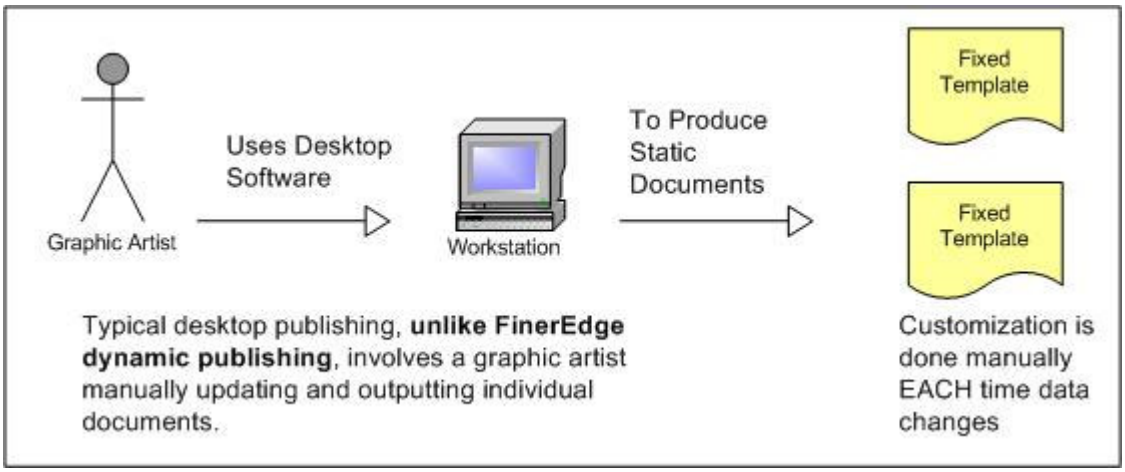
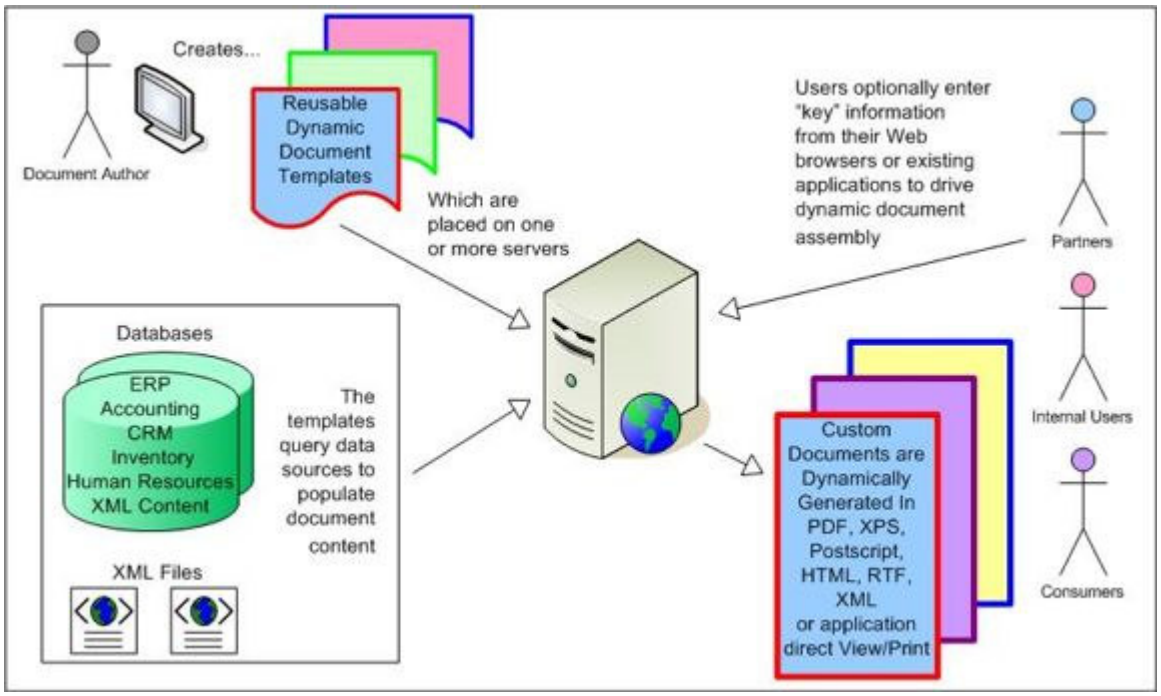
Dynamic Publishing vs. Static Publishing
What You Might Not Know

While there has been progress in the area of static publishing products that largely depend upon manually formatted pages, until recently very little has been accomplished in the area of automated dynamic publishing from a practical perspective. Manual or even partially automated static publishing systems, such as a number of Desk Top Publishing systems, typically employ an underlying model whereby most pages are individually formatted using absolute page positioning methods, while also usually storing information about the documents themselves within closed proprietary binary file formats, as seen in Figure 1.



(Figure 1)

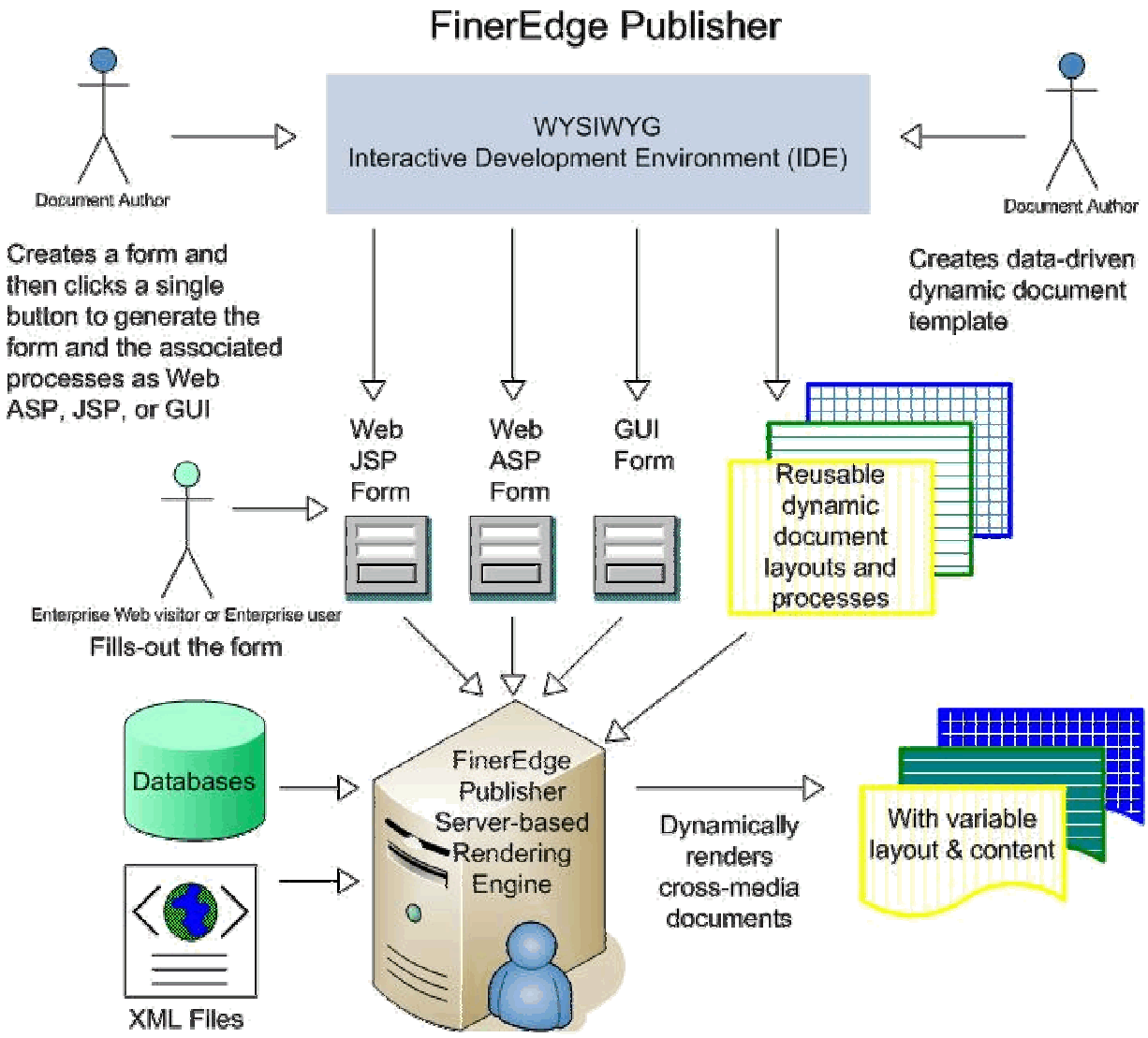
In contrast to manually oriented static publishing, dynamic publishing systems automate the production of documents by heavily drawing upon a customer's existing information from multiple data sources, such as their own production relational databases. These databases are often maintained and updated by existing production systems that successfully operate within the customer's organization. Furthermore, dynamic publishing systems normally have extensive facilities to widely vary a document's look and feel based upon highly flexible conditional formatting, which can even be driven directly by the content of the customer's external database information or even from end-users entering data from Web or application forms as seen in Figure 2.



(Figure 2)

Unlike many static publishing systems, dynamic publishing systems such as FinerEdge Publisher are built specifically for automation and can easily exist in both a back-end server environment or within a traditional client-based platform. While an automated dynamic publishing solution must be very quick without consuming excessive resources such as memory, the advantages that can be realized by operating such a system on a cluster of back-end servers has enormous potential for creating customized on-the-fly output for the clients of a customer that wish to access the resulting document output from the web.

FinerEdge Publisher can simultaneously produce precisely detailed and fully scoped cross-media outputs, such as 1000+ page catalogs, in many industry accepted standards such as PDF and PostScript, XPS, RTF, and XHTML. But while FinerEdge Publisher can automatically produce very large document outputs from multiple data sources, the dynamic rendering model employed by FinerEdge Publisher opens the door to also creating smaller document cross-sections and sub-catalogs on-the-fly, all from the same document structure. To drive this document production, FinerEdge Publisher also generates and automatically deploys complete turnkey solutions in a number of well-known web environments such as Active Server Pages (ASP) and Java Server Pages (JSP). An illustration of how all of this comes together can be seen in Figure 3.



(Figure 3)

Implementing this type of automated environment with well-understood markup elements is a real challenge that FinerEdge Publisher has succeeded in surpassing. Along with the dynamic pagination of documents using commonly understood markup elements, complex situations arise as a factor of all of the possible combinations of elements, including nested tables, that can be embedded within each other to any level required. In turn, this mandates that many simultaneous flows of information be maintained over page boundaries and be precisely resumed on subsequent pages. Individual complex elements containing other embedded elements such as nested tables, images, and text can also be measured on-the-fly and moved to subsequent pages based upon detailed criteria, which might also be ascertained directly from a customer's database information.

FinerEdge Publisher is an industrial-strength solution that can also format the variable length flows of information with respect to page-relative criteria such as top, bottom, left, and right page areas, which can further vary based upon additional properties such as odd and even pages, gutters, variable page lengths, altered page numbering schemes, and many more criteria. Finally, highly flexible forward and backward document references facilitate the automatic creation of commonly known elements such as table of contents, indexes, dictionary-style appearances, and virtually any type of cross-referencing imaginable.

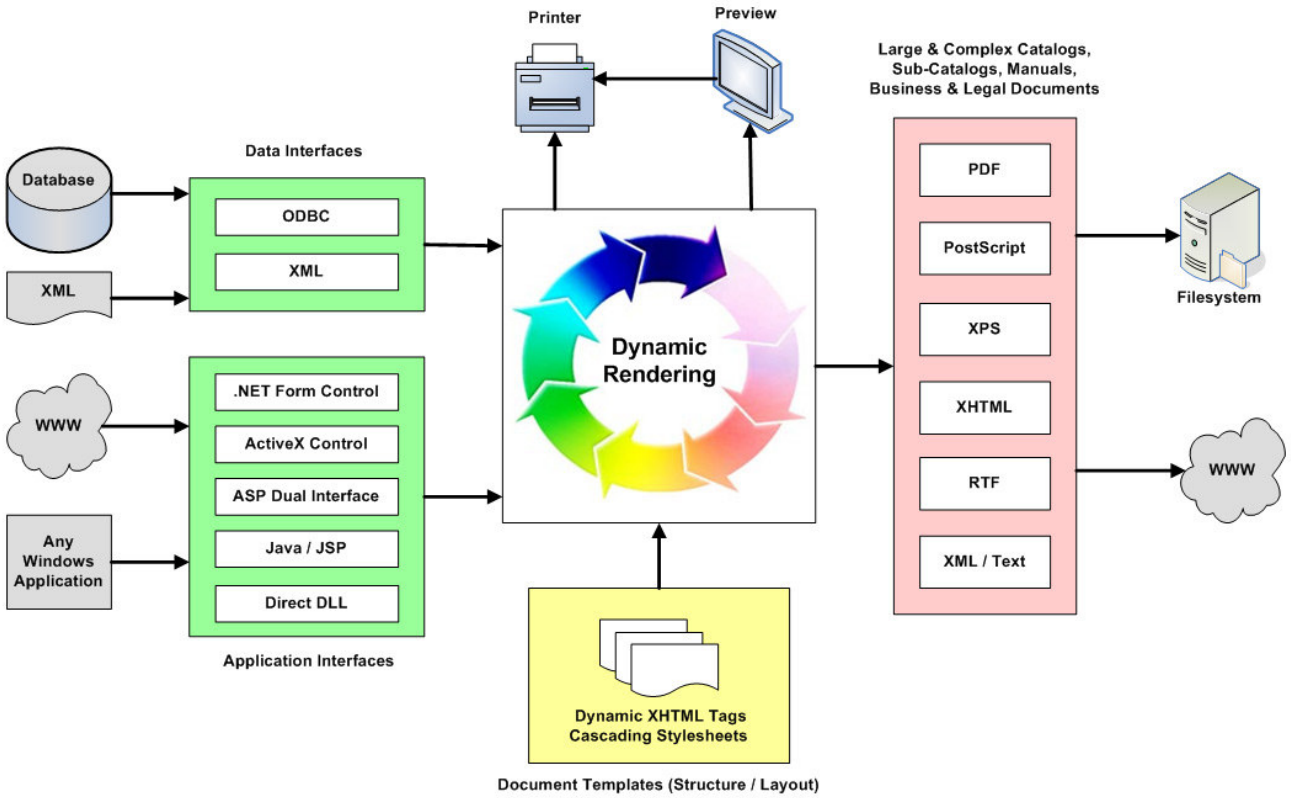
FinerEdge® Publisher – Overview of Features

An Integrated Development Environment simultaneously supports a WYSIWYG view, document design view, mark-up view, document generation view, form WYSIWYG view, and form generation view. A built-in document trace facility allows a document designer to perform any type of detailed document refinement. In addition, FinerEdge Publisher supports the automatic generation and deployment of both Active Server Pages (ASP) and Java Server Pages (JSP) to drive the generation of documents from front-end forms. Of course, all document structures are stored in an open XML-compliant format that adheres to XHTML and Cascading Style Sheet (CSS) standards. Also, Unicode is consistently implemented throughout the FinerEdge Publisher system’s rendering engine and all of its components.

Behind the WYSIWYG and visual design views, FinerEdge Publisher implements a set of commonly understood content elements such as table, div, and span. A common set of style properties along with content elements allow customized and widely varied appearances to be realized. Also, a rich set of built-in functions greatly enhances the types of customizations that can be attained during the automated dynamic creation of output document formats. Lastly, FinerEdge Publisher encapsulates all formatting instructions within user-defined methods that encourage both reuse and standardization.

Feature Chart - A technical illustration can be seen in Figure 4.

Feature Category	Feature Set
Input Sources	Database, XML data, or directly from a web or GUI application. All data inputs can be used interchangeably within the same document. The comprehensive built-in database import facilities support a visual designer that can utilize multiple data sources simultaneously, and that further visually supports all join types. The database designer can also generate entire documents or portions of documents based upon the selection of customizable themes.
Output Formats	PDF, PostScript, XHTML, XPS, RTF, direct viewing, direct printing, and textual. Output formats such as PDF and PostScript support multiple versions along with Open Prepress Initiative instructions for images. Cross-media navigation links can be generated for applicable formats such as PDF, XHTML, RTF, and direct viewing.
Image Types	TIFF, PNG, JPG, GIF, and BMP. All image-relative subtypes are supported for each of the major image types. Transparency is also supported on applicable image types such as TIFF, PNG, and GIF. Images can either be automatically scaled to fit formatting specifications or the formatting can be automatically adjusted to fit the true image sizes.
Application Interfaces	• An automation server for Active Server Pages and other applications • ActiveX controls for direct viewing and document generation • A Java Native Interface for Java Programs • A Java Bean designed specifically for Java Server Pages • A direct DLL-based Application Program Interface



(Figure 4)

More information, including Flash demos and a free trial download can be found at:
http://www.fineredge.com