

RadiantQ jQuery Gantt Package

© RadiantQ, 2009 - 2018

Table of Contents

UserGuide	7
GanttControl or FlexyGantt	7
Introduction	9
Installation	9
About Samples	10
HTML Samples	11
Angular Samples	12
React Samples	17
MVC Samples	21
WebForms Samples	22
PHP Samples	23
TypeScript Samples	24
Deploying Gantt	25
IIS	26
Single HTML Sample	26
Entire HTML Samples	28
External Dependencies	31
IE 8 Support	33
Gantt Control	34
Getting Started	35
In HTML	35
In Angular	42
In React	48
In ASP.NET	50
In ASP.NET MVC	56
In PHP	63
In TypeScript	68
Gantt Basics	73
GanttTable	74
Setting up the GanttTable	74
GanttTable Editing	75
Runtime Interaction	79
Activities or Tasks	82
Dependencies	84
Keeping Dependant tasks "sticky"	86
Calendars	88
WorkTimeSchedule	89
Calendar and CalendarWithExceptions	94
Time Scale Header	96
Resource Assignment	97
Binding to Resource Strings	97
Binding to Resource Objects	99
Resource Specific Calendars	103
Assignments VS Task Duration	108
Critical Paths	110
WBS Support	112
Undo/Redo	116
Enabling Undo/Redo	116
Adding Undo actions Programmatically	118
Creating Custom Undo Actions	120

Data Binding	121
About Task Field Types	121
XML Data	122
Using RadiantQ Binding	126
Using Knockout	130
Persisting Changes	132
Persisting Changes Immediately	132
Persisting Changes in Bulk	136
EndTime in DataSource	139
EndTime field without Effort field	139
EndTime field with Effort field	141
Data Binding in PHP	142
Persist Changes Immediately	144
Persist Changes in Bulk	147
MS Project Export/Import	150
Scheduling Features	152
Resource Level Schedules	152
Resource Leveling	153
Task Level Schedules	154
Resource Load View	157
Behavior Customization	158
Tasks Filtering	158
Disabling Runtime Features	160
Global ReadOnly	160
Global Selective ReadOnly settings	163
Task specific ReadOnly settings	170
GanttTable Customization	171
Enable Row Drag and Drop	171
Enable edit mode in single click	174
Enable HeaderMenu	175
Enable GanttTable startEdit and endEdit options	176
Gantt Chart Customization	178
Overriding Dependency Setup Behavior	178
Appearance Customization	179
Gantt CSS Styles	179
ContextMenus	180
Chart Look and Feel	183
Task Bar Look and Feel	184
Custom Look and Feel	184
Custom Look for Specific Task Bars	186
Custom UI on Task Bars	187
TaskBarBackgroundTemplate	190
Task Bar Labels	194
Task Bar Redraw	197
Task Popup	199
Gantt Table Customization	203
GanttTable Custom Column	203
GanttTable Column Editable Settings	205
GanttTable Alternative Row background	207
How Tos	208
How to find a task/activity by it's ID?	208
How to find ActivityView by its ID?	209
How to listen to changes made by the end-user on the tasks?	210

What are the different ways to improve the performance of the gantt with a large set of tasks?	213
How to visually select a row given it's task id?	215
How to remove a dependency?	216
How to get all dependencies of an activity?	217
How to Expand all/Collapse all summary tasks dynamically?	218
How to get hold of bound activity in row click?	219
How to add new task using the context menu?	220
How to refresh Gantt with new data with same schema	221
How to refresh Gantt with new data with new schema	222
How to export current Gantt data to JSON?	223
How to listen the Activity CollectionChanges in GanttControl?	224
How to add resource to an activity programmatically	225
How to enable "Undo/Redo" feature for custom columns ?	226
How to add a new resource in resource dataset without reloading Gantt?	229
How to save Gantt options in cookies?	230
How to notify when dependency connection is failed?	231
How to make selection and hover effect?	232
How to make Custom Progress Calculation?	233
How to add custom resource dropdown for custom column?	234
How to enable paging while autoscrolling in GanttChart ?	236
FlexyGantt	237
Getting Started	238
In HTML	238
In Angular	244
In React	251
In ASP.NET	254
In ASP.NET MVC	262
In PHP	269
In TypeScript	275
FlexyGantt Basics	280
DataBinding	281
XML Data	281
Using RadiantQ Binding	286
Using Knockout	290
Task Template	292
Editing Tasks	296
Multi Column Tree Grid	298
FlexyTable Editing	302
Time Scale Header	303
Time Scale Header Customization	303
Dependency Lines	304
Performance Optimization Options	307
Assigning Schedules	312
Schedule for Rendering	312
Row Specific Schedule for Rendering	314
Appearance Customization	316
Chart Look and Feel	317
Overlapped Tasks Look	317
Task Labels	326
Task Popups	332
Custom UI In Task Row	335
RefreshRowBackground	335
RefreshRowForeground	338

Row Background	340
Gantt Look and Feel	341
ContextMenus	343
Behavior Customization	345
FlexyTable Customization	346
Enable Row Drag and Drop	346
Events	349
Task Bar Vertical Dragging	349
How Tos	351
How to Convert X to Time and Time to X in the Chart?	351
How to refresh Gantt with new data with same schema	352
How to listen the taskbar size changes in FlexyGantt?	353
How to listen the taskbar click event in FlexyGantt?	354
How to show different taskbars in single row?	355
How to prevent Taskbar Overlapping?	357
How to add tooltip for a custom element ?	360
How to maintain the current state of the Expand/Collapse nodes after data source reset?	362
How to make reflect the bound data property changes into the gantt (view)?	363
How to listen the node Expand/Collapse state in FlexyGantt?	365
How to add data asynchronously while on expanding the node?	366
How to insert/remove items dynamically?	368
How to set the overlapping feature?	369
Common Topics	372
Getting Started	373
Cleaning Up Src Folder	373
Gantt Basics	375
Virtualization In Gantt	375
Round To Options	377
Appearance Customization	379
Gantt CSS Styles	379
Special Lines	383
Custom Chart Background	385
Time Span Highlighting	390
ContextMenus	393
Localized Strings	394
Themes	396
Creating Gradient Backgrounds dynamically in code	401
Chart Look and Feel	403
Time Scale Header Customization	403
Default Time Scale Header	404
Header Text Format and Header Height Customization	406
Zooming Programmatically	411
Custom Time Scale Headers	412
End-User Time Scale Header Operations	417
Gantt Chart AnchorTime	419
GanttChart View Width	420
Browse To Task Cues	421
TimeSpan Paging	423
Events	425
GanttChart Events	425
Scheduling Features	426

Resource Load View	426
Printing and Export	429
Printing	429
Export	447
JS Patterns	449
RequireJS	449
Backbone.JS	452
Bootstrap	456
How Tos	463
How to bring a task into view in the gantt chart?	463
How to vertically scroll and bring a task into View?	464
How to pass a argument to server in MVC Wrapper?	465
How to pass some AjaxSettings to ajax in MVC?	466
How to improve the performance of the Gantt while setting multiple options?	467
How to send exported image to server?	468
How to adjust the chart's zoom level to make it show all the tasks in the project?	469
How to customize the React Gantt Component?	471
How to communicate with gantt component from your own component in React?	472
How to render the multiple react gantt component in single page?	475
JSON Data	478
Troubleshooting	481
Why the gantt is not showing in the page and how to fix it?	481
How to enable scroll bars for Mac OS ?	483
Why React gantt application renders a blank page and how to fix it ?	484

UserGuide

RadiantQ jQuery Gantt Package GanttControl or FlexyGantt

The RadiantQ jQuery Gantt package consists of 2 rich Gantt widgets:

[GanttControl](#)

The GanttControl is ideal for visualizing project-tasks and has these key features.

- Binds to a flat list of tasks with hierarchies defined through an "indent-level" property.
- Includes a built-in scheduling engine with built-in "working time" support.
- Visualizes and enforces dependency constraints.
- The UI is pre-defined inside the control's templates.

[Flexy Gantt](#)

The FlexyGantt is ideal for visualizing resource-utilization and has these key features.

- Binds to a hierarchical list of business-objects. The leaf nodes represents a "resource". Each resource can contain multiple "tasks" all of which are displayed in the same row in the gantt chart on the right.
- No scheduling or dependency constraints supported.
- The task bar look and feel is defined at the application level via templates.

Comparison Matrix

Feature/Control	GanttControl	FlexyGantt
Project Scheduling	Y	Y
Dependency Constraints	Y	N
Dependency Lines	Y	Y
Multiple tasks per row	N	Y
Time Scale Headers	Y	Y
User Interaction	Y	Y
Multi-column tree list on the left	Y	Y
DataSource	Flat List	Flat or any Hierarchical

Gantt Visualization Patterns

While we list some common patterns here, our controls are generic enough to support a lot of different patterns.

Gantt for Effort Driven Scheduling (default behavior)

- Use a GanttControl to list a hierarchy of tasks.
- Your data source usually has a StartTime and Effort field for the task.
- The End Time for the task is automatically calculated based on the resource assignments.

- Gantt schedules the task's times based on configurable working times, set dependency constraints, etc.

Gantt for Fixed Duration Scheduling

- Use a GanttControl to list a hierarchy of tasks.
- Your data source usually has a StartTime and EndTime field (although an Effort is recommended) for the task.
- Multiple Resources can be assigned to a task, but that does not affect the duration of the task (set the AdjustDurationOnAssignment property to false).
- Gantt schedules the task's times based on configurable working times, set dependency constraints, etc.

Gantt for Resource Allocation Views

- Use a FlexyGantt to visualize a hierarchy of resources.
- Visualize the hierarchy in the table on the left.
- For each row (resource) in the table, visualize a list of tasks or a single task in the chart - in the same row.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

Introduction

RadiantQ jQuery Gantt Package

Installation

Installation on Development Machine

The install for the jQuery Gantt Package is available in a .msi or .zip file that you can install/unzip to anywhere in your system. The install comes with samples and documentation and the source .js files. It is this .js files you simply have to include in your web project.

<install folder>\Src - All the Source files required to include jQuery Gantt in your applications.

<install folder>\SampleBrowser.htm - A combined sample that showcases all the html samples included.

<install folder>\SampleBrowser.csproj - The VS.NET project that allows you to take a look at the sample html source and also debug.

<install folder>\Samples - Folder containing all individual sample htm files.

<install folder>\SampleBrowserSrc - Some custom js and css files used in the above Sample Browser application.

<install folder>\PlatformSamples\MVCSamples - As the name implies, a sample project with MVC aspx pages, that illustrates using jQuery Gantt in MVC.

<install folder>\PlatformSamples\ASPNETWebFormsSamples - As the name implies, a sample project with aspx pages, that illustrates using jQuery Gantt in ASP.

<install folder>\PlatformSamples\PHPSamples - As the name implies, a sample project with PHP pages, that illustrates using jQuery Gantt in PHP.

<install folder>\PlatformSamples\TypeScriptSamples - As the name implies, a sample project with typescript and htm pages, that illustrates using jQuery Gantt in TypeScript.

<install folder>\PlatformSamples\AngularSamples - As the name implies, a sample project with Angular, that illustrates using jQuery Gantt in Angular

<install folder>\PlatformSamples\React - As the name implies, a sample project with React, that illustrates using jQuery Gantt in React.

<install folder>\PlatformSamples\SQLBindings - As the name implies, a sample project to bind the SQL data with gantt, that illustrates using jQuery Gantt in ASP, MVC4 and MVC3

<install folder>\Documentation - The Users Guide for the jQuery Gantt Package.

RadiantQ jQuery Gantt Package

About Samples

The install includes samples that illustrate using the gantt in straight HTML, Angular, MVC, WebForms, PHP, TypeScript and React. Each of these have lots of samples illustrating the different functionality.

[HTML Samples](#)

[Angular Samples](#)

[MVC Samples](#)

[WebForms Samples](#)

[PHP Samples](#)

[TypeScript Samples](#)

[React Samples](#)

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

HTML Samples

HTML Samples

Straight individual HTML samples are in this folder:

<install path>/Samples/*.htm

Sample Browser

There is a "Sample Browser" which lets you browse all these samples conveniently. The HTML for which is at:

<install path>/SampleBrowser.htm

Opening HTMLs directly in the browser

In Windows, you can simply double click the sample *.htm samples to open and run them in the browsers. This usually works fine in IE and Firefox, whereas Chrome doesn't run this for security reasons.

Running with VS.NET Project

The <install path>/SampleBrowser.csproj is a convenient way to run the Sample Browser and all the samples, if you have Visual Studio installed.

Running samples from a local web server

To run our samples in a local server, please follow these steps for IIS (follow similar steps for other servers):

[Deploying a Single HTML Sample in IIS](#)

[Deploying Entire HTML Samples in IIS](#)

© RadiantQ 2009-2018. All Rights Reserved.

RadiantQ jQuery Gantt Package

Angular Samples

Angular Samples

You can find the Angular Samples in the following path,

<install path>/PlatformSamples/AngularSamples/app/Samples

NOTE: You can also find the optional VisualStudio project or solution for Angular in the following path,

<install path>/PlatformSamples/AngularSamples/Angular4DemoVS2015.csproj

1) How to run the Angular Sample

You can run the Angular Gantt sample using NPM(cmd), Visual Studio 2015+ versions or Visual Studio Code. For all these 3 approaches, you need to first use npm to install the dependent 'node_modules' as mentioned in the steps below. (The 'node_modules' package dependencies are already specified in 'package.json' file.)

Also, make sure to update your npm version to 5.4.1 or later. (Use "*npm version*" to determine your version). To update to the latest, use cmd line: "*npm install npm@latest -g*".

Steps to run the Angular sample using NPM:

1. Browse to the project directory (*./PlatformSamples/AngularSamples*) in command prompt using "Run as administrator" option.
2. Type "*npm install*". This will install the different dependant modules in the 'node_modules' directory.
3. Run "*npm start*".
4. The sample automatically runs in default browser. If not, copy the localhost and paste it in browser.

Steps to run the Angular sample using Visual Studio 2015+:

1. Install 'node_modules' package (*Step 1 and Step 2 from NPM steps above*)
2. Then open '.csproj' with Visual Studio 2015 and greater versions from project directory and run it as usual.

Steps to run the Angular sample using Visual Studio Code:

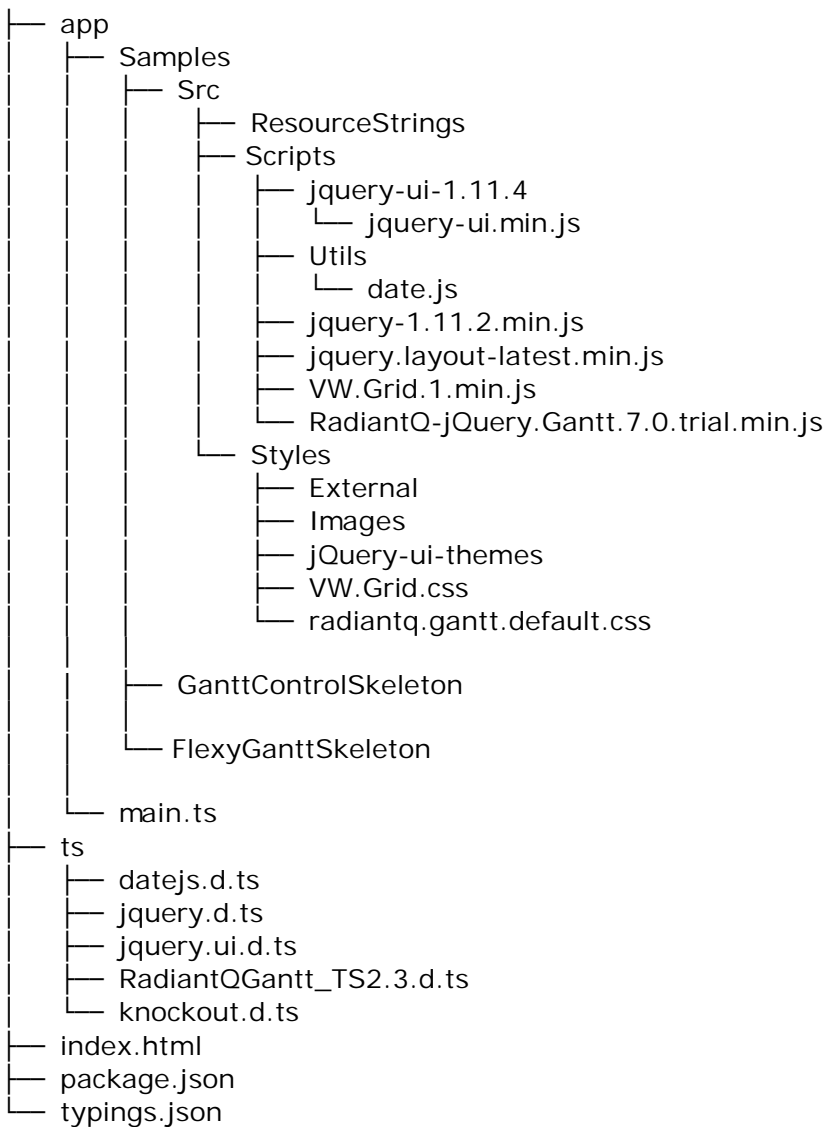
1. Open Visual Studio Code IDE
2. Import Angular gantt project using *File --> Open Folder*
3. Open command window using *View --> Integrated Terminal* and select *TERMINAL* tab
4. Then follow *Step 2 to 4 from NPM steps above*.

2) Angular Gantt Structure and Installation

The Angular Gantt package v4.0 is available in <install path>/PlatformSamples/AngularSamples

This platform includes only the necessary files and folders to build and run the basic Angular samples.

Create your project directory and structure it as you need. Ours looks like this,



The Angular Gantt files can be listed as follows.

- package.json
- index.html
- ts folder
- app folder

package.json

The 'package.json' file specifies the project name, description, server and also contains the Angular necessary packages which can be installed during npm installation(explained below).

This JSON file also includes packages like angular-cli, typescript, typings under 'devDependencies'. Here, the user can also add some additional packages based on their project requirement.

index.html

The 'index.html' contains all the gantt source references. It has the root tag with sample's name ' <GanttControlSkeleton> </GanttControlSkeleton>' which detects by the export class to find whether the sample belongs to GanttControl or FlexyGantt. And please be sure to comment other available root tags apart from <GanttControlSkeleton>

'ts' folder

The TypeScript declaration files such as `datejs.d.ts`, `jquery.d.ts`, `RadiantQGantt_TS2.3.d.ts` etc., were available in this 'ts' folder since Angular gantt is based on TypeScript.

'app' folder

The 'app' folder consists of `main.ts`, `Samples` folder.

The following files and directory were available in this 'app' and 'Samples' folder.

- `Samples` folder -
- `main.ts`
- `SampleBrowser` folder

- `main.ts`

The 'main.ts' imports the angular components such as `BrowserModule`, `platformBrowserDynamic`, etc., and also gantt classes.

The declaration of our gantt export classes takes place using '@NgModule({})' decorator. And it also bootstraps the export class '*RQGanttSample*'.

Here, the user can also enable production mode using method '`enableProdMode()`'

```
import { enableProdMode } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { Http, Response, HttpClientModule, JsonpModule } from '@angular/http';
import { NgModule } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { Routes, RouterModule } from '@angular/router';
// RadiantQ components.
import { DataService } from './data.service';
import { RQGanttControl, RQFlexyGantt, Column, getClientTemplate,
getClientEditorTemplate, rqTemplateBinder, getTaskItemTemplate,
getParentTaskItemTemplate } from './RQGanttSettings';
import { SampleBrowser } from './SampleBrowser/SampleBrowser';
import { FlexyGanttSkeleton } from './Samples/FlexyGanttSkeleton/FlexyGanttSkeleton';
import { GanttControlSkeleton } from
'./Samples/GanttControlSkeleton/GanttControlSkeleton';
import { GanttControlCustomDataBinding } from
'./Samples/GanttControlCustomDataBinding/GanttControlCustomDataBinding';
import { ResourceLoadView } from './Samples/ResourceLoadView/ResourceLoadView';

// Route config let's you map routes to components
const routes = [
  {
    path: 'GanttControlCustomDataBinding',
    component: GanttControlCustomDataBinding
  },
  {
    path: 'GanttControlSkeleton',
    component: GanttControlSkeleton
  },
  {
    path: 'FlexyGanttSkeleton',
    component: FlexyGanttSkeleton
  },
  {
    path: 'ResourceLoadView',
    component: ResourceLoadView
  },
  {
    path: '',
    redirectTo: '/GanttControlCustomDataBinding',
    pathMatch: 'full'
  }
];
export const appRouterModule = RouterModule.forRoot(routes, { useHash: true }); //
'useHash' - To avoid 404 error while manually refreshing URL.

//enableProdMode();
@NgModule({
  imports: [
    BrowserModule,
    HttpClientModule,
    JsonpModule,
    appRouterModule
  ],
  declarations: [
    getParentTaskItemTemplate,
    getTaskItemTemplate,
    getClientTemplate,
    getClientEditorTemplate,
    rqTemplateBinder,
    Column,
    RQFlexyGantt,
    RQGanttControl,
    ResourceLoadView,
    GanttControlCustomDataBinding,
    GanttControlSkeleton,
```

main.ts

'Samples' folder

The 'Samples' folder consists of various samples and it's respective html, css, ts and json files. It also contains the 'Src' folder in it.

- 'Src' folder

The 'Src' folder contains the source files which requires to run our gantt samples such as ResourceStrings, Scripts, Styles.

Samples folder

- GanttControlSkeleton sample folder
 - GanttControlSkeleton.ts
 - GanttControlSkeleton.html
 - GanttControlSkeleton.css
 - GanttControlSkeleton.json
- FlexyGanttSkeleton sample folder
 - FlexyGanttSkeleton.ts
 - FlexyGanttSkeleton.html
 - FlexyGanttSkeleton.css
 - FlexyGanttSkeleton.json
- GanttControlCustomDataBinding sample
- ResourceLoadView sample

These samples were briefly explained here :-

- o [GanttControlSkeleton](#)
- o [FlexyGanttSkeleton](#)

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

React Samples

React Samples

You can find the React Samples in the following path,

<install path>/PlatformSamples/React/Samples.

NOTE: You can also find the optional React Visual Studio project or solution in the following path,

<install path>/PlatformSamples/React/GanttInReact.csproj.

How to run the React Sample

You can run the React Gantt sample using NPM(cmd), Visual Studio 2017 versions or Visual Studio Code. For all these 3 approaches, you need to first use npm to install the dependent 'node_modules' as in the steps below. (The 'node_modules' package dependencies are already specified in 'package.json' file.)

Also, make sure to update your npm to latest version(Use "npm version" to determine your version). To update to the latest, use cmd line: "npm install npm@latest -g".

Steps to run the React sample using NPM:

1. Browse to the project directory (.\\PlatformSamples\\React) in command prompt using "Run as administrator" option.
2. Type "*npm install*". This will install the dependant modules in the node_modules folder.
3. Run "*npm start*".
4. The sample automatically runs in default browser. Otherwise, copy the port and paste in any browser.

Steps to run the React sample using Visual Studio 2017:

1. Install webpack and webpack-cli globally by using the "*npm install webpack -g*" and "*npm install webpack-cli -g*".
2. Install 'node_modules' package (*Step 1 and Step 2 from NPM step above*)
3. Then open '.csproj' with Visual Studio 2017 from project directory and run it as usual.

Steps to run the React sample using Visual Studio Code:

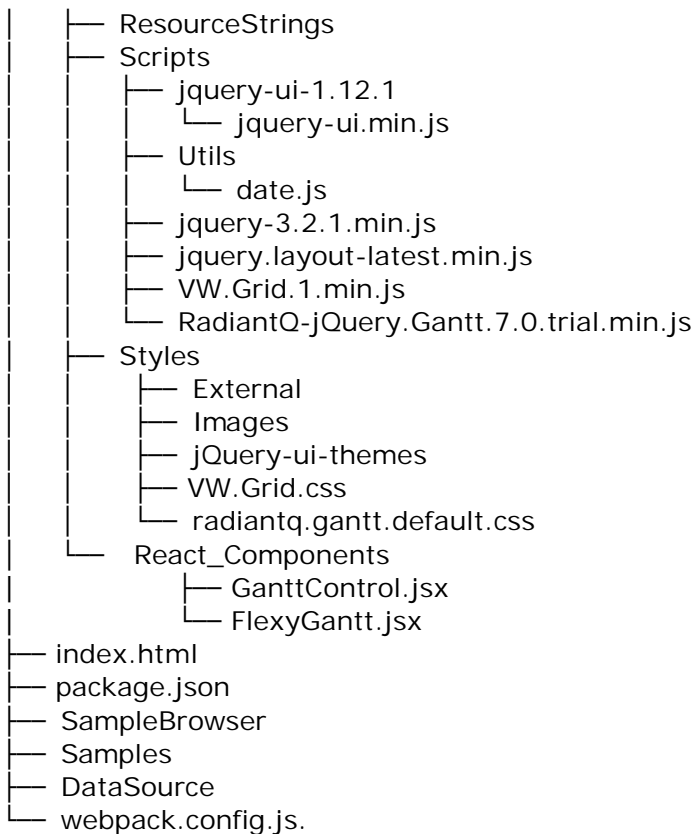
1. Open Visual Studio Code IDE
2. Import React gantt project using *File --> Open Folder*
3. Open command window using *View --> Integrated Terminal* and select *TERMINAL* tab
4. Then follow the *Steps 2 to 4 from NPM step above*.

React Gantt Structure and Installation

The React Gantt package is available in <install path>/PlatformSamples/React

Create your project directory and structure it as you need. Ours looks like this,

└─ Src



The React Gantt files can be listed as follows.

- package.json.
- Index.html.
- webpack.config.js.
- Src folder.
- Samples folder.
- DataSource folder.

package.json

The 'package.json' file specifies the project name, description and React, Webpack and Babel packages which can be installed during npm installation.

Configuring Webpack

Here, we set up webpack for our React Gantt by adding desired config in webpack.config.js file.

```
const path = require('path');
const webpack = require('webpack');
```

```
module.exports = {
  entry: {
    // Here, Webpack to start bundling our app at following paths.
    GanttControlSkeleton: './Samples/GanttControlSkeleton/GanttControlSkeleton',
    FlexyGanttSkeleton: './Samples/FlexyGanttSkeleton/FlexyGanttSkeleton.jsx'
  },
```

```
  output: {
    // Output our app to the dist/ directory with Sample name. (eg:
    dist/SampleBrowser.js)
```

```
    path: path.resolve(__dirname, 'dist'),
    filename: '[name].js',
    publicPath: 'dist/'
  },
```

```
  devServer: {
    inline: true,
    port: 3000
  },
```

```
  // devtool enhances the debugging process
  devtool: 'inline-source-map',
```

```
  resolve: {
    // add alias for application code directory
    alias: {
      RQSrc: path.resolve(__dirname, 'Src'),
      Samples: path.resolve(__dirname, 'Samples'),
    },
  },
```

```
  //Webpack uses loaders to translate the file before bundling them.
```

```
  module: {
    rules: [
      {
        test: /\.jsx?$/,
        loader: 'babel-loader',
        exclude: /node_modules/,
        //Configuring babel
        query: {
          presets: ['es2015', 'react']
        }
      },
      {
        test: /\.css$/,
        use: [
          "style-loader",
          "css-loader"
        ]
      },
      {
        test: /\.(png|jp(e*)g|svg)$/,
        use: [{
          loader: 'url-loader',
          options: {
            limit: 8000,
            name: 'images/[hash]-[name].[ext]'}
        }]
      }
    ],
  },
};
```

webpack.config.js

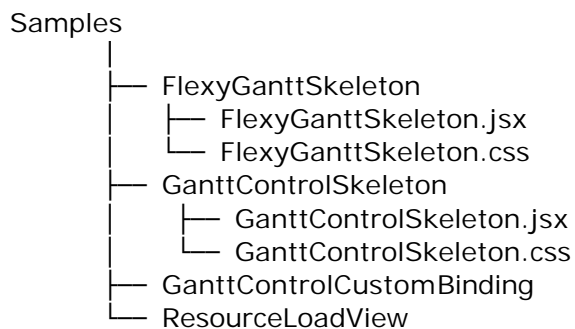
To know more about webpack please refer the following link [Webpack](#) .

'Src' folder

The 'Src' folder contains the source files which requires to run our gantt such as ResourceStrings, Scripts, Styles and it also includes 'React_components' folder, which contains the "GanttControl.jsx" and "FlexyGantt.jsx", these components are exported their functionality and it can be used anywhere by importing the Component.

'Samples' folder

It contains the samples to illustrate our features and usage.



It was briefly discussed in :-

- [GanttcontrolSkeleton.jsx](#)
- [FlexyGanttSkeleton.jsx](#)

RadiantQ jQuery Gantt Package

MVC Samples

MVC samples using the Gantt MVC Extensions are available for both MVC 3 and 4. You can find these projects in the following paths:

<install path>/PlatformSamples/MVCSamples/MVC3RazorDemo/MVCRazorDemo2010.csproj

<install path>/PlatformSamples/MVCSamples/MVC4RazorDemo/MVCRazorDemo2010.csproj

The assemblies implementing these extensions are here (and are referenced by the above projects):

<install path>/Src/bin/DotNET4MVC3/RadiantQ.Web.JQGantt.dll

<install path>/Src/bin/DotNET4MVC4/RadiantQ.Web.JQGantt.dll

Sample Contents

a) All the cshtml samples are in this folder within the project folder:

<project folder>/Views/Home/

b) and the controllers providing data are within this folder:

<project folder>/Controllers/

© RadiantQ 2009-2018. All Rights Reserved.

RadiantQ jQuery Gantt Package
WebForms Samples

WebForms samples using the Gantt Controls are included in the install here:

<install path>/PlatformSamples/ASPNETWebFormsSamples/ASPNetGanttDemo2010.csproj

The assembly implementing the WebForms Gantt controls in .NET 4.0 are here:

You can link to either of these assemblies (they differ only in their reference to MVC which is irrelevant when you are using WebForms):

<install path>/Src/bin/DotNET4MVC3/RadiantQ.Web.JQGantt.dll

<install path>/Src/bin/DotNET4MVC4/RadiantQ.Web.JQGantt.dll

Sample Contents

a) You can find the individual aspx sample pages here:

<install path>/PlatformSamples/ASPNETWebFormsSamples/Samples

b) and the handlers which provide data to ganttt are in the folder:

<install path>/PlatformSamples/ASPNETWebFormsSamples/DataSources/

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

PHP Samples

PHP Samples

PHP samples using the Gantt PHP extensions can be loaded from the PHP SampleBrowser here:

In VS.NET project

Simply open:

<install path>/PlatformSamples/PHPSamples/RadiantQ.Web.PHP.JQGantt.phpproj

This opens in VS.NET with the extension "PHP Tools for Visual Studio" installed.

In Eclipse

To work in Eclipse, create a new Eclipse project in this folder:

<install path>\PlatformSamples\PHPSamples

... and include all the files and folders into that project before you run.

Sample Contents

a) Individual PHP sample files are here:

<install path>/PlatformSamples/PHPSamples

b) The PHP Gantt Interfaces, classes, serializers, etc. are in PHP wrappers here:

<install path>/PlatformSamples/PHPSamples/lib

c) The Required Source JS and CSS are here:

<install path>/PlatformSamples/Samples/Src

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

TypeScript Samples

TypeScript Samples

TypeScript Samples are compressed and placed here:

<install path>/PlatformSamples/PlatformSamples/TypeScriptSamples/TypeScriptSamples.zip.

To begin with, unzip the contents of the above zip into the same folder.

Sample Browser project

This VS.NET project lets you browse all the Gantt TypeScript samples:

<install path>/PlatformSamples/PlatformSamples/TypeScriptSamples/
RadiantQGanttTypeScript.csproj

Sample Contents

a) You can find the individual TS samples here:

<install path>/PlatformSamples/TypeScriptSamples/Samples

b) The TypeScript Gantt (and other dependant libraries') definition SRC files are here:

<install path>/PlatformSamples/TypeScriptSamples/ts

c) The SRC JS and CSS are here:

<install path>/PlatformSamples/TypeScriptSamples/ts/Samples/Src

© RadiantQ 2009-2018. All Rights Reserved.

RadiantQ jQuery Gantt Package

Deploying Gantt

Deploying Gantt

The files under the <install path>\Src folder in our install should be typically referenced in your web pages and also be deployed in your server. Let us take a closer look at the contents of this folder:

- Scripts - This contains all the jQuery utility/plugin/widget files that are necessary for the gantt as well as the jQuery Gantt script files.
- Styles - CSS and Image files that define the Gantt look and feel.
- themes - CSS and Images files of the plugins and widgets used by the gantt.
- ResourceStrings - Localized string dictionaries are defined here. The default English strings (required by default) and strings for few other cultures will be loaded from these files. You can add more localized string dictionaries if the default supported dictionaries don't suffice.
- bin - Contains the jQuery Gantt MVC extension assembly that should be referenced in the ASP.NET MVC projects.

Note on json files

The server will have to be setup to support uploading of files with json extensions.

By default servers do not serve .json file types(no wildcard MIME type). Therefore a 404 not found is thrown when they are accessed. So you will have to add a MIME type to allow it to serve that type of file. You can set it at the site level or at the server level.

To set this for the entire server in IIS:

Open the properties for the server in IIS Manager and click MIME Types
Click "New". Enter "json" for the extension and "application/data" for the MIME type.

To set this for the site level in IIS:

If you want to manually add support to your site, you can just add the following to your web.config in the system.webServer section:

```
<staticContent>  
  <mimeMap fileExtension=".json" mimeType="application/data" />  
</staticContent>
```

Note on Date.js files

You can see that there is a Src\Scripts\Utils\date.js file which is a mandatory file to be included in your pages.

And for globalization scenarios, you should include one of the several extension files based on the culture you are currently running your pages. For example, if you are running in German culture, you should also include this js file in your page: Src\Scripts\Utils\globalization\de-DE.js.

IIS

RadiantQ jQuery Gantt Package

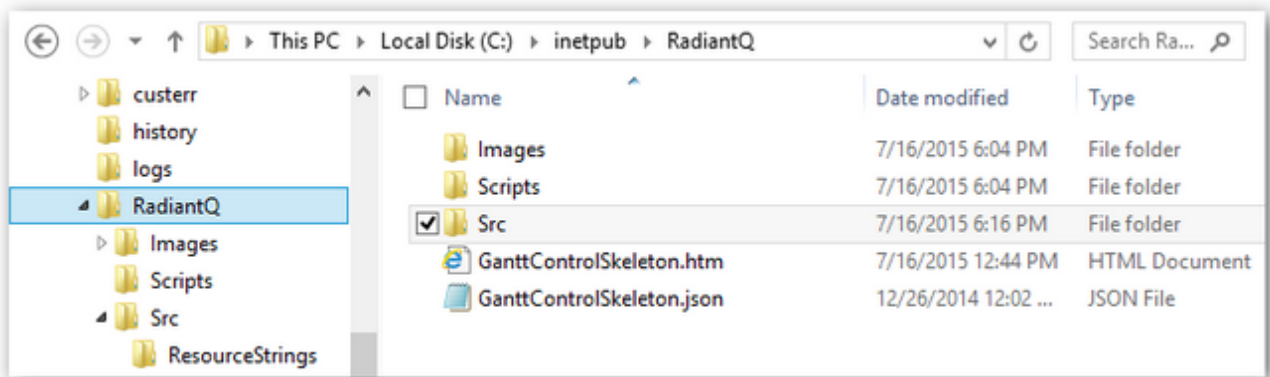
Single HTML Sample

Deploying Single Gantt HTML Samples in IIS

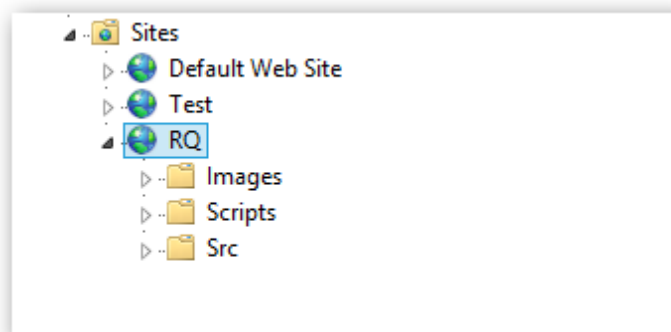
Here we will go through the steps involved in deploying our samples that are part of our Install to an IIS server.

To deploy single gantt sample in IIS you need to copy over following files from install path into your server.

- RadiantQ jQuery Gantt Package\Samples\GanttControlSkeleton.htm
- RadiantQ jQuery Gantt Package\Samples\GanttControlSkeleton.json
- RadiantQ jQuery Gantt Package\web.config
- RadiantQ jQuery Gantt Package\Src (exclude the bin folder).



Files inside IIS Server path



Same Files as above in IIS Manager

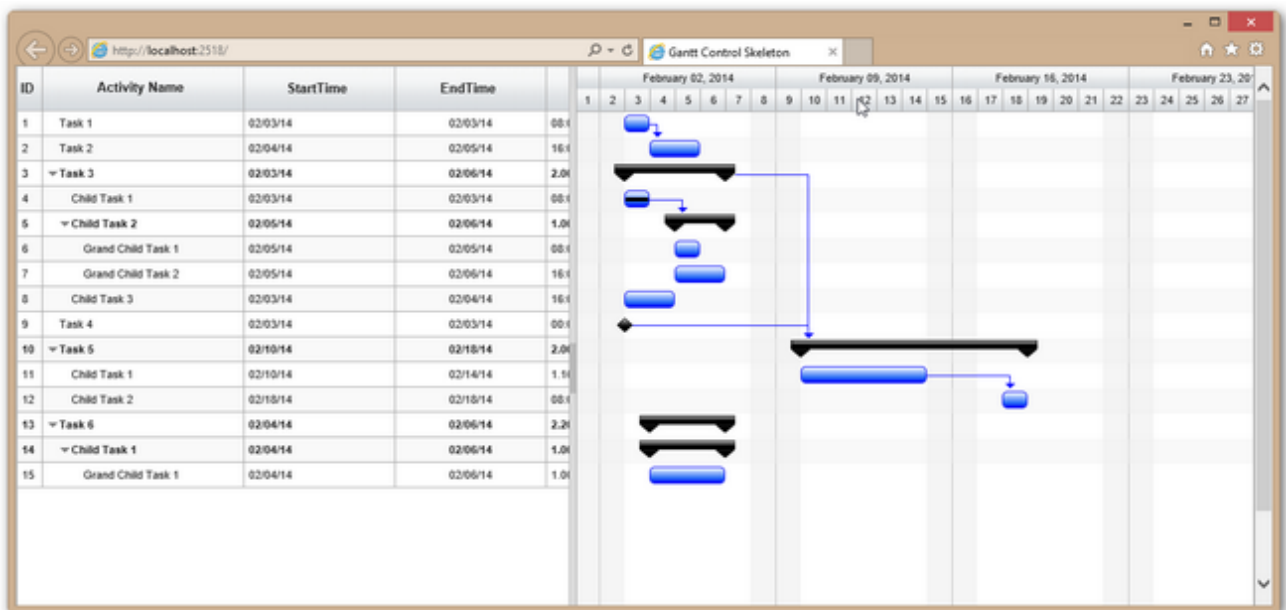
As mentioned in the previous topic [Deploying Gantt](#), ensure the mime type for .json, so that the server serves the sample JSON files.

MIME Types

Use this feature to manage the list of file name extensions and associated content types served as static files by the Web server.

Group by: No Grouping		
Extension	MIME Type	Entry Type
.jpe	image/jpeg	Local
.jpeg	image/jpeg	Local
.jpg	image/jpeg	Local
.js	application/javascript	Local
.json	application/json	Local
.jsx	text/jscrip	Local
.latex	application/x-latex	Local
.lit	application/x-ms-reader	Local
.lpk	application/octet-stream	Local
.lsf	video/x-la-asf	Local
.lsx	video/x-la-asf	Local
.lzh	application/octet-stream	Local
.m13	application/x-msmediaview	Local
.m14	application/x-msmediaview	Local

JSON Mime type added in IIS



Samples running in your local IIS

RadiantQ jQuery Gantt Package

Entire HTML Samples

Deploying Gantt HTML Samples in IIS

Here we will go through the steps involved in deploying our samples that are part of our Install to an IIS server.

To deploy gantt in IIS you need to copy over following files from install path into your server.

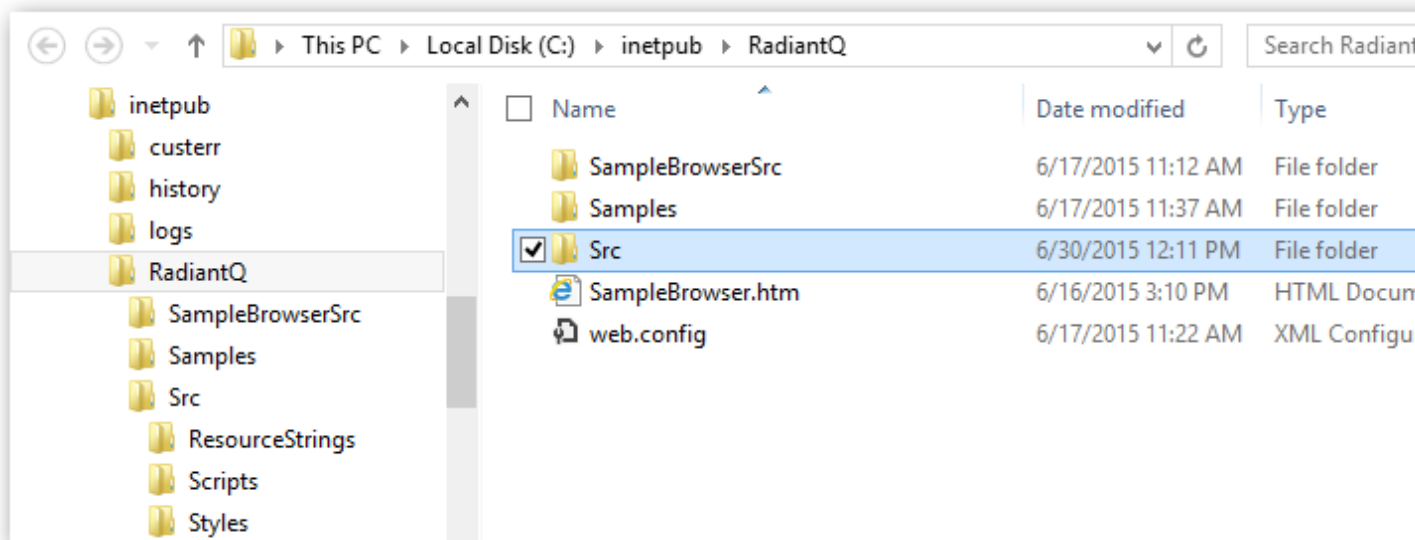
RadiantQ jQuery Gantt Package\SampleBrowser.htm

RadiantQ jQuery Gantt Package\web.config

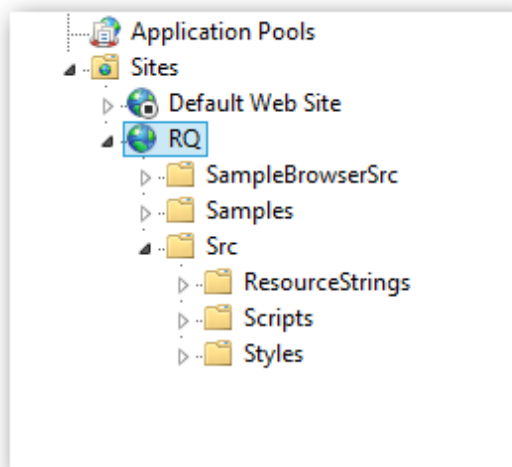
RadiantQ jQuery Gantt Package\SampleBrowserSrc (The JS files required to run the Sample Browser)

RadiantQ jQuery Gantt Package\Samples (The individual sample files).

RadiantQ jQuery Gantt Package\Src (exclude the bin folder).

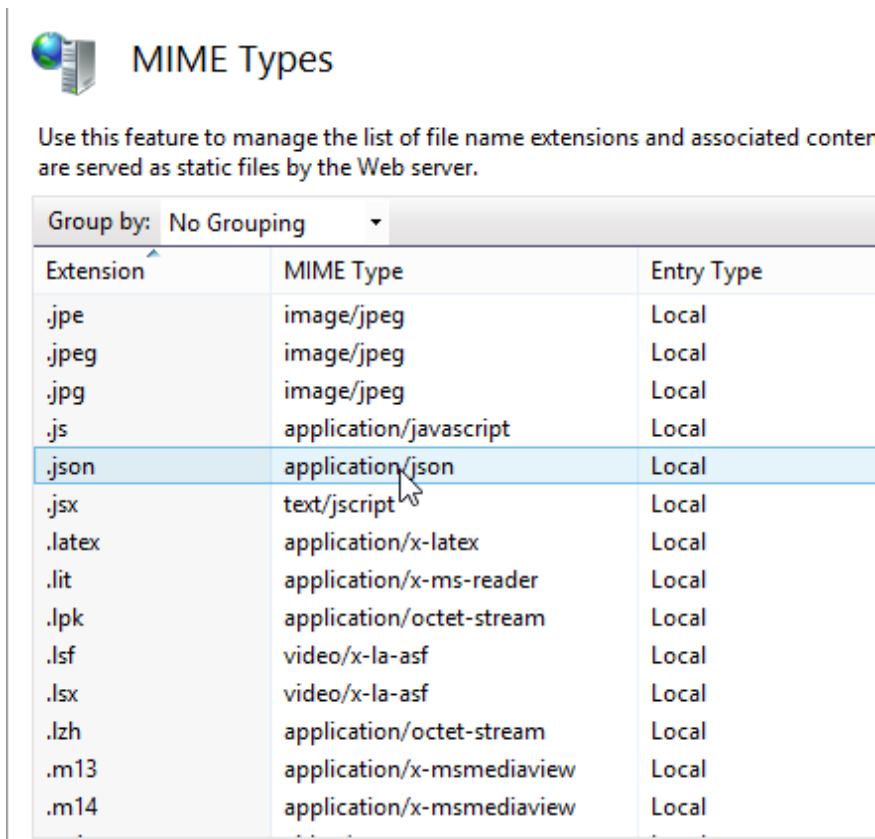


Files inside IIS Server path



Same Files as above in IIS Manager

As mentioned in the previous topic [Deploying Gantt](#), ensure the mime type for .json, so that the server serves the sample JSON files.



Use this feature to manage the list of file name extensions and associated content types that are served as static files by the Web server.

Extension	MIME Type	Entry Type
.jpe	image/jpeg	Local
.jpeg	image/jpeg	Local
.jpg	image/jpeg	Local
.js	application/javascript	Local
.json	application/json	Local
.jsx	text/jscript	Local
.latex	application/x-latex	Local
.lit	application/x-ms-reader	Local
.lpk	application/octet-stream	Local
.lsf	video/x-la-asf	Local
.lsx	video/x-la-asf	Local
.lzh	application/octet-stream	Local
.m13	application/x-msmediaview	Local
.m14	application/x-msmediaview	Local

JSON Mime type added in IIS

RadiantQ
Project Management Controls

Project Gantt

- GanttControlCustomDataBinding
- GanttControlCriticalPath
- ResourceLeveling
- ResourceLoadView
- GanttControlUndoRedo
- PerformanceTestSample
- GanttControlCustomAppearance
- GanttControlTableCustomization
- WBSEnabledTasks
- BrowseToTasksCues
- GanttControlBindingToEndTime
- GanttControlFiltering
- GanttBoundToXML
- Flexy Gantt
- Common

Version : 5.01.1

DEMO CODE

Update Critical Paths Clear Critical Paths

ID	Activity Name	StartTime	EndTime
1	Task 1	02/03/14	02/03/14
2	Task 2	02/04/14	02/05/14
3	Task 3	02/03/14	02/06/14
4	Child Task 1	02/03/14	02/03/14
5	Child Task 2	02/05/14	02/06/14
6	Grand Child Task 1	02/05/14	02/05/14
7	Grand Child Task 2	02/05/14	02/06/14
8	Child Task 3	02/03/14	02/04/14
9	Task 4	02/03/14	02/03/14
10	Task 5	02/10/14	02/18/14
11	Child Task 1	02/10/14	02/14/14
12	Child Task 2	02/18/14	02/18/14
13	Task 6	02/04/14	02/05/14
14	Child Task 1	02/04/14	02/05/14

February 02, 2014

Resource 1

Show Description

Copyright © 2014 RadiantQ | All Rights Reserved

Samples running in your local IIS

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

External Dependencies

Dependencies

The jQuery Gantt Package is dependent on the javascript files that are listed below:

- [jQuery](#) lib - The jQuery Gantt is built on top of the latest jQuery version. (See below for supported versions)
- [jquery.layout-latest.js](#) - Extension that provides resizing functionality using a splitter.
- [knockout](#) (optional) - Provides MVVM support in jQuery apps
 - [knockout-2.0.0.js](#)
 - [knockout.mapping-latest.debug.js](#)
- [date.js](#) - Provides extended date APIs.
- [jquery.contextMenu.js](#) (optional) - Used for context menu support in the Gantt for older than jQuery UI 1.11.4 versions.
- [jquery.xml2json.js](#) (optional) - Use this to convert your data source from XML format to JSON before binding to the gantt.
- [html2canvas](#) (optional) - Necessary for the Printing or Export to image features.
- [canvg.js](#) (optional) - To dynamically create the svg gradient images.

What jQuery Versions are Supported?

The jQuery Gantt Package is compatible with the following versions:

- jQuery 1.x
 - From : jQuery 1.6
 - To : jQuery 1.11.3
- jQuery 2.x(does not support Internet Explorer 8)
 - From : jQuery 2.0.0
 - To : jQuery 2.1.4
- jQuery UI
 - From : jQuery UI 1.9.0
 - To : jQuery UI 1.11.4
- Knockout
 - From : knockout-2.0.0
 - To : knockout-3.2.0

Our samples in the install link to the jQuery 1.11.2 and jQuery UI 1.11.4 as shown below:

```
<script src="Src/Scripts/jquery-1.11.2.js" type="text/javascript"></script>
<script type="text/javascript" src="Src/Scripts/jquery-ui-1.11.4/jquery-ui.min.js"></script>
```

If you want to link to a different version of jQuery, you simply have to include the corresponding JS files in the sub-folders and then link to them like this:

```
<script src="Src/Scripts/jquery-1.7.1.js" type="text/javascript"></script>
<script type="text/javascript" src="Src/Scripts/jquery-ui-1.9.1/jquery-ui.min.js"></script>
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

IE 8 Support

IE 8 Support

Though a very old version with not sufficient support for latest HTML and JS features, owing to it's sufficiently large install base, we have extended our widgets to include support for this browser. One key challenge in supporting IE 8 is it's inability to support property extenders for objects. So, we have provided method variants for our properties which you can access with a "_M" prefix.

For example, here are some properties in our APIs that work with the later browsers:

```
var selectActivityView = ganttControl.SelectedItem;
var selectIndex = ganttControl.SelectedIndex;
var zeroTimeSpan = RQTimeSpan.Zero
var startTime = activity.StartTime;
var endTime = activity.EndTime;
```

The above is not compatible with IE8. You should instead use these APIs to create an IE8 compatible application:

```
var selectActivityView = ganttControl.SelectedItem_M();
var selectIndex = ganttControl.SelectedIndex_M();
var zeroTimeSpan = RQTimeSpan.Zero_M();
var startTime = activity.StartTime_M();
var endTime = activity.EndTime_M();
```

We have used the above approach in all our samples in our install to make them IE8 compatible. You can either choose to use the property based API or the method based API depending on whether or not you want to support IE8.

NOTE: When running in IE 8, you will see a console warning like this: "It appears that the gantt is loaded in a IE browser version earlier than IE9. Some features are not supported in earlier than IE9. Or you might be running IE 9+ in Compatibility View mode, which is also not supported by the gantt." This is a warning message for developers. The unsupported features are listed below.

- Printing and ExportToImage features.
- Row Drag And Drop feature in GridTable.
- Labels to the right of the task bar.
- Dynamically creating the Gradient images.

Gantt Control

Getting Started

RadiantQ jQuery Gantt Package

In HTML

Your First Gantt

Let us start with creating a new project directory, for example MyFirstGantt.

1) Gantt Widget Source JS files

To begin with, you need the JS Source files that are required by the Gantt Widget. These files are in the <install path>/Src folder. Simply copy over the whole Src folder into the above directory. This folder also has other dependant css files, etc. You can delete the "Src/bin" folder as that's not required for this HTML sample.

2) Sample Utility JS files

Some JS files with utility functions that enables inline-editing in the grid, etc. are in the <install path>/Samples/Scripts folder. Copy over that entire Scripts directory as well into MyFirstGantt.

3) Sample JSON Data

Create a SampleData.json file containing a list of sample tasks to be displayed in the gantt.

SampleData.json content:

```
[{
  "Name" : "Task 1",
  "ID" : 1,
  "StartTime" : "2012-02-02T00:00:00Z",
  "Effort" : "8:00:00",
  "Description" : "Description of Task 1"
},
{
  "Name" : "Task 2",
  "ID" : 2,
  "PredecessorIndices" : "1",
  "StartTime" : "2012-02-03T00:00:00Z",
  "Effort" : "16:00:00",
  "Description" : "Description of Task 2"
},
{
  "Name" : "Task 3",
  "ID" : 3,
  "StartTime" : "2012-02-02T00:00:00Z",
  "Effort" : "1.12:30:00",
  "ProgressPercent" : 90,
  "Description" : "Description of Task 3"
},
{
  "Name" : "Child Task 1",
  "ID" : 4,
  "IndentLevel" : 1,
  "StartTime" : "2012-02-03T00:00:00Z",
  "Effort" : "8:00:00",
  "ProgressPercent" : 75,
  "Description" : "Description of Task 3/Child Task 1"
},
{
  "Name" : "Child Task 2",
  "ID" : 5,
  "IndentLevel" : 1,
  "PredecessorIndices" : "4+8",
  "Description" : "Description of Task 3/Child Task 2"
},
{
  "Name" : "Grand Child Task 1",
  "ID" : 6,
  "IndentLevel" : 2,
  "StartTime" : "2012-02-03T00:00:00Z",
  "Effort" : "8:00:00",
  "Description" : "Description of Task 3/Child Task 1/Grand Child 1"
},
{
  "Name" : "Grand Child Task 2",
  "ID" : 7,
  "IndentLevel" : 2,
  "StartTime" : "2012-02-03T00:00:00Z",
  "Effort" : "16:00:00",
  "Description" : "Description of Task 3/Child Task 1/Grand Child 2"
},
{
  "Name" : "Child Task 3",
  "ID" : 8,
  "IndentLevel" : 1,
  "StartTime" : "2012-02-02T00:00:00Z",
  "Effort" : "16:00:00",
  "Description" : "Description of Task 3/Child Task 3"
},
{
  "Name" : "Task 4",
  "ID" : 9,
  "StartTime" : "2012-02-02T00:00:00Z",
```

4) HTML file including the Gantt Widget

Create a new HTML file inside the project directory (MyFirstGantt) and reference the following source files. Remember to link to the right version of the RadiantQ jQuery Gantt in the last reference below.

```
<head>
  <title></title>
  <link href="Src/Styles/jquery-ui-themes/smoothness/jquery-ui.css" rel
="stylesheet" />
  <link href="Src/Styles/radiantq.gantt.default.css" rel="stylesheet" />
  <link href="Src/Styles/VW.Grid.css" rel="stylesheet" />
  <script src="Src/Scripts/jquery-1.11.2.min.js" type="text/javascript"></script>
  <script type="text/javascript" src
="Src/Scripts/jquery-ui-1.11.4/jquery-ui.min.js" ></script>
  <script type="text/javascript" src="Src/Scripts/jquery.layout-latest.min.js"></
script>
  <script src="Src/Scripts/Utils/date.js" type="text/javascript"></script>
  <script src="Src/ResourceStrings/en-US.js" type="text/javascript"></script>
  <script src='Src/Scripts/VW.Grid.1.min.js' type='text/javascript'></script>
  <script src='Src/Scripts/RadiantQ-jQuery.gantt.5.0.trial.min.js' type
='text/javascript'></script>
</head>
```

Initializing the Gantt Table

Now you have to setup the different column you want to show in the GanttTable. You can do so by defining a collection of column objects like below.

```

var columns = [
  {
    field: "Activity_M().ID_M()",
    title: "ID",
    iseditable: false,
    width: 25
  },
  {
    field: "Activity_M().ActivityName_M()",
    title: "Activity Name",
    width: 200,
    editor: RadiantQ.Default.Template.ProjectGanttExpandableTextboxEditor(),
    template:
RadiantQ.Default.Template.ProjectGanttExpandableTextBlockTemplate()
  },
  {
    field: "Activity_M().StartTime_M()",
    title: "StartTime",
    width: 150,
    format: "MM/dd/yy",
    editor: "<input data-bind='ActivityTimeBinder:Activity_M().StartTime_M'
/>"
  },
  {
    field: "Activity_M().EndTime_M()",
    title: "EndTime",
    width: 150,
    format: "MM/dd/yy",
    cellalign: "center",
    editor: "<input data-bind='value:Activity_M().EndTime_M'
data-getvalueName='getDate' data-setvaluenam='setDate' data-valueUpdate='onBlur'
data-role=\"DateTimePicker\" />"
  },
  {
    field: "Activity_M().Effort_M()",
    title: "Effort",
    format: "" /*to call the .toString()*/,
    width: 100,
    editor: "<input data-bind='value:Activity_M().Effort_M' '
data-role=\"DurationPicker\" />"
  },
  {
    field: "Activity_M().ProgressPercent_M()",
    title: "ProgressPercent",
    width: 150,
    editor: "<input data-bind='value:Activity_M().ProgressPercent_M'
data-role=\"spinner\" data-options='{\"min\":0, \"max\": 100}' />"
  },
  {
    field: "Activity_M().Assignments_M()",
    title: "Resource",
    iseditable: false,
    template: '<div> ${
RadiantQ.Gantt.ValueConverters.ConverterUtils.GetResourcesText(data.Activity_M().Assi
gnments_M(), false) } </div>',
    width: 100
  },
  {
    field: "Activity.PredecessorIndexString",
    title: "PredecessorIndex",
    isParentEditable: false,
    template: "<div>${data.PredecessorIndexString || '' }</div>",
    editor: "<input data-bind='value:PredecessorIndexString'/>",
    width: 150
  }
];

```

Creating Gantt

Now include code to retrieve the json file you created above and then initialize the GanttControl widget binding it with the loaded data.

```

<script type="text/javascript">
  this.jsonData = null;
  var self = this;

  $.holdReady(true);
  $.ajax({
    type: "GET",
    dataType: 'json',
    url: 'SampleData.json',
    converters:
    {
      "text json": function (data) {
        //console.log(data);
        return $.parseJSON(data, true
        /*converts date strings to date objects*/
        , true
        /*converts ISO dates to local dates*/
        );
      }
    },
    success: function (data) {
      self.jsonData = data;
      $.holdReady(false);
    }
  });

  $(document).ready(function () {
    var anchorTime = new Date(2014, 01, 02, 00, 00, 00);
    var $gantt_container = $("#gantt_container");
    $gantt_container.GanttControl({
      ProjectStartDate: anchorTime,
      DataSource: self.jsonData,
      GanttTableOptions:{
        columns:columns
      },
      IDBinding: new RadiantQ.BindingOptions("ID"),
      NameBinding: new RadiantQ.BindingOptions("Name"),
      IndentLevelBinding: new RadiantQ.BindingOptions("IndentLevel"),
      StartTimeBinding: new RadiantQ.BindingOptions("StartTime"),
      EffortBinding: new RadiantQ.BindingOptions("Effort"),
      PredecessorIndicesBinding: new RadiantQ.BindingOptions(
"PredecessorIndices"),
      ProgressPercentBinding: new RadiantQ.BindingOptions(
"ProgressPercent"),
      DescriptionBinding: new RadiantQ.BindingOptions("Description"),
      GanttChartTemplateApplied: function (sender , args) {
        var $GanttChart = args.element;
        $GanttChart.GanttChart({ AnchorTime: anchorTime });
      }
    });
  });
</script>

<body>
  <!-- Div that will be transformed into the gantt widget above.-->
  <div id="gantt_container" style="height:550px;"></div>
  <!-- GanttTable and Gantttable contents comes here.-->
</body>

```


RadiantQ jQuery Gantt Package

In Angular

Angular Gantt sample

- Sample HTML template

The project gantt samples should be created using the element '`<RQ:GanttControl> </RQ:GanttControl>`'.

You can set the gantt options such as Datasource, AnchorTime, ResizeToFit, UseVirtualization etc., as illustrated here.

The datasource for gantt can be either from JSON or from local function(looped item source).

The gantt columns should be created with available properties. The client and editor templates is to be set using '`<ng-template> </ng-template>`' specifying it's absolute template references(prefixed with '#') as illustrated here.

```

<RQ:GanttControl [DataSourceUrl]
="/app/Samples/GanttControlSkeleton/GanttControlSkeleton.json" [AnchorTime]
="2018-02-02" Height="500px"
[ResizeToFit]="false" [AfterGanttWidgetInitializedCallback]
="AfterGanttWidgetInitializedCallback">

  <GanttTableOptions>
    <Columns>
      <Column field="Activity.ID" [width]="25" title="ID" [iseditable]
="false"></Column>
      <Column field="Activity.ActivityName" [width]="200" title="Activity Name"
[iseditable]="true" [isParentEditable]="true" [clientTemplate]="pgNameTemplate"
[clientEditorTemplate]="pgNameEditorTemplate"></Column>
      <Column field="Activity.StartTime" [width]="150" title="StartTime"
[format]="dateFormat" [clientEditorTemplate]="startTimeEditor"></Column>
      <Column field="Activity.EndTime" [width]="150" title="EndTime" [format]
="dateFormat" [clientEditorTemplate]="endTimeEditor"></Column>
      <Column field="Activity.Effort" [width]="100" title="Effort"
[clientTemplate]="effortTemplate" [clientEditorTemplate]="effortEditor"></Column>
      <Column field="Activity.ProgressPercent" [width]="100" title
="ProgressPercent" [clientEditorTemplate]="progressPercentEditor"></Column>
      <Column field="Activity.PredecessorIndexString" [width]="100" title
="Predecessor Index" [isParentEditable]="false" [clientTemplate]
="predecessorIndexTemplate" [clientEditorTemplate]="predecessorIndexEditor"></Column>
    </Columns>
  </GanttTableOptions>
</RQ:GanttControl>

<!--Column Templates-->
<ng-template #pgNameTemplate>
  <div #nameTemp>
    <div class="rq-grid-expand-indentWidth" [rqTemplateBinder]
="nameColTempl.style_indentWidth"></div>
    <div class="arrowContainer" [rqTemplateBinder]
="nameColTempl.style_arrowCont">
      <div id="arrow" [rqTemplateBinder]="nameColTempl.class_arrow" onclick="
RadiantQ.Gantt.ExpanderOnClick(this,event)"></div>
    </div>
    <div class="rq-grid-expander-text" data-bind="text:Activity.ActivityName"></
div>
    <div [getClientTemplate]="nameTemp.innerHTML"></div>
  </div>
</ng-template>
<ng-template #effortTemplate>
  <div #effortTemp>
    <div [getClientTemplate]="effortTemp.innerHTML" data-bind
="text:Activity.CumulativeEffort.toString()"></div>
  </div>
</ng-template>
<ng-template #predecessorIndexTemplate>
  <div #predTemp>
    <div [getClientTemplate]="predTemp.innerHTML" data-bind
='text:Activity.PredecessorIndexString'></div>
  </div>
</ng-template>

<!--Column Editors-->
<ng-template #pgNameEditorTemplate>
  <div #nameEditor>
    <div class="rq-grid-expand-indentWidth" [rqTemplateBinder]
="nameColTempl.style_indentWidth"></div>
    <div [rqTemplateBinder]="nameColTempl.style_arrowCont" class
="arrowContainer">
      <div id="arrow" onclick="RadiantQ.Gantt.ExpanderOnClick(this,event)"
[rqTemplateBinder]="nameColTempl.class_arrow"></div>
    </div>
    <div class="rq-grid-expander-text"><input data-bind

```

- Sample JSON Data

Create a SampleData.json file containing a list of sample tasks to be displayed in the gantt.

```
[{
  "Name" : "Task 1",
  "ID" : 1,
  "StartTime" : "2018-02-02T00:00:00Z",
  "Effort" : "8:00:00",
  "Resources" : "1",
  "Description" : "Description of Task 1"
},
{
  "Name" : "Task 2",
  "ID" : 2,
  "PredecessorIndices" : "1",
  "StartTime" : "2018-02-03T00:00:00Z",
  "Effort" : "16:00:00",
  "Resources" : "1",
  "Description" : "Description of Task 2"
},
{
  "Name" : "Task 3",
  "ID" : 3,
  "StartTime" : "2018-02-02T00:00:00Z",
  "Effort" : "1.12:30:00",
  "ProgressPercent" : 90,
  "Description" : "Description of Task 3"
},
{
  "Name" : "Child Task 1",
  "ID" : 4,
  "IndentLevel" : 1,
  "StartTime" : "2018-02-03T00:00:00Z",
  "Effort" : "8:00:00",
  "ProgressPercent" : 100,
  "Description" : "Description of Task 3/Child Task 1"
},
{
  "Name" : "Child Task 2",
  "ID" : 5,
  "IndentLevel" : 1,
  "PredecessorIndices" : "4+8",
  "Description" : "Description of Task 3/Child Task 2"
},
{
  "Name" : "Grand Child Task 1",
  "ID" : 6,
  "IndentLevel" : 2,
  "StartTime" : "2018-02-03T00:00:00Z",
  "Effort" : "8:00:00",
  "Description" : "Description of Task 3/Child Task 1/Grand Child 1"
},
{
  "Name" : "Grand Child Task 2",
  "ID" : 7,
  "IndentLevel" : 2,
  "StartTime" : "2018-02-03T00:00:00Z",
  "Effort" : "16:00:00",
  "Description" : "Description of Task 3/Child Task 1/Grand Child 2"
},
{
  "Name" : "Child Task 3",
  "ID" : 8,
  "IndentLevel" : 1,
  "StartTime" : "2018-02-02T00:00:00Z",
  "Effort" : "16:00:00",
  "Description" : "Description of Task 3/Child Task 3"
},
{
  "Name" : "Task 4",
```

json

- Sample TS

The 'GanttControlSkeleton.ts' file imports the TypeScript libraries from top-level 'ts' folder.

The sample template(GanttControlSkeleton.html) and its corresponding CSS (GanttControlSkeleton.css) can be declared using '@Component({})' decorator with selector as 'GanttControlSkeleton' specified as sample root tag in index.html

Here, the export class pass the value for gantt options which sets as template in 'GanttControlSkeleton.html' such as DataSource, AnchorTime, ResizetoFit, Columns and bindings.

```

/// <reference path="../../ts/radiantqgantts2.3.d.ts" />
/// <reference path="../../ts/jquery.ui.d.ts" />
/// <reference path="../../ts/datejs.d.ts" />

import { Component, ElementRef, ViewEncapsulation } from '@angular/core';

@Component({
  selector: 'GanttControlSkeleton',
  templateUrl: './app/Samples/GanttControlSkeleton/GanttControlSkeleton.html',
  styleUrls: ['./app/Samples/GanttControlSkeleton/GanttControlSkeleton.css'],
  encapsulation: ViewEncapsulation.None, // Fix: Styles for child elements not
  applied due to encapsulation(shadow dom).
})
export class GanttControlSkeleton {
  private $gcSkeleton: HTMLElement = null;
  nameColTempl = {
    style_indentWidth: { key: 'style', value: 'height: 1px; width:
    ${data.IndentWidth_M()}px' },
    style_arrowCont: { key: 'style', value: 'width: 12px; display:
    ${data.IsParent_M() ? "block" : "none"}' },
    class_arrow: { key: 'class', value: '${ data.IsExpanded_M() ? "
    rq-grid-expand-arrow rq-grid-collapse-arrow" : "rq-grid-expand-arrow"}
    rq-Ignore-click' }
  };
  dateFormat = Date["CultureInfo"]["formatPatterns"]["shortDate"];

  constructor(private elementRef: ElementRef) {
    this.$gcSkeleton = elementRef.nativeElement;
  }

  AfterGanttWidgetInitializedCallback($gantts_container: any) {
    // This gets fired once after the gantt widget is intialized with user
    options.
    var ganttControl = $gantts_container.data('GanttControl');
    // Here, user can customize the gantt using this 'ganttsControl' instance.
  };
}

```

GanttControlSkeleton.ts

The 'AfterGanttWidgetInitializedCallback' function gets fired once after the gantt widget is initialized with user options. Here, user can customize the gantt using the 'ganttsControl' instance

The gantt widget initialization process takes place in 'RQGanttSettings.ts' with user options.

This is illustrated in the following path:

In Angular :
\PlatformSamples\AngularSamples\app\Samples\GanttControlSkeleton\..

Also, illustrated to import both 'GanttControl and FlexyGantt' samples in one page in the following path:

In Angular : \PlatformSamples\AngularSamples\app\Samples\ResourceLoadView\...

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

In React

React Gantt sample.

Import the react libraries and the "GanttControl" component from its desired path like this, 'RQSrc/React_Components/GanttControl.jsx'.

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import GanttControl from 'RQSrc/React_Components/GanttControl.jsx';
import './GanttControlSkeleton.css';

export default class GanttControlSkeleton extends React.Component{
  constructor(){
    super();
    this.anchorTime = new Date("2014-02-01T00:00:00-00:00");
  }

  GanttChartTemplateApplied(sender, args) {
    var $GanttChart = args.element;
    $GanttChart.GanttChart({ AnchorTime: this.anchorTime });
  }

  render(){
    var height =
      {
        height: '495px'
      };
    return(
      <div style={height}>
        { /* passing Url,ProjectStartDate,GanttChartTemplateApplied as a
        props of GanttControl */}
        <GanttControl Url= './DataSource/GanttControlSkeleton.json'
          ProjectStartDate={this.anchorTime}
          GanttChartTemplateApplied={(sender, args) => this
          .GanttChartTemplateApplied(sender, args)}>/>
      </div>
    );
  }
}
```

```
ReactDOM.render(<GanttcontrolSkeleton />, document.getElementById('container'));
GanttcontrolSkeleton.jsx
```

The ReactDOM.render method can render the "GanttcontrolSkeleton" component into the DOM. Here, the second argument "container" specifies the mount element. Mount element defined inside the "index.html".

This is illustrated in the following path:

In React : \PlatformSamples\React\Samples\GanttcontrolSkeleton
 \GanttcontrolSkeleton.jsx.

-0-

*RadiantQ jQuery Gantt Package***In ASP.NET**

Create a new ASP.NET project in Visual Studio:

VS 2012 :FILE --> New --> Project --> Visual C# --> Web --> ASP.NET Web Forms Application, create a project.

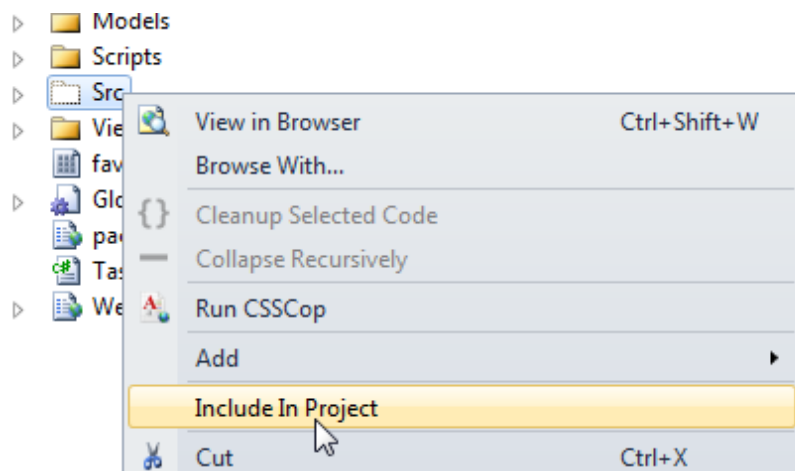
VS 2010 :FILE --> New --> Project --> Visual C# --> Web --> ASP.NET Web Application, create a project.

1) Gantt Widget Source JS files

To begin with, you need the JS Source files that are required by the Gantt Widget. These files are in the <install path>/Src folder. Copy over this folder into the above Project folder (though this folder is very big in size, it contains the required CSS, etc for all Themes, locales, etc. and not all of them will be loaded inside your page).

Go ahead and remove the bin folder inside this Src folder.

Then in the project's Solution Explorer click on the "Show All Files" toolbar item to show this newly included Src folder and include that in the project.



include Src folder in project

2) Sample Utility JS files

Some JS files with utility functions are in the <install path>/Samples/Scripts folder. Copy over the contents of the Scripts folder from the above install path into the Scripts folder in your project folder (when you create a new project a Scripts folder would be automatically created in your project folder).

Then include the newly added Script files into your project following the same procedure as in the previous step.

3) Create a sample Data Source(JSON Data)

You will typically use an Entity Model, ADO.NET, etc. to retrieve data from a database. But, to keep things simple, we will create a simple list of "tasks" and return it to the client from the server.

Create a new type called TaskInfo to represent the task instances. In Solution Explorer right click on the Project name, then Add --> New Item --> class (call it TaskInfo.cs) and define a class like below.

```
public class TaskInfo
{
    public string Name { get; set; }
    public int IndentLevel { get; set; }
    public DateTime StartTime { get; set; }
    public string Effort { get; set; }
    public double ProgressPercent { get; set; }
    public string Resources { get; set; }
    public int ID { get; set; }
    public string PredecessorIndices { get; set; }
    public int SortOrder { get; set; }
    public string Description { get; set; }
    public object Tag { get; set; }
    public int Priority { get; set; }
    public DateTime PlannedStartTime { get; set; }
    public DateTime PlannedEndTime { get; set; }
    public double Cost { get; set; }
    public DateTime EndTime { get; set; }
}
```

Then, add a new Generic handler to the project; this is going to provide the data source to the client page.

To add a Generic handler in the visual studio, in Solution Explorer right click on the Project name, then Add --> New Item --> Generic Handler, and create a new instance naming it, for example, DataHandler.ashx.

In the generic handler (DataHandler.ashx) create a list of TaskInfos, convert that list to a json using JavaScriptSerializer or any other "JSON serializer" to convert it to json data and add that to the Response.

```

public class DataHandler : IHttpHandler
{
    DateTime dt = DateTime.Today;
    public void ProcessRequest(HttpContext context)
    {
        List<TaskInfo> taskItems = new List<TaskInfo>
        {
            new TaskInfo { Name = "Task 1", ID = 1, StartTime =dt, Effort=TimeSpan
.FromDays(1).ToString(), Description = "Description of Task 1" },
            new TaskInfo { Name = "Child Task 1", ID = 2,IndentLevel=1, StartTime
=dt, Effort=TimeSpan.FromDays(2).ToString(), Description = "Description of Task 2" },
            new TaskInfo { Name = "Task 3", ID = 3, StartTime =dt, Effort=TimeSpan
.FromDays(4).ToString(), Description = "Description of Task 3" },
            new TaskInfo { Name = "Child Task 1", ID = 4,IndentLevel=1, StartTime
=dt, Effort=TimeSpan.FromDays(3).ToString(), Description = "Description of Task 4" },
            new TaskInfo { Name = "Child Task 2", ID = 5, IndentLevel=2, StartTime
=dt, Effort=TimeSpan.FromDays(1).ToString(), Description = "Description of Task 5" },
            new TaskInfo { Name = "Task 6", ID = 6, StartTime =dt, Effort=TimeSpan
.FromDays(2).ToString(), Description = "Description of Task 6" },
            new TaskInfo { Name = "Task 7", ID = 7, StartTime =dt, Effort=TimeSpan
.FromDays(1).ToString(), Description = "Description of Task 7" },
            new TaskInfo { Name = "Child Task 1", ID = 8,IndentLevel=1, StartTime
=dt, Effort=TimeSpan.FromDays(2).ToString(), PredecessorIndices="6+2" }
        };
        System.Web.Script.Serialization.JavaScriptSerializer serializer = new
System.Web.Script.Serialization.JavaScriptSerializer();
        context.Response.Write(serializer.Serialize(taskItems));
    }

    public bool IsReusable
    {
        get
        {
            return false;
        }
    }
}

```

4) ASPX file including the Gantt Widget

Create the .aspx sample file

In the visual studio SolutionExplorer right click on the Project name, then Add --> New Item --> Web Form(call it WebForm1.aspx).

Include the following namespace in the created aspx.

```

<%@ Register Assembly="RadiantQ.Web.JQGantt" Namespace="RadiantQ.WebForms.JQGantt"
TagPrefix="RQ" %>

```

Add the *RadiantQ.Web.JQGantt.dll* to your project reference, you can find the dll here:
<install folder> \Src\bin\DotNET4MVC4\RadiantQ.Web.JQGantt.dll.

Reference the following source files in the aspx inside the <head> tag. Remember to link to the right version of the RadiantQ jQuery Gantt in the last one line reference below.

```

<head runat="server">
  <title></title>
  <link id="themeChooser" href="<%= Page.ResolveClientUrl(
"~/Src/Styles/jquery-ui-themes/smoothness/jquery-ui.css") %>" rel="stylesheet" />
  <link id="default" href="<%= Page.ResolveClientUrl(
"~/Src/Styles/radiantq.gantt.default.css") %>" rel="stylesheet" />
  <link href="<%= Page.ResolveClientUrl("~/Src/Styles/VW.Grid.css") %>" rel
="stylesheet" />
  <script src="<%= Page.ResolveClientUrl("~/Src/Scripts/jquery-1.11.2.min.js") %>"
type="text/javascript"></script>
  <script type="text/javascript" src="<%= Page.ResolveClientUrl(
"~/Src/Scripts/jquery-ui-1.11.4/jquery-ui.min.js") %>"></script>
  <script type="text/javascript" src="<%= Page.ResolveClientUrl(
"~/Src/Scripts/jquery.layout-latest.min.js") %>"></script>
  <script src="<%= Page.ResolveClientUrl("~/Src/Scripts/Utils/date.js") %>" type
="text/javascript"></script>
  <script type="text/javascript" src="<%= Page.ResolveClientUrl(
"~/Src/ResourceStrings/en-US.js") %>"></script>
  <script type="text/javascript" src="<%= Page.ResolveClientUrl("~/Src
/Scripts/VW.Grid.1.min.js") %>"></script>
  <script type="text/javascript" src="<%= Page.ResolveClientUrl("~/Src
/Scripts/RadiantQ-jQuery.gantt.5.0.trial.min.js") %>"></script>
  <script src="<%= Page.ResolveClientUrl("~/Src/Scripts/RQGantt_Init.min.js") %>"
type="text/javascript"></script>
</head>

```

Creating Gantt

Now include code to retrieve the json file you created above and then initialize the GanttControl widget binding it with the loaded data.

In the WebForm1.aspx, within the default <form> tag in <body>, add this tag:

```

<body>
  <form id="form1" runat="server">
    <RQ:GanttControl ID="gantt" DataSourceUrl="DataHandler.ashx" Height="500px"
runat="server"/>
    <div>
    </div>
  </form>
</body>

```

Initializing the Gantt Table

Now you have to setup the different columns you want to show in the GanttTable. You can do so by defining columns inside GanttTableOptions property in GanttControl.

```

    <RQ:GanttControl ID="gantt" DataSourceUrl="DataHandler.ashx" Height="500px"
runat="server">
    <GanttTableOptions>
        <Columns>
            <GanttBase:Column field="Activity.ID" width="25" title="ID"
iseditable="false"></GanttBase:Column>
            <GanttBase:Column field="Activity.ActivityName" width="200" title
="Activity Name" clientTemplate="projectGanttNameTemplate" clientEditorTemplate
="projectGanttNameEditor"></GanttBase:Column>
            <GanttBase:Column field="Activity.StartTime" width="100" title
="StartTime" format="MM/dd/yy" clientEditorTemplate="startTimeEditor"></GanttBase:
Column>

                <GanttBase:Column field="Activity.EndTime" width="100" title
="EndTime" format="MM/dd/yy" clientEditorTemplate="endTimeEditor"></GanttBase:Column>
            <GanttBase:Column field="Activity.Effort" width="100" title
="Effort" format="" clientEditorTemplate="effortEditor"></GanttBase:Column>
            <GanttBase:Column field="Activity.ProgressPercent" width="100"
title="ProgressPercent" clientEditorTemplate="progressEditor"></GanttBase:Column>
            <GanttBase:Column field="Activity.Assignments" width="100" title
="Resource" iseditable="false" clientTemplate="resourceTemplate"></GanttBase:Column>
            <GanttBase:Column field="Activity.PredecessorIndexString" width
="100" title="Predecessor Index" clientEditorTemplate="predecessorEditor"
clientTemplate="predecessorTemplate" isParentEditable="false"></GanttBase:Column>
        </Columns>
    </GanttTableOptions>
</RQ:GanttControl>

<!-- Column Template-->
<script id="projectGanttNameTemplate" type="text/x-jquery-tmpl">
    <div class="rq-grid-expand-indentWidth" style="height: 1px; width: ${data
.IndentWidth_M()}px"></div>
    <div style="width: 12px; display: ${data.IsParent_M() ? "block" : "none" }"
class="arrowContainer">
        <div onclick="ExpanderOnClick(this,event)" id="Div2" class="${
data.IsExpanded_M() ? " rq-grid-expand-arrow rq-grid-collapse-arrow": "
rq-grid-expand-arrow"} rq-Ignore-click"></div>
    </div>
    <div class="rq-grid-expander-text">${data.Activity.ActivityName}</div>
</script>
<script id="predecessorTemplate" type="text/x-jquery-tmpl">
    <div>${data.PredecessorIndexString_M() || '' }</div>
</script>

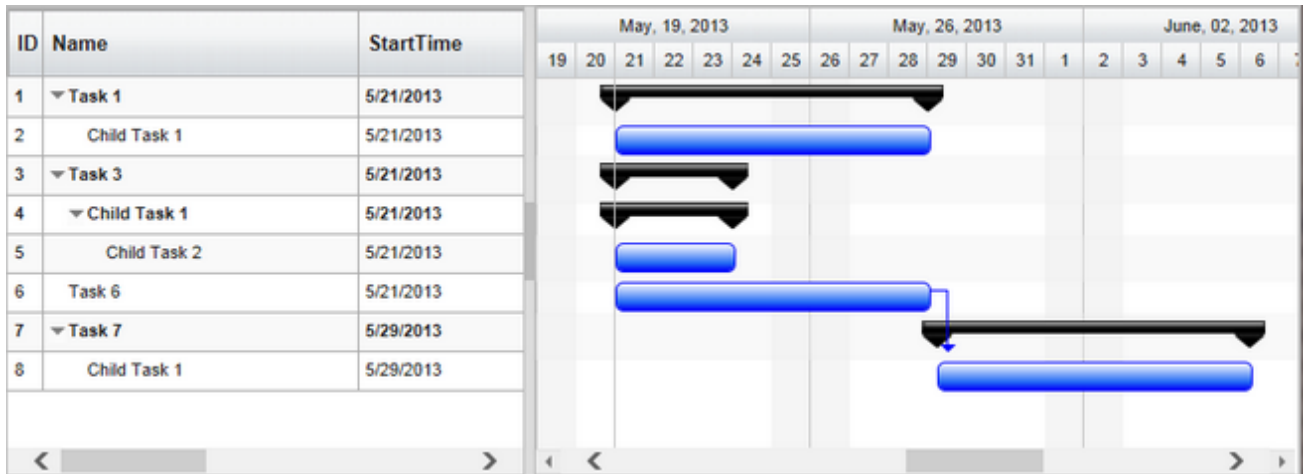
<!--Column Editors -->
<script id="startTimeEditor" type="text/x-jquery-tmpl">
    <input data-bind='ActivityTimeBinder: Activity_M().StartTime_M' />
</script>
<script id="endTimeEditor" type="text/x-jquery-tmpl">
    <input data-bind='value: Activity_M().EndTime_M' data-getvaluename='getDate'
data-setvaluename='setDate' data-valueupdate='onBlur' data-role="DateTimePicker" />
</script>
<script id="effortEditor" type="text/x-jquery-tmpl">
    <input data-bind='value: Activity_M().Effort_M' data-role="DurationPicker" />
</script>
<script id="progressEditor" type="text/x-jquery-tmpl">
    <input data-bind='value: Activity_M().ProgressPercent_M' data-role="spinner"
data-options='{ "min":0, "max": 100}' />
</script>
<script id="predecessorEditor" type="text/x-jquery-tmpl">
    <input data-bind='value: PredecessorIndexString' />
</script>

```

The gantt is now fully setup to display tasks returned from the ashx handler.

Make WebForm1.aspx the "Start Page". You can do so by right-clicking on this file in Solution Explorer and selecting "Set as Start Page".

Here is the resultant gantt:



Sample In Browser

Finally, take a look at this [topic](#) that show how you can clean up the Src folder in your project to remove unnecessary files.

For a sample that shows how to persist changes back in the database, please browse to this topic on [Persisting Changes](#).

RadiantQ jQuery Gantt Package

In ASP.NET MVC

Create a new ASP.NET MVC project in Visual Studio:

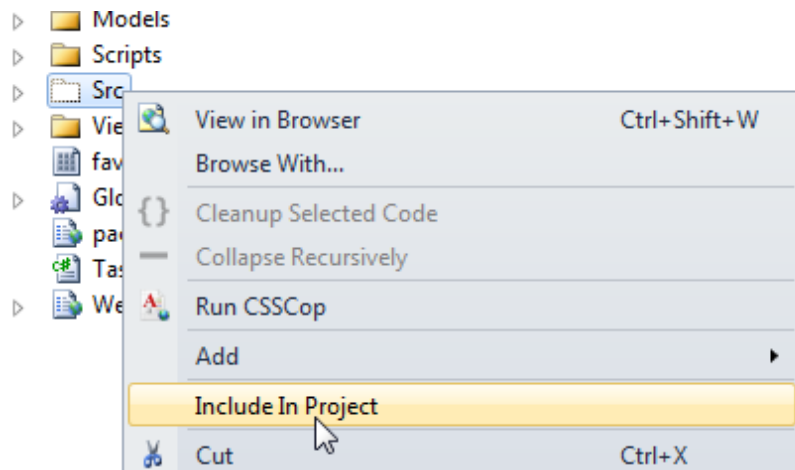
FILE --> New --> Project --> ASP.NET MVC 4 Web Application --> select Internet Application(Select Razor Engine)

1) Gantt Widget Source JS files

To begin with, you need the JS Source files that are required by the Gantt Widget. These files are in the <install path>/Src folder. Copy over this folder into the above Project folder (though this folder is very big in size, it contains the required CSS, etc for all Themes, locales, etc. and not all of them will be loaded inside your page).

Go ahead and remove the bin folder inside this Src folder.

Then in the project's Solution Explorer click on the "Show All Files" toolbar item to show this newly included Src folder and include that in the project.



include Src folder in project

2) Sample Utility JS files

Some JS files with utility functions are in the <install path>/Samples/Scripts folder. Copy over the contents of the Scripts folder from the above install path into the Scripts folder in your project folder (when you create a new project a Scripts folder would be automatically created in your project folder).

Then include the newly added Script files into your project following the same procedure as in the previous step.

3) Create a sample Data Source(JSON Data)

You will typically use an Entity Model, ADO.NET, etc. to retrieve data from a database. But, to keep things simple, we will create a simple list of "tasks" and return it to the client from the server.

Create a new type called TaskInfo to represent the task instances. In Solution Explorer right click on the Project name, then Add --> New Item --> class (call it TaskInfo.cs) and define a class like below.


```

public class TaskInfo
{
    public string Name { get; set; }
    public int IndentLevel { get; set; }
    public DateTime StartTime { get; set; }
    public string Effort { get; set; }
    public double ProgressPercent { get; set; }
    public string Resources { get; set; }
    public int ID { get; set; }
    public string PredecessorIndices { get; set; }
    public int SortOrder { get; set; }
    public string Description { get; set; }
    public object Tag { get; set; }
    public int Priority { get; set; }
    public DateTime PlannedStartTime { get; set; }
    public DateTime PlannedEndTime { get; set; }
    public double Cost { get; set; }
    public DateTime EndTime { get; set; }
}

```

Create a sample Data Source

Prepare a sample list of the above TaskInfo instances that represents the tasks in a project. This method must be inside the HomeController class(Controllers/HomeController.cs)

```

public class HomeController : Controller
{
    public ActionResult GetProjectGanttItemsource()
    {
        DateTime dt = DateTime.Today;

        List<TaskInfo> taskItems = new List<TaskInfo>
        {
            new TaskInfo { Name = "Task 1", ID = 1, StartTime =dt, Effort=
            TimeSpan.FromDays(1).ToString(), Description = "Description of Task 1" },
            new TaskInfo { Name = "Child Task 1", ID = 2,IndentLevel=1,
            StartTime =dt, Effort=TimeSpan.FromDays(2).ToString(), Description =
            "Description of Task 2" },
            new TaskInfo { Name = "Task 3", ID = 3, StartTime =dt, Effort=
            TimeSpan.FromDays(4).ToString(), Description = "Description of Task 3" },
            new TaskInfo { Name = "Child Task 1", ID = 4,IndentLevel=1,
            StartTime =dt, Effort=TimeSpan.FromDays(3).ToString(), Description =
            "Description of Task 4" },
            new TaskInfo { Name = "Child Task 2", ID = 5, IndentLevel=2,
            StartTime =dt, Effort=TimeSpan.FromDays(1).ToString(), Description =
            "Description of Task 5" },
            new TaskInfo { Name = "Task 6", ID = 6, StartTime =dt, Effort=
            TimeSpan.FromDays(2).ToString(), Description = "Description of Task 6" },
            new TaskInfo { Name = "Task 7", ID = 7, StartTime =dt, Effort=
            TimeSpan.FromDays(1).ToString(), Description = "Description of Task 7" },
            new TaskInfo { Name = "Child Task 1", ID = 8,IndentLevel=1,
            StartTime =dt, Effort=TimeSpan.FromDays(2).ToString(), PredecessorIndices=
            "6+2" }
        };

        return this.Json(taskItems, JsonRequestBehavior.AllowGet);
    }
}

```

4) RadiantQ Assembly and Script References

Add the *RadiantQ.Web.JQGantt.dll* to your project reference, you can find the dll here:
<install folder>\Src\bin\DotNET4MVC4\RadiantQ.Web.JQGantt.dll. (Or use the MVC3 equivalent)

In your *_Layout.cshtml*, include the following script and css references required by the gantt:

Remember to link to the right version of the jquery Gantt package in the last but one line reference below.

```
<script src="~/Src/Scripts/jquery-1.11.2.min.js"></script>
<link id="themeChooser" href
="~/Src/Styles/jquery-ui-themes/smoothness/jquery-ui.css" rel="stylesheet" />
<link id="default" href="~/Src/Styles/radiantq.gantt.default.css" rel
="stylesheet" />
<link id="gridCss" href="~/Src/Styles/VW.Grid.css" rel="stylesheet" />
<script src="~/Src/Scripts/jquery-ui-1.11.4/jquery-ui.min.js"></script>
<script type="text/javascript" src="~/Src/Scripts/jquery.layout-latest.min.js"></
script>
<script src="~/Src/Scripts/Utils/date.js"></script>
<script src="~/Src/ResourceStrings/en-US.js"></script>
<script src='~/Src/Scripts/VW.Grid.1.min.js' type='text/javascript'></script>
<script src='~/Src/Scripts/RadiantQ-jQuery.Gantt.5.0.trial.min.js' type
='text/javascript'></script>

<script src='~/Src/Scripts/RQGantt_Init.min.js' type='text/javascript'></script>
```

Also make sure you are not including the jQuery file in *_Layout.cshtml* (since we are referencing it above). By default the jQuery file including line look likes this in *_Layout.cshtml*,

```
@Scripts.Render("~/bundles/jquery")
```

And finally, include the following namespaces in *Web.config* within the existing *System.Web* tag as follows

```
<configuration>
<System.Web>
<pages>
<namespaces>
<add namespace="RadiantQMVC" />
<add namespace="RadiantQ.Web.JQGantt" />
<add namespace="RadiantQ.Web.JQGantt.Common" />
</namespaces>
</pages>
</System.Web>
</configuration>
```

5) CSHTML file including the Gantt Widget

Create the *.cshtml* sample file

In the visual studio right click on the views --> Home folder and Add --> View (call it as *view1.cshtml*)

And Include this Following namespaces in your *cshtml* page.

```
@using RadiantQMVC
@using RadiantQ.Web.JQGantt;
@using RadiantQ.Web.JQGantt.Common
```

Creating GanttControl

Now include code to retrieve the json file you created above and then initialize the GanttControl widget binding it with the loaded data.

(This could be added right at the bottom of the newly created cshtml file)

```
@Html.JQProjectGantt(
    new JQProjectGanttSettings()
    {
        ControlId = "gantt_container",

        DataSourceUrl = new Uri("/Home/GetProjectGanttItemsource", UriKind
.RelativeOrAbsolute),
        Options = new ProjectGanttOptions()
        {
            IDBinding= new Binding("ID"),
            NameBinding = new Binding("Name"),
            IndentLevelBinding = new Binding("IndentLevel"),
            StartTimeBinding = new Binding("StartTime"),
            EffortBinding = new Binding("Effort"),
            PredecessorIndicesBinding= new Binding("PredecessorIndices"),
            ProgressPercentBinding= new Binding("ProgressPercent"),
            DescriptionBinding= new Binding("Description")
        }
    })
<!-- Div that will be transformed into the gantt widget above.-->
<div id="gantt_container" style="height:450px;">
</div>
```

Initializing the Gantt Table

Now you have to setup the different column you want to show in the GanttTable. You can do so by defining GanttTableOptions as shown below.

(Only copy over the GanttTableOptions related code from below to the already copied over JQProjectGanttSettings code).

```
@Html.JQProjectGantt(
    new JQProjectGanttSettings()
    {
        ControlId = "gantt_container",

        DataSourceUrl = new Uri("/Home/GetProjectGanttItemsource", UriKind
.RelativeOrAbsolute),
        Options = new ProjectGanttOptions()
        {

            GanttTableOptions = new GanttTableOptions()
            {

                Columns = new Columns(){
                    new Column(){
```

```

        field= "Activity.ID",
        title= "ID",
        width= 40,
        iseditable=false
    },
    new Column(){
        field= "Activity.ActivityName",
        title= "Name",
        width= 200,
        clientEditorTemplate= "projectGanttNameEditor",
        clientTemplate = "projectGanttNameTemplate"
    },
    new Column(){
        field= "Activity.StartTime",
        title= "StartTime",
        width= 150,
        format= "MM/dd/yy",
        cellalign= "center",
        editor= "<input
data-bind='ActivityTimeBinder:Activity.StartTime' data-getvalueName='getDate'
data-setvaluenname='setDate' data-valueUpdate='onClose' data-role=\"DateTimePicker\"
/>"
    },
    new Column(){
        field= "Activity.EndTime",
        title= "EndTime",
        width= 150,
        format= "MM/dd/yy",
        cellalign= "center",
        editor= "<input data-bind='value:Activity.EndTime'
data-getvalueName='getDate' data-setvaluenname='setDate' data-valueUpdate='onClose'
data-role=\"DateTimePicker\" />"
    },
    new Column(){
        field= "Activity.Effort",
        title= "Effort",
        format= "" /*to call the .toString()*/,
        width= 100,
        editor= "<input data-bind='value:Activity.Effort'
style='height:18px' data-role=\"DurationPicker\" />"
    },
    new Column(){
        field= "Activity.ProgressPercent",
        title="ProgressPercent",
        width= 100,
        editor= "<input style='height:18px'
data-bind='value:Activity.ProgressPercent' data-role=\"spinner\"
data-options='{\"min\":0, \"max\": 100}' />"
    },
    new Column(){
        field= "Activity.Assignments",
        title= "Resource",
        iseditable=false,
        template=
"<div>${RadiantQ.Gantt.ValueConverters.ConverterUtils.GetResourcesText(data.Activity_
M().Assignments_M(), false)}</div>",
        editor= "<input
data-bind='ResourcePickerBinder:Activity.Assignments' />",
        width= 200
    },
    new Column(){

```

```

        field= "Activity.PredecessorIndexString",
        title= "PredecessorIndices",
        template= "<div>${data.PredecessorIndexString || ''
}</div>",
        editor= "<input data-bind='value:PredecessorIndexString'
/>",
        width= 150
    }
}
},
IDBinding= new Binding("ID"),
NameBinding = new Binding("Name"),
IndentLevelBinding = new Binding("IndentLevel"),
StartTimeBinding = new Binding("StartTime"),
EffortBinding = new Binding("Effort"),
PredecessorIndicesBinding= new Binding("PredecessorIndices"),
ProgressPercentBinding= new Binding("ProgressPercent"),
DescriptionBinding= new Binding("Description")
}
})

```

Finally, define the following templates used in the Name column definition.

```

<script id="projectGanttNameTemplate" type="text/x-jquery-tmpl">
<div class="rq-grid-expand-indentWidth" style="height: 1px; width: ${data.IndentWidth}
px"></div>
    <div style="width: 12px; display: ${data.IsParent ? "block" : "none" }" class
="arrowContainer">
        <div onclick="ExpanderOnClick(this,event)" id="arrow" class="${
data.IsExpanded ? " rq-grid-expand-arrow rq-grid-collapse-arrow":
"rq-grid-expand-arrow"} rq-Ignore-click"></div>
    </div>
    <div class="rq-grid-expander-text">${data.Activity.ActivityName}</div>
</script>
<script id="projectGanttNameEditor" type="text/x-jquery-tmpl">
    <div class="rq-grid-expand-indentWidth" style="height: 1px; width: ${data
.IndentWidth_M()}px"></div>
    <div style="width: 12px; display: ${data.IsParent_M() ? "block" : "none" }"
class="arrowContainer">
        <div onclick="ExpanderOnClick(this,event)" id="arrow" class="${
data.IsExpanded_M() ? " rq-grid-expand-arrow rq-grid-collapse-arrow":
"rq-grid-expand-arrow"} rq-Ignore-click"></div>
    </div>
    <div class="rq-grid-expander-text"><input data-bind='value:
Activity_M().ActivityName_M' /></div>
</script>

```

The gantt is now fully setup to display tasks returned from the controller.

Create a ActionResult in Homecontroller that will return your page as view

```

public ActionResult View1()
{
    return View("View1");
}

```

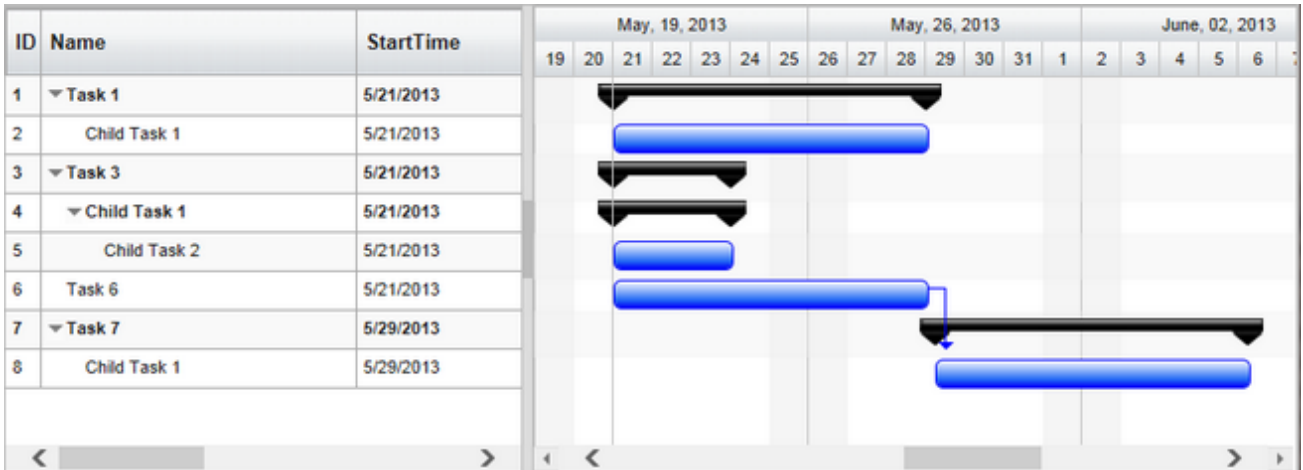
In App_Start/RouteConfig.cs change the routes action to "View1" (to make it your default View)

```

routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "View1", id = UrlParameter.Optional
}
);

```

Here is the resultant gantt:



Sample In Browser

Finally, take a look at this [topic](#) that show how you can clean up the Src folder in your project to remove unnecessary files.

For a sample that shows how to persist changes back in the database, please browse to this topic on [Persisting Changes](#).

© RadiantQ 2009-2018. All Rights Reserved.

RadiantQ jQuery Gantt Package

In PHP

Your First Gantt

Let us start with creating a new project directory, for example MyFirstGantt.

1) Gantt Widget Source JS files

To begin with, you need the JS Source files that are required by the Gantt Widget. These files are in the <install path>/Src folder. Simply copy over the whole Src folder into the above sample directory. This folder also has other dependant css files. You can delete the "Src/bin" folder as that's not required for this PHP sample.

Then copy over the PHP library files which are inside the <install path>/PlatformSamples/PHPSamples/lib folder into the sample directory.

2) Sample Utility JS files

Some JS files with utility functions that enables inline-editing in the grid, etc. are in the <install path>/Samples/Scripts folder. Copy over that entire Scripts directory as well into MyFirstGantt.

3) Sample JSON Data

Create a SampleData.json file containing a list of sample tasks to be displayed in the gantt.

SampleData.json content:

```
[{
  "Name" : "Task 1",
  "ID" : 1,
  "StartTime" : "2012-02-02T00:00:00Z",
  "Effort" : "8:00:00",
  "Description" : "Description of Task 1"
},
{
  "Name" : "Task 2",
  "ID" : 2,
  "PredecessorIndices" : "1",
  "StartTime" : "2012-02-03T00:00:00Z",
  "Effort" : "16:00:00",
  "Description" : "Description of Task 2"
},
{
  "Name" : "Task 3",
  "ID" : 3,
  "StartTime" : "2012-02-02T00:00:00Z",
  "Effort" : "1.12:30:00",
  "ProgressPercent" : 90,
  "Description" : "Description of Task 3"
},
{
  "Name" : "Child Task 1",
  "ID" : 4,
  "IndentLevel" : 1,
  "StartTime" : "2012-02-03T00:00:00Z",
  "Effort" : "8:00:00",
  "ProgressPercent" : 75,
  "Description" : "Description of Task 3/Child Task 1"
},
{
  "Name" : "Child Task 2",
  "ID" : 5,
  "IndentLevel" : 1,
  "PredecessorIndices" : "4+8",
  "Description" : "Description of Task 3/Child Task 2"
},
{
  "Name" : "Grand Child Task 1",
  "ID" : 6,
  "IndentLevel" : 2,
  "StartTime" : "2012-02-03T00:00:00Z",
  "Effort" : "8:00:00",
  "Description" : "Description of Task 3/Child Task 1/Grand Child 1"
},
{
  "Name" : "Grand Child Task 2",
  "ID" : 7,
  "IndentLevel" : 2,
  "StartTime" : "2012-02-03T00:00:00Z",
  "Effort" : "16:00:00",
  "Description" : "Description of Task 3/Child Task 1/Grand Child 2"
},
{
  "Name" : "Child Task 3",
  "ID" : 8,
  "IndentLevel" : 1,
  "StartTime" : "2012-02-02T00:00:00Z",
  "Effort" : "16:00:00",
  "Description" : "Description of Task 3/Child Task 3"
},
{
  "Name" : "Task 4",
  "ID" : 9,
  "StartTime" : "2012-02-02T00:00:00Z",
```


4) PHP file including the Gantt Widget

Create a new PHP file inside the project directory (MyFirstGantt) and reference the following source files. Remember to link to the right version of the RadiantQ jQuery Gantt in the last reference below.

```
<!DOCTYPE html>
  <link href="Src/Styles/jquery-ui-themes/smoothness/jquery-ui.css" rel
="stylesheet" />
  <link href="Src/Styles/radiantq.gantt.default.css" rel="stylesheet" />
  <link href="Src/Styles/VW.Grid.css" rel="stylesheet" />
  <script src="Src/Scripts/jquery-1.11.2.min.js" type="text/javascript"></script>
  <script src="Src/Scripts/jquery-ui-1.11.4/jquery-ui.min.js" type
="text/javascript"></script>
  <script type="text/javascript" src="Src/Scripts/jquery.layout-latest.min.js"></
script>
  <script src="Src/Scripts/Utils/date.js" type="text/javascript"></script>
  <script src="Src/ResourceStrings/en-US.js" type="text/javascript"></script>
  <script src='Src/Scripts/VW.Grid.1.min.js' type='text/javascript'></script>
  <script src='Src/Scripts/RadiantQ-jQuery.gantt.5.0.trial.min.js' type
='text/javascript'></script>>
  <script src="Src/Scripts/RQGantt_Init.min.js" type="text/javascript"></script>
  <!-- It automatically includes required PHP gantt extension files-->
  require_once "lib/AutoLoad.php"
```

Creating Gantt

Initialize the GanttControl widget as follows within your PHP pages

```

<?php
//Gantt Settings
$ganttSettings = new RadiantQ\Gantt\GanttSettings();
//GanttControl Options.
$options= new RadiantQ\Gantt\ProjectGanttOptions();
$ganttSettings->DataSourceUrl= "SampleData.json";

//column defintions
$IDcolumn = new RadiantQ\Gantt\Column();
$IDcolumn->field= "Activity.ID";
$IDcolumn->title= "ID";
$IDcolumn->width= 40;
$IDcolumn->iseditable=false;

$NameColumn = new RadiantQ\Gantt\Column();
$NameColumn->field="Activity.ActivityName";
$NameColumn->title= "Name";
$NameColumn->width= 200;
$NameColumn->clientEditorTemplate= "projectGanttNameEditor";
$NameColumn->clientTemplate = "projectGanttNameTemplate";

$STColumn = new RadiantQ\Gantt\Column();
$STColumn->field="Activity.StartTime";
$STColumn->title= "StartTime";
$STColumn->width= 150;
$STColumn->format= "MM/dd/yy";
$STColumn->editor= "<input
data-bind='ActivityTimeBinder:Activity_M().StartTime_M' />";

$ETColumn = new RadiantQ\Gantt\Column();
$ETColumn->field="Activity.EndTime";
$ETColumn->title= "EndTime";
$ETColumn->width= 150;
$ETColumn->format= "MM/dd/yy";
$ETColumn->editor= "<input data-bind='value:Activity_M().EndTime_M'
data-getvalueName='getDate' data-setvalueName='setDate' data-valueUpdate='onBlur'
data-role=\"DateTimePicker\" />";

$EffortColumn = new RadiantQ\Gantt\Column();
$EffortColumn->field= "Activity.Effort";
$EffortColumn->title= "Effort";
$EffortColumn->format= "" /*to call the .toString()*/;
$EffortColumn->template= '<div> ${
data.Activity_M().CumulativeEffort_M().toString() } </div>';
$EffortColumn->editor= "<input data-bind='value:Activity_M().Effort_M'
data-role=\"DurationPicker\" />";
$EffortColumn->width= 100;

$ProgressColumn = new RadiantQ\Gantt\Column();
$ProgressColumn->field= "Activity.ProgressPercent";
$ProgressColumn->title="ProgressPercent";
$ProgressColumn->width= 100;
$ProgressColumn->editor= "<input style='height:18px'
data-bind='value:Activity.ProgressPercent' data-role=\"spinner\"
data-options='{\"min\":0, \"max\": 100}' />";

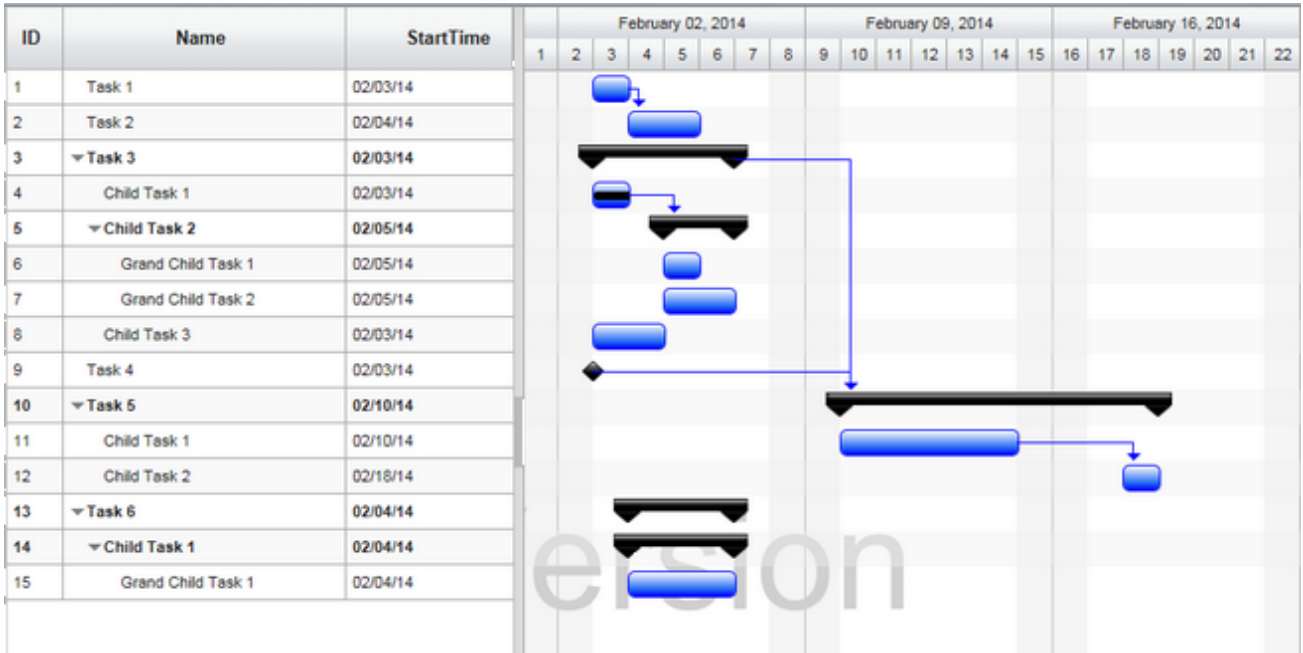
$ResourceColumn = new RadiantQ\Gantt\Column();
$ResourceColumn->field= "Activity.Assignments";
$ResourceColumn->title= "Resource";
$ResourceColumn->iseditable=false;
$ResourceColumn->template=
"<div>\${RadiantQ.Gantt.ValueConverters.ConverterUtils.GetResourcesText(data.Activity
_M().Assignments_M(), false)}</div>";
$ResourceColumn->editor= "<input
data-bind='ResourcePickerBinder:Activity.Assignments' />";
$ResourceColumn->width= 200;

```

The content of your MyFirstGantt directory should look like this:

- Src
- Scripts
- json file
- PHP file

Open the FirstGanttSample.PHP in the browser and you should see this in the browser:



FirstGanttSample in browser

Finally, take a look at this [topic](#) that show how you can clean up the Src folder in your project to remove unnecessary files.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

In Typescript

Create a new ASP.NET project in Visual Studio:

VS 2012 : FILE --> New --> Project --> Installed--> Templates --> Other Language --> TypeScript, create a project.

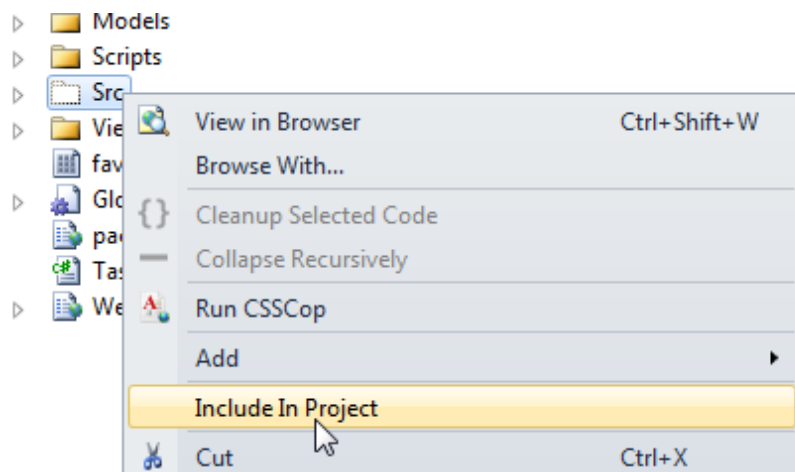
The Gantt Package includes the necessary Gantt TypeScript interfaces to help you develop your Web application just like any other Type-Safe language with compile time checks.

1) Gantt Widget Source JS files

To begin with, you need the JS Source files that are required by the Gantt Widget. These files are in the <install path>/Src folder. Copy over this folder into the above Project folder (though this folder is very big in size, it contains the required CSS, etc for all Themes, locales, etc. and not all of them will be loaded inside your page).

Go ahead and remove the bin folder inside this Src folder.

Then in the project's Solution Explorer click on the "Show All Files" toolbar item to show this newly included Src folder and include that in the project.



include Src folder in project

2) Sample Utility JS files

Some JS files with utility functions that enables inline-editing in the grid, etc. are in the <install path>/Samples/Scripts folder. Copy over the contents of the Scripts folder from the above install path into the Scripts folder in your project folder (when you create a new project a Scripts folder would be automatically created in your project folder).

Then include the newly added Script files into your project following the same procedure as in the previous step.

3) Sample JSON Data

Create a SampleData.json file containing a list of sample tasks to be displayed in the gantt.

SampleData.json content:

```
[{
  "Name" : "Task 1",
  "ID" : 1,
  "StartTime" : "2012-02-02T00:00:00Z",
  "Effort" : "8:00:00",
  "Description" : "Description of Task 1"
},
{
  "Name" : "Task 2",
  "ID" : 2,
  "PredecessorIndices" : "1",
  "StartTime" : "2012-02-03T00:00:00Z",
  "Effort" : "16:00:00",
  "Description" : "Description of Task 2"
},
{
  "Name" : "Task 3",
  "ID" : 3,
  "StartTime" : "2012-02-02T00:00:00Z",
  "Effort" : "1.12:30:00",
  "ProgressPercent" : 90,
  "Description" : "Description of Task 3"
},
{
  "Name" : "Child Task 1",
  "ID" : 4,
  "IndentLevel" : 1,
  "StartTime" : "2012-02-03T00:00:00Z",
  "Effort" : "8:00:00",
  "ProgressPercent" : 75,
  "Description" : "Description of Task 3/Child Task 1"
},
{
  "Name" : "Child Task 2",
  "ID" : 5,
  "IndentLevel" : 1,
  "PredecessorIndices" : "4+8",
  "Description" : "Description of Task 3/Child Task 2"
},
{
  "Name" : "Grand Child Task 1",
  "ID" : 6,
  "IndentLevel" : 2,
  "StartTime" : "2012-02-03T00:00:00Z",
  "Effort" : "8:00:00",
  "Description" : "Description of Task 3/Child Task 1/Grand Child 1"
},
{
  "Name" : "Grand Child Task 2",
  "ID" : 7,
  "IndentLevel" : 2,
  "StartTime" : "2012-02-03T00:00:00Z",
  "Effort" : "16:00:00",
  "Description" : "Description of Task 3/Child Task 1/Grand Child 2"
},
{
  "Name" : "Child Task 3",
  "ID" : 8,
  "IndentLevel" : 1,
  "StartTime" : "2012-02-02T00:00:00Z",
  "Effort" : "16:00:00",
  "Description" : "Description of Task 3/Child Task 3"
},
{
  "Name" : "Task 4",
  "ID" : 9,
  "StartTime" : "2012-02-02T00:00:00Z",
```

4) HTML file including the Gantt Widget

Create a new HTML file in your project and reference the following source files. Remember to link to the right version of the RadiantQ jQuery Gantt in the last reference below.

```
<head>
  <title></title>
  <link href="Src/Styles/jquery-ui-themes/smoothness/jquery-ui.css" rel
="stylesheet" />
  <link href="Src/Styles/radiantq.gantt.default.css" rel="stylesheet" />
  <link href="Src/Styles/VW.Grid.css" rel="stylesheet" />
  <script src="Src/Scripts/jquery-1.11.2.min.js" type="text/javascript"></script>
  <script src="Src/Scripts/jquery-ui-1.11.4/jquery-ui.min.js" type
="text/javascript"></script>
  <script type="text/javascript" src="Src/Scripts/jquery.layout-latest.min.js"></
script>
  <script src="Src/Scripts/Utils/date.js" type="text/javascript"></script>
  <script src="Src/ResourceStrings/en-US.js" type="text/javascript"></script>
  <script src='Src/Scripts/VW.Grid.1.min.js' type='text/javascript'></script>
  <script src='Src/Scripts/RadiantQ-jQuery.gantt.5.0.trial.min.js' type
='text/javascript'></script>
</head>

<body>
  <div id="pagecontent" style="height: 600px;">
    <div id="ganttt_container" style="height: 100%;">
      </div>
    </div>
</body>
```

5) TypeScript file.

Add a new TypeScript file(myApp.ts) next to that HTML in your project and refer the resultant js in html.

```
<head>
  other script fils.
  <script src=myApp.js type='text/javascript'></script>
</head>
```

6) Create Ganttcontrol widget inside the TypeScript file.

```

$.ajax({
  type: "GET",
  dataType: 'json',
  url: 'GanttControlSkeleton.json',
  converters:
  {
    "text json": function (data) {
      //console.log(data);
      return $.parseJSON(data, true
      /*converts date strings to date objects*/
      , true
      /*converts ISO dates to local dates*/
      );
    }
  },
  success: function (data) {
    loadGantt(data);
  }
});
function loadGantt(datasourcejson) {

  var columns = [
    {
      field: "Activity_M().ID_M()",
      title: "ID",
      width: 20
    },
    {
      field: "Activity_M().ActivityName_M()",
      title: "Activity Name",
      width: 200,
      editor:
RadiantQ.Default.Template.ProjectGanttExpandableTextboxEditor(),
      template:
RadiantQ.Default.Template.ProjectGanttExpandableTextBlockTemplate()
    },
    {
      field: "Activity_M().StartTime_M()",
      title: "StartTime",
      width: 100,
      format: "MM/dd/yy",
      cellalign: "center",
      editor: "<input data-bind='
ValueBinder.ActivityTimeBinder:Activity_M().StartTime_M' />"
    },
    {
      field: "Activity_M().EndTime_M()",
      title: "EndTime",
      width: 100,
      format: "MM/dd/yy",
      cellalign: "center",
      editor: "<input data-bind='value:Activity_M().EndTime_M'
data-getvalueName='getDate' data-setvalueName='setDate'
data-valueUpdate='onBlur' data-role=\"DateTimePicker\" />"
    },
    {
      field: "Activity_M().Effort_M()",
      title: "Effort",
      format: "" /*to call the .toString()*/,
      width: 100,
      editor: "<input data-bind='value:Activity_M().Effort_M'
style='height:18px' data-role=\"DurationPicker\" />"
    },
    {
      field: "Activity_M().ProgressPercent_M()",
      title: "ProgressPercent",
      width: 100,

```

Finally, take a look at this [topic](#) that show how you can clean up the Src folder in your project to remove unnecessary files.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

Gantt Basics

GanttTable

RadiantQ jQuery Gantt Package

Setting up the GanttTable

In this topic, we go through the different steps required to setup columns in the GanttTable portion of the gantt. Note that the gantt does not automatically populate the table columns, you will have to set this up with column definitions. Among other things this gives you the flexibility to customize the columns shown in the table (take a look at the VWGrid user guide).

At the bottom of this topic, you will see references to samples where a full-fledged implementation of these features can be seen.

Creating GanttTable column

Each columns in the GanttTable should be defined via column definition. This is the column definition for the Start Time Column:

```
{
  field: "Activity_M().StartTime_M()",
  title: "StartTime",
  width: 150,
  format: "MM/dd/yy",
  editor: "<input data-bind='ActivityTimeBinder:Activity_M().StartTime_M' />"
},
```

This is illustrated in most of our samples. For example,

In HTML : ..\Samples\GanttControlTableCustomization.htm.
In ASP.NET MVC : ..\Views\Home\ProjectGantt\GanttControlTableCustomization.cshtml.
In ASP.NET : ..\Samples\ProjectGantt\GanttControlTableCustomization.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

GanttTable Editing

VWGrid supports incell and popup editing, incell is the default editing mode. The following steps illustrates how to setup editing in GanttTable.

Incell Editing:

Step 1: Create an editor template for the column.

Define a custom cell-editor template.

```
<script id="projectGanttNameEditor" type="text/x-jquery-tmpl">
  <div class="rq-grid-expand-indentWidth" style="height: 1px; width: ${data
  .IndentWidth_M()}px;"></div>
  <div style="width: 12px; display: ${data.IsParent_M() ? "block":"none" }" class
  ="arrowContainer">
    <div onclick="ExpanderOnClick(this,event)" id="arrow" class="${
  data.IsExpanded_M() ? " rq-grid-expand-arrow rq-grid-collapse-arrow":
  "rq-grid-expand-arrow"} rq-Ignore-click"></div>
  </div>
  <div class="rq-grid-expander-text"><input data-bind="value:
  Activity_M().ActivityName_M" /></div>
</script>
```

Step 2: Specify the Editor in Column definition

Specify the above editor in the Column definition.

```
{
  field: "Activity_M().ActivityName_M()",
  title: "Activity Name",
  width: 200,
  editor: $.trim($("#projectGanttNameEditor").html()),
  template: RadiantQ.Default.Template.ProjectGanttExpandableTextBlockTemplate()
},
```

Step 3: Specify the property, if editable and if parent editable

You must specify which property to edit, in the column definition using "column.field" field. The object that will be in the data context while editing a row is the "activity view" instance, so you can refer any property or sub-property of the activity view instance.

So, you can simply reference the properties of the activity view instance like this: **field: "Activity.ActivityName"**.

Or you can reference any property in your data bound object representing the task, like this: **field: "Activity.DataSource.Cost"** this is illustrated in ..Samples/GanttControlCostTracking.htm)

```

$ganttt_container = $('#ganttt_container');
$ganttt_container.GanttControl({
    .....
    .....
    GanttTableOptions: {
        columns: [{
            field: "Activity.DataSource.Cost",
            title: "Cost",
            editor: "<input data-bind='value:Activity.DataSource.Cost'
data-role=\"spinner\" />",
            template: "<div>${ToDollarString(data)}</div>",
            width: 100,
            iseditable: true,
            isParentEditable: false
        }],
        startEdit: function (cell, dataItem, column) {
            // Preventing the row edit when cost is geater than 1000
            if (data.activity.DataSource.Cost >= 1000)
                return false;
        }
    },
    .....
    .....
});

```

This is illustrated in this samples:

In HTML : ..\Samples\GanttControlTableCustomization.htm.
 In ASP.NET MVC : ..\Views\Home\ProjectGantt\GanttControlTableCustomization.cshtml.
 In ASP.NET : ..\Samples\ProjectGantt\GanttControlTableCustomization.aspx.

Popup Editing :

Step 1: Create an custom editor in Popup.

By default, the Gantt provides you an expandable textbox element for editing the name column value. You can use the following custom editor function to use input element instead of expandable textbox for editing the name value in popup.

```

function nameEditor($elem, options, data, ispopupEditing) {
    if (ispopupEditing)
        return "<input data-bind='value:Activity_M().ActivityName_M' />";
    else
        return RadiantQ.Default.Template.ProjectGanttExpandableTextboxEditor();
}

```

Step 2: Specify the Editor in Column definition

Specify the above editor function in the Column definition.

```
Columns=[{
    field: "Activity_M().ActivityName_M()",
    title: "Activity Name",
    width: 100,
    editor: nameEditor,
    template: RadiantQ.Default.Template.ProjectGanttExpandableTextBlockTemplate()
},
.....
],
```

Step 3: Specify the property for Popup Editmode in GanttTableOptions

To enable "popup editing" you have to set the Editmode to "popup" . Popup will open when user clicks on a row which is already selected.

```
$gantt_container.GanttControl({
    GanttTableOptions: {
        editmode: "popup"
    }
});
```

ID	Activity Name
1	Task 1
2	Task 2
3	Task 3
4	Child Task 1
5	Child Task 2
6	Grand Child
7	Grand Child
8	Child Task 3
9	Task 4
10	Task 5
11	Child Task 1
12	Child Task 2
13	Task 6
14	Child Task 1
15	Grand Child

This is illustrated in this samples:

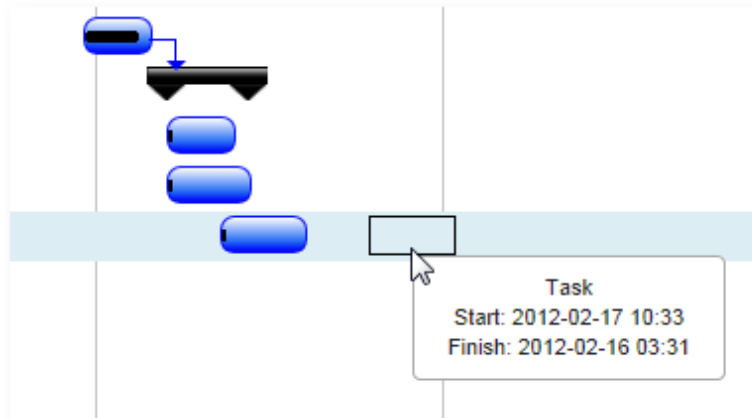
In HTML : ..\Samples\TaskEditingDialog.htm.
 In ASP.NET MVC : ..\Views\Home\Common\TaskEditingDialog.cshtml.
 In ASP.NET : ..\Samples\Common\TaskEditingDialog.aspx.

RadiantQ jQuery Gantt Package

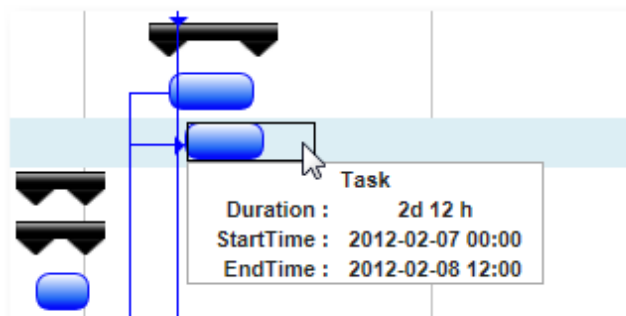
Runtime Interaction

Editing Tasks

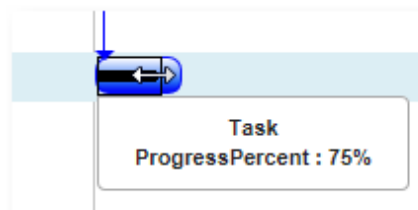
A task's StartTime can be changed by simply dragging them to the left or right in the GanttChart.



The task's EndTime can be changed by resizing it.



The task's ProgressPercent can also be visually changed by dragging the progress bar.



The above can also be edited directly on the activity instance and the changes will be reflected in the chart.

Editing Dependency Lines

You can connect 2 tasks with a dependency line visually in the GanttChart.



Existing dependency lines can also be deleted or edited through the dependencies list in the gantt model and the changes will automatically reflect in the chart.

Adding a dependency line visually as above results in a "FinishToStart" type dependency, but if you want to specify a different type of dependency, you will have to provide a UI where the dependency type can be edited and the changed applied on the dependency instance in the model directly.

Example Predecessors field values:

9 - Task with ID 9.

9SS - Task with ID 9 and with StartToStart dependency.

9SS + 4 - Task with ID 9, with StartToStart dependency and with a 4 hour lag.

9SS, 8FF - Task with ID 9 with StartToStart dependency and Task with ID 8 and with FinishToFinish dependency.

Indent and Outdent

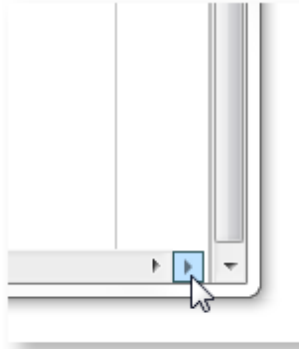
The parent-child relationship between adjacent tasks can be affected by indenting and outdenting them. There is no built-in interactivity to let the end-user indent or outdent tasks (this is mainly because of the lack of context menu support in the framework). However, you can provide menus or buttons to indent/outdent selected tasks with a call to GanttControl.Indent and GanttControl.Outdent methods (as illustrated in several of our samples).

Multiple Indent and Outdent

User can indent/outdent multiple rows by selecting them using ctrl+click or shift+click.This feature will not support for undo/redo scenario.

Time Span

To scroll past the visible time span, the end user has to scroll all the way to the right and then use the pane scroll button to scroll past the visible span.



Similarly scroll all the way to the left and click on the pane scroll button to scroll beyond the left most visible time.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Activities or Task

Activity or Task

The basic unit of work represented in the Gantt control is referred to as an Activity or a Task. The control's API usually refer to them as Activity while the samples and documentation refer to them as Tasks. The control assumes no distinction between the two.

There are typically a hierarchy of parent (summary) and child activities in a project and in fact, the GanttControl's model consists of an Activities hierarchy in hierarchical data structures. But, in real world, the activities list is easily represented by a flat list and so the GanttControl primarily binds to a flat list of activities.

Indent Level

In a flat list of activities, each activity is assigned an Indent Level which define the parent/child relationship.

For example, consider an activity hierarchy like this:

```
Summary Task 1
  Child Task 1
  Child Task 2
```

In a flat list, these will be represented by objects with values like these:

```
{TaskName="Summary Task 1", IndentLevel=0}
{TaskName="Child Task 1", IndentLevel=1}
{TaskName="Child Task 1", IndentLevel=1}
```

See how the IndentLevel setting dictates the parent/child relationship.

StartTime, EndTime and Effort

Every activity in the Gantt control will have a StartTime and Effort associated with it.

Duration Calculation Example

For a given StartTime and Effort, the duration will be based on the current schedule ([WorkTimeSchedule](#)) of the project.

For example:

- If StartTime is 12/25 8AM (Fri) and the Effort is (24 hours)
- And if the WorkTimeSchedule is (8 hours per day starting at 8AM, Mon - Fri).

Then task will require 3 (24/8) working days and taking into account non-working days (Sat and Sun), the task's EndTime would be at 12/29 (Tue) 4PM.

If the WorkTimeSchedule is set to null (24 hours X 7 days a week), then EndTime would be simply StartTime + Effort.

The duration of the task will also be adjusted based on the [resource assignments](#).

If the activity is a parent activity then it's StartTime and EndTime are representation of and based on its child activity times.

Dependencies, if any, setup between activities can also influence the StartTime of an activity. More on this in the [Dependencies](#) section.

The user can change the StartTime of an activity by simply moving the task left or right in the chart or by editing the value in the table. Similarly, a task's Effort can be edited by the end-user by resizing the width of the activity in the chart or by editing it's value in the table.

ProgressPercent

Optionally, the progress made so far in accomplishing a task can be represented in the control. A field in the bound task object can provide this value which should be between 0-100.

SortOrder

If you are going to allow moving tasks up/down or inserting a task in between other tasks (which is almost always the case), you have to include a SortOrder field in your data. Once you do so, follow this pattern:

- a) When you retrieve the data from the table, include a "Sort By" in your SQL query to sort by this field, ascending. Or sort the retrieved list by this field before binding it to the gantt.
- b) Set the SortOrderBinding property in the gantt to this field.

Note: If you are going to add a SortOrder field to an existing table, make sure to give appropriate values from 1 to N representing their order, instead of simply specifying 0 for all the tasks.

Assigned Resources

One or more resources can be assigned to an activity with the ability to assign the resource full-time (100%) or more, or less, defined by the assignments AllocationUnit. The duration of the task is automatically updated based on the number of resources assigned to it.

The GanttControl can also do resource-levelling to avoid resource overload.

PreferredStartTime

This optional setting allows you to associate a preferred StartTime for an activity which allows the Gantt to try to anchor the activity to this time whenever possible. Persisting this setting is essential to automatically move hierarchies of activities back and forth.

RadiantQ jQuery Gantt Package

Dependencies

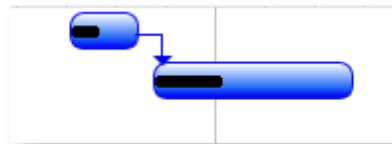
Activity Dependencies

Activities within a project could have dependencies on other activities. The dependent activity's start time is then constrained by the predecessor activity's times. The GanttControl supports these 4 different types of dependency constraints:

Dependency Type	Constraint behavior
FinishToStart (Default)	The ToActivity's start time cannot be earlier than the FromActivity's end time.
StartToStart	The ToActivity's start time cannot be earlier than the FromActivity's start time.
FinishToFinish	The ToActivity's end time cannot be earlier than the FromActivity's end time.
StartToFinish	The ToActivity's end time cannot be earlier than the FromActivity's start time.

There can also be a "lag" specified between the 2 times above.

A FinishToStart dependency between these 2 activities with a lag of 4 hours



is represented as follows:

```
{TaskName="Task 1", TaskID="5" ....}
```

```
{TaskName="Task 2", TaskID="6", PredecessorIndices="5+4"}
```

The PredecessorIndices property indicates a dependency with activity ID "5" and a lag of "4 hours". "5-4" would correspond to a negative lag of "-4 hours". If you want this to be something else then set the LagStringUnitsInHours property appropriately (24, if you want it to indicate the number of days).

The dependency type is indicated with a corresponding suffix ("FS", "SF", "SS" or "FF") as follows:

```
{TaskName="Task 2", TaskID="6", PredecessorIndices="5SF"}
```

The above example indicates a StartToFinish dependency. Specifying the "FS" (Finish To Start) suffix is optional since that's the default.

Multiple predecessors

If an activity is constrained by more than 1 predecessor activity, then the dependency is represented as follows:

PredecessorIndices="4, 2, 9+3"

Simply separate the different predecessor activity IDs by commas.

Editing the Dependencies

End-users can simply drag and drop a connection line from one activity to another in the gantt chart to setup a dependency connection between the two. This results in the default FinishToStart type of dependency.

Validating the predecessor-dependency setting

During runtime the gantt control validates all predecessor settings before applying it. For example, a parent cannot be a predecessor to it's child. The gantt prevents applying such dependency settings.

Preventing Dependency Constraints Conditionally

You can choose to prevent getting the dependency constraints applied on some activities, conditionally as follows, by listening to the ShouldEnforceDependencyConstraintsOnActivity event.

```
var $gantControl = $gant_container.data("GanttControl");
$gantControl.ShouldEnforceDependencyConstraintsOnActivity.subscribe(Gantt_ShouldEnforceDependencyConstraints);

// To prevent dependency constraints from shifting an activity that has already progressed.
function Gantt_ShouldEnforceDependencyConstraints(sender, args) {
    if (args.Activity_M().ProgressPercent_M() > 0)
        args.Enforce = false;
    else
        args.Enforce = true;
}
```

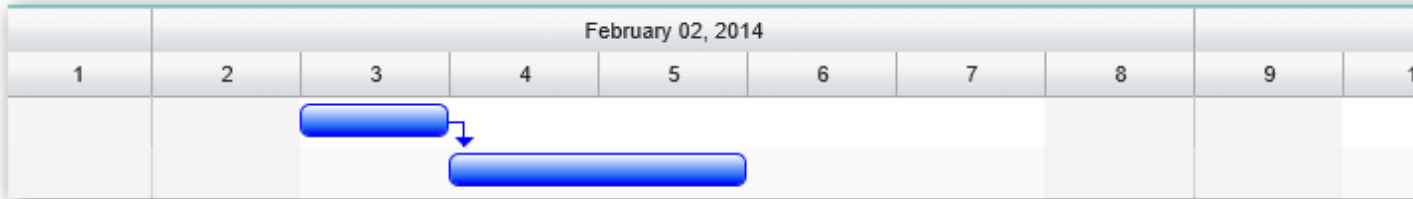
© RadiantQ 2009-2018. All Rights Reserved.

-0-

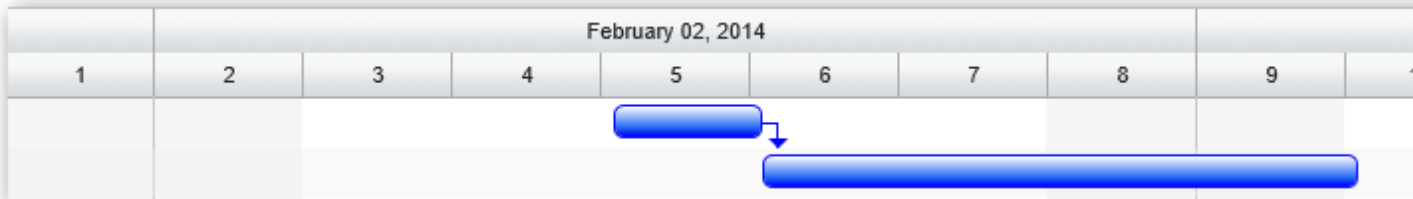
RadiantQ jQuery Gantt Package

Keeping Dependant tasks "sticky"

When there are 2 connected tasks (Task A and Task B) like below:



... Task B would move to the right as Task A is moved, like this:



However, you might also want it to move backwards as Task A is moved back.

While this is supported, Task B can move back only until it's "preferred start time".

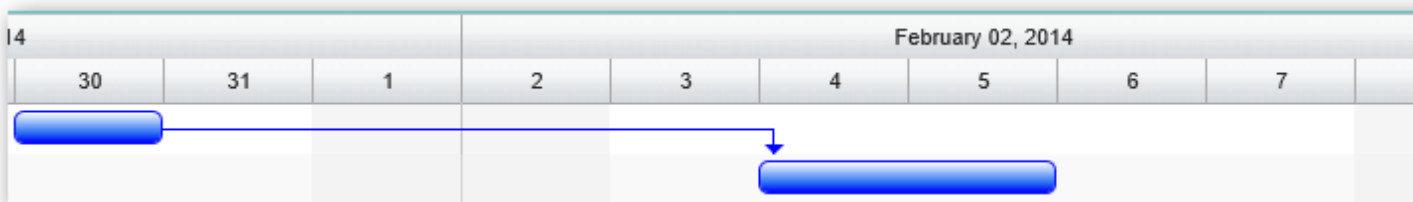
So, what exactly is a task's preferred start time? This varies based on your settings.

Scenario 1:

You have not setup PreferredStartTimeBinding in the gantt. In other words, the underlying task object does not contain a property with the "preferred start time" value.

In this case, when the task is loaded in the gantt, it's start time becomes it's preferred start time.

So, Task B cannot go backwards as Task A goes back:



Scenario 2:

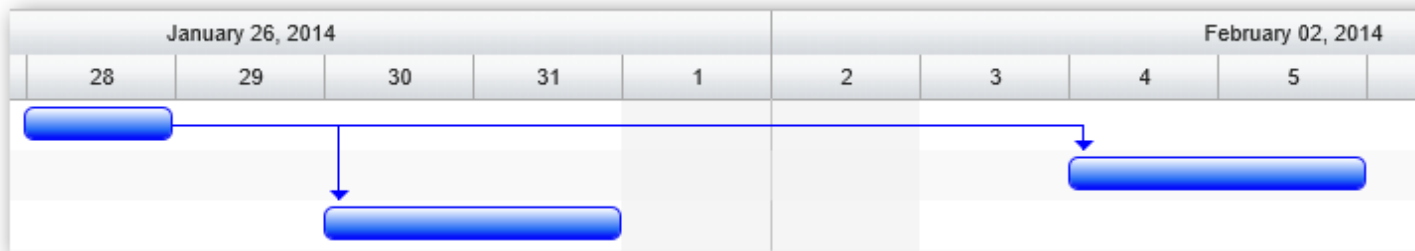
You have setup PreferredStartTimeBinding and the task has a property called PreferredStartTime.

```
$gantt_container = $('#gantt_container');
$gantt_container.GanttControl({
    PreferredStartTimeBinding: new RadiantQ.BindingOptions(
"PreferredStartTime"),
});
```

Then when you create a new task, you give it a proper PreferredStartTime like this:

```
var gantt = $gantt_container.data("GanttControl");
gantt.AddNewItem({
  "Name" : "Task 3",
  "ID" : 3,
  "PredecessorIndices" : "1",
  "StartTime" : "2014-02-04T00:00:00Z",
  "PreferredStartTime":"2014-01-30T00:00:00Z",
  "Effort" : "16:00:00",
  "Resources" : "1",
  "Description" : "Description of Task 3"
});
```

Then you will see that the task will move backwards until the specified PreferredStartTime is hit.



Note that the gantt will update this PreferredStartTime value for a task whenever the user moves a task manually (as opposed to the task getting moved automatically because of dependency).

RadiantQ jQuery Gantt Package

Calendars

Projects usually have to be associated with Calendars representing these:

- Week days
- Week day working hours (with or without breaks)
- Exception Days
- Holidays
- Certain days with varying working times.

Calendars can then be associated with an entire Project (for all the tasks in the project) or to specific Resources (then the times for tasks assigned to these resources will be calculated based on the Resource's calendar).

Calendars can be programatically defined using the [WorkTimeSchedule](#) type. There are also commonly used schedules, built-in, that you can simply reuse.

Calendars can also be declared in a specific [string format](#) which can then be converted to a WorkTimeSchedule instance using tools provided.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

*RadiantQ jQuery Gantt Package***WorkTimeSchedule**

Working Times

It's important to provide the GanttControl the correct "working times" for the tasks in the project so that the time span of the tasks can be appropriately determined based on the required duration of the task.

WorkTimeSchedule

This type lets you define a set of working days and times. You can create a custom schedule and assign it to the `GanttControl.WorkTimeSchedule` property.

For example,

Using the built-in schedules

In HTML

```
$('#container').GanttControl({
    ....
    ....
    WorkTimeSchedule: RadiantQ.Gantt.WorkTimeSchedule.Schedule8X5
});
```

In ASP.NET MVC

```
<%= Html.JQProjectGantt(
    new JQProjectGanttSettings()
    {
        ControlId = "gantt_container",
        Options = new ProjectGanttOptions()
        {
            WorkTimeSchedule = "RadiantQ.Gantt.WorkTimeSchedule.Schedule8X5",
            ...
        }
    }
)%>
```

In ASP.NET

```
<RQ:GanttControl WorkTimeSchedule="RadiantQ.Gantt.WorkTimeSchedule.Schedule8X5" ID="gantt" ... />
```

Using custom schedules

In HTML

```
$('#container').GanttControl({
    ....
    ....
    WorkTimeSchedule: new RadiantQ.Gantt.WorkTimeSchedule("Monday -
Thursday, 10 Hours", Create4Days10HoursSchedule)
});
```

In ASP.NET MVC

```

<script type="text/javascript">

    function Create8X5ScheduleWithHolidays(date) {
        // Holidays for 2012
        if (date == null)
            return;
        if (date.equals(new Date(2012, 1, 13)) ||
            date.equals(new Date(2012, 1, 14)) ||
            date.equals(new Date(2012, 1, 17)) ||
            date.equals(new Date(2012, 5, 4)) ||
            date.equals(new Date(2012, 6, 13)))
            return null;

        // Simply reuse an existing schedule or use your own logic here.
        return RadiantQ.Gantt.WorkTimeSchedule.EightHoursByFiv
eDaysScheduleProvider(date);
    }
    Object.defineProperty(this, "FourDaysTenHoursSchedule", {
        get: function () {
            return new RadiantQ.Gantt.WorkTimeSchedule("Monday - Thursday, 10
Hours", Create8X5ScheduleWithHolidays); ;
        },
        enumerable: true,
        configurable: false
    });
</script>
...
<%= Html.JQProjectGantt(
    new JQProjectGanttSettings()
    {
        ControlId = "ganttt_container",
        Options = new ProjectGanttOptions()
        {
            WorkTimeSchedule = "FourDaysTenHoursSchedule",
            ...
        }
    }
)%>

```

In ASP.NET

```

<script type="text/javascript">

    function Create8X5ScheduleWithHolidays(date) {
        // Holidays for 2012
        if (date == null)
            return;
        if (date.equals(new Date(2012, 1, 13)) ||
            date.equals(new Date(2012, 1, 14)) ||
            date.equals(new Date(2012, 1, 17)) ||
            date.equals(new Date(2012, 5, 4)) ||
            date.equals(new Date(2012, 6, 13)))
            return null;

        // Simply reuse an existing schedule or use your own logic here.
        return
        RadiantQ.Gantt.WorkTimeSchedule.EightHoursByFiveDaysScheduleProvider(date);
    }
    Object.defineProperty(this, "FourDaysTenHoursSchedule", {
        get: function () {

            return new RadiantQ.Gantt.WorkTimeSchedule("Monday - Thursday, 10 Hours",
                Create8X5ScheduleWithHolidays); ;
        },
        enumerable: true,
        configurable: false
    });
</script>
...
<RQ:GanttControl ID="gantt" WorkTimeSchedule = "FourDaysTenHoursSchedule" .... />

```

If you are going to set this property, make sure to do so before assigning the ItemsSource on the GanttControl. If you have to dynamically change this setting during runtime, then everytime you do so, you will have to reset the ItemsSource property to null and then back to your source for the new schedule to take effect. (This is not handled internally for performance reasons).

Built-in Schedules

- The default schedule used is a "Monday to Friday, 8AM to 4PM" schedule defined by the RadiantQ.Gantt.WorkTimeSchedule.Schedule8X5 instance.
- There is also a "Monday to Friday, 24 hours a day" schedule defined by the RadiantQ.Gantt.WorkTimeSchedule.Schedule24X5 instance.
- If you want a continuous "24 X 7" schedule simply set GanttControl.WorkTimeSchedule property to null.

Custom Schedules

When you look at the examples below, you will see that creating custom schedules are a snap.

- For example, if you want to create a "Monday to Thursday, 10 hours a day" schedule, you can do so as follows:

```

function Create4Days10HoursSchedule(date){
    switch (date.getDayName()) {
        case "Monday":
        case "Tuesday":
        case "Wednesday":
        case "Thursday":
            var intervals = new RadiantQ.Gantt.TimePeriodCollection();
            intervals.Add(new
RadiantQ.Gantt.TimePeriod(date.clone().addHours(8.0), null, new RQTimeSpan(0, 10, 0,
0, 0)));
                return intervals;
                break;
            default:
                return null;
        }
    }
}

```

b) For a "Monday to Thursday, 24 hours a day" schedule, you can do as follows:

```

function Create4Days10HoursSchedule(date){
    switch (date.getDayName()) {
        case "Monday":
        case "Tuesday":
        case "Wednesday":
        case "Thursday":
            var intervals = new RadiantQ.Gantt.TimePeriodCollection();
            intervals.Add(new
RadiantQ.Gantt.TimePeriod(date.clone().addHours(8.0), null, new RQTimeSpan(1, 0, 0,
0, 0)));
                return intervals;
                break;
            default:
                return null;
        }
    }
}

```

c) To include Holidays in your schedule use a custom schedule like this:

```

function Create8X5ScheduleWithHolidays(date){
    // Holidays for 2012
    if (date.equals(new Date(2012, 1, 13)) ||
        date.equals(new Date(2012, 1, 14)) ||
        date.equals(new Date(2012, 1, 17)) ||
        date.equals(new Date(2012, 5, 4)) ||
        date.equals(new Date(2012, 6, 13)))
        return null;

    // Simply reuse an existing schedule or use your own logic here.
    return
RadiantQ.Gantt.WorkTimeSchedule.EightHoursByFiveDaysScheduleProvider(date);
}

```

Take a look at the sample GanttControlCustomSchedule for an illustration of this feature.

-0-

RadiantQ jQuery Gantt Package

Calendar and CalendarWithExceptions

The Calendar and CalendarWithExceptions are used to create a [WorkTimeSchedule](#) from a string which defines the working/non-working days and working/non-working time.

Here is some example Calendar definition strings.

Calendar Name	Calendar definition string
8X5 Calendar	MON 08:00:00 16:00:00;TUE 08:00:00 16:00:00;WED 08:00:00 16:00:00;THU 08:00:00 16:00:00;FRI 08:00:00 16:00:00;SAT 08:00:00 16:00:00
24X7 Calendar	Mon 00:00:00 23:00:00;Tue 00:00:00 23:00:00;Wed 00:00:00 23:00:00;thu 00:00:00 23:00:00;fri 00:00:00 23:00:00;sat 00:00:00 23:00:00;sun 00:00:00 23:00:00;
24X7 With Holidays	Mon 00:00:00 23:00:00;Tue 00:00:00 23:00:00;Wed 00:00:00 23:00:00;thu 00:00:00 23:00:00;fri 00:00:00 23:00:00;sat 00:00:00 23:00:00;sun 00:00:00 23:00:00; HOL 2013-9-16;TeamOut:2013-9-18;

The Calendar definition string holds the following information

1. Working Days of the week - The first part of the string is where you would specify a "working day" and the working times in that day. For example, "**MON 08:30:00 12:30:00, 13:30:00 17:30:00;**" which means Monday is a working day with working times from 8.30AM to 12.30PM and 1.30PM to 5.30PM. Any day not represented is considered a day off. Separate different day definitions with a semicolon (;) and multiple time breaks must be separated by commas (,).
2. Holidays - The second part of the string lets you specify holidays by date. Start this part with a HOL keyword prefix, followed by the different dates separated by semi-colons. Specific holidays can optionally include a name. For an example see above list.

Code Example

```
var baseCalendar = new RadiantQ.Gantt.Calendar(/*Calendar Name */ "8X5 Calendar",
/*Calendar Definition string*/ "MON 08:00:00 16:00:00;TUE 08:00:00 16:00:00;WED
08:00:00 16:00:00;THU 08:00:00 16:00:00;FRI 08:00:00 16:00:00;SAT 08:00:00 16:00:00"
);

//To create a work time schedule from Calendar from the above string.
var workTimeSchedule = RadiantQ.Gantt.Calendar.CreateSchedule(baseCalendar);

$('#container').GanttControl({
....
....
    WorkTimeSchedule: workTimeSchedule
});
```

CalendarWithExceptions

Used to define some exceptions (working days, non-working days, week days and week day times) that overrides a base calendar defined above.

Here is an example CalendarWithExceptions definition string.

```

WorkDaySpecific OT:2014-1-13 TimePeriod 08:00:00 20:00:00;OT:2015-5-23;
NonWorkDaySpecific Meeting:2014-1-27;Meeting:2014-2-1;Meeting:2014-3-10;
WeekDaySpecific Sun 00:00:00 1.00:00:00;

```

The CalendarWithExceptions definition string holds the following information

1. Workday Specific- A string that defines a specific day (non-working day in the base calendar) as a working day and it's working time. Separate multiple entries with a semicolon (;). Each entry could include an optional name like "OT" above and can also include optional times (if not specified, the times from the weekday will be used.).
2. NonWorkDay Specific - A string that defines a specific day (working day in the base calendar) as a non-working day.
3. WeekDay Specific - A string that defines the working times of specific weekdays, overriding what was specified in the base calendar.

Code Example

```

var baseCalendar = new RadiantQ.Gantt.Calendar(/*Calendar Name */ "8X5 Calendar",
    /*Calendar Definition string*/ "MON 08:00:00 16:00:00;TUE 08:00:00
16:00:00;WED 08:00:00 16:00:00;THU 08:00:00 16:00:00;FRI 08:00:00 16:00:00;SAT
08:00:00 16:00:00");

var expCalendar = new RadiantQ.Gantt.CalendarWithExceptions(/* base Calendar */
baseCalendar,
    /*Calendar with Exceptions Definition string*/
    "WorkDaySpecific OT:2014-1-13 TimePeriod 08:00:00 20:00:00;OT:2015-5-23;
NonWorkDaySpecific Meeting:2014-1-27;Meeting:2014-2-1;Meeting:2014-3-10;
WeekDaySpecific Sun 00:00:00 1.00:00:00;");

//To create a work time schedule from the above CalendarWithExceptions.
var workTimeSchedule =
RadiantQ.Gantt.CalendarWithExceptions.CreateWorkTimeSchedule(expCalendar);

$('#container').GanttControl({
....
....
    WorkTimeSchedule: workTimeSchedule
});

```

RadiantQ jQuery Gantt Package

Time Scale Header

Please refer to these topics for more info:

- [Time Scale Header customization](#)
- [End-User Time Scale Header operations](#)

© RadiantQ 2009-2018. All Rights Reserved.

-0-

Resource Assignment

RadiantQ jQuery Gantt Package

Binding to Resource Strings

Resources as Simple Strings

The most simplest way to provide a set of resources that are available for assignment in a project is with an array of resource names as follows:

```
var resources = new Array("Resource1", "Resource2")
```

Your task object should then expose a string property that can hold the resource assignment information as a comma separated list of resource names, as follows:

```
// Bound task instance.
{
  Name: "Task 6",
  ID: 13,
  StartTime: new Date().addDays(2).Date(),
  Effort: new RQTimeSpan(0, 16, 0, 0, 0),
  Resources: "Resource 1, Resource 2"
},
```

To indicate to the GanttControl that the resource assignment information is in the property "Resources", setup this binding:

In HTML

```
$('#container').GanttControl({
  ....
  ....
  ResourceItemsSource: resources,
  AssignedResourcesBinding: new RadiantQ.BindingOptions("Resources")
});
```

The gantt will then read and write resource assignment information into this property of the bound object.

Partial Resource Assignments

A partial resource assignment is easily specified in the assignment string as follows:

```

var resources = new Array({
    ResourceID: 1,
    ResourceName: "Resource 1"
},
{
    ResourceID: 3,
    ResourceName: "Resource 3[50%]"
},
{
    ResourceID: 2,
    ResourceName: "Resource 2",
});
$('#container').GanttControl({
    ....
    ....
    ResourceItemsSource: resources
});

```

This is illustrated in this samples:

In HTML : ..\Samples\TaskEditingDialog.htm.

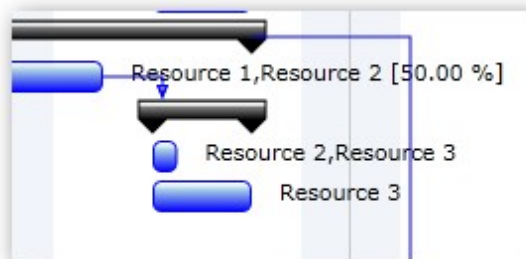
In ASP.NET MVC : ..\Views\Home\ProjectGantt\TaskEditingDialog.cshtml.

In ASP.NET : ..\Samples\ProjectGantt\TaskEditingDialog.aspx.

See [Assignments VS Task Duration](#) for more info on how this affects the task's duration.

Hiding Resource text to the right of the bar

Normally, the assigned resources are shown to the right of the task bar in the gantt chart.



Resource names in gantt chart.

But, you can hide this if necessary as follows:

```

$('#container').GanttControl({
    ....
    ....
    ShowAssignedResourcesText : false,
});

```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Binding to Resource Objects

Resources objects

Often in your business layer, resources are probably represented as a list of instances of a complex type with various properties. Assign this list to the gantt as follows:

In HTML

```

    $('#container').GanttControl({
        ....
        ....
        ResourceItemsSource: new Array({ ResourceID: 1, ResourceName:
"Resource 1" },
                                   { ResourceID: 2, ResourceName:
"Resource 2" }),
        ResourceIDBinding: new RadiantQ.BindingOptions("ResourceID"),
        ResourceNameBinding: new RadiantQ.BindingOptions("ResourceName")
    });

```

In ASP.NET MVC

```

<%= Html.JQProjectGantt(
    new JQProjectGanttSettings()
    {
        ControlId = "ganttt_container",
        Options = new ProjectGanttOptions()
        {
            ResourceItemsSource = new List<ResourceInfo>(){ new ResourceInfo(){
ResourceID=1,ResourceName="Resource 1"}, new ResourceInfo(){
ResourceID=2,ResourceName="Resource 2" }},
        }
    })%>

```

In ASP.NET

```

<script runat="server">
    protected void gantt_Load(object sender, EventArgs e)
    {
        this.gantt.ResourceItemsSource = new List<ASPNetGanttDemo.DataSources.
ResourceInfo>()
        {
            new ASPNetGanttDemo.DataSources.ResourceInfo(){ ResourceID = 1,
ResourceName = "Resource 1" },
            new ASPNetGanttDemo.DataSources.ResourceInfo(){ ResourceID = 2,
ResourceName = "Resource 2", CustomScheduleString = "Mon 8:00:00 16:00:00;Tue 8:00:00
16:00:00;Wed 8:00:00 16:00:00" },
            new ASPNetGanttDemo.DataSources.ResourceInfo(){ ResourceID = 3,
ResourceName = "Resource 3" }
        };
        this.gantt.ResourceIDBinding = new RadiantQ.Web.JQGantt.Common.Binding(
"ResourceID");
        this.gantt.ResourceNameBinding = new RadiantQ.Web.JQGantt.Common.Binding(
"ResourceName");
    }
</script>

<RQ:GanttControl ID="gantt" OnLoad="gantt_Load" />

```

For ASP.NET and ASP.NET MVC, have to create a `ResourceInfo` type in you project.

```

public class ResourceInfo
{
    /// <summary>
    /// The name of the resource. Typically referenced in the
GanttControl.ResourceNameBinding property.
    /// </summary>
    public string ResourceName { get; set; }

    /// <summary>
    /// The unique id of the resource. Typically referenced in the
GanttControl.ResourceIDBinding property.
    /// </summary>
    public int ResourceID { get; set; }
    /// <summary>
    /// A string that represents the custom schedule of this resource. Typically
referenced in the GanttControl.ResourceScheduleBinding property.
    /// This is usually of the format "MON 08:30:00 12:30:00, 13:30:00 17:30:00; TUE
9:30:00 5:30:00; WED.....". Exclude days when resource will not be working.
    /// A 24 hour day is represented just as "MON" for example.
    /// </summary>
    public string CustomScheduleString { get; set; }
    /// <summary>
    /// A string that represents the custom schedule of this resource. Typically
referenced in the GanttControl.ResourceScheduleBinding property.
    /// </summary>
    public string CustomSchedule { get; set; }
}

```

You should then specify how a resource be identified in the model as well as how it should be identified in the UI. You can do this as above.

With this setup the resources will be represented in the model as a comma separated list of resource ids, as follows:

In HTML

```
// Bound task instance where the Resources property specifies the
resource assignments.
{
    Name: "Task 6",
    ID: 13,
    StartTime: new Date().addDays(2).Date(),
    Effort: new RQTimeSpan(0, 16, 0, 0, 0),
    Resources: "1, 2"
},
```

In ASP.NET MVC And ASP.NET

```
List<TaskInfo> taskItems = new List<TaskInfo>
{
    new TaskInfo { Name = "Grand Child Task 1", ID = 6, IndentLevel = 2,
Resources = "2,3", StartTime = dt, Effort = TimeSpan.Parse("08:00:00").ToString() },
    ...
};
```

To indicate to the GanttControl that the resource assignment information is in the property "Resources", setup this binding:

In HTML

```
$('#container').GanttControl({
    ....
    ....
    AssignedResourcesBinding: new RadiantQ.BindingOptions("Resources")
});
```

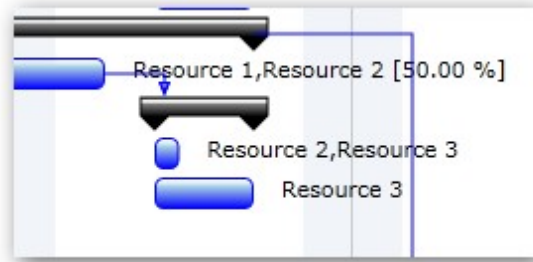
In ASP.NET MVC

```
<%= Html.JQProjectGantt(
    new JQProjectGanttSettings()
    {
        ControlId = "ganttt_container",
        Options = new ProjectGanttOptions()
        {
            //You have to create a resource type in your controller
            AssignedResourcesBinding= Binding("Resources"),
        }
    }
)%>
```

In ASP.NET

```
<RQ:GanttControl ID="ganttt" DataSourceUrl="../../DataSources/TaskListHandler.ashx"
Height="500px" runat="server" >
    <AssignedResourcesBinding Property="Resources" />
</RQ:GanttControl>
```

Due to the above ResourceNameBinding setup, the UI of the gantt will use the names of the resources rather than the id of the resources in the gantt chart as follows:



Resource names in gantt chart.

This is illustrated in this samples:

- In HTML : ..\Samples\GanttControlSkeletonWithResources.htm.
- In ASP.NET MVC : ..\Views\Home\ProjectGantt\GanttControlSkeletonWithResources.cshhtml.
- In ASP.NET : ..\Samples\ProjectGantt\GanttControlSkeletonWithResources.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

RadiantQ jQuery Gantt Package

Resource Specific Calendars

Resources with custom schedules

Your resource objects could contain information about their custom schedules that can be bound to the gantt. This information should eventually be converted to a `WorkTimeSchedule` instance.

a)

There are some built-in utilities that can convert a string representation of this custom schedule (in a predefined format) into a `WorkTimeSchedule` instance.

So, for example, if your resources are defined as follows:

In HTML

```
$('#container').GanttControl({
    ....
    ....
    ResourceItemsSource: new Array({ ResourceID: 1, ResourceName:
"Resource 1" },
                                { ResourceID: 2, ResourceName:
"Resource 2",
                                CustomScheduleString: "Mon 8:00:00
16:00:00;Tue 8:00:00 16:00:00;Wed 8:00:00 16:00:00" }),
    ResourceIDBinding: new RadiantQ.BindingOptions("ResourceID"),
    ResourceNameBinding: new RadiantQ.BindingOptions("ResourceName"),

    // Then you can setup this binding in the GanttControl:
    ResourceScheduleBinding: { Property: "CustomScheduleString",
Converter: RadiantQ.Gantt.StringToWorkTimeScheduleConverter }
});
```

In ASP.NET MVC

```
<%= Html.JQProjectGantt(
    new JQProjectGanttSettings()
    {
        ControlId = "ganttt_container",
        Options = new ProjectGanttOptions()
        {

            ResourceItemsSource = new List<ResourceInfo>() { new ResourceInfo() {
ResourceID = 1, ResourceName = "Resource 1" },
                new ResourceInfo() { ResourceID = 2, ResourceName = "Resource 2",
CustomScheduleString = "Mon 8:00:00 16:00:00;Tue      8:00:00 16:00:00;Wed
8:00:00 16:00:00" } },
            ResourceIDBinding= new Binding("ResourceID"),
            ResourceNameBinding= new Binding("ResourceName"),
            // Then you can setup this binding in the GanttControl:
            ResourceScheduleBinding = new Binding("CustomScheduleString",
"RadiantQ.Gantt.StringToWorkTimeScheduleConverter")
        }
    }
)%>
```

The gantt will then use this custom schedule for the resources.

In ASP.NET

```

<script runat="server">
    protected void gantt_Load(object sender, EventArgs e)
    {
        this.gantt.ResourceItemsSource = new List<ASPNetGanttDemo.DataSources.
ResourceInfo>()
        {
            new ASPNetGanttDemo.DataSources.ResourceInfo(){ ResourceID = 1,
ResourceName = "Resource 1" },
            new ASPNetGanttDemo.DataSources.ResourceInfo(){ ResourceID = 2,
ResourceName = "Resource 2", CustomScheduleString = "Mon 8:00:00 16:00:00;Tue 8:00:00
16:00:00;Wed 8:00:00 16:00:00" },
            new ASPNetGanttDemo.DataSources.ResourceInfo(){ ResourceID = 3,
ResourceName = "Resource 3" }
        };
    }
</script>

<RQ:GanttControl ID="gantt" DataSourceUrl="../../DataSources/TaskListHandler.ashx"
runat="server" >
    <ResourceIDBinding Property="ResourceID" />
    <ResourceNameBinding Property="ResourceName" />
    <ResourceScheduleBinding Property="CustomScheduleString" Converter
="RadiantQ.Gantt.StringToWorkTimeScheduleConverter" />
</RQ:GanttControl>

```

The gantt will then use this custom schedule for the resources.

b)

Alternatively if you have a property that exposes a WorkTimeSchedule instance, that can be directly be bound to the gantt as follows:

```

{ ResourceID: 2, ResourceName: "Resource 2", CustomSchedule: new
RadiantQ.Gantt.WorkTimeSchedule("Custom Schedule", CustomScheduleForRes2) }

```

In HTML

```

function CustomScheduleForRes2(date) {
    // You can choose to take into account holidays here.
    // For better performance cache the intervals for dates instead of
creating a new collection every time.
    if ((date.getDayName() != "Saturday") && (date.getDayName() != "Sunday")
&& (date.getDayName() != "Friday")) {
        var intervals = new RadiantQ.Gantt.TimePeriodCollection();
        intervals.Add(new
RadiantQ.Gantt.TimePeriod(date.clone().addHours(8.0), null, new RQTimeSpan(0, 8, 0,
0, 0)));
        return intervals;
    }
    else {
        return null;
    }
}

```

In HTML


```

    $('#container').GanttControl({
        ....
        ....
        ResourceScheduleBinding: new RadiantQ.BindingOptions("CustomSchedule"
    )
    });

```

In ASP.NET MVC

```

<script type="text/javascript">

    function CustomScheduleForRes2(date) {
        // You can choose to take into account holidays here.
        // For better performance cache the intervals for dates instead of creating a
        new collection every time.
        if ((date.getDayName() != "Saturday") && (date.getDayName() != "Sunday") &&
(date.getDayName() != "Friday")) {
            var intervals = new RadiantQ.Gantt.TimePeriodCollection();
            intervals.Add(new RadiantQ.Gantt.TimePeriod(date.clone().addHours(8.0),
null, new RQTimeSpan(0, 8, 0, 0, 0)));
            return intervals;
        }
        else {
            return null;
        }
    }

    Object.defineProperty(window, "CustomSchedule",
    {
        get: function () {
            return new RadiantQ.Gantt.WorkTimeSchedule("Custom Schedule",
CustomScheduleForRes2)
        },
        set: function (value) {
        },
        enumerable: true,
        configurable: true
    });

</script>

```

In ASP.NET MVC

```

<%= Html.JQProjectGantt(
    new JQProjectGanttSettings()
    {
        ControlId = "ganttt_container",
        AfterGanttWidgetInitializedCallback = "AfterGanttWidgetInitializedCallback",
        DataSourceUrl = new Uri("/Home/GetProjectGanttItemsource", UriKind
.RelativeOrAbsolute),
        Options = new ProjectGanttOptions()
        {
            ResourceItemsSource = new List<FlexyGanttMVCSample.Controllers.
ResourceInfo>() { new FlexyGanttMVCSample.Controllers.ResourceInfo() { ResourceID =
1, ResourceName = "Resource 1", CustomSchedule = "CustomSchedule" }, new
FlexyGanttMVCSample.Controllers.ResourceInfo() { ResourceID = 3, ResourceName =
"Resource 3" }, new FlexyGanttMVCSample.Controllers.ResourceInfo() { ResourceID = 2,
ResourceName = "Resource 2" } },
            ResourceIDBinding = new Binding("ResourceID"),
            ResourceNameBinding = new Binding("ResourceName"),
            ResourceScheduleBinding = new Binding("CustomSchedule"),
        }
    })%>

```

In ASP.NET

```

<script type="text/javascript">
    function CustomScheduleForRes2(date) {
        // You can choose to take into account holidays here.
        // For better performance cache the intervals for dates instead of creating a
new collection every time.
        if ((date.getDayName() != "Saturday") && (date.getDayName() != "Sunday") &&
(date.getDayName() != "Friday")) {
            var intervals = new RadiantQ.Gantt.TimePeriodCollection();
            intervals.Add(new RadiantQ.Gantt.TimePeriod(date.clone().addHours(8.0),
null, new RQTimeSpan(0, 8, 0, 0, 0)));
            return intervals;
        }
        else {
            return null;
        }
    }

    Object.defineProperty(window, "CustomSchedule",
    {
        get: function () {
            return new RadiantQ.Gantt.WorkTimeSchedule("Custom Schedule",
CustomScheduleForRes2)
        },
        set: function (value) {
        },
        enumerable: true,
        configurable: true
    });
</script>

```

In ASP.NET

```
<script runat="server">
    protected void gantt_Load(object sender, EventArgs e)
    {
        this.gantt.ResourceItemsSource = new List<ASPNetGanttDemo.DataSources.
ResourceInfo>()
        {
            new ASPNetGanttDemo.DataSources.ResourceInfo(){ ResourceID = 1,
ResourceName = "Resource 1" },
            new ASPNetGanttDemo.DataSources.ResourceInfo(){ ResourceID = 2,
ResourceName = "Resource 2", CustomSchedule = "CustomSchedule"},
            new ASPNetGanttDemo.DataSources.ResourceInfo(){ ResourceID = 3,
ResourceName = "Resource 3" }
        };
    }
</script>

<RQ:GanttControl ID="gantt" DataSourceUrl="../../DataSources/TaskListHandler.ashx"
runat="server" >
    <ResourceIDBinding Property="ResourceID" />
    <ResourceNameBinding Property="ResourceName" />
    <ResourceScheduleBinding Property="CustomSchedule" />
</RQ:GanttControl>
```

This is illustrated in this samples:

In HTML : ..\Samples\GanttControlSkeletonWithResources.htm.
In ASP.NET MVC : ..\Views\Home\ProjectGantt\GanttControlSkeletonWithResources.cshtml.
In ASP.NET : ..\Samples\ProjectGantt\GanttControlSkeletonWithResources.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

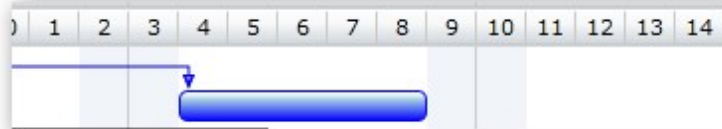
-0-

RadiantQ jQuery Gantt Package

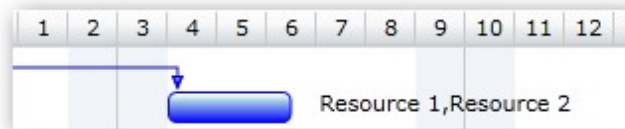
Assignments VS Task Duration

Effort Driven Scheduling

By default, When multiple resources are assigned to a task the task's duration is automatically reduced to reflect the multiple assignments.



Duration before resource assignment



Duration after resource assignment

When a resource has a custom schedule then the following logic is used to calculate the duration of the task. Similarly when a task is assigned a resource partially:

- Divide the task's effort equally between all the assigned resources.
- Compute the required duration for each resource to complete their assigned portion.
- The task's duration will be the largest of these individual resource-durations.

Fixed Duration Scheduling

Alternatively you can turn off this behavior and prevent the duration from changing when multiple resources are assigned by setting this:

In HTML

```
$gantt_container.GanttControl({
  DataSource: jsonData,
  AdjustDurationOnAssignment: false,
});
```

In ASP.NET MVC

```
@Html.JQProjectGantt(
  new JQProjectGanttSettings()
  {
    ControlId = "gantt_container",
    DataSourceUrl = new Uri("/Home/GanttControlItemSource", UriKind
.RelativeOrAbsolute),
    Options = new ProjectGanttOptions()
    {
      AdjustDurationOnAssignment = false
    }
  }
);
```

In ASP.NET

```
<RQ:GanttControl ID="gantt" AdjustDurationOnAssignment="false"
  DataSourceUrl="../../TaskListHandler.ashx" runat="server" >
</RQ:GanttControl>
```

This is illustrated in this samples:

In HTML : ..\Samples\GanttControlSkeletonWithResources.htm.
In ASP.NET MVC : ..\Views\Home\ProjectGantt\GanttControlSkeletonWithResources.cshtml.
In ASP.NET : ..\Samples\ProjectGantt\GanttControlSkeletonWithResources.aspx.

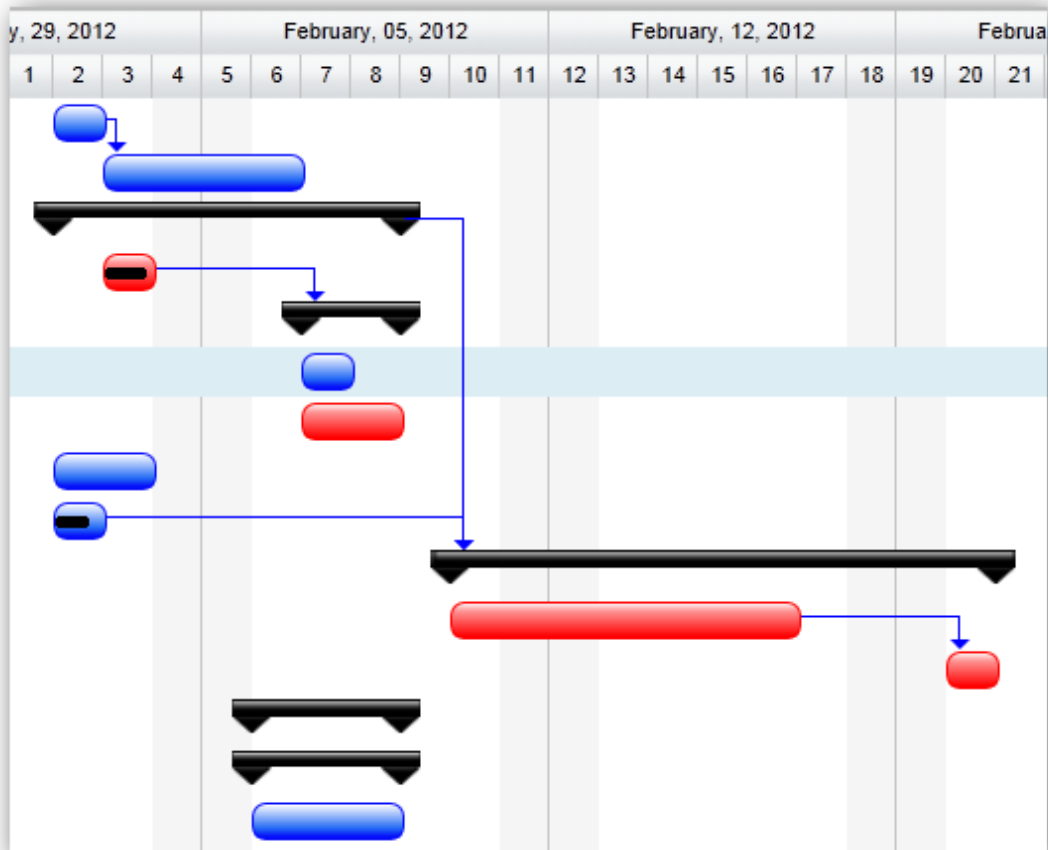
© RadiantQ 2009-2018. All Rights Reserved.

-0-

*RadiantQ jQuery Gantt Package***Critical Paths**

A project's Critical Path is defined by a list of tasks whose delay will directly or indirectly affect the project's finish date. An activity could indirectly affect the project finish date because of the presence of dependencies that would in turn affect other activities move past the current project end time.

It's often required to highlight such critical activities for proper project planning.



Sample Gantt with critical path highlighted

The GanttControl provides built-in functionality to determine and visualize the critical path of the project.

Determining Critical Paths in code

The GanttControl.GetCriticalPathActivities method returns the list of activities that fall under the critical path. For example:

```
GetCriticalPathActivities(/*RQTimeSpan*/ timeBuffer);
```

The 1st argument above is a time-buffer which specifies how further away an activity should be from potentially affecting the project dead line. If an activity is within this time-buffer then it will also be classified as "critical".

Highlighting Critical Paths

To highlight critical path activities, first cache the critical path activities in a static array as follows:

```

    var CriticalPathActivities = new Array();
    // In Button_Click for example:
    // The first argument to GetCriticalPathActivities is a time-buffer which
    // speifies the "safe distance"
    // that an activity's end time should be away from affecting the project
    // deadline.
    CriticalPathActivities =
$gantControl.GetCriticalPathActivities(RQTimeSpan.Zero);
    $gantControl.RedrawChartRows();

```

Then setup a binding that will provide custom color information for activities as follows:

```

    var tTemplate = "<div class='rq-gc-taskbar'
style='background-image:${UpdateBackgroundColorBinding(data)} !important;
border-color:${ UpdateBorderColorBinding(data)} !important'><div
id='GanttTaskBarLabel' class='rq-gc-taskbar-label'></div></div>";

    function UpdateBackgroundColorBinding(data) {
        var isCritical = CriticalPathActivities.indexOf(data) != -1;
        // for background-image
        if (isCritical)
            return 'url(Images/redBar.png)';
        return 'url(Src/Styles/Images/TaskBar.png)';
    }
    function UpdateBorderColorBinding(data) {
        var isCritical = CriticalPathActivities.indexOf(data) != -1;
        // for background-color
        if (isCritical)

            return 'red';
        return '#050DFA';
    }

```

The resultant Gantt screenshot is shown above.

This is illustrated in this samples:

In HTML : ..\Samples\GanttControlCriticalPath.htm.
 In ASP.NET MVC : ..\Views\Home\ProjectGantt\GanttControlCriticalPath.cshtml.
 In ASP.NET : ..\Samples\ProjectGantt\GanttControlCriticalPath.aspx.

RadiantQ jQuery Gantt Package

WBS Support

WBS Support

Work Breakdown Structure (WBS) codes are outline numbers that you can apply to tasks and edit to match your business needs.

You can setup a default outline numbering scheme which will be used by the gantt as the user adds/deletes/moves tasks around. And you can also allow your end-users to edit the WBS codes and provide their own for specific tasks.

NOTE: WBS is illustrated in the sample Samples\WBSEnabledTasks. Use that as a companion resource while you read through this topic.

Turning On WBS for tasks

To begin with, you can turn on this feature by providing a method to the ProvideWBSID that will compute the WBS for a given task. For example:

```
$('#gantt_container').GanttControl({
    .....
    .....
    WBSIDBinding: new RadiantQ.BindingOptions("WBSID"),
    ProvideWBSID: ProvideWBSIDHandler,
});

// Generate WBS codes of the form 1.10.1
function ProvideWBSIDHandler(sender, args) {
    var parent = args.Activity.Parent;
    var childIndex = (args.GetActivityChildIndex() + 1).toString();
    if (parent == null)
        args.NewWBSID = childIndex;
    else
        args.NewWBSID = parent.DefaultWBSID + "." + childIndex;
}
```

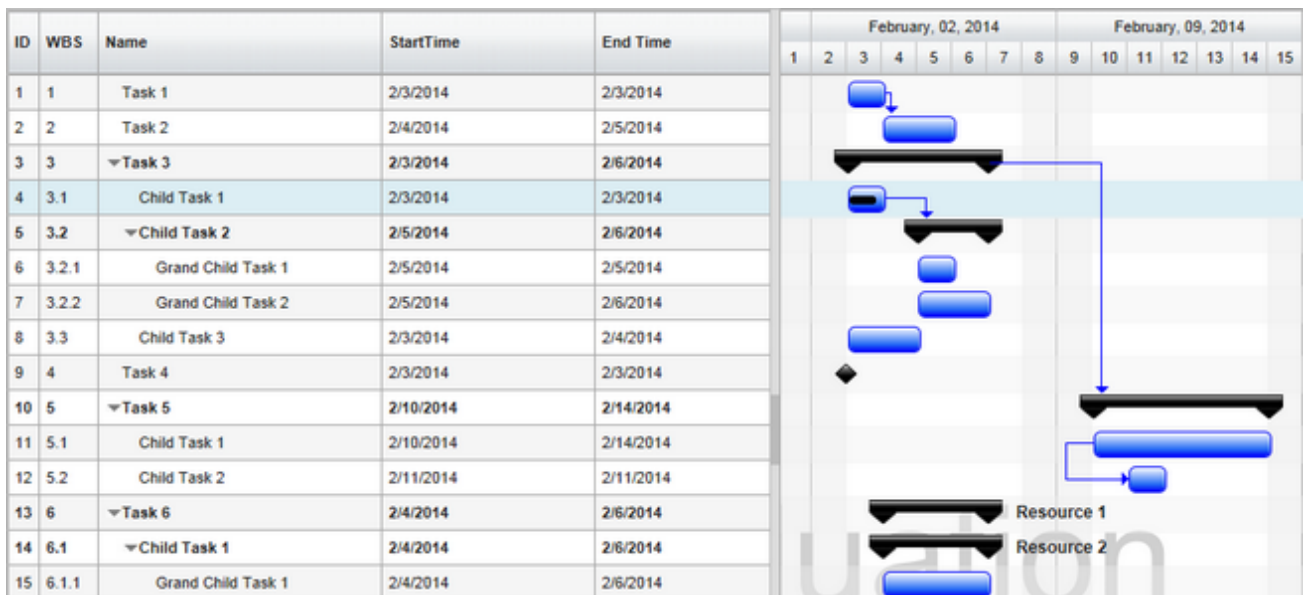
This will set an corresponding, appropriate WBS code on all the activities that are in the model. You can visualize this WBS in the Gantt UI by adding a column to the Gantt's grid as follows:


```

var columns = [
  {
    field: "Activity_M().ID_M()",
    title: "ID",
    width: 25,
    iseditable: false
  },
  {
    field: "Activity_M().ActivityName_M()",
    title: "Activity Name",
    width: 200,
    editor: RadiantQ.Default.Template.ProjectGanttExpandableTextboxEditor(),
    template:
RadiantQ.Default.Template.ProjectGanttExpandableTextBlockTemplate()
  },
  {
    field: "Activity_M().WBSID_M()",
    title: "WBS",
    isParentEditable: false,
    editor: "<input data-bind='value:Activity_M().WBSID_M' />",
    width: 100
  },
  .....
  .....
]

```

This will add the WBS code to all the activities, and keep it updated as you move the tasks around.



Gantt With WBS

To access a WBS of any activity, you can use the Activity.WBSID property.

Persisting WBS Codes

You can now optionally choose to persist the WBS codes into your database by setting up the WBSIDBinding property.

```

$( '#gantt_container' ).GanttControl({
  .....
  .....
  WBSIDBinding: new RadiantQ.BindingOptions("WBSID")
});

```

This will persist the WBS code assigned to a task into your data source that you can reference from outside the gantt, if necessary.

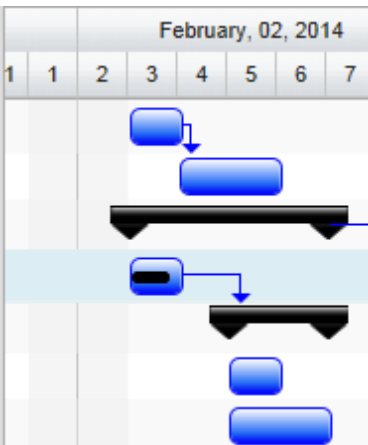
Enable End-User Editing

You can also optionally choose to allow the end-user to edit the WBS codes assigned to a task. First enable editing in the WBS column as below:

```
var columns = [
  {
    field: "Activity_M().WBSID_M()",
    .....
    .....
    editor: "<input data-bind='value:Activity_M().WBSID_M' />",
  }
]
```

User can then start editing and specify any string value they see fit as the WBS code for any task:

ID	WBS	Name
1	1	Task 1
2	2	Task 2
3	3	▼Task 3
4	3.1	Child Task 1
5	3.2	▼Child Task 2
6	3.2.1	Grand Child Task 1
7	3.2.2	Grand Child Task 2



WBS Editing

The gantt will then preserve this WBS code as the task is moved around in the hierarchy. To distinguish such user-specified WBS and auto-generated WBS, the activity interface has the following properties:

Activity.WBSID - The current WBSID assigned to this activity. This could be the auto-generated one or end-user edited one.

Activity.IsAutoWBSID - This bool property indicates if the above WBSID property has a auto-generated value (true) or a end-user provided value(false).

Activity.DefaultWBSID - The auto-generated WBSID for this activity.

The end-user can also clear a custom WBSID that he might have set on a task by simply editing and clearing the text in the TextBox. This will make the gantt assign the auto generated WBS value to the task.

The edited values are then persisted into the database if the WBSIDBinding is set.

Note that while persisting, to distinguish auto-generated and custom WBS values, the custom values are prefixed with a "*". If you browse these WBS values directly in the database, remember to trim this "*" out before you use it.

This is illustrated in this samples:

In HTML : ..\Samples\WBSEnabledTasks.htm.

In ASP.NET MVC : ..\Views\Home\ProjectGantt\WBSEnabledTasks.cshtml.

In ASP.NET : ..\Samples\ProjectGantt\WBSEnabledTasks.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

Undo/Redo

RadiantQ jQuery Gantt Package

Enabling Undo/Redo

Enabling Undo/Redo

The GanttControl has support for tracking user actions and building an undo/redo stack that the users can then operate on. You can turn on this feature as follows:

```
$('#gantt_container').GanttControl({
    EnableRecordingActions: true,
});
```

Once you turn on this feature all operations on the gantt made by the user now gets recorded:

- Moving, Resizing Tasks in the Gantt Chart.
- Connecting Tasks with a dependency in the Gantt Chart.
- Progress percent resize.
- All editing operations in the grid.
- Dragging tasks around in the grid, to change their order.
- Expand/Collapse of tasks in the view.
- etc.

Note that when you enable Undo in the gantt, all operations made on the gantt should go through this ActionManager, otherwise undo actions become unreliable. In other words, you should not make any changes on the gantt model programmatically without going through the ActionManager. Refer to this [topic](#) on how to add custom actions to the ActionManager.

This is illustrated in this samples:

In HTML : ..\Samples\GanttControlUndoRedo.htm.

Undo and Redo

You can then allow users to Undo or Redo actions on the top of the stack using the following APIs:

```
var ganttControl = $('#gantt_container').data('GanttControl');
// To Undo the action on top of the undo stack
ganttControl.ActionManager.Undo();

// To Redo the action on top of the redo stack
ganttControl.ActionManager.Redo();
```

Undo / Redo List

The list of Undo / Redo commands is exposed by the ActionManager in this list:

```
var undoList = ganttControl.ActionManager.EnumUndoableActions();
var redoList = ganttControl.ActionManager.EnumRedoableActions();
```

You can use this to a UI list (ListControl, ComboBox, etc.) to show your users the list of actions in the undo/redo stack.

Note: Multiple indent/outdent feature will not support when undo/redo feature is enabled.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Adding Undo actions Programmatically

You will often have to add custom actions into the gantt's ActionManager when you have the Undo feature enabled. As mentioned in the previous topic, when Undo is enabled, you should not make any changes on the gantt model programmatically without going through the ActionManager.

Some common actions that you will have to add manually into the ActionManager are:

Adding new Tasks

Use the built-in AddAction to add a new task in the gantt.

```
var newTask = {
    "Name": "New Task ",
    "ID": ganttControl.Model.GetNewID(),
    "StartTime": new Date("2014/02/02"),
    "Effort": new RQTimeSpan(0, 12, 30, 0, 0),
    "ProgressPercent": 50,
    "Description": "Description of Task"
};
var addAction = new RadiantQ.Gantt.AddAction(ganttControl, newTask);
ganttControl.ActionManager.RecordAction(addAction);
```

This Action removes the added task object from the list on undo and adds it back again on redo.

Deleting an existing Task

Use the build-in DeleteAction action to delete an existing task in the gantt:

```
var activity = ganttControl.SelectedActivity;
if (activity)
{
    if (activity.ChildActivities_M().length > 0)
    {
        alert("Children have to be deleted before parent to support Undo.");
        return;
    }
    var delAction = new RadiantQ.Gantt.DeleteAction(ganttControl, activity);
    ganttControl.ActionManager.RecordAction(delAction);
}
else
    alert("Select an activity first, before delete.");
```

This Action caches the deleted object and adds it back into the list on undo.

More built-in Actions

There are a whole lot of other built-in undo actions in the RadiantQ.Windows.Controls.Gantt namespace like the following:

- InsertAction
- IndentAction
- OutdentAction
- MoveSelectedItemsUpAction
- MoveSelectedItemsDownAction

- etc.

This is illustrated in this samples:

In HTML : ..\Samples\GanttControlUndoRedo.htm.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Creating Custom Undo Actions

Creating Custom Undoable Actions is very easy. Create a custom type like the following:

```
yourCustomAction = function(){
    this.DeleteDependencyAction = function()
    {
    }
    this.ExecuteCore = function()
    {
        // Code to execute (redo) the action
    }

    this.UnExecuteCore = function()
    {
        // Code to undo the action.
    }
}
```

And create a new action and add it to the ActionManager as follows:

```
var yourAction = new yourCustomAction(ganttControl, activity, ...whatever else...);
ganttControl.ActionManager.RecordAction(yourAction);
```

Note that when you create a custom action type, remember not to cache any objects that could technically become deleted from the gantt model or view and would become obsolete by the time undo happens. For example, do not cache IActivity instances, instead cache the id of the activity using which you can look up an IActivity instance whenever necessary.

Data Binding

RadiantQ jQuery Gantt Package

About Task Field Types

Task Field Types

The gantt expects the bound fields of the task object to be of a certain type as listed below. If they are not of the required type, you will have to setup converters to provide the gantt the values in appropriate type.

Properties	Data Type
ID (via IDBinding)	number or string
Name (via NameBinding)	string
StartTime (via StartTimeBinding)	DateTime
Effort (via EffortBinding)	RQTimeSpan instance
EndTime (via EndTimeBinding)	DateTime instance. This scenario is discussed in this topic .
PreferredStartTime (via PreferredStartTimeBinding)	DateTime instance
IndentLevel (via IndentLevelBinding)	number
PredecessorIndices (via PredecessorIndicesBinding)	string referencing IDs of predecessors, separated by commas.
ProgressPercent (via ProgressPercentBinding)	number
Resources (via AssignedResourcesBinding)	string referencing one or more resource IDs separated by commas.

Effort type

This should represent the effort for a task represented in RQTimeSpan.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

XML Data

Using XML Data

XML is another common way web services expose data to the client. Here is some code that shows how the XML data retrieved has to be "massaged" before binding it to the gantt.

In HTML

```
$.ajax({
  type: "GET",
  dataType: 'xml text',
  url: 'GanttData.xml',
  converters:
  {
    "xml text": function (data) {
      // We use the xml2json jquery plugin to convert xml to json.
      var json = $.xml2json(data).task;
      return $.parseJSON(window.JSON.stringify(json), true
      /*converts date strings to date objects*/
      , true
      /*converts ISO dates to local dates*/
      );
    }
  },
  success: function (data) {
    self.jsonData = data;
    $.holdReady(false);
  }
});
```

And binding the above json list to the Gantt,

In HTML

```
$ganttControl.GanttControl({
  ProjectStartDate: anchorTime,
  DataSource: self.jsonData,
  ....
  ....
});
```

In ASP.NET MVC

```

var AjaxSettings =
{
  dataType: 'xml text',
  converters:
  {
    "xml text": function (data) {
      // We use the xml2json jquery plug-in to convert xml to json.
      var json = $.xml2json(data).TaskInfo;
      var str = window.JSON.stringify(json);
      var parse = $.parseJSON(str, true
      /*converts date strings to date objects*/
      , true
      /*converts ISO dates to local dates*/
      );
      return parse;
    }
  }
};

```

And pass the AjaxSettings to Gantt,

In ASP.NET MVC

```

@Html.JQProjectGantt(
  new JQProjectGanttSettings()
  {
    ControlId = "ganttt_container",
    AfterGanttWidgetInitializedCallback =
    "AfterGanttWidgetInitializedCallback",
    DataSourceUrl = new Uri("/Home/GanttControlXMLItemSource", UriKind
    .RelativeOrAbsolute),
    AjaxSettings = "AjaxSettings",
    Options = new ProjectGanttOptions()
    {
      IDBinding = new Binding("ID"),
      NameBinding = new Binding("Name"),
      IndentLevelBinding = new Binding("IndentLevel"),
      StartTimeBinding = new Binding("StartTime"),
      EffortBinding = new Binding("Effort"),
      PredecessorIndicesBinding = new Binding("PredecessorIndices"),
      ProgressPercentBinding = new Binding("ProgressPercent"),
      GanttChartOptions = new GanttChartOptions()
      {
        AnchorTime = DateTime.Today
      }
    }
  })

```

In ASP.NET

```

var AjaxSettings =
{
  dataType: 'xml text',
  converters:
  {
    "xml text": function (data) {
      // We use the xml2json jquery plug-in to convert xml to json.
      var json = $.xml2json(data).TaskInfo;
      var str = window.JSON.stringify(json);
      var parse = $.parseJSON(str, true
      /*converts date strings to date objects*/
      , true
      /*converts ISO dates to local dates*/
      );
      return parse;
    }
  }
};

```

And pass the AjaxSettings to Gantt,

In ASP.NET

```

<RQ:GanttControl ID="gantt" DataSourceUrl
="../../DataSources/GanttDataBoundToXML.ashx" AjaxSettings = "AjaxSettings"
AfterGanttWidgetInitializedCallback="AfterGanttWidgetInitializedCallback"
LocalizationResourceFilePath="~/Src/ResourceStrings/"
runat="server" />

```

NOTE: If you are using namespaces in your XML (for example, <ns:StartTime>...</ns:StartTime>), then the above xml2json works slightly differently between browsers.

IE: In IE, the above element is referenced in the containing parent object with a property named "ns:StartTime".

Other browsers: The above element is referenced with a property named "StartTime".

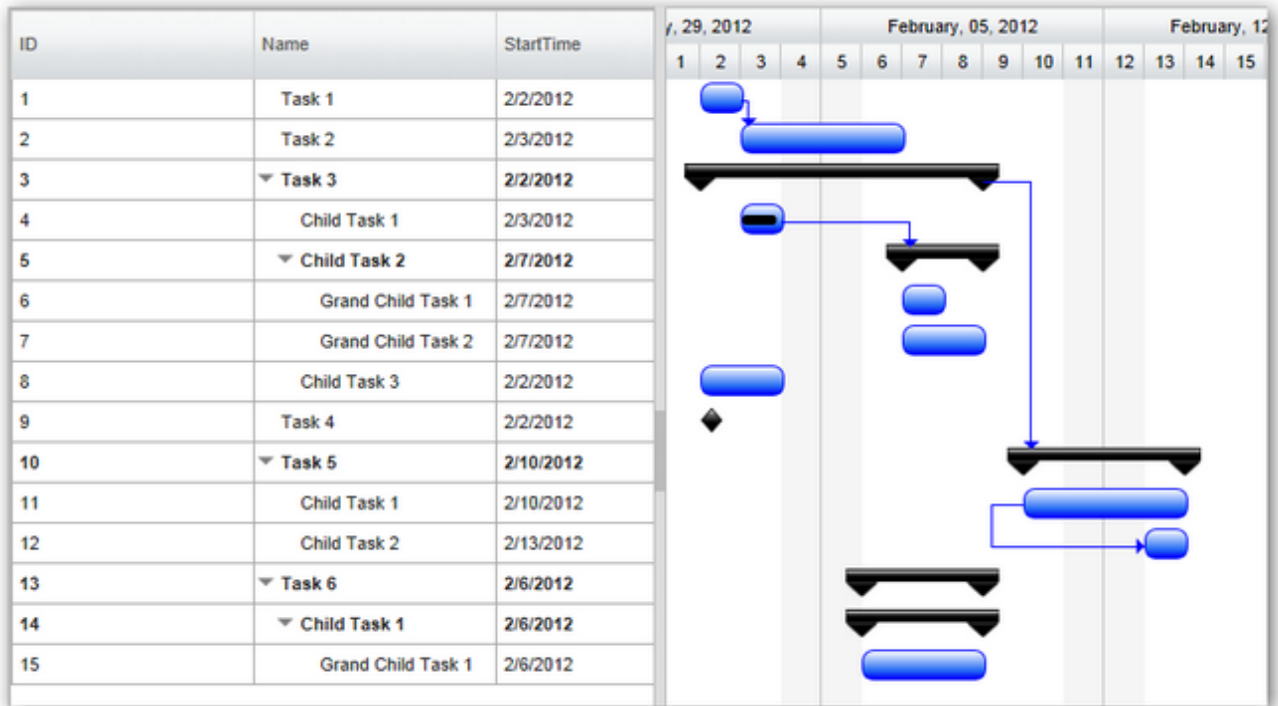
To workaround this issue, you can determine what the property name would be by probing into your first task list item as follows:

```

// For eg: Using the namespace "ns" in xml,
<task Name="Task 1" ID="1" ns:StartTime="2012-02-02T00:00:00Z" Effort="08:00:00"
Description="Description of Task1"></task>
// In Firefox the property ends up being StartTime and in IE it's ns:StartTime;
var taskStartTimeProperty = self.jsonData[0]["ns:StartTime"] ? "ns:StartTime" :
"StartTime";

```

Here is the resultant gantt:



This is illustrated in this samples:

- In HTML : ..\Samples\GanttBoundToXML.htm.
- In ASP.NET MVC : ..\Views\Home\ProjectGantt\GanttBoundToXML.cshtml.
- In ASP.NET : ..\Samples\ProjectGantt\GanttBoundToXML.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

RadiantQ jQuery Gantt Package

Using RadiantQ Binding

Why RadiantQ Binding?

RadiantQ Provides a highly customizable Binding engine which let you to bind the model, it improves performance of the Gantt, while comparing to other Binding engine with gantt and it does not require any model changes.

You will need data bindings only if you want some portions of the gantt to dynamically update it's UI when the underlying value changes. This is how dynamic updates are possible / setup in the grid and chart area of the gantt, in the absence of RadiantQ Binding:

Dynamic UI in the Grid Area

The grid cells as setup in our samples use our internal binding support to automatically listen to changes in the underlying activity's values and update itself. So, there is no need for bindings here.

Dynamic UI in the Chart Area

Take a look at the different ways to customize the look and feel of the task bars [here](#) using templates. While for the most part the templates allow you to customize the look based on the underlying activity's property values, they don't automatically redraw the bars when a value affecting the look of the bars change. Except by forcing a redraw of a bar manually.

Model Changes.

The Bound property should trigger the property changes event when it's value changes. Here is an example(if you are creating the model in project.)

```
function Task() {

    //the argument is optional.
    this.PropertyChanged = new ObjectEvent("PropertyChanged");

    var cost = 1;
    Object.defineProperty(this, "Cost",
    {
        get: function () { return cost; },
        set: function (newVal) {
            cost = newVal;
            // rise the PropertyChanged, in setter, to notify the property changes in
bindings.
            this.PropertyChanged.raise(this, {
                PropertyName: "Cost",
                value: cost
            });
        }
    });

    //.. other property definition.
}
```

This is not a most common case, the data may come from different server or service in json format. In this case you don't have to create a brand new model for data to trigger the property changes instead of you can inject the get/set in data using the RadiantQ.Gantt.Utills.InjectGetAndSetOnData utility. Here is an example for that.

```
$.ajax({
  type: "GET",
  dataType: 'json',
  url: 'GanttControlCostTracking.json',
  converters:
  {
    "text json": function (data) {
      return $.parseJSON(data, true /*converts date strings to date objects*/,
true /*converts ISO dates to local dates*/, function (key, value) {

        if (key == "Cost") {
          //To inject get and set to trigger the property changed.
          RadiantQ.Gantt.Utils.InjectGetAndSetOnData(this, key);
          return value;
        }
        return value;
      });
    }
  },
  success: function (data) {
    //data - source for the ProjectGantt.
  }
});
```

Note: Binding Activity's property does not require the Get/set injection.

View Changes.

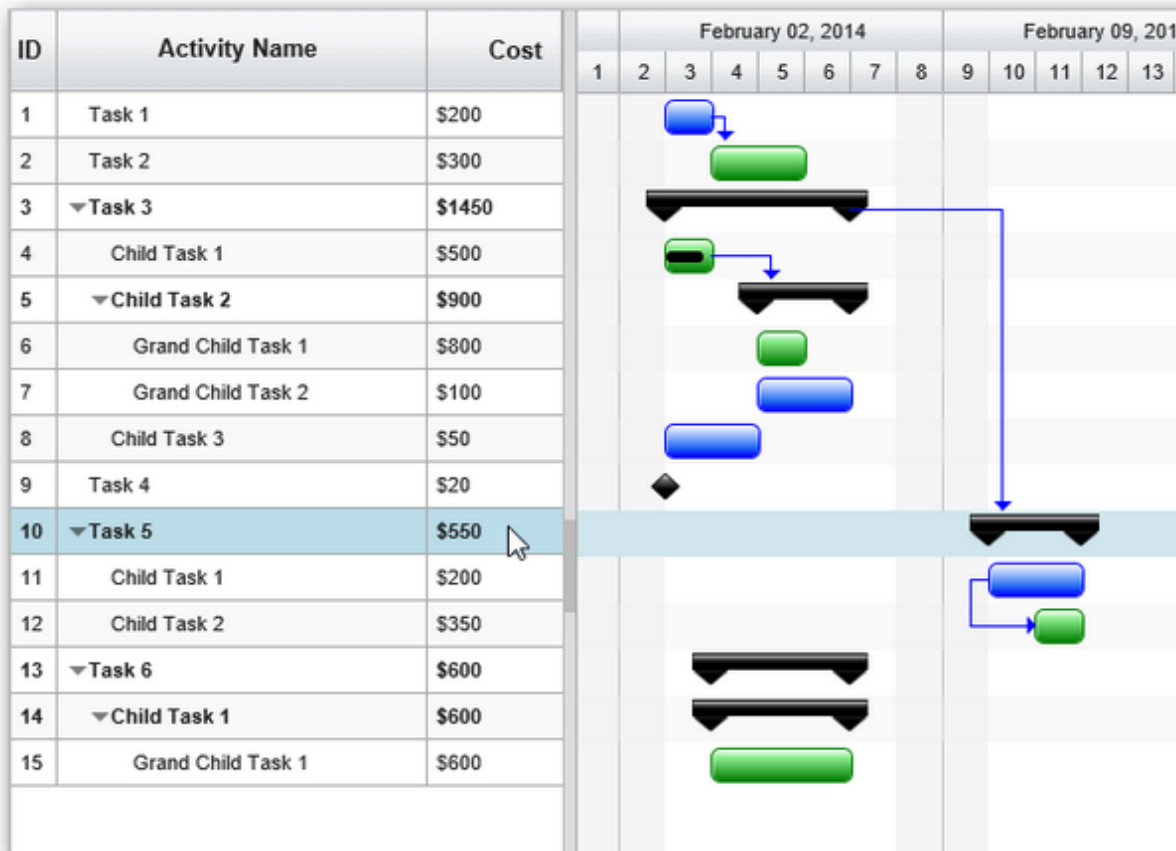
Binding should be specified using the data-bind attribute.

```
//TaskColor - The name of the custom binder.
//Activity.DataSource.Cost - The name of the bound property.
var taskTemplate = "<div class='rq-gc-taskbar'
data-bind='TaskColor:Activity.DataSource.Cost' ><div id='GanttTaskBarLabel'
class='rq-gc-taskbar-label'></div></div>";

// Initialize the FlexyGantt widget.
$gantt_container.GanttControl({
  TaskItemTemplate: taskTemplate,
  //..other options.
});
```

Custom Binder.

```
//A custom binder which changes the task bar background-image based on the cost.
RadiantQ.Binder.TaskColor = function ($elem, role, value, data) {
  this.element = $elem;
  this.value = value;
  this.data = data;
  //called while initializing the TaskColor binding.
  this.init = function () {
    var cost = value.getter(data);
    if (cost > 200) {
      this.element[0].style.setProperty('background-image',
"url(Src/Styles/Images/greenBar.png)", "important");
      this.element.css("border-color", "green", "!important");
    }
    else {
      this.element[0].style.setProperty('background-image',
"url(Src/Styles/Images/TaskBar.png)", "important");
      this.element.css("border-color", "blue", "!important");
    }
  }
  //called when cost is changed.
  this.refresh = function () {
    this.init();
  }
}
}
```



Project Gantt Custom task bar

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Using Knockout

Using Knockout(KO)

[Knockout](#) is a JavaScript library that helps you to create clean, rich, responsive display with a clean underlying data model - using an MVVM approach. Any time you have sections of UI that update dynamically (e.g., changing depending on the user's actions or when the bound data source changes), KO can help you implement it more simply and maintainably.

Why Knockout in gantt?

By default Gantt has its own [data binding](#) utilities that are light-weight, providing templating support, observable collections, etc. These utilities can be used outside the gantt as well. So, you don't really need to use KO.

But, if your page has to use KO for any other reason, then you can use the same in the Gantt as well.

Gantt Setup with Knockout

To begin with add the JSON task list to the Knockout "view model" as follows:

```
var mappingFilter = {
  // Do not let Knockout transform the TimeSpan objects.
  'Effort': {
    create: function (data) {
      return ko.utils.unwrapObservable(data.data);
    }
  }
}

viewModel = {
  Tasks: ko.mapping.fromJS(self.jsonData, mappingFilter)
};
```

Define a Knockout template, which usually has some way to customize the look of the gantt based on an activity property value. In this case if the "ProgressPercent" is < 50, we draw the bars in red and green otherwise.

```
// Customizing the taskbar color based on ProgressPercent
var tTemplate = "<div data-bind=\"attr: { 'class':
RQDataContext.ProgressPercent() > 50 ? 'rq-gc-taskbar bluebar-style' :
'rq-gc-taskbar greenbar_style'}\" ><div data-bind=\"text:
RQDataContext.Description()\"
class='rq-gc-taskbar-label'></div></div>";

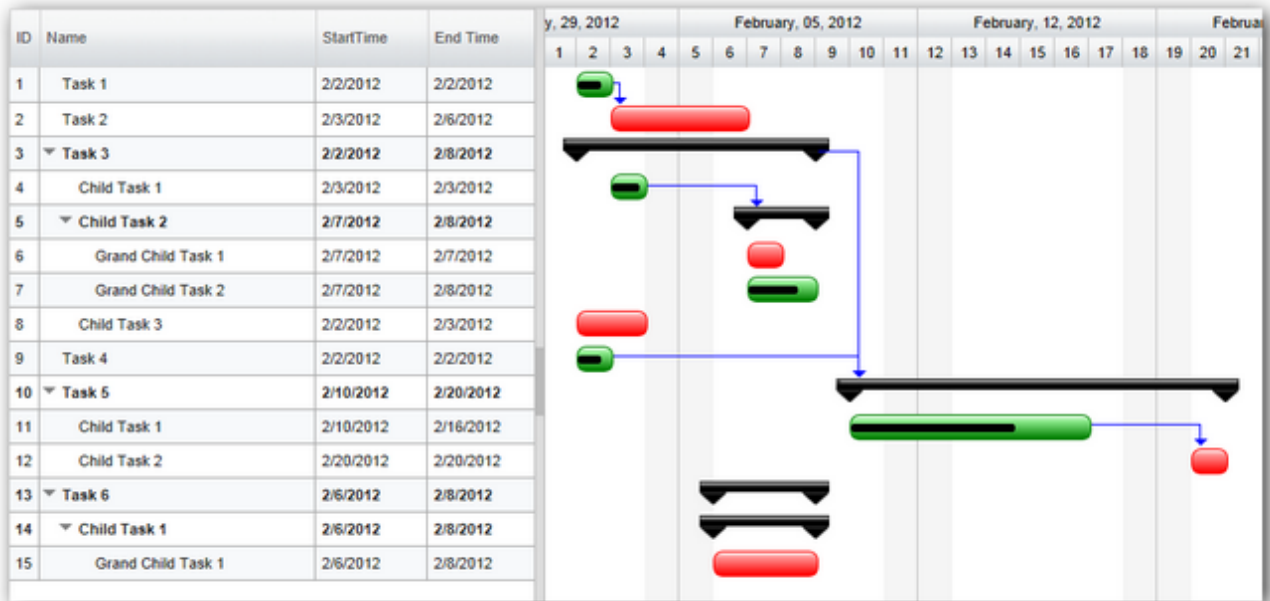
$ganttControl.GanttControl({
  ProjectStartDate: anchorTime,
  DataSource: viewModel.Tasks(),
  TaskItemTemplate: tTemplate,
  ....
  ....
});
```

And since we are using Knockout, the task bars will update themselves as and when the progress value changes.

Note: "RQDataContext" will simply be your original bound data object representing the task

data.

Here is the resultant gantt:



This is illustrated in this samples:

- In HTML : ..\Samples\GanttControlSkeletonUsingKO.htm.
- In ASP.NET MVC : ..\Views\Home\ProjectGantt\GanttControlSkeletonUsingKO.cshtml.
- In ASP.NET : ..\Samples\ProjectGantt\GanttControlSkeletonUsingKO.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

Persisting Changes

RadiantQ jQuery Gantt Package

Persisting Changes Immediately

We will discuss how to persist changes immediately as they happen in the client, using AJAX. The next topic shows how to [Persisting changes in bulk](#).

Caching Changes

Even though we are saving changes immediately you would still need a cache because, often a single operation by the user (for example, moving a task) would result in changes to multiple tasks (dependant tasks get moved, parent bars get resized, etc.) and when you are listening to changes in your data source, you want to wait until all these changes are completed before sending them to the server for persistence.

Here we are listening to the PropertyChanged event which will get called for every changes made to the gantt bound task properties and CollectionChanged to subscribe the PropertyChanged event. Take a look at [this topic](#) for information on other ways to listen to task property value changes.

```

var timer = null;
gantControl = $gant_container.data("GanttControl");
var activityViews = gantControl.ActivityViews;
for (var i = 0; i < activityViews.length; i++) {
    var view = activityViews[i];
    var activity = gantControl.ActivityViews[i].activity;

    var task = activity.DataSource_M();
    //To listen the task changes.
    task.PropertyChanged.subscribe(PropertychangeNotifier);
}

//To listen the task collection changes.
activityViews.CollectionChanged.subscribe(function (event, ui) {
    if (event.type === "insert") {
        var task = ui.items[0].Activity_M().DataSource_M();
        task.PropertyChanged.subscribe(PropertychangeNotifier);
    }
    else {
        var task = ui.items[0].Activity_M().DataSource_M();
        task.PropertyChanged.unsubscribe(PropertychangeNotifier);
    }
});
var updatedTasks = new RadiantQ.Gantt.Dictionary();
var timer = null;
//This will be called only once when many task changed.
function PropertychangeNotifier(sender, args) {
    updatedTasks.Add(sender.ID, sender);
    //To update the server using timer, so that it will update the server only once
when the task and depended tasks changed
    if (timer) {
        clearTimeout(timer);
    }
    timer = setTimeout(function () {
        UpdateModifiedTasks();
    }).bind(this), 500);
}

function UpdateModifiedTasks() {
    if (updatedTasks.length == 0)
        return;
    // $.post('/Home/UpdateModifiedTasks',
    // { updatedTasks: JSON.stringify(updatedTasks) },
    // function (data) {
    //     alert(data);
    // });
    updatedTasks = new RadiantQ.Gantt.Dictionary();
}

```

Updating Added and removed Task To Database Immediately

When a task is added to the GanttControl, send it to the server right away for persisting.

```

$("#addRow").click(function () {
    var newTask = getNewTask();
    //To add a task to GanttControl.
    ganttControl.AddNewItem(newTask);
    UpdateAddedTasks(newTask);
    return false;
});

function UpdateAddedTasks(addedTask) {
    // $.post('/Home/UpdateAddedTasks',
    //     { addedTasks: JSON.stringify(addedTasks) },
    //     function (data) {
    //         alert(data);
    //     });
}

```

When the task is removed from gantt, it might potentially remove more tasks if it's a parent. So, make sure to notify the server with all those task IDs.

```

$("#remove").click(function () {
    var activitys = null;
    var activity = ganttControl.SelectedActivity;
    if (activity) {
        var ganttItemSource = ganttControl.options.DataSource;
        var removedTaskIds = [];
        //To Remove a task from GanttControl.
        activitys = ganttControl.RemoveActivity(activity.ID);
        for (var i = 0; i < activitys.length; i++) {
            var boundData = activitys[i].DataSource;
            removedTaskIds.push(boundData.ID);
            //To Remove a task from GanttControl bound data source.
            ganttItemSource.splice(ganttItemSource.indexOf(boundData), 1);
        }
        UpdateRemovedTasks(removedTaskIds);
    }
    else
        alert("Please select an item in the table first.");
    return false;
});

function UpdateRemovedTasks(removedTaskIds) {
    //$.post('/Home/UpdateRemovedTasks',
    //     { removedTaskIds: JSON.stringify(removedTaskIds) },
    //     function (data) {
    //         alert(data);
    //     });

    removedTaskIds = [];
}

```

For a live example of the approach discussed here, refer to one of these samples:

In HTML : ..\Samples\GanttControlEditsImmediateSave.htm.

In ASP.NET MVC : ..\SQLBinding\ASPNetSamples\GanttControlEditsImmediateSave.cshtml.

In ASP.NET : ..\SQLBinding\MVC4Samples\GanttControlEditsImmediateSave.aspx.

RadiantQ jQuery Gantt Package

Persisting Changes in Bulk

This topic describes how persisting changes in Bulk using a button click. The previous topic shows [how to Persisting Changes Immediately](#).

Caching Changes

Cache the changes made by the user so you can send the changes to the server in bulk at a later time. Here we are listening to the ActivityUpdated event which will get called for all changes made to the gantt bound task properties. Take a look at [this topic](#) for information on other ways to listen to task property value changes.

```
//To listen to activity changes.
ganttControl.Model.ActivityUpdated.subscribe(PropertychangeNotifier);
// To listen to property changes in the bound task object.
function PropertychangeNotifier(sender, args) {
    var boundData = args.DataSource;
    // Caching updated tasks, unless it's already in the cached tasks list.
    if (!addedTasks.Contains(boundData.ID))
        updatedTasks.Add(boundData.ID, boundData);
}
```

Updating Added and Removed Tasks in Bulk.

When a task is added to gantt, add the task object to a dictionary (later parsed as string) .

```
//To cache the newly added task objects.
var addedTasks = new RadiantQ.Gantt.Dictionary();

$("#addRow").click(function () {
    var newTask = getNewTask();
    ganttControl.AddNewItem(newTask);
    //Add the newly added task to Dictionary.
    addedTasks.Add(newTask.ID, newTask);
    return false;
});
```

When the task is removed from the gantt, cache it's task id. Remove this task from the "addedTasks" and "updatedTasks" list, if it's there.


```

//To cache the Id of the removed tasks
var removedTaskIds = [];

// To remove the selected task.
$("#remove").click(function () {
    var activitys = null;
    var activity = gantt.SelectedActivity;
    if (activity) {
        var ganttItemSource = gantt.options.DataSource;
        activitys = gantt.RemoveActivity(activity.ID);
        for (var i = 0; i < activitys.length; i++) {
            var boundData = activitys[i].DataSource;
            // To check and remove the task from addedTasks and updatedTask list.
            cacheRemovedTask(boundData.ID);
            // To remove the task from bounded datasource
            ganttItemSource.splice(ganttItemSource.indexOf(boundData), 1);
        }
    }
    else
        alert("Please select an item in the table first.");
});

cacheRemovedTask = function (removedTaskID) {
    removedTaskIds.push(removedTaskID);
    //To remove the removed task from addedTasks list
    if (addedTasks.Contains(removedTaskID))
        addedTasks.Remove(removedTaskID);
    //To remove the removed task from updatedTasks list
    if (updatedTasks.Contains(removedTaskID))
        updatedTasks.Remove(removedTaskID);
}

```

When the user is ready to save the changes, he could click on a "Save" button for example in whose handler you could call the UpdateModifiedTasks() method below to save the changes in the server.

```

$("#save").click(function () {
    //$.ajax({
    //    type: "POST",
    //    url: '../..../DataSource/Save.ashx',
    //    dataType: "json",
    //    data: { addedTasks: JSON.stringify(addedTasks.asArray), removedTaskIds:
JSON.stringify(removedTaskIds) },
    //    async: false,
    //});
    alert("Data is updated!");
    //clearing the cached values;
    addedTasks = new RadiantQ.Gantt.Dictionary();
    removedTaskIds = [];
    updatedTasks = new RadiantQ.Gantt.Dictionary();
});

```

For a live example of the approach discussed here, refer to one of these samples:

In HTML : ..\Samples\GanttControlCustomDataBinding.htm.
 In ASP.NET MVC : ..\SQLBinding\MVC4Samples\Views\Home\Index.cshtml.
 In ASP.NET : ..\SQLBinding\MVC4Samples\SQLDataBinding.aspx.

EndTime in DataSource

RadiantQ jQuery Gantt Package

EndTime field without Effort field

Often, you might have a data source with EndTime field rather than an Effort field for a task.

In such cases, you have to first setup EffortBinding using a converter as follows:

```

var effortConverter = {
  Convert: function (value, parameter) {
    var task = parameter;
    return effortConverter.GetTaskEffort(task);
  },
  ConvertBack: function (value, parameter) {
    // Compute EndTime based on StartTime and Effort.
    var duration = value;
    var task = parameter;

    // Note that "WorkTimeSchedule.Schedule8X5" is the same schedule used
    // by the GanttControl
    // in this app. If you use a different schedule in the GanttControl,
    // use that same schedule here as well:
    return
    RadiantQ.Gantt.WorkTimeSchedule.Schedule8X5.GetEnd(task.StartTime, duration);
  },

  GetTaskEffort: function (task) {
    // Note that "WorkTimeSchedule.Schedule8X5" is the same schedule used
    // by the GanttControl
    // in this app. If you use a different schedule in the GanttControl,
    // use that same schedule here as well:
    if (task.StartTime && task.EndTime) {
      if (task.StartTime && task.EndTime)
        return
        RadiantQ.Gantt.WorkTimeSchedule.Schedule8X5.GetEffort(task.StartTime, task.EndTime);
    }
    return RQTimeSpan.Zero;
  }
};

$gantt_container.GanttControl({
  .....
  .....
  EffortBinding: new RadiantQ.BindingOptions("EndTime",
RadiantQ.Gantt.BindingMode.TwoWay, new effortConverter()),
  EndTimeBinding: new RadiantQ.BindingOptions("EndTime"),
  .....
  ....
});

```

Also see how you will have to setup the EndTimeBinding so that changes made to the EndTime are persisted back into your data's EndTime property.

This is also illustrated in this sample that's part of our install: <install path>\Samples\GanttControlBindingToEndTime.htm

(The above sample also sets the `AdjustDurationOnAssignment` property to false to make the gantt work in "Fixed Duration" mode. You can choose to keep it true to make the gantt adjust the duration of a task when multiple assignments are made.)

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

EndTime field with Effort field

There are 2 reasons why you could choose to have an EndTime field in your data source along with your Effort field.

- a) You might want this EndTime information in some other context, not involving the gantt.
- b) Help gantt's load time performance. This is especially helpful when you have long running tasks or a lot of tasks.

In this scenario you can simply bind your EndTime field to the gantt through the EndTimeBinding property. The gantt will then use this value while loading as well as keep this value updated as the user moves/resizes the task around.

```
$gantt_container.GanttControl({
    .....
    .....
    EffortBinding: new RadiantQ.BindingOptions("Effort"),
    EndTimeBinding: new RadiantQ.BindingOptions("EndTime"),
    .....
    ....
});
```

NOTE: When you do setup EndTimeBinding like this, if you adjust your project's schedule (to add holidays, for example), these EndTimes could become potentially invalid. So, make sure to get these EndTimes adjusted whenever the schedule changes.

This is also illustrated in the following sample that is part of our install: <install path>\Samples\TrackingEndTimeInDataSource.htm

RadiantQ jQuery Gantt Package

Data Binding in PHP

Here we discuss how to bind the gantt to a database table in PHP and then discuss how to persist changes made by the end user into the database in the 2 child topics.

Establishing Connection

We cover the following 2 database types.

- MYSQL Database
- SQLite database(.db) file

a) MYSQL connection is established using PHP PDO statement using 'DatabaseUtilities' (which is inside ../PHPWrappers\RadiantQ.Web.PHP.JQGantt\lib) which can be called as follows,

```
// Establishes connection for MYSQL.
$db = new DatabaseUtilities('mysql:host=localhost;dbname=taskdb', 'tasks',
$properties, 'root', 'password');
```

Above, the 'mysql host and database name' is given as the first argument, table name is the second argument, the 'root' and 'password' are the third and fourth arguments respectively.

b) For establishing a connection to an SQLite database(.db) file, the arguments of DatabaseUtilities are as follows,

```
// Establishes connection for SQLite Database file.
//Username and password are not necessary for sqlite db file. Hence null values are
passed as arguments.
$db = new DatabaseUtilities('sqlite:../Tasks.db', 'tasks', $properties, null, null);
```

In this the first argument is the 'path to the database file'. The second is the 'tablename'.

Note : The following implementation for enabling CRUD operations are common for both MYSQL and SQLite.

To Fetch data from Database

To read data from database you have to call the "fetchTask" function as below,

```
// To read data from database.
$result = $db->fetchTask();
```

The fetchTask() performs the following operation to read data from database.

```
public function fetchTask()
{
    // Create SQL select statement
    $statement = $db->prepare("SELECT * FROM tasks");
    // Execute the statement
    $statement->execute();
    // The result of 'read' operation is all tasks from the tasks table.
    $result = $statement->fetchAll(PDO::FETCH_CLASS);
    return $result;
}
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Persisting Changes Immediately

We will discuss how to persist changes immediately as they happen in the client, using as POST.

Caching Changes

Even though we are saving changes immediately you would still need a cache because, often a single operation by the user (for example, moving a task) would result in changes to multiple tasks (dependant tasks get moved, parent bars get resized, etc.) and when you are listening to changes in your data source, you want to wait until all these changes are completed before sending them to the server for persistence.

Here we are listening to the ActivityUpdated event which will get called for all changes made to the gantt bound task properties. Take a look at [this topic](#) for information on other ways to listen to task property value changes.

```

var timer = null;
//To listen to the activity changes.
ganttControl.Model.ActivityUpdated.subscribe(PropertychangeNotifier);
// To listen to property changes in the bound task object.
function PropertychangeNotifier(sender, args) {
    var boundData = args.DataSource;
    // Caching updated tasks, unless it's already in the added tasks list.
    if (!addedTasks.Contains(boundData.ID))
        updatedTasks.Add(boundData.ID, boundData);
        if (timer) {
            clearTimeout(timer);
        }
        timer = setTimeout(function () {
            UpdateModifiedTasks();
        }.bind(this), 500);
}

function UpdateModifiedTasks() {
    var UpdatedTasksJson = JSON.stringify(updatedTasks.asArray,
MySQLDateFormatConversion);
//Posting to the same page.
$.post("<?php echo $_SERVER['PHP_SELF'];?>",
    { UpdatedTasks: JSON.stringify(UpdatedTasksJson) },
    function (data) {
        alert(data);
    });
}

function MySQLDateFormatConversion(key, value) {
    if (key == "StartDate" || key == "EndDate" || key == "PreferredStartTime")
        value = new Date(value).toString('yyyy-MM-dd hh:mm:ss');
    return value;
}

//Server side.
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $dbh->UpdateTasks($_POST["UpdatedTasks"]);
}

```

Updating Added and removed Task To Database Immediately

When a task is added to the GanttControl, send it to the server right away for persisting.

```
$("#addRow").click(function () {
    var newTask = getNewTask();
    //To add a task to GanttControl.
    ganttControl.AddNewItem(newTask);
    UpdateAddedTasks(newTask);
    return false;
});

function UpdateAddedTasks(addedTask) {
    var AddedTasksTasksJson = JSON.stringify(addedTask, MySQLDateFormatConversion);
    //Posting to the same page.
    $.post("<?php echo $_SERVER['PHP_SELF'];?>",
        { AddedTasks: JSON.stringify(AddedTasksTasksJson) },
        function (data) {
            alert(data);
        });
}

//Server side.
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $dbh->AddTasks($_POST["AddedTasks"]);
}
```

When the task is removed from gantt, it might potentially remove more tasks if it's a parent. So, make sure to notify the server with all those task IDs.

```
$("#remove").click(function () {
    var activitys = null;
    var activity = ganttControl.SelectedActivity;
    if (activity) {
        var ganttItemSource = ganttControl.options.DataSource;
        var removedTaskIds = [];
        //To Remove a task from GanttControl.
        activitys = ganttControl.RemoveActivity(activity.ID);
        for (var i = 0; i < activitys.length; i++) {
            var boundData = activitys[i].DataSource;
            removedTaskIds.push(boundData.ID);
            //To Remove a task from GanttControl bound data source.
            ganttItemSource.splice(ganttItemSource.indexOf(boundData), 1);
        }
        UpdateRemovedTasks(removedTaskIds);
    }
    else
        alert("Please select an item in the table first.");
    return false;
});

function UpdateRemovedTasks(removedTaskIds) {
    //Posting to the same page.
    $.post("<?php echo $_SERVER['PHP_SELF'];?>",
        { "RemovedTaskIds": JSON.stringify(removedTaskIds) },
        function (data) {
            alert(data);
        });
    removedTaskIds = [];
}

//Server side.
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $dbh->AddTasks($_POST["RemovedTaskIds"]);
}
```

RadiantQ jQuery Gantt Package

Persisting Changes in Bulk

This topic describes how persisting changes in Bulk using a button click.

For a live example of the approach discussed here, refer to this PHP sample:
\Samples\ProjectGantt\DataBaseSample.php

Caching Changes

Cache the changes made by the user so you can send the changes to the server in bulk at a later time. Here we are listening to the ActivityUpdated event which will get called for all changes made to the gantt bound task properties. Take a look at [this topic](#) for information on other ways to listen to task property value changes.

```
//To listen the activity changes.
ganttControl.Model.ActivityUpdated.subscribe(PropertychangeNotifier);
// To listen to property changes in the bound task object.
function PropertychangeNotifier(sender, args) {
    var boundData = args.DataSource;
    // Caching updated tasks, unless it's already in the added tasks list.
    if (!addedTasks.Contains(boundData.ID))
        updatedTasks.Add(boundData.ID, boundData);
}
```

Updating Added and Removed Tasks in Bulk.

When a task is added to gantt, add the task object to the dictionary(later parsed as string) .

```
//To cache the newly added task objects.
var addedTasks = new RadiantQ.Gantt.Dictionary();

$("#addRow").click(function () {
    var newTask = getNewTask();
    ganttControl.AddNewItem(newTask);
    //Add the newly added task to Dictionary.
    addedTasks.Add(newTask.ID, newTask);
    return false;
});
```

When the task is removed from gantt, cache the task ids in an array and also check if the task exists in the added and updated dictionary, and if it does, remove it from those dictionaries.

```

//To cache the Id of the removed tasks
var removedTaskIds = [];

// To remove the selected task.
$("#remove").click(function () {
    var activitys = null;
    var activity = gantt.SelectedActivity;
    if (activity) {
        var ganttItemSource = gantt.options.DataSource;
        activitys = gantt.RemoveActivity(activity.ID);
        for (var i = 0; i < activitys.length; i++) {
            var boundData = activitys[i].DataSource;
            // To check and remove the task from addedTasks and updatedTask list.
            cacheRemovedTask(boundData.ID);
            // To remove the task from bounded datasource
            ganttItemSource.splice(ganttItemSource.indexOf(boundData), 1);
        }
    }
    else
        alert("Please select an item in the table first.");
});

cacheRemovedTask = function (removedTaskID) {
    removedTaskIds.push(removedTaskID);
    //To remove the removed task from addedTasks list
    if (addedTasks.Contains(removedTaskID))
        addedTasks.Remove(removedTaskID);
    //To remove the removed task from updatedTasks list
    if (updatedTasks.Contains(removedTaskID))
        updatedTasks.Remove(removedTaskID);
}

```

When the user is ready to save the changes, he could click on a "Save" button for example in whose handler you could called the UpdateModifiedTasks() method below to save the changes in the server.

```

$("#save").click(function () {
    var UpdatedTasksJson = JSON.stringify(updatedTasks.asArray,
    MySQLDateFormatConversion);
    var AddedTasksJson = JSON.stringify(addedTasks.asArray,
    MySQLDateFormatConversion);
    $.post("<?php echo $_SERVER['PHP_SELF'];?>?type=update", { AddedTasks:
    AddedTasksJson, RemovedTaskIds: removedTaskIds, UpdatedTasks: UpdatedTasksJson },
    function (data) {

        });
        UpdatedTasks = new RadiantQ.Gantt.Dictionary();
        AddedTasks = new RadiantQ.Gantt.Dictionary();
        RemovedTaskIds = [];
    });

function MySQLDateFormatConversion(key, value) {
    if (key == "StartDate" || key == "EndDate" || key == "PreferredStartTime")
        value = new Date(value).toString('yyyy-MM-dd hh:mm:ss');
    return value;
}
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $dbh->SaveTasks($_POST, 'tasks');
}

```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

MS Project Export/Import

Microsoft Project Export

There is built in support for exporting the gantt contents into MS Project compatible XML, that can be opened in Project.

This is illustrated in the sample:

```
<install path>\Samples\GanttControlExportToProjectXML
```

To export the contents of a gantt into Project XML, you can create a Project document from an empty Project XML in memory and add tasks and resources as shown below:

```
saveProject = new RadiantQ.ProjectModel.Project(JsonData);
saveProject.initFrom(ganttControl.Model);
for (var i = 0; i < ganttControl.ActivityViews.length; i++) {
    .....
    .....
}
saveProject.calculateTasksEndTime();
var xml = js2xmlparser("Project", saveProject.projectElem);
window.JSON.toXML(savedJson, "Project");
```

The full source can be seen in the sample referenced above.

Microsoft Project Import

MS Project XML format contains a whole lot of features, not all of which are used by / compatible with the gantt. When you try to import a Project XML into the gantt, you can expect to see the following information imported and the rest ignored:

Task Specific Information

- Task Name
- Duration
- Start
- Finish
- Predecessors with lags and the following types:
 - Finish to Start
 - - Start to Finish
 - - Finish to Finish
 - - Start to Start
 - - Percent complete
- Resource Assignment (including multiple resource assignments, partial assignments)
- Description
- Task Hierarchy (Indent Level)
- Custom Project Schedules

APIs allow you to open a stream of Project XML files, create a Project Document in memory and initialize a gantt from this as shown below.

This is also illustrated in the sample <install path>\Samples\GanttControlMSProjectXMLBinding

```

function readBlob() {
    var fileElement = $('#files');
    var file = fileElement[0].files[0];
    var reader = new FileReader();
    // If we use onloadend, we need to check the readyState.
    reader.onloadend = function (evt) {
        if (evt.target.readyState == FileReader.DONE) { // DONE == 2
            var txt = evt.target.result;
            var parser = new DOMParser();
            var xml = parser.parseFromString(txt, "text/xml");
            init(xml);
        }
    };
    fileElement.replaceWith(fileElement = fileElement.clone(true));
    reader.readAsText(file);
}

function init(xml)
{
    var json = $.xml2json(xml);
    project = new RadiantQ.ProjectModel.Project(json);

    $ganttt_container.GanttControl({ "DataSource": null, "ResourceItemsSource"
: null });
    var delayUpdates1 = new RadiantQ.Gantt.Utils.DelayUpdates()
    {
        $ganttt_container.GanttControl({
            WorkTimeSchedule: project.BaseCalendar.Schedule,
            ResourceItemsSource: project.ResourceList,
            DataSource: project.Tasks
        });
    }
    delayUpdates1.Dispose();

    var delayUpdates2 = new RadiantQ.Gantt.Utils.DelayUpdates()
    {
        gantt = $ganttt_container.data("GanttControl");
        var $GanttChart = gantt.GetGanttChart();
        $GanttChart.GanttChart({ AnchorTime:
project.Tasks[0].StartDate.clone() });
    }
    delayUpdates2.Dispose();
}
}

```

Take a look at the above sample for a fully functional example.

Scheduling Features

RadiantQ jQuery Gantt

Resource Level Schedules

Refer to this [topic](#) where this is discussed.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt

Resource Leveling

Resource Leveling

Easily level resources with a single call using the LevelResources method.

```
var ganttControl = $ganttt_container.data("GanttControl") || $ganttt_container.data("radiantqGanttControl");
```

```
gantttControl.LevelResources (boolIncludePartiallyCompletedTasks, dateTimeStart);
```

boolIncludePartiallyCompletedTasks - Specifies whether or not partially completed tasks should be included in the leveling algorithm.

dateTimeStart - The start time for leveling - usually the current or a time in future.

The GanttControl parses through all the tasks, determines over allocation (any thing more than 100% of a resource usage at a specific time) and then moves the tasks into the future to clear such over allocation of resources.

This is illustrated in the "Samples\ResourceLeveling.htm" sample.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt

Task Level Schedules

Tasks with custom schedules

Specific tasks could occasionally be associated with a custom schedule that dictates when (time of the day and working days) the task could be performed. This could be different from the overall project schedule.

Tasks with custom schedules will then follow these rules:

- Start and End times will be restricted to fall within this schedule.
- When rendered within a "day header", the beginning of the time unit will be the start of the schedule in that day and end of the time unit will be the end of the schedule in that day.

Your task objects could contain information about their custom schedules that can be bound to the gantt. This information could be in any format, but should eventually be converted to a `WorkTimeSchedule` instance.

a)

There are some built-in utilities that can convert a string representation of this custom schedule (in a predefined format) into a `WorkTimeSchedule` instance.

```

var jsonData = GetSampleData();

function GetSampleData() {
    //var tasks = [];
    tasks.push(new CustomTask({ TaskName: "Summary 1", Schedule: "", TaskID:
"9", SortOrder: 1, StartTime: new Date().Date(), RequiredEffort: TimeSpan.parse(
"08:00:00"), Description: "Description of Task 1" }));
    //tasks.push(task);
    tasks.push(new CustomTask({ TaskName: "Task 1", IndentLevel: 1,
Schedule: "", TaskID: "10", SortOrder: 2, PredecessorIndices: "9", StartTime: new
Date().Date(), RequiredEffort: TimeSpan.parse("16:00:00"), Description: "Description
of Task 2" }));
    tasks.push(new CustomTask({ TaskName: "Task 2", IndentLevel: 1,
ScheduleDesc: "9AM to 5PM, Sun to Thu", Schedule: "sun 09:00:00 17:00:00; mon
09:00:00 17:00:00; tue 09:00:00 17:00:00; wed 09:00:00 17:00:00; thu 09:00:00
17:00:00;", TaskID: "11", PredecessorIndices:"10", SortOrder: 3, StartTime: new
Date().Date(), RequiredEffort: TimeSpan.parse("12:30:00"), CompletedEffort:
TimeSpan.parse("5:00:00"), Description: "Description of Task 3" }));
    tasks.push(new CustomTask({ TaskName: "Summary 2", ScheduleDesc: "8AM to
4PM, Mon to Fri", Schedule: "mon 08:00:00 16:00:00; tue 08:00:00 16:00:00; wed
08:00:00 16:00:00; thu 08:00:00 16:00:00; fri 08:00:00 16:00:00;", TaskID: "12",
SortOrder: 4, StartTime: new Date().Date(), AssignedResources: "Resource 1 , Resource
2", RequiredEffort: TimeSpan.parse("08:00:00"), CompletedEffort: TimeSpan.parse(
"2:00:00"), Description: "Description of Task 3/Child Task 1" }));
    tasks.push(new CustomTask({ TaskName: "Task 3", ScheduleDesc: "1PM to
5PM, Tue to Sat", Schedule: "tue 13:00:00 17:00:00; wed 13:00:00 17:00:00; thu
13:00:00 17:00:00; fri 13:00:00 17:00:00; sat 13:00:00 17:00:00;", TaskID: "13",
IndentLevel: 1, SortOrder: 5, StartTime: new Date().Date(), RequiredEffort:
TimeSpan.parse("16:00:00"), CompletedEffort: TimeSpan.parse("4:00:00"),
PredecessorIndices: "12+8", Description: "Description of Task 3/Child Task 2" }));
    tasks.push(new CustomTask({ TaskName: "Task 4", ScheduleDesc: "7AM to
3PM, Mon, Wed, Fri", Schedule: "mon 07:00:00 15:00:00; wed 07:00:00 15:00:00; fri
07:00:00 15:00:00;", TaskID: "14", IndentLevel: 1, SortOrder: 6, StartTime: new
Date().Date(), RequiredEffort: TimeSpan.parse("08:00:00"), CompletedEffort:
TimeSpan.parse("4:00:00"), Description: "Description of Task 3/Child Task 1/Grand
Child 1" }));

    return tasks;
}

```

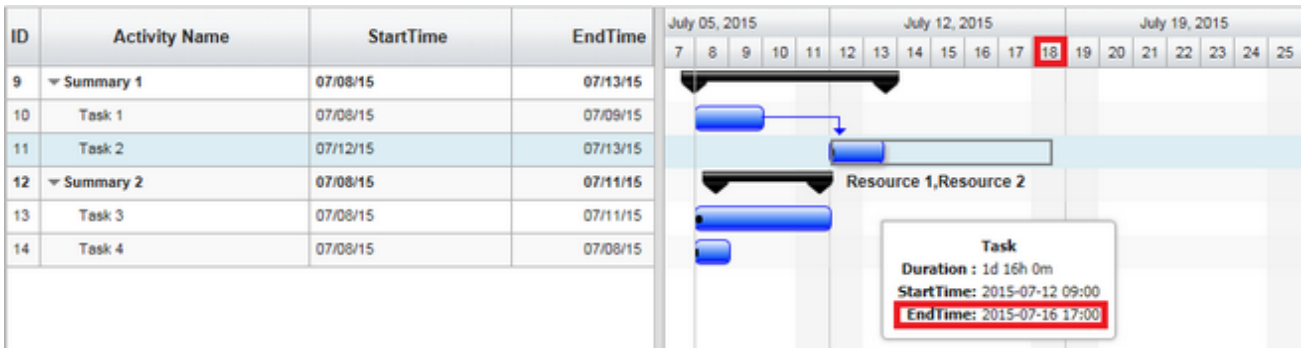
Then you can setup this binding in the GanttControl:

```

$gantt_container = $('#gantt_container');
$gantt_container.GanttControl({
    ScheduleBinding: new RadiantQ.BindingOptions("Schedule", "TwoWay",
RadiantQ.Gantt.StringToWorkTimeScheduleConverter),
});

```

The gantt will then use this custom schedule for the tasks.



Gantt doesn't allow you to resize the bar to a non-working time.

This is illustrated in the Samples\TaskLevelSchedules sample.

RadiantQ jQuery Gantt

Resource Load View

Please see [this](#) topic for more information.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

Behavior Customization

RadiantQ jQuery Gantt Package

Tasks Filtering

Tasks Filtering

Often you might require to visualize a filtered view of the tasks in your project. For example,

- All tasks that are incomplete (< 100% progress).
- All tasks assigned to "Resource A"
- etc.

You could be tempted to simply filter the bound list of rows and then bind it to the gantt. However, that would break relationships between tasks within your project (dependencies, parent/child relationship, etc.). So, you will always have to bind the entire list of tasks to the gantt and then provide filtering criteria to the gantt, so that the gantt renders only the tasks you are interested in, while maintaining the relationship between the different tasks at the model level.

Filter definition

Filtering the tasks is supported in an old fashioned way. You provide a "filter function" that will be called once for each task to determine whether a task should be filtered in(shown) or out(hidden).

For example, this would be the filtering function that filters in tasks that are not yet complete and filters out tasks that are complete:

```
var IncompleteTasksFilter = function(activity)
{
    if (activity.ProgressPercent == 100)
        return false;
    else
        return true;
}
```

A return value of *true* for an activity includes that activity in the view and *false* will exclude the activity from the view.

Note that this function will only be called for non-summary/leaf tasks. The summary task's will be shown if there is at least one child filtered-in and hidden if all it's children are filtered out.

Once you have setup filters like this, you simply have to set the FilterActivities options of the gantt control to the above function and call the Filter method like this:

```
$("#IncompleteTasks").click(function () {
    ganttControl.options.FilterActivities = IncompleteTasksFilter;
    ganttControl.Filter();
});
```

In the code above, you will see that as soon as you call Filter, your filtering method will be called once for each activity in the project.

Filtering is a one time operation, so any changes made to the activity after the event will not alter its visibility. In the example above, after filtering, if a task's progress changes to 100%, it will not automatically get filtered out, you will have to trigger Filter() method again for that to happen.

Resetting Filter

Simply call the ResetFilter method to clear the filter currently applied.

```
gantControl.ResetFilters();
```

This is illustrated in this samples:

In HTML	: ..\Samples\GanttControlFiltering.htm.
In ASP.NET MVC	: ..\Views\Home\ProjectGantt\GanttControlFiltering.cshtml.
In ASP.NET	: ..\Samples\ProjectGantt\GanttControlFiltering.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

Disabling Runtime Features

RadiantQ jQuery Gantt Package

Global ReadOnly

Mark the control as ReadOnly

To disable all kinds of editing by the end-user, simply set the GanttControl.IsReadOnly property to true and set iseditable column property to false.


```

var ganttControl = $gantt_container.data("GanttControl");
$gantt_container.GanttControl('option', 'IsReadOnly', true);

//To make the Column not editable in ganttTable.

var columns = [
{
    field: "Activity_M().ID_M()",
    title: "ID",
    width: 25,
    iseditable: false
},
{
    field: "Activity_M().ActivityName_M()",
    title: "Activity Name",
    width: 200,
    iseditable: false,
    editor: RadiantQ.Default.Template.ProjectGanttExpandableTextboxEditor(),
    template:
RadiantQ.Default.Template.ProjectGanttExpandableTextBlockTemplate()
},
{
    field: "Activity_M().StartTime_M()",
    title: "StartTime",
    width: 150,
    iseditable: false,
    format: "MM/dd/yy",
    editor: "<input data-bind='ActivityTimeBinder:Activity_M().StartTime_M'
/>"
},
{
    field: "Activity_M().EndTime_M()",
    title: "EndTime",
    width: 150,
    iseditable: false,
    format: "MM/dd/yy",
    editor: "<input data-bind='value:Activity_M().EndTime_M'
data-getvalueName='getDate' data-setvaluenam='setDate' data-valueUpdate='onBlur'
data-role=\"DateTimePicker\" />"
},
{
    field: "Activity_M().Effort_M()",
    title: "Effort",
    iseditable: false,
    format: "" /*to call the .toString()*/,
    width: 100,
    editor: "<input data-bind='value:Activity_M().Effort_M'
data-role=\"DurationPicker\" />"
},
{
    field: "Activity_M().ProgressPercent_M()",
    title: "ProgressPercent",
    width: 100,
    iseditable: false,
    editor: "<input data-bind='value:Activity_M().ProgressPercent_M'
data-role=\"spinner\" data-options='{\"min\":0, \"max\": 100}' />"
},
{
    field: "Activity_M().Assignments_M()",
    title: "Resource",
    iseditable: false,
    editor: "<input
data-bind='ResourcePickerBinder:Activity_M().Assignments_M' />",
    template: '<div> ${
RadiantQ.Gantt.ValueConverters.ConverterUtils.GetResourcesText(data.Activity_M().Assi
gnments_M(), false) } </div>',
    width: 100
}

```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Global Selective ReadOnly settings

Selective ReadOnly settings

a) StartTimes are ReadOnly:

To prevent the end user from changing the start time of tasks in the Gantt Chart and Table, use these settings:

```

var columns = [
    {
        field: "Activity_M().ID_M()",
        title: "ID",
        width: 25,
        iseditable: false
    },
    {
        field: "Activity_M().ActivityName_M()",
        title: "Activity Name",
        width: 200,
        editor: RadiantQ.Default.Template.ProjectGanttExpandableTextboxEditor(),
        template:
RadiantQ.Default.Template.ProjectGanttExpandableTextBlockTemplate()
    },
    {
        field: "Activity_M().StartTime_M()",
        title: "StartTime",
        width: 150,
        format: "MM/dd/yy",
        editor: "<input data-bind='ActivityTimeBinder:Activity_M().StartTime_M'
/>"
    },
    {
        field: "Activity_M().EndTime_M()",
        title: "EndTime",
        width: 150,
        format: "MM/dd/yy",
        editor: "<input data-bind='value:Activity_M().EndTime_M'
data-getvalueName='getDate' data-setvalueName='setDate' data-valueUpdate='onBlur'
data-role=\"DateTimePicker\" />"
    },
    {
        field: "Activity_M().Effort_M()",
        title: "Effort",
        format: "" /*to call the .toString()*/,
        width: 100,
        editor: "<input data-bind='value:Activity_M().Effort_M'
data-role=\"DurationPicker\" />"
    },
    {
        field: "Activity_M().ProgressPercent_M()",
        title: "ProgressPercent",
        width: 100,
        editor: "<input data-bind='value:Activity_M().ProgressPercent_M'
data-role=\"spinner\" data-options='{\"min\":0, \"max\": 100}' />"
    },
    {
        field: "Activity_M().Assignments_M()",
        title: "Resource",
        editor: "<input
data-bind='ResourcePickerBinder:Activity_M().Assignments_M' />",
        template: '<div> ${
RadiantQ.Gantt.ValueConverters.ConverterUtils.GetResourcesText(data.Activity_M().Assi
gnments_M(), false) } </div>',
        width: 100
    },
    {
        field: "Activity.PredecessorIndexString",
        title: "PredecessorIndex",
        isParentEditable: false,
        template: "<div>${data.PredecessorIndexString || '' }</div>",
        editor: "<input data-bind='value:PredecessorIndexString' />",
        width: 150
    }
    ];
$ganttt_container.GanttControl('option', 'IsStartTimesReadOnly', true );

```

b) Efforts are ReadOnly:

To prevent the end user from changing the required effort for tasks in the Gantt Chart and Table, use these settings:

```

var columns = [
    {
        field: "Activity_M().ID_M()",
        title: "ID",
        width: 25,
        iseditable: false
    },
    {
        field: "Activity_M().ActivityName_M()",
        title: "Activity Name",
        width: 200,
        editor: RadiantQ.Default.Template.ProjectGanttExpandableTextboxEditor(),
        template:
RadiantQ.Default.Template.ProjectGanttExpandableTextBlockTemplate()
    },
    {
        field: "Activity_M().StartTime_M()",
        title: "StartTime",
        width: 150,
        format: "MM/dd/yy",
        editor: "<input data-bind='ActivityTimeBinder:Activity_M().StartTime_M'
/>"
    },
    {
        field: "Activity_M().EndTime_M()",
        title: "EndTime",
        width: 150,
        format: "MM/dd/yy",
        editor: "<input data-bind='value:Activity_M().EndTime_M'
data-getvalueName='getDate' data-setvalueName='setDate' data-valueUpdate='onBlur'
data-role=\"DateTimePicker\" />"
    },
    {
        field: "Activity_M().Effort_M()",
        title: "Effort",
        format: "" /*to call the .toString()*/,
        width: 100,
        editor: "<input data-bind='value:Activity_M().Effort_M'
data-role=\"DurationPicker\" />"
    },
    {
        field: "Activity_M().ProgressPercent_M()",
        title: "ProgressPercent",
        width: 100,
        editor: "<input data-bind='value:Activity_M().ProgressPercent_M'
data-role=\"spinner\" data-options='{\"min\":0, \"max\": 100}' />"
    },
    {
        field: "Activity_M().Assignments_M()",
        title: "Resource",
        editor: "<input
data-bind='ResourcePickerBinder:Activity_M().Assignments_M' />",
        template: '<div> ${
RadiantQ.Gantt.ValueConverters.ConverterUtils.GetResourcesText(data.Activity_M().Assi
gnments_M(), false) } </div>',
        width: 100
    },
    {
        field: "Activity.PredecessorIndexString",
        title: "PredecessorIndex",
        isParentEditable: false,
        template: "<div>${data.PredecessorIndexString || '' }</div>",
        editor: "<input data-bind='value:PredecessorIndexString' />",
        width: 150
    }
    ];
$ganttt_container.GanttControl('option', 'IsEffortReadOnly', true);

```

c) Dependencies are ReadOnly:

To prevent the end user from changing the dependencies between tasks in the Gantt Chart and Table, use these settings:

```

var columns = [
    {
        field: "Activity_M().ID_M()",
        title: "ID",
        width: 25,
        iseditable: false
    },
    {
        field: "Activity_M().ActivityName_M()",
        title: "Activity Name",
        width: 200,
        editor: RadiantQ.Default.Template.ProjectGanttExpandableTextboxEditor(),
        template:
RadiantQ.Default.Template.ProjectGanttExpandableTextBlockTemplate()
    },
    {
        field: "Activity_M().StartTime_M()",
        title: "StartTime",
        width: 150,
        format: "MM/dd/yy",
        editor: "<input data-bind='ActivityTimeBinder:Activity_M().StartTime_M'
/>"
    },
    {
        field: "Activity_M().EndTime_M()",
        title: "EndTime",
        width: 150,
        format: "MM/dd/yy",
        editor: "<input data-bind='value:Activity_M().EndTime_M'
data-getvalueName='getDate' data-setvalueName='setDate' data-valueUpdate='onBlur'
data-role=\"DateTimePicker\" />"
    },
    {
        field: "Activity_M().Effort_M()",
        title: "Effort",
        format: "" /*to call the .toString()*/,
        width: 100,
        editor: "<input data-bind='value:Activity_M().Effort_M'
data-role=\"DurationPicker\" />"
    },
    {
        field: "Activity_M().ProgressPercent_M()",
        title: "ProgressPercent",
        width: 100,
        editor: "<input data-bind='value:Activity_M().ProgressPercent_M'
data-role=\"spinner\" data-options='{\"min\":0, \"max\": 100}' />"
    },
    {
        field: "Activity_M().Assignments_M()",
        title: "Resource",
        editor: "<input
data-bind='ResourcePickerBinder:Activity_M().Assignments_M' />",
        template: '<div> ${
RadiantQ.Gantt.ValueConverters.ConverterUtils.GetResourcesText(data.Activity_M().Assi
gnments_M(), false) } </div>',
        width: 100
    },
    {
        field: "Activity.PredecessorIndexString",
        title: "PredecessorIndex",
        isParentEditable: false,
        template: "<div>${data.PredecessorIndexString || '' }</div>",
        editor: "<input data-bind='value:PredecessorIndexString' />",
        width: 150
    }
    ];

$ganttt_container.GanttControl('option', 'IsDependencyLinesReadOnly', true);

```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Task specific ReadOnly settings

Task specific ReadOnly settings

You can also make certain tasks read-only by binding to some property values of the task. For example, you can make the tasks that are 100% complete, read-only as follows:

```
var $gantt_container = $('#gantt_container');
$gantt_container.GanttControl({
  ProjectStartDate: anchorTime,
  DataSource: source,
  // Don't allow dependency connections or time-editing of tasks that are already
  // 100% completed.
  IsTaskReadOnlyBinding: {
    Property: "activity.ProgressPercent", Converter: function (value, src,
target) {
      return (value > 99);
    }
  },
});
```

GanttTable Customization

RadiantQ jQuery Gantt Package

Enable Row Drag and Drop

Enabling Row Drag and Drop

Row drag and drop in the table view is disabled by default in the Gantt. You can however enable/disable this feature by setting the gantt's `CanUserReorderRows` option to `true/false`. When enabled, by moving the mouse over the row header will provide you a drag cue using which you can drag one or more selected rows:



ID	Name	StartTime
1	Task 1	2/3/2014
2	Task 2	2/4/2014
3	▼Task 3	2/3/2014
4	Child Task 1	2/3/2014
5	▼Child Task 2	2/5/2014
6	Grand Child Task 1	2/5/2014
7	Grand Child Task 2	2/5/2014
8	Child Task 3	2/3/2014
9	Task 4	2/3/2014

Drag cue to help drag selected rows

You can then move the selected row(s) to a new position, in between other tasks. However, you can also drop them as children of another task with the below setting.

Drop as Child

By default, the gantt lets you drop the selected tasks above or below another existing task like this:

ID	Name	StartTime
1	Task 1	2/3/2014
2	Task 2	2/4/2014
3	▼Task 3	2/3/2014
4	Child Task 1	2/3/2014
5	▼Child Task 2	2/5/2014
6	Grand Child Task 1	2/5/2014
7	Grand Child Task 2	2/5/2014
8	Child Task 3	2/3/2014

Dragging a task next to another task

However, you can optionally also allow dropping a set of tasks as a child of another task when the mouse is over the middle of this destination task. You can turn this feature on with the `EnableDropAsChild` property:

```
var $gantt_container = $('#gantt_container');
    $gantt_container.GanttControl({
        CanUserReorderRows: true,
        EnableDropAsChild: true,
        .....
        .....
        GanttChartTemplateApplied: function (sender , args) {
            .....
        }
    });
```

ID	Name	StartTime
1	Task 1	2/3/2014
2	Task 2	2/4/2014
3	▼Task 3	2/3/2014
4	Child Task 1	2/3/2014
5	▼Child Task 2	2/5/2014
6	Grand Child Task 1	2/5/2014
7	Grand Child Task 2	2/5/2014
8	Child Task 3	2/3/2014

Dropping a task into another task to make it a child of that task.

Drag Events

Before dragging starts, the `GanttControl.BeforeRowsDragStart` event is raised to allow you to prevent dragging for specific rows. The arguments for this event allow you to cancel some row-drags selectively:

```

var ganttcontrol = $gantt_container.data('GanttControl');
ganttcontrol.BeforeRowsDragStart.subscribe(function (args) {
    var rowIndex = args.DragRowIndex;
    if (rowIndex == 0) {
        // Can it should be drag? If not, set args.Cancel = true;
    }

    var dragTask = ganttControl.ActivityViews[rowIndex].Activity;
    if (dragTask._name == 'Task 3') {
        // Can it should be drag? If not, set args.Cancel = true;
    }
});

```

While dragging, the `GanttControl.SelectedRowsDrag` event is raised to allow you to selectively disallow dragging tasks into some locations. The arguments for this event allow you to cancel some drop positions selectively:

```

var ganttcontrol = $gantt_container.data('GanttControl');
ganttcontrol.SelectedRowsDrag.subscribe(function (args) {
    var dragOverRowIndex = args.DragOverRowIndex;
    // Note that "SelectedItem" will give you the parent of Activity.
    if (dragOverRowIndex == 0) {
        // Can it dropped way on top? If not, set args.Cancel = true;
    }
    else if (dragOverRowIndex == ganttcontrol.ActivitiesViews.length) {
        // can the selected task be dropped way at the bottom? If not,
set args.Cancel = true;
    }
    else {
        var dropTask = ganttControl.ActivityViews[dragOverRowIndex];
        // Can the selected task be dropped above this?

        // To prevent dragging tasks at the top level.
        if (selectedTask.IndentLevel == 0)
            args.Cancel = true;
    }
});

```

A sample that illustrates this feature can be located here in the install:

```

In HTML           : ..\Samples\GanttControlCustomDataBinding.htm.
In ASP.NET MVC    : ..\Views\Home\ProjectGantt\GanttControlCustomDataBinding.cshtml.
In ASP.NET        : ..\Samples\ProjectGantt\GanttControlCustomDataBinding.aspx.

```

© RadiantQ 2009-2018. All Rights Reserved.

RadiantQ jQuery Gantt Package

Enable Single Click Edit Option.

Enabling edit mode in a single click

There is a "enableSingleClickEdit" option in "GanttTableOptions" which allow you to enable the editing on a single click. The default value is false.

Here is the code example:

```
$('#gantt_container').GanttControl({  
  //To edit grid cell in single click.  
  GanttTableOptions = {  
    enableSingleClickEdit: true  
  }  
});
```

RadiantQ jQuery Gantt Package

Enable GanttTable HeaderMenu

Enabling the Headermenu option in GanttTable

"EnableHeaderMenu" option is used to display the header menu options in the Column header. By default "EnableHeaderMenu " is false, You can however enable this feature by setting EnableHeaderMenu option to true.

AllowColumnFreezing:

Set "AllowColumnFreezing" option in the GanttTableOptions as false to hide the column lock/unlock option. By default, this option is true.

HeaderMenuMode:

User can change the mode of header menu by setting "HeaderMenuMode":ContextMenu option in GanttTableOptions. To set the header menu mode option,EnableHeaderMenu option should be set as true. By default HeaderMenuMode will be "IconClick".

Here is the code example:

```
$('#gantt_container').GanttControl({
  GanttTableOptions: {
    EnableHeaderMenu: true,
    HeaderMenuMode: "ContextMenu",
    AllowColumnFreezing: false
  }
});
```

RadiantQ jQuery Gantt Package

Handling the editing feature of GanttTable

GanttTable has an option to handle the editing the feature of GanttTable using the "startEdit" and "endEdit" options.

startEdit

To let decide whether the editing should begin or not at the time of edit begins. It will be useful when you prevent the editing for specific row/cell value based on your own conditions. This option get trigger when the user start editing the table cell values and return false to prevent the editing.

Here is the code example:

```
$('#gantt_container').GanttControl({
  GanttTableOptions: {
    startEdit: function (cell, dataItem, column) {
      // Preventing the Recurring Tasks row from edit.
      if (dataItem.Activity_M().DataSource_M().RecurringInfo_M() != null)
        return false;
    }
  },
});
```

endEdit

To let decide whether the changed value should submit or not after the editing is done. It will be useful when you need to handle the value before submitting, and cancelling the submit process based on some validations. This option get trigger when the user submitting the edited cell value and return false to prevent the submitting.

Here is the code example:

```
$('#gantt_container').GanttControl({
  GanttTableOptions: {
    endEdit: function (cell, dataItem, column) {
      // Preventing the Recurring Tasks row from edit.
      if (dataItem.Activity_M().DataSource_M().RecurringInfo_M() != null)
        return false;
    }
  },
});
```

This is illustrated in this samples:

In HTML : ..\Samples\RecurringTasks.htm.

-0-

Gantt Chart Customization

RadiantQ jQuery Gantt Package

Overriding Dependency Setup Behavior

By default when the user connects 2 tasks a "Finish to Start" dependency will be setup. You can override this behavior to change the dependency type or simply cancel/prevent dependency between specific tasks by providing a `NewDependencyValidator` method as follows:

```
var $gantt_container = $('#gantt_container');
$gantt_container.GanttControl({
    DataSource: self.jsonData,
});

var GanttControl = $gantt_container.data("GanttControl");
GanttControl.Model.NewDependencyValidator = NewDependencyValidator;
```

Then implement the above method as follows:

```
function NewDependencyValidator(/*IActivity*/ from, /*IActivity*/ to,
/*NewDependencyAddScenarioType*/ addType) {
    // Setup a workaround to change default dependency types to FF rather than FS.
    if (addType == RadiantQ.Gantt.Model.NewDependencyAddScenarioType.UserDragDrop) {
        // If the user has connected 2 tasks in the Chart to setup dependency, first
        // cancel that op, by returning false below....
        setTimeout(function () {
            // .... and instead create a new FF dependency here.
            GanttControl.Model.CreateNewDependency(from, to,
            RadiantQ.Gantt.DependencyType.FinishToFinish, RQTimeSpan.Zero);
        }, 0);

        return false;
    }
    else {
        // All other scenarios, simply return true to allow normal processing.
        return true;
    }
}
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

Appearance Customization

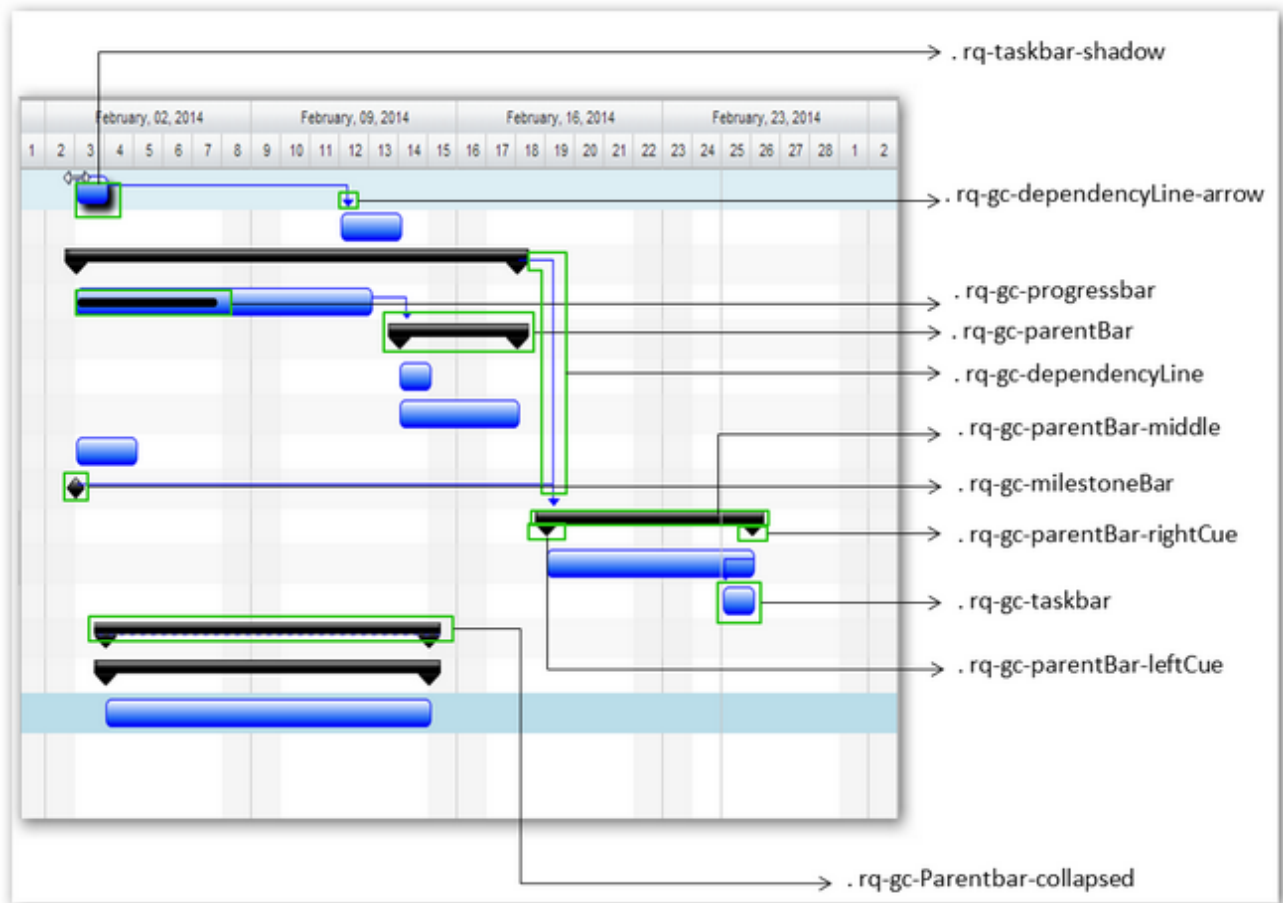
RadiantQ jQuery Gantt Package

Gantt CSS Styles

Styles common to both GanttControl and FlexyGantt are discussed in this [topic](#).

Here we discuss some styles that are specific to the GanttControl.

Gantt Control Specific Styles



© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package ContextMenus

The GanttControl employs several built-in context menus for different portions of the gantt as discussed below.

But, to begin with you have a couple of options to select a context menu framework that the gantt should work with. See this [topic](#) for more information on this.

GanttControl ContextMenus

The gantt provides an abstract API through which you customize the context menu items to be shown in the gantt, no matter which of the above scenarios applies to you.

You can customize the context menus by adding, removing items in the context menu.

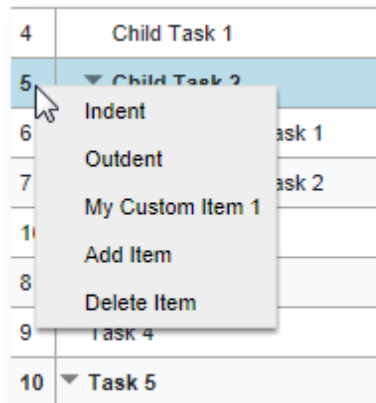
The following context menus are built-in:

- Gantt Table Context Menu
- Gantt Chart Task Bar Context Menu
- Gantt Chart DependencyLine Context Menu

(TheSamples\GanttCustomizingMenus sample illustrates the below features.)

Gantt Table Context Menu

This context menu is shown when right clicking on the table row. As seen in the screen-shot, there are built-in items like "Indent" and "Outdent" which you can augment with custom items like "My Custom Item 1" , "Add Item" and "Delete Item".



Built-in context menu with a custom item

You can add custom items like below.

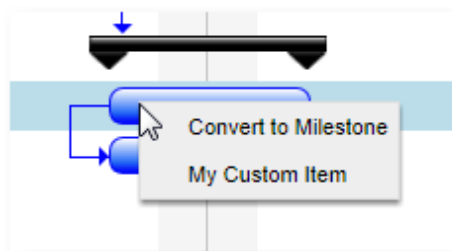
```

var tableContextMenu = $('#ganttControl').data("GanttControl").TableContextMenu;
var ganttControl = $ganttControl.data("GanttControl");
var tableMenuItems = [
    { keyName: "MyCustomItem", name: "My Custom Item 1", icon: "My
Custom Item 1", callback: function (key, opt){
        var dataContext = grid.GetDataFromRow(opt.$trigger[0]);
        alert("TaskInfo context in Table: TaskName: " +
dataContext.Activity_M().ActivityName_M() + ", StartTime: " +
dataContext.Activity_M().StartTime_M());
    }
},
    { keyName: "AddItem", name: "Add Item", icon: "Add Item",
callback: function (key, opt) {
        var newRow = {
            "Name": "New Task",
            "ID": ganttControl.Model.GetNewID(),
            "StartTime": anchorTime.clone(),
            "Effort": new RQTimeSpan(0, 12, 30, 0, 0),
            "ProgressPercent": 90,
            "Cost": 100,
            "Description": "Description of Task"
        };
        ganttControl.InsertNewItemAsSiblingBelow(newRow,
ganttControl.SelectedIndex, true);
    }
},
    { keyName: "DeleteItem", name: "Delete Item", icon: "Delete
Item", callback: function (key, opt) {
        var selectedItem = ganttControl.SelectedItem;
        ganttControl.RemoveActivity(selectedItem.Activity.ID);
    }
}
]
//Passing "false" as an second argument will disable the default contextMenu items.
Default menu items are always enabled.
tableContextMenu.AddNewItems(tableMenuItems);

```

Gantt Chart Task Bar Context Menu

The context menu shown when right-clicking on the task bar has by default a "Convert to Milestone" item. You can add custom items to the menu as shown in the screen-shot.



The above mentioned CustomizingMenus sample illustrates how this can be implemented.

Here is the code snippet taken from the above sample to add a custom item.

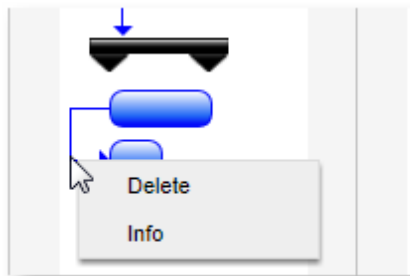
```

var taskContextMenu = $('#ganttControl').data("GanttControl").TaskContextMenu;
taskMenuItems = [
    { name: "My Custom Item", icon: "My Custom Item", callback:
function (key, opt) {
    var dataContext = opt.$trigger[0].parentElement["data-grid-item"];
    alert("TaskInfo context in Table: TaskName: " +
dataContext.Activity.ActivityName + ", StartTime: " +
dataContext.Activity.StartTime);
    }
}
];
//Passing "false" as an second argument will disable the default contextMenu items.
Default menu items are always enabled.
taskContextMenu.AddNewItems(taskMenuItems);

```

Gantt Chart Dependency Line Context Menus

Right clicking on the dependency lines show this context menu which will allow you to delete it. This can be customized along the same lines.



Here is the code snippet.

```

var dependencyContextMenu = $('#ganttControl').data("GanttControl")
).DependencyContextMenu;
dependencyMenuItems = [
    { keyName: "Info", name: "Info", icon: "Info", callback:
function (key, opt) {
    var dataContext =
$(opt.$trigger[0].parentElement).data().GanttDependencyLine.options.DependencyView;
    alert("From Activity : " +
dataContext.StartActivity.ActivityName + "," + dataContext.StartTime + "
    To Activity" + dataContext.EndActivity.ActivityName + "," +
dataContext.EndTime);
    }
}
];
//Passing "false" as an second argument will disable the default contextMenu items.
Default menu items are always enabled.
dependencyContextMenu.AddNewItems(dependencyMenuItems , false);

```

Chart Look and Feel

Task Bar Look and Feel

RadiantQ jQuery Gantt Package

Custom Look and Feel

Task Bar Look and Feel

By default task bars are in blue gradient as shown below. This default look changes if you choose one of our built-in themes, though.

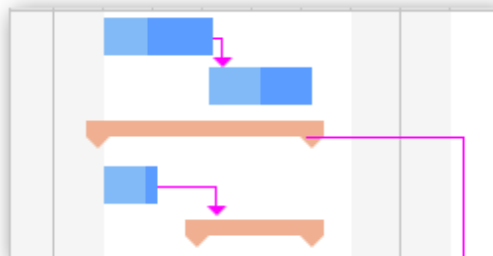


Default Task Bar

Customizing Task Bar Look Using Custom Templates

You can also fully customize the look and feel of task bars by just overriding the taskbars style. Below is the css code in html

```
<style type="text/css">
  .rq-gc-taskbar
  {
    background-image: none;
    background-color: #3796D2;
    border: 1px solid black;
    border-radius: 0px;
  }
  .rq-gc-progressbar
  {
    border-radius: 0px;
    background-color: #47E260;
    margin: -1px 0px 0px -1px;
    height: 100%;
  }
</style>
```



Templatized TaskBar

TheSamples\TemplatizedTaskBar sample illustrates how to do this.

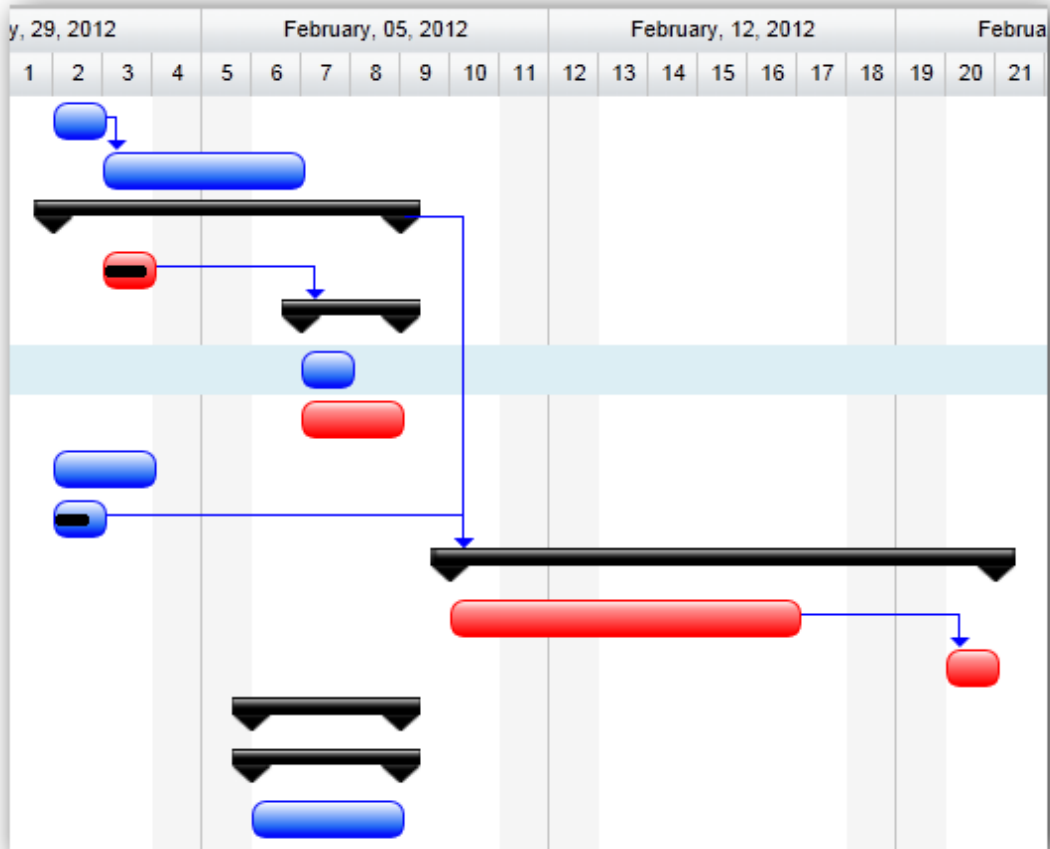
-0-

RadiantQ jQuery Gantt Package

Custom Look for Specific Task Bars

Customizing a specific Task Bar's Look

Sometimes you have to customize the look and feel of individual task bars based on some value in the bound object instance representing each task. The [Critical Paths](#) topic will explain, for example, how to customize taskbar look and feel based on the Critical Paths in the project.



RadiantQ jQuery Gantt Package

Custom UI on Task Bars

By default task bars has the progress bar UI. The task bar can be customize by jQuery templates, to render the templates you have to set that to GanttControl.Options.TaskItemTemplate.

Here is the example that shows how to render the time spend bars in task bar.

Sample JSON.

```
[
  {
    "Name": "Task 1",
    "ID": 1,
    "StartTime": "2014-02-02T00:00:00Z",
    "Effort": "08:00:00",
    "Description": "Description of Task 1",
    "TimeSpent": "08:00:00"
  },
  {
    "Name": "Task 2",
    "ID": 2,
    "PredecessorIndices": "1",
    "StartTime": "2014-02-03T00:00:00Z",
    "Effort": "16:00:00",
    "Description": "Description of Task 2",
    "TimeSpent": "08:00:00"
  }
]
```

CSS

```
.timespent-bar {
  margin-top: 6px;
  height: 5px !important;
  z-index: 11 !important;
  position: relative;
  border-radius: 7px !important;
  background-color: red !important;
  border-color: red !important;
  -ms-touch-action: none;
  float: left;
}
```

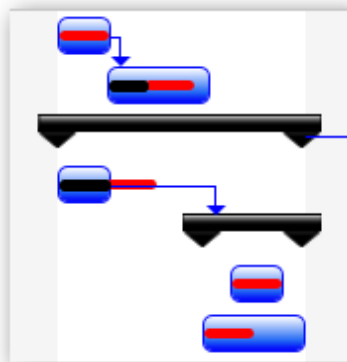
JS Code

```

//Task bar template which has the time spend bar
var tTemplate = "<div class='rq-gc-taskbar'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='start-resizeThumb'></div><div class='timespent-bar'></div></div>";
var $gantt_container = $('#gantt_container');
$gantt_container.GanttControl({
  ProjectStartDate: anchorTime,
  DataSource: self.jsonData,
  IDBinding: new RadiantQ.BindingOptions("ID"),
  NameBinding: new RadiantQ.BindingOptions("Name"),
  IndentLevelBinding: new RadiantQ.BindingOptions("IndentLevel"),
  StartTimeBinding: new RadiantQ.BindingOptions("StartTime"),
  EffortBinding: new RadiantQ.BindingOptions("Effort"),
  PredecessorIndicesBinding: new RadiantQ.BindingOptions("PredecessorIndices"),
  ProgressPercentBinding: new RadiantQ.BindingOptions("ProgressPercent"),
  DescriptionBinding: new RadiantQ.BindingOptions("Description"),
  TaskItemTemplate: tTemplate,
  OnTaskBarLoad: function (taskbarInstance) { //Time spend bar logic.
    var ganttChart = taskbarInstance.GanttChart;
    //To get hold of the data source.
    var DataSource = taskbarInstance.ActivityView.Activity.DataSource;
    //ProgressEndTime
    var progressEnd = taskbarInstance.ActivityView.Activity.ProgressEndTime;
    var timeSpend = DataSource.TimeSpend;
    var taskbarInstancestart = taskbarInstance.ActivityView.Activity.StartTime;
    var timeSpendStart = progressEnd;
    var timeSpendEnd =
taskbarInstance.options.WorkTimeSchedule.GetEnd(progressEnd, timeSpend);
    var timeSpendStartX = ganttChart.ConvertTimeToX(progressEnd);
    var timeSpendBarWidth =
TimeComputingUtils.ConvertToUnitsOfBaseScaleType(taskbarInstance.GanttChart.options.B
aseTimeScaleType, progressEnd, timeSpendEnd,
taskbarInstance.GanttChart.options.WorkTimeSchedule) *
taskbarInstance.GanttChart.options.BaseTimeUnitWidth;

    $(this).find("div.timespent-bar").css("width", timeSpendBarWidth + "px");
    $(this).find("div.timespent-bar").css("margin-left",
taskbarInstance.ProgressBarRectangle.width() + "px");
  }
});

```



Custom UI On Task Bars

*RadiantQ jQuery Gantt Package***TaskBarBackgroundTemplate****TaskBarBackgroundTemplate**

It's common to customize the look and feel of a task bar with custom task backgrounds to visualize more information available in the bound objects.

For example, there could be a base line start and end properties in your bound list elements that you want to visualize in the gantt chart just so you know which tasks went past-due.

You might want to visualize such tasks that went past-due with a "red bar" indicating the planned end time for such tasks.

To achieve this first define a TaskBarBackgroundTemplate.

This property in the GanttControl lets you define the UI for your custom task background.

In HTML

```
// Custom template to indicate tasks that went past due
// Note that a GanttTaskItemBar widget options will be the DataContext for all those
// UI elements,
// So, you can refer to it's properties in the binding code below,
// 1) To refer to a property in the IActivity instance, the binding expression will
// look like this:
//    // "${ActivityView.Activity.StartTime}"
// 2) The refer to a property in the underlying data source, the binding expression
// will look like this:
//    // "${ActivityView.Activity.DataSource.PlannedEndTime}"
var taskBarBGTemplate = '<div class="baseline-style" style="width: ${widthConverter(
data)); margin:${leftConverter(data)}"
title="${taskBarBGTplTooltip(data)}"></div>';

$ganttControl.GanttControl({
  ProjectStartDate: anchorTime,
  DataSource: self.jsonData,
  AnchorTime: anchorTime,
  TaskBarBackgroundTemplate: baseLineTemplate
});
```

In ASP.NET MVC

```

@Html.JQProjectGantt(
    new JQProjectGanttSettings()
    {
        ControlId = "ganttt_container",
        AfterGanttWidgetInitializedCallback = "AfterGanttWidgetInitializedCallback",
        DataSourceUrl = new Uri("/Home/GanttControlItemSource", UriKind
.RelativeOrAbsolute),
        Options = new ProjectGanttOptions()
        {
            IDBinding = new Binding("ID"),
            NameBinding = new Binding("Name"),
            IndentLevelBinding = new Binding("IndentLevel"),
            StartTimeBinding = new Binding("StartTime"),
            EffortBinding = new Binding("Effort"),
            PredecessorIndicesBinding = new Binding("PredecessorIndices"),
            ProgressPercentBinding = new Binding("ProgressPercent"),
            DescriptionBinding = new Binding("Description"),
            TaskBarBackgroundTemplate = "<div class='baseline-style' style='width:
${widthConverter(data)}; margin:${leftConverter(data)}'
title='${taskBarBGTplTooltip(data)}'></div>",
            GanttChartOptions = new GanttChartOptions()
            {
                AnchorTime = DateTime.Today
            }
        }
    }
)

```

In ASP.NET

```

<RQ:GanttControl ID="ganttt" DataSourceUrl="../../TaskListHandler.ashx"
    TaskBarBackgroundTemplate="<div class='baseline-style' style='width: ${
widthConverter(data)}; margin:${leftConverter(data)}' title='${ taskBarBGTplTooltip(
data)}'></div>"
    AfterGanttWidgetInitializedCallback="this.AfterGanttWidgetInitializedCallback"
    LocalizationResourceFilePath="../../Src/ResourceStrings/"
    Height="500px" runat="server" />

```

And the leftConverter and widthConverter would look like this:

```

// Calculating the width of the TaskBarBackgroundTemplate
function widthConverter(data) {
    var ganttChart = data.GanttChart;
        var DataSource = data.ActivityView.Activity_M().DataSource_M();
        var plannedStart = DataSource.PlannedStartTime;
        var plannedEnd = DataSource.PlannedEndTime;

        // Use this utility in GanttChart to determine the location of the past
due bar.
        var rightX = plannedEnd ? ganttChart.ConvertTimeToX(plannedEnd) : 0;

        var leftX = plannedStart ? ganttChart.ConvertTimeToX(plannedStart) : 0;

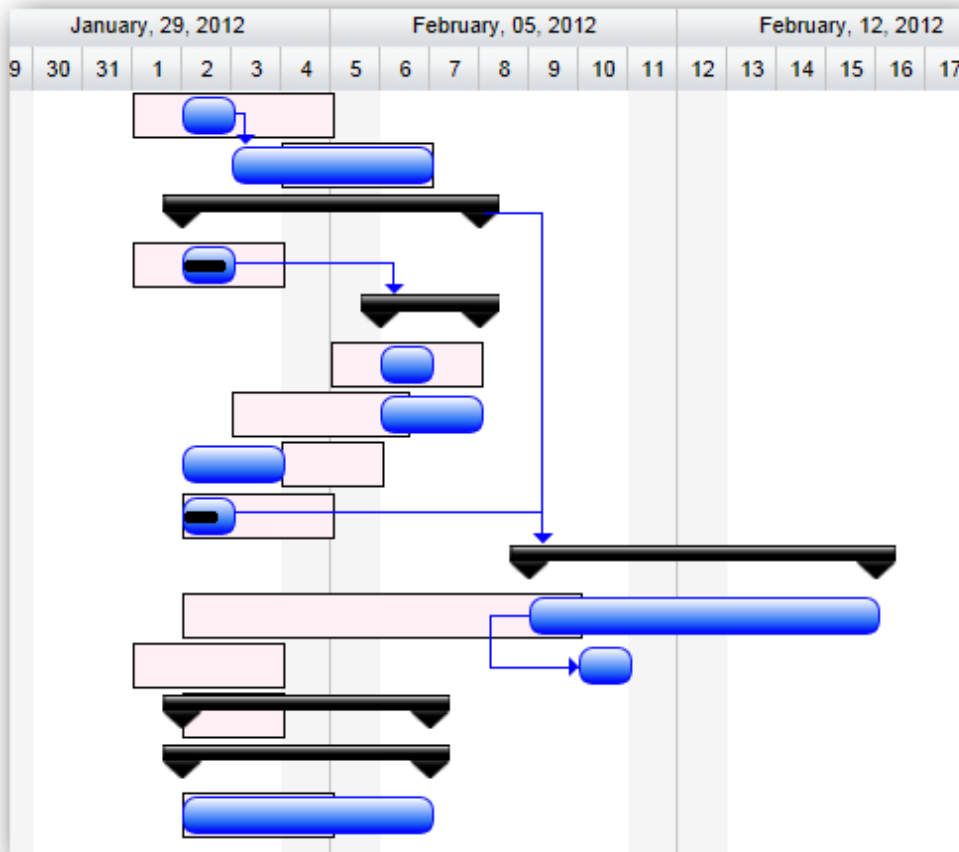
        return (rightX - leftX) + "px";
}

// Calculating the left margin for TaskBarBackgroundTemplate
function leftConverter(data) {
    var ganttChart = data.GanttChart;
    var plannedStart = data.ActivityView.Activity.DataSource.PlannedStartTime;
    // Return the setting for the margin.
    return "1px 0px 1px " + (plannedStart ? ganttChart.ConvertTimeToX(plannedStart) :
0) + "px";
}

// Using the bound data in tooltip for TaskBarBackgroundTemplate
function taskBarBGTplTooltip(data) {
    data.rendered = function () {
        $(this.nodes).tooltip({
            content: function (val) {
                var toolTipDateFormat = 'dd-MMM-yyyy';
                var ds = data("tplItem"
).data.ActivityView.Activity_M().DataSource_M();
                var PStartTime =
ds.PlannedStartTime.toString(toolTipDateFormat);
                var PEndTime = ds.PlannedEndTime.toString(toolTipDateFormat);
                return "<div align='center'><span
style='font-weight:bold'>BaseLine</span></div><div><span
style='font-weight:bold'>Start:</span> " + PStartTime + "</div><div><span
style='font-weight:bold'>End:</span> " + PEndTime + "</div>";
            }
        });
    }
    return "";
}
}

```

The above settings will result in the gantt tasks rendered with baseline bars.



Baseline indicated using background template

This is illustrated in this samples:

- In HTML : ..\Samples\GanttControlTemplateSample1.htm.
- In ASP.NET MVC : ..\Views\Home\ProjectGantt\GanttControlTemplateSample1.cshtml.
- In ASP.NET : ..\Samples\ProjectGantt\GanttControlTemplateSample1.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

RadiantQ jQuery Gantt Package

Task Bar Labels

Task Bar Labels

You also have an option to show some text to the right side of the taskbar using [jQuery-template](#) , below code shows how to do this. See how a custom "tTemplate" is defined.

In HTML

```
var tTemplate = "<div data-bind=\"attr: { 'class': RQDataContext.ProgressPercent()  
> 50 ? 'rq-gc-taskbar bluebar-style' : 'rq-gc-taskbar greenbar_style'}\" ><div  
data-bind=\"text: RQDataContext.Description()\"  
class='rq-gc-taskbar-label'></div></div>\"";  
  
var $gantt_container = $('#gantt_container');  
$gantt_container.GanttControl({  
  
    DataSource: viewModel.Tasks(),  
    TaskItemTemplate: tTemplate,  
    KnockoutObjectName: "viewModel",  
    GanttTableOptions: {  
        columns: columns,  
    },  
    GanttChartTemplateApplied: function (sender , args) {  
        ...  
    }  
});
```

In ASP.NET MVC

```

@Html.JQProjectGantt(
    new JQProjectGanttSettings()
    {
        ControlId = "ganttt_container",
        AfterDataRetrievedCallback = "AfterDataRetrievedCallback",
        AfterGanttWidgetInitializedCallback = "AfterGanttWidgetInitializedCallback",
        DataSourceUrl = new Uri("/Home/GanttControlItemSource", UriKind
.RelativeOrAbsolute),
        Options = new ProjectGanttOptions()
        {
            IDBinding = new Binding("ID"),
            NameBinding = new Binding("Name"),
            IndentLevelBinding = new Binding("IndentLevel"),
            StartTimeBinding = new Binding("StartTime"),
            EffortBinding = new Binding("Effort"),
            PredecessorIndicesBinding = new Binding("PredecessorIndices"),
            ProgressPercentBinding = new Binding("ProgressPercent"),
            KnockoutObjectName = "viewModel",
            GanttTableOptions = {
                columns: columns,
            },
            // KO Templates are used here to color the bars differently based on the
            progress value. KO will trigger this binding every time the bound value changes and
            affect the look appropriately.
            TaskItemTemplate = "<div data-bind=\"attr: { 'class':
RQDataContext.ProgressPercent() > 50 ? 'rq-gc-taskbar bluebar-style' : 'rq-gc-taskbar
greenbar_style'}\" ><div data-bind=\"text: RQDataContext.Description()\"
class='rq-gc-taskbar-label'></div></div>",
            GanttChartOptions = new GanttChartOptions()
            {
                AnchorTime = DateTime.Today
            }
        }
    })

```

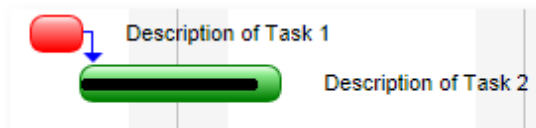
In ASP.NET

```

<script runat="server">
    protected void ganttt_Load(object sender, EventArgs e)
    {
        this.ganttt.TaskItemTemplate = "<div data-bind=\"attr: { 'class':
RQDataContext.ProgressPercent() > 50 ? 'rq-gc-taskbar bluebar-style' : 'rq-gc-taskbar
greenbar_style'}\" ><div data-bind=\"text: RQDataContext.Description()\"
class='rq-gc-taskbar-label'></div></div>";
    }
</script>

<RQ:GanttControl ID="ganttt" OnLoad="ganttt_Load" HierarchyResolverFunction
="ResolverFunction".... runat="server" />

```



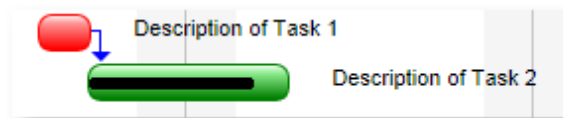
Task Bar with custom Labels on the right.

TheSamples\GanttControlSkeletonUsingKO sample illustrates how to display label in gantt chart.

User can also able to move label to any side of the taskbar by overriding

"rq-gc-taskbar-label" class in html and can set margins and padding. For example if you want to place rq-gc-taskbar-label top-right of taskbar you can do it like this.

```
<style type="text/css">
  .rq-gc-taskbar-label
  {
    padding-top: 0px;
    float: right;
    height: auto;
    width: auto;
  }
</style>
```



© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Task Bar Redraw

Forcing a Redraw of the Bars

Gantt enables you to redraw the whole chart rows by calling "RedrawChartRows" GanttControl method.

The task bar look and feel is defined using templates as shown above. And these templates can also use bindings to customize the look and feel based on the underlying data-object properties. However, the look does not automatically change when the underlying data changes. There are 2 ways to ensure this:

- Using KnockOut. This is illustrated this in [topic](#). This would in fact result in a much more cleaner code with the MVVM approach. However, this adds a level of complexity to your pages.
- Simply redraw the bars whenever a data object value changes. This is what this API allows you to do.

Here is some sample code that illustrates a use case from our GanttControlCriticalPath sample.

```
// Template to show the critical task.
var tTemplate = "<div class='rq-gc-taskbar'
style='background-image:${UpdateBackgroundColorBinding(data)} !important;
border-color:${ UpdateBorderColorBinding(data)} !important'><div
id='GanttTaskBarLabel' class='rq-gc-taskbar-label'></div></div>";

function updateCriticalPaths(sender, e) {
    // The first argument to GetCriticalPathActivities is a time-buffer which
    // speifies the "safe distance"
    // that an activity's end time should be away from affecting the project
    // deadline.
    CriticalPathActivities =
$gantttControl.GetCriticalPathActivities(RQTimeSpan.Zero);
    $gantttControl.RedrawChartRows();
}
function clearCriticalPaths(sender, e) {
    CriticalPathActivities = [];
    $gantttControl.RedrawChartRows();
}

function UpdateBackgroundColorBinding(data) {
    var isCritical = CriticalPathActivities.indexOf(data) != -1;
    // for background-image
    if (isCritical)
        return 'url(Images/redBar.png)';
    return 'url(Src/Styles/Images/TaskBar.png)';
}
function UpdateBorderColorBinding(data) {
    var isCritical = CriticalPathActivities.indexOf(data) != -1;
    // for background-color
    if (isCritical)
        return 'red';
    return '#050DFA';
}
```

The\GanttControlCriticalPath sample illustrates a use case for this method.

Redraw API s.

RadiantQ GanttControl allows you to redraw the single chart row and also enable you to redraw the whole chart rows. Below code will illustrates how to redraw single row and whole rows in chart.

```
$("#RedrawChartRows").click(function () {
    $gantttControl.RedrawChartRows();
});
$("#RedrawChartRow").click(function () {
    var activityview = ganttControl.SelectedItem;
    $gantttControl.RedrawChartRow(activityview);
});
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Task Popups

Task Popup

Popups are shown when the end-user interacts with the task, when they move it, resize it, connect 2 tasks, resize the progress information, etc. While the default content might be good enough and appropriate enough for most scenarios, you can replace these popups with custom popups if necessary.

These are the properties in GanttChart that will allow you do this:

- ResizeInfoPopup
- ProgressResizePopup (used only by GanttControl)
- MovingInfoPopup
- ConnectingInfoPopup (used only by GanttControl)

The class reference for these properties contain information on the DataContext that will be applied on these custom popup templates.

Let us see how to customize the MovingInfoPopup using a template with an example.

(Note that the RadiantQ template library is used internally to convert the template provided to actual dom elements.)

```

<script id="MovingInfoPopup" type="text/x-jquery-tmpl">
  <table class="draggingContent" style="width: 180px;">
    <tr>
      <td colspan='2' align='center'>${RadiantQ_TaskString} </td>
    </tr>
    <tr>
      <td>${RadiantQ_StartString} : ${data.StartTime.toString('yyyy-MM-dd
HH:mm')} </td>
    </tr>
    <tr>
      <td>${RadiantQ_FinishString} : ${data.EndTime.toString('yyyy-MM-dd
HH:mm')} </td>
    </tr>
  </table>
</script>

<script id="ResizeInfoPopup" type="text/x-jquery-tmpl">
  <table class='draggingContent' style='width: 180px;'>
    <tr>
      <td colspan='2' align='center'>Task </td>
    </tr>
    <tr>
      <td style='float: right;'>Duration : </td>
      <td>#if (data.Duration){# #= data.Duration.days_M() #d #=
data.Duration.hours_M() #h #}# </td>
    </tr>
    <tr>
      <td style='float: right;'>StartTime : </td>
      <td>${data.StartTime.toString('yyyy-MM-dd HH:mm')}</td>
    </tr>
    <tr>
      <td style='float: right;'>EndTime :</td>
      <td>${data.EndTime.toString('yyyy-MM-dd HH:mm')}</td>
    </tr>
  </table>
</script>

```

In HTML

```

$('#container').FlexyGantt({
  MovingInfoPopup: $("#MovingInfoPopup").html(),
  ResizeInfoPopup: $("#ResizeInfoPopup").html()
});

```

In ASP.NET MVC

```

@Html.JQFlexyGantt(
  new JQFlexyGanttSettings()
  {
    ..
    Options = new FlexyGanttOptions()
    {
      GanttChartOptions = new GanttChartOptions()
      {
        AnchorTime = DateTime.Today,
        MovingInfoPopupID = "MovingInfoPopup",
        ResizeInfoPopupID = "ResizeInfoPopup",
      }
    }
  }
)

```


In ASP.NET

```

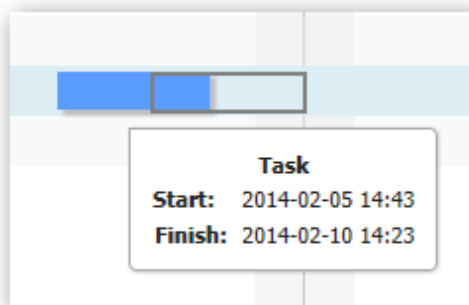
<RQ:FlexyGantt ID="ganttt" AfterDataRetrievedCallback="AfterDataRetrievedCallback"
  AfterGanttWidgetInitializedCallback="AfterGanttWidgetInitializedCallback"
  ParentTaskEndTimeProperty="OverallEndTime"
  ParentTaskStartTimeProperty="OverallStartTime"
  TaskEndTimeProperty="EndTime"
  MovingInfoPopupID="MovingInfoPopup"
  ResizeInfoPopupID="ResizeInfoPopup"
  TaskStartTimeProperty="StartTime"
  HierarchyResolverFunction="ResolverFunction"
  DataSourceUrl="../../DataSources/FGTaskListHandler.ashx"
  LocalizationResourceFilePath="../../Src/ResourceStrings/"
  Height="500" runat="server"
  TaskItemTemplate="<div class='rq-gc-taskbar'></div>"
  ProgressBarTemplate="<div class='rq-gc-progressbar'></div>"

  ParentTaskItemTemplate="<div class='parentBar-style'><div
class='rq-gc-parentBar-leftCue'></div><div class='rq-gc-parentBar-middle'
></div><div class='rq-gc-parentBar-rightCue'></div>"

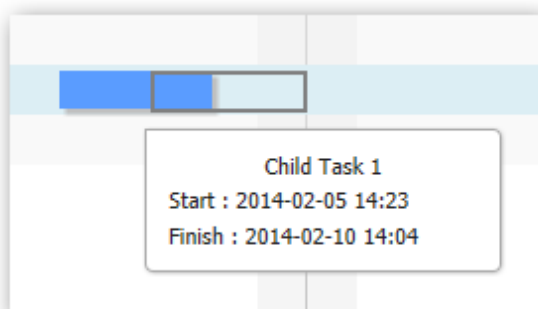
```

This is illustrated in this samples:

In HTML : ..\Samples\TemplatizedTaskBar.htm.
 In ASP.NET MVC : ..\Views\Home\ProjectGantt\TemplatizedTaskBar.cshtml.
 In ASP.NET : ..\Samples\ProjectGantt\TemplatizedTaskBar.aspx.



Built-in MovingInfoPopup



Customized MovingInfoPopup

© RadiantQ 2009-2018. All Rights Reserved.

-0-

Gantt Table Customization

RadiantQ jQuery Gantt Package

GanttTable Custom Column

VWGrid is highly customizable, you can create your own cell template, cell editing template and binders. This topic will show how to add a custom Column to GanttTable(derived form VWGrid).

Here is the example, which shows how to add a Priority column, the Priority value in data source is an integer but GanttTable will show the corresponding status as a string.

```
//priority to status converter.
function priorityTextConverter(data, value) {
    var priority = data.Activity.DataSource.Status.Priority;
    if (priority != undefined) {
        switch (priority.toString()) {
            case "1":
                return "Low";
                break;
            case "2":
                return "Medium";
                break;
            case "3":
                return "High";
                break;
        }
    }
}

//Column definition.
{
    field: "Activity.DataSource.Status.Priority",
    title: "Priority",
    width: 100,
    editor: "<select data-bind='priorityEditor:Activity.DataSource.Status.Priority'
></select>",
    //customized cell template, priorityTextConverter is called for each and every
    //priority cell.
    template: "<div>${priorityTextConverter(data)}</div>"
},
```

Similarly, you can display the status image in a Priority Cue Column.

```
//priority to status image converter
function priorityImageConverter(data, value) {
    var priority = data.Activity.DataSource.Status.Priority;
    if (priority != undefined) {
        switch (priority.toString()) {
            case "1":
                return '';
                break;
            case "2":
                return '';
                break;
            case "3":
                return '';
                break;
        }
    }
}

//Column definition.
{
    field: "Activity.DataSource.Status.Priority",
    title: "Priority Cue",
    width: 100,
    template: "<div>${priorityImageConverter(data)}</div>"
},
```

By default GanttTable listens the property changes and updates the corresponding cell in table. Here, the Priority value changes will reflect in both Priority and Priority Cue column, GanttTable identifies these column by its field value ("Activity.DataSource.Status.Priority" in this case).

Note: Please take a look at the "Samples\GanttControlTableCustomization" sample where these features are illustrated.

RadiantQ jQuery Gantt Package

GanttTable Column Editable Settings

Disable editing For a particular Column.

Set the `iseditable` to `false` in column definition.

```
{
  field: "Activity.DataSource.Cost",
  title: "Cost",
  editor: "<input data-bind='value:Activity.DataSource.Cost'
data-role=\"spinner\" />",
  template: "<div>${ToDollarString(data)}</div>",
  width: 100,
  iseditable: false
},
```





Custom editing.

By default, the VWGrid provides you an input element for editing the value, which can of course be extended with a UI Widget. However, you can also make the VWGrid use a different element for editing the values, using the editor template. This is illustrated in the sample ... Samples/GanttControlTableCustomization.htm where we use a `select` html element to edit the value of the priority column.

```
RadiantQ.Binder.priorityEditor = function () {

  this.init = function($elem, role, value, data)
  {
    var source = ["Low", "Medium", "High"];
    for (var i = 0; i < source.length; i++) {
      $elem.append("<option value=" + (i + 1) + ">" + source[i] + "</option>");
    }
    //To set the value to select.
    $elem.val(value.getter(data));
    //select change event.
    $elem.change(function () {
      //To update the boundobject.
      data.Activity.DataSource.Status.Priority = $(this).val();
    });
  }
}

// Column definition
{
  field: "Activity.DataSource.Status.Priority",
  title: "Priority",
  width: 100,
  isParentEditable: false,
// priorityEditor is a custom binder
  editor: "<select data-bind='priorityEditor:Activity.DataSource.Status.Priority'
></select>",
  template: "<div>${priorityTextConverter(data)}</div>"
}
```

08:00:00	High	
1.00:00:00	Low	
1.00:00:00	Low Medium	
1.00:00:00	High	

Using select html element to edit.

Enable Summary row Column Editable.

By default all Summary columns are not editable other than the name column. To allow editing, set the `isParentEditable` to true in the `fcolumn` definition.

```
var columns = [
  .....
  .....
  {
    field: "Activity.StartTime",
    title: "StartTime",
    width: 150,
    isParentEditable: true,
    format: "MM/dd/yy",
    editor: "<input data-bind='ActivityTimeBinder:Activity.StartTime' />"
  },
  .....
  .....
];
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

GanttTable Alternative Row background

Alternative rows in the GanttTable has the `.rq-grid-alternative-background` class. In this class you can specify the background of the alternative row background color.

CSS style

```
.rq-grid-alternative-background {  
    background-color:#F7F9FB;  
}
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

How Tos

RadiantQ jQuery Gantt Package

How to find a task/activity by it's ID?

How to find a task/activity by it's ID?

```
// Returns the Gantt Model's internal activity/task provided an id.  
var ganttControl = $('#gantt_container').data("GanttControl");  
var activity = ganttControl.Model.GetActivityById(id);
```

And if you want to retrieve the bound data item in your data source, you can do this:

```
// Task is the bound data source item  
var taskSource = activity.DataSource;
```


RadiantQ jQuery Gantt Package

How to find ActivityView by its ID?

How to find a activity view by it's ID?

```
// Returns the Gantt Model's internal activity/task provided an id.  
var ganttControl = $('#gantt_container').data("GanttControl");  
var activityView = ganttControl.ActivityViews.GetActivityViewByID(id);
```

And if you want to retrieve the bound data item in your data source, you can do this:

```
//To get the activity.  
var activity = activityView.Activity;  
// Task is the bound data source item  
var taskSource = activity.DataSource;
```

RadiantQ jQuery Gantt Package

How to listen to changes made by the end-user on the tasks?

There are a few different ways to do this, go through all the options below and choose the right one for you.

- 1) ..\Samples\GanttBoundToXML.htm.
- 2) ..\Samples\TrackingEndTimeInDataSource.htm

Option 1

Listen to the `GanttControl.ActivityTimeChanged` event, which is raised only when the end-user changes the start or end time (duration) of an activity in the `GanttChart`.

```
//Will notify when the task start/end changed.
gantControl.ActivityTimeChanged.subscribe(activity_timeChanged);
function activity_timeChanged(sender, args) {
    if (args.Type == "DragEnd") {
        alert("Task Moved, New Start and End is" + args["StartTime"] + args["EndTime"]);
    }
    else if (args.Type == "ResizeEnd") {
        alert("Task Resized, New End is" + args["EndTime"]);
    }
}
```

Note that if you added the ability to edit the times in the Grid, this event will not be raised when the user edits the times in the Grids.

Option 2

To get a comprehensive notification of all changes happening on an activity, consider listening to the `Model.ActivityUpdated` event. This event will be raised for all the properties in your task object that are bound to the gantt (like Start Time, End Time, Predecessors, Progress, etc.). This will be raised both changes that are initiated from the Chart and in the Grid.

Occasionally, you might be showing some custom data in the Gantt's Grid, in a custom column and if the user edits the values in those columns, the above event will not be raised for that.

Code sample to listen to this event:

```
$gantControl = $gant_container.data("GanttControl");
$gantControl.Model.ActivityUpdated.subscribe(activity_Updated);

function activity_Updated(sender, args) {
    var taskSource = activity.DataSource; // to get hold of the underlying data source.
    // The activity has the latest values edited by the user.
}
```

Note that this event will only call after the `GanttControl` loads. In other words, this will not be raised for changes made to your task objects (typically the times) while the Gantt's Model is getting initialized. While initializing, if the Model sees that the times in your task objects need to be adjusted (to take into account project schedule, dependencies, etc.) it will do so.

Option 3

The best way to keep notified of changes is to listen directly to your bound task object's

property changed events. Java Script objects, of course, don't emit any change notification, however, there are 2 ways to enable them to notify changes:

a) Enable Change Notification through Knockout

If you have applied Knockout on your data objects then they will be notifying changes.

```
function propertyChangeNotifier(value) {
    //To get currently modified task object field name.
    var propName = this.PropertyName;
    //To get the modified task object.
    var task = this.TaskObject;
    // alert("PropertyName : " + propName + "New Value : " + value);
}

ko.utils.arrayForEach(viewModel.Tasks(), function (taskObject) {

    for (var fieldName in taskObject) {
        var field = taskObject[fieldName];
        if (field.subscribe) { //{ PropertyName: fieldName, TaskObject: taskObject }
is the context for the subscribe method
            field.subscribe(propertyChangeNotifier, { PropertyName: fieldName,
TaskObject: taskObject });
        }
    }
});
```

b) Insert Change Notifying code

If you are not using KO, our library comes with a simple utility that will insert change notifying code into your objects and then you can listen to changes in the data. Here is the code that lets you do that:

```
$gantt_container.GanttControl({
    DataSource: self.jsonData,
    // This will make the bound objects trigger change notifications.
    CanInsertPropertyChangeTriggeringPropertiesInData: true
    ....
});

ganttControl = $gantt_container.data("GanttControl");

var activityViews = ganttControl.ActivityViews;
for (var i = 0; i < activityViews.length; i++) {
    var view = activityViews[i];
    var activity = view.Activity;
    //To listen the task property changes.
    var task = activity.DataSource;
    task.PropertyChanged.subscribe(PropertychangeNotifier);
}

function PropertychangeNotifier(sender, args) {
    //Sender - holds the bounded task object
    //arg.PropertyName - holds the property which was changed by end user(like
StartTime, Effort, name etc..)
    //arg.PropertyName - holds the updated value for the arg.PropertyName
}
```

When a task is newly added, you will have to listen to it's PropertyChanged events too. This is discussed in this [topic](#).

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

What are the different ways to improve the performance of the gantt with a large set of tasks?

a) GanttControl.WorkTimeSchedule property

Note that by default the GanttControl uses a "5 days a week, 8 hours a day" work-time schedule to calculate the duration of a task. But, it's possible that in your gantt application, you don't care for specific working-times. You might simply want to follow a 24 X 7 schedule where a task with 24 hour effort would have a duration of 1 day (in a 8 X 5 schedule the duration would be 3 days or more if it spans across weekends).

To set a 24 X 7 schedule which in turn would improve load time performance, do this:

```
$gantt_container.GanttControl({
  // other options..
  WorkTimeSchedule: null
});
```

b) Disable Time Validations while building the Model

The gantt model does a lot of validations when the activity's times are set. But, validating the times might be unnecessary, if you know for a fact that the persisted times are proper (the start and end times don't fall under non-working times, etc.). To turn off validation for such scenarios, do this on gantt initialization:

```
$gantt_container.GanttControl({
  // other options..
  EnforceDependencyConstraints:false,
  ValidateDependencySetting:false
});
```

.... and turn it back on like this after tue gantt is initialized,

```
$ganttControl = $gantt_container.data("GanttControl");
$ganttControl.Model.EnforceDependencyConstraints = true;
$ganttControl.Model.ValidateDependencySetting = true;
```

c) Enable EndTimeBinding

If your tasks have a field with EndTime information and if this field is reliably updated by the gantt and not by any other source, then you can setup the EndTimeBinding in the gantt. This will make the gantt use the pre-calculated EndTime value instead of calculating it on load, thus improving performance. More on this is discussed [here](#) and [here](#).

d) Avoiding Multiple GanttChart Redraws during load

After the Gantt is initialized, the following settings changes will redraw the GanttChart, so these settings are best set during initializing the gantt.

- 1) Changing the BaseTimeUnitWidth (Zoom level).
- 2) Changing AnchorTime (paging and setting chart start and end changes the anchor time).
- 3) Changing ViewWidth and ResizeToFit.

Here is an example of how the above are best set during init:

```

var yearHeader = self.yearHeaderLine();
var monthHeader = self.monthHeaderLine();
tmshs.add(yearHeader);
tmshs.add(monthHeader);

$gantt_container.GanttControl({
  TimeScaleHeaders: tmshs,
  DataSource: jsonData,
  GanttTableOptions: {
    columns: columns,
  },
  GanttChartTemplateApplied: function (sender, args) {
    var $GanttChart = args.element;
    $GanttChart.GanttChart({ AnchorTime: anchorTime, ViewWidth: 2000,
  ResizeToFit: false, });
  }
});

```

Here is an example that shows incorrect setting of the above properties resulting in multiple redraws:

```

var yearHeader = self.yearHeaderLine();
var monthHeader = self.monthHeaderLine();
tmshs.add(yearHeader);
tmshs.add(monthHeader);
var $ganttChart = null;

$gantt_container.GanttControl({
  DataSource: self.jsonData,
  GanttTableOptions: {
    columns: columns,
  },
  GanttChartTemplateApplied: function (sender, args) {
    $ganttChart = args.element;
  }
});

// Too late
$gantt_container.GanttControl({
  TimeScaleHeaders: tmshs
});
$ganttChart.GanttChart({ AnchorTime: newAnchortime });

```

e) Efficient Redraws

See this [topic](#) that discusses how you can sandwich multiple property changes in a BeginUpdate/EndUpdate calls to avoid multiple redraws.

RadiantQ jQuery Gantt Package

How to visually select a row given it's task id?

How to visually select a row given it's task id?

The code below shows how to select an activity who's task id is 5.

```
//To get hold of Ganttcontrol
var ganttControl = $('#gantt_container').data("GanttControl");
//To get the ActivityView corresponding to task id 5
var activity = ganttControl.Model.GetActivityById(5);
//To select the row.
ganttControl.SelectedItem = ganttControl.ActivityViews.GetIActivityView(activity);
```

RadiantQ jQuery Gantt Package

How to remove a dependency?

How to remove a dependency?

1) How to delete the dependency, knowing it's instance itself?

```
// Removing dependency by instance.  
gantControl.Model.Dependencies.remove(dependency);
```

2) How to remove all dependencies for a specific Activity?

This API removes all the dependencies where the activity is a predecessor or a successor.

```
// Removing all dependencies of activity.  
gantControl.Model.Dependencies.removeDependenciesOf(activity);
```

3) Given the from and to activity ids, how to remove their dependency if any?

```
// Get predecessor dependencies by to activity.  
var preds = gantControl.Model.Dependencies.GetPredecessors(toAct);  
// Get successor dependencies by from activity.  
var succs = gantControl.Model.Dependencies.GetSuccessors(fromAct);  
  
var intersectDeps = preds.filter(function (n) {  
    return succs.indexOf(n) != -1  
});  
if (intersectDeps.length > 0) {  
    // Removing the dependency.  
    gantControl.Model.Dependencies.remove(intersectDeps[0]);  
}
```


RadiantQ jQuery Gantt Package

How to get all dependencies of an activity?

How to get all dependencies of an activity?

Use the "**GetDependenciesByActivity**" method in Dependencies collection which will return all the dependencies of the specified activity, where the activity is both a successor and predecessor of the dependency.

```
// Gets an array of dependencies for the specified activity.  
var deps = ganttControl.Model.Dependencies.GetDependenciesByActivity(activity);
```

RadiantQ jQuery Gantt Package

How to Expand all/Collapse all summary tasks dynamically?

Gantt has some inbuilt .apis which enables you to collapses all and expand all summary tasks bars during run time. Below code will explain you how to use those .apis.

```
$(document).ready(function () {
    var $ganttt_container = $('#ganttt_container');

    $ganttt_container.GanttControl({
        DataSource: self.jsonData,
        .....
        .....
    });

    var ganttControl = $ganttt_container.data("GanttControl") || $ganttt_container.data
("radiantqGanttControl");

    $("#collapseButton").click(function () {
        ganttControl.CollapseAll();
    });

    $("#expandButton").click(function () {
        ganttControl.ExpandAll();
    });
});
```

RadiantQ jQuery Gantt Package

How to get hold of bound activity in row click?

This topic shows how to get the bound activity in row click event.

Here is the code :

```
var activityView = null;
var activityName = null;
//GanttChart table( which holds the Chart rows ).
var $gantChartTable = ganttControl.GetGanttChart().find("table.chartRowsContainer");

//Click event of the chart.
$gantChartTable.click(function (sender) {
    var $target = $(sender.target);
    var $tr = $target.closest("tr");
    //To get bound data from row.
    var activityView = ganttControl.options.GanttTable.data("VWGrid"
).GetDataFromRow($tr);
    activityName = activityView.Activity.ActivityName;
    alert(activityName);
});

var $gantTable = ganttControl.GetGanttTable().uiGridBody;

//Double click event of the table.
$gantTable.click(function (sender) {
    var $target = $(sender.target);
    var $tr = $target.closest("tr");
    //To get bound data from row.
    var activityView = ganttControl.options.GanttTable.data("VWGrid"
).GetDataFromRow($tr);
    activityName = activityView.Activity.ActivityName;
    alert(activityName);
});
```

RadiantQ jQuery Gantt Package

How to add new task using the context menu?

This topic shows how to add a new task to GanttControl using GanttTable Context menu

Here is the code:

```
var gantt = $('#gantt_container').data("GanttControl");
var tableContextMenu = gantt.TableContextMenu;
taskMenuItems = [
  {
    keyName: "MyCustomItem", name: "ADD Task", callback: function (key, opt) {
      var newRow = getNewTask();
      //To add new task.
      gantt.AddNewItem(newRow);
    }
  }
]
tableContextMenu.AddNewItems(taskMenuItems, true);
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

How to refresh Gantt with new data with same schema

Refreshing with new data with same schema

To refresh gantt with same schema (same bound property names) you only have to set the DataSource of GanttControl to null first and then reset DataSource property again with the new data. The below code will explain how to do this.

```
// First set the source to null and then set the new source.
$gantt_container.GanttControl("option", "DataSource", null);
$gantt_container.GanttControl({
    WorkTimeSchedule: schedule,
    DataSource: data
});
```

RadiantQ jQuery Gantt Package

How to refresh Gantt with new data with new schema

Refreshing with new data with new schema

To refresh gantt with a new schema (data source has different bound property names) you have to set the DataSource of GanttControl to null first, reset DataSource property again with the new data and also reset all other binding settings as well. Check the below code.

```
    // First set the source to null and then set all binding properties
    and the set the new source.
    $ganttt_container.GanttControl("option", "DataSource", null);
    $ganttt_container.GanttControl({
        WorkTimeSchedule: getSelectedSchedule($schedule.val()),
        IDBinding: new RadiantQ.BindingOptions("TaskID"),
        NameBinding: new RadiantQ.BindingOptions("TaskName"),
        IndentLevelBinding: new RadiantQ.BindingOptions(
"TaskIndentLevel"),
        StartTimeBinding: new RadiantQ.BindingOptions("TaskStartTime"),
        EffortBinding: new RadiantQ.BindingOptions("TaskEffort"),
        PredecessorIndicesBinding: new RadiantQ.BindingOptions(
"TaskPredecessorIndices"),
        ProgressPercentBinding: new RadiantQ.BindingOptions(
"TaskProgressPercent"),
        DescriptionBinding: new RadiantQ.BindingOptions(
"TaskDescription"),
        DataSource: data
    });
```

RadiantQ jQuery Gantt Package

How to export current Gantt data to JSON?

By default Gantt will automatically update the underlying bound JSON task when the activity changed. So, you can simply export the bound list of tasks using **JSON.stringify**:

```
//To generate the json from the gantt's DataSource
$("#exportToJSON").click(function () {
    var jsonObject = gantt.options.DataSource;
    // Specify the fields you want to export, because when the
    CanInsertPropertyChangeTriggeringPropertiesInData gantt option is set
    // it will add some custom property to your datasource which you typically do not
    want to export.
    alert(JSON.stringify(jsonObject, ["ID", "Name", "StartTime", "Effort",
    "Description", "IndentLevel", "Resources", "PredecessorIndices", "ProgressPercent"
    ]));
})
```

RadiantQ jQuery Gantt Package

How to listen the Activity CollectionChanges in GanttControl?

Sometime we might want to listen the activity collection changes to, for example, subscribe to the property change event for newly added tasks at runtime. Here is the code that shows how to do that.

```
gantControl = $gant_container.data("GanttControl");
var activityViews = gantControl.ActivityViews;
//To listen to the task collection changes.
activityViews.CollectionChanged.subscribe(function (event, ui) {
    if (event.type === "insert") {
        //task is the newly added task
        var task = ui.items[0].Activity.DataSource;
        task.PropertyChanged.subscribe(PropertychangeNotifier);
    }
    else {
        //task is the removed task
        var task = ui.items[0].Activity.DataSource;
        task.PropertyChanged.unsubscribe(PropertychangeNotifier);
    }
});
```

To know more about PropertyChanged event take look at this [topic](#).

RadiantQ jQuery Gantt Package

How to add resource to an activity programmatically?

The code below illustrates how to add resource to activity in button click.

```
// Adding resource to activity in button click.
$("#addResource").click(function () {

    var activityview = ganttControl.SelectedItem;
    if (activityview) {
        var activity = activityview.Activity;

        var assgns = activity.Assignments;
        //1 is the resource ID.
        var selResource = activity.Model.GanttResources._resByID.GetItemByKey(1)
        if (selResource && assgns.ContainsResource(selResource) == false)
            assgns.add(new RadiantQ.Gantt.Model.ResourceAssignment(selResource));
    }
    else
        alert("Please select an item in the table first.");
    return false;
});
```

RadiantQ jQuery Gantt Package

How to enable Undo/Redo feature for custom fields ?

Here is the code that illustrates how to enable Undo/Redo feature for custom field(cost).

```

$.ajax({
    .....
    .....
    success: function (data) {
        self.jsonData = data;

        // Listening property changes for custom column(Cost) to let Gantt update -
        Undo/Redo.
        for (i = 0; i < data.length; i++)
            InjectGetAndSet.call(data[i], "Cost", data[i].Cost);

        $.holdReady(false);
    }
});
function InjectGetAndSet(key, value) {
    this.PropertyChanged = new ObjectEvent("PropertyChanged");
    if (key == "Cost") {
        var catchVal = parseInt(value);
        this[key + "_M"] = function (newValue) {
            if (arguments.length == 0) {
                return catchVal;
            }
            else {
                if (newValue != catchVal) {
                    catchVal = newValue;
                    this.PropertyChanged.raise(this,
                        {
                            PropertyName: key,
                            Value: newValue
                        });
                }
            }
        }
    }
    if (RadiantQ.CanUseDefineProperty) {
        Object.defineProperty(this, key, {
            get: function () {
                return catchVal;
            },
            set: function (newValue) {
                if (newValue != catchVal) {
                    catchVal = newValue;
                    this.PropertyChanged.raise(this,
                        {
                            PropertyName: key,
                            Value: newValue
                        });
                }
            },
            enumerable: true,
            configurable: true
        });
    }
}
function ToDollarString(data) {
    if (data.activity.DataSource_M().Cost_M() == null)
        data.activity.DataSource_M().Cost_M(0);
    return "$" + data.activity.DataSource_M().Cost_M();
}

//Column definition.
var columns = [
    {
        field: "Activity.DataSource.Cost",
        title: "Cost",
        editor: "<input data-bind='value:Activity.DataSource.Cost'
data-role='\"spinner\"' data-options='{\"min\":0}'/>",

```

Enabling Undo/Redo

The GanttControl has support for tracking user actions and building an undo/redo stack that the users can then operate on. You can turn on this feature as follows:

```
$('#gantt_container').GanttControl({  
    EnableRecordingActions: true,  
});
```

Now, you can enable Undo/Redo feature for custom column(cost) in a table.

Undo and Redo

You can then allow users to Undo or Redo actions on the top of the stack using the following APIs:

```
var ganttControl = $('#gantt_container').data('GanttControl');  
// To Undo the action on top of the undo stack  
ganttControl.ActionManager.Undo();  
  
// To Redo the action on top of the redo stack  
ganttControl.ActionManager.Redo();
```

Similarly, you can enable Undo/Redo feature for different custom fields.

RadiantQ jQuery Gantt Package

How to add a new resource in resource dataset without reloading Gantt?

Add a new resource to the ResourceItemsSource using the "gantControl.AddNewResourceItem" method. Here is the code that illustrates how to add a new resource in resource dataset in button click at runtime.

```
$(document).ready(function () {  
    .....  
    .....  
    var ganttControl = $gantt_container.data("GanttControl");  
    var resourceItemsSource = ganttControl.options.ResourceItemsSource;  
    var count = resourceItemsSource.length;  
    $("#addResources").click(function () {  
        count++;  
  
        var resourceData = {  
            ResourceID: count,  
            ResourceName: "Resource " + count  
        };  
        // Adding resource at run time.  
        ganttControl.AddNewResourceItem(resourceData);  
    });  
});
```

RadiantQ jQuery Gantt Package

How to save Gantt options in cookies?

How to save the Gantt options in cookies?

When saving state in cookies is disabled (default setting) the end-user will not get back any changes made in Gantt view, after refreshing the page or closing file. If `SaveStateInCookie` option is set as `true` then the changes made in `ZoomLevel`, `TablePanelWidth`, `TaskCollapsingState` options are saved in cookies(memory) by using the below code. It will always return the saved state.

Here is the code example:

```
$('#gantt_container').GanttControl({
  SaveStateInCookie: true,
  /*property: SaveStateInCookieOptions
  To specifies the save state options.
  */
  SaveStateInCookieOptions: {
    /*property: ZoomLevel
    To save the chart zoom level */
    ZoomLevel: true,
    /*property: TablePanelWidth
    To save the width of the table panel */
    TablePanelWidth: true,
    /*property: TaskCollapsingState
    To save the collapsing state of the task */
    TaskCollapsingState: false
  }
});
```

© RadiantQ 2009-2018. All Rights Reserved.

-o-

RadiantQ jQuery Gantt Package

How to notify when dependency connection is failed?

Notification of dependency failer

User can use "DependencyFailedNotifierCallBack" function to notify when the dependency connection is failed.

Here is the code example:

```
$('#gantt_container').GanttControl({
  DependencyFailedNotifierCallBack: function (from, to) {
    // This function noitifies once the dependency connection gets failed(also
    // due to Circular Dependency).
    // Here Circular Dependency attempted from "Task 1" to "Task 2".
    if (window["console"] && console["error"])
      console.error("Circular Dependency attempt from : " +
from.ActivityName_M() + " to " + to.ActivityName_M());
  }
});
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

How to make selection and hover effect?

Selection and hover effect

"ApplySelectionAndHoverEffect" option is used to highlight the selected or hovered row in a gantt. Selection and hover effect option is enable by default. You can however enable/disable this feature by setting the "ApplySelectionAndHoverEffect" option to true/false.

Here is the code example:

```
$('#gantt_container').GanttControl({  
  ApplySelectionAndHoverEffect: false  
});
```


RadiantQ jQuery Gantt Package

How to make Custom Progress Calculation?

CustomProgressCalculation

A custom logic that will be used, to calculate the progress value of a summary bar. It gives you the option to calculate the progress value. Set this option before setting the ItemsSource property to the Gantt.

Here is the code example:

```
$('#ganttt_container').GanttControl({
    CustomProgressCalculation: CustomProgressCalculationCallback,
});

// Implements progress calculation logic that's similar to MS Project.
function CustomProgressCalculationCallback(parentActivity) {
    var totalEffort = RQTimeSpan.Zero_M();
    var completedEffort = RQTimeSpan.Zero_M();

    for (var i = 0; i < parentActivity.ChildActivities_M().length; i++) {
        var child = parentActivity.ChildActivities_M()[i];
        var childEffort = child.CumulativeEffort_M();
        totalEffort = totalEffort.add(childEffort);
        completedEffort =
completedEffort.add(RQTimeSpan.fromTicks(childEffort.Ticks_M() *
(child.ProgressPercent_M() / 100)));
    }

    if (totalEffort == RQTimeSpan.Zero_M())
        return 0.0;
    else
        return (completedEffort.Ticks_M() / totalEffort.Ticks_M()) * 100.0;
}
```

This is illustrated in this samples:

In HTML : ...\\Samples\\GanttTaskBarBrowseToCue.htm

© RadiantQ 2009-2018. All Rights Reserved.

RadiantQ jQuery Gantt Package

How to add dropdown for custom column?

This topic shows how to add a custom dropdown for given column.

Here is the code:

```
//Column definition.
var columns = [
{
  field: "CustomColumn",
  title: "Custom Column",
  width: 80,
  template: '<div> ${ data.Activity_M().DataSource.CustomColumn || "" } </div>',
  editor: "<input data-bind='MultiPicker:data.DataSource' />",
}
];

// Binder function for multiselect items
Binder.MultiPicker = function ($elem, role, value, actview) {
  var selectedItems = actview.Activity.DataSource.CustomColumn || [];
  var items = ["A", "B", "C", "D"];

  // Using the in-built 'ResourcePicker' widget and customizing it.
  $elem.ResourcePicker({
    "source": items,
    SelectedResources: selectedItems,
    open: function () {
      actview.GC.PreventDefaultFns = true;
    },
    close: function () {
      actview.GC.PreventDefaultFns = false;
    },
    destroy: function () {
      var resourcePicker = $(this).data('ResourcePicker');
      if (resourcePicker == undefined || resourcePicker == null)
        return;

      if (resourcePicker.IsChanged() == true)
        actview.Activity.DataSource.CustomColumn =
resourcePicker.selectedItems;
    }
  });
}
```


RadiantQ jQuery Gantt Package

How to enable paging while autoscrolling in GanttChart ?

How to enable paging while Autoscrolling in GanttChart ?

" EnablePagingOnAutoScroll" option is used to enable an automatic paging on autoscroll feature while horizontally dragging the taskbar in GanttChart. By default " EnablePagingOnAutoScroll " is false, You can however enable this feature by setting the GanttChart option "EnablePagingOnAutoScroll" to true.

Here is the code example:

```
$('#gantt_container').GanttControl({
  GanttChartTemplateApplied: function (sender, args) {
    var $GanttChart = args.element;
    $GanttChart.GanttChart({
      EnablePagingOnAutoScroll: true
    });
  },
});
```

FlexyGantt

Getting Started

RadiantQ jQuery Gantt Package

In HTML

Your First Gantt

Let us start with creating a new project directory, for example MyFirstGantt.

1) Gantt Widget Source JS files

To begin with, you need the JS Source files that are required by the Gantt Widget. These files are in the <install path>/Src folder. Simply copy over the whole Src folder into the above directory. This folder also has other dependant css files, etc. You can delete the "Src/bin" folder as that's not required for this HTML sample.

2) Sample Utility JS files

Some JS files with utility functions that enables inline-editing in the grid, etc. are in the <install path>/Samples/Scripts folder. Copy over that entire Scripts directory as well into MyFirstGantt.

3) Sample JSON Data

Create a SampleData.json file containing a list of sample tasks to be displayed in the FlexyGantt.

SampleData.json content:

```
[{
  "TName" : "Team1",
  "PStartTime" : "2012-04-02T00:00:00Z",
  "PEndTime" : "2012-04-18T00:00:00Z",
  "Resources" : [{
    "RName" : "JohnH",
    "PStartTime" : "2012-04-04T00:00:00Z",
    "PEndTime" : "2012-04-15T00:00:00Z",
    "Tasks" : [{
      "TaskName" : "Task 1",
      "StartTime" : "2012-04-03T00:00:00Z",
      "EndTime" : "2012-04-12T00:00:00Z",
      "Progress" : 20
    }]
  }],
},
{
  "RName" : "VictorG",
  "PStartTime" : "2012-04-06T00:00:00Z",
  "PEndTime" : "2012-04-18T00:00:00Z",
  "Tasks" : [{
    "TaskName" : "Task 1",
    "StartTime" : "2012-04-03T00:00:00Z",
    "EndTime" : "2012-04-18T00:00:00Z",
    "Progress" : 20
  }]
},
{
  "RName" : "JasonS",
  "PStartTime" : "2012-04-06T00:00:00Z",
  "PEndTime" : "2012-04-18T00:00:00Z",
  "Tasks" : [{
    "TaskName" : "Task 1",
    "StartTime" : "2012-04-06T00:00:00Z",
    "EndTime" : "2012-04-12T00:00:00Z",
    "Progress" : 20
  }],
  {
    "TaskName" : "Task 2",
    "StartTime" : "2012-04-12T00:00:00Z",
    "EndTime" : "2012-04-18T00:00:00Z",
    "Progress" : 70
  }
}]
},
{
  "TName" : "Team2",
  "PStartTime" : "2012-04-10T00:00:00Z",
  "PEndTime" : "2012-04-20T00:00:00Z",
  "Resources" : [{
    "RName" : "BalajiN",
    "PStartTime" : "2012-04-08T00:00:00Z",
    "PEndTime" : "2012-04-18T00:00:00Z",
    "Tasks" : [{
      "TaskName" : "Task 1",
      "StartTime" : "2012-04-08T00:00:00Z",
      "EndTime" : "2012-04-20T00:00:00Z",
      "Progress" : 20
    }]
  }],
},
{
  "RName" : "LiM",
  "PStartTime" : "2012-04-12T00:00:00Z",
  "PEndTime" : "2012-04-18T00:00:00Z",
  "Tasks" : [{
    "TaskName" : "Task 1",
    "StartTime" : "2012-04-08T00:00:00Z",
```

4) HTML file including the Gantt Widget

Create a new HTML file inside the project directory (MyFirstGantt) and reference the following source files. Remember to link to the right version of the RadiantQ jQuery Gantt in the last reference below.

```
<head>
  <title></title>
  <link href="Src/Styles/jquery-ui-themes/smoothness/jquery-ui.css" rel
="stylesheet" />
  <link href="Src/Styles/radiantq.gantt.default.css" rel="stylesheet" />
  <link href="Src/Styles/VW.Grid.css" rel="stylesheet" />
  <script src="Src/Scripts/jquery-1.11.2.min.js" type="text/javascript"></script>
  <script src="Src/Scripts/jquery-ui-1.11.4/jquery-ui.min.js" type
="text/javascript"></script>
  <script type="text/javascript" src="Src/Scripts/jquery.layout-latest.min.js"></
script>
  <script src="Src/Scripts/Utils/date.js" type="text/javascript"></script>
  <script src="Src/ResourceStrings/en-US.js" type="text/javascript"></script>
  <script src='Src/Scripts/VW.Grid.1.min.js' type='text/javascript'></script>
  <script src='Src/Scripts/RadiantQ-jQuery.gantt.5.0.trial.min.js' type
='text/javascript'></script>
</head>
```

Initializing the Flexy Table

Now you have to setup the different column you want to show in the FlexyTable. You can do so by defining a html table element as follows (insert the below code snippet right above the end of the body tag above):

```
var columns = [
  {
    field: "Name",
    title: "Name",
    editor:
RadiantQ.Default.Template.FlexyGanttExpandableTextBoxEditor("nameConverter"),
    template:
RadiantQ.Default.Template.FlexyGanttExpandableTextBlockTemplate("nameConverter")
  }
];
```

Creating Gantt

Now include code to retrieve the json file you created above and then initialize the FlexyGantt widget binding it with the loaded data.

```
<script type="text/javascript">

  var self = this;
  var jsonData;

  // Retrieve JSON Data from file.
  $.holdReady( true );
  $.ajax( {
    type: "GET",
    dataType: 'json',
```



```

url: 'FlexyGanttSkeleton.json',
converters:
  {
    "text json": function ( data )
    {
      return $.parseJSON(data, true /*converts date
strings to date objects*/, true /*converts ISO dates to local dates*/);
    }
  },
success: function ( data )
{
  self.jsonData = data;
  $.holdReady( false );
}
} );
$(document).ready(function () {
  // Determine the time the timeline in the gantt should scroll to.
  var anchorTime = self.jsonData[0].PStartTime.clone();

  // The template that defines the look for the task bars.
  "rq-gc-taskbar" is a built-in style that defines a default look for the task bars.
  var tTemplate = "<div class='rq-gc-taskbar'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div></div>"
;

  var $gantt_container = $("#gantt_container");
  // Initialize the FlexyGantt widget.
  $gantt_container.FlexyGantt({
    DataSource: self.jsonData,
    GanttTableOptions: {
      columns: columns
    },
    //the FlexyGantt is bound to resolve the hierarchy of
Team/Resources/Tasks.
    resolverFunction: function (data) {
      // If data is wrapped by KO, then data itself could be a
function and so we pick the object from the function.
      if ($.isFunction(data)) {
        data = data()[0];
      }

      if (data["Resources"] != undefined) {
        if ($.isFunction(data["Resources"]))
          return data["Resources"]();
        else
          return data["Resources"];
      }
      else if (data["RName"] != undefined) {
        if (data["Tasks"] != undefined) {
          return null;
        }
        else
          return new Array();
      }
      return null;
    },
    TaskStartTimeProperty: "StartTime",
    ParentTaskStartTimeProperty: "PStartTime",
    TaskItemTemplate: tTemplate,
    ParentTaskItemTemplate: pTemplate,
    TaskEndTimeProperty: "EndTime",

```

```

        ParentTaskEndTimeProperty: "PEndTime",
        TasksListProperty: "Tasks",
        TaskTooltipTemplate: $("#TaskTooltipTemplate").html()
    });

    // to get the name from the bounded list
    function nameConverter(flexyNodeData) {
        var data;
        // The grid calls this converter with flexyNodeData as a arg.
        if (flexyNodeData)
            data = flexyNodeData.Data();
        // The grid calls this converter with flexyNodeData as a
datacontext.
        else
            data = this.data;

        if (data["TName"])
            return data["TName"];
        else if (data["RName"])
            return data["RName"];
        else if (data["TaskName"])
            return data["TaskName"];

        return;
    };

    // to get the short time format.
    var toolTipDateFormat = Date.CultureInfo.formatPatterns.shortDate + ' '
+ "HH:mm:ss";
    function startTimeTooltipConverter() {
        if (this.data["PStartTime"])
            return this.data["PStartTime"].toString(toolTipDateFormat);
        else if (this.data["StartTime"])
            return this.data["StartTime"].toString(toolTipDateFormat);

        return null;
    }

    function endTimeTooltipConverter() {
        if (this.data["PEndTime"])
            return this.data["PEndTime"].toString(toolTipDateFormat);
        else if (this.data["EndTime"])
            return this.data["EndTime"].toString(toolTipDateFormat);

        return null;
    }

</script>

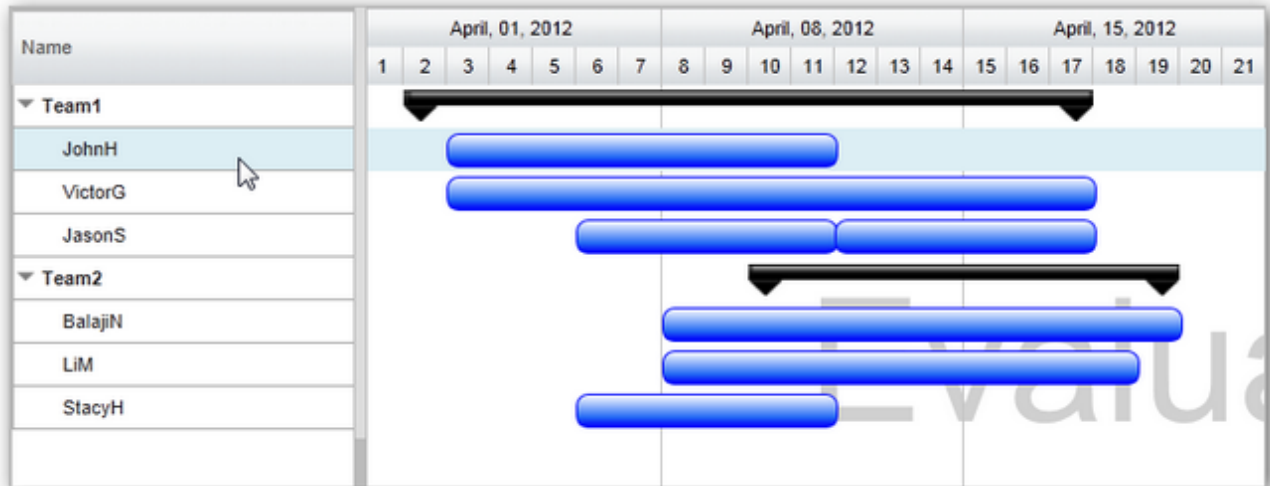
<body>
    <!-- Div that will be transformed into the FlexyGantt widget above.-->
    <div id="gantt_container" style="height: 550px;">
    </div>
</body>

```

The content of your MyFirstGantt directory should look like this:

- Src
- Scripts
- json file
- html file

Here is the resultant gantt:



Sample In

Browser

Finally, take a look at this [topic](#) that show how you can clean up the Src folder in your project to remove unnecessary files.

RadiantQ jQuery Gantt Package

In Angular

Angular Gantt sample

- Sample HTML template

The flexygantt samples should be created using the element '`<RQ:FlexyGantt></RQ:FlexyGantt>`' .

You can set the gantt options such as Datasource, AnchorTime, ResizeToFit, UseVirtualization etc., as illustrated here.

Here, the datasource for gantt is from the class function(i.e looped item source).

The gantt columns should be created with available properties. The client and editor templates is to be set using '`<ng-template></ng-template>`' specifying it's absolute template references(prefixed with '#') as illustrated here.

```

<RQ:FlexyGantt [DataSourceUrl]
="/app/Samples/FlexyGanttSkeleton/FlexyGanttSkeleton.json" [AnchorTime]
="'2018-04-01'"
    [ResizeToFit]="false" Height="500px"
    [TaskItemTemplate]="taskItemTemplate" [ParentTaskItemTemplate]
="parentTaskItemTemplate"
    [TaskStartTimeProperty]="'StartTime'" [ParentTaskStartTimeProperty]
="'PStartTime'"
    [TaskEndTimeProperty]="'EndTime'" [ParentTaskEndTimeProperty]
="'PEndTime'"
    [TasksListProperty]="'Tasks'" [resolverFunction] ="resolverFunction"
    TaskTooltipTemplate="{userOptions.taskTooltipTemplate}"
    [AfterGanttWidgetInitializedCallback]
="AfterGanttWidgetInitializedCallback">

    <GanttTableOptions>
        <Columns>
            <Column field="Name" title="Name" [clientTemplate]="fgNameTemplate"
[clientEditorTemplate]="fgNameEditorTemplate"></Column>
        </Columns>
    </GanttTableOptions>
</RQ:FlexyGantt>

<!--TaskItemTemplate-->
<ng-template #taskItemTemplate>
    <div id="ng-templ-taskBar" #tTemp>
        <div class='rq-gc-taskbar' [getTaskItemTemplate]="tTemp.innerHTML"><div class
='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div class
='start-resizeThumb'></div></div>
    </div>
</ng-template>

<!--ParentTaskItemTemplate-->
<ng-template #parentTaskItemTemplate>
    <div id="ng-templ-parentBar" #pTemp>
        <div class='rq-gc-parentBar' [getParentTaskItemTemplate]="pTemp.innerHTML"><
div class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='start-resizeThumb'></div><div class='rq-gc-parentBar-leftCue'></div><div
class='rq-gc-parentBar-middle'></div><div class='rq-gc-parentBar-rightCue'></div></
div>
    </div>
</ng-template>

<!--Column Templates-->
<ng-template #fgNameTemplate>
    <div #nameTemp>
        <div class="rq-grid-expand-indentWidth" [rqTemplateBinder]
="nameColTempl.style_indentWidth"></div>
        <div [rqTemplateBinder]="nameColTempl.style_arrowCont" class
="arrowContainer">
            <div id="arrow" onclick="RadiantQ.Gantt.ExpanderOnClick(this,event)"
[rqTemplateBinder]="nameColTempl.class_arrow"></div>
            </div>
            <div class="rq-grid-expander-text" data-bind="text:nameConverter"></div>
            <div [getClientTemplate]="nameTemp.innerHTML"></div>
        </div>
    </div>
</ng-template>

<!--Column Editors-->
<ng-template #fgNameEditorTemplate>
    <div #nameEditor>
        <div class="rq-grid-expand-indentWidth" [rqTemplateBinder]
="nameColTempl.style_indentWidth"></div>
        <div [rqTemplateBinder]="nameColTempl.style_arrowCont" class
="arrowContainer">
            <div id="arrow" onclick="RadiantQ.Gantt.ExpanderOnClick(this,event)"
[rqTemplateBinder]="nameColTempl.class_arrow"></div>
            </div>
    </div>
</ng-template>

```

FlexyGanttSkeleton.html

- Sample JSON Data

Create a SampleData.json file containing a list of sample tasks to be displayed in the gantt.

```
[{
  "TName" : "Team1",
  "PStartTime" : "2018-04-02T00:00:00Z",
  "PEndTime" : "2018-04-18T00:00:00Z",
  "Resources" : [{
    "RName" : "JohnH",
    "PStartTime" : "2018-04-04T00:00:00Z",
    "PEndTime" : "2018-04-15T00:00:00Z",
    "Tasks" : [{
      "TaskName" : "Task 1",
      "StartTime" : "2018-04-03T00:00:00Z",
      "EndTime" : "2018-04-12T00:00:00Z",
      "Progress" : 20,
      "Priority" : "High"
    }]
  }],
},
{
  "RName" : "VictorG",
  "PStartTime" : "2018-04-06T00:00:00Z",
  "PEndTime" : "2018-04-18T00:00:00Z",
  "Tasks" : [{
    "TaskName" : "Task 1",
    "StartTime" : "2018-04-03T00:00:00Z",
    "EndTime" : "2018-04-18T00:00:00Z",
    "Progress" : 20,
    "Priority" : "Medium"
  }]
},
{
  "RName" : "JasonS",
  "PStartTime" : "2018-04-06T00:00:00Z",
  "PEndTime" : "2018-04-18T00:00:00Z",
  "Tasks" : [{
    "TaskName" : "Task 1",
    "StartTime" : "2018-04-06T00:00:00Z",
    "EndTime" : "2018-04-12T00:00:00Z",
    "Progress" : 20,
    "Priority" : "High"
  }],
  {
    "TaskName" : "Task 2",
    "StartTime" : "2018-04-12T00:00:00Z",
    "EndTime" : "2018-04-18T00:00:00Z",
    "Progress" : 70,
    "Priority" : "Medium"
  }
}]
},
{
  "TName" : "Team2",
  "PStartTime" : "2018-04-10T00:00:00Z",
  "PEndTime" : "2018-04-20T00:00:00Z",
  "Resources" : [{
    "RName" : "BalajiN",
    "PStartTime" : "2018-04-08T00:00:00Z",
    "PEndTime" : "2018-04-18T00:00:00Z",
    "Tasks" : [{
      "TaskName" : "Task 1",
      "StartTime" : "2018-04-08T00:00:00Z",
      "EndTime" : "2018-04-20T00:00:00Z",
      "Progress" : 20,
      "Priority" : "High"
    }]
  }],
  {
    "RName" : "LiM",

```

FlexyGanttSkeleton.json

- Sample TS

The 'FlexyGanttSkeleton.ts' file imports the TypeScript libraries from top-level 'ts' folder.

The sample template(FlexyGanttSkeleton.html) and its corresponding CSS(FlexyGanttSkeleton.css) can be declared using '@Component({ })' decorator with selector as FlexyGanttSkeleton specified as sample root tag in index.html

Here, the export class pass the value for gantt options which sets as template in FlexyGanttSkeleton.html' such as DataSource, AnchorTime, ResizetoFit, Columns and bindings.


```

/// <reference path="../../ts/radiantqgantts2.3.d.ts" />
/// <reference path="../../ts/jquery.ui.d.ts" />
/// <reference path="../../ts/datejs.d.ts" />

import { Component, ElementRef, ViewEncapsulation } from '@angular/core';

@Component({
  selector: 'FlexyGanttSkeleton',
  templateUrl: './app/Samples/FlexyGanttSkeleton/FlexyGanttSkeleton.html',
  styleUrls: ['./app/Samples/FlexyGanttSkeleton/FlexyGanttSkeleton.css'],
  encapsulation: ViewEncapsulation.None // Fix: Styles for child elements not
  applied due to encapsulation(shadow dom).
})
export class FlexyGanttSkeleton {
  private userOptions = new Object();
  nameColTempl = {
    style_indentWidth: { key: 'style', value: 'height: 1px; width:
    ${RadiantQ.Gantt.LevelToIndentWidth(data.Level(), data.IsParentType())}px' },
    style_arrowCont: { key: 'style', value: 'width: 12px; display:
    ${data.IsParentType() ? (data.HierarchicalItem.CanShowCue() ? \"block\" : \"none\") :
    \"none\" }' },
    class_arrow: { key: 'class', value: '${data.HierarchicalItem.IsExpanded() ?
    \" rq-grid-expand-arrow rq-grid-collapse-arrow\": \"rq-grid-expand-arrow\"}
    rq-Ignore-click' }
  };
  private rqFlexyGanttElem: HTMLElement = null;
  constructor(private elementRef: ElementRef) {
    this.rqFlexyGanttElem = elementRef.nativeElement;
    userGanttSettings(this.userOptions);
  }

  AfterGanttWidgetInitializedCallback($gantts_container: any) {
    // This gets fired once after the gantt widget is intialized with user
    options.
    var flexyGantt = $gantts_container.data('FlexyGantt');
    // Here, user can customize the gantt using 'flexyGantt' instance.
  };

  resolverFunction(data) {
    if (data["Resources"] != undefined) {
      return data["Resources"];
    }
    return null;
  }
}

function userGanttSettings(userOptions) {
  userOptions.taskTooltipTemplate = "<div class='TaskTooltip'><div
  align='center'>Name: ${nameConverter(data)} </div><div><span>StartTime :
  ${startTimeTooltipConverter(data)} </span><br/><span>EndTime :
  ${endTimeTooltipConverter(data)} </span></div></div>";
}

// to get the name from the bounded list
function nameConverter(flexyNodeData, value) {
  var data;
  // The grid calls this converter with flexyNodeData as a arg.
  if (flexyNodeData.Data)
    data = flexyNodeData.Data();
  // The grid calls this converter with flexyNodeData as a datacontext.
  else
    data = flexyNodeData;
  if (value == undefined) {
    if (data["TName"] != undefined)
      return data["TName"];
    else if (data["RName"] != undefined)
      return data["RName"];
    else if (data["TaskName"] != undefined)

```

FlexyGanttSkeleton.ts

The 'AfterGanttWidgetInitializedCallback' function gets fired once after the gantt widget is initialized with user options. Here, user can customize the gantt using the 'gantControl' instance

The gantt widget initialization process takes place in 'RQGanttSettings.ts' with user options.

This is illustrated in the following path:

In Angular :
\PlatformSamples\AngularSamples\app\Samples\FlexyGanttSkeleton\...

Also, illustrated to import both 'GanttControl and FlexyGantt' samples in one page in the following path:

In Angular : \PlatformSamples\AngularSamples\app\Samples\ResourceLoadView\...

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

In React

React Gantt sample

Import the react libraries and the 'FlexyGantt' component from its desired path
'RQSrc/React_Components/FlexyGantt.jsx';

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import FlexyGantt from 'RQSrc/React_Components/FlexyGantt.jsx';
import './FlexyGanttSkeleton.css';

// To get the name from the bounded list .
// Converter should be define globally using window
function initConverters() {
  window.nameConverter = function (flexyNodeData, value) {
    var data;
    // The grid calls this converter with flexyNodeData as a arg.
    if (flexyNodeData.Data)
      data = flexyNodeData.Data();
    // The grid calls this converter with flexyNodeData as a datacontext.
    else
      data = flexyNodeData;
    if (value == undefined) {
      if (data["TName"] != undefined)
        return data["TName"];
      else if (data["RName"] != undefined)
        return data["RName"];
      else if (data["TaskName"] != undefined)
        return data["TaskName"];
    }
    else {
      if (data["TName"] != undefined)
        data["TName"] = value;
      else if (data["RName"] != undefined)
        data["RName"] = value;
      else if (data["TaskName"] != undefined)
        data["TaskName"] = value;
    }
  };

  // To get the short time format.
  var tooltipDateformat = Date.CultureInfo.formatPatterns.shortDate + ' ' +
  "HH:mm:ss";
  window.startTimeTooltipConverter = function (data) {

    if (data["PStartTime"])
      return data["PStartTime"].toString(tooltipDateformat);
    else if (data["StartTime"])
      return data["StartTime"].toString(tooltipDateformat);

    return null;
  }
  window.endTimeTooltipConverter = function (data) {
    if (data["PEndTime"])
      return data["PEndTime"].toString(tooltipDateformat);
    else if (data["EndTime"])
      return data["EndTime"].toString(tooltipDateformat);

    return null;
  }
}

export default class FlexyGanttSkeleton extends React.Component {
  constructor() {
    super();
    initConverters();
    this.columns = [{
      field: "Name",
      title: "Name",
      editor: RadiantQ.Default.Template.FlexyGanttExpandableTextBoxEditor(
"nameConverter"),
      template:
RadiantQ.Default.Template.FlexyGanttExpandableTextBlockTemplate("nameConverter")

```

FlexyGanttSkeleton.jsx.

The ReactDOM.render method can render the "FlexyGanttSkeleton" component into the DOM. Here, the second argument "container" specifies the mount element . Mount element defined inside the "index.html".

This is illustrated in the following path:

In React : \PlatformSamples\React
\Samples\FlexyGanttSkeleton\FlexyGanttSkeleton.jsx.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

*RadiantQ jQuery Gantt Package***In ASP.NET**

Create a new ASP.NET project in Visual Studio:

VS 2012 :FILE --> New --> Project --> Visual C# --> Web --> ASP.NET Web Forms Application, create a project.

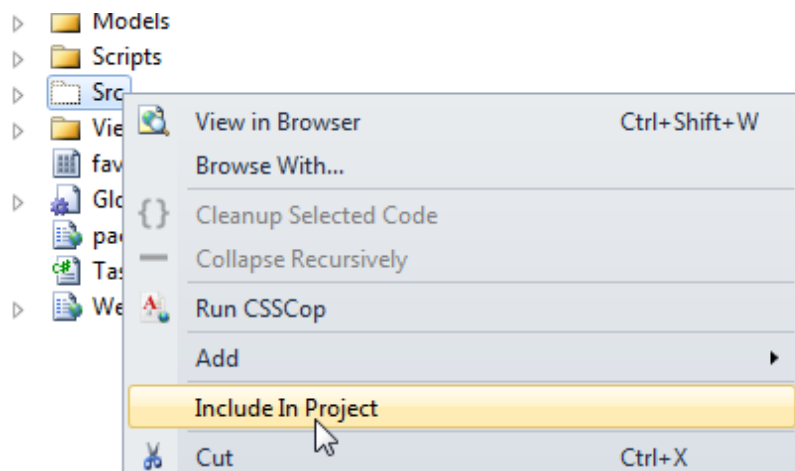
VS 2010 :FILE --> New --> Project --> Visual C# --> Web --> ASP.NET Web Application, create a project.

1) Gantt Widget Source JS files

To begin with, you need the JS Source files that are required by the Gantt Widget. These files are in the <install path>/Src folder. Copy over this folder into the above Project folder (though this folder is very big in size, it contains the required CSS, etc for all Themes, locales, etc. and not all of them will be loaded inside your page).

Go ahead and remove the bin folder inside this Src folder.

Then in the project's Solution Explorer click on the "Show All Files" toolbar item to show this newly included Src folder and include that in the project.



include Src folder in project

2) Sample Utility JS files

Some JS files with utility functions that enables inline-editing in the grid, etc. are in the <install path>/Samples/Scripts folder. Copy over the contents of the Scripts folder from the above install path into the Scripts folder in your project folder (when you create a new project a Scripts folder would be automatically created in your project folder).

Then include the newly added Script files into your project following the same procedure as in the previous step.

3) Create a sample Data Source(JSON Data)

You will typically use an Entity Model, ADO.NET, etc. to retrieve data from a database. But, to keep things simple, we will create a simple list of "projects" and return it to the client from the server.

Create a new type called Project and Task. In Solution Explorer right click on the Project name, then Add --> New Item --> class (call it project.cs) and define a class like below.

```
public class Project
{
    public Project() { this.Tasks = new System.Collections.ObjectModel.
ObservableCollection<Task>(); }
    public string PName { get; set; }
    public System.Collections.ObjectModel.ObservableCollection<Task> Tasks { get;
set; }
    public DateTime OverallStartTime { get; set; }
    public DateTime OverallEndTime { get; set; }
}

public class Task
{
    public string Status;
    public string TaskName { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public double Progress { get; set; }
    public bool IsOverlapping { get; set; }
    public object Tag { get; set; }
}
```

Then, add a new Generic handler to the project; this is going to provide the data source to the client page.

To add a Generic handler in the visual studio, in Solution Explorer right click on the Project name, then Add --> New Item --> Generic Handler, and create a new instance naming it, for example, FGTaskListHandler.ashx.

In the generic handler (FGTaskListHandler.ashx) create a list of TaskInfos, convert that list to a json using JavaScriptSerializer or any other "JSON serializer" to convert it to json data and add that to the Response.

```
public class FGTaskListHandler : IHttpHandler
{
    public void ProcessRequest(HttpContext context)
    {
        DateTime dtS = DateTime.Now;
        System.Collections.ObjectModel.ObservableCollection<Project> projects = new
System.Collections.ObjectModel.ObservableCollection<Project>();

        Project prj = new Project() { PName = "Project1", OverallStartTime = dtS +
TimeSpan.FromDays(1), OverallEndTime = dtS + TimeSpan.FromDays(6) };
        prj.Tasks.Add(new Task() { StartTime = dtS, EndTime = dtS + TimeSpan
.FromDays(2), TaskName = "John's Task 3", Progress = 20 });
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(3), EndTime =
dtS + TimeSpan.FromDays(4), TaskName = "John's Task 2", Progress = 50 });
        projects.Add(prj);

        prj = new Project() { PName = "Project2", OverallStartTime = dtS + TimeSpan
.FromDays(1.5), OverallEndTime = dtS + TimeSpan.FromDays(5.5) };
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(1), EndTime =
dtS + TimeSpan.FromDays(4), TaskName = "Victor's Task", Progress = 20 });
        projects.Add(prj);

        prj = new Project() { PName = "Project3", OverallStartTime = dtS + TimeSpan
.FromDays(2), OverallEndTime = dtS + TimeSpan.FromDays(5) };
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(1), EndTime =
dtS + TimeSpan.FromDays(4), TaskName = "Jason's Task 1", Progress = 20 });
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(7), EndTime =
dtS + TimeSpan.FromDays(9), TaskName = "Jason's Task 2", Progress = 70 });
        projects.Add(prj);

        prj = new Project() { PName = "Project4", OverallStartTime = dtS + TimeSpan
.FromDays(0.5), OverallEndTime = dtS + TimeSpan.FromDays(3.5) };
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(1.5), EndTime
= dtS + TimeSpan.FromDays(4), TaskName = "Vicky's Task", Progress = 20 });
        projects.Add(prj);

        prj = new Project() { PName = "Project5", OverallStartTime = dtS + TimeSpan
.FromDays(2), OverallEndTime = dtS + TimeSpan.FromDays(6) };
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(2.2), EndTime
= dtS + TimeSpan.FromDays(3.8), TaskName = "Oleg's Task 1", Progress = 50 });
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(5), EndTime =
dtS + TimeSpan.FromDays(6), TaskName = "Oleg's Task 2", Progress = 50 });
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(8), EndTime =
dtS + TimeSpan.FromDays(9.6), TaskName = "Oleg's Task 3", Progress = 20 });
        projects.Add(prj);

        prj = new Project() { PName = "Project6", OverallStartTime = dtS + TimeSpan
.FromDays(2.5), OverallEndTime = dtS + TimeSpan.FromDays(4.5) };
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(0.8), EndTime
= dtS + TimeSpan.FromDays(2), TaskName = "Kim's Task", Progress = 90 });
        projects.Add(prj);

        prj = new Project() { PName = "Project7", OverallStartTime = dtS + TimeSpan
.FromDays(5), OverallEndTime = dtS + TimeSpan.FromDays(7.5) };
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(1.5), EndTime
= dtS + TimeSpan.FromDays(4), TaskName = "Balaji's Task 1", Progress = 20 });
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(5), EndTime =
dtS + TimeSpan.FromDays(8), TaskName = "Balaji's Task 2", Progress = 70 });
        projects.Add(prj);

        prj = new Project() { PName = "Project8", OverallStartTime = dtS + TimeSpan
.FromDays(3), OverallEndTime = dtS + TimeSpan.FromDays(6.3) };
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(1.75), EndTime
= dtS + TimeSpan.FromDays(2.25), TaskName = "Li's Task", Progress = 20 });
        projects.Add(prj);
    }
}
```


4) ASPX file including the Gantt Widget

Create the .aspx sample file

In the visual studio SolutionExplorer right click on the Project name, then Add --> New Item --> Web Form(call it WebForm1.aspx).

Include the following namespace in the created aspx.

```
<%@ Register Assembly="RadiantQ.Web.JQGantt" Namespace="RadiantQ.WebForms.JQGantt"
TagPrefix="RQ" %>
```

Add the *RadiantQ.Web.JQGantt.dll* to your project reference, you can find the dll here:
<install folder>\Src\bin\DotNET4MVC4\RadiantQ.Web.JQGantt.dll.

Reference the following source files in the aspx inside the <head> tag. Remember to link to the right version of the RadiantQ jQuery Gantt in the last one line reference below.

```
<head runat="server">
  <title></title>
  <link id="themeChooser" href="<%= Page.ResolveClientUrl(
~/Src/Styles/jquery-ui-themes/smoothness/jquery-ui.css)" %>" rel="stylesheet" />
  <link id="default" href="<%= Page.ResolveClientUrl(
~/Src/Styles/radiantq.gantt.default.css)" %>" rel="stylesheet" />
  <link href="<%= Page.ResolveClientUrl("~/Src/Styles/VW.Grid.css)" %>" rel
="stylesheet" />
  <script src="<%= Page.ResolveClientUrl("~/Src/Scripts/jquery-1.11.2.min.js)" %>"
type="text/javascript"></script>
  <script type="text/javascript" src="<%= Page.ResolveClientUrl(
~/Src/Scripts/jquery-ui-1.11.4/jquery-ui.min.js)" %>"></script>
  <script type="text/javascript" src="<%= Page.ResolveClientUrl(
~/Src/Scripts/jquery.layout-latest.min.js)" %>"></script>
  <script src="<%= Page.ResolveClientUrl("~/Src/Scripts/Utils/date.js)" %>" type
="text/javascript"></script>
  <script type="text/javascript" src="<%= Page.ResolveClientUrl(
~/Src/ResourceStrings/en-US.js)" %>"></script>
  <script type="text/javascript" src="<%=
Page.ResolveClientUrl("~/Src/Scripts/VW.Grid.1.min.js)" %>"></script>
  <script type="text/javascript" src="<%=
Page.ResolveClientUrl("~/Src/Scripts/RadiantQ-jQuery.gantt.5.0.trial.min.js)" %>"></
script>
  <script src="<%= Page.ResolveClientUrl("~/Src/Scripts/RQGantt_Init.js)" %>" type
="text/javascript"></script>
</head>
```

Creating FlexyGantt

Now include code to retrieve the json file you created above and then initialize the GanttControl widget binding it with the loaded data.

In the WebForm1.aspx, within the default <form> tag in <body>, add this tag:

```
<body>
  <form id="form1" runat="server">
    <RQ:FlexyGantt runat="server" ID="gantt" Height="500"
      DataSourceUrl="FGTaskListHandler.ashx"
      HierarchyResolverFunction="ResolverFunction"
```

```

        TaskStartTimeProperty="StartTime"
        TaskEndTimeProperty="EndTime"
        TaskItemTemplate="<div class='rq-gc-taskbar'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div></div>"
        ParentTaskItemTemplate="<div class='rq-gc-parentBar'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='parentLeftPoly-style'></div><div
class='parentMiddleBar-style'></div><div
class='rq-gc-parentBar-rightCue'></div></div></div>"
        ParentTaskEndTimeProperty="OverallEndTime"
        ParentTaskStartTimeProperty="OverallStartTime" />
    <div>
    </div>
</form>
</body>

```

Initializing the Gantt Table

Now you have to setup the different columns you want to show in the GanttTable. You can do so by defining columns inside GanttTableOptions property in GanttControl.

```

<RQ:FlexyGantt runat="server" ID="gantt" Height="500"
    DataSourceUrl="../../DataSources/FGTaskListHandler.ashx"
    HierarchyResolverFunction="ResolverFunction"
    TaskStartTimeProperty="StartTime"
    TaskEndTimeProperty="EndTime"
    TaskItemTemplate="<div class='rq-gc-taskbar'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div></div>"
    ParentTaskItemTemplate="<div class='parentBar-style' ><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='rq-gc-parentBar-leftCue'></div><div class='rq-gc-parentBar-middle'></div><div
class='rq-gc-parentBar-rightCue'></div></div>"
    ParentTaskEndTimeProperty="OverallEndTime"
    ParentTaskStartTimeProperty="OverallStartTime">
    <GanttTableOptions>
        <Columns>
            <GanttBase:Column field="Name" title="Name" clientTemplate
="flexyGanttNameColumnTemplate" clientEditorTemplate
="flexyGanttExpandableTextBoxEditor"></GanttBase:Column>
        </Columns>
    </GanttTableOptions>
</RQ:FlexyGantt>

<script id="flexyGanttNameColumnTemplate" type="text/x-jquery-tmpl">
    <div class="rq-grid-expand-indentWidth" style="height: 1px; width: ${
RadiantQ.Gantt.LevelToIndentWidth(data.Level(), data.IsParentType())}px"></div>
    <div style="width: 12px; display: ${data.IsParentType() ? "block" : "none"
} " class="arrowContainer">
        <div onclick="ExpanderOnClick(this,event)" id="arrow" class
="${data.HierarchicalItem.IsExpanded() ? "rq-grid-expand-arrow rq-grid-collapse-arrow"
: "rq-grid-expand-arrow"} rq-Ignore-click"></div></div>
        <div class="rq-grid-expander-text"> ${nameConverter(data)} </div>
    </script>
<script id="flexyGanttExpandableTextBoxEditor" type="text/x-jquery-tmpl">
    <div class="rq-grid-expand-indentWidth" style="height: 1px; width: ${
RadiantQ.Gantt.LevelToIndentWidth(data.Level(), data.IsParentType())}px"></div>
    <div style="width: 12px; display: ${data.IsParentType() ? "block" : "none"
} " class="arrowContainer">
        <div onclick="ExpanderOnClick(this,event)" id="Div3" class
="${data.HierarchicalItem.IsExpanded() ? "rq-grid-expand-arrow rq-grid-collapse-arrow"
: "rq-grid-expand-arrow"} rq-Ignore-click"></div></div>
        <div class="rq-grid-expander-text">
            <input data-bind="value: nameConverter" /></div>
    </script>

```

The gantt is now fully setup to display Projects returned from the ashx handler.

Converters Functions

Here are some converter functions that provide value to FlexyGantt from the bound objects.

```

<script type="text/javascript">
    //to resolve the hierarchical data source.
    function ResolverFunction(data) {
        // If data is wrapped by KO, then data itself could be a function and so we
        pick the object from the function.
        if ($.isFunction(data)) {
            data = data()[0];
        }
        if (data['PName'] != undefined) {
            if (data['Tasks'] != undefined) {
                if ($.isFunction(data['Tasks']))
                    return data['Tasks']();
            }
        }
    }

```

```

        else
            return data['Tasks'];
        }
        else
            // Return an empty array to keep this a collapsible parent with no
            children. Return null to make this a leaf node.
            return new Array();
        }
        return null;
    }
}

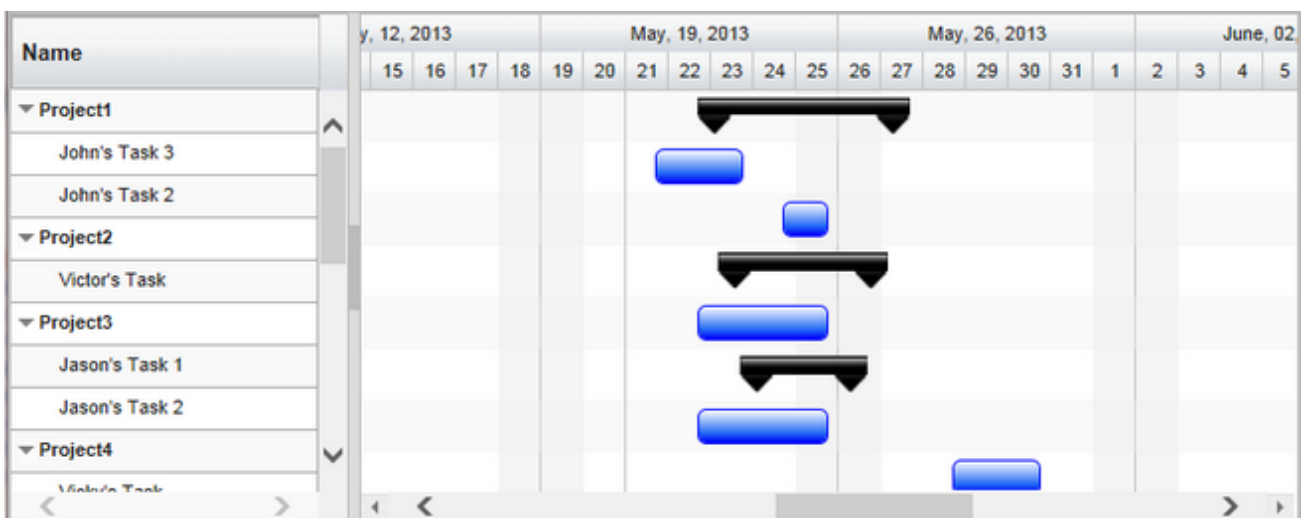
//To resolve the hierarchical data source.
nameConverter = function (flexyNodeData) {
    var data;
    // The grid calls this converter with flexyNodeData as a arg.
    if (flexyNodeData)
        data = flexyNodeData.Data();
    // The grid calls this converter with flexyNodeData as a datacontext.
    else
        data = this.data;
    if (data["PName"])
        return data["PName"];
    else if (data["TaskName"])
        return data["TaskName"];
    return;
}
}
</script>

```

The gantt is now fully setup to display tasks returned from the ashx handler.

Make WebForm1.aspx the "Start Page". You can do so by right-clicking on this file in Solution Explorer and selecting "Set as Start Page".

Here is the resultant gantt:



Sample In Browser

Finally, take a look at this [topic](#) that show how you can clean up the Src folder in your project to remove unnecessary files.

-0-

RadiantQ jQuery Gantt Package

In ASP.NET MVC

Create a new ASP.NET MVC project in Visual Studio:

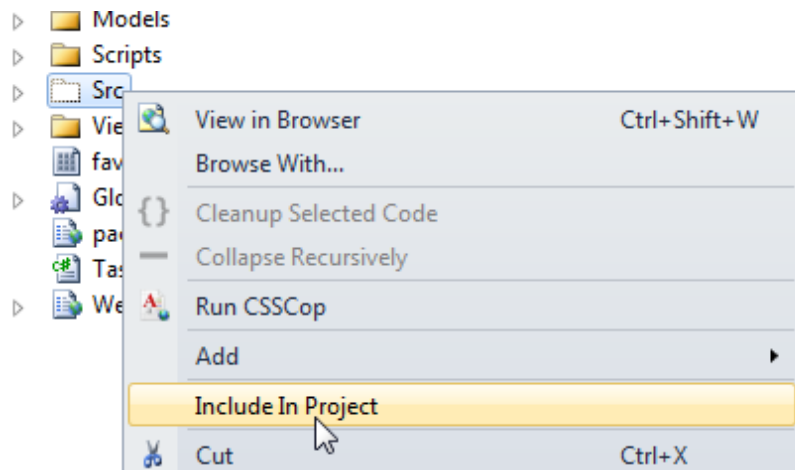
FILE --> New --> Project --> ASP.NET MVC 4 Web Application --> select Internet Application(Select Razor Engine)

1) Gantt Widget Source JS files

To begin with, you need the JS Source files that are required by the Gantt Widget. These files are in the <install path>/Src folder. Copy over this folder into the above Project folder (though this folder is very big in size, it contains the required CSS, etc for all Themes, locales, etc. and not all of them will be loaded inside your page).

Go ahead and remove the bin folder inside this Src folder.

Then in the project's Solution Explorer click on the "Show All Files" toolbar item to show this newly included Src folder and include that in the project.



include Src folder in project

2) Sample Utility JS files

Some JS files with utility functions that enables inline-editing in the grid, etc. are in the <install path>/Samples/Scripts folder. Copy over the contents of the Scripts folder from the above install path into the Scripts folder in your project folder (when you create a new project a Scripts folder would be automatically created in your project folder).

Then include the newly added Script files into your project following the same procedure as in the previous step.

3) Create a sample Data Source(JSON Data)

You will typically use an Entity Model, ADO.NET, etc. to retrieve data from a database. But, to keep things simple, we will create a simple list of "projects" and return it to the client from the server.

Create a new type called Project and Task. In Solution Explorer right click on the Project name, then Add --> New Item --> class (call it project.cs) and define a class like below.

```
public class Project
{
    public Project() { this.Tasks = new System.Collections.ObjectModel.
ObservableCollection<Task>(); }
    public string PName { get; set; }
    public System.Collections.ObjectModel.ObservableCollection<Task> Tasks { get;
set; }
    public DateTime OverallStartTime { get; set; }
    public DateTime OverallEndTime { get; set; }
}

public class Task
{
    public string Status;
    public string TaskName { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public double Progress { get; set; }
    public bool IsOverlapping { get; set; }
    public object Tag { get; set; }
}
```

Create a sample Data Source

Prepare a sample hierarchical list that will be bound to the Gantt. Prepare a sample list of the above project instances. This method must be inside the HomeController class
Controllers/HomeController.cs

```

public class HomeController : Controller
{
    public ActionResult GetFlexyGanttItemSource()
    {
        DateTime dtS = DateTime.Now;

        System.Collections.ObjectModel.ObservableCollection<Project> projects =
new System.Collections.ObjectModel.ObservableCollection<Project>();

        Project prj = new Project() { PName = "Project1", OverallStartTime = dtS
+ TimeSpan.FromDays(1), OverallEndTime = dtS + TimeSpan.FromDays(6) };
        prj.Tasks.Add(new Task() { StartTime = dtS, EndTime = dtS + TimeSpan
.FromDays(2), TaskName = "John's Task 3" });
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(3), EndTime
= dtS + TimeSpan.FromDays(4), TaskName = "John's Task 2" });
        projects.Add(prj);

        prj = new Project() { PName = "Project2", OverallStartTime = dtS +
TimeSpan.FromDays(1.5), OverallEndTime = dtS + TimeSpan.FromDays(5.5) };
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(1), EndTime
= dtS + TimeSpan.FromDays(4), TaskName = "Victor's Task" });
        projects.Add(prj);

        prj = new Project() { PName = "Project3", OverallStartTime = dtS +
TimeSpan.FromDays(2), OverallEndTime = dtS + TimeSpan.FromDays(5) };
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(1), EndTime
= dtS + TimeSpan.FromDays(4), TaskName = "Jason's Task 1" });
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(7), EndTime
= dtS + TimeSpan.FromDays(9), TaskName = "Jason's Task 2" });
        projects.Add(prj);

        prj = new Project() { PName = "Project4", OverallStartTime = dtS +
TimeSpan.FromDays(0.5), OverallEndTime = dtS + TimeSpan.FromDays(3.5) };
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(1.5),
EndTime = dtS + TimeSpan.FromDays(4), TaskName = "Vicky's Task" });
        projects.Add(prj);

        prj = new Project() { PName = "Project5", OverallStartTime = dtS +
TimeSpan.FromDays(2), OverallEndTime = dtS + TimeSpan.FromDays(6) };
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(2.2),
EndTime = dtS + TimeSpan.FromDays(3.8), TaskName = "Oleg's Task 1" });
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(5), EndTime
= dtS + TimeSpan.FromDays(6), TaskName = "Oleg's Task 2" });
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(8), EndTime
= dtS + TimeSpan.FromDays(9.6), TaskName = "Oleg's Task 3" });
        projects.Add(prj);

        prj = new Project() { PName = "Project6", OverallStartTime = dtS +
TimeSpan.FromDays(2.5), OverallEndTime = dtS + TimeSpan.FromDays(4.5) };
        prj.Tasks.Add(new Task() { StartTime = dtS + TimeSpan.FromDays(0.8),
EndTime = dtS + TimeSpan.FromDays(2), TaskName = "Kim's Task" });
        projects.Add(prj);

        var result = this.Json(projects, JsonRequestBehavior.AllowGet);
        return result;
    }
}

```

4) CSHTML file including the Gantt Widget

Create the .cshtml sample file

In the visual studio right click on the views folder and Add -> View (call it as view1.cshtml)

Add the *RadiantQ.Web.JQGantt.dll* to your project reference , you can find the dll here:
<install folder> \Src\bin\DotNET4MVC4\RadiantQ.Web.JQGantt.dll.

Include the following namespaces in Web.config

```
<pages>
  <namespaces>
    <add namespace="RadiantQMVC" />
    <add namespace="RadiantQ.Web.JQGantt" />
    <add namespace="RadiantQ.Web.JQGantt.Common" />
  </namespaces>
</pages>
```

And Include this Following namespace in your cshtml page.

```
@using RadiantQMVC
@using RadiantQ.Web.JQGantt;
@using RadiantQ.Web.JQGantt.Common
```

Include the following script reference. Make sure to link to the right version of the jQuery Gantt Package source in the last but one reference below.

```
<script src="~/Src/Scripts/jquery-1.11.2.min.js"></script>
<link id="themeChooser" href
="~/Src/Styles/jquery-ui-themes/smoothness/jquery-ui.css" rel="stylesheet" />
<link id="default" href="~/Src/Styles/radiantq.gantt.default.css" rel
="stylesheet" />
<link id="gridCss" href="~/Src/Styles/VW.Grid.css" rel="stylesheet" />
<script src="~/Src/Scripts/jquery-ui-1.11.4/jquery-ui.min.js"></script>
<script type="text/javascript" src="~/Src/Scripts/jquery.layout-latest.min.js"></
script>
<script src="~/Src/Scripts/Utils/date.js"></script>
<script src="~/Src/ResourceStrings/en-US.js"></script>
<script src='~/Src/Scripts/VW.Grid.1.min.js' type='text/javascript'></script>
<script src='~/Src/Scripts/RadiantQ-jQuery.Gantt.5.0.trial.min.js' type
='text/javascript'></script>
<script src='~/Src/Scripts/RQGantt_Init.min.js' type='text/javascript'></script>
```

Make sure you are not including the jQuery file in _Layout.cshtml, by default the jQuery file including line look likes this in _Layout.cshtml,

```
@Scripts.Render("~/bundles/jquery")
```

Creating FlexyGantt

Now include code to retrieve the json file you created above and then initialize the FlexyGantt widget binding it with the loaded data.

```
@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        ControlId = "ganttt_container",
```

```

        DataSourceUrl = new Uri("/Home/GetFlexyGanttItemSource", UriKind
.RelativeOrAbsolute),
        Options = new FlexyGanttOptions()
        {
            HierarchyResolverFunction = "ResolverFunction",
            ParentTaskStartTimeProperty = "OverallStartTime",
            ParentTaskEndTimeProperty = "OverallEndTime",
            TaskStartTimeProperty= "StartTime",
            TaskEndTimeProperty= "EndTime",
            TaskItemTemplate = "<div class='rq-gc-taskbar'><div
class='rq-taskbar-dragThumb'></div><div class='start-resizeThumb'></div><div
class='rq-taskbar-resizeThumb'></div></div>",
            ParentTaskItemTemplate = "<div class='parentBar-style'><div
class='rq-taskbar-dragThumb'></div><div class='start-resizeThumb'></div><div
class='rq-taskbar-resizeThumb'></div><div class='rq-gc-parentBar-leftCue'></div><div
class='rq-gc-parentBar-middle'></div><div
class='rq-gc-parentBar-rightCue'></div></div>",
            GanttChartOptions = new GanttChartOptions()
            {
                AnchorTime = DateTime.Today
            }
        }
    }
)

<!-- Div that will be transformed into the FlexyGantt widget above.-->
<div id="gantt_container" style="height: 450px;">
</div>

```

Initializing the Gantt Table

Now you have to setup the different column you want to show in the GanttTable. You can do so by defining GanttTableOptions

```

@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        ControlId = "gantt_container",
        DataSourceUrl = new Uri("/Home/GetFlexyGanttItemSource", UriKind
.RelativeOrAbsolute),
        Options = new FlexyGanttOptions()
        {
            GanttTableOptions = new GanttTableOptions()
            {
                Columns = new Columns(){
                    new Column(){
                        field = "Name",
                        title = "Name",
                        width = 110,
                        clientEditorTemplate="flexyGantNameEditor",
                        clientTemplate = "flexyGanttNameColumnTemplate"
                    }
                }
            },
            HierarchyResolverFunction = "ResolverFunction",
            TaskTooltipTemplateID="TaskTooltipTemplate",
            ParentTaskStartTimeProperty = "OverallStartTime",
            ParentTaskEndTimeProperty = "OverallEndTime",

```

```

        TaskStartTimeProperty= "StartTime",
        TaskEndTimeProperty= "EndTime",
        TaskItemTemplate = "<div class='rq-gc-taskbar'><div
class='rq-taskbar-dragThumb'></div><div class='start-resizeThumb'></div><div
class='rq-taskbar-resizeThumb'></div></div>",
        ParentTaskItemTemplate = "<div class='parentBar-style'><div
class='rq-taskbar-dragThumb'></div><div class='start-resizeThumb'></div><div
class='rq-taskbar-resizeThumb'></div><div class='rq-gc-parentBar-leftCue'></div><div
class='rq-gc-parentBar-middle'></div><div
class='rq-gc-parentBar-rightCue'></div></div>",
        GanttChartOptions = new GanttChartOptions()
        {
            AnchorTime = DateTime.Today
        }
    }
}
)

```

Converters Functions

Here are some converter functions that provide value to FlexyGantt from the bound objects.

```

<script type="text/javascript">
    //to resolve the hierarchical data source.
    function ResolverFunction(data) {
        // If data is wrapped by KO, then data itself could be a function and so we
        pick the object from the function.
        if ($.isFunction(data)) {
            data = data()[0];
        }
        if (data['PName'] != undefined) {
            if (data['Tasks'] != undefined) {
                if ($.isFunction(data['Tasks']))
                    return data['Tasks']();
                else
                    return data['Tasks'];
            }
            else
                // Return an empty array to keep this a collapsible parent with no
                children. Return null to make this a leaf node.
                return new Array();
        }
        return null;
    }

    //To resolve the hierarchical data source.
    nameConverter = function (flexyNodeData) {
        var data;
        // The grid calls this converter with flexyNodeData as a arg.
        if (flexyNodeData)
            data = flexyNodeData.Data();
        // The grid calls this converter with flexyNodeData as a datacontext.
        else
            data = this.data;
        if (data["PName"])
            return data["PName"];
        else if (data["TaskName"])
            return data["TaskName"];
        return;
    }
</script>

```

The gantt is now fully setup to display tasks returned from the controller .

Make view1.cshtml as "Start Page".

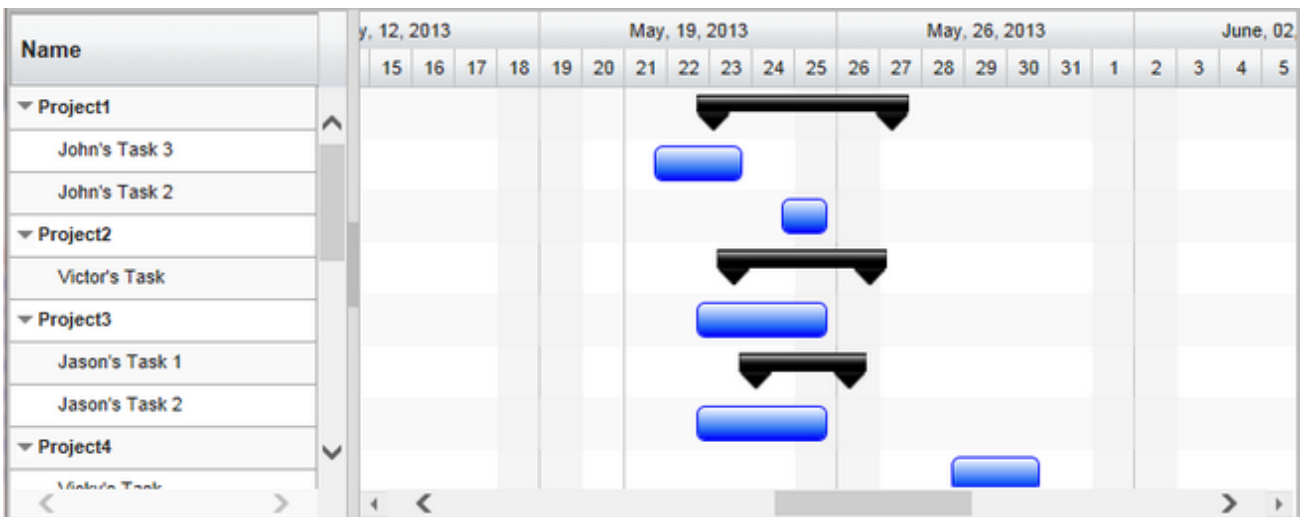
create a ActionResult in Homecontroller that will return your page as view

```
public ActionResult View1()
{
    return View("View1");
}
```

In App_Start/RouteConfig.cs change the routes action to "View1"

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "View1", id = UrlParameter.Optional
}
);
```

Here is the resultant gantt:



Sample In Browser

Finally, take a look at this [topic](#) that show how you can clean up the Src folder in your project to remove unnecessary files.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

In PHP

Your First Gantt

Let us start with creating a new project directory, for example MyFirstGantt.

1) Gantt Widget Source JS files

To begin with, you need the JS Source files that are required by the Gantt Widget. These files are in the <install path>/Src folder. Simply copy over the whole Src folder into the above sample directory. This folder also has other dependant css files. You can delete the "Src/bin" folder as that's not required for this PHP sample.

Then copy over the PHP library files which are inside the <install path>/PlatformSamples/PHPSamples/lib folder into the sample directory.

2) Sample Utility JS files

Some JS files with utility functions that enables inline-editing in the grid, etc. are in the <install path>/Samples/Scripts folder. Copy over that entire Scripts directory as well into MyFirstGantt.

3) Sample JSON Data

Create a SampleData.json file containing a list of sample tasks to be displayed in the gantt.

SampleData.json content:

```
[{
  "TName" : "Team1",
  "PStartTime" : "2012-04-02T00:00:00Z",
  "PEndTime" : "2012-04-18T00:00:00Z",
  "Resources" : [{
    "RName" : "JohnH",
    "PStartTime" : "2012-04-04T00:00:00Z",
    "PEndTime" : "2012-04-15T00:00:00Z",
    "Tasks" : [{
      "TaskName" : "Task 1",
      "StartTime" : "2012-04-03T00:00:00Z",
      "EndTime" : "2012-04-12T00:00:00Z",
      "Progress" : 20
    }]
  }],
},
{
  "RName" : "VictorG",
  "PStartTime" : "2012-04-06T00:00:00Z",
  "PEndTime" : "2012-04-18T00:00:00Z",
  "Tasks" : [{
    "TaskName" : "Task 1",
    "StartTime" : "2012-04-03T00:00:00Z",
    "EndTime" : "2012-04-18T00:00:00Z",
    "Progress" : 20
  }]
},
{
  "RName" : "JasonS",
  "PStartTime" : "2012-04-06T00:00:00Z",
  "PEndTime" : "2012-04-18T00:00:00Z",
  "Tasks" : [{
    "TaskName" : "Task 1",
    "StartTime" : "2012-04-06T00:00:00Z",
    "EndTime" : "2012-04-12T00:00:00Z",
    "Progress" : 20
  }],
  {
    "TaskName" : "Task 2",
    "StartTime" : "2012-04-12T00:00:00Z",
    "EndTime" : "2012-04-18T00:00:00Z",
    "Progress" : 70
  }
}]
},
{
  "TName" : "Team2",
  "PStartTime" : "2012-04-10T00:00:00Z",
  "PEndTime" : "2012-04-20T00:00:00Z",
  "Resources" : [{
    "RName" : "BalajiN",
    "PStartTime" : "2012-04-08T00:00:00Z",
    "PEndTime" : "2012-04-18T00:00:00Z",
    "Tasks" : [{
      "TaskName" : "Task 1",
      "StartTime" : "2012-04-08T00:00:00Z",
      "EndTime" : "2012-04-20T00:00:00Z",
      "Progress" : 20
    }]
  }],
},
{
  "RName" : "LiM",
  "PStartTime" : "2012-04-12T00:00:00Z",
  "PEndTime" : "2012-04-18T00:00:00Z",
  "Tasks" : [{
    "TaskName" : "Task 1",
    "StartTime" : "2012-04-08T00:00:00Z",
```

4) PHP file including the Gantt Widget

Create a new PHP file inside the project directory (MyFirstGantt) and reference the following source files. Remember to link to the right version of the RadiantQ jQuery Gantt in the last reference below.

```
<!DOCTYPE html>
  <link href="Src/Styles/jquery-ui-themes/smoothness/jquery-ui.css" rel
="stylesheet" />
  <link href="Src/Styles/radiantq.gantt.default.css" rel="stylesheet" />
  <link href="Src/Styles/VW.Grid.css" rel="stylesheet" />
  <script src="Src/Scripts/jquery-1.11.2.min.js" type="text/javascript"></script>
  <script src="Src/Scripts/jquery-ui-1.11.4/jquery-ui.min.js" type
="text/javascript"></script>
  <script type="text/javascript" src="Src/Scripts/jquery.layout-latest.min.js"></
script>
  <script src="Src/Scripts/Utils/date.js" type="text/javascript"></script>
  <script src="Src/ResourceStrings/en-US.js" type="text/javascript"></script>
  <script src='Src/Scripts/VW.Grid.1.min.js' type='text/javascript'></script>
  <script src='Src/Scripts/RadiantQ-jQuery.gantt.5.0.trial.min.js' type
='text/javascript'></script>>
  <script src="/Samples/Src/Scripts/RQGantt_Init.min.js" type="text/javascript"></
script>
  <!-- It automatically includes required PHP gantt extension files-->
  require_once "lib/AutoLoad.php"
```

Creating Gantt And Gantt Table

Initialize the GanttControl widget and Gantt Table as follows

```

<?php
//Gantt Settings
$gantSettings = new RadiantQ\Gantt\GanttSettings();

//GanttControl Options.
$options= new RadiantQ\Gantt\FlexyGanttOptions();
$gantSettings->DataSourceUrl = "FlexyGanttSkeleton.json";

//column defintions
$NameColumn = new RadiantQ\Gantt\Column();
$NameColumn->field="Name";
$NameColumn->title= "Name";
$NameColumn->clientEditorTemplate= "flexyGantNameEditor";
$NameColumn->clientTemplate = "flexyGanttNameColumnTemplate";

$GanttTableOptions->columns = array($NameColumn);

$options->GanttTableOptions = $GanttTableOptions;
$options->HierarchyResolverFunction = "ResolverFunction";
$options->TaskTooltipTemplateID = "TaskTooltipTemplate";
$options->TaskStartTimeProperty = "StartTime";
$options->TasksListProperty = "Tasks";
$options->ParentTaskStartTimeProperty = "PStartTime";
$options->TaskItemTemplate = "<div class='rq-gc-taskbar'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div></div>"
;
$options->ParentTaskItemTemplate = "<div class='parentBar-style'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='rq-gc-parentBar-leftCue'></div><div class='rq-gc-parentBar-middle'></div><div
class='rq-gc-parentBar-rightCue'></div></div>";
$options->TaskEndTimeProperty = "EndTime";
$options->ParentTaskEndTimeProperty = "PEndTime";

//GanttChart options.
$GanttChartOptions = new RadiantQ\Gantt\GanttChartOptions();
$GanttChartOptions->AnchorTime = new DateTime("2014-04-03");
$GanttChartOptions->TimeIndicatorLineOption = 1;
$options->GanttChartOptions = $GanttChartOptions;
$gantSettings->Options = $options;
//Ganttcontrol initialization.
$gant = new RadiantQ\UI\FlexyGantt($gantSettings);
//Setting attributes to Ganttcontrol elements.
$gant->setStyleAttribute("width","100%")->setStyleAttribute("height","100%"
)->setAttribute("id","gant_container");
//To render the FlexyGantt.
echo $gant->Render();
?>
//Name template and editor with expander cue.
<script id="flexyGanttNameColumnTemplate" type="text/x-jquery-tmpl">
    <div class="rq-grid-expand-indentWidth" style="height: 1px; width: ${RadiantQ
.Gantt.LevelToIndentWidth(data.Level(), data.IsParentType())}px"></div>
    <div style="width: 12px; display: ${data.IsParentType() ? "block" : "none" }"
class="arrowContainer">
        <div onclick="ExpanderOnClick(this,event)" id="arrow" class="${
data.HierarchicalItem.IsExpanded() ? "rq-grid-expand-arrow rq-grid-collapse-arrow":
"rq-grid-expand-arrow"} rq-Ignore-click"></div>
        </div>
        <div class="rq-grid-expander-text">${nameConverter(data)}</div>
</script>

<script id="flexyGantNameEditor" type="text/x-jquery-tmpl">
    <div class="rq-grid-expand-indentWidth" style="height: 1px; width: ${RadiantQ
.Gantt.LevelToIndentWidth(data.Level(), data.IsParentType())}px"></div>
    <div style="width: 12px; display: ${data.IsParentType() ?

```


Script

```

// to get the name from the bounded list
function nameConverter(flexyNodeData, value) {
    var data;
    // The grid calls this converter with flexyNodeData as a arg.
    if (flexyNodeData.Data)
        data = flexyNodeData.Data();
        // The grid calls this converter with flexyNodeData as a datacontext.
    else
        data = flexyNodeData;
    if (value == undefined) {
        if (data["TName"] != undefined)
            return data["TName"];
        else if (data["RName"] != undefined)
            return data["RName"];
        else if (data["TaskName"] != undefined)
            return data["TaskName"];
    }
    else {
        if (data["TName"] != undefined)
            data["TName"] = value;
        else if (data["RName"] != undefined)
            data["RName"] = value;
        else if (data["TaskName"] != undefined)
            data["TaskName"] = value;
    }
    return;
};

//to resolve the hierarchical data source.
function ResolverFunction(data) {
    // If data is wrapped by KO, then data itself could be a function and so we pick
    the object from the function.
    if ($.isFunction(data)) {
        data = data()[0];
    }

    if (data["Resources"] != undefined) {
        if ($.isFunction(data["Resources"]))
            return data["Resources"]();
        else
            return data["Resources"];
    }
    else if (data["RName"] != undefined) {
        if (data["Tasks"] != undefined) {
            return null;
        }
        else
            // Return an empty array to keep this a collapsible parent with no
            children. Return null to make this a leaf node.
            return new Array();
    }
    return null;
}

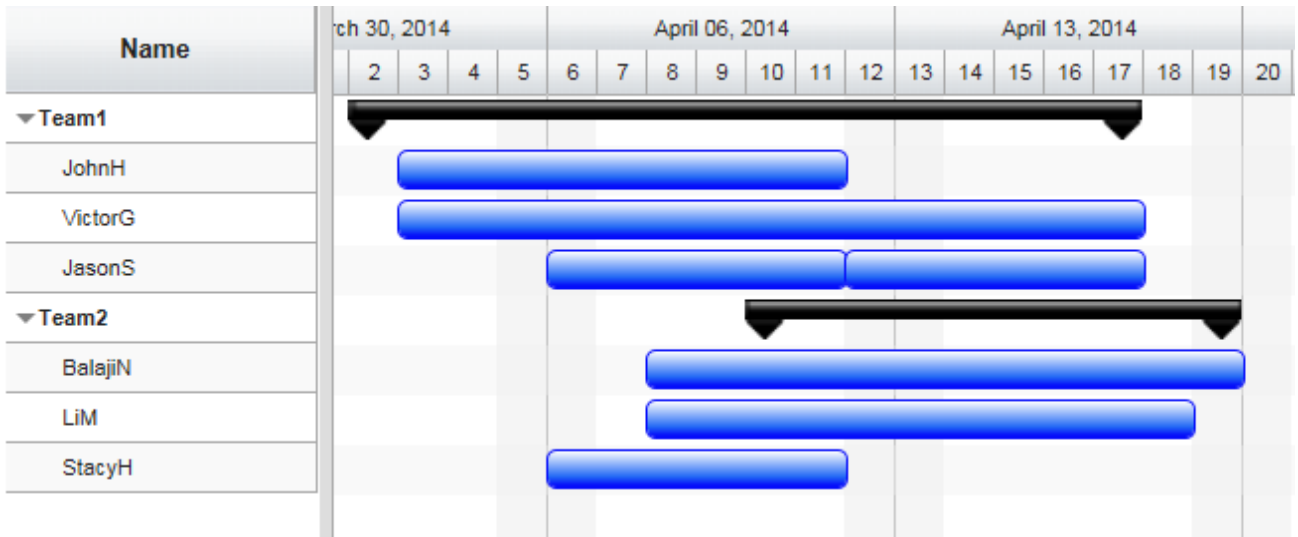
```

The content of your MyFirstGantt directory should look like this:

- Src
- Scripts
- json file

- PHP file

Open the FirstGanttSample.PHP in the browser and you should see this in the browser:



FirstGanttSample in browser

Finally, take a look at this [topic](#) that show how you can clean up the Src folder in your project to remove unnecessary files.

© RadiantQ 2009-2018. All Rights Reserved.

-o-

RadiantQ jQuery Gantt Package

In Typescript

Create a new ASP.NET project in Visual Studio:

VS 2012 : FILE --> New --> Project --> Installed--> Templates --> Other Language --> TypeScript, create a project.

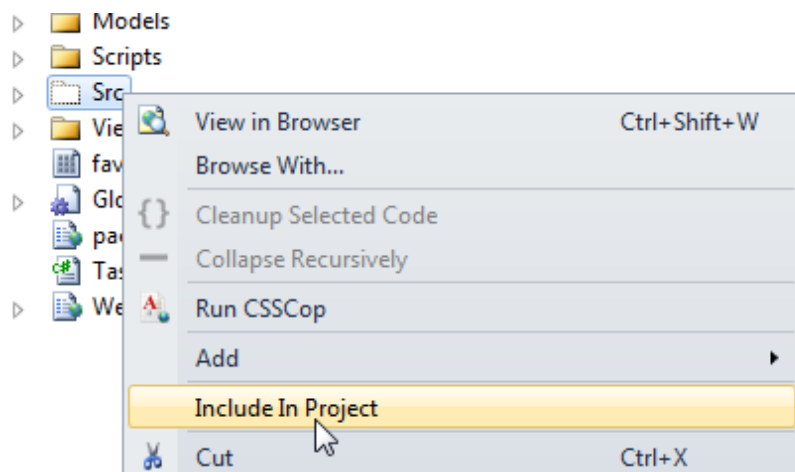
The Gantt Package includes the necessary Gantt TypeScript interfaces to help you develop your Web application just like any other Type-Safe language with compile time checks.

1) Gantt Widget Source JS files

To begin with, you need the JS Source files that are required by the Gantt Widget. These files are in the <install path>/Src folder. Copy over this folder into the above Project folder (though this folder is very big in size, it contains the required CSS, etc for all Themes, locales, etc. and not all of them will be loaded inside your page).

Go ahead and remove the bin folder inside this Src folder.

Then in the project's Solution Explorer click on the "Show All Files" toolbar item to show this newly included Src folder and include that in the project.



include Src folder in project

2) Sample Utility JS files

Some JS files with utility functions that enables inline-editing in the grid, etc. are in the <install path>/Samples/Scripts folder. Copy over the contents of the Scripts folder from the above install path into the Scripts folder in your project folder (when you create a new project a Scripts folder would be automatically created in your project folder).

Then include the newly added Script files into your project following the same procedure as in the previous step.

3) Sample JSON Data

Create a SampleData.json file containing a list of sample tasks to be displayed in the gantt.

SampleData.json content:

```
[{
  "TName" : "Team1",
  "PStartTime" : "2012-04-02T00:00:00Z",
  "PEndTime" : "2012-04-18T00:00:00Z",
  "Resources" : [{
    "RName" : "JohnH",
    "PStartTime" : "2012-04-04T00:00:00Z",
    "PEndTime" : "2012-04-15T00:00:00Z",
    "Tasks" : [{
      "TaskName" : "Task 1",
      "StartTime" : "2012-04-03T00:00:00Z",
      "EndTime" : "2012-04-12T00:00:00Z",
      "Progress" : 20
    }]
  }],
},
{
  "RName" : "VictorG",
  "PStartTime" : "2012-04-06T00:00:00Z",
  "PEndTime" : "2012-04-18T00:00:00Z",
  "Tasks" : [{
    "TaskName" : "Task 1",
    "StartTime" : "2012-04-03T00:00:00Z",
    "EndTime" : "2012-04-18T00:00:00Z",
    "Progress" : 20
  }]
},
{
  "RName" : "JasonS",
  "PStartTime" : "2012-04-06T00:00:00Z",
  "PEndTime" : "2012-04-18T00:00:00Z",
  "Tasks" : [{
    "TaskName" : "Task 1",
    "StartTime" : "2012-04-06T00:00:00Z",
    "EndTime" : "2012-04-12T00:00:00Z",
    "Progress" : 20
  }],
  {
    "TaskName" : "Task 2",
    "StartTime" : "2012-04-12T00:00:00Z",
    "EndTime" : "2012-04-18T00:00:00Z",
    "Progress" : 70
  }
}]
},
{
  "TName" : "Team2",
  "PStartTime" : "2012-04-10T00:00:00Z",
  "PEndTime" : "2012-04-20T00:00:00Z",
  "Resources" : [{
    "RName" : "BalajiN",
    "PStartTime" : "2012-04-08T00:00:00Z",
    "PEndTime" : "2012-04-18T00:00:00Z",
    "Tasks" : [{
      "TaskName" : "Task 1",
      "StartTime" : "2012-04-08T00:00:00Z",
      "EndTime" : "2012-04-20T00:00:00Z",
      "Progress" : 20
    }]
  }],
},
{
  "RName" : "LiM",
  "PStartTime" : "2012-04-12T00:00:00Z",
  "PEndTime" : "2012-04-18T00:00:00Z",
  "Tasks" : [{
    "TaskName" : "Task 1",
    "StartTime" : "2012-04-08T00:00:00Z",
```

4) HTML file including the Gantt Widget

Create a new HTML file in your project and reference the following source files. Remember to link to the right version of the RadiantQ jQuery Gantt in the last reference below.

```
<head>
  <title></title>
  <link href="Src/Styles/jquery-ui-themes/smoothness/jquery-ui.css" rel
="stylesheet" />
  <link href="Src/Styles/radiantq.gantt.default.css" rel="stylesheet" />
  <link href="Src/Styles/VW.Grid.css" rel="stylesheet" />
  <script src="Src/Scripts/jquery-1.11.2.min.js" type="text/javascript"></script>
  <script src="Src/Scripts/jquery-ui-1.11.4/jquery-ui.min.js" type
="text/javascript"></script>
  <script type="text/javascript" src="Src/Scripts/jquery.layout-latest.min.js"></
script>
  <script src="Src/Scripts/Utils/date.js" type="text/javascript"></script>
  <script src="Src/ResourceStrings/en-US.js" type="text/javascript"></script>
  <script src='Src/Scripts/VW.Grid.1.min.js' type='text/javascript'></script>
  <script src='Src/Scripts/RadiantQ-jQuery.gantt.5.0.trial.min.js' type
='text/javascript'></script>
</head>

<body>
  <div id="pagecontent" style="height: 600px;">
    <div id="ganttt_container" style="height: 100%;">
      </div>
    </div>

    <!--The template for the custom tooltip. -->
    <script id="TaskTooltipTemplate" type="text/x-jquery-tmpl">
      <div class='TaskTooltip'>
        <div align='center'>Name: ${nameConverter()}</div>
        <div>
          <span > StartTime : ${startTimeTooltipConverter()}
          </span></br>
          <span > EndTime : ${endTimeTooltipConverter()}
          </span>
        </div>
      </div>
    </script>
</body>
```

5) TypeScript file.

Add a new TypeScript file(myApp.ts) next to that HTML in your project and refer the resultant js in html.

```
<head>
  other script fils.
  <script src=myApp.js type='text/javascript'></script>
</head>
```

6) Create Ganttcontrol widget inside the TypeScript file.

```

$.ajax({
  type: "GET",
  dataType: 'json',
  url: 'GanttControlSkeleton.json',
  converters:
  {
    "text json": function (data) {
      //console.log(data);
      return $.parseJSON(data, true
      /*converts date strings to date objects*/
      , true
      /*converts ISO dates to local dates*/
      );
    }
  },
  success: function (data) {
    loadGantt(data);
  }
});
function loadGantt(datasourcejson) {
  var anchorTime = self.jsonData[0].PStartTime.clone();
  var columns = [
    {
      field: "Name",
      title: "Name",
      editor:
RadiantQ.Default.Template.FlexyGanttExpandableTextBoxEditor("FlexyGanttSkeleton_nameC
onverter"),
      template:
RadiantQ.Default.Template.FlexyGanttExpandableTextBlockTemplate("FlexyGanttSkeleton_n
ameConverter")
    }
  ];
  // The template that defines the look for the task bars. "rq-gc-taskbar" is a
  // built-in style that defines a default look for the task bars.
  var tTemplate = "<div class='rq-gc-taskbar'><div
  class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div></div>"
  ;
  var $ganttt_container = $("#ganttt_container");
  // Initialize the FlexyGantt widget.
  $ganttt_container.FlexyGantt({
    DataSource: self.jsonData,
    //the FlexyGantt is bound to resolve the hierarchy of Team/Resources/Tasks.
    resolverFunction: function (data) {
      // If data is wrapped by KO, then data itself could be a function and so we
      // pick the object from the function.
      if ($.isFunction(data)) {
        data = data()[0];
      }

      if (data["Resources"] != undefined) {
        if ($.isFunction(data["Resources"]))
          return data["Resources"]();
        else
          return data["Resources"];
      }
      else if (data["RName"] != undefined) {
        if (data["Tasks"] != undefined) {
          return null;
        }
        else
          return new Array();
      }
      return null;
    },
    GanttTableOptions: {
      columns: columns
    },
  },

```

Finally, take a look at this [topic](#) that show how you can clean up the Src folder in your project to remove unnecessary files.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

FlexyGantt Basics

DataBinding

RadiantQ jQuery Gantt Package

XML Data

Using XML Data

XML is another common way web services expose data to the client. Here is some code that shows how the XML data retrieved has to be "massaged" before binding it to the gantt.

In HTML

```
$.ajax({
  type: "GET",
  dataType: 'xml text',
  url: 'FlexyGanttData.xml',
  converters:
  {
    "xml text": function (data) {
      // We use te xml2json jquery plugin to convert xml to json.
      var json = $.xml2json(data).task;
      return $.parseJSON(window.JSON.stringify(json), true
      /*converts date strings to date objects*/
      , true
      /*converts ISO dates to local dates*/
      );
    }
  },
  success: function (data) {
    self.jsonData = data;
    $.holdReady(false);
  }
});
```

And binding the above json list to the Gantt,

In HTML

```

$gantt_container.FlexyGantt({
    DataSource: self.jsonData,
    //the FlexyGantt is bound to resolve the hierarchy of
Team/Resources/Tasks.
    resolverFunction: function (data) {
        // If data is wrapped by KO, then data itself could be a function
and so we pick the object from the function.
        if (data["resource"] != undefined) {
            // While converting XML to JSON, if there is only one
resource in the team then the xml2json utility funciton will make the resource a
child of team object. Here we are
            // converting it to an array with 1 resource.
            if ($.isArray(data["resource"]) == false) {
                var res = data["resource"];
                data["resource"] = new Array(res);
            }
            return data["resource"];
        }
        else if (data["RName"] != undefined) {
            // While converting XML to JSON, if there is only one task in
the resource then the xml2json utility funciton will make the resource a child of
team object. Here we are
            // converting it to an array with 1 task.
            if (data["task"] != undefined) {
                if ($.isArray(data["task"]) == false) {
                    var task = data["task"];
                    data["task"] = new Array(task);
                }
                return null;
            }
            else
                // Return an empty array to keep this a collapsible parent
with no children. Return null to make this a leaf node.
                return new Array();
        }
        return null;
    },
    ....
    ....
});

```

In ASP.NET MVC

```

var AjaxSettings = {
  dataType: 'xml text',
  converters:
  {
    "xml text": function (data) {
      // We use te xml2json jquery plugin to convert xml to json.
      var json = $.xml2json(data).Project;
      for (var i = 0; i < json.length; i++) {
        var project = json[i];
        var task = project['ArrayOfTask']['Task']
        if (task) {
          if (task.length == undefined)
            json[i]["Tasks"] = new Array(json[i]['ArrayOfTask']['Task']);
          else
            json[i]["Tasks"] = json[i]['ArrayOfTask']['Task'];
        }
      }
      return $.parseJSON(window.JSON.stringify(json), true
        /*converts date strings to date objects*/
        , true
        /*converts ISO dates to local dates*/
        );
    }
  }
};

```

And pass the AjaxSettings to FlexyGantt,

In ASP.NET MVC

```

@Html.JQFlexyGantt(
  new JQFlexyGanttSettings()
  {
    ControlId = "ganttt_container",
    AfterGanttWidgetInitializedCallback = "AfterGanttWidgetInitializedCallback",
    DataSourceUrl = new Uri("/Home/GetFlexyGanttSkeletonXMLDataSource", UriKind
.RelativeOrAbsolute),
    AjaxSettings = "AjaxSettings",
    Options = new FlexyGanttOptions()
    {
      HierarchyResolverFunction = "ResolverFunction",
      ...
    }
  }
)

```

In ASP.NET

```

var AjaxSettings = {
  dataType: 'xml text',
  converters:
  {
    "xml text": function (data) {
      // We use te xml2json jquery plugin to convert xml to json.
      var json = $.xml2json(data).Project;
      for (var i = 0; i < json.length; i++) {
        var project = json[i];
        var task = project['ArrayOfTask']['Task']
        if (task) {
          if (task.length == undefined)
            json[i]["Tasks"] = new Array(json[i]['ArrayOfTask']['Task']);
          else
            json[i]["Tasks"] = json[i]['ArrayOfTask']['Task'];
        }
      }
      return $.parseJSON(window.JSON.stringify(json), true
        /*converts date strings to date objects*/
        , true
        /*converts ISO dates to local dates*/
        );
    }
  }
};

```

And pass the AjaxSettings to FlexyGantt,

In ASP.NET

```

<RQ:FlexyGantt ID="gantt" ParentTaskEndTimeProperty="OverallEndTime"
AfterGanttWidgetInitializedCallback="AfterGanttWidgetInitializedCallback"
ParentTaskStartTimeProperty="OverallStartTime" TaskEndTimeProperty="EndTime"
TaskStartTimeProperty="StartTime" AjaxSettings="AjaxSettings" .. />

```

>>>>

NOTE: If you are using namespaces in your XML (for example, <ns:StartTime>...</ns:StartTime>), then the above xml2json works slightly differently between browsers.

IE: In IE, the above element is referenced in the containing parent object with a property named "ns:StartTime".

Other browsers: The above element is referenced with a property named "StartTime".

To workaroud this issue, you can determine what the property name would be by probing into your first task list item as follows:

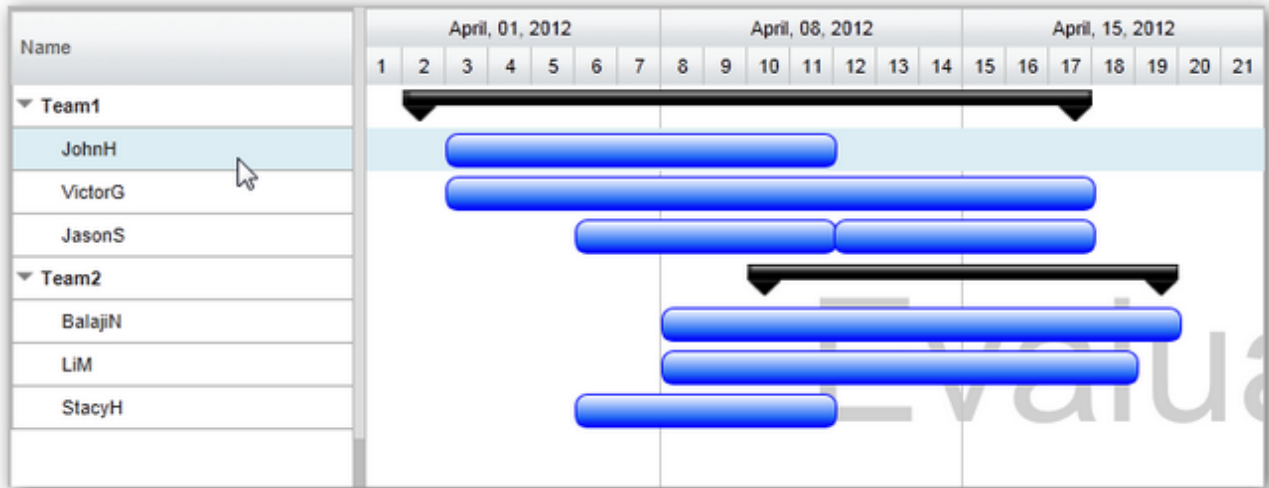
```

// For eg: Using the namespace "ns" in xml,
<task TaskName="Task 1" ns:StartTime="2012-04-03T00:00:00Z" EndTime="
2012-04-12T00:00:00Z" Progress="20"></task>
// In Firefox the property ends up being StartTime and in IE it's ns:StartTime;
var taskStartTimeProperty = self.jsonData[0]["ns:StartTime"] ? "ns:StartTime" :
"StartTime";

```

>>>>

Here is the resultant gantt:



This is illustrated in this samples:

- In HTML : ..\Samples\FlexyGanttBoundToXML.htm.
- In ASP.NET MVC : ..\Views\Home\FlexyGanttBoundToXML.cshtml.
- In ASP.NET : ..\Samples\ProjectGantt\FlexyGanttBoundToXML.aspx.

RadiantQ jQuery Gantt Package

Using RadiantQ Binding

Why RadiantQ Binding?

RadiantQ Provides a highly customizable Binding engine which let you to bind the model, it improves performance of the Gantt, while comparing to other Binding engine with gantt and it does not require any model changes.

You will need data bindings only if you want some portions of the gantt to dynamically update it's UI when the underlying value changes. This is how dynamic updates are possible / setup in the grid and chart area of the gantt, in the absence of RadiantQ Binding:

Dynamic UI in the Grid Area

The grid cells as setup in our samples use our internal binding support to automatically listen to changes in the underlying activity's values and update itself. So, there is no need for bindings here.

Dynamic UI in the Chart Area

Take a look at the different ways to customize the look and feel of the task bars [here](#) using templates. While for the most part the templates allow you to customize the look based on the underlying activity's property values, they don't automatically redraw the bars when a value affecting the look of the bars change. Except by forcing a redraw of a bar manually.

Model Changes.

The Bound property should trigger the property changes event when it's value changes. Here is an example(if you are creating the model in project.)

```
function Employee() {
    //the argument is optional.
    this.PropertyChanged = new ObjectEvent("PropertyChanged");

    var endTime = new Date();
    Object.defineProperty(this, "EndTime",
    {
        get: function () { return endTime; },
        set: function (newVal) {
            endTime = newVal;
            // rise the PropertyChanged, in setter, to notify the property changes in
bindings.
            this.PropertyChanged.raise(this, {
                PropertyName: "EndTime",
                value: endTime
            });
        }
    });
});

//.. other property definition.
}
```

This is not a most common case, the data may come from different server or service in json format. In this case you don't have to create a brand new model for data to trigger the property changes instead of you can inject the get/set in data using the RadiantQ.Gantt.Utills.InjectGetAndSetOnData utility. Here is an example for that.

```
$.ajax({
  type: "GET",
  dataType: 'json',
  url: 'FlexyGanttCustomTaskLook.json',
  converters:
    {
      "text json": function (data) {
        //arg[1] can convert Iso date string to JS date object
        //arg[2] can convert Iso date to local date by default is true.
        return $.parseJSON(data, true /*converts date strings to date
objects*/, true /*converts ISO dates to local dates*/, function (key, value) {

          if (key == "EndTime") {
            //To inject get and set to trigger the property chnaged.
            RadiantQ.Gantt.Utills.InjectGetAndSetOnData(this, key);
            return value;
          }
          return value;
        });
      }
    },
  success: function (data) {
    //data - source for the FlexyGantt.
  }
});
```

View Changes.

Binding should be specified using the data-bind attribute.

```
//TaskColor - The name of the custom binder.
//EndTime - The name of the bound property.
var taskTemplate = '<div class="rq-gc-taskbar" data-bind="TaskColor: EndTime"><div
class="rq-taskbar-dragThumb"></div><div class="rq-taskbar-resizeThumb"></div><div
class="lable" data-bind="TaskColor : EndTime"></div></div>';

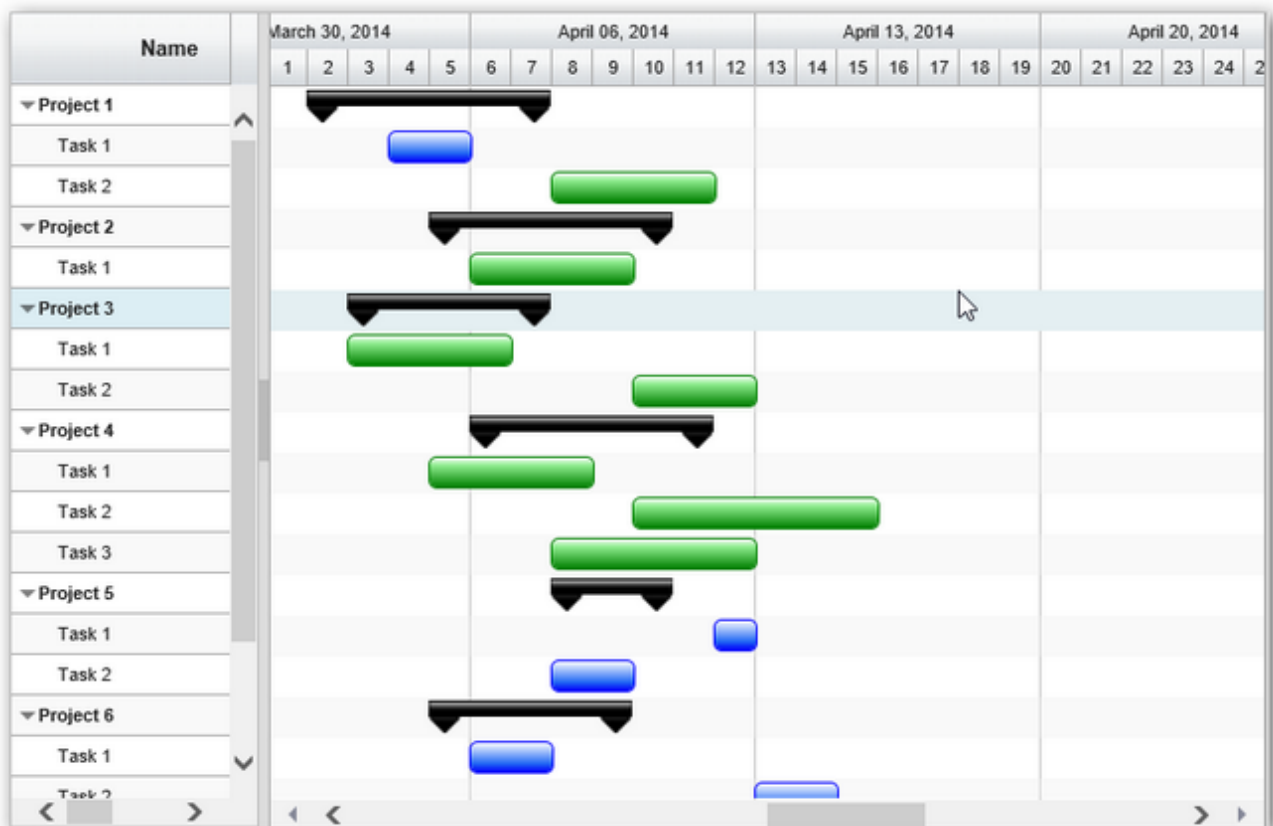
// Initialize the FlexyGantt widget.
$gant_container.FlexyGantt({
  TaskItemTemplate: taskTemplate,
  //..other options.
});
```

Custom Binder.

```

//A custom binder which changes the task bar background-image based on the duration.
RadiantQ.Binder.TaskColor = function ($elem, role, value, data) {
  this.element = $elem;
  this.value = value;
  //called while initializing the TaskColor binding.
  this.data = data;
  //called while initializing the TaskColor binding.
  this.init = function () {
    var duration = new RQTimeSpan(data.EndTime - data.StartTime);
    var durationdays = duration.days;
    if (durationdays > 2) {
      this.element[0].style.setProperty('background-image',
"url(Src/Styles/Images/GreenBar.png)", "important");
      this.element.css("border-color", "green", "!important");
    }
    else {
      this.element[0].style.setProperty('background-image',
"url(Src/Styles/Images/TaskBar.png)", "important");
      this.element.css("border-color", "blue", "!important");
    }
  }
  //called when endtime is changed.
  this.refresh = function () {
    this.init();
  }
}
}

```



FlexyGantt custom task look.

This is illustrated in this samples:

In HTML : ..\Samples\FlexyFlexyGanttCustomTaskLook.htm.
In ASP.NET MVC : ..\Views\Home\FlexyGantt\FlexyGanttCustomTaskLook.cshtml.
In ASP.NET : ..\Samples\FlexyGantt\FlexyGanttCustomTaskLook.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Using Knockout

Using Knockout(KO)

[Knockout](#) is a JavaScript library that helps you to create clean, rich, responsive display with a clean underlying data model - using an MVVM approach. Any time you have sections of UI that update dynamically (e.g., changing depending on the user's actions or when the bound data source changes), KO can help you implement it more simply and maintainably.

Why Knockout in gantt?

By default Gantt has its own [data binding](#) utilities that are light-weight, providing templating support, observable collections, etc. These utilities can be used outside the gantt as well. So, you don't really need to use KO.

But, if your page has to use KO for any other reason, then you can use the same in the Gantt as well.

Gantt Setup with Knockout

To begin with add the JSON task list to the Knockout "view model" as follows:

```
viewModel = {
  Tasks: ko.mapping.fromJS(self.jsonData)
};
```

And providing the collection into Gantt,

In HTML

```
// The template that defines the look for the parent task bars.
var pTemplate = "<div class='rq-gc-parentBar'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='parentLeftPoly-style'></div><div class='parentMiddleBar-style'></div><div
class='rq-gc-parentBar-rightCue'></div></div>";

// The template that defines the look for the task bars. "rq-gc-taskbar" is a
built-in style that defines a default look for the task bars.
var tTemplate = "<div class='rq-gc-taskbar'><div
class='rq-taskbar-dragThumb'></div><div
class='rq-taskbar-resizeThumb'></div></div>";

// Initialize the FlexyGantt widget.
$('#container').FlexyGantt({
  DataSource: viewModel.Tasks(),
  TaskStartTimeProperty: "StartTime",
  KnockoutObjectName="viewModel"
  ....
  ....
});
```

In ASP.NET MVC

```

@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        ControlId = "ganttt_container",
        DataSourceUrl = new Uri("/Home/GetFlexyGanttItemSource", UriKind
.RelativeOrAbsolute),
        Options = new FlexyGanttOptions()
        {
            KnockoutObjectName = "viewModel",
            ...
        }
    }
)

```

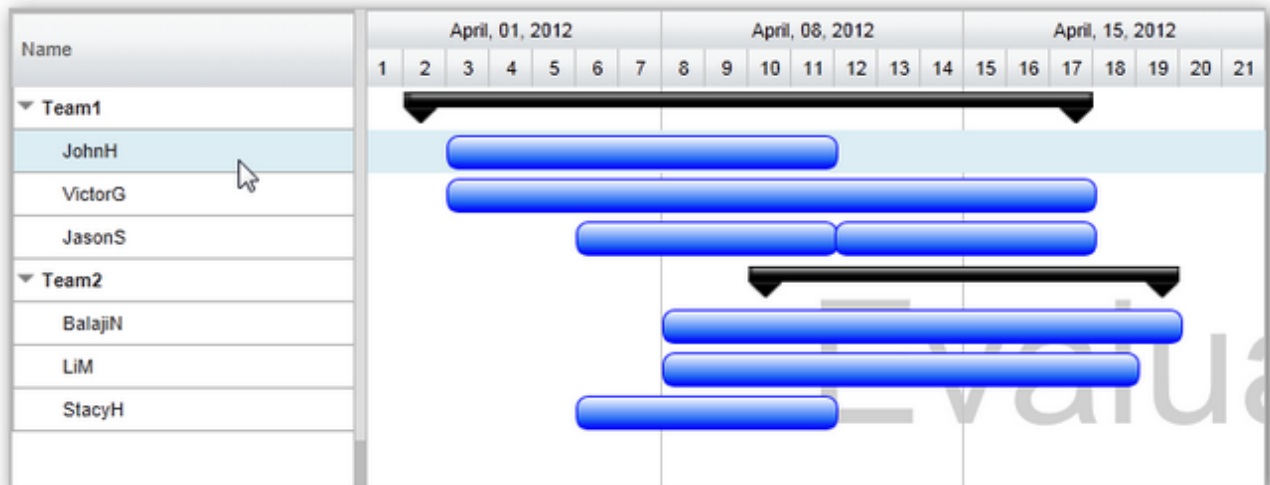
In ASP.NET

```

<RQ:FlexyGantt runat="server" ID="ganttt" Height="500"
    HierarchyResolverFunction="ResolverFunction"
    TaskStartTimeProperty="StartTime" KnockoutObjectName = "viewModel"
    TaskEndTimeProperty="EndTime"
    ... />

```

Here is the resultant gantt:



This is illustrated in this samples:

```

In HTML           : ..\Samples\FlexyGanttUsingKO.htm.
In ASP.NET MVC    : ..\Views\Home\ProjectGantt\FlexyGanttUsingKO.cshtml.
In ASP.NET        : ..\Samples\ProjectGantt\FlexyGanttUsingKO.aspx.

```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Task Template

Task Templates

The look and feel of the task bars has to be defined via the TaskItemTemplate and ParentTaskItemTemplate options in the FlexyGantt widget.



Task and Summary Task look

The template is usually a html string with one DOM element (usually a div) that represents the child or parent task bar in the chart area. This DOM element will be laid out at the appropriate position and length in the chart area.

RadiantQ template(...\RadiantQTemplate\RadiantQTemplate.chm) library is used internally to convert the template provided to actual dom elements.

Example Template that defines the above look for the child and parent bars,

In HTML

```
var pTemplate = "<div class='rq-gc-parentBar'><div
class='parentLeftPoly-style'></div><div
class='parentMiddleBar-style'></div><div
class='rq-gc-parentBar-rightCue'></div></div>";
var tTemplate = "<div class='rq-gc-taskbar'></div>";
$('#container').FlexyGantt({
  TaskItemTemplate: tTemplate,
  ParentTaskItemTemplate: pTemplate
});
```

In ASP.NET MVC

```

@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        ControlId = "ganttt_container",
        DataSourceUrl = new Uri("/Home/GetFlexyGanttItemSource", UriKind
.RelativeOrAbsolute),
        Options = new FlexyGanttOptions()
        {
            TaskItemTemplate = "<div class='rq-gc-taskbar'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div></div>"
,
            ParentTaskItemTemplate = "<div class='rq-gc-parentBar'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='parentLeftPoly-style'></div><div class='parentMiddleBar-style'></div><div
class='rq-gc-parentBar-rightCue'></div></div>",
            ..
        }
    }
)

```

In ASP.NET

```

<RQ:FlexyGantt runat="server" ID="ganttt" Height="500"
    TaskItemTemplate="<div class='rq-gc-taskbar'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div></div>"
    ParentTaskItemTemplate="<div class='rq-gc-parentBar'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='parentLeftPoly-style'></div><div class='parentMiddleBar-style'></div><div
class='rq-gc-parentBar-rightCue'></div></div>"
    ... />

```

Below are the styles referenced in above templates,

```
.rq-gc-taskbar
{
    height: 18px;
    background-image: url(TaskBar.png);
    -moz-background-size: 100% 100%; /* Firefox 3.6 */
    background-size: 100% 100%;
    background-repeat: no-repeat;
    border: 1px solid #050DFA;
    border-radius: 7px;
    float: left;
    position: absolute;
    z-index: 10;
}

.rq-gc-parentBar
{
    position: absolute; width: 500px ; height: 20px; z-index: 10;
}

.parentLeftPoly-style
{
    background-image: url('poly.png');
    width: 20px;
    position: absolute;
    height: 100%;
    background-size: 100% 100%;
    background-repeat: no-repeat;
    background-position: center;
    float: left;
}

.parentMiddleBar-style
{
    background-image: url('bar.png');
    width: 100%;
    position: absolute;
    height: 50%;
    background-repeat: no-repeat;
    background-size: 100% 100%;
    float: left;
}

.rq-gc-parentBar-rightCue
{
    background-image: url('poly.png');
    width: 20px;
    height: 100%;
    background-size: 100% 100%;
    background-repeat: no-repeat;
    background-position: center;
    float: right;
}
```

This is illustrated in this samples:

In HTML : ..\Samples\FlexyGanttCustomTaskLook.htm.

In ASP.NET MVC : ..\Views\Home\FlexyGantt\FlexyGanttCustomTaskLook.cshtml.

In ASP.NET : ..\Samples\FlexyGantt\FlexyGanttCustomTaskLook.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

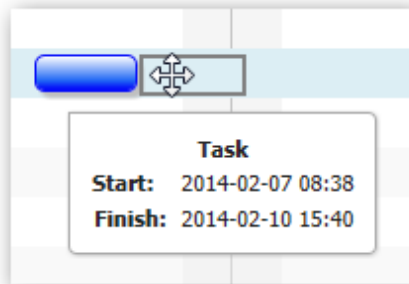
-0-

RadiantQ jQuery Gantt Package

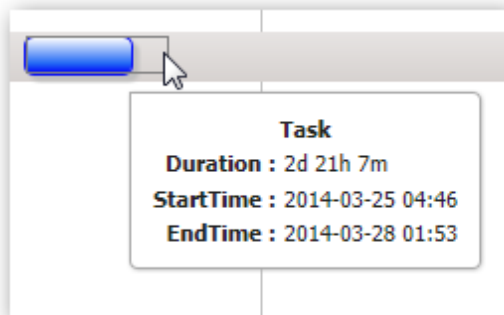
Editing Tasks

Editing Tasks

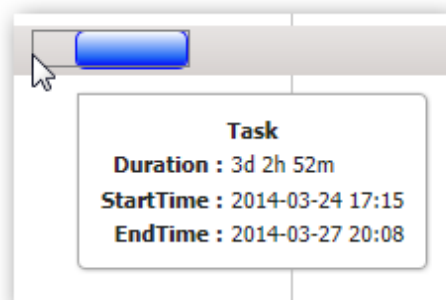
A task's StartTime can be changed by simply dragging them to the left or right in the Chart.



The task's EndTime can be changed by resizing it.



The task's StartTime can be changed by resizing it.



Resizing tasks to the left

The default template we use in most of our samples and documentation (for example in the previous [Task Template](#) topic) does not include the ability to resize at the start of the task bar. You can enable this by placing the "rq-taskbar-dragThumb" and "rq-taskbar-resizeThumb" Div's inside task templates like below


```
var pTemplate = "<div class='rq-gc-parentBar'><div  
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div  
class='parentLeftPoly-style'></div><div class='parentMiddleBar-style'></div><div  
class='rq-gc-parentBar-rightCue'></div></div>";  
  
var tTemplate = "<div class='rq-gc-taskbar'><div  
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div></div>";
```

Take a look at [Task Popups](#) topic for more information about customizing the default popup.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Multi Column Tree Grid

Multi Column Tree using FlexyTable

FlexyTable is the widget (derives from VWGrid) that is used to visualize a collapsible multi-column tree list in the left of the gantt.

In your code, set columns collection to columns property in GanttTableOptions.

In HTML

```
//all the FlexyTable Columns are defined here
var columns = [
{
  field: "Name",
  title: "Name",
  //RadiantQ.Default.Template.FlexyGanttExpandableTextBlockTemplate returns a
pre-designed template(which a expander cue and some divs)
  //nameConverter - a function name, which will return the name based on the data.
  template: RadiantQ.Default.Template.FlexyGanttExpandableTextBlockTemplate(
"nameConverter")
}
];

var $gantt_container = $("#gantt_container");
// Initialize the FlexyGantt widget.
$gantt_container.FlexyGantt({
  DataSource: self.jsonData,
  GanttTableOptions: {
    columns: columns
  },
  .....
});
```

In ASP.NET MVC

```
@Html.JQFlexyGantt(
  new JQFlexyGanttSettings()
  {
    ControlId = "gantt_container",
    GanttTableOptions = new GanttTableOptions()
      {
        Columns = new Columns(){
          new Column(){
            field = "Name",
            title = "Name",
            width = 110,
            //flexyGantNameEditor is a template script id.
            clientEditorTemplate="flexyGantNameEditor"
          }
        }
      },
    .....
  }
)
```

In ASP.NET

```

<RQ:FlexyGantt runat="server" ID="gantt" Height="500"
...>
  <GanttTableOptions>
    <Columns>
      <GanttBase:Column field="Name" title="Name" clientTemplate
="flexyGanttNameColumnTemplate" clientEditorTemplate
="flexyGanttExpandableTextBoxEditor"></GanttBase:Column>
    </Columns>
  </GanttTableOptions>
</RQ:FlexyGantt>

```

Each columns in the FlexyTable should be defined via Column definition

When you define the template, note that the DataContext for each row is a "node" in the underlying hierarchy and since this is a hierarchy, the type of bound object in each row could be different. Use Converters in the template to display appropriate data in the cells like the nameConverter below.

```

var columns = [
{
  field: "Name",
  title: "Name",
  //RadiantQ.Default.Template.FlexyGanttExpandableTextBlockTemplate returns a
pre-designed expandable text box template.
  //nameConverter - a function name, which will return the name based on the data.
  template: RadiantQ.Default.Template.FlexyGanttExpandableTextBlockTemplate(
"nameConverter")
}];

//nameConverter is global method.
var nameConverter = function (flexyNodeData, value) {
  var data;
  // The grid calls this converter with flexyNodeData as a arg.
  if (flexyNodeData)
    data = flexyNodeData.Data();
  // The grid calls this converter with flexyNodeData as a datacontext.
  else
    data = this.data;
  if (data["PName"])
    return data["PName"];
  else if (data["TaskName"])
    return data["TaskName"];
  return;
}

```

Here is the another example column definition for FlexyTable.

```

var columns = [{
  .....
  template: "<div>${progressConverter(data)}</div>",
}];

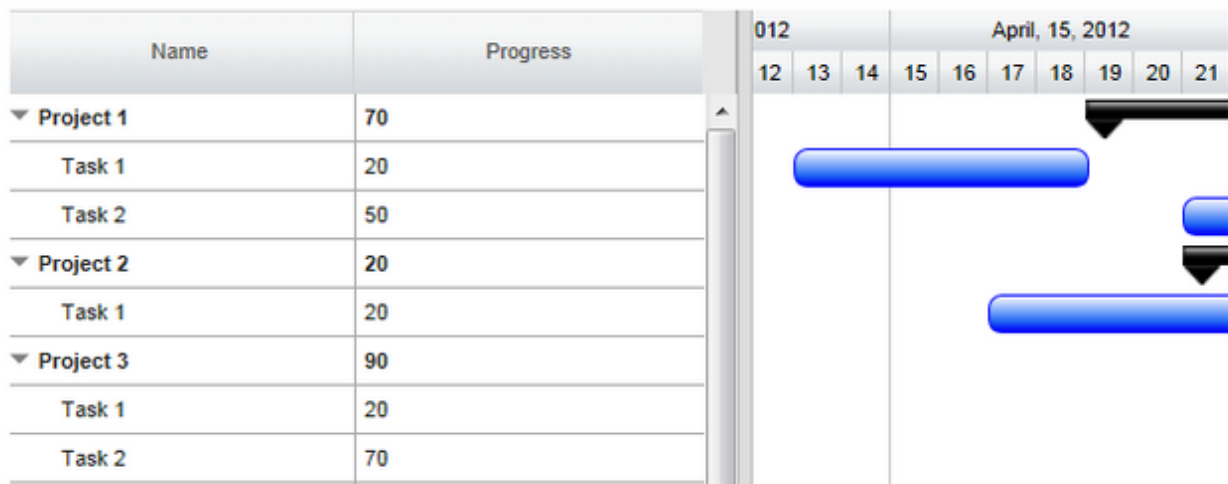
```

Make sure you defined the progress column in FlexyTable.

```
function progressConverter(data)
{
    if (data.Data()["PName"] )
    {
        var tasks = data.Data().Tasks;
        var totalprogress = 0;
        for(var i = 0; i < tasks.length; i ++ ){
            totalprogress += tasks[i].Progress
        }
        return totalprogress;
    }
    else if (data.Data()["TaskName"] )
        return data.Data()["Progress"];

    return 0;
}
```

Here is the resultant FlexyTable rendered in the gantt:



This is illustrated in this samples:

In HTML : ..\Samples\FlexyGanttCustomTaskLook.htm.

In ASP.NET MVC : ..\Views\Home\FlexyGantt\FlexyGanttCustomTaskLook.cshtml.

In ASP.NET : ..\Samples\FlexyGantt\FlexyGanttCustomTaskLook.aspx.

Alternative Row background

Alternative rows in the GanttTable has the `.rq-grid-alternative-background` class. In this class you can specify the background of the alternative row background color.

CSS style

```
.rq-grid-alternative-background {
    background-color:#F7F9FB;
}
```

© RadiantQ 2009-2018. All Rights Reserved.

RadiantQ jQuery Gantt Package

FlexyTable Editing

The jQuery Gantt widget supports editing the task's properties, in-line, within the FlexyTable. The following step illustrates how to setup editing in the FlexyTable.

Input is the default editor. you can change that by setting the editor template.

```
var columns = [
{
  field: "Name",
  title: "Name",
  //RadiantQ.Default.Template.FlexyGanttExpandableTextBoxEditor returns a
  pre-designed template(with an expander cue and input).
  editor: RadiantQ.Default.Template.FlexyGanttExpandableTextBoxEditor(
"nameConverter"),
  template: RadiantQ.Default.Template.FlexyGanttExpandableTextBlockTemplate(
"nameConverter")
}];

// An optional nameConverter global method in case the Name cell has to show values
from different kinds of objects in the bound hierarchy (like Project, Task, etc.).
var nameConverter = function (flexyNodeData, value) {
  var data;
  // The grid calls this converter with flexyNodeData as the arg.
  if (flexyNodeData.Data)
    data = flexyNodeData.Data();
  // The tooltip calls this converter with bound data as the arg.
  else
    data = flexyNodeData;
  if (data["PName"])
    return data["PName"];
  else if (data["TaskName"])
    return data["TaskName"];
  return;
}
```

For ASP.NET and ASP.NET MVC project, do the above steps in AfterGanttWidgetInitializedCallback function.

This is illustrated in these samples:

In HTML : All FlexyGantt samples.
In ASP.NET MVC : All FlexyGantt samples.
In ASP.NET : All FlexyGantt samples.

© RadiantQ 2009-2018. All Rights Reserved.

Time Scale Header

RadiantQ jQuery Gantt Package

Time Scale Header Customization

Please refer this [topic](#).

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Dependency Lines

Dependency Lines

Yes, even render dependency lines between bars in FlexyGantt. The FlexyGantt doesn't enforce any times or constraints as mentioned already, but these dependency lines might help visualize some dependencies that you have between tasks in your model.

Dependency Information

You could use a like this represent dependency lines in your sample:

```
FGDependencyHandler.resolveDependencyInfo(task, "ID", "Predecessor");
```

Note: This `resolveDependencyInfo` implemented in "FGDependencyHandler.js" file which is in sample level scripts folder that you can customize it as you needed.

The **Predecessor**(StartTask) and **ID**(EndTask) should refer to objects in your model that represent tasks (bars in the chart area).

Here are the available dependency types that you can use,

Dependency Type
FinishToStart (Default)
StartToStart
FinishToFinish
StartToFinish

which is a type defined by Gantt. This allows you to define how you want the lines to be rendered between the Start and End task, but it's optional, in which case the usual FinishToStart approach will be used.

Then create a list of `DependencyInfo` instances and bind it to the gantt as follows:

```
$gantt_container.FlexyGantt({
  DependencyListSource: depLinesList
});
```

Specifying Bindings in Sample

Assuming each row is represented by an `Employee` with these properties:

```
Employee { EName, EmployeeTasks }
```

where `EmployeeTasks` is a list of type `Task`:

```
Task{ TaskName, StartTime, EndTime }
```

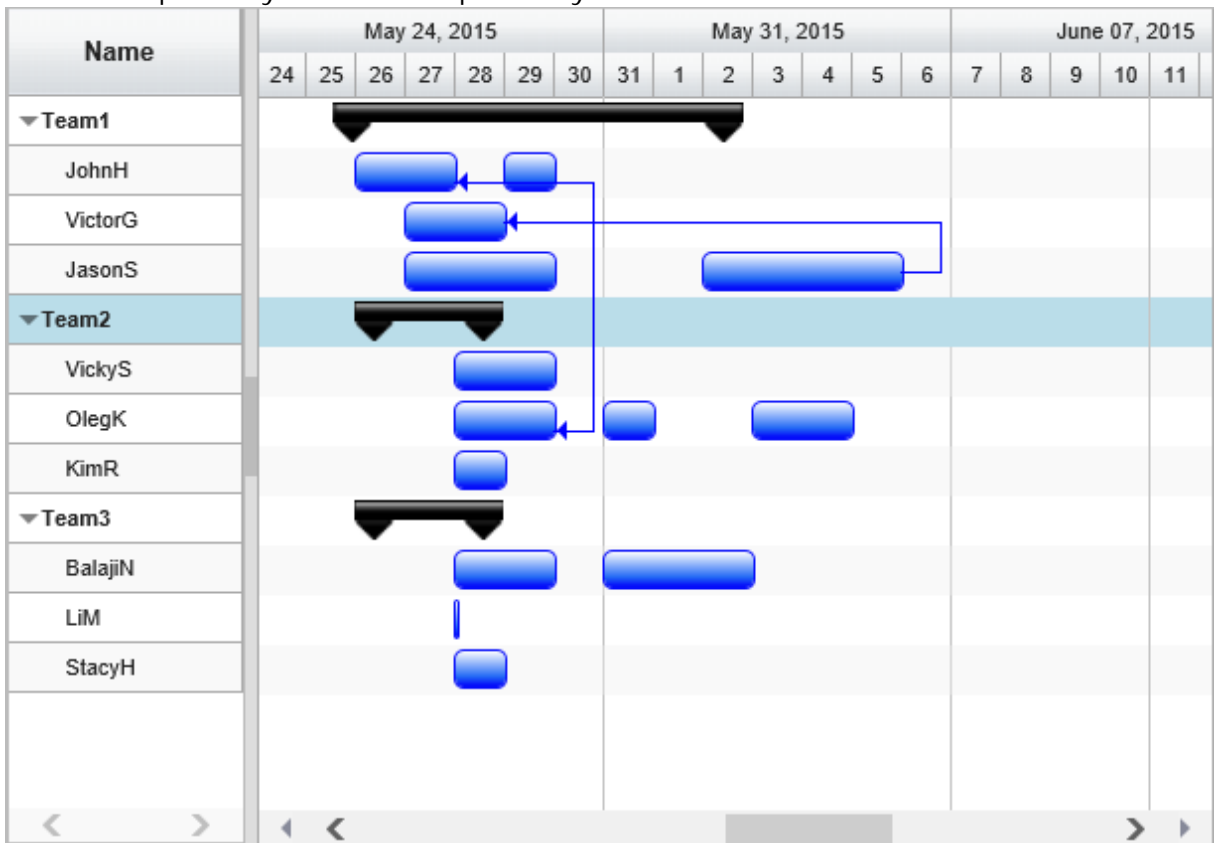
Then in `Sample` you can specify these bindings to let the gantt automatically figure out the relationships and render the dependency lines:


```

$gantt_container.FlexyGantt({
    DependencyLineStartRowItemBinding: new RadiantQ.BindingOptions(
"StartTask.Employee"),
    DependencyLineEndRowItemBinding: new RadiantQ.BindingOptions(
"EndTask.Employee"),
    DependencyLineStartItemBinding: new RadiantQ.BindingOptions(
"StartTask"),
    DependencyLineEndItemBinding: new RadiantQ.BindingOptions("EndTask"),
    DependencyLineTypeBinding: new RadiantQ.BindingOptions(
"DependencyType"),
    DependencyLineStartItemYPosAdjustmentBinding: new
RadiantQ.BindingOptions("StartTask", RadiantQ.Gantt.BindingMode.TwoWay, new
YPosAdjustmentConverter()),
    DependencyLineEndItemYPosAdjustmentBinding: new
RadiantQ.BindingOptions("EndTask", RadiantQ.Gantt.BindingMode.TwoWay, new
YPosAdjustmentConverter()),
});

```

Here is a sample FlexyGantt with dependency lines between tasks:



FlexyGantt with dependency lines

This is illustrated in this samples:

- In HTML : ..\Samples\FlexyGanttWithDepLines.htm.
- In ASP.NET MVC : ..\Views\Home\FlexyGantt\FlexyGanttWithDepLines.cshtml.
- In ASP.NET : ..\Samples\FlexyGantt\FlexyGanttWithDepLines.aspx.

RadiantQ jQuery Gantt Package

Performance Optimization Options

Vertical UI Virtualization

Please see [this](#) topic on how to enable vertically virtualized rendering in the chart area.

Horizontal UI Virtualization

By default the FlexyGantt renders all the tasks within a currently visible row in the chart area, even if the tasks are currently out of view. This is the default behavior for the following reasons:

- a) Zooming and paging the chart will be smooth.
- b) If you have adorning templates that results in UI that renders outside the task's time span, this ensures that in some scenarios, even if the tasks are not visible, it's adorners might be.
- c) In changing row-heights scenario, because of overlapped tasks a row's height won't change dynamically as you scroll horizontally.

However there are scenarios where you might have to render hundreds of thousands of tasks per row. This will definitely slow down the gantt as it tries to render all the tasks within the visible rows. This can be avoided by instructing the FlexyGantt to only render those tasks that are currently in view as follows:

In HTML

```
$gantt_container.FlexyGantt({
  UseTimeRangeFilteredTasksInRows:true,
  FilterTasksByTimeRange: function (sender, args) {
    FilterTasksByTimeRange(sender, args);
  },
  TimeRangeFilteredTasksTimeBuffer:"2.00:00:00"
});

function FilterTasksByTimeRange(sender, args) {
  var leftMostRunInView = BinarySearchLeftMostVisibleIndex(args.SourceList, 0,
args.SourceList.length - 1);
  var runs = args.SourceList;
  var filteredRuns = new Array();
  if (leftMostRunInView != -1) {
    var i = leftMostRunInView;
    while (true) {
      var run = runs[i];
      if (ViewLocation(run) == 0) {
        filteredRuns.push(runs[i]);
      }
      else
        break;

      if (i == runs.length - 1)
        break;

      i++;
    }
  }
  args.FilteredList = filteredRuns;
}
```

In ASP.NET MVC

```

<script type="text/javascript">
function FilterTasksByTimeRange(sender, args) {
    var leftMostRunInView = BinarySearchLeftMostVisibleIndex(args.SourceList, 0,
args.SourceList.length - 1);
    var runs = args.SourceList;
    var filteredRuns = new Array();
    if (leftMostRunInView != -1) {
        var i = leftMostRunInView;
        while (true) {
            var run = runs[i];
            if (ViewLocation(run) == 0) {
                filteredRuns.push(runs[i]);
            }
            else
                break;

            if (i == runs.length - 1)
                break;

            i++;
        }
    }
    args.FilteredList = filteredRuns;
}
</script>

@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        ControlId = "ganttt_container",
        AfterGanttWidgetInitializedCallback = "AfterGanttWidgetInitializedCallback",

        DataSourceUrl = new Uri("/Home/GetServerStatusHugeDataSource", UriKind
.RelativeOrAbsolute),
        Options = new FlexyGanttOptions()
        {
            TaskTooltipTemplateID = "TaskTooltipTemplate",
            TasksListProperty = "STasks",
            TaskStartTimeProperty = "StartTime",
            TaskItemTemplate = "<div style='background-color:
${taskTemplateConverter(data)};' class='taskBar'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div></div>",
            TaskEndTimeProperty = "EndTime",
            GanttChartOptions = new GanttChartOptions()
            {
                AnchorTime = DateTime.Today
            },
            UseTimeRangeFilteredTasksInRows=true,
            FilterTasksByTimeRange="FilterTasksByTimeRange",
            TimeRangeFilteredTasksTimeBuffer= "2.00:00:00"
        }
    }
)

```

In ASP.NET

```

<script type="text/javascript">
function FilterTasksByTimeRange(sender, args) {
    var leftMostRunInView = BinarySearchLeftMostVisibleIndex(args.SourceList, 0,
args.SourceList.length - 1);
    var runs = args.SourceList;
    var filteredRuns = new Array();
    if (leftMostRunInView != -1) {
        var i = leftMostRunInView;
        while (true) {
            var run = runs[i];
            if (ViewLocation(run) == 0) {
                filteredRuns.push(runs[i]);
            }
            else
                break;

            if (i == runs.length - 1)
                break;

            i++;
        }
    }
    args.FilteredList = filteredRuns;
}
</script>

<RQ:FlexyGantt runat="server" ID="gantt" Height="500"
AfterGanttWidgetInitializedCallback="AfterGanttWidgetInitializedCallback"
DataSourceUrl="../../DataSources/ServerStatusHugeSource.ashx"
HierarchyResolverFunction="ResolverFunction"
TaskTooltipTemplateID="TaskTooltipTemplate"
TasksListProperty="STasks"
UseTimeRangeFilteredTasksInRows="true"
FilterTasksByTimeRange="FilterTasksByTimeRange"
TimeRangeFilteredTasksTimeBuffer= "2.00:00:00"
... />

```

Let us look at the above 3 settings in detail:

a) UseTimeRangeFilteredTasksInRows: **true**

This tells the gantt that we only want to display the tasks that are currently within the chart's view span. The gantt will then by default, parse through the entire list of tasks to be rendered within a row, determine which tasks are currently view and render only those. This can however be further optimized by handling the FilterTasksByTimeRange event as discussed below.

b) TimeRangeFilteredTasksTimeBuffer: "2.00:00:00"

This tells the gantt to apply a buffer to the chart's time range while filtering.

c) FilterTasksByTimeRange : **function** (sender, args) {
 FilterTasksByTimeRange(sender, args);
 },

As mentioned above, parsing the entire list of tasks to be rendered within a row, every time the chart's time range changes (due to paging or zooming) is not very efficient. Knowing the

bound data more intimately, you can optimize this "filtering" better within your applications.

You can do so by listening to the above event. In your listener you can use the most optimal way to determine which set of tasks fall under the current chart view span.

For example, if the tasks in your bound list are all ordered by time, then doing a binary search to determine the left most visible item, in the current view span, would improve the time taken to filter the list exponentially.

This is illustrated in this samples:

In HTML : ..\Samples\ServerStatusWithHugeData.htm.

In ASP.NET MVC : ..\Views\Home\FlexyGantt\ServerStatusWithHugeData.cshtml.

In ASP.NET : ..\Samples\FlexyGantt\ServerStatusWithHugeData.aspx.

In this sample, using this approach, some 16000 tasks per row are visualized within the FlexyGantt, without any degradation in loading, paging and zooming times.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

Assigning Schedules

RadiantQ jQuery Gantt Package

Schedule for Rendering

FlexyGantt with a Schedule

You can assign a schedule to FlexyGantt which will then be used by the gantt while rendering and user interaction.

For example,

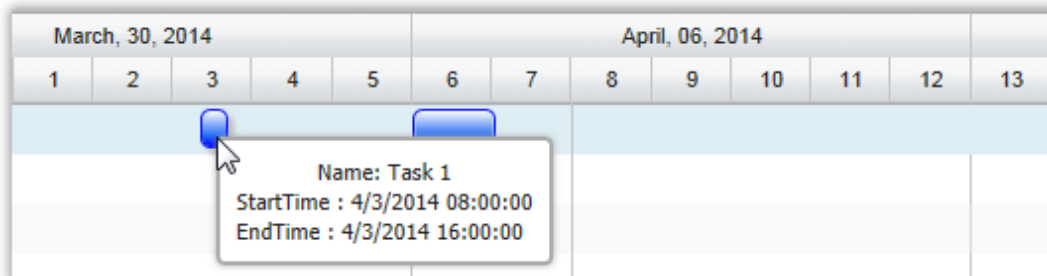
In HTML

```
var $gantt_container = $("#gantt_container");
// Initialize the FlexyGantt widget.
$gantt_container.FlexyGantt({
    DataSource: self.jsonData,
    ....
    WorkTimeSchedule: RadiantQ.Gantt.WorkTimeSchedule.Schedule8X5
});
```

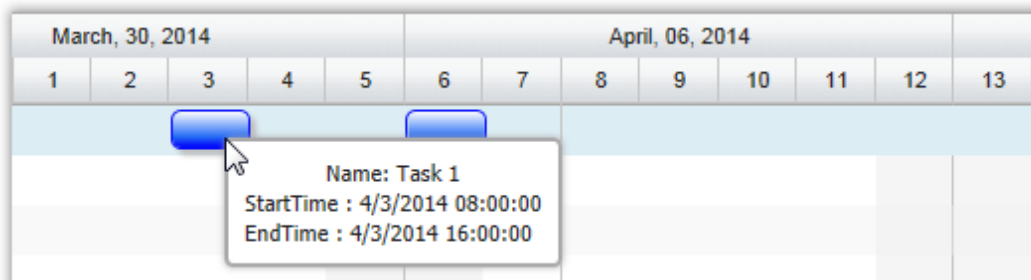
Sized-to-Fit Rendering

See how a task spanning 8 hours is rendered without and with a schedule assigned.

With a null schedule:



With a 8X5 schedule:

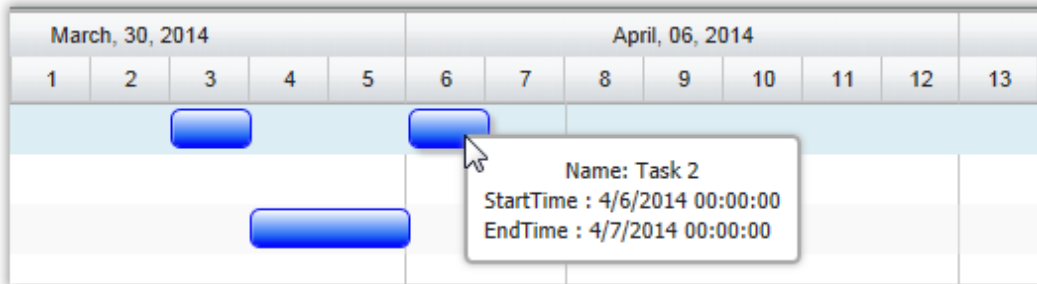


The schedule will also be applied when the user resizes or moves the task.

Times outside the schedule

While the FlexyGantt will adjust the times as the user resizes or moves a task, it will not adjust the times to fit the schedule on load. So, if your task data has times that does not fit the specified schedule, the FlexyGantt will simply render it without adjusting.

See how a task falling on a Sunday is rendered as is, without adjusting for the assigned schedule.



This is illustrated in this samples:

In HTML : ..\Samples\FGCustomResourceLevelSchedules.htm.

In ASP.NET MVC : ..\Views\Home\FlexyGantt\FGCustomResourceLevelSchedules.cshtml.

In ASP.NET : ..\Samples\FlexyGantt\FGCustomResourceLevelSchedules.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

*RadiantQ jQuery Gantt Package***Row Specific Schedule for Rendering**

Row Specific Schedule for Rendering

Sometimes, you might be plotting times on a resource each of which has a custom schedule.

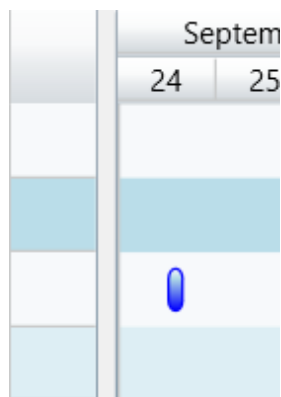
For example, you might want to plot assignments made to a machine that is running on 3 different shifts:

Shift 1 - 12AM to 8AM

Shift 2 - 8AM to 4PM

Shift 3 - 4PM to 12AM

An assignment on Shift 2, running from 12PM to 4PM would appear like this normally in the FlexyGantt:



Plotting 12PM to 4PM in a FlexyGantt

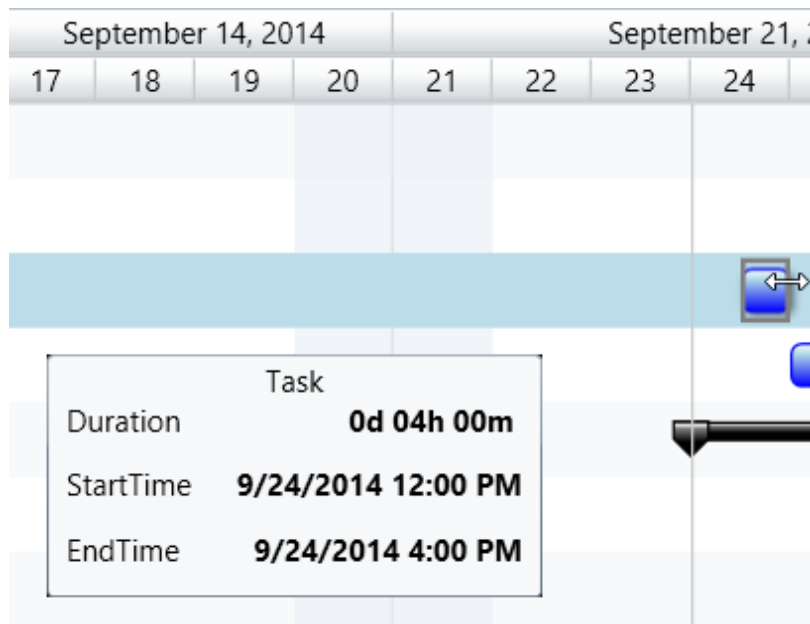
If you then want to Size-to-fit the times for these shifts, you will have to supply the corresponding, varying schedule of each row to the gantt.

You can do so by first including a "Schedule" property in the json data object.

Then set this property in FlexyGantt to let the gantt use this schedule while rendering bars in the row corresponding to this data.

```
// Initialize the FlexyGantt widget.
$gantt_container.FlexyGantt({
    .....
    .....
    // Converter will provide permission to customize the schedule value..
    RowRenderingScheduleBinding: { Property: "Schedule", Converter:
scheduleConverter },
});
```

This will then render the 4 hour bar like this within that row:



4 hour bar sized to fit an 8 Hour Shift Schedule.

Resizing and Shifting will also restrict the times within the specified schedule.

This is illustrated in this samples:

- In HTML : ..\Samples\FGCustomResourceLevelSchedules.htm.
- In ASP.NET MVC : ..\Views\Home\FlexyGantt\FGCustomResourceLevelSchedules.cshtml.
- In ASP.NET : ..\Samples\FlexyGantt\FGCustomResourceLevelSchedules.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

Appearance Customization

Chart Look and Feel

RadiantQ jQuery Gantt Package

Overlapped Tasks Look

Overlapped Tasks Look

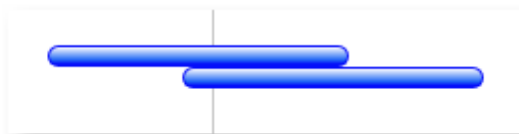
By default when there are tasks with overlapping time spans, they simply get rendered on top of each other like below.



Overlapping Bars

Shrunk Height

But you have the option to automatically make them render shrunk in height one-below the other within the same row as below.



Overlapping Bars with Shrunk Height

This is easily achieved by setting "OverlappedTasksRenderingOptimization" option in FlexyGantt widget.

In HTML

```
$( '#container' ).FlexyGantt({  
    OverlappedTasksRenderingOptimization: "ShrinkHeight"  
    ...  
});
```

In ASP.NET MVC

```

@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        ControlId = "ganttt_container",
        AfterDataRetrievedCallback = "AfterDataRetrievedCallback",
        AfterGanttWidgetInitializedCallback =
"\"AfterGanttWidgetInitializedCallback\"",
        DataSourceUrl = new
Uri("/Home/GetOverlappedTasksChangingRowHeightSource", UriKind.RelativeOrAbsolute),
        Options = new FlexyGanttOptions()
        {
            AnchorTime = DateTime.Today,
            HierarchyResolverFunction = "ResolverFunction",
            TaskTooltipTemplateID = "TaskTooltipTemplate",
            OverlappedTasksRenderingOptimization =
OverlappedTasksRenderingOptimization.ShrinkHeight,
            ...
        }
    }
)

```

In ASP.NET

```

<RQ:FlexyGantt runat="server" OnLoad="flexyGantt_Load" ID="ganttt" Height="500"
  AfterGanttWidgetInitializedCallback="AfterGanttWidgetInitializedCallback"
  HierarchyResolverFunction="ResolverFunction"
  AfterDataRetrievedCallback="AfterDataRetrievedCallback"
  DataSourceUrl
="\"../DataSources/OverlappedTasksChangingRowHeightSourceSource.ashx\"
  TaskTooltipTemplateID="TaskTooltipTemplate"
  OverlappedTasksRenderingOptimization="ShrinkHeight"
  ..
/>

```

Data Binding the bars to the overlapped state

You can also change the look and feel of overlapping bars based on whether or not they are overlapping using RadiantQ binding, below code will allow you to do that,

First insert a new property called `IsOverlapping` into your JSON objects representing the tasks and trigger them when it gets changed, like this:

```

function insertIsOverlappingObject(jsonData) {
    for (var tIndex = 0; tIndex < jsonData.length; tIndex++) {
        var resources = jsonData[tIndex].Resources;
        if (resources) {
            for (var rIndex = 0; rIndex < resources.length; rIndex++) {
                tasks = resources[rIndex].Tasks;
                if (tasks) {
                    for (var index = 0; index < tasks.length; index++) {
                        tasks[index]["IsOverlapping"] = null;
                    }
                }
            }
        }
    }
}

```

```

RadiantQ.Gantt.Utils.InsertPropertyChangedTriggeringProperty(tasks[index], [
"\"IsOverlapping\""], true);

```

Then create the RadiantQ binder and bind it to the gantt

```
RadiantQ.Binder.OverlappingBinder = function ($elem, role, value, data) {
    this.element = $elem;
    this.value = value;
    this.data = data;

    this.init = function () {
        this.element.removeClass("bluebar_style");
        this.element.removeClass("redbar_style");
        if (this.data.IsOverlapping == true)
            this.element.addClass("redbar_style");
        else
            this.element.addClass("bluebar_style");
    }
    this.refresh = function () {
        this.init();
    }
}
```

In HTML

Then specify data-binding like this in the task template to make it appear red when overlapped and blue when not.

```
var tTemplate = "<div class= 'rq-gc-tooltip rq-gc-taskbar'
data-bind='OverlappingBinder:IsOverlapping'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='start-resizeThumb'></div></div>";
```

In ASP.NET MVC

```

<script type="text/javascript">
  // will call when source is retrieved form server
  function AfterDataRetrievedCallback(jsonData) {
    insertIsOverlappingObject(jsonData);
  }
</script>

@Html.JQFlexyGantt(
  new JQFlexyGanttSettings()
  {
    ControlId = "ganttt_container",
    AfterDataRetrievedCallback = "AfterDataRetrievedCallback",
    AfterGanttWidgetInitializedCallback = "AfterGanttWidgetInitializedCallback",
    DataSourceUrl = new Uri("/Home/GetOverlappedTasksChangingRowHeightSource",
UriKind.RelativeOrAbsolute),
    Options = new FlexyGanttOptions()
    {
      HierarchyResolverFunction = "ResolverFunction",
      OverlappedTasksRenderingOptimization =
OverlappedTasksRenderingOptimization.ShrinkHeight,
      TasksListProperty = "EmployeeTasks",
      FlatItemsSourceCreated = "FlatItemsSourceCreated",
      TaskStartTimeProperty = "StartTime",
      ParentTaskStartTimeProperty = "OverallStartTime",
      TaskItemTemplate = "<div class= 'rq-gc-tooltip rq-gc-taskbar'
data-bind='OverlappingBinder:IsOverlapping'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='start-resizeThumb'></div></div>",
      ParentTaskItemTemplate = "<div class='rq-gc-parentBar'><div
class='dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='parentLeftPoly-style'></div><div class='parentMiddleBar-style'></div><div
class='rq-gc-parentBar-rightCue'></div></div>",
      TaskEndTimeProperty = "EndTime",
      ParentTaskEndTimeProperty = "OverallEndTime",
      ..
    }
  }
)

```

In ASP.NET


```

<script runat="server">
    protected void flexyGantt_Load(object sender, EventArgs e)
    {
        this.gantt.TaskItemTemplate = "<div class='rq-gc-taskbar'
data-bind='OverlappingBinder:IsOverlapping'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div></div>";
    }
</script>

<script type="text/javascript">
    // will call when source is retrieved form server
    function AfterDataRetrievedCallback(jsonData) {
        insertIsOverlappingObject(jsonData);
    }
</script>

<RQ:FlexyGantt runat="server" OnLoad="flexyGantt_Load" ID="gantt" Height="500"
    AfterGanttWidgetInitializedCallback="AfterGanttWidgetInitializedCallback"
    HierarchyResolverFunction="ResolverFunction"
    AfterDataRetrievedCallback="AfterDataRetrievedCallback"
    DataSourceUrl
= "../DataSources/OverlappedTasksChangingRowHeightSourceSource.ashx"
    TaskTooltipTemplateID="TaskTooltipTemplate"
    OverlappedTasksRenderingOptimization="ShrinkHeight"
    TaskStartTimeProperty="StartTime"
    TasksListProperty="EmployeeTasks"
    ParentTaskStartTimeProperty="OverallStartTime"
    ParentTaskItemTemplate="<div class='rq-gc-parentBar'><div
class='dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='parentLeftPoly-style'></div><div class='parentMiddleBar-style'></div><div
class='rq-gc-parentBar-rightCue'></div></div>"
    TaskEndTimeProperty="EndTime"
    ParentTaskEndTimeProperty="OverallEndTime" />

```



Overlapping bars with custom formatting

This is illustrated in this samples:

In HTML : ..\Samples\OverlappedTasksRendering.htm.
 In ASP.NET MVC : ..\Views\Home\FlexyGantt\OverlappedTasksRendering.cshtml.
 In ASP.NET : ..\Samples\FlexyGantt\OverlappedTasksRendering.aspx.

Dynamically Varying Row Heights On Overlap

Another very useful feature available in FlexyGantt is the ability to bind the height of a row to the overlapped state of the tasks in that row. This is possible because the gantt keeps the `MaxOverlappingBlocksRowCount` value for each row (in `FlexyNodeData` instance) up to date with the maximum number of overlapping bars within the different blocks in each row. And binding to this property in the `RowHeightBinding` will give you the desired effect.

JavaScript Code.

```
$('#container').FlexyGantt({
  .....
  RowHeightBinding:
  {
    Property: 'MaxOverlappingBlocksRowCount',
    Converter: function (value) {
      return value * 30;
    }
  }
});
```

MVC Wrapper Code

```

<script type="text/javascript">

    function RowHeightBindingConverter(value) {

        return value * 30;

    }
</script>

@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        ControlId = "gantt_container",
        AfterDataRetrievedCallback = "AfterDataRetrievedCallback",
        AfterGanttWidgetInitializedCallback = "AfterGanttWidgetInitializedCallback",
        DataSourceUrl = new Uri("/Home/GetOverlappedTasksChangingRowHeightSource",
UriKind.RelativeOrAbsolute),
        Options = new FlexyGanttOptions()
        {
            HierarchyResolverFunction = "ResolverFunction",
            TaskTooltipTemplateID = "TaskTooltipTemplate",
            OverlappedTasksRenderingOptimization =
OverlappedTasksRenderingOptimization.ShrinkHeight,
            RowHeightBinding = new Binding("MaxOverlappingBlocksRowCount",
"RowHeightConverter"),
            TaskStartTimeProperty = "StartTime",
            TasksListProperty = "EmployeeTasks",
            ParentTaskStartTimeProperty = "OverallStartTime",
            TaskItemTemplate = "<div style='height:20px' class= 'rq-gc-tooltip
rq-gc-taskbar' data-bind='OverlappingBinder:IsOverlapping'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='start-resizeThumb'></div></div>",
            ParentTaskItemTemplate = "<div class='rq-gc-parentBar'><div
class='dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='parentLeftPoly-style'></div><div class='parentMiddleBar-style'></div><div
class='rq-gc-parentBar-rightCue'></div></div>",
            TaskEndTimeProperty = "EndTime",
            ParentTaskEndTimeProperty = "OverallEndTime",
            GanttChartOptions = new GanttChartOptions()
            {
                AnchorTime = DateTime.Today
            }
        }
    }
)

```

In ASP.NET

```

<script runat="server">
    protected void flexyGantt_Init(object sender, EventArgs e)
    {
        this.gantt.RowHeightBinding = new RadiantQ.Web.JQGantt.Common.Binding(
"MaxOverlappingBlocksRowCount", "RowHeightConverter");
    }
</script>

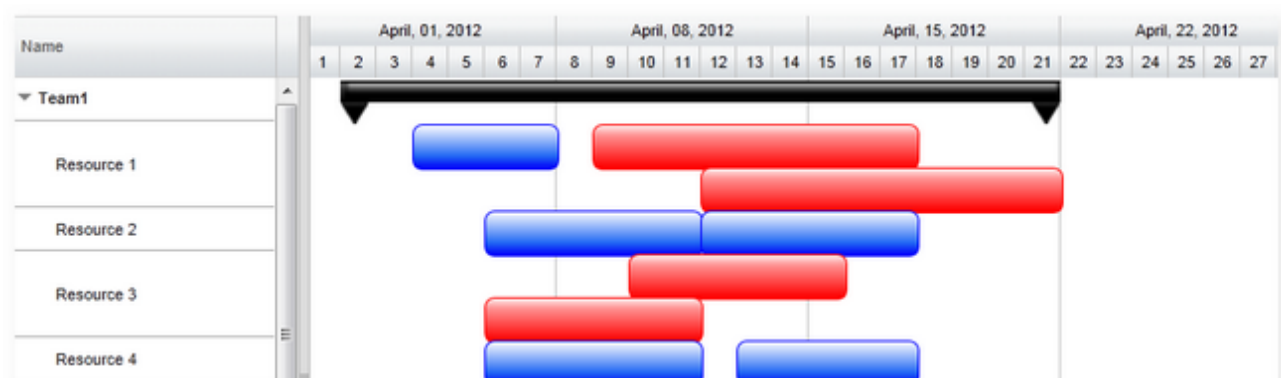
<script type="text/javascript">

    function RowHeightBindingConverter(value) {

        return value * 30;
    }
</script>

<RQ:FlexyGantt runat="server" OnLoad="flexyGantt_Init" ID="gantt" Height
="500"
    AfterGanttWidgetInitializedCallback="AfterGanttWidgetInitializedCallback"
    AfterDataRetrievedCallback="AfterDataRetrievedCallback"
    DataSourceUrl
="..\..\DataSources/OverlappedTasksChangingRowHeightSourceSource.ashx"
    HierarchyResolverFunction="ResolverFunction"
    TaskTooltipTemplateID="TaskTooltipTemplate"
    OverlappedTasksRenderingOptimization="ShrinkHeight"
    TaskStartTimeProperty="StartTime"
    TaskListProperty="EmployeeTasks"
    ParentTaskStartTimeProperty="OverallStartTime"
    TaskItemTemplate="<div style='height:20px' class='rq-gc-taskbar'
data-bind='OverlappingBinder:IsOverlapping'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='start-resizeThumb'></div></div>"
    ParentTaskItemTemplate="<div class='rq-gc-parentBar'><div
class='dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='parentLeftPoly-style'></div><div class='parentMiddleBar-style'></div><div
class='rq-gc-parentBar-rightCue'></div></div>"
    TaskEndTimeProperty="EndTime"
    ParentTaskEndTimeProperty="OverallEndTime" />

```



Row Heights varying based on overlapped state of bars

This is illustrated in this samples:

In HTML : ..\Samples\OverlappedTasksChangingRowHeights.htm.

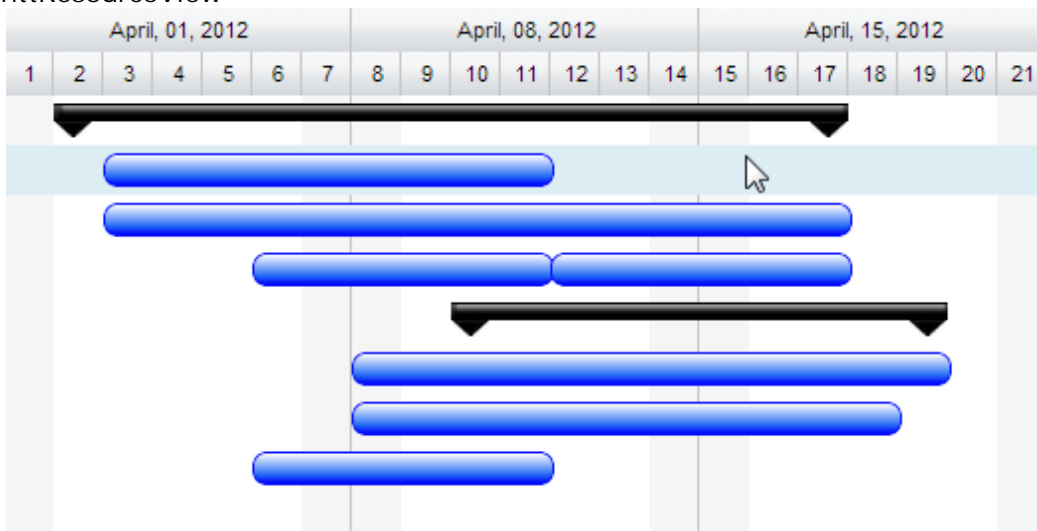
In ASP.NET MVC : ..\Views\Home\FlexyGantt\OverlappedTasksChangingRowHeights.cshtml.
In ASP.NET : ..\Samples\FlexyGantt\OverlappedTasksChangingRowHeights.aspx.

MarginForBars

This static property controls the top and bottom margin of the taskbar in FlexyGantt. by default its 2. you must set this before FlexyGantt created

```
RadiantQ.FlexyGantt.TaskItemControl.MarginForBars = 4;  
// FlexyGantt creation.
```

FlexyGanttResourceView



MarginForBars set to 4

RadiantQ jQuery Gantt Package

Task Labels

Taskbar Labels

You might have to render icons, shapes, text, etc. right next to the task bars to indicate status, annotation, etc. This is easily implemented by extending the default task template by defining a custom TaskItemTemplate, discussed in detail, in [this](#) section.

For example, you might want to mark certain tasks with a "Critical" text to it's right as follows:



Label to the right of Task bar

You can use below code to achieve this.

Define a task template like below. This template contains the default UI elements that make the task bar and an additional div (with class "rq-gc-taskbar-label") that represents the label.

In HTML

```

var pTemplate = "<div class='parentBar-style'><div class='dragThumb'></div><div
class='rq-taskbar-resizeThumb'></div><div class='parentLeftPoly-style'></div><div
class='rq-gc-parentBar-middle'></div><div
class='rq-gc-parentBar-rightCue'></div></div>";
// The template that defines the look for the task bars. "taskbar-style" is a
built-in style that defines a default look for the task bars.
var tTemplate = "<div class='rq-gc-taskbar' data-bind='TaskColor:EndTime'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='lable' data-bind='TaskColor:EndTime'></div></div>";
var $ganttt_container = $("#ganttt_container");
// Initialize the FlexyGantt widget.
$ganttt_container.FlexyGantt({
    DataSource: self.jsonData,
    TaskItemTemplate: tTemplate,
    ParentTaskItemTemplate: pTemplate
});

RadiantQ.Binder.TaskColor = function ($elem, role, value, data) {
    this.element = $elem;
    this.data = data;
    this.init = function () {
        var duration = new RQTimeSpan(data.EndTime - data.StartTime);
        var durationdays = duration.days;
        if (durationdays > 2) {
            if (this.element.hasClass("lable")) {
                updateText(this.element, data.EndTime);
                this.element.css("color", "green");
            }
            else {
                this.element[0].style.setProperty('background-image', "url('" +
greenBarImageURL + "')", "important");
                this.element.css("border-color", "green", "!important");
            }
        }
        else {
            if (this.element.hasClass("lable")) {
                this.element.css("color", "blue");
                updateText(this.element, data.EndTime);
            }
            else {
                this.element[0].style.setProperty('background-image',
"url(.../Src/Styles/Images/TaskBar.png)", "important");
                this.element.css("border-color", "blue", "!important");
            }
        }
    }
    this.refresh = function () {
        this.init();
    }
};

function updateText(element, endTime) {
    if (endTime < new Date())
        $(element).text("Critical");
    else
        $(element).text("");
}

```

In ASP.NET MVC

```

<script type="text/javascript">
RadiantQ.Binder.TaskColor = function ($elem, role, value, data) {
    this.element = $elem;
    this.data = data;
    this.init = function () {
        var duration = new RQTimeSpan(data.EndTime - data.StartTime);
        var durationdays = duration.days;
        if (durationdays > 2) {
            if (this.element.hasClass("lable")) {
                updateText(this.element, data.EndTime);
                this.element.css("color", "green");
            }
            else {
                this.element[0].style.setProperty('background-image', "url('" +
greenBarImageURL + "')", "important");
                this.element.css("border-color", "green", "!important");
            }
        }
        else {
            if (this.element.hasClass("lable")) {
                this.element.css("color", "blue");
                updateText(this.element, data.EndTime);
            }
            else {
                this.element[0].style.setProperty('background-image',
"url(../../Src/Styles/Images/TaskBar.png)", "important");
                this.element.css("border-color", "blue", "!important");
            }
        }
    }
    this.refresh = function () {
        this.init();
    }
};

function updateText(element, endTime) {
    if (endTime < new Date())
        $(element).text("Critical");
    else
        $(element).text("");
}
</script>

```

```

@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        ControlId = "gantt_container",
        AfterGanttWidgetInitializedCallback =
"AfterGanttWidgetInitializedCallback",
        DataSourceUrl = new Uri("/Home/GetFlexyGanttItemSource", UriKind
.RelativeOrAbsolute),
        Options = new FlexyGanttOptions()
        {
            HierarchyResolverFunction = "ResolverFunction",
            TaskTooltipTemplateID = "TaskTooltipTemplate",
            TaskStartTimeProperty = "StartTime",
            ParentTaskStartTimeProperty = "OverallStartTime",
            TaskItemTemplate = "<div class='rq-gc-taskbar'
data-bind='TaskColor:EndTime'><div class='rq-taskbar-dragThumb'></div><div
class='rq-taskbar-resizeThumb'></div><div class='lable'
data-bind='TaskColor:EndTime'></div></div>",
            ParentTaskItemTemplate = "<div class='parentBar-style'><div
class='dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='parentLeftPoly-style'></div><div class='rq-gc-parentBar-middle'></div><div
class='rq-gc-parentBar-rightCue'></div></div>",
            TaskEndTimeProperty = "EndTime",
            ParentTaskEndTimeProperty = "OverallEndTime",
            GanttChartOptions = new GanttChartOptions()

```


In ASP.NET

```

<script type="text/javascript">
RadiantQ.Binder.TaskColor = function ($elem, role, value, data) {
    this.element = $elem;
    this.data = data;
    this.init = function () {
        var duration = new RQTimeSpan(data.EndTime - data.StartTime);
        var durationdays = duration.days;
        if (durationdays > 2) {
            if (this.element.hasClass("lable")) {
                updateText(this.element, data.EndTime);
                this.element.css("color", "green");
            }
            else {
                this.element[0].style.setProperty('background-image', "url('" +
greenBarImageURL + "')", "important");
                this.element.css("border-color", "green", "!important");
            }
        }
        else {
            if (this.element.hasClass("lable")) {
                this.element.css("color", "blue");
                updateText(this.element, data.EndTime);
            }
            else {
                this.element[0].style.setProperty('background-image',
"url(../../Src/Styles/Images/TaskBar.png)", "important");
                this.element.css("border-color", "blue", "!important");
            }
        }
    }
    this.refresh = function () {
        this.init();
    }
};

function updateText(element, endTime) {
    if (endTime < new Date())
        $(element).text("Critical");
    else
        $(element).text("");
}
</script>

<RQ:FlexyGantt ID="gantt" AfterDataRetrievedCallback="AfterDataRetrievedCallback"
AfterGanttWidgetInitializedCallback="AfterGanttWidgetInitializedCallback"
ParentTaskEndTimeProperty="OverallEndTime"
ParentTaskStartTimeProperty="OverallStartTime"
TaskEndTimeProperty="EndTime"
TaskStartTimeProperty="StartTime"
HierarchyResolverFunction="ResolverFunction"
DataSourceUrl="../../DataSources/FGTaskListHandler.ashx"
LocalizationResourceFilePath="../../Src/ResourceStrings/"
Height="500" runat="server"
TaskItemTemplate="<div class='rq-gc-taskbar'
data-bind='TaskColor:EndTime'><div class='rq-taskbar-dragThumb'></div><div
class='rq-taskbar-resizeThumb'></div><div class='lable'
data-bind='TaskColor:EndTime'></div></div>"
ParentTaskItemTemplate="<div class='parentBar-style'><div
class='dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='parentLeftPoly-style'></div><div class='rq-gc-parentBar-middle'></div><div
class='rq-gc-parentBar-rightCue'></div></div>" />

```

And remember to define the rq-gc-taskbar-label CSS as follows.

In the margin-right, specify "-40px" so that the text will always be rendered 40 pixels to the right of the bar.

```
<style type="text/css">
  .rq-gc-taskbar-label {
    padding-top: 2px;
    float: right;
    margin-right: -40px;
  }
</style>
```

This is illustrated in these samples:

In HTML : ..\Samples\FlexyGanttCustomTaskLook.htm.
In ASP.NET MVC : ..\Views\Home\FlexyGantt\FlexyGanttCustomTaskLook.cshtml.
In ASP.NET : ..\Samples\FlexyGantt\FlexyGanttCustomTaskLook.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Task Popups

Task Popup

Popups are shown when the end-user interacts with the task, when they move it and resize it. While the default content might be good enough and appropriate enough for most scenarios, you can replace these popups with custom popups in necessary.

These are the properties in GanttChart that will allow you do this:

- ResizeInfoPopup
- MovingInfoPopup

The class reference for these properties contain information on the DataContext that will be applied on these custom popup instances and you can construct your custom popup accordingly.

The custom look of the popup should be defined as template.

```
<script id="MovingInfoPopup" type="text/x-jquery-tmpl">
  <table class="draggingContent" style="width: 180px;">
    <tr>
      <td colspan='2' align='center'>${RadiantQ_TaskString} </td>
    </tr>
    <tr>
      <td>${RadiantQ_StartString} : ${data.StartTime.toString('yyyy-MM-dd
HH:mm')} </td>
    </tr>
    <tr>
      <td>${RadiantQ_FinishString} : ${data.EndTime.toString('yyyy-MM-dd
HH:mm')} </td>
    </tr>
  </table>
</script>

<script id="ResizeInfoPopup" type="text/x-jquery-tmpl">
  <table class='draggingContent' style='width: 180px;'>
    <tr>
      <td colspan='2' align='center'>Task </td>
    </tr>
    <tr>
      <td style='float: right;'>Duration : </td>
      <td>#if (data.Duration){# # = data.Duration.days_M() #d # =
data.Duration.hours_M() #h #}# </td>
    </tr>
    <tr>
      <td style='float: right;'>StartTime : </td>
      <td>${data.StartTime.toString('yyyy-MM-dd HH:mm')}</td>
    </tr>
    <tr>
      <td style='float: right;'>EndTime :</td>
      <td>${data.EndTime.toString('yyyy-MM-dd HH:mm')}</td>
    </tr>
  </table>
</script>
```

In HTML

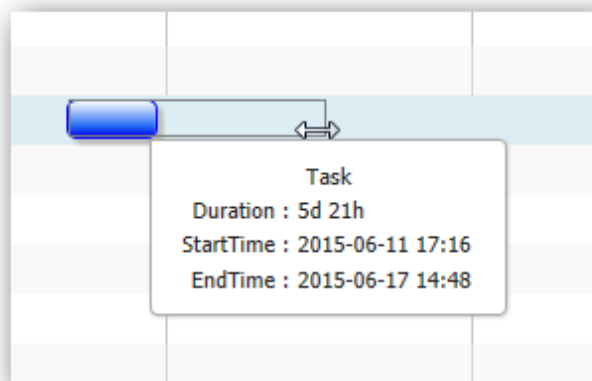
```
$('#container').FlexyGantt({
    MovingInfoPopup: $("#MovingInfoPopup").html(),
    ResizeInfoPopup: $("#ResizeInfoPopup").html()
});
```

In ASP.NET MVC

```
@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        ..
        Options = new FlexyGanttOptions()
        {
            GanttChartOptions = new GanttChartOptions()
            {
                AnchorTime = DateTime.Today,
                MovingInfoPopupID = "MovingInfoPopup",
                ResizeInfoPopupID = "ResizeInfoPopup",
            }
        }
    }
)
```

In ASP.NET

```
<RQ:FlexyGantt ID="gantt" AfterDataRetrievedCallback="AfterDataRetrievedCallback"
AfterGanttWidgetInitializedCallback="AfterGanttWidgetInitializedCallback"
ParentTaskEndTimeProperty="OverallEndTime"
ParentTaskStartTimeProperty="OverallStartTime"
TaskEndTimeProperty="EndTime"
MovingInfoPopupID="MovingInfoPopup"
ResizeInfoPopupID="ResizeInfoPopup"
TaskStartTimeProperty="StartTime"
HierarchyResolverFunction="ResolverFunction"
DataSourceUrl="../../DataSources/FGTaskListHandler.ashx"
LocalizationResourceFilePath="../../Src/ResourceStrings/"
Height="500" runat="server"
TaskItemTemplate="<div class='rq-gc-taskbar' data-bind='TaskColor:EndTime'><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='lable' data-bind='TaskColor:EndTime'></div></div>"
ParentTaskItemTemplate="<div class='parentBar-style'><div
class='dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='parentLeftPoly-style'></div><div class='rq-gc-parentBar-middle'></div><div
class='rq-gc-parentBar-rightCue'></div></div>" />
```



Customized Task Popup

This is illustrated in this samples:

In HTML : ..\Samples\FlexyGanttCustomTaskLook.htm.
In ASP.NET MVC : ..\Views\Home\FlexyGantt\FlexyGanttCustomTaskLook.cshtml.
In ASP.NET : ..\Samples\FlexyGantt\FlexyGanttCustomTaskLook.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

Custom UI In Task Row

RadiantQ jQuery Gantt Package

RefreshRowBackground

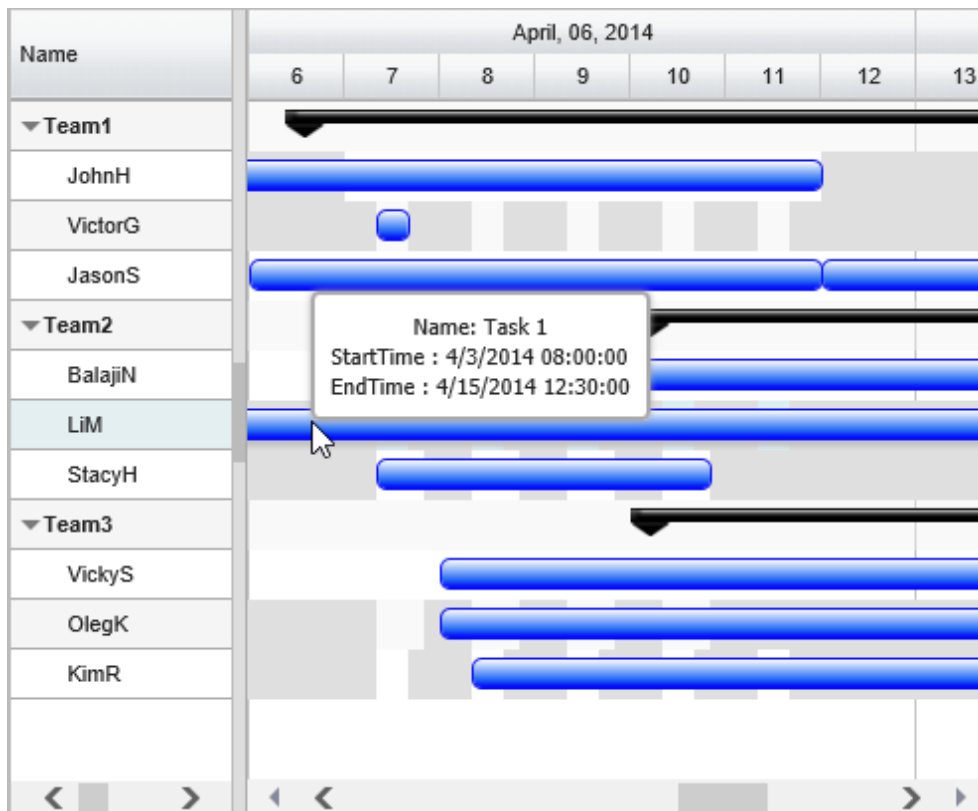
Custom UI in Row Background

Occasionally, you might have to render some custom UI in a FlexyGantt row's background. For example, to visualize "down times" for your resource.

You do so by first setting `EnableCustomRowBackground` option to true in FlexyGantt. This will prompt the FlexyGantt to insert a transparent div in the row background spanning the entire chart's view span and then raise the `RefreshRowBackground` event where you can insert your custom UI elements into the 'background div'.

The code below illustrates how this can be setup. In this sample, we render some "gray rectangles" that represent non-working hours in a defined schedule. We also use a utility function, `RenderNonWorkingHours`, to easily render these rectangles. However, you can simply add any child UI elements to the *bgDiv*.

```
$gantt_container.FlexyGantt({
    .....
    .....
    EnableCustomRowBackground: true,
    GanttChartTemplateApplied: function (sender , args) {
        var flexyGantt = $gantt_container.data("FlexyGantt");
        var GanttChart = flexyGantt.GetGanttChart().data("GanttChart");
        // RefreshRowBackground event is raised every time the row's span changes
        GanttChart.RefreshRowBackground.subscribe(function (bgDiv, args) {
            // bgDiv - The div inside which you should add your custom UI
            elements.
            // args -
            // args.GanttChart - Reference to the GanttChart widget.
            // args.Data - Reference to the FlexyNodeData of the row.
            var data = args.Data._data;
            var scheduleID = data.ScheduleID;
            var schedule = getSchedule(scheduleID);
            if (schedule)
                RadiantQ.FlexyGantt.RenderNonWorkingHours(bgDiv, 'grayBGStyle',
args.GanttChart, schedule);
        });
    }
});
```



Non working times in a schedule rendered with gray background

Note that if you have to render some rectangular divs at specific times, the following utility methods will be of help:

ConvertTimeToX: Will Converts a specified datetime to it's X location.

```
var x = ganttChart.ConvertTimeToX(DateTime);
```

ConvertToNextWorkingTime: If the specified datetime is a "working time" returns that, and if not returns the next closest valid work time.

```
var workingTime = workTimeSchedule.ConvertToNextWorkingTime(DateTime);
```

ConvertToNextWorkingTimeIntervals: Returns the different TimeIntervals that the work will spawn given the start time and duration of the activity.

```
var viewSpan = new TimeSpan(viewComputedEnd - viewComputedStart);
var timePeriods = workTimeSchedule.ConvertToNextWorkingTimeIntervals(
viewComputedStart, viewSpan, null);
```

This is illustrated in this samples:

In HTML : ..\Samples\FGCustomResourceLevelSchedules.htm.

In ASP.NET MVC : ..\Views\Home\FlexyGantt\FGCustomResourceLevelSchedules.c.shtml.

In ASP.NET : ..\Samples\FlexyGantt\FGCustomResourceLevelSchedules.aspx.

RadiantQ jQuery Gantt Package

RefreshRowForeground

Custom UI in Row Foreground:

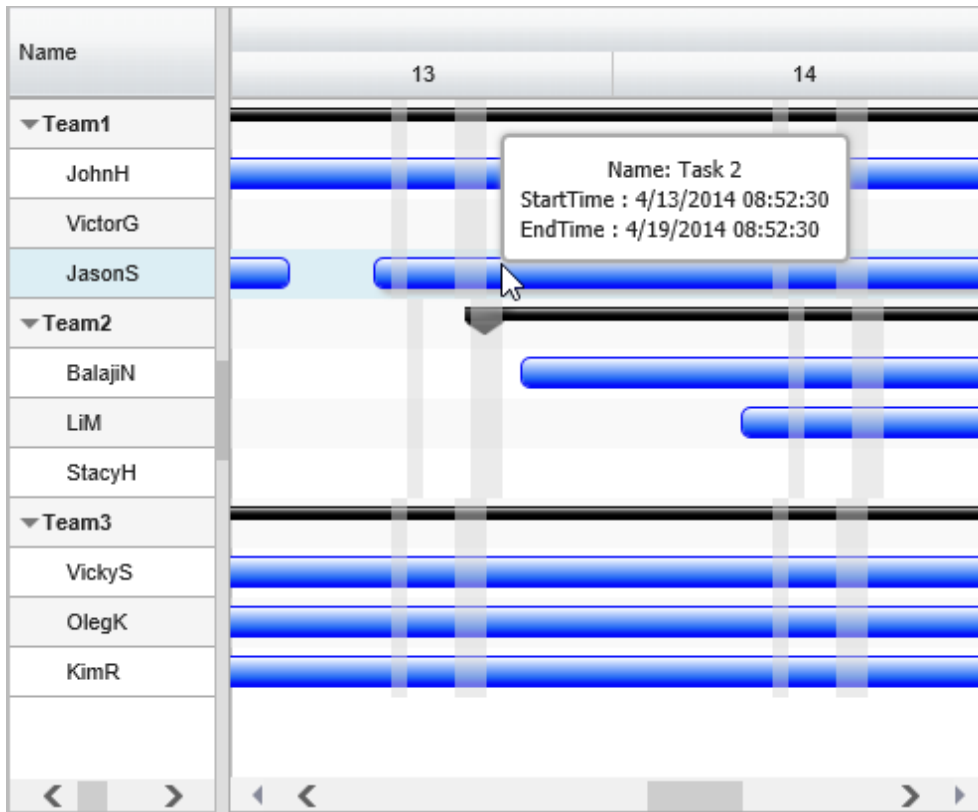
Similar to the above [topic\(RefreshRowBackground\)](#), you might want to occasionally render some custom UI on top of the tasks in the row. For example, to indicate "breaks" at specific times in the chart.

You do so by first setting EnableCustomRowForeground to true and then subscribing to RefreshRowForeground. Below is some sample code that shows how to use the RenderNonWorkingHours to render some rectangular divs indicating 'breaks'.

```

$gantt_container.FlexyGantt({
    .....
    .....
    EnableCustomRowForeground: true,
    GanttChartTemplateApplied: function (sender , args) {
        var flexyGantt = $gantt_container.data("FlexyGantt");
        var GanttChart = args.element;
        // RefreshRowForeground event is raised every time the row's span changes
        GanttChart.RefreshRowForeground.subscribe(function (fgDiv, args) {
            // fgDiv - The div inside which you should add your custom UI
            elements.
                // args - It contain the two values.
                // args.GanttChart - Reference to the GanttChart widget.
                // args.Data - Reference to the FlexyNodeData of the row.
            var data = args.Data._data;
            var breaksID = data.BreaksID;
            var break = getBreaks(breaksID);
            if (break)
                RadiantQ.FlexyGantt.RenderNonWorkingHours(fgDiv, 'grayFGStyle',
args.GanttChart, break);
        });
    }
});

```



Break times in a schedule rendered with gray background

RadiantQ jQuery Gantt Package

Row Background

Selected Row Background

Use the `.ui-selected` css style to customize the selected row background, as all the selected rows will include the `.ui-selected` style.

Here is the default style for the selected row.

```
ui-selected
{
  background:#badde9;
}
```

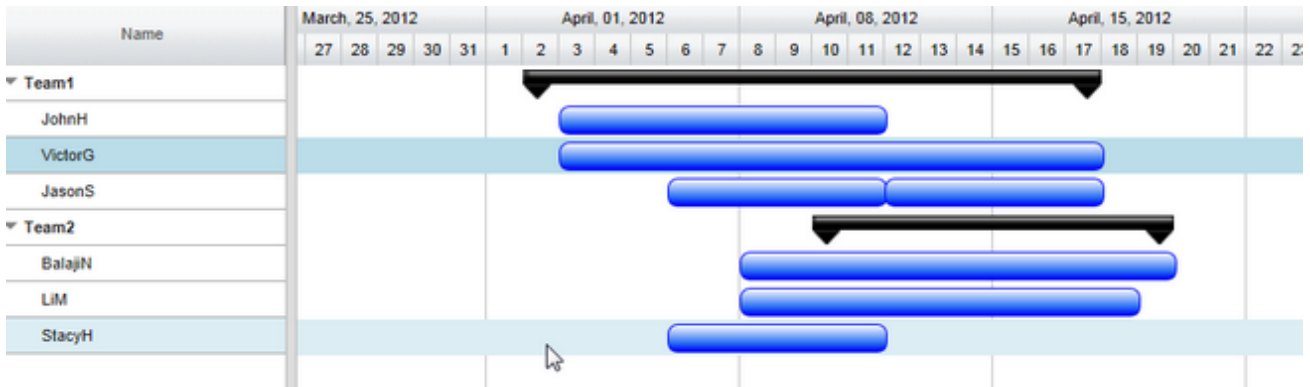
Hovering Row Background

Similarly, use the `.rq-row-hover` css style to customize the selected row background, as all the hovering rows will include the `.rq-row-hover` style.

here is the default style for a hovering row.

```
.rq-row-hover
{
  background:#dceef4;
}
```

the resultant will be like this.



Default Selected row and hovering row styles.

RadiantQ jQuery Gantt Package

Gantt Look and Feel

Splitter position

Often you want to show less of the table on the left and more of the chart on the right. This is easily implemented by setting the value to the `TablePanelWidth` option in `FlexyGantt`.

In HTML

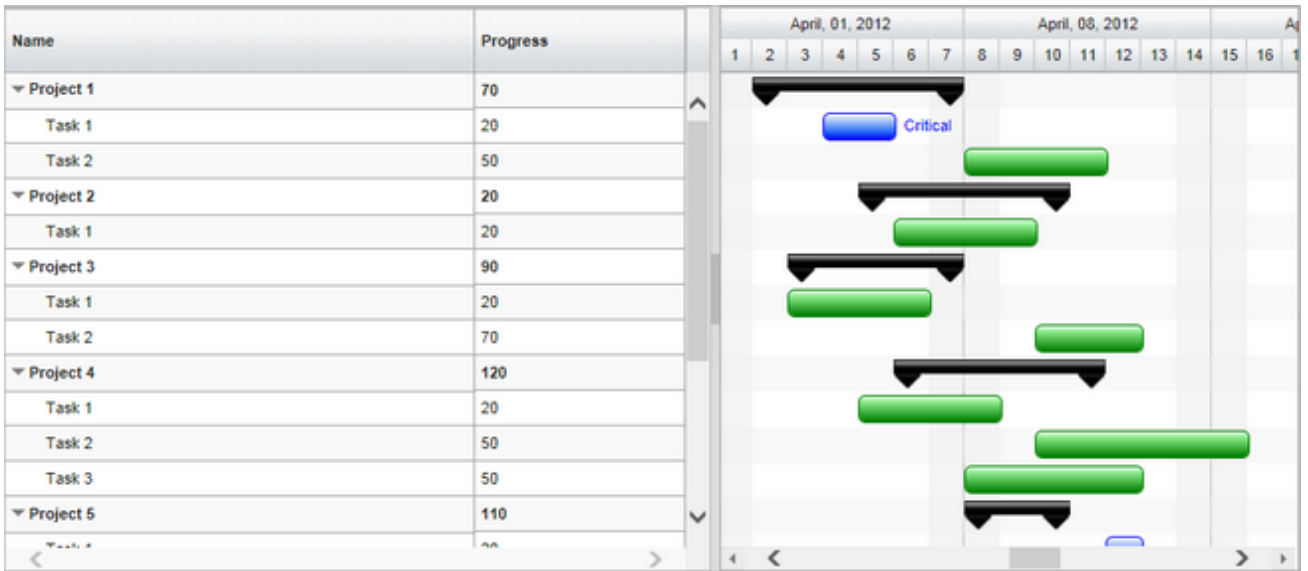
```
$( '#container' ).FlexyGantt( {
    TablePanelWidth: 500, //the default Value is 200
    ....
});
```

In ASP.NET MVC

```
@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        Options = new FlexyGanttOptions()
        {
            ...
            TablePanelWidth= 500,
            SpecialLineInfos = new System.Collections.ObjectModel.
ObservableCollection<SpecialLineInfo>() { new SpecialLineInfo() { LineColor="green",
LineDateTime=DateTime.Today.AddDays(1), ToolTipText="now" } },
            GanttChartOptions = new GanttChartOptions()
            {
                AnchorTime = DateTime.Today,
                MovingInfoPopupID = "MovingInfoPopup",
                ResizeInfoPopupID = "ResizeInfoPopup",
            }
        }
    }
)
```

In ASP.NET

```
<RQ:FlexyGantt ID="gantt" AfterDataRetrievedCallback="AfterDataRetrievedCallback"
TablePanelWidth= "500" ../>
```



GanttTable Width is 500

Hiding the Grid

setting 0 to the TablePanelWidth will hide the GanttTable.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

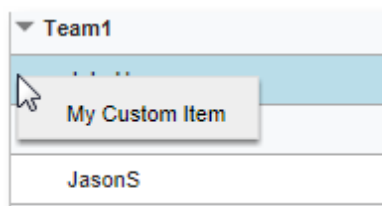
ContextMenus

The FlexyGantt includes the following built-in context menus which are empty by default (so not shown when right clicked on). However you can add custom items into it based on the context.

But to begin with you have a couple of options to select a context menu framework that the FlexyGantt should work with. See this [topic](#) for more information on this.

FlexyTable Row Context Menu

The table row context menu is shown when right clicking on the row in the FlexyTable. The built-in context menu is empty, but you can add custom items to it as shown below:



FlexyGantt Task Bar Context Menu

The FlexyGantt's task bar context menu is shown when right-clicking on the task bars in the FlexyGantt. The built-in context menu is empty, however you can add custom items into it based on the context.



Here is the sample code snippet from the above:

```
var taskContextMenu = $('#container').data("FlexyGantt").TaskContextMenu;
taskMenuItems = [
    { keyName: "MyCustomItem", name: "My Custom Item", icon: "My
Custom Item", callback: function (key, opt) {
        var dataContext = opt.$trigger[0].DataContext;
        alert("TaskInfo context : TaskName: " + dataContext.TaskName + ",
StartTime: " + dataContext.StartTime);
    }
}
]
taskContextMenu.AddNewItems(taskMenuItems);

var tableContextMenu = $('#container').data("FlexyGantt").TableContextMenu;
tableMenuItems = [
    { keyName: "MyCustomItem", name: "My Custom Item", icon: "My
Custom Item", callback: function (key, opt) {
        var dataContext = grid.GetDataFromRow(opt.$trigger[0]);
        alert("FlexyTable row context : Resource: " +
(dataContext.Data().RName || dataContext.Data().TName));
    }
}
]
tableContextMenu.AddNewItems(tableMenuItems);
```

This is illustrated in this samples:

In HTML : ..\Samples\CustomizingMenus.htm.
In ASP.NET MVC : ..\Views\Home\FlexyGantt\CustomizingMenus.cshtml.
In ASP.NET : ..\Samples\FlexyGantt\CustomizingMenus.aspx.

Behavior Customization

FlexyTable Customization

RadiantQ jQuery Gantt Package

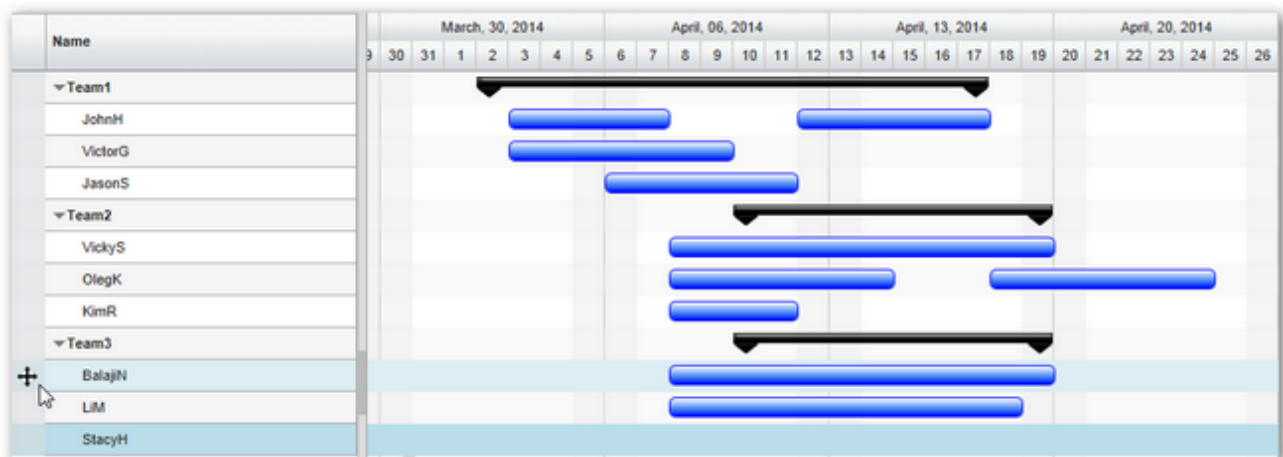
Enable Row Drag and Drop

Enabling Row Drag And Drop

You can turn on drag and drop of rows in the FlexyGantt's grid as follows with the `CanUserReorderRows`, `EnableDropAsChild` settings:

```
var $gantt_container = $('#gantt_container');
    $gantt_container.FlexyGantt({
        CanUserReorderRows: true,
        EnableDropAsChild: true,
        .....
        .....
        GanttChartTemplateApplied: function (sender , args) {
            .....
        }
    });
```

This will allow you to start dragging rows:



Dragging a FlexyGantt row

Next, listen to drag & drop events as follows:

```

var fxgantt = $gantt_container.data('FlexyGantt');

fxgantt.BeforeRowsDragStart.subscribe(function (args) {
    // Here the args will give you the top selected row index. You can optionally,
    // prevent the dragging here.
    var selectedItems = fxgantt.SelectedItems;
    for (var i = 0; i < selectedItems.length; i++) {
        if (selectedItems[i].Data().TName == "Team2") {
            args.Cancel = true;
            return;
        }
    }
});
fxgantt.SelectedRowsDrop.subscribe(function (args) {
    // Here the args will give you the rows that are being dragged and the
    // destination row, including whether the user wanted the rows to be dropped as
    // sibling or children.

    // This will be the case when Teams are dragged.
    var selectedItem = fxgantt.SelectedItem;
    if (selectedItem.IsParentType() == false) {
        alert("Only Employees can be dragged");
        args.Cancel = true;
        return;
    }
    // This will be the case when the user chooses to drop next to a node rather than
    // within a node.
    else if (args.DropAsChildren == false) {
        args.Cancel = true;
        alert("Can only be dropped as children.");
    }
    // User choose to drop within a node
    else {
        var dropRowData = fxgantt.FlatHierarchicalItemsList[args.DestinationIndex];
        // Do this only when the destination is a Team
        if (dropRowData.data.TName == "Team1") {
            args.Cancel = true;
            return;
        }

        if (dropRowData._parentItem == null || dropRowData._parentItem == undefined)
        {
            // Save the selected rows, before deleting them from the UI:
            var selectedRows = fxgantt.SelectedItems.slice(0);

            var index = GetRowIndex(dropRowData.corrFlexyNodeData);
            var dropTeam = GetTeam(index);

            // Move all the selected employees into the destination team.
            for (var i = 0; i < selectedRows.length; i++) {
                var empIndex = GetRowIndex(selectedRows[i]);
                var employee = selectedRows[i].Data();
                var currentTeam = GetTeam(empIndex);
                if (currentTeam != dropTeam) {
                    // Delete from old Team
                    $.observable(currentTeam.Resources).remove(employee);
                    // Add to new Team
                    $.observable(dropTeam.Resources).insert(employee);
                }
            }
        }
        else {
            alert("Can only be dropped on Teams");
            args.Cancel = true;
        }
    }
});

```

A sample that illustrates this feature can be located here in the install:

In HTML : ..\Samples\MovingResourceRows.htm.
In ASP.NET MVC : ..\Views\Home\FlexyGantt\MovingResourceRows.cshtml.
In ASP.NET : ..\Samples\FlexyGantt\MovingResourceRows.aspx.

-0-

Events

RadiantQ jQuery Gantt Package

Task Bar Vertical Dragging

Task Bar Vertical Dragging

By default, the task bar can be dragged horizontally from one time to another. But, there is also support for dragging the task bars vertically to support moving bars from one row to another.

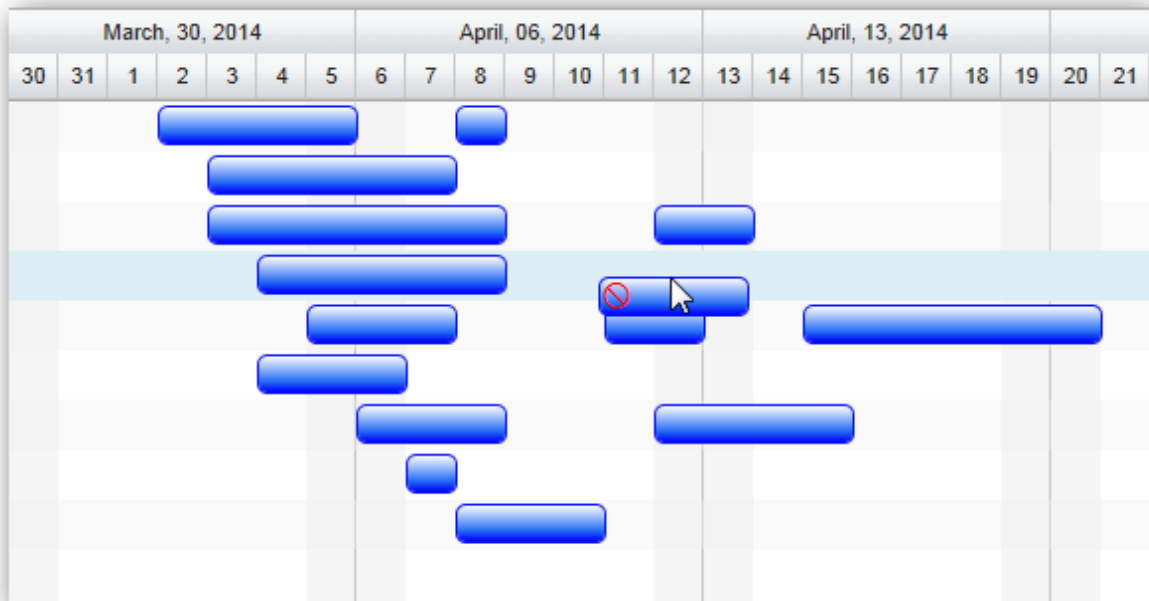
You can turn on this feature with this boolean property:

```
RadiantQ.FlexyGantt.ShiftTrackerGlobal.IncludeVerticalMovingSupport = true;
```

Once you turn on this feature, appropriate events will be raised as the user drags the bars around where you can determine if a drop is supported at a particular location and process the drop. These are the events raised that you should listen to:

```
RadiantQ.FlexyGantt.ShiftTrackerGlobal.BeginVerticalDrag.subscribe(function (sender, args) {  
    // This event is raised before the starting of dragging.  
});  
RadiantQ.FlexyGantt.ShiftTrackerGlobal.VerticalDragOver.subscribe(function (sender, args) {  
    // This event is raised on the process of dragging.  
});  
RadiantQ.FlexyGantt.ShiftTrackerGlobal.VerticalDragDrop.subscribe(function (sender, args) {  
    // This event is raised after you dropped over the row.  
});
```

Among other things you can implement a drag cursor that can be shown at the drop location as the user drag a bar around, as follows:



Dragging a bar with a special no-drop cue

This feature is illustrated in the following sample:

- In HTML : ..\Samples\SingleGanttTaskDandD.htm.
- In ASP.NET MVC : ..\Views\Home\FlexyGantt\SingleGanttTaskDandD.cshtml.
- In ASP.NET : ..\Samples\FlexyGantt\SingleGanttTaskDandD.aspx.

How Tos

RadiantQ jQuery Gantt Package

How to Convert X to Time and Time to X in the Chart?

How to Convert X to Time and Time to X in the Chart?

Often you might want to convert an X location in the Gantt Chart to a Date value (based on the current time scale headers settings) and vice versa. This is easily implemented using the following methods in GanttChart:

```
var gantt = $gantt_container.data("FlexyGantt");
// xPos is a x value in GanttChart co-ordinates
gantt.GetGanttChart().data('GanttChart').ConvertXToTime(xPos);
// xPos is a x value in GanttChart co-ordinates
gantt.GetGanttChart().data('GanttChart').ConvertTimeToX(aDateTime);
```

RadiantQ jQuery Gantt Package

How to refresh Gantt with new data with same schema

Refreshing with new data with same schema

To refresh gantt with same schema (same bound property names) you only have to set the DataSource of GanttControl to null first and then reset DataSource property again with the new data. The below code will explain how to do this.

```
// First set the source to null and then set the new source.  
$gantt_container.FlexyGantt("option", "DataSource", data);
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

How to listen the taskbar size changes in FlexyGantt?

Sometime we might want to listen the taskbar size changes to update the div position in which is inside the taskbar div for example progress bar in flexygantt.

```
// The template that defines the look for the task bars. "taskbar-style" is a
built-in style that defines a default look for the task bars. The method
taskBarSizeChanged(passed as attribute value of SizedChanged will be called when the
taskbar position recomputed.
    var tTemplate = "<div SizeChanged ='taskBarSizeChanged'><div
class='dragThumb'></div><div class='progress-bar'></div></div>";

//Called when taskbar size chnged. Here we update the progress bar's width.
function taskBarSizeChanged(boundData, ganttChart) {
    var startInX = ganttChart.ConvertTimeToX(boundData.StartTime);
    var endInX = ganttChart.ConvertTimeToX(boundData.EndTime);
    var progressEndPos = startInX + ((endInX - startInX) * boundData.progress / 100);
    var progressEndTime = ganttChart.ConvertXToTime(progressEndPos);
    var width = ganttChart.ConvertTimeToX(progressEndTime) -
ganttChart.ConvertTimeToX(boundData.StartTime);           //To cache the element
after the template rendered.

    var progressBar = $("div.progress-bar", this);
    progressBar.width(width);
}
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

How to listen the taskbar click event in FlexyGantt?

We can listen the taskbar click event on "OnTaskBarLoad" function like below.

```
$gantt_container.FlexyGantt({
    .....
    .....
    OnTaskBarLoad: function (task) {
        var $taskbar = task.element;
        $taskbar.unbind("click.onTaskBarClick");//To unbind the click event.
        $taskbar.bind("click.onTaskBarClick", $.proxy(function (event) {
            $(document).unbind(".TaskItemControl");//To disable the pop-up.
            alertTaskName(task);
            event.stopPropagation();
        }));
    },
    .....
    .....
});

function alertTaskName(task)
{
    alert("Name:" + task.options.Data.TaskName);
}
```

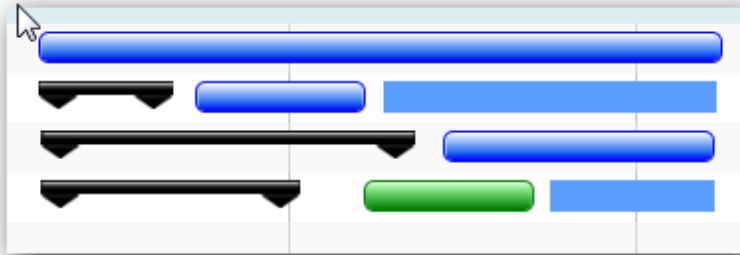
© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

How to show different taskbars in single row?

Sometime user wants to differentiate the tasks from the others tasks in a same row to represent and highlight the taks, Here It is possible to show the different type of taskbars in the single row of the GanttChart.



Customized Task Bars

sample json:

```
[
  {
    "ResourceGroupName" : "Group1",
    "ResourceGroupID" : "Grp1",
    "Tasks": [
      {
        "TaskName": "Task 1",
        "StartTime": "2014-04-02T00:00:00Z",
        "EndTime": "2014-04-09T08:00:00Z",
        "Type": "Order"
      },
      .....
    ],
    "Resources" : [
      {
        "ResourceName" : "Resource1",
        "ResourceID" : "Res1",
        "Tasks": [
          {
            "TaskName": "Task 2",
            "StartTime": "2014-04-03T00:00:00Z",
            "EndTime": "2014-04-09T06:00:00Z",
            "Type": "ProductionOrder"
          },
          .....
        ]
      },
      .....
    ]
  },
  .....
]
```

Sample Code:

```

// The template that defines the look for the task bars. "rq-gc-taskbar" is a
// built-in style that defines a default look for the task bars. And added some more
// elements as a child which contain the style of the parentbar.
var tTemplate = "<div class='rq-gc-taskbar'><div
class='rq-gc-parentBar-leftCue parentStyle'></div><div class='rq-gc-parentBar-middle
parentStyle'></div><div class='rq-gc-parentBar-rightCue parentStyle'></div><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div><div
class='start-resizeThumb'></div></div>";

// Setting the TaskItemTemplate and ParentTaskItemTemplate as same template.
$gantt_container.FlexyGantt({
    .....
    .....
    TaskItemTemplate: tTemplate,
    ParentTaskItemTemplate: tTemplate,
    OnTaskBarLoad: function (task) {
        // Check for the Type.
        if (this[0].DataContext.Type == "ProductionOrder") {
            //If it is "ProductionOrder" then Add new class to remove the
            TaskBarStyle to make the TaskBar like the ParentBar.
            $(this).addClass('customBarStyle');
        }
        else {
            // If it is a Not a ProductionOrder then remove the child elements
            from the bar to show the TaskBar alone.
            $('.parentStyle', $(this)).remove();
            //If it is "StockOrder" then Add new class to customize the taskbar
            if(this[0].DataContext.Type == "StockOrder")
                $(this).addClass('templatedTaskbar');
            //If it is "StockOrder" then Add new class to customize the taskbar
            if (this[0].DataContext.Type == "Maintenance")
                $(this).addClass('greenBarStyle');
        }
    },
    .....
    .....
});

```

This is illustrated in this samples:

In HTML : ..\Samples\FlexyGanttTaskbarCustomization.htm.
 In ASP.NET MVC : ..\Views\Home\FlexyGantt\FlexyGanttTaskbarCustomization.cshtml.
 In ASP.NET : ..\Samples\FlexyGantt\FlexyGanttTaskbarCustomization.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

How to prevent Taskbar Overlapping?

In some cases user don't want their tasks to overlap with the other, the below code describes how to prevent this by shifting the overlapped tasks away from the currently resized or moved task.

Sample Code:

```

$(document).ready(function () {
  var flexyGantt = $gantt_container.data("FlexyGantt");
  flexyGantt.TaskTimeChanged.subscribe(function (sender, args) {
    var flexyNodeData = args.FlexyNodeData;
    $currentChartRow = flexyNodeData.ChartRow();
    var rowData = $currentChartRow[0]["data-grid-item"];

    var movedTask = args.DataSource; // moved or resized bar.

    // Persisting the new values into the data source right away. If we
    // don't do this, the gantt will do this after you return from this function.
    movedTask.StartTime = args.StartTime;
    movedTask.EndTime = args.EndTime;

    var inspectTimeRange = {
      StartTime: movedTask.StartTime,
      EndTime: movedTask.EndTime
    }

    var redrawBars = false;
    var currentRowTasks = rowData.Data().Tasks;

    //Sorting tasks by startTime.
    currentRowTasks.sort(function (a, b) { return a.StartTime -
    b.StartTime });

    if (args.Type == "ResizeStart")//task resize.
    {
      //Moving tasks from right to left while resizing task's start.
      for (var i = currentRowTasks.length - 1; i >= 0; i--) {
        var task = currentRowTasks[i];
        if (movedTask != task) {
          var isOverlapping = preventOverlapping(inspectTimeRange,
task, redrawBars, args);
          if (isOverlapping)
            inspectTimeRange.StartTime = task.StartTime;
        }
      }
    }
    else {
      //Moving tasks from left to right while resizing end and while
      //moving tasks.
      for (var i = 0; i < currentRowTasks.length; i++) {
        var task = currentRowTasks[i];
        if (movedTask != task) {
          var isOverlapping = preventOverlapping(inspectTimeRange,
task, redrawBars, args);
          if (isOverlapping)
            inspectTimeRange.EndTime = task.EndTime;
        }
      }
    }

    // Force a redraw (or if you use a change notifying model like
    // knockout, the gantt will then auto update)
    if (redrawBars == true) {
      $currentChartRow.find("td").data("radiantqTasksListControl"
).RedrawTaskRow()
    }

    // Here you know that changes are made into the "moved task" as well
    // as the "adjusted task". You can now call the server to validate.
  });

  function preventOverlapping(inspectTimeRange, task, redrawBars, args) {

```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

How to add tooltip for a custom element ?

Sometimes, the user likes to set the custom element inside the taskbar and add tooltip for that custom element. The below code will explain how to do this.

```
// The template that defines the look for the task bars. "rq-gc-taskbar" is a
// built-in style that defines a default look for the task bars. Here, we add class and
// style for the custom element.
var tTemplate = "<div class='rq-gc-taskbar' data-bind='TaskColor: EndTime'><span
class='headerTxt' style='display:none;'>radiantq</span><div
class='rq-taskbar-dragThumb'></div><div class='rq-taskbar-resizeThumb'></div></div>";

// Applying your styles on demand.
$flexyGantt_container = $('#flexyGantt_container');
$flexyGantt_container.FlexyGantt({
    .....
    .....
});

// Applying your styles on demand.
updateTextVisibility($flexyGantt_container.data("FlexyGantt"));

function updateTextVisibility(gantt) {
    var chart = gantt.GetGanttChartInstance();
    chart.AfterChartRowsRender.subscribe(function () {
        $(".headerTxt").css('display', 'inline-block').RQTooltip(getTooltipContent);
    });
}

// Function to return the content of headerTxt's tooltip.
function getTooltipContent($this, evt) {
    // To remove the taskbar's tooltip in headerTxt element.
    $(".TaskTooltip").parent().remove();

    // To prevent the taskbar's tooltip from displaying in headerTxt element.
    evt.stopPropagation();

    // Returns the content of the tooltip.
    return "<div>Custom Tooltip</div>";
}
}
```

Update custom element visibility after printing to keep the on demand styles.

```
$("#print").click(function () {
    var options = new RadiantQ.Gantt.PrintOptions();
    options.AfterPrintComplete.subscribe(function () {
        // Applying your styles on demand.
        updateTextVisibility($flexyGantt_container.data("FlexyGantt"));
    });
    RadiantQ.Gantt.RQPrint(selectedGantt, options);
});
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

How to maintain the current state of the Expand/Collapse nodes after data source reset?

Sometimes, you want to maintain the current state of the Expand/Collapse nodes. Here is the code that shows how to maintain the current state of the Expand/Collapse node after datasource reset in button click.

```

$(document).ready(function () {
$gantt_container = $("#gantt_container");
// Initialize the FlexyGantt widget.
$gantt_container.FlexyGantt({
.....
.....
FlatItemsSourceCreated: function (sender, args) {
    var cachedItemsList = clonedHierItemsList;
    if (cachedItemsList && args.Source.FlatItemsSource) {
        for (var i = 0; i < cachedItemsList.length; i++) {
            // Update the expand and collapse state based on previous gantt view.
            if (cachedItemsList[i].isExpanded)
                args.Source.FlatItemsSource[i].HierarchicalItem.IsExpanded(true);
            else
                args.Source.FlatItemsSource[i].HierarchicalItem.IsExpanded(false);
        }
    }
};

flexyGantt = $gantt_container.data("FlexyGantt");
var ganttDataSource = flexyGantt.options.DataSource;
var clonedHierItemsList = [];
$("#reset").click(function () {
    jQuery.extend(true, clonedHierItemsList, flexyGantt.FlatHierarchicalItemsList)

    //flexyGantt.options.DataSource = null; // In Flexygantt, it's not necessary to
    null the data source.
    $gantt_container.FlexyGantt("option", "DataSource", ganttDataSource);
});

```

RadiantQ jQuery Gantt Package

How to make reflect the bound data property changes into the gantt (view)?

To make reflect the bound data property changes into the gantt(view) you have to refresh the grid cell with currently changed item to update the view during Undo/Redo. Here is the code that shows how to do that.

```

$.ajax({
    .....
    .....
    success: function (data) {

        for (var i = 0; i < data.length; i++) {
            var resources = data[i].Resources;
            // For each task object
            for (var j = 0; j < resources.length; j++) {
                var tasks = resources[j].Tasks;
                var resource = resources[j];
                // Make RName fire PropertyChanged on value change.
                RadiantQ.Gantt.Utils.InjectGetAndSetOnData(resource, "RName");
                resource.PropertyChanged.subscribe(onPropertyChanged);

                function onPropertyChanged(data, args) {
                    if (args.PropertyName == "RName") {
                        var flexyGantt = $("#gantt_container").data("FlexyGantt");
                        var grid = flexyGantt.grid;
                        var curItem =
flexyGantt.FlatItemsSource.GetItemFromDataSource(data);
                        // Refresh grid cell with current changed item to update the
view during "Undo/Redo".
                        if (curItem)
                            grid.RefreshItem(curItem);
                    }
                }
            }
        }

        self.jsonData = data;
        $.holdReady(false);
    }
});

//Column definition.
var columns = [
    {
        field: "RName",
        title: "Name",
        editor: RadiantQ.Default.Template.FlexyGanttExpandableTextBoxEditor(
"nameConverter"),
        template:
RadiantQ.Default.Template.FlexyGanttExpandableTextBlockTemplate("nameConverter"),
        isParentEditable: false,
        width:200
    }
];

```

RadiantQ jQuery Gantt Package

How to listen the node Expand/Collapse state in FlexyGantt?

Sometimes, user might want to listen the `NodeExpansionStateChanged` event, which is raised when the node Expand/Collapse changed state in FlexyGantt. The below code will explain how to do this.

```
$(document).ready(function (){
    .....
    .....

    var flexyGantt = $gantt_container.data("FlexyGantt");
    flexyGantt.FlatItemsSource.NodeExpansionStateChanged.subscribe(function
    (hierData) {
        var isExpanded = hierData.IsExpanded();
    });
});
```

RadiantQ jQuery Gantt Package

How to add data asynchronously while on expanding the node?

"dynamicResolverFunction" is called everytime the user expands a node in the grid and which will add the child data asynchronously while on expanding the node. Here is the code that shows how to do that.

```

$( "#gantt_container" )..FlexyGantt({
    .....
    .....
    dynamicResolverFunction: function (data, hierItem) {
        if (hierItem.isExpanding) {
            if (data.Resources.length == 0) {
                // To let gantt know the child items processed async.
                hierItem.isUpdatingChildListAsync = true;
                getChildTasksAsync(data, hierItem);
                return [];
            }
            else
                return data.Resources;
        }
        // Return an empty array to stay the row as parent.
        return [];
    }

    getChildTasksAsync = function (data, hierItem) {
        // Doing this in a 2 secs timeout, just to simulate loading data from server.
        setTimeout(function () {
            var childData = new Array();
            for (i = 1; i <= 50; i++) {
                var resources = {
                    "RName": "Resource" + i,
                    "Tasks": [{
                        "TaskName": "Task 1",
                        "StartTime": Date.today().addDays(-10),
                        "EndTime": Date.today().addDays(1),
                        "Progress": 20
                    },
                    {
                        "TaskName": "Task 2",
                        "StartTime": Date.today().addDays(2),
                        "EndTime": Date.today().addDays(20),
                        "Progress": 50
                    }
                ]
            }
            childData.push(resources);
        }
        hierItem.UpdateDynamicResolvedChildList(childData);
        return childData;
    }, 2000);
});

```

This is illustrated in this samples:

In HTML : ..\Samples\FGLoadChildrenOnDemand.htm

In ASP.NET MVC : ..\Views\Home\FlexyGantt\FGLoadChildrenOnDemand.cshtml.
In ASP.NET : ..\Samples\FlexyGantt\FGLoadChildrenOnDemand.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

How to insert/remove items dynamically?

We recommend you to add or remove items in FlexyGantt items using "\$.observable" utility to let Gantt to update those changes immediately.

Available methods of *\$.observable*

- `$.observable(data).insert(item);`
To insert the item into the data at the end. This is equivalent of push method in array.
- `$.observable(data).insert(index, item);`
To insert the item into the given index in data.
- `$.observable(data).remove(index);`
To remove the item at the given index in data.

Here is the code usage,

```
// Inserting new Team at the index of 1.
$.observable(jsonData).insert(1/*Index to insert*/, newTeam);
// Inserting new Resource at the index of 1.
$.observable(jsonData[0].Resources).insert(1/*Index to insert*/, newResource);
// Inserting new Task at the end of items.
$.observable(jsonData[0].Resources[0].Tasks).insert(1/*Index to insert*/, newTask);

// Removing Team at the index of 1.
$.observable(jsonData).remove(1/*Index to remove*/);
// Removing Resource at the index of 1.
$.observable(jsonData[0].Resources).remove(1/*Index to remove*/);
// Removing Task at the index of 0.
$.observable(jsonData[0].Resources).remove(0/*Index to remove*/);
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

How to handle the overlapping state?

"AreTasksOverlappingCallBack" is used to handle the state of overlapping whether the task is overlapped or not. User can set the overlapping state by just returning the true or false in this callback function.

Here is the code Example.

```

$('#gantt_container').FlexyGantt({
  GanttChartTemplateApplied: function (sender, args) {
    var $GanttChart = args.element;
    $GanttChart.GanttChart({
      AreTasksOverlappingCallBack: function (entity, taskItemControl,
isOverlapped) {
        if (isOverlapped) {
          // Checking whether the overlapping task's type and the other
task's type on same row is same.
          // Here we checked, 'Task 1' and 'Task 2' (Team1-Resource2)
          if (entity.options.Data.IsOverlapType &&
taskItemControl.options.Data.IsOverlapType)
            return true; // Enables overlapping for specic task's types.
          else
            return false; // Breaks the default overlapping behaviour.
        }
      },
    });
  },
});

```

In this below json data have "IsOverlapType": true for which task needs to be handled in "AreTasksOverlappingCallBack" function to prevent default overlapping feature.

```
[
  {
    "TName": "Team1",
    "PStartTime": "2014-04-02T00:00:00Z",
    "PEndTime": "2014-04-22T00:00:00Z",
    "Resources": [
      {
        "RName": "Resource 1",
        "PStartTime": "2014-04-04T00:00:00Z",
        "PEndTime": "2014-04-22T00:00:00Z",
        "Tasks": [
          {
            "TaskName": "Task 1",
            "StartTime": "2014-04-06T00:00:00Z",
            "EndTime": "2014-04-14T00:00:00Z",
            "Progress": 20,
            "IsOverlapType": true
          },
          {
            "TaskName": "Task 2",
            "StartTime": "2014-04-12T00:00:00Z",
            "EndTime": "2014-04-18T00:00:00Z",
            "Progress": 90,
            "IsOverlapType": true
          }
        ]
      },
      {
        "RName": "Resource 2",
        "PStartTime": "2014-04-06T00:00:00Z",
        "PEndTime": "2014-04-18T00:00:00Z",
        "Tasks": [
          {
            "TaskName": "Task 1",
            "StartTime": "2014-04-10T00:00:00Z",
            "EndTime": "2014-04-16T00:00:00Z",
            "Progress": 20
          },
          {
            "TaskName": "Task 2",
            "StartTime": "2014-04-06T00:00:00Z",
            "EndTime": "2014-04-12T00:00:00Z",
            "Progress": 70
          }
        ]
      },
      {
        "RName": "Resource 3",
        "PStartTime": "2014-04-06T00:00:00Z",
        "PEndTime": "2014-04-06T00:00:00Z",
        "Tasks": [
          {
            "TaskName": "Task 1",
            "StartTime": "2014-04-06T00:00:00Z",
            "EndTime": "2014-04-12T00:00:00Z",
            "Progress": 20,
            "IsOverlapType": true
          },
          {
            "TaskName": "Task 2",
            "StartTime": "2014-04-13T00:00:00Z",
            "EndTime": "2014-04-18T00:00:00Z",
            "Progress": 70,
            "IsOverlapType": true
          }
        ]
      }
    ]
  }
]
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

Common Topics

Getting Started

RadiantQ jQuery Gantt Package

Cleaning Up Src Folder

In the Getting Started sections, we had recommended to copy over the <install path>/Src folder into your project file. But, this folder does include a lot of files that are not necessary for all projects.

These are the files you should decide whether or not to keep in your project.

1) Globalization Date files.

These files define date-time formats for specific cultures.

The default date-time formats are defined in this file:

..\Src\Scripts\Utils\date.js

You can choose to delete everything or keep only some of the files inside this folder:

..\Src\Scripts\Utils\globalization

2) jquery.xml2json.js

This file is only needed if you are going to convert any data from XML to JSON.

3) jQuery files

By default Gantt SRC includes the jQuery 1.7, 1.8, 1.9 and 2.0 files. You will only need to reference one of these, so you can remove the rest.

4) knockout js

If you are not using any KO binding in your project, you can remove the following files.

knockout-2.0.0.js

knockout-2.2.1.js

knockout-2.3.0.js

If you are using KO, refer only to one of the above files.

Include this file below as well if you are referencing any one of the above.

knockout.mapping-latest.debug.js

5) jQuery-ui-themes

The gantt widget always needs the base(Src\Styles\jQuery-ui-themes\smoothness) CSS files. So, don't delete this folder.

However, you can remove the rest of the sub-folders in the <Src\Styles\jQuery-ui-themes> folder, whichever themes you don't plan to use.

6) Sample level Script files

html2canvas.js- used in Printing/Image export.

canvg.js - Used by the RadiantQ APIs that generate a gradient image dynamically during

runtime (for use typically as background in task bars)

If you are not using a feature, you can remove the corresponding files.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

Gantt Basics

RadiantQ jQuery Gantt Package

Virtualization In Gantt

Gantt uses 2 views to render the tasks or resources - the grid or table view and the chart view. These two views can be rendering virtualized as discussed below.

Use of Virtualization

Virtualization in the context of gantt is rendering only those elements that are currently visible in the vertical scroll view. This allows us to support visualizing 1000s of tasks at the same time without unduly affecting the rendering performance, etc.

Without turning on Virtualization in the gantt, the gantt would typically begin to slow down when there are more than 2000 rows to render. But, with virtualization turned on, you will see much improved performance for any number of rows (but note that with the GanttControl, building the gantt's internal model could slow down as the number of task items goes up).

Setting Virtualization:

You can turn on virtualization in the gantt as follows (this enables virtualization on both the grid and chart position):

In HTML

```
// Setting "UseVirtualization" option in gantt to "true" will turn on virtualization
in the both gantt table and chart.
$gantt_container.GanttControl({
    UseVirtualization: true,
});
```

In ASP.NET MVC

```
// Setting "UseVirtualization" option in gantt to "true" will turn on virtualization
in the both gantt table and chart.
@Html.JQProjectGantt(
    new JQProjectGanttSettings()
    {
        Options = new ProjectGanttOptions()
        {
            ..
            UseVirtualization=true,
            GanttChartOptions = new GanttChartOptions()
            {
                AnchorTime = DateTime.Today
            }
        }
    })
```

In ASP.NET

```
<RQ:GanttControl ID="gantt" UseVirtualization="true" .." Height="500px" runat="server"
/>
```

There is however a drawback in turning virtualization on with regards to rendering dependency lines. Since only the visible rows are rendered, if a dependency line's start or end task is not visible then the line will only be partly rendered and if both the start and end tasks are not visible, then the dependency line will not be rendered at all.

Chart View Virtualization

Optionally, you can turn on virtualization in the gantt chart region alone as follows:

In HTML

```
// Setting "UseChartVirtualization" option in gantt to "true" will turn on
virtualization only in the gantt chart.
$gantt_container.GanttControl({
    UseChartVirtualization: true,
});
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Round To Options

Round To Options

By default, the gantt allows the end user to specify times down to seconds. This is often undesirable, you might only want the end-user to specify times in days, half-days, hours, etc. The gantt exposes a property that allows you to specify this "round up" settings:

In HTML

```
$('#container').FlexyGantt({
  DataSource: self.jsonData,
  RoundTimeEditsTo: RadiantQ.Gantt.RoundToOptions.Day,
  ....
  ....
})
```

In ASP.NET MVC

```
@Html.JQFlexyGantt(
  new JQFlexyGanttSettings()
  {
    ControlId = "gantt_container",
    AfterGanttWidgetInitializedCallback = "AfterGanttWidgetInitializedCallback",
    DataSourceUrl = new Uri("/Home/GetFlexyGanttItemSource", UriKind
.RelativeOrAbsolute),
    Options = new FlexyGanttOptions()
    {
      RoundTimeEditsTo = RoundToOptions.Day
    }
  }
)
```

In ASP.NET

```
<RQ:FlexyGantt runat="server" ID="gantt" Height="500" RoundTimeEditsTo="day" ... />
```

The other round to options available are:

Auto, // currently uses 15 min rounding

FifteenMinutes,

ThirtyMinutes,

Hour,

Day,

HalfDay

This round up setting is enforced when the end-user tries to resize the length of a task, edit the times of a task, etc.

This is illustrated in this samples:

In HTML : ..\Samples\RoundsTimesToDay.htm.
In ASP.NET MVC : ..\Views\Home\ProjectGantt\RoundsTimesToDay.cshhtml.
In ASP.NET : ..\Samples\ProjectGantt\RoundsTimesToDay.aspx.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

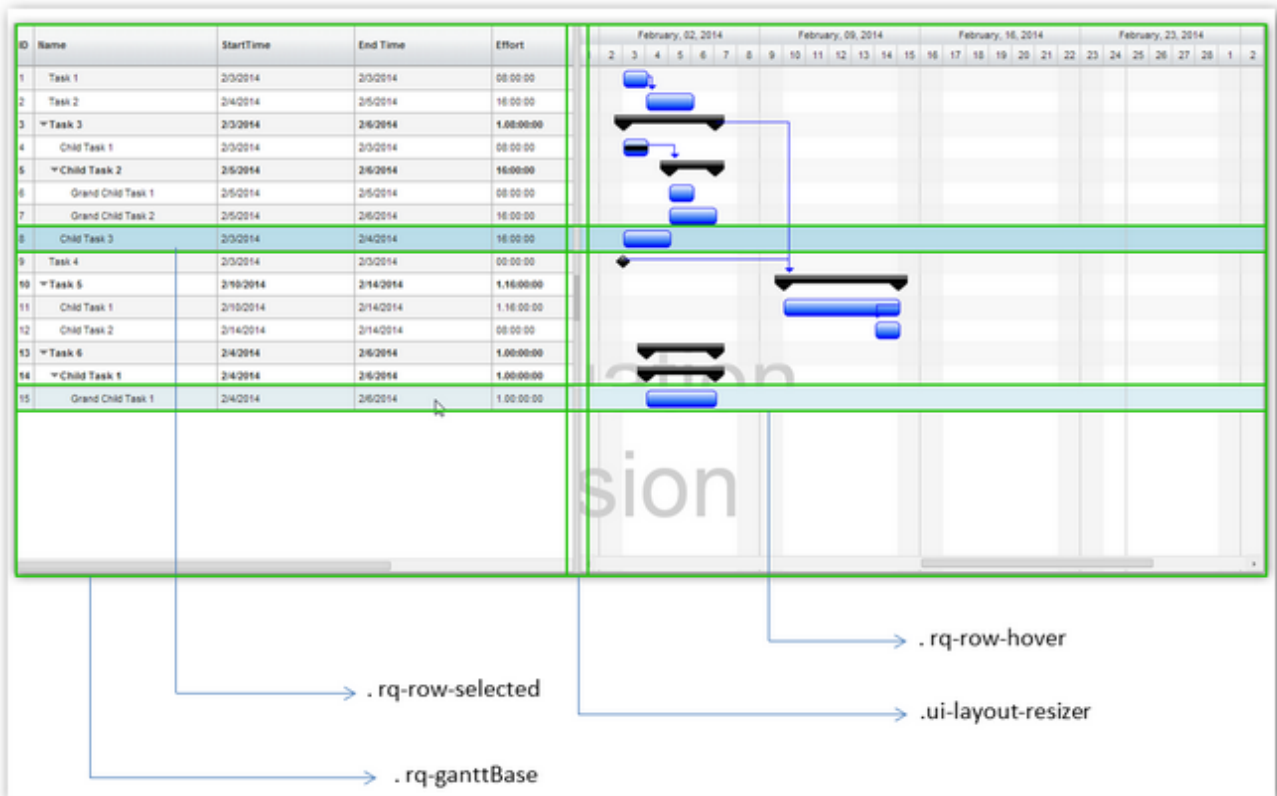
Appearance Customization

RadiantQ jQuery Gantt Package

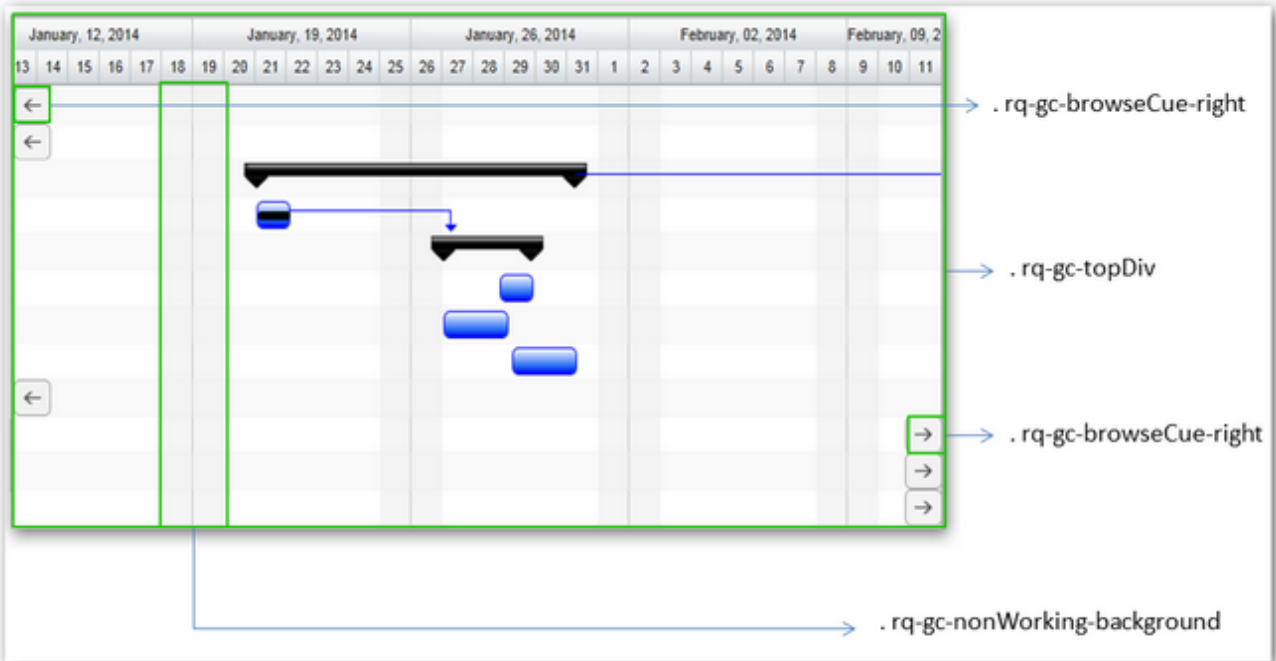
Gantt CSS Styles

The CSS styles defined by the gantt in the radiantq.gantt.default.css file is annotated against the corresponding portion of the gantt.

Top Level Gantt Styles



Commonly referenced Chart Styles



Less commonly referenced Chart Element Styles

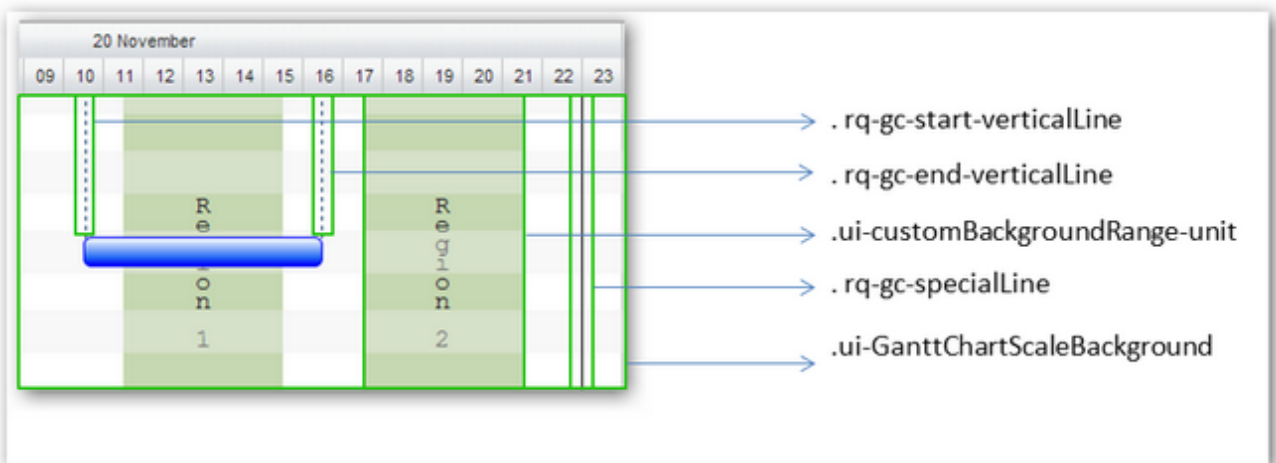
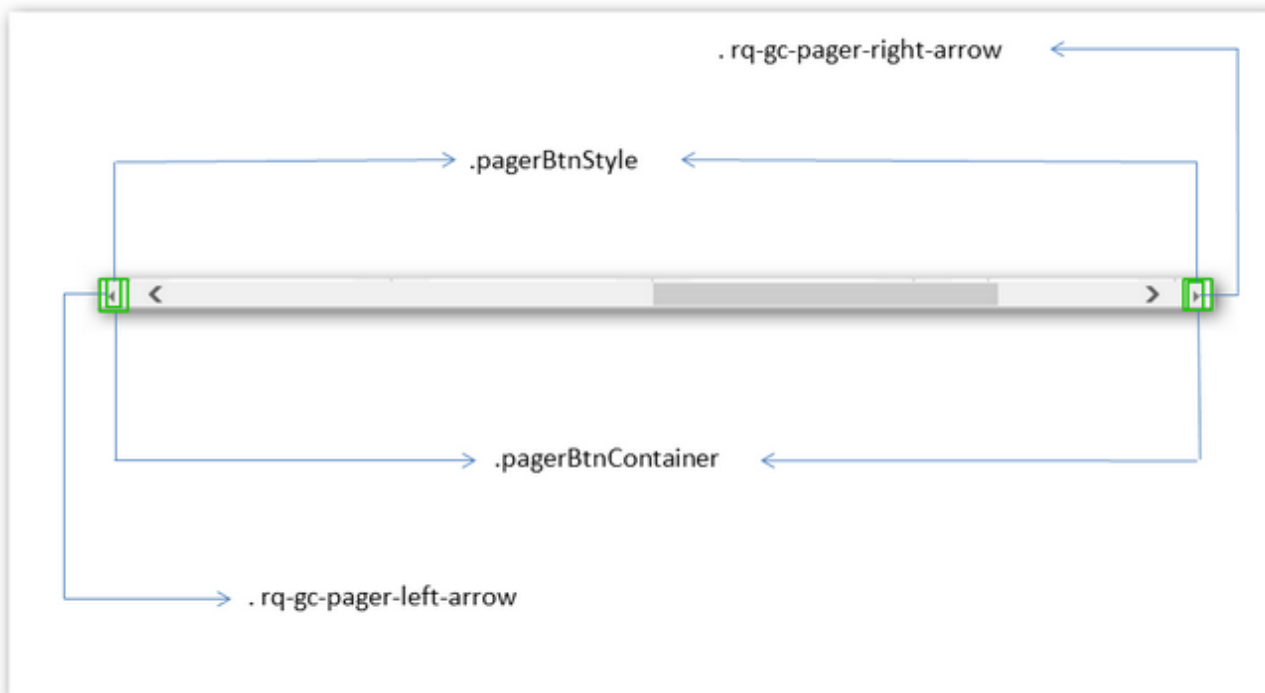
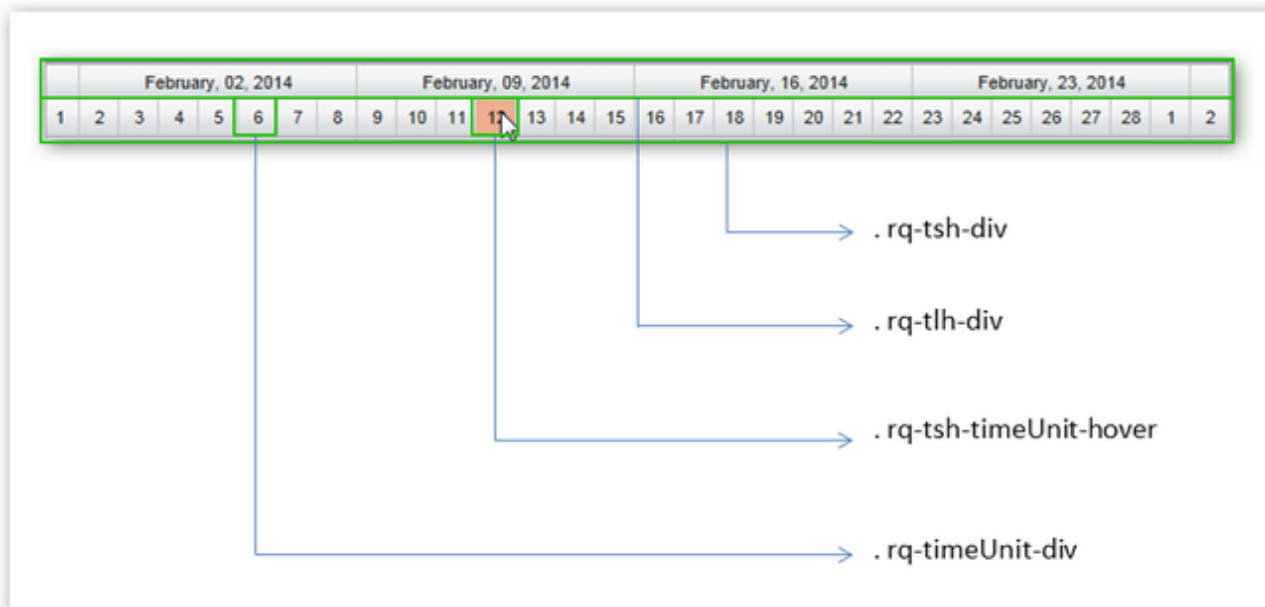


Chart Pager Buttons Styles



Time Line Header Styles



Grid Styles

ID	Name	StartTime	End Time	Effort
1	Task 1	2/3/2014	2/3/2014	08:00:00
2	Task 2	2/4/2014	2/5/2014	16:00:00
3	Task 3	2/3/2014	2/6/2014	1:08:00:00
4	Child Task 1	2/3/2014	2/3/2014	08:00:00
5	Child Task 2	2/5/2014	2/6/2014	16:00:00
6	Grand Child Task 1	2/5/2014	2/5/2014	08:00:00
7	Grand Child Task 2	2/5/2014	2/6/2014	16:00:00
8	Child Task 3	2/3/2014	2/4/2014	16:00:00
9	Task 4	2/3/2014	2/3/2014	00:00:00
10	Task 5	2/10/2014	2/14/2014	1:16:00:00
11	Child Task 1	2/10/2014	2/14/2014	1:16:00:00
12	Child Task 2	2/14/2014	2/14/2014	08:00:00
13	Task 6	2/4/2014	2/6/2014	1:00:00:00

Annotations:

- `.gantt-table-header` points to the header row.
- `.ui-table-head-bg` points to the background of the header row.
- `.rq-grid-collapse-arrow` points to the collapse arrow in the ID column of row 3.
- `.rq-grid-alternative-background` points to the background of row 9.
- `.rq-grid-expand-arrow` points to the expand arrow in the ID column of row 13.

RadiantQ jQuery Gantt Package

Special Lines

Special Lines

Special Lines are lines that can be drawn at specific times in the GanttChart to indicate milestones, deadlines or other times of interest.

Here is a sample code that will draw a line at a specific time, with the specific color:

In HTML

```
var SpecialLineInfos = new ObservableCollection();
var deadLine = new RadiantQ.Gantt.SpecialLineInfo()
deadLine.LineDateTime = projectStart.addDays(20);
deadLine.ToolTipText = 'DeadLine';
deadLine.LineColor = 'green';
SpecialLineInfos.add(deadLine);

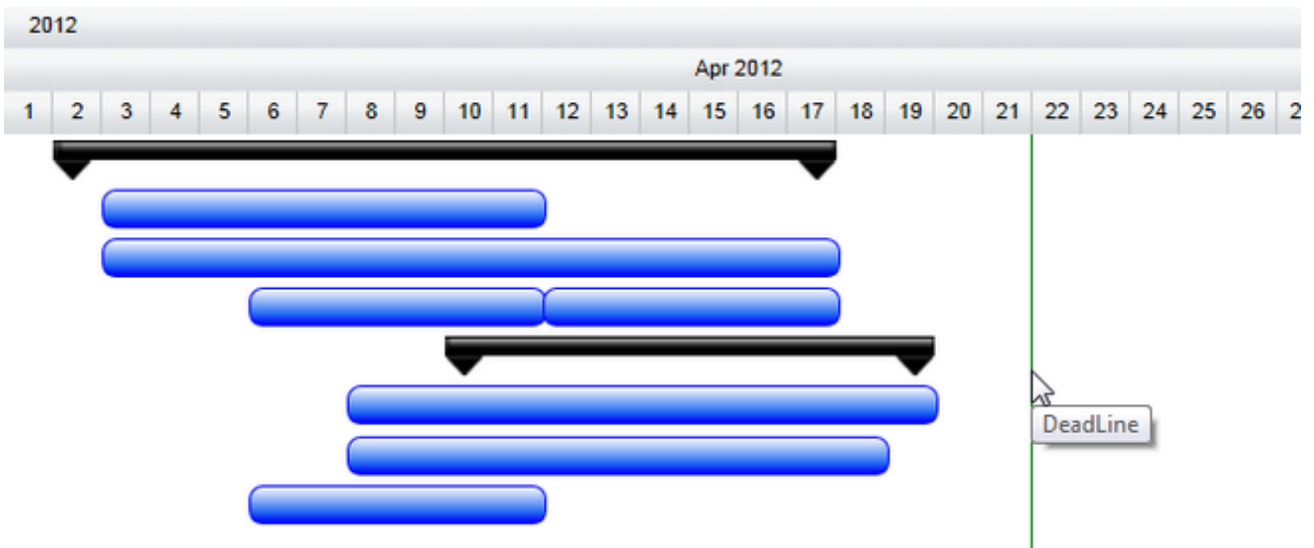
$('#container').FlexyGantt( {
    ...
    SpecialLineInfos: SpecialLineInfos
});
```

In ASP.NET MVC

```
@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        ControlId = "ganttt_container",
        ..
        Options = new FlexyGanttOptions()
        {
            SpecialLineInfos = new System.Collections.ObjectModel.
ObservableCollection<SpecialLineInfo>() { new SpecialLineInfo() { LineColor="green",
LineDateTime=DateTime.Today.AddDays(1), ToolTipText="DeadLine" } },
            GanttChartOptions = new GanttChartOptions()
            {
                AnchorTime = DateTime.Today,
                MovingInfoPopupID = "MovingInfoPopup",
                ResizeInfoPopupID = "ResizeInfoPopup"
            }
        }
    }
)
```

In ASP.NET

```
<script runat="server">
    protected void gantt_Load(object sender, EventArgs e)
    {
        this.gantt.SpecialLineInfos = new System.Collections.ObjectModel.
ObservableCollection<RadiantQ.Web.JQGantt.Common.SpecialLineInfo>() { new
RadiantQ.Web.JQGantt.Common.SpecialLineInfo() { LineColor="green", LineDateTime=
DateTime.Today.AddDays(1), ToolTipText="now" } };
    }
</script>
<RQ:FlexyGantt ID="gantt" OnLoad="gantt_Load"... />
```



GanttChart with special lines

RadiantQ jQuery Gantt Package

Custom Chart Background

CustomBackgroundRanges

Often you might want to "mark" certain time-ranges in the gantt with a custom background to indicate phases or stages in the project. This can be implemented using an API like below. Note that you can either add discreet set of ranges or provide "repeating ranges" via an event.

(This code is from the sample "...Samples\CustomChartBackgroundRanges")

In HTML

```

$(document).ready( function ()
{
    var cgcb = self.CreateCustomChartBackgroundRanges();
    .....
    var $ganttt_container = $("#ganttt_container");
    // Initialize the FlexyGantt widget.
    $ganttt_container.FlexyGantt({
        CustomChartBackgroundRanges: cgcb,
        ...
    } );
} );
// Provides custom range info
function CreateCustomChartBackgroundRanges() {
    var customRangesInfos = new RadiantQ.Gantt.CustomRangesInfos();

    var repeatingRange= RadiantQ.Template("<div class='repeatingRange'
style='background-color:\\#eae4e4;'><div class='childDiv'>" + this
.stringToNewLineChart('Repeating Range') + "</div></div>");
    var phase1 = RadiantQ.Template("<div class='phase1'
style='background-color:\\#c6d8b1;'><div class='childDiv'>" + this
.stringToNewLineChart('Region 1') + "</div></div>");
    var phase2 = RadiantQ.Template("<div class='phase1'
style='background-color:\\#c6d8b1;'><div class='childDiv'>" + this
.stringToNewLineChart('Region 2') + "</div></div>");

    // Discreet Ranges
    customRangesInfos.DiscreetCustomRanges.add(new
RadiantQ.Gantt.CustomRangeInfo(new Date(2014, 10, 20, 6, 0, 0), new Date(2014, 10,
20, 10, 0, 0), phase1));
    customRangesInfos.DiscreetCustomRanges.add(new
RadiantQ.Gantt.CustomRangeInfo(new Date(2014, 10, 20, 12, 0, 0), new Date(2014, 10,
20, 16, 0, 0), phase2));

    customRangesInfos.ProvideRepeatingCustomRanges.subscribe(ProvideRepeatingCustomRanges
Event);

    function ProvideRepeatingCustomRangesEvent(sender, args) {
        var viewSpan = new RQTimeSpan(args.ViewEndTime - args.ViewStartTime);
        // Don't provide repeating ranges above certain zoom levels.
        if (viewSpan.getTotalDays() > 5)
            return;

        var start = args.ViewStartTime.Date().clone();
        while (start < args.ViewEndTime) {
            var end = start.clone().addHours(5);
            args.CustomRanges.push(new
RadiantQ.Gantt.CustomRangeInfo(start.clone(), end.clone(), repeatingRange));
            start = start.clone().addDays(1);
        }
    }
    return customRangesInfos;
}
}

```

In ASP.NET MVC

```

<script type="text/javascript">
    // Provides custom range info
    function CreateCustomChartBackgroundRanges() {
        var customRangesInfos = new RadiantQ.Gantt.CustomRangesInfos();
        var repeatingRange = RadiantQ.Template("<div class='repeatingRange'
style='background-color:\\#eae4e4;'><div class='childDiv'>" + this
.stringToNewLineChart('Repeating Range') + "</div></div>");
        var phase1 = RadiantQ.Template("<div class='phase1'
style='background-color:\\#c6d8b1;'><div class='childDiv'>" + this
.stringToNewLineChart('Region 1') + "</div></div>");
        var phase2 = RadiantQ.Template("<div class='phase1'
style='background-color:\\#c6d8b1;'><div class='childDiv'>" + this
.stringToNewLineChart('Region 2') + "</div></div>");

        // Discreet Ranges
        var today=new Date().Date();
        customRangesInfos.DiscreetCustomRanges.add(new
RadiantQ.Gantt.CustomRangeInfo(new Date(today.clone().addHours(6)), new
Date(today.clone().addHours(10)), phase1));
        customRangesInfos.DiscreetCustomRanges.add(new
RadiantQ.Gantt.CustomRangeInfo(new Date(today.clone().addHours(12)), new
Date(today.clone().addHours(16)), phase2));

customRangesInfos.ProvideRepeatingCustomRanges.subscribe(ProvideRepeatingCustomRanges
Event);
        function ProvideRepeatingCustomRangesEvent(sender, args) {
            var viewSpan = new RQTimeSpan(args.ViewEndTime - args.ViewStartTime);
            // Don't provide repeating ranges above certain zoom levels.
            if (viewSpan.getTotalDays() > 5)
                return;

            var start = args.ViewStartTime.Date().clone();
            while (start < args.ViewEndTime) {
                var end = start.clone().addHours(5);
                args.CustomRanges.push(new
RadiantQ.Gantt.CustomRangeInfo(start.clone(), end.clone(), repeatingRange));
                start = start.clone().addDays(1);
            }
        }
        return customRangesInfos;
    }
</script>

```

```

@Html.JQProjectGantt(
    new JQProjectGanttSettings()
    {
        ControlId = "ganttt_container",
        AfterGanttWidgetInitializedCallback="AfterGanttWidgetInitializedCallback",
        DataSourceUrl = new Uri("/Home/GanttControlItemSource_EffortInHours", UriKind
.RelativeOrAbsolute),
        Options = new ProjectGanttOptions()
        {
            BaseTimeUnitWidth = 20,
            RowHeight = 25,
            TimeScaleHeaders = new TimeScaleHeaderDefinitions(){
                new TimeScaleHeaderDefinition(){
                    Type = TimeScaleType.Days,
                    Name="daysHeader",
                    TextFormat="dd MMMM",
                },
                new TimeScaleHeaderDefinition(){
                    Type = TimeScaleType.Hours,
                    Name="hourHeader"
                }
            },
            CustomChartBackgroundRanges = "CreateCustomChartBackgroundRanges",

```

In ASP.NET

```

<script type="text/javascript">
    // Provides custom range info
    function CreateCustomChartBackgroundRanges() {
        var customRangesInfos = new RadiantQ.Gantt.CustomRangesInfos();
        var repeatingRange = RadiantQ.Template("<div class='repeatingRange'
style='background-color:\\#eae4e4;'><div class='childDiv'>" + stringToNewLineChart(
'Repeating Range') + "</div></div>");
        var phase1 = RadiantQ.Template("<div class='phase1'
style='background-color:\\#c6d8b1;'><div class='childDiv'>" + stringToNewLineChart(
'Region 1') + "</div></div>");
        var phase2 = RadiantQ.Template("<div class='phase1'
style='background-color:\\#c6d8b1;'><div class='childDiv'>" + stringToNewLineChart(
'Region 2') + "</div></div>");

        // Discreet Ranges
        var today=new Date().Date();
        customRangesInfos.DiscreetCustomRanges.add(new
RadiantQ.Gantt.CustomRangeInfo(new Date(today.clone().addHours(6)), new
Date(today.clone().addHours(10)), phase1));
        customRangesInfos.DiscreetCustomRanges.add(new
RadiantQ.Gantt.CustomRangeInfo(new Date(today.clone().addHours(12)), new
Date(today.clone().addHours(16)), phase2));

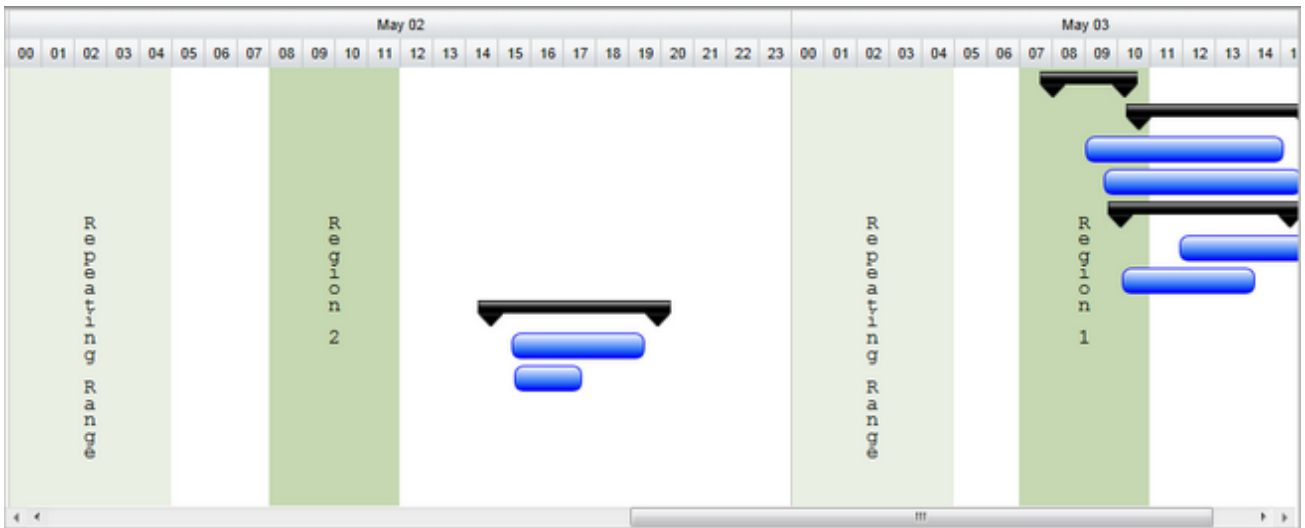
customRangesInfos.ProvideRepeatingCustomRanges.subscribe(ProvideRepeatingCustomRanges
Event);

    function ProvideRepeatingCustomRangesEvent(sender, args) {
        var viewSpan = new TimeSpan(args.ViewEndTime - args.ViewStartTime);
        // Don't provide repeating ranges above certain zoom levels.
        if (viewSpan.getTotalDays() > 5)
            return;
        var start = args.ViewStartTime.Date().clone();
        while (start < args.ViewEndTime) {
            var end = start.clone().addHours(5);
            args.CustomRanges.push(new
RadiantQ.Gantt.CustomRangeInfo(start.clone(), end.clone(), repeatingRange));
            start = start.clone().addDays(1);
        }
    }
    return customRangesInfos;
}
</script>

<RQ:GanttControl ID="ganttt" DataSourceUrl="../../../DataSources/TaskListHandler.ashx"
TimeRangeHighlightBehavior="HighlightInChartOnHeaderMouseHover"
Height="500px" runat="server">
    <TimeScaleHeaders>
        <GanttBase:TimeScaleHeaderDefinition Type="Days" Name="daysHeader"
TextFormat="dd MMMM" />
        <GanttBase:TimeScaleHeaderDefinition Type="Hours" Name="hoursHeader"/>
    </TimeScaleHeaders>
</RQ:GanttControl>

```

Here is the resultant gantt with custom background ranges:



GanttChart with custom background ranges

This is illustrated in this samples:

- In HTML : ..\Samples\CustomChartBackgroundRanges.htm.
- In ASP.NET MVC : ..\Views\Home\Common\CustomChartBackgroundRanges.cshtml.
- In ASP.NET : ..\Samples\Common\CustomChartBackgroundRanges.aspx.

RadiantQ jQuery Gantt Package

Time Span Highlighting

Time Span Highlighting

The gantt provides the option to highlight the time line in the chart area when the user moves the mouse over the time line headers. This feature is easily turned on with this setting:

Here is a sample code that shows how to setup the time range highlight behavior

In HTML

```
/ Initialize the FlexyGantt widget.
$('#container').FlexyGantt({

    DataSource: self.jsonData,
    //to remove this behavior,you have to set
    "RadiantQ.Gantt.TimeRangeHighlightBehavior.None"

    TimeRangeHighlightBehavior:RadiantQ.Gantt.TimeRangeHighlightBehavior.HighlightInChart
    OnHeaderMouseHover,
    .....
});
```

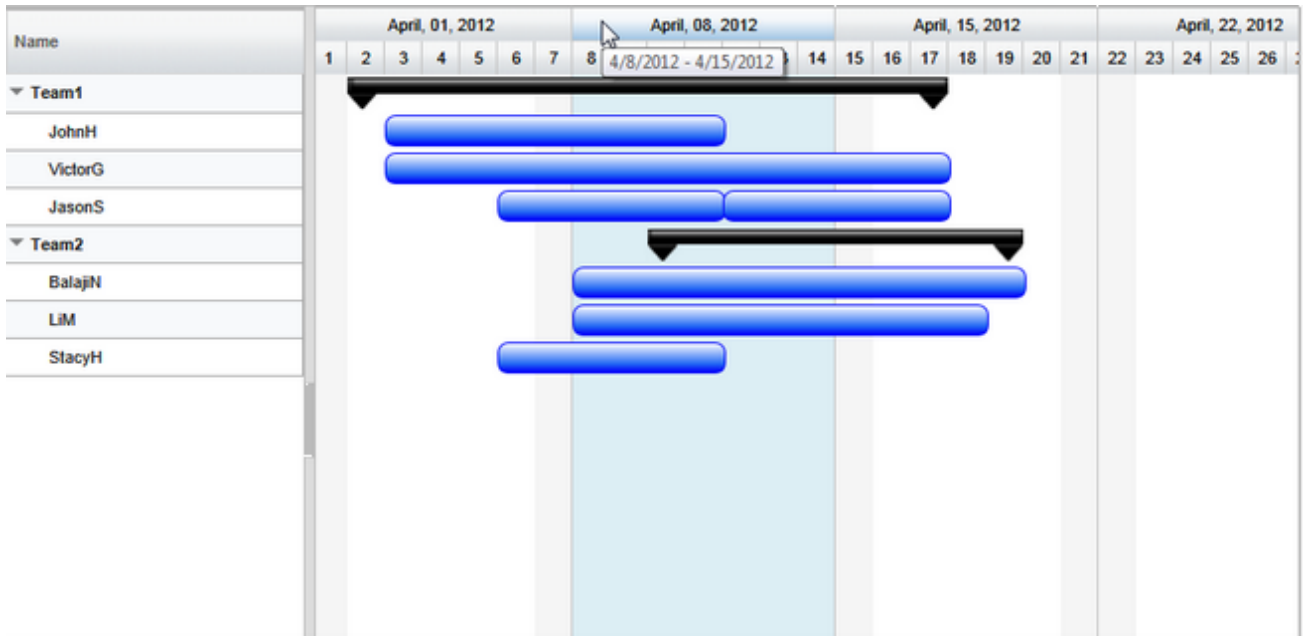
In ASP.NET MVC

```
@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        ControlId = "gantt_container",

        Options = new FlexyGanttOptions()
        {
            ...
            TimeRangeHighlightBehavior =
TimeRangeHighlightBehavior.HighlightInChartOnHeaderMouseHover,
            GanttChartOptions = new GanttChartOptions()
            {
                AnchorTime = DateTime.Today,
                MovingInfoPopupID = "MovingInfoPopup",
                ResizeInfoPopupID = "ResizeInfoPopup",
            }
        }
    }
)
```

In ASP.NET

```
<RQ:FlexyGantt ID="gantt" TimeRangeHighlightBehavior =
"HighlightInChartOnHeaderMouseHover" ... />
```



Customizing the default non working background and highlight color

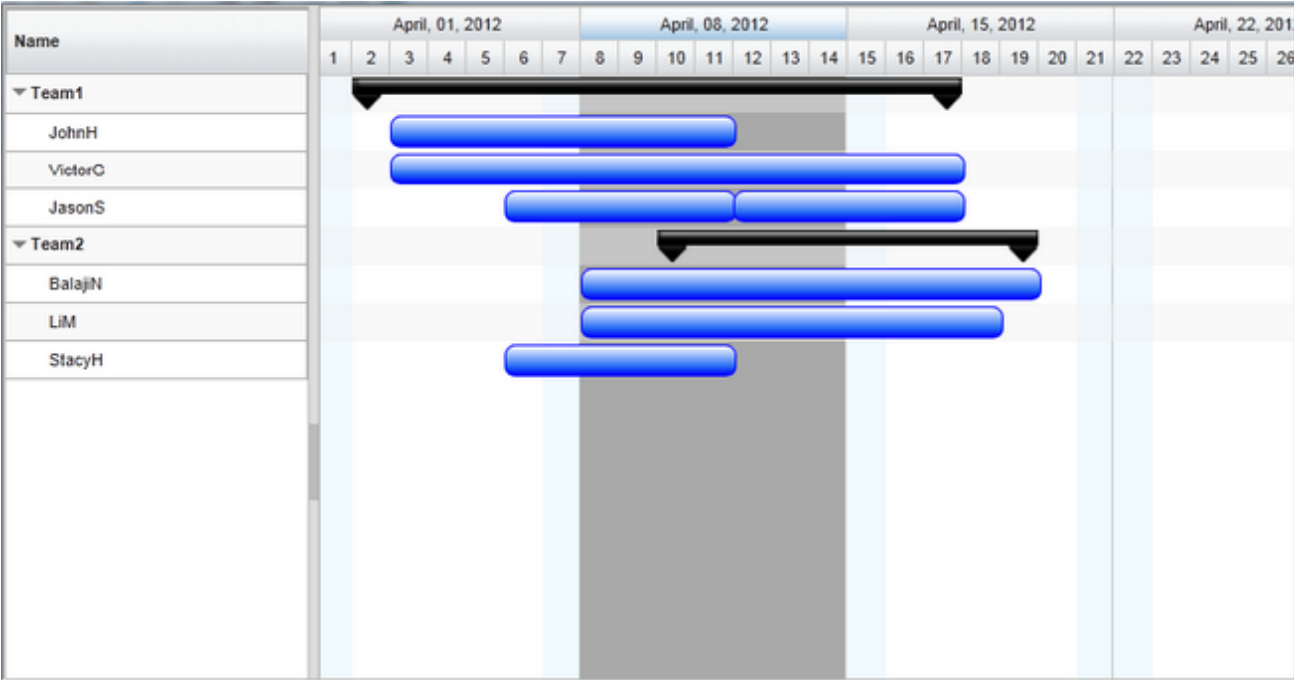
This is easily implemented by redefining the CSS styles defined in the `radiantq.gantt.default.css` file.

To change NonWorking day background in chart.

```
.rq-gc-nonWorking-background
{
    background-color: #F0FAFF !important;
    border: none !important;
    background-image: none !important;
}
```

To change time line hover style in chart.

```
.timeunitBG-mouseIn
{
    background: darkgray !important;
    border: none !important;
}
```



Gantt's non-working background and hover background redefined.

RadiantQ jQuery Gantt Package ContextMenus

The Gantt Controls employs several built-in context menus for different portions of the gantt. It also provides a couple of options as to the context menu framework you want to use.

a) ContextMenus with jQuery UI 1.11.4 and later

Beginning jQuery UI 1.11.4, there is support for Context Menu which the Gantt library utilizes for its built-in context menus. So, if you are linking to this version or later of the jQuery UI (see below), then there is nothing else to refer to for Context Menu support. The gantt will automatically start showing context menus based on menus in jQuery UI.

```
<script src="Src/Scripts/jquery-1.11.2.min.js" type="text/javascript"></script>
<script type="text/javascript" src
="Src/Scripts/jquery-ui-1.11.4/jquery-ui.min.js" ></script>
```

b) ContextMenus with earlier jQuery UI versions

Since these jQuery UI versions do not have any built-in context menu support, the gantt expects you to include the "jquery.contextMenu.js" plugin and "jquery.contextMenu.css" to provide context menu support. You simply have to add a reference to this plugin script file in your html page:

```
<link href="Src/Styles/jquery.contextMenu.css" rel="stylesheet" type="text/css" />
<script src="Src/Scripts/jquery.contextMenu.js" type="text/javascript"></script>
```

... and the gantt will automatically start showing context menus built from this plugin.

Support for Other ContextMenu frameworks

If there is a context menu plugin that you would like for the gantt to use, an extension needs to be built for that.

We will continue to add support for popular context menu plugins in future.

Customizing the context menu items is discussed within the GanttControl and FlexyGantt topics.

RadiantQ jQuery Gantt Package

Localized Strings

Localizing Strings used by the RadiantQ Gantt widget

All the strings used in the RadiantQ Gantt control are enabled for localization. We have localized gantt for several languages that are distributed as part of the install at this location:
Src\ResourceStrings

In HTML

```
//To load the RadiantQ culture JS and date JS based on the specified cultureName
RadiantQ.Culture.Load(cultureName);

//Or you have to manually include the script files in head tag.

<script src="Src/ResourceStrings/en-US.js"></script>
<script src="Src/Scripts/Utils/globalization/en-US.js"></script>
```

In ASP.NET MVC

```
@*scripts loads based on the page culture e*@

@*date js globalization file*@
<script src="~/Src/Scripts/Utils/globalization/@string.Format("{0}.js",
System.Globalization.CultureInfo.CurrentCulture.ToString())"></script>
@*jQuery UI for globalization file for datetime picker. *@
<script src="~/Scripts/jquery.ui.datepicker-@string.Format("{0}.js",
System.Globalization.CultureInfo.CurrentCulture.TwoLetterISOLanguageName)"></script>

@*RadiantQ globalization file *@
<script src="~/Src/ResourceStrings/@string.Format("{0}.js", System.Globalization.
CultureInfo.CurrentCulture.ToString())"></script>
```

In ASP.NET

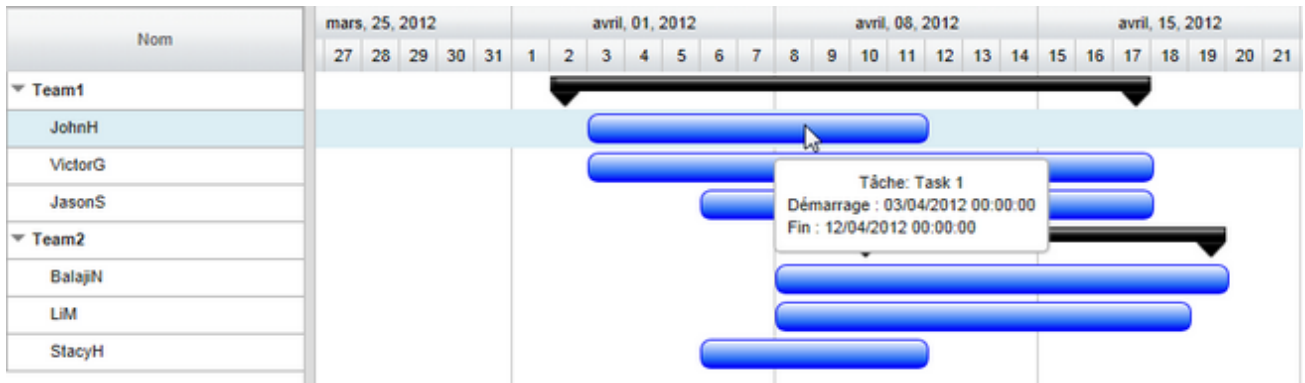
```
@*scripts loads based on the page SelectedValue e*@

<script src="<%= Page.ResolveClientUrl("~/Src/Scripts/Utils/globalization/"+this
.lang.SelectedValue + ".js") %>" ></script>
<script src="<%= Page.ResolveClientUrl("~/Src/ResourceStrings/"+this
.lang.SelectedValue + ".js") %>" ></script>
<script src="<%= Page.ResolveClientUrl("../Scripts/jquery.ui.datepicker-"+this
.lang.SelectedValue.Split('-')[0] + ".js") %>" ></script>
```

By default the Gantt widgets comes with localized string files for the fr ,de ,es ,ru, ja, nb, zh,

us, no cultures.

Here is the resultant



This is illustrated in this samples:

In HTML : ..\Samples\LocalizedStrings.htm.
 In ASP.NET MVC : ..\Views\Home\Common\LocalizedStrings.cshtml.
 In ASP.NET : ..\Samples\Common\LocalizedStrings.aspx.

Localized Date strings in the Gantt

The [date.js](#) that the gantt uses is an open source library that supports 150+ cultures. To know more about this take a look at [this](#) .

The Localized Date strings in the Gantt widgets are based on the date js file included in the web page. By default all the samples reference the "Date.JS" file, this file is for "en-US" culture.

To support a different culture, include that culture's date js file in your page.

Static file inclusion.

```
<!-- Set the CultureInfo to fr-FR (French) -->
<script src="Src/Scripts/Utils/globalization/fr-FR.js" type="text/javascript"></script>
```

Note: Make sure the Date script is included before the Gantt widget is created.

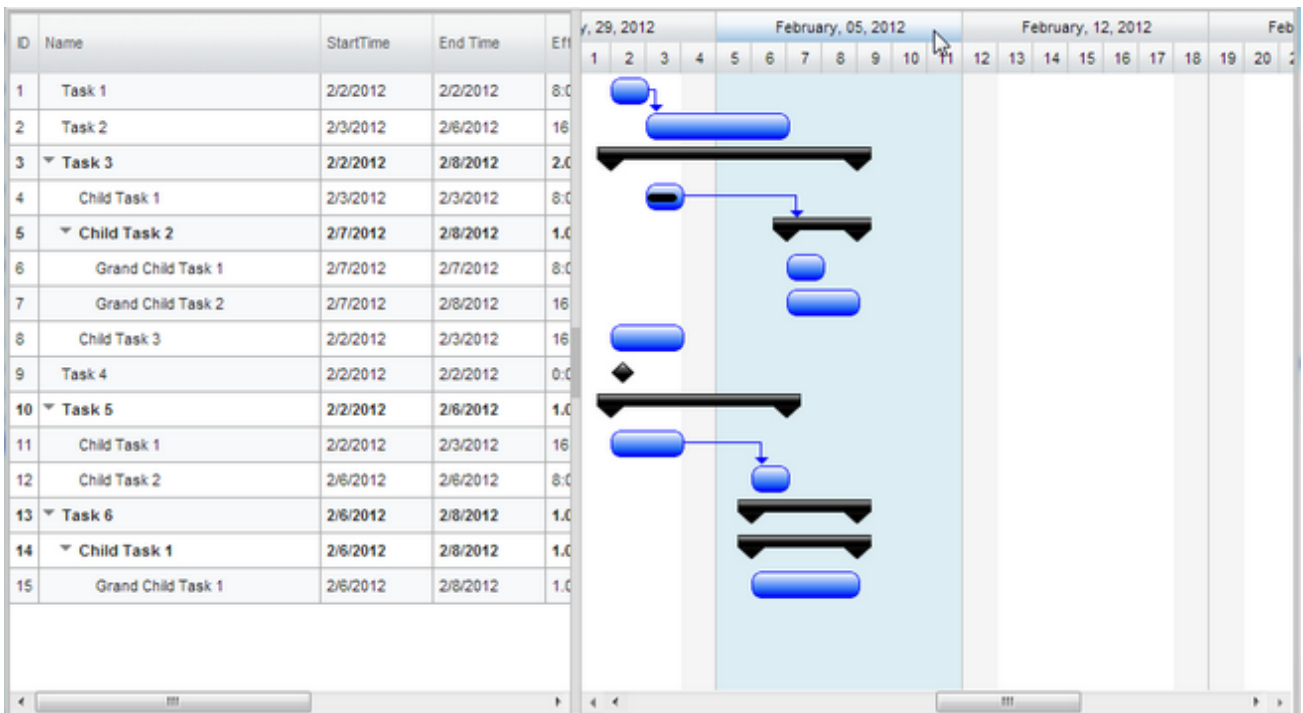
RadiantQ jQuery Gantt Package Themes

The jQuery Gantt widgets come with a professional looking default look which you can customize to your needs. But the gantt widgets can also take on the look of a jQuery UI theme if you choose to include incorporate that theme in your page.

Default Theme

The default style for the Gantt is defined in the file radiantq.gantt.default.css.

```
<!-- Default theme-->
<link href="Src/Styles/radiantq.gantt.default.css" rel="stylesheet" type="text/css" />
```



jQuery Gantt Default Style.

The above CSS file is in the ..\Src\Styles folder.

Customizing the default gantt look

This is easily implemented by redefining the CSS styles defined in the radiantq.gantt.default.css file.

To change Task Bar style and the Dependency Line Style in chart.

```
.taskbar-style
{
    height: 20px;
    background-image: url(Images/greenBar.png) !important ;
    -moz-background-size: 100% 100% !important; /* Firefox 3.6 */
    background-size: 100% 100% !important;
    background-repeat: repeat !important;
    border: 1px solid green !important;
    border-radius: 7px !important;
    float: left !important;
}
```

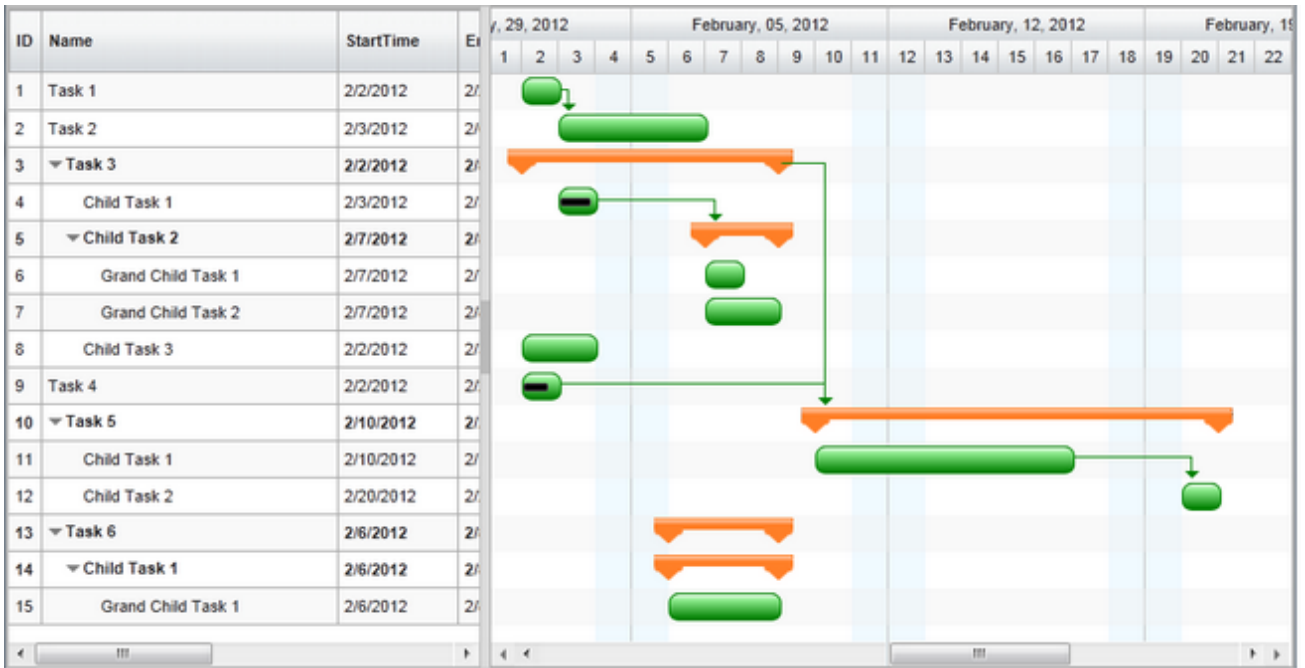
```
    position: absolute !important;
    z-index: 10 !important;
}
.rq-gc-dependencyLine-arrow
{
    border-top-color: green !important;
    background: transparent !important;
    border-bottom-color: green !important;
    border-color: green !important;
}
```

To change Summary Bar style in chart.

```
.parentLeftPoly-style
{
    background-image: url(Images/Darkness_poly.png);
    width: 20px;
    position: absolute;
    height: 100%;
    background-size: 100% 100%;
    background-repeat: no-repeat;
    background-position: center;
    float: left;
}

.rq-gc-parentBar-middle
{
    background-image: url(Images/Darkness_parentmiddle.png);
    width: 100%;
    position: absolute;
    height: 50%;
    background-repeat: no-repeat;
    background-size: 100% 100%;
    float: left;
}

.rq-gc-parentBar-rightCue
{
    background-image: url(Images/Darkness_poly.png);
    width: 20px;
    height: 100%;
    background-size: 100% 100%;
    background-repeat: no-repeat;
    background-position: center;
    float: right;
}
```



Customized jQuery Gantt default Style.

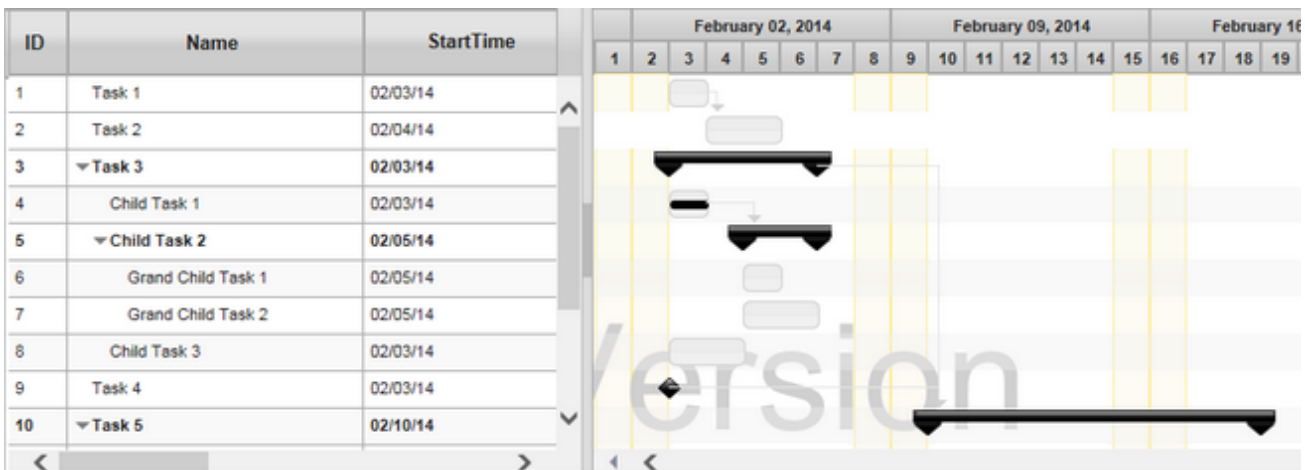
Including jQuery UI Theme

For the jQuery UI theme css to affect the gantt, you must include the radiantq.gantt.theme.css (instead of the above *default.css) followed by the ui theme css file, specifically in that order.

```

<!-- Gantt overridable style-->
<link href="Src/Styles/radiantq.gantt.theme.css" rel="stylesheet" type="text/css" />
<!--jQuery ui le-frog theme-->
<link href="Src/themes/Themes/smoothness/jquery-ui.css" rel="stylesheet" type="text/css" />
    
```

Note: If, by mistake, you include radiantq.gantt.default.css (instead of the *gantt.theme.css above), then the jQuery UI themes will not have any effect on the gantt.



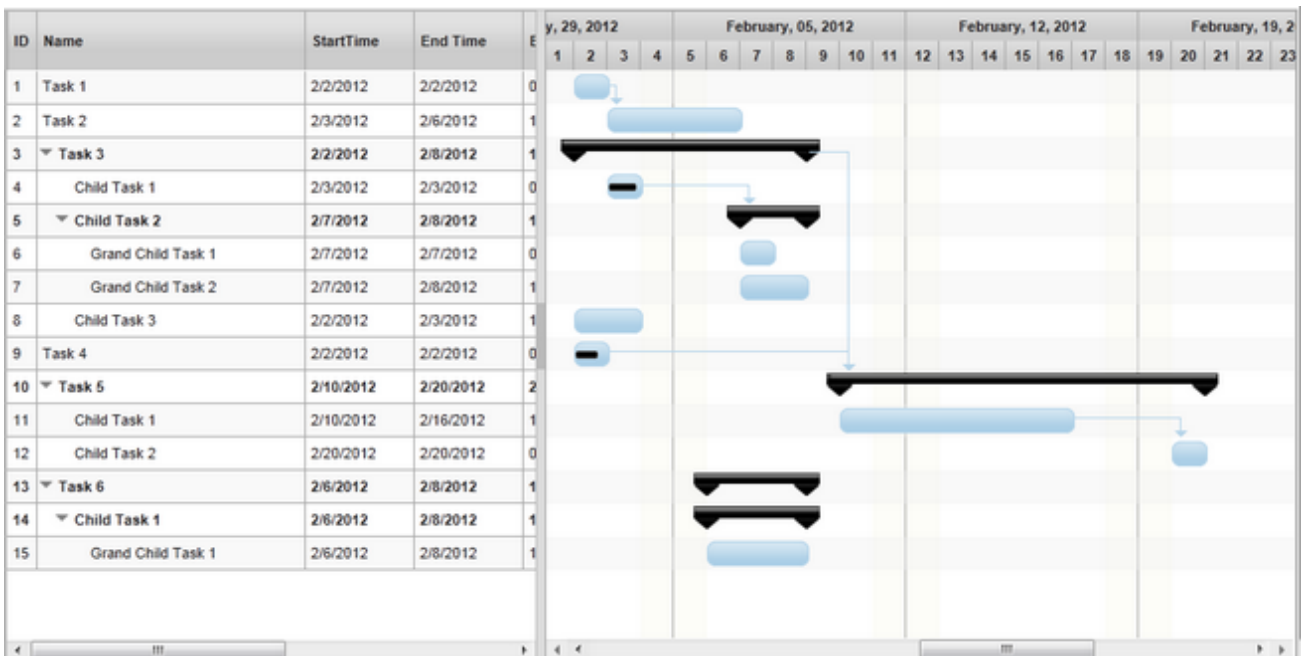
Gantt using jQuery UI smoothness theme.

All themes are illustrated in this sample in our install: ..\Samples\ThemesSupport.htm sample.

Customizing the jQuery UI themed look

To change Task Bar Style and Dependency Line Style in the chart.

```
.ui-state-default{
    border: 1px solid rgb(182,213,234) !important;
    background: rgb(182,213,234) url(Images/Cupertino_taskbar.png) 50% 50% repeat-x
!important;
    font-weight: normal ;
    color: #ffffff ;
}
.rq-gc-dependencyLine
{
    background-clip: padding-box !important;
    background-color:rgb(182,213,234);
    border-width:0px !important;
    margin-top:0px !important;
    background-image:none !important;
}
.rq-gc-dependencyLine-arrow
{
    border-color:rgb(182,213,234);
    border-top-width:5px !important;
    background:transparent !important;
}
}
```



Customized *jQuery UI smoothness theme*.

-0-

RadiantQ jQuery Gantt Package

Creating Gradient Backgrounds dynamically in code

Creating Gradient Backgrounds dynamically in code

To be able to specify a gradient background for a taskbar, for example, you would typically have to create an image with some gradient colors and then specify that as background for a taskbar div style. This approach is used for example in our sample:

"..\Samples\GanttControlCriticalPath.htm"

However, this is very cumbersome as you will have to use other image editing tools to create this. So, we have created a much easier to use GetGradientImageUrl API, that creates an appropriate gradient image in memory that you can set as the background of any taskbar (any HTML element).

Note: This feature requires you to reference the "canvg.js" JS file in your web page. This file is available at ..\Samples\Scripts\canvg.js in our installation.

Usage Scenario 1 (setting gradient for bar templates):

This code sample shows how to use this API when you are defining the custom look and feel for a FlexyGantt task bar in JS. (Same approach can be used when defining the look and feel of a GanttControl task bar.)

```
// setGradientAsBackground({x1:value,y1:value,x2:value,y2:value}, [stops])
// x1 and y1 (starting point) define the direction of the gradient
// x2 and y2 (end point) define the direction of the gradient.
var greenBarImageUrl = RadiantQ.Gantt.GetGradientImageUrl(
    { x1: 0, y1: 0, x2: 0, y2: 100 },
    [{ offset: '0', 'stop-color': '#FFFFFF' }, { offset: '0.25', 'stop-color':
'lightgreen' }, { offset: '1', 'stop-color': 'green' }]
);

// Then reference this url in the template for the task bars.
var tTemplate = '<div class="taskbar-style" style="background-image:url(' +
greenBarImageUrl + ') !important;border-color:green !important"></div>';
```

This is illustrated in the sample "..\Samples\FlexyGanttCustomTaskLook.htm" sample.

Usage Scenario 2 (setting the gradient for any html element from JS):

In this sample code below, calling the setGradientAsBackground API for the list of jquery elements specified by their style name would set the background property on the specified style to this internally generated image.

```
// setGradientAsBackground({x1:value,y1:value,x2:value,y2:value}, [stops])
// x1 and y1 (starting point) define the direction of the gradient
// x2 and y2 (end point) define the direction of the gradient.
$(".taskbar-style").setGradientAsBackground(
    { x1: 0, y1: 0, x2: 0, y2: 100 },
    [{ offset: '0', 'stop-color': '#FFFFFF' }, { offset: '1', 'stop-color': '#3796D2'
}]
);
```

Supported Gradient Types

1) Linear Gradient:

LinearGradient should be in the format of {x1, y1, x2, y2} coordinates and stops.

Horizontal gradients are created when y1 and y2 are equal and x1 and x2 differ.

For LeftToRight gradient: {x1:0, y1:0, x2:100, y2:0}

For RightToLeft gradient: {x1:100, y1:100, x2:0, y2:100}

Vertical gradients are created when x1 and x2 are equal and y1 and y2 differ.

For TopToBottom gradient: {x1:0, y1:0, x2:0, y2:100}

For BottomToTop gradient: {x1:100, y1:100, x2:100, y2:0}

Angular gradients are created when x1 and x2 differ and y1 and y2 differ

For top-left to bottom-right gradient: {x1:0, y1:0, x2:100, y2:100}

For top-right to bottom-left gradient: [x1:100,y1:0,x2:0,y2:100]

For bottom-left to top-right gradient: {x1:0, y1:100, x2:100, y2:0}

For bottom-right to top-left gradient: {x1:100, y1:100, x2:0, y2:0}

2) Creating the Radial Gradient:

RadialGradient should be in the format of {cx, cy, r} - coordinates and stops.

The cx and cy of RadialGradient define the width and height of the shape to fill
r defines the radius of the gradient

For top-left gradient: {cx:0, cy:0, r:50}

For top-center gradient: {cx:50, cy:0, r:50}

For top-right gradient: {cx:100, cy:0, r:50}

For middle-left gradient: {cx:0, cy:50, r:50}

For middle-center gradient: {cx:50, cy:50, r:50}

For middle-right gradient: {cx:100, cy:50, r:50}

For bottom-left gradient: {cx:0, cy:100, r:50}

For bottom-center gradient: {cx:50, cy:100, r:50}

For bottom-right gradient: {cx:100, cy:100, r:50}

NOTE : Currently we are not supporting Gradient Backgrounds dynamically in ie8.

© RadiantQ 2009-2018. All Rights Reserved.

Chart Look and Feel

RadiantQ jQuery Gantt Package Time Scale Header Customization

NOTE: All references to "gantt" in this topic refers to both [GanttControl](#) and [FlexyGantt](#) widgets.

-0-

RadiantQ jQuery Gantt Package

Default Time Scale Header.

The Time Scale Headers refer to the headers in the gantt chart that define how the horizontal span of the gantt chart should be divided into days, weeks, etc.

Default Time Scale Header.

As seen above the default headers consist of a header row of type "Days" and another of type "Weeks". This is the default definition in the Gantt widget's TimeScaleHeaders option. If the headers are not specified in Gantt it automatically renders the default header.

April, 15, 2012							April, 22, 2012							April, 29, 2012							May, 06, 2012						
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12

To add/remove header rows, you will have to add/remove TimeScaleHeaderDefinition instances to the above collection property.

The code below shows how to add a headers in GanttChart.

In HTML

```

var self = this;
$(document).ready(function(){

    var tmshs = new RadiantQ.Gantt.TimeScaleHeaderDefinitions();
    tmshs.HeaderCollection.add(self.monthHeaderLine());
    tmshs.HeaderCollection.add(self.weekHeaderLine());
    tmshs.HeaderCollection.add(self.dayHeaderLine());

    $('#container').FlexyGantt({
        TimeScaleHeaders : tmshs,
        .....
    });
});

function weekHeaderLine(){
    var weeksHeader = new RadiantQ.Gantt.TimeScaleHeaderDefinition();
    weeksHeader.Type = RadiantQ.Gantt.TimeScaleType.Weeks;
    return weeksHeader;
}
function monthHeaderLine(){
    var monthsHeader = new RadiantQ.Gantt.TimeScaleHeaderDefinition();
    monthsHeader.Type = RadiantQ.Gantt.TimeScaleType.Months;
    return monthsHeader;
}
function dayHeaderLine(){
    var daysHeader = new RadiantQ.Gantt.TimeScaleHeaderDefinition();
    daysHeader.TextFormat = "ddd";
    daysHeader.Type = RadiantQ.Gantt.TimeScaleType.Days;
    return daysHeader;
}

```

In ASP.NET MVC

```

@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        ControlId = "ganttt_container",
        AfterDataRetrievedCallback = "AfterDataRetrievedCallback",
        AfterGanttWidgetInitializedCallback =
            "AfterGanttWidgetInitializedCallback",
        DataSourceUrl = new Uri("/Home/GetFlexyGanttItemSource", UriKind
            .RelativeOrAbsolute),
        Options = new FlexyGanttOptions()
        {
            TimeScaleHeaders = new TimeScaleHeaderDefinitions(){
                new TimeScaleHeaderDefinition(){
                    Type = TimeScaleType.Months

                },
                new TimeScaleHeaderDefinition(){
                    Type = TimeScaleType.Weeks
                },
                new TimeScaleHeaderDefinition(){
                    Type = TimeScaleType.Days
                }
            }
        }
    }
)

```

In ASP.NET

```

<RQ:GanttControl ID="ganttt" DataSourceUrl="../../DataSources/TaskListHandler.ashx"
    AfterGanttWidgetInitializedCallback
    = "AfterGanttWidgetInitializedCallback"
    TimeRangeHighlightBehavior="HighlightInChartOnHeaderMouseHover"
    Height="500px" runat="server" >
    <TimeScaleHeaders>
        <GanttBase:TimeScaleHeaderDefinition Type="Years" Name
    = "yearHeader" />
        <GanttBase:TimeScaleHeaderDefinition Type="Months" Name
    = "monthHeader" TextFormat="MMM yyyy" />
        <GanttBase:TimeScaleHeaderDefinition Type="Days" Name
    = "daysHeader" />
    </TimeScaleHeaders>

</RQ:GanttControl>

```

December 2012														January 2013													
December, 23, 2012							December, 30, 2012							January, 06, 2013							January, 13, 2013						
Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu		

Monthly, Weekly and Daily Headers

RadiantQ jQuery Gantt Package

Header Text Format and Height Customization

Header Height

You can specify a custom height for each of the headers in the GanttChart using the `TimeScaleHeaderDefinition.HeaderHeight` property.

For example, to change the height of the top-most header:

In HTML

```
var self = this;
$(document).ready(function(){

    var tmshs = new RadiantQ.Gantt.TimeScaleHeaderDefinitions();
    tmshs.HeaderCollection.add(self.monthHeaderLine());
    tmshs.HeaderCollection.add(self.weekHeaderLine());
    tmshs.HeaderCollection.add(self.dayHeaderLine());

    $('#container').FlexyGantt({
        TimeScaleHeaders : tmshs,
        .....
    });
});

function weekHeaderLine(){
    var weeksHeader = new RadiantQ.Gantt.TimeScaleHeaderDefinition();
    weeksHeader.Type = RadiantQ.Gantt.TimeScaleType.Weeks;
    return weeksHeader;
}
function monthHeaderLine(){
    var monthsHeader = new RadiantQ.Gantt.TimeScaleHeaderDefinition();
    monthsHeader.Type = RadiantQ.Gantt.TimeScaleType.Months;
    monthsHeader.HeaderHeight=25;
    return monthsHeader;
}
function dayHeaderLine(){
    var daysHeader = new RadiantQ.Gantt.TimeScaleHeaderDefinition();
    daysHeader.TextFormat = "ddd";
    daysHeader.Type = RadiantQ.Gantt.TimeScaleType.Days;
    return daysHeader;
}
```

In ASP.NET MVC

```

@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        ControlId = "gantt_container",
        AfterDataRetrievedCallback = "AfterDataRetrievedCallback",
        AfterGanttWidgetInitializedCallback =
            "AfterGanttWidgetInitializedCallback",
        DataSourceUrl = new Uri("/Home/GetFlexyGanttItemSource", UriKind
            .RelativeOrAbsolute),
        Options = new FlexyGanttOptions()
        {
            TimeScaleHeaders = new TimeScaleHeaderDefinitions(){
                new TimeScaleHeaderDefinition(){
                    Type = TimeScaleType.Months

                },
                new TimeScaleHeaderDefinition(){
                    Type = TimeScaleType.Weeks
                },
                new TimeScaleHeaderDefinition(){
                    Type = TimeScaleType.Days,HeaderHeight=22
                }
            }
        }
    })

```

In ASP.NET

```

<RQ:GanttControl ID="gantt" DataSourceUrl="../../../DataSources/TaskListHandler.ashx"
    AfterGanttWidgetInitializedCallback
    = "AfterGanttWidgetInitializedCallback"
    TimeRangeHighlightBehavior="HighlightInChartOnHeaderMouseHover"
    LocalizationResourceFilePath="../../../Src/ResourceStrings/"
    Height="500px" runat="server" >
    <TimeScaleHeaders>
        <GanttBase:TimeScaleHeaderDefinition Type="Years" Name
    = "yearHeader" />
        <GanttBase:TimeScaleHeaderDefinition Type="Months" Name
    = "monthHeader" TextFormat="MMM yyyy" />
        <GanttBase:TimeScaleHeaderDefinition Type="Days" Name
    = "daysHeader" HeaderHeight="22" />
    </TimeScaleHeaders>

    </RQ:GanttControl>

```

Changing the chart header's height will automatically change the height of the grid's header.

Header Text Format

a) The `TimeScaleHeaderDefinition` type lets you specify the format string that will be internally used in the `Date().toString` method to arrive at the text that is to be displayed in the headers above.

Set a custom `TextFormat` value to customize this. The supported format strings are discussed [here](#).

In HTML

```
function dayHeaderLine(){
    var daysHeader = new RadiantQ.Gantt.TimeScaleHeaderDefinition();
    daysHeader.TextFormat = "ddd";
    daysHeader.Type = RadiantQ.Gantt.TimeScaleType.Days;
    return daysHeader;
}
```

In ASP.NET MVC And ASP.NET

```
new TimeScaleHeaderDefinition()
{
    Type = TimeScaleType.Days,
    TextFormat="ddd"
}
```

December 2012														January 2013													
December, 23, 2012							December, 30, 2012							January, 06, 2013							January, 13, 2013						
Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu		

Custom format in day headers.

b) The gantt also provides you the ability to specify multiple text formats and intelligently displays the format that best fits the available width. For example:

```
headers.Add(new TimeScaleHeaderDefinition() { Type = TimeScaleType.Days,
TextFormat="M|dd" });
```

In HTML

```
function dayHeaderLine(){
    var daysHeader = new RadiantQ.Gantt.TimeScaleHeaderDefinition();
    daysHeader.TextFormat = "MMMM dd |dd";
    daysHeader.Type = RadiantQ.Gantt.TimeScaleType.Days;
    return daysHeader;
}
```

In ASP.NET MVC And ASP.NET

```
new TimeScaleHeaderDefinition()
{
    Type = TimeScaleType.Days,
    TextFormat="MMMM dd |dd"
}
```

... will cause the gantt to first try to render the date in "MMMM dd" format like this:

2012													
Apr 2012													
April 06	April 07	April 08	April 09	April 10	April 11								

Day headers rendered in "MMMM dd" format.

... and when the headers size is not big enough to accommodate the text, it would render in

"dd" format like this:

2012						
Apr 2012						
06	07	08	09	10	11	12

Day headers rendered in "dd" format

Also, if the default formatting options don't suffice, you can provide custom header texts as follows:

// Customize the weekly header's text:

In HTML

```
function weeksHeaderLine()
{
    var weeksHeader = new RadiantQ.Gantt.TimeScaleHeaderDefinition();
    weeksHeader.Type = ns_gantt.TimeScaleType.Weeks;
    weeksHeader.ProvideCustomHeaderText.subscribe( provideCustomHeaderText );
    function provideCustomHeaderText( sender, args )
    {
        var jan1st = new Date( args.dateTime.getFullYear(), 0, 1, 0, 0, 0, 0 );
        var days = new TimeSpan( args.dateTime - jan1st ).days;
        var weeks = days / 7;
        args.Text = "W " + weeks.toString();
        args.TooltipText = "Week of " + args.dateTime.toShortDateString();
    }
    return weeksHeader;
}
```

In ASP.NET MVC And ASP.NET

```
<script >
    function provideCustomHeaderText(sender, args) {
        var jan1st = new Date(args.dateTime.getFullYear(), 0, 1, 0, 0, 0, 0);
        var days = new TimeSpan(args.dateTime - jan1st).days;
        var weeks = days / 7;
        args.Text = "W " + weeks.toString();
        args.TooltipText = "Week of " + args.dateTime.toShortDateString();
    }
</script>

new TimeScaleHeaderDefinition()
{
    Type = TimeScaleType.Weeks,
    ProvideCustomHeaderText="provideCustomHeaderText"
}
```

W 14				W 15				W 16				W 17															
08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	01	02	03	04	05

Weeks in the format "W XX"

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package Zooming Programmatically

Zooming - Programmatically

You can specify a zoom level programmatically by changing the `$(selector).Gantt ('option', 'BaseTimeUnitWidth')` property as follows:

```
// For zooming out 50%
```

```
$( selector).FlexyGantt( 'option', 'BaseTimeUnitWidth', 37.5 ) // Default is 25.0; Max is 50.
```

The above value changes as the end-user zooms in and out the header during runtime.

This is called "base time unit width", because it indicates the width (in pixels) for the current "base time scale type" header which is indicated by the `BaseTimeScaleType` property. The `BaseTimeScaleType` property does not usually change during the life of the gantt control, even when there are headers being added and removed or shown/hidden. It's value is usually `RadiantQ.Gantt.TimeScaleType.Days` unless you have a `RadiantQ.Gantt.TimeScaleType.Hours` header in the time scale. So, a value of 20 for `BaseTimeUnitWidth` usually means - draw the base time scale type time units (usually Days) 20 pixels wide. A time unit representing a week would be drawn 140 pixels wide and so on.

There are also (settable) `BaseTimeUnitWidthMinimum` and `BaseTimeUnitWidthMaximum` which dictate the allowed range of zooming operations. This specified range will be enforced while setting the zoom value programmatically or by the end-user.

RadiantQ jQuery Gantt Package Custom Time Scale Headers

Custom Time Scale Headers

If the built-in headers ranging from Minutes to Years are not sufficient, you can create custom headers with custom ranges as follows..

// Custom Semester (Half-Yearly) header.

```
In HTML
```

```

var self = this;
$(document).ready(function(){

    var tmshs = new RadiantQ.Gantt.TimeScaleHeaderDefinitions();
    //Step 1: Define the custom header and add it to the headers collection
    ....
    tmshs.HeaderCollection.add(self.semiAnnualHeadercustomHeaderLine());

    $('#container').FlexyGantt({
        TimeScaleHeaders : tmshs,
        .....
    });
});

function semiAnnualHeadercustomHeaderLine()
{
    var semiAnnualHeader = new ns_gantt.TimeScaleHeaderDefinition();
    semiAnnualHeader.name( "semiAnnualHeader" );
    semiAnnualHeader.Type = ns_gantt.TimeScaleType.Custom;
    // See how it's important to specify an approximate "hint" for the gantt to be
    // able render this properly.
    semiAnnualHeader.CustomTimeScaleTypeHint = new
    RadiantQ.Gantt.CustomTimeScaleTypeHint( 6, ns_gantt.TimeScaleType.Months );
    semiAnnualHeader.ProvideCustomHeaderText.subscribe( provideCustomHeaderText );
    semiAnnualHeader.ProvideCustomTimeIntervals.subscribe(
    ProvideCustomTimeIntervals );

    // Step 2: Provide the custom intervals for this header for a given time span
    function ProvideCustomTimeIntervals( sender, args )
    {
        var start = args.ViewStartTime.clone();
        if(args.ViewStartTime.getMonth() < 6)
            start = new Date(start.getFullYear(), 6, 1);
        else
            start = new Date(start.getFullYear() + 1, 0, 1);

        for(var dateTime = start.clone(); args.ViewEndTime.compareTo(dateTime) >= 0;
        dateTime = dateTime.clone().addMonths(6)){
            var t = dateTime.clone().addMonths(6);
            args.TimeIntervals.push(dateTime.clone());
        }
    }
    // Step 3: Provide the header text for this custom headers
    function provideCustomHeaderText(sender, args){
        args.TooltipText = args.dateTime.getFullYear().toString();
        var half = ((args.dateTime.getMonth()) / 6) + 1;
        args.Text = "H" + Math.floor(half); // Displays H1 or H2
    }

    return semiAnnualHeader;
}

```

In ASP.NET MVC

```

<script type="text/javascript">
    // Step 1: Provide the custom intervals for this header for a given time span
    function ProvideCustomTimeIntervals( sender, args )
    {
        var start = args.ViewStartTime.clone();
        if(args.ViewStartTime.getMonth() < 6)
            start = new Date(start.getFullYear(), 6, 1);
        else
            start = new Date(start.getFullYear() + 1, 0, 1);

        for(var dateTime = start.clone(); args.ViewEndTime.compareTo(dateTime) >= 0;
dateTime =      dateTime.clone().addMonths(6)){
            var t = dateTime.clone().addMonths(6);
            args.TimeIntervals.push(dateTime.clone());
        }
    }
    // Step 2: Provide the header text for this custom headers
    function provideCustomHeaderText(sender, args){
        args.TooltipText = args.dateTime.getFullYear().toString();
        var half = ((args.dateTime.getMonth()) / 6) + 1;
        args.Text = "H" + Math.floor(half); // Displays H1 or H2
    }

</script>

@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        ControlId = "gantt_container",
        DataSourceUrl = new Uri("/Home/GetFlexyGanttItemSource", UriKind
.RelativeOrAbsolute),
        Options = new FlexyGanttOptions()
        {
            TimeScaleHeaders = new TimeScaleHeaderDefinitions(){
                .....

                new TimeScaleHeaderDefinition(){
                    Type = TimeScaleType.Custom,
                    CustomTimeScaleTypeHint = new CustomTimeScaleTypeHint(6,
Months),

                    ProvideCustomTimeIntervals="ProvideCustomTimeIntervals",
                    ProvideCustomHeaderText="provideCustomHeaderText"
                }
            },
            ...
        }
    }
})

```

In ASP.NET

```

<script type="text/javascript">
    // Step 1: Provide the custom intervals for this header for a given time span
    function ProvideCustomTimeIntervals( sender, args )
    {
        var start = args.ViewStartTime.clone();
        if(args.ViewStartTime.getMonth() < 6)
            start = new Date(start.getFullYear(), 6, 1);
        else
            start = new Date(start.getFullYear() + 1, 0, 1);

        for(var dateTime = start.clone(); args.ViewEndTime.compareTo(dateTime) >= 0;
dateTime =    dateTime.clone().addMonths(6)){
            var t = dateTime.clone().addMonths(6);
            args.TimeIntervals.push(dateTime.clone());
        }
    }
    // Step 2: Provide the header text for this custom headers
    function provideCustomHeaderText(sender, args){
        args.TooltipText = args.dateTime.getFullYear().toString();
        var half = ((args.dateTime.getMonth()) / 6) + 1;
        args.Text = "H" + Math.floor(half); // Displays H1 or H2
    }

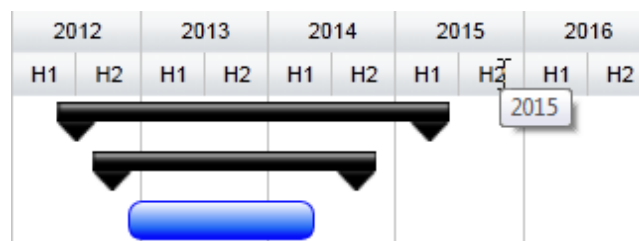
</script>

<RQ:GanttControl ID="gantt" DataSourceUrl="../../../DataSources/TaskListHandler.ashx"
    AfterGanttWidgetInitializedCallback="AfterGanttWidgetInitializedCallback"

    TimeRangeHighlightBehavior="HighlightInChartOnHeaderMouseHover"
    LocalizationResourceFilePath="../../../Src/ResourceStrings/"
    Height="500px" runat="server" >
    <TimeScaleHeaders>
        <GanttBase:TimeScaleHeaderDefinition Type="Years" Name="yearHeader" />
        <GanttBase:TimeScaleHeaderDefinition Type="Custom" Name="customHeader"
ProvideCustomTimeIntervals="ProvideCustomTimeIntervals" ProvideCustomHeaderText
="provideCustomHeaderText" >
            <CustomTimeScaleTypeHint EquivalentType="Months" EquivalentUnits="6" />
        </GanttBase:TimeScaleHeaderDefinition>
    </TimeScaleHeaders>
</RQ:GanttControl>

```

Here is the resultant custom header:



Custom Semester (half yearly) headers

-0-

RadiantQ jQuery Gantt Package

End-User Time Scale Header Operations

End User Zooming

End users can zoom the time-range using the mouse-wheel or mouse down + drag on the headers.

The current zoom level is specified in the BaseTimeUnitMinimum property and there are also (settable) BaseTimeUnitWidthMinimum and BaseTimeUnitWidthMaximum which dictate the allowed range of zooming operations. This specified range will be enforced while setting the zoom value programmatically or by the end-user. See the previous [topic](#) for more information on BaseTimeUnitWidthMinimum.

Note that while zooming-in, if the time-units in some headers are too small to display then that whole header is collapsed. For example:



Year, Month and Week headers before zooming in.



Month and Week headers are collapsed after zooming in.

You can actually customize the zooming behavior by setting the GanttChart.ZoomOptions with one or more of these enums:

```
RadiantQ.Gantt.ChartZoomOptions =
{
    // Turns off all zooming capabilities.
    None : 0,
    // Zooms when using the mouse when on the time scale header.
    MouseWheelZoomOnTimeScaleHeaders : 0x02,
    // Zooms when mouse wheel is used on the gantt chart.
    MouseWheelZoomOnGanttChart : 0x04,
    // Zooms when left mouse down on the time span header and dragged.
    LeftMouseButtonDownDrag : 0x08
}
```

End User Panning

End users can pan the time-range using Ctrl + Mouse Down + Drag on the headers.

When GanttChart.ResizeToFit is false (default setting) the end-user can only pan until he reaches the end of the scrollable view to the left or right. After which, the end-user will have to page using the pager buttons.



Right Pager Button .

Again, you can customize this behavior by setting the GanttBase.ScrollOptions property with one or more of these enums:

```
RadiantQ.Gantt.ChartScrollOptions=  
{  
    // Turns off all scrolling capabilities.  
    None : 0x00,  
    // Left mouse down on the time scale header and drag to scroll it.  
    LeftMouseButtonDownDrag : 0x01,  
    // Ctrl + Left mouse down on the time scale header and drag to scroll it.  
    CtrlAndLeftMouseButtonDownDrag : 0x02  
}
```

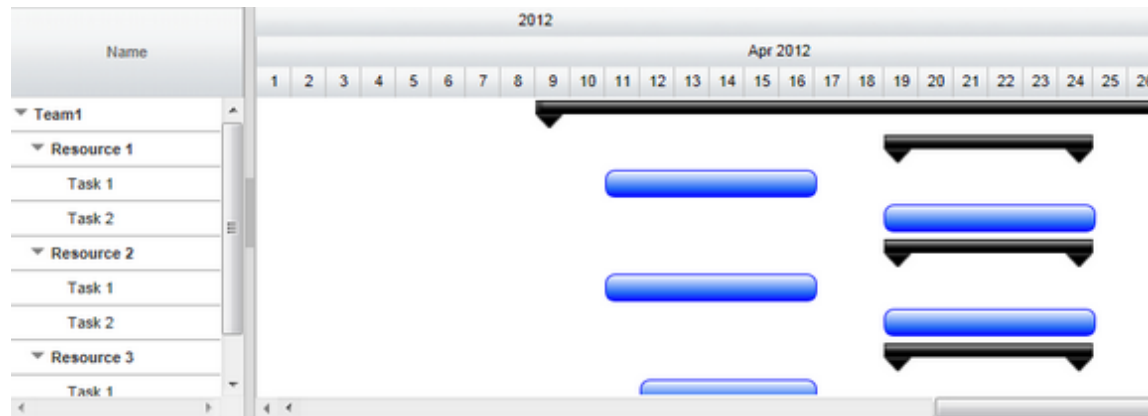
© RadiantQ 2009-2018. All Rights Reserved.

-o-

RadiantQ jQuery Gantt Package Gantt Chart AnchorTime

AnchorTime (and Start and End Times of the View)

The GanttChart also has a concept of "AnchorTime" which is the time that will be visible to the left of the chart when the chart loads. Note that this is not the time that dictates the start time of the view. In other words, AnchorTime also dictates the time at the middle of the view.



GanttChart loaded with AnchorTime set to April 2st 2012.

When ResizeToFit is set, the AnchorTime is shown right in the middle of the view.

To explicitly specify the start time of the view use the SetStartTime method.

The End-Time of the view cannot be explicitly set. It varies based on the current zoom level (BaseTimeUnitWidth setting). Take a look at the "...Samples\TaskScaleStartAndEnd" sample for utility methods that will let you do this easily.

© RadiantQ 2009-2018. All Rights Reserved.

RadiantQ jQuery Gantt Package

GanttChart View Width

GanttChart View Width

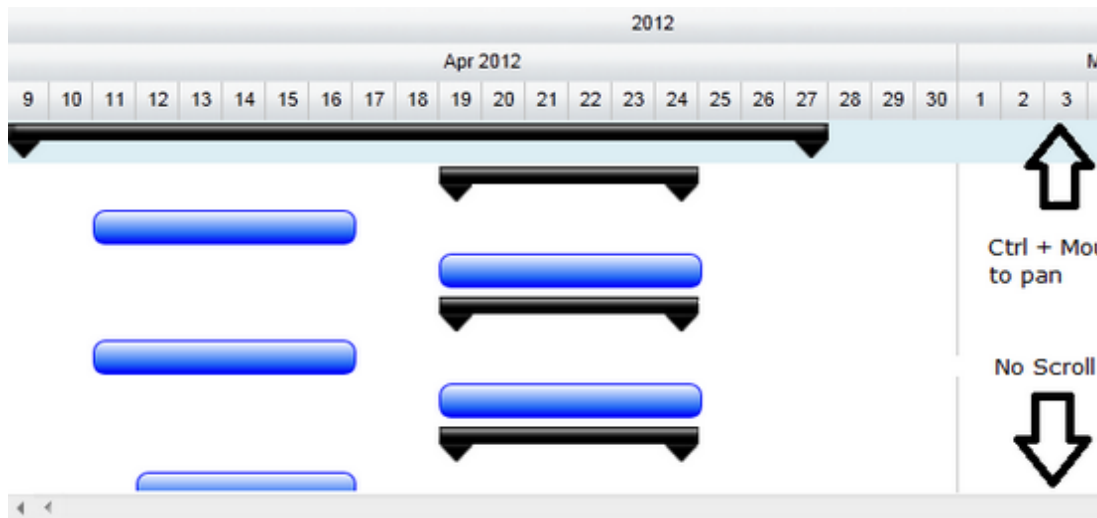
By default, the GanttChart is presented in a horizontally scrollable view whose width is specified via the ViewWidth property.

```
$(selector).FlexyGantt( 'option', 'ViewWidth' ,3000) // Default is 2000; Allowed values are 300 - 10000.
```



Alternatively, you could use the \$(selector).Gantt('option', 'ResizeToFit') property to make the view to resize to fit the available space. In this mode, the end-user can also easily pan the view through a Ctrl + Mouse Down + Drag on the chart headers.

```
$(selector).FlexyGantt( 'option', 'ResizeToFit' ,true)
```

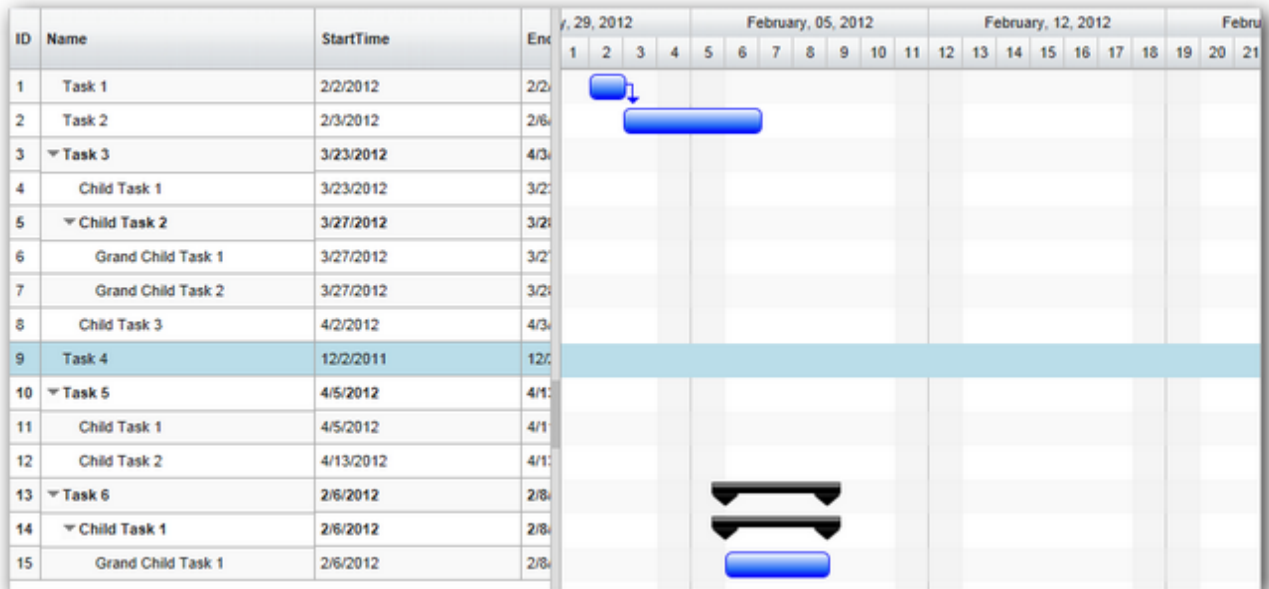


ResizeToFit mode where end-users can pan the view with Ctrl + Mouse Down + drag

*RadiantQ jQuery Gantt Package***Browse To Task Cues****Browse To Task Cues**

Many a times, at least some tasks are not in the current view as you scroll through the gantt chart's timeline. When this happens you are usually not sure whether the hidden task is to the left of the timeline or to the right of the timeline.

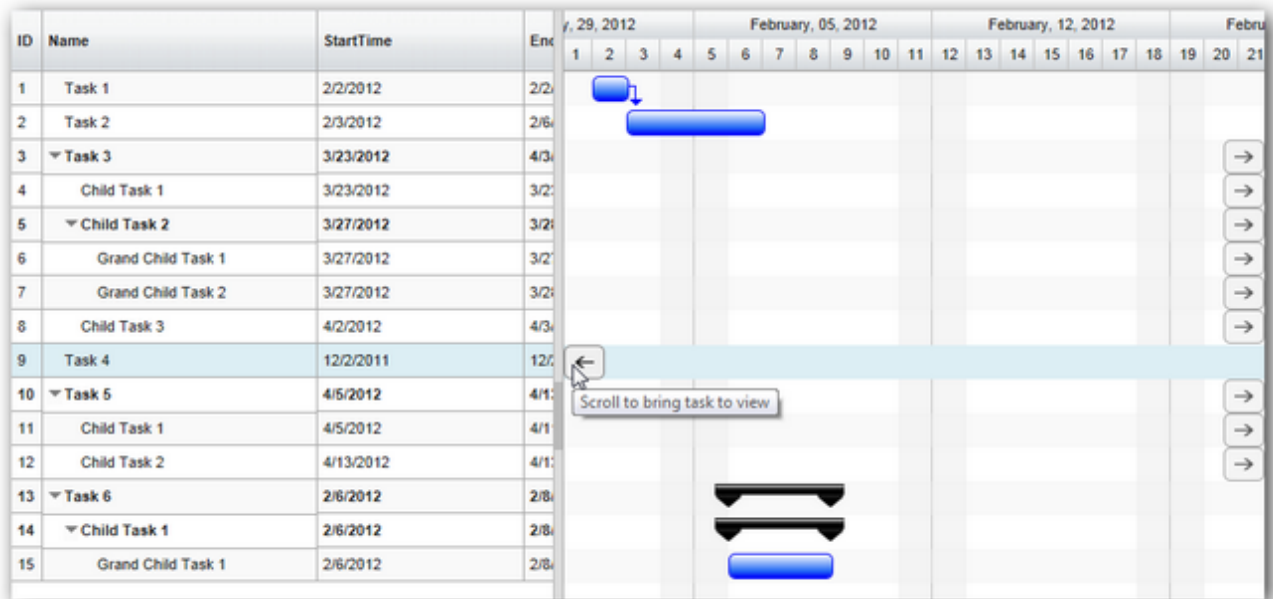
Take a look at this sample gantt chart view:



Gantt with some tasks out of view

In the above chart, several tasks in the middle are out of view and it's not very clear whether those tasks are to the left or to the right of the timeline, making this a bit annoying for the end-user.

With the "Browse To Task Cues" feature turned on, you can make this easier on the end-user with cues appearing to illustrate where exactly the hidden tasks are (to the right or left).



Gantt with Browse to Task Cues On

This gives a much clearer picture to the user and also gives him an option to easily bring a task into view (by just clicking on the cue button).

This feature is easily turned on automatically when you specify the look and feel to be used for these cues, through the `TaskBarBrowseToCueLeftTemplate` and `TaskBarBrowseToCueRightTemplate` as follows:

In ProjectGantt:

```
$gantt_container.GanttControl({
  DataSource: self.jsonData,
  TaskBarBrowseToCueLeftTemplate: "<button></button>",
  TaskBarBrowseToCueRightTemplate: "<button></button>"
});
```

In FlexyGantt:

```
$gantt_container.FlexyGantt({
  DataSource: self.jsonData,
  TaskBarBrowseToCueLeftTemplate: "<button></button>",
  TaskBarBrowseToCueRightTemplate: "<button></button>"
});
```

This is also illustrated in this sample that's part of our install:

ProjectGantt sample: <install path>\Samples\GanttTaskBarBrowseToCue.htm

FlexyGantt sample: <install path>\Samples\FlexyGanttBrowseToTasksCues.htm

RadiantQ jQuery Gantt Package

TimeSpan Paging

Accessing the Pager Buttons

There are buttons at the bottom right and bottom left to page the view right and left respectively. You can access them and disable them for example as follows:

In HTML

```
$('#container').FlexyGantt({
    .....
    .....
    GanttChartTemplateApplied: function (sender , args) {
        var ganttChart = args.element;
        //setting anchor time to chart
        ganttChart.setOption("AnchorTime", anchorTime);
        //To disable pager buttons
        ganttChart.setOption("EnablePagerButtonRight", false);
        ganttChart.setOption("EnablePagerButtonLeft", false);
    }
});
```

In ASP.NET MVC

```
@Html.JQFlexyGantt(
    new JQFlexyGanttSettings()
    {
        Options = new FlexyGanttOptions()
        {
            ...
            GanttChartOptions = new GanttChartOptions()
            {
                AnchorTime = DateTime.Today,
                EnablePagerButtonRight= false,
                EnablePagerButtonLeft= false
            }
        }
    }
)
```

In ASP.NET

```
<script runat="server">
    protected void gantt_Load(object sender, EventArgs e)
    {
        this.gantt.GanttChartOptions.EnablePagerButtonLeft = false;
        this.gantt.GanttChartOptions.EnablePagerButtonRight = false;
    }
</script>

<RQ:GanttControl ID="gantt" OnLoad="gantt_Load" .. runat="server" />
```

Customize Paging

Sometimes you will want to "page" by a custom span, rather than the default paging behavior. You can do so, by listening to the above pager button clicks and page by your preferred time span.

In the example below, we page by 5 days, instead of the default paging behavior.

In HTML

```

$gantt_container = $('#gantt_container');
$gantt_container.FlexyGantt({
.....
.....
    GanttChartTemplateApplied: function (sender , args) {
        var $GanttChart = args.element;
        $GanttChart.GanttChart({
            AnchorTime: anchorTime,
            BeforePagingGanttChart: function (sender, e) {
                var HScrollbar = $GanttChart.data("GanttChart").HScrollbar;
                if (sender == "LeftPagerButton" && HScrollbar[0].scrollLeft
== 0) {
                    this.AnchorTime = this.AnchorTime.clone().addDays(-5);
                }
                if (sender == "RightPagerButton" && HScrollbar.width() +
HScrollbar[0].scrollLeft == HScrollbar[0].scrollWidth) {
                    this.AnchorTime = this.AnchorTime.clone().addDays(5);
                }
            }
        });
    }
});

```

In ASP.NET , ASP.NET MVC

```

AfterGanttWidgetInitializedCallback = function () {

    ganttChart = $("#rq_gantt").data("FlexyGantt").GetGanttChart();
    ganttChart.GanttChart({
        BeforePagingGanttChart: function (sender , e) {
            var HScrollbar = ganttChart.data("GanttChart").HScrollbar;
            if (sender == "LeftPagerButton" && HScrollbar[0].scrollLeft == 0) {
                this.AnchorTime = this.AnchorTime.clone().addDays(-5);
            }
            if (sender == "RightPagerButton" && HScrollbar.width() +
HScrollbar[0].scrollLeft == HScrollbar[0].scrollWidth) {
                this.AnchorTime = this.AnchorTime.clone().addDays(5);
            }
        }
    });
});

```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

Events

RadiantQ jQuery Gantt Package

GanttChart Events.

GanttChart Events

Chart Time Span Changing.

The Chart's time span could change due to a variety of reasons - the end-user zooming, paging, AnchorTime change, etc. A reliable way to listen to all these changes is like this:

```
// end-user zooming
$('#flexyGantt_container').FlexyGantt({
  // Raised when the user zooms the view.
  Zoomed: function ( sender, args )
  {
  }
});

//end-user paging.
$('#flexyGantt_container').FlexyGantt({
  // Raised when the user zooms the view.
  AnchorTimeChanged: function ( sender, args )
  {
  }
});
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

Scheduling Features

RadiantQ jQuery Gantt Package

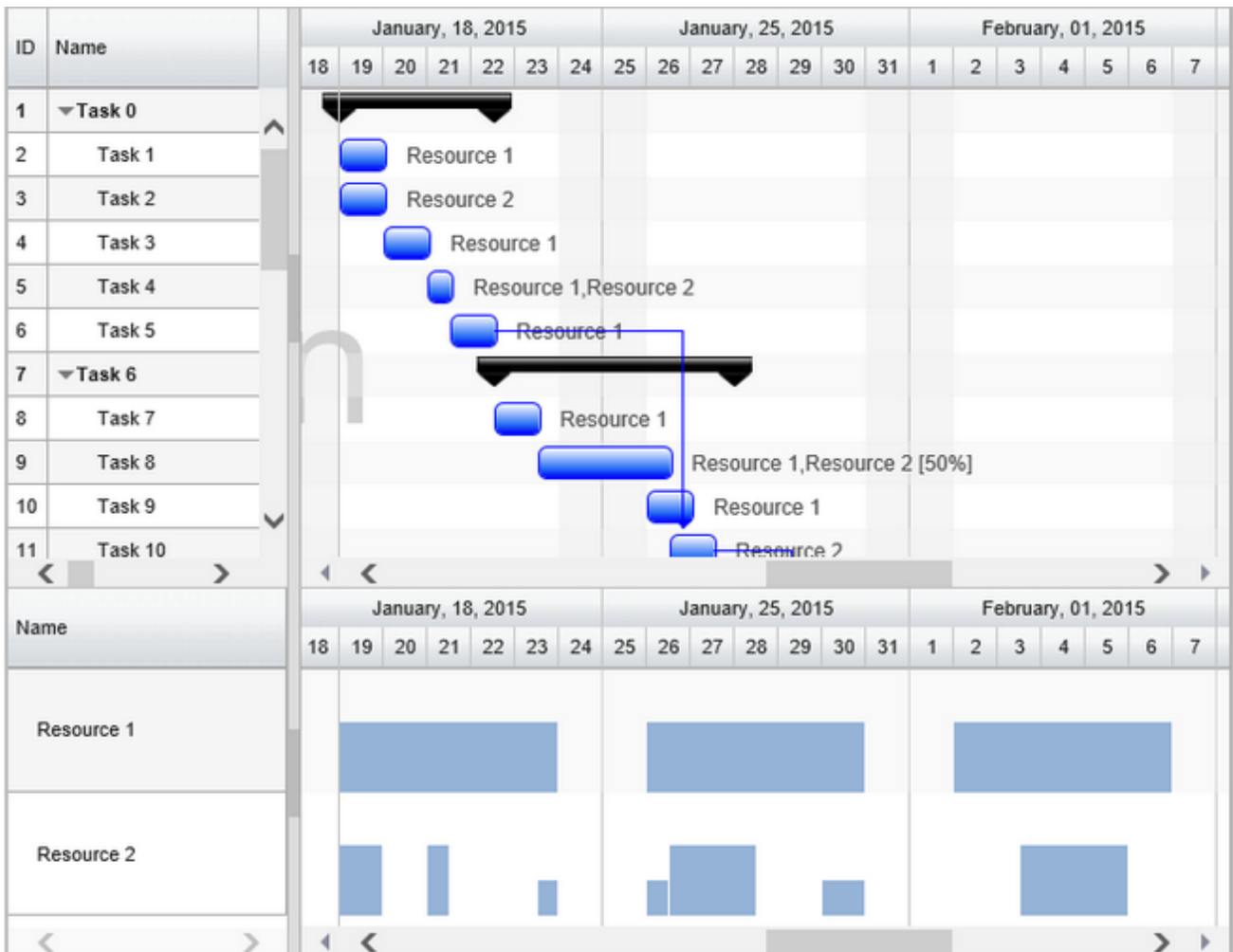
Resource Load View

Resource Load View

The Gantt library now includes some utility classes that helps you compute the resources' load over a specific time span, take that information and build a resource load view using or FlexyGantt.

ResourceLoadTracker is the utility class that is capable of providing you load information for a resource given a GanttControl's Model instance.

Here is the Resource Load View (at the bottom) built using the above ResourceLoadTracker and the FlexyGantt.



Resource Load View

This sample that illustrates this feature can be located here in the install:

In HTML : ..\Samples\ResourceLoadView.htm.
 In ASP.NET MVC : ..\Views\Home\ProjectGantt\ResourceLoadView.cshtml.
 In ASP.NET : ..\Samples\ProjectGantt\ResourceLoadView.aspx.

Step 1: ResourceLoadTracker

When you are ready to get the load information from the ResourceLoadTracker, you can do as follows:

```
var model = ganttControl.Model;
var _loadTracker = new RadiantQ.Gantt.Model.ResourceLoadTracker(model);

$flexyGantt_container.FlexyGantt({ DataSource:
_loadTracker.ResourceLoadLists_M().asArray });
```

Initialize the tracker with the following:

Model instance - the tracker then builds the load information for all the Resources that are currently in the Model.

Then simply assign the ResourceLoadLists as the FlexyGantt's ItemsSource as above.

Note that the above list contains a ResourceLoadList instance for each Resource in the GanttControl's Model.

A ResourceLoadList then contains the load information as an array of LoadInfo instances (LoadInfos property) like this :

```
{
  TimePeriod: 1/15/2014 8AM - 2/15/2014 4PM;
  LoadPerc: 110; // A double value
},
etc.
```

Step 2: Customizing FlexyGantt

Apply appropriate templates to the FlexyGantt to make it show the load information as bars within the time ranges.

```
// The template that defines the look for the task bars. "rq-gc-taskbar"
is a built-in style that defines a default look for the task bars.
var tTemplate = "<div class='taskBar' style=\"background-color:
${UpdateBackgroundColorBinding(data)}; !important;
height:${UpdateHeightBinding(data)} !important; margin-top:${UpdateTopMargin(data)}
!important;\"></div>";
$flexyGantt_container = $('#flexyGantt_container');
$flexyGantt_container.FlexyGantt({
  TaskItemTemplate: tTemplate,
  TaskEndTimeProperty: "TimePeriod.End",
  TasksListProperty: "LoadInfos",
  .....
  .....
});
```

Note that the height and color of the bars is bound to the LoadPerc value in the TaskItemTemplate above.

The required converters are implemented in the above mentioned sampled.

Printing and Export

RadiantQ jQuery Gantt Package

Printing

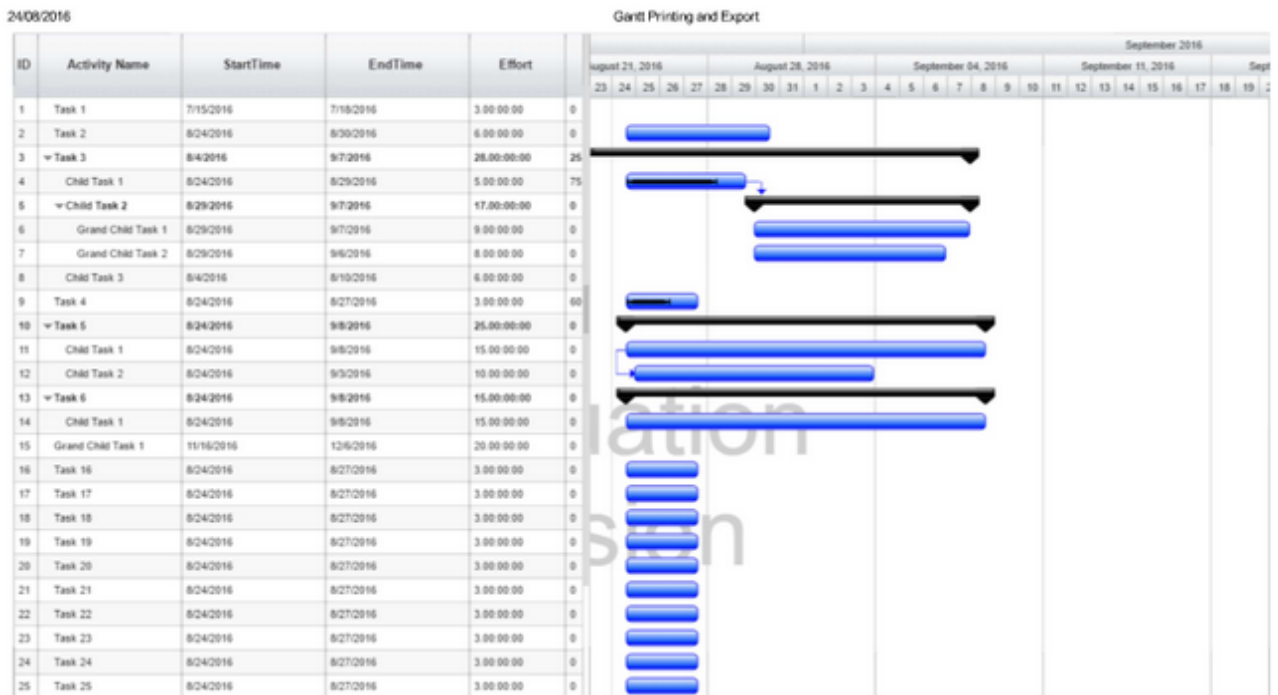
Printing

By default, the gantt will print with the currently visible grid table and timeline in the chart with about 25 rows per page.

For example,

```
RadiantQ.Gantt.RQPrint($gantt_container.data("GanttControl"));
```

Print output,



24/08/2016

Gantt Printing and Export

ID	Activity Name	StartTime	EndTime	Effort	September 2016																												
					August 21, 2016							August 28, 2016							September 04, 2016							September 11, 2016							
					23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
26	Task 26	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 26]																											
27	Task 27	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 27]																											
28	Task 28	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 28]																											
29	Task 29	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 29]																											
30	Task 30	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 30]																											
31	Task 31	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 31]																											
32	Task 32	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 32]																											
33	Task 33	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 33]																											
34	Task 34	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 34]																											
35	Task 35	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 35]																											
36	Task 36	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 36]																											
37	Task 37	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 37]																											
38	Task 38	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 38]																											
39	Task 39	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 39]																											
40	Task 40	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 40]																											
41	Task 41	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 41]																											
42	Task 42	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 42]																											
43	Task 43	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 43]																											
44	Task 44	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 44]																											
45	Task 45	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 45]																											
46	Task 46	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 46]																											
47	Task 47	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 47]																											
48	Task 48	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 48]																											
49	Task 49	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 49]																											
50	Task 50	8/24/2016	8/27/2016	3:00:00:00	0	[Gantt bars for Task 50]																											

<http://localhost:50178/Samples/GanttPrintingAndExport.htm>

2/3

Default Gantt printing (page

2)

24/08/2016 Gantt Printing and Export

ID	Activity Name	StartTime	EndTime	Effort	September 2016																							
					August 21, 2016					August 28, 2016					September 04, 2016					September 11, 2016					Septem			
					23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
51	Task 51	8/24/2016	8/27/2016	3.00:00:00	0																							
52	Task 52	8/24/2016	8/27/2016	3.00:00:00	0																							
53	Task 53	8/24/2016	8/27/2016	3.00:00:00	0																							
54	Task 54	8/24/2016	8/27/2016	3.00:00:00	0																							
55	Task 55	8/24/2016	8/27/2016	3.00:00:00	0																							

<http://localhost:60176/Samples/GanttPrintingAndExport.htm>

3/3

Default Gantt printing (page

3)

There are some print options available to customize the default behavior as described below.

1) Customizing the number of rows in page

Use the `NoOfRowsInPage` option before printing to desire how many rows should print per page.

For example,

```
var printOptions = new RadiantQ.Gantt.PrintOptions();
printOptions.NoOfRowsInPage = 50;

RadiantQ.Gantt.RQPrint($gantt_container.data("GanttControl"), printOptions);
```

Print output,

24/08/2016 Gantt Printing and Export

ID	Activity Name	StartTime	EndTime	Effort	Gantt Chart																			
					August 21, 2016				August 28, 2016				September 04, 2016											
					23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11
1	Task 1	7/15/2016	7/18/2016	3.00:00.00	█																			
2	Task 2	8/24/2016	9/30/2016	6.00:00.00																				
3	Task 3	8/4/2016	9/7/2016	26.00:00.00	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	
4	Child Task 1	8/24/2016	8/26/2016	5.00:00.00	█	█	█	█	█															
5	Child Task 2	8/29/2016	9/7/2016	17.00:00.00																				
6	Grand Child Task 1	8/29/2016	9/7/2016	9.00:00.00																				
7	Grand Child Task 2	8/29/2016	9/6/2016	8.00:00.00																				
8	Child Task 3	8/4/2016	8/18/2016	6.00:00.00	█	█	█	█	█	█														
9	Task 4	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
10	Task 5	8/24/2016	8/30/2016	26.00:00.00	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	
11	Child Task 1	8/24/2016	8/30/2016	15.00:00.00	█	█	█	█	█	█														
12	Child Task 2	8/24/2016	9/3/2016	10.00:00.00	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	
13	Task 6	8/24/2016	9/6/2016	15.00:00.00	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	
14	Child Task 1	8/24/2016	8/30/2016	15.00:00.00	█	█	█	█	█	█														
15	Grand Child Task 1	11/18/2016	12/6/2016	20.00:00.00																				
16	Task 16	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
17	Task 17	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
18	Task 18	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
19	Task 19	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
20	Task 20	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
21	Task 21	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
22	Task 22	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
23	Task 23	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
24	Task 24	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
25	Task 25	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
26	Task 26	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
27	Task 27	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
28	Task 28	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
29	Task 29	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
30	Task 30	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
31	Task 31	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
32	Task 32	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
33	Task 33	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
34	Task 34	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
35	Task 35	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
36	Task 36	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
37	Task 37	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
38	Task 38	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
39	Task 39	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
40	Task 40	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
41	Task 41	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
42	Task 42	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
43	Task 43	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
44	Task 44	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
45	Task 45	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
46	Task 46	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
47	Task 47	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
48	Task 48	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
49	Task 49	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																
50	Task 50	8/24/2016	8/27/2016	3.00:00.00	█	█	█	█																

rows per page (page 1)

Gantt printing with the given number of

24/08/2016 Gantt Printing and Export

ID	Activity Name	StartTime	EndTime	Effort	August 21, 2016							August 26, 2016							September 04, 2016							
					23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11		
51	Task 51	8/24/2016	8/27/2016	3:90:00.00	0																					
52	Task 52	8/24/2016	8/27/2016	3:90:00.00	0																					
53	Task 53	8/24/2016	8/27/2016	3:90:00.00	0																					
54	Task 54	8/24/2016	8/27/2016	3:90:00.00	0																					
55	Task 55	8/24/2016	8/27/2016	3:90:00.00	0																					

Gantt printing with the given number of rows per page (page 2)

2) Printing only the Gantt Chart

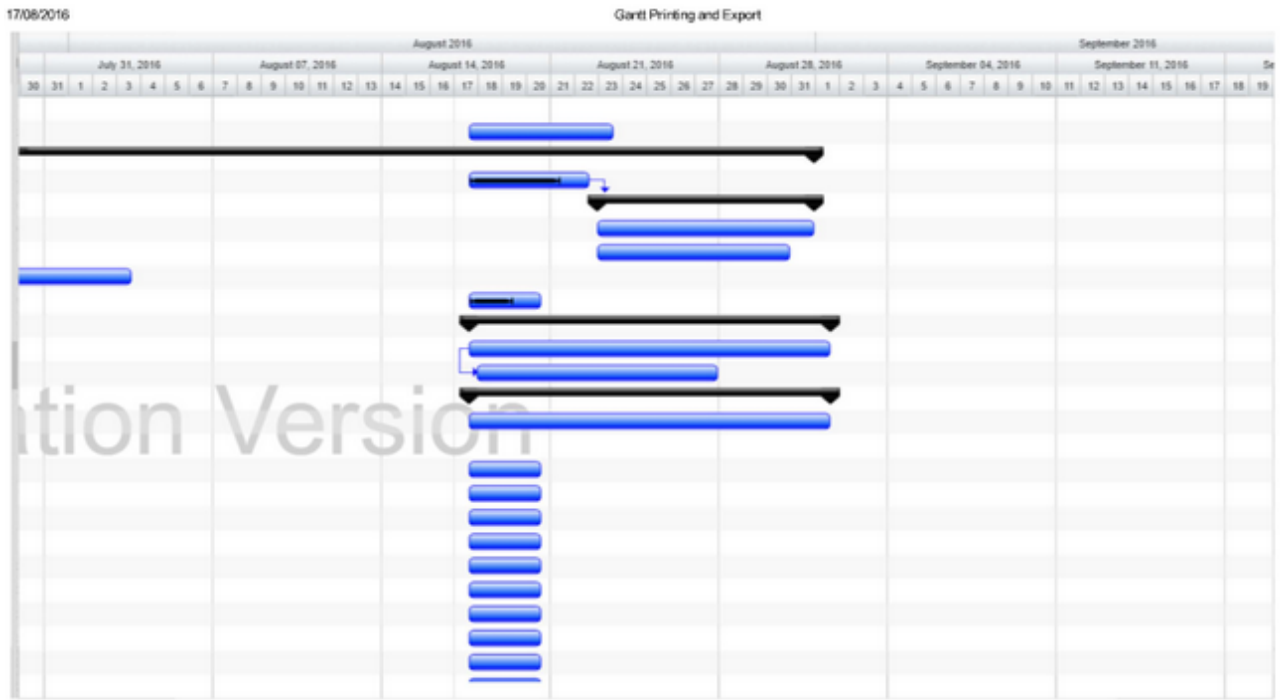
Enable ShowGanttChartOnly option before printing to hide the grid table and to print only the gantt chart.

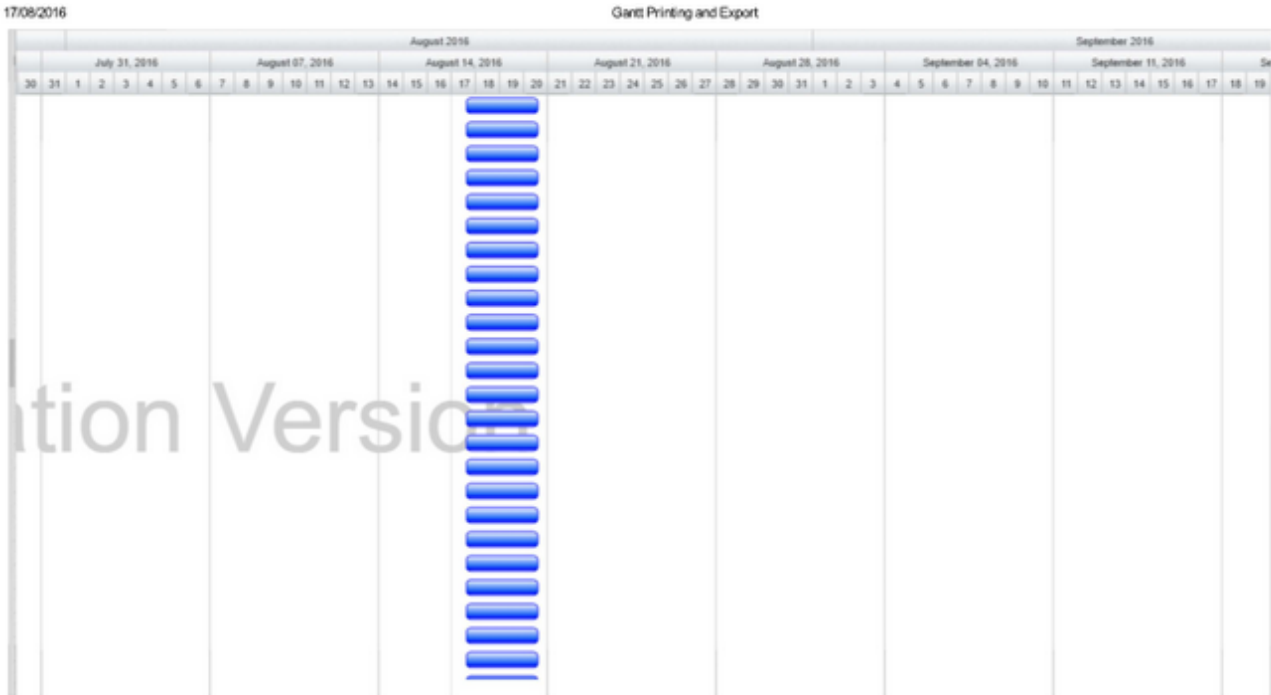
For example,

```
var printOptions = new RadiantQ.Gantt.PrintOptions();
printOptions.ShowGanttChartOnly = true;

RadiantQ.Gantt.RQPrint($gantt_container.data("GanttControl"), printOptions);
```

Print output,





gantt chart (page 2)

Gantt printing on showing only the



Rectangular Snip

<http://localhost:60178/Samples/GanttPrintingAndExport.htm>

3/3

gant chart (page 3)

Gantt printing on showing only the

3) Printing only the Grid table

Enable ShowGanttTableOnly option before printing to hide the gantt chart and to show only the grid table.

For example,

```
var printOptions = new RadiantQ.Gantt.PrintOptions();
printOptions.ShowGanttTableOnly = true;

RadiantQ.Gantt.RQPrint($gantt_container.data("GanttControl"), printOptions);
```

Print output,

17/08/2016

Gantt Printing and Export

ID	Activity Name	StartTime	EndTime	Effort	ProgressPercent	Resource	PredecessorIndex
1	Task 1	7/8/2016	7/11/2016	3.00:00:00	0		
2	Task 2	8/17/2016	8/23/2016	6.00:00:00	0		
3	Task 3	7/28/2016	8/31/2016	28.00:00:00	25		
4	Child Task 1	8/17/2016	8/22/2016	5.00:00:00	75		
5	Child Task 2	8/22/2016	8/31/2016	17.00:00:00	0	4-8	
6	Grand Child Task 1	8/22/2016	8/31/2016	9.00:00:00	0		
7	Grand Child Task 2	8/22/2016	8/30/2016	8.00:00:00	0		
8	Child Task 3	7/28/2016	8/3/2016	6.00:00:00	0		
9	Task 4	8/17/2016	8/20/2016	3.00:00:00	60		
10	Task 5	8/17/2016	9/1/2016	25.00:00:00	0		
11	Child Task 1	8/17/2016	9/1/2016	15.00:00:00	0		
12	Child Task 2	8/17/2016	8/27/2016	10.00:00:00	0		1155-8
13	Task 6	8/17/2016	9/1/2016	15.00:00:00	0		
14	Child Task 1	8/17/2016	9/1/2016	15.00:00:00	0		
15	Grand Child Task 1	11/9/2016	11/29/2016	20.00:00:00	0		
16	Task 16	8/17/2016	8/29/2016	3.00:00:00	0		
17	Task 17	8/17/2016	8/29/2016	3.00:00:00	0		
18	Task 18	8/17/2016	8/29/2016	3.00:00:00	0		
19	Task 19	8/17/2016	8/29/2016	3.00:00:00	0		
20	Task 20	8/17/2016	8/29/2016	3.00:00:00	0		
21	Task 21	8/17/2016	8/29/2016	3.00:00:00	0		
22	Task 22	8/17/2016	8/29/2016	3.00:00:00	0		
23	Task 23	8/17/2016	8/29/2016	3.00:00:00	0		
24	Task 24	8/17/2016	8/29/2016	3.00:00:00	0		
25	Task 25	8/17/2016	8/29/2016	3.00:00:00	0		

<http://localhost:60178/Samples/GanttPrintingAndExport.htm>

1/3

table (page 1)

Gantt printing on showing only the gantt

17/08/2016

Gantt Printing and Export

ID	Activity Name	StartTime	EndTime	Effort	ProgressPercent	Resource	PredecessorIndex
26	Task 26	8/17/2016	8/29/2016	3.00:00:00	0		
27	Task 27	8/17/2016	8/29/2016	3.00:00:00	0		
28	Task 28	8/17/2016	8/29/2016	3.00:00:00	0		
29	Task 29	8/17/2016	8/29/2016	3.00:00:00	0		
30	Task 30	8/17/2016	8/29/2016	3.00:00:00	0		
31	Task 31	8/17/2016	8/29/2016	3.00:00:00	0		
32	Task 32	8/17/2016	8/29/2016	3.00:00:00	0		
33	Task 33	8/17/2016	8/29/2016	3.00:00:00	0		
34	Task 34	8/17/2016	8/29/2016	3.00:00:00	0		
35	Task 35	8/17/2016	8/29/2016	3.00:00:00	0		
36	Task 36	8/17/2016	8/29/2016	3.00:00:00	0		
37	Task 37	8/17/2016	8/29/2016	3.00:00:00	0		
38	Task 38	8/17/2016	8/29/2016	3.00:00:00	0		
39	Task 39	8/17/2016	8/29/2016	3.00:00:00	0		
40	Task 40	8/17/2016	8/29/2016	3.00:00:00	0		
41	Task 41	8/17/2016	8/29/2016	3.00:00:00	0		
42	Task 42	8/17/2016	8/29/2016	3.00:00:00	0		
43	Task 43	8/17/2016	8/29/2016	3.00:00:00	0		
44	Task 44	8/17/2016	8/29/2016	3.00:00:00	0		
45	Task 45	8/17/2016	8/29/2016	3.00:00:00	0		
46	Task 46	8/17/2016	8/29/2016	3.00:00:00	0		
47	Task 47	8/17/2016	8/29/2016	3.00:00:00	0		
48	Task 48	8/17/2016	8/29/2016	3.00:00:00	0		
49	Task 49	8/17/2016	8/29/2016	3.00:00:00	0		
50	Task 50	8/17/2016	8/29/2016	3.00:00:00	0		

<http://localhost:60178/Samples/GanttPrintingAndExport.htm>

2/3

gantt table (page 2)

Gantt printing on showing only the

17/08/2016

Gantt Printing and Export

ID	Activity Name	StartTime	EndTime	Effort	ProgressPercent	Resource	PredecessorIndex
51	Task 51	8/17/2016	8/29/2016	3.00:00:00	0		
52	Task 52	8/17/2016	8/29/2016	3.00:00:00	0		
53	Task 53	8/17/2016	8/29/2016	3.00:00:00	0		
54	Task 54	8/17/2016	8/29/2016	3.00:00:00	0		
55	Task 55	8/17/2016	8/29/2016	3.00:00:00	0		

<http://localhost:60178/Samples/GanttPrintingAndExport.htm>

3/3

Gantt printing on showing only the gantt

table (page 3)

4) Printing by range index

Use the `StartRowIndex` and `EndRowIndex` options before printing to print gantt within the specified row indices.

For example,

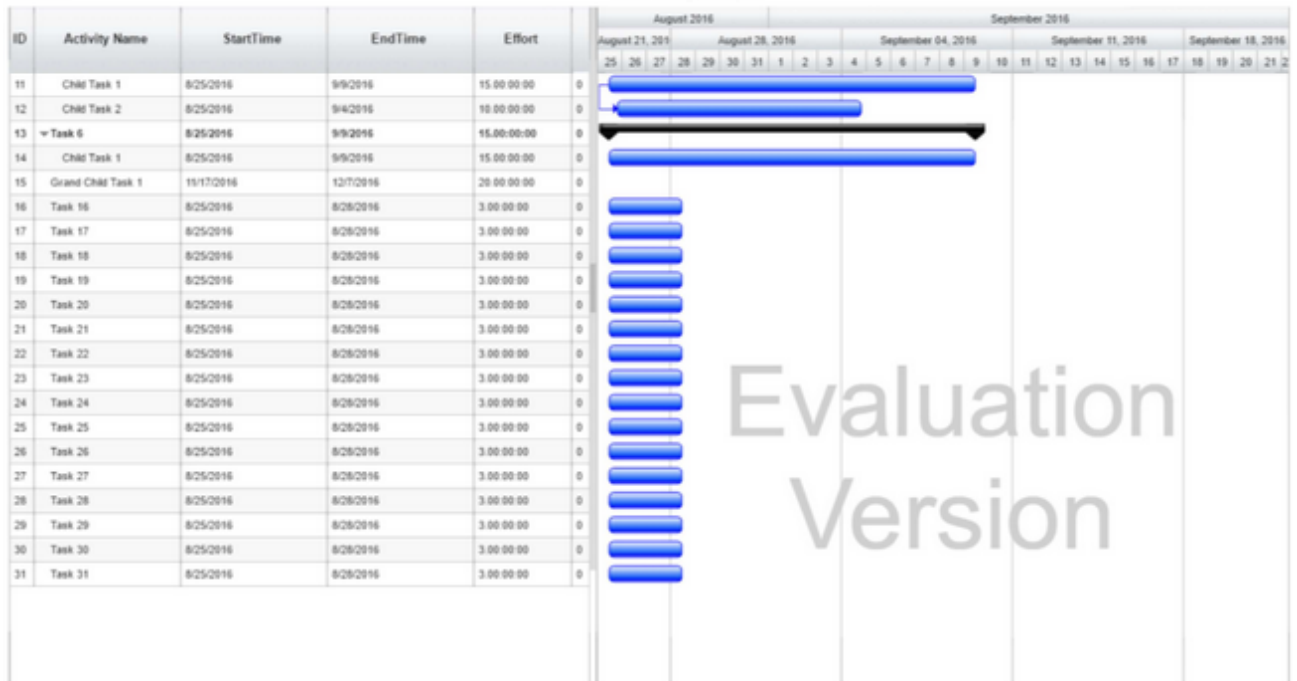
```
var printOptions = new RadiantQ.Gantt.PrintOptions();
printOptions.StartRowIndex = 10;
printOptions.EndRowIndex = 30;

RadiantQ.Gantt.RQPrint($ganttt_container.data("GanttControl"), printOptions);
```

Print output,

25/08/2016

Gantt Printing and Export



Evaluation
Version

<http://localhost:60178/Samples/GanttPrintingAndExport.htm>

1/1

row indices

Gantt printing within the specified

5) Customizing the title of the page

Use the Title option to overwrite the default title of the print output.

For example,

```
var printOptions = new RadiantQ.Gantt.PrintOptions();
printOptions.Title = "Custom Print Title";

RadiantQ.Gantt.RQPrint($gantt_container.data("GanttControl"), printOptions);
```

6) Selecting the Columns to print

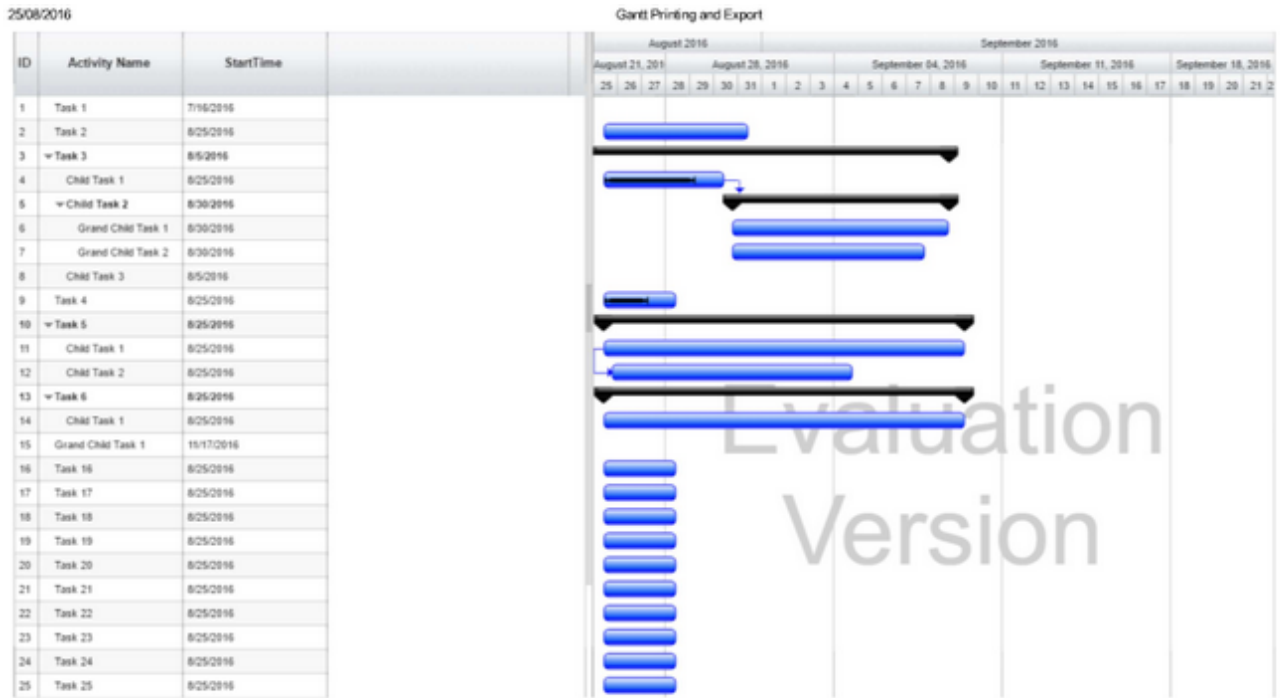
Use the VisibleColumnIndices to specify the column indices which should be included in the print output.

For example,


```
var printOptions = new RadiantQ.Gantt.PrintOptions();
printOptions.VisibleColumnIndices = [0, 1, 2];
```

```
RadiantQ.Gantt.RQPrint($gantt_container.data("GanttControl"), printOptions);
```

Print output,



columns (page 1)

Gantt output with a subset of the

25/08/2016

Gantt Printing and Export

ID	Activity Name	StartTime	August 2016																																		
			August 21, 2016							August 28, 2016							September 04, 2016							September 11, 2016							September 18, 2016						
			25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22						
26	Task 26	8/25/2016																																			
27	Task 27	8/25/2016																																			
28	Task 28	8/25/2016																																			
29	Task 29	8/25/2016																																			
30	Task 30	8/25/2016																																			
31	Task 31	8/25/2016																																			
32	Task 32	8/25/2016																																			
33	Task 33	8/25/2016																																			
34	Task 34	8/25/2016																																			
35	Task 35	8/25/2016																																			
36	Task 36	8/25/2016																																			
37	Task 37	8/25/2016																																			
38	Task 38	8/25/2016																																			
39	Task 39	8/25/2016																																			
40	Task 40	8/25/2016																																			
41	Task 41	8/25/2016																																			
42	Task 42	8/25/2016																																			
43	Task 43	8/25/2016																																			
44	Task 44	8/25/2016																																			
45	Task 45	8/25/2016																																			
46	Task 46	8/25/2016																																			
47	Task 47	8/25/2016																																			
48	Task 48	8/25/2016																																			
49	Task 49	8/25/2016																																			
50	Task 50	8/25/2016																																			

Evaluation Version

<http://localhost:60178/Samples/GanttPrintingAndExport.htm>

2/3

columns (page 2)

Gantt output with a subset of the



<http://localhost:60178/Samples/GanttPrintingAndExport.htm>

3/3

Gantt output with a subset of the columns (page 3)

7) Customizing the Gantt Chart Timeline

Use the `ViewStartTime` and `ViewEndTime` options while printing to customize the timeline with which to print the gantt chart.

For example,

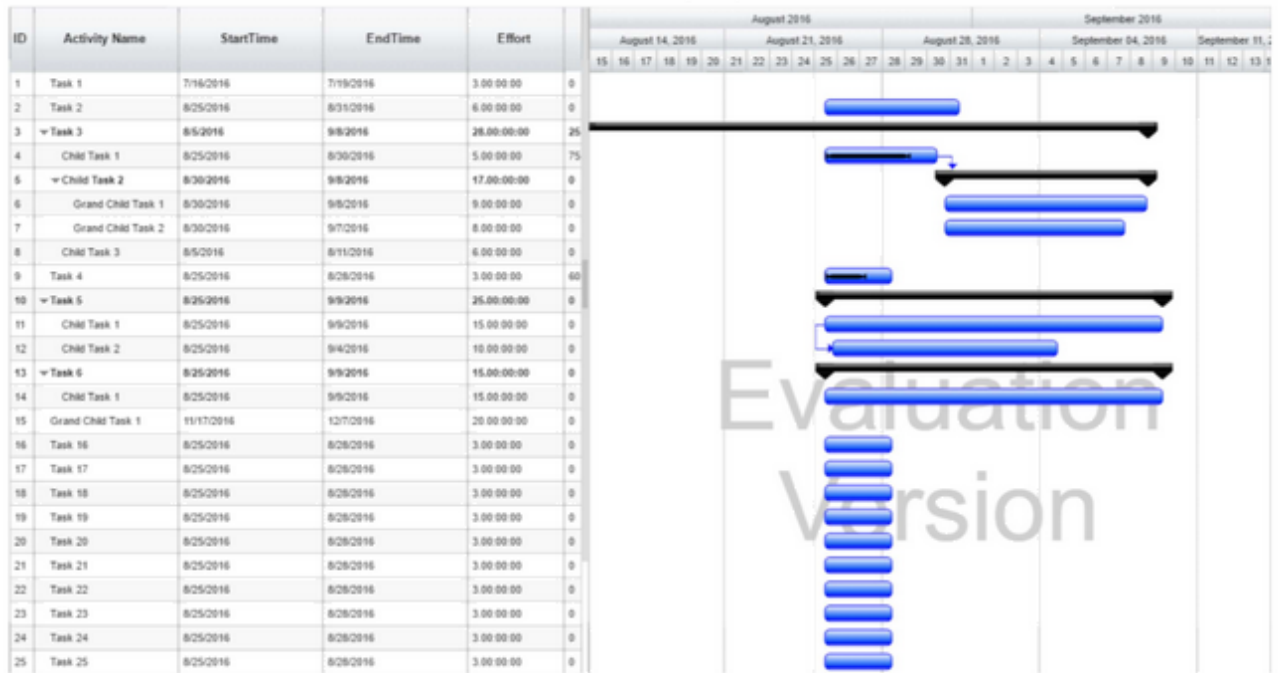
```
var printOptions = new RadiantQ.Gantt.PrintOptions();
printOptions.ViewStartTime= new Date(2016, 7, 15);
printOptions.ViewEndTime= new Date(2016, 8, 15);

RadiantQ.Gantt.RQPrint($gantt_container.data("GanttControl"), printOptions);
```

Print output,

25/08/2016

Gantt Printing and Export



Evaluation Version

ViewEndTime (page 1)

Gantt output with the given ViewStartTime and

25/08/2016 Gantt Printing and Export

ID	Activity Name	StartTime	EndTime	Effort	August 2016														September 2016												
					August 14, 2016					August 21, 2016					August 28, 2016				September 04, 2016				September 11, 2016								
					15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10
51	Task 51	8/25/2016	8/28/2016	3.00:00:00	0																										
52	Task 52	8/25/2016	8/28/2016	3.00:00:00	0																										
53	Task 53	8/25/2016	8/28/2016	3.00:00:00	0																										
54	Task 54	8/25/2016	8/28/2016	3.00:00:00	0																										
55	Task 55	8/25/2016	8/28/2016	3.00:00:00	0																										



<http://localhost:60178/Samples/GanttPrintingAndExport.htm>

33

Gantt output with the given ViewStartTime and ViewEndTime (page 3)

Required references for this feature

This feature requires you to include the "html2canvas.js" file in your web page. This file is available at ..\Samples\Scripts\html2canvas.js.

Gantt Printing is illustrated in "..\Samples\GanttPrintingAndExport.htm" sample.

NOTE : Currently we are not supporting Printing and ExportingToImage in IE8.

© RadiantQ 2009-2018. All Rights Reserved.

RadiantQ jQuery Gantt Package

Export

Export To Image

By default, the gantt will export to an image with the full grid table and the currently visible timeline in the chart. In export to image, it will prompt you to download the image with the extension of ".png", which you can save to your machine. Gantt can be exported to ".png" and ".jpg" image formats only.

To Export to Image:

```
$ganttElement.data("GanttControl").ExportToImage();
    (or)
$ganttElement.GanttControl("ExportToImage");
```

You can customize the format in which the image is exported as using the Extension option. 'jpg' and 'png' are the supported formats. 'png' is the default.

Here is the code example:

```
Gantt.ExportToImage({ Title: "fileName", Format: 'jpg' });
    (or)
Gantt.ExportToImage({ Title: "fileName", Format: 'png' });
```

In some cases, instead of saving the image to machine, you have to send the image to server. Take a look at this [topic](#) for more info on this.

Export To PDF

Gantt doesn't have any built in support to export as PDF but it provides a canvas which can be converted as a PDF files using jspdf utility.

Here is an Example.

```
var printOptions = new RadiantQ.Gantt.PrintOptions();
RadiantQ.Gantt.RQPrint($ganttElement.data("GanttControl"), printOptions,
afterCanvasGeneratedCallBack/*enabling 'ExportToPDF' option*/);
```

The afterCanvasGeneratedCallBack is a method which will provide the Gantt as canvas.

Here is the example code that shows how to convert Gantt canvas to PDF:

```
function afterCanvasGeneratedCallBack(canvasArr) {
    var doc = new jsPDF('l', 'pt', [canvasArr[0].width, canvasArr[0].height]);
    for (var i = 0; i < canvasArr.length; i++) {
        var canvas = canvasArr[i];
        var imgData = canvas.toDataURL('image/png');
        doc.addImage(imgData, 'png', 0, 0, canvas.width, canvas.height);
    }
    doc.save('Gantt.pdf');
    // return 'false' to enable PDF option.
    // return 'true' to continue with default printing.
    return false;
}
```

The jspdf.debug.js is placed inside the <Installedolder>/Scripts folder. You can get more information about the jsPDF file in the [jsPDF documentation](#) online.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

JS Patterns

RadiantQ jQuery Gantt Package

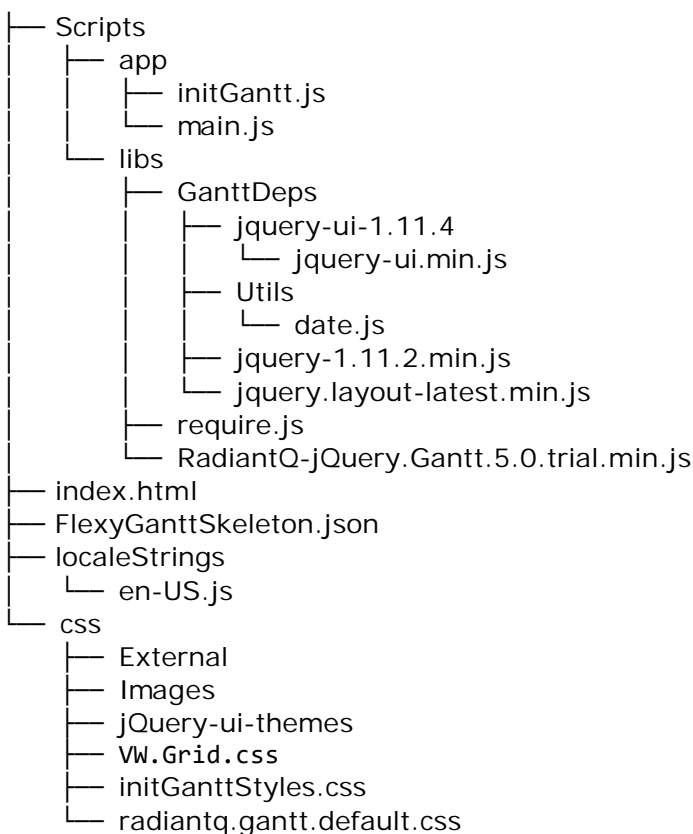
RequireJS

[RequireJS](#) is an implementation of AMD (Asynchronous Module Definition), an API for declaring modules and loading them asynchronously when they are needed. RequireJS can help you organize your code with modules and will manage for you the asynchronous and parallel downloads of your files. Since scripts are loaded only when needed, it reduces the loading time of your page, which is great!

In this topic we discuss how to use the jQuery Gantt in a RequireJS project.

The discussion in this topic is illustrated in the ..\PatternSamples\RequireJS sample.

Create your project directory and structure it as you need. Ours looks like this,



In index.html, include code to:

- "data-main Entry Point", The data-main attribute is a special attribute that require.js will check to start script loading. Take a look at this [RequireJS Docs](#) for more information about this.
- The templates to implement the gantt.
- The config object as the global variable required before require.js is loaded and have the values applied automatically. Add the RadiantQ and dependant files in this config object. Take a look at this [RequireJS Doc](#) for more information.

```

<head>
  <script data-main="Scripts/app/main.js" src="Scripts/libs/require.js"></script>
  <script>
    requirejs.config({
      "baseUrl": "Scripts/app",
      "paths": {
        "jquery": "../libs/GanttDeps/jquery-1.11.2.min",
        "jquery-ui": "../libs/GanttDeps/jquery-ui-1.11.4/jquery-ui.min",
        "jquery.layout": "../libs/GanttDeps/jquery.layout-latest.min",
        "dateJS": "../libs/GanttDeps/Utils/date",
        "resourceString": "../../localeStrings/en-US",
        "underscore": "../libs/underscore",
        "backbone": "../libs/backbone",
        "router": "../router",
        "models": "../models/model",
        "views": "../views/view",
        "collections": "../collections/collection",
        "VWGrid": "../libs/VW.Grid.1.min.js",
        "RadiantQ": "../libs/RadiantQ-jQuery.Gantt.5.0.min.js"
      },
      "shim": {
        "jquery-ui": ["jquery"],
        "jquery.layout": ["jquery", "jquery-ui"],
        "backbone": ["underscore"],
        "router": ["backbone"],
        "models": ["router", "RadiantQ"],
        "collections": ["models"],
        "views": ["collections"],
        "RadiantQ": ["jquery", "jquery-ui", "jquery.layout", "dateJS",
"resourceString", "VWGrid"]
      }
    });
  </script>
</head>

```

The main.js should contain code that needs execute before initializing the Gantt, for example, "loading the ItemSource from json file".

```

var jsonData = [];
define(["RadiantQ"], function () {
  $.ajax({
    type: "GET",
    dataType: 'json',
    url: 'FlexyGanttSkeleton.json',
    converters:
    {
      "text json": function (data) {
        return $.parseJSON(data, true /*converts date strings to date
objects*/, true /*converts ISO dates to local dates*/);
      }
    },
    success: function (data) {
      jsonData = data;
      require(["app/initGantt"]);
    }
  });
});

```

In initGantt.js:

```
// Gantt widget initialization & related codes goes here.
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

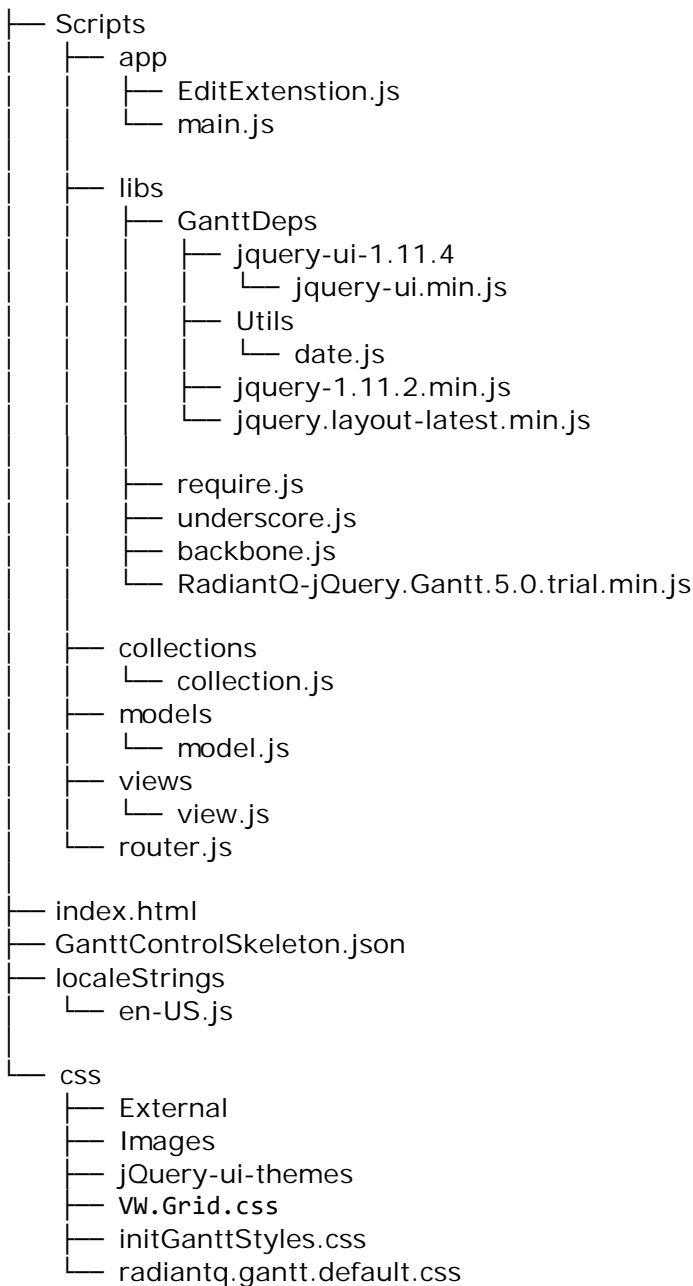
*RadiantQ jQuery Gantt Package***Backbone.JS**

[Backbone.js](#) gives structure to web applications by providing models with key-value binding and custom events, collections with a rich API of enumerable functions, views with declarative event handling, and connects it all to your existing API over a RESTful JSON interface.

In this topic we discuss how to use the jQuery Gantt in a BackboneJS project.

The discussion in this topic is illustrated in the ..\PatternSamples\BackboneJS sample.

Create your project directory and structure it as you need. The sample above looks like this,



In index.html, include the following code:

- "data-main Entry Point", The data-main attribute is a special attribute that require.js will check to start script loading. Take a look at this [RequireJS Docs](#) for more information about this.
- The templates to implement the gantt.
- The config object as the global variable required before require.js is loaded and have the values applied automatically. Add the RadiantQ and dependant files in this config object. The above mentioned *RequireJS Doc* has more information.

```

<head>
  <script data-main="Scripts/app/main.js" src="Scripts/libs/require.js"></script>
  <script>
    requirejs.config({
      baseUrl: "Scripts/app",
      paths: {
        "jquery": "../libs/GanttDeps/jquery-1.11.2.min",
        "jquery-ui": "../libs/GanttDeps/jquery-ui-1.11.4/jquery-ui.min",
        "jquery.layout": "../libs/GanttDeps/jquery.layout-latest.min",
        "dateJS": "../libs/GanttDeps/Utils/date",
        "resourceString": "../../localeStrings/en-US",
        "underscore": "../libs/underscore",
        "backbone": "../libs/backbone",
        "router": "../router",
        "models": "../models/model",
        "views": "../views/view",
        "collections": "../collections/collection",
        "VWGrid": "../libs/VW.Grid.1.min.js",
        "RadiantQ": "../libs/RadiantQ-jQuery.Gantt.5.0.min.js"
      },
      "shim": {
        "jquery-ui": ["jquery"],
        "jquery.layout": ["jquery", "jquery-ui"],
        "backbone": ["underscore"],
        "router": ["backbone"],
        "models": ["router", "RadiantQ"],
        "collections": ["models"],
        "views": ["collections"],
        "RadiantQ": ["jquery", "jquery-ui", "jquery.layout", "dateJS",
"resourceString", "VWGrid"]
      }
    });
  </script>
</head>

```

In main.js you should create a new router after the "models", "views", "collections" and "RadiantQ" files are loaded.

```

// The object to handle the extended models, views & collections.
var BB = {
  Models: {},
  Collections: {},
  Views: {}
};

define(["jquery", "backbone", "models", "views", "collections", "RadiantQ"], function
() {
  // Starting the new router.
  var router = new BB.Router();
  Backbone.history.start();
});

```

In router.js you should create a new view of the Gantt.

```
// Backbone "router" is very useful for any application/feature that needs URL
routing/history capabilities.
BB.Router = Backbone.Router.extend({
  routes: {
    "": "index"
  },
  index: function () {
    // Create a new view for Gantt.
    var ganttView = new BB.Views.GanttView();
  }
});
```

In view.js:

```
BB.views.GanttView = Backbone.View.extend({
  el: $("#gantt_container"),
  model: BB.Models.Task,
  initialize: function () {
    this.collection = new BB.Collections.Tasks();
    _.bindAll(this, 'render');

    // To get the json data from given collection url,
    this.collection.fetch({
      converters: {
        "text json": function (data) {
          return $.parseJSON(data, true/*converts date strings to date
objects*/, true/*converts ISO dates to local dates*/);
        }
      },
      success: this.render
    });
  },
  render: function () {
    this.initGantt();
  },
  initGantt: function () {
    // Gantt widget initialization & related codes goes here.
  }
});
```

In model.js:

```
BB.Models.Task = Backbone.Model.extend({
  defaults: {
    ID: -1,
    Name: "",
    StartTime: new Date(),
    Effort: RQTimeSpan.Zero,
    PreferredStartTime: Date.MinValue,
    IndentLevel: 0,
    IsOpen: false,
    ProgressPercent: 0,
    Resources: "",
    PredecessorIndices: "",
    SortOrder: 0,
    Description: "",
    Tag: null,
    Priority: 0
  },
  initialize: function () {
  }
});
```

In collection.js:

```
BB.Collections.Tasks = Backbone.Collection.extend({
  model: BB.Models.Task,
  initialize: function () {
    this.url = 'GanttControlSkeleton.json';
  }
});
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

Bootstrap

In this topic we discuss how to use the RadiantQ jQuery Gantt along with [Bootstrap](#), in the process addressing any issues that need to be addressed.

Note: A working sample of this topic can be found in the `..\PatternSamples\Bootstrap` folder.

Including Bootstrap Files

a) Bootstrap js files

Include the Bootstrap JS files in a folder like this from our sample: `..\PatternSamples\Bootstrap\js`.

b) Bootstrap css files

Include the Bootstrap CSS files in a folder like this from our sample: `..\PatternSamples\Bootstrap\css`.

c) Fonts

Bootstrap uses some fonts and icon files that are used in it's Navbar and other controls. Include this Bootstrap folder in a path like this: `..\PatternSamples\Bootstrap\fonts`

Including Gantt References

a) HTML file referencing the Gantt Widget

First copy over all required Gantt files into the project folder. Please reference the above PatternSamples sample for the full set of files to include. This is also discussed in detail in the [GanttControl Getting Started](#) and [FlexyGantt Getting Started](#) topic.

Create a new HTML file inside the project directory (`GanttWithBootstrap.html`) and reference the following source files and the corresponding Bootstrap related js files and css files.


```

<head>
  <title></title>
  <link href="Src/Styles/jquery-ui-themes/smoothness/jquery-ui.css" rel="stylesheet"
 />
  <link href="Src/Styles/radiantq.gantt.default.css" rel="stylesheet" />
  <link href="css/bootstrap.min.css" rel="stylesheet"/>
  <link href="css/jquery.bootstrap-touchspin.css" rel="stylesheet"/>
  <link href="css/bootstrap-datetimepicker.css" rel="stylesheet"/>
  <link href="css/GanttWithBootstrap.css" rel="stylesheet"/>
  <script src="Src/Scripts/jquery-1.11.2.min.js" type="text/javascript"></script>
  <script src="Src/Scripts/jquery-ui-1.11.4/jquery-ui.min.js" type
="text/javascript"></script>
  <script type="text/javascript" src="Src/Scripts/jquery.layout-latest.min.js"></
script>
  <script src="Src/Scripts/Utils/date.js" type="text/javascript"></script>
  <script src="Src/ResourceStrings/en-US.js" type="text/javascript"></script>
  <script src='Src/Scripts/VW.Grid.1.min.js' type='text/javascript'></script>
  <script src='Src/Scripts/RadiantQ-jQuery.Gantt.5.0.min.js' type
='text/javascript'></script>
  <script src="js/bootstrap.min.js"></script>
  <script src="js/bootstrap-contextmenu.js"></script>
  <script src="js/jquery.bootstrap-touchspin.js"></script>
  <script src="js/moment.js"></script>
  <script src="js/bootstrap-datetimepicker.js"></script>
</head>

```

b) Sample JSON Data

Create a GanttWithBootstrap.json file containing a list of sample tasks to be displayed in the gantt.

Addressing conflicts between Bootstrap and jQuery UI

The Gantt widgets use jQuery UI and Bootstrap has some conflicts with certain jQuery UI plugins. We will address those here.

a) Tooltip styling conflicts.

To avoid some naming conflict between the jQuery UI Tooltip and Bootstrap tooltip, include this script:

```

<script>
  /** Handle jQuery plugin naming conflict between jQuery UI and Bootstrap
  ***/
  var bootstrapTooltip = $.fn.tooltip.noConflict();
  $.fn.bstooltip = bootstrapTooltip;
</script>

```

b) Sizing issue in Bootstrap css

Bootstrap.min.css redefines box-sizing to an undesirable setting and has to be reset for the dom elements within the gantt. We do so by including the following css:

```

#gantt_container *{
  -webkit-box-sizing: content-box !important;
  -moz-box-sizing: content-box !important;
  box-sizing: content-box !important;
}

```

The above code will be inside "..\PatternSamples\Bootstrap\css\GanttWithBootstrap.css", "GanttWithBootstrap.css" contains all sample level css styles definitions.

Including Bootstrap Features within the Gantt

a) Using bootstrap-contextmenu.js instead of Gantt's jQuery UI based built-in context menu.

The RadiantQ jQuery Gantt provides an easy way to plugin other context menu frameworks, thereby letting the gantt use them instead of it's built-in context menu framework. Below code plugs in the Bootstrap ContextMenu into the RadiantQ Gantt.

```

RadiantQ.Gantt.ContextMenuImpl.ContextMenu = function (className, scope) {
  var self = this;
  this.ItemClicked = new ObjectEvent("ItemClicked");
  this.BeforeContextMenu = new ObjectEvent("BeforeContextMenu");
  this.selector = className;
  var contextMenu = this;
  this._position = { x: 0, y: 0 };
  this.Items = new RadiantQ.Gantt.Dictionary();

  $(scope).contextmenu({
    target: '#context-menu',
    before: function (e, context) {
      $("#context-menu ul").empty();
      if (contextMenu.Items.length != 0) {
        self.BeforeContextMenu.raise(context, e);
        // execute code before context menu if shown
        var menuItems = contextMenu.Items.asArray;
        for (var i = 0; i < menuItems.length; i++) {
          var name = menuItems[i].name;
          var $li = $("<li keyName=" + menuItems[i].keyName + "
><a tabindex='-1' href='#' style='padding:3px 10px'>" + name + "</a></li>");
          if (menuItems[i].disabled == true)
            $li.addClass("disabled");
          $("#context-menu ul").append($li);
        }
        return true;
      } else {
      }
    },
    scopes: className,
    onItem: function (context, e) {
      // execute on menu item selection
      var key = $(e.currentTarget).attr("keyName");
      self.ItemClicked.raise(key, context);
    }
  });

  this.AddNewItems = function (menuItems, enableDefaultItems) {
    if (enableDefaultItems == false) {
      this.Items = new RadiantQ.Gantt.Dictionary();
      for (var i = 0; i < menuItems.length; i++)
        this.Items.Add(menuItems[i].keyName, menuItems[i]);
    }
    else {
      for (var i = 0; i < menuItems.length; i++)
        this.Items.Add(menuItems[i].keyName, menuItems[i]);
    }
  }
  return this;
}

<div id="context-menu">
  <ul class="dropdown-menu custom-dropdown-menu" role="menu">
  </ul>
</div>

```

b) Bootstrap datetime picker and spinner

We now use a 3rd party bootstrap [datetime picker](#) and [spinner](#) for editing the table StartTime, EndTime and Progress columns in the Gantt table.

Note that we have also included the appropriate JS and CSS files required for these plugins in our sample.

c) Bootstrap navigation bar html with buttons to interact with Gantt

```

<!-- /.navbar-start -->
<div style="height:45px">
  <nav class="navbar navbar-default navbar-fixed-top" role="navigation" style
="min-height:40px; height:40px">
    <div class="container" style="width:auto">
      <!-- Brand and toggle get grouped for better mobile display -->
      <div class="navbar-header">
        <button type="button" class="navbar-toggle" data-toggle="collapse"
data-target="#bs-example-navbar-collapse-1">
          <span class="sr-only">Toggle navigation</span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
      </div>
      <!-- Collect the nav links, forms, and other content for toggling -->
      <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
        <ul class="nav navbar-nav">
          <li title="Add" id="addRow">
            <a href="#" class="btn btn-default btn-custom-default
btn-sm"><span class="glyphicon custom-glyphicon glyphicon-plus"></span></a>
          </li>

          <li id="InsertNewItemAsSiblingBelow" title="Insert NewItem As
SiblingBelow">
            <a href="#" class="btn btn-default btn-custom-default
btn-sm"><span class="glyphicon custom-glyphicon glyphicon-th"></span></a>
          </li>
          <li id="InsertNewItemAsChildOf" title="Insert NewItem As
ChildOf">
            <a href="#" class="btn btn-default btn-custom-default
btn-sm"><span class="glyphicon custom-glyphicon glyphicon-th-list"></span></a>
          </li>

          <li title="delete" id="remove">
            <a href="#" class="btn btn-default btn-custom-default
btn-sm"><span class="glyphicon custom-glyphicon glyphicon-remove"></span></a>
          </li>
          <li title="Move up" id="moveup">
            <a href="#" class="btn btn-default btn-custom-default
btn-sm"><span class="glyphicon custom-glyphicon glyphicon-open"></span></a>
          </li>
          <li title="Move down" id="movedown">
            <a href="#" class="btn btn-default btn-custom-default
btn-sm"><span class="glyphicon custom-glyphicon glyphicon-save"></span></a>
          </li>
          <li title="Indent" id="indent">
            <a href="#" class="btn btn-default btn-custom-default
btn-sm"><span class="glyphicon custom-glyphicon glyphicon-arrow-right"></span></a>
          </li>
          <li title="Outdent" id="outdent">
            <a href="#" class="btn btn-default btn-custom-default
btn-sm"><span class="glyphicon custom-glyphicon glyphicon-arrow-left"></span></a>
          </li>
        </ul>
      </div>
      <!-- /.navbar-collapse -->
    </div>
  <!-- /.container -->
</nav>
</div>
<!-- /.navbar-End -->

```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

How Tos

RadiantQ jQuery Gantt Package

How to bring a task into view in the gantt chart?

Sometime we need a situation to bring the task/time in to view using a button click or some DOM events, to do that RadiantQ Gantt provide a method called BringTimeIntoView in GanttChart this method is common for both Project and FlexyGantt.

Here is the some example code.

```
//To get hold of GanttControl. For flexyGantt get the pass the "FlexyGantt" as data
argument.
var gantt = $gantt_container.data("GanttControl");
//To get the Ganttchart instance.
var ganttChart = gantt.GetGanttChart().data("GanttChart");
//Task StartTime which we want to bring into view (i.e this is an example)
var taskstart = gantt.options.DataSource[0].StartTime;
//To bring the task into view using it's start time.
ganttChart.BringTimeIntoView(taskstart);
```

RadiantQ jQuery Gantt Package

How to vertically scroll and bring a task into View?

RadiantQ Gantt lets you bring a Gantt Row into view knowing the underlying bound data.

Example code for Project Gantt.

You can do so knowing the activity's ID:

```
//To get hold of the GanttControl instance by its element.
var ganttControl = $gantt_container.data("GanttControl")
// To get Activity view (i.e example)
//Each row in Project Gantt bound to an ActivityView.
var activityView = ganttControl.ActivityViews[30];
//Scroll by Activity view.
ganttControl.ScrollIntoView(activityView);
//select the activity view.
ganttControl.SelectedItem = activityView;
```

Example code for FlexyGantt.

First get hold of the FlexyNodeData instance as follows:

```
//To get hold of the flexy gantt instance.
var flexyGantt = $gantt_container.data("FlexyGantt");
//To get the FlexyNode data, using it's index.
//Each row in FlexyGantt bound to an FlexyNode.
var flexyNodeData = flexyGantt.FlatItemsSource.FlatItemsSource[22];
//Scroll to the FlexyNodeData.
flexyGantt.ScrollIntoView(flexyNodeData);
//select the FlexyNodeData.
flexyGantt.SelectedItem = flexyNodeData;
```


RadiantQ jQuery Gantt Package

How to pass a argument to server in MVC Wrapper?

Sometimes we have a situation to pass an arguments to sever, based on the arguments we have to retrieve data from server in this case, we can use the `DataSourceUrlArguments` property.

Here is the code example.

```
<%= Html.JQFlexyGantt(  
    new JQFlexyGanttSettings()  
    {  
        ControlId = "gantt_container",  
        DataSourceUrlArguments = "{total_tasks:window.taskCount}",  
        DataSourceUrl = new Uri("/Home/GetPerformanceTestSampleSource", UriKind  
.RelativeOrAbsolute),  
        Options = new FlexyGanttOptions()  
        {  
            ....  
        }  
    }  
)%>
```

to kown more about this take a look at the `..\MVCWrapperDemo\Views\Home\PerformanceTestSample.aspx` sample.

RadiantQ jQuery Gantt Package

How to pass some AjaxSettings to ajax in MVC?

In MVC wrapper all the sever calls are initiated from our soruce, so if you need to pass some options to ajax use the AjaxSettings property.

```

<script type="text/javascript">

    var AjaxSettings =
    {
        dataType: 'xml text',
        converters:
        {
            "xml text": function (data) {
                console.log(data);
                // We use te xml2json jquery plugin to convert xml to json.
                var json = $.xml2json(data).XMLTaskInfo;

                return $.parseJSON(window.JSON.stringify(json), true
                /*converts date strings to date objects*/
                , true
                /*converts ISO dates to local dates*/
                );
            }
        }
    };

</script>

@Html.JQProjectGantt(
    new JQProjectGanttSettings()
    {
        ControlId = "gantt_container",
        DataSourceUrl = new Uri("/Home/GetProjectGanttXMLItemsource", UriKind
.RelativeOrAbsolute),
        AjaxSettings="AjaxSettings" ,
        Options = new ProjectGanttOptions()
        {
            AnchorTime = DateTime.Today,
            BaseTimeUnitWidth = 50,
            RowHeight = 25,
            WorkTimeSchedule = null,
            FlatActivitiesListCreated = "FlatActivitiesListCreated",
        }
    })

```

RadiantQ jQuery Gantt Package

How to improve the performance of the Gantt while setting multiple options?

Changing some settings in the gantt could trigger expensive redraws of the gantt. In such cases it would be good if you can temporarily block redrawing the gantt until you have updated all the settings.

You can do so by sandwiching your settings code within the BeginUpdate and EndUpdate methods. This will prevent GanttChart from redrawing until the EndUpdate method is called.

Code Example:

```
$Gantt.FlexyGantt("BeginUpdate");  
  
$GanttChart.GanttChart('AnchorTime', anchorTime);//To change the anchortime.  
$GanttChart.GanttChart('BaseTimeUnitWidth', baseTimeUnitWidth);//To change the  
baseTimeUnitWidth.  
$GanttCart.GanttChart('ResizeToFit', resizeToFit);//To change the resizeToFit.  
  
$Gantt.FlexyGantt("EndUpdate");
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

How to send exported image to server?

Using the `AfterCanvasGenerated` callback, the canvas that contains the gantt can be read. Then the canvas can be converted to image data using `"canvas.toDataURL()"` method. You can send the obtained image data to server using an ajax call like below.

Here is the code example:

```
function afterCanvasGeneratedCallback(canvas) {
    canvasDataURL = canvas.toDataURL("image/jpeg");

    // canvasDataURL holds the url of the image. Instead of alert you can do ajax
    // call to pass the image to server
    alert(canvasDataURL);

    // prevents the default behaviour of export to image feature.
    return false;
}
Gantt.ExportToImage({ AfterCanvasGenerated: afterCanvasGeneratedCallback });
```

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

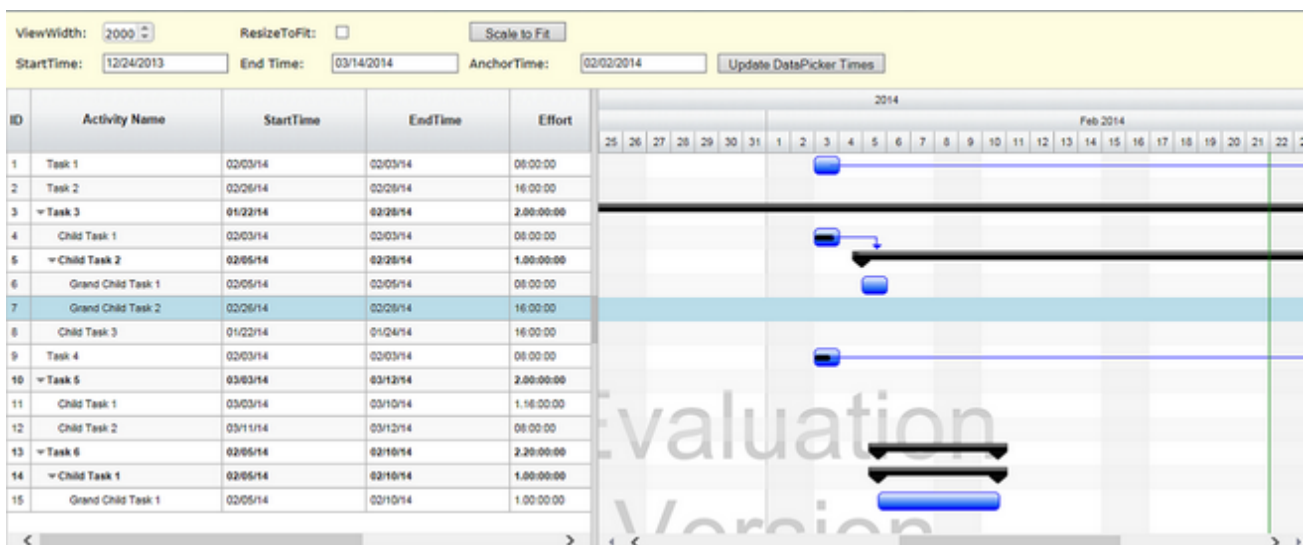
How to adjust the chart's zoom level to make it show all the tasks in the project?

This topic shows how to change the zoom level of the GanttChart to show all the tasks in view.

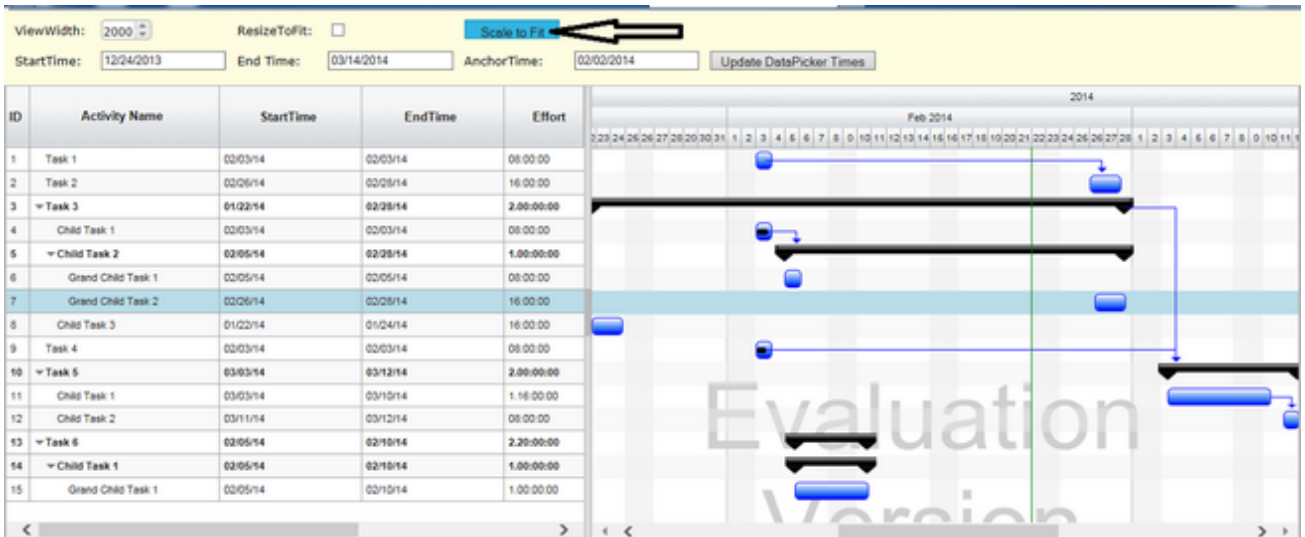
Here is the code.

```
//Gantt Chart Instance
var ganttChartInstance = $GanttChart.data("GanttChart");
//Earlier Task Start
var earliestTaskStart = ganttControl.Model.Activities.StartTime;
//later Task Start
var latestTaskEnd = ganttControl.Model.Activities.EndTime.addDays(1);
ganttChartInstance.SetStartTime(earliestTaskStart);
ganttChartInstance.TrySetEndTime(latestTaskEnd);
```

Here is the default gantt with hiding three or more tasks in view:



Here is the resultant gantt with showing all the tasks in view:



This is illustrated in this samples:

- In HTML : ..\Samples\TimeScaleStartAndEnd.htm.
- In ASP.NET MVC : ..\Views\Home\Common\TimeScaleStartAndEnd.cshtml.
- In ASP.NET : ..\Samples\Common\TimeScaleStartAndEnd.aspx.

RadiantQ jQuery Gantt Package

How to customize the React Gantt Component ?

a) How to populate the own datasource ?

Instead of passing Url as a props, you can also populate your own data by passing `DataSource` as a props of `GanttControl`.

If you want to populate data with AJAX calls, then you should define "async :false" on the ajax request. The code below illustrates how the `GanttcontrolSkeleton` component calls the `GanttControl` component with `DataSource` as the props .

```
class GanttcontrolSkeleton extends React.Component{
  render(){
    return(
      <div >
        <GanttControl
          // Set DataSource option as props.
          DataSource= {this.data}/* your data */
        />
      </div>
    );
  }
}
```

b) How to define customize columns in React Component ?

You can also customize the columns data and it should set as "columns" option in `GanttControl` component. The below code will explain how to do this.

```
class GanttcontrolSkeleton extends React.Component {
  render() {
    return (
      <div>
        <GanttControl
          // Set columns option as props.
          columns = {this.columns} /* your column data */
        />
      </div>
    );
  }
}
```

RadiantQ jQuery Gantt Package

How to communicate with gantt component from your own component in React?

Sometimes, one component may be depended on any other component. The communication between the components can possible by passing property. Here is the code that shows how to do that,


```

class ButtonContainer extends React.Component {
  constructor(props) {
    super(props);
    this.ganttControl = this.props.ganttControl;
  }

  // componentWillReceiveProps will updates the newly received ganttcontrol
  props.
  componentWillReceiveProps(nextProps) {
    if (nextProps.ganttControl !== this.ganttControl) {
      if (this.ganttControl !== null)
        this.ganttControl = nextProps.ganttControl;
    }
  }

  // To add a task.
  addRow() {
    var newTask = this.getNewTask();
    this.ganttControl.AddNewItem(newTask);
    //Add the newly added task to Dictionary.
    this.addedTasks.Add(newTask.ID, newTask);
  }

  // To return a new task object.
  getNewTask() {
    return {
      "Name": "New Task ",
      "ID": this.ganttControl.Model.GetNewID(),
      "StartTime": new Date("2014/02/02"),
      "Effort": new RQTimeSpan(0, 12, 30, 0, 0),
      "ProgressPercent": 50,
      "Description": "Description of Task"
    };
  }

  render() {
    return (
      <div id="Div1">
        <button id="addRow" onClick={() => this.addRow()}>Add task</button>
      </div>
    );
  }
}

```

```

class GanttControlCustomDataBinding extends React.Component {
  constructor() {
    super();
    this.state = { ganttControl: null }
  }

  // ganttControl data rerendered once the GanttLoaded event was triggered.
  GanttLoaded($gantt_container,ganttControl) {
    this.setState({ ganttControl: ganttControl })
  }

  // Here, GanttControlCustomDataBinding renders multiple Component .
  render() {
    return (
      <div>
        <ButtonContainer ganttControl={this.state.ganttControl}/>
        <GanttControl
          .....
          .....
          GanttLoaded={(ganttControl) => this.GanttLoaded(ganttControl)}
        />
      </div>
    );
  }
}

```

Note: By default ganttControl state value is null and it should be available after the Gantt is loaded. So, updating the state value in "GanttControlCustomDataBinding" component by setState method and this should be listened in your component by using the "componentWillReceiveProps" lifecycle method.

This is illustrated in this samples:

In React :
.\Samples\GanttControlCustomDataBinding\GanttControlCustomDataBinding.jsx.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

How to render the multiple react gantt component in a single page?

Import the react libraries and the "GanttControl and FlexyGantt" component from its desired path and render both components into the MultiGantt Component like below,

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import GanttControl from '../RQ_Components/GanttControl.jsx';
import FlexyGantt from '../RQ_Components/FlexyGantt.jsx';

// To get the name from the bounded list .
// Converter should be define globally using window
function initConverters() {
    window.nameConverter = function (flexyNodeData, value) {
        var data;
        // The grid calls this converter with flexyNodeData as a arg.
        if (flexyNodeData.Data)
            data = flexyNodeData.Data();
        // The grid calls this converter with flexyNodeData as a datacontext.
        else
            data = flexyNodeData;
        if (value == undefined) {
            if (data.Resource && data.Resource.ResourceName_M != undefined)
                return data.Resource.ResourceName_M();
            else
                return data["ActivityName"];
        }
        else {
            if (data.Resource && data.Resource.ResourceName_M != undefined)
                data.Resource.ResourceName_M(value);
            else
                data["ActivityName"] = value;
        }
        return;
    };

    // to get the short time format.
    var tooltipDateformat = Date.CultureInfo.formatPatterns.shortDate + ' ' +
    "HH:mm:ss";
    window.startTimeTooltipConverter = function (data) {
        if (data["PStartTime"])
            return data["PStartTime"].toString(tooltipDateformat);
        else if (data["StartTime"])
            return data["StartTime"].toString(tooltipDateformat);

        return null;
    }
    window.endTimeTooltipConverter = function (data) {
        if (data["PEndTime"])
            return data["PEndTime"].toString(tooltipDateformat);
        else if (data["EndTime"])
            return data["EndTime"].toString(tooltipDateformat);

        return null;
    }
}

// This samples renders the multiple react components in single page.
class MultiGantt extends React.Component {
    constructor() {
        super();
        this.anchorTime = new Date("2014-02-01T00:00:00-00:00");
        initConverters();
        this.columns = [{
            field: "Name",
            title: "Name",
            editor: RadiantQ.Default.Template.FlexyGanttExpandableTextBoxEditor(
            "nameConverter"),
            template:
            RadiantQ.Default.Template.FlexyGanttExpandableTextBlockTemplate("nameConverter")
        }
    ]};
}

```

This is illustrated in this samples:

In React : `.\Samples\ResourceLoadView\ResourceLoadView.jsx`

© RadiantQ 2009-2018. All Rights Reserved.

-0-

*RadiantQ jQuery Gantt Package***JSON Data**

JSON is most common data-interchange language, which can be easily parsed into a JavaScript object. JSON however doesn't define any standard way to represent the date time and time span format. The server code could choose to represent this in many different ways. In this topic we are going to describe how to parse such date-time and time-span strings into their corresponding JS objects.

RadiantQ has an extended version of the \$.parseJSON which is used to convert JSON to JS object and it takes the following arguments.

Arguments	Description
jsonString (string)	A well-formed JSON string which is converted into the returned Java Script object.
canConvterDefaultFormat (boolean)	To specify whether to convert the default date string into date object
ConvertTolocalTime (boolean)	if its true it converts GMT dates to local time.
DataConverter (function)	An optional function which if specified, is called for each string value in the json string, to be converted into a JS object.

Here is the example.

```
$.parseJSON(jsonString, true, true);
```

Default Date and TimeSpan formats supported

The default date string formats which the \$.parseJSON can handle are:

- 1) YYYY-MM-ddTHH:mm:ssZ
- 2) YYYY-MM-ddTHH:mm:ss
- 3) .NET Date (Date(1224043200000) / Date(-1224043200000))
- 4) date time with time Zone (Tue May 05 2015 13:14:17 GMT+0530 (India Standard Time))

The default time span formats supported are:

- 1) HH:mm:ss
- 2) dd.HH:mm:ss / d.HH:mm:ss

Here is an example JSON data.

```
[{
  "Name" : "Task 1",
  "ID" : 1,
  "StartTime" : "2014-02-02T00:00:00Z",
  "Cost" : "200",
  "Effort" : "8:00:00",
  "Description" : "Description of Task 1"
},
{
  "Name" : "Task 2",
  "ID" : 2,
  "Cost" : "300",
  "PredecessorIndices" : "1",
  "StartTime" : "2014-02-03T00:00:00Z",
  "Effort" : "1.20:00:00",
  "Description" : "Description of Task 2"
}]
```

Handling custom Date and TimeSpan strings

The 4th "data converter" argument can be used to convert a custom string format to a appropriate object.

Here is an example for that.

NOTE: Assume the file name is sampleTask.json, containing this data:

```
[{
  "Name" : "Task 1",
  "ID" : 1,
  "StartTime" : "\\Date(1414782000000+0500)\\/",
  "Cost" : "200",
  "Effort" : "8:00:00",
  "Description" : "Description of Task 1"
},
{
  "Name" : "Task 2",
  "ID" : 2,
  "Cost" : "300",
  "PredecessorIndices" : "1",
  "StartTime" : "\\Date(1409511600000+0500)\\/",
  "Effort" : "1.20:00:00",
  "Description" : "Description of Task 2"
}]
```

Call RadiantQ \$.parseJSON with this "data converter".

```
$.ajax({
  type: "GET",
  dataType: 'json',
  url: "sampleTask.json",
  ContentType: "application/json; charset=utf-8",
  converters:
  {
    "text json": function (data) {
      return $.parseJSON(data, true,
        true,
        //Data converter function,
        //key - is the name for the field.
        //value - field value.
        function (key, value) {

          if (key == "StartTime") {
            //Custom string format expression.
            var dateNet = /\Date\((([0-9]+)([+])([0-9]+))\)\//;

            if (dateNet.test(value)) {
              var regex = value.match(dateNet);
              var millisecs = parseInt(regex[1]);
              //Convert string into the date object.
              return new Date(millisecs);
            }
          }
        }
      );
    },
    success: function (data) {
      // data hold the js array of tasks.
    }
  }
});
```


Troubleshooting

RadiantQ jQuery Gantt Package

Why the gantt is not showing in the page and how to fix it?

Why the gantt is not showing in the page and how to fix it?

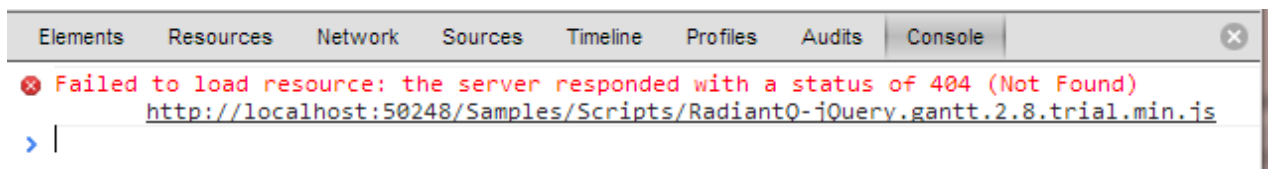
Here are a few scenarios that will lead to an empty page, without the gantt showing.

(Note that you can usually show a browser's console window by pressing "F12".)

a) Failed to load JS Script files

An empty page will be displayed and you get the "Failed to load resource: the server responded with status of 404 (Not Found)" error in browser console, which usually means that the reference to the js script files were incorrect.

The below image shows that "RadiantQ-jQuery.gantt.2.8.trial.min.js" file is not found.

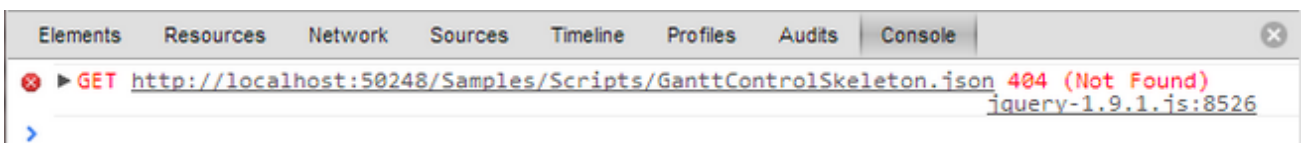


To fix this issue:

Simply make sure that the referenced path, "Samples/Scripts" is correct or make sure that the script files are, in fact, available in that path.

b) Failed to load data source (web service or JSON file)

The console log will contain the error message of missing web-service or json files.



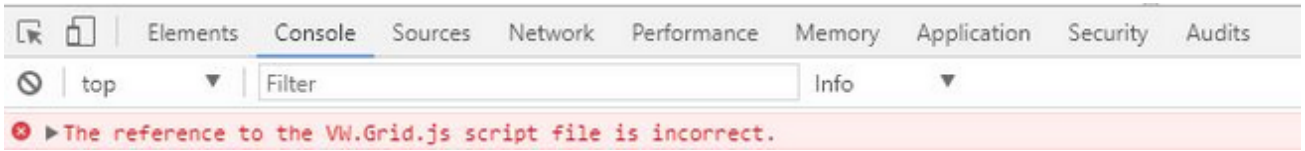
To fix this issue:

Make sure that the path to the web-service or json file referenced is correct.

Note : When you deploy a json file to the server (usually for testing purposes), servers, by default, do not serve such .json file types(no wildcard MIME type). Therefore a 404 not found is thrown when they are accessed. Take a look at [Deploying Gantt](#) topic (Note on json files) to see how to enable the server to serve json files.

c) Forgets to refer (or) wrongly referred "

Splitter only displayed and you get "The reference to the VW.Grid.js script file is incorrect" error in browser console, which usually means that the user forgets to refer (or) wrongly referred "VW.Grid.js".



To fix this issue:

Simply make sure that the "VW.Grid.js" file referred properly.

© RadiantQ 2009-2018. All Rights Reserved.

-0-

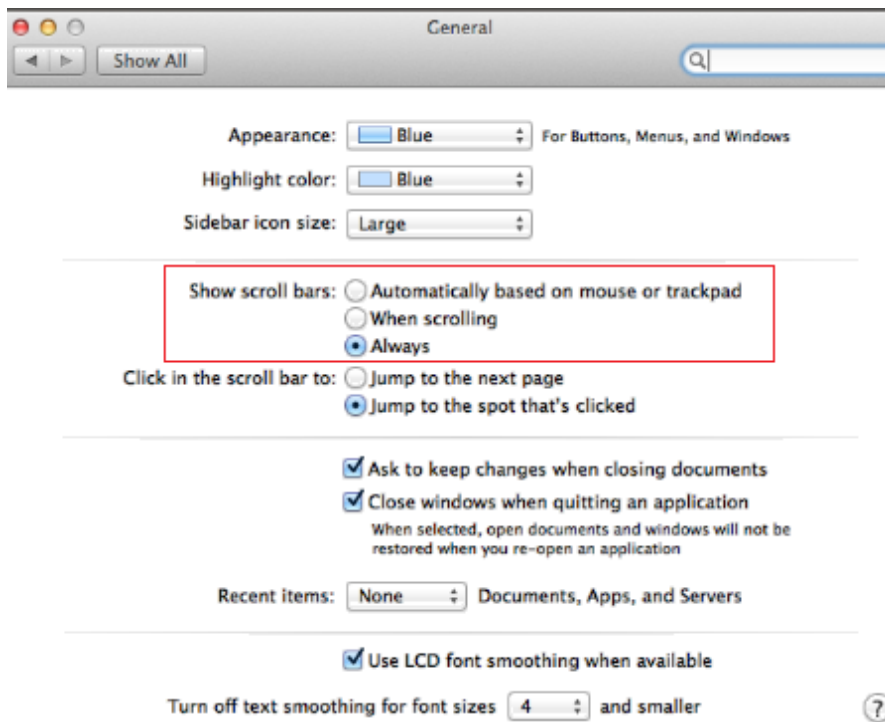
RadiantQ jQuery Gantt Package

How to enable scroll bars for Mac OS ?

This topic shows how to enable/disable scroll bars in Mac OS for any browser.

By default, the scroll bars are not enabled in Mac. You can enable the scroll bars in System Preferences by following the below steps,

- Click the apple menu at the top-left of the screen, then select "System Preferences".
- Click on "General" section and choose the required option under "Show scroll bars" heading.



© RadiantQ 2009-2018. All Rights Reserved.

-0-

RadiantQ jQuery Gantt Package

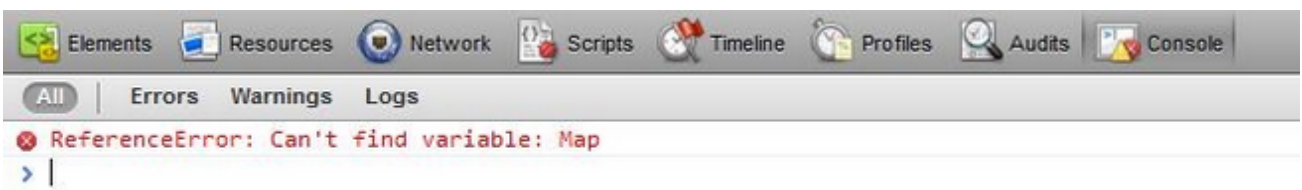
Why React gantt application renders a blank page and how to fix it ?

Why React gantt application renders a blank page and how to fix it ?

Here, the below possible solution might help you if you encounter the problem like "ReferenceError: Can't find variable: Map" error in the browser console window by pressing "F12" .

React 16 (or) React Fiber doesn't support Older Browser.

An blank page will be displayed and you get the "ReferenceError: Can't find variable: Map" error in browser console, which means that the older browser natively not supported React Fiber features of "Map" and "Set" .



To fix this issue:

To support older browser, you need to include a "babel-polyfill" in the entry point of the bundled application. Because, React Fiber depends on the collection types "Map" and "Set" .

- Adding "babel-polyfill" in the entry array of webpack.config.js.

```
module.exports = {  
  entry: ["babel-polyfill", "./app.js"]  
};
```

(OR)

- Otherwise, you should import "babel-polyfill" at the top of entry file like this (app.js).

```
import 'babel-polyfill';
```

© RadiantQ 2009-2018. All Rights Reserved.

Index

- -

How to enable "Undo/Redo" feature for custom columns ? 226

- ! -

'TimeScaleHeaders' 7

- **A** -

About Samples 10
About Task Field Types 121
Activities or Tasks 82
add/remove 7
Adding Undo actions Programmatically 118
Angular Samples 12
Assignments VS Task Duration 108

- **B** -

Backbone.JS 452
Binding to Resource Objects 99
Binding to Resource Strings 97
Bootstrap 456
Browse To Task Cues 421

- **C** -

Calendar and CalendarWithExceptions 94
Calendars
Chart Look and Feel 403
Cleaning Up Src Folder 373
Common Topics 372
ContextMenus 180, 343, 393
Creating Custom Undo Actions 120
Creating Gradient Backgrounds dynamically in code 401
Critical Paths 110
Custom Chart Background
Custom Look and Feel 184
Custom Look for Specific Task Bars 186
Custom Time Scale Headers 412

- **D** -

Data Binding
Data Binding in PHP 142
Default Time Scale Header 404
Dependency Lines 304

Deploying Gantt 25
Disabling Runtime Features 160

- E -

Editing Tasks 296
Enable edit mode in single click 174
Enable GanttTable startEdit and endEdit options 176
Enable HeaderMenu 175
Enable Row Drag and Drop 171, 346
Enabling Undo/Redo 116
End-User Time Scale Header Operations 417
EndTime field with Effort field 141
EndTime field without Effort field 139
EndTime in DataSource 139
Entire HTML Samples 28
Export 447
External Dependencies 31

- F -

Feedback
FlexyGantt 7, 7
FlexyTable Editing 302

- G -

Gantt Chart AnchorTime 419
Gantt CSS Styles 379, 179
Gantt Customization using Color Palette
Gantt Look and Feel 341
Gantt Table Customization 203
Gantt Table Customization 203
GanttChart Events 425
GanttChart View Width 420
GanttControl 7
GanttTable Alternative Row background 207
GanttTable Column Editable Settings 205
GanttTable Custom Column 203
GanttTable Editing 75
Getting Started 373
Global ReadOnly 160
Global Selective ReadOnly settings 163

- H -

Header Text Format and Header Height Customization 406
How to add a new resource in resource dataset without reloading Gantt? 229
How to add custom resource dropdown for custom column? 234
How to add data asynchronously while on expanding the node? 366
How to add new task using the context menu? 220
How to add resource to an activity programmatically 225
How to add tooltip for a custom element ? 360
How to adjust the chart's zoom level to make it show all the tasks in the project? 469
How to bring a task into view in the gantt chart? 463
How to communicate with gantt component from your own component in React? 472
How to Convert X to Time and Time to X in the Chart? 351
How to customize the React Gantt Component? 471
How to enable paging while autoscrolling in GanttChart ? 236

How to enable scroll bars for Mac OS ? 483
How to Expand all/Collapse all summary tasks dynamically? 218
How to export current Gantt data to JSON? 223
How to find a task/activity by it's ID? 208
How to find ActivityView by its ID? 209
How to get all dependencies of an activity? 217
How to get hold of bound activity in row click? 219
How to improve the performance of the Gantt while setting multiple options? 467
How to insert/remove items dynamically? 368
How to listen the Activity CollectionChanges in GanttControl? 224
How to listen the node Expand/Collapse state in FlexyGantt? 365
How to listen the taskbar click event in FlexyGantt? 354
How to listen the taskbar size changes in FlexyGantt? 353
How to listen to changes made by the end-user on the tasks?
How to maintain the current state of the Expand/Collapse nodes after data source reset? 362
How to make Custom Progress Calculation? 233
How to make reflect the bound data property changes into the gantt(view)? 363
How to make selection and hover effect? 232
How to notify when dependency connection is failed? 231
How to pass a argument to server in MVC Wrapper? 465
How to pass some AjaxSettings to ajax in MVC? 466
How to prevent Taskbar Overlapping? 357
How to refresh Gantt with new data with new schema 222
How to refresh Gantt with new data with same schema 352, 221
How to remove a dependency? 216
How to render the multiple react gantt component in single page? 475
How to save Gantt options in cookies? 230
How to send exported image to server? 468
How to set the overlapping feature? 369
How to show different taskbars in single row? 355
How to vertically scroll and bring a task into View? 464
How to visually select a row given it's task id? 215
How Tos 463
HTML Samples 11

- I -

IE 8 Support 33
image export 7
In Angular 42, 244
In Angular 42, 244
In ASP.NET 254, 50
In ASP.NET MVC 56, 262
In ASP.NET MVC 56, 262
In HTML 238, 35
In PHP 63, 269
In React 48, 251
In TypeScript 68, 275
Installation 9

- J -

JSON Data 478

- K -

Keeping Dependant tasks "sticky" 86
knockout 31

- L -

Localized Strings 394

- M -

MS Project Export/Import 150
Multi Column Tree Grid 298
MVC Samples 21

- O -

Overlapped Tasks Look 317
Overriding Dependency Setup Behavior 178

- P -

parseJSON.js 31
Performance Optimization Options 307
Persist Changes Immediately 144
Persist Changes in Bulk 147
Persisting Changes 132
Persisting Changes Immediately 132
Persisting Changes in Bulk 136
PHP Samples 23
Print Settings in XML 7
Printing 429

- R -

RadiantQ.Gantt.TimeScaleHeaderDefinitions 403
RefreshRowBackground 335
RefreshRowForeground 338
RequireJS 449
Resource Level Schedules 152
Resource Leveling 153
Resource Load View 426, 157
Resource Specific Calendars 103
Round To Options 377
Row Background 340
Row Specific Schedule for Rendering 314
Runtime Interaction 79

- S -

Schedule for Rendering 312
Setting up the GanttTable 74
Single HTML Sample 26
Special Lines 383

- T -

Task Bar Labels 194
Task Bar Look and Feel 184

Task Bar Redraw 197
Task Bar Vertical Dragging 349
Task Labels 326
Task Level Schedules 154
Task Popup 199
Task Popups 332
Task specific ReadOnly settings 170
Task Template 292
TaskBarBackgroundTemplate 190
Tasks Filtering 158
Themes 396
Time Scale Header 303, 96
Time Scale Header customization 403, 7, 303
Time Scale Header Customization 403, 7, 303
Time Span Highlighting 390
TimeScaleHeaderDefinition 7
TimeSpan Paging 423
TypeScript Samples 24

- U -

Using Knockout 130, 290
Using RadiantQ Binding 286, 126

- V -

Virtualization In Gantt 375

- W -

WBS Support 112
WebForms Samples 22
What are the different ways to improve the performance of the gantt with a large set of tasks? 213
Why React gantt application renders a blank page and how to fix it ? 484
Why the gantt is not showing in the page and how to fix it? 481
WorkTimeSchedule 89

- X -

XML Data 122, 281

- Z -

Zooming Programmatically 411

