Application Designer

Introduction

Version 8.3.4

July 2014

Application Designer

This document applies to Application Designer Version 8.3.4.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: CIT-CONCEPTS-834-20140722

Table of Contents

Preface

Application Designer focuses on building user interfaces for composite applications. It allows you to create rich internet applications based on the AJAX technology. It helps IT departments to develop and deploy rich internet and composite applications very rapidly by providing a design and runtime environment that does not require skills in JavaScript, HTML and CSS. It allows you to create rich interfaces (that is: clients that behave like thick clients). The GUI either runs inside a web browser without any plugins - or runs inside the so called Application Designer SWT Client, a Java-based thin client.

The efficiency during the development of user interfaces is based on:

- A rich library of professional controls.
- A simple binding between the controls inside your layout definition and the application running on the server side.
- Tools for creating your layout in a WYSIWYG mode, for creating server-side Java code, as well as defining literals outside the code and for translating these literals.

The usability at runtime is similar to what you are familiar with when working with native GUIs, for example, written in C++ or Java Swing - though the GUI is running 100% inside the browser:

- No page-flickering caused by permanent page-reloading when interacting with the server.
- Very fast and efficient data transfer between browser client and web/application server.

Due to the open architecture, you can easily connect to existing application logic on the server side as well as easily extend the control library with your own controls or even manipulate existing controls.

This introduction is organized under the following headings:

Concept	About layout creation and the server side architecture.
Content Page - Workplace/Portal	About the creation of highly interactive browser pages.
Development Process	About developing, testing, debugging and delivering your application.
Tools	An overview of the tools that are used for the development tasks.

1 Concept

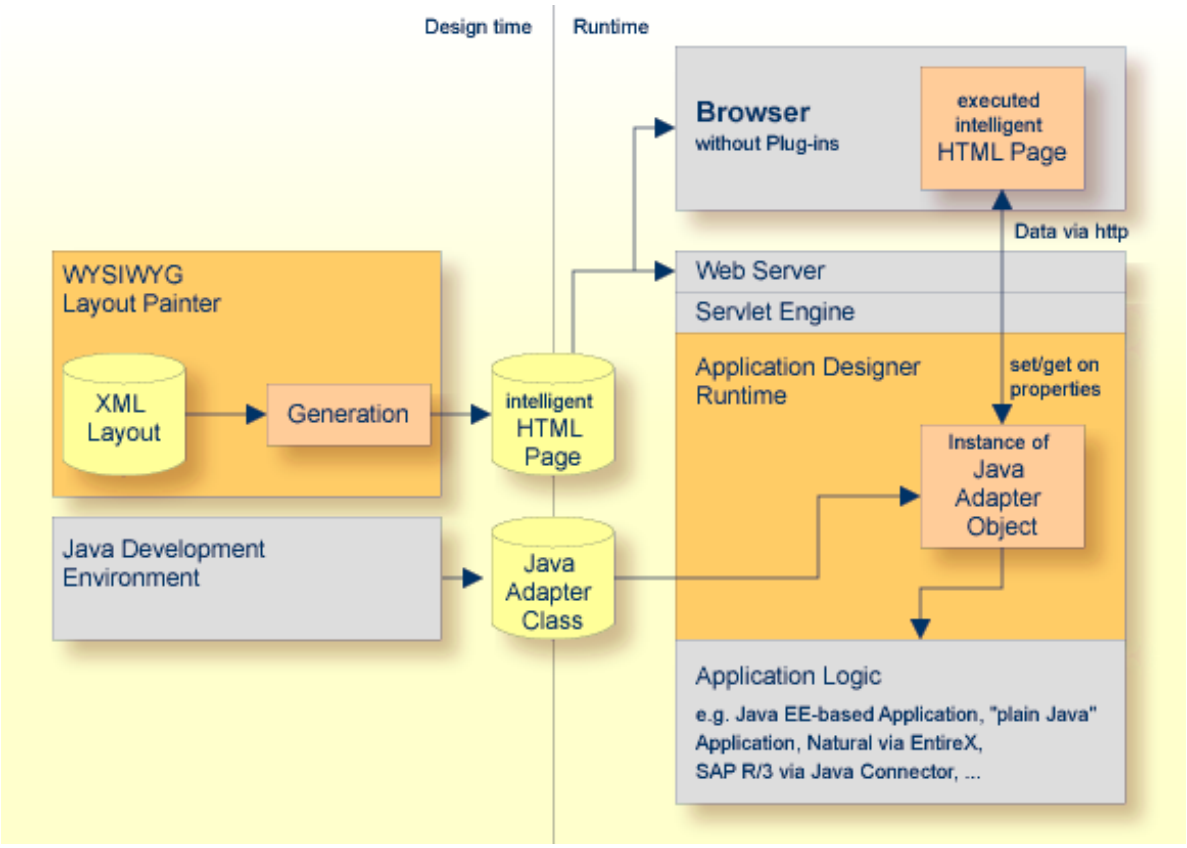
Layout Creation - Binding to Application

The Application Designer concept follows the commonly accepted paradigm of separating the layout (view) of an application as much as possible from the application logic itself.

To write a page (view) as a front-end to an application, you proceed as follows:

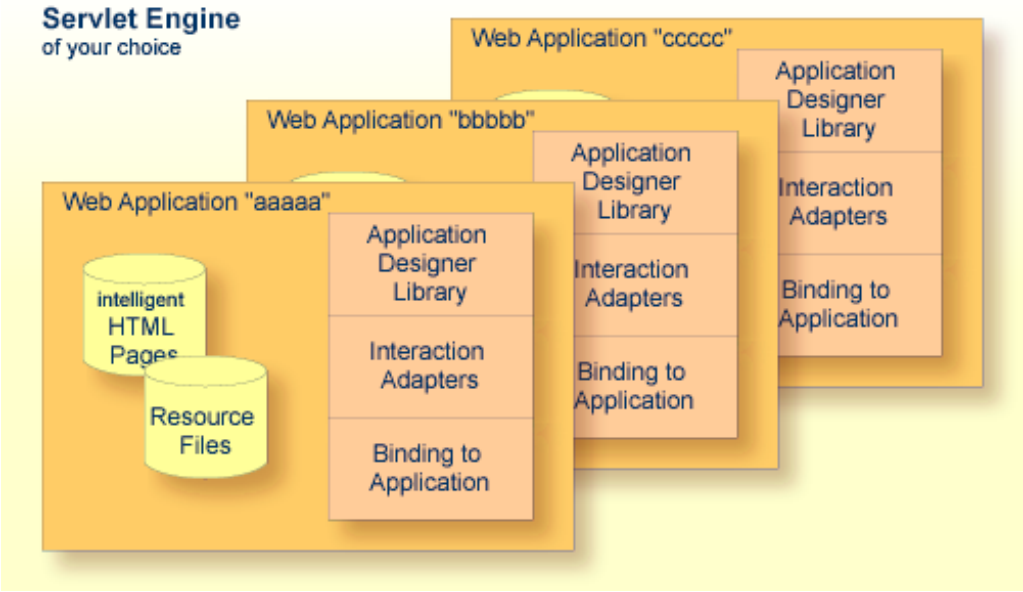
- You design the layout of the page using the WYSIWYG Layout Painter. The result is kept in an XML layout description file. The Layout Painter provides a code generator that you can use to generate an HTML page from the XML layout description. The layout definition both contains the controls used on a page and keeps information on how these controls are bound to properties and methods of a server-side Java object. The generated HTML page contains HTML and JavaScript statements - and is called “intelligent HTML page” in this documentation. The page is executed within a browser at runtime.
- You write a small Java class. This class handles all properties and methods defined within the layout definition. The class is executed in the Application Designer server environment, i.e. it does not run on the browser client but on the server side.

The class does not provide a large amount of logic but connects to existing application logic. This is the reason why it is called the “interaction adapter” in this documentation.



Server Side Architecture

The Application Designer server runtime is a web application which is based on the Java Servlet standard. It can be deployed multiple times to a servlet engine of your choice. XML layout definitions, generated intelligent HTML pages, adapter classes and other resource files are part of this web application.



You either install Application Designer as a web application inside your servlet engine and build your application modules around it, or you add and integrate the Application Designer environment into an existing web application.

In case of a multiple-deployment setup, each deployed instance of Application Designer is a self-contained, independent environment, completely separate from the other instances. With the example shown in the graphics above, you have three independent instances. Pages of the first instance are, for example, accessible via *http://host:port/aaaaa/...*, pages of the second instance via *http://host:port/bbbbb/...*, and so on.

2 Content Page - Workplace/Portal

Application Designer simplifies the creation of highly interactive browser pages. Pages that are created on the basis of Application Designer are accessible by a normal HTTP URL. It is very easy to pass parameters through the URL into the page. The semantics of the pages is not defined in any way and is left open to the application.

Typically an application consists of a couple of pages that are started from some environment applications. There are certain usage scenarios:

- You may build content pages that are responsible, for example, for editing a certain business document or a certain master data object. You can integrate these pages into an existing portal environment. In this case, the portal environment typically builds the frame around the contents: through the frame, a user accesses the functions to be executed. Application Designer pages run inside the existing portal.
- You may use the Application Designer workplace in order to embed content pages into a workplace environment. The Application Designer workplace is an Application Designer application itself that is responsible for building up a flexible function tree on the left. Content can be started from the function tree. It is possible to have multiple content pages open at the same time - this can be compared with multi document interfaces you may know from word processing systems.
- You may use Application Designer to develop your own workplace environment in which you arrange your functions to be accessed by the user in exactly the way you want.

3 Development Process

Developing, Testing and Debugging your Application

The development process consists of the following steps:

1. You create the layout with the Layout Painter.
2. You write the adapter program in a Java development environment. Optionally, you may use the Code Assistant tool which simplifies the creation of code. The compiled adapter classes are written to a certain Application Designer directory.

For better structuring large projects inside one Application Designer web application, a project management capability is available that allows you to structure your web application project into several units.

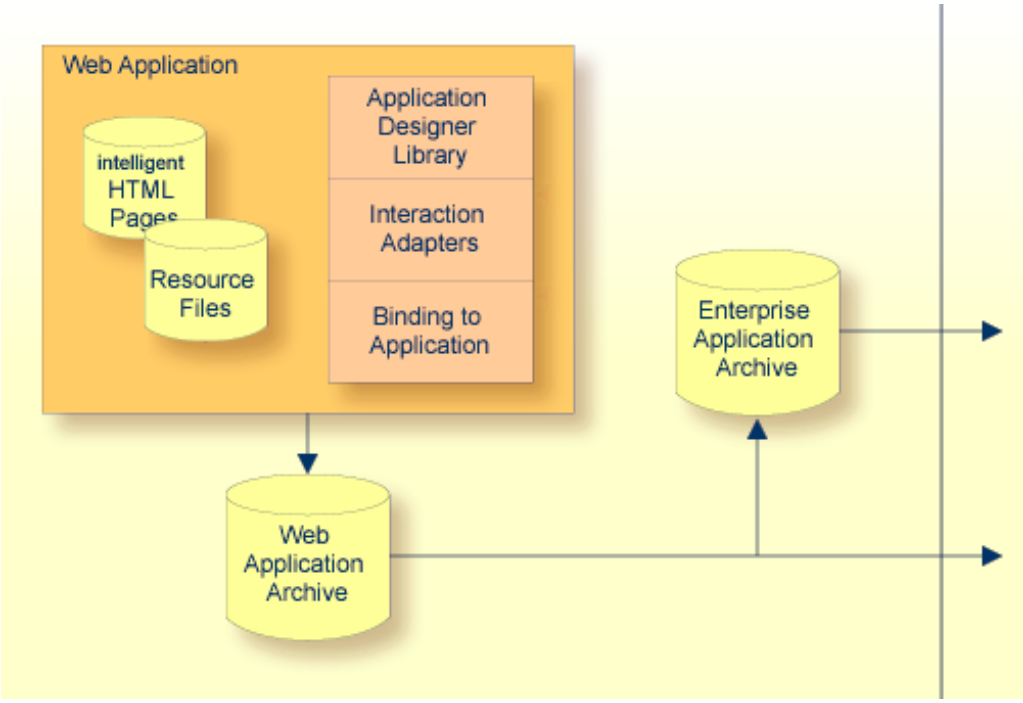
Deployment is done according to the application deployment rules: the web application around Application Designer is packaged and deployed as a *.war* file (web application archive) into web server environments or application server environments.

Debugging can be done by either connecting to the Application Designer server in remote debugging mode or by starting the Application Designer server directly inside a debugger.

Delivering your Application

Your application(s) that use Application Designer are packaged as web applications. Depending on your application's structure, Application Designer can be delivered in several ways:

- Straight delivery of web applications (*.war*) files.
- Integration of the web application (*.war*) files into enterprise application archive (*.ear*) files.


Web Application
Application Designer Library
Interaction Adapters
Binding to Application
intelligent HTML Pages
Resource Files
Enterprise Application Archive
Web Application Archive

4 Tools

Development Workplace versus Eclipse Plug-in

- **Development Workplace**
  The development workplace is a browser-based development environment. In the workplace, you can create, edit and run layouts using the different tools which help you maintain these layout definitions. From within the workplace, you can also access demo applications and documentation. For more information, see the *Development Workplace* documentation.
- **Eclipse Plug-in**
  For the development of layouts in an Eclipse environment, the Eclipse plug-in Ajax Developer is available. It supports the major features from the development workplace in an Eclipse environment. For more information, see the *Ajax Developer* documentation.

Important Tools

The most important tools that are used for the development tasks are briefly described below.

- **Layout Painter**
  The Layout Painter is an Application Designer server application, which means that the editor runs in the browser or Eclipse and the application logic runs inside the Application Designer server. In the Layout Painter, you specify the layout of your pages using an XML tree. An active WYSIWYG preview is provided in which you can test your applications.

  Detailed information is provided in:

  - *Layout Painter* in the *Development Workplace* documentation.
  - *Layout Painter* in the *Ajax Developer* documentation.
- **Layout Manager**
  While the Layout Painter allows you to edit one layout definition, the Layout Manager allows you to manage all layout definitions of a project. Mass operations like regeneration of all intelligent HTML pages of a selected project can be done with this tool.

  For detailed information, see *Layout Manager* in the *Development Workplace* documentation.
- **Code Assistant**
  The Code Assistant - like the Layout Painter - is an Application Designer server application and runs inside the browser or Eclipse. You use its code generator to speed up development of Java adapter classes. All set/get methods of referenced properties can be generated easily using this tool. The Code Assistant is integrated in the Layout Painter.

  Detailed information is provided in:

  - *Using the Code Assistant* in the *Development Workplace* documentation.

- *Using the Code Assistant* in the *Ajax Developer* documentation.

- **Literal Assistant**
  Optionally, you can define all literals inside a page layout - the “text IDs”. The text for a text ID is derived from the multi language management at runtime. The Literal Assistant is a tool for editing and translating these literals. It is integrated in the Layout Painter.

  Detailed information is provided in:

  - *Using the Literal Assistant* in the *Development Workplace* documentation.
  - *Using the Literal Assistant* in the *Ajax Developer* documentation.

- **Style Sheet Editor**
  All controls that are contained inside the Application Designer control library are rendered using a style sheet. Application Designer delivers a variety of predefined styles but also allows you to generate custom styles sheets. To simplify the creation of custom style sheets, the Style Sheet Editor is available. With this tool, you can both define the very basic style elements (main colors to be used) as well as change the style definitions of the controls on the lowest level.

  Detailed information is provided in:

  - *Style Sheet Editor* in the *Development Workplace* documentation.
  - *Style Sheet Editor* in the *Ajax Developer* documentation.

- **WAR Packager**
  To deliver your web applications built with Application Designer, you need to create a web archive (*.war* file). This file is either created by tools you might select from your development environment or by the WAR Packager: You can create the *.war* file automatically, using an easy-to-use graphical user interface.

  For detailed information, see *WAR Packager* in the *Development Workplace* documentation.

Java Development Environment

You can use a standard Java development environment to develop the adapter classes for your layouts. The Ajax Developer plug-in provides extended support for Eclipse (open source), which is one of the most popular Java IDEs. For detailed information, see the *Ajax Developer* documentation.

Alternatively, you can also use different Java development environments such NetBeans (open source) and JBuilder. However, Application Designer does not provide any enhanced support for these environments as it does for Eclipse.

Depending on the functional capabilities, you can also use the development environment for remote debugging. If your development environment does not provide a remote debugger, you can use other debugging tools such as the JSwat debugger. See *Appendix F - Using JSwat for Debugging*.