CUBA Platform. Developer’s Manual

Preface

This manual provides the reference information for the CUBA platform and covers the most important topics of developing business applications with it.

Intended Audience

This manual is intended for application developers using the CUBA platform. The following technologies knowledge is required to use the platform:

- Java Standard Edition
- Relational databases (SQL, DDL)

Further Reading

This manual and other documentation related to the CUBA platform can be found at www.cuba-platform.com/manual.

Video materials and presentations that can help you to understand the platform are available at www.cuba-platform.com/tutorials.

You can also check out online demo applications at www.cuba-platform.com/online-demo.

Knowledge of the following technologies and frameworks will be helpful to get a deeper understanding of the platform:

- Gradle build system
- Spring Framework
- Java Persistence API
- EclipseLink framework
- Vaadin web applications framework
- HTML / CSS / JavaScript
- Java Servlets
- Java Swing

Feedback

If you have any suggestions for improvement of this Manual, please contact support at www.cuba-platform.com/support/topics.

If you find a mistake in the documentation, please specify the number of the chapter and attach a small portion of the surrounding text to facilitate the search.

1. Introduction

This chapter provides information about the CUBA platform features and requirements.

1.1. Overview

CUBA platform is an ideal tool for development teams working on line-of-business applications, typically having extensive data model, hundreds of screens and complex business logic.

Based on a mainstream technology stack, CUBA platform brings unparalleled productivity by utilizing a rich set of ready to use data-aware components, extensive scaffolding, visual interface designer and hot deploy.

Open architecture allows a developer to customize any part of the framework, providing high levels of control and flexibility. Developers have the freedom to use popular Java IDEs and have full access to the source code.

CUBA applications fit seamlessly into the corporate IT environment, supporting major databases and application servers, as well as popular aPaaS clouds. Streamlined clustered deployment ensures scalability and failover, while a generic REST API enables easy integration with other systems.

1.2. Technical Requirements

Minimum requirements for development using CUBA platform:

- Memory – 4 GB
- Hard drive space – 5 GB
- Operating system: **Microsoft Windows**, **Linux** or **Mac OS X**

1.3. Release Notes

CUBA platform changelog is available at http://files.cuba-platform.com/cuba/platform/platform-6.0-changelog.html.

2. Installation and Setup

Minimum software requirements are as follows:

- **Java SE Development Kit (JDK) 8**. It is recommended that you use **Oracle Java HotSpot VM**.

  In order to build and run projects outside Studio, you need to set the path to the JDK root directory in the `JAVA_HOME` environment variable, e.g. `C:\Program Files\Java\jdk1.8.0_60`. On Windows, you can do this at **Computer → Properties → Advanced System Settings → Advanced → Environment variables.** The value of the variable should be added to the **System variables** list.

- Java IDE: **IntelliJ IDEA Community Edition 13+** or **Eclipse 4.3+**. We recommend using **IntelliJ IDEA**.

In the most basic scenario, the built-in **HyperSQL** (http://hsqldb.org) can be used as the database server. This is sufficient for exploring the platform capabilities and application prototyping. For building production applications, it is recommended to install and use one of the full-featured DBMS supported by the platform, like **PostgreSQL** for instance.

The web interface of the platform-based applications supports all popular browsers, including **Google Chrome**, **Mozilla Firefox**, **Safari**, **Opera 15+**, **Internet Explorer 9+**, **Microsoft Edge**.

2.1. CUBA Studio Installation

Prerequisites:

- Make sure that Java SE Development Kit (JDK) 8 is installed by running the following command in the console:

  ```
  java -version
  ```

  The command should return the Java version, e.g. `1.8.0_60`.

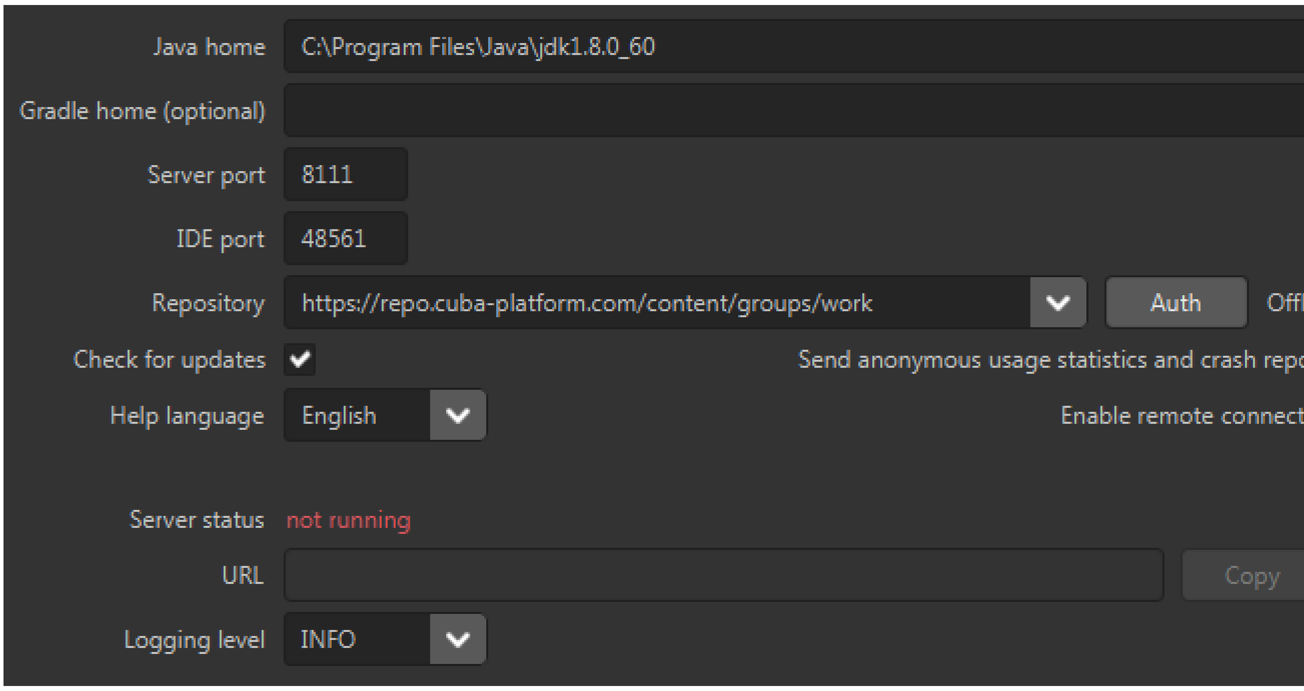
- If you connect to the internet via a proxy server, some Java system properties must be passed to the JVM running Studio and Gradle. These properties are explained here: http://docs.oracle.com/javase/8/docs/technotes/guides/net/proxies.html (see properties for HTTP and HTTPS protocols).

  It is recommended to set these properties system-wide in the `JAVA_OPTS` environment variable. The Studio launch script passes `JAVA_OPTS` to the Java executable.

In order to install CUBA Studio, take the following steps:

1. Download an appropriate installer or ZIP archive from https://www.cuba-platform.com/download.
2. Run the installer or unzip the archive to a local directory, e.g. `c:/work/studio`
3. Launch the installed application or open the command line, go to `bin` directory and run `studio`
4. In the **CUBA Studio Server** window, enter the following parameters:
   - **Java home** − JDK installation to be used for building and running projects. If you have set the `JAVA_HOME` environment variable as described in the beginning of this chapter, it will appear in this field. Otherwise, Studio will try to find your Java installation itself.
   - **Gradle home** − leave this field empty; in this case, the required **Gradle** distribution will be downloaded automatically.

     If you want to use a local Gradle distribution, enter the path to the respective directory in the field. For project build system to work correctly, **Gradle 2.6** is required.
   - **Server port** − CUBA Studio server port (the default port is 8111).
   - **IDE port** − IDE plugin listening port (the default port is 48561).
   - **Repository** − binary artifacts repository URL and authentication parameters.


Java home C:\Program Files\Java\jdk1.8.0_60
Gradle home (optional)
Server port 8111
IDE port 48561
Repository https://repo.cuba-platform.com/content/groups/work
Check for updates ✓
Send anonymous usage statistics and crash repo
Help language English
Enable remote connect
Server status not running
URL
Logging level INFO

5. The following options are also available:
   - **Check for updates -** check for new versions on every start.
   - **Help language -** built-in help language.
   - **Offline -** enable working with projects without an Internet connection, provided that all the required libraries have been previously downloaded from the repository.
   - **Send anonymous statistics and crash reports -** enable Studio to send error statistics to developers.
   - **Enable remote connection -** by default, it is assumed that Studio runs on localhost. Select this checkbox if you need to connect to this Studio instance from a remote host.
6. Click **Start** to run the Studio server.

   The server will download, run, and connect to the Gradle daemon. This may take a significant amount of time on first startup; on subsequent launches, this will take just a few seconds.

   After that, the web server will be started, and the URL of the Studio interface will appear in the **URL** field. By clicking →, you can open the address in your default web browser; by clicking **Copy** you can copy the address to the clipboard.

7. Open the specified address in the web browser.
8. In the Studio web interface, click **Create new** to create a new project, or **Import** to add an existing one to the **Recent** list.
9. Once the project is opened, the Studio will download the source code of the platform base projects and save it to the local folder. Before building the project, it is recommended to wait until the download is finished and make sure that the background task indicator in the bottom left corner has faded out.

2.2. IDE Integration

Take the following steps to integrate Studio with **IntelliJ IDEA** or **Eclipse**:

1. Open or create a new project in the Studio.
2. Switch to **Project properties** section and click **Edit**. Select the required **Java IDE** by checking **IntelliJ IDEA** or **Eclipse**.
3. Select **Build** > **Create or update <IDE> project files** in the Studio menu. The corresponding files will be created in the project directory.
4. For IntelliJ IDEA integration:
    a. Run IntelliJ IDEA 13+ and install **CUBA Framework Integration** plugin, from the plugin repository: **File > Settings > Plugins > Browse Repositories**.
5. For Eclipse integration:
    a. Run Eclipse 4.3+, open **Help > Install New Software**, add `http://files.cuba-platform.com/eclipse-update-site` repository and install the **CUBA Plugin**.
    b. In the **CUBA** section of the **Window > Preferences** menu, check **Studio Integration Enabled**, and click **OK**.

Please note that **IDE: on port 48561** label has appeared in the bottom left corner of the Studio. Now the corresponding source code files will be opened in IDE when you click **IDE** buttons in the Studio.

3. Quick Start

This section describes the process of creating an application using **CUBA Studio**. Similar information is provided in the videos available at www.cuba-platform.com/quickstart.

Make sure that the necessary software is already installed and set up on your computer, see Installation and Setup.

Key stages of our application development:

1. Data model development including creation of entities describing application domain and corresponding database tables.
2. Development of the user interface screens enabling to create, view, update and delete data model entities.

3.1. Application Details

The application should maintain information about the customers and their orders.

A customer has the following attributes:

- Name
- E-mail

Order attributes:

- Ownership by a customer
- Date
- Amount

The application UI should contain:

- Customers browser screen;
- Customer editor screen, containing as well the list of this customer's orders;
- General orders browser screen;
- Order editor screen.

The application should support user interface in English and Russian.

3.2. Creating a Project

1. Start CUBA Studio and open its web interface (See CUBA Studio Installation).
2. Click **Create new**.
3. Specify the name of the new project in the **Project name** field of the **New project** window – for example, `sales`. The name should contain only Latin letters, numbers and underscores. Think carefully on the project name at this stage, as changing it later on will require complex manual intervention.
4. The following fields below will be automatically populated:
   - **Project path** – the path to the new project directory. You can select the directory manually by clicking the **…** button next to the field. The **Select folder** window will appear with the list of folders n your hard drive. You can select one of those, or create a new directory by clicking the **+** button.
   - **Project namespace** – the namespace which will be used as a prefix for entity names and database tables. The namespace can consist of Latin letters only and should be as short as possible. For example, if the project name is `sales_2`, the namespace can be `sales` or `sal`.
   - **Root package** − the root package of Java classes. It can be adjusted later, but the classes generated at project creation will not be moved.
   - **Base projects version** – the platform version used in the project. The platform artifacts will be automatically downloaded from the repository on project build.
5. Click **OK**. Empty project will be created in the specified `sales` directory and the main Studio window will open.
6. Assemble the project: select option **Build** > **Assemble project** in the Studio main menu. At this stage all required libraries will be downloaded and project artifacts will be assembled in `build` subdirectories of the modules.
7. Create the database on the local **HyperSQL** server: select option **Run** > **Create database** in the menu. The database name is the same as project namespace by default.
8. Select **Run** > **Deploy** menu option. **Tomcat** server with the deployed application will be installed in the project `build` subdirectory.
9. Select **Run** > **Start application server** option. The link next to the **Web application** caption in the status panel will become available in a few seconds so you will be able to open the application directly from Studio.

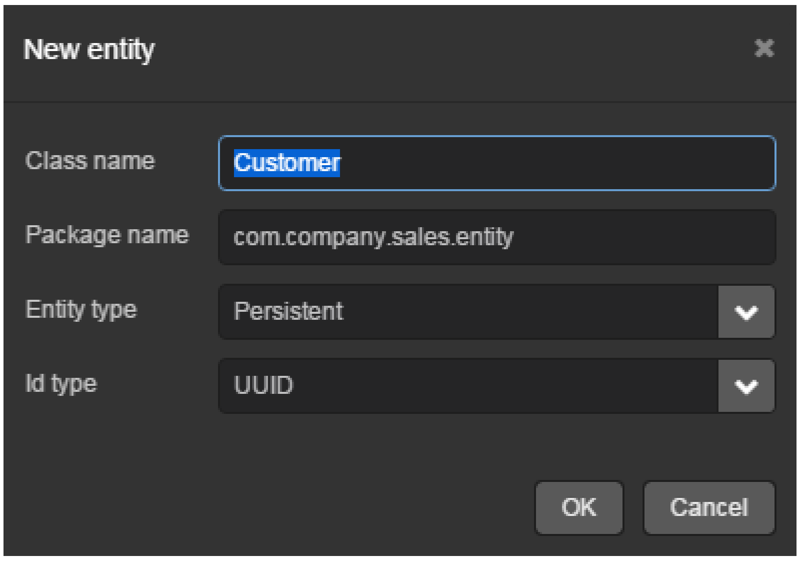
   The username and password are `admin` / `admin`.

   The running application contains two main menu items (**Administration** and **Help**), as well as security and administration subsystems functionality.

3.3. Creating Entities

Let us create the `Customer` entity class.

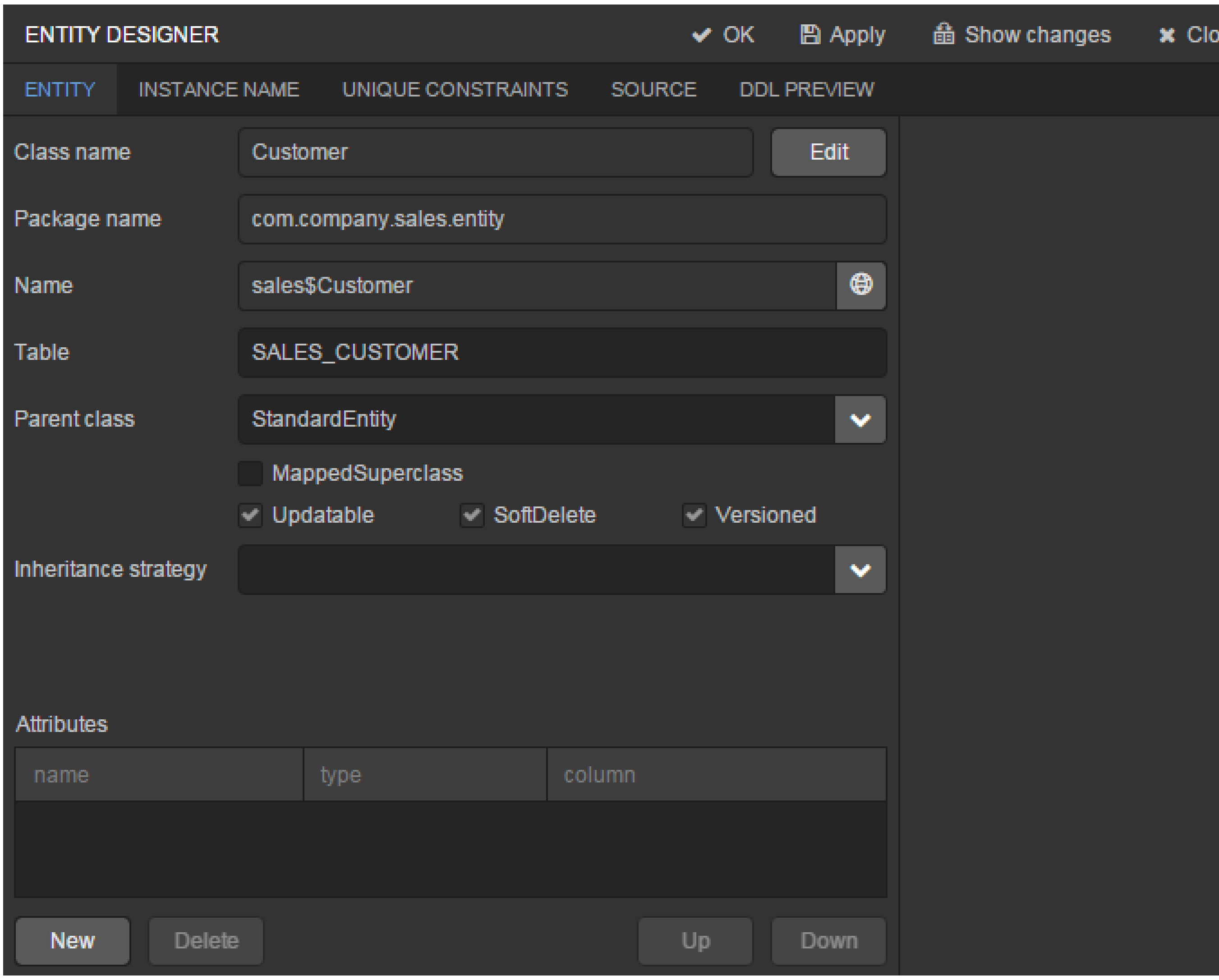
- Go to the **Entities** tab in the navigation section and click **New entity**. The **New entity** dialog window will appear.
- Enter the name of the entity class – `Customer` – in the **Class name** field.



New entity

Class name: Customer
Package name: com.company.sales.entity
Entity type: Persistent
Id type: UUID

- Click **OK**. The entity designer page will be displayed in the workspace.


ENTITY DESIGNER

ENTITY | INSTANCE NAME | UNIQUE CONSTRAINTS | SOURCE | DDL PREVIEW

Class name	Customer
Package name	com.company.sales.entity
Name	sales$Customer
Table	SALES_CUSTOMER
Parent class	StandardEntity
	☐ MappedSuperclass
	✓ Updatable ✓ SoftDelete ✓ Versioned
Inheritance strategy	

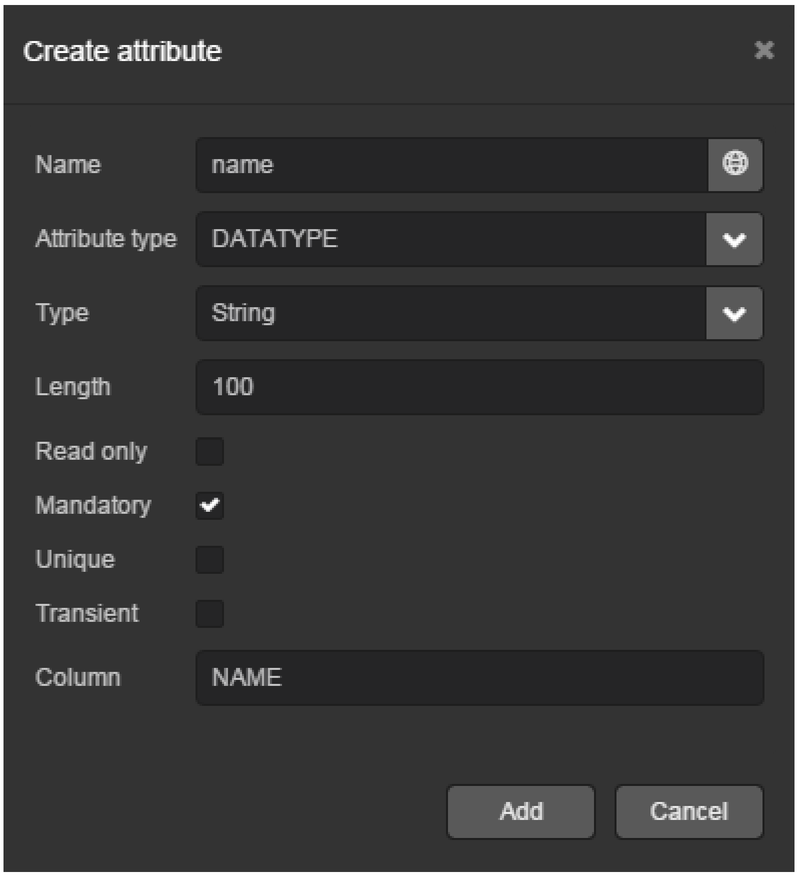
Attributes

name	type	column

- The entity name and the database table name will be automatically generated in the **Name** and the **Table** fields respectively.
- Leave the existing value – `StandardEntity` - in the **Parent class** field.
- Leave the **Inheritance strategy** field blank.
- Click button next to the **Name** to open the **Localized message** window. Specify localization for the entity name for the available languages in it.

Next, let us create entity attributes. To do this, click the **New** button below the **Attributes** table.

- **Create attribute** window will appear. Enter the name of the entity attribute – `name`, in the **Name** field. Select `DATATYPE` value in the **Attribute type** list, specify `String` attribute type in the **Type** field and then set the length of the text attribute to 100 characters in the **Length** field. Check the **Mandatory** box. The name of the database table column will be automatically generated in the **Column** field.


Create attribute

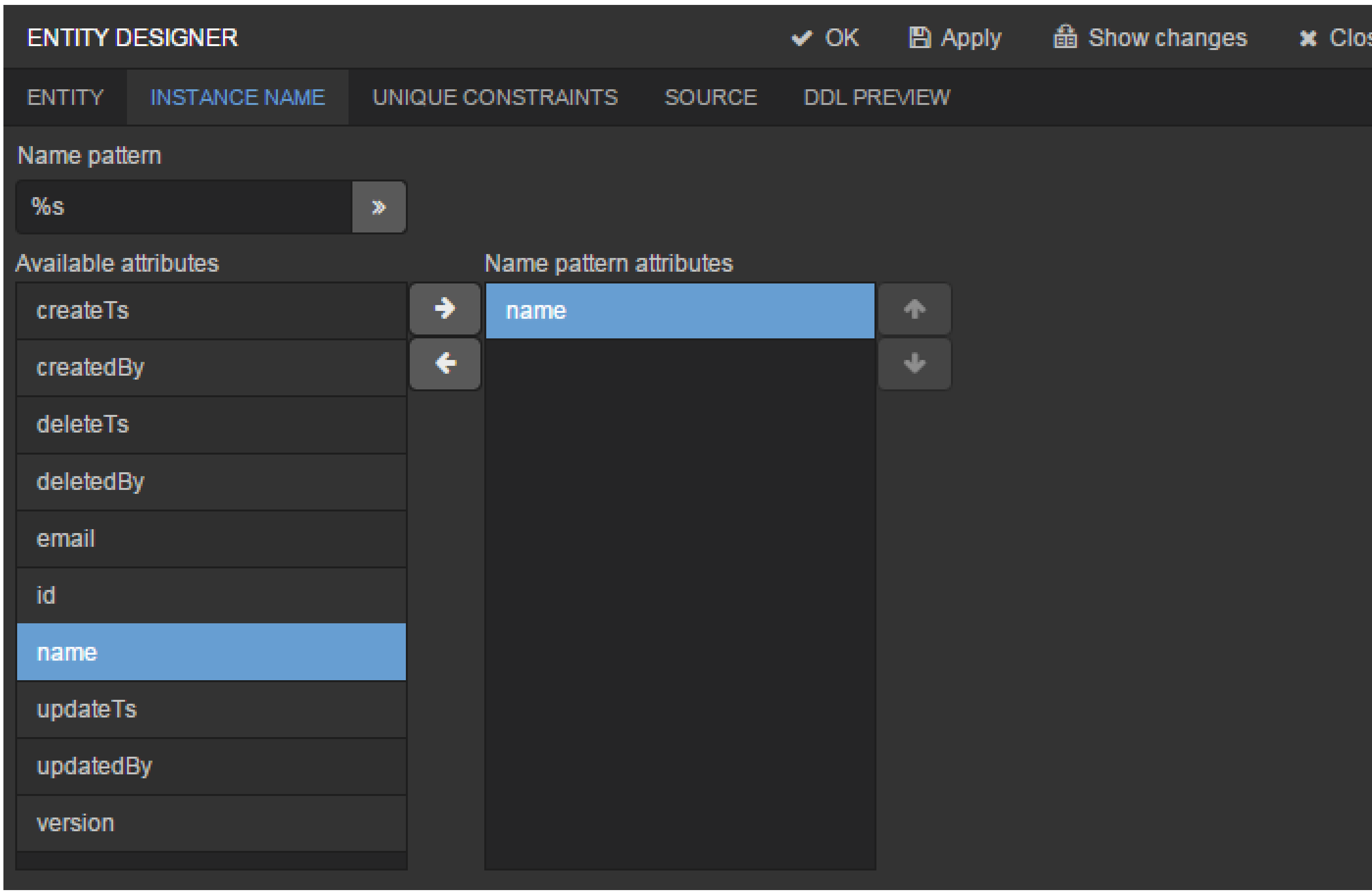
Name	name
Attribute type	DATATYPE
Type	String
Length	100
Read only	☐
Mandatory	✓
Unique	☐
Transient	☐
Column	NAME

Now click button next to the attribute name to open the **Localized message** window. Localize the attribute name in the available languages.

Click **Add** to add the attribute.

- `email` attribute is created in the same way but the value in **Length** field should be set to `50`.

After creating the attributes, go to the **Instance name** tab in the entity designer to specify Name pattern. Select the **name** attribute in the **Available attributes** list and move it to the **Name pattern attributes** list by clicking the button with the right arrow on it.



Name pattern: %s

Available attributes: createTs, createdBy, deleteTs, deletedBy, email, id, name, updateTs, updatedBy, version

Name pattern attributes: name

Customer entity creation is now complete. Click **OK** in the top panel to save the changes and close the page.

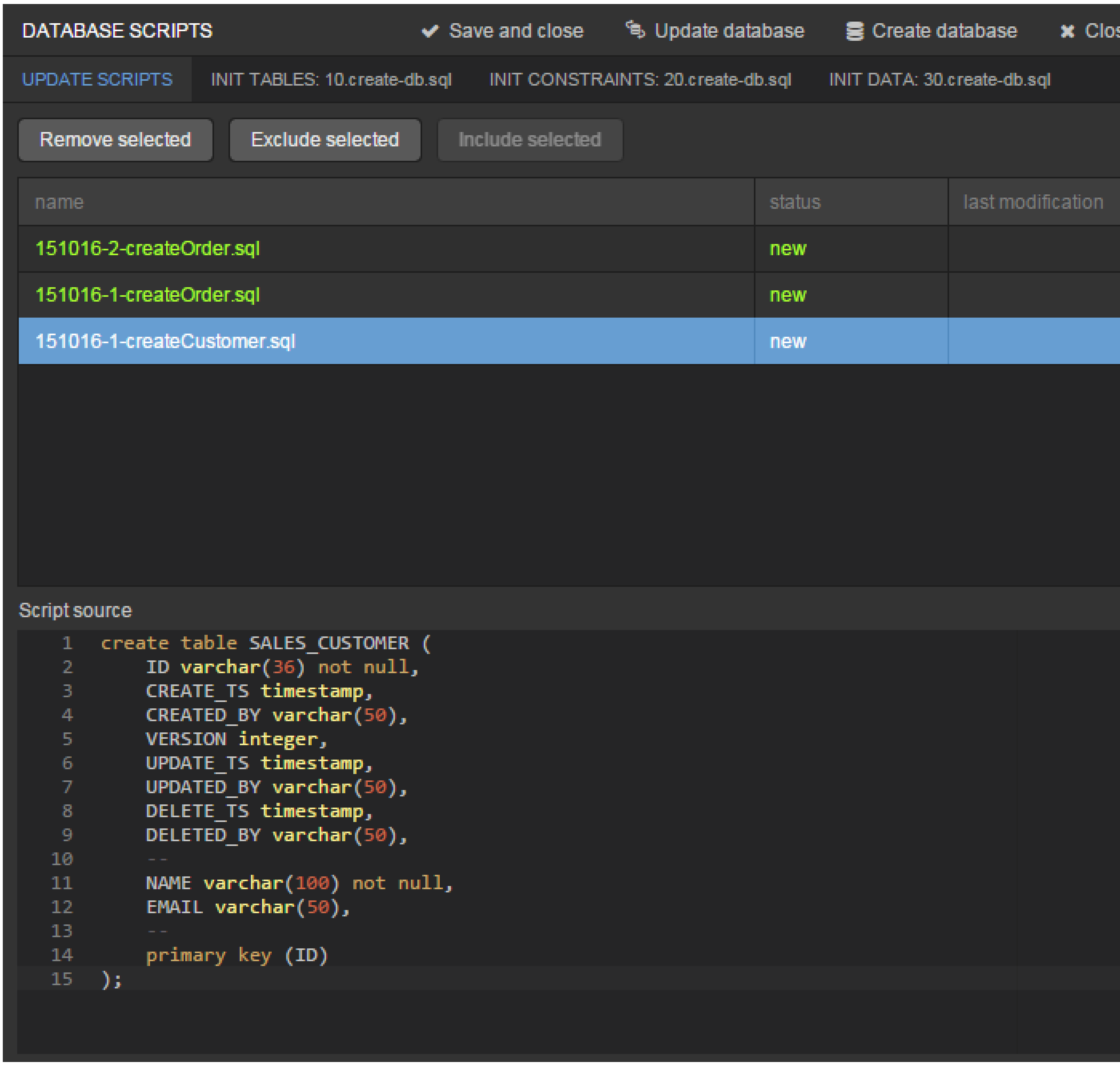
Let us create the `Order` entity. Click **New entity** option on the **Entities** tab. Enter the **Class name** – `Order`. The entity should have the following attributes:

- **Name** – `customer`, **Attribute type** – `ASSOCIATION`, **Type** – `Customer`, **Cardinality** – `MANY_TO_ONE`.
- **Name** – `date`, **Attribute type** – `DATATYPE`, **Type** – `Date`. Check **Mandatory** box for `date` attribute.
- **Name** – `amount`, **Attribute type** – `DATATYPE`, **Type** – `BigDecimal`.

Specify localized caption for each of the attributes by clicking the 🌐 button next to the attribute name.

3.4. Creating Database Tables

It is sufficient to click **Generate DB scripts** button in **Entities** tab on the navigation panel to create database tables. After that, **Database scripts** page will open. Both incremental DB update scripts from the current state (**Update scripts**) and initial DB creation scripts (**Init tables**, **Init constraints**, **Init data**) will be generated on this page.



name	status	last modification
151016-2-createOrder.sql	new	
151016-1-createOrder.sql	new	
151016-1-createCustomer.sql	new	

```
create table SALES_CUSTOMER (
    ID varchar(36) not null,
    CREATE_TS timestamp,
    CREATED_BY varchar(50),
    VERSION integer,
    UPDATE_TS timestamp,
    UPDATED_BY varchar(50),
    DELETE_TS timestamp,
    DELETED_BY varchar(50),
    --
    NAME varchar(100) not null,
    EMAIL varchar(50),
    --
    primary key (ID)
);
```

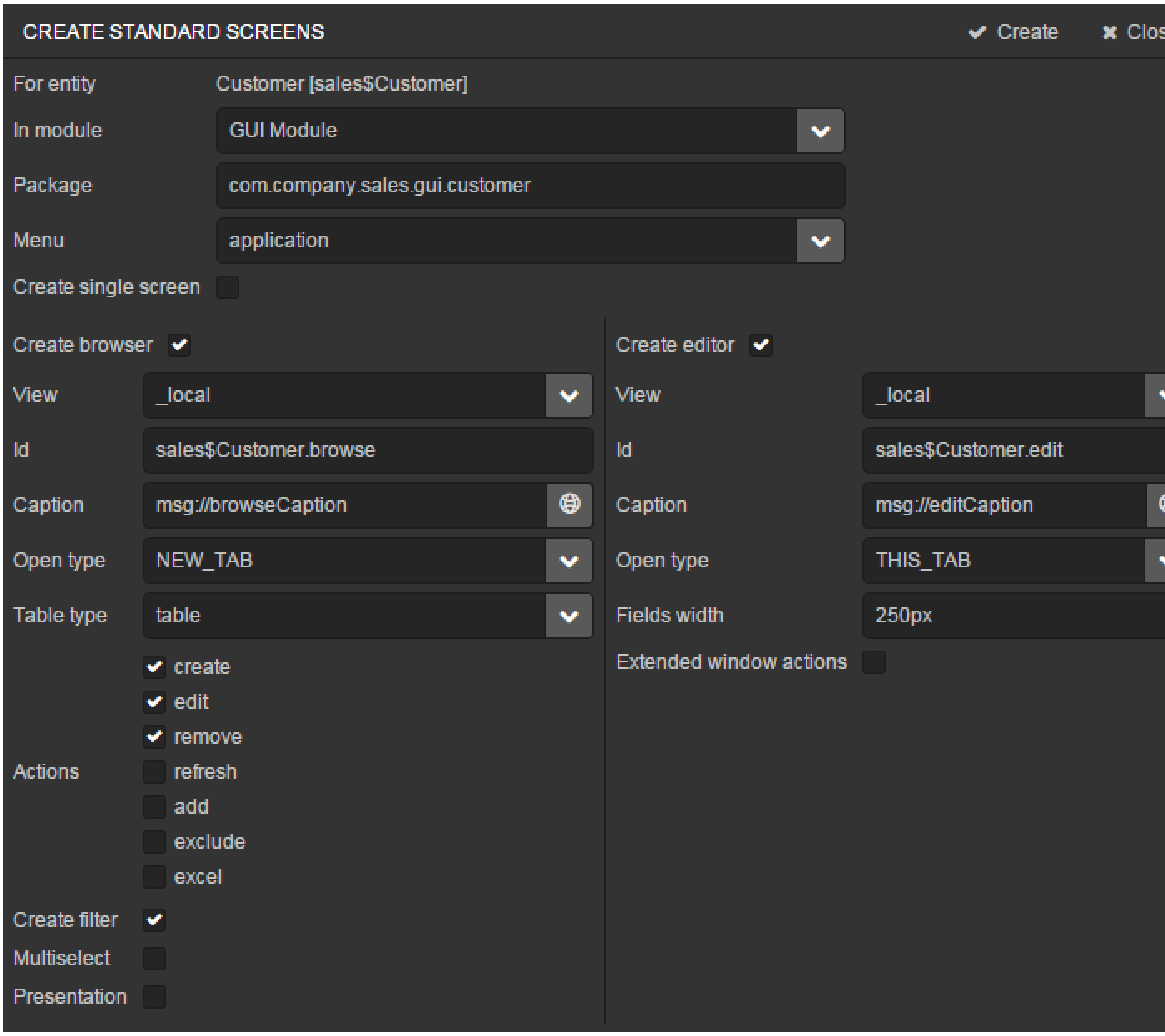
Click **Save and close** button to save the generated scripts. To run update scripts, stop the running application using the **Run** > **Stop application server** command, then select **Run** > **Update database**.

3.5. Creating User Interface Screens

Now we will create screens for customers and orders data management.

3.5.1. Screens for Customer

Select `Customer` entity in the **Entities** tab on the navigation panel to create standard screens for viewing and editing Customers. Click **Create standard screens** link at the bottom of the section. After that, **Create standard screens** page will appear.


CREATE STANDARD SCREENS ✔ Create ✖ Clos

For entity	Customer [sales$Customer]
In module	GUI Module
Package	com.company.sales.gui.customer
Menu	application
Create single screen	☐

Create browser	✓	Create editor	✓
View	_local	View	_local
Id	sales$Customer.browse	Id	sales$Customer.edit
Caption	msg://browseCaption	Caption	msg://editCaption
Open type	NEW_TAB	Open type	THIS_TAB
Table type	table	Fields width	250px
		Extended window actions	☐
Actions	✓ create		
	✓ edit		
	✓ remove		
	☐ refresh		
	☐ add		
	☐ exclude		
	☐ excel		
Create filter	✓		
Multiselect	☐		
Presentation	☐		

All fields in this dialog are already populated with default values, there is no need to change them. Click the **Create** button.

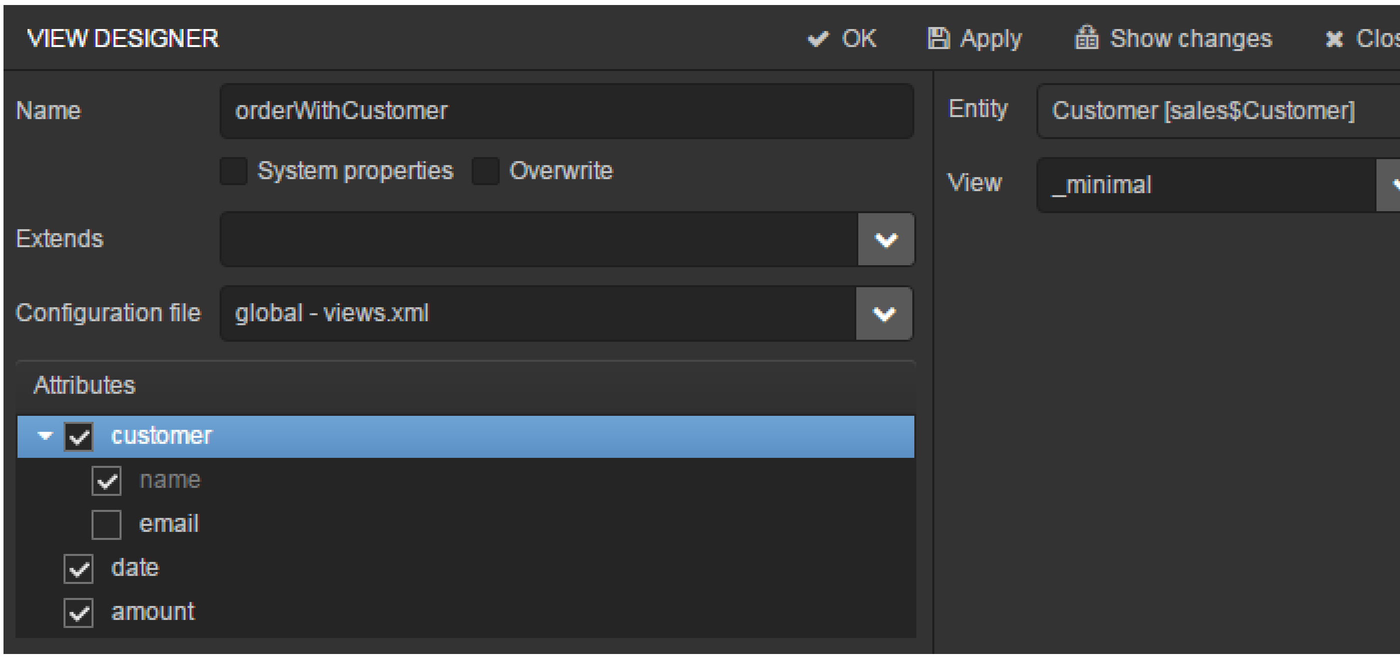
`customer-edit.xml` and `customer-browse.xml` items will appear in **GUI Module** on **Screens** tab of the navigation panel.

You can specify localized captions for the screens. For this, select a screen and click **Edit** to open the screen designer page. Go to the **Properties** tab. Click the [] button next to the **Caption** field and specify screen names in different locales. Alternatively, you can open `messages.properties` item located in the screens package and edit `browseCaption` and `editCaption` messages for available locales.

3.5.2. Order Screens

`Order` entity has the following distinction: since one of the attributes is the `Order.customer` reference attribute, you should define a view including this attribute (standard `_local` view does not include reference attributes).

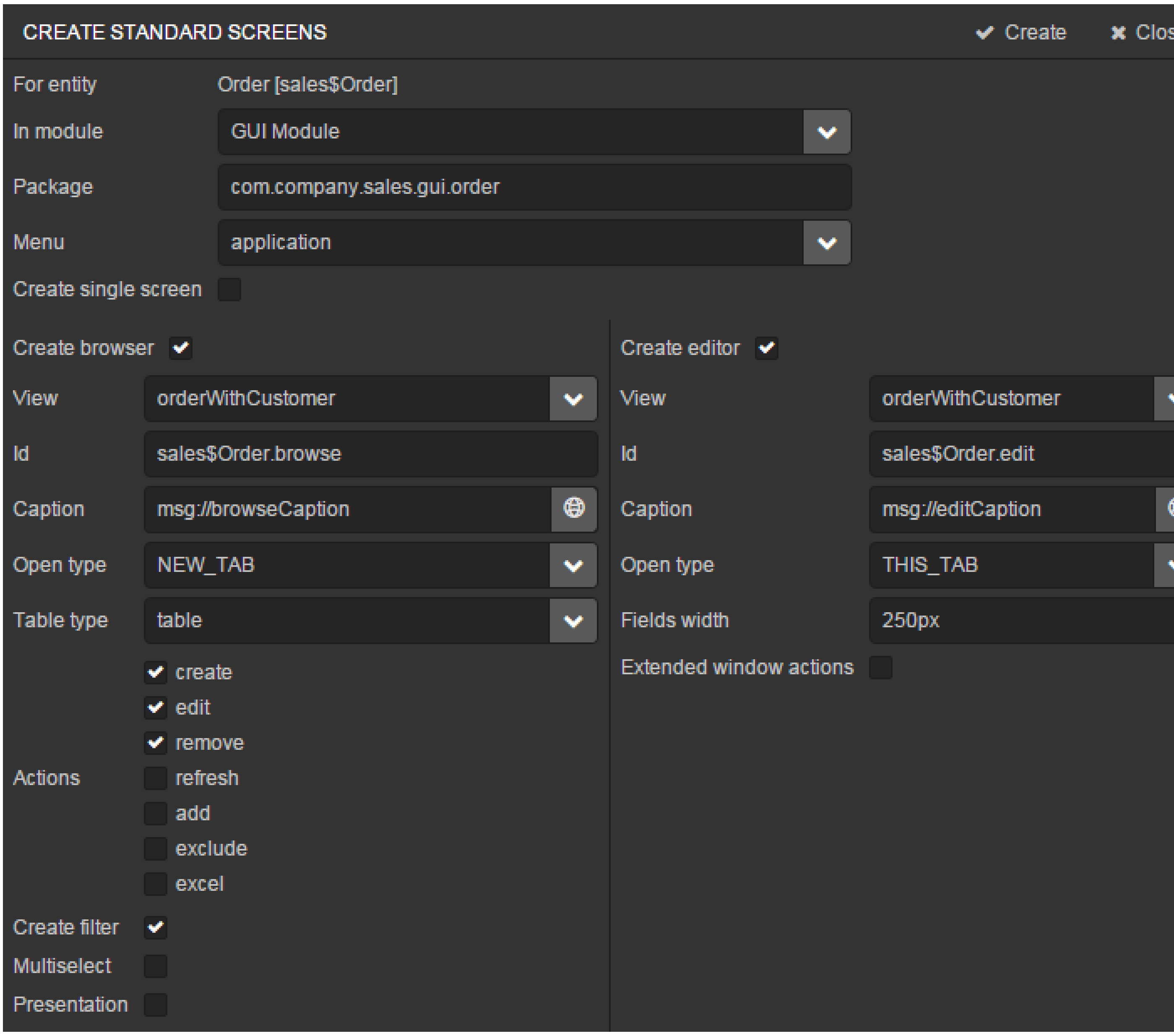
Go to the **Entities** tab on the navigation panel, select the `Order` entity and click the **New view** button. View designer page will open. Enter `orderWithCustomer` as the view name, click on `customer` attribute and select `_minimal` view for the `Customer` entity in the panel on the right.



Name: orderWithCustomer
Extends:
Configuration file: global - views.xml
Entity: Customer [sales$Customer]
View: _minimal

Click **OK** in the top panel.

After that, select the `Order` entity and click **Create standard screens**. Select `orderWithCustomer` in the **View** fields in both browser and editor panels of the **Create standard screens** page and click **Create**.



For entity	Order [sales$Order]
In module	GUI Module
Package	com.company.sales.gui.order
Menu	application
Create single screen	☐

Create browser	✓	Create editor	✓
View	orderWithCustomer	View	orderWithCustomer
Id	sales$Order.browse	Id	sales$Order.edit
Caption	msg://browseCaption	Caption	msg://editCaption
Open type	NEW_TAB	Open type	THIS_TAB
Table type	table	Fields width	250px
Actions	✓ create	Extended window actions	☐
	✓ edit		
	✓ remove		
	☐ refresh		
	☐ add		
	☐ exclude		
	☐ excel		
Create filter	✓		
Multiselect	☐		
Presentation	☐		

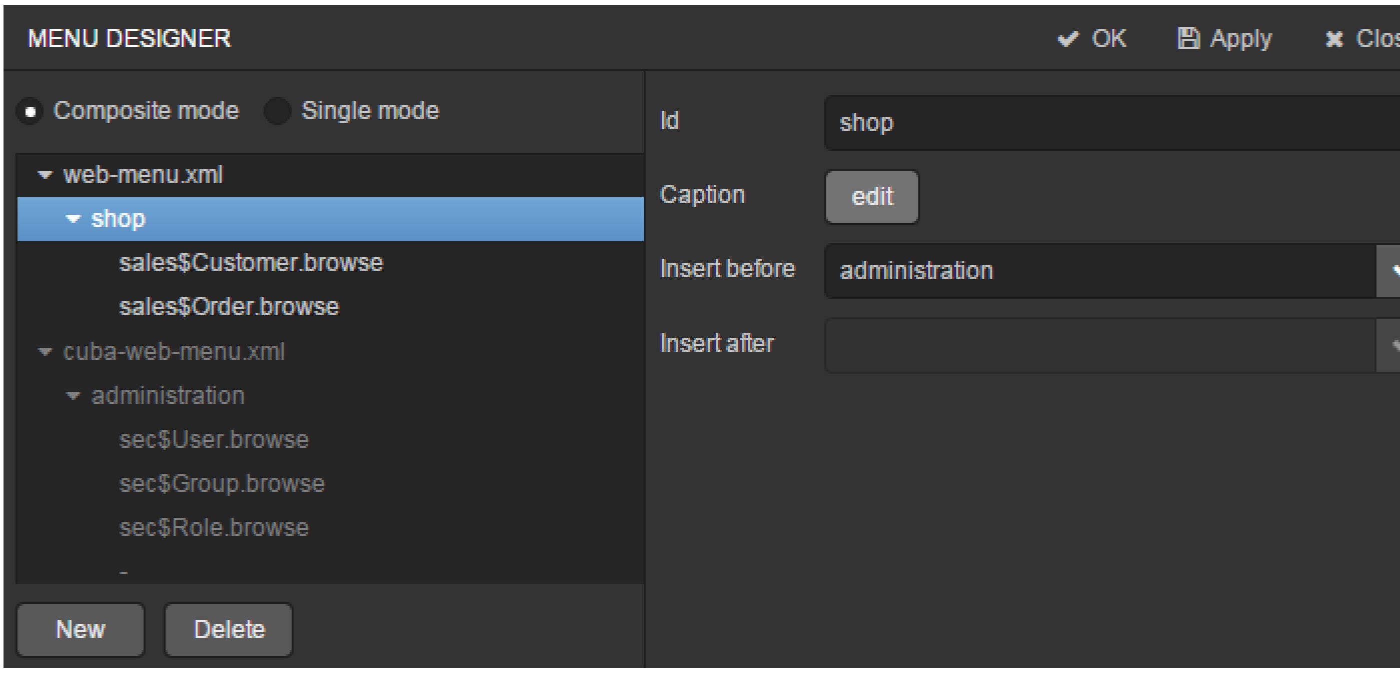
`order-edit.xml` and `order-browse.xml` items will appear in the **GUI Module** on the **Screens** tab of the navigation panel.

You can specify localized captions for the Order screens as described above for the Customer screens.

3.5.3. Application Menu

At the moment of their creation, the screens were added to the **application** menu item of the default application menu. Let us rename it. Switch to the **Main menu** tab on the navigation panel and click **Edit**. The **Menu designer** page will open. Select the `application` menu item to edit its properties.

Enter the new value of the menu identifier – `shop` – in the **Id** field, then click the **Caption edit** button and set localized names of the menu item.



● Composite mode ○ Single mode

- web-menu.xml
  - shop
    - sales$Customer.browse
    - sales$Order.browse
- cuba-web-menu.xml
  - administration
    - sec$User.browse
    - sec$Group.browse
    - sec$Role.browse
    - -

Id	shop
Caption	edit
Insert before	administration
Insert after	

After editing the menu, click **OK** in the top panel.

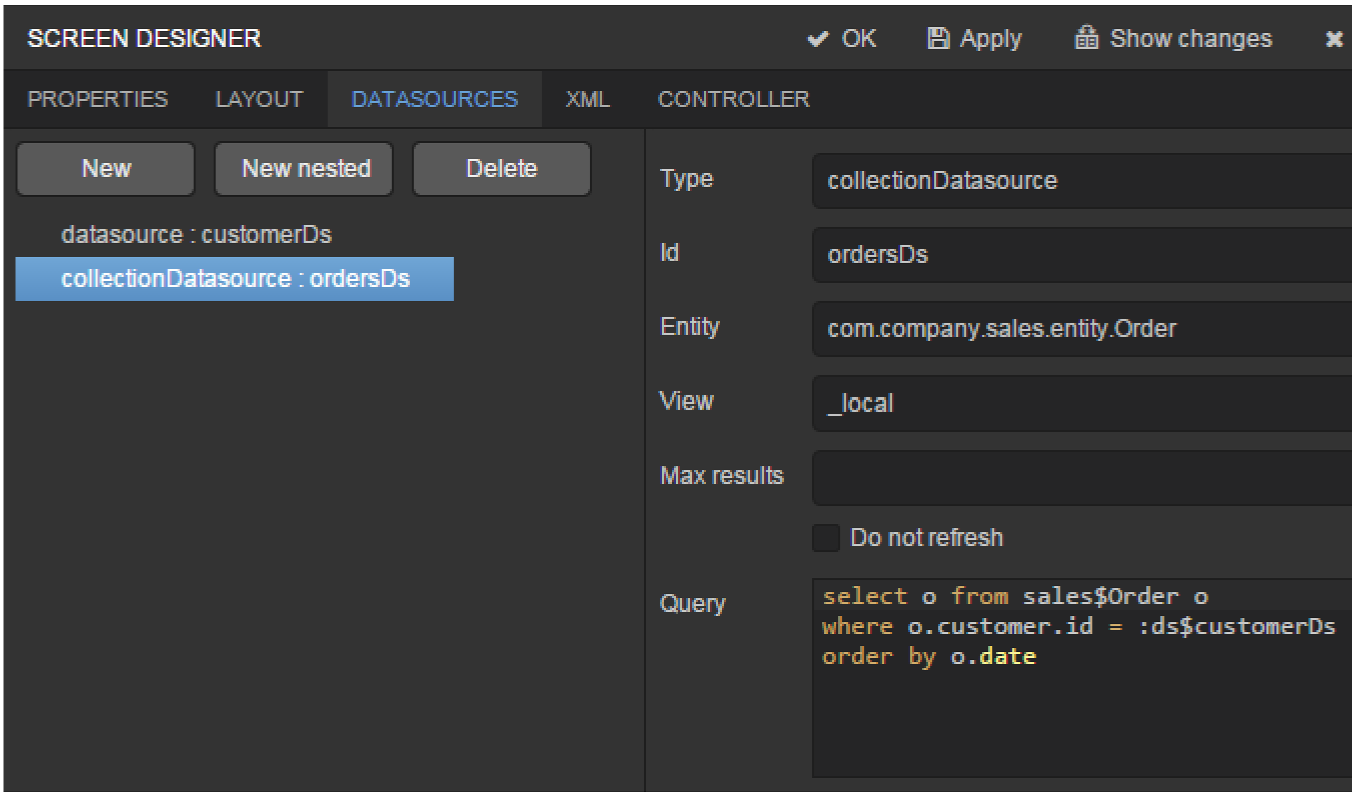
3.5.4. Customer Editor With a List of Orders

Do the following to display the list of Orders in the Customers edit screen:

- Go to the **Screens** tab on the navigation panel. Choose `customer-edit.xml` screen and click **Edit**.
- Go to the **Datasources** tab on the screen designer page and click **New**.
- Select the newly created data source in the list. Its attributes will appear in the right part of the page.
- Specify `collectionDatasource` in the **Type** field.
- In **Id** field enter the data source identifier – `ordersDs`.
- Select `com.sample.sales.entity.Order` entity in the **Entity** list.
- Select `_local` view in the **View** list.
- Enter the following query in the **Query** field:

  ```
  select o from sales$Order o where o.customer.id = :ds$customerDs
  order by o.date
  ```

  The query contains orders selection criterion with `ds$customerDs` parameter. The parameter value named like `ds${datasource_name}` will contain id of the entity selected in `datasource_name` datasource at the moment, in this case it is the id of the Customer being edited.



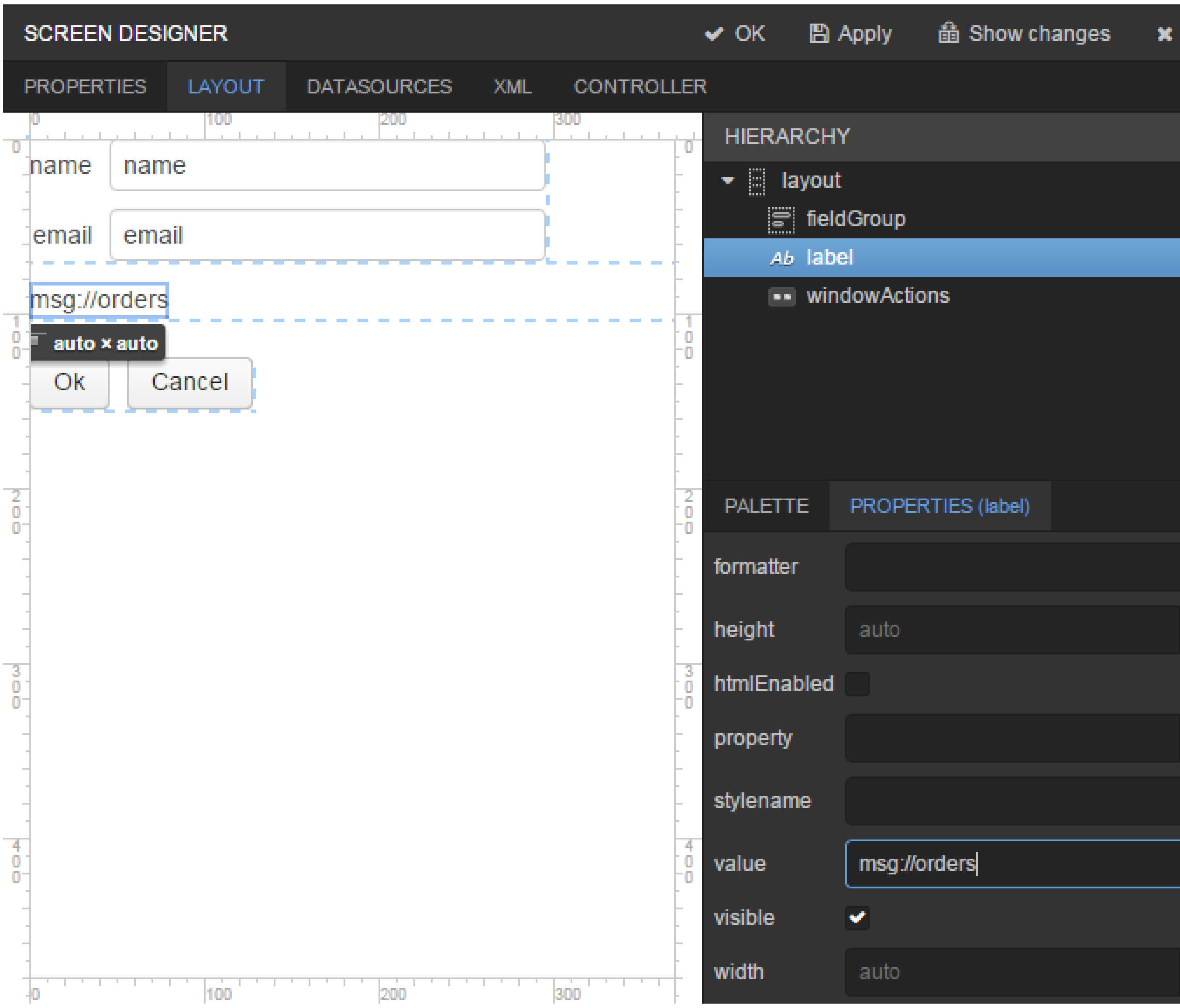
datasource : customerDs
collectionDatasource : ordersDs

Type	collectionDatasource
Id	ordersDs
Entity	com.company.sales.entity.Order
View	_local
Max results	
	Do not refresh

Query

```
select o from sales$Order o
where o.customer.id = :ds$customerDs
order by o.date
```

- Click **Apply** to save the changes.
- Next go to the **Layout** tab in the screen designer and find the `Label` component in the components palette. Drag this component to the screen components hierarchy panel and place it between `fieldGroup` and `windowActions`. Go to the **Properties** tab in the properties panel. Enter `msg://orders` in the **value** field. Click the button next to the **value** field and define label values in available languages.

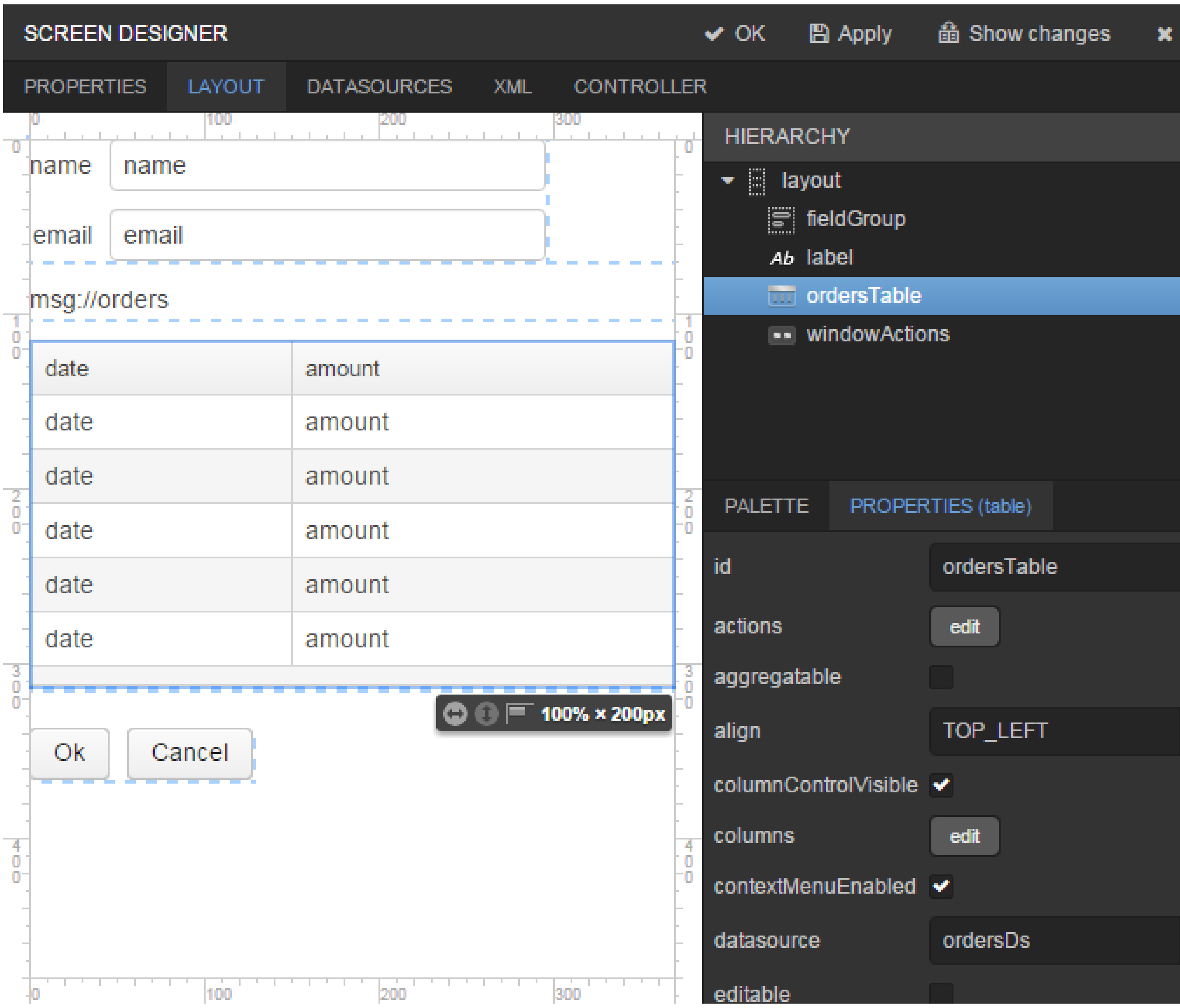


formatter
height: auto
htmlEnabled
property
stylename
value: msg://orders
visible: ✓
width: auto

Tip If the application is not intended to be used in multiple languages, the value in the **value** field can be entered straight in the required language.

- Drag `Table` from the components palette to components hierarchy panel and place it between `label` and `windowActions`. Select this component in the hierarchy and specify table size in properties on the **Layout** tab: set `100%` in the **width** field and `200px` in the **height** field.

  Go to the **Properties** tab. Set `ordersTable` value as **id**, choose `orderDs` from the list of available datasources.



id	ordersTable
actions	edit
aggregatable	
align	TOP_LEFT
columnControlVisible	✓
columns	edit
contextMenuEnabled	✓
datasource	ordersDs
editable	

- Click **OK** in the top panel to save the changes in the screen.

3.6. Running the Application

Now let us see how the created screens look in the actual application. Select **Run** > **Restart application server**.

Log in selecting English language in the login window. Open the **Sales** > **Customers** menu item:

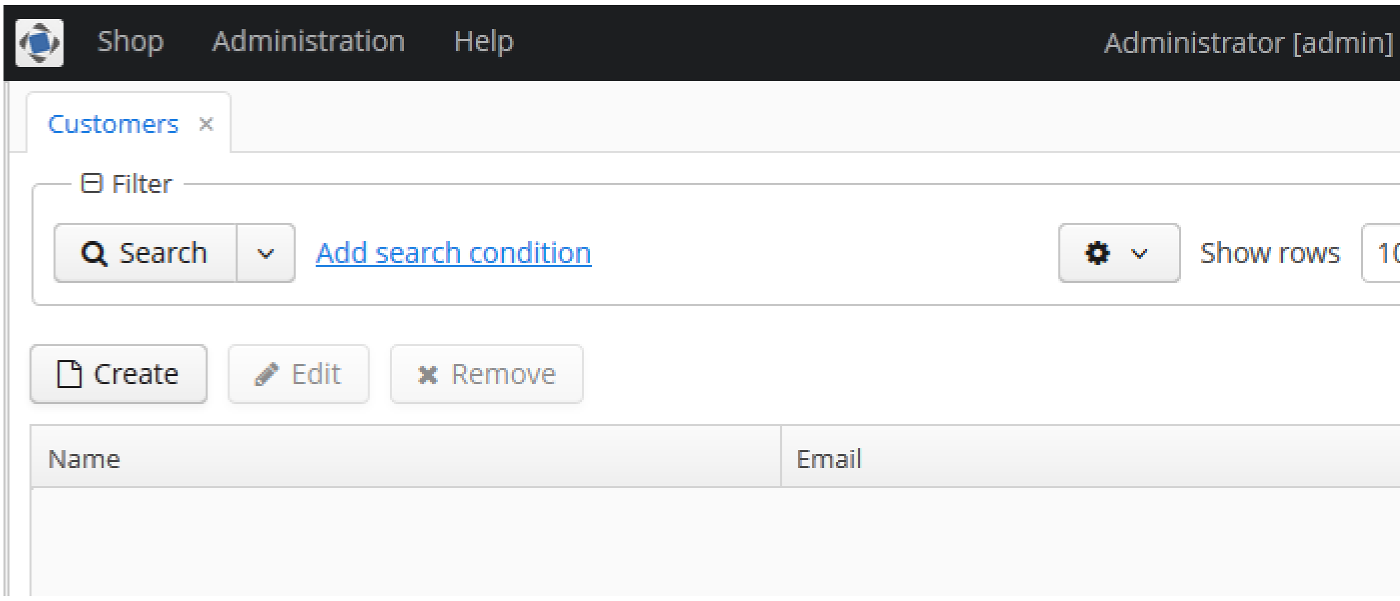
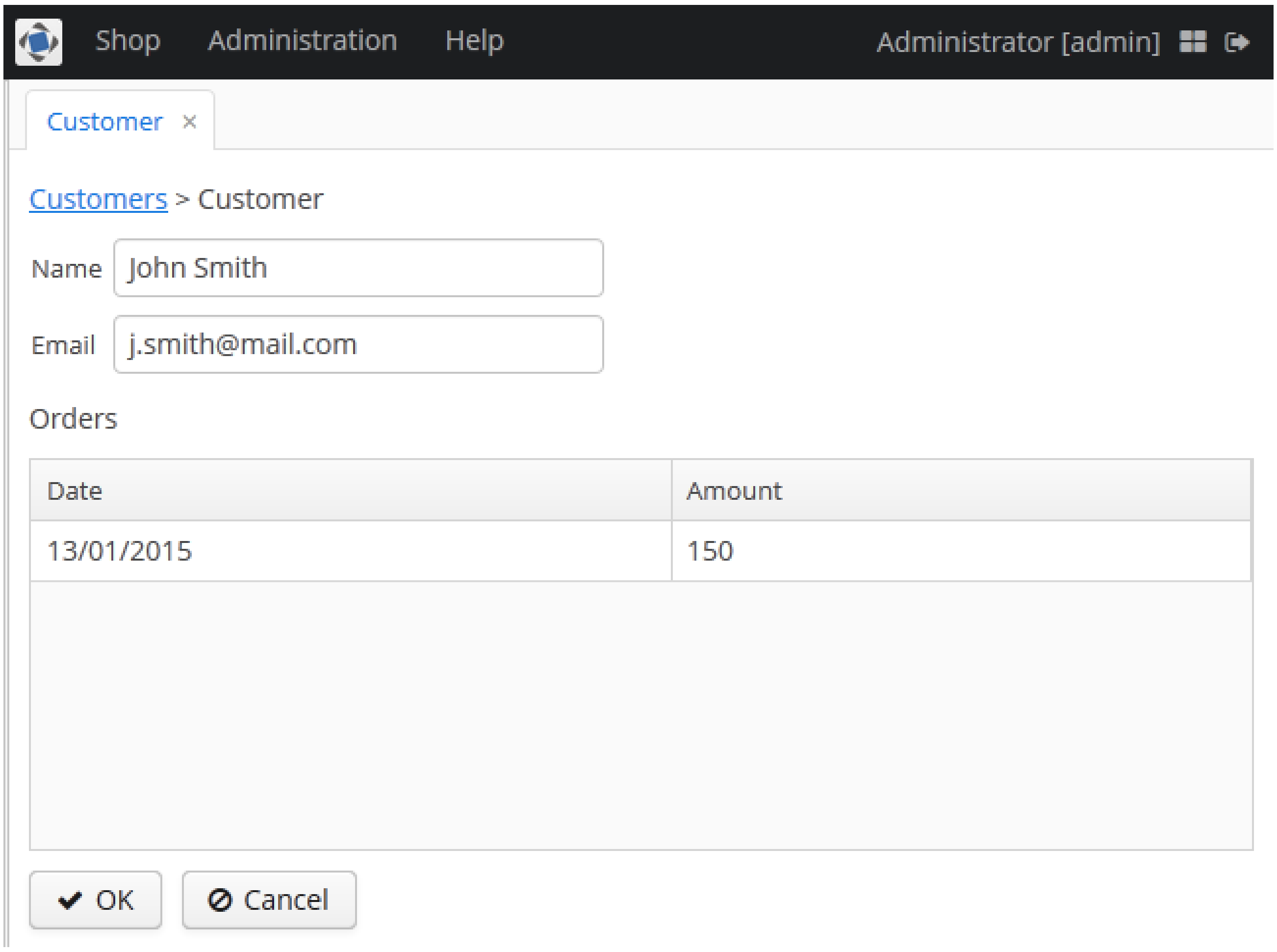


Figure 1. The Customers browser

Click **Create**:



Name: John Smith

Email: j.smith@mail.com

Orders

Date	Amount
13/01/2015	150

Figure 2. The Customer editor screen

Open the **Sales** > **Orders** menu item:

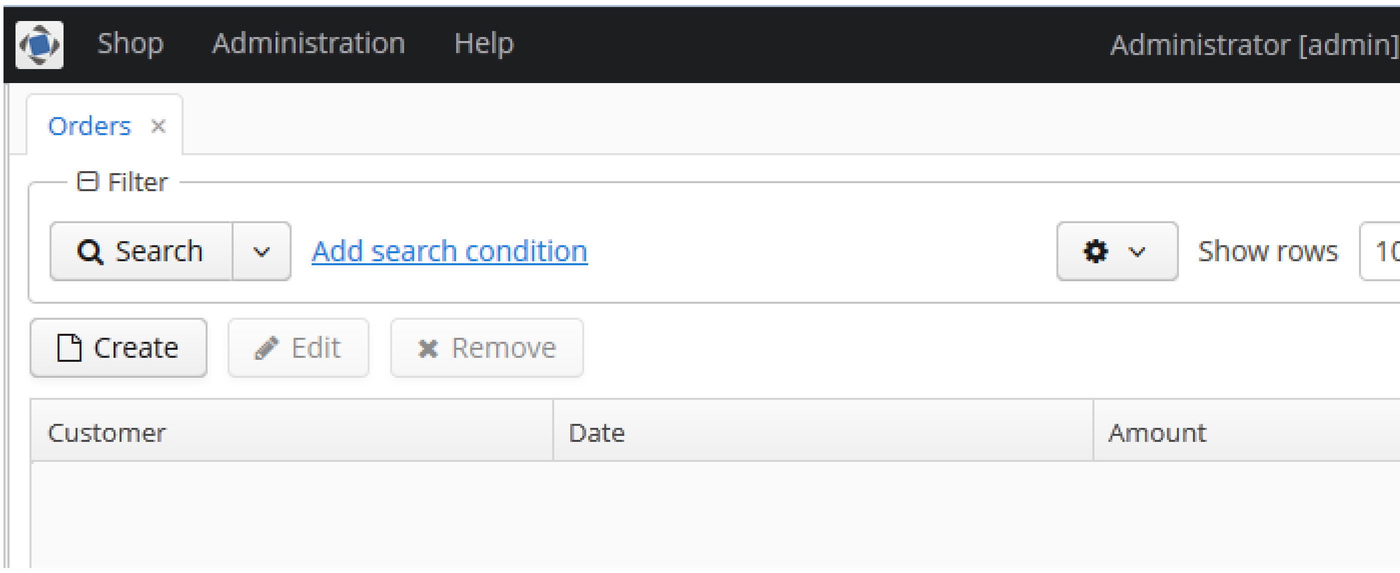


Figure 3. The Orders browser

Click **Create**:

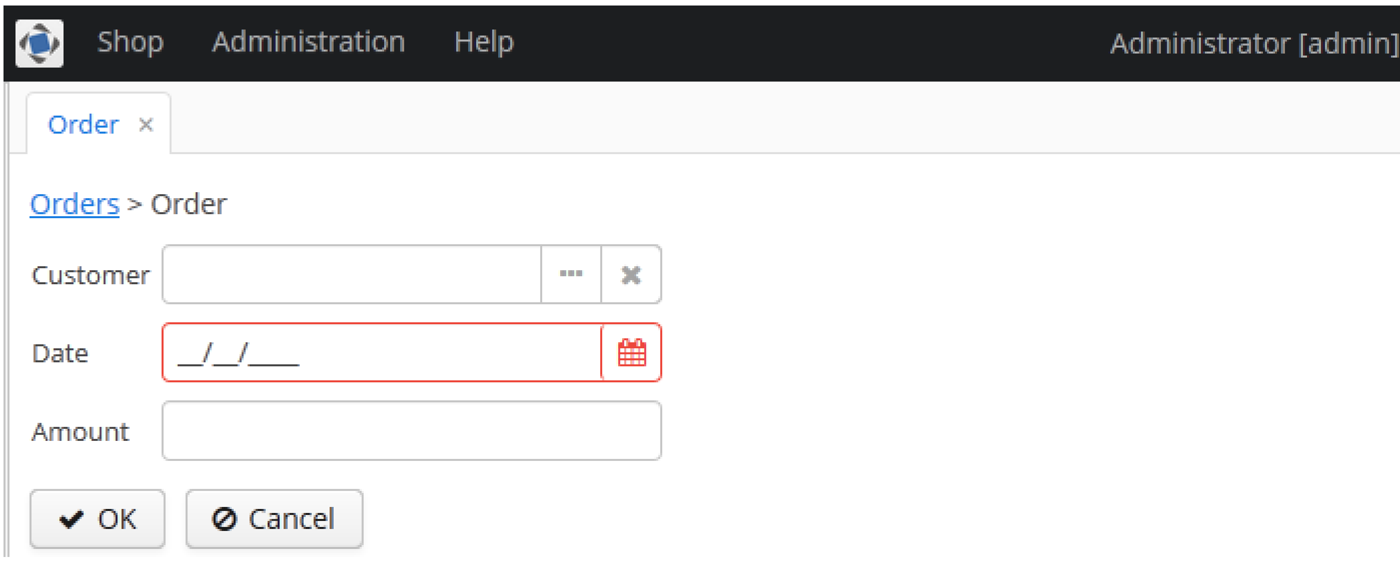


Figure 4. The Order editor