



Web Application Framework IDE Guide

Sun Java™ Studio Enterprise 7 2004Q4

Sun Microsystems, Inc.
www.sun.com

Part No. 819-0729-10
December 2004, Revision A

Submit comments about this document at: <http://www.sun.com/hwdocs/feedback>

Copyright © 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

This distribution may include materials developed by third parties. Sun, Sun Microsystems, the Sun logo and Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc.

UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Products covered by and information contained in this service manual are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright © 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés.

Cette distribution peut comprendre des composants développés par des tierces parties. Sun, Sun Microsystems, le logo Sun et Java sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Les produits qui font l'objet de ce manuel d'entretien et les informations qu'il contient sont regis par la législation américaine en matière de contrôle des exportations et peuvent être soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes biologiques et chimiques ou du nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers des pays sous embargo des Etats-Unis, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exclusive, la liste de personnes qui font l'objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites. LA

DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.



Adobe PostScript

Contents

- Before You Begin 7**

- 1. Sun Java Studio Enterprise 7 Overview 13**
 - Sun Java Studio Enterprise 7 Projects 15
 - Creating a New Sun Java Studio Enterprise 7 Project 15
 - Opening a Sun Java Studio Enterprise 7 Project 16
 - Sun Java Studio Enterprise 7 Windows 16
 - Filesystems Window 17
 - Project Window 19
 - Runtime Window 21
 - Web Application Framework Apps Window 23

- 2. Web Application Framework Apps Window Overview 25**
 - Web Application Framework Apps Root Node 27
 - Web Application Framework Application Node 28
 - Settings & Configuration Node 33
 - General Node 34
 - File Upload Node 37
 - Logging Node 40
 - Deployment Descriptor Node 43

Component Libraries Node	44
Design-Time Resources Node	45
JDBC Datasources Node	46
Web Application Framework Datasource Node Contextual Menu Commands	47
Templates Node	47
Sun Java System Identity Server Node	48
Documents Node	49
Application Classes Node	50
Java Package Folder Node	52
Module Folder Node	54

3. Web Application Framework Component Nodes 59

Page Component Node	59
Java Source Node	63
JSP Pages Node	64
JSP Node	66
Non-Visual Components Node	70
Non-Visual Component Node	73
Visual Components Node	75
Visual Component Node	78
Pagelet Component Node	80
Model Component Node	81
Java Source Node	85
Model Field Group Node	85
Model Field Node	86
Model Operation Node	88
Command Component Node	89
Java Source Node	92

4. Sun Java Studio Enterprise 7 Tool Options	93
Properties Property Sheet	94
Expert Property Sheet	96
Index	99

Before You Begin

The *Web Application Framework IDE Guide* introduces developers to the Explorer window of the Sun™ Java™ Studio Enterprise 7 2004Q4 developer environment, and various views of an application's filesystem structure as accessed and presented through the Explorer tabs.

Before You Read This Book

Before starting, you should be familiar with concepts used in building web applications using existing Java 2 Platform, Enterprise Edition (J2EE™ platform) web technologies such as servlets and JavaServlet Pages™ (JSP™ pages).

The following resources can provide additional information :

- *Java 2 Platform, Enterprise Edition Specification*
<http://java.sun.com/j2ee/download.html#platformspec>
- *The J2EE Tutorial*
<http://java.sun.com/j2ee/tutorial>
- *Java Servlet Specification Version 2.3*
<http://java.sun.com/products/servlet/download.html#specs>
- *JavaServer Pages Specification Version 1.2*
<http://java.sun.com/products/jsp/download.html#specs>

Note – Sun is not responsible for the availability of third-party Web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

How This Book Is Organized

[Chapter 1](#) provides an overview of the various parts of the Sun Java Studio Enterprise 7 development environment (hereafter referred to as IDE) and emphasizes the use of the visual tools for developing Web Application Framework applications.

[Chapter 2](#) provides an overview of the Web Application Framework tab and explores each of this tab's primary nodes

[Chapter 3](#) provides an overview of the various nodes that visually represent the major Web Application Framework components that you create in your Web Application Framework application.

[Chapter 4](#) provides an overview of the Sun Java Studio Enterprise 7 tool options.

Typographic Conventions

Typeface	Meaning	Examples
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your <code>.cvspass</code> file. Use <code>DIR</code> to list all files. Search is complete.
AaBbCc123	What you type, when contrasted with on-screen computer output	> login Password:
<i>AaBbCc123</i>	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the <i>User's Guide</i> . These are called <i>class</i> options. You <i>must</i> save your changes.
<code>AaBbCc123</code>	Command-line variable; replace with a real name or value	To delete a file, type DEL <i>filename</i> .

Related Documentation

Java Studio Enterprise documentation includes books and tutorials delivered in Acrobat Reader (PDF) format, release notes, online help, and tutorials delivered in HTML format.

Documentation Available Online

The documents described in this section are available from the `docs.sun.com`SM web site and from the Documentation link from the Sun Java Studio Enterprise Developers Source portal (<http://developers.sun.com/jsenterprise>).

The `docs.sun.com` web site (<http://docs.sun.com>) enables you to read, print, and buy Sun Microsystems manuals through the Internet.

- *Sun Java Studio Enterprise 7 Release Notes* - part no. 819-0905-10
Describes last-minute release changes and technical notes.

- *Sun Java Studio Enterprise 7 Installation Guide* (PDF format) - part no. 817-7971-10
Describes how to install the Sun Java Studio Enterprise 7 integrated development environment (IDE) on each supported platform and includes other pertinent information, such as system requirements, upgrade instructions, server information, command-line switches, installed subdirectories, database integration, and information on how to use the Update Center.
- *Building J2EE Applications* - part no. 819-0819-10
Describes how to assemble EJB modules and web modules into a J2EE application and how to deploy and run a J2EE application.
- Web Application Framework documentation (PDF format)
 - *Web Application Framework Component Author's Guide* - part no. 819-0724-10
Describes the Web Application Framework component architecture and the process to design, create, and distribute new components.
 - *Web Application Framework Component Reference Guide* - part no. 819-0725-10
Describes the components available in the Web Application Framework Library.
 - *Web Application Framework Overview* - part no. 819-0726-10
Introduces the Web Application Framework and what it is, how it works, and what sets it apart from other application frameworks.
 - *Web Application Framework Tutorial*- part no. 819-0727-10
Introduces the mechanics and techniques to build a web application using the Web Application Framework tools.
 - *Web Application Framework Developer's Guide* - part no. 819-0728-10
Provides the steps to create and use application components that can be assembled to develop an application using the Web Application Framework and explains how to deploy the application in most J2EE containers.
 - *Web Application Framework IDE Guide* - part no. 819-0729-10
Describes the various parts of the Sun Java Studio Enterprise 7 2004Q4 IDE and emphasizes the use of the visual tools for developing a Web Application Framework application.
 - *Web Application Framework Tag Library Reference* - part no. 819-0730-10
Gives a brief introduction to the Web Application Framework tag library, as well as a comprehensive reference to the tags available within the library.

Tutorials

Sun Java Studio Enterprise 7 tutorials help you understand the features of the IDE. Each tutorial provides techniques and code samples that you can use or modify in developing more substantial applications. All tutorials illustrate deployment with Sun Java System Application Server.

All tutorials are available from the Tutorials and Code Camps link off the Developers Source portal, which you can access from within the IDE by choosing Help > Examples and Tutorials.

- **QuickStart guides** provide an introduction to the Sun Java Studio IDE. Start with a QuickStart tutorial if you are either new to the Sun Java Studio IDE or want a quick introduction to a particular feature. These tutorials describe how to develop simple web and J2EE applications, generate web services, and how to get started with UML modeling and Refactoring. QuickStarts take minutes to complete.
- **Tutorials** focus on a single feature of the Sun Java Studio IDE. Try these if you are interested in the details of a particular feature. Some tutorials build an application from the ground up, while others build on provided source files, depending on the focus of the example. You can complete a tutorial in an hour or less.
- **Narrated Tutorials** use video to illustrate a feature or technique. Try a narrated tutorial for a visual overview of the IDE or an in-depth presentation of a particular feature. You can complete a narrated tutorial in a few minutes. You can also start and stop a narrated tutorial at any point you wish.

Online Help

Online help is available in the Sun Java Studio Enterprise 7 IDE. You can open help by pressing the help key (F1 in Microsoft Windows environments, Help key in the Solaris environment), or by choosing Help → Contents. Either action displays a list of help topics and a search facility.

Documentation in Accessible Formats

The documentation is provided in accessible formats that are readable by assistive technologies for users with disabilities. You can find accessible versions of documentation as described in the following table.

Type of Documentation	Format and Location of Accessible Version
Books and tutorials	HTML at http://docs.sun.com
Tutorials	HTML at the Examples and Code Camps link from the Developers Source portal at http://developers.sun.com/jsenterprise
Release notes	HTML at http://docs.sun.com

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. Email your comments to Sun at this address:

`docfeedback@sun.com`

Please include the book's title (*Web Application Framework IDE Guide*) and its part number (819-0729-10) in the subject line of your email.

Sun Java Studio Enterprise 7 Overview

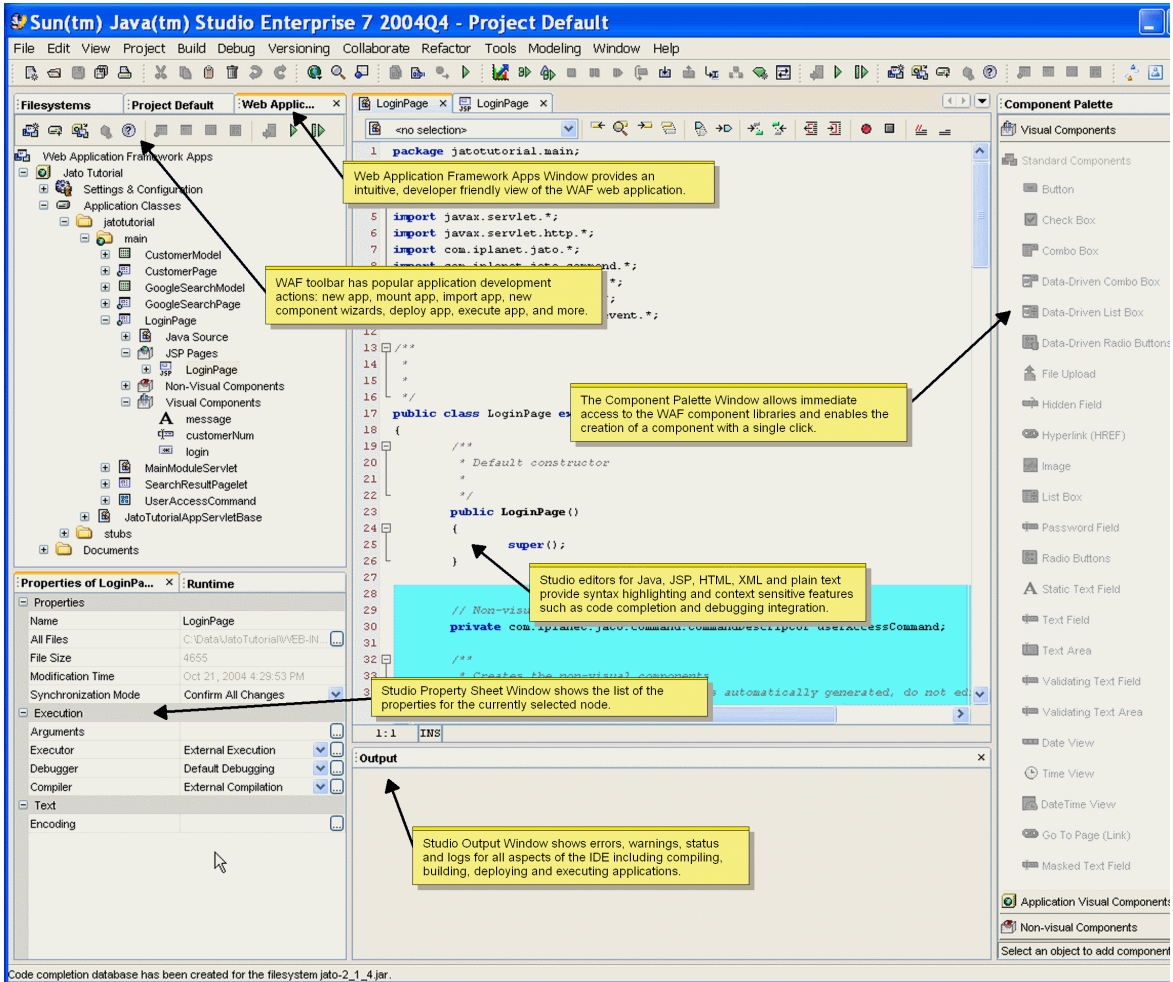
This chapter provides an overview of the various parts of the Sun Java Studio Enterprise 7 development environment (hereafter referred to as the IDE) and emphasizes the use of the visual tools for developing web applications using Sun Java Studio's Web Application Framework (formerly Sun™ ONE Application Framework and JATO).

This document focuses on a few specific dockable windows of the IDE. In previous versions of the IDE these windows made up the tabs of the Explorer window. In the latest version of the IDE these tabs are now normal IDE windows; they are listed in the Windows menu of the IDE. The windows and IDE features discussed in this document include views of your application's file system structure, application structure and runtime integration in the IDE's environment. Some of these tabs might be outside the scope of this document.

A completed version of the *JatoTutorial* application is used as a reference for demonstrating the various features of the Web Application Framework tools. If you have completed the *JatoTutorial*, you are familiar with the objects of this Web Application Framework application. If you have not completed the *JatoTutorial*, you might find it helpful to do so before proceeding with this document.

For complete documentation on the basic features of the NetBeans-based Sun Java Studio Enterprise 7 software, visit the NetBeans online documentation at: <http://www.netbeans.org/kb/using-netbeans/36/index.html>.

The following figure shows a complete view of the Sun Java Studio Enterprise 7 IDE featuring the Web Application Framework.



Sun Java Studio Enterprise 7 Projects

Sun Java Studio Enterprise 7 projects provide you with an IDE *sandbox*, which is a way to isolate different development environments that you can customize for several different development efforts. When you run the Sun Java Studio Enterprise 7 software for the first time, you are working in the default Sun Java Studio Enterprise 7 project.

For example, suppose you have an assignment that requires working with J2ME for wireless applications. You could mount all the necessary libraries and file systems, set compiler, editor, and build options, and more, that are unique for that particular project.

In another assignment, you might be working with CORBA, RMI, and rich GUI clients. This might require the mounting of several other kinds of libraries and file systems, and it might require very different build configurations.

Another scenario might be that you are tasked with maintaining a Web application framework at the company for which you are employed. In your spare time, you might work on an open source application server effort. These two projects could be placed into separate Sun Java Studio Enterprise 7 projects: *Real Job Project* and *Open Source Project*.

Note – Before you create or open a project, make sure that all Sun Java Studio Enterprise 7 invoked processes are shutdown first to ensure a quick and clean project switchover. Go to the Runtime window in the Explorer window, expand the Processes node, and terminate all running processes (right-click the process and select Terminate Process).

For additional information about Sun Java Studio Enterprise 7 projects, visit the NetBeans online documentation at:

http://usersguide.netbeans.org/gwd/gwd_project_setup.html#projects

Creating a New Sun Java Studio Enterprise 7 Project

To create a new Sun Java Studio Enterprise 7 project select the Project > Project Manager menu option, click New, and provide a project name when prompted. A new Sun Java Studio Enterprise 7 project is created and opens. You can configure the project as required.

Opening a Sun Java Studio Enterprise 7 Project

To open another Sun Java Studio Enterprise 7 project, select the Project > Project Manager menu option. From the list of projects, select the project that you want to open and click the Open button. If the Open button is not enabled when you select the project name, that project is the currently open project.



Caution – If you want to mount multiple Web Application Framework applications in the same Sun Java Studio Enterprise 7 project, be sure that all of those applications are using the same version of the Web Application Framework JAR (`jato.jar`). You might not encounter any issues at all, but there could be compilation ambiguities.

Sun Java Studio Enterprise 7 Windows

The Sun Java Studio Enterprise 7 usually has many windows docked together as tabbed windows. While the layout and visibility of these windows is fully configurable, in this document we use a layout with three windows:

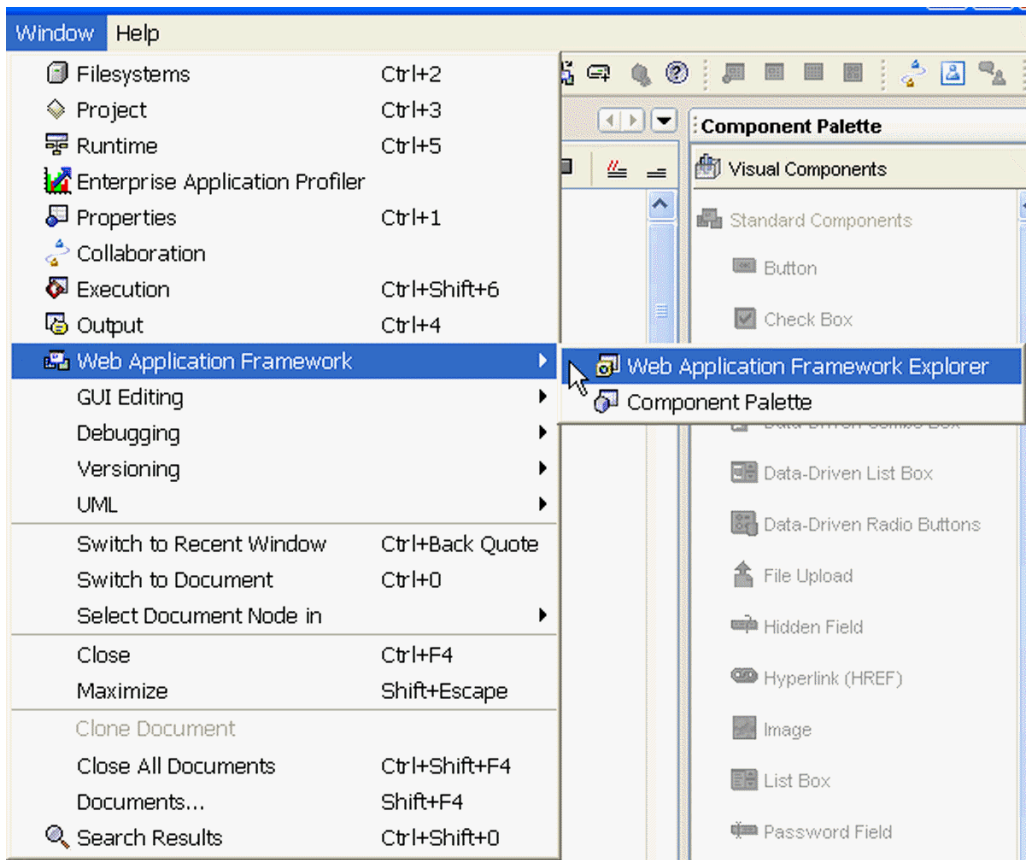
- Filesystems
- Project
- Web Application Framework Apps

at the upper left and two windows:

- Properties
- Runtime

at the lower left.

However, this can vary, as the Sun Java Studio Enterprise 7 allows for extreme layout customization. Each of these window expose a view of the Sun Java Studio Enterprise environment in a different way. See the image below which shows the Windows menu listing all the windows available in the IDE.

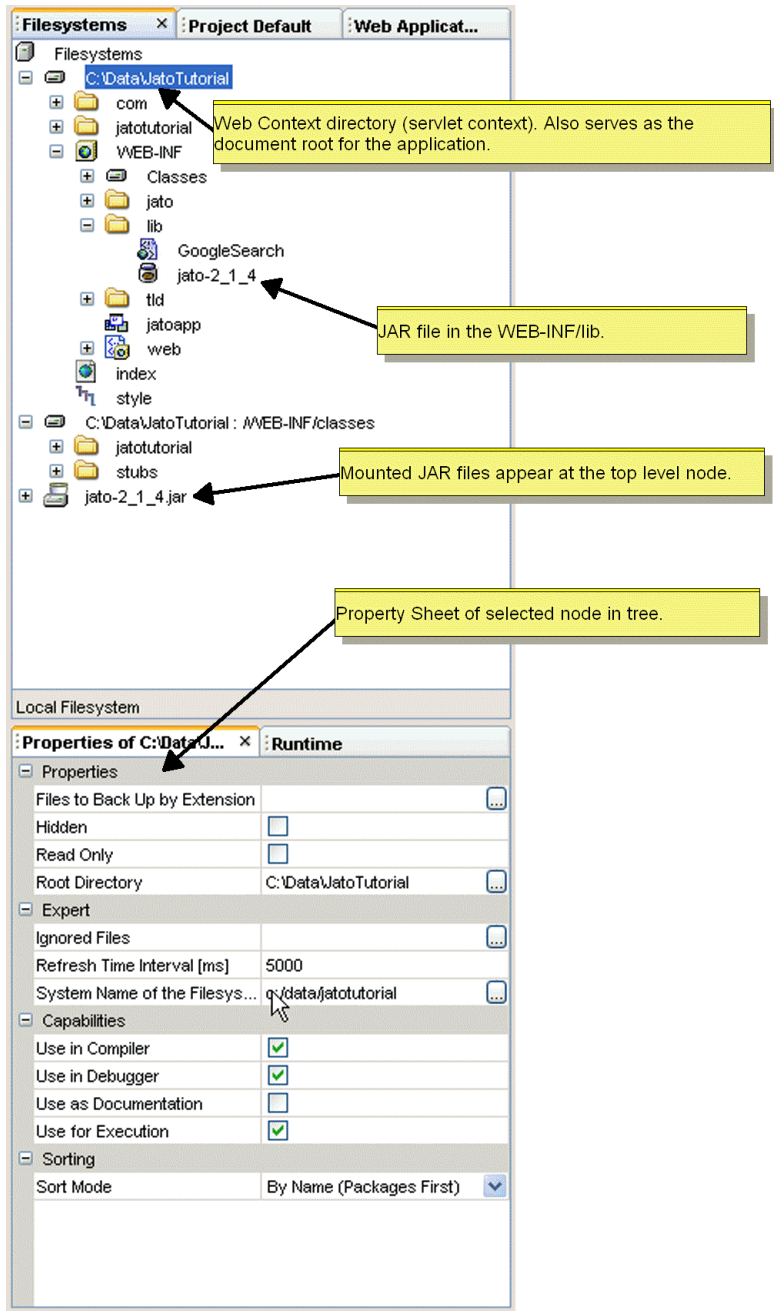


For additional information about the Sun Java Studio Enterprise 7 Explorer window, visit the NetBeans online documentation at:

http://usersguide.netbeans.org/gwd/gwd_project_setup.html#explorer

Filesystems Window

The Filesystems window of the Explorer window displays all currently mounted directories, as shown in the following figure.



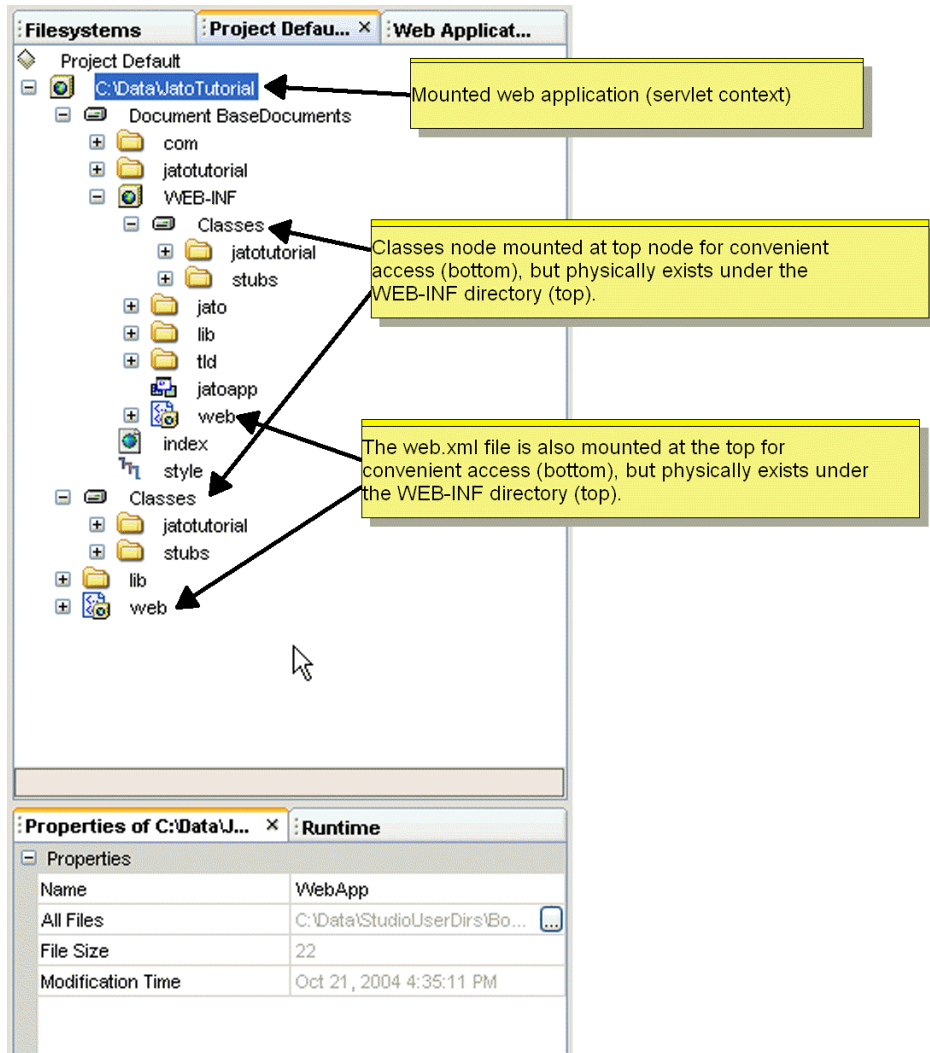
When you create a new Web Application Framework application, the Web Context directory (also known as servlet context or base Web application directory) is mounted automatically. Any JAR files located in the `WEB-INF/lib` directory of your Web Application Framework application are also mounted. Although these JAR files only physically exist in the `WEB-INF/lib` directory, they are displayed at the top level node.

As you add Web Application Framework components that require additional libraries (for example, `jaxrpc-api.jar`, and others for Web Service models) or when you manually copy JAR files to your application's `WEB-INF/lib` directory, the Sun Java Studio Enterprise 7 automatically mounts these new JAR files. The Sun Java Studio Enterprise 7 might not recognize a new JAR file immediately. This is dependent upon the refresh rate setting for your Sun Java Studio Enterprise 7 configuration.

Besides the *virtual* view of the JAR files, the Filesystems view provides the raw directory and file layout as found in your operating system's file system. As a Web Application Framework application developer, you do not need to spend much, if any, time in this view.

Project Window

For Web applications, such as a Web Application Framework application, the Project window displays a logical (flattened) view of your Web application's directory structure, as shown in the following figure.



Like the JAR files in the Filesystems window, some *buried* nodes are mounted at the top level node for convenient access. As a Web Application Framework application developer, you probably never need to view this window.

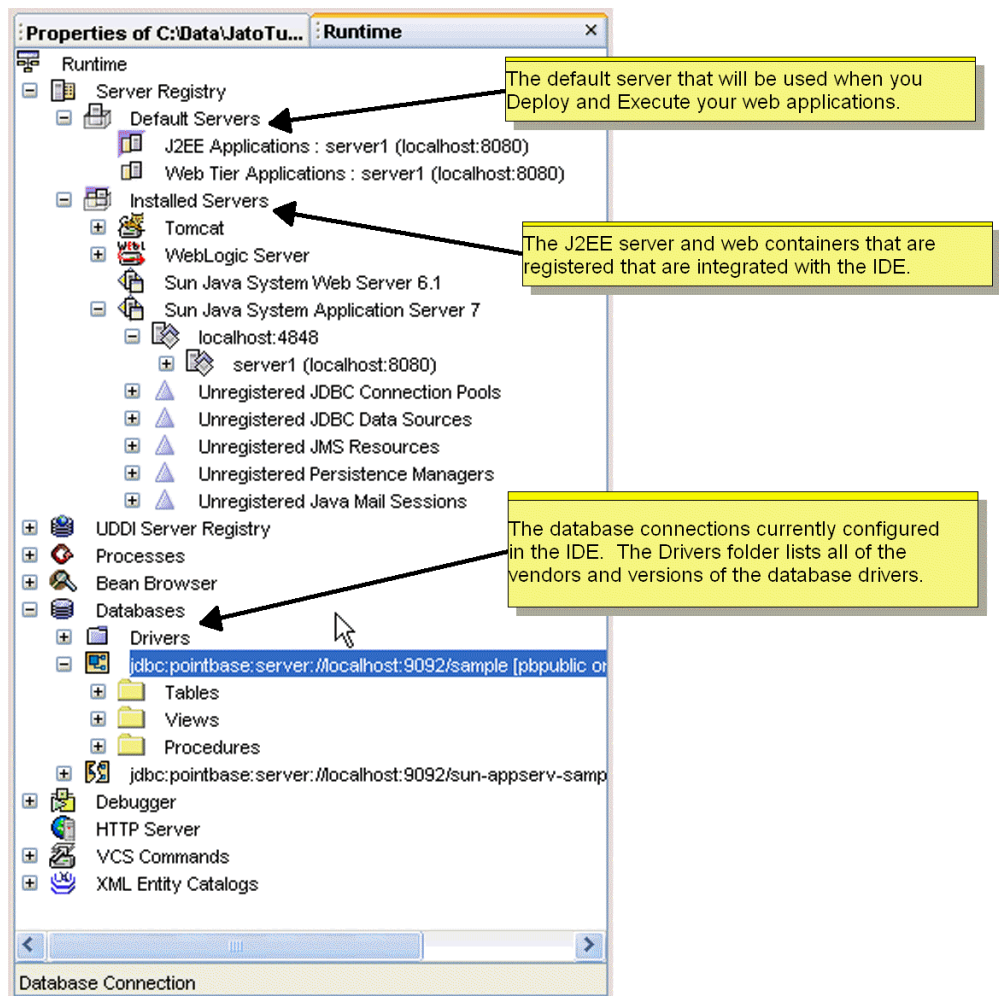
Runtime Window

The Runtime window shows various resources available in the Sun Java Studio Enterprise 7 in the form a tree node structure.

Nodes that you might interact with while developing applications (not limited to Web applications or Web Application Framework web applications) are as follows:

- Server Registry
- Processes
- Databases

The following figure shows the Runtime window.



Server Registry Node

Server Registry contains a list of all the servlet containers that can be managed and utilized from within the Sun Java Studio Enterprise 7. Tomcat and Sun ONE Application Server are two such servlet containers that come with the Sun Java Studio Enterprise 7. You might have to visit the Sun Java Studio Enterprise 7 Update Center to download and install the module if it has not already been installed by default.

Processes Node

Processes lists any processes that were launched (and are still running) by the Sun Java Studio Enterprise 7 IDE. When you compile your application, the compilation process displays in Processes until it has finished its task. When you test run your application, the servlet container process is listed here.

The servlet container remains running until you shut it down. To shut down any process launched by the Sun Java Studio Enterprise 7 IDE, go to the Processes list, right-click the desired process, and select Terminate Process.

Shut down all processes launch by the Sun Java Studio Enterprise 7 IDE before closing the Sun Java Studio Enterprise 7 IDE or switching Sun Java Studio Enterprise 7 projects. Sun Java Studio Enterprise 7 projects are managed (created, deleted, opened) from the Sun Java Studio Enterprise 7 Project menu option. Sun Java Studio Enterprise 7 projects are outside the scope of this document.

Databases Node

Databases is a list of all the RDBMS drivers and connections that have been configured in the Sun Java Studio Enterprise 7. There is a Drivers folder subnode, and potentially a list of database connections.

The Drivers folder node has a list of many vendor database driver versions. If the drivers for a particular database are available to the Sun Java Studio Enterprise 7, the Drivers folder node is enabled. Otherwise, it is disabled (a diagonal, red line through its icon). You can add new drivers by right-clicking the Drivers folder node, selecting the Add Driver action, and supplying the required driver information in the dialog box that is presented. You must also make the driver available to the Sun Java Studio Enterprise 7 by placing it in the Sun Java Studio Enterprise 7 `lib/ext` directory.

The connections, if any, can be connected (represented by a complete icon), or disconnected (represented by a broken icon). To connect a connection, right-click the connection node and select the Connect or Connect As action. You are prompted for the required connection information (username, password, and so on). To complete

the connection, the target database server must be running, so be sure the preferred database server is started and accessible. To disconnect, right-click the connection and select the Disconnect action.

You can add new connections by right-clicking the Databases node and selecting the Add Connection action. You are prompted with the database type, location, and connection information. Before attempting to create a connection to the database, be sure, as described earlier in this section ([Databases Node](#)), that the proper driver is available to the Sun Java Studio Enterprise 7.

Web Application Framework Apps Window

The Web Application Framework Apps window provides a user friendly view of your Web application. You can mount one or more Web Application Framework applications in this window. Mounting a Web Application Framework application also mounts it as a Filesystem in the Filesystems window, and as a Web module in the Project window. Within each mounted Web Application Framework application there are three top level nodes, as follows:

- Settings & Configuration
- Application Classes
- Documents

Each of these nodes is introduced in [Chapter 2, Web Application Framework Apps Window Overview](#), with details on creating and mounting Web Application Framework applications.

Web Application Framework Apps Window Overview

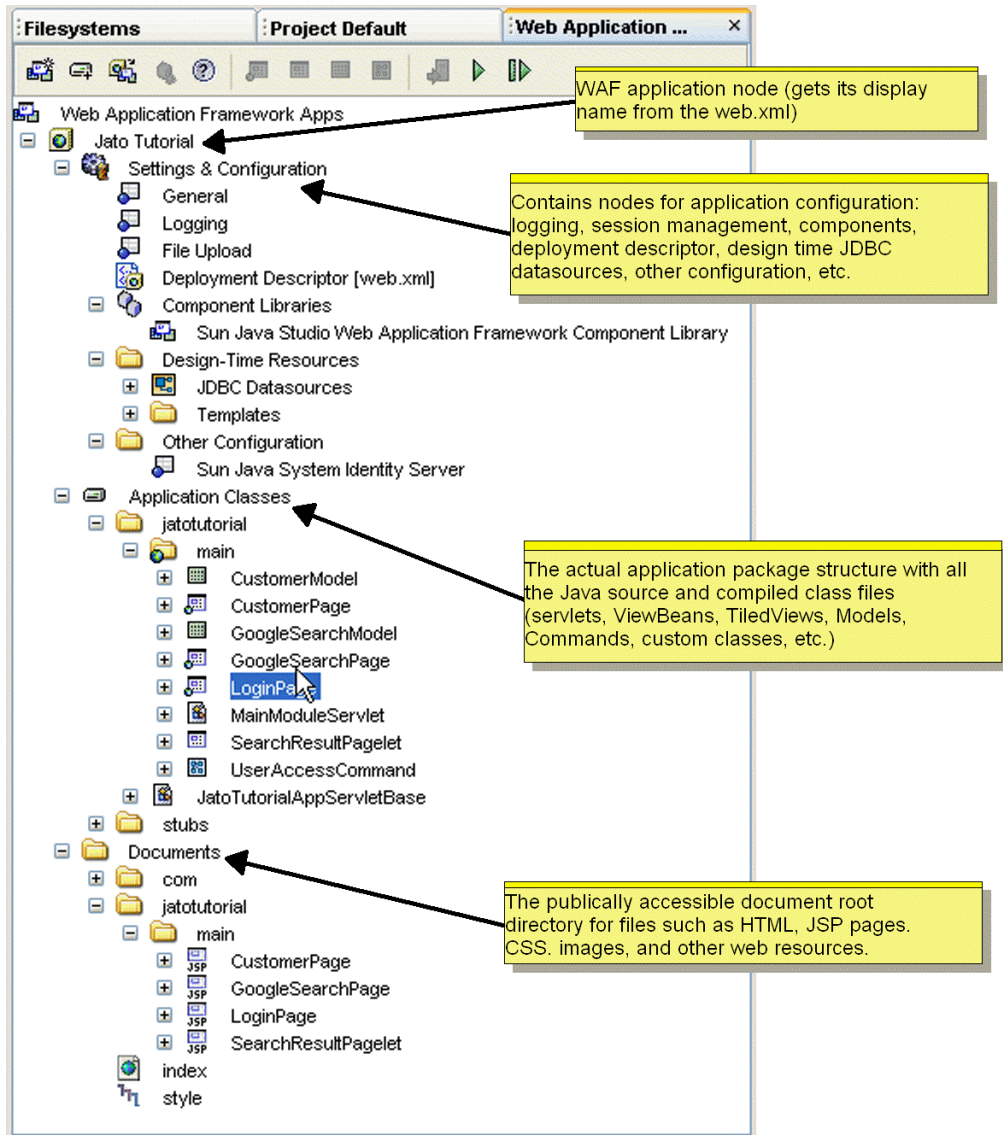
This chapter provides an overview of the Web Application Framework Apps window and explores each of this window's primary nodes.

The Web Application Framework Apps window presents your Web Application Framework application in an intuitive and logical view.

There are three primary nodes under the top level application node, as follows:

- Settings & Configuration
- Application Classes
- Documents

The following figure shows the Web Application Framework Apps window and its three primary nodes.



You will spend most of your time in the Application Classes node where you will select Web Application Framework components to configure properties and add events and custom code. This document explores each of these nodes in detail.

Web Application Framework Apps Root Node

The node at the top of the Web Application Framework Apps window is the *anchor* node under which you mount one or more Web Application Framework applications. When you have two or more Web Application Framework applications mounted, they are considered separate applications that will be deployed as such (as separate WAR files).

Web Application Framework Apps Root Node Contextual Menu Commands

When you select the Web Application Framework Apps Root Node and right-click, the contextual menu displays menu items described below:

▼ [action] New Web Application Framework App

To create a new Web Application Framework application, select the Sun Java Studio Enterprise 7 menu option File | New (or click the New toolbar button, which should be the first icon at the top left of the Sun Java Studio Enterprise 7, located just below the File menu option). The Choose Template panel is displayed. The Choose Template panel contains all of the available Sun Java Studio Enterprise 7 template wizards.

The Web Application Framework wizards can be found in the Web Application Framework node. Expand this node and select Application. Alternatively, you can click the New Web Application Framework App toolbar button found on the toolbar located at the top of the Web Application Framework window. A subset of this toolbar might also be visible on the Sun Java Studio Enterprise 7 main toolbar (just under the menu options). All of these techniques invoke the same wizard. Creating a new Web Application Framework application effectively mounts the new application.

▼ [action] Mount Web Application Framework App

If you have a Web Application Framework application (version 2.x) that was previously created, but is not mounted in the Sun Java Studio Enterprise 7, and you want to have it mounted, you can do so by clicking the Mount Web Application Framework toolbar button which is located just to the right of the New Web Application Framework Apps toolbar button.

There is also a Sun Java Studio Enterprise 7 File menu option and a right-click action on the root node of the Web Application Framework Apps window. This node is called Web Application Framework App. Using any of these techniques launches a file system browser/chooser dialog. You can navigate the file system on your computer or network to where the desired Web Application Framework application

is located. Folders that are the root of a Web Application Framework application will be displayed with the Web Application Framework application icon (three overlapping rectangles). Select the Web Application Framework application that you want and click the Mount button found in the dialog window. Your Web Application Framework application will be appropriately mounted into the Sun Java Studio Enterprise 7.

▼ [action] Download Components

Currently, this action opens a browser window to the Web Application Framework web page. The intent of this action is to target a website that would be repository of third party Web Application Framework components that could be downloaded for use in your Web Application Framework application. Such a repository does not currently exist. For more details on creating third party Web Application Framework components, consult the *Web Application Framework Component Author's Guide*.

Web Application Framework Application Node

Directly under the Web Application Framework Apps root node there will be zero, one, or more mounted Web Application Framework application nodes. Each mounted Web Application Framework application node is depicted by a yellow cube with a green globe in the center of the cube. The name of these nodes is the *display name* of the Web application. The display name is an actual element entry in the deployment descriptor file (`web.xml`) located in the `WEB-INF` directory of the Web application. The actual file system directory name might be something entirely different. This is known as the *web context name* of your Web application. It is also the *web documents root* directory.

Web Application Framework Application Node Contextual Menu Commands

This node has several contextual menu commands. You will use a few of these actions to prepare your Web Application Framework application for execution.

▼ [action] Execute

You will probably not use the Execute action for this node since this action does not directly execute any Web Application Framework page components in your Web Application Framework application.

The Execute action launches a browser window and attempts to load the `index.html` file in the documents root of your Web application. If a file named `index.html` does not exist, it loads the *welcome* file that is declared in the deployment descriptor file (`web.xml`) located in the `WEB-INF` directory of your Web application.

A default *welcome* file is created when you create a Web Application Framework application. It is a simple HTML file with a standard welcome message. This default welcome page is named `index.html` and is located in the Documents folder of your Web application. You can customize this file, but it is not a mandatory component of a Web Application Framework application.

If there is no `index.html` file in the documents root directory, and there is no *welcome* file declared in the deployment descriptor, a standard directory listing of the documents root is displayed in the browser window.

The `index.html` or *welcome* page can be configured to redirect to a particular Web Application Framework page component in your Web Application Framework application (login page, main menu page, and so on).

▼ [action] Execute (Force Reload)

This performs the same behavior as the Execute action, except that it forces the application to be redeployed to the target server, and the server to be restarted. This ensures that any changes are picked up, rather than using the Web application components that might be in memory. If the application needs to be compiled, it compiles (compile all) the application as needed before it is deployed and executed.

▼ [action] Deploy

The deploys the Web application to the target server where it can be executed. Use this action when you have made a change to the Web application and want to perform a test run. If the application needs to be compiled, it will compile (compile all) the application as needed before it is deployed.

▼ [action] Add Component Library

If you have created a custom or acquired a third party Web Application Framework component library, this action allows you to navigate to its file system location and select it. The component library will be copied to your application's `WEB-INF/lib` directory and the IDE mounts it. The library must be a Web Application Framework component library. This action is expecting a certain configuration file (`complib.xml`) contained within the JAR file that declares it to be such. For more details on creating third party component libraries, consult the *Web Application Framework Component Author's Guide*.

It might take the Sun Java Studio Enterprise 7 some time to recognize the new library file addition. This time interval is a property of the Filesystems root node in the Filesystems window. You can configure this to be a shorter interval if you choose, but do not make it so frequent that the IDE performance is hindered. Alternatively, you can manually force the Sun Java Studio Enterprise 7 to refresh its folder state by right-clicking the folder node that is the parent of the new file addition (`WEB-INF/lib` in this case), and select the Refresh Folder action.

▼ [action] Compile

This compiles files in the current directory that are new or have changed since the last compile. The up-to-date check is done by comparing timestamps between the source (`.java`) and products (`.class`) of the compile. This command does not compile the files in subfolders.

▼ [action] Compile All

This compiles only those files that are new or have changed since the last compile, including the files in subfolders.

▼ [action] Build

This deletes the `.class` files in the folder and recompiles the source files. This command does not remove `.class` files or compile source files in subfolders.

▼ [action] Build All

This deletes all `.class` files within a folder and its subfolders and then compiles all files within the folder and subfolders.

▼ [action] Identify Modules

This searches through the `web.xml` file for the element entries that identify a particular package as a module package. It is a refresh action for module identification.

▼ [action] Rename

This changes the *display name* of your Web application. The display name is an actual element entry in the deployment descriptor file (`web.xml`) in the application's `WEB-INF` directory.

▼ [action] Export WAR File

This packages your application using the file name you specify and copies to the location you choose. Use this action when you need to create a deployable WAR file for your application so you can run it in a servlet container external to the Sun Java Studio Enterprise 7 (for production deployment, to test run in another container, or to send to another developer for testing).

▼ [action] Unmount Application

This unmounts (removes) the Web Application Framework application from the Sun Java Studio Enterprise 7 environment (Filesystems, Project and Web Application Framework Apps tabs). It does not actually delete it from disk.

▼ [action] Tools

This provides access to standard Sun Java Studio Enterprise 7 tools and is outside the scope of this document.

▼ [action] Properties

This displays the property sheet for the selected node in a new window.

Web Application Framework Application Node Properties

▼ [property] Content Language

This is the default content type for pages that are executed in the Web application. The content type can be changed programmatically as needed for specific pages.

▼ [property] Context Root

This is the servlet context—the first piece of the URL following the `server:port`.

For example, `http://<server:port>/<contextroot>`

This context root is populated with the value you supplied for the *web context name* in the Web Application Framework application wizard. You probably will not need to modify this property.

▼ [property] Name

This is the *display Name* of the Web application. The display name is an actual element entry in the deployment descriptor file (`web.xml`) located in the `WEB-INF` directory of the Web application.

▼ [property] Template

This determines whether this node is a template or not (*True/False*).

▼ [property] Web Module Group

This identifies the path of the web module group file to which the Web Application Framework application belongs. See the IDE's online help for more information on creating web module groups.

▼ [property] Extra Files

This is used by the Export WAR File action. You can specify any additional files (that might not be in the Web applications directory structure) to be included in the generated WAR file.

▼ [property] Filter

This is used by the Export WAR File action and can be used to exclude certain types of files from the generated WAR file. It is common to exclude the Java source files (.java) from the WAR file so that the application source code is not published to the production server.

▼ [property] Debugger

The Tools | Options menu opens the Options window. The debugger types are listed under the Debugging and Executing node. These debugger types are the possible options for this property. You can create a custom debugger type based upon one of the current types if required.

▼ [property] Executor

The Tools | Options menu opens the Options window. The executor types are listed under the Debugging and Executing node. These executor types are the possible options for this property. You can create a custom executor type based upon one of the current types, if required.

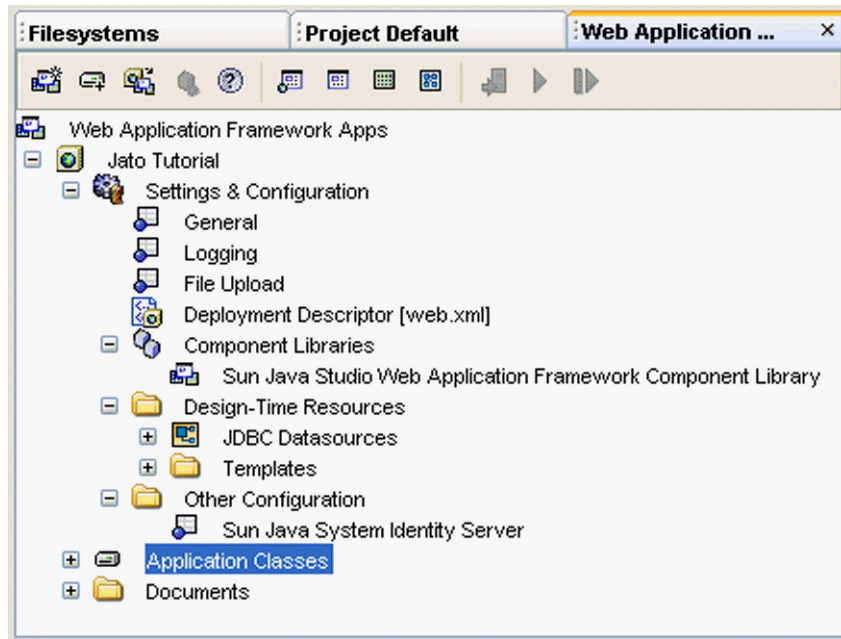
▼ [property] Target Server

This determines which servlet container will handle the execution of the Web application. The default for this property is *Default Web Server*. The default Web server is configured in the Runtime window under the Server Registry node. Only servers that are registered in the Sun Java Studio Enterprise 7 can be selected for this property. You can change the server that handles the execution of a single Web application without the need to change it globally for all other Web applications.

Settings & Configuration Node

The Settings & Configuration node has many resources for configuring the design time and runtime environment of your Web Application Framework application.

The following figure shows the Settings & Configuration node.



Settings & Configuration Node Contextual Menu Commands

▼ [none]

Settings & Configuration Node Properties

▼ [none]

Settings & Configuration Subnodes

The Settings & Configuration node has five subnodes, as follows:

- General
- Logging
- Deployment Descriptor
- Component Libraries
- Design-Time Resources
- Other Configuration

General Node

The General Node is an area for miscellaneous type Web application configuration.

General Node Contextual Menu Commands

- ▼ [none]

General Node Properties

The General node currently has two properties that control the runtime behavior of the applications.

- ▼ [property] Generate Unique URLs Node

When set to *True*, this ensures that, during your design/test cycles, each test run launches the browser and sends a unique URL to guarantee that your browser does not use a cached version of your page. This is accomplished by appending a named-value pair that has a unique ID per test run. The default value is *false*, which means do not use unique URLs.

- ▼ [property] Strict Session Timeout Node

When set to *True*, this forces you to implement session timeout handling code for your Web Application Framework application. Failure to do so results in session timeout exception messages being displayed in the browser when successive test runs are attempted. To avoid these messages, you must ensure that every browser instance sharing the same process space is closed (this might also include mail clients, as is the case with Netscape).

The following figure shows the *friendly* HTTP session timeout message displayed in the browser.

Application Error

The HTTP session has timed out

Notes for application developers:

- To prevent users from seeing this error message, override the `onSessionTimeout()` method in the module `Servlet` and take action specific to the application
 - To see a stack trace from this error, see the source for this page
-

To see the actual stack trace of an HTTP session timeout, view the HTML source of the message.


```
<!-- Exception stack trace -->
<!--
javax.servlet.ServletException: This session has timed out
at
com.iplanet.jato.ApplicationServletBase.onSessionTimeout(ApplicationServletBase.java:1222)
at
com.iplanet.jato.ApplicationServletBase.fireSessionTimeoutEvent(ApplicationServletBase.java:1079)
at
com.iplanet.jato.ApplicationServletBase.fireSessionEvents(ApplicationServletBase.java:834)
at
com.iplanet.jato.ApplicationServletBase.processRequest(ApplicationServletBase.java:609)
at
com.iplanet.jato.ApplicationServletBase.doPost(ApplicationServletBase.java:474)
)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:760)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
...
```

The other option is to implement session timeout handling logic that redirects the user to the login page. To implement the session timeout handling you want for your Web Application Framework application, override the `onSessionTimeout` event in your application's application servlet class

(`jatotutorial.JatoTutorialAppServletBase` in the tutorial example). For more details on handling session timeouts, consult the *Web Application Framework Developer's Guide*.

When the Strict Session Timeout property is set to *false* (the default), session timeouts are ignored and the test run is completed as requested. Automatic session timeout handling is accomplished by using a context parameter (`jato:enforceStrictSessionTimeout`) in the deployment descriptor (`web.xml`) which is passed into the `ModuleServlet` class at runtime, as shown in the following figure.

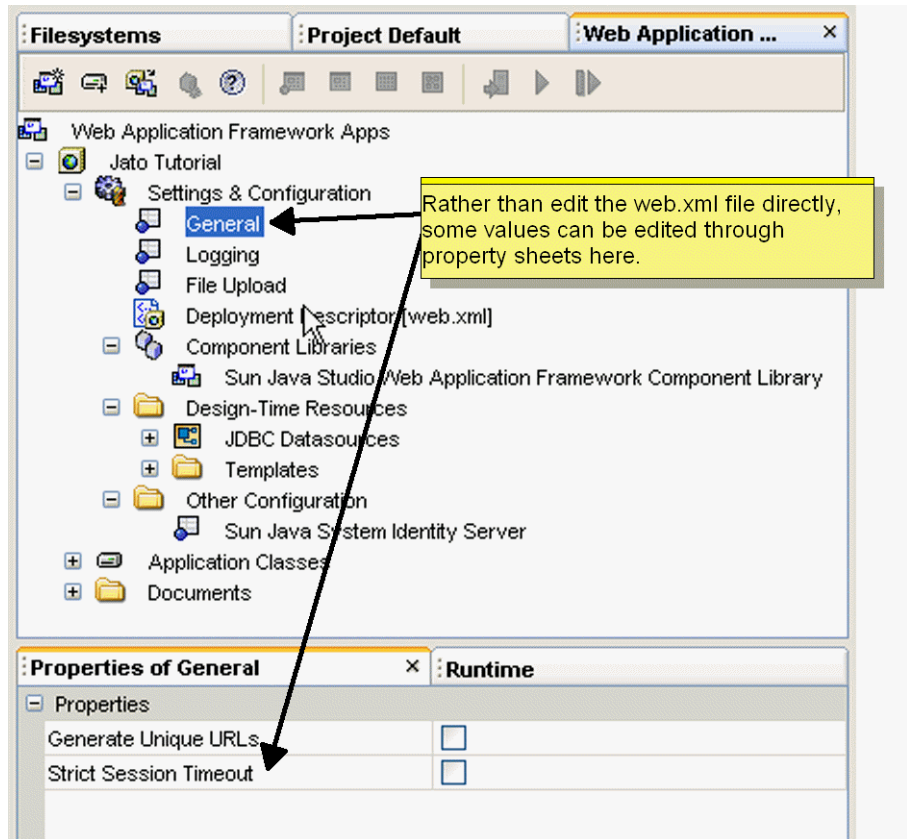
```
<web-app>
  <icon>
    <small-icon>sun logo</small-icon>
  </icon>
  <display-name>Jato Tutorial</display-name>
  <context-param>
    <param-name>jato:jatotutorial.main.*:moduleURL</param-name>
    <param-value>../main</param-value>
  </context-param>
  <context-param>
    <param-name>jato:enforceStrictSessionTimeout</param-name>
    <param-value>true</param-value>
  </context-param>
```



The context-param as seen in the raw web.xml file (deployment descriptor)

If this parameter is *false* or *does not exist*, then the Web Application Framework application handles the session timeout by ignoring it and displaying a message at the bottom of the resulting page notifying the developer to explicitly handle session timeouts.

The following figure shows the Strict Session Timeout property setting.



File Upload Node

The file upload node has several properties that enable your application to perform file uploads. This feature requires your application to be run in a Servlet v2.3 compliant servlet container.

File Upload Node Contextual Menu Commands

▼ [none]

File Upload Properties

For more details, see `com.ipplanet.jato.MutipartFormServletFilter` javadocs.

▼ [property] Auto Delete Temp Files

When set to *true*, the filter will mark temporary files for deletion when the VM exits. This setting might be used as a fallback strategy, but there is no guarantee that the container's VM will exit gracefully and delete these files. Furthermore, creating and marking hundreds or thousands of files for deletion will certainly add to the memory burden of the container. Instead, applications should delete temporary files when they are through with them. Note that deleting files from the container might require granting the *delete file* privilege in the container's security policy.

▼ [property] Enable File Upload Servlet Filter

When set to *True*, the remaining uneditable properties will become enabled (editable).

▼ [property] File Size Limit

This is the maximum byte size of uploaded file content. For example, if a user uploads three files in one request, each one of the files might not be larger than this limit. Note that this limit is a soft limit, meaning that if uploaded content exceeds this limit, the content will be discarded but the request will still proceed normally, allowing for handling of the size violation by the application. This limit defaults to 1 MB, but developers should customize it to be as small as possible for their applications.

▼ [property] File Size Limit (Hard)

This is the maximum byte size of uploaded file content before an error occurs and normal request processing is stopped. For example, if a user uploads three files in one request, if one or more of the files exceeds this limit, all files will be discarded and the filter will signal an error condition. Upon a hard size limit violation, the filter will issue an HTTP redirect to an error handler URL.

The redirect URL can be configured via the *Request Failure Redirect URL property*. If no redirect URL has been specified, the filter throws an exception to immediately end the request. This limit defaults to 2 MB, but developers should customize it to be as small as possible for their applications.

▼ [property] Request Failure Redirect URL

This is the URL to redirect the client to if the file upload size limit is exceeded, or parsing of the request fails. This parameter might be an arbitrary URL either within or outside of the current application. The URL is sent as an HTTP 302 redirect.

▼ [property] Request Size Limit

This is the maximum byte size of the entire incoming request. If a request violates this limit, request processing stops and the input stream is discarded. The filter then handles the violation as it would for a content size hard limit violation. This limit defaults to 4 MB, but developers should customize it to be as small as possible for their applications.

▼ [property] Temp File Directory

This is the directory in which to create temporary files. If this parameter is not specified, the JDK's default temp directory will be used.

▼ [property] Use Temp Files

When set to *True*, the filter saves uploaded file content into temporary files in the temp directory. Otherwise, uploaded file content is stored in memory only in multipart content objects.

▼ [property] Default Character Encoding

Optional character encoding default used only if a value has not been set for a `ServletRequest` scoped attribute named by constant `CHARACTER_ENCODING_OVERRIDE_ATTRIBUTE_NAME` and assigned a string value of a supported character encoding.

▼ [property] Use Request Character Encoding

When set to *true*, the `ServletRequest.getCharacterEncoding()` value is used for decoding only if a value has not been set for a `ServletRequest` scoped attribute named by constant `CHARACTER_ENCODING_OVERRIDE_ATTRIBUTE_NAME`.



Caution – Temporary files should be used, since reading files into memory can cause serious production scalability problems. This parameter defaults to *True*.

Logging Node

The Logging Node enables very basic and easy to use tracing in your Web Application Framework application. This feature leverages the behaviors of the `com.ipplanet.jato.util.Log` class. Context parameters in the deployment descriptor file are used by Web Application Framework to enable/disable logging automatically. The Web Application Framework has some built-in log statements for tracing and debugging, but it is up to you to add logging with the proper log levels to your own application code.

Logging Node Contextual Menu Commands

- ▼ [none]

Logging Node Properties

The properties of the Logging node all affect the deployment descriptor file (`web.xml` file in the Web application's `WEB-INF` directory). This allows you to change the logging behavior without recompiling the application.

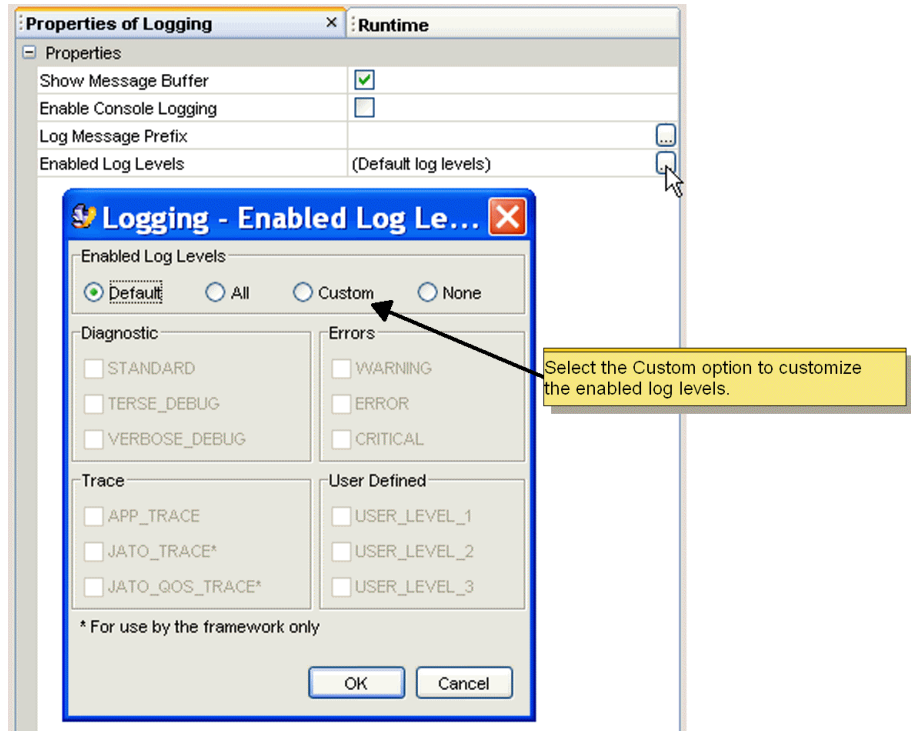
- ▼ [property] Enable Console Logging

This controls whether the output of `com.ipplanet.JATO.Log` CLASS will be sent to the server console in addition to the standard servlet context log (Output window in the Sun Java Studio Enterprise 7 or the console window for the server).

- ▼ [property] Enabled Log Levels

This allows the developer to enable a *default* subset of the log levels, *all* of the levels, *none* of the levels, or a *custom* subset of the log levels.

The following figure shows the Enabled Log Level Property Editor.



The Enabled Log Level editor can be displayed by clicking the ellipses button in the Enabled Log Levels property sheet value box.

```
2003-08-07 14:07:19 [JATO_TRACE] Servlet[jatotutorial_main@3098834]: Enabled
log levels: MANDATORY | STANDARD | VERBOSE_DEBUG | TERSE_DEBUG | JATO_TRACE |
JATO_QOS_TRACE | APP_TRACE | WARNING | ERROR | CRITICAL | USER_LEVEL_1 |
USER_LEVEL_2 | USER_LEVEL_3
```

```
2003-08-07 14:07:19 [JATO_TRACE] Servlet[jatotutorial_main@3098834]: Setting
parameter "jato:echoLogToSystemOut" = "true" (java.lang.Boolean)
```

```
2003-08-07 14:07:19 [JATO_TRACE] Servlet[jatotutorial_main@3098834]: Setting
parameter "jato:jatotutorial.main.*:moduleURL" = "../main" (java.lang.String)
```

```
2003-08-07 14:07:19 [WARNING] Servlet[jatotutorial_main@3098834]: The servlet
"jatotutorial_main" is NOT enforcing strict session timeouts. Be sure to turn
on strict session timeout handling or implement the onSessionTimeout() event
before putting this application into production.
```

```
2003-08-07 14:07:19 [JATO_TRACE] Excluded method
"jatotutorial.main.LoginViewBean.beginChildDisplay" from registration
```

```
2003-08-07 14:07:19 [JATO_TRACE] Excluded method
"jatotutorial.main.LoginViewBean.endChildDisplay" from registration
```

▼ [property] Log Message Prefix

This is a customizable string that identifies a log message that is being traced via the logging API built into the framework. This prefix string is prepended to the trace statements. For example, if the string `*>*>*` were entered as the prefix, then a trace statement would appear as follows:

```
2003-08-07 14:07:19 *>*>* [JATO_TRACE] Excluded method
"jatotutorial.main.LoginViewBean.beginChildDisplay" from registration
```

▼ [property] Show Message Buffer

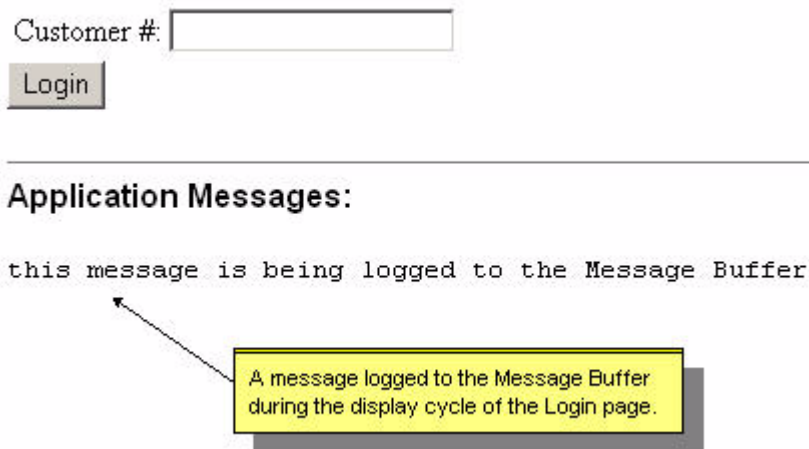
Messages logged using the `appMessage` method are written to the *message buffer*. The contents of the message buffer are written to the end of the HTML page being displayed.

Following is an example of a message being logged to the message buffer using the `appMessage` method in the `beginComponentDisplay` event of the `LoginViewBean` class.

```
public void beginComponentDisplay(DisplayEvent event)
    throws ModelControlException
{
    appMessage("this message is being logged to the Message Buffer");
}
```

The `appMessage` method is convenient for quick troubleshooting of your applications. Most often this method would be used in `ViewBean` and `TiledView` classes and is an instance method of all the container view classes. This is just one of many places this logging method can be implemented.

The following figure shows the result of this message buffer logging.



If `Show Message Buffer` is enabled, the contents of the message buffer are displayed. Otherwise, the buffer is ignored. This makes it simple to enable these messages in the development environment and disable in the production environment.

Deployment Descriptor Node

The Deployment Descriptor node is just a link to the `web.xml` file in the application's `WEB-INF` directory. This file is what identifies the application as a Web application. The deployment descriptor file contains various configurations that the servlet container uses to manage your Web application (not just Web Application Framework web applications) at runtime. The deployment descriptor contains information such as display name, context parameters, servlet mappings, taglib declarations, and much more. The Web Application Framework tools allow the

novice Web application developer to ignore the tedious details of configuring this file properly. However, the J2EE savvy developer can manipulate this file either directly by opening the file in a text or XML editor, or by using the property sheet in the Sun Java Studio Enterprise 7 when this node is selected.

Deployment Descriptor Node Contextual Menu Commands

▼ [action] Edit

This opens the deployment descriptor in the Sun Java Studio Enterprise 7 editor. Caution should be exercised when modifying this document. Some of the entries are managed by the Web Application Framework tools. Improper modification of such entries could prevent the tool from recognizing those entries properly.

Deployment Descriptor Node Properties

All of the properties directly map to the element entries of the deployment descriptor. The Web Application Framework tools module does not add any specific properties above and beyond what is already provided by the Sun Java Studio Enterprise 7 Web module. See the Web module documentation for more details concerning the deployment descriptor file node properties.

Component Libraries Node

The Component Libraries node lists all of the Web Application Framework component library JAR files that are mounted in the `WEB-INF/lib` directory of your Web Application Framework application. Every Web Application Framework application contains the Web Application Framework Standard Component Library (WAF SCL).

You might eventually create your own reusable Web Application Framework component library, or you might purchase one from a third party component vendor. When you place the component library JAR file in your Web Application Framework application's `WEB-INF/lib` directory, the Sun Java Studio Enterprise 7 recognizes and mounts the new library, and it is listed under this node.

This list of component libraries are for quick referencing of which component libraries are accessible in your Web Application Framework application and cannot be modified in any way. For more details on Web Application Framework component authoring, consult the *Web Application Framework Component Author's Guide*.

Component Libraries Node Contextual Menu Commands

▼ [action] Add Component Library

This is identical to the action with the same name for the Web Application Framework App Base Folder Node (see [\[action\] Add Component Library](#) above).

Component Libraries Node Properties

▼ [none]

Component Libraries Subnodes

All of the subnodes of the Component Library node are either the Web Application Framework Standard Component Library (included with the Web Application Framework installation), third party Web Application Framework component libraries that you have obtained and add to your Web Application Framework application, or custom Web Application Framework component libraries that you have created and added to your Web Application Framework application. For more details on Web Application Framework component authoring, consult the *Web Application Framework Component Author's Guide*.

Design-Time Resources Node

This contains two subnodes: the JDBC Datasources node and the Templates node.

Design-Time Resources Node Contextual Menu Commands

A new JDBC datasource can be created by right-clicking the JDBC Datasources node and selecting Add Datasource. Once a datasource is created, it can be deleted, but it cannot be modified.

Design-Time Resources Node Properties

▼ [none]

Design-Time Resources Subnodes

The Design-Time Resources node contains two subnodes: the JDBC Datasources node and the Templates node.

JDBC Datasources Node

The JDBC Datasources node contains a list of all the JDBC datasources you created in your Web Application Framework application using the JDBC Datasource wizard. These datasources are only used at design-time to gather schema data for the target databases so that you are able to select tables and columns and stored procedures when building JDBC SQL table and stored procedure models using the wizards. These datasources are not involved in how a database connection is created or obtained by the servlet container and/or your Web Application Framework application during runtime. The appropriate runtime JDBC and JNDI configurations must be performed using the tools provided with your target production servlet container.

JDBC Datasources Node Contextual Menu Commands

▼ [action] Add JDBC Datasource

This creates a new Web Application Framework datasource based upon the database connections that are configured in the Runtime window under the Databases node. Make sure there is a database connection created and the database server is running and accessible before you create a new Web Application Framework datasource.

JDBC Datasources Node Properties

▼ [none]

JDBC Datasources Subnodes

The only subnodes of the JDBC Datasources node will be the actual Web Application Framework datasources you create.

Web Application Framework Datasource Node Contextual Menu Commands

▼ [action] Delete

Once you create a Web Application Framework datasource, the only thing you can do to it is delete it. If you need to modify a Web Application Framework datasource, you must delete it and recreate it.

Web Application Framework Datasource Node Properties

▼ [none]

Templates Node

The Templates node is a storage location for all templates that the page/pagelet component wizards use to generate new JSP page/pagelet files. You can add a custom JSP page/pagelet to the list of templates.

Templates Node Contextual Menu Commands

The Templates node is presented as a typical Java package node. It has all of the folder actions that any other folder node would have.

Templates Node Properties

The Templates node is presented as a typical Java package node. It has all of the folder properties that any other folder node would have.

Templates Subnodes

The subnodes of the Templates node can be additional levels of folders and the actual JSP page/pagelet templates.

Sun Java System Identity Server Node

The Sun Java System Identity Server (formerly Sun ONE Identity Server) node has several properties that enable you to configure your application to use Identity Server Security features. These properties are only applicable when the application is deployed using the Sun Java System Application Server (formerly Sun ONE Application Server).

Sun Java System Identity Server Node Contextual Menu Commands

▼ [none]

Sun Java System Identity Server Node Properties

The Security Constraints, the Security Roles, and the Mapped Security Roles properties are the same properties displayed on the node for the deployment descriptor file (`web.xml` in the `WEB-INF` directory of the Web application). These properties are shown on the Sun Java System Identity Server node for convenience.

▼ [property]Enable Identity Server Security

This property determines whether the web module is enabled to use the Identity Server Security features or not (True/False). The default value is set to False.

▼ [property]Security Constraints

The Security Constraint property shows the number of security constraints defined for the web module. A security constraint is a mapping of a web resource to a group of roles or a transport requirement. The default value is No Security Constraints. To specify the security constraints, click the Security Constraints property value field, then click the ellipses button to display the Security Constraints property editor.

▼ [property]Security Roles

The Security Roles property shows the number of security roles that are defined for the web module. A role is an abstract logical grouping of users that is defined by the application developer or assembler. Security roles can either allow or deny access to different resources within the web module. The default value is No Security Roles. To specify security roles, click the Security Roles property value field, then click the ellipses button to display the Security Roles Property Editor.

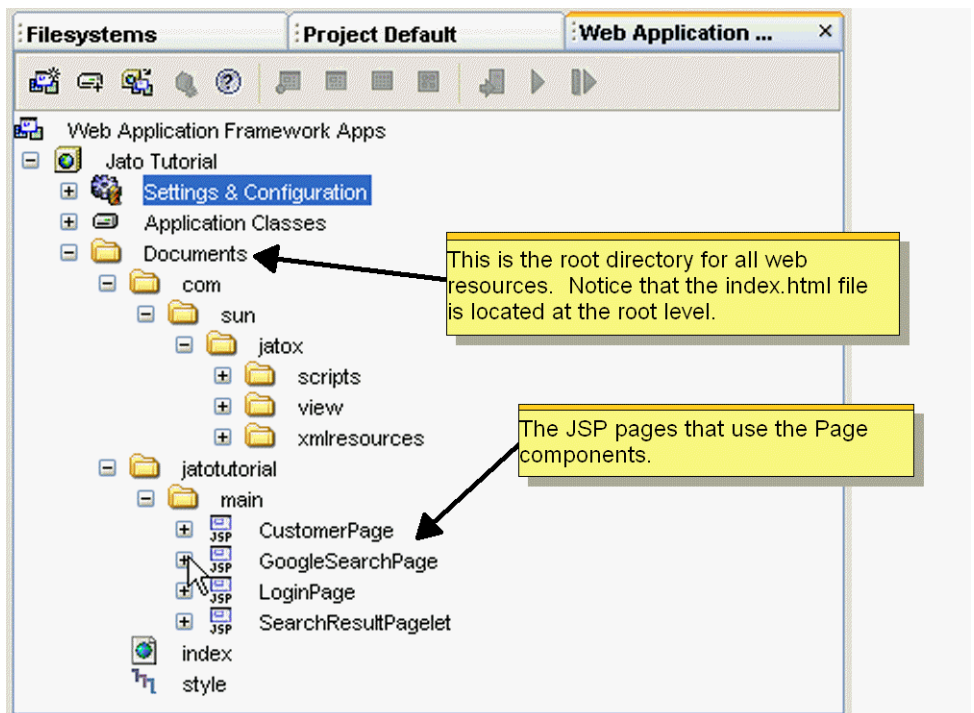
▼ [property]Mapped Security Roles

The Mapped Security Roles property shows the roles that are mapped to principals (users) or groups in the currently active realm of the web module. This property maps logical J2EE roles to Sun Java System Application Server groups and/or principals. The default value is 0 Roles. To map roles to principals or groups, click the Mapped Security Roles value field, and then click the ellipses button to display the Mapped Security Roles editor.

Documents Node

The Documents node is the Web document root that contains Web resources such as HTML, JSP, CSS, images, and so on.

The following figure shows the Documents Node, the root directory for all Web resources.



Notice that Web Application Framework JSP pages are structured in a directory layout parallel to the packages of the page and pagelet components in the application. This is a recommended convention, but the developer is free to locate the JSPs anywhere within the servlet context directory structure.

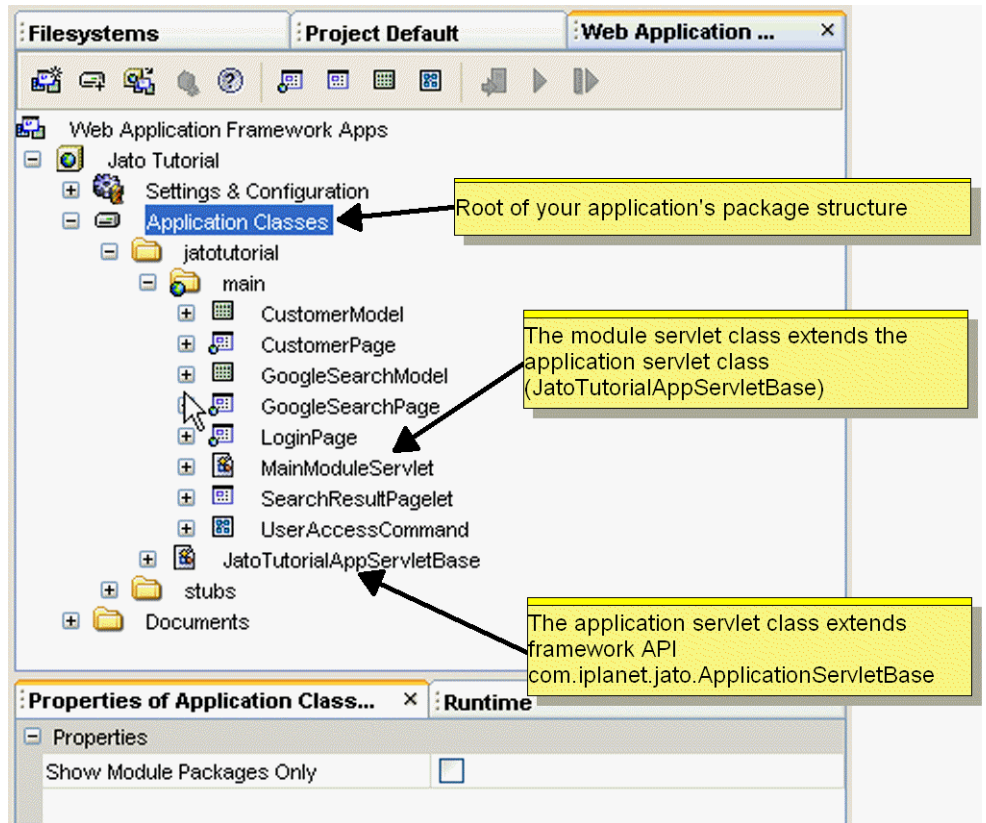
There are two key reasons for the parallel directory convention. First, it makes finding JSPs that are using the ViewBeans easier for developers, especially those new to Web Application Framework. It is easier to have the same directory structure for both classes and JSPs since a given page is comprised of both. If a developer knows where one is, he or she immediately knows exactly where the other is. Second, and most important, this parallel directory structure naturally inherits the same namespace scoping rules as the application code. Therefore, if there is more than one module in an application, classes AND JSPs with the same name will not conflict. This allows complete module independence.

An illustration of this would be if you had a *menu* page (`MenuViewBean.java` and `Menu.jsp`) in `module1` and a *menu* page in another module, `module 2`. Without the parallel directory structure, the two menu JSP files would need unique filenames (`Menu.jsp` and `Menu2.jsp`, for example).

Application Classes Node

The Application Classes node contains the complete Java package structure of your Web Application Framework application. Application Classes node is the root package of your application.

The following figure shows the Application Classes node.



Application Classes Node Contextual Menu Commands

The only actions you are likely to need on this node are the compile/build actions.

▼ [action] Compile

This compiles files in the current directory that are new or have changed since the last compile. The up-to-date check is done by comparing timestamps between the source (.java) and products (.class) of the compile. This command does not compile the files in subfolders.

▼ [action] Compile All

This compiles only those files that are new or have changed since the last compile, including the files in subfolders.

▼ [action] Build

This deletes the `.class` files in the folder and recompiles the source files. This command does not remove `.class` files or compile source files in subfolders.

▼ [action] Build All

This deletes all `.class` files within a folder and its subfolders and then compiles all files within the folder and subfolders.

Application Classes Node Properties

▼ [property] Show Module Packages Only

This allows you to hide all non-Module packages (packages that do not have a module servlet which is declared as such in the deployment descriptor). This can be convenient when you have a complex and deep package structure, and you only want to see those packages that have been marked as a Module folder.

Application Classes Subnodes

Possible subnodes of the Application Classes node are standard Java package folders, Web Application Framework module folders (which are also Java packages), and Java classes.

Java Package Folder Node

All folders under the Application Classes node are Java package folders. You are able to design and develop an application that has a package structure that is as complex as you want. Some (at least one) of those packages will be declared as a Web Application Framework module folder. Module folders are discussed in more detail below (see [Module Folder Node](#)).

Java Package Folder Node Contextual Menu Commands

▼ [action] Compile

This compiles files in the current directory that are new or have changed since the last compile. The up-to-date check is done by comparing timestamps between the source (`.java`) and products (`.class`) of the compile. This command does not compile the files in subfolders.

▼ [action] Compile All

This compiles only those files that are new or have changed since the last compile, including the files in subfolders.

▼ [action] Build

This deletes the `.class` files in the folder and recompiles the source files. This command does not remove `.class` files or compile source files in subfolders.

▼ [action] Build All

This deletes all `.class` files within a folder and its subfolders and then compiles all files within the folder and subfolders.

▼ [action] Add

This is a container for other menu actions: Command, Model, Module Folder, Page (ViewBean), and Subpage (ContainerView). These options all launch the wizard that guides you through creating the respective components. These actions have the same effect as using the Web Application Framework toolbar buttons (at the top of the Web Application Framework Apps window), or selecting the Sun Java Studio Enterprise 7 menu option, File | New, then expanding the Web Application Framework node and selecting one of the component creation wizards.

If you right-click a folder node under the Application Classes node, some additional actions are available. The Add menu item is a shortcut to some of the Web Application Framework component wizards.

Some Web Application Framework components are required to be created inside of a module folder (see [Module Folder Node](#)). The wizards will prevent you from creating components with this requirement.

▼ [action] Convert to Module

This transforms a regular Java package into a Web Application Framework module folder. Selecting this action displays a dialog that provides inputs to convert the standard package to a module folder. If servlet classes exist in the target package, they will be presented as the *module servlet candidates*, or you can enter a class name to create a new module servlet class.

You can create new module folders by right-clicking the desired base package node (such as Application Classes, `jatutorial`, `main`, or any other package in Application Classes), and selecting Add, then Module. You will be prompted to enter the name of a new module servlet.

Java Package Folder Properties

▼ [property] Module

Non-module Java packages in a Web Application Framework application can be converted into module folders by changing the property from *False* to *True*. For more details, see [\[action\] Convert to Module](#).

▼ [property] Name

This is the name of the folder.

▼ [property] Sort Mode

The subnodes of a module, or any folder in the Explorer, can be unsorted, sorted by type, name, or name (packages first).

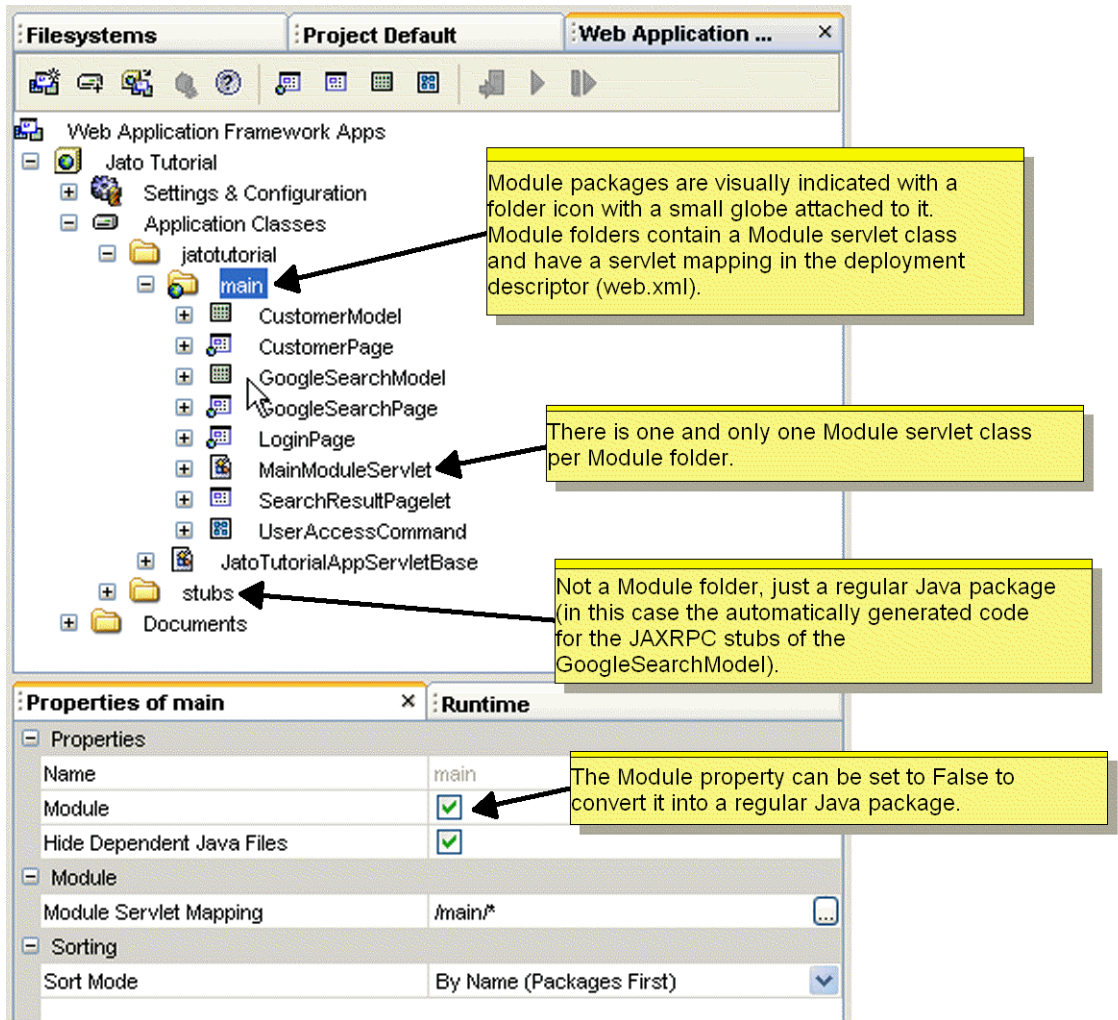
Module Folder Node

A Web Application Framework module folder node is a traditional Java package folder node badged with a small greenish globe in the bottom left corner. The IDE will recognize and badge a Java package folder as a valid Web Application Framework module folder if the folder contains a Web Application Framework module servlet and the module servlet is registered properly in the deployment descriptor file (`web.xml` in the `WEB-INF` directory of the Web application). The IDE will automatically manage the creation of valid Web Application Framework module folders for you. The IDE also supports the automatic conversion of Java package folders to Web Application Framework module folders.

A Web Application Framework application node typically contains at least one Web Application Framework module folder. Larger Web Application Framework applications might contain more than one Web Application Framework module folder. The module folder offers developers the opportunity to logically partition larger applications.

A module folder typically contains Web Application Framework components such as pages, pagelets, and models. It might also contain the arbitrary application resources that you can place in any Java package folder. The essential feature of Web Application Framework module folders is that Web Application Framework page components are only executable when they are in Web Application Framework module folders. Web Application Framework pages, pagelets, models, and command components can be created in arbitrary Java package folders, but Web Application Framework page components can only be executed from within Web Application Framework folders.

The following figure shows a Module Folder node.



Module Folder Node Contextual Menu Commands

▼ [action] Compile

This compiles files in the current directory that are new or have changed since the last compile. The up-to-date check is done by comparing timestamps between the source (.java) and products (.class) of the compile. This command does not compile the files in subfolders.

▼ [action] Compile All

This compiles only those files that are new or have changed since the last compile, including the files in subfolders.

▼ [action] Build

This deletes the `.class` files in the folder and recompile the source files. This command does not remove `.class` files or compile source files in subfolders.

▼ [action] Build All

This deletes all `.class` files within a folder and its subfolders, and then compile all files within the folder and subfolders.

▼ [action] Add

This is a container for other menu actions: Command, Model, Module Folder, Page (ViewBean), and Subpage (ContainerView). These options all launch the wizard that guides you through creating the respective components. These actions have the same effect as using the Web Application Framework toolbar buttons (at the top of the Web Application Framework Apps window), or selecting the Sun Java Studio Enterprise 7 menu option, File | New, then expanding the Web Application Framework node and selecting one of the component creation wizards.

If you right click a folder node under the Application Classes node, some additional actions are available. The Add menu item is a shortcut to some of the Web Application Framework component wizards.

Module Folder Node Properties

▼ [property] Module

Module folders have a property named `Module`. It is always *True* when it is a Module. This means that this folder has a `ModuleServlet` class (`MainModuleServlet` in this case), and it has some entries in the deployment descriptor (`web.xml`).

The entries look like as follows:


```
<context-param>
  <param-name>jato:jatotutorial.main.*:moduleURL</param-name>
  <param-value>../main</param-value>
</context-param>

<servlet>
  <servlet-name>jatotutorial_main</servlet-name>
  <servlet-class>jatotutorial.main.MainModuleServlet</servlet-class>
</servlet>
<servlet-mapping>

<servlet-name>jatotutorial_main</servlet-name>
  <url-pattern>/main/*</url-pattern>
</servlet-mapping>
```

If you were to change the Module property to *False*, these entries would be removed from the deployment descriptor, and the module folder would become a regular Java package. To convert it back to a module folder, change the property back to *True*, or right-click the non-module folder, and select the Convert to Module action. For more details, see [\[action\] Convert to Module](#).

Note – The Web Application Framework tools enable customization of some of the deployment descriptor file entries without directly editing the file. However, direct file editing is possible if you are Web application deployment descriptor and Web Application Framework savvy. Improper modification of some the entries could cause unpredictable results for your design-time and run-time environments.

▼ [property] Module Servlet Mapping

This property facilitates servlet invocation based upon mapping a URL pattern from an end user request to a servlet (the module servlet). For this example, `/main/*` is the pattern. This means that any URL with the pattern of `http://<server>/<servletcontext>/main/<anything-at-all>`, invokes the `MainModuleServlet` servlet class.

▼ [property] Sort Mode

The subnodes of a module, or any folder in the Explorer, can be unsorted, sorted by type, name, or name (packages first).

Web Application Framework Component Nodes

This chapter provides an overview of the various nodes that visually represent the major Web Application Framework components that you create in your Web Application Framework application.

Details about these nodes and Web Application Framework specific contextual menu actions are presented below.

Each application specific page, pagelet, model, and command component is a subtype of the corresponding Web Application Framework component which was selected during the application component creation phase. The IDE introspects the base component and dynamically presents component properties which are declared by the library component from which the application component derives. This section focuses on those contextual menu commands and properties which are common to all components of a given type.

Page Component Node

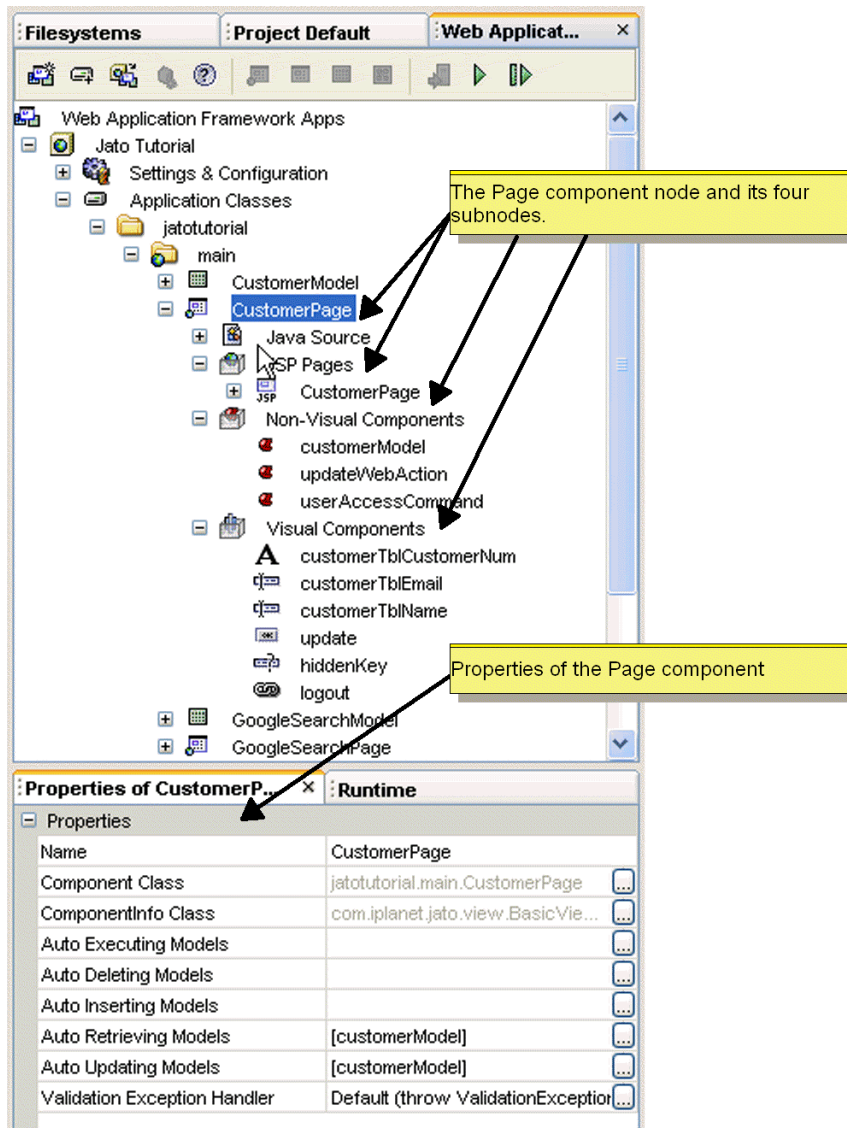
Web Application Framework page components are the primary visual objects in a Web Application Framework application. At runtime each Web Application Framework page component can be invoked by a URL and returns a document.

A Web Application Framework page component consists of a Java class and zero or more associated JSPs. The IDE assists in the construction and configuration of Web Application Framework pages by exposing the structure of the page as a collection of subnodes. These subnodes are described below.

As you add, subtract, and declaratively configure the component's subnodes, the IDE automatically updates the component's Java source file so that it remains in sync with the component's configuration. The automatically generated Java source code is protected from user modification by being guarded in protected blocks. The IDE does not allow you to edit these protected blocks because they are subject to regeneration whenever the component's configuration changes.

Each application page component is a subtype of the Web Application Framework page component which you selected during the page creation phase. The IDE introspects the base page component and dynamically presents those component properties which are declared by the library component from which the application component derives.

The following figure shows the Page Component node and its four subnodes.



Page Component Node Contextual Menu Commands

▼ [action] Open

This opens the page component's Java source file and jumps to the class in the Source Editor.

▼ [action] Edit Default JSP

This opens the page component's default JSP file and jumps to the page in the Source Editor. Each page component might be associated with zero or more JSPs. One of the associated JSP's is designated as the default JSP.

▼ [action] Error Information

This displays a dialog which describes a list of one or more error messages. The messages describe the configuration inconsistencies so you can take corrective action before proceeding with further configuration or test running of the component.

This command only appears in the node's contextual menu when the component's internal state is in error due to some configuration inconsistency. Whenever the component's internal state is in error, the component node is badged with the standard error or warning badge. The warning or error badge disappears when you have corrected the errors.

▼ [action] Events

This launches a submenu which lists the event handler methods suitable for implementation in this component. The names of already implemented event handlers appear in bold. Those which have not yet been implemented are not in bold.

When you select a bold event handler from the list, the IDE opens the component's Java source and positions your cursor at the start of the event handler. When you select a non-bold event handler, the IDE adds the corresponding event handler's method stub to the component's Java source file and position your cursor above it.

The list of available event handler methods is component specific and declared by the component from which the current component is derived. Some components do not declare event handlers. This menu command is inactive if the component does not declare any event handlers.

▼ [action] Component Methods

This launches a submenu which lists useful component methods available for this component to implement. The names of already implemented component methods appear in bold. Those which have not yet been implemented are not in bold.

When you select a bold component method from the list, the IDE opens the component's Java source and positions your cursor at the start of the method. When you select a non-bold component method, the IDE adds the corresponding method stub to the component's Java source file and positions your cursor above it.

The list of available component methods is component specific and is declared by the component from which the current component is derived. Some components do not declare component methods. Typically, component methods are a subset of available base class methods which are specifically intended for override by subclasses and deemed important enough by the library component's author to merit exposure in this convenient manner. This menu command is inactive if the component does not declare any event handlers.

▼ [action] Execute Page

This test runs the page. This command launches a browser and invokes the proper URL to request the current page from the IDE's currently configured servlet container. This command does not redeploy the application. Changes made since the last deploy are not reflected in the test run.

▼ [action] Execute Page (Redeploy)

This test runs the page. This command automatically compiles the current application, deploys the current application to the IDE's currently configured servlet container, and launches a browser and invokes the proper URL to request the current page.

▼ [action] Rename

This sets the name of the component class and renames the component's Java source file.

▼ [action] Delete

This deletes the page component. This command deletes the component's Java source file. It also prompts and asks if you also want to delete any JSP files formerly associated with this page.

▼ [action] Cut, Copy, and Paste

This provides the conventional cut, copy, and paste behavior.

Do not attempt to copy a page from one Web Application Framework application into another Web Application Framework application if the source page contains references to other objects within the source application. The references are not preserved across Web Application Framework application boundaries.

Page Component Node Properties

▼ [property] Component Class

This specifies this component's fully qualified class name. This property is non-editable.

▼ [property] ComponentInfo Class

This specifies this component's fully qualified `ComponentInfo` class. This property is non-editable.

▼ [property] Name

This sets the name of the component class.

Page Component Subnodes

A Page node has four immediate subnodes as follows:

- Java Source
- JSP Pages
- Non-Visual Components
- Visual Components

Each of these nodes might have subnodes of its own.

Java Source Node

The Java source node provides node level access to this component's Java source file. The Web Application Framework component node parents the Java source node in order to emphasize that the application component is logically comprised of more than just the Java node. The Java source node is identical to the conventional Sun Java Studio Enterprise 7 Java source node with the following exceptions:

- It is parented by a Web Application Framework component node.
- Its node name is fixed as *Java Source*.
- Its contextual menu cut, delete, and rename commands are disabled.

Use the component's cut, delete, and rename commands instead.

The Java Source node contextual menu commands and properties are described fully in the Sun Java Studio Enterprise 7 documentation.

JSP Pages Node

The JSP Pages node lists all of the JSPs that are *using* the page component—the page component and the JSP are *associated*. A given page might be associated with zero or more JSPs. Therefore, this node might contain zero or more JSP subnodes. For any given runtime client request a single associated JSP is used to render the page's content. However, Web Application Framework pages might be coded to conditionally and dynamically utilize any one of its associated JSPs for a give request. This Web Application Framework feature is known as parallel content. For more details on parallel content, consult the *Web Application Framework Developer's Guide*.

Once a JSP is associated with a given Web Application Framework page, the IDE keeps the page component and its associated JSP in sync. As child visual components are added to the page component, the IDE automatically adds a JSP tag to any and all associated JSPs. Similarly, as child visual components are renamed and deleted, the tags are automatically deleted or renamed in order to maintain design time synchronization. For example, if you add a `TextField` child visual component named *foo* to a page component, a `jato:textField` tag is inserted into all the associated JSPs listed in the JSP Pages node.

This synchronization is one way. Modifications to the page component are automatically reflected in the JSP, but not vice versa. The page component is primary, and the JSP secondary. The JSP uses the page component in a standard *uses* relationship.

A given JSP might only be formally associated with one Web Application Framework page. However, since Web Application Framework pages might contain child visual component of type `pagelet`, one can easily construct arbitrary compositional hierarchies of page and `pagelet`, and achieve maximal reuse of application level visual components.

An associated JSP has a `jato:useViewBean` tag specifying the class name of the page component as shown below:

```
<jato:useViewBean className="jatotutorial.main.CustomerPage">
```

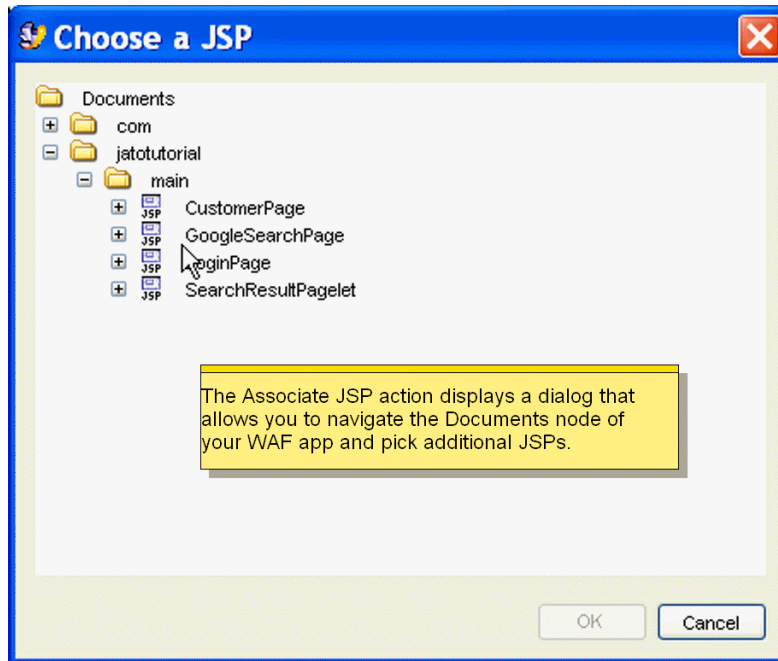
JSP Pages Node Contextual Menu Commands

▼ [action] Associate JSP

This associates an additional JSP with the page component. Select a JSP to associate from the list of available JSPs under the current application document root. The OK button in the *Choose a JSP* Dialog is only enabled when you have selected a JSP which is not associated with any other page at this time. This command modifies the

associated JSP's content so that it is configured with the appropriate Web Application Framework tag library descriptor (tld) directive and a Web Application Framework JSP tag which declares the use relationship between the associated JSP and this page.

The following figure shows the Choose a JSP Editor.



▼ [action] Add JSP

This creates a new JSP page and associates it with this page. The newly generated JSP is automatically populated with JSP tags that correspond to all of this page and this page's visual component children. When the new JSP page is created, it is added to this page's list of associated JSP pages, and a new JSP subnode appears under the JSP Pages Node.

▼ [action] Change Order

This displays a dialog with a list of the current node's subnodes. The dialog has buttons (Move Up and Move Down) that allow you to manipulate the display order of the subnodes.

JSP Pages Node Properties

▼ [property] Default JSP

This specifies the URI for the associated JSP which is currently designated as the Default JSP for this page. The Default JSP is the JSP used to render the current page when this page is test run within the Sun Java Studio Enterprise 7, unless otherwise overridden by page specific code. This property is non-editable.

The value of the current setting for the Default JSP property is reflected in the page component's generated Java source code, as follows:

```
setDefaultDisplayURL("/jatotutorial/main/CustomerPage.jsp");
```

JSP Pages Subnodes

There can be zero or more JSP Page Nodes under the JSP Pages Node.

JSP Node

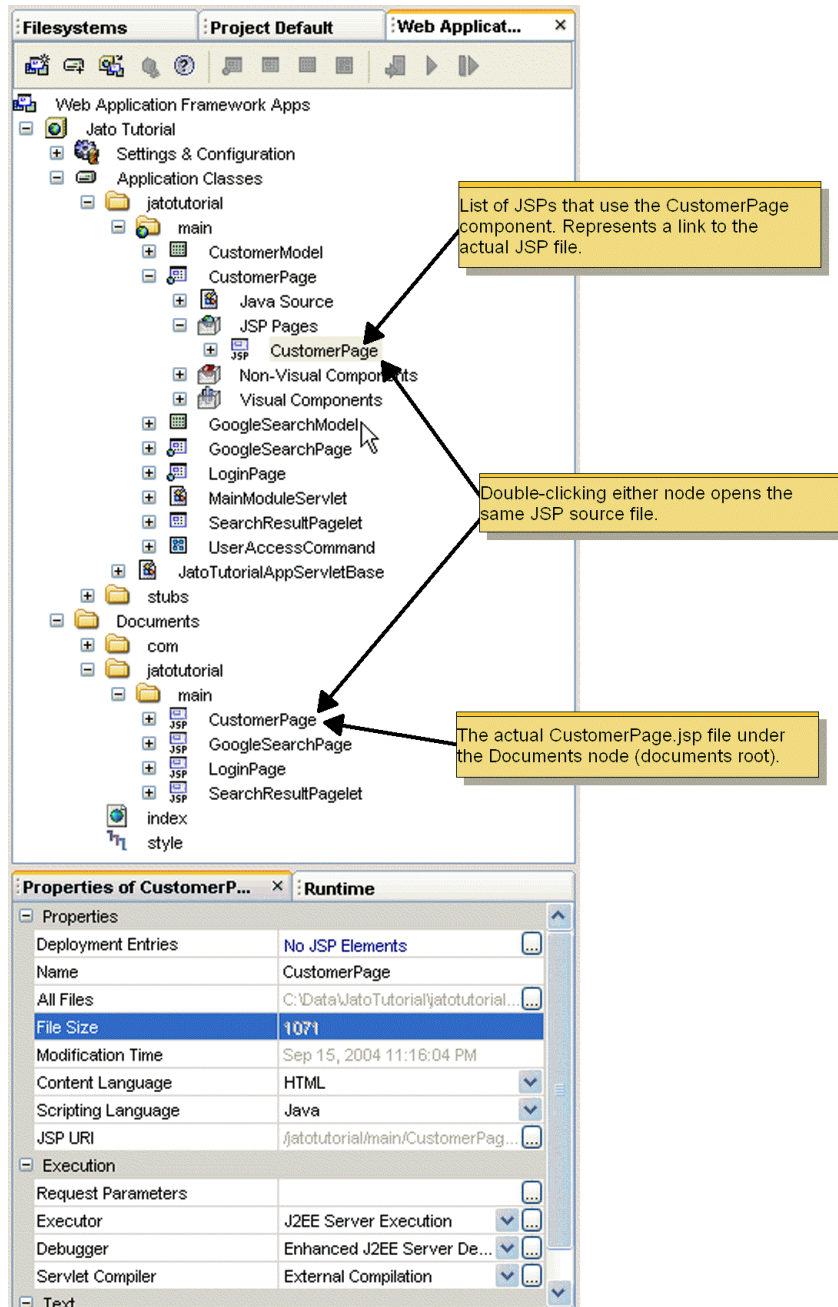
The JSP Node provides node level access to a page component's formerly associated JSP file. The JSP node is identical to the conventional Sun Java Studio Enterprise 7 JSP node.

Though the JSP file is physically located under the application's document root directory, and the page component is physically located under the application's `WEB-INF/classes` directory, the IDE creates a copy of the JSP node so that Web Application Framework page component developers can have easy access to the associated JSP from within the node hierarchy of the page component.

Note that this is a copy of the node only. There is only one physical JSP file, even though its JSP node appears in two places. This visually confirms that a given JSP is associated with a given page component, and that the IDE automatically synchronizes the content of the JSP as modifications are made to the page component that the JSP uses.

You can see that the JSP node exists in two node hierarchies simultaneously, the document node hierarchy, and the page node hierarchy.

The following figure shows the JSP node.



The JSPs under the Documents node represent the actual JSP files, while the JSPs under the JSP Pages node are simply links to the actual file. This means that if you delete a JSP under the JSP Pages node, you are just removing the visual association

it has with the page component in the Sun Java Studio Enterprise 7. However, if you delete a JSP under the Documents node, you are deleting the physical file from disk.

JSP Node Contextual Menu Commands

The JSP node contextual menu commands are identical to the conventional Sun Java Studio Enterprise 7 JSP node contextual menu commands with the exception of the commands detailed here. The remaining JSP node contextual menu commands are described fully in the Sun Java Studio Enterprise 7 documentation.

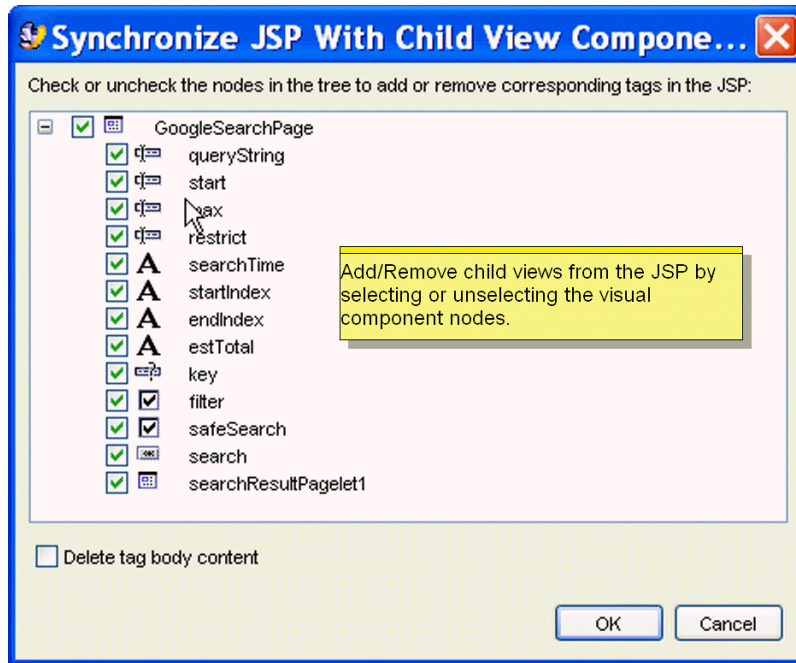
▼ [action] Synchronize to View

This presents a synchronization editor that allows you to selectively add and remove JSP tags that correspond to the page's visual component hierarchy. This command is merely a convenience utility. You can, of course, manually edit the JSP to add or remove JSP tags. However, the *Synchronize to View* editor offers a more convenient way of introducing or removing tags in a batch mode.

This editor presents the page component's complete visual hierarchy in a simplified node tree. All of the children from the page component's Visual Components node display as subnodes. In more complex visual component hierarchies, there might be pagelet children, a TiledView for example, that have child visual components as well. The complete view hierarchy, no matter how complex and deep, displays in this editor.

All visual components from the page that are currently referenced by tags in the JSP appear selected (checked). All visual components from the page that are currently unreferenced by tags in the JSP appear as unselected (unchecked). By unchecking and/or checking the various nodes, you can perform a batch modification of the JSP.

This figure shows the synchronization editor.



You can remove undesired tags by unselecting (unchecking) the corresponding node. If a child is also a pagelet (like a TiledView), unselecting it also unselects all of its children, and they are all deleted.

You can add a tag by selecting (checking) the corresponding node. Selecting a pagelet child initially selects all of its children. You may selectively uncheck any of the nested children if you want to do so.

The Delete Tag Body Content checkbox at the bottom of the Synchronization dialog allows you to control the tag deletion policy as it pertains to tags which have existing body content (for example, HREF or pagelet tags). You might choose to have the tool also delete the body content, or leave it intact even while the tag itself is deleted. The Delete tag body content checkbox is unselected by default to prevent existing body content from being deleted without your specific consent.

▼ [action] Set as Default

This sets the current JSP as the Default JSP for the page component.

▼ [action] Delete

This deletes the current JSP node from the JSP Pages node and breaks the formal association between the JSP and the page component. However, this command does not physically delete the JSP file from disk. By invoking this command you are just

removing the visual association between the JSP file and the current page component. If you want to delete the physical JSP file, delete the actual JSP node under the Document root.

JSP Page Node Properties

The JSP node properties are identical to the conventional Sun Java Studio Enterprise 7 JSP node properties, with the exception of the properties detailed here. The remaining JSP node properties are described fully in the Sun Java Studio Enterprise 7 documentation.

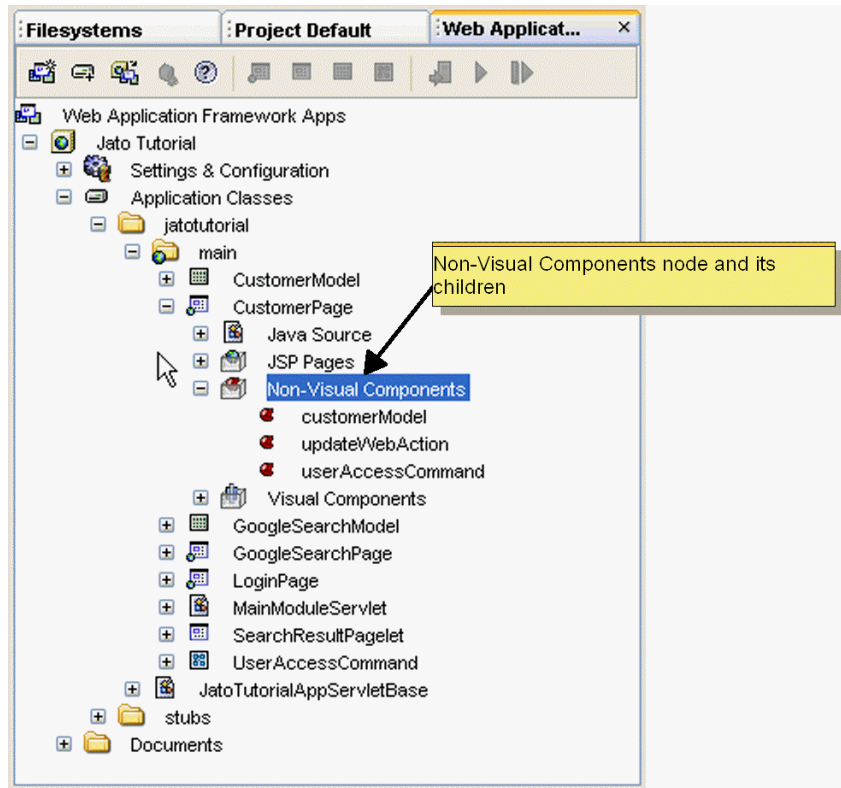
▼ [property] JSP URI

This specifies the JSP file location under the application document root. This is purely a convenience property which allows developers to see at a glance the physical location of the JSP file. This property is non-editable.

Non-Visual Components Node

Non-visual components are those non-visual components used by the page component. For example, a page component might *use* or *refer* to zero or more Models, zero or more Command components. These references can be configured declaratively as non-visual component nodes. Component libraries might introduce arbitrary non-visual components for use as well. The IDE generates the corresponding Java code to reflect the declarative configuration of these non-visual resources.

The following figure shows the non-visual components node and its children.



Because these non-visual components are essentially just JavaBeans, the types of components that can appear here is limitless, and therefore, the property sheets for each type will vary. Typically, you will define non-visual components, because your page's visual components contain properties with values that must be set to refer to non-visual components. The classic example of this internal *uses* relationship is the visual component's `ModelReference` property which is satisfied by referring to a non-visual `ModelReference` component. Some of the visual component property editors implicitly create non-visual components when you configure those properties. You might also explicitly add non-visual components to the page.

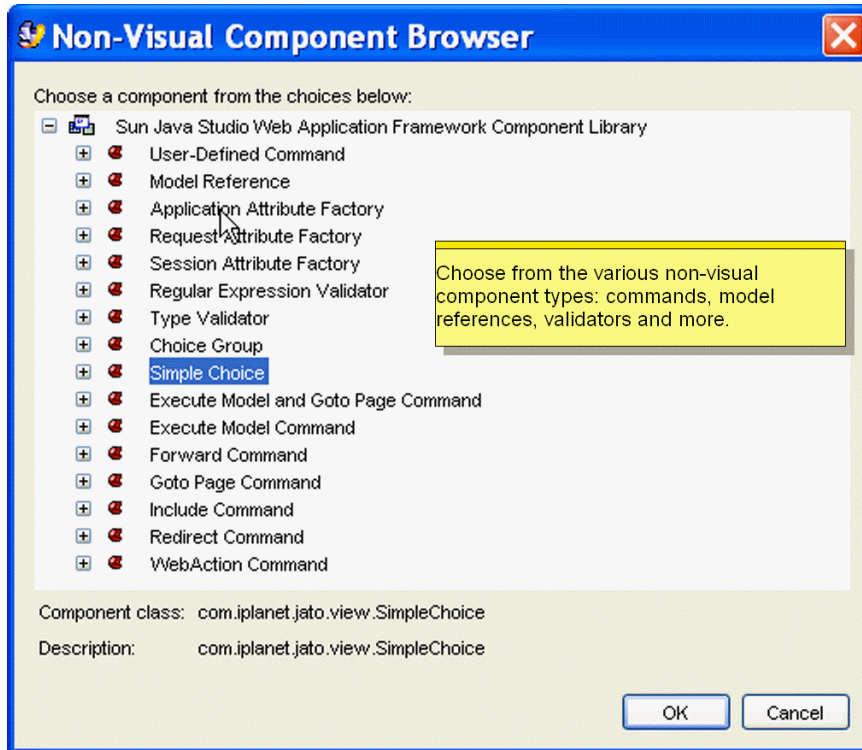
A given non-visual component might be used by zero or more visual components within the current page.

Non-Visual Components Node Contextual Menu Commands

▼ [action] Add Non-Visual Component

This explicitly adds an arbitrary new non-visual component. Select from the list of available non-visual components in the Web Application Framework component libraries that are registered in your Web Application Framework application.

The following figure shows the Non-Visual Component Browser.



▼ [action] Change Order

This displays a dialog with a list of the current node's subnodes. The dialog has buttons (Move Up and Move Down) that allow you to manipulate the subnodes display order.

Non-Visual Components Node Properties

▼ [none]

Non-Visual Components Subnodes

The non-visual components subnodes consists of zero or more non-visual component nodes.

Non-Visual Component Node

A non-visual component node provides declarative configuration for a non-visual component which is instantiated in the scope of the current page. The IDE automatically generates the code to instantiate the non-visual component within the page's Java source file.

Non-Visual Component Node Contextual Menu Commands

▼ [action] Cut, Copy, and Paste

This provides conventional cut, copy and paste behavior.

▼ [action] Delete

This deletes the non-visual component node. Upon deletion of a non-visual component, any visual components which have property values still holding a reference to the deleted non-visual component are badged with a warning. The error information available on the page node describes the stale property references in detail so that you can clean them up.

Non-Visual Component Node Properties

Each non-visual component has three property tabs:

- Properties
- Component Properties
- Code Generation

The Properties and Code Generation property tabs contain the same properties for all non-visual components and are described below.

The Component Properties window, however, contains properties which are specific to a given non-visual component type. Since the non-visual components are arbitrary JavaBeans, these bean specific properties are dynamically discovered by the IDE and presented for editing in the Component Properties window. These non-visual component specific properties are not documented here since they are component specific.

▼ [property] Component Class

This specifies the fully qualified class name for the non-visual component. This property is non-editable.

▼ [property] Name

This specifies the non-visual component's logical name. This is the name that appears wherever the non-visual component is referenced by visual component properties within the scope of this page component. The IDE also factors the value of the name property into the code it generates to manage the construction and access to the non-visual component within the page's Java source file. The exact code which is generated varies depending on the values of the three related code generation properties:

- Access Privilege
- Scop
- Session Attribute Name

▼ [property] Access Privilege

This specifies the access privilege which the IDE's code generation engine applies when it generates the code which provides access to the non-visual component instance within the page's Java source.

▼ [property] Scope

This specifies a scope setting that allows you to influence the IDE's code generation engine when it generates the code which provides access to the non-visual component instance within the page's Java source.

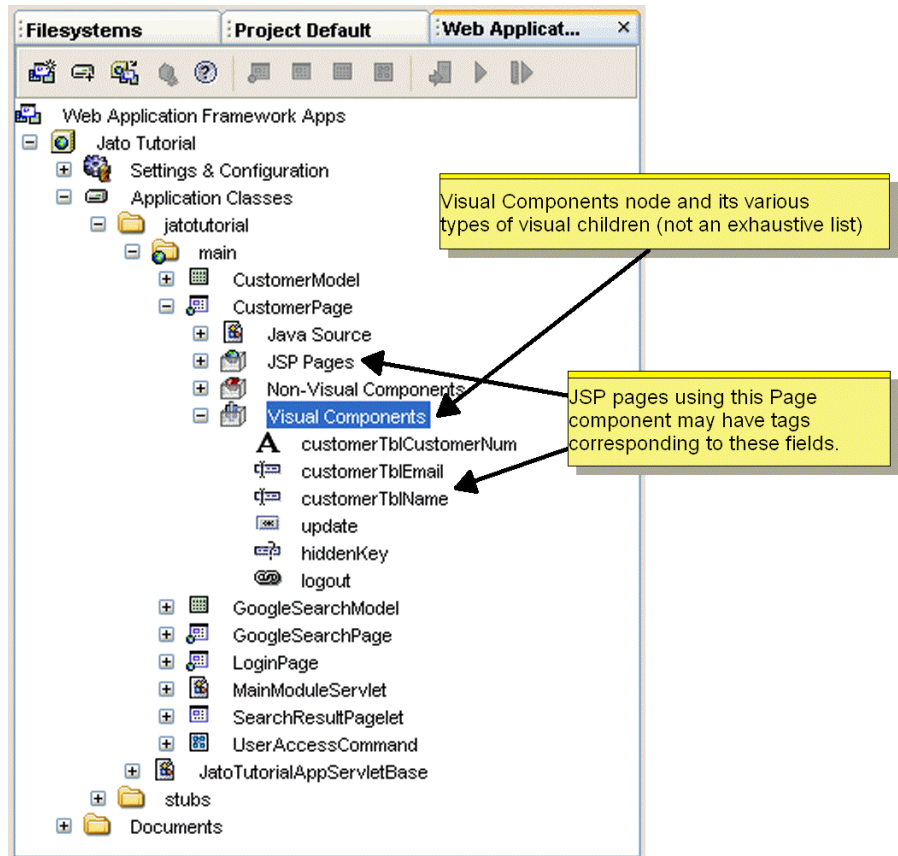
▼ [property] Session Attribute Name

This specifies an HTTP session attribute name. If the scope property is set to HTTP Session, the code generation engine uses the specified session attribute name when it generates the code that provides access to the non-visual component instance within the page's Java source.

Visual Components Node

The Visual Components node contains zero or more child visual component nodes. This is the heart of the visual component story. The page node forms the root of the hierarchy. It might contain zero or more visual component subnodes. Any visual components subnode which is a pagelet component may itself have child visual components. This allows for an arbitrarily deep visual component hierarchy.

The following figure shows the Visual Components node and its various types of visual children.



Visual Component Node Contextual Menu Command

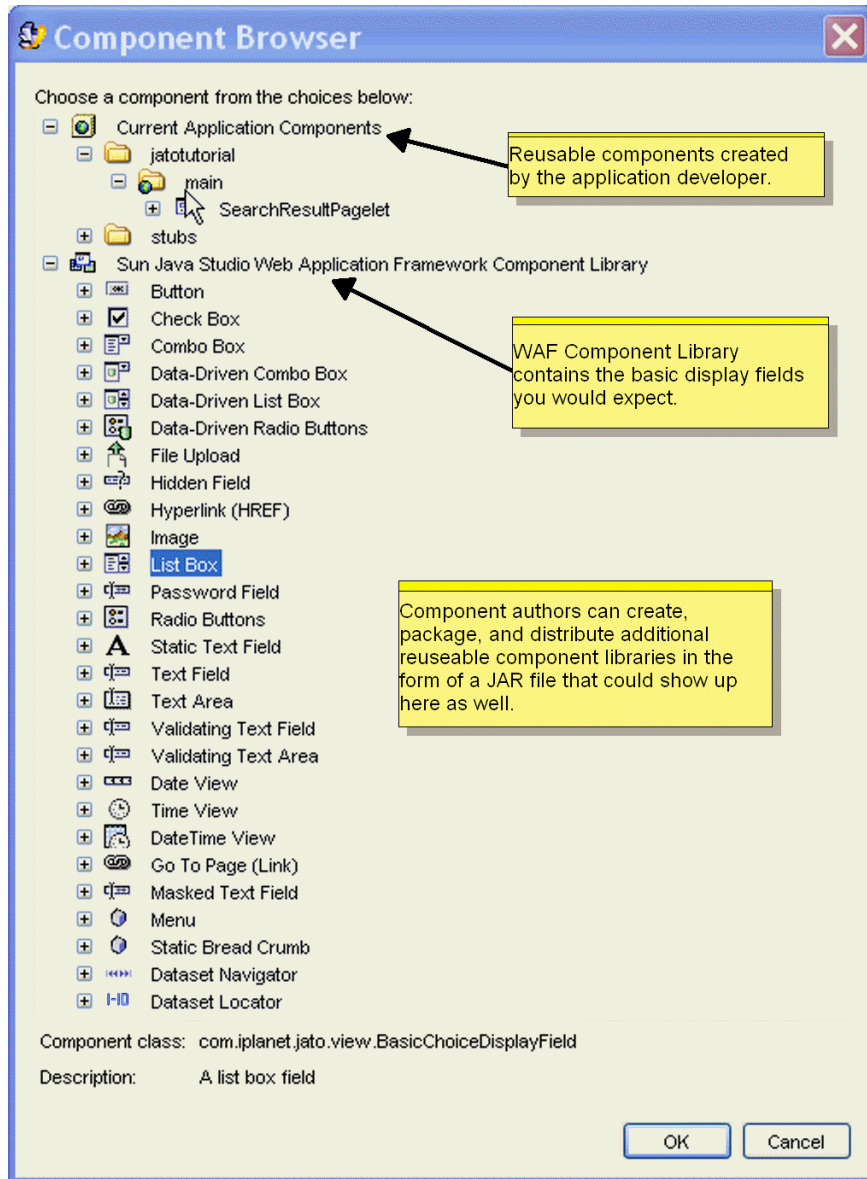
▼ [action] Add Visual Component

This adds a new visual component node. This command presents a component chooser editor. The effect of choosing a component from the component chooser is functionally equivalent to adding a visual component by clicking on the Component palette. They are two equally valid ways to add a visual component.

The Component Browser editor displays with at least two nodes: *Current Application Components* and the *Web Application Framework Component Library*.

If you added a third party Web Application Framework component library JAR files to your `WEB-INF/lib` directory, you would see additional nodes representing those component libraries as well. For more information about creating your own reusable, distributable Web Application Framework component libraries, consult the *Web Application Framework Component Author's Guide*.

The following figure shows the Component Browser.



For every visual component node which is added to the page in this manner, the IDE generates a component constructor invocation, a component accessor method, and a class constant for the component instance name. Additionally, a corresponding component specific JSP tag is inserted in all JSPs using the page component.

All of this automation facilitates a more consistent and less error prone application. If you require more flexibility with the creation of your visual components, manual creation is an option. See the *Web Application Framework Developer's Guide*.

▼ [action] Paste

Paste a visual component. When pasting a visual component which was copied or pasted from a different page, any non-visual components referred to by the original visual component node is not automatically carried forward into the new page.

▼ [action] Change Order

This displays a dialog with a list of the current node's subnodes. The dialog has buttons (Move Up and Move Down) that allow you to manipulate the subnodes display order.

Visual Component Node Properties

▼ [none]

Visual Component Subnodes

The visual component subnode consists of zero or more visual component nodes.

Visual Component Node

A visual component node might be a simple visual component node (for example, text fields, choice fields, command fields, and so on) or a pagelets node (for example, TiledViews, ContainerViews, and TreeViews). A pagelet node is expandable into its own visual component subnodes, thereby allowing for an arbitrary visual component hierarchy.

Visual Component Node Contextual Menu Commands

▼ [action] Events

This launches a submenu which lists the child specific event handler methods suitable for implementation within the parent component's source code file. The names of already implemented event handlers appear in bold. Those which have not yet been implemented are not in bold. When you select a bold event handler from

the list, the IDE opens the parent component's Java source and positions your cursor at the start of the event handler. When you select a non-bold event handler, the IDE adds the corresponding event handler's method stub to the parent component's Java source file and positions your cursor above it. Since the event handlers need to be child component specific, the IDE generates the event handler methods so that the method name is uniquely named, based on the child component's instance name. For example if the event handler appears as *beginDisplay* in the events menu, and the visual component instance is named *foo*, then the IDE generates a handler method named *beginFooDisplay*.

The list of available child component event handler methods is determined by the parent component's ability to dispatch child component specific events. This ability is declared by the component from which the current page component is derived. The visual component node's event contextual menu command is inactive if the page component does not declare any event handlers for the current child component type.

Visual Component Node Properties

All visual component nodes share a few common properties. Each visual component also has an arbitrary number of visual component specific properties which the IDE dynamically discovers and presents for your configuration within the property sheet.

▼ [property] Component Class

This specifies this visual component's fully qualified class name. This property is non-editable.

▼ [property] ComponentInfo Class

This specifies this visual component's fully qualified `ComponentInfo` class. This property is non-editable.

▼ [property] Name

Specifies the visual component's instance name. The IDE also factors the value of the name property into the code it generates to manage the construction and access to this visual component within the page's Java source file.

▼ [property] Pre-initialization Code

This specifies a block of arbitrary Java code which is inserted by the code generation engine immediately following the invocation of the visual component's constructor. This pre-initialization code is therefore inserted before any of the explicit property setters are invoked. This allows you to insert code into the otherwise non-editable section of the page's Java file.

▼ [property] Post-initialization Code

This specifies a block of arbitrary Java code which is inserted by the code generation engine immediately after all of the automatically generated component specific property setter invocation code. This allows you to insert code into the otherwise non-editable section of the page's Java file.

Pagelet Component Node

Pagelet nodes are virtually identical to Page nodes. They both have similar subnodes, contextual menu commands, and properties. There is one notable exception. Pagelets nodes are not executable the way Page nodes are. This is because pagelets are subpages, or page fragments. They are intended to provide visual building blocks for other more complex visual components and ultimately for inclusion, directly or indirectly as children of page components. Page components form the root of an arbitrary visual component hierarchy. The pagelets form the branches of a given visual component hierarchy. A pagelet may contain zero or more other pagelet children. A given pagelet may be a child of any number of other pagelet or page components.

One example of a pagelet is the TiledView component. The TiledView has built-in iteration behavior. It is commonly used to iterate over a result set to display the values of the columns of each of the rows (tiles).

Pagelet Component Node Contextual Menu Commands

The pagelet contextual menu commands are identical to the Page node's contextual menu commands, except that the pagelet node does not have an execution command. Pagelets cannot be test run. They are not addressable via URL the way pages are.

See [Page Component Node Contextual Menu Commands](#).

Pagelet Component Node Properties

See [Page Component Node Contextual Menu Commands](#).

Pagelet Component Subnodes

See [Page Component Node Contextual Menu Commands](#).

Special Note on Pagelet JSP Nodes

It is perfectly acceptable for a Pagelet node not to have any associated JSP nodes. This may sound counterintuitive until you realize that the Web Application Framework Pagelet is first and foremost a visual component hierarchy. The rendering of the visual component hierarchy is controlled by corresponding JSP tags, but the JSP tags do not need to be isolated to a pagelet specific JSP fragment.

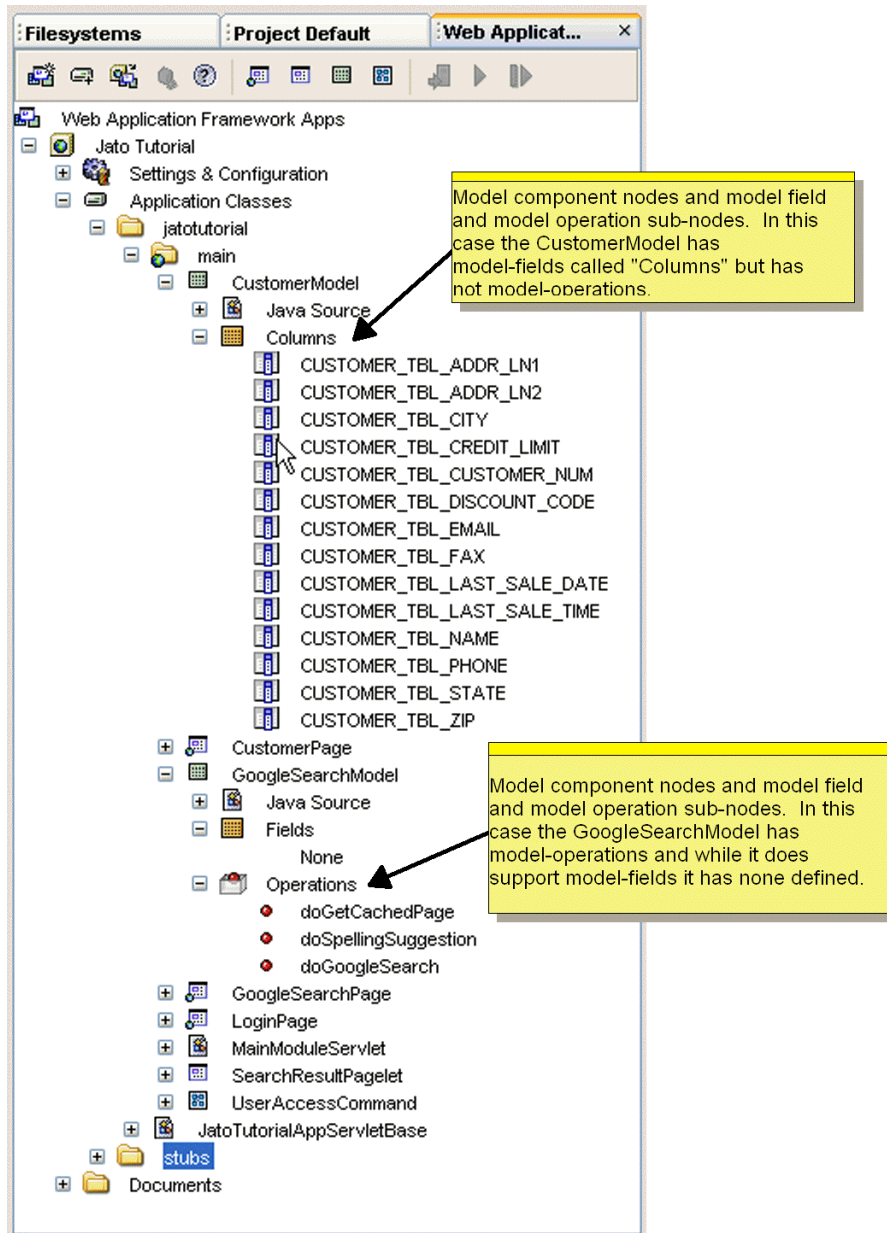
In the classic pagelet use case, the tags for rendering the pagelet's own visual component hierarchy might be isolated in reusable JSP fragment(s) and formally associated with the Pagelet component as a JSP subnode(s). If this is the case, whenever the pagelet is added as a child to another container, the IDE adds a single JSP include tag to the parent visual component's JSP file(s). The include tag includes the pagelet's JSP fragment in conventional JSP include fashion. This is the approach to take when you want the pagelet's rendering to be identical in each context in which it appears as a child.

However, there are cases where you may want a pagelet's visual layout to vary as that pagelet is used within different parent components. In that case, you might intentionally refrain from explicitly associating any JSPs with the pagelet. In this case, whenever the pagelet is added as a child to another visual component, the IDE adds tags for the pagelet and its children to the parent visual component's JSP file(s), thereby creating a deep tag hierarchy in the parent's JSP file(s). This allows the parent JSP to be edited as a single document, which some JSP authors prefer. The Web Application Framework offers you the choice and allows you to decide on a pagelet specific basis.

Model Component Node

Models are classes that serve as a storage mechanism and/or a proxy to data for the application's views (pages and pagelets). There are several types of models that ship with Web Application Framework: JDBC SQL Query, JDBC Stored Procedure, Web Service, Resource Bundle, Bean Adapter, Custom, HTTP Session, Tree, Object Adapter, and more.

The following figure shows examples of model component nodes.



Model Component Node Contextual Menu Commands

▼ [action] Open

This opens the component's Java source file and jumps to the class in the Source Editor.

▼ [action] Events

This launches a submenu which lists the event handler methods suitable for implementation in this component. The names of already implemented event handlers appear in bold. Those which have not yet been implemented are not in bold. When you select a bold event handler from the list, the IDE opens the component's Java source and positions your cursor at the start of the event handler. When you select a non-bold event handler, the IDE adds the corresponding event handler's method stub to the component's Java source file and positions your cursor above it. The list of available event handler methods is component specific and declared by the component from which the current component is derived. Some components do not declare event handlers. This menu command is inactive if the component does not declare any event handlers.

▼ [action] Design Actions

This launches a submenu which lists the special design actions available for this component. A design action is an arbitrary function provided by the component author which may operation on or mutate the configuration of the component. A design action can be silent, or it might display a customizer for the component.

▼ [action] Component Methods

This launches a submenu which lists useful component methods available for this component to implement. The names of already implemented component methods appear in bold. Those which have not yet been implemented are not in bold. When you select a bold component method from the list, the IDE opens the component's Java source and positions your cursor at the start of the method. When you select a non-bold component method the IDE adds the corresponding method stub to the component's Java source file and positions your cursor above it.

The list of available component methods is component specific and declared by the component from which the current component is derived. Some components will not declare component methods. Typically, component methods are a subset of available base class methods which are specifically intended for override by subclasses and deemed important enough by the library component's author to merit exposure in this convenient manner. This menu command is inactive if the component does not declare any event handlers.

▼ [action] Rename

Sets the name of the component class and renames the component's Java source file.

▼ [action] Delete

Deletes the component. This command deletes the component's Java source file.

▼ [action] Cut, Copy, and Paste

Provides the conventional cut, copy and paste behavior.

Model Component Node Properties

▼ [property] Component Class

Specifies this component's fully qualified class name. This property is non-editable.

▼ [property] ComponentInfo Class

Specifies this component's fully qualified ComponentInfo class. This property is non-editable.

▼ [property] Name

Sets the name of the component class.

Model Component Subnodes

Different types of models can have slightly different subnodes. All models have a Java Source subnode just like the page and pagelet nodes. Model components may have zero or more model field group nodes and models also may support a model operation group node.

The optional model field group node logically represents a set of zero or more named model fields. Visually, the model field group node is parent to zero or more model field nodes. The type of the model field is arbitrary and decided by the component author. The label for the model field group node usually changes depending upon the type of model field it exposes.

All the model fields across all the model field groups must have unique names. In other words, there is one set of uniquely named model fields for the entire model but the component author may allow the model fields to be partitioned into different groups so that different types of fields are supported.

It is possible for a model to have zero model field groups, in which case views will bind to the models with anonymous bindings entered manually. It is also possible for a model to only expose internally generated automatic fields through the binding chooser even though there are no model field nodes or model field group nodes for

that model. All this depends upon the decisions made by the component author and the kind of data which the model is managing. A model component that exposes hundreds or thousands of bindings could be designed. In this case, it is not desirable to represent all the fields as nodes, and the component author would instead provide a specialized chooser for the view binding.

The optional model operation group node logically represents a set of zero or more named model operations. Visually, the model operation group node is parent to zero or more model operation nodes. The label for the model operation group node might change depending upon the type of model operation it exposes. However, the label is usually *Operations*.

Java Source Node

The Java Source Node provides node level access to this component's Java source file. The Web Application Framework component node parents the Java source node to emphasize that the application component is logically comprised of more than just the Java node. The java source node is identical to the conventional Sun Java Studio Enterprise 7 Java source node with the following exceptions:

- It is parented by a Web Application Framework component node.
- Its node name is fixed as *Java Source*.
- Its contextual menu cut, delete, and rename commands are disabled. Use the component's cut, delete, and rename commands instead.

The Java Source node contextual menu commands and properties are described fully in the Sun Java Studio Enterprise 7 documentation.

Model Field Group Node

This is a node that parents a group of zero or more model field nodes.

Model Field Group Node Contextual Menu Commands

▼ [action] Add model field

Add a new model field to the group. A new subnode is added for the model field. The name of the field defaults to a unique name derivative of a basename defined by the model component author. The developer might immediately change that name as desired. The addition of the model field updates the generated code section of the model's Java file.

▼ [action] Paste

This pastes a model field.

▼ [action] Change Order

This displays a dialog with a list of the current node's subnodes. The dialog has buttons (Move Up and Move Down) that allow you to manipulate the subnodes display order.

Model Field Group Node Properties

▼ [none]

Model Field Group Node Subnodes

There are zero or more model field subnodes for each model field group node

Model Field Node

This provides direct access to the model field properties and editing actions.

Model Field Node Contextual Menu Commands

▼ [action] Cut, Copy, and Paste

This provides conventional cut, copy, and paste behavior. Upon a move of a model field to another model, any visual components bound to that field will have their binding eliminated.

▼ [action] Rename

This renames the model field node. Upon a rename of a model field any visual components bound to that field will have their binding updated.

▼ [action] Delete

This deletes the model field node. Upon a delete of a model field, any visual components bound to that field will have their binding eliminated.

Model Field Node Properties

All model field nodes share a few common properties which will be described here. Each model field also has an arbitrary number of specific properties which the IDE dynamically discovers and presents for your configuration within the property sheet.

▼ [property] Name

This specifies the model field's instance name. The IDE also factors the value of the name property into the code it generates to manage the construction and access to this model field within the page's Java source file.

Model Operation Group Node

This is a node that parents a group of zero or more model operation nodes.

Model Operation Group Node Contextual Menu Commands

▼ [action] Add model operation

This adds a new model operation to the group. A new subnode is added for the model operation. The name of the field defaults to a unique name derivative of a basename defined by the model component author. The developer might immediately change that name as desired. The addition of the model operation updates the generated code section of the model's Java file.

▼ [action] Paste

This pastes a model operation.

▼ [action] Change Order

This displays a dialog with a list of the current node's subnodes. The dialog has buttons (Move Up and Move Down) that allow you to manipulate the subnodes display order.

Model Operation Group Node Properties

▼ [none]

Model Operation Group Node Subnodes

There are zero or more model operation subnodes for the model operation group.

Model Operation Node

Provides direct access to the model operation properties and editing actions.

Model Operation Node Contextual Menu Commands

▼ [action] Cut, Copy, and Paste

This provides conventional cut, copy and paste behavior. Upon a move of a model operation to another model, any non-visual components indicating this operation name in a property are not updated. It is up to the developer to manually reset operation name value on non-visual components.

▼ [action] Rename

This renames the model operation node. Upon a rename of a model operation, any non-visual components indicating this operation name in a property are not updated. It is up to the developer to manually reset operation name value on non-visual components.

▼ [action] Delete

This deletes the model field node. Upon a delete of a model field, any non-visual components indicating this operation name in a property are not updated. It is up to the developer to manually reset operation name value on non-visual components.

Model Operation Node Properties

All model operation nodes share a few common properties which will be described here. Each model operation also has an arbitrary number of specific properties which the IDE dynamically discovers and presents for your configuration within the property sheet.

▼ [property] Class

This specifies this model operation's fully qualified class name. This property is non-editable.

▼ [property] Name

This specifies the model operations's instance name. The IDE also factors the value of the name property into the code it generates to manage the construction and access to this model operation within the page's Java source file.

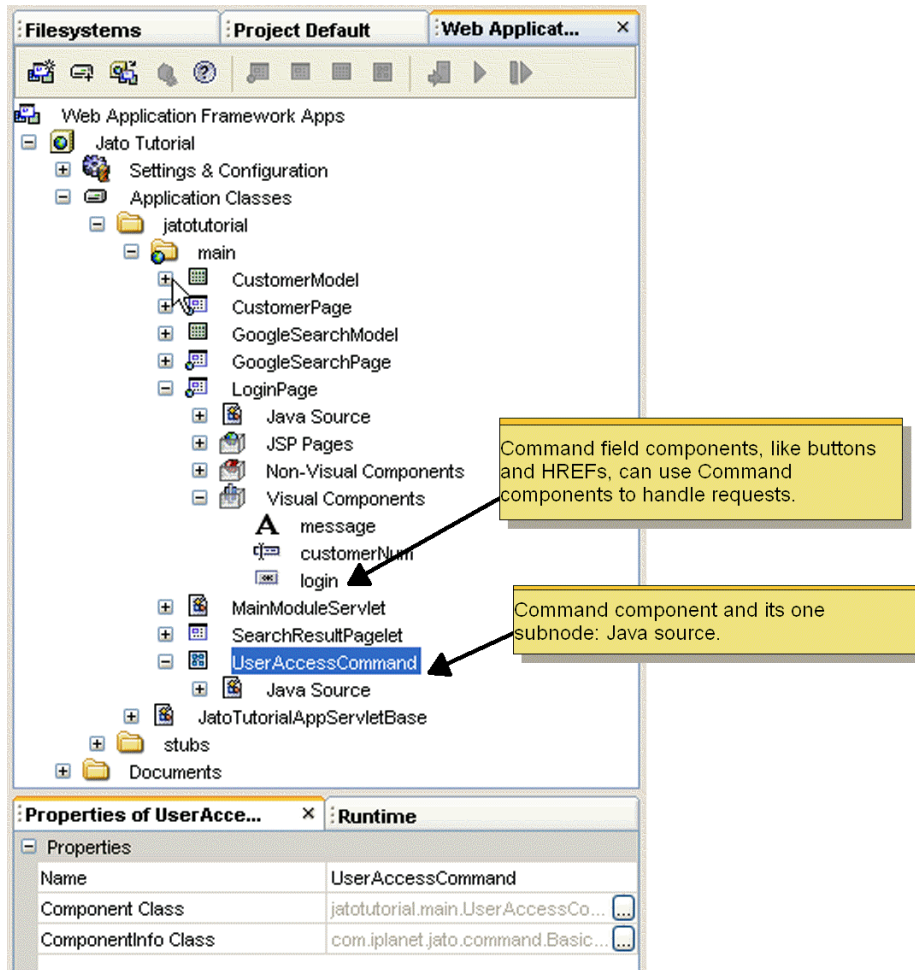
Command Component Node

Commands are the controller components of your application. A command can be created to handle specific types of actions when a command field (button or HREF) is clicked, for example.

A single command component can be reused by many different command fields throughout your application. Behavior can be further customized, based upon parameters that are passed into the command component by the many command fields using the command component.

For more details on command component authoring, consult the *Web Application Framework Developer's Guide* and the *Web Application Framework Component Author's Guide*.

The following figure shows examples of command component nodes.



Command Component Node Contextual Menu Commands

▼ [action] Open

This opens the component's Java source file and jumps to the class in the Source Editor.

▼ [action] Events

This launches a submenu which lists the event handler methods suitable for implementation in this component. The names of already implemented event handlers appear in bold. Those not yet implemented are not in bold.

When you select a bold event handler from the list, the IDE opens the component's Java source and positions your cursor at the start of the event handler. When you select a non-bold event handler, the IDE adds the corresponding event handler's method stub to the component's Java source file and position your cursor above it.

The list of available event handler methods is component specific and declared by the component from which the current component is derived. Some components will not declare event handlers. This menu command is inactive if the component does not declare any event handlers.

▼ [action] Component Methods

This launches a submenu which lists useful component methods available for this component to implement. The names of already implemented component methods appear in bold. Those which have not yet been implemented are not in bold.

When you select a bold component method from the list, the IDE opens the component's Java source and positions your cursor at the start of the method. When you select a non-bold component method the IDE adds the corresponding method stub to the component's Java source file and positions your cursor above it.

The list of available component methods is component specific and declared by the component from which the current component is derived. Some components will not declare component methods.

Typically, component methods are a subset of available base class methods which are specifically intended for override by subclasses and deemed important enough by the library component's author to merit exposure in this convenient manner. This menu command is inactive if the component does not declare any event handlers.

▼ [action] Rename

This sets the name of the component class and renames the component's Java source file.

▼ [action] Delete

This deletes the component. This command deletes the component's Java source file.

▼ [action] Cut, Copy, and Paste

This provides the conventional cut, copy, and paste behavior.

Command Component Node Properties

▼ [property] Component Class

This specifies this component's fully qualified class name. This property is non-editable.

▼ [property] ComponentInfo Class

This specifies this component's fully qualified ComponentInfo class. This property is non-editable.

▼ [property] Name

This sets the name of the component class.

Command Component Subnode

Command nodes only have one subnode: the Java Source node.

Java Source Node

The Java source node provides node level access to this component's Java source file. The Web Application Framework component node parents the Java source node to emphasize that the application component is logically comprised of more than just the Java node. The java source node is identical to the conventional Sun Java Studio Enterprise 7 Java source node with the following exceptions.

- It is parented by a Web Application Framework component node.
- Its node name is fixed as *Java Source*.
- Its contextual menu cut, delete and rename commands are disabled. Use the component's cut, delete, and rename commands instead.

The Java Source node contextual menu commands and properties are described fully in the Sun Java Studio Enterprise 7 documentation.

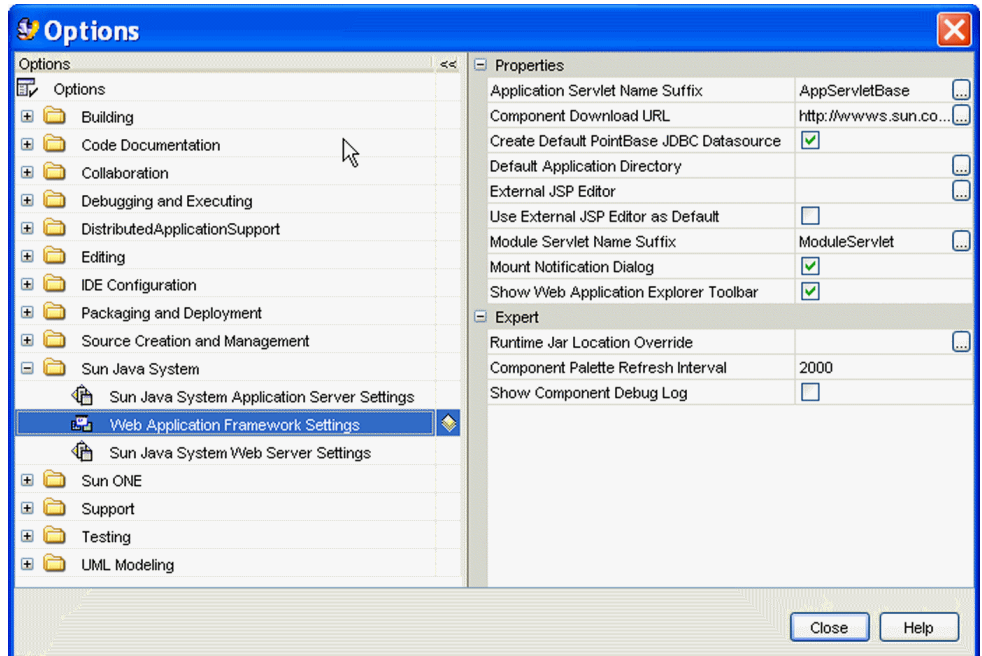
Sun Java Studio Enterprise 7 Tool Options

This chapter provides an overview of the Sun Java Studio Enterprise 7 tool options.

Like all IDEs, the Sun Java Studio Enterprise 7 has general configuration options that apply to the entire IDE and to specific modules of the IDE, like the Web Application Framework, for example. This section focuses on those properties that can be configured for the Web Application Framework module.

Select Tools -> Options from the IDE's main menu.

The *Options* window displays, as shown in the following figure.



With all the top level nodes collapsed (closed), locate the Sun Java System node and expand it. Then select the Web Application Framework Settings node. This displays the property sheet tabs (Properties and Expert) on the right portion of the Options window.

Properties Property Sheet

The following list of properties reside in the Properties window of the Web Application Framework Settings property sheet.

▼ [property] Application Servlet Name Suffix

When you create a new Web Application Framework application using the new application wizard, you are prompted to provide a name for the a servlet class known as the app servlet class of your Web Application Framework application. The default suffix appended to this class is *AppServletBase*. However, this is merely a convention and not a requirement.

This property allows you to customize the suffix that is appended to the name of all app servlet class names when you create a new Web Application Framework application.

▼ [property] Component Download URL

Specifies a default URL (<http://www.sun.com/sunone/>) that is used when the Download Components contextual menu option of the Web Application Framework root node (in the Web Application Framework Apps window) is selected.

The purpose of this URL is meant to provide a shortcut to a website that would contain a repository of third party Web Application Framework components that developers could download and use in their Web Application Framework applications. At the time this document was authored, no such web-based component repository has been constructed.

▼ [property] Create Default PointBase JDBC Datasource

When you create a new Web Application Framework application using the new application wizard, a default JDBC Datasource object, name `jdbc/jdbc-pointbase`, is created for you. This datasource is preconfigured to access the sample PointBase database that ships with the Sun Java Studio Enterprise 7 software. See JDBC Datasources for more details.

Setting this property to *False* prevents the new application wizard from creating the default JDBC Datasource. There is no negative side effect or runtime performance penalty of having this JDBC Datasource exist in your Web Application Framework application. In fact, the JDBC Datasource is only used for design-time activities.

▼ [property] Default Application Directory

When you create a new Web Application Framework application using the new application wizard, you are prompted for a location (directory) in which to locate your application. The default location is your Sun Java Studio Enterprise 7 user directory. This default location will vary depending upon the platform (Unix, Linux, or Windows) and is configured when the Sun Java Studio Enterprise 7 is installed.

This property allows you with the opportunity to provide a custom default location when you create a new Web Application Framework application. This saves you the trouble of navigating to your preferred application development location with every new application.

This property is also used for the default location when you attempt to mount a Web Application Framework application. For more information on creating and mounting Web Application Framework applications, refer to Section 1 of this document.

▼ [property] External JSP Editor

If editing the raw contents of the JSP is out of your domain of expertise and you would rather use a 3rd party HTML/JSP WYSIWYG editor, you can specify the executable path to the desired application in the Web Application Framework setting's External JSP Editor property by selecting Tools | Options from the main menu of the IDE, then expand the Sun Java System node, and select Web Application Framework Settings.

When you right-click a JSP in your Web Application Framework application, you can select the Edit in External Editor action to start the external editor and load the JSP. Be careful not to have the JSP open in both the external editor and the Sun Java Studio Enterprise 7 at the same time to avoid file locking/overwriting issues.

▼ [property] Module Servlet Name Suffix

When you create a new Web Application Framework application using the new application wizard, or when you create a new module folder, you are prompted to provide a name for the a servlet class known as the module servlet class. See module folder node in this document for more details. The default suffix appended to this class is `ModuleServlet`. However, this is merely a convention and not a requirement.

This property allows you to customize the suffix that is appended to the name of all module servlet class names when you create a new module servlet class.

▼ [property] Mount Notification Dialog

When set to *True*, the *mount notification* dialog is shown when a Web Application Framework application filesystem is mounted.

▼ [property] Show Web Application Explorer Toolbar

When set to *True*, the Web Application Framework toolbar is docked at the top of the Web Application Framework Apps window in the Explorer window.

▼ [property] Use External JSP Editor as Default

If you have specified an external JSP editor in the *External JSP Editor* property, then setting this option to *True* launches that external editor as the default editor instead of the Sun Java Studio Enterprise 7 editor. This means that double-clicking a JSP (with or without jato tag markup) anywhere within the context of a Web Application Framework application opens it in the specified external editor.

If you want to open a JSP in the Sun Java Studio Enterprise 7 editor, simply right click the JSP and select the *Open* contextual menu command. If you are in a Web application that is not a Web Application Framework application, the default is to open it in the Sun Java Studio Enterprise 7 editor. In other words, this property only affects JSPs inside a Web Application Framework application.

Expert Property Sheet

The following list of properties reside in the Expert window of the Web Application Framework Settings property sheet.

▼ [property] Component Palette Refresh Interval

Controls how often the Web Application Framework Component Palette refreshes its contents. If you were to add a new component library that contained third party Web Application Framework components, or when you create new page, pagelet and model components in your Web Application Framework application, those components are exposed in the Component Palette on the next refresh interval.

You do not want, nor is there any need, to set this refresh rate to check more frequently than the default. You do not want to overwhelm the IDE with refreshing the Component Palette every five milliseconds.

▼ [property] Runtime JAR Location Override

The location of a Web Application Framework runtime file JAR (*jato.jar*) to be used in place of the one that is shipped with the Sun Java Studio Enterprise 7 tools module.

▼ [property] Show Component Debug Log

When set to *True*, a log file in the IDE is shown that might help component developers debug their components or component libraries.

Index

A

- Add Connection action, 23
- Application Classes Node, 50
- Application Classes Node Contextual Menu
 - Commands, 51
- Application Classes Node Contextual Menu
 - Commands, action
 - Build, 52
 - Build All, 52
 - Compile, 51
 - Compile All, 51
- Application Classes Node Properties, 52
- Application Classes Node property
 - Show Module Packages Only, 52
- Application Classes Subnodes, 52
- Application Framework Datasource Node
 - Contextual Menu Commands, action
 - Delete, 47
- Application Node Contextual Menu Commands, action
 - Add Component Library, 29
 - Build, 30
 - Build All, 30
 - Compile, 30
 - Compile All, 30
 - Deploy, 29
 - Execute, 28
 - Execute (Force Reload), 29
 - Export WAR File, 31
 - Properties, 31
 - Rename, 30
 - Tools, 31

- Unmount Application, 31
- Application Node property
 - Content Language, 31
 - Context Root, 31
 - Debugger, 32
 - Executor, 32
 - Extra Files, 32
 - Filter, 32
 - Name, 31
 - Target Server, 32
 - Template, 32
- appMessage method, 42

B

- base Web application directory, 19
- buried nodes, 20

C

- Command Component Node, 89
- Command Component Node Contextual Menu
 - Commands, 90
- Command Component Node Contextual Menu
 - Commands, action
 - Cut, Copy, and Paste, 91
 - Delete, 91
 - Events, 90
 - Open, 90
 - Rename, 91
- Command Component Node Properties, 92
- Command Component Node, property
 - Component Class, 92
 - ComponentInfo Class, 92

- Name, 92
- Command Component Subnode, 92
- Component Libraries Node, 44
- Component Libraries Node Contextual Menu Commands, 45
- Component Libraries Node Contextual Menu Commands, action
 - Add Component Library, 45
- Component Libraries Node Properties, 45
- Component Libraries Subnodes, 45
- connections and RDBMS drivers, listed in Databases node, 22
- connections, connected or disconnected, 22
- create/open project, before doing, 15
- Creating a New Sun Java Studio Project, 15
- customize development environments, 15

D

- Databases, 21
- Databases Node, 22
- Deployment Descriptor Node, 43
- Deployment Descriptor Node Contextual Menu Commands, 44
- Deployment Descriptor Node Contextual Menu Commands, action
 - Edit, 44
- Deployment Descriptor Node Properties, 44
- Design-Time Resources Node, 45, 48
- Design-Time Resources Node Contextual Menu Commands, 45
- Design-Time Resources Node Properties, 46
- Design-Time Resources Subnodes, 46
- development environments, customize, 15
- directories, currently mounted, displayed in
 - Filesystems window, 17
- directory
 - raw, and file layout, 19
 - Web Context, 19
- Disconnect action, 23
- documentation
 - NetBeans-based Sun Java Studio Enterprise 7, 13
- Documents Node, 49
- Drivers folder node
 - disabled, 22
 - enabled, 22

- Drivers folder subnode, 22

E

- Enabled Log Level editor, 41
- Expert Property Sheet, 96
- Expert window, property
 - Component Palette Refresh Interval, 96
 - Runtime JAR Location Override, 96
 - Show Component Debug Log, 97

F

- File Upload
 - Node, 37
 - Properties, 38
- File Upload Node
 - Contextual Menu Commands, 38
- File Upload property
 - Auto Delete Temp Files, 38
 - Default Character Encoding, 39
 - Enable File Upload Servlet Filter, 38
 - File Size Limit, 38
 - File Size Limit (Hard), 38
 - Request Failure Redirect URL, 39
 - Request Size Limit, 39
 - Temp File Directory, 39
 - Use Request Character Encoding, 39
 - Use Temp Files, 39

- Filesystems Window, 17

G

- General Node, 34
- General Node Contextual Menu Commands, 34
- General Node Properties, 34
- General Node property
 - Generate Unique URLs Node, 34
 - Strict Session Timeout Node, 34

J

- JAR files, 19
- JAR files, when you manually copy to WEB-INF/lib directory, 19
- JAR in multiple Web Application Framework applications, 16
- JatoTutorial, 13
- Java Package Folder Node, 52

- Java Package Folder Node Contextual Menu
 - Commands, 52
- Java Package Folder Node Contextual Menu
 - Commands, action
 - Add, 53
 - Build, 53
 - Build All, 53
 - Compile, 52
 - Compile All, 53
 - Convert to Module, 53
- Java Package Folder Properties, 54
- Java Package Folder Properties property
 - Module, 54
 - Name, 54
 - Sort Mode, 54
- Java Source Node, 63, 85, 92
- JDBC Datasources Node, 46
- JDBC Datasources Node Contextual Menu
 - Commands, 46
- JDBC Datasources Node Contextual Menu
 - Commands, action
 - Add JDBC Datasource, 46
- JDBC Datasources Node Properties, 46
- JDBC Datasources Subnodes, 46
- JSP Node, 66
- JSP Node Contextual Menu Commands, 68
- JSP Node Contextual Menu Commands, action
 - Delete, 69
 - Set as Default, 69
 - Synchronize to View, 68
- JSP Page Node Properties, 70
- JSP Page Node property
 - JSP URI, 70
- JSP Pages Node, 64
- JSP Pages Node Contextual Menu Commands, 64
- JSP Pages Node Contextual Menu Commands action
 - Add JSP, 65
 - Associate JSP, 64
 - Change Order, 65
- JSP Pages Node Properties, 66
- JSP Pages Node property
 - Default JSP, 66
- JSP Pages Subnodes, 66

L

- lib/ext directory, contains drivers available to
 - Studio, 22
- libraries, additional, 19
- Logging Node, 40
- Logging Node Contextual Menu Commands, 40
- Logging Node Properties, 40
- Logging Node property
 - Enable console Logging, 40
 - Enabled Log Levels, 40
 - Log Message Prefix, 42
 - Show Message Buffer, 42

M

- message buffer, 42
- Model Component Node, 81
- Model Component Node Contextual Menu
 - Commands, 82
- Model Component Node Contextual Menu
 - Commands, action
 - Component Methods, 83
 - Cut, Copy, and Paste, 84
 - Delete, 84
 - Design Actions, 83
 - Events, 83
 - Open, 82
 - Rename, 83
- Model Component Node Properties, 84
- Model Component Node property
 - Component Class, 84
 - ComponentInfo Class, 84
 - Name, 84
- Model Component Subnodes, 84
- Model Field Group Node, 85
- Model Field Group Node Contextual Menu
 - Commands, 85
- Model Field Group Node Contextual Menu
 - Commands action
 - Add model field, 85
 - Change Order, 86
 - Paste, 86
- Model Field Group Node Properties, 86
- Model Field Group Node Subnodes, 86
- Model Field Node, 86
- Model Field Node Contextual Menu Commands, 86

- Model Field Node Contextual Menu Commands, action
 - Cut, Copy, and Paste, 86
 - Delete, 86
 - Rename, 86
- Model Field Node Properties, 87
- Model Field Node property
 - Name, 87
- Model Operation Group Node, 87
- Model Operation Group Node Contextual Menu Commands, 87
- Model Operation Group Node Contextual Menu Commands, action
 - Add model operation, 87
 - Change Order, 87
 - Paste, 87
- Model Operation Group Node Properties, 88
- Model Operation Group Node Subnodes, 88
- Model Operation Node, 88
- Model Operation Node Contextual Menu Command, 88
- Model Operation Node Contextual Menu Command, action
 - Cut, Copy, and Paste, 88
 - Delete, 88
 - Rename, 88
- Model Operation Node Properties, 88
- Model Operation Node property
 - Class, 89
 - Name, 89
- Module Folder Node, 54
- Module Folder Node Contextual Menu Commands, 55
- Module Folder Node Contextual Menu Commands, action
 - Add, 56
 - Build, 56
 - Build All, 56
 - Compile, 55
 - Compile All, 56
- Module Folder Node Properties, 56
- Module Folder Node property
 - Module, 56
 - Module Servlet Mapping, 57
 - Sort Mode, 57

- mount multiple Web Application Framework applications, 16
- mounting Application Framework application, 23
- multiple Web Application Framework applications, mount, 16

N

- NetBeans online documentation, 15
- NetBeans-based Sun Java Studio Enterprise 7 documentation, 13
- Non-Visual Component Node, 73
- Non-Visual Component Node Contextual Menu Commands, 73
- Non-Visual Component Node Contextual Menu Commands, action
 - Cut, Copy, and Paste, 73
 - Delete, 73
- Non-Visual Component Node Properties, 73
- Non-Visual Component Node property
 - Access Privilege, 74
 - Component Class, 74
 - Name, 74
 - Scope, 74
 - Session Attribute Name, 74
- Non-Visual Components Node, 70
- Non-Visual Components Node Contextual Menu Commands, 72
- Non-Visual Components Node Contextual Menu Commands, action
 - Add Non-Visual Component, 72
 - Change Order, 72
- Non-Visual Components Node Properties, 73
- Non-Visual Components Subnodes, 73

O

- Opening a Sun Java Studio Project, 16
- Options window, Sun Java Studio, 93
- overview, Sun Java Studio Enterprise 7 IDE, 13
- Overview, Web Apps Window, 25

P

- Page Component Node, 59
- Page Component Node Contextual Menu Commands, 61
- Page Component Node Contextual Menu Commands, action

- Component Methods, 61
- Cut, Copy, and Paste, 62
- Delete, 62
- Edit Default JSP, 61
- Error Information, 61
- Events, 61
- Execute Page, 62
- Execute Page (Redeploy), 62
- Open, 61
- Rename, 62
- Page Component Node Properties, 63
- Page Component Node property
 - Component Class, 63
 - ComponentInfo Class, 63
 - Name, 63
- Page Component Subnodes, 63
- Pagelet Component Node, 80
- Pagelet Component Node Contextual Menu
 - Commands, 80
- Pagelet Component Node Properties, 81
- Pagelet Component Subnodes, 81
- Preface, 7 to 12
- primary nodes, Settings & Configuration,
 - Application Classes, and Documents, figure, 25
- primary nodes, Web Application Framework Apps
 - window, 25
- Processes, 21
- Processes Node, 22
- processes, launched and running - in Processes
 - Node, 22
- Project Window, 19
- project, before create/open, 15
- project, default Studio, 15
- Properties Property Sheet, 94
- Properties window, property
 - Application Servlet Name Suffix, 94
 - Component Download URL, 94
 - Create Default PointBase JDBC Datasource, 94
 - Default Application Directory, 95
 - External JSP Editor, 95
 - Module Servlet Name Suffix, 95
 - Mount Notification Dialog, 95
 - Show Web Application Explorer Toolbar, 96
 - Use External JSP Editor as Default, 96

R

- raw directory and file layout, 19
- RDBMS drivers, 22
- refresh rate setting dependency, Studio
 - configuration, 19
- Root Node Contextual Menu Commands, action
 - Download Components, 28
 - Mount Web Application Framework App, 27
 - New Web Application Framework App, 27
- Root Node Contextual Menu Commands, Web
 - Application Framework Apps, 27
- Root Node, Web Application Framework Apps, 27
- Runtime Window, 21

S

- sandbox, IDE, 15
- Server Registry, 21
- Server Registry Node, 22
- servlet containers, Tomcat and Application
 - Server, 22
- servlet context, 19
- session timeout message, 34
- Settings & Configuration Node, 33
- Settings & Configuration Node Contextual Menu
 - Commands, 33
- Settings & Configuration Node Properties, 33
- Settings & Configuration Subnodes, 34
- Special Note on Pagelet JSP Nodes, 81
- Strict Session Timeout property set to false, 36
- Strict Session Timeout property setting, figure, 36
- Studio, projects, 15
- Sun Java Studio Enterprise 7 IDE Overview, 13
- Sun Java Studio Enterprise 7 Tool Options, 93
- Sun Java Studio Overview, 13 to 23
- Sun Java Studio Project, creating new, 15
- Sun Java Studio Project, opening, 16
- Sun Java Studio Tool Options, 93 to 97
- Sun Java Studio Workspaces, 16

T

- Templates Node, 47
- Templates Node Contextual Menu Commands, 47
- Templates Node Properties, 47
- Templates Subnodes, 47

Tomcat in Server Registry Node, 22

tool options, 93

U

user friendly view of Web application, Web Apps window provides, 23

V

vendor database driver versions, listed in Drivers

Folder subnode of Databases node, 22

view, Sun Java Studio Enterprise 7 showing Web Application Framework Workspace, 13

Visual Component Node, 78

Visual Component Node Contextual Menu Command, 76

Visual Component Node Contextual Menu Command, action

Add Visual Component, 76

Change Order, 78

Paste, 78

Visual Component Node Contextual Menu Commands, 78

Visual Component Node Contextual Menu Commands, action

Events, 78

Visual Component Node Properties, 78, 79

Visual Component Node property

Component Class, 79

ComponentInfo Class, 79

Name, 79

Post-initialization Code, 80

Pre-initialization Code, 80

Visual Component Subnodes, 78

Visual Components Node, 75

W

Web Application Framework Application Node, 28

Web Application Framework Apps Window

Overview, 25 to 57

Web Application Framework Component Nodes, 59 to 92

Web Context directory, 19

Web Server, Default - property, 32

Windows, Studio, 16