

SYBASE®

Working with JSP Targets

PowerBuilder®

11.0

DOCUMENT ID: DC77883-01-1100-01

LAST REVISED: May 2007

Copyright © 1991-2007 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase trademarks can be viewed at the Sybase trademarks page at <http://www.sybase.com/detail?id=1011207>. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book	ix	
CHAPTER 1	Working with JSP Targets	1
	About JavaServer Pages	1
	How JavaServer Pages work	2
	What a JSP contains	2
	Application logic in JSPs	3
	About JSP targets	3
	Object models	4
	What kinds of applications can you develop?	5
	Do you need to know Java or HTML?	6
	Advantages of the JSP target development environment	6
	JSP Web Target object model classes.....	7
	Using the JSP target development environment	9
	About the editors	12
	Tools for editing.....	12
	System options and editor preferences.....	14
	Working in an integrated Web delivery environment.....	15
CHAPTER 2	Developing JSP Applications	19
	Setting up JSP targets	19
	JSP Target wizard	20
	Creating a JSP target	22
	Adding deployment configurations	23
	Defining connection profiles	25
	Importing files from an existing Web site.....	26
	Working with files in a JSP target.....	26
	Types of files	26
	Adding content	28
	Using the System Tree	29
	Creating Web pages	34
	Adding content and style to your Web pages.....	34
	Building and deploying JSP targets.....	35

CHAPTER 3	Working with the HTML Editor	37
	HTML editor views	37
	Page view	37
	Source view	40
	Preview view	41
	Choosing a view to work in.....	42
	Switching between views	42
	Opening the HTML editor and setting options.....	43
	Opening the HTML editor	43
	Saving your work and closing the editor.....	45
	Giving your page a title.....	46
	Formatting HTML source display	47
	Basic editing in Page and Source views	49
	Using the PowerBuilder menu.....	49
	Formatting tips.....	51
	Using the System Tree.....	51
	Properties for HTML elements	53
	Undo and Redo	56
	Finding and changing text	56
	Using the Script editor	57
	Correspondences of common elements	57
	Headings and paragraphs	57
	Lists.....	59
	Character formatting.....	61
	Inserting special symbols	62
	Links and anchors	62
	More complex formatting.....	64
	Absolute positioning	67
	About absolute positioning	67
	What you can do	67
	Toggling between static and absolute positioning.....	69
	Setting absolute positioning options.....	70
	Manipulating an absolutely positioned element.....	70
	JSP page authoring.....	71
	JSP actions	72
	Directives.....	76
	Custom tags	79
	Error handling.....	80
CHAPTER 4	Working with Style Sheets and Framesets	83
	About style sheets.....	83
	Working with styles.....	84
	Syntax for style attributes and selectors.....	84
	Working with IDs and classes	85

About the JSP Target style and style sheet editors	85
The Style Sheet editor tab page interface	87
Integration with other JSP target editors	87
Basic editing with the style sheet editors	88
Creating an external style sheet.....	89
Importing an existing style sheet	89
Linking an external style sheet to an HTML page	91
Embedding style definitions in an HTML page	92
Opening an existing style sheet	92
Using the Inline Style editor.....	93
Adding selectors for HTML elements, classes, and IDs.....	94
Removing items from a style sheet	98
Editing frames and framesets	98
About the Frameset editor	98
Creating a new frameset document	99
Modifying a frameset	100
Modifying frame properties.....	101

CHAPTER 5 Working with Images, Other Media, and Components 103

Images and image maps.....	103
Inserting images	103
Creating image maps	106
Multimedia.....	107
Components.....	108
Viewing available components	109
Inserting a component.....	109
Design-time controls.....	111
The Java class path	112
Class path values	112
Using the class path	113
The custom tag library search path.....	113

CHAPTER 6 Writing Scripts 115

About scripts	115
Editing scripts	115
Scripting languages.....	117
Types of scripts	117
Objects in an HTML document.....	120
JSP scripting elements.....	121
Inserting a scripting element	122
Implicit objects	122
Scopes	124

	Procedures for editing scripts.....	124
	Choosing an object or event for scripting.....	124
	Assigning an ID to an object in the document.....	125
	Creating a new script.....	125
	Writing the code.....	126
	Finding and changing code.....	130
	Setting default formats for scripts in the Script editor.....	130
	Techniques and tips for creating scripts.....	131
	Position of scripts.....	131
	URLs in scripts.....	131
CHAPTER 7	Working with Application Servers and Transaction Servers ..	133
	Integrating with application servers.....	133
	Working with server scripts.....	135
	Using the Web Target object model.....	136
	Accessing database content from your JSP target.....	139
	Managing page data.....	140
	About page parameters and variables.....	140
	Using page parameters in server scripts.....	141
	Using session variables in server scripts.....	143
	Samples for retrieving and displaying data.....	145
	Integrating with EAServer.....	150
	Accessing components.....	151
CHAPTER 8	JSP Web Services.....	155
	About JSP Web services.....	155
	Using the JSP Web Service Proxy wizard.....	155
	Files added by the wizard and files required by the server ...	156
	Using the UDDI browser in the wizard.....	157
	SOAP processing in JSP targets.....	159
	Adding a custom tag for Web services.....	160
CHAPTER 9	Developing 4GL JSP Pages.....	163
	About 4GL JSP pages.....	163
	Developing pages.....	164
	Creating a new 4GL JSP page.....	165
	Enabling 4GL mode in an existing page.....	167
	Adding content to 4GL JSP pages.....	169
	Using parameters and variables.....	169
	Setting up page parameters.....	171
	Setting parameter bindings on the linking page.....	172
	Setting up page and session variables.....	173

Accessing EAServer components	174
About EAServer integration	175
Working with EAServer components	175
Setting up EAServer login variables	177
Adding controls	178
Binding controls to properties of EAServer components	179
Binding controls to page data	181
Disabling server scripting for a control	182
Writing server scripts	183
Responding to events on your page	184
Adding scripts to 4GL JSP pages	186
Writing scripts to access EAServer components	188
How page request processing works	189
Disabling 4GL mode	191

CHAPTER 10	Setting Up Page Navigation	193
	About page navigation	193
	Managing client hyperlinks	196
	Managing client form submission	197
	Managing server redirection	200

CHAPTER 11	Using the Web DataWindow Design-Time Control	203
	About the Sybase Web DataWindow DTC	203
	Web DataWindow support	203
	Server-side environment	206
	Benefits of using the Web DataWindow DTC	207
	Adding a DataWindow to a JSP page	207
	Creating a page that has a Web DataWindow DTC	207
	What you see in Page view	209
	What you see in Source view	210
	What you see in a non-4GL JSP page	211
	What you see in a 4GL JSP page	212
	Using Web Target object model classes	213
	Setting Web DataWindow DTC properties	213
	Selecting the source for a DataWindow object	213
	Selecting a database profile	215
	Controlling the behavior of the DTC	217
	Setting the bind type and values for retrieval arguments	219
	Defining links	220
	Selecting a Web DataWindow generator	221
	Editing existing Web DataWindow DTC properties	222
	DataWindow presentation styles and data sources	223

- Binding data to DataWindow retrieval arguments 224
 - Constants 225
 - Control Values 225
 - JavaScript Expressions 226
 - Page Parameters 227
 - Variables 229
- Defining hyperlinks on objects in a DataWindow 230
- Custom tag library for the Web DataWindow 232

- CHAPTER 12 Building and Deploying JSP Targets 233**
 - About building and deploying JSP targets 233
 - Building JSP targets 234
 - The deployment process 237
 - Working with JSP servers 237
 - Setting up a deployment configuration 238
 - Editing a JSP deployment configuration 239
 - General deployment options 239
 - JSP deployment options 241
 - Enterprise Portal deployment options 251
 - Deploying a JSP target 254
 - Running a JSP target 255
 - Troubleshooting 4GL JSP pages 255
 - Displaying runtime errors 255
 - Displaying trace messages 257
 - Troubleshooting JSP targets 259
 - Problems deploying and running JSPs 259
 - Troubleshooting JSP Web services 260
 - Additional resources for JSPs and Web services 261
- Index 263**

About This Book

Audience	This book is for developers who build Web applications using Java Server Pages (JSP).
How to use this book	This book provides an overview of the JSP target features of PowerBuilder®.
Related documents	Reference information for these features is available in the <i>JSP Target Reference</i> , the <i>DataWindow Reference</i> , and the online Help.
Other sources of information	<p>Use the Sybase Getting Started CD, the SyBooks CD, and the Sybase Product Manuals Web site to learn more about your product:</p> <ul style="list-style-type: none">• The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.• The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format. Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader. Refer to the <i>SyBooks Installation Guide</i> on the Getting Started CD, or the <i>README.txt</i> file on the SyBooks CD for instructions on installing and starting SyBooks. <p>• The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network. To access the Sybase Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.</p>

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

- 1 Point your Web browser to Technical Documents at <http://www.sybase.com/support/techdocs/>.
- 2 Click Certification Report.
- 3 In the Certification Report filter select a product, platform, and timeframe and then click Go.
- 4 Click a Certification Report title to display the report.

❖ **Finding the latest information on component certifications**

- 1 Point your Web browser to Availability and Certification Reports at <http://certification.sybase.com/>.
- 2 Either select the product family and product under Search by Product; or select the platform and product under Search by Platform.
- 3 Select Search to display the availability and certification report for the selection.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

- 1 Point your Web browser to Technical Documents at <http://www.sybase.com/support/techdocs/>.
- 2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on EBFs and software maintenance**

- 1 Point your Web browser to the Sybase Support Page at <http://www.sybase.com/support>.
- 2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.
- 3 Select a product.

- 4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the “Technical Support Contact” role to your MySybase profile.

- 5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.



About this chapter

This chapter describes JSP targets and how you work with them in PowerBuilder.

Contents

Topic	Page
About JavaServer Pages	1
About JSP targets	3
JSP Web Target object model classes	7
Using the JSP target development environment	9
Working in an integrated Web delivery environment	15

About JavaServer Pages

JavaServer Pages (JSP) technology provides a quick, easy way to create Web pages with both static and dynamic content. JSPs are text-based documents that contain static markup, usually in HTML or XML, as well as Java content in the form of scripts and/or calls to Java components. JSPs extend the Java Servlet API and have access to all Java APIs and components.

You can use JSPs in many ways in Web-based applications. As part of the J2EE application model, JSPs typically run on a Web server in the middle tier, responding to HTTP requests from clients, and invoking the business methods of Enterprise JavaBeans (EJB) components on a transaction server.

JSP pages built with PowerBuilder support:

- Version 1.2 of the JavaServer Pages specification
- Version 2.3 of the Java Servlet specification

PowerBuilder supports custom tag libraries that use the JSP 1.2 format.

How JavaServer Pages work

JSP pages are executed in a JSP engine (also called a JSP container) that is installed on a Web or application server. The JSP engine receives a request from a client and delivers it to the JSP page. The JSP page can create or use other objects to create a response. For example, it can forward the request to a servlet or an EJB component, which processes the request and returns a response to the JSP page. The response is formatted according to the template in the JSP page and returned to the client.

Translating into a servlet class

In PowerBuilder, JSP pages are deployed to the server in source form. If a JSP page is in source form, the JSP engine typically translates the page into a class that implements the servlet interface and stores it in the server's memory.

Depending on the implementation of the JSP engine, translation can occur at any time between initial deployment and the receipt of the first request. As long as the JSP page remains unchanged, subsequent requests reuse the servlet class, reducing the time required for those requests.

Requests and responses

Some JSP engines can handle requests and responses that use several different protocols, but all JSP engines can handle HTTP requests and responses. The `JspPage` and `HttpJspPage` classes in the `javax.servlet.jsp` package define the interface for the compiled JSP, which has three methods:

- `jspInit()`
- `jspDestroy()`
- `_jspService(HttpServletRequest request, HttpServletResponse response)`

What a JSP contains

A JSP contains static template text that is written to the output stream. It also contains dynamic content that can take several forms:

- Directives provide global information for the page, or include a file of text or code.
- Scripting elements (declarations, scriptlets, and expressions) manipulate objects and perform computations.
- Standard tags, or actions, perform common actions such as instantiating a JavaBeans component or getting or setting its properties, downloading a plug-in, or forwarding a request.
- Custom tags perform additional actions defined in a custom tag library.

“JSP page authoring” on page 71 provides a brief description of each of these types of dynamic content. For more detailed information, see the JavaServer Pages at <http://java.sun.com/products/jsp/index.html>, or one of the many books about JavaServer Pages technology.

Application logic in JSPs

The application logic in JSPs can be provided by components such as servlets, JavaBeans, and Enterprise JavaBeans (EJBs), customized tag libraries, scriptlets, and expressions. Scriptlets and expressions hold the components and tags together in the page.

JavaBeans	You can easily use JavaBeans components in a JSP with the <code>useBean</code> tag. For more information, see “ <code><jsp:useBean></code> ” on page 73.
Enterprise JavaBeans	To use an EJB component, you need to use JNDI to establish an initial naming context for the EJB’s home interface. You could do this in a scriptlet, using a JavaBeans component or a custom tag.
Custom tag libraries	Custom tag libraries define a set of actions to be used within a JSP for a specific purpose, such as handling SQL requests. See “Custom tags” on page 79.

About JSP targets

A JSP target is a collection of files and components used to create a Web application. A Web application can represent part or all of a Web site. Web applications created using PowerBuilder JSP targets deliver dynamic, interactive content by integrating database data, scripting for client- and server-side events, and providing access to components stored on middle-tier servers. The PowerBuilder development environment simplifies how you develop and maintain JSP Web applications.

Development environment	PowerBuilder lets you develop both PowerScript targets and JSP targets in a workspace. You can add as many targets to a workspace as you want, and open and edit objects in multiple targets. If your JSP application uses <code>DataWindow®</code> and <code>EAServer</code> components, you can work with all of them in a single workspace. PowerBuilder provides an intuitive user interface, combined with wizards that step you through complex or repetitive tasks to help you develop pages faster and spend less time on Web site maintenance.
-------------------------	---

Deployment environment

You must deploy PowerBuilder Web applications to JSP page servers. JSP targets can be deployed to Apache Tomcat, Sybase EAServer, or servers compatible with JSP 1.2 or later.

The page servers can in turn access components on transaction servers such as EAServer. If you use Tomcat as a JSP server, you can still access components running on EAServer in your Web applications. With JSP targets, you can use 4GL Web technology to manage page data and easily integrate middle-tier components into Web pages.

You can choose to deploy a JSP target as a Web application to:

- EAServer (see the PowerBuilder Release Bulletin for supported versions)
- Apache Tomcat 4.1.13 and later
- Any other JSP 1.2 server for which you can configure command-line deployment capabilities
- Sybase Enterprise Portal instead of a JSP server

For more information, see the JavaServer Pages specification at <http://java.sun.com/products/jsp/index.html>, and the Java Servlets specification at <http://java.sun.com/products/servlet/index.html>.

For more information on 4GL Web technology, see Chapter 9, “Developing 4GL JSP Pages.”

Object models

What is an object model?

Object models provide developers with a scripting environment by creating and instantiating objects—and exposing their properties, methods, and events.

In a distributed environment such as the Web, object models can be client side or server side. In a client script, objects belong to the Document Object Model of the client browser. In a server script, objects belong to the object model existing on or deployed to the selected server.

You can view representations of client-side and server-side object models in the Language page of the System Tree.

About the Web Target object model

The Web Target object model streamlines the process of developing and deploying JSP applications. During the development phase, the object model hides many of the platform-specific details you would otherwise need to know to write server pages. At deployment time, the object model takes care of mapping your platform-independent code to each application server platform you choose to target.

The structures and objects in the Web Target object model are defined in Java classes for JSP targets.

When you deploy a JSP page that uses the Web Target object model, the target automatically adds an object model file to your deployed application and imports the contents of that file into your page. The object model file resolves references you make to JSP target objects to appropriate objects in the target application server. The deployment controller imports an object model file into any JSP file containing one or more server scripts that use the Web Target object model.

For more information about the Web target object model, see and “Using the Web Target object model” on page 136.

What kinds of applications can you develop?

The applications you build with PowerBuilder for JSP targets can include simple text-based pages as well as complex JSP pages with:

- Client- and server-side scripting
- Database content
- Web DataWindows
- Components, such as EAServer components (including Enterprise JavaBeans) or ActiveX controls
- Component transaction server access

Dynamic, data-driven applications

Dynamic business-critical Web applications typically use application servers to display data stored in a database and present interactive interfaces through which users execute business transactions in real time. The Web DataWindow, easily created within the JSP Target development environment, gives your page real-time access to databases for retrieval and update.

Open applications

JSP targets you build in PowerBuilder support an open architecture. The basic Web Target object model supports server-side programming for multiple application servers, enabling you to develop JSP targets for deployment to multiple servers. To provide dynamic content for your Web applications, you can create server-side scripts in Java for JSP targets, or you can create client-side scripts in JavaScript or any ECMA-compliant script (VBScript, JScript, and others).

4GL applications

4GL extensions to the Web Target object model provide server-side event processing and generate server-side code automatically from selections you make in the JSP target user interface. 4GL applications can be deployed to EAServer, Tomcat, or other JSP 1.2 or later compatible JSP servers.

Do you need to know Java or HTML?

HTML is one of the underlying technologies for your Web site, so it helps to know what it can and cannot do. However, you can edit pages in the HTML editor without knowing HTML syntax. Page view (one of three views) in the HTML editor feels more like a word processor than a code editor.

You can also create styles in the Style Sheet editor without knowing the syntax for style definitions. If you do know HTML, the editors help you create more complex HTML layouts like tables and forms.

With JSP targets, you can edit pages in the HTML editor without knowing Java syntax. When you drag and drop controls onto a page, the HTML editor adds code that you can see in the Source view. For 4GL pages, this includes Java code to construct the control using 4GL object model classes.

If you need to create scripts, you certainly need to know about the objects on your page and their events, as well as the syntax of your scripting language. In addition to providing an object view of your document, the System Tree shows you the classes, methods, and properties of the Web target.

Advantages of the JSP target development environment

The JSP target development environment simplifies the configuration and coding tasks for your applications. Wizards and dialog boxes let you provide the information an application needs while the development tool takes care of implementation details.

Simplifies Web application creation You use the JSP target wizard to create a new JSP target. The JSP Target wizard prompts you to select a JSP server and select connection properties for the server. The wizard also steps you through the Deployment Configuration wizard screens.

The Source Controlled JSP Target wizard allows you to add a JSP target to your workspace when the target is checked in to source control.

Helps automate deployment configuration After you create a JSP target, you can manually run the Deployment Configuration wizard. You access the Deployment Configuration wizard through the JSP target properties sheet. When you set up a deployment configuration, you specify the type of server you want your application to run on. You can choose either Tomcat or EAServer as your JSP server.

Dynamically extends supported object models The Web Target object model extends the programming interface for your Web pages by simplifying how you include connections to databases, Web DataWindows, and EAServer components, and how you handle error reporting. The entries you make in dialog boxes generate server scripts that you can extend and customize.

The Web Target object model supports an open architecture.

Automates link management The build process for JSP targets verifies the links between files, writing warnings for broken links or bad syntax to the Output window. It does not attempt to fix the links and it does not prevent deployment of the target files.

Enables the use of a team environment If you create JSP targets in a team environment, you can control file access through the source control system you have configured for your workspace. You work with the source control system the same way as for PowerScript targets in PowerBuilder, except that for JSP targets, you do not need to compile the files you obtain from the source control server.

JSP Web Target object model classes

Classes in the Web Target object model handle the complexities of data transfer, HTML generation, and JavaScript generation for client scripts. The non-4GL part of the JSP object model provides a set of utility Java classes that implement this functionality and encapsulate most of the JSP page's implicit objects. These object classes can be used on non-4GL Web pages as well as 4GL JSP pages. The JSP deployment controller imports the JSP object model for each JSP page it deploys.

For more information about the server scripts added by the JSP deployment controller, see “Transformations for JSP targets” on page 237.

Use of constructors For JSP targets, you must assign a variable of the correct class type before you can create an instance of the object or call methods on it. For more information about JSP constructors for Web Target object model classes, see the *JSP Target Reference*.

Object model changes for JSP targets A few Web Target object methods have not been implemented for JSP targets and are therefore obsolete. The psServer GetConnection and MapPath methods and the SetSQL method on the PSCommandClass object are not implemented for JSP targets.

The GetValue method on the PSCursor object does not return a value of a set datatype and therefore cannot be used with JSP pages. You can use the methods in Table 1-1 to return values of a specific datatype.

Table 1-1: Object model methods for JSPs

Method	Return value datatype
GetColumn<DataType>(String strColName) where <DataType> can be Boolean, Byte, Double, Float, Int, Long, Short, or String	Corresponds to <i>DataType</i> used in method name
GetColumn<DataType>(int iColNo) where <DataType> can be Boolean, Byte, Double, Float, Int, Long, Short, or String	Corresponds to <i>DataType</i> used in method name
GetColumnLength(String strColName)	int
GetColumnLength(int iCol)	int
GetColumnName(int iCol)	String
GetColumnType(int iCol)	int
GetColumnName(int iCol)	String
GetPrecision(int iCol)	int
GetResultSet()	ResultSet
GetResultSetMetaData()	ResultSetMetaData
GetScale(int iCol)	int

These methods are described in more detail in the *JSP Target Reference*. If the syntax of a method is target dependent, the *JSP Target Reference* indicates the proper syntax to use for each target type.

Server-side events When you enable 4GL functionality on a JSP page, you can rely on an event-driven infrastructure to handle many of the details of coding server scripts. The JSP 4GL object model provides foundation classes for the event-driven infrastructure, such as the server control classes, the DataWindow class, server variables, and parameter classes.

In JSP targets, you must script a return value for the server-side events of a 4GL JSP page that have a return value as part of their event signature. If there is no return statement, a servlet translation error occurs at runtime. The following server-side events on the 4GL psPage object have boolean return types: BeforeAction, BeforeGenerate, FirstTime, RequestStart, ServerError, and Validate.

You can use the SetTrace method to trace the server-side events on a generated page. The psPage object in the JSP Web Target object model also has a method for checking if tracing is on before you call other trace methods multiple times. The IsTrace method returns a boolean and takes no arguments.

Variables

When you create variables on the Variables page of the Page Properties dialog box for 4GL JSP pages, you must associate a datatype with each variable. The following variable datatypes are supported in 4GL JSP pages: boolean, byte, char, double, float, int, long, short, and String.

For more information on setting page and session variables in a 4GL JSP page, see “Setting up page and session variables” on page 173.

Using the JSP target development environment

When you work in the JSP target environment, you do so within the context of a PowerBuilder workspace. Inside this workspace a JSP target includes all of the files you need to produce a JSP application. When you create or open a JSP target, a comprehensive set of Web development features is available to you.

JSP target tools

The JSP target development environment provides the following development and authoring features:

- **System Tree** As an active resource for programming information, the System Tree lists language elements for HTML and object models for scripting. It lets you view the list of controls on a Web page and the properties and methods available to them. It also lets you view components and component methods available on EAServer servers, and custom tag libraries that you want to use with your JSP applications.
- **HTML editor** The primary development tool is the HTML editor. In it, you typically use the Page view to add controls and the Source view to edit text.

The editor lets you include lists, links and anchors, tables, forms, images, components, and other features in your Web pages.

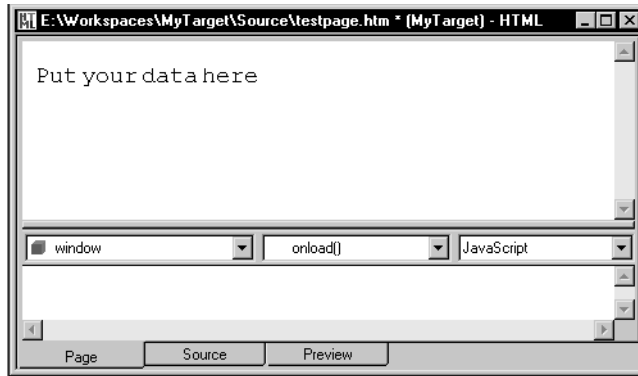


Table 1-2: HTML editor views

Editor view	Description
Page view	A fully formatted, editable view of your content
Source view	An editable view of the tags and content
Preview view	A non-editable view that lets you test how the document should appear

- **Script editor** The script editor supports writing both client- and server-side scripts. The editor is available as an integrated part of the HTML editor or as a standalone tool.

Table 1-3: Script editor features

Editor type	Features
Integrated Script editor	Appears as a pane at the bottom of the HTML editor. You can select an object or event from your current page for scripting. Drag-and-drop programming and InstaCode help you choose the objects and properties to include in your code.
Standalone Web Script editor	Appears as a separate window. You can create standalone scripts in it, and then access those scripts from a number of pages.

- **Cascading Style Sheet editor** You can define styles for Web pages by creating style definitions in the Cascading Style Sheet editor. Quick access to styles through a tabbed dialog box lets you create embedded and inline styles as well as separate style sheets.

- **Frameset editor** The Frameset editor helps you to edit and work with framesets in a document. A Frameset wizard helps with the initial creation of frameset documents.
- **Wizards** PowerBuilder wizards guide you through setup tasks, such as creating workspaces, targets, JSP pages, 4GL JSP pages, DataWindow elements, script files, EA Server components, and many others. You must add a JSP target to your workspace before you can use the page or script creation wizards.
- **4GL JSP pages** When you develop 4GL JSP pages, you can easily create page parameters and also variables that you can bind to controls. On 4GL JSP pages you can also select and code server-side events from the integrated Script editor.
- **Design-time controls (DTCs)** Design-time controls create basic HTML and script from information you provide in property sheets that display when you drop a DTC on a JSP page in the HTML editor.

The Web DataWindow DTC provides an easy way to access a database from a JSP page. It displays dynamic database content in a variety of presentation styles and supports inserts, updates, and deletes against the database.

- **To-Do List** The To-Do List tracks your progress in completing tasks for your targets. The To-Do List for a JSP target works the same way as it does for PowerScript targets in PowerBuilder.
- **Deployment controllers** The deployment controllers manage server-specific coding and configuration. When you create content and scripts using the Web Target object model, you can create one version of your source files rather than one for each server destination. The JSP target deployment controller automatically modifies the scripts for compatibility with the servers that you select for your Web site deployment.
- **Link management** A JSP target displays information in the Output window about broken links from one file to another whenever you build a target. This gives you the opportunity to fix links before you deploy the target.

About the editors

PowerBuilder includes several editors for preparing JSP pages.

Table 1-4: JSP target editors

Editor	Description
HTML editor	Provides views that let you: <ul style="list-style-type: none"> • Edit in a WYSIWYG word-processing window (Page view) • Edit in a color-coded window of HTML tags and content (Source view) • Preview the page as it would appear in a browser (Preview view)
Script editor	Either standalone or integrated with an HTML or JSP page. The Script editors let you: <ul style="list-style-type: none"> • Write scripts for objects and events in the page • Save scripts in external files that you link to the page
Style Sheet editor	Either global, embedded, or inline (available from property sheets of elements and controls on your page). Style Sheet editors let you: <ul style="list-style-type: none"> • Create external style sheets, embedded styles, and inline styles • Edit styles in tabbed pages that hide style sheet syntax • View the output of editing from a Source tab
Frameset editor	Lets you define frames graphically and specify the HTML or JSP pages to display in the frames.

Tools for editing

Several tools help you develop content in the editors.

Toolbars

There are several toolbars that include buttons for:

- Saving, undo/redo, using the clipboard, searching, and setting properties



- Inserting HTML elements such as lines, tables, images, hyperlinks, anchors, form controls, tables, DataWindows, and components



- Applying character formatting (font, style, and size changes), bold, italic, underlining, and various alignment changes



You can manage and customize the toolbars using the Toolbars dialog box that you access from the Tools>Toolbars menu. In the dialog box, you can turn on PowerTips and, by clicking the Customize button, examine toolbar icons and their commands in the Customize dialog box.

System Tree

The System Tree is an active resource. It provides a quick reference list of programming information.

The System Tree has four tabs and lists the HTML tags, language syntax, and object models that you use in the HTML and Script editors. There is information for the major browsers and scripting languages. You can view the client-side object models for Microsoft and Netscape browsers, and syntax information for VBScript and JavaScript.

The System Tree window displays by default when you start PowerBuilder for the first time. You can hide or display the System Tree using the System Tree button on the PowerBar or by selecting Window>System Tree.

For more information on using the System Tree with JSP targets, see Chapter 2, “Developing JSP Applications.”

Wizards for HTML elements

PowerBuilder provides several tools to help you create the more complex HTML elements: frame, table, and form. After you create the element, you can edit the tags in Source view. For tables and forms, you can add content to the element in Page view.

Frames

The Frameset wizard is available from the New dialog box. You can graphically lay out the frames and specify an HTML or JSP document for each frame. After you create the frameset, you can edit the No Frames section in the editor.

If you want to change the frameset specifications, you can make the changes in Source view or in the Frameset Properties dialog box available from the Frames view pop-up menu.

Tables

The Table wizard is available on the Table menu in either the Page view or Source view of the HTML editor. You can specify the columns and rows, and the alignment and color attributes. You can also add content to the cells.

After you leave the Table wizard, you can edit the table content in Page view or Source view. In Page view, you can also use the Table menu to manipulate (insert, delete, merge, split) the table's rows, columns, and cells. In Source view, you can directly edit the TR and TD elements for the table.

Other JSP-related wizards

Information on related JSP target wizards is available elsewhere in this manual or in the *DataWindow Programmer's Guide*.

Table 1-5: Location of information about JSP target wizards

Wizard type	Where to find information
Target wizards	Chapter 2, "Developing JSP Applications"
Web page wizards	Chapter 3, "Working with the HTML Editor"
Style Sheet wizard	Chapter 4, "Working with Style Sheets and Framesets"
Deployment Configuration wizard	Chapter 12, "Building and Deploying JSP Targets"
JavaScript Caching wizard	<i>DataWindow Programmer's Guide</i>

Design-time controls

PowerBuilder provides a design-time control, the Web DataWindow DTC, that lets you use DataWindow objects you have created in PowerBuilder or InfoMaker to specify data you want to display. When you insert a Web DataWindow DTC, PowerBuilder uses the DataWindow object definition to generate HTML and server-side scripting logic for the page.

For more information on the Web DataWindow DTC, see Chapter 11, "Using the Web DataWindow Design-Time Control."

System options and editor preferences

To make changes to PowerBuilder system options, select Tools>System Options from the PowerBuilder menu.

The System Options dialog box has six tabs that apply to JSP targets in PowerBuilder, including a tab that allows you to add search paths for custom tag libraries.

The Java tab page of the System Options dialog box allows you to include search paths—in addition to the paths defined in the system CLASSPATH variable—for applets and JavaBeans. For information on specific fields in the System Options dialog box, see the online Help and the PowerBuilder *User's Guide*.

Before you start developing Web content, you can set preferences for the JSP Target Script editors. For information about configuring the Script editors, see “Formatting HTML source display” on page 47 and “Setting default formats for scripts in the Script editor” on page 130.

Working in an integrated Web delivery environment

In Web delivery environments, application and transaction servers play a vital role in delivering dynamic content to Web site users by extending the capabilities of a Web server and integrating database management systems (DBMS) into the delivery strategy.

An application server processes server scripts to produce customized pages, whereas a transaction server manages components that encapsulate business logic and manage database connections. An application server can integrate with, but does not require, a transaction server.

Server types

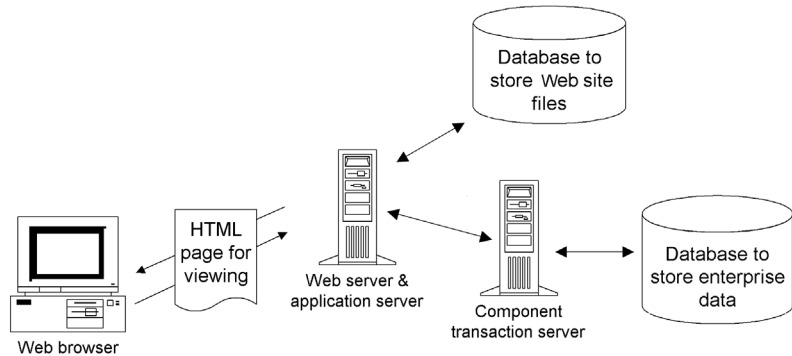
Web delivery environments use several types of application and transaction servers.

Table 1-6: Server types in Web delivery environments

This type of server	Performs these actions
Application server as a Web server; personal Web server	Manages requests for Web pages
Application server as a dynamic page server	Processes server-side scripts
Component transaction server	Provides access to components that process business logic
Database management system	Provides database access

These servers can run on one machine, or run on a number of machines for load balancing. The following illustration shows an environment where the Web server and application server run on the same system, and the component transaction server on another system. In this example, the application server uses its own database to store all of the files included in the Web site:

Figure 1-1: Web server environment example



Web servers and application servers

You can use any commercially available Web server that can communicate with the application servers where you deploy a JSP target.

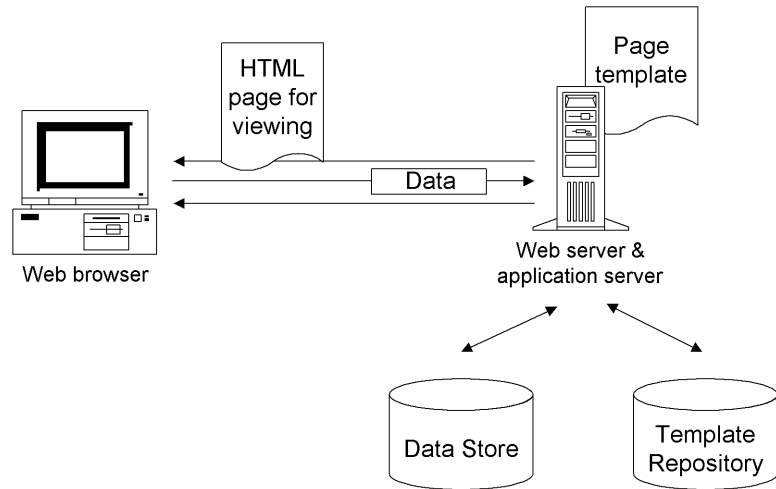
Table 1-7: Application servers and Web servers

Application server	Web server
Tomcat	Apache Tomcat or any Web server that can communicate with the Tomcat application server
EAServer	EAServer or any Web server that can communicate with the EAServer application server
Other JSP 1.2 servers (You can use third-party command line tools to deploy a JSP target to other JSP servers.)	Any Web server that can communicate with the application server where you deploy the JSP target

These application servers create dynamic pages on the fly by processing server-side scripts. The scripts are part of a template (source) page. A template page can contain HTML and client scripts as well as server scripts.

The following figure shows how an application server integrates into a Web delivery environment (without a transaction server):

Figure 1-2: Web delivery environment without transaction server



Transaction server

Transaction servers are used in multitier applications to host executable components. They make it possible to shift processing to the middle tier, enabling application clients (such as Web pages) to be thin. They also handle database connections, thereby distributing the processing load and making it easy to manage connections through connection caching and pooling.

The EAServer component transaction server can host various kinds of components, including Java classes, JavaBeans, Enterprise JavaBeans (EJBs), servlets, JSPs, and PowerBuilder objects. JSP targets provide ready access to a server and its components, including Web DataWindow server components.

DBMS

A key feature of dynamic Web pages is the ability to retrieve and update database information. A JSP target's support for application server technologies makes it easy to incorporate dynamic database content into Web pages.

Using the Web DataWindow

Adding a Web DataWindow to a Web page facilitates retrieving and updating database information. See the *DataWindow Programmer's Guide*.

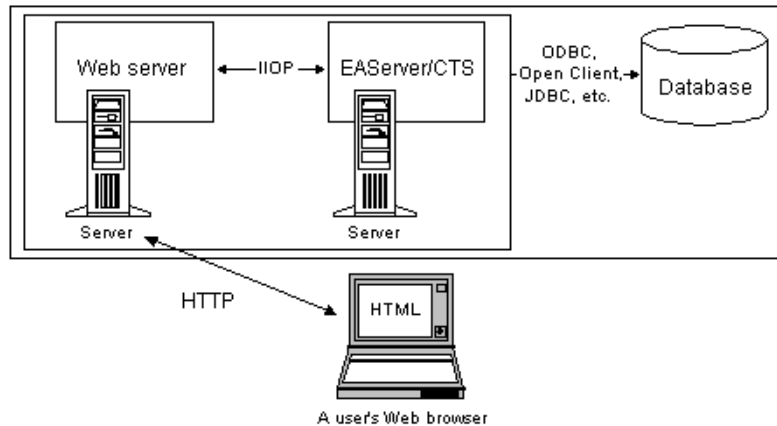
EAServer environment

EAServer provides the following services:

- HTTP server
- Component transaction server (CTS)
- JSP server

The basic architecture and communications protocols used by the transaction server and the page and personal servers are shown below:

Figure 1-3: Web delivery environment with transaction server



Can be a single server

The Web server and the component transaction server can be on the same server machine.

4GL JSP pages provide enhanced integration with EAServer environments. They make it easy for you to access EAServer components, bind properties of those components to controls on your page, manage page data, and simplify server scripting tasks.

JSP and EAServer only

You cannot use 4GL JSP pages if you use a transaction server other than EAServer.

For how to work with 4GL JSP pages, see Chapter 9, “Developing 4GL JSP Pages.”

About this chapter

This chapter describes how to create and work with JSP targets to develop Web applications.

Contents

Topic	Page
Setting up JSP targets	19
Working with files in a JSP target	26
Creating Web pages	34

Setting up JSP targets

A JSP target provides the physical and management structure for the folders and files within it. When you work with a JSP target, you do so within the context of a workspace. You must set up the JSP target before you can begin developing content. After you set up a JSP target, you can add new content or content based on existing files that you import.

To produce most JSP targets, you must complete the following tasks :

- Create a JSP target
- Configure the JSP target to meet your JSP environment delivery requirements by modifying JSP target properties
- Define connection profiles and the folder structure for your files
- Import existing files you want to use
- Create Web pages with the JSP page wizards and the HTML editor
- Add HTML elements and controls, including design-time controls, to your Web pages using drag-and-drop programming
- Build and deploy your target
- Test your pages to make sure that they appear and work as planned
- Deploy your Web application to a production environment where client browsers can access your Web site

JSP Target wizard

You create a JSP target using the JSP Target wizard. The JSP Target wizard creates a target with Source and Build folders and a deployment configuration. JSP pages are deployed as a Web application in a Web Archive (WAR) file.

The JSP Target wizard includes a server-specific deployment page with properties that depend on the selection you make in the JSP server page of the wizard. Any properties you specify in the wizard, except for the deployment configuration name, can be modified after target creation in the Deployment Configuration Properties dialog box for the JSP target.

Specifying a server type

When you select a server type, the wizard presents a page where you specify how to connect to the server.

Table 2-1: Server-specific selections in the JSP Target wizard

JSP server	Server-specific selections
EAServer	The EAServer profile and the HTTP port you want to use
Tomcat	The deployment folder (typically, the webapps folder under the Tomcat installation directory), the HTTP server and port name, and a login name and password
Command Line	Command lines you can use to deploy your Web application to any JSP 1.2 compatible server

Custom command-line deployment

Custom deployment configuration properties

You can use the custom command-line deployment configuration to set commands for Java command-line build tools such as Apache Ant, or to deploy a target to JSP 1.2 servers other than EAServer and Tomcat, such as IBM WebSphere or BEA WebLogic. The configuration screen for custom command-line deployment includes the following fields and check boxes:

Table 2-2: Custom command-line deployment configuration properties

Property	Field type	Use this field to
Description	Text box	Set a description for your deployment configuration.
Command	Column	Add one or more command lines in a defined sequence of execution. You can move command lines up or down in the command sequence by using the arrow controls. You can enter macros.

Property	Field type	Use this field to
Start-up directory	Column	(Optional) Change the current directory to the location you want before executing the command you entered for the same row in the Command column. This is equivalent to a <code>cd</code> command to change directories.
Show messages in output window	Check box	Display messages from the command-line tool in the PowerBuilder Output window (selected by default).
Automatically generate WAR file	Check box	Generate the target WAR file when you deploy the target (selected by default). You should clear this check box if you generate the WAR file from a command you enter in the Command column.
Abort execution on error	Check box	Halt the execution of command lines once an error is detected (selected by default).
Command time-out	Text box	(Optional) Enter a time-out value in seconds that applies to all the command lines you enter.

Command-line macros for custom deployment

You can use macros on any of the command lines you enter in the list of commands for your custom deployment configuration. There are five macros available for use on the command lines you enter in the JSP Target wizard (or in the Deployment Configuration wizard):

Table 2-3: Command-line macros

Macro text	Pasted text	Value added by macro
Configuration Directory	\$(ConfigDir)	Location of the local copy directory that you specify in a subsequent page of the wizard.
Display Name	\$(DisplayName)	Display name from <i>web.xml</i> for the target. You also specify this in the wizard.
WAR Filename	\$(WARFile)	File name, but not the full path. If you do not use a command line to build the WAR file, the value is specified in a subsequent page of the wizard.
Build Directory	\$(BuildDir)	Full path to the build directory for the target that you specified in the JSP Target wizard.
Source Directory	\$(SourceDir)	Full path to the source directory for the target that you specified in the JSP Target wizard.

Creating a JSP target

Creating a target is the first step in producing most JSP targets.

❖ **To create a JSP target:**

- 1 From an open workspace, select File>New
or
In the Workspace tab of the System Tree, right-click the workspace name, and select New from the pop-up menu.
- 2 On the Target page of the New dialog box, click the JSP Target wizard.

The JSP Target wizard starts. You use the JSP Target wizard to create a JSP Web site that you deploy to a JSP 1.2 or later component server such as Tomcat or EAServer.

What the Source Controlled JSP Target wizard does

The Source Controlled JSP Target wizard opens a JSP target that has been checked in to a source control system

- 3 Follow the instructions on the wizard pages.

Target Properties dialog box

After you create a JSP target, you can modify the target properties and add deployment configurations from the Target Properties dialog box. You access the Target Properties dialog box from the pop-up menu for a JSP target in the System Tree.

The Target Properties dialog box for a JSP target has the following options for property selections:

Table 2-4: Target properties dialog box for a JSP site target

On this page	Set these options
Options	The path of the JSP target's Source folder and Build folder.
Deploy	The local and remote deployment configurations for your target. You can also set the deployment priority of the various deployment configurations, create new deployment configurations, make changes to existing configurations, and remove deployment configurations.
Run	The start page for the target and the deployment configuration you want to use for running when you click the Run button from the PowerBar or select Run>Run Target from the PowerBuilder menu.

Adding deployment configurations

After you create a target, you can add deployment configurations and change settings for the target from the property pages for the target. For information about deploying a JSP target, see Chapter 12, “Building and Deploying JSP Targets.”

Local and target deployment configurations

Before you add a deployment configuration, you should decide whether you want to create a test (local) configuration, a shared or production (target) configuration, or both. You can deploy using both types of configuration at the same time.

Local configurations are stored in your registry, whereas target configurations are stored in the PBT file. PBT files with target configurations can be shared in a source control system, but users have to make certain that any configuration paths, database profiles, and target mappings have identical names on all machines that use the source-controlled target configurations.

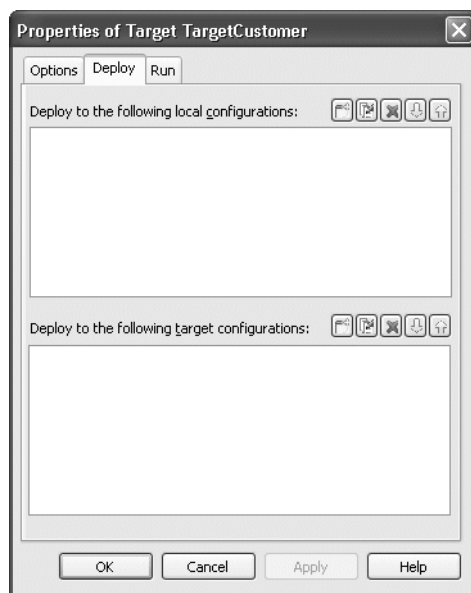
Multiple deployment configurations

You can add multiple deployment configurations to the list of local or target configuration profiles. You can change the order of the configurations in the list by using the wavy arrow keys above the configuration list boxes. A check box next to each deployment configuration in the list lets you select which of the configurations you want to use the next time you deploy your JSP target. When you add a configuration, it is selected for deployment by default.

❖ **To add deployment configurations:**

- 1 Right-click the JSP target on the Workspace tab of the System Tree and select Properties from the pop-up menu.
- 2 Click the Deploy tab.

The Deploy page is one of three pages in the Properties dialog box for a JSP target.



- 3 Click the Create New Configuration button for either a local deployment configuration or a target deployment configuration.

The Name New Deployment Configuration wizard starts.

- 4 Follow the instructions in the Name New Deployment Configuration wizard. In the wizard, you provide the following information:
 - A name for the deployment configuration
 - Whether you are using the default object model or no object model
 - The JSP server to deploy to
 - If your JSP server is EAServer, the EAServer profile
 - If your JSP server is EAServer, the HTTP server name and port
 - How to handle failures (deploy all or nothing or only successful files)
 - A folder name for a local copy of the deployed files
 - The WAR file name

The JSP server can also be Tomcat or Command Line.

After you click Finish on the last wizard page, the new deployment configuration displays in one of the list boxes on the Deploy page of the Target Properties dialog box. You can use the toolbar above the list box to edit, delete, or change the order of the configurations in the list.

- 5 Click the Run tab and select a start page for your JSP target.

If you want to start your Web application from the design-time environment, you do not need the complete URL, but you can give the complete URL, including the server name, if you want to.

Choosing a URL

You can use the Choose URL dialog box (URL picker) that you access from the ellipsis button next to the Start Page field. When you use Choose URL, only the file name portion of the URL that you select is added to the field.

- 6 Select a deployment configuration that you want to use for running.

The deployment configurations you created for the current JSP target are available for selection from the Deploy Configuration For Running drop-down list.

Defining connection profiles

If you plan to use database connections or EAServer connections, you must define connection profiles for these types of connections:

Profile type	Create for
Database connections	Database connections that your pages use directly, and connections to be used by Web DataWindow objects
Connections to EAServer	Components your application will access that are stored on the server

You set up database connection profiles from the Tools>Database Profile menu item. You can set up EAServer profiles either from the Tools>EAServer Profile menu item or by right-clicking anywhere on the Components tab of the System Tree.

For information on setting up these profiles, see *Connecting to Your Database*.

Importing files from an existing Web site

You can import folders or multiple files into your JSP target. If you want to use an existing Web site as the starting point for development, you can import the site into a JSP target.

The site you import must be a file-based Web site. The folder structure of the Web site becomes the folder structure in the JSP target. The JSP target creates a map of the resources used by the site and tracks the content, links, and components.

Once you have imported files or a complete Web site, you can modify the content and organizational structure to suit your Web application.

- ❖ **To import files or an existing Web site into a JSP target:**
 - 1 Right-click the JSP target on the Workspace tab of the System Tree.
 - 2 Select Import Files or Import Folder from the pop-up menu.
 - 3 Select the files or the folder you want to import and click OK or Open.

Working with files in a JSP target

When building a Web application that uses EAServer components and DataWindows, you can use the JSP target environment to develop these separate components in the same workspace. A JSP target lists and tracks all of the files and folders in your Web site. The target identifies the root directory used to store source files and specifies build and deploy options.

Types of files

A JSP target can contain various types of files:

- HTML files
- JSP files
- Scripts
- Component files
- Accessory files, such as images and video files

HTML files	HTML files determine the presentation for your Web applications. You can use HTML pages in JSP targets, but you must import them as accessory files rather than create them directly in the JSP target.
JSP files	You can use JSPs in many ways in Web-based applications that you deploy to a JSP server. JSPs are invoked by a Web server in the middle tier in response to HTTP requests from Web clients. As part of the J2EE application model, JSPs can invoke, in turn, the business methods of Enterprise JavaBeans (EJB) components on a transaction server.
Scripts	<p>Scripts drive application behavior on both the client side and the server side. You can write scripts in the Script editor in a number of languages including JavaScript, JScript, and VBScript, as well as server scripts in Java.</p> <p>Client-side scripting Client-side scripts contain instructions that the browser executes on the user's local machine. Client-side scripts can use syntax, functions, and objects supported by the major browsers.</p> <p>Server-side scripting Server-side scripts contain instructions that an application server or Web server executes before sending a Web page to a client browser. These scripts provide a way to include conditional execution, looping, and other programming structures in your Web pages. They can also provide access to integrated server systems such as a DBMS or EAServer.</p> <p>Server-side scripts can take advantage of the Web Target object model, which uses a set of language structures and objects. Although many of the objects and programming structures are common to a number of application server technologies, a subset is specific to the EAServer environment.</p>
Components	<p>You can include the following types of components in your Web pages to deliver the content and functionality your site users need:</p> <ul style="list-style-type: none">• Web DataWindow DTCs• EAServer components• Java applet and JavaBean components• Microsoft ActiveX controls• Netscape plug-ins• Custom tag libraries and their supporting classes
Accessory files	Web sites include several other types of files, such as images, video files, and audio files. You can import accessory files into a JSP target from another location. See "Importing files from an existing Web site" on page 26.

Adding content

After you create a JSP target, you are ready to begin developing content. You should probably start by setting up the folder structure for your target.

Adding new folders

When you add content to a JSP target, you use folders to set up a logical directory structure for the content. When you deploy a JSP target, the deployment engine replicates the folder structure on the server system. It also processes the content for the target Web server and application server, and rewrites link information to fit the directory structure.

❖ **To add a new folder to a JSP target:**

- 1 On the Workspace tab of the System Tree, right-click a JSP target (or a folder under a JSP target), then select New Folder from the pop-up menu.
- 2 Right-click the new folder that displays under your JSP target and select Rename from the pop-up menu.
- 3 Type in the name you want to give to your folder.

Adding new JSP files

With a folder structure in place, you can begin adding files to those folders. The JSP page of the New dialog box has wizards for creating the various types of files you can edit in a JSP target.

Table 2-5: Web page wizards

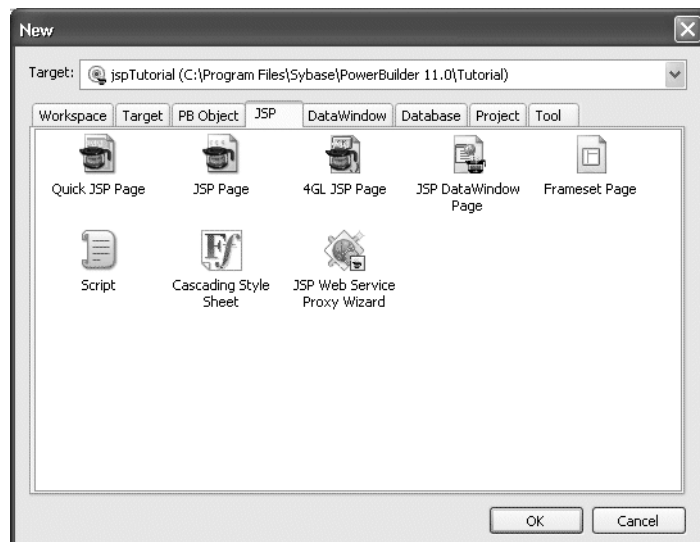
Select this	To do this
Quick JSP Page	Open a new, unnamed JSP page in the HTML editor
JSP Page	Create a new JSP page, specify design settings such as background color for the page, and open the page in the HTML editor
4GL JSP Page	Create a new 4GL JSP page, specify design settings such as the background for the page, select error reporting settings, page parameters, and EAServer components, and open the page in the HTML editor
JSP DataWindow Page	Create a new JSP page that contains a Web DataWindow design-time control (DTC) and specify design settings such as the background for the page and configuration information for the DTC, including the EAServer profile
Frameset Page	Create a new frameset page for your Web pages
Script	Open a new, unnamed script file in the standalone Script editor

Select this	To do this
Cascading Style Sheet	Open a new external style sheet file (and optionally reference an existing style sheet) in the Style Sheet editor
JSP Web Service Proxy	Create a proxy object to use Web services

❖ **To add new JSP files to a JSP target:**

- Right-click a JSP target, or a folder under a JSP target, then click New from the pop-up menu.

The JSP page of the New dialog box displays:



Selecting an item in the New dialog box launches a wizard that helps you set up a new JSP file.

Using the System Tree

The System Tree is an active resource for page development, providing an organized framework for developing your targets, pages, and components. Its four tab pages let you browse items available to your Web pages.

You can view the properties for any item in the System Tree by right-clicking the item and selecting Properties from the pop-up menu. On the Language, Components, and Page tab pages, you can view, but not change, properties.

Table 2-6: System Tree tab pages

System Tree tab page	Displays this content
Workspace	Workspace contents—including targets, folders, files, and libraries
Page	The page components of the active page open in the HTML editor
Language	HTML tags, client- and server-side object models (listing object properties, methods, and events), and client-side scripting language elements (including JavaScript and VBScript syntax elements)
Components	ActiveX controls, plug-ins, Java applets, JavaBeans, EAServer components, and custom tag libraries available to your target

You can drag and drop HTML elements, scripting resources, and other components from a System Tree tab page onto Web pages open in the HTML editor.

Workspace tab

The Workspace tab lets you manage your JSP targets. From it you change target properties, add, remove, and rename files and folders, migrate JSP targets created in earlier versions of PowerBuilder, and build, deploy, and run JSP targets. The pop-up menu for a JSP target gives you access to these features:

Table 2-7: Workspace tab pop-up menu selections for JSP targets

Pop-up menu item	Feature
New	Displays PowerBuilder New dialog box
New Folder	Creates a new folder in the target
Import Files	Allows you to select files to import to the target
Import Folder	Allows you to select a folder to import to the target
Incremental Build	Builds only files in the JSP target that have changed since the previous build
Full Build	Builds all files in the JSP target
Migrate JSP Target	Migrates a JSP target created in PowerBuilder 10 or 10.5 to PowerBuilder 11.
Deploy	Deploys the target according to the deployment configurations selected in the Target Properties dialog box
Deploy To EP	Deploys a JSP target to Sybase Enterprise Portal (EP) rather than a JSP server
Run	Starts a browser that opens to the page you specify in the Start Page field in the Run tab page of the Target Properties dialog box
Remove Target	Deletes the target from the workspace
Show	Toggles display of items in the System Tree
Properties	Opens the Target Properties dialog box

Page tab	<p>The Page tab shows the hierarchy of objects on the current page in the HTML editor. The page tab also lists properties, methods, and events for:</p> <ul style="list-style-type: none">• Predefined Microsoft Internet Explorer page objects• Predefined HTML objects• Predefined JavaScript objects• User-defined controls• Server-side EAServer components placed on a 4GL JSP page <p>You can create script to refer to the object or its methods, properties, and events by dragging it from the Page tab to the Source view of the HTML editor, or to the integrated Script editor.</p>
Language tab	<p>The language tab provides quick access to:</p> <ul style="list-style-type: none">• HTML elements and attributes The most commonly used HTML elements appear in this list. The elements are organized alphabetically as well as by category to make browsing easy. Elements appear in categories such as format, headers, image, and multimedia. Attributes appear alphabetically under elements. You can drag an HTML element or attribute from the Language tab to the Page view or the Source view in the HTML editor, or to the Script editor.• Script language syntax The syntax elements for JavaScript and VBScript. You can drag a script syntax element from the Language tab to the Source view in the HTML editor, or to the Script editor.• Object models The objects, and their methods, properties, and events, for the Microsoft and Netscape client-side object models. Methods, properties, and events are also listed for the Web Target and Active Server Pages server-side object models, as well as for JSP implicit objects. You can drag an object model from the Language tab to the Source view in the HTML editor, or to the Script editor.
Components tab	<p>The Components tab of the System Tree lists client-side and server-side ActiveX controls, plug-ins, Java applets, and JavaBeans installed on your system, and EAServer components accessible from your system.</p>

Information about installed components

A JSP target gets information about installed components from the Windows registry, browser plug-in directories, MIME extensions recognized by your primary browser, and user-specific folder lists. It relies on the Java classpath, if set on your system, to find Java applets and JavaBeans.

Custom tag libraries for JSP targets must be listed (or contained in directories that are listed) on the JSP page of the System Options dialog box. You must then make sure the classes for the custom tag libraries are available to your JSP target.

The list of servers on the Components tab is populated from the EAServer profiles configured for PowerBuilder. A profile must be configured for a server for it to appear on the Components tab. The server must be running for you to see the components available in its repository.

To give a page access to a component listed on the Components tab, drag the component from the Components tab to the Page view or Source view in the HTML editor.

Drag and drop

You can drag files, HTML tags, objects, and methods from the System Tree into the HTML or Script editors, or you can copy from the System Tree and paste into an editor window. The result depends on the item you drag or copy, on the System Tree, and on the editor view:

Table 2-8: Drag-and-drop elements from the System Tree

From	To	Item	Result
Workspace tab	Page view or Source view	Text file (such as HTM, JS, JSP)	Creates a hyperlink to a file from the current page. If the HTML editor is not already open, opens the item dragged or copied in the appropriate editor.
	Page view or Source view	Image file (such as GIF, JPG, and so on)	Creates image element with link to image file in its SRC attribute.
Page tab	Source view or integrated Script editor	Object method, event, or property available on current page	Inserts the appropriate dot notation to fully qualify the object method, event, or property name.

From	To	Item	Result
Language tab	Page view or Source view	HTML element (tag) that does not have an associated JSP Target property sheet	If no text is selected in the HTML editor, the element is inserted, but you must be in, or switch to, Source view to put the cursor between the start and end tags and add content. In Page view, the editor formats selected text according to the element you dragged.
	Page view or Source view	HTML element (tag) that has a JSP Target property sheet	JSP Target property sheet displays. You can fill in property sheet fields before the element (control) is added to the open page in the editor.
	Page view, Source view, or Script editor	Object model method, event, or property	Inserts the appropriate dot notation with text in brackets that you must replace to fully qualify the method, event, or property.
Language tab	Page view, Source view, or Script editor	Scripting language function, keyword, operator, or escape sequence	Inserts the appropriate dot notation to fully qualify the scripting language syntax.
Components tab	4GL page in Source view	EAServer component	Opens the Page properties dialog box to the EAServer page on which the control is associated with the page.
	Script editors	EAServer component	Inserts the appropriate dot notation to fully qualify the component.
	Page view or Source view	Custom Tag Library (JSP target), Plug-in, or ActiveX	Opens the JSP target property sheet associated with object. (To add a custom tag library class, you must first add the tag library to the page.)
	Script editors	Applet or JavaBean method, Plug-in, or ActiveX	Inserts the object name or identifier.

Copying items You can also copy items from the System Tree and paste them to the Page view or Source view.

❖ **To copy items from the System Tree:**

- 1 Right-click the item you want to copy and select Copy from the pop-up menu.
- 2 Right-click in the Page view or Source view, and select Paste from the pop-up menu.

Migrating JSP targets If you open an existing workspace created in PowerBuilder 9 or 10 using File>Open Workspace or File>Recent Workspace, the new workspace could contain JSP targets created in PowerBuilder 9 or 10. To migrate a JSP target to PowerBuilder 11.0, right-click the JSP target, select Migrate JSP Target from the pop-up menu, and click OK.

After the JSP target migration is complete, the JSPs in the target use the HTMLGenerator110 component instead of the HTMLGenerator100 or the HTMLGenerator105 component, class IDs for design-time controls are changed to PowerBuilder 11.0 class IDs, the original JSP files are saved to files with UTF-8 encoding, and the original page directive character set is changed to UTF-8 for each JSP in the target.

Creating Web pages

The remainder of this book provides detailed information about how you develop pages within a JSP target, and how you use the development environment to produce Web applications. This section gives an overview of the types of tasks you need to complete to develop a page, and gives references to sections in this book that describe how to complete these tasks.

Adding content and style to your Web pages

You use the tools available in a JSP target to add content and style to your Web application.

Table 2-9: Where to find information about page content and style

For information about this topic	See this chapter
Opening a page in the HTML editor and adding text, images, and other page elements	Chapter 3, “Working with the HTML Editor”
Using absolute positioning on a page	Chapter 3, “Working with the HTML Editor”
Setting up page formatting using style sheets	Chapter 4, “Working with Style Sheets and Framesets”
Developing dynamic Web pages	Chapter 8, “JSP Web Services” Chapter 7, “Working with Application Servers and Transaction Servers”
Developing dynamic Web pages for deployment to EAServer	Chapter 9, “Developing 4GL JSP Pages”
Writing client and server scripts	Chapter 6, “Writing Scripts” Chapter 7, “Working with Application Servers and Transaction Servers” Chapter 9, “Developing 4GL JSP Pages”
Adding database forms for retrieval and update using the Web DataWindow design-time control	Chapter 11, “Using the Web DataWindow Design-Time Control”
Adding components such as Java applets, JavaBeans, and EAServer components to a page	“Using the System Tree” on page 29
Adding custom tags and custom tag libraries to a JSP	“Custom tags” on page 79

Building and deploying JSP targets

When you build a JSP target or JSP target files, the target or the files are copied from the target Source directory to the target Build directory. Building a target before you deploy it can be useful to verify links and make sure they work.

You can build JSP target files in a separate action, but when you deploy a JSP target or JSP target files, the target files are built automatically before being deployed. You can find more information in Chapter 12, “Building and Deploying JSP Targets.”

You can make sure that your Web pages appear and function as planned by inspecting the pages during development and then after deployment. Web servers are important for testing and deployment.

You can develop Web applications that are independent of the application server used for production deployment. If your server-side scripting uses the Web Target object model, then mapping information supplied during deployment translates between the Web Target object model and the object model for the application server you choose.

Table 2-10: Where to find information about testing JSP targets

For information about this topic	See this chapter
Viewing the appearance of your page in a browser or in the HTML editor's Preview view	Chapter 3, "Working with the HTML Editor"
Using customized troubleshooting tools for 4GL JSP pages	Chapter 9, "Developing 4GL JSP Pages"
Viewing the deployed pages in the browsers you want your application to support	Chapter 12, "Building and Deploying JSP Targets"

Working with the HTML Editor

About this chapter

The HTML editor can be used to edit HTML pages and JSP pages in JSP targets. This chapter introduces you to the HTML editor.

For information on the Style Sheet and Frameset editors, see Chapter 4, “Working with Style Sheets and Framesets.”

For information on the Script editor, see Chapter 6, “Writing Scripts.”

Contents

Topic	Page
HTML editor views	37
Opening the HTML editor and setting options	43
Basic editing in Page and Source views	49
Correspondences of common elements	57
Absolute positioning	67
JSP page authoring	71

HTML editor views

The HTML editor has three views: Page, Source, and Preview. Each view provides a different way of working with your HTML project.

Page view

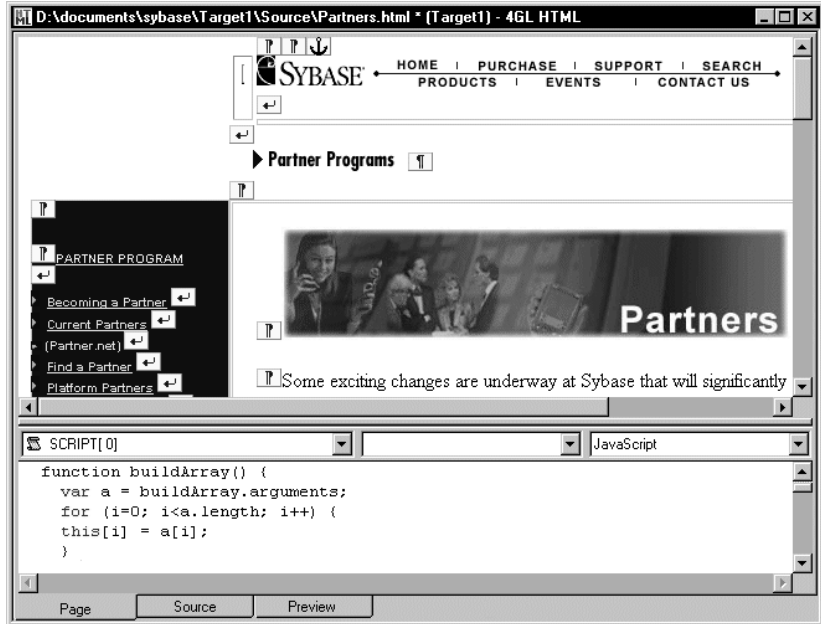
Page view provides WYSIWYG editing for an HTML page without requiring knowledge of HTML tagging. Use Page view as your main editing environment or to supplement the editing you do in Source view.

Hiding page view

You can hide page view by selecting Design>Options and clearing the Show Page View check box. To see the change, close the HTML editor and then open it again.

Modifying the WYSIWYG view

Although Page view provides WYSIWYG editing, you can show HTML tags and JSP directives on your page through a toggle switch in the PowerBuilder Design menu or in the Page view pop-up menu. The tags display in symbol form inside yellow blocks. This image shows a page in Page view with the Show Non-Visual Tags menu item selected.



Part of a client-side script is displayed in the integrated Script editor at the bottom of Page view. For more information about the Script editor, see Chapter 6, "Writing Scripts."

The same page looks like this when the Show Non-Visual Tags item is not selected:



Forms on non-4GL pages and script elements

In Page view, you can also see FORM elements around controls that you insert on a non-4GL page. You can see icons for client and server scripts that you add to the page, and on JSP pages, you can see icons for JSP directives and custom tag library classes that you add to a page. These elements and icons remain visible in Page view even after you clear the Show Non-Visual Tags item, but they are not visible in the Preview view.

Basic document structure

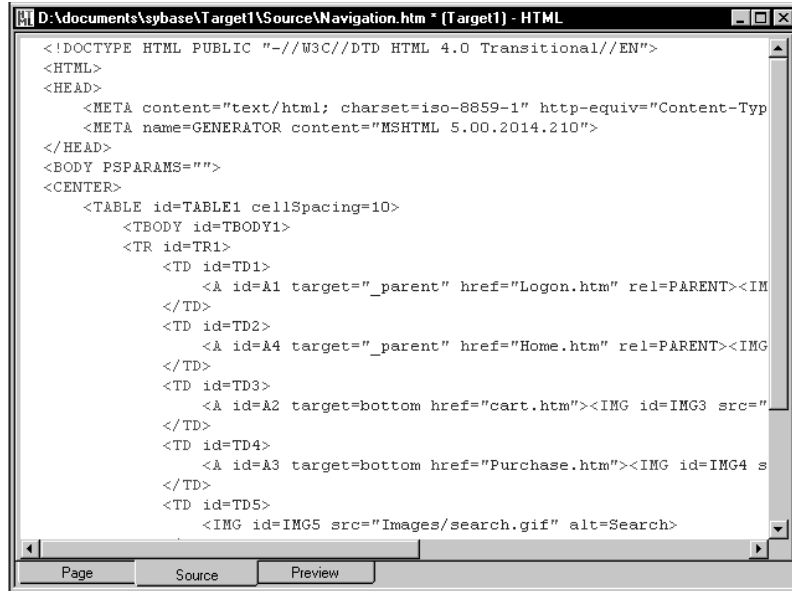
The basic document structure (HTML, HEAD, and BODY elements) is supplied when the page is first created. The editor creates the structure for you. Any text you type in Page view is inserted in the Body section of the HTML or JSP document.

The HEAD element includes a document title and can include links to external style sheets.

For more information on adding a page title from Page view, see “Giving your page a title” on page 46. For information on style sheets, see Chapter 4, “Working with Style Sheets and Framesets.” For information on other basic editing techniques, see “Basic editing in Page and Source views” on page 49.

Source view

Source view gives you total control over the HTML tags and content of your file, including the Head section and scripts. You can view and edit content as well as the HTML tags and their attributes.



```

D:\documents\sybase\Target1\Source\Navigation.htm * (Target1) - HTML
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <META content="text/html; charset=iso-8859-1" http-equiv="Content-Typ
  <META name=GENERATOR content="MSHTML 5.00.2014.210">
</HEAD>
<BODY PPARAMS="">
<CENTER>
  <TABLE id=TABLE1 cellSpacing=10>
    <TBODY id=TBODY1>
      <TR id=TR1>
        <TD id=TD1>
          <A id=A1 target="_parent" href="Logon.htm" rel=PARENT><IM
        </TD>
        <TD id=TD2>
          <A id=A4 target="_parent" href="Home.htm" rel=PARENT><IMG
        </TD>
        <TD id=TD3>
          <A id=A2 target=bottom href="cart.htm"><IMG id=IMG3 src="
        </TD>
        <TD id=TD4>
          <A id=A3 target=bottom href="Purchase.htm"><IMG id=IMG4 s
        </TD>
        <TD id=TD5>
          <IMG id=IMG5 src="Images/search.gif" alt=Search>
  </TABLE>

```

Basic document structure

When you create a new HTML file and switch to Source view, the basic document structure has already been created for you, including the HTML, HEAD, and BODY elements.

Using menu items and tools

In Source view you can type the tags manually, use menu items or tools, such as wizards, toolbar buttons, element property sheets, or the System Tree, and drag-and-drop items to add content and formatting. You can then edit these tags and attributes in the source code.

For more information on editing techniques in Source view, see “Basic editing in Page and Source views” on page 49.

Formatting the HTML source

Source view in the HTML editor can format your HTML code to make it more readable. Use the Editors tab of the Options Properties dialog box to specify formatting. You display the Options Properties dialog box by selecting the Design>Options menu item from the HTML editor menu bar.

You can also select an option to format the source code automatically, or trigger the formatting manually from the Source view pop-up menu.

For more information on formatting Source view display, see “Formatting HTML source display” on page 47.

Preview view

Preview view lets you test the work you do in Page and Source view. It defaults to the Microsoft Internet Explorer (IE) browser to display your current page and execute the client scripts it contains. Server scripts are ignored.

Choosing a view to work in

Page view and Source view provide very different ways of working with HTML. This table lists some of the advantages and disadvantages of each view.

Table 3-1: Choosing a view to work in

View	Advantages	Disadvantages
Page	<ul style="list-style-type: none">• Displays paragraph and character formatting• Provides easy manipulation of tables and absolute positioning of elements• Does not require detailed HTML knowledge• Focuses on content, not HTML tags• Incorporates Script editor	<ul style="list-style-type: none">• No manual control over the layout of the HTML source code• Extra HTML tags added for formatting purposes• Some HTML elements not supported• Cannot edit Title section of document, although you can use Page Properties to set the title and you can add LINK and STYLE tags from the Format menu
Source	<ul style="list-style-type: none">• Complete control of HTML source code layout• Use formatting menu items or edit HTML tags directly• Edit the whole document in one window, including HEAD and scripts	<ul style="list-style-type: none">• Must know HTML• Concentration on formatting instead of content

Edit in Source view to keep HTML source from being reformatted

If you do not want the editor to alter the layout of your HTML source, use Source view instead of Page view for all of your editing. By default, if you edit in Page view, the source will be reformatted.

Switching between views

You can switch between views by clicking the tabs at the bottom of the editing pane. Each view preserves its own insertion point. When you switch back to a view, especially if you have done no editing, the cursor will be where you left it.

If you make changes in a view, it can affect the cursor position in another view. For example, if changes you make in Source view cause to recalculate the layout, the cursor moves to the start of the file. If in Page view you delete or insert text that is before a Source view insertion point, the insertion point moves accordingly.

Remembering views from a previous session

PowerBuilder provides an option to automatically reload any editor windows that were open when you ended your previous session. If you select this option, the next time you open your workspace, the HTML editor will redisplay the pages you had open in the previous session.

When reloading an HTML editor window, PowerBuilder displays the view that you were last in (Page, Source, or Preview).

❖ **To reload pages that were open when you ended a previous session:**

- 1 Select Tools>System Options from the PowerBuilder menu.
- 2 Select the Workspaces tab, check Reload Painters When Opening Workspace, and click OK.

Opening the HTML editor and setting options

The following procedures describe how to use the HTML editor to prepare HTML files. HTML tags and their attributes can be dragged and dropped from the Language tab of the System Tree onto a page in the HTML editor. The tags are organized alphabetically or by category.

Opening the HTML editor

You can open the HTML editor with a new file that you create with a JSP page wizard, or you can open the editor with an existing JSP file, regardless of whether it is in your current JSP target.

Creating a new document

PowerBuilder has several wizards that help you create new JSP pages. The main JSP page wizards are listed below.

Table 3-2: JSP page wizards

Web page wizard	Use this to
Quick JSP Page	Create an HTML page without any content (HEAD and BODY elements are visible in Source view as soon as you begin to add content to the page).
JSP Page	Create an HTML page with a file name and optional content, such as a title, an associated style sheet, a background image or color selection, a header based on the title, and a footer with the page creation date.
4GL JSP Page	Create a 4GL JSP page with error and trace options, parameter definitions, and EAAServer component selections in addition to the standard content options of the Web Page wizard.
JSP DataWindow Page	Creates an HTML page with a Web DataWindow. You define the DataWindow source and connection information in wizard screens.

❖ **To start the editor with a new document:**

- 1 Select File>New.
Click the JSP tab in the New dialog box.
- 2 Double-click a JSP page wizard icon.
- 3 Follow the instructions in the wizard.
- 4 When the HTML editor opens, begin editing in Page view
or
Click the Source tab to edit in Source view.

Starting the editor with an existing document

PowerBuilder can automatically open text, style sheets, scripts, and image files. A file's treatment is based on its extension:

- Text files with TXT extensions are opened for editing in the PowerBuilder File editor.
- Text files with HTM, HTML, or JSP extensions are opened for editing in the HTML editor.
- Style sheet files are opened in the standalone Style Sheet editor.
- Script files are opened in the standalone Script editor.
- Image files are opened in a browsing window for viewing only.

Dragging files onto an open page in the HTML editor

Text files, style sheets, or script files can also be referenced as hyperlinked documents. If you drag and drop a file from the System Tree (or from an external file management system) onto an open page in the HTML editor, the dragged file is treated as a hyperlink reference, and the Hyperlink Properties dialog box displays.

❖ To open an existing JSP target file:

- Double-click the file in the Workspace tab of the System Tree

❖ To open a file that is not part of a target:

- Select File>Open from the menu bar, select a file type in the Files Of Type text box, and browse to find the file in the Open dialog box

Saving your work and closing the editor

❖ To save changes to a file:

- Select File>Save

For a new file, File>Save displays the Save As dialog box so you can name the file.

❖ To create another copy of the file:

- Select File>Save As

If you try to close the editor without saving, it prompts you to save the changes.

Giving your page a title

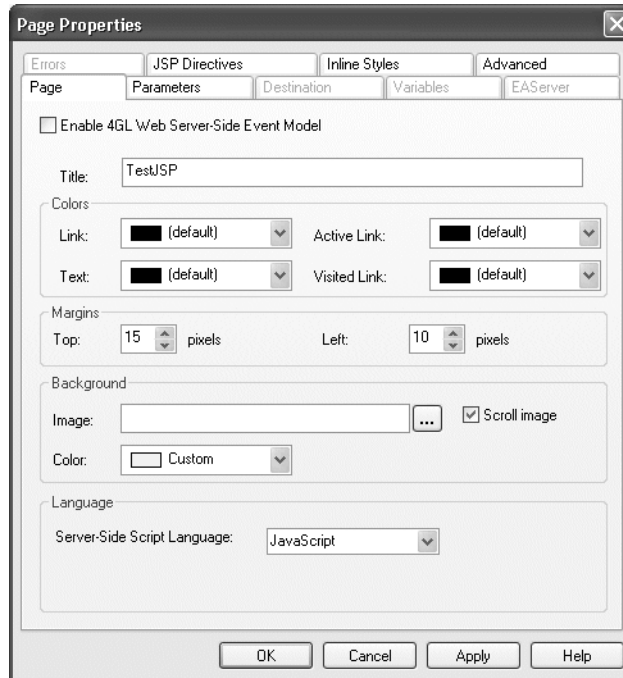
Use the Page Properties dialog box to add a title and to set inline styles and parameters for your HTML or JSP page.

4GL page properties for JSP targets

If you select the Enable 4GL Web Server Side Event Model check box on the Page tab of the Page Properties dialog box, you enable additional dialog box tabs. These tabs allow you to define server-scriptable properties for your page. For information on 4GL JSP pages, see Chapter 9, “Developing 4GL JSP Pages.”

❖ **To give your page a title:**

- 1 Right-click anywhere in a document in Page view and choose Page Properties from the pop-up menu.
- 2 Type a title for your page in the Title text box on the Page tab of the Page Properties dialog box.



Formatting HTML source display

Why use HTML source formatting

Source view allows you to format your HTML source code for readability. This feature enables you to specify your own HTML source formatting rules. It overrides the default formatting that the editor applies when you edit a file in Page view or use a tool or menu item to generate code in Source view.

This feature is important if you use Page view to develop or modify your HTML files. Page view edits your HTML source code behind the scenes, then formats that code according to its own rules for indenting, new lines, and so on. Although this default formatting is generally adequate, it might differ from the coding style you want.

Source formatting options enable you to override the default formatting generated from Page view with your own code formatting rules.

Invoking HTML source formatting

If you select the Format Source option (on the Editors tab of the Options Properties dialog box), PowerBuilder automatically performs HTML source formatting in these cases:

- When you switch to Source view from other HTML editor views after making changes
- When you perform operations in Source view that generate code

Whether or not you check the Format Source option, you can invoke HTML source formatting manually at any time by selecting the Format Source command on the Source view pop-up menu.

Preserving your own source formatting

If you prefer to format your HTML source code manually and keep it in that format, use Source view instead of Page view for all your editing, and do not select the Format Source command from the Source view pop-up menu.

❖ To change the Source view display formatting:

- 1 Select Design>Options.

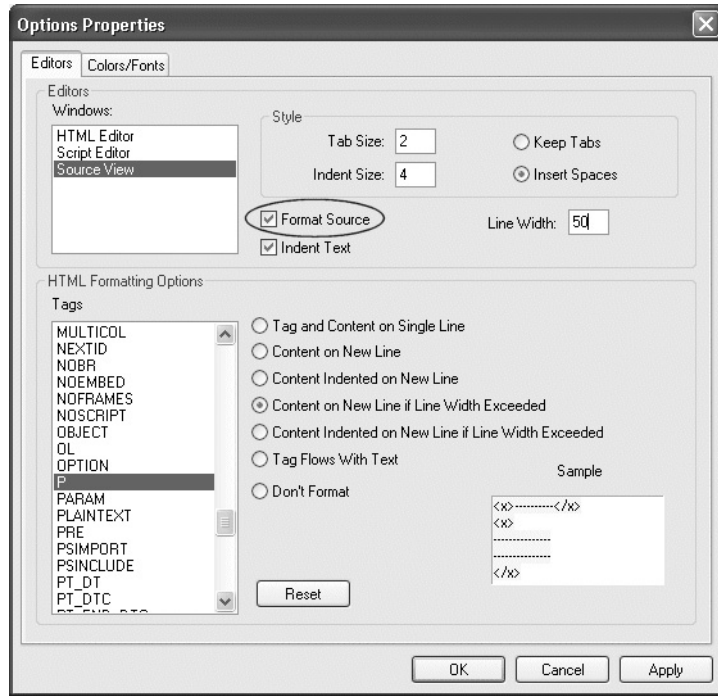
The Options Properties dialog box opens to the Editors page.

- 2 In the Windows list, highlight Source View.

Select the Format Source check box.

Type or select other options in the Editors panel.

- 3 Select an HTML tag or Script tag from the Tags list.
Select a radio button for the type of formatting you want for that tag.



- 4 Repeat the previous step for as many types of tags as you want to format, and click Apply.
- 5 Select the Colors/Fonts tab.
In the Window list, highlight Source View.
- 6 Select a display element in the Types list.
Modify the display element colors and fonts as desired.
- 7 Repeat the previous step for as many display element types as you want to format, and click OK.

Basic editing in Page and Source views

There are several methods for adding and changing content on your page in Page and Source views of the HTML editor:

- Using options on the PowerBuilder menu
- Clicking toolbar buttons
- Using keystrokes assigned to format settings
- Dragging elements from the System Tree into the editor
- Dragging HTML content saved in the PowerBuilder Clip Window into the editor
- Adding styles from a style sheet

Validating HTML source code

If you use toolbar or menu items to insert content or format settings in Page view, the HTML editor generates valid HTML code in Source view. If you use the same toolbar or menu items in Source view, you must make sure your choice of insertion point does not corrupt other HTML tags or otherwise invalidate the HTML structure of your document.

Using the PowerBuilder menu

PowerBuilder menu items can be used to add new content or modify existing content and formatting in Page or Source view.

Table 3-3: PowerBuilder menu items for adding or formatting page content

Menu item	Type of content	Description of new or existing page content
Format	Paragraph formatting	You use the Format menu to choose a paragraph style for the current paragraph. When you choose a style, the editor puts HTML tags like P, H1, PRE, or ADDRESS around the paragraph. The tags are visible in Source view.
	Character formatting	The Character menu item lets you change the font and apply bold and italic to selected text.

Menu item	Type of content	Description of new or existing page content
Table	HTML tables	Launch the table wizard to add a table to your page in Page or Source view. You can also modify table settings and formatting for the table (or table items) you select in Page view.
Insert	Non-text content	Lets you add line breaks, rules, images, and components such as Java applets, ActiveX controls, and data for plug-ins. You can use DataWindow design-time controls (DTC) to generate and edit database forms, application server access, and more.
Position	Content positioning	Lets you turn on absolute positioning for certain kinds of HTML elements. Use absolute positioning to place elements anywhere on the two-dimensional space of your page just by dragging them.
Edit	Document, control, or paragraph information	Select the Properties menu to view and set default information for the document, selected control, or selected paragraph.
	Style sheet links	Select the Global Style Sheet menu item to select an external link to a style sheet and include embedded styles in the Head section of the page.

Do not format empty paragraphs

When you use the Format>Paragraph dialog box in Page view, do not format empty paragraphs. Type some text and then apply the format. Otherwise, the format you choose will be discarded.

Formatting tips

Paragraph and character formatting

Here are tips for some common formatting activities.

Table 3-4: Tips for common formatting activities

To	Do this
Change heading styles	Select the text or put the insertion point somewhere in the heading and use Format>Paragraph and choose a heading level
Create lists from paragraphs	Select the paragraphs or some portion of the paragraphs and use Format>Paragraph and choose a list type
Change font or font characteristics of selected text	Select the text and use Format>Character and choose the preferred font type, style, or size

Paragraph style

The paragraph styles you can select from the Paragraph dialog box are not identical to the HTML tags that are generated in the source code. For correspondences between paragraph styles and HTML tags, see “Headings and paragraphs” on page 57.

Clip window

If you regularly use a particular document template, you can store the HTML or JSP file in the PowerBuilder Clip Window (up to 2048 characters per clip entry) for easy reuse. Use Source view when your template includes a Head section or controls. In Page view, only text elements in the Body section of the page get copied to or from the Clip Window.

Redundant formatting

Both the Format menu and System Tree tend to add elements without removing elements that are redundant. By using keystrokes and the toolbar, you can do a better job of changing existing formatting without adding redundant tags. Avoid a buildup of elements that cancel each other out by checking Source view and removing redundant tags.

Using the System Tree

The Language page of the System Tree lists all the HTML elements supported by the common browsers. Use the System Tree to insert HTML elements and their attributes into your document.

Using drag and drop

Most HTML elements have start and end tags. You can insert HTML tags in Page or Source view using drag-and-drop. You can set attributes of elements through property sheets. When you select text and then drag an element from the System Tree, the editor puts the element's start tag before the selection and its end tag after the selection.

Do's and don'ts for using the System Tree with Page view

HTML elements dragged from the System Tree do not have any context. You must make sure elements and their attributes are properly nested. Because of this, the System Tree is more useful when you work in Source view. In Page view, it is better to select text before dragging or pasting elements from the System Tree onto the text.

Some HTML elements require you to select text in a page in the HTML editor before you can drag the elements from the System Tree to the page. Other elements (like lists) are not suited for insertion by dragging, because the results will not be properly nested. Here are some tips:

- *Do* drag to insert single-tag elements, like BR and HR. You can also use the Insert menu (or the toolbar line icon for a horizontal line).
- *Do* drag to apply simple character formatting to selected text. You can also use keystrokes or the toolbar.
- *Do* drag to apply the FONT tag to selected text. You can also use Format>Character.
- *Do* drag to change a paragraph to a heading or a heading to another heading type. The whole paragraph is affected regardless of selected text.
- *Do not* drag to create complex nested elements, like lists. Use Format>Paragraph to convert existing paragraphs to a list.

Selecting text before dragging from the System Tree

When you drag onto selected text in either Page or Source view, the text will be formatted according to the chosen element.

For some elements, such as FONT, if you drag to an insertion point, you will not see any effect, but an element is inserted anyway. In Page view you cannot position the cursor between the start and end tags, so what you type next is not affected by the element.

Other elements, such as the anchor <A> and <TABLE> elements, display a properties dialog box. If you close the properties dialog box without specifying properties, future changes to element attributes must be made using Source view.

Inserting an element from the System Tree

❖ **To insert an element from the System Tree:**

- 1 Select text in the editor if applicable, or insert the cursor at the insertion point.
- 2 Use drag-and-drop or copy and paste to move the element from the Language tab of the System Tree to the editor where you want it to appear.

To surround the selected text with the element's start and end tag, you must place the element on top of the selected text.

If applicable, the properties dialog box for the new element displays.

- 3 Fill in any properties as appropriate.

Inserting an attribute for an element from the System Tree

You should use the System Tree to insert an attribute for an element only when you are editing in Source view. Dragging an attribute to the page in Page view only adds the text for the attribute—plus an equals sign—to the open page in a displayable format. (In this case it is not added as an attribute of an element.)

❖ **To insert an attribute for an element from the System Tree:**

- 1 Insert the element.
- 2 Drag the attribute from the Language tab of the System Tree to its correct position within the element's brackets.
- 3 Enter a value for the attribute.

Setting attributes from a properties dialog box

You can also set attribute values using the properties dialog box for an element.

Properties for HTML elements

Each HTML element has a properties dialog box for its attributes. The first tab, labeled with the element name, displays settings for the common attributes. You can define inline styles on the Inline Styles tab, or add other attributes on the Advanced tab.

In many cases, when you insert a new instance of the element, PowerBuilder displays the properties dialog box so that you can set attribute values. You can display the properties dialog box anytime.

If you give names or IDs to HTML elements, the pop-up menu that displays the available properties dialog boxes uses your name instead of the generic HTML element.

Displaying element properties in Page view

Because many tags cannot be displayed in Page view, you cannot always target a particular element. The pop-up menu includes items for all the elements in effect at the insertion point. For example, when you click on a link, you can choose to view the properties dialog box for the <A> (link), <P> (paragraph), or <BODY> element.

❖ **To display the properties dialog box for an element in Page view:**

- 1 Right-click the element's text.
- 2 Select one of the property menu items on the pop-up menu, such as Paragraph Properties or Body Properties.

Displaying element properties in Source view

❖ **To display the properties dialog box for an element in Source view:**

- Right-click the element's start tag or end tag and select Properties from the pop-up menu
or
Click to set the insertion point inside the element's brackets and select Edit>Properties from the menu bar.

Using the Inline Styles tab

If you want to override styles defined in an external style sheet or in an embedded style, you can do so on the Inline Styles tab.

Browser-specific implementation of style hierarchy

Some browsers might not permit the overriding of external styles with inline styles, or might have different implementations of style hierarchies. You should always test the appearance of styles and style overrides with the browsers that will be used to view your Web site.

❖ To add or modify a style definition using the Inline Styles tab:

- 1 In Source view, right-click the element's start tag or end tag and select Properties from the pop-up menu
or
In Page view, select the viewable object and right-click to access the Properties dialog box.
- 2 Select the Inline style type.
- 3 Select the Inline radio button and click the Edit button.
- 4 Specify style definitions for the selected tag on the tabbed pages of the Inline Style dialog box. To modify the font of the selected item:
 - Select the Font from the Available Font window.
 - Use the arrow key to add the font to the Selected Font window. Select the new font in the Selected Font window and click OK.
- 5 In the Style Attributes and Values source box, view the styles selected. Click Apply or OK when finished defining styles.

For more information on styles, see Chapter 4, “Working with Style Sheets and Framesets.”

Using the Advanced tab

If an attribute you want to set does not correspond to a property on the main tab of the element properties dialog box, you can set it on the Advanced tab.

The attributes you enter are not verified as valid. Make sure you check the HTML reference or the Language tab of the System Tree for valid attributes.

❖ To add an attribute on the Advanced tab:

- 1 From Source view right-click the element's start tag or end tag and select Properties from the pop-up menu
or
In Page view select the viewable object, and right-click to access the Properties dialog box.
- 2 Double-click to type an appropriate value in the Attribute Name column and the Value column.
- 3 Do not include quotes when you specify the value.

If the value requires quotes (for example, if it includes spaces), they are inserted automatically.

❖ **To remove a setting:**

- 1 Click anywhere on the line for the attribute you want to delete.
- 2 Press the Delete key or the Delete button.

Undo and Redo

While you remain in a single view, you can use Edit>Undo multiple times to undo each change you make. You can also use Edit>Redo multiple times.

If you switch to another view, all the changes you made in the first view become a single set of changes. When you use Undo after switching views, all the changes made in the previous view are undone at once.

Finding and changing text

❖ **To find or replace text in Page view or Source view:**

- 1 Select Edit>Find or Edit>Replace on the menu bar

or

Right-click a page in the HTML editor and select Find or Replace from the pop-up menu.

Settings in the dialog box let you control the direction of the search and whether uppercase and lowercase letters must match the search string.

- 2 Specify a search string in the Find text box.

In either view, you search for the text as you see it displayed. In Source view, you can search for HTML tags and property values.

If you need to change many element tags or property values, switch to Source view. It is the most efficient way to make multiple similar changes.

Special characters

The editor does not support searching for special characters, such as line breaks and tabs.

Using the Script editor

In Page view, the Script editor lets you associate scripts with objects and events in the HTML document. You can also define new scripts and functions that are independent of an object. The editor handles the HTML syntax for scripts automatically. You can save a script on the page itself or in an external file.

For more information, see Chapter 6, “Writing Scripts.”

Correspondences of common elements

When you insert a control or choose a paragraph or character style to include on your page, PowerBuilder adds HTML syntax to Source view that enables Web browsers to render the object or style selected.

Headings and paragraphs

You can add headings and paragraphs to your page from a dialog box that you open from Page view or Source view with the Format>Paragraph menu command. Table 3-5 lists the paragraph styles in the Format Paragraph dialog box for headings and paragraphs.

Table 3-5: Format menu items for paragraphs and headings

Paragraph style in Page view	HTML tag in Source view	Description
Normal	<P></P>	A standard paragraph
Formatted	<PRE></PRE>	A paragraph that preserves all spacing including extra white space and is usually displayed in a monospaced font
Address	<ADDRESS> </ADDRESS>	Usually displayed in italic
Heading 1 to Heading 6	<H1></H1> to <H6></H6>	Headings of various levels

Other styles in the Format Paragraph dialog box can be used to format lists. For information about formatting lists, see “Lists” on page 59.

❖ **To create a heading in Page view:**

- 1 Type the heading text and leave the insertion point in the heading paragraph.
- 2 Select Format>Paragraph from the menu bar.
- 3 In the Paragraph Style list box, select one of the heading styles (Heading 1 through Heading 6).

❖ **To create a heading in Source view:**

- 1 In the Body section, select the heading text
or
Put the insertion point where you want the heading to appear.
- 2 Select Format>Paragraph from the menu bar.
- 3 In the Paragraph list box, select one of the heading styles (Heading 1 through Heading 6).
- 4 After you click OK, the Header Properties dialog box appears. Add text if new, change the properties if you want, and click OK.

❖ **To format text with a paragraph style in Page view:**

- 1 Type at least some of the paragraph text and leave the insertion point in the paragraph.
- 2 Select Format>Paragraph from the menu bar.
- 3 In the Paragraph Style list box, select one of the paragraph styles.

❖ **To format text with a paragraph style in Source view:**

- 1 Select all the text of the paragraph
or
Put the insertion point in the Body section where you want the paragraph to appear.
- 2 Select Format>Paragraph from the menu bar.
- 3 In the Paragraph Style list box, select one of the paragraph styles.
- 4 After you click OK, the Paragraph Properties dialog box appears. Change the properties if you want to, and click OK.
- 5 If you did not select the paragraph text in step 1, type the text now between the paragraph's start and end tags.

Lists

There are several list types available in the Format>Paragraph dialog box. The basic types are numbered and bulleted. In most browsers, Menu and Directory List styles also appear as bulleted lists.

Table 3-6: Format menu items for lists

Paragraph style in Page view	HTML tag in Source view	Description
Numbered List	 	An ordered list
Bulleted List	 	An unordered list
Directory List	<DIR> </DIR>	A directory list
Menu List	<MENU> </MENU>	A menu list
Definition Term	<DL><DT> </DL>	The definition term in a definition list
Definition	<DD>	The definition value in a definition list

If you do not see the formatting you specify, you must make sure that list item tags () precede each item in a regular (ordered or unordered) list. For a definition list, you must make sure that the correct definition tags (<DT> or <DD>) precede all the terms and definitions in the list. You must verify the positioning of the tags in Source view.

The two-part definition lists are more complicated. Procedures for using them are described separately.

Ordered and unordered lists in Page view

❖ **To create a new list:**

- 1 With the insertion point in an empty paragraph, choose Format>Paragraph from the menu bar and select the type of list you want.

The editor inserts a number or a bullet.

- 2 Type the item text and press enter.

The editor inserts another numbered or bulleted paragraph.

- 3 Continue typing items and pressing enter.
If you press enter with the insertion point at the end of any list item, the editor inserts another item.
 - 4 When you have finished, end the list by pressing enter in an empty list item.
The editor removes the last empty bullet or number and changes the paragraph style to Normal.
- ❖ **To change paragraphs into list items:**
- 1 Highlight a group of paragraphs.
 - 2 Select Format>Paragraph from the menu bar and choose the type of list you want.

Definition lists in Page view

Each item in a definition list has two parts: the term or phrase being defined, and the definition.

- ❖ **To create a definition list:**
- 1 Type the first term.
 - 2 With the cursor in the term paragraph, select Format>Paragraph from the menu bar and select the Definition Term paragraph style.
 - 3 Back in the editor, press ENTER and type the term's definition.
 - 4 In the editor, press ENTER and repeat the steps to create terms and definitions
or
Press ENTER twice to end the list.

Typing definitions and formatting paragraphs in separate procedures

Each time you press ENTER, you create another element of the same type. If you're in a Definition Term, pressing ENTER creates another term. Instead of formatting each paragraph right after you type it, you can apply formatting as needed to selected paragraphs.

Lists in Source view

For regular lists The Format Paragraph dialog box inserts the list container in your document. You must insert LI elements for each list item.

For definition lists The Format Paragraph dialog box inserts DL, DT, and DD elements when you choose the Definition Term and Definition styles. You might have to type one or more of these elements directly in the source code.

❖ **To insert the list container:**

- 1 If the list items are already in the document, select all the items

or

Put the insertion point in the body where the list should be.

- 2 Choose Format>Paragraph and select a list style.

When you click OK, the properties dialog box for the list appears. The list style name shown matches the element, not the styles of the first dialog box. (For example, Numbered List is now called Ordered List for the OL element.)

- 3 Set properties if you want to, and click OK.
- 4 For regular lists, add tags before the list items.
For definition lists, make sure <DT> and <DD> tags are included before the appropriate definition list items.

Character formatting

You can apply character formatting to selected text, or you can choose settings so that the formatting applies to the next text you type.

❖ **To change font characteristics:**

- Select Format>Character from the menu bar and specify settings in the Font dialog box.

Applying simple formatting

The Format Character dialog box always inserts the FONT element, even if you want only to turn on bold or italic. Use keystrokes or toolbars to apply simple formatting.

Inserting special symbols

Insert Symbol can be used in Source or Page view.

❖ **To insert special symbols or accented characters:**

- 1 Set the insertion point in Source view or Page view.
- 2 Select Insert>Symbol from the menu bar.
- 3 In the Insert Symbol dialog box, select a symbol.

The named entity or numeric value of the symbol displays in the Equivalent Escape Sequence box.

- 4 Click OK.

Links and anchors

When you type a URL that uses an HTTP protocol directly in Page view, the editor automatically turns it into a hyperlink. If you want to display different text for the hyperlink, you should use the following procedure.

❖ **To create a hyperlink:**

- 1 Select the text that you want to display for the link or set the insertion point where you want to add the link.

- 2 Choose Format>Hyperlink from the menu bar.

The Hyperlink Properties dialog box displays.

- 3 If you did not select text in step 1, type the text you want to display for the hyperlink in the Text of the Hyperlink text box.

You cannot type text if you selected text in step 1. Instead, the Text of the Hyperlink text box is grayed, and it displays the text that you selected.

- 4 Enter the URL for the link in the Destination text box

or

Click the browse button to open the Choose URL dialog box.

When you click OK, the selected text becomes a hyperlink and is underlined. In Source view, you can enter the link ID and text between the A tags.

- ❖ **To create an anchor that can be a target of a hyperlink:**
 - 1 Select text or set the insertion point where the anchor should be.
 - 2 Choose Format>Hyperlink from the menu bar.
 - 3 Click the Advanced tab and type the attribute name in the left column and the value (name) of the anchor on the right. Include quotes around the value.

- ❖ **To link to an anchor within your document:**
 - 1 Select the text that will be the link.
 - 2 Choose Format>Hyperlink from the menu.
 - 3 In the Link text box, type a pound sign (#) followed by the anchor name.

- ❖ **To transform a URL into a hyperlink:**
 - 1 Enter the URL in your document, followed by a space. The URL does not need to be complete—it needs just enough for the editor to recognize it as a URL. The text will be turned into an underlined active link target.
 - 2 Edit the underlined text if you want to.

Spaces in link text

If you select the link text and begin typing, you replace the selected text as usual. When you type a space, the editor takes you out of the link so that you can type normal text. To create link text that includes spaces, you can:

- Type the text with no spaces and insert the spaces afterward.
- Type the text with spaces, then cut the text that is no longer part of the link but should be, and paste it at the end of the link.
- Select all but the first or last character of the displayed URL, type the link text, then delete the non-selected characters from the URL when you have finished.

Correcting link problems When you paste a relative link into Page view, the extra text *about:* might sometimes appear in the HREF. If this happens, use Source view to remove the extra text.

More complex formatting

Forms

The Insert menu has items for several types of form fields. The menu items insert the HTML elements displayed in the following table:

Table 3-7: HTML elements added to Web page by Insert menu items

Menu item	HTML element
Single Line Text	Input TYPE=TEXT
Text Area	Textarea
Text (for 4GL pages only)	Object
Check Box	Input TYPE=CHECKBOX
Radio Button	Input TYPE=RADIO
List Box	Select
Push Button	Input TYPE=BUTTON, SUBMIT, or RESET
Image Button	Input TYPE=IMAGE
DataWindow	Object

If you insert any of these items into a non-4GL page, FORM tags are also inserted automatically as long as the insertion point is not already inside a FORM element. In Page view, you can add the FORM element yourself by selecting all the fields you want to include in a form, then dragging the FORM element from the Language tab of the System Tree to the selection.

Do not add FORM tags to a 4GL-enabled Web page

When you work with a 4GL page, the page itself is a form, and therefore all forms are submitted as a single form. Existing FORM tags must be manually removed from a 4GL page.

❖ **To insert form fields:**

- 1 Select Insert>Form Field from the menu bar and select a type of form field from the cascading menu.
- 2 Add text to the form by typing before and after the inserted fields.

If you are working in Page view, check Source view to make sure text and fields are nested correctly inside the FORM element.

Tables

Adding new tables

You can use the Table wizard to add a table to your Web page.

❖ **To insert a table:**

- 1 Select Table>Table Wizard from the menu bar.
- 2 Use the Table wizard to specify the number of rows and columns and to specify formatting for the table, individual rows, and individual cells.
- 3 Type the content of cells in the Create Table dialog box or directly in the document.

Reorganizing existing tables

You can manipulate rows, columns, or cells in an existing table from the Table menu or from a pop-up menu when you right-click on the table items you want to modify. Table actions are available only in Page view.

Table 3-8: Actions for manipulating table rows, columns, or cells

Action	What it does
Insert Row	<p>Inserts a new row above the current one.</p> <p>The new row will contain the same number of cells as the current row, with the same COLSPAN attributes, cell attributes, and styles.</p>
Insert Column	<p>Inserts a new column to the left of the current one.</p> <p>The new column will contain the same number of cells as the current column. The individual cell attributes are copied cell for cell from the current column to the new one.</p>
Insert Cell	<p>Inserts a single cell to the left of the current one.</p> <p>If your selection includes more than one cell, the current cell is defined as the one that's leftmost and topmost in the selection. When the new cell is inserted, individual cell attributes are copied from the current cell to the new one.</p>
Delete Row	<p>Deletes the selected rows.</p> <p>If your selection includes more than one row, all rows containing any portion of the selection will be deleted. It is not necessary to select the entire contents of a row.</p>
Delete Column	<p>Deletes the selected columns.</p> <p>If your selection includes more than one column, it must be within a single row. All columns containing any portion of the selection will be deleted. It is not necessary to select the entire contents of a column.</p>

Action	What it does
Delete Cell	Deletes the selected cells. If your selection includes more than one cell, all cells containing any portion of the selection will be deleted. It is not necessary to select the entire contents of a cell.
Merge Cells	Merges two or more cells into a single cell. All cells containing any portion of your selection will be merged. It is not necessary to select the entire contents of a cell. When cells are merged, their contents are concatenated in the remaining cell. The merged cells assume the attributes of the cell that was leftmost and topmost in the selection.
Split Cell	Splits one cell into two. The selected cell is split horizontally—an empty cell is added to its right.

❖ **To manipulate rows, columns, or cells in an existing table:**

- 1 In Page view, highlight text in the rows, columns, or cells you want to manipulate.
- 2 Select Table from the menu bar
or
Right-click the highlighted text and select Table from the pop-up menu.
- 3 Select the menu item for the action you want from the Table menu.

The overall table width is not altered when you perform any of the table actions. Instead, the cell widths are adjusted. When you are working in Page view, table cells might appear equal in size. By selecting the table and using the mouse, you can expand or shrink the width of the columns.

Other formatting

To use absolute positioning for elements on a page, see "Absolute positioning" next. To add images, components, and other non-text content, see Chapter 5, "Working with Images, Other Media, and Components."

Absolute positioning

You can use absolute positioning on the HTML editor's Page view. The following sections describe how it works and what it can do for you.

About absolute positioning

Dynamic HTML allows HTML elements to be positioned on a page, independent of their position within the HTML stream. An absolutely positioned HTML element has its position attribute set to absolute instead of static.

Absolutely positioned elements also have a z-index, which specifies the visual order of overlapping absolutely positioned elements (and how absolutely positioned elements are ordered relative to elements in the HTML stream).

An absolutely positioned element is also known as a two-dimensional (2D) element. A statically positioned element is known as a one-dimensional (1D) element. A relatively positioned element (an element with its position attribute set to relative) is treated as a 1D element.

Browser specificity

Absolute positioning is implemented differently in Netscape and Internet Explorer. The HTML editor implements absolute positioning that is optimized for Internet Explorer. This implementation does not work with Netscape browsers. (Absolute positioning in Netscape requires the use of LAYER tags or STYLE tags with a position property.)

What you can do

The HTML editor's Page view makes it easy to work with absolutely positioned elements. By using the Position menu, you can toggle an element from static positioning to absolute positioning and vice versa. Once an element uses absolute positioning, you can place it anywhere on the 2D space of your page just by dragging it.

The Position menu also enables you to:

- Change the z-index of an absolutely positioned element
- Move (nudge) an absolutely positioned element by a specified number of pixels
- Lock an absolutely positioned element in place to prevent it from being inadvertently moved or resized
- Constrain absolutely positioned elements to move only horizontally or vertically when you drag them
- Adjust the invisible grid that Page view provides to help you place absolutely positioned elements

Elements that can be absolutely positioned

You can use absolute positioning on the following kinds of elements:

APPLET	HR	OBJECT
BUTTON	IFRAME	SELECT
DIV	IMG	SPAN
EMBED	INPUT	TABLE
FIELDSET	MARQUEE	TEXTAREA

Absolute positioning is not available for other kinds of elements or for design-time controls (DTCs).

Using style sheets for absolute positioning

If you want to use the same position definitions in a number of files, you can set these values in an external style sheet.

For more information, see Chapter 4, “Working with Style Sheets and Framesets.”

Toggleing between static and absolute positioning

In Page view of the HTML editor, you can use absolute positioning to place HTML controls anywhere on the 2D space of your page.

The z-index style attribute of a new absolutely positioned element is initially set higher than all other absolutely positioned elements in its document or container. As a result, that element will display in front of older absolutely positioned elements.

In addition, the new element's z-index always begins as a positive value, causing that element to display in front of the HTML (1D) stream of the page. (Absolutely positioned elements with a negative z-index display behind the HTML stream.)

❖ **To toggle from static positioning to absolute positioning:**

- 1 Select a control in Page view by clicking its outside edge.

The control should now display a dotted border. (If you see a slashed border, click that border to make it dotted.)

- 2 Select Position>Use Absolute Positioning from the menu bar
or

Right-click the dotted border and select Position>Use Absolute Positioning from the pop-up menu.

- 3 Drag the control anywhere you want on the page.

❖ **To toggle from absolute positioning to static positioning:**

- 1 Select a control in Page view by clicking its outside edge.

The control should now display a dotted border. (If you see a slashed border, click that border to make it dotted.)

- 2 Select Position>Use Absolute Positioning from the menu bar
or

Right-click the dotted border and select Position>Use Absolute Positioning from the pop-up menu.

The control automatically moves from its absolute position to its position within the HTML stream of the page.

Setting absolute positioning options

❖ **To set absolute positioning options for Page view:**

- 1 Select Position from the menu bar.
- 2 Select one of these actions from the Position menu:

Action	What it does
Constrain Positioning	Toggles constrain mode on or off. In constrain mode, absolutely positioned elements move along only one axis at a time (either X or Y) when you drag them. This enables you to adjust an element's horizontal position without affecting its vertical position and vice versa.
Set Grid Size	Sets the cell size of the invisible grid that absolutely positioned elements snap to when you drag them. You specify the X and Y values for the cell size (in pixels) in the Set Grid Cell Size dialog box.

Manipulating an absolutely positioned element

❖ **To manipulate an absolutely positioned element:**

- 1 Select a control in Page view by clicking its outside edge.

The control should now display a dotted border. (If you see a slashed border, click that border to make it dotted.)

- 2 Select Position from the menu bar

or

Right-click the dotted border and select Position from the pop-up menu.

- 3 Select one of these actions from the Position menu:

Action	What it does
Bring To Front	Brings the element to the front of its document or container. This sets the element's z-index style attribute to the highest of all the absolutely positioned elements on its side of the HTML stream, adjusting the z-index of other elements as necessary.
Send To Back	Sends the element to the back of its document or container. This sets the element's z-index style attribute to the lowest of all the absolutely positioned elements on its side of the HTML stream, adjusting the z-index of other elements as necessary.
Bring Forward	Brings the element forward by one z-index layer in its document or container, adjusting the z-index of other elements as necessary.
Send Backward	Sends the element backward by one z-index layer in its document or container, adjusting the z-index of other elements as necessary.
Bring Above Text	Brings the element in front of the HTML stream of the page by making its z-index a positive value.
Send Below Text	Sends the element in back of the HTML stream of the page by making its z-index a negative value.
Nudge Element	Moves the element from its current X and Y coordinates by the number of pixels you specify in the Nudge Object dialog box.
Lock Element	Locks the element in place to prevent it from being inadvertently moved or resized in Page view. Lock Element prevents you from changing the element's X and Y coordinates, but not its z-index. It does not affect the runtime behavior of the element.

JSP page authoring

JSP pages can be written in any well-formed language, including XML, but they are usually written in HTML. In PowerBuilder, when you create a new Web page, the page wizard gives it the extension *.jsp* by default instead of *.htm*. You edit JSP pages in much the same way as any other HTML page.

JSP authoring elements

Standard HTML elements, controls, and client-side scripts are available to JSP pages. In addition, JSP-specific elements are available in the development environment for editing JSP Web pages:

- JSP actions
- Directives
- JSP scripting elements
- Custom tags
- About JSP Web services

Page view icons

In the Page view, JSP standard actions and scripting elements are represented by icons showing the element’s delimiters. When you select a scripting element or a 4GL server-side event, Java is the only language available in the script editor.

Table 3-9: Icons displaying in Page view for JSP-specific elements

Icon	Description
<%>	Server-side scriptlet
<%=>	Server-side expression
<%!>	Server-side declaration
<jsp:>	Standard action, such as <jsp:useBean ... >
</jsp:>	Close tag of standard action, such as </jsp:useBean>
<jsp:/>	Self-closing standard action, such as <jsp:getProperty ... />
<ctl:>	Custom tag, such as <j2ee:action ... >
</ctl:>	Close tag of custom tag, such as </j2ee:action>
<ctl:/>	Self-closing custom tag, such as <j2ee:action ... />
<?:>	Unknown custom tag
<%@ins>	Include page directive, such as <%@ include ... %>

JSP actions

Actions are standard tags that perform common actions. All JSP standard actions use the prefix jsp.

You can insert any of the following actions:

- <jsp:useBean>** The `useBean`, `getProperty`, and `setProperty` actions are all used with JavaBeans components. The `useBean` `id` attribute is the name of the bean and corresponds to the `name` attribute for `getProperty` and `setProperty`. The `useBean` action locates or instantiates a JavaBeans component:
- ```
<jsp:useBean id="labelLink" scope="session"
class="LinkBean.labelLink" />
```
- The bean class and classes required by the bean class must be deployed under a JavaCode base that is available to the Web Application where the JSP is installed.
- <jsp:getProperty>** The `getProperty` action gets the value of a JavaBeans component property so that you can display it in a result page:
- ```
<jsp:getProperty name="labelLink" property="url" />
```
- <jsp:setProperty>** The `setProperty` action sets a property value or values in a JavaBeans component:
- ```
<jsp:setProperty name="labelLink" property="url"
value="<%= labelLink.getURL() %>" />
```
- <jsp:include>** The `include` action includes a static file or sends a request to a dynamic file:
- ```
<jsp:include page="cart.html" flush="true" />
```
- <jsp:forward>** The `forward` action forwards a client request to an HTML file, JSP file, or servlet for processing:
- ```
<jsp:forward page="/jsp/datafiles/ListSort.jsp" />
```
- <jsp:param>** The `param` action specifies request parameters in the body of an `include` or `forward` action. It can also be used in the body of a `params` action.
- ```
<jsp:forward page="/jsp/datafiles/ListSort.jsp" />
  <jsp:param name="bgColor" value="blue" />
</jsp:forward>
```
- <jsp:params>** The `params` action can be used only in the body of a plugin action to enclose the applet parameters specified by `param` actions.
- <jsp:plugin>** The `plugin` action downloads plug-in software to the Web browser to execute an applet or JavaBeans component. It generates HTML `<embed>` or `<object>` elements in the page. You can use the `params` and `param` actions to specify parameters required by the plug-in, and the `fallback` action to specify the text that displays if the browser does not support `<embed>` or `<object>` elements:
- ```
<jsp:plugin type=applet code="Calc.class"
codebase="/mathutils" >
 <jsp:params>
```

```
<jsp:param name="multiplier"
 value="multipliers/tax.val" />
</jsp:params>
<jsp:fallback>
 <p> unable to start plugin </p>
</jsp:fallback>
</jsp:plugin>
```

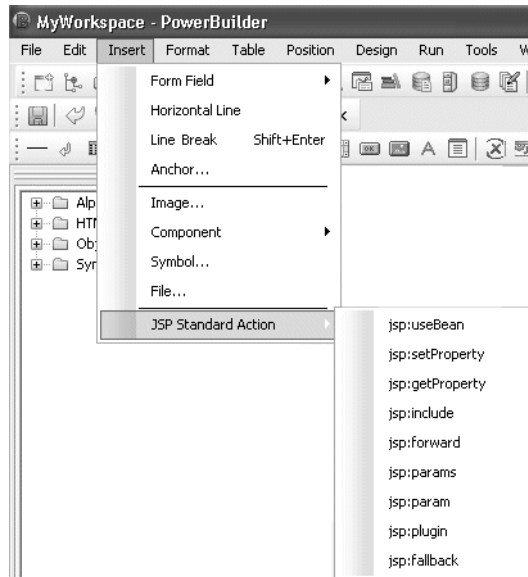
<jsp:fallback>

The fallback action can be used only in the body of a plugin action to specify the text that displays if the browser does not support <embed> or <object> elements.

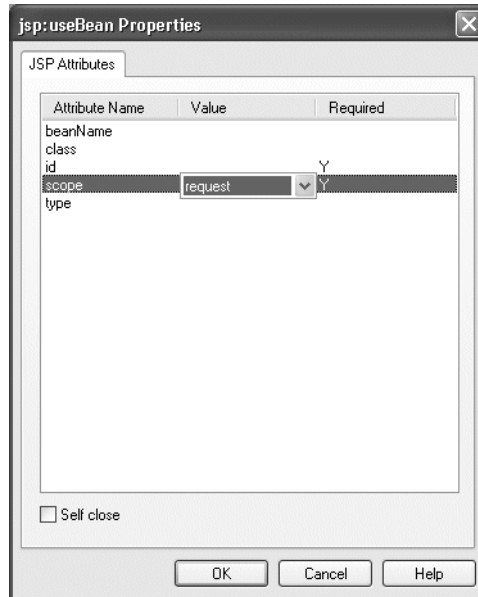
## Inserting an action

### ❖ To insert an action in a JSP page:

- 1 Select Insert>JSP Standard Action from the menu bar and select an action:



- 2 In the dialog box that displays, specify the values of the action's attributes. A Y in the Required column indicates that you must specify a value for the attribute:



For a description of each of the values available for the scope attribute of the `<jsp:usebean>` action, see “Scopes” on page 124.

## Adding applets and JavaBeans

Adding applets and JavaBeans to a JSP page inserts the appropriate JSP action. To view JavaBeans and applets on the Components tab of the System Tree, you must make sure that the component you want and the *WTInfo110.jar* file are included in the Java class path. The *WTInfo110.jar* is installed in the *Sybase\Shared\Web Targets* directory. It should be included in the class path by default.

### Adding applets

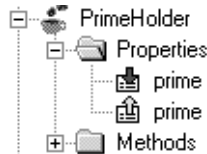
When you drag an applet from the Components tab to a JSP page in Page view or Source view, the `jsp:plugin` Properties dialog box displays with default values for the applet you selected. When you click OK, the applet is added to the page in a `jsp:plugin` action tag.

Adding JavaBeans and JavaBean properties

When you add an applet to a JSP page, you must make sure the applet classes are stored in a location accessible to client browsers. You can assign this location, using a file or http protocol, to the codebase attribute of the `jsp:plugin` directive.

When you drag a JavaBean from the Components tab to a JSP page in Page view or Source view, the `jsp:useBean` Properties dialog box displays with default values for the JavaBean you selected. When you click OK, the JavaBean is added to the page in a `jsp:useBean` action tag. If the JavaBean is in a class file, the class file is added to the `Web-Inf\classes` directory for your target. If the JavaBean is in an archive file, the archive file is added to the `Web-Inf\lib` directory for your target.

JavaBean properties with both read and write permissions are listed twice on the Components tab: one time for the read property and another time for the write property. The icon for the read-enabled property is a yellow arrow pointing upward. The icon for the write-enabled property is a green arrow pointing downward.



When you drag a read-enabled JavaBean property from the Components tab to a JSP page, the `jsp:getProperty` Properties dialog box displays with default values for the JavaBean property you selected. When you drag a write-enabled JavaBean property from the Components tab to a JSP page, the `jsp:setProperty` Properties dialog box displays with default values for the JavaBean property you selected.

## Directives

Directives are messages to the JSP engine that provide global information for the page or include a file of text or code. Directives begin with the character sequence `<%@` followed by the name of the directive and one or more attribute definitions. They end with the character sequence `%>`.

There are three directives: `page`, `include`, and `taglib`.



**Page directive** The page directive defines attributes that apply to an entire JSP page, including language, the class being extended, packages imported for the entire page, the size of the buffer, and the name of an error page. For example:

```
<%@ page language="java" import="mypkg.*"
 session="true" errorPage="ErrorPage.jsp" %>
```

For more information about error pages, see “Error handling” on page 80.

**Include directive** The include directive includes a static file, parsing the file’s JSP elements:

```
<%@ include file="header.htm" %>
```

---

**Include directive and include standard tag** Note that the include directive parses the file’s contents, whereas the include tag does not.

---

**Taglib directive** The taglib directive defines the name of a tag library and its prefix for any custom tags used in a JSP page:

```
<%@ taglib uri="http://www.mycorp/printtags"
 prefix="print" %>
```

If the tag library with the prefix print includes an element called doPrintPreview, this is the syntax for using that element later in the page:

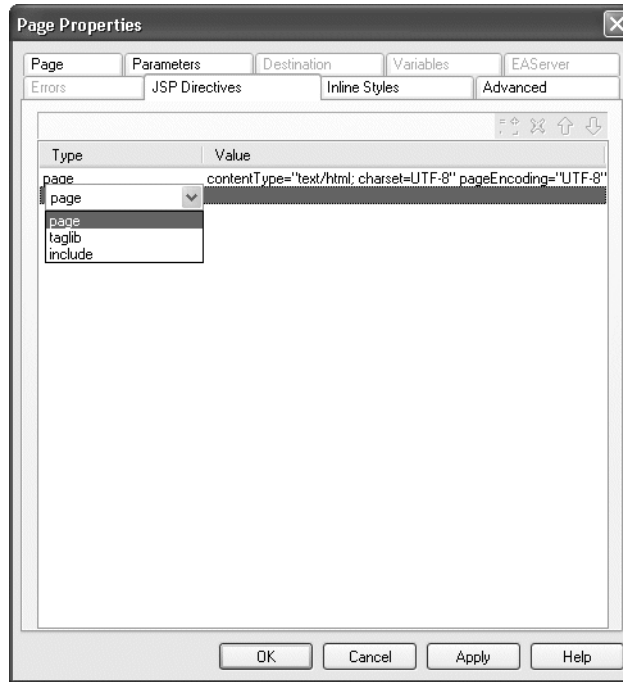
```
<print:doPrintPreview>
...
</print>
```

For more information, see “Custom tags” on page 79.

## Inserting a directive

- ❖ **To insert a directive in a JSP page:**
  - 1 Right-click inside a page in Page view and select Page Properties from the pop-up menu  
*or*  
Right-click inside the <BODY . . . > tag in Source view and select Properties from the pop-up menu.
  - 2 In the Page Properties or Body Properties dialog box, select the JSP Directives tab and click the New icon.

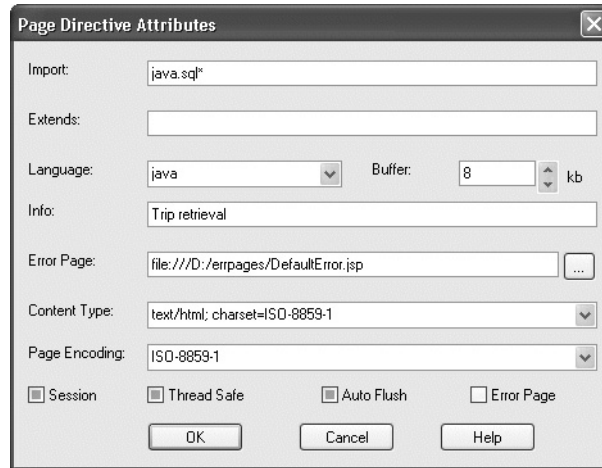
- 3 Select the type of directive you want to add in the drop-down list box in the Name column.



- 4 Click inside the Value column, then click the Browse (...) button that displays at the right of the Value column.

- 5 Complete the dialog box that displays.

The type of dialog box that displays depends on the type of directive you are adding. The Page Directive Attributes dialog box looks like this:



## Custom tags

Custom tags, also called tag extensions or custom actions, extend the capabilities of JSP pages. Tag libraries define a set of actions to be used within a JSP page for a specific purpose, such as handling SQL requests. The tag libraries you use in PowerBuilder can be built using another tool, although you can create custom tags for Web services using a PowerBuilder wizard (see “About JSP Web services” on page 155).

The URI identifying a tag library is associated with a Tag Library Descriptor (TLD) file and with tag handler classes.

### Tag handlers

A tag handler is a Java class that defines the semantics of an action. The implementation class for the JSP instantiates a tag handler object for each action in the page. Tag handler objects implement the `javax.servlet.jsp.tagext.Tag` interface, which defines basic methods required by all tag handlers, including `doStartTag` and `doEndTag`. The `BodyTag` interface extends the `Tag` interface by adding methods that enable the handler to manipulate its body.

Packaging tag libraries

To associate a tag library with a JSP page, you use a `taglib` directive that identifies the URI where the tag library's TLD file can be located. The TLD file must be in (or deployed to) the class path of the JSP container and is usually placed in the Web application's `WEB-INF/tlds` directory. The class files for the tag library must also be in the class path of the JSP container. Typically they are placed in the Web application's `WEB-INF/classes` directory or in a JAR file in the `WEB-INF/lib` directory.

For information on adding a `taglib` directive to a JSP page, see "Taglib directive" on page 77.

Using tag libraries in PowerBuilder JSP pages

In PowerBuilder, you can add tag libraries to a JSP page from the Components tab of the System Tree as well as from the JSP Directives page of the Page Properties dialog box. A tag library must be in the PowerBuilder custom tag library search path in order to be listed on the Components tab. You can add directories or tag library descriptor files to the custom tag library search path on the JSP page of the System Options dialog box.

When you add a tag library to a JSP page, a dialog box prompts you to enter a prefix. The prefix you enter is used as a shorthand entry to refer to the tag library when you add a tag from the library to the page. PowerBuilder automatically includes the path to the TLD file in the `web.xml` file for the target to which the page belongs. PowerBuilder also adds an entry for the tag library on the Tag Libraries page of the Deployment Configuration Properties dialog box for the target.

For more information about the Tag Libraries page of the Deployment Configuration Properties dialog box, see "Tag Libraries" on page 245.

## Error handling

When a client request is processed, runtime errors can occur in the body of the implementation class for the JSP or in Java code that is called by the page. These exceptions can be handled in the code in the JSP page, using the Java language's exception mechanism.

Uncaught exceptions

Exceptions that are thrown from the body of the implementation class that are not caught can be handled using an error page. You specify the error page using a page directive. Both the client request and the uncaught exception are forwarded to the error page. The `java.lang.Throwable` exception is stored in the `javax.HttpServletRequest` instance for the client request using the `putAttribute` method, with the name `javax.servlet.jsp.jspException`.

Using an error page  
JSP

If you specify a JSP page as the error page, you can use its implicit exception variable to obtain information about the exception. The exception object is of type `java.lang.Throwable` and is initialized to the `Throwable` reference when the uncaught exception is thrown. For more information about the exception object, see “Implicit objects” on page 122.

To specify an error page for a JSP, set its `errorPage` attribute to the URL of the error page in a page directive:

```
<%@ page errorPage="ErrorPage.jsp" %>
```

To define a JSP as an error page, set its `isErrorPage` attribute to `true`:

```
<%@ page isErrorPage="true" %>
```

This sample error page uses the exception object’s `toString` method to return the name of the class of the object causing the exception and the result of the `getMessage` method for the object. If no message string was provided, `toString` returns only the name of the class.

The example also uses the `getParameterNames` and `getAttributeNames` methods of the request object to obtain information about the request.

```
<%@ page language="java" import="java.util.*"
 isErrorPage="true" %>
<H1 align="Center">Exceptions</H1>

<%= exception.toString() %>
<%! Enumeration parmNames; %>
<%! Enumeration attrNames; %>

Parameters:
<%
 parmNames = request.getParameterNames();
 while (parmNames.hasMoreElements()) {
%>

<%= parmNames.nextElement().toString() %>
<%
 }
%>

Attributes:<%
 attrNames = request.getAttributeNames();
 while (attrNames.hasMoreElements()){
%>

<%= attrNames.nextElement().toString() %>
<%
 }
%>
```



# Working with Style Sheets and Framesets

About this chapter

This chapter describes the Style Sheet editor and the Frameset editor for JSP targets. It does not attempt to teach the use of HTML tagging or Web design.

Contents

<b>Topic</b>	<b>Page</b>
About style sheets	83
About the JSP Target style and style sheet editors	85
Basic editing with the style sheet editors	88
Editing frames and framesets	98

## About style sheets

Style sheet files store common design and layout information separately from the page content of HTML or JSP files. By using style sheets, you can:

- Create a standard design for your HTML pages that can be reused for additional pages as needed.
- Make it easy to change style definitions within a Web site. Editing a style sheet propagates any style change across all of the HTML pages that use that style sheet.
- Separate document design from content development, letting some team members concentrate on the design while other team members develop content.

## Working with styles

You can define and modify style definitions in external style sheets or inside an HTML page. Precedence rules govern how your document appears when the browser finds overlapping style definitions. Styles for the same element might be defined in any of the following (listed inversely to the order of precedence as defined for the Microsoft Internet Explorer browser):

- **External style sheets** store style definitions in separate files external to HTML pages. These files are also known as cascading style sheets because styles can be defined at different levels, with a browser's interpretation of the styles cascading from one level to another.
- **Embedded styles** can be used to create new styles for HTML elements or to modify the appearance of styles from style definitions in an external style sheet. The new or modified styles are included in a `STYLE` element tag that you add to the Head section of an HTML page.
- **Inline styles** can be used to create new styles for HTML elements or to modify the appearance of embedded and external styles. You define inline styles as style attributes of elements on an HTML page.
- **Classes and IDs** can have styles of their own. These styles are linked to particular objects and classes, not to particular elements.
- **Scripts** can modify any style. With recent versions of HTML, any style on a page is considered an object. The Style Sheet editor lets you create style objects associated with IDs. For information about the Script editor, see Chapter 6, "Writing Scripts."

## Syntax for style attributes and selectors

For external style sheets and embedded styles, a selector is the link between an HTML element and a style attribute. The selector specifies what element is to be affected by a declaration for a specific style attribute. The style is that part of the rule that sets forth what the effect will be. In this example, the selector is `H1` and the style is `color:red`:

```
H1 {color: red }
```

When this declaration is included in an embedded style tag on an HTML page or in an external style sheet linked to the page, all `H1` elements on the page appear in red (unless overridden by inline styles or scripts for particular elements).



You can use the JSP Target style sheet editors to assign specific style attributes to HTML elements and selectors through a user-friendly interface. The editors insert the correct syntax for your style definitions onto your HTML page or external style sheet.

## Working with IDs and classes

Classes and IDs, as well as HTML elements, are implemented as selectors in external or embedded style sheets. In the terminology for style sheets, a selector is an element to which a style definition is assigned. Selector elements are *not* enclosed in angle brackets (<>).

If you define classes and IDs in external style sheets, they are available to all of the files that share that style sheet. You can define style characteristics for an ID or class, then assign that ID or class as an attribute to an HTML element.

## About the JSP Target style and style sheet editors

Style sheet editors simplify the process for creating cascading style sheets or embedded or inline styles through the use of tabs and property sheets. You can access the following style and style sheet editors in PowerBuilder:

**Table 4-1: Style sheet editors available for JSP targets**

Style sheet editor	How to open	What to use it for
Global	Select Edit>Global Style Sheet when a JSP page is open in the HTML editor.	Add links to external files or embedded styles to the current HTML page. Can use to create styles for HTML elements, classes, and IDs.
Standalone	Double-click a CSS file in the System Tree or, when a Web page is not open in the HTML editor, drag and drop a CSS file into the editor area. The Cascading Style Sheet wizard also opens this editor.	Create or modify external files. Can use to create styles for HTML elements, classes and IDs. The external file must be linked to a Web page for these styles to be used by the page.

Style sheet editor	How to open	What to use it for
Inline	Click the Inline Styles tab from the property sheet for any HTML element in the current Web page in the HTML editor. Select the Inline radio button and click Edit.	Create or modify styles for the selected HTML element in the current Web page.

Style sheet components

The style sheet editors inside PowerBuilder are composed of the following components:

**Style sheet tree** The left pane of the Global Style Sheet editor provides a list of current embedded styles and links to external style sheets, as well as to elements included in embedded style tags. When you add a new element, or selector, it appears in the left pane under the embedded style to which it was added. The left pane of the standalone Style Sheet editor includes a list of all styles in the open style sheet. Imported style sheets are also displayed in the left pane of these style sheet editors.

**Style sheet tab pages** The right panes of the Global Style Sheet editor and the standalone Style Sheet editor provide a series of tab pages that give you quick access to style attributes. The Inline Styles editor interface is composed entirely of these style sheet tab pages.

The tab pages group similar attributes: font style, margin settings, and so on. When you select attributes for an element, the editor inserts the correct syntax for your style definition.

**Style sheet source** The standalone Style Sheet editor tab pages include a Source tab page. The other tab pages generate style sheet syntax that you can view in the Source tab page. You can also use this page to copy and paste elements between CSS files.

Support for CSS2

The style sheet editors support styles for both formatting and layout as specified in the CSS2 (Cascading Style Sheets, level 2) specification. Current versions of the Internet Explorer and Netscape browsers implement CSS2.

For information about the CSS2 specification, go to the Web site for the World Wide Web Consortium at <http://www.w3.org>.

## The Style Sheet editor tab page interface

All JSP Target style sheet editors provide tab pages that allow you to create and modify style or style sheet attributes easily. The tabs are grayed if a selected element in the style sheet editor or HTML page does not support any of the attributes available on the tab page, or if no element is selected.

The following table lists the attributes available for each tab page.

**Table 4-2: Style attributes available in the Style Sheet editor**

Tab page	Style attribute
Advanced	Aural attributes and the following style attributes: content, counter increment, counter reset, direction, marker offset, quotes, text shadow, and unicode-bidi
Background	Background position, attachment, image, repeat, and color
Border	Border width, color, and style
Font	Font family, size, color, style, variant, weight, stretch, size adjustment, and line height
List	Attributes for list items: image, position, and type
Margin	Margins for all sides of an element
Padding	Padding for all sides of an element
Print	Printing attributes: page size, page break before, page break after, page break inside, marks, orphans, and widows
Source	HTML source code (standalone Style Sheet editor only)
Table	Table attributes: caption side, layout, border collapse, empty cells, speak header, and border spacing
Text	Text alignment, decoration, transform, white space, indent, letter spacing, and word spacing
Visual	Display, position, visibility, clear, z-index, overflow, vertical alignment, clip, and cursor

## Integration with other JSP target editors

The editors used in JSP targets provide integrated support for creating and maintaining style components in your projects.

## **HTML editor**

The HTML editor provides direct access to the style sheet editor. When you are working on a page in the HTML editor, you can open the Global Style Sheet editor to:

- Link the current page to external style sheets
- Embed styles by inserting the STYLE element in the Head section of the HTML page

You can use the Inline Styles editor to add inline styles through the property sheets for individual elements on the current Web page.

The style sheet editors add the appropriate syntax to your file.

## **Script editor**

In the Script editor, you add style objects (typically generic IDs created with the Style Sheet editor) to scripts. Scripts let you produce dynamic style implementations. These scripts can be internal or external to an HTML document.

If you plan to implement selector IDs in scripts, you should be familiar with the naming conventions for these components in the scripting language you use.

## **Basic editing with the style sheet editors**

The style sheet editors and wizard allow you to create external style sheets and link them to your Web pages. You can also create embedded styles and inline styles and make style assignments to HTML elements and selectors using the style sheet editors.

## Creating an external style sheet

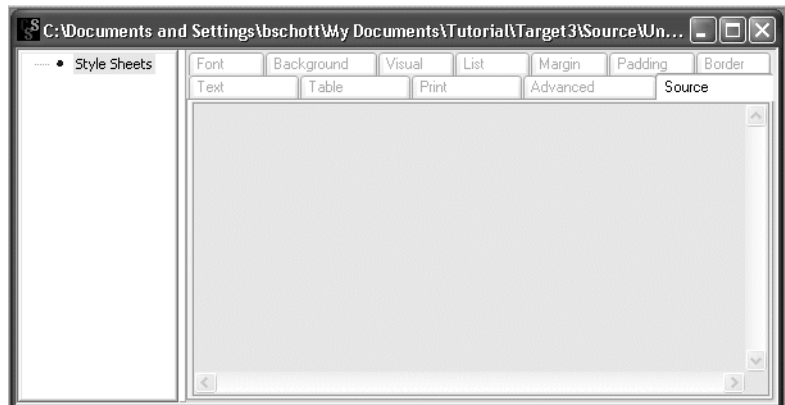
You can use the Cascading Style Sheet wizard to create a new style sheet. You can also use the wizard to link the new style sheet to an existing style sheet.

❖ **To create an external style sheet:**

- 1 Select File>New.
- 2 In the New dialog box, click the JSP tab and double-click the Cascading Style Sheet icon.
- 3 Follow the instructions in the wizard to complete the entries required.

You can specify a name for the style sheet you want to create and, optionally, you can link it to an existing style sheet. When you click Finish, the new style sheet displays in the standalone Style Sheet editor.

No styles or links to existing style sheets are defined in this style sheet:



## Importing an existing style sheet

You can import a style sheet in the Global Style Sheet editor and in the standalone Style Sheet editor. The style sheet editors include a pointer to the imported style sheet using the @import rule.

The @import rule allows you to import style rules from other style sheets. Any @import rules must precede all rule sets in a style sheet. The @import keyword must be followed by the URI of the style sheet you want to include.

### Import rules and client browsers

Not all browsers support @ rules. You should make sure the browsers that will be used to view your Web site support these rules before you link a style sheet through the @import rule.

---

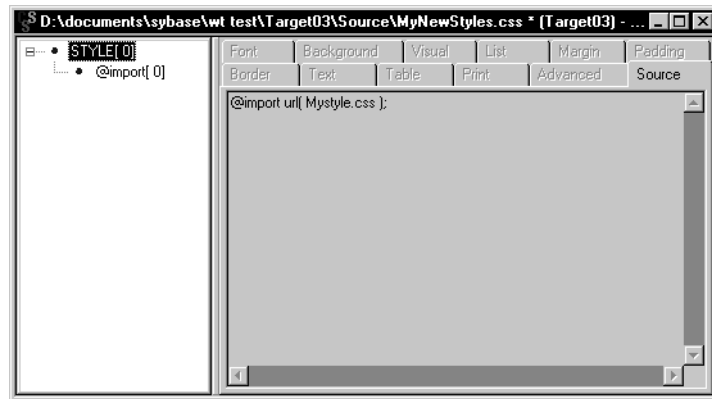
❖ **To import an existing style sheet:**

- 1 Right-click the STYLE[#] item (or the style ID) in the left pane of a style sheet editor.

This is the top item in the standalone Style Sheet editor. This is a second-level item in the Global Style Sheet editor. If you assign an ID to a style sheet in the Global Style Sheet editor, the ID for the style sheet replaces the generic STYLE[#] listing in the left pane of the editor.

- 2 Select Insert@import from the pop-up menu.
- 3 In the Choose URL dialog box, specify the URL of the style sheet that you want to import, save, and click OK.

A pointer is added to the chosen style sheet. You can view the source code for the pointer directly in the Source tab of the standalone Style Sheet editor while the topmost item in the left pane is selected:



To view the source code generated by the Global Style Sheet editor, you must look at the current Web page in the Source view of the HTML editor.

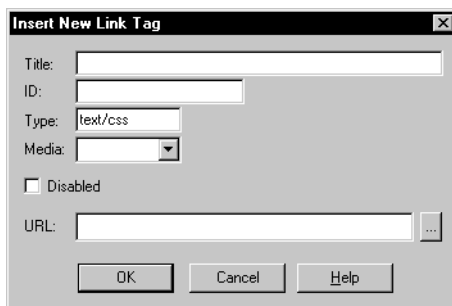
## Linking an external style sheet to an HTML page

After you create style sheets for your project, you can link them to your documents using the Global Style Sheet editor.

❖ **To link an external style sheet to an HTML document:**

- 1 Open the HTML file to which you want to link a style sheet, and select Edit>Global Style Sheet from the HTML editor menu bar.
- 2 Right-click an item in the left pane of the Global Style Sheet editor, and select Insert <LINK> Tag item from the pop-up menu.

The Insert New Link Tag dialog box displays.



- 3 (Optional) Type a title and ID in the appropriate text boxes, select a media type from the Media drop-down list, and select or clear the Disabled check box.
- 4 Click the browse (...) button to select a file or type the name of the external style sheet to be linked to your Web page. Click OK.

When you close the Global Style Sheet editor, the editor inserts the new LINK tag in the Head section of your HTML file with any optional attributes you selected.

## Embedding style definitions in an HTML page

You can use the Global Style Sheet editor to add embedded styles to the current page in the HTML editor.

---

### Embedded styles and client browsers

Older browsers might not recognize the STYLE element. You can surround the style definitions with comment tags (as you would for SCRIPT elements) to direct these browsers to ignore the embedded style definitions. You must add the comment tags (<!-->) directly in the Source view for the HTML page.

---

#### ❖ To add or embed style definitions in an HTML document:

- 1 Open the HTML file in which you want to embed style definitions and select Edit>Global Style Sheet from the HTML editor menu bar.
- 2 Right-click an item in the left pane of the Global Style Sheet editor, and select Insert <STYLE> Tag from the pop-up menu.
- 3 (Optional) In the Insert New Style Tag dialog box, specify a title and ID in the appropriate text boxes, select a media type from the Media drop-down list, and select or clear the Disabled check box.
- 4 In the left pane, right-click the newly specified style or another style that you want to edit. Then define styles and click OK.

When you close the Global Style Sheet editor, the editor inserts the new STYLE element in the Head section of your HTML file, along with any selector styles for HTML elements, classes, and IDs that you add.

Separate STYLE tags are generated in the HTML page for each STYLE element you add to the left pane of the Global Style Sheet editor. Selector styles are added to the HTML page only between STYLE tags that correspond to the STYLE element under which they appear in the left pane of the style sheet editor.

## Opening an existing style sheet

You open an existing style sheet in the standalone Style Sheet editor. Although you can import a style sheet and modify embedded styles with the Global Style Sheet editor, you can open or edit an external style sheet only by using the standalone editor.



**❖ To open an existing style sheet for editing:**

- Right-click the file in the System Tree and select Edit from the pop-up menu

*or*

Drag an existing CSS file from the System Tree to the editor area for PowerBuilder.

If you drag a CSS file while an HTML page is open in the HTML editor, the file does not open in the Style Sheet editor, but becomes the target of a hyperlink instead.

For information on linking style sheets to HTML pages, see “Linking an external style sheet to an HTML page” on page 91.

**❖ To open a file that is not part of your JSP target:**

- From a drag-and-drop file viewer, such as Microsoft Windows Explorer, drag the file into the JSP Target workspace

*or*

Select File>Open from the PowerBuilder menu bar and browse to find the file in the Open dialog.

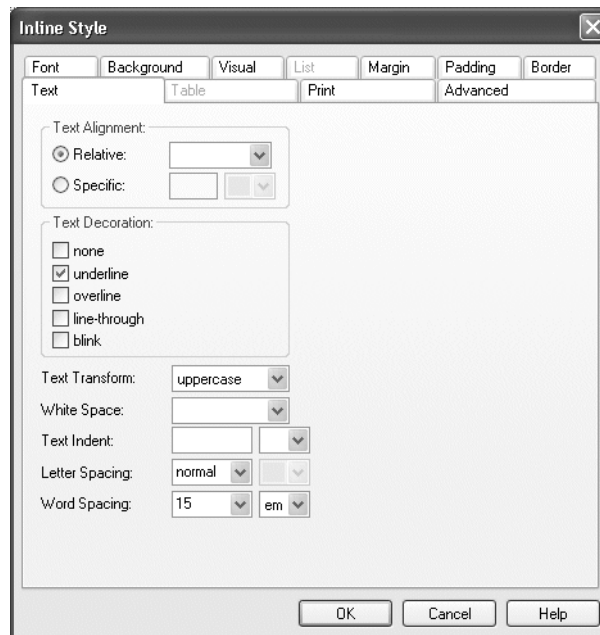
## Using the Inline Style editor

You open the Inline Style editor from the properties dialog box for HTML elements for the current page in the HTML editor. You can set inline style for an element when you drag that element from the System Tree to an HTML page.

**❖ Using the inline style editor to modify HTML tag elements:**

- 1 From the Page view or Source view of the HTML editor, right-click the desired HTML element or control and select Properties from the pop-up menu.
- 2 Select the Inline Style tab of the properties dialog box, select the Inline radio button, and click Edit.
- 3 On the tab pages of the Inline Style editor, specify the type of style you want to add by selecting or typing values for the style attributes.

Some of the tab pages or items on the tab pages might be grayed if they are inappropriate for the selected element. This is the Inline Style editor for a button control:



You can view the generated inline styles in the Style Attributes And Values list box on the Inline Styles page of the element property sheet, and in the Source view of the HTML editor.

## Adding selectors for HTML elements, classes, and IDs

You can define styles for HTML tags, classes, and IDs in the Global Style Sheet editor or in the standalone Style Sheet editor.

### Global Style Sheet editor restrictions

In the Global Style Sheet editor, you must right-click an embedded STYLE element or an item at a level below an embedded STYLE element. If you right-click a LINK element or the topmost StyleSheets item, the Insert HTML Tag Selector menu item is grayed.

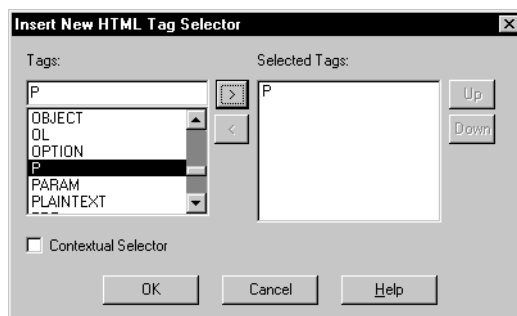
You can right-click any item in the left pane of the standalone Style Sheet editor to add selectors to an external style sheet.

## Defining styles for HTML element selectors

### ❖ To define styles for HTML element selectors:

- 1 Right-click an item in the left pane of a style sheet editor, then select Insert HTML Tag Selector from the pop-up menu. (See “Global Style Sheet editor restrictions” on page 94.)
- 2 In the Insert New HTML Tag Selector dialog box, double-click the tag for which you want to define a style  
*or*  
Select the tag to be added to your style sheet and click the > button.

The selected tag is copied into the Selected Tag list box:



- 3 (Optional) Select the Contextual Selector check box and add another tag that you nest inside the first tag.

The styles you select are applied to the nested element only when it is nested below the element at the top of the Selected Tag list box. You can change positions of nested elements by selecting one of the elements in the list box and clicking the Up or Down buttons.

- 4 Click OK and select the new element in the left pane of the style sheet editor.
- 5 On the tab pages in the right pane of the style sheet editor, specify the type of style you want to add for the element by selecting or typing values for the style attributes.
- 6 After you finish setting the style definitions, you can insert another element in the external or embedded style sheet.

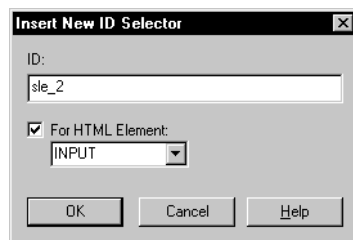
## Defining styles for ID Selectors

ID selectors define style rules for an individual element. You can create an ID in the context of an element, or as a generic identifier. However, even as a generic identifier, an ID should be assigned to only one element in a document. IDs can be used within HTML elements or in scripts.

JSP Target style sheet editors use generic IDs as style objects. Several scripting languages, such as JavaScript and VBScript, can manipulate these objects to dynamically change the appearance of elements associated with the ID.

### ❖ To define styles for ID selectors:

- 1 Right-click an item in the left pane of a style sheet editor, then select Insert ID Selector from the pop-up menu. (See “Global Style Sheet editor restrictions” on page 94.)
- 2 In the Insert New ID Selector dialog box, identify the new ID for the style sheet.
- 3 To create a generic ID, click OK  
*or*  
To assign the ID to an HTML tag, select the For HTML Element check box, then select an element and click OK.



- 4 With the new ID selected in the left pane, define styles for the ID in the tab pages in the right pane.

## Defining styles for class selectors

Class selectors define style rules in the context of a specific HTML element or as a generic component. When applied to a specific element, the class is available only with the associated tag. Generic classes are available as attributes for any tag.

For example, to apply a class at the tag level, you can define a tag as:

```
H1.NewStyle {font-family:arial; font_color:navy;}
```

In an HTML document, the tag would be referenced as H1.NewStyle. The NewStyle class is available only to H1 tags.

As a generic class, you could define the NewStyle class this way:

```
NewStyle {font-family:arial; font_color:navy;}
```

To implement this style in an H1 tag, you would use this syntax:

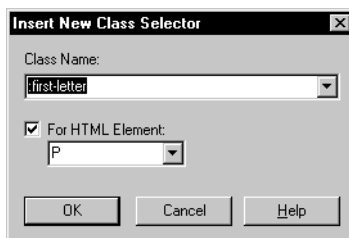
```
<H1 CLASS="NewStyle">
```

The Insert New Class Selector dialog includes the following well-known classes (also called pseudo-classes):

:active	:focus
:after	:hover
:before	A:active
:first-child	A:hover
:first-letter	A:link
:first-line	A:visited

❖ **To define styles for classes:**

- 1 Right-click an item in the left pane of a style sheet editor, then select Insert Class Selector from the pop-up menu. (See “Global Style Sheet editor restrictions” on page 94.)
- 2 Select or type a class name in the Insert New Class Selector dialog box.  
You can define styles for a new class or for existing classes.
- 3 Click OK to create a generic class  
*or*  
To assign the class to an HTML tag, select the For HTML Element check box. Then select an element and click OK.



- 4 With the new class selected in the left pane, define styles for the class in the tab pages in the right pane.

## Removing items from a style sheet

You can remove style selectors, embedded styles, and linked style sheets in the Global Style Sheet editor, and you can remove imported style sheets and style selectors in the standalone Style Sheet editor.

---

### Removing styles using the Inline Styles editor

To remove styles using the Inline Styles editor, you can either set each style attribute to a null value or remove the attributes directly in the Source view of the HTML editor.

---

❖ **To remove items from a style sheet:**

- 1 In the left pane of the style sheet editor, right-click the item you want to remove.
- 2 From the pop-up menu, select Delete.

## Editing frames and framesets

A Frameset document is an HTML page with preset frame divisions. These frames can be used to display the content of other HTML pages. When you open a Frameset file in PowerBuilder, the file displays in the Frameset editor.

### About the Frameset editor

There are four frameset views in the Frameset editor: Frames, Source, Preview, and No Frames. You can make modifications to the frames or frameset in the Frames, Source, and No Frames views. No Frames view displays the page as seen with a Web browser that does not support frames.

#### Frames view

Frames view displays the frames you have defined and the contents of the pages. Frames view lets you drag and drop Web pages from your JSP target to frame panes in your frameset. You can also modify the size of the frames by stretching the frame panes. Data cannot be placed directly in the frame panes.

- Source view** Source view lets you edit in the source file. As with Source view in the rest of the JSP target environment, Source view provides the most flexible editing environment. In Source view you can drag and drop items from the component or language tab in the System Tree area.
- Preview view** Preview is a display-only view. It shows what the page would look like when viewed with a Frame-enabled Web browser.
- No Frames view** The No Frames view displays the page as it is seen with a Web browser that does not support frames. No Frames view lets you add text to the page that is not seen when you switch back to the Frames view.

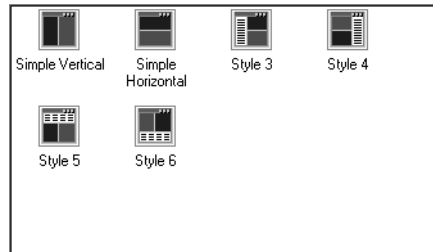


## Creating a new frameset document

You can use the Frameset wizard to create a new frameset document.

- ❖ **To create a new frameset document:**
  - 1 Right-click a target and select New from the pop-up menu  
or  
Select File>New from the menu bar.

- 2 In the New dialog, click the Files tab, then double-click the Frameset Page icon.
- 3 In the Frameset wizard, specify file information for the frameset document and click Next.
- 4 Select one of the six layout choices and click Next.



If you want to specify a different frameset pattern with more than three frame panes, you must do that later from the Source view of the Frameset editor, or by using the Split Horizontally and Split Vertically commands from the pop-up menu on frames in the Frames view.

- 5 Click Finish.

When you complete the entries in the wizard, the editor displays the frame structure in the Frames view of the Frameset editor.

## Modifying a frameset

Formatting a frameset page

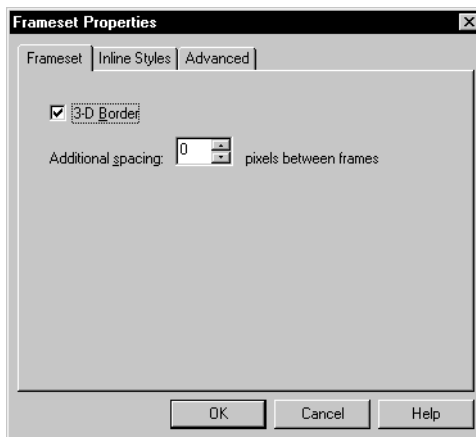
The Frameset editor lets you modify the look of your frame pages and allows you to make connections to other Web pages.

❖ **To modify a frameset page format:**

- 1 Open the frameset file in the Frameset editor.



- 2 In Frames view, right-click a Frame pane and choose Frameset Properties from the pop-up menu  
*or*  
In Source view, right-click a Frameset tag and choose Properties from the pop-up menu.



You can use the Frameset page of the dialog box to add or remove a border between frames of the frameset and to set the spacing in pixels between the frames. On the Advanced page of this dialog box, you can add or modify frameset attributes, such as percentage of the page for frames in ROWS and COLS.

For information on the Inline Styles page, see “Using the Inline Style editor” on page 93.

## Modifying frame properties

From the Frame Properties dialog box, you can modify the properties of frames and change or select the URL of a file to be displayed in a frame. You can modify the sizes, borders, fonts, and other related format items of the individual frame elements within a frameset.

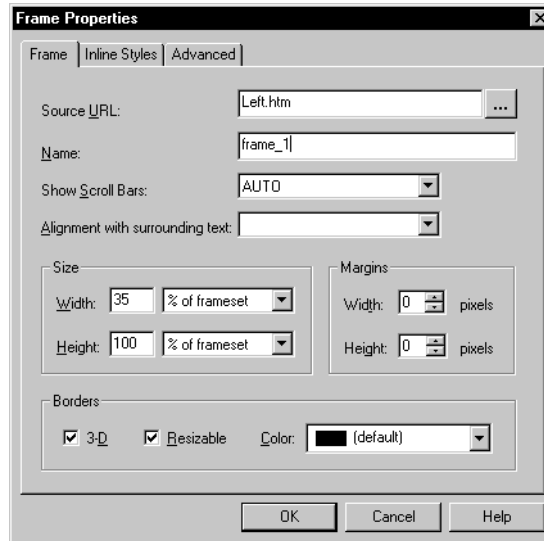
❖ **To modify frame properties:**

- 1 Open the frameset file in the Frameset editor.

- 2 In Frames view, right-click a Frame pane and choose Properties from the pop-up menu

*or*

In Source view, right-click a Frame tag and choose Properties from the pop-up menu.



# Working with Images, Other Media, and Components

## About this chapter

This chapter describes how to add images, sound, video, and other components to your Web pages using the System Tree and JSP Target editors and toolbars.

## Contents

<b>Topic</b>	<b>Topic</b>
Images and image maps	103
Multimedia	107
Components	108
The Java class path	112
The custom tag library search path	113

## Images and image maps

You can use your favorite image editing tool to create image files and add them to your HTML page or JSP file. You can use any image format supported by the major browser vendors.

## Inserting images

You can insert images in both Page view and Source view of the HTML Editor. You can define attributes for the inserted image in the Image Properties dialog box.

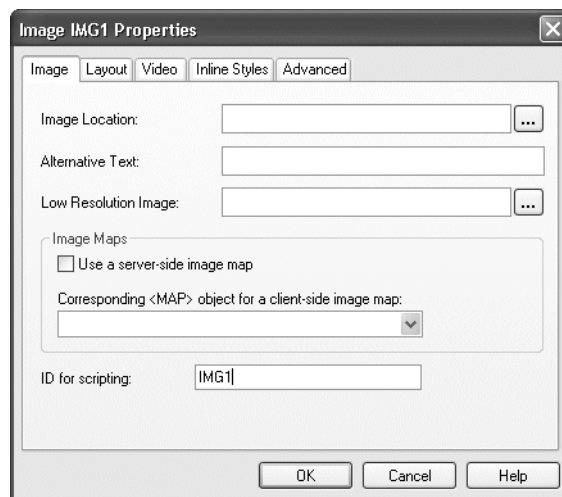
**Table 5-1: Attributes you can set in the Image Properties dialog box**

Image properties page	IMG element attributes definable through user interface
Image	SRC, ALT, LOWSRC, ISMAP, USEMAP, ID
Layout	ALIGN, WIDTH, HEIGHT, HSPACE, VSPACE, BORDER
Video	DYNSRC, CONTROLS, START, LOOP
Inline Styles	All inline style attributes (see Chapter 4, “Working with Style Sheets and Framesets”)
Advanced	All attributes and attribute values you want to set for the image object

❖ **To insert an image:**

- 1 Put the insertion point where you want to insert the image.
- 2 Select Insert>Image from the menu bar.

The Image Properties dialog box displays.



- 3 Click the Browse button (...) to select an image file using the URL Picker *or* Type the URL for the image in the Image Location text box.
- 4 Set other image attributes as needed, and click OK.

## Setting height and width attributes for images

After you insert an image in Page view, the editor displays the dimensions of the image in the Properties dialog box. However, these are default values and are not automatically included in the HTML source.

If you resize the image or explicitly set different values, the HEIGHT and WIDTH attributes and their values will be included. Use the following steps if you want the *default* values included in the HTML source.

❖ **To have HEIGHT and WIDTH attributes added to the HTML source:**

- 1 In Page view, right-click the image and select Properties from the pop-up menu.
- 2 On the Layout tab, change the Height and Width values, or retype them, and click Apply.

Source view will now show HEIGHT and WIDTH in the IMG element.

---

**If you know the image dimensions**

If you know the image dimensions, work in Source view and type the HEIGHT and WIDTH attributes in the IMG element. For example:

```

```

---

## Converting the image to a hyperlink

❖ **To make the image a hyperlink:**

- 1 Select the image:
  - In Source view, select the entire IMG tag.
  - In Page view, drag the cursor over the image or use the Shift and arrow keys to highlight the entire image. (Do not click on the image so that the resizable selection border displays.)
- 2 Select Format>Hyperlink from the menu bar, and create the link.

For information on formatting hyperlinks, see Chapter 3, “Working with the HTML Editor.”

## Creating image maps

An image map is an image that links to different files or URLs depending on the area of the image clicked by the user. Not all browsers support client-side image maps, but you can still make this work on the server side.

You can set up the same image to be a client-side and server-side image map. The client-side processing takes priority in browsers that support it.

### Creating a client-side image map

This is how you set up a client-side image map.

❖ **To create a client-side image map:**

- 1 In Page view or Source view of the HTML editor, select Insert>Image to insert the image in your document.
- 2 On the Image tab of the Image Properties dialog box, type a pound sign (#) followed by the name of a MAP element (which you will create next) in the text box for the Corresponding <MAP> Object For A Client-Side Image Map, and then click OK.

In the Source view for the current HTML document, the IMG element includes the USEMAP attribute that is assigned to the map name you entered:

```

```

- 3 In Source view, insert a MAP element in the document and assign it a NAME attribute that matches the name you typed for the USEMAP attribute:

```
<MAP NAME="mymap" ></MAP>
```

You can add this to the document source before or after the IMG element, or in the Head section.

- 4 Inside the MAP element, add the AREA elements required to identify different regions of the image. A basic syntax is shown here:

```
<AREA SHAPE="shape" COORDS="x1, y1, x2, y2. . ."
 HREF="url" >
```

## Creating a server-side image map

This is how you set up a server-side image map.

❖ **To create a server-side image map:**

- 1 In Page view or Source view of the HTML editor, select Insert>Image to insert the image into your document.
- 2 On the Image tab of the Image Properties dialog box, select the Use A Server-Side Image Map check box and click OK.

This adds the ISMAP attribute to the IMG element in the source for the current Web page.

- 3 Select the image and use Format>Hyperlink to add a hyperlink.

The Hyperlink Properties dialog box opens to the Hyperlink page. For more information, see “Converting the image to a hyperlink” on page 105.

- 4 In the Destination text box on the Hyperlink page, specify a URL that points to the server program that processes the image map and ends with a path to the map file.
- 5 Install a map file on the server that describes the shapes within the image and their URLs.

The format of the map file depends on your server.

## Multimedia

In addition to attributes for managing single images, the IMG element includes advanced attributes for video images. You can create or modify video attributes settings on the Video tab of the Image Properties dialog box. These video settings are not available in all browsers.

Elements for other media and effects are best entered directly in Source view. You can use the System Tree to view a list of available attributes for these elements. Many of the less frequently used elements do not have a custom properties dialog box, just an Advanced tab in which you can enter the attributes and the attribute values you want to set.

Examples of HTML elements for other media and effects include BGSOUND and MARQUEE. Components such as applets and ActiveX enable customized effects and user interaction.

## Components

The Components page of the System Tree lists the components that are installed on your PC. There are several categories:

- **ActiveX Controls** The registered components are self-categorizing. The categories you see depend on what is installed on your system. The categories of greatest interest are:
  - Web design-time control (DTC)
  - Controls that are safely scriptable  
If a control does not identify itself as safely scriptable, this does not mean that it is unsafe.
  - Controls (all registered controls)
- **Plugins** The plug-ins installed in the Netscape and Internet Explorer Plugins directories.
- **Java Applets and Java Beans** The applets and JavaBeans that are in the class path.
- **EAServer Servers** Lists the servers for which you have defined EAServer profiles. You can see the packages, components, and component methods available on accessible servers. You can insert the components on 4GL JSP pages only.
- **Custom Tag Libraries** Lists the custom tag libraries in the paths you specify on the JSP page of the System Options dialog box. To use the custom tag libraries, you must make sure that the classes of the libraries are available to the server where you deploy your JSP target. You can insert custom tag libraries on JSP target pages only.

Right-clicking anywhere in the Components page produces a pop-up menu with access to the System Options and the EAServer Profiles dialog box.

In Page view, components display as they would appear on the page, but they do not execute. To interact with components for testing, use Preview view or open a browser window.



## Viewing available components

To use the tools for inserting components, you must have the components installed and available on your system. If they are not, you have to know how to fill in the OBJECT, EMBED, and APPLET property dialogs with the correct values.

❖ **To view the components available on your system:**

- 1 In the System Tree, select the Components tab.
- 2 Expand the branches for the different types of components.

The Java branches might be slow to expand because every Java file in the path must be examined to determine its type.

## Inserting a component

You can insert an ActiveX control, applet, plug-in, JavaBean, EAServer component, or custom tag library by dragging it from the System Tree to the current page in the HTML editor.

For information on inserting EAServer components, see “Integrating with EAServer” on page 150 and “Accessing EAServer components” on page 174.

If an ActiveX component has not identified itself as safely scriptable, the editor displays a warning. To interact with the component and view its custom property pages, you must allow it to initialize and run scripts.

---

### Disabling the warning

To disable the warning (which comes from Internet Explorer), start the Internet control panel from your Windows Start menu. Then, using Custom, change the settings for Initialize And Script ActiveX Controls Not Marked As Safe. If you make this change, be aware that your system is more vulnerable when you browse the Web with Internet Explorer.

---

❖ **To insert a component into a Web page:**

- 1 Drag the component from the System Tree to the current page  
*or*  
Set the insertion point in the current page and select Insert>Component>*Component Type* from the menu bar.

You can drag and drop EAServer components to a 4GL JSP page and custom tag libraries to JSPs, but you cannot use the Insert menu to add EAServer components or custom tag libraries to your page.

- 2 If a warning about initializing and scripts displays, click Yes.

PowerBuilder displays the properties dialog box for the OBJECT, EMBED, or APPLET element.

An applet inserted from the Insert menu or toolbar might not display immediately on exiting the properties dialog box. You can force it to display by making any small change to the file in Source view.

- 3 Set properties as needed, especially the Name (For Forms Or Scripting) at the bottom of the *ComponentType* Properties dialog box.

It is important to supply valid values for parameters on the component's custom property dialog. Parameters can have invalid values because:

- A value was not specified on the component's property sheet and the component did not supply a valid default value
- An invalid value was specified on the property sheet and the component did not do appropriate error checking
- The component expected a value but the property sheet had not yet been displayed

These problems occur most often with applets but can occur with other components too.

- 4 For an ActiveX control, click the Control Properties button to display and edit the control's custom property dialog box.

---

### **Control Properties button**

This button is enabled only when you are working in Page view.

---

After you close the property dialog boxes, an OBJECT element is inserted in the document and the control displays in Page view.

## Design-time controls

Design-time controls (DTCs) are ActiveX controls that write HTML into your document while you edit. They provide custom property pages where you can specify options that affect the HTML.

For information on using the Sybase Web DataWindow DTC, see Chapter 11, “Using the Web DataWindow Design-Time Control.”

## Viewing and editing DTC properties

In the editor, you can view the control's custom property pages and make changes. When you click OK, the HTML for the control is regenerated.

- ❖ **To view or make changes in the control's custom property dialog box:**
  - 1 In Source view, right-click the DTC's METADATA tag, the subsequent OBJECT tag, or generated code, and then select Properties from the pop-up menu.
  - 2 In Page view, right-click the object and select the custom menu item for the control's properties.

## Working with the generated HTML

Normally you should not modify any of the HTML generated for a DTC because the changes you make will be lost the next time you modify the control properties and regenerate the output.

It is possible to insert the DTC-generated HTML without the control itself. If you choose Insert>Component>ActiveX from the HTML editor's menu bar and check the Generate Static Output property (on the Design-time tab), you get the HTML produced by the DTC but not the METADATA comments or OBJECT element. You can then modify this static output as you like without any worry of accidentally re-executing the DTC and overwriting your modifications.

## The Java class path

The Java class path is a list of directories and archive files that specify where to look for applets, JavaBeans, and other files containing Java code.

PowerBuilder uses the class path to find the Java classes it lists in the System Tree.

PowerBuilder searches the directories in the class path and their subdirectories. It also searches the contents of the ZIP and JAR archive files listed in the class path. It determines whether any of the files are Java CLASS files and whether they are applets or JavaBeans. If it finds more than one CLASS file with the same name, only the first one appears in the System Tree.

## Class path values

When you use JSP targets, there are two possible values for the class path:

- **System default class path** This value is stored in the CLASSPATH environment variable.
- **System options class path** The Java page of the System Options dialog box lets you list directories and archive files to be searched in addition to those listed in the system default class path (without changing the system default value).

To go to the Java page of the System Options dialog, you can either:

- Select Tools>System Options from the menu bar and click the Java tab
- Right-click anywhere in the Components tab of the System Tree and select Java Classpath from the pop-up menu

If the Java VM has already started, you can see the effects of your changes by restarting PowerBuilder and expanding a Java branch on the Components tab of the System Tree.

## Using the class path

The class path value in use during your JSP target session depends on how the Java VM starts. PowerBuilder starts the Java VM in these situations:

- The first time you look at a page that contains a Java applet, the Java VM starts automatically using the default system class path. The system options class path is not used.
- When you expand a Java branch on the Components tab of the System Tree, the Java VM starts using a combination of the system options and system default class paths.

Once the Java VM is started, it remains running until you exit your workspace.

When you include a Java applet in your document, the Applet element provides a CODEBASE attribute that specifies where to find the applet on the server. The class path setting on the user's computer is irrelevant.

---

### Viewing results of class path changes in the System Tree

If you use the Java tab of the System Options dialog box to change the class path after the Java VM is started in your current session, you must restart PowerBuilder to see the result. Even then, your changes take effect only after the Java VM starts as a result of actions in the System Tree.

---

## The custom tag library search path

Custom tags in a JSP file perform actions defined in a custom tag library. PowerBuilder supports custom tag libraries that use the JSP 1.2 format. On the JSP page of the System Options dialog box, you can indicate paths that you want PowerBuilder to search to add tag library descriptor (TLD) files to the Components tab of the System Tree. You do not need to restart PowerBuilder for any changes you make to the custom tag library path to take effect.

To go to the JSP page of the System Options dialog, you can either:

- Select Tools>System Options from the menu bar and click the JSP tab.
- Right-click anywhere in the Components tab of the System Tree and select Custom Tag Libraries Search Path from the pop-up menu.

To add a tag library to a JSP page, you can insert a taglib page directive or you can drag a tag library from the Components tab of the System Tree to a target page in the HTML editor.

When you add a custom tag library to a page, you need to specify a prefix to identify the custom tag as well as the location where the TLD file can be found relative to the root of the Web application. The folder that contains the deployed Web application has a *WEB-INF* subdirectory. Typically the TLD files are deployed to the *tlds* subdirectory of the *WEB-INF* directory. You must also remember to make your custom tag library classes available on the class path of the application server.

For more information about custom tag libraries, see “Custom tags” on page 79. For information about the Custom Tag Library page of the Deployment Configuration dialog box, see “Tag Libraries” on page 245.

# Writing Scripts

About this chapter

This chapter describes how to include scripts with your JSP targets and how to edit them.

Contents

Topic	Page
About scripts	115
JSP scripting elements	121
Procedures for editing scripts	124
Techniques and tips for creating scripts	131

## About scripts

Scripts for a Web site include event handlers for HTML objects, client-side scripts associated with a document, and server-side scripts run before a document is downloaded to a browser. Client-side scripts can be written in several languages. Server-side scripts must be written in Java. Which languages you use depends on the browsers you want to support or the application server your site uses.

Code snippets can be saved in the Clip window and dropped into client- or server-side scripts as needed. Scripts can be saved as part of an HTML or JSP document, or in a separate file (one script per file).

## Editing scripts

The HTML editor and the Script editor provide a flexible approach to writing scripts. You can work in the Script editor that is integrated with the HTML editor or in the standalone Web Script editor.

## Working in the HTML editor

The integrated Script editor is a pane in Page view of the HTML editor. It provides an organized view of your scripts, supports color-coding based on the selected scripting language, and provides facilities for saving and testing scripts. When you add a script in Page view, the editor automatically inserts appropriate HTML elements and attributes for the script into your document.

An HTML page can contain many scripts. The integrated Script editor lets you focus on one script at a time. Using the Script toolbar, from the three drop-down lists select an object, an event, and a scripting language. If an object you select is a script, the event drop-down is blank.

In Source view, you can edit the HTML script elements and the script itself in a single window. The scripts are intermixed with the rest of the document content.

## Working in the standalone Web Script editor

The standalone Web Script editor is a separate window, independent of the HTML editor. Scripts can be created here and saved to a file, becoming independent code you can reuse in many documents and projects.

You can open the Web Script editor by selecting the Script wizard in the New dialog box or by dragging and dropping a script file to the PowerBuilder editor area. The Web Script editor supports the following extensions: *ASP*, *JS*, *PSS*, *SSC*, *SSS*, *STS*, and *VBS*. However, the editor allows you to save a script file with any extension you want.

## Using the Clip window

The PowerBuilder Clip Window button opens a window in which you can store bits of code you use frequently. You can copy text to the Clip window to be saved and then drag or copy this text to the Script editor when you want to use it.

The Clip window displays a list of named clips and a preview of the information contained in each. It provides buttons to move Clip window contents to the clipboard, copy clipboard contents to the Clip window, rename a clip, and delete a clip. Clips you save in one workspace are available in all your workspaces. You can hide or display the Clip window by using the Clip Window button on the PowerBar or selecting Window>Clip.



## Where and when to save scripts

By default, the scripts you write in the integrated Script editor are stored in the JSP page or HTML document. They are saved when you save the document. If you use the standalone Web Script editor to write a script, you can incorporate the script into a document by pointing to the file or copying the script into the document.

When you save a client script in an external file from the integrated Script editor, the editor inserts an SRC attribute to point to the file. When you edit the script, the editor gets the script from the file and saves it again when you save the document.

## Scripting languages

JSP targets support several client scripting languages, including JavaScript and VBScript. For server scripting, JSP targets support only Java.

For more information about working with application servers, see Chapter 7, “Working with Application Servers and Transaction Servers.”

**Choosing a scripting language** When you write a script, you specify which scripting language you are using. The editor uses this information to:

- Add a LANGUAGE attribute to the SCRIPT element
- Recognize the language syntax for color coding

## Types of scripts

There are several ways to write scripts in an HTML page or JSP. The following examples show the HTML elements for each type of script as you would see the script in Source view. In Page view, you do not see this syntax.

### Inline event handlers

The code for an inline script is included in the start tag of an HTML element. It is assigned to a property associated with an event.

```
<BODY LANGUAGE=JavaScript
onload='alert ("Confidential!"); '>
```

---

### Script for objects that are not HTML elements

If an object such as a document has events but is not an HTML element, the script is contained in a `SCRIPT` element and uses the `FOR` property to associate itself with the object and the `EVENT` property to associate itself with an event:

```
<SCRIPT LANGUAGE=JavaScript
FOR="document" EVENT="afterupdate">
alert("Confidential!");
</SCRIPT>
```

---

In the integrated Script editor, use the Script toolbar to select an object and an event. The script is stored in the property associated with the event.

## Client scripts

Client scripts are not necessarily associated with an event. In HTML or JSP documents, surround these scripts with `SCRIPT` tags. (The integrated Script editor can do this for you.) The scripts are evaluated as they are loaded with the Web page. They can include functions that can be called by other scripts.

```
<SCRIPT Language="JavaScript">
function navigate(myform) {
 durl= (myform.mylist.options
 [myform.mylist.selectedIndex].value);
 location.href=durl;
}
</SCRIPT>
```

In the integrated Script editor, use the pop-up menu to create a new client script. Client scripts are part of an array of Script objects.

## Server scripts

You can write scripts that are run on the server before the document is sent to a Web browser.

To create a server script, use the pop-up menu to create a new server script. The HTML that marks the script varies depending on the server you choose.

For information on how to insert a server script in your page, see “Creating a new script” on page 125. For more information about working with application servers, see Chapter 7, “Working with Application Servers and Transaction Servers.”

## Scripts in external files

You can store any script in a separate file. The `SCRIPT` element has an `SRC` attribute to refer to the file. For JavaScript, the file usually has a *JS* extension.

```
<SCRIPT Language="JavaScript" SRC="script1.js">
</SCRIPT>
```

There are two ways to create an external script file:

- In the integrated Script editor, the Save As External File pop-up menu item stores the current client script in a file and points to that file with the `SRC` attribute.
- The standalone Web Script editor creates a script with no specific association to an HTML document. To use the script in an HTML document, switch to Source view to edit the `SRC` attribute to point to the standalone script file.

## Other scripts

**Read-only scripts** Scripts that have been automatically generated are marked as read-only. These scripts are usually associated with design-time controls (DTCs) and are part of the `METADATA` information for the control. You can view them but not change them in the Script editor.

If you change the script in Source view and later cause the DTC to regenerate its output, the script is rewritten and your changes are lost.

---

### Static output for design-time controls

In some cases, you might want to insert generated scripts for a DTC without the control itself. If you choose `Insert>Component>ActiveX` from the menu bar and select the Generate Static Output check box (on the Design-time tab), you will get the scripts produced by the DTC but not the `METADATA` comments or `<OBJECT>` element. You can then modify this static output without accidentally re-executing the DTC and overwriting your modifications.

---

## Objects in an HTML document

An HTML document is made up of objects organized in a hierarchy. Initial releases of JavaScript used a small set of objects. With the latest updates to HTML, any component of the page—a paragraph, a link, a table cell—is an object.

### IDs for HTML objects

To help identify objects when scripting, you can give them names or IDs. For example, instead of referring to each member of the array of link objects (anchor elements) on a page by using the default document object identifier (document.A with a numeric index), you could name the links to identify them at a glance:

```
<A ID=beginnerguidelink
 HREF="http://www.finance.com/info/index.html">
 Beginner's Guide to Finance
```

**Assigning an ID** You can assign an ID in the HTML editor using the HTML element's properties dialog box. You can also type the ID attribute into the document directly in Source view.

### HTML objects in the Script toolbar

All commonly scripted objects are included in the integrated Script editor's Object drop-down list. To identify each object in the Object list box, a JSP target uses the following rules of precedence:

- If the object *has a name*, the name is used as the identifier.
- If the object *has an ID but no name*, the ID is used as the identifier.
- If the object *does not have a name or an ID*, it is identified by its array index. (For example, the first image on a page would have the identifier IMG[0].)

**Radio buttons** Radio buttons in a form are handled differently. Because radio buttons are linked together by assigning them the same name, the identifier for a radio button is its ID. If no ID is given, the ID is the radio button array index.

**Client and server scripts** Client scripts and server scripts are identified by their array indexes. For example, the first client script on a page would have the identifier SCRIPT[0] and the first server script would have the identifier ServerScript[0].

**About array indexes** Array indexes for items without an ID, including scripts, correspond to the items' positions in the document. If you insert an item earlier in the document or if you move items around, the index value associated with the item will change and any scripts for the item will now be associated with the new index value.

For example, if you create three client scripts and then move the third script, `SCRIPT[2]`, to the Head section, that script becomes `SCRIPT[0]`, `SCRIPT[0]` becomes `SCRIPT[1]`, and `SCRIPT[1]` becomes `SCRIPT[2]`. If you change the order in Page view, you might need to open the Source view to force the editor to update the array index numbers.

## JSP scripting elements

Scripting elements manipulate objects and perform computations. The character sequence that precedes a scripting element depends on the element's type: `<%` for a scriptlet, `<%=` for an expression, and `<%!` for a declaration. Scriptlets, expressions, declarations, and server-side comments are all closed with the sequence `%>`.

### Scriptlets

A scriptlet contains a code fragment valid in the page-scripting language (usually Java, but other languages can be defined in the page directive):

```
<% cart.processRequest(request); %>
```

### Expressions

An expression contains an expression valid in the page-scripting language:

```
Value="<%= request.getParameter("amount") %>"
```

### Declarations

A declaration declares variables or methods valid in the page-scripting language:

```
<%! Connection myconnection; String mystring; %>
```

### Comments

You can add two types of comments to a JSP file:

- HTML comments optionally contain an expression. They are sent to the client and can be viewed in the page source:

```
<!-- Copyright (C) 2002 Acme Software -->
```

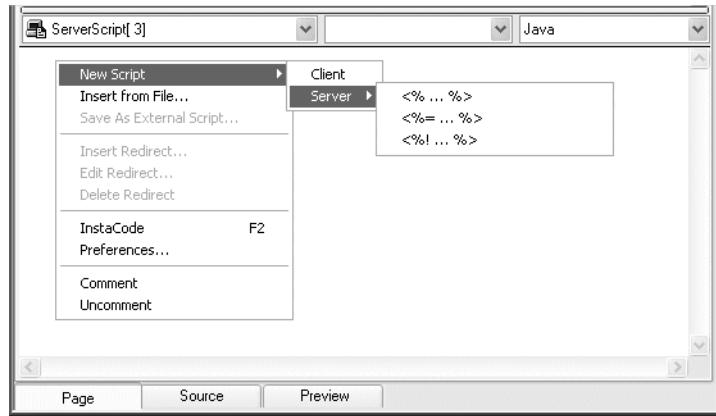
- Hidden comments document the source file and are not sent to the client:

```
<%-- Add new module here --%>
```

To insert a comment, type it in the Source view.

## Inserting a scripting element

- ❖ **To insert a scripting element in a JSP page:**
  - 1 Open a JSP page, select the Page tab, and right-click in the Script editor.
  - 2 From the pop-up menu, select New Script>Server>JSP and then the delimiters for the type of scripting element you want.



- 3 Type the script, expression, or declaration in the Script editor.

## Implicit objects

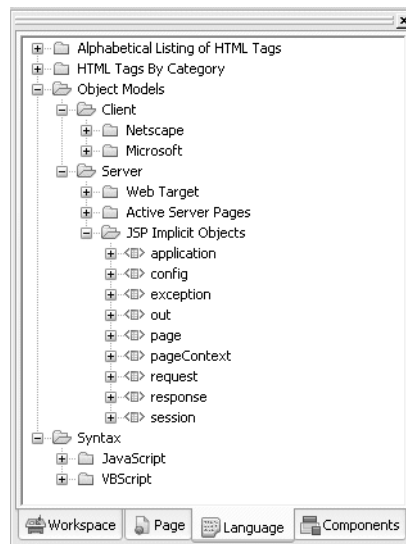
When a JSP page processes a request, it has access to a set of implicit objects, each of which is associated with a given scope. Other objects can be created in scripts. These created objects have a scope attribute that defines where the reference to that object is created and removed.

References to an object created in script are stored in the pageContext, request, session, or application implicit object, according to the object's scope.

**Table 6-1: Implicit objects for a JSP target**

Implicit object	Description	Scope
request	The request triggering the servlet invocation.	Request
response	The response to the request that triggered the servlet invocation.	Page
pageContext	The page context for this JSP.	Page
session	The session object created for the requesting client (if any).	Session
application	The servlet context obtained from the servlet configuration, as in the call <code>getServletConfig().getContext()</code> .	Application
out	An object that writes to the output stream.	Page
config	The <code>ServletConfig</code> instance for this JSP.	Page
page	The instance of this page's implementation class that is processing the current request. A synonym for this when the programming language is Java.	Page
exception	The uncaught <code>Throwable</code> exception that caused the error page to be invoked.	Page

Implicit objects display on the Language tab page in the System Tree.



Implicit objects other than the exception object are always available within scriptlets and expressions. If the JSP is an error page (the page directive's `isErrorPage` attribute is set to true), the exception implicit object is also available.

You can often use an implicit object or a Web Target object model wrapper class to obtain the same functionality. For example, calling `out.println` in a server-side event is equivalent to calling `psDocument.Write`.

For more information about the exception implicit object, see “Error handling” on page 80. For more information about server-side events, see “Writing server scripts” on page 183.

## Scopes

There are four scopes for objects in a JSP application.

**Table 6-2: Scopes for objects in a JSP application**

Scope	Description
Page	Accessible only in the page in which the object is created. Released when the response is returned or the request forwarded.
Request	Accessible from pages processing the request in which the object is created. Released when the request has been processed.
Session	Accessible from pages processing requests in the same session in which the object is created. Released when the session ends.
Application	Accessible from pages processing requests in the same application in which the object is created. Released when the runtime environment reclaims the <code>ServletContext</code> .

## Procedures for editing scripts

In the integrated Script editor, you can create scripts for object events or independent scripts.

### Choosing an object or event for scripting

- ❖ **To write a script for an object event:**
  - 1 In the rightmost drop-down list, select a scripting language.



- 2 In the Script toolbar, select an object in the leftmost drop-down list.

The object list box lists all client and server scripts you create. In addition, it lists all scriptable HTML objects (objects for which events are triggered).

- 3 In the center drop-down list, select an event.

If the object you have selected is an independent script and does not have events, the list box is blank. If you have enabled the 4GL event model, server-side events are listed in blue. In the integrated Script editor, you can write a server-side script for an object only if you enable the 4GL event model for the page.

The script you write is saved in the HTML page, associated with the HTML object and its event attribute.

## Assigning an ID to an object in the document

A JSP target assigns default IDs when you create commonly scripted objects. You can assign or change the ID for any object from Page view or Source view of the HTML editor.

❖ **To assign or change an ID:**

- 1 Right-click the object and select the Properties menu item for the element from the pop-up menu  
*or*  
Select the HTML element and select Edit>Properties from the menu bar.
- 2 In the ID For Scripting text box, type a new value for the object.

---

### **IDs for style definitions**

You can assign IDs for style definitions in the Style Sheet editor.

---

## Creating a new script

When you select New Script from the pop-up menu in the integrated Script editor, the script is inserted in the Body section, after any existing scripts. If the script needs to be in the Head section, you can switch to Source view and move the script. In Source view, you can insert scripts anywhere in the document.

❖ **To create a new top-level script in the HTML document:**

- 1 Right-click in the script editor.
- 2 Select New Script from the pop-up menu and choose Server or Client from the cascading menu.
- 3 If you select Server, choose from three script delimiters for your HTML tag delimiters:

`<% ... %>`, `<%! ... %>`, and `<%= ... %>`

For information about writing server scripts, see Chapter 7, “Working with Application Servers and Transaction Servers,” and Chapter 9, “Developing 4GL JSP Pages.”

❖ **To create a script in a separate file:**

- 1 Select File>New from the menu bar.
- 2 In the New dialog, click the JSP tab.
- 3 On the JSP tab page, double-click the Script icon.

The Web Script editor displays in its own window, not as a pane within the HTML editor.

## Writing the code

The script editors provide many techniques to help you write scripts easily. You can:

- Build syntax using the System Tree
- Use InstaCode to complete object names and select from property lists (integrated Script editor only)
- Save and reuse syntax with the Clip window
- Copy and paste, including pasting text with generated document.write statements
- Use code in external files

## System Tree

The Language page of the System Tree lists objects and language syntax for several scripting languages.

The Current page of the System Tree lists the objects in the HTML document. The standard objects are listed along with any objects to which you have attached an ID. The Current page is empty if you are working in the standalone Web Script editor.

- ❖ **To insert a language element or object from the System Tree into a script editor:**
  - Drag the item to the editor, which inserts the fully qualified object name or property at the cursor position  
*or*  
Right-click an item in the System Tree, choose Copy from the pop-up menu, and paste the item (CTRL + V) into the script editor.

## InstaCode

In the integrated Script editor, InstaCode helps you write code by providing a list of objects, properties, and methods that are appropriate completions to code you have started to type. As you type in the script editor, you can press F2 to get a list of suggestions.

InstaCode completes code in two ways:

- When you type *part of an object name*, press F2 to see a list of all the objects that begin with those letters.
- When you type *the dot after an object name*, press F2 to see a list of properties and methods for that object.

The list of suggestions depends on the context. In a client script, you see objects belonging to the document object model and objects you have inserted in the HTML page, such as HTML elements and components. In a server script, you see JSP implicit objects and objects belonging to the Web Target object model.

- ❖ **To use InstaCode:**
  - 1 Type part of an object name and press F2.

- 2 Select an item from the list. (To close the list without making a selection, press ESC or click the close box.)
  - To select using keystrokes:
    - a Use the arrow keys or press a letter key to highlight an item.
    - b Press ENTER to insert the item in the document and close the window.
  - To select using the mouse:
    - a Click to highlight an item.
    - b Double-click to insert that item in the document and close the list.

### Examples

❖ **To insert the window object and the alert method in a client script:**

- 1 On a blank line or after a space, type "w" and press F2.
- 2 Select "window" from the list.
- 3 Type "." after "window" and press F2.
- 4 Select "alert(message)" from the list.

The script now contains the code "window.alert(message)". Edit the method argument if necessary.

❖ **To insert the scroll method for the window object in a client script:**

- 1 On a blank line or after a space, type "s" and press F2. Because window is the default object, the list displays window methods and properties.
- 2 Select "scroll(x,y)" from the list.

The script now contains the code "scroll(x,y)".

❖ **To insert the Write method for the psDocument object in a server script:**

- 1 On a blank line or after a space, type "ps" and press F2.
- 2 Select "psDocument" from the list.
- 3 Type "." and press F2.
- 4 Select "Write(string)" from the list.

The script now contains the code "psDocument.Write(string)".

## Code in external files

You can reuse code by copying it into another file or by referring to the code in a separate file through the SRC attribute for the SCRIPT element.

❖ **To get code from a file and insert it in a JSP Target Script editor:**

- Select Insert From File from the pop-up menu. The code appears in the script editor, and there is no further connection with the original file.

Scripts in HTML documents can be stored in external files, rather than in the HTML document. When the browser sees an SRC reference in a SCRIPT tag, it requests the script file from the server.

❖ **To put a script saved in the page into an external file:**

- 1 Right-click in the integrated Script editor and choose Save As External Script from the pop-up menu.
- 2 In the Save dialog, you can include the external file in the target by saving the file in the local directory for the JSP target.

❖ **To associate the script with code in an existing external file:**

- 1 Right-click in the integrated Script editor and choose New Script. Choose Client for the type of script.
- 2 Switch to Source view and add an SRC attribute to the SCRIPT element. For the value of the SRC attribute, you can assign:
  - A relative path to specify a file in the target:

```
<SCRIPT LANGUAGE=JavaScript
 SRC="../../common/frameset/test.js"></SCRIPT>
```

- An absolute path to any file (such a path might not be valid when you deploy the target)

The HTML editor recognizes the reference to the external file and displays the code in the integrated Script editor (when the script object is selected in the first drop-down list) as if it were part of the HTML document. The icon next to the object name in the drop-down list displays an additional image (of a paper sheet with a folded-back corner) to indicate that the script is saved separately.

To reference a server script that is stored in a separate file, you need to use the PSIMPORT tag. For more information on the PSIMPORT tag, see the *JSP Target Reference*.

## Finding and changing code

❖ **To find or replace text in the current script:**

- 1 Click inside the script that you want to search.
- 2 Select Edit>Find or Edit>Replace from the PowerBuilder menu.

Settings in the Find and Replace dialog boxes let you control the direction of the search. You can also select options restricting searches to whole words only or to strings matching the case of the search text.

If you need to find or replace text across many scripts on a page, switch to Source view. The Find and Replace dialog boxes look through all the scripts in a single pass.

---

### **Searches for special characters**

The script editors do not support searching for special characters, such as line breaks and tabs.

---

## Setting default formats for scripts in the Script editor

You can specify the font size, text colors, and how the tab key behaves for the the Script editor (as well as the Source view).

❖ **To configure the editor:**

- 1 With a page open in the HTML editor, select Design>Options from the menu bar.
- 2 On the Editors tab, select Script Editor in the Windows list box and change settings as appropriate.

You can make only minimal style selections for the Script editor, determining tab size and indent size, and whether to maintain tabs for blank space. You can also add or delete scripting languages for the Script editor. The Source view allows you to format individual HTML elements.

- 3 Click the Colors/Fonts tab, select a view in the Windows list box, and change the color and font setting as appropriate.

You can change colors and fonts for both the Script editor and the Source view.

For more information about applying formatting to text, see “Formatting HTML source display” on page 47.

## Techniques and tips for creating scripts

### Position of scripts

It is standard practice to put client-side scripts in the Head section of your document, but the Page view in the HTML editor does not handle scripts in the Head section.

You can put client-side scripts in the Body section, but:

- Scripts are not allowed within a TEXTAREA or SELECT tag
- There are restrictions for server scripts that build client scripts

### Balanced HTML that the editor can understand

Page view in the HTML editor requires valid HTML, so if you use a script to build part of an element and use straight HTML for the rest of the element, the editor will have trouble displaying the page containing the element in Page view. If you want to use Page view, make sure the HTML on your page is balanced. Start and finish an element in HTML only or in a script only. Do not start in one mode and finish in the other.

For example, if a script is to provide data for a table, you might think it would be easiest to code the table's heading row in HTML and then include a script for the data. To take advantage of the editor, however, it would be better to create the heading row in the script too, even if it seems like more work. When you save a page containing unbalanced client- and server-side script, the editor might alter part of the script that displays in the integrated Script editor.

### URLs in scripts

A JSP target manages links in HTML pages and JSPs. It remaps the URLs according to the directory structure of the target or deployed site. However, a JSP target cannot manage URLs in scripts the way it manages URLs in HTML attributes. Therefore, when you build URLs dynamically in a script, use methods provided in the Web Target object model to get information about the server and directory structure so that the URLs reflect the current environment.

The following methods belong to psDocumentClass and can be used to build a URL dynamically:

- File
- FileExtension
- Path
- Site

**Example**

To build a URL, your script might include code like this:

```
string filefortoday = "myfile"
"http://" + psDocument.Site() + "/" + filefortoday +
.htm";
```



# Working with Application Servers and Transaction Servers

## About this chapter

This chapter describes how to work with JSPs in a Web delivery environment that contains an application server, a transaction server, or both, in order to create dynamic Web content for your Web site.

## Contents

Topic	Page
Integrating with application servers	133
Working with server scripts	135
Using the Web Target object model	136
Accessing database content from your JSP target	139
Managing page data	140
Integrating with EAServer	150

## Integrating with application servers

If you want to deliver dynamic content for your Web site, you must integrate an application server (to process server-side scripts) into your Web site delivery strategy. With JSPs, you can take advantage of 4GL extensions to the Web Target object model. The 4GL JSP page interface can handle many of the coding details for you.

For 4GL and non-4GL JSP targets, you write server-side scripts in Java, although you can use JavaScript for client-side scripting. You can also use the JSP implicit object model in your server scripts.

For more information about the implicit object model, see “Implicit objects” on page 122.

## What an application server does

An application server processes code on the server system before a Web server sends a page to a Web browser. By taking advantage of the capabilities of an application server, you can include conditional execution, looping, and other programming structures in your Web pages.

Application servers process template or source files to return dynamic content. The server evaluates server scripts when the page is requested and generates the HTML page, which it sends to the client browser.

For an overview on how application servers fit into an integrated Web delivery environment, see Chapter 1, “Working with JSP Targets.”

Dynamic Web pages

A dynamic page is a page that is generated each time it is accessed. Using application servers to create dynamic pages helps you enhance your Web site. Dynamic Web pages can:

- Respond to input from a browser, returning data requested by the user.  
For example, a user can complete a form on a Web page, then view another page in response to data entered on the form.
- Customize the output for each user.  
After a user provides information on a Web page (such as areas of interest, or level of expertise), the content delivered to the user’s browser can be fine-tuned to the information provided.
- Customize the output for the display capabilities of the Web browser.  
Different page presentations can be generated based on the type of browser a client uses to access the Web site.

Processing dynamic Web pages

The template for a dynamic Web page typically provides all of the following content before and after processing by an application server:

**Table 7-1: Content processing by an application server**

<b>Template contents before processing</b>	<b>Contents after processing</b>
HTML-encoded text	HTML-encoded text
Embedded Web DataWindows or database queries with instructions for formatting the retrieved data	Formatted database information
Server-side scripts	Results of execution of server scripts
Client-side scripts	Client-side scripts

Users do not see the unprocessed template page in their browsers. They see the page content after the application server processes the template page, and after their browser processes client-side scripts.

Two ways to create dynamic Web pages

When you are working with a JSP target, you can create template pages in two ways:

- **4GL JSP pages** provide an event-driven infrastructure that lets you create JSP pages with dynamic content easily. The Web Target object model extensions that underlie these pages handle many of the coding details required to produce dynamic pages.

For more information about 4GL JSP pages, see Chapter 9, “Developing 4GL JSP Pages.”

- **Server-scripted pages** let you create all of your own server scripts. JSP implicit objects and the Web Target object model without 4GL extensions provide a number of server objects that you integrate into your scripts. Other server-side object models are available for deployment to specific server types.

For more information about writing server scripts without the 4GL Web page technology, see “Working with server scripts” next and “Managing page data” on page 140.

## Working with server scripts

For JSP targets, you write server scripts in Java. You can embed server scripts in your HTML pages or place them in separate script files. You can import a server script file that contains code for processing by an application server, or you can include a server script file if its contents need only to be redirected, without modification, from the server to client-side browsers.

Using script files

By storing code in separate script files, you can centralize common functions, thereby simplifying development and maintenance. Once you have created a script file, you can import this file into Web pages and into other script files, or you can insert the file contents into a page as embedded script. You can use the standalone Script editor to create separate script files.

For information on using the standalone Script editor, see Chapter 6, “Writing Scripts.”

Embedding server scripts

If you want server scripts to appear within an HTML page, you can write them in the integrated Script editor or insert them from a script file.

An embedded server script is delimited by HTML tags. When you create a server script for a JSP target in the integrated Script editor, you can choose from three script delimiters for your HTML tag delimiters: `<% ... %>`, `<%! ... %>`, and `<%= ... %>`

❖ **To embed a server script on a Web page:**

- 1 With a Web page open in the HTML editor, right-click in the integrated Script editor.
- 2 Select **New Script>Server>delimitertype** from the pop-up menu.

The server script delimiters you select appear in the Source view for your Web page. A new server script icon appears in the Page view, which you can position on the page, and a server script item appears (as the current item) in the first drop-down list of the integrated Script editor.

## Using the Web Target object model

Object model file

When you use the Web Target object model, the name of the object model file imported for the JSP application server is JSPOBJECT110.JAR.

Basic Web target object model classes

The object model file includes the following Web target classes:

**Table 7-2: Web Target object model classes**

Class	Description
PSCommandClass	Defines a SQL statement or stored procedure that can be reused multiple times on the same page
PSCursorClass	Represents a result set that is the output of a database retrieval operation
PSDocumentClass	Describes the current document
PSConnectionClass	Allows you to connect to a database
PSServerClass	Represents the application server environment in which Web pages run
PSSessionClass	Describes information that needs to persist for the duration of a particular session between a Web client and a Web site
PSErrorClass	Provides access to errors captured by the application server

**Pre-instantiated objects** The Web Target object model automatically creates unique instances of the following objects:

**Table 7-3: Instantiated classes in Web Target object model**

Class	Pre-instantiated object name
PSDocumentClass	psDocument
PSServerClass	psServer
PSSessionClass	psSession

In your scripts, you always refer to these objects. You do not need to instantiate PSDocumentClass, PSServerClass, and PSSessionClass.

**Object comparison** Web target objects are converted to JSP objects.

**Table 7-4: Conversion of Web target objects to JSP objects**

Web target class	JSP object
PSConnectionClass	Connection
PSCursorClass	ResultSet
PSDocumentClass	request, response
PSServerClass	pageContext
PSSessionClass	session

Objects to support the Web DataWindow

Typically you use the Web DataWindow DTC to integrate DataWindows into your Web application. The Web Target object model also supplies objects that enable you to instantiate and manipulate Web DataWindow controls. The following table lists the classes you can instantiate in a script to set up access to a Web DataWindow. For details about these classes, see the *JSP Target Reference*.

**Table 7-5: Classes for Web DataWindow instantiation**

Class	Description
PSConnectionParmsClass	Specifies the database connection parameters required for a Web DataWindow control to connect to a database.
PSDataWindowClass	Creates a new object for a Web DataWindow control. This object lets you add a DataWindow control (that you create in DataWindow Designer, PowerBuilder, or InfoMaker) to your page.
PSDataWindowSourceClass	Creates a new source parameter object. The object specifies an existing definition of a Web DataWindow control.

Class	Description
PSJaguarConnection	Specifies the connection information required to connect to a server component on EAServer. This component provides interoperability between the Web DataWindow control and the application server.
PSNamedConnectionParmsClass (obsolete)	Specifies the database connection information required to connect to a named (cached or profiled) database.

4GL extensions to the Web target object model

A psPage server object is created for each 4GL JSP page you create.

Objects that you place on a 4GL JSP page are assigned the PSSERVERSCRIPTABLE attribute by default. This attribute allows you to write server-side scripts (in addition to client-side scripts) to access properties, methods, and events for these objects.

---

**References to controls require a psPage prefix**

References to controls *must* be prefixed with *psPage* for server-side processing. Using a prefix was unnecessary in earlier versions of PowerBuilder. The change is needed now to make the pages thread safe.

---

4GL JSP pages rely on the psPage server object and the following classes for each supported object:

**Table 7-6: 4GL Classes and objects**

Object	Description
psPage	Represents a 4GL JSP page on the server, encapsulates the other server objects available to 4GL JSP pages, and controls page processing
PSButtonClass	Represents a client-side button on the server
PSCheckBoxClass	Represents a client-side check box control on the server
PSDropDownListClass	Represents a client-side drop-down list control on the server
PSImageClass	Represents a client-side image on the server
PSLinkClass	Represents a client-side hyperlink (anchor element) on the server
PSPasswordClass	Represents a client-side text box control on the server
PSRadioGroupClass	Represents a client-side group of radio button controls on the server

Object	Description
PSSStaticTextClass	Lets you manipulate the specified text from a server script
PSTextAreaClass	Represents a client-side multiline text box control on the server
PSTextClass	Represents a client-side single-line text box control on the server
PSWebDataWindowClass	Represents a Web DataWindow control on the server

## Accessing database content from your JSP target

If you want to access database content from your Web application, you should define a database profile to make this connection available throughout your JSP target. A database profile is a named set of parameters stored in the registry that defines a connection to a particular database.

### Defining database profiles

To define a database connection profile, select the DB Profiles button from the PowerBar. Use the Database Profiles dialog box and the interface-specific Database Profile Setup dialog box to define your profile. The database profile specifies the parameters for connecting to the database, including a user ID and password, a server name, a database name, and other optional information.

For more information about defining database profiles, see *Connecting to Your Database*.

### Using database profiles in a DTC

After you set up database access for a JSP target, you can add a Web DataWindow DTC to your page or write scripts that otherwise reference a database profile. In the Properties dialog box for the Web DataWindow DTC, you select both the DataWindow object you want to use and the database connection profile used by that DataWindow.

For more information, see “Selecting a database profile” on page 215.

### Using database profiles in a script

If you choose not to use the Web DataWindow DTC and instead create a database connection from a script, you can add logic to the connection definition that gets executed during page processing by an application server.

For information on writing scripts to access a database, see the *DataWindow Programmer's Guide*.

### Setting up database connections on a component server

When you use the Web DataWindow (with or without the DTC), it is the Web DataWindow server component that interacts with the database, so you need to set up database connections on the server where the component is running.

To make sure you have the same results at design time and runtime, it is preferable to select the same database connection type in the DataWindow painter (when you create a DataWindow object) that you plan on using on the server where the DataWindow is deployed (and that you select from a database connection drop-down list if you are using the Web DataWindow DTC).

For more information on the database connectivity software available on EA Server, see *Connecting to Your Database* and the *DataWindow Programmer's Guide*.

## Managing page data

Displaying Web content dynamically (in response to a user's actions) enhances the user experience of your Web site. Passing data from one page to another or sharing data among pages lets you generate the dynamic content for your site. When a user moves from one page to another, server scripts interpret the values of parameters and variables to define the content sent to the user's browser.

When discussing data transfer between Web pages:

- The linking page is the page on which a user action initiates a move (redirect) to another page
- The target page is the destination page of a move from a linking page

## About page parameters and variables

Parameters and variables let you share page data in different ways. Basically you define page parameters for information that will be passed to your page; you define page variables to contain values internal to the page.

### Parameters

**A page parameter** is a named value that you can pass from one page to another. Usually a page parameter is appended to a URL as a query string or is submitted with a form. After the value is passed from a page, a server script can access the value and use it as needed in generating the target page.

### Variables

**A page variable** is a temporary value that can be saved and made accessible to other pages, including the page where the variable is set. A page variable is available as long as a page is active.



**A session variable** is like a page variable, but with a longer life span. A session variable is available for the length of the user's browser session. You use a session variable to make the user's information available to many pages.

**Variables and login data** Variables, either page or session, can be used to store login information for a user's site visit. This lets that person visit a number of pages without needing to log in again.

4GL JSP pages

4GL JSP pages give you a straightforward way to define and keep track of parameters and variables during development. The 4GL extensions to the Web Target object model manage the parameters and variables when the page is processed by a JSP server.

For information about 4GL JSP pages, see Chapter 9, "Developing 4GL JSP Pages." For information about referring to parameters and variables in scripts on 4GL JSP pages, see "Adding scripts to 4GL JSP pages" on page 186.

## Using page parameters in server scripts

If you are not using the 4GL interface to manage your page parameters, you can access parameters submitted to a page by writing server scripts.

Non-4GL pages

To make full use of the page parameters, you can have the server script generate a client script. This server-side script generates a client-side script that sets a text box value (on a Web page that is not 4GL enabled) to the *id* parameter:

```
psDocument.WriteLine("<script>");
psDocument.WriteLine("function setValues()");
psDocument.WriteLine("{");
 psDocument.WriteLine("myForm.sle_1.value=' " +
 psDocument.GetParam("id")+ "'");
psDocument.WriteLine("}");
psDocument.WriteLine("</script>");
```

You can then code a client-side event (such as an onclick event for a button or the onload event for the page) to call the `setValues` function in the generated client-side script.

Passing a parameter in an anchor element

Passing page parameters from one page to another requires identifying the parameter (of the target page) on the linking page, then accessing the value in a server script on the target page. You can set up an anchor element or a form on the linking page to link to the page parameter of the target page.

On the linking page, create an anchor element (<A>). In the HREF attribute, specify the target URL with a query string appended to it. The query string can have one or more name-value pairs. A question mark separates the query string from the URL and an ampersand separates each name-value pair. The format is:

```
url?name1=value1&name2=value
```

**Example: passing data in a query string** Here a link on the linking page goes to the target page *nextpage.htm*. There are two values passed in the query string: Data and Name. On the target page, the page parameter names are called Data and Name and their values are "1" and "Jane". Note that parameter names are case sensitive.

```
Jane's data

```

Passing a parameter in a form

On the linking page, create a form for which the action is the URL of the target page. When the user submits the form, the form field names and values are passed to the target page as page parameters. Depending on the form method (GET or POST), the parameters are formatted as a query string or sent separately. No matter which method you use, server scripts on the target page see the values as page parameters.

**Example: passing data from a form** Here a form on the linking page asks the user to specify a name and a number. On the target page, *nextpage.htm*, the page parameters are called "Name" and "Data", and their values are whatever the user entered in the form fields.

```
<FORM id=FORM1 name=NameAndData action="nextpage.htm"
method=post>

User Name: <INPUT id=INPUT1 name="Name" type="TEXT">
Number of requests: <INPUT id=INPUT2 name="Data"
type="TEXT">
<INPUT id=INPUT3 name="Submit" type=submit>
</FORM>
```

Accessing the value of a page parameter

In a server script on the target page, you can get the value of page parameters with the GetParam method.

**Example** Here the script gets the value of "Name" and "Data".

```
username = psDocument.GetParam("Name");
userdata = psDocument.GetParam("Data");
```

## Using session variables in server scripts

The `psSession` object allows you to keep track of user login information and other data that you want to make available to all the pages in your Web application during a user's browser session. The `psSession` object also keeps track of user activity so that a user's session can be terminated if it becomes inactive.

The actual behavior of the session object depends on its implementation on each application server, but typically a session object is instantiated only if you try to access it. Session variables are properties of the `psSession` object. You create properties as you need them simply by setting them in a server script. For complete information about session objects, see the documentation for your application server.

### Setting the length of a session

The `psSession` object handles the lifespan of the session as well as session variables. When the user does not access the server for a specified number of minutes, the session times out. If you use the `psSession` object to manage user login information, the login information disappears when the user is inactive for the specified amount of time.

**Ending a session** One way to implement a timeout is by destroying the `psSession` object on the application server so that the `psSession` object and its properties no longer exist until you set new properties. If you query a property after the session ends, the `GetValue` method on the session object returns null. Pages that rely on shared information must handle the disappearance of that information. If your page gets values of `psSession` properties, your code should check for null and handle the situation of an expired session.

**Changing the length of a session** You can change the length of a session either in the server configuration or dynamically in a script. The server's own session object stores the session length in its `timeout` property. You can change the lifespan of a session by:

- Setting the server's timeout property using a `psSession` object
- Writing code that uses the correct property name for the server
- Writing conditional code tailored to each server you want to support

### Creating and setting the value of a session variable

You create a session variable with the `SetValue` method. This method creates the variable, if necessary, and sets the variable's value. If the `psSession` object does not exist, calling this method instantiates it. Creating the `psSession` object triggers timeout management.

**Example: creating a session variable** This script sets the value of the userid and password session variables:

```
psSession.SetValue("userid", "jdoe");
psSession.SetValue("password", "mydogsname");
```

You can also set the timeout property in server script.

**Example: specifying the application server** This code uses the `ObjectModelType` method to determine the current application server and sets the timeout property as named on that server:

```
if (psServer.ObjectModelType() == "JSPObject")
{
 psSession.SetValue("timeOut", 30);
}
```

The JSP target server timeout value is set in the *web.xml* file for the target. You can modify this on the JSP Options page of the Deployment Configuration Properties dialog box for the target. To change the timeout dynamically in a JSP session, you can call the `setMaxInactiveInterval` method on the session object:

```
session.setMaxInactiveInterval(1800);
```

The *web.xml* value for the session timeout is in minutes, whereas the value for the argument used by the `setMaxInterval` method is in seconds.

Getting the value of a session variable

The `GetValue` method gets a value from a property of the `psSession` object. If the property does not exist, `GetValue` returns null. The property does not exist if the session has timed out.

**Example: getting a session variable** This code gets the user's ID. If the ID does not exist, the user is redirected to the `login.htm` page.

In a real application, you would want to explain to the user what happened.

```
string curruser
curruser = psSession.GetValue("userid");
if (curruser == null)
{
 psDocument.Redirect("login.htm");
}
```

## Samples for retrieving and displaying data

This section presents separate examples for a JSP target without 4GL extensions to illustrate the objects and methods you use to:

- Get the value of a page parameter
- Establish a database connection and handle database errors
- Create a SQL query and use the page parameter in the WHERE clause
- Display the query results in a table with link formatting

In these examples, the target page receives the value of a department passed from a link on another page to a target page parameter. It then retrieves the names and IDs of employees who work in that department.

The employee names are then displayed in a table with a link to an employee detail page. The employee ID is in a query string of the link so that it can be used in another query on the detail page.

The complete source for these examples is provided at the end of this section.

Getting the value of a page parameter

The `GetParam` method for the `psDocument` object accesses the value of a page parameter. You can assign the value to a variable and use that variable in other scripts on the page.

**Example: getting the value of a page parameter** This script would appear after the first heading in the file. It assigns the page parameter value to the string variable `curr_dept`. It also writes the department name on the page:

```
string curr_dept;
curr_dept = psDocument.GetParam("Dept");
psDocument.Write("<P>Employees for department ");
psDocument.Write(curr_dept + "</P>");
```

Establishing a database connection

The `psServer` instance is automatically instantiated in your Web target and is available to server scripts on every page. Using `psServer` methods, you can define new connections at execution time or you can access connections you defined in PowerBuilder. (When you deploy a Web target, the connection information is made available to the application server.)

**Example: connecting to a database** Here the `GetConnection` method for `psServer` instantiates a `PSConnectionClass` object using the connection profile `Employees`. If an error occurs, the code calls a `WriteError` function to display error information:

```
PSConnectionClass conn;
conn = psServer.GetConnection("Employees");
rows = 0;
if (conn.GetError() != null)
{
 WriteError("GetConnection", conn);
 return;
}
```

The `WriteError` function is called only if the error object is not null. The script that defines the `WriteError` function is shown in the next example and is placed in the `Head` section of the document. The arguments for `WriteError` are:

- The method that caused the error
- The instance of the `PSConnectionClass` object

The `WriteError` function calls the `GetError` method for the connection object to get the first instance of the `PSErrorClass` object. An error object is available only if an error has occurred; otherwise, `GetError` returns null.

Handling database errors

After getting error information, the `GetError` function writes the connection name, error code, error message, and the name of the function that failed in the document. The `GetCode` and `GetMessage` methods for `PSErrorClass` get the error code and message.

**Example: handling database errors** This is the code for the `WriteError` function:

```
PSErrorClass errobj;
function WriteError(function_called, connName)
{
 errobj = connName.GetError();
 str = errobj.GetCode() + " " + errobj.GetMessage();
 psDocument.Write("<P>Error: ");
 psDocument.Write (function_called + " " + str);
 psDocument.Write("</P>");
 return;
}
```

Using the page parameter in a SQL query

After you establish a connection, you can retrieve data with a SQL statement and store the result set in a `PSCursorClass` object. To retrieve data, this code:

- Builds a string that is the SQL statement. The `curr_dept` variable, which holds the page parameter value, is incorporated into the WHERE clause.
- Uses the string with the SQL statement as an argument for the `CreateCursor` method. This method belongs to the `PSConnectionClass` object.
- Assigns the returned result set to the newly instantiated `PSCursorClass` object called "mycursor".
- Checks whether a database error occurred and calls the `WriteError` function if necessary.

**Example: retrieving and storing data** The code that creates the cursor and retrieves data looks like this:

```
//build a SQL statement
sqlquery = " SELECT \"employees\".\"fname\" , " +
 " \"employees\".\"lname\" , " +
 " \"employees\".\"empid\" " +
 " FROM \"employees\" " +
 " WHERE \"employees\".\"deptid\" = " +
 "\"" + curr_dept + "\"";
// Do the query and assign the result set to
PSCursorClass mycursor;
mycursor = conn.CreateCursor(sqlquery);
if (conn.GetError() != null){
 WriteError("CreateCursor", conn);
}
return;
}
```

Displaying the query results in a table with link formatting

After the rows are retrieved, methods for the `PSCursorClass` object provide access to the data. Code that writes HTML for displaying data is mixed with method calls that get the data from the `PSCursorClass` object. This code:

- Calls `GetRowCount`, a method of the `PSCursorClass` object, to find out how many rows are in the result set
- Writes HTML for the Table element
- Writes HTML to close the Table element
- Loops through the rows in the result set

For each row in the result set, the code:

- Writes HTML for a table row with one cell.
- Writes an anchor element (<A>) tag with a query string using data from the second column (empid). The GetValue method for the PSCursorClass object gets the data.
- Writes text inside the anchor element, using GetValue to get the employee first name from the first column (0) and the last name from the second column (1). Column numbers start with 0 and correspond to the columns in the SQL SELECT statement.
- Writes HTML that closes the A, TD, and TR elements.
- Calls the MoveNext method for the PSCursorClass object to go to the next row in the result set.

**Example: processing rows** The code that processes the rows looks like this:

```
rows = mycursor.GetRowCount();
// Write Table start tag
psDocument.Write("<TABLE BORDER=1>");
// Loop over retrieved rows
// where rows variable is the row count
for (var i=0; i < rows; i++)
{
 // Write TR and TD start tags
 psDocument.Write("<TR><TD>");

 // Write A element with employee ID in query string
 psDocument.Write(" <A HREF=\"detail.htm?Key=");
 psDocument.Write(mycursor.GetValue(2) + "\">");

 // Write first and last names
 psDocument.Write(mycursor.GetValue(0) + " ");
 psDocument.Write(mycursor.GetValue(1));

 // Write A, TD, and TR end tags
 psDocument.Write("<TD><TR>");
 // Go to next row in result set mycursor.MoveNext();
}
// Write Table end tag
psDocument.Write("</TABLE>");
```



## Complete example

The complete page looks like this in Source view:

```

<HTML>
<HEAD>
<%
// function for displaying error information
function WriteError(function_called, connName)
{
 PSErrorClass errobj;
 errobj = connName.GetError();
 str = errobj.GetCode() + " " + errobj.GetMessage();
 psDocument.Write("<P>Error: ");
 psDocument.Write (function_called + " " + str);
 psDocument.Write("</P>");
 return;
}
%>
</HEAD>
<BODY>
<H1>Employees</H1>
<P><% // Get page parameter and write value in document
string curr_dept;
curr_dept = psDocument.GetParam("Dept");
psDocument.Write("<P>Employees for department ");
psDocument.Write(curr_dept + "</P>");%></P>

<P><% // Get a connection
conn = psServer.GetConnection("Employees");
rows = 0;
if (conn.GetError() != null) {
 WriteError("GetConnection", conn);
 return;
}
// Construct the SQL statement
sqlquery = " SELECT \"employees\".\"fname\" , " +
" \"employees\".\"lname\" , " +
" \"employees\".\"empid\" " +
" FROM \"employees\" " +
" WHERE \"employees\".\"deptid\" = " +
"'" + curr_dept + "'";
// Retrieve the data
PSCursorClass mycursor
mycursor = conn.CreateCursor(sqlquery);
if (conn.GetError() != null) {
 WriteError("CreateCursor", conn);
 return;
}

```

```
// Get the number of rows retrieved
rows = mycursor.GetRowCount();

psDocument.Write("<TABLE BORDER=1>");
for (var i=0; i < rows; i++) {
 // Write TR and TD start tags
 psDocument.Write("<TR><TD>");
 // Write A element with employee ID in query string
 psDocument.Write(" <A HREF=\"detail.htm?Key=\"");
 psDocument.Write(mycursor.GetValue(2) + "\">");
 // Write first and last names
 psDocument.Write(mycursor.GetValue(0) + " ");
 psDocument.Write(mycursor.GetValue(1));
 psDocument.Write("<TD><TR>");
 mycursor.MoveNext();
}
psDocument.Write("</TABLE>");
%>
</BODY>
</HTML>
```

## Integrating with EAServer

EAServer is a middle-tier component transaction server that hosts executable business objects called components. You can access components on EAServer from your Web applications.

EAServer can host various types of components including PowerBuilder objects and Java classes or beans. Although these components have different origins, they have several things in common. They:

- Encapsulate business logic that one or more applications need to execute
- Shift processing to the middle-tier EAServer server, enabling application clients to remain thin
- Are stored in packages and contain methods that clients can call to perform specific operations

Developers typically create components using tools such as PowerBuilder. These tools enable developers to build classes or objects and deploy them directly to EAServer as components.

Access to components

You can take advantage of the JSP Target user interface to access EAServer components by using:

- **4GL JSP pages** to help you create server scripts that access EAServer components.
- **Web DataWindow DTCs** to let you access Web DataWindow server components available on EAServer. The Web DataWindow retrieves, processes, and displays data on a page, and EAServer manages the database connection.

4GL JSP pages

4GL JSP pages give you quick access to EAServer components by:

- Letting you bind page controls to EAServer component properties
- Giving your scripts access to EAServer components as represented by variables
- Providing drag-and-drop programming access to methods of EAServer components

For more information about 4GL JSP pages see Chapter 9, “Developing 4GL JSP Pages.”

Web DataWindow DTC

The Web DataWindow DTC provides a design-time interface that lets you select the DataWindow objects to include on a Web page. It also permits you to override certain settings of the DataWindow object before the Web DataWindow displays in the client browser.

4GL JSP pages also provide enhancements for Web DataWindow objects available through EAServer. A server representation of a Web DataWindow object (the PSWebDataWindowClass object) allows you to write server-side scripts for DataWindow events and methods. For more information about the Web DataWindow DTC, see Chapter 11, “Using the Web DataWindow Design-Time Control.”

## Accessing components

Before you can view information about or include an EAServer component on your Web page, you must make sure a profile exists for the server that contains a component you want to use. A connection from a JSP target to EAServer uses the Internet Inter-ORB Protocol (IIOP). You must make sure the EAServer has an IIOP listener configured before you set up a connection for your JSP target.

With a connection to the server established, you can:

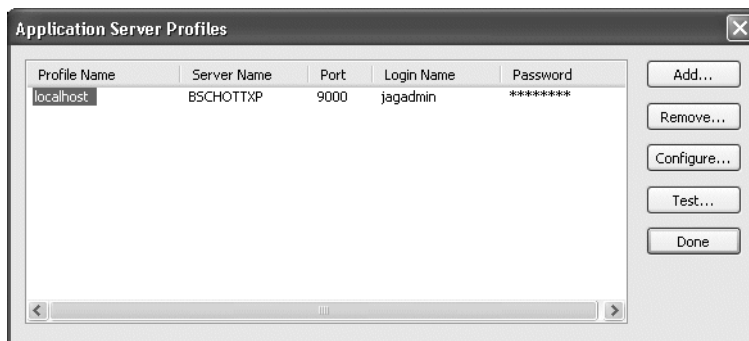
- View a list of accessible servers from the Components tab page of the System Tree
- Display the packages, components, and methods installed on EAServer
- Get information about a component or method

You can drag and drop an EAServer component from the System Tree to a 4GL JSP page that is open in the Page view of the HTML editor. This opens the Page Properties dialog box to the EAServer page and adds the component to the list of components available to the Web page.

❖ **To define an EAServer connection profile:**

- 1 Select Tools>Application Server Profiles from the PowerBuilder menu  
*or*  
Right-click anywhere in the Components tab page of the System Tree and select EAServer Servers from the pop-up menu.

The Application Server Profiles dialog box displays.



- 2 Edit the list of EAServer profiles.

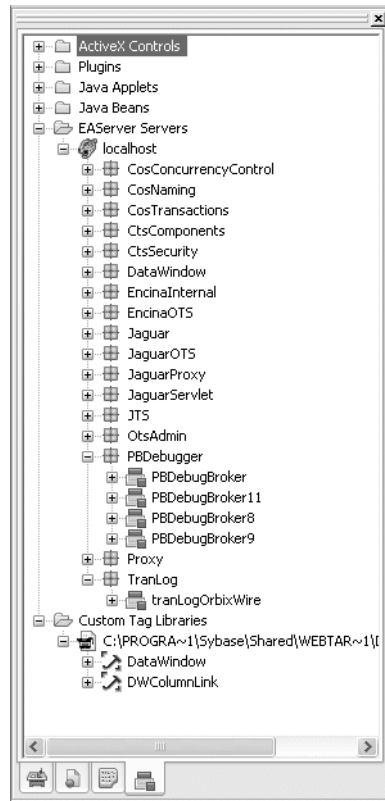
You can add, modify (configure), or remove EAServer connections as needed. You can also select a connection and test it. You should make sure the server is running before you test the connection.

- 3 Click Done to save your changes.

An item representing the server appears under the EAServer Servers node on the Components tab page of the System Tree.

Viewing components

After you establish a connection to an EAServer for your JSP target, you can see the components and the methods for those components installed on the server.



You can view the list of server components that you have added to a page from the Page Properties dialog box for 4GL JSP pages. When you add a Web DataWindow DTC to a page, you can select a DataWindow component from the Web DataWindow DTC Properties dialog box; the DataWindow HTMLGenerator110 component is selected and enabled by default.

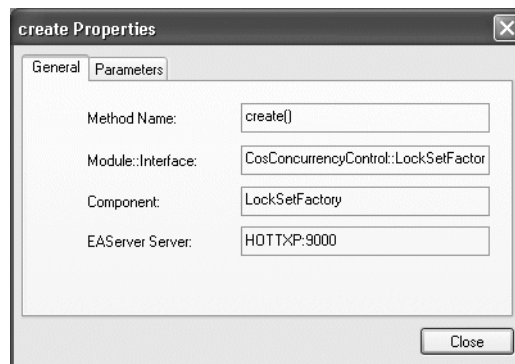
Getting information about components and methods

After you define a connection, you can get information about components and component methods from the System Tree.

❖ **To get information about EAServer components and methods:**

- 1 From the Components tab of the System Tree, expand the EAServer Servers branch.
- 2 Expand the appropriate server and package to find the component you want information about  
*or*  
Expand the appropriate server, package, and component to find the method you want information about.
- 3 Right-click the component or method and select Properties from the pop-up menu.

The properties dialog box for the selected component or method displays:



Method properties have a Parameters tab to display the parameters of the component method.

- 4 (Optional) Click the Parameter tab to see information about the parameters the method uses.

About this chapter

This chapter describes JSP Web services.

Contents

<b>Topic</b>	<b>Page</b>
About JSP Web services	155
Using the JSP Web Service Proxy wizard	155
SOAP processing in JSP targets	159
Adding a custom tag for Web services	160

## About JSP Web services

You can use Web services in your JSP by generating custom tags for them. The JSP Web Service Proxy wizard on the Web page of the New dialog box creates a custom tag library with the information necessary for calling a Web service in a JSP.

For more information on custom tags and custom tag libraries, see “Custom tags” on page 79 and “Adding a custom tag for Web services” on page 160.

## Using the JSP Web Service Proxy wizard

The JSP Web Service Proxy wizard collects information such as the location of a Web Services Description Language (WSDL) file, the service, and port. You can specify overrides to the WSDL file for a custom bean name, Java class name, Java package name, tag library descriptor (TLD) name, JAR file name, output variables, and the selection of operations within a service.

---

### PowerBuilder client Web services wizard

The JSP Web Service Proxy wizard is similar to the Web Service Proxy wizard on the Project page of the New dialog box, but the latter creates a PowerBuilder client for a Web service and can only be used in PowerScript targets.

For more information about Web services and the Web Service Proxy project wizard, see the chapter on Web services in *Application Techniques*.

---

## Files added by the wizard and files required by the server

Files added by the JSP Web Service Proxy wizard

The JSP Web Service Proxy wizard adds a TLD and a JAR file containing TLD classes to JSP target subdirectories. When you deploy the JSP target, these files are deployed along with the other target files. Table 8-1 lists the files created by the JSP Web Service Proxy wizard.

**Table 8-1: Files added to target WEB-INF subdirectories**

File added by wizard	Description
tlds\Service_Name.tld	A default file name is provided by the Web service WSDL, but the JSP Web Services Proxy wizard lets you rename the TLD file that it adds to the target
lib\Service_Name.jar	A default file name is provided by the Web service WSDL, but the JSP Web Services Proxy wizard lets you rename the JAR file that it adds to the target

---

### Files created on a full build of your JSP target

When you build a JSP target for the first time, PowerBuilder creates *Jaguar.properties* and *Database.properties* files in the target's *WEB-INF/Class* directory. These files contain connection information from your current PowerBuilder database and EAServer profiles. If you modify these profiles before you deploy a JSP target, you should perform a full build to make sure the *Jaguar.properties* and *Database.properties* files contain up-to-date connection information.

For information about performing a full build, see “Building JSP targets” on page 234.

---



Other files required on the JSP server

At runtime, the JSP server where you deploy your target must have additional files in its class path for a Web service to work:

```
axis.jar
commons-discovery.jar
commons-httpclient.jar
commons-logging.jar
dom.jar
jaxrpc.jar
log4j-1.2.8.jar
pbwst110.jar
saaj.jar
sybasewst.jar
wsdl4j.jar
xercesImpl-2.1.0.jar
xml-apis.jar
```

These files are not deployed with your JSP target. If your server does not have these files in its class path, you can copy them to a directory in the server class path from the Sybase\Shared\PowerBuilder\WEB-INF\lib directory. For Tomcat, you should copy them to the Tomcat Shared\Lib directory.

For HTTPS connections over SSL, you also need to copy the following files to your JSP server class path:

```
jcrt.jar
jnet.jar
jsse.jar
```

## Using the UDDI browser in the wizard

PowerBuilder provides live access to Universal Description, Discovery, and Integration (UDDI) registries for both PowerScript and JSP targets. The UDDI service is an industry-wide effort to bring a common standard for business-to-business integration. It defines a set of standard interfaces for accessing a database of Web services.

The UDDI browser is incorporated into the Web Service Proxy wizard and the JSP Web Service Proxy wizard. You open UDDI search pages by clicking the Search From UDDI button on the Select WSDL File page of these wizards.

The UDDI Search page has three required search fields and four search options listed in the following table:

**Table 8-2: UDDI search fields and options**

Search field or option	Description
UDDI profile	Editable drop-down list for the name of a UDDI operator. You can associate a UDDI profile with a query URL. The drop-down list allows you to select predefined profiles for the Microsoft and IBM public UDDI registries.
Query URL	Text box that displays the URL for the Web service registry in which you want to find a Web service. If you selected a predefined profile in the UDDI Profile drop-down list, the URL associated with that profile displays in the text box. You can also enter a query URL and associate the URL with a profile name by clicking the Save Profile button.
Search For	Text box for entering the keyword you want to use in a UDDI search.
In	Drop-down list for “Service Names” (default) or “Business Names.”
Exact Match	Check box option. If selected, limits the search to the current value in the Search For drop-down list.
Case Sensitive	Check box option. If selected, limits the search to the capitalization used by the current value in the Search For drop-down list.
Sort	Radio button option. Sorts search results in ascending or descending order.
Maximum Rows	Spin button option. Limits the number of search results returned to the number that you enter in this spin button control.

Which wizard page the UDDI search displays next depends on whether you are searching a keyword in business names or service names:

- **For a business name search** The Select Business wizard page returns a list of business names and descriptions that meet your search criteria. After you select a business name and click Next, a list of service names is returned on the Select Service wizard page, along with a service description and WSDL file name for each service listed.
- **For a service name search** The Select Service wizard page returns a list of service names along with a business name, service description, and WSDL file name for each service listed.

After you select a service on the Select Service page of a wizard, the UDDI search is complete and you continue your selections on the remaining pages of the wizard.

## SOAP processing in JSP targets

### Datatype support

JSP targets in PowerBuilder use the Apache Software Foundation's Axis software for Simple Object Access Protocol (SOAP) processing. The Axis software includes support for user-defined complex datatypes and document-type WSDL files.

Axis provides a WSDL2 Java tool that builds Java proxies and skeletons for Web services with WSDL descriptions. Axis follows the Java API for the XML-Based Remote Procedure Calls (JAX-RPC) specification when generating Java client bindings from the WSDL descriptions and generates only those bindings necessary for the client. Table 8-3 shows the type of Java file generated from each entry type in the WSDL file.

**Table 8-3: Java client bindings generated from WSDL file**

For each WSDL	Java class generated
Type in the types section	JavaBean
complexType	Holder if this type is used as an in/out parameter
portType	Java interface
binding	Stub class
service	Service interface and service implementation (the locator)

In a JSP target, the authoring tool (HTML editor) is Unicode enabled, so you can input text in multiple languages on a single page. It accepts UTF-16 Unicode text only; however, JSP files with ANSI-encoded or UTF-8 text can still be imported in the editor. Text with these encodings is automatically converted to UTF-16.

### Custom tag support in JSP targets

Custom tags to be used in a JSP target are processed at design time. The code to process a custom tag invoking a Web service is generated in Java and compiled using the Java compiler in the JDK. The generated code uses the Java client binding class generated by the Axis toolkit.

The main purpose of the generated code is to provide the glue needed to call and pass arguments from a custom tag in a JSP to the Axis toolkit. The custom tag provides the ability to access both input and output arguments through attributes (for input variables) and scripting variables (for output variables). In addition, the ability to access the return value is provided by a scripting variable.

For a custom tag to function properly, three components must be created:

- **JavaBean to handle custom tag at runtime** Each bean provides support for one operation in a WSDL file. All beans that support the same service in the WSDL file are placed in the same package. The default name of the package is the service name. The name of the package can be overridden by the user in the wizard.

For each argument in the Web service there is an instance variable in the class. If the argument is an input variable to the Web service, there is a *setArgumentName* method. If the argument is an output variable from the Web service, there is a *getArgumentName* method.

After the code has been generated, it is compiled to a *.class* file

- **TagLib directive in the JSP file** When a Web service is added to a JSP page, a directive is added to the top of the JSP page to import the appropriate tag library.
- **Tag Library Descriptor (TLD) file** A TLD is one of the three key components required for the use of a custom JSP tag. A TLD is an XML document that describes a tag library. A TLD contains information about the tag library as a whole and about each tag contained in the library. The generated TLD files are placed in the WEB-INF/services directory of the target application.

## Adding a custom tag for Web services

Once you have generated your custom tag, it appears in the System Tree. When you drag it to a JSP, the tag library is automatically associated with the page. You must specify the input and output arguments for the custom tags for the Web services in the JSP. Output arguments are stored in the *pageContext* of the JSP container.

Deployment of the custom tag for Web services is the same as deployment of any custom tag in PowerBuilder. See “Editing a JSP deployment configuration” on page 239 for details.

Custom tags for Web services throw a `JspTagException` for nonrecoverable errors. The `JspTagException` contains information about the root cause of the exception and the point where the error occurred in processing the custom tag. This exception can be caught in a Try-Catch block or mapped to a specific error page in the Deployment Configuration Properties dialog box for the JSP target. An error page can also be specified in a page directive. See “Error handling” on page 80 and “Error Mapping” on page 245 for details.

Example using a  
currency exchange  
Web service

This example, using the currency exchange service available on the XMethods Web site at <http://www.xmethods.com/sd/CurrencyExchangeService.wsdl>, demonstrates how custom tags for Web services on a JSP are defined to a JSP container. You proceed with these steps after generating a TLD and JAR file for the service using the JSP Web Service Proxy wizard. The remainder of this example assumes that you accept all the wizard defaults after selecting the *CurrencyExchangeService.wsdl* file.

First you declare the custom tag library to the JSP. This makes all of the tags in the library available to the JSP. The exchange prefix allows for easy reference to the tag library. You can drag the library from the Components tab of the System Tree to the JSP in the HTML editor to add this code.

```
<%@ taglib
 uri="WEB-INF/tlds/CurrencyExchangeService.tld"
 prefix="exchange" %>
```

Once the tag library is available to the page, you declare the input arguments for the custom tags in a server script.

```
<%
 String firstCountry = "usa";
 String secondCountry = "japan";
%>
```

Next, you invoke the Web service through the custom tag, passing the input parameters in a server-side expression.

```
<exchange:getRate country1="<%= firstCountry %>"
 country2="<%= secondCountry %>" />
```

Then you get the value of the *returnValue* variable from the custom tag and display it. This value is set when the tag is executed.

```
<P>The exchange rate between "<%= firstCountry %>" and
"<%= secondCountry %>" is:
<%= CurrencyExchangeService_getRate_returnValue %></P>
```

# Developing 4GL JSP Pages

## About this chapter

This chapter describes how to create and develop 4GL JSP pages using extensions to the Web Target object model. By enabling 4GL functionality, you can take advantage of a rich user interface that simplifies how you develop Web sites with dynamic content.

## Contents

Topic	Page
About 4GL JSP pages	163
Developing pages	164
Using parameters and variables	169
Accessing EAServer components	174
Adding controls	178
Writing server scripts	183
How page request processing works	189
Disabling 4GL mode	191

## Before you begin

For information on troubleshooting 4GL JSP pages, see “Troubleshooting 4GL JSP pages” on page 255.

## About 4GL JSP pages

4GL JSP pages are enhanced Web pages that incorporate extensions to the Web Target object model to generate template (source) code for dynamic Web pages. 4GL JSP pages rely on the object model to handle the complexities of data transfer, HTML generation, and Java or JavaScript generation for server scripts. With many of the implementation details taken care of for you, you can concentrate on designing your pages and coding the application logic.

4GL JSP pages integrate with other Web pages on your Web site. They are suitable for sharing data with other pages across your site and accessing components installed on EAServer. 4GL JSP pages provide enhanced support for Web DataWindow objects on HTML or JSP pages. You cannot hand-code a Web DataWindow on a 4GL JSP page.

4GL JSP pages can be deployed to EAServer, Tomcat, or other JSP 1.2 or higher servers that support command line deployment.

4GL JSP pages help you:

- Manage page data among Web pages (using page parameters, page variables, and session variables)
- Access data from EAServer components
- Bind data to controls on your page
- Manage page navigation
- Create server scripts with minimal coding effort

## Developing pages

With 4GL functionality enabled, you can rely on an event-driven infrastructure to handle many of the details of coding server scripts for your Web pages. As with other Web target files, you edit 4GL JSP pages in the HTML editor and write scripts in one of the script editors. The integrated Script editor provides additional support for some of the features available in 4GL pages.

You can create new 4GL pages or you can modify existing pages to enable 4GL processing after you add the pages to your Web target. If you change a non-4GL page to 4GL mode, you must manually remove any existing FORM tags in Source view. Each 4GL JSP page is represented as a single form. You cannot use nested FORM tags on these pages.



## Creating a new 4GL JSP page

You create new 4GL JSP pages using the 4GL JSP Page wizard. If you want to change an existing page to 4GL mode, you can enable 4GL mode by selecting a check box in the Page Properties dialog box for the page.

The 4GL JSP Page wizard prompts you for basic design information, options for error reporting, page parameters, and EAServer components that you want to use on the page. If you are unsure about values to enter for a particular wizard field, you can leave the field blank or accept the field default, then add or change the information later in the Page Properties dialog box.

**Table 9-1: 4GL JSP Page wizard page design information**

Specify this	To do this
Style sheet	Select an existing style sheet for the page.
Background image	Select an image that displays as the page background.
Scroll image	Make the background image scroll with the page.
Background color	Select a color for the page background.
Header based on title	Add a page header that is the same as the title. This header appears on the printed version of the page.
Date created footer	Add a page footer that shows the date the page was created. This footer appears on the printed version of the page.

The 4GL JSP Page wizard also includes a page with selections for the specialized error reporting capabilities available to 4GL pages:

**Table 9-2: Error reporting selections for 4GL pages**

Select this	To do this
Show runtime errors in alert message box	Display page processing errors in an alert message box
Show runtime errors as text at the top of the page	Display page processing errors at the top of a generated page
Show runtime errors as text at the bottom of the page	Display page processing errors at the bottom of a generated page
Enable trace	Display detailed information about page processing during page development

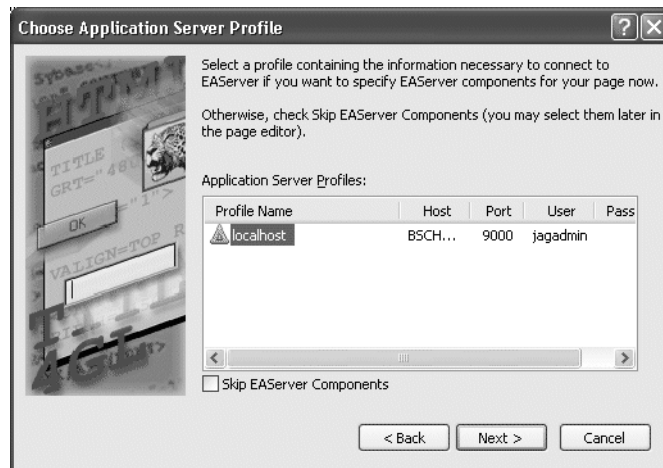
❖ **To create a new 4GL JSP page:**

- 1 From a JSP target select File>New from the menu bar  
or  
Right-click a target or target folder in Workspace view and select New from the pop-up menu.

- 2 Click the JSP tab in the New dialog box.  
Make sure the JSP target to which you want to add a 4GL JSP page is selected in the Target drop-down list box.
- 3 Click the 4GL JSP Page wizard icon to start the wizard, then click Next.
- 4 On the Specify New 4GL JSP File page of the 4GL JSP Page wizard, enter the title and file name for the page you want to create.
- 5 Provide design information for your Web page in subsequent wizard pages.
- 6 Specify options for error reporting and click Next.
- 7 Specify any page parameters that will provide input values for the page, and click Next.

You can add page parameters at a later time from the Page Properties dialog box for your page.

- 8 Select an EAServer profile, making sure the EAServer you select is running, and click Next to list the available components on this server  
*or*  
Select the Skip EAServer Components check box if you want to select components at a later time, then click Next and skip the next step in this procedure.

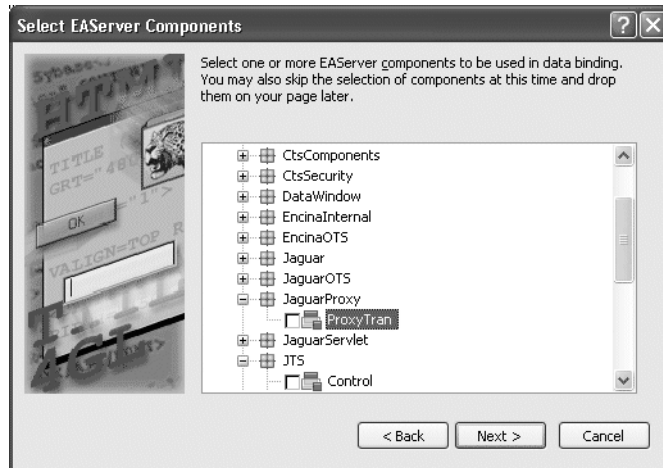


This wizard page lists EAServer profiles already configured in your development environment.

**Make sure your EAServer is running**

If you click Next without selecting the Skip EAServer Components check box, the wizard attempts to connect to the server that is selected in the Choose EAServer Profile dialog box.

- 9 Select the EAServer components you want your 4GL JSP page to access, and click Next.



- 10 On the last wizard page, review the summary of page properties, and click Finish.

## Enabling 4GL mode in an existing page

You change an existing page in your JSP target to 4GL mode by opening the page in the HTML editor and selecting a check box in the Page Properties dialog box. Some of the pages of the Page Properties dialog box are available only to Web pages for which you select the 4GL mode. After you select the 4GL mode check box, you can add other 4GL properties to the page and access the server-side event model in the integrated Script editor.

**Table 9-3: Property pages of the Page Properties dialog box**

On this page	Specify this
Page	Presentation details for the page.
Destination	A self-link or a link to a target page, including parameters and parameter bindings for the page. Available to 4GL pages only.

<b>On this page</b>	<b>Specify this</b>
Parameters	Page parameters for the current page and default values for those parameters, if any. You cannot enter default values for parameters on a non-4GL page.
Variables	Variables that your page will use. Available to 4GL pages only.
EAServer	Variables that represent EAServer components. Available to 4GL pages only.
Errors	Type of error reporting the page uses. Available to 4GL pages only.
JSP Directives	Add directives (page, include, or taglib) to your page. Available in JSP targets only.
Inline Styles	Style properties for the page.
Advanced	Additional attributes for the page. (Some user interface properties, such as a target page link, are also listed here.)

---

### **Post-conversion manipulation**

If you change an existing page to 4GL mode, you must manually remove any existing FORM tags and you must make sure that each control on the page has a unique name. If you want to take advantage of 4GL functionality with a control you added before changing the page to 4GL mode, you must select the Server Side Scriptable check box on the control property sheet (or add the PSSERVERSCRIPTABLE attribute for the corresponding INPUT or OBJECT tag in Source view).

---

#### **❖ To enable 4GL mode in an existing page:**

- 1 Open the page in the HTML editor.
- 2 Right-click in the page, then select Page Properties from the pop-up menu.
- 3 On the Page tab of the Page Properties dialog box, select the Enable 4GL Web Server Side Event Model check box.

When you select the check box, the 4GL property pages of the Page Properties dialog box are enabled.

- 4 (Optional) Specify properties for the 4GL page on the newly enabled pages of the Page Properties dialog box.

## Adding content to 4GL JSP pages

After you open a new Web page in the HTML editor, you are ready to start developing the page contents. The Page tab page of System Tree displays information about any objects and controls that you add to your page.

When you develop a page, you typically add content in a prescribed order. For example, you define parameters and variables as a first step, in order to access those values from objects or scripts as you build the page. The following procedure shows the steps you will most likely follow to develop 4GL JSP pages.

❖ **To develop a 4GL JSP page:**

- 1 Add page parameters, page variables, and session variables.  
See “Using parameters and variables” on page 169.
- 2 Add access to properties of EAServer components.  
See “Accessing EAServer components” on page 174.
- 3 Insert form controls, and then bind these controls to parameters, variables, or EAServer component properties already accessible on your page.  
See “Adding controls” on page 178.
- 4 Set up page navigation.  
See Chapter 10, “Setting Up Page Navigation.”
- 5 Write scripts for the events on your page.  
See “Writing server scripts” on page 183.

## Using parameters and variables

Web pages rely on page parameters and page variables to share data between Web pages or to pass data from one page to another. For more information about page parameters and variables, see “Managing page data” on page 140.

### 4GL advantage

4GL JSP pages provide a straightforward way to keep track of parameters and variables during development. You can define parameters and variables from the Page Properties dialog box. The 4GL extensions to the Web Target object model help manage the parameters and variables when the page is processed by JSP servers.

You can set target page parameters for a non-4GL JSP page, but to take advantage of the JSP Target user interface for parameter binding, the linking page must be 4GL enabled.

Navigation style

The type of value you can pass to a target page parameter depends on the navigation style you select. You can navigate to another page through a hyperlink, a form submit, or a server-side redirect script. However, if you use only the hyperlink navigation style, you lose the advantages of 4GL functionality for passing parameters.

The 4GL form submit style is equivalent to setting FORM element attributes as follows: the ACTION attribute to the target URL and the METHOD attribute to POST. The entire 4GL page is considered as a single form. You select the target URL on the Destination tab of the Page Properties dialog box. Parameters on the target page are automatically bound to parameters, variables, or controls on the linking page that have the same names.

For more information on navigation style and its effects on passing parameters, see “About page navigation” on page 193.

Steps in passing parameters

Typically, the steps involved in passing parameters from one page to another include:

- Setting up parameters for a target page from the 4GL JSP Page wizard or the Parameters page of the Page Properties dialog box.
- Setting up the parameters, variables, or control values on the linking page that you want to send to the target page.
- Selecting the binding type and setting the value you want to bind from the linking page to the target page.

---

**Automatic selection of binding type and bind value**

When you use a 4GL form submit navigation style (by selecting the target page on the Destination tab of the property sheet for the linking page), binding type and bind value are automatically selected. These noneditable selections are based on name matching between parameters on the target page and parameters, variables, or controls on the linking page.

---

For information on referring to parameters and variables in scripts on 4GL JSP pages, see “Adding scripts to 4GL JSP pages” on page 186.

## Setting up page parameters

You use page parameters on a target page to manage data sent from a linking page. If no data is sent and the target page is 4GL-enabled, the target page can use default values for its parameters. You set default values for parameters on the Parameters page of the Page Properties dialog box. Default values for page parameters are valid only on 4GL-enabled pages.

---

### **If a default value is not set**

If no data is sent from a linking page for a given parameter and a default value is not set on the 4GL target page, the only value available to the target page is an empty string ("").

---

On 4GL JSP pages, parameter values also become page variables that you can access in server scripts.

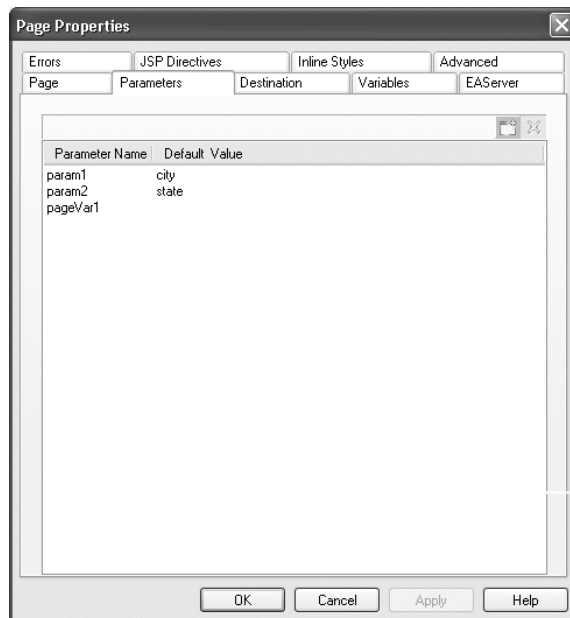
Identifying parameters  
on a target page

To pass data from one page to another, you set up parameters on the target page to receive the data from a linking page. You can set up parameters on the linking page following the same procedure.

### ❖ **To set page parameters for a target page:**

- 1 Right-click a Web page in the HTML editor, then select Page Properties from the pop-up menu.
- 2 In the Page Properties dialog box, click the Parameters tab.
- 3 On the Parameters page, click the New button.
- 4 Under the Parameter Name column, type the name of the parameter.

(Optional) If your page is 4GL-enabled, type the default value for the parameter under the Default Value column, which is the value set for a parameter when a linking page does not pass a value for the parameter.



- 5 Repeat steps 3 and 4 for each parameter you want to add.
- 6 Click Apply.

## Setting parameter bindings on the linking page

In a Web application, you often pass data from a linking page to a target page that has page parameters defined for receiving the data. At design time, after you set page parameters on a target page, you can see those parameters in the user interface for a linking page. On the linking page, depending on the navigation style you select, you can specify the target for page navigation in a dialog box. The linking page must be 4GL-enabled.

There are three principle navigation styles: hyperlink, form submit, and server redirect. The user interface for binding parameters is different for each navigation style. To view and bind the target page parameters from the linking page using the different navigation styles, see “About page navigation” on page 193.



## Setting up page and session variables

You create page and session variables for use in server scripts. The variables for a 4GL JSP page are available to all server scripts, including events and blocks of server scripts used to generate a section of a page. The Web Target user interface makes it easy to bind variables from a linking page to parameters from a target page.

For more information about page variables and session variables, see “Managing page data” on page 140. For information about passing parameters using different navigation styles, see “About page navigation” on page 193.

### Variable properties

Table 9-4 shows the values you can assign to the properties of a variable:

**Table 9-4: Properties of variables**

Property	Value
Data Type	boolean, byte, char, double, float, int, long, short, or String
Life Time	Page or session
Client Access	None, read only, or read/write

A read/write variable lets users set a value on a page in their browser. A server action for the page returns the client-entered value to the server.

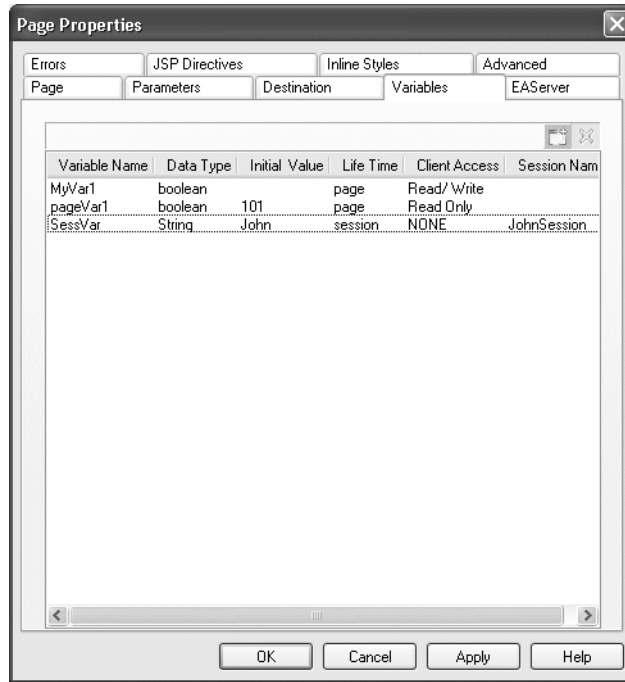
### Defining variables

When you define variables for a page, you can set the scope and the client access attributes for each variable.

❖ **To define page or session variables:**

- 1 Right-click in a 4GL JSP page open in the HTML editor, then select Page Properties from the pop-up menu.
- 2 In the Page Properties dialog box, click the Variables tab.

- 3 On the Variables page, click the New button, and then specify values for the variable.



- 4 Repeat step 3 for each variable you want to define.
- 5 Click Apply.

## Accessing EAServer components

4GL JSP pages provide tight integration with EAServer servers. Using 4GL JSP pages facilitates access to EAServer components, component properties, and component methods.

For a deployed 4GL JSP page, the connection to the EAServer server associated with the page is made before the page is loaded in a client browser. If component stubs are available to the page server, components accessed by the page get instantiated at load time.

## About EAServer integration

4GL JSP pages integrate EAServer components in several ways, by:

- Binding controls to the properties of an EAServer component to incorporate data returned from the EAServer component

For information about binding controls to the property of an EAServer component, see “Binding controls to properties of EAServer components” on page 179.

- Writing a server script that directly accesses an EAServer component

You can drag an EAServer component or a method on an EAServer component directly from the System Tree to a server script. For more information about adding EAServer methods to a page, see “Writing scripts to access EAServer components” on page 188.

- Writing a server script that manipulates a variable representing an EAServer component. For more information about working with variables that represent EAServer components, see “Making properties of EAServer components available for binding” on page 175 and “Writing scripts to access EAServer components” on page 188.

## Working with EAServer components

After you define an EAServer profile in PowerBuilder, the 4GL JSP Page wizard and the Components tab of the System Tree list the components available on the server. You can select components in the wizard, or drag components from the System Tree to your page to give your page access to the components.

### Making properties of EAServer components available for binding

Providing page access to a component lets you use it for data binding. Data binding lets you bind a property of an EAServer component to a page control, associating the control with the property value.

Properties available for binding comprise a standardized set of set and get methods. In general, components are available for binding if they display a get method that does not require arguments.

The 4GL JSP Page wizard lets you select components to add to your page. You can also defer adding components until after the page is created.

❖ **To make properties of EAServer components available for binding:**

- 1 Drag an EAServer component from the Components tab of the System Tree to the Page view of a 4GL JSP page in the HTML editor.

---

**Start EAServer**

The server must be running before you can see the server components in the System Tree.

---

The EAServer page of the Page Properties dialog box displays. This page supplies the component name, a default variable name (based on the component name), and a default scope (page) for the variable.



- 2 Change any of these values as needed and click OK.

Now the variable representing the component is available to your page.

For information about binding controls to EAServer component properties, see “Binding controls to properties of EAServer components” on page 179.

Getting information about EAServer components

After your page has access to an EAServer component, you can display information about the characteristics of that component on the Page tab of the System Tree (as well as on the Components tab). You can find the component under the EAServer Objects branch of the Server Side node of the Page tab. The Properties menu item on the pop-up menu for the component displays the properties of the component.

---

**Drag and drop from the Page tab**

If you drag an EAServer component from the Page tab of the System Tree to the integrated Script editor or the Source view, the name of the component is added to the open script or the source code. The Page Properties dialog box does not display as it does when you drag and drop the same component from the Components tab to the Page view.

---

## Setting up EAServer login variables

4GL JSP pages make it easy to set up client login for pages that access EAServer components. If your application prompts users for user name and password, you can bind this data to either a page or a session variable. By using a session variable, a user can log in once during a browser session, and then access multiple EAServer components from the server using the same user name and password.

❖ **To set up login variables:**

- 1 Right-click in a 4GL JSP page open in the HTML editor, then select Page Properties from the pop-up menu.
- 2 In the Page Properties dialog box, click the EAServer tab.
- 3 Click the New button on the EAServer page.

The cursor displays in a new line in the list box of components for the page. An ellipsis button displays on the same line.

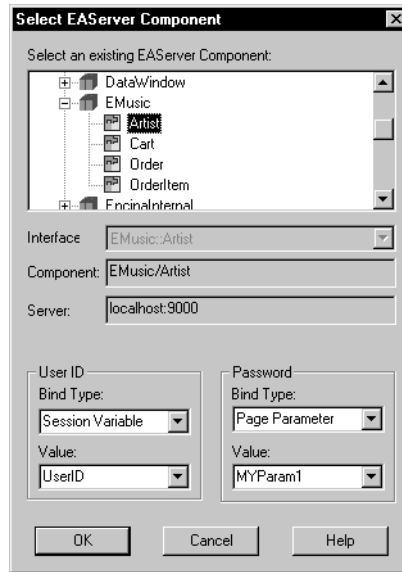
- 4 Click the ellipsis button on the new line under Component Name.

The Select EAServer Component dialog box displays.

- 5 Specify the user ID (user name) and password for the component, and the appropriate binding type for each value.

You can bind the User ID and Password for the component to a constant value, a page parameter, a page variable, or a session variable.

For more information about binding, see “About page navigation” on page 193.



## Adding controls

You add controls to 4GL JSP pages the same way you add them to other Web pages. When you add controls to 4GL JSP pages, however, the server scriptable property is turned on by default.

Server scriptable property

The **server scriptable** property does the following:

- Creates a server object that represents the control

For server scripts, you access the object as a page variable that has the same name as the control (which is why controls must have unique names). The control is added under the Server Side branch in the Page tab of the System Tree when the server scriptable property is selected.

- Supports binding to properties of EAServer components and other page data (from page parameters, page variables, and session variables)

**Supported controls**

4GL pages provide enhanced support for HTML form field controls. Several controls present special conditions on 4GL JSP pages:

- **Hidden text** This is a client-side control. It is not server scriptable, but is available for 4GL-enabled Web pages only.
- **Static text** The Static Text control is a specialized text field that can be manipulated by server scripts. The client cannot change the value of this text. The Static Text control is available on 4GL JSP pages only.
- **Check box** The value of a check box sent to the server from a 4GL JSP page is a boolean value (T or F). The Value To Send To Server field on the Checkbox Properties dialog box for a 4GL JSP page is grayed.
- **Standard button** A standard button on a 4GL JSP page can work like a submit button if you add a server redirect on the ServerAction event. However, if you code the client-side onclick event to return False, the ServerAction event will not be triggered.
- **Submit button** The client-side onclick event for a submit button is triggered only on a 4GL-enabled Web page. If you code this event to return False, a form submit (either to the current page in a self-link or to a URL that you select on the Destination page of the Page Properties dialog box) does not occur and the ServerAction event is not triggered.
- **Radio button** Radio buttons are different from other controls because they function as a group. The server scriptable property is either enabled or disabled for all buttons in the group. Each button in the group uses the same binding and has the same properties. If you make a change to a binding or a property for one button, the change takes effect for the others too.

## **Binding controls to properties of EAServer components**

Binding a server-scriptable control to the property of an EAServer component gives the control access to the data encapsulated by that property. It also automates the process of moving data to and from the component. For components that encapsulate data from a database, binding a property of that component to a control lets you quickly get data and update it.

Adding the component to a page

Before you can bind a control to the value of an EAServer component property, the component value must be available to your page. You can add a component to an existing page by selecting the component on the EAServer page of the Page Properties dialog box or by dragging and dropping it onto the page from the System Tree. The component must have at least one get method for a property that you want to bind to your control. The required get method must not include any arguments.

For more information about accessing a property for an EAServer component, see “Accessing EAServer components” on page 174.

Using component stubs at runtime

For the binding to work at runtime, component stubs must be available to the page server. Component stubs can be generated automatically from PowerBuilder.

---

### **Regenerating component stubs**

EAServer stubs are regenerated only if you select a full rebuild for your target deployment rebuild option. You can make this selection in the Deployment Configuration wizard or in the Deployment Configuration Properties dialog box. By default, the default rebuild option for a deployment configuration is incremental. If the incremental rebuild option is selected, the time it takes to deploy a 4GL target that uses EAServer components can be significantly reduced.

---

The directory containing the component stubs must be included in the classpath used by the page server. If the page server caches pages it generates, and if you modify a component and regenerate the stubs after loading a page from your Web server, you may need to stop and restart the Web server to see changes on the client side.

### **❖ To bind an EAServer component property to a control:**

- 1 Select Insert>Form Field, then select the type of control (such as Single Line Text) you want to bind to a component property.

The Properties dialog box for the control displays.

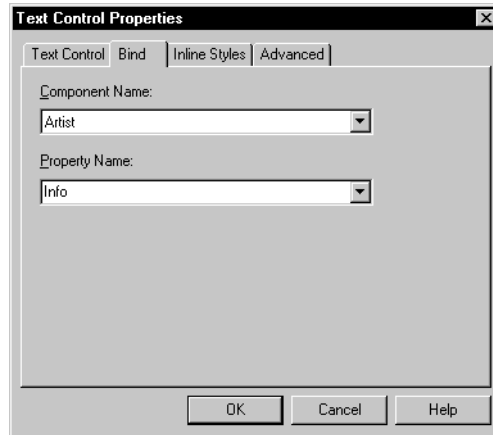
- 2 On the main page of the control properties dialog box, type the name of the control and other information as needed.
- 3 On the Bind page, select an EAServer component.

The list includes only EAServer components that have properties available for binding.



- 4 Select a component property.

The property name is the property that provides access to the component through a get method defined for it in EAServer. The get method must not take any arguments for it to appear in the Property Name drop-down list.

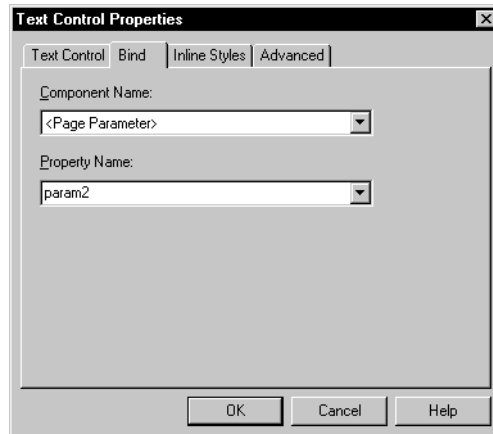


## Binding controls to page data

You can also bind a server-scriptable control to the value of a page parameter, page variable, or session variable. The Bind page in the Properties dialog box for each control lists the properties available for each of these components.

- ❖ **To bind page data (the value of a parameter or variable) to a control:**
  - 1 Select Insert>Form Field, then select the type of control (such as Single Line Text) you want to add to your Web page.  
The Properties dialog box displays.
  - 2 On the main page of the control properties dialog box, type the name of the control and other information as needed.
  - 3 On the Bind page, select Page Parameter, Page Variable, or Session Variable as the component you want to bind to the control.

- 4 Select the name of a parameter or a variable that appears in the property list:



## Disabling server scripting for a control

Disabling server scripting for a control makes the control unavailable for binding to input data. The control is inaccessible to server scripts and cannot pass data back to the server.

❖ **To disable server scripting for a control:**

- 1 Right-click the control on a page open in the HTML editor, then select Properties from the pop-up menu

*or*

Insert a new control.

The Properties dialog box for the control displays.

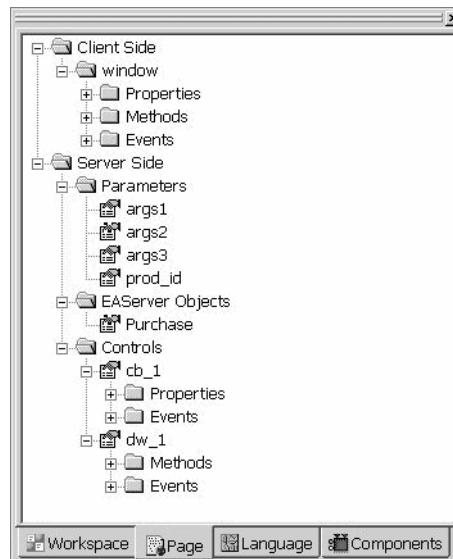
- 2 On the initial page, clear the Server Side Scriptable check box.

## Writing server scripts

When editing Web pages in Page view of the HTML editor, you use the integrated Script editor to add scripts for events appropriate to the context in which you are working. You add scripts for the server events and write other server scripts the same way you do for client event scripts. For more information about using the integrated Script editor, see Chapter 6, “Writing Scripts.”

### System Tree

The System Tree lists the objects, methods (including EA Server methods), properties, events, parameters, and variables you can access from server scripts:



You can drag any of these items from the System Tree and drop them onto the Script editor (or into Source view of the HTML editor). When you drop methods or properties into a script, the appropriate format for the call appears; you need only supply the arguments.

### psPage object

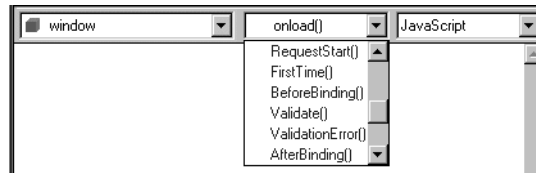
The psPage object represents a 4GL JSP page. It is a global object on the server that encapsulates the extensions to the Web Target object model and controls page processing for 4GL JSP pages. For information about page processing, see “How page request processing works” on page 189.

## Responding to events on your page

An event-driven architecture is the foundation for working with 4GL JSP pages. Writing scripts to respond to server events controls the data that displays on your Web page.

**Server events** In the events list of the Script editor, server events appear in blue text. You must enable the 4GL Web server-side event model to display these events in the events list (a 4GL JSP page is required). When you write a script to handle an event, an icon identifies which events have associated scripts.

Server events for a page appear in the events list for the window object; server events for a control appear in the events list for a control object. Here the events list shows a partial listing of the server events available for the page (window object):



**Client events** You can also write scripts for client-side JavaScript events. Client events appear in black text in the events list. You do not need to enable the 4GL mode to write scripts for these events.

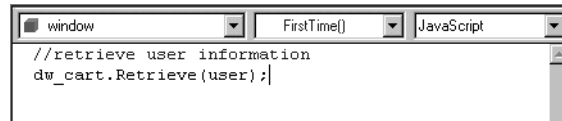
Summary of principal events for a page

For most 4GL JSP pages, you should add scripts to handle initialization, response to page controls, and validation.

**Table 9-5: Typical server-side events to script on 4GL JSP pages**

To do this	Write a script to handle this event
Initialize page the first time a user visits it	FirstTime
For pages that use self-navigation, initialize a page on subsequent visits during the same user session	BeforeBinding
Respond to an action that a user performed using a page control (such as clicking a button)	ServerAction
Validate a page	Validate and ValidationError

**Example 1: Initialize a page for a first visit** Here the page retrieves the data about a user and displays it in the Web DataWindow `dw_cart` when the page initializes:

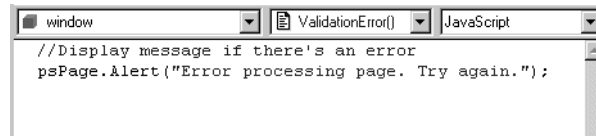


```

window
FirstTime()
JavaScript
//retrieve user information
dw_cart.Retrieve(user);

```

**Example 2: Validate page** If the `ValidationError` event is fired in response to a page validation error, the user sees the following message in an alert box:



```

window
ValidationError()
JavaScript
//Display message if there's an error
psPage.Alert("Error processing page. Try again.");

```

Summary of optional events for a page

The following events are available on 4GL JSP pages:

**Table 9-6: Server-side events for 4GL JSP pages**

This event	Occurs
RequestStart	At the very beginning of page processing, before server-side objects have been created and before any data binding or variable retrieval.
AfterBinding	After the controls have been bound to the input data and all validation has been done, but before any actions are performed.
BeforeAction	After data binding and validation and just before performing any action.
AfterAction	After all actions have been performed but before page generation.
BeforeGenerate	Before any generation happens. It is triggered both when the page is requested for the first time and when a self-navigation is done.
AfterGenerate	When all generation has taken place.
RequestFinish	After all generation is complete. It is the last event to occur on the page.
ServerError	When the <code>ReportError</code> method is called. It can be used to alert you when something goes wrong during processing.

**psDocumentWrite**

HTML generation occurs after the BeforeGenerate event. Do not place psDocument.Write in a script before this event. The appearance of the resulting page would be unpredictable.

---

Events for page controls

4GL JSP pages also provide events for the various types of controls.

**Table 9-7: Server-side events for controls on a page**

For these controls	These events are available
SingleLineText	Validate, ValidationError, ItemChanged
TextArea	Validate, ValidationError, ItemChanged
RadioButton group	ItemChanged
ListBox	ItemChanged
PushButton	ServerAction
CheckBox	ItemChanged
StaticText	ServerAction
DataWindow	AfterAction, AfterRetrieve, AfterUpdate, BeforeAction, BeforeRetrieve, BeforeUpdate, OnDBError, Validate, ValidationError

For a description of server-side events on Web DataWindows, see the *DataWindow Reference*. For a description of the server-side events on other controls on 4GL pages, see the *JSP Target Reference*.

In addition to viewing the events available for a control from the events list in the integrated Script editor, you can expand a control on the Page tab page of the System Tree to see a list of events for that control.

❖ **To view a list of events available for a control:**

- 1 On the Page tab page of the System Tree, click the name of the control.
- 2 Expand the item for the control, then expand its Events folder.

## Adding scripts to 4GL JSP pages

The extensions to the Web Target object model give you other ways to customize a page by writing scripts to access:

- Properties and methods for the psPage object
- Methods for objects that represent controls

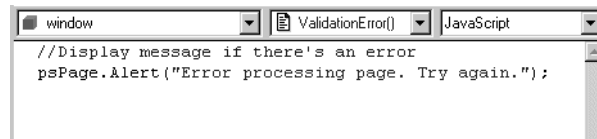
## Properties and methods of the psPage object

The psPage object represents an entire 4GL JSP page. You can add properties as well as methods for the psPage object to your page by dragging them from the System Tree, dropping them into the appropriate place in the Script editor (or in the Source view of the HTML editor), then defining arguments.

For a list and description of psPage properties and methods, see the *JSP Target Reference*.

All psPage methods (except Redirect) help fine-tune error reporting for your page. The psPage Alert method lets you display a client-side Alert box to make sure that users of your Web application see important messages. If you use the Alert method to inform users about validation errors, it lets them correct entries that are not in the correct syntax.

Here is how you can use the Alert method in a script in response to a ValidationError event:



```

window | ValidationError() | JavaScript
//Display message if there's an error
psPage.Alert("Error processing page. Try again.");

```

## Adding scripts for properties of controls

Objects for controls also have associated properties that you can access in server-side scripts:

**Table 9-8: Typical properties for controls on a 4GL JSP page**

Property	Description
name	The name of the control. This is a read-only property.
value	The label for the control.
visible	Sets whether or not the client control is generated. If not visible, there is no access to the client control.
enabled	Sets whether or not the control allows focus. (This property works only in browsers that support the DISABLED attribute.)

Typically you set values for object properties on the property pages for the control rather than in server scripts.

## Using the psPage prefix

When referring to read-write variables in script for client-side events, it is best to include the psPage prefix before the variable name. Otherwise, client-modified values might not be passed on to a target page; initial values are passed if the prefix is not included in the script. Page parameters cannot be accessed in client-side script. You can optionally use the psPage prefix for the names of controls on the page. In server-side scripts, you must use the psPage prefix for the names of controls on the page.

**Example 1: Client-side code** This script in a client-side onchange event sets the *v1* read-write variable to a value the client enters in the *sle\_1* text box (an alert message should not be prefixed with *psPage* on the client side):

```
alert("This is the client-side onchange event");
psPage.v1=psPage.sle_1.value;
```

**Example 2: Server-side code** This same script in a server-side event (such as *ItemChanged* or *ServerAction*) should also include a *psPage* prefix for controls:

```
psPage.Alert("This is a server-side event");
v1=psPage.sle_1.value;
```

Although page controls require the *psPage* prefix, parameters and variables do not require the *psPage* prefix.

For more information on scripting, see Chapter 6, “Writing Scripts,” and Chapter 7, “Working with Application Servers and Transaction Servers.”

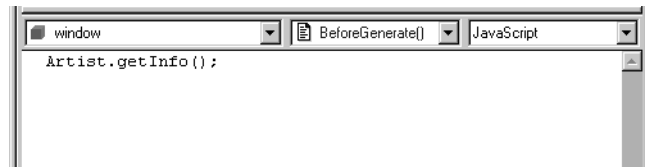
## Writing scripts to access EAServer components

4GL JSP pages provide ready access to EAServer components from server scripts. You can drag and drop a method for an EAServer component into a script or access a variable for an EAServer component.

Adding EAServer methods to server scripts

You can drag any component method visible in the System Tree to a server script.

The following illustration shows the result of dragging the *getInfo* method of the *Artist* component to the integrated Script editor in the script for the *BeforeGenerate* server-side event. The method is called when the *BeforeGenerate* event is triggered for the page.



If you drag and drop a method that requires arguments, you simply type the arguments in the Script editor.

For how to set up access to an EAServer server to view components and component methods installed on a server, see “Working with EAServer components” on page 175.



Manipulating variables that represent EAServer components

Scripts can also access EAServer components represented as variables on your page. Whenever you drag an EAServer component from the System Tree and drop it on your page, the component is available as a variable. The variable name is the same as the component name unless you change it.

❖ **To view the EAServer variables for your page:**

- 1 Right-click in a 4GL JSP page open in the HTML editor, then select Page Properties from the pop-up menu.
- 2 In the Page Properties dialog box, click the EAServer tab.

On the EAServer page, you see the list of EAServer components and the associated variables.

## How page request processing works

When processing a 4GL JSP page, the Web Target object model calls events for the psPage object in a specified order. The psPage global object controls the event model and the communication between pages.

Sequence for processing pages

When a browser requests a page, the JSP application server processes a 4GL JSP page in this order:

- 1 Starts page processing
  - Initializes and restores page variables and parameters
  - Creates the server object model
- 2 Performs page-specific processing
  - Validates the page and binds input data if data is submitted to the current page (typically in preparation for redirection to another page)
  - Binds values for EAServer component properties to controls
- 3 Generates the new page

The following tables summarize the processing of 4GL JSP pages and show the order in which server-side events are triggered during that processing:

**Table 9-9: Order of events at the start of page processing**

Page processing	Events
Initiate page request	psPage.RequestStart
First page request	psPage.FirstTime
Reload page with data submitted to the reloaded page (typically in preparation for redirection to another page)	psPage.BeforeBinding

**Table 9-10: Order of events that perform page-specific processing**

Page processing	Events
A control had a specified value on request and the value of that control changed	control.Validate
The validation fails	control.ValidationError
The validation succeeds	control.ItemChanged
Data binds to component properties; sets the value of the property of a corresponding EAServer component (for each server object)	none
Validate page	psPage.Validate
The page or any control on the page fails validation	psPage.ValidationError
Binding and validation complete	psPage.AfterBinding
Page control initiates a form submit	psPage.BeforeAction control.ServerAction psPage.AfterAction
For a first request or for pages that require data binding, sets the value of the property of a corresponding EAServer component (for each server object)	none

**Table 9-11: Order of events that generate the new page**

Page processing	Events
If the script does not redirect to another page, process the page template as before. Include other server scripts.	psPage.BeforeGenerate
If the script does not redirect to another page, generation is complete.	psPage.AfterGenerate
Generation is complete. This event is always triggered.	psPage.RequestFinish

## Disabling 4GL mode

Usually there is no reason to disable 4GL mode for a page. If you do so, you should be aware of the impact of this change.

If you turn off 4GL mode on the Page page in the Page Properties dialog box, the page cannot use any of the processing done by the 4GL JSP pages. In this case, the events do not produce any action because the server objects representing the controls are not available.

Server event scripts that you created remain in the source file. If you want to remove those scripts, you must edit the source file and delete them. You must also edit the source to place FORM tags around existing fields or controls whose values you want to submit to a different page.



About this chapter

This chapter describes how you can link one page to another and pass data from one page to the next in 4GL JSP pages and non-4GL JSP pages.

Contents

Topic	Page
About page navigation	193
Managing client hyperlinks	196
Managing client form submission	197
Managing server redirection	200

## About page navigation

One of the major tasks when creating a set of Web pages is managing how one page links to another and how data is passed from one page to the next.

Navigation styles

You can link from one page to another using any of three navigation styles:

- A hyperlink using the HTML `<A>` element
- A form submit for the page
- A server redirection using self-navigation (where the form representing the current page submits back to itself)

All three styles let you bind parameters to target pages, but the level of support for binding varies. Which navigation style you choose depends on how you want data passed, whether or not the data requires processing by the server, and the flexibility you need. Most 4GL JSP pages navigate to other pages by server redirection, which provides the most flexible way to navigate from one page to another. Server redirection can also be used to navigate from non-4GL JSP pages, but more coding is required. Table 10-1 next shows the advantages and disadvantages of the three navigation styles.

**Table 10-1: Navigation style advantages and disadvantages**

Navigation	Advantages	Disadvantages
Hyperlink	<ul style="list-style-type: none"> <li>• Jumps directly to another page without server processing</li> <li>• Works well with a range of different target pages when processing is unnecessary before moving to the target page</li> <li>• Allows non-matching names for parameter binding (binding defined in HREF attribute in page source)</li> </ul>	<ul style="list-style-type: none"> <li>• Requires the values of parameters to be available when the page is generated.</li> <li>• Does not return to the server to process current page before moving to the next page. (Not all parameters and variables are available to the target page.)</li> <li>• Does not support server actions.</li> <li>• Does not support validation before navigation.</li> <li>• Does not allow the user to recover from errors caused by entering data in an invalid format.</li> </ul>
Form submit	<ul style="list-style-type: none"> <li>• Jumps directly to another page without server processing</li> </ul>	<ul style="list-style-type: none"> <li>• Requires the target page to define parameter names to match the parameters passed.</li> <li>• Provides only the data from the fields on the page form.</li> <li>• Does not support validation before navigation.</li> </ul>
Server redirect	<ul style="list-style-type: none"> <li>• Supports validation and server actions</li> <li>• Processes the values of parameters before passing them to a target page</li> <li>• Allows non-matching names for parameter binding (binding defined in server script)</li> <li>• Retains the values of parameters and makes all the data available for parameter binding</li> <li>• Lets you redirect the client to another page from the middle of a server script</li> <li>• Lets you stop navigation if validation fails</li> </ul>	<ul style="list-style-type: none"> <li>• Loads a current page, then the target page in the user's browser (making this method slower than the other two styles).</li> <li>• Does not allow you to add FORM elements to a 4GL Web page. (A FORM represents an entire 4GL Web page.) You can still insert form field controls, but they are not surrounded by FORM tags in Source view.</li> </ul>

**Target pages** When you develop Web pages, you can specify a target page that already exists, or one that you plan to create. If you specify a page that already exists, you can bind values from the linking page to the parameters of the target page. If you specify a page that does not yet exist, you can set parameters on the linking page; then, when you create the target page, you specify which parameters it requires of those being passed from the linking page. For more information about page parameters, see “Setting up page parameters” on page 171.

**Parameter binding** When you establish navigation from one page to another, you can bind values from the linking page to parameters on the target page. 4GL Web pages support several types of parameter bindings.

**Table 10-2: Parameter binding for 4GL JSP pages**

Type of binding	Description	Navigation style restrictions
Constant	A fixed value	Not available for form submit.
Page Variable	A variable whose value is set on the server	Only initial value used by hyperlink (cannot be changed by client). Variable name must match parameter name on target page for form submit.
Page Parameter	A parameter whose value is set on the server	Only initial value used by hyperlink (cannot be changed by client). Parameter name must match parameter name on target page for form submit.
Expression	The value of an expression set on the server	Not available for form submit.
Control	The value property of a server object	Not available for hyperlink.

The user interface for binding parameters is different for each navigation style:

**Table 10-3: Information about navigation style and parameter binding**

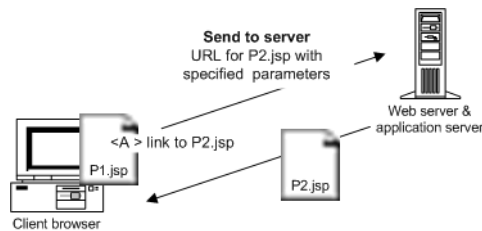
Navigation style	For information, see
Hyperlink	"Managing client hyperlinks" next
Form submit	"Managing client form submission" on page 197
Server redirect	"Managing server redirection" on page 200

## Managing client hyperlinks

For client hyperlinks, an HTML anchor `<A>` element directly links one page to another. On the linking page, you specify the parameters (including the type of binding and the value for the parameter) you want to pass to parameters on the target page.

The following drawing illustrates how parameters are made available to a target page from a hyperlink on a linking page in a JSP target:

**Figure 10-1: Passing parameters in a hyperlink**



The server sets the parameters sent to the target page. Only initial values are passed to the target parameters; client-entered values are not processed.

❖ **To pass parameters with a client hyperlink:**

- 1 Drag an HTML A element from the Language tab of the System Tree to a 4GL JSP page in the HTML editor (Page or Source view)

*or*

Click the hyperlink tool button in the painter bar for the HTML editor with a 4GL JSP page open in Page or Source view.

The Hyperlink Properties dialog box displays.

- 2 On the Hyperlink page, specify link information, including a target page destination.  
Make sure the Server-Side Scriptable check box is selected.
- 3 On the Parameters page, specify page parameters for the target page, the type of binding for each parameter, and the value from the current page that you want to bind to each parameter.

---

**If not a 4GL JSP page**

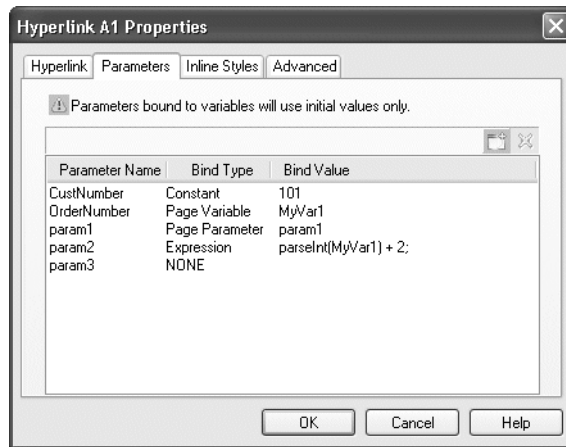
If the linking page is not a 4GL JSP page, the Parameters tab is grayed.

---



If the linking page is a 4GL JSP page and the target page exists and has page parameters defined for it, these parameters display automatically under the Parameter Name column. They are available for binding to constants, expressions, page parameters, and page variables from the current page.

You can add new parameters to the list and define them later in the target page. Parameters and variables on the current page automatically display under the Bind Value column when you select a bind type of Page Parameter or Page Variable, respectively.



- 4 Click OK after you finish setting the hyperlink properties.

For information about setting hyperlinks on DataWindow objects and columns, see Chapter 11, “Using the Web DataWindow Design-Time Control.”

## Managing client form submission

When a form submits directly to another page, the data passed to the target page must already be available from the linking page. The following data is available to the target page:

- Values for server-scriptable controls
- Client-side parameters (including parameter values set on the client)

On 4GL JSP pages, a *form represents a page*. When a user action submits a page form, the form can submit to the same page (self-navigation) to refresh a page in the user's browser or to prepare for a server redirection. Otherwise, it can be submitted directly to another page (form submit). For more information about server redirection, see "Managing server redirection" on page 200.

For a form submit, the names of the client controls must map to (match) the names of the parameters on the target page. To take advantage of the Web Target interface for the form submit navigation style, you must first set the parameter names on the target page.

---

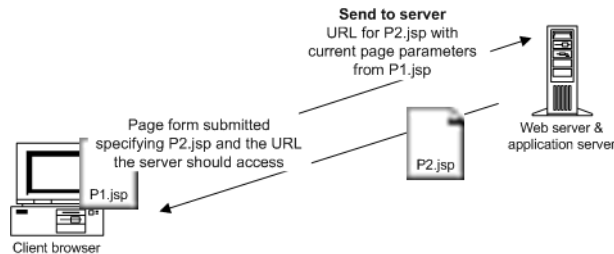
### No forms on 4GL JSP pages

You should not add FORM tags to 4GL JSP pages. If you change a non-4GL page to 4GL mode, you must manually remove any FORM tags in Source view. For 4GL JSP pages, the entire page is represented as a single form.

---

The following drawing illustrates how parameters get passed when a page form submits directly to another form in a non-4GL JSP target:

**Figure 10-2: Passing parameters in a form submit**



For a fuller description of how to pass parameters using the form submit navigation style, see "Using parameters and variables" on page 169.

❖ **To set parameter bindings for a 4GL form submit:**

- 1 Right-click the linking page in the HTML editor, then select Page Properties from the pop-up menu.

The Page Properties dialog box displays.

- 2 Click the Parameters tab  
*or*  
Click the Variables tab.

- 3 Click the New button, and add a parameter or variable with the same name as a parameter on the target page.

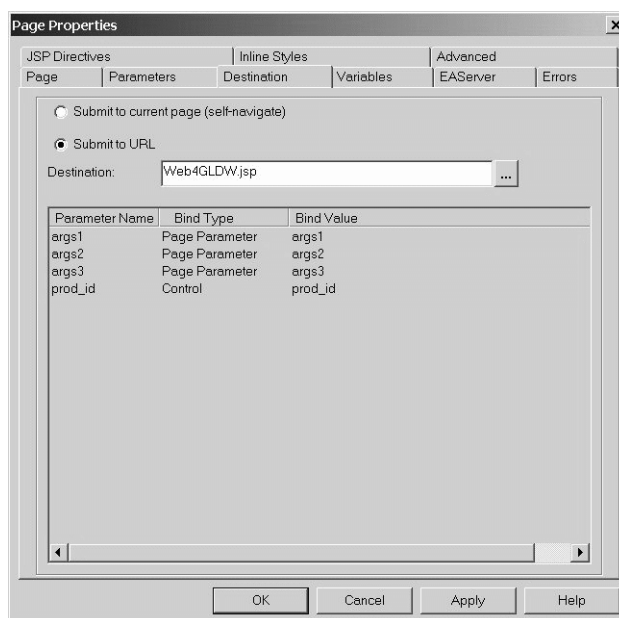
For parameter binding purposes, the names you type are case sensitive. The parameter or variable name on the linking page must exactly match the parameter name on the target page.

To add a parameter, see “Setting up page parameters” on page 171. To add a variable, see “Setting up page and session variables” on page 173.

- 4 Repeat steps 2 and 3 for each parameter and variable you want to add to the current (linking) page.
- 5 Click the Destination tab on the open (linking) page.
- 6 Select the Submit To URL radio button option. Click the browse (...) button and select your target page from the Choose URL dialog box.

The parameters of the target page are listed in the Parameter Name column of the grayed list box. You cannot directly modify the items in this list.

If a parameter or variable name on the current (linking) page matches a parameter on the target page, the matching name is listed in the Bind Value column. The Bind Type indicates whether the matching name is a page parameter or page variable on the current page.



7 Click Apply.

## Managing server redirection

4GL JSP page navigation

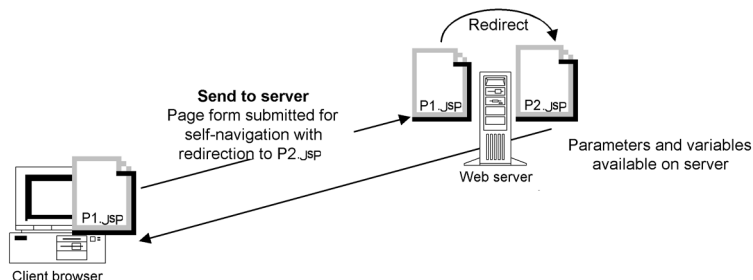
4GL JSP pages provide added support for navigation between pages. Server redirection is the navigation style of choice when a parameter value passed to another page relies on user input, or when you want to validate user input. The 4GL JSP page generates the server script that specifies the target page and the values that get passed to parameters on the target page.

A server redirection can initiate a link to another page from anywhere in a server script. Typically you add a Redirect call in a ServerAction event for a command or picture button, or in the AfterAction event for the psPage object or a Web DataWindow control that you place on the page.

All server data is available for parameter binding, because the redirection occurs in the server script. The page gets processed by the server, enabling the page to take advantage of all variables available there.

The following drawing illustrates how parameters get processed in a 4GL JSP application when a server redirection displays another page. Parameters are passed the same way for JSP targets, except for the application server name. Also, the target and linking pages in a JSP application would typically have *.jsp* extensions instead of the *.htm* extensions displayed in the drawing.

**Figure 10-3: Passing parameters in a server redirect**



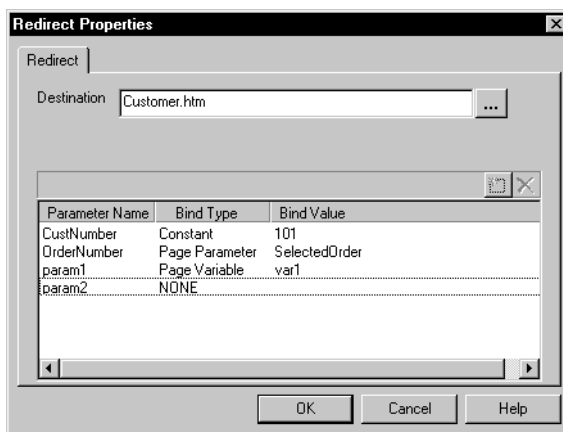
Setting up a server redirection

You can set up a server redirection from the integrated Script editor by selecting a server event, then specifying the target page and the parameters to pass to that page. The script is created for you.

❖ **To set up a server redirection from the integrated Script editor:**

- 1 In a server script or in the script for a control with a server action event selected in the events list, right-click in the Script editor.
- 2 Select Insert Redirect from the pop-up menu.
- 3 In the Redirect Properties dialog box, specify the target file or URL.
- 4 Specify the target page parameters, the type of binding for each parameter, and the value you want to bind to the parameter.

If the target page exists with input parameters identified for it, these parameters appear in the list of parameters on the Redirect page. You can edit these parameters and add new ones.



- 5 When you finish setting the Redirect Properties, click OK.

The Script editor inserts a block of code. You can modify or remove this block of code using the pop-up menu.

#### Changing parameter bindings

You can change the binding for parameters from the Redirect Properties dialog box.

❖ **To change parameter bindings or parameter values for a server redirection:**

- 1 Right-click in the Redirect code in the integrated Script editor.
- 2 Select Edit Redirect from the pop-up menu.
- 3 In the Redirect Properties dialog box, make any changes needed to the parameter definitions.



# Using the Web DataWindow Design-Time Control

About this chapter

This chapter describes the Web DataWindow design-time control (DTC) you can add to your JSP target pages.

Contents

<b>Topic</b>	<b>Page</b>
About the Sybase Web DataWindow DTC	203
Adding a DataWindow to a JSP page	207
Setting Web DataWindow DTC properties	213
Editing existing Web DataWindow DTC properties	222
DataWindow presentation styles and data sources	223
Binding data to DataWindow retrieval arguments	224
Defining hyperlinks on objects in a DataWindow	230
Custom tag library for the Web DataWindow	232

## About the Sybase Web DataWindow DTC

The Web DataWindow DTC (design-time control) lets you add database-driven content to your Web applications.

### Web DataWindow support

Thin-client implementation

The Web DataWindow DTC provides support for the features available in the Web DataWindow, a thin-client DataWindow implementation. As a thin client, the Web DataWindow does not require any runtime components on the client—only a standard Web browser.

The Web DataWindow has three types of implementation:

- **XML Web DataWindow** Separate XML (content), XSLT (layout), and CSS (style) with a subsequent transformation to XHTML
- **XHTML Web DataWindow** XHTML content only
- **HTML Web DataWindow** HTML content only

Security setting requirements

The Web DataWindow DTC is an ActiveX control. To use the Web DataWindow DTC, you must be able to run ActiveX controls on the development machine and script ActiveX controls marked safe for scripting.

You can change your machine's ActiveX security settings in the Internet Options dialog box that you access from the Control Panel or from the Tools menu of the Internet Explorer browser. Changing the Default Level settings to Medium is all that is required to enable you to use the Web DataWindow DTC.

Since the DTC is used only at design time, client security settings do not require modification. Web DataWindow pages are still visible to client browsers with High security settings.

Use existing DataWindow objects

The Web DataWindow DTC lets you create Web applications that include DataWindow objects created in PowerBuilder or InfoMaker. When you add a Web DataWindow DTC to a Web page, the Sybase Web DataWindow DTC Properties dialog box displays.

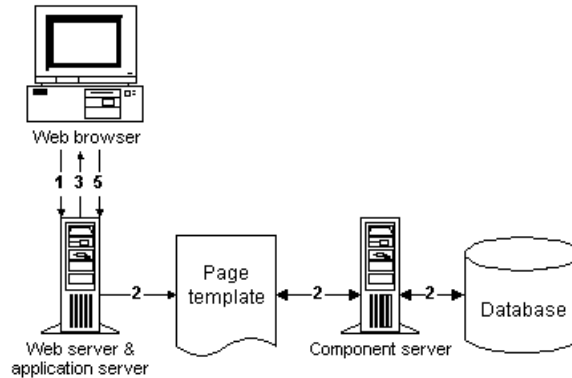
From this dialog box you can select the DataWindow object you want, the location and connection information required by the server, and bindings for any retrieval arguments. The script for accessing the Web DataWindow is automatically inserted in your page.



What happens in a Web application

Here is what happens when a client browser requests a page with an XHTML or HTML Web DataWindow on it:

**Figure 11-1: Architecture required for use with Web DataWindow**



- 1 The user's browser requests the URL for the page template.
- 2 Server-side scripts in the template run, calling the server component methods that generate the Web DataWindow code.
- 3 The Web page with the Web DataWindow is delivered to the browser.
- 4 The user interacts with the DataWindow.
- 5 Actions by the user cause the URL, with added action parameters, to be sent to the server. The actions are communicated to the server component, which causes modifications to the regenerated DataWindow, and the cycle continues again with step 2.

---

**When a client browser requests a page with an XML Web DataWindow**

When an XML Web DataWindow is generated, the server-side and client-side processes are more complex than with XHTML or HTML generation. For information about what happens when a client browser requests a page with an XML Web DataWindow on it, see the *DataWindow Programmer's Guide*.

---

## Server-side environment

The Web DataWindow works with a server component hosted in EAServer. The Web DataWindow retrieves and manipulates data from your enterprise database. When you create a DataWindow object, you can set properties that determine how the DataWindow object appears in the browser. The server component can generate XML, XSLT, and CSS (with subsequent transformation to XHTML), XHTML directly, or HTML for your DataWindow object.

Additional EAServer support

When you set up EAServer, the DataWindow HTMLGenerator110 is installed for you as a default server component. It contains many methods that you can display on the Components tab of the System Tree and invoke in server-side method calls.

You can also create a custom DataWindow server component and deploy it to your component server for greater flexibility. However, if your custom component does not implement the HTMLGenerator110 interface, you cannot use the Sybase Web DataWindow DTC. (You must manually code the DataWindow connection information in a server script on your page.)

You can use the Web DataWindow Container component wizard to create a custom component that implements the HTMLGenerator110 interface. The main advantage of the wizard-created component is that it deploys all the DataWindow objects in a selected target with the connection information encoded in the component.

For information on creating and deploying a custom server component or a Web DataWindow Container component, see the *DataWindow Programmer's Guide*. For information on DataWindow methods, see the *DataWindow Reference*.

## Benefits of using the Web DataWindow DTC

The Web DataWindow DTC provides an easy way to access a database from a Web page. The DTC offers the following benefits:

- **Enhanced productivity** Using the Web DataWindow with the Web DataWindow DTC decreases the amount of code you need to write. A Web DataWindow DTC generates the scripts that access a DataWindow server component, as well as the scripts and HTML to render the page. It also generates the scripts that allow the server component to access a database.
- **Reusability** You can use a DataWindow object in as many Web pages as you like.
- **Ease of maintenance** Whenever the data-driven requirements for a Web page change, you do not need to rewrite the server scripts in the page; instead you can simply modify the DataWindow object and update the properties for the DTC if needed.

## Adding a DataWindow to a JSP page

You can use the JSP DataWindow Page wizard to create a new JSP page with a DataWindow on it. For existing pages, you add a DataWindow to your page using the Web Target toolbar or the Insert>Form Field menu in the Page view or Source view of the HTML editor. You can also drag and drop a DataWindow (or a Web DataWindow Container component) onto a Web page in the HTML editor.

If you want your page to access a Web DataWindow component on EA Server, both the wizard and the property pages give you quick access to a list of the components available.

## Creating a page that has a Web DataWindow DTC

- ❖ **To create a new page with a Web DataWindow DTC:**
  - 1 In an open JSP Target workspace, select File>New from the menu bar.
  - 2 Click the JSP tab of the New dialog box.
  - 3 Double-click the JSP DataWindow Page wizard icon.

- 4 Follow the instructions in the wizard to complete the entries required.

---

**The new page is not 4GL-enabled**

If you want to enable 4GL processing for a new JSP DataWindow page, you need to select the Enable 4GL Web Server Side Event Model check box in the Page Properties dialog box for the page after you create it.

---

❖ **To insert a new Web DataWindow DTC in an existing HTML page:**

- 1 In Page view or Source view of the HTML editor, put the insertion point where you want the control to appear.
- 2 Select Insert>Form Field>DataWindow from the menu bar  
*or*  
Click the DataWindow button on the Insert toolbar.

---

**Drag and drop**

You can also drag the Sybase Web DataWindow DTC control from the Components tab of the System Tree to an open page in the HTML editor Page view or Source view. You can find the control under the ActiveX Controls>Web Design-Time Controls branch on the Components tab.

---

The Sybase Web DataWindow DTC Properties dialog box displays.

- 3 Specify Web DataWindow DTC properties by making the following selections:

Property selection	For information
Source for the DataWindow object	See “Selecting the source for a DataWindow object” on page 213
Database profile	See “Selecting a database profile” on page 215
Web DataWindow generator	See “Selecting a Web DataWindow generator” on page 221

Specifying properties also typically involves taking these actions:

Typical actions	For information
Binding data to DataWindow retrieval arguments	See “Binding data to DataWindow retrieval arguments” on page 224
Defining hyperlinks on objects in a DataWindow	See “Defining hyperlinks on objects in a DataWindow” on page 230

Typical actions	For information
Changing presentation details stored in the DataWindow definition	See “DataWindow presentation styles and data sources” on page 223
Scripting client- or server-side events on the DataWindow	See “Choosing an object or event for scripting” on page 124

4 Do one of the following:

- **In Page view** Click OK in the Sybase Web DataWindow DTC Properties dialog box.
- **In Source view** Click OK in the Sybase Web DataWindow DTC Properties dialog box and then click OK to close the Edit Design Time Control dialog box.

---

#### Using the Edit Design Time Control dialog box

You can do any of the following in the Edit Design Time Control dialog box:

- Click OK to insert the control in the page
  - Click Cancel to cancel the insert operation
  - Click Properties to redisplay the Sybase Web DataWindow DTC Properties dialog box
- 

## What you see in Page view

The labels from the header band of the DataWindow object that you select in the Web DataWindow Properties dialog box (or in the Web DataWindow wizard) display in Page view. If you did not specify a DataWindow object, you see only an empty box—with a title based on the default DataWindow control name to represent the Web DataWindow.

If you saved data in your DataWindow object, you can also see data in Page view—unless you use a Web DataWindow Container component as the source for your DataWindow object.



Artist	Name	Year
Ace of Base	The Sign	1993
Beatles	Abbey Road	1965

When you deploy the page, the generated source HTML and script is passed to the server.

## What you see in Source view

When you insert a Web DataWindow DTC in a Web page, the following text is added to the page source between two METADATA comments:

- An `<OBJECT>` element that embeds the Web DataWindow DTC ActiveX in the page. The control provides the information to manage the server component that generates the client control. The OBJECT element has parameters that keep track of various kinds of information about the DTC definition.

For example, the OBJECT SourceFileName parameter specifies the PBL, PSR, or SRD file that contains the DataWindow object definition. This value is set to an empty string if a Web DataWindow Container component is the DataWindow source.

- Server script generated by the DTC that provides logic for accessing the Web DataWindow server component. To allow you to target multiple application servers from the same source page, platform-independent code is generated that takes advantage of the capabilities of the Web Target object model.

The OBJECT element in the METADATA comment is required only when you are authoring the page; it is not needed at execution time. However, when the application server processes the page, it executes the generated server script on the page and returns the resulting HTML to the Web browser.

**Working with the generated text**

Typically, you do not modify any of the text generated for the Web DataWindow DTC. Changes you make are lost the next time you modify the properties of the control and regenerate the text.

**What you see in a non-4GL JSP page**

You can see the generated source for your page in the Source view of the HTML editor. The following example shows the OBJECT element (and some of the PARAM tags) added to the Source view for a Web DataWindow DTC in a non-4GL JSP page. They are wrapped in a METADATA comment:

```
<!--METADATA TYPE="DesignerControl" startspan
<OBJECT classid=CLSID:C77CAE91-7F08-11D2-BF7B-00C04F79FC8E height=171 id=OBJECT1
<PARAM NAME="SourceFileName" VALUE="D:\Program Files\eMusic\eMusic.pbl"></PA
<PARAM NAME="DataWindowObject" VALUE="d_cdlist"></PARAM>
<PARAM NAME="StandaloneDataWindow" VALUE=""></PARAM>
<PARAM NAME="AbsoluteLibPath" VALUE="0"></PARAM>
<PARAM NAME="DataConnection" VALUE="eMusic_pb"></PARAM>
<PARAM NAME="EnableWeightSettings" VALUE="0"></PARAM>
<PARAM NAME="EnableNameOverride" VALUE="0"></PARAM>
<PARAM NAME="EnableRowsOverride" VALUE="0"></PARAM>
<PARAM NAME="AllowDataEntry" VALUE="0"></PARAM>
<PARAM NAME="Events" VALUE="1"></PARAM>
<PARAM NAME="Validation" VALUE="1"></PARAM>
<PARAM NAME="Formatting" VALUE="0"></PARAM>
```

The code for the DTC includes source and connection information. It calls the Generate method on the server component inside a server script immediately following the closing OBJECT tag and just before the closing METADATA comment:

```
<PARAM NAME="UserID" VALUE="jagadmin"></PARAM>
<PARAM NAME="Passwd" VALUE=""></PARAM>
<PARAM NAME="ComponentName" VALUE="DataWindow/HTMLGenerator100"></PARAM>
<PARAM NAME="OneTrip" VALUE="0"></PARAM>
<PARAM NAME="EnableJaguar" VALUE="1"></PARAM>
<PARAM NAME="ProfileName" VALUE="local"></PARAM>

</OBJECT>
-->
<%
 jagConn = new PSJaguarConnection("myhost:9000", "jagadmin", "", "DataWindow
dwSource = new PSDataWindowSourceClass("eMusic.pbl", "d_cdlist");
dbConn = new PSConnectionParmsClass("ConnectionString='DSN=eMusic;UID=dba;PWD=sc
webDW = new PSDataWindowClass("webDW", false, jagConn, dwSource, dbConn);
webDW.RetrievalArgs[0] = psDocument.GetParam("category");

 webDW.Generate();
%>
<!--METADATA TYPE="DesignerControl" endspan-->
```

## What you see in a 4GL JSP page

When placed on a 4GL JSP page, the Web DataWindow DTC becomes an object of type PSWebDataWindowClass (an extension to the Web Target object model) with its own server-side methods and events. These are listed under the Web Target object model node on the Language tab of the System Tree.

The OBJECT element on a 4GL JSP page includes a special CREATE attribute that sets the source and connection information for the DataWindow object (similar to the code that is generated inside a server script on a non-4GL JSP page):

```
<!--METADATA TYPE="DesignerControl" startspan
<OBJECT classid=CLSID:C77CAE91-7F08-11D2-BF7E-00C04F79FC8E height=171 id=OBJECT1
name=webDW width=658 PSSERVERSCRIPTABLE="true" Create="
var jagConn = new PSJaguarConnection("myhost:9000";, "jagadmin
var dwSource = new PSDataWindowSourceClass("D:/Program Files/eMusic/eMus
var dbConn = new PSConnectionParmsClass("ConnectString='DSN=eMusic;UID=db
var webDW = new PSWebDataWindowClass(name, false, jagConn, dwSource, dbConn);
webDW.bAutomaticRetrieve = true;
webDW.addArgument(DYNAMIC_VALUE_PAGE_VAR, "category");
return webDW; ">
<PARAM NAME="SourceFileName" VALUE="D:\Program Files\eMusic\eMusic.pbl"></PA
<PARAM NAME="DataWindowObject" VALUE="d cdlist"></PARAM>
```

The server script generated on a 4GL JSP page calls only the GenerateXMLWeb method or the GenerateXHTML method on the server component:

```
<PARAM NAME="ProfileName" VALUE="remote"></PARAM>
</OBJECT>
-->
<%
dw_1.Generate();
%>
<!--METADATA TYPE="DesignerControl" ends-->
```

On the Page view of the HTML Editor, you can code server-side events (in 4GL JSP mode only) from the integrated Script editor. After you select a Web DataWindow control in the first drop-down list in the Script editor, you can select a server-side event in the second drop-down list. Server-side events display in blue, and client-side events display in black.



## Using Web Target object model classes

The Sybase Web DataWindow DTC uses the following classes of the Web Target object model to set up the database and component server connections:

- PSConnectionParmsClass
- PSDataWindowClass
- PSDataWindowSourceClass
- PSJaguarConnection

For more information, see the *JSP Target Reference*.

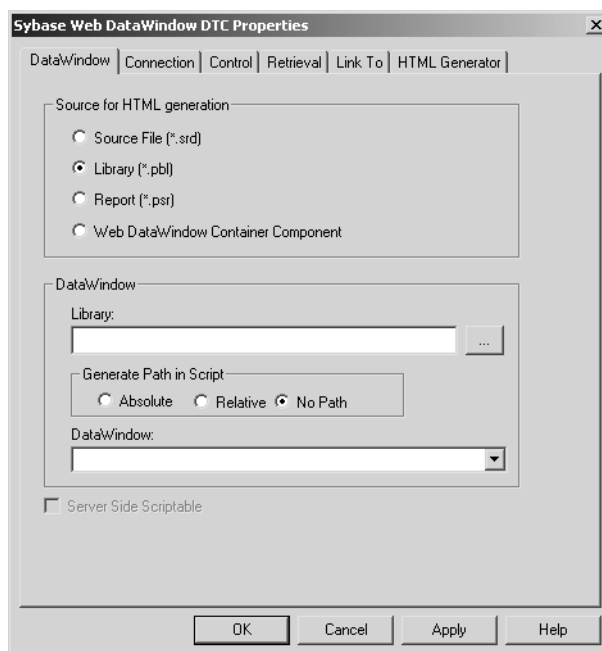
## Setting Web DataWindow DTC properties

The Web DataWindow DTC property page has six tab pages that you use for setting DTC properties.

## Selecting the source for a DataWindow object

You create a DataWindow object in PowerBuilder or InfoMaker. If you use a DataWindow or report (PSR) in a PowerBuilder Library (PBL) or an exported source (SRD) file as the source for your Web DataWindow, you must make sure that the file is available to the application (page) server. This means either that the file containing the DataWindow definition must be deployed to the system path of the server, or you must specify an absolute path to the file.

To specify an absolute path, you use the DataWindow page of the Sybase Web DataWindow DTC Properties dialog box:



---

**When the source is a Web DataWindow Container component**

If you use a Web DataWindow Container component as the source for a Web DataWindow, you must build the project you create with the Web DataWindow Container Component wizard and deploy the component directly to the component server for your Web application.

---

**Generating the path in script**

PowerBuilder allows you to deploy a DataWindow that you select in the Web DataWindow DTC as part of your JSP target. The Generate Path in Script field on the DataWindow page of the Web DataWindow DTC Properties dialog box contains three radio buttons: Absolute, Relative, and No Path.

After you add or import a PBL or PSR to your current target path and select that PBL or PSR as your DataWindow source, you can select the Relative radio button. If the Relative radio button is selected when you deploy your target, the PBL or PSR will be deployed with the target. Typically, you would select the Relative radio button only if your page server also functions as a DataWindow component server.

---

### **Deploying more than once**

If EAServer has loaded a DataWindow from your target and you attempt to deploy the target a second time while the Relative radio button is selected, your PBL might be locked and the deployment might fail. To avoid this, you can disable instance pooling for the component in EAServer Manager (EAServer 5.x) or in the EAServer Web Administration Console (EAServer 6.0 or later). After you have finished testing and editing the target containing the DataWindow object, you can enable instance pooling.

Instance pooling provides better performance in a production environment when a component instance can be reinitialized and reused for multiple clients.

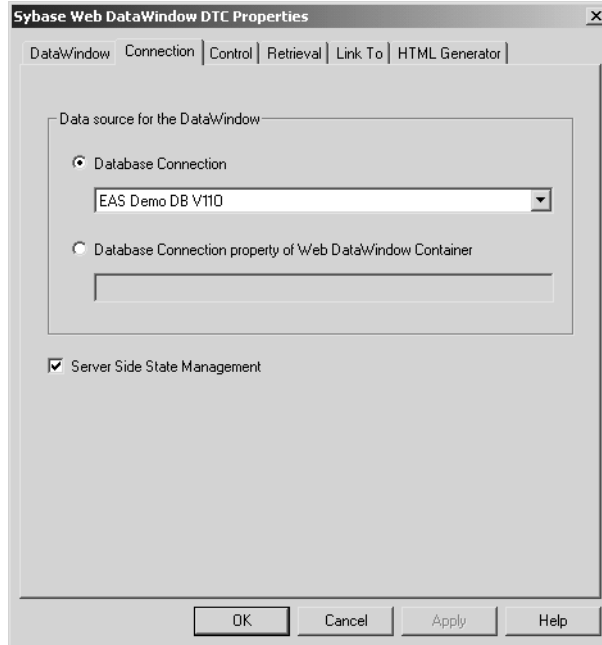
---

If you select the Absolute radio button, you must make sure that the path to the DataWindow source is the same on your development machine as on the machine that hosts the server. If you intend to deploy the DataWindow source manually to the system path of the server machine, you can select the No Path radio button.

## **Selecting a database profile**

On the Connection tab page of the Sybase Web DataWindow DTC Properties dialog box, you can override the database connection defined for a DataWindow object in a Web DataWindow Container component. A default value for the database connection is set when a Web DataWindow Container is created. The database connection is a container component property that you can also modify in EAServer Manager or in the EAServer Web Administration Console.

When you build your DataWindow, you must define a data source connection. In PowerBuilder, you must also set up a database profile to define access to the data source connection. Database profiles that you define in PowerBuilder automatically populate the Database Connection drop-down list on the Connection page of the Sybase Web DataWindow DTC Properties dialog box. For information on defining a database profile, see *Connecting To Your Database*.



---

**Connection information not needed for PSR file**

You do not need—and cannot select—a database connection for a report definition that you get from a PSR file. Report data is embedded in the report with no connection to the database.

---

You must make sure that the application server can use the database connection defined in your database profile to connect to the data source for the DataWindow object. See your server documentation for the types of connection and the connection options it supports.

In JSP targets, the deployment controller creates a *Database.properties* file that it deploys to the server with your target. The *Database.properties* file contains the connection information from all the database profiles defined in PowerBuilder on the development machine.

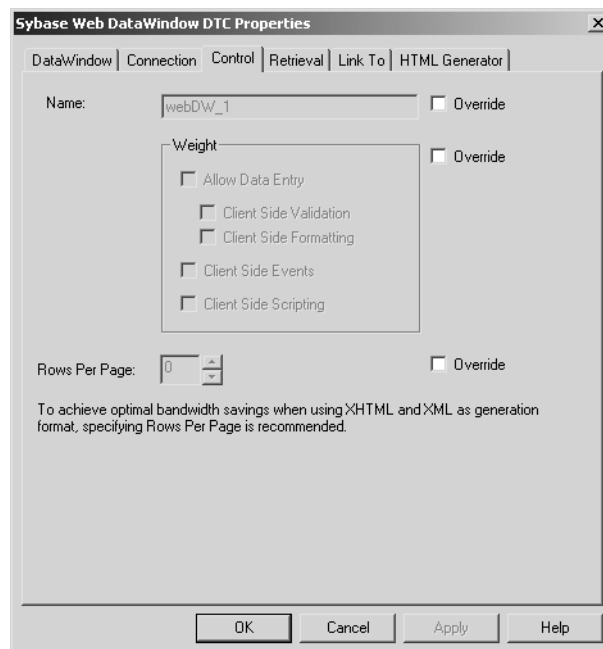
If you use a Web DataWindow Container component

If you use a Web DataWindow Container component, you should make sure that the database connection you define on your local machine is also defined as a connection cache on the EAServer machine to which you deploy the component. If you are using a JDBC connection and do not define a connection cache, you will get a runtime error when you try to use a DataWindow definition from the container component.

With a Web DataWindow Container component, you also have the option of overriding the database connection defined in the component.

## Controlling the behavior of the DTC

The Control properties page displays name and behavior properties for the Web DataWindow DTC.



In the Control tab page you can modify specific settings for the instance of the DataWindow object on your page. You can define settings on this page to override these values set in the DataWindow object:

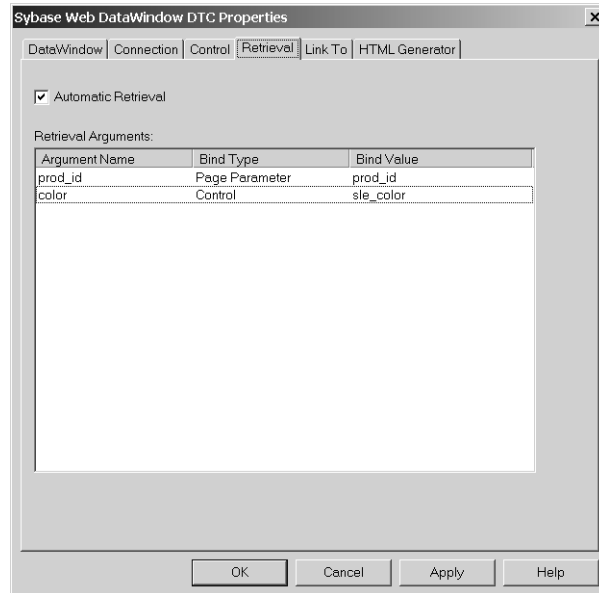
<b>Values you can override</b>	<b>Description</b>
DataWindow name	The name identifies the client-side Web DataWindow control. You can use this name in client-side scripts that you write. The Override check box lets you change the name of this instance of the Web DataWindow control.
Weight	The weight identifies the type of functionality included on your HTML page, including whether you allow client data entry, client-side events, and client-side scripting. As you include more functionality on your page, the size of the control increases. The largest (heaviest) but most feature-rich objects support both client-side formatting and client-side scripting.
Number of rows per page	The Rows Per Page property shows the number of rows displayed on each page. Override lets you change the number of rows that are displayed. With Override selected, you can specify that the Web DataWindow DTC display fewer rows than the number defined in the DataWindow object.

Instead of using overrides

Instead of using the Control tab page overrides, you can change the original (default) values for these settings in the DataWindow painter. You do this on the HTML Generation page of the Properties view for the DataWindow object that you selected in the Web DataWindow DTC.

## Setting the bind type and values for retrieval arguments

If the Web DataWindow object has one or more retrieval arguments, then the Retrieval tab page displays the names of the retrieval arguments defined for the Web DataWindow object. You can specify retrieval argument bind type and bind values for the Web DataWindow DTC.



Specifying the bind type

The bind type is the type of data that will be passed to the Web DataWindow. The Bind Type column has a drop-down list box that allows you to specify how the Web DataWindow DTC will get a value for each retrieval argument:

- **Control** For 4GL JSP pages only, the value property of a server object.
- **Constant** A fixed value.
- **Page Parameter** A parameter whose value is set on the server, either as a value passed from one page to another or as a default value set for the page if no value is passed.

---

### If you type in a parameter name for a 4GL JSP page

If you type in a parameter name for a 4GL JSP page (instead of selecting it from the drop-down list box), make sure to add the parameter to the list on the Parameters tab of the Page Properties dialog box.

---

- **Page Variable** A variable whose value is set on the server. This selection is available only on 4GL JSP pages.

Specifying bind values

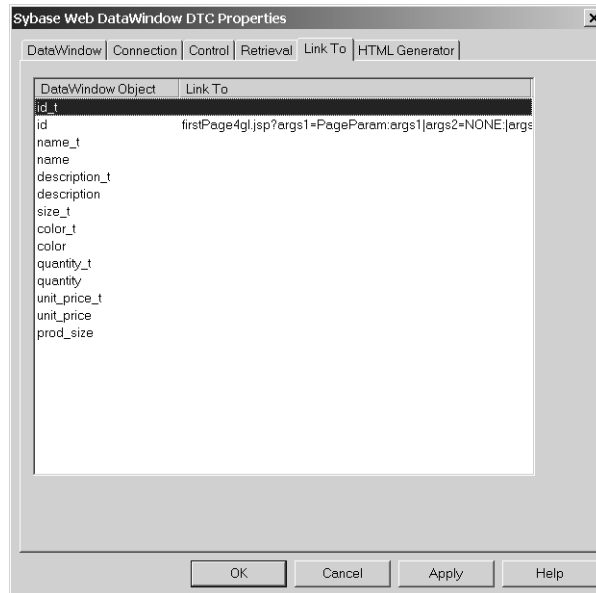
Bind values are the data values that will be passed to the DataWindow as retrieval arguments. The value you specify depends on the option you select in the Bind Type drop-down list box. Available page parameters, page variables, and control values automatically populate the Bind Value list when you select these types in Bind Type.

For more information about binding data to retrieval arguments, see “Binding data to DataWindow retrieval arguments” on page 224.

## Defining links

The Link To property page displays a list of the columns, text, computed fields, and graphical elements for your DataWindow object, and lets you define links for them for the Web DataWindow DTC.

The properties on the Link To tab apply when you use a DataWindow object that has a Tabular or Grid presentation style.



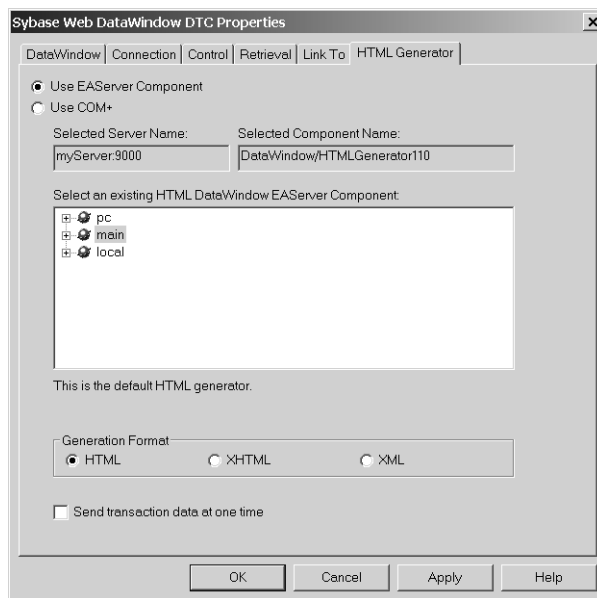


DataWindow object column	List of the columns, text, computed fields, and graphical elements available for linking from the DataWindow object. Links on columns work only when the columns are read-only. You can change a column to read-only by setting its tab order to 0 in the DataWindow painter. (In the DataWindow painter, you can also replace the column with a text field that uses a DataWindow expression for the column. You can then define links for the text field that contains the column data.)
Link To column	<p>The URL of the target page that gets displayed when a user clicks on the specified object. To add a target with defined parameters, click in the box under Link To in the row that specifies the column name. You define the link in the Link Definition dialog box.</p> <p>For information about defining hyperlinks on an object in a DataWindow, see “Defining hyperlinks on objects in a DataWindow” on page 230.</p>

## Selecting a Web DataWindow generator

EAServer hosting

If your Web DataWindow generator component is hosted in EAServer, you select the particular Web DataWindow generator that you want to use in the HTML Generator page of the Web DataWindow DTC Properties dialog box:



You must first define a profile for the EAServer machine you want to use. For information on defining an EAServer profile, see “Accessing components” on page 151. In JSP targets, the deployment controller creates a *Jaguar.properties* file that it deploys to the server with your target. The *Jaguar.properties* file contains the EAServer information from all the EAServer profiles defined in PowerBuilder on the development machine.

From the Sybase Web DataWindow DTC Properties dialog box, you can select a custom component that implements the default generator or a Web DataWindow Container component that you want to use for the generation.

---

### Using a Web DataWindow Container component

If you select a Web DataWindow Container component as the source for your DataWindow, you cannot select a different component for the generation.

---

By default, a DataWindow that you add to a Web page in the HTML Editor uses the HTMLGenerator110 component on an EAServer machine to generate the DataWindow as HTML. It is likely, though, that you might want to generate your Web DataWindow as XML (with subsequent transformation to XHTML and CSS) or as XHTML directly. The GenerateXMLWeb and GenerateXHTML methods that enable you to do that are in the PSDataWindowClass and PSWebDataWindowClass of the Web target object model. For information about the Web DataWindow implementations, see the *DataWindow Programmer's Guide*.

### COM+ hosting

You cannot select a generator component on COM+ from the Web DataWindow DTC. However, if you are using such a component for generating a DataWindow, you must select the Use COM+ radio button on the HTML Generator page of the Sybase Web DataWindow DTC Properties dialog box.

To access a COM component from a JSP page, you must use a Java-COM bridge. For more information, see the white paper *How to set up a JSP that uses a DW DTC to access the Web DW on MTS via Tomcat* on the Sybase Web site at <http://www.sybase.com/detail?id=1029911>.

## Editing existing Web DataWindow DTC properties

You can edit Web DataWindow DTC properties in Page view or Source view.

**❖ To edit Web DataWindow DTC properties in Page view:**

- 1 Right-click the DTC object in Page view and select Sybase Web DataWindow DTC Properties from the pop-up menu.

The Sybase Web DataWindow DTC Properties dialog box displays.

- 2 Make the changes you want to the property settings and click OK.

**❖ To edit Web DataWindow DTC properties in Source view:**

- 1 Right-click the METADATA or OBJECT tag for the DataWindow source code and select Properties from the pop-up menu.

A representation of the control displays in the Edit Design Time Control dialog box, and the Sybase Web DataWindow DTC Properties dialog box displays with the current settings for this control.

- 2 Make the changes you want to the property settings and click OK.
- 3 Click OK to close the Edit Design Time Control dialog box.

## DataWindow presentation styles and data sources

The Web DataWindow supports most PowerBuilder DataWindow functionality.

### Presentation styles

The Web DataWindow and the Web DataWindow DTC support the following presentation styles

- Freeform
- Tabular
- Grid
- Group
- TreeView
- N-Up
- Crosstab
- Label

Data sources                      The Web DataWindow and the Web DataWindow DTC support the following DataWindow data sources:

- Quick Select
- SQL Select
- Query
- Stored Procedure

For more information            For complete information about designing DataWindow objects for the Web DataWindow, see the *DataWindow Programmer's Guide*.

## Binding data to DataWindow retrieval arguments

The Web DataWindow DTC can bind data from the current page or a linking page as values for retrieval arguments, allowing you to control what data is retrieved. You can use retrieval arguments to allow a single page to retrieve different sets of data depending on a user's selection, or to enable the reuse of a single DataWindow object in the design of many pages.

To use retrieval arguments, you define them when you create the DataWindow object in PowerBuilder or InfoMaker. You make the retrieval arguments part of the WHERE clause for the SQL statement. Then, when you select the DataWindow object for your Web DataWindow, the retrieval arguments you defined are automatically listed in the Argument Name column on the Retrieval tab of the Sybase Web DataWindow DTC Properties dialog box.

You can have the following types of binding for your retrieval arguments, depending on whether or not your DataWindow Web page is 4GL enabled:

**Table 11-1: Binding types for Web DataWindow retrieval arguments**

Binding type	4GL enabled	Not 4GL enabled
Constants	Yes	Yes
Control Values	Yes	No, but you can use page parameters to bind control values (sent in a form submit or query string) to retrieval arguments
JavaScript Expressions	No	Yes, but not available for JSP targets

Binding type	4GL enabled	Not 4GL enabled
Page Parameters	Yes, but the parameter you bind must be on the list of parameters in the Page Properties dialog box	Yes
Variables	Yes	No

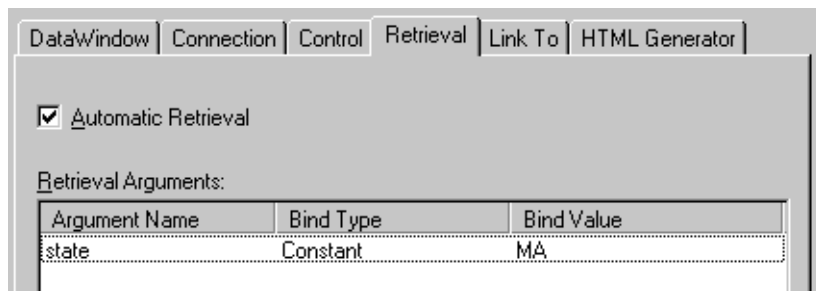
## Constants

Use a constant when the retrieval argument is always the same in the particular page you are designing. You specify the value directly on the Retrieval tab of the Sybase Web DataWindow DTC Properties dialog box.

### Example

**Using a constant** This example has a single page called *emplist.htm* in the Massachusetts section of your site. The DataWindow object has a retrieval argument called *state*, which allows you to use the same DataWindow object on other pages that list employees by state.

On the Retrieval page of the Sybase Web DataWindow DTC Properties dialog box, you can type MA as a constant value for the state retrieval argument.



## Control Values

Use a control when the value you want to bind to the retrieval argument is specified in a server-scriptable control on the same page as the Web DataWindow DTC. You can set a server-scriptable control value for a retrieval argument on a 4GL JSP page only.

## JavaScript Expressions

Use a JavaScript expression when the retrieval argument value requires processing before the DataWindow retrieves data. You can include variables that have been defined in another script on the same page in your JavaScript expression. Binding a retrieval argument to a JavaScript expression is not possible in a JSP target. To use a variable, you must declare the variable in a server script that runs before the Web DataWindow DTC.

### Verify in the Source view

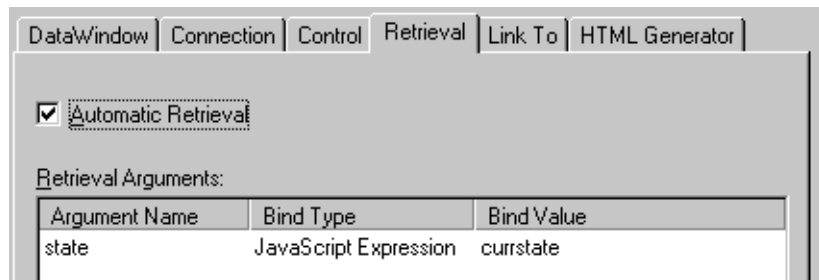
You can look at the page in Source view to make sure scripts appear on the page in the right order.

#### Example 1

**Using a JavaScript variable** In this example, a server script establishes the value of the State retrieval argument. In the script, which must be executed before the DataWindow scripts on the page, the variable *currstate* is assigned a value. The script can do any other processing that you need.

```
var currstate = "MA";
```

On the Retrieval page of the Sybase Web DataWindow DTC Properties dialog box, specify *currstate* as a JavaScript expression for the state retrieval argument.



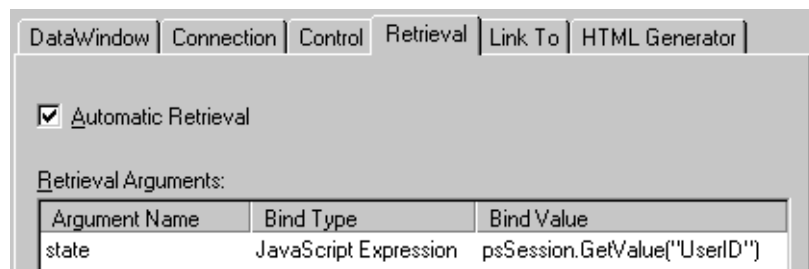
#### Example 2

**Using an expression** In this example, a server script on a login confirmation page for employees created a Userid property for the psSession object. The value was passed to the confirmation page as a page parameter.

```
psSession.SetValue("Userid",
psDocument.GetParam("loginid"));
```

On some other page in the Web application, a Web DataWindow DTC can use the Userid value by specifying this expression on the Retrieval tab.

```
psSession.GetValue("Userid")
```



## Page Parameters

Use a page parameter when the value for the retrieval argument is specified on another page. You can add page parameters on the Parameters tab page of the Page Properties dialog box. The parameters you specify automatically appear in the Bind Value drop-down list on the Retrieval tab of the Sybase Web DataWindow DTC Properties dialog box when you select Page Parameters as the bind type.

**For 4GL JSP pages** When the linking page is 4GL enabled, the JSP Target user interface lets you specify parameters (and variables, expressions, and so on) to pass to the target page using different navigation styles: Hyperlink, Form Submit, or Server Redirect. After you link these values to parameters that you specify on a target page containing a Web DataWindow, you can select the parameters from the Bind Value drop-down list to bind them to your retrieval arguments.

For more information on navigation styles for 4GL pages, see Chapter 10, “Setting Up Page Navigation.”

**For non-4GL JSP pages** When the linking page is not 4GL enabled, you can use the Hyperlink or the Form Submit navigation styles, but you must manually edit or verify the source code rather than rely on the Web Target user interface to generate this for you. The following table shows the tasks required on the linking page to submit a value as a parameter for a retrieval argument on the target page.

**Table 11-2: Using a parameter from a non-4GL linking page as a retrieval argument**

Navigation style	Tasks on linking page
Hyperlink	Set HREF attribute for the A element to the target URL. Append a page parameter to the URL using a query string.
Form Submit	Make sure all controls (whose values you want to bind to the retrieval argument on a target page) are wrapped in a FORM element. Set the ACTION attribute of the FORM element to the target URL.

On the Retrieval tab of the Sybase Web DataWindow DTC (in the target page), select Page Parameter as the bind type and type in the parameters you are passing in the Bind Value column. The parameters you type must match the parameters you submit from the linking page.

For more information about page parameters, see “Managing page data” on page 140.

## Example 1

**Setting up and using a page parameter** In this example, the user selects a state and views a list of employees in that state. Two Web pages are involved: the linking page has a form for selecting the state and the target page has a Web DataWindow DTC. This example demonstrates one way to pass page parameters from a linking page that is not 4GL enabled.

The first page, *Empstate.htm*, includes this form:

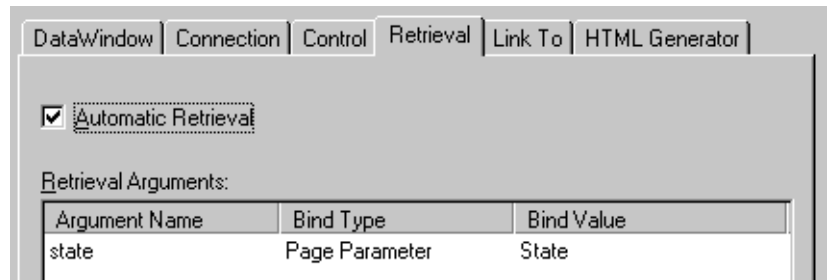
```
<FORM id=FORM1 name=EmployeeState action="emplist.htm"
 method=post>
Choose a state:
 <SELECT id=SELECT1 name=State size=3>
 <OPTION value=MA>Massachusetts
 <OPTION value=CA>California
 <OPTION value=TX>Texas
 </SELECT>
 <INPUT value=Go id=INPUT1 name=Submit type=submit>
</FORM>
```

When the user clicks the button labeled Go, the target page *Emplist.htm* displays. *Emplist.htm* has two page parameters with names that match the two form fields: State and Submit.

In the Web DataWindow DTC, the DataWindow object has a retrieval argument called *state*. On the Retrieval tab of the Properties dialog box, you make the connection between the state retrieval argument, which is listed automatically, and the State page parameter.



The names of the form field and the target page parameter must match in capitalization. In the form above, the NAME attribute of the SELECT element is State; therefore, the page parameter name in the Bind Value column must also be State, with the same capitalization.



### Example 2

**Page parameters passed from <A> elements** An alternative to the Form Submit method is a list of hyperlinks. On the linking page *Empstate.htm*, several HTML anchor elements could include a query string as part of the target URL. In each anchor element, the query string assigns a different value to the name State.

```

 Massachusetts

 California

 Texas
```

## Variables

Use a page variable when the value you want to bind to a retrieval argument is specified in a variable on the same page as the Sybase Web DataWindow DTC. You can set page or session variables for the retrieval argument only on 4GL JSP pages.

## Defining hyperlinks on objects in a DataWindow

The Web DataWindow DTC can pass data in query strings to a target page. You can use the Link To tab page in the Sybase Web DataWindow DTC Properties dialog box to generate hyperlinks around headers and labels, computed fields that are not calculated on the client, graphical elements in a DataWindow, or read-only columns.

### Setting links on columns

You can change a column to read-only by setting its tab order to 0, its Protect property to 1, or its Edit.DisplayOnly property to Yes. Hyperlinks can also be set around text objects that use DataWindow expressions to display data from database columns.

When you click in the Link To column next to a DataWindow object, a browse (...) button displays to the right in the row that you clicked. The browse button opens the Link Definition dialog box, where you specify the target page for the link and the data to bind to target page parameters.

### Bind types

The types of values you can pass to the target page from a DataWindow object are:

- **Control** Select this to pass the value of any control from the current page to the target page. The initial value of the control is passed, even if the control is editable at runtime. You can use a control as a bind type in a 4GL Web page only.
- **Constant** Use a constant when you know the value you want to pass to the target page. Type the value directly in the Bind Value column of the Link Definition dialog box.
- **Database column** Use a database column when you want to pass data for a column from the row a user clicks. You can bind column data to any DataWindow object you select on the Link To tab. If you select a column (label) as the DataWindow object link, the column value you bind is not restricted to the column you selected for the link; you can pass data from another column, such as a column that is not displayed.
- **DataWindow Expression** Use a DataWindow expression to pass a value derived from retrieved data for the DataWindow.
- **JavaScript Expression** Use a script variable when you want to pass a value that was calculated in a previously-run server script on the current page, or that you can specify as an expression. The variable's value is not derived from the retrieved data for the DataWindow, although it could refer to other data you have retrieved. You cannot use a JavaScript expression as a bind type in a JSP target.

- **Page Parameter** Use a page parameter when you want to pass on, as is, to the target page, a value that was passed to the current page. The value is not derived from the retrieved data for the DataWindow.
- **Variable** Use a page variable to pass on the value of a variable on the current page. You can use a variable as a bind type in a 4GL Web page only.

❖ **To link to parameters on other pages**

- 1 Right-click on a Web DataWindow DTC and select Sybase Web DataWindow DTC Properties.
- 2 Click the Link To tab.
- 3 Select a DataWindow object to link to (for example, a header or a picture).
- 4 Click under the Link To column for the object you selected.  
An ellipsis button displays in the row where you clicked.

- 5 Click the ellipsis button.

The Link Definition dialog box displays.

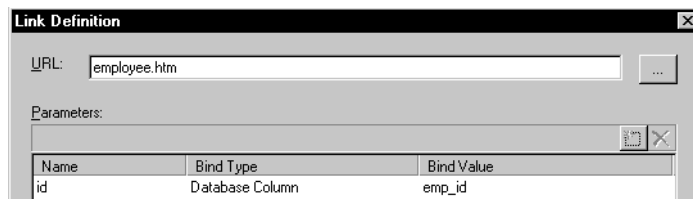
- 6 Click the browse (...) button to open the Choose URL dialog box, select a Web page or file to which you want to link, and click OK.

You return to the Link Definition dialog box. The name of the target page displays in the URL box. Existing parameters on the target page display under the Name column. You can type in additional parameters, but you need to define them later on the target page.

- 7 Select a bind type (for one of the target page parameters) from the Bind Type drop-down list.

The Bind Value drop-down list is automatically populated for certain bind type selections.

- 8 Select or type a value in the Bind Value drop-down list, and click OK.



## Custom tag library for the Web DataWindow

You can use the Web DataWindow custom tag library to specify the parameters and values required by a Web DataWindow on a JSP page. The tag library is defined in the file *DataWindow110.tld*. To use the tag library, place the *DataWindow110.tld* file in a *WEB-INF/tlds* directory in your Web applications Source directory. The tag classes are included in the *jspobject.jar* file that is deployed with all PowerBuilder JSP Web applications.

The tag library contains two tags, DataWindow and DWColumnLink. The DWColumnLink tag is an inner tag—it can be used inside the DataWindow tag only. On 4GL JSP Web pages, you must set the fourGLWeb attribute of the DataWindow tag to true.

Attributes have three subelements: name, required, and rtxprvalue. The rtxprvalue element is optional and indicates whether the attribute's value can be dynamically calculated at runtime.

For more information about the DataWindow and DWColumnLink tags in the Web DataWindow custom tag library, see the *JSP Target Reference*.

# Building and Deploying JSP Targets

About this chapter

This chapter describes the production process for JSP targets, and explains how to build and deploy JSP targets from your workspace.

Contents

Topic	Page
About building and deploying JSP targets	233
Building JSP targets	234
The deployment process	237
Setting up a deployment configuration	238
Editing a JSP deployment configuration	239
Enterprise Portal deployment options	251
Deploying a JSP target	254
Running a JSP target	255
Troubleshooting 4GL JSP pages	255
Troubleshooting JSP targets	259

## About building and deploying JSP targets

You build and deploy JSP targets to:

- Test part or all of a JSP target
- Move a completed JSP target into a production environment

The build phase

Building a JSP target prepares your files for deployment and verifies links from a JSP page to another file. You can build a target anytime during development to get information about broken links so that you can fix them. You must build a target or individual files before deployment to make the files available for deployment. The deployment process does this automatically.

The deployment phase

Deploying a JSP target processes target files from the Build folder and moves them to the runtime environment you specify. Typically you deploy your application in a local test environment before deploying it to the production servers.

You can deploy JSP targets to:

- **EAServer** The JSP container included with EAServer
- **Tomcat** The official reference implementation for Java servlet and JSP technologies from the Apache Software Foundation's Jakarta project
- **Other JSP servers** Any server that supports the JSP 1.2 specifications
- **Sybase Enterprise Portal** Instead of a JSP server

## Building JSP targets

The build phase for a JSP target:

- Provides link verification
- Creates the *Jaguar.properties* and *Database.properties* files from current EAServer and database profiles, and places them in the target Web-Inf\classes directory
- Moves the files to a Build folder under the *target* folder to make the files available for deployment

You can build an entire JSP target, or just one file. Building a file lets you quickly make sure that links from the file work.

Link verification

When the build process verifies the links from one file to another in the target, it displays information about broken links in the Output window. It also verifies the syntax—but not the integrity—of links outside the target.

Double-clicking the broken link in the Output window opens the file in the HTML Editor. You can choose to fix the link or not. If you do not fix the link, the deployed files will also have a broken link.

The Web target processes the following HTML attributes during link verification:

ACTION	CODEBASE	HTTP-EQUIV
BACKGROUND	DYNSRC	SRC
CODE	HREF	

Elements that use these attributes include:

**Table 12-1: HTML elements parsed for link integrity**

Element	Attributes processed
A	HREF
Applet	CODE, CODEBASE
Base	HREF
Bgsound	SRC
Body	BACKGROUND
Form	ACTION
Frame	SRC
Img	SRC, DYNSRC
Input	SRC (for TYPE="image")
IsIndex	ACTION
Layer	SRC, BACKGROUND
Link	HREF
Meta	HTTP-EQUIV
Script	SRC
Object	CODE, CODEBASE
Table	BACKGROUND
TD	BACKGROUND
TH	BACKGROUND

#### Files in Build folder

The JSP target build process copies files from the target Source folder to the target Build folder. These folders are visible in the JSP target development environment (in the Library painter or in the System Tree when the root is set to My Computer).

Files from the Build folder are processed during deployment to a Web site. When you explicitly build a target, you can choose to build all of the files in the target, or only those files that have changed since the last build.

How to build a JSP target

Invoking the build process whenever you save a file gives you timely information about the links in the file, and ensures that you have a copy of your changes in the Build folder, ready for deployment. You can also build a file or an entire target at any time.

**Table 12-2: Build selection options**

Select this menu item	To do this
Full Build	Build all the files in the target and regenerate the <i>Jaguar.properties</i> and <i>Database.properties</i> files
Incremental Build	Build target files that have changed since the last build

❖ **To build a JSP target:**

- On the Workspace tab page of the System Tree, right-click a JSP target and select Full Build or Incremental Build from the pop-up menu.

You can also set a deployment option to specify if you want to do a full build or incremental build when you deploy your JSP target. You must do a full build to make sure connection properties for new database or EA Server profiles are available to a JSP target. For information on setting this option, see “Editing a JSP deployment configuration” on page 239.

If you want to build all of the files within a workspace, you can use the workspace Run menu. When you build a workspace, the build processes target files within the active workspace. A workspace build can also be full or incremental.

❖ **To build an individual file:**

- On the Workspace tab page of the System Tree, right-click a file, then select Build from the pop-up menu.

❖ **To build multiple files (without building a target or workspace):**

- 1 In the List view of the Library painter, display the Source directory under your JSP target directory.
- 2 Use the CTRL key to select all the files you want to build.  
You can use the SHIFT key instead to select consecutive files in the list.
- 3 Right-click the selected files and select Build from the pop-up menu.



## The deployment process

The deployment process

The deployment process involves three phases:

- **Get** Retrieves a file from the Build folder
- **Transform** Processes the contents of the file, changing HTML tags and scripts, or adding server scripts as necessary to suit the target application server
- **Put** Writes out the deployed pages to your local or network file system, to an FTP site, or directly to an application server

When you deploy a target, the deployment controller executes these three phases for each JSP and script file you deploy.

About deployment configurations

A deployment configuration is a named set of instructions for deployment. You can deploy to one configuration, or to as many as you like. Deployment configurations can be stored in a target file or in your Windows registry.

When you define a configuration, you specify the type of server to deploy the Web site files to, and other information about your site that the deployment controller requires to transform files into the syntax used by the server.

You can use the JSP Target wizard to define a deployment configuration for a JSP server. Otherwise, you set up and modify deployment configurations from the JSP target properties dialog box.

## Working with JSP servers

The JSP deployment controller provides program logic that performs the Get, Transform, and Put phases of pre-deployment, deployment, and post-deployment procedures as required.

Transformations for JSP targets

When you deploy a JSP target, the JSP deployment controller adds the following server scripts to the top of each JSP page:

```
<%@ page import="com.sybase.powerbuilder.jspobject.*" %>
<%
 // global instance for the page
 PSDocumentClass psDocument = new PSDocumentClass
 (request, response, out, application);
 PSSessionClass psSession = new
 PSSessionClass(session);
 PSServerClass psServer = new
 PSServerClass(psDocument);
%>
```

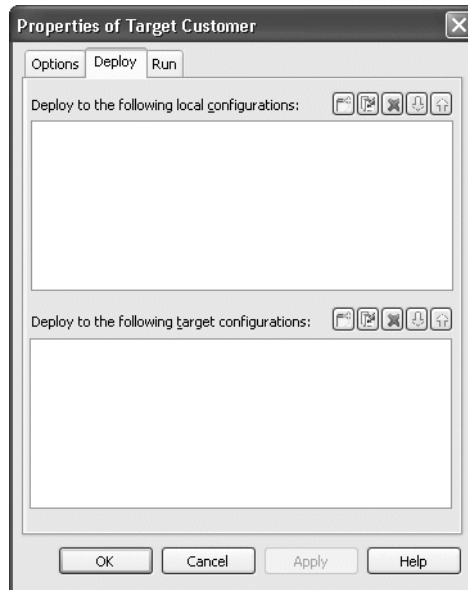
## Setting up a deployment configuration

From a Target Properties dialog box for a JSP target, you can set up both local and target configurations:

- **Local configuration** A local deployment configuration is stored in your local registry for personal use. Use a local configuration if you have a test server to deploy, and if other developers will not be editing the target.
- **Target configuration** A target deployment configuration is stored in the target file for use by anyone with access to the target. Typically, you use target configurations if you share a deployment configuration with other developers or check your target files into a source control system.

❖ **To set up a deployment configuration:**

- 1 From the Workspace tab of the System Tree, right-click the JSP target, then select Properties from the pop-up menu.
- 2 In the Properties of Target *TargetName* dialog box, click the Deploy tab.
- 3 On the Deploy page, click one of the create new configuration buttons (local or target) to set up a new local or target deployment configuration:



The New Deployment Configuration wizard starts.

- 4 Follow the instructions on the wizard pages.

## Editing a JSP deployment configuration

The Deployment Configuration Properties dialog box for a JSP target includes properties for:

- "General deployment options" next
- "JSP deployment options" on page 241

When you deploy the JSP target, PowerBuilder builds a Web Archive (WAR) file in the deployment configuration folder created by the JSP Web Target wizard. The WAR file contains the JSP files you added to the target, any classes or JAR files you added to the *Source* folder, and a *web.xml* file that conforms to the Document Type Definition (DTD) for Web applications. The Web application is automatically deployed to the server you selected in the target wizard.

The *web.xml* file is the deployment descriptor for the Web application. The deployment configuration properties you define in the JSP Options section of the JSP Deployment Configuration Properties dialog box are written to the *web.xml* file.

In general, you should not edit the *web.xml* file manually. The changes you make in the Deployment Configuration Properties dialog box are propagated to copies of the *web.xml* file in the WAR file and appropriate subdirectories of your JSP target.

You open the Deployment Configuration Properties dialog box for a JSP target in the same way as for a Web site target: select Properties from the target's pop-up menu and double-click the deployment configuration you want to view or edit on the Deploy page of the Target Properties dialog box.

### General deployment options

The general deployment options of a JSP target allow you to:

- Enter a description for the deployment configuration
- Enter server information
- Include the JSP object model in the deployment archive
- Select a build strategy and how you want to handle deployment errors

You cannot change the deployment configuration name from the Deployment Configuration Properties dialog box. If you want to create a different deployment configuration name, close this dialog box and start the Name New Deployment Configuration wizard from the Deploy page of the Properties dialog box for the target.

The general deployment options for a JSP target consist of four main selection pages that you access from the tree view in the Deployment Configuration Properties dialog box:

**Table 12-3: General deployment configuration options for a JSP target**

Deployment selection page	Description
General	Type a description for the deployment configuration.
Server Information	<p>Lists the server types to which you can deploy. The current selection is highlighted. Additional selections are available depending on which server you select:</p> <p><b>EAServer</b> Select a deployment profile and an HTTP port.</p> <p><b>Tomcat</b> Select the deployment folder and the HTTP server and port. You can also select a login name and password, and choose to stop and restart the server automatically after the target is deployed.</p> <p><b>Custom Command Line</b> Type deployment commands required for deploying the target WAR file to a JSP server other than EAServer or Tomcat. You can use macros to build the command lines and select options allowing you to abort deployment on detection of an error, show deployment messages in the output window, or create the target WAR file from the command line (by blocking PowerBuilder from generating the WAR file).</p>

Deployment selection page	Description
Object Model	Select whether you want to deploy the JSP object model with your JSP target. You can select the default JSP object model only.
Deploy What?	<p>Select Deploy All Or Nothing to make sure that nothing gets deployed when one of the files selected for deployment fails the build or predeployment processing. Select Deploy Only Successful Files to prevent failure of a single file from affecting deployment of other files in the target.</p> <p>The Rebuild field lets you select whether to use an incremental or full rebuild of files you select for deployment with the current configuration. For targets that use 4GL pages, EAServer stubs are regenerated only if you select a full rebuild.</p> <p><b>Local Copy Folder</b> When you build the JSP target, PowerBuilder generates a WAR file containing JSP files and supporting objects in the folder you specify as the Local Copy Folder. You can clear the Make Local Copy Of Deployed Files check box only if you selected the Deploy Only Successful Files option.</p>

## JSP deployment options

The JSP options that you specify in the Deployment Configuration Properties dialog box are added to the *web.xml* deployment descriptor for the WAR file that contains the Web application. These properties are defined under the following dialog box headings:

JSP options	Mime Mapping	Security
Context Params	Welcome Files	Environment
Filters	Error Mapping	EJBs
Listeners	Tag Libraries	
Servlets	Resource References	

### JSP options

Web Application Name

The Web Application Name is the display name used on the server to identify a deployed WAR file.

Description

Use the description box to provide any information that might be required by the consumer of the application.

Session Timeout	Session Timeout is a specified time in minutes after which the server will terminate servlet sessions. This value applies to all the servlets within an application. A value of 0 indicates that servlet sessions never expire.
Distributable	<p>Web applications can run on only one Java VM at any one time. To override this rule, you must mark the Web application as distributable in the deployment descriptor. However, the application must conform to additional requirements. A distributable Web application cannot use <code>setAttribute</code> and <code>putValue</code> methods to place objects into a <code>javax.servlet.http.HttpSession</code> object unless the object is one of the following types:</p> <ul style="list-style-type: none"><li>• <code>java.io.Serializable</code></li><li>• <code>javax.ejb.EJBObject</code></li><li>• <code>javax.ejb.EJBHome</code></li><li>• <code>javax.transaction.UserTransaction</code></li><li>• <code>javax.naming.Context</code> object for the <code>java:comp/env</code> context</li></ul> <p>For more information, see the Java Servlet specification, available at <a href="http://java.sun.com/products/servlet/index.html">http://java.sun.com/products/servlet/index.html</a>.</p>

## Context Params

The Context Params page is where you specify the value of parameters that convey initialization information for the Web application, such as a Web master's address or the name of a system that holds critical data. They can be retrieved using the `getInitParameter` and `getInitParameterNames` methods of the `ServletContext` interface.

In a JSP page, the parameter can be retrieved in a scriptlet using the application implicit object, for example:

```
<%
 String iURL = application.getInitParameter("iURL");
%>
```

## Filters

Filter content

You can write a filter to modify requests and responses and then declare it on the Filters page. Filters implement the `javax.servlet.Filter` interface.

**Table 12-4: Filter content properties for a JSP target**

Setting	Value
Filter Name	Specify the name of the filter, for example, Image Filter
Filter Class	Specify the fully qualified class name of the filter, for example, com.acme.ImageServlet
Init Parameters	Specify initialization parameter names and values for each filter you select

For more information about filters, see the Java Servlet specification or the *Web Application Programmer's Guide* in the EAServer documentation.

#### Filter mapping

The container uses the filter mappings you specify on the Filter Mapping page to determine how to apply the filters that have been defined to requests. You can apply a filter to a single servlet by specifying its name, or to a group of servlets and other Web content by specifying a URL pattern. For example, `\*` specifies that a filter applies to all servlets in the Web application. The filters are applied in the order in which they appear in the list of filter-mapping elements in the deployment descriptor.

## Listeners

You can provide listener classes by implementing one or more of the listener classes in the Servlet API. Listeners can support event notifications or manage resources or state. You package the listener classes in the WAR file and list them in the deployment descriptor in the order in which they are to be invoked.

## Servlets

#### Servlet details

Use the servlet pages to describe a servlet class or JSP page used in the Web application. Click **New** to give the servlet or page a short name that can be used to reference it. Then select **Servlet Class** or **JSP Filename** from the drop-down list box. For servlets, you must specify the fully qualified class name in the text box next to the drop-down list box.

If you want to see target JSPs listed in the management tool for your server, you must enter a short name for each JSP, select **JSP Filename** from the drop-down list box, and enter the JSP file name in the text box next to the drop-down list box. However, this information is not required for access to the JSPs from a client browser.

You can specify the following properties for each servlet or JSP from the Deployment Configuration Properties dialog box: **Load on Startup**, **Init Param**, **Role references**, and **Servlet mapping URL pattern**.

**Load on Startup** Load on Startup indicates whether you want a servlet loaded and initialized when the application is deployed. Otherwise, the servlet class is loaded when the first client requests it. Servlet classes that perform lengthy processing in the init method can be loaded at startup so that the first client to invoke the servlet does not experience increased response time.

A value of 0 or a positive integer requires the container to load the servlet when the application is deployed. Servlets with a low Load on Startup value are loaded before those with a higher value. If you do not specify a value, or if you specify a negative integer, the container can load the servlet at any time.

**Init Param** Use the Init Param table to assign values of parameters specifying setup information for the servlet or JSP page. In a JSP page, the parameter can be retrieved in a scriptlet using the config implicit object, for example:

```
<%
 String initVal = config.getInitParameter("initVal");
%>
```

**Role references** Role references provide a mechanism for an application to map a role name used in the application’s code to a security role defined in its deployed environment.

**Table 12-5: Role reference properties for a JSP target**

Setting	Value
Name	Name of the security role used as a parameter to the <code>isCallerInRole</code> method
Description	(Optional) A comment to explain how the property is used
Link	The security role (see “Roles” on page 249) to which this reference should be linked

**Servlet mapping URL pattern** A servlet mapping defines the association between a URL pattern and a servlet. This mapping is used to map requests to servlets. The default is `/ServletTargetName`, for example, `/MyServlet`.

If the container handling the request is a JSP container, a URL containing a `.jsp` extension is implicitly mapped.

## Mime Mapping

Specify mime mappings to ensure that the Web container knows how to associate a file extension with a mime type. For example, if you specify `.txt` as the extension, you must specify a predefined mime type such as `text/plain`.



## Welcome Files

The welcome file list contains an ordered list of welcome file elements to be used when the container receives a valid partial request. A valid partial request is a request for a URI that corresponds to a directory entry in the WAR not mapped to a Web component.

For example, if the container receives a request for `//myhost:8080/myapp/mydir`, and `mydir` is not mapped to a servlet or JSP file, then if the welcome file list includes the mapping `mydir/index.html`, `index.html` is displayed.

## Error Mapping

You can customize what the client sees when an error or an exception is generated by specifying the locations of error pages for different kinds of errors. Error pages you specify here are used for servlets and for any JSP pages that do not specify an error page for the error type.

In the left column, you can specify an HTTP error code, for example 404, or a fully qualified class name of a Java exception type. In the right column, specify where to find the resource in the Web application relative to the root of the Web application. The value of the location must have a leading forward slash (/). For example, `/404.html`.

PowerBuilder adds elements to the target *Web.xml* file that depend on the value in the Error column:

**Table 12-6: Element added to target *Web.xml* file**

Error column value	Element added to <i>Web.xml</i> file
Y	<error-code>
N	<exception-type>

If you specify an exception class in the left column rather than an HTTP error code, you *must* change the Error column value to N. Otherwise you might have problems deploying or running the target, depending on the JSP container to which you deploy, or try to deploy, your target.

## Tag Libraries

If the Web application uses one or more tag libraries, you can make sure that the Web container can locate them by specifying a mapping for each tag library in the deployment descriptor. If you selected tag libraries in the JSP Web Target wizard, they display here.

You use a taglib directive to refer to a tag library in a JSP page. For example:

```
<%@ taglib uri="/WEB-INF/tlds/mycalc.tld" prefix="mc" %>
```

The uri attribute specifies the uniform resource locator (URI) for the TLD file relative to the root of the Web application. You can map this path to a short name in the deployment descriptor. Specify the name you want to use in the Tag Library URI column, and the location relative to the root of the Web application in the Descriptor File Location column. The value of the location must have a leading forward slash (/). For example, /WEB-INF/tlds/Testlibrary\_1\_3.tld.

If you specify /mycalc as the short name for the /WEB-INF/tlds/mycalc.tld, the taglib directive can be written like this:

```
<%@ taglib uri="/mycalc" prefix="mc" %>
```

## Resource References

### References

To be platform independent, an application should refer to resources within the operating environment in which it is deployed, rather than having a specific location coded within the application. The J2EE specification defines a mechanism for an application to obtain resource references in its deployed environment. Resource references are used to obtain database connections, JavaMail sessions, URL factories, and JMS connection factories.

**Table 12-7: Resource reference properties for a JSP target**

Setting	Value
Name	Specify the JNDI name used to refer to a resource. Use the prefix mail/ for JavaMail references, jdbc/ for data source references, url/ for java.net.URL references, and jms/ for javax.jms references. For example, if your code refers to java:comp/env/jdbc/MyDatabase, enter jdbc/MyDatabase.
Type	Use one of these resources: <ul style="list-style-type: none"> <li>• javax.sql.DataSource for JDBC connections</li> <li>• java.net.URL for URL factories</li> <li>• javax.mail.Session for mail sessions</li> <li>• javax.jms.QueueConnectionFactory for a JMS queue</li> <li>• javax.jms.TopicConnectionFactory for a JMS topic</li> </ul>

Setting	Value
Authentication	Enter: <ul style="list-style-type: none"> <li>• <code>Container</code> if the container signs on to the resource manager on behalf of the servlet component. The methodology used to sign on is server specific.</li> <li>• <code>Application</code> if the application signs on programmatically to the resource manager.</li> <li>• <code>Servlet</code> if the servlet (not the container) signs on programmatically to the resource manager.</li> </ul>
Sharing Scope	By default, connections to a resource manager can be shared by other components that use the resource in the same transaction context, optimizing the use of connections. Select <code>Unshareable</code> if the application cannot share connections to the resource.
Description	(Optional) A comment to explain how the property is used.

### Environment references

Resource environment references allow the JSP page to use logical names to refer to administered objects associated with resources. These references must be bound to administered objects in the deployment environment.

**Table 12-8: Environment reference properties for a JSP target**

Setting	Value
Name	Specify a name for a reference to an administered object associated with resources, such as a JMS message queue. The name is relative to the <code>java:comp/env</code> context, for example, <code>jms/MyQueue</code> .
Type	Specify the type of the resource, for example, <code>javax.jms.Queue</code> .
Description	(Optional) A comment to explain how the property is used.

## Security

### Security constraints

Security constraints let you control access to a Web resource collection. A Web resource collection identifies the resources, defined by URL patterns, and the HTTP methods on those resources, to which the security constraints apply. The security constraints define the roles authorized to use the Web resource collection (authorization constraint) and the level of transport security required of the client server (user data constraint).

You define the Web resource collection and its constraints on the Security Constraints page.

If you do not assign a user role, no user has access to the resources in the specified collection. If you do not specify HTTP methods, the constraints apply to all methods.

**Table 12-9: Security constraint properties for a JSP target**

Setting	Value
Name	Specify a name for the Web resource collection.
URL Pattern	Select one or more URL patterns to specify the resources in this Web application to which the constraints apply.
HTTP Methods	(Optional) Specify the HTTP methods to which the constraints apply. If you do not specify any methods, the constraints apply to all methods.
Authorized Roles	Select the roles authorized to access the collection of Web resources defined in the URL Pattern and HTTP Methods boxes. You can define roles on the Roles page of the Deployment Configuration Properties dialog box.
Transport Guarantee	Establish a level of transport security appropriate for the Web resources you are protecting. If you use basic or form-based authentication, passwords and other sensitive information are not protected for confidentiality. If you have sensitive information that you want to protect, establish a security constraint that uses a greater level of protection: <ul style="list-style-type: none"> <li>• NONE – uses insecure HTTP. SSL-protected sessions require more overhead than insecure HTTP sessions. Use none for transport guarantee if you do not need the added confidentiality of SSL.</li> <li>• INTEGRAL – uses an SSL-protected session that checks for data integrity.</li> <li>• CONFIDENTIAL – uses an SSL-protected session to ensure that all message content, including the client authenticators, is protected for confidentiality as well as data integrity. A confidential transport guarantee has more overhead than none.</li> </ul>

Login configuration

Protected resources on a server can be partitioned into separate protection spaces. Each protection space can be configured with a specific security scheme, such as an authentication protocol or authorization database. When a Web server asks a client to authenticate a user, it passes a realm to the client. A realm is a string that defines a protection space.

---

**Use of the term realm**

In J2EE applications, the term realm is also used to refer to a security policy domain. In this deployment descriptor, it refers to the string passed as part of HTTP basic authentication.

---

The client passes the user name and password to the Web server, and the Web server authenticates the user in the specified realm. The login-config element is used to configure the authentication method, the realm name that should be used for this application, and the attributes that are needed by the form login mechanism.

**Table 12-10: Login authentication properties for a JSP target**

Setting	Value
Authentication Method	Select the authentication method to be used to configure the authentication mechanism for the Web application: <ul style="list-style-type: none"> <li>• BASIC – the server asks the client for a user name and password. You must also provide a realm name.</li> <li>• DIGEST – advanced form of BASIC authentication using an MD5 message-digest hash of the credentials and a unique value supplied by the server. The password is not sent in clear, unencrypted text as with BASIC authentication.</li> <li>• FORM – the Web application developer creates an HTML login page, where the client enters a user name and password. The entire HTML page is sent to the server. You also create an error page that is returned to the client in the event of a server error.</li> <li>• CLIENT-CERT – the client connects to the server using SSL tunneled within HTTP. The client must provide a certificate that the server accepts and authenticates.</li> </ul>
Realm Name	Specify the realm name to be used in HTTP basic authentication.
Form Login Page	Specify the location in the Web application where the page to be used for login can be found. The path begins with a leading / and is interpreted relative to the root of the Web application.
Form Error Page	Specify the location in the Web application where the error page that is displayed when login fails can be found. The path begins with a leading / and is interpreted relative to the root of the Web application.

## Roles

A security role is a grouping of permissions that a given type of user of an application must have to successfully use an application and its components. The Roles page allows you to define security roles—for example, `admin` or `user`—that you can associate with specific resources on the Security Constraints page.

## Environment

Environment properties allow you to specify global read-only data for use by all the JSP pages in the Web application.

Servlets and JSP pages must use JNDI to retrieve environment properties, using the prefix `java:comp/env` in JNDI lookups. Unlike context initialization properties, environment properties can have datatypes other than `java.lang.String`.

The deployment descriptor catalogs the environment properties used by your servlets and JSP pages, as well as each property's Java datatype and default value. You can tailor the values to match a server's configuration. For example, you might have environment properties to specify the name of a logging file or to tune cache usage.

**Table 12-11: Environment properties for a JSP target**

Setting	Value
Name	Specifies the JNDI name, relative to the <code>java:comp/env</code> prefix, used in servlet and JSP code to refer to this resource.
Type	Select the Java datatype of the property from the drop-down list box. The specified type must have a constructor that takes a single <code>java.lang.String</code> argument.
Value	The initial or post-deployment value of the property, specified as text that is valid for the type constructor that takes a single <code>java.lang.String</code> argument.
Description	(Optional) A comment to explain how the property is used.

## EJBs

EJBs that support the EJB 2.0 specification can have both remote and local interfaces.

### EJB references

When servlets and JSP pages reference remote EJBs, the EJB reference in the deployment descriptor is used to instantiate proxies for EJB home interfaces. EJB references must be catalogued in the deployment descriptor so that the Web application does not depend on a specific naming configuration. When deploying the Web application, a site administrator can specify site-specific EJB home names.

**Table 12-12: EJB reference properties for a JSP target**

Setting	Value
Name (New button)	Click New to create a new remote reference to an enterprise bean. Specifies the JNDI name used to refer to this EJB.
Type	Choose Session for session beans or Entity for entity beans.

Setting	Value
Home Interface	The Java class name of the EJB home interface, specified in dot notation. For example, <code>com.sybase.myBeanHome</code> .
Remote Interface	The Java class name of the EJB remote interface, specified in dot notation. For example, <code>com.sybase.myBeanRemote</code> .
Description	(Optional) A comment to describe the EJB reference.
Link	The JNDI name of an instance of the specified EJB that is installed in the server where the Web application is to be deployed.

## Local references

Servlets and JSP pages can reference EJBs running in the same Java VM using local interfaces. The settings for EJB local references are analogous to the settings for EJB references, which are used when the EJB is not running in the same Java VM.

**Table 12-13: Local EJB reference properties for a JSP target**

Setting	Value
Name (New button)	Click New to create a new local reference to an enterprise bean. Specifies the JNDI name used to refer to this EJB.
Type	Choose Session for session beans or Entity for entity beans.
Local Home	The Java class name of the EJB local home interface, specified in dot notation. For example, <code>com.sybase.shopping.LocalCartHome</code> .
Local Interface	The Java class name of the EJB local interface, specified in dot notation. For example, <code>com.sybase.shopping.LocalCart</code> .
Description	(Optional) A comment to describe the local EJB reference.
Link	The JNDI name of an instance of the specified EJB that is installed in the server where the Web application is to be deployed.

## Enterprise Portal deployment options

You can deploy a JSP target to a Sybase Enterprise Portal (EP) rather than a JSP server. You can use the Deployment Configuration Properties dialog box to add or change deployment information for an EP target.

## General information for EP deployment

Table 12-14 describes the general information settings you must enter before you deploy to EP.

**Table 12-14: General information for EP deployment**

Setting	Description
EP Host	The EP host name, including the domain (for example: <code>mycomputer.sybase.com</code> ).

Setting	Description
EP Port	The HTTP port of the host machine.
User	A registered Portal Studio user with permissions to create and approve Portlets, Pages, and PageGroups.
Password	The password for the named user.
RID	A resource ID with which the user is associated. The RFID drop-down is automatically populated after you enter the EP Host and EP Port and connect to the EP server.

Building a portlet for EP deployment

Table 12-15 describes the information settings to create a portlet for the EP host.

**Table 12-15: Building a portlet for EP deployment**

Setting	Description
Portlet Name	A unique name for the portlet. The default name for the portlet you create is the same as the name of the target.
Portlet Type (unlabeled drop-down)	<p><b>Select Web Application</b> if the JSP element references a WAR file deployed on the application server. For this portlet type, you must enter the following information:</p> <ul style="list-style-type: none"> <li>• <b>WebApp Display Name</b> The name displayed for the Web application</li> <li>• <b>Initial Resource</b> The initial JSP page to display</li> <li>• <b>WAR File Name</b> The name of the WAR file, including the <i>.war</i> extension</li> </ul> <p><b>Select Remote URL</b> if the JSP element does not reference a WAR file. For this portlet type, you enter:</p> <ul style="list-style-type: none"> <li>• <b>Remote URL</b> A remote URL that calls a Web application</li> <li>• <b>Input Parameters (Optional)</b> The input parameter names as defined by JSP code</li> </ul>
Set Refresh Rate	Select a refresh time for the portlet you create.
Set Default Height	Select a default height for the portlet you create.
Display Last Refreshed	Select this check box to display the date and time the portlet was last refreshed.
iFrame	Select this check box to display the portlet in an HTML <IFRAME> element.
No Popups	Select this check box to prevent pop-up windows from displaying with your portlet.



Building a page for EP deployment

Table 12-16 describes the information settings for building or creating pages for the EP portlet.

**Table 12-16: Building a page for EP portlets**

Setting	Description
Create Page	If selected, creates a new page and adds the portlet to the page. You must enter the following information: <ul style="list-style-type: none"> <li>• <b>Page Name</b> A unique page name</li> <li>• <b>Choose Layout</b> The layout type that you want for the page (two-column 50/50 by default)</li> </ul>
Select Page	If selected, adds the portlet to an existing page. You must enter: <ul style="list-style-type: none"> <li>• <b>Page Name</b> An approved page created by the current user</li> </ul>
Position	The position of the portlet on the page in ascending order: the leftmost column starts at position 101, and the next column (if the page layout is not Full) starts at 201. Position for a portlet in a third column (if the page layout is 3 column) is 301.

Building a page group for EP deployment

Table 12-17 describes the information settings for building or creating a page group for the EP pages.

**Table 12-17: Building a page group for EP pages**

Setting	Description
Create Page Group	If selected, creates a new page group and adds the new portlet page to the group. You must enter the following information: <ul style="list-style-type: none"> <li>• <b>Page Group Name</b> A unique page group name</li> </ul>
Select Page Group	If selected, adds the page to an existing page group. You must enter: <ul style="list-style-type: none"> <li>• <b>Page Group Name</b> An approved page group created by the current user</li> </ul>
Position	The position of the page in the page group. The position starts at 101. The page group lists the pages that belong to it in ascending order.

## Deploying a JSP target

After you set up local or target deployment configurations, you can deploy a JSP target for testing or production whenever you want.

❖ **To deploy a JSP target using selected configurations:**

- 1 Right-click the target, and select Properties from the pop-up menu.
- 2 Click the Deploy tab, then select only the configurations you want to use.

When you deploy a target, it is deployed to all selected configurations. You can also choose the order of deployment to the selected configurations by moving configurations up or down in the configuration list.

- 3 (Optional) Click the Run tab, select a start page for your target, and select the deployment configuration for running.

---

### Selecting a start page

You can run the JSP target from the PowerBuilder Run menu if you select a valid start page. If you define a server and port in your current deployment configuration, you can enter a relative URL in the Start Page text box, preceding the start page with a forward slash. For example, you could use the relative path `/First.jsp` or `/MyFolder/First.jsp` as a relative URL.

You can also enter a complete URL, in which case the server, port, and mapping selections in your current deployment configuration are ignored when you run the target from PowerBuilder.

---

- 4 Click OK to close the target properties dialog box.
- 5 Right-click the target, and select Deploy from the pop-up menu.

The output window displays messages and lets you know if the deployment is successful or if errors are encountered.

---

### Deploying more than one target at once

You can also deploy a workspace with multiple JSP targets.

---

## Running a JSP target

After you deploy a JSP target, you can view the Web site files from your browser. Make sure your Web server is running. You may also need to start a component server. You need to make sure that a start page was defined for your deployment configuration when you deployed the Web target.

For information on defining a target start page, see “Deploying a JSP target” on page 254.

❖ **To view deployed files:**

- On the Workspace tab of the System Tree, right-click a target, then select Run from the pop-up menu  
*or*  
Select Run>Run  
*or*  
Select Run>Select and Run, and then select the target to run.

## Troubleshooting 4GL JSP pages

4GL JSP pages provide two troubleshooting features:

- Displaying runtime errors
- Displaying trace messages

You can enable these features when you set up your file in the 4GL JSP Page wizard or in the Page Properties dialog box for your page.

### Displaying runtime errors

4GL JSP pages provide centralized error processing that reports errors occurring before page generation, such as errors generated when server events are triggered during page processing. When you display runtime errors for production pages, the messages tell your users about problems they might encounter when they view a page.

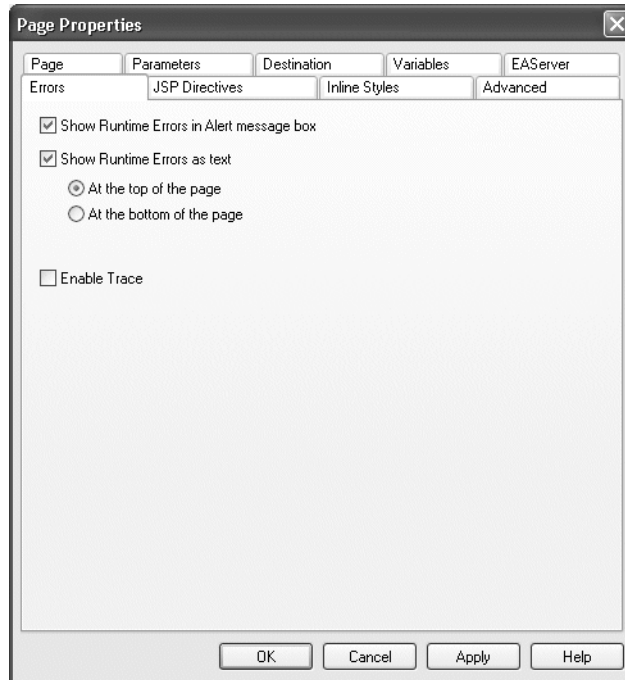
Enabling runtime error reporting

You enable the reporting of runtime errors on the Errors page of the Page Properties dialog box. You choose where the errors get displayed.

❖ **To enable reporting of runtime errors:**

- 1 Right-click on a 4GL JSP page open in the HTML editor, then select Page Properties from the pop-up menu.
- 2 Click the Errors tab.
- 3 On the Errors page, select how you want runtime errors displayed.

You can display errors on a page, in an alert box, or in both places:



Writing scripts to customize runtime error reporting

Another way to report runtime errors is by writing scripts that call methods and properties on the psPage object. The ReportError method triggers the ServerError event, then depending on the return value from ServerError, adds an error to the error log.

The `psPage` object has the following properties to support error reporting:

**Table 12-18: `psPage` properties that support error handling**

Use this property	To do this
<code>showErrorsOnPage</code>	Generate processing errors when the page is generated
<code>showErrorsAtTop</code>	Display processing errors at the top or bottom of the page
<code>showErrorsInAlert</code>	Display processing errors in a separate alert box

To display errors elsewhere on a page, use the `BeforeGenerate` event to make sure the errors are available when the page generates.

The `psPage` object also has these methods for displaying errors:

**Table 12-19: `psPage` methods for displaying errors**

Use this method	To do this
<code>WriteErrorsToDocument</code>	Define a precise location where error messages are to appear on a page
<code>TestCompError</code>	Check whether a method on an <code>EAServer</code> component caused any errors

For more information about the methods and properties that support error reporting on the `psPage` object, see the online *Web and JSP Target Reference*.

## Displaying trace messages

Tracing code for 4GL JSP pages helps troubleshoot server processing problems you might encounter as you develop your pages. With tracing enabled, you can view details about the processing of your page, including all the server-side events that are triggered. Trace messages appear at the top of your page.

### Disable for production pages

Be sure to disable tracing when you deploy your Web target to a production environment, so that your production pages do not display the messages.

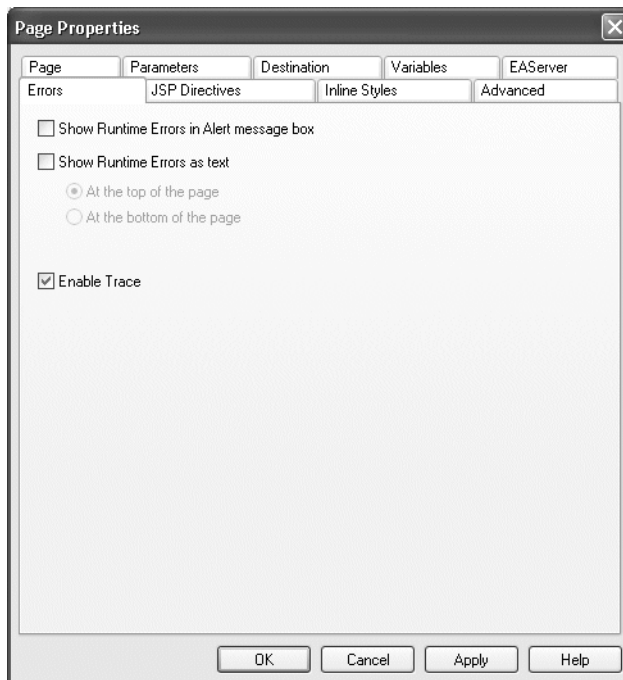
#### Enabling tracing

You enable trace messages on the Errors page of the Page Properties dialog box.

#### ❖ To enable tracing:

- 1 Right-click on a 4GL JSP page open in the HTML editor, then select Page Properties from the pop-up menu.
- 2 Click the Errors page.

- 3 On the Errors tab page, select Enable Trace.



Writing scripts to customize tracing

You customize tracing by writing scripts that call methods on `psPage`. These methods add a message to the trace and control the appearance of the text. The `psPage` object provides the following methods:

- `IsTrace`
- `SetTrace`
- `Trace`
- `TraceIndent`
- `TraceOutdent`

For details about these methods, see the online *JSP Target Reference*.

## Troubleshooting JSP targets

Several common problems can occur during JSP deployment or at runtime. Problems can also arise during the processing of a WSDL file by the JSP Web Service Proxy wizard. A few of these problems are described in this section, as well as some steps you can take to resolve them.

### Problems deploying and running JSPs

Problems deploying JSPs can be specific to a particular JSP server. PowerBuilder provides an interface for deployment of JSPs to Tomcat and to EAServer, but you can also use the command line tool to deploy JSPs to other servers that support the JSP 1.2 specification.

For more information, see “Custom command-line deployment” on page 20.

#### Problem running JSPs deployed to Tomcat

After you deploy a JSP application to Tomcat, you might need to shut down the Tomcat server and restart it before you can run the application from a client browser.

If you are using the JSP in a Tomcat server with a component running in EAServer, you might need to change the HTTP listener port for EAServer or Tomcat before you restart Tomcat. EAServer 5.x and Tomcat servers use port 8080 as the default HTTP listener port. (Port 8000 is the default HTTP listener port for EAServer 6.0 and later.) If you do not change the HTTP listener port on either server, you can still run both servers, but you must restart the Tomcat server before starting or restarting EAServer.

#### Integer parsing problem

When deploying JSPs to EAServer, it is possible to run into problems creating Integers. The issue is that in a JSP file the following code fails to compile in EAServer:

```
Integer doesNotWork = new Integer (0);
```

To fix this issue, the JSP should contain:

```
Integer doesWork = new Integer ("0");
```

Quotation marks are required around the integer (0) in the construction of the Integer.

Problems deploying after setting a target exception page

If you specify an exception class (as opposed to an HTML error code) in the Error Mapping pane of the Deployment Configuration Properties dialog box for a JSP target, you must change the default value in the Error column from "Y" to "N". Otherwise, you will not be able to deploy your target to EAServer or run your target application on a different JSP server.

For more information about JSP target deployment properties, see “Editing a JSP deployment configuration” on page 239.

Problems deploying to an upgraded version of EAServer

PowerBuilder installs *easclient.jar* and *easj2ee.jar* files to the *Sybase\Shared\Web Targets* directory. These JAR files must be compatible with the EAServer to which you deploy your JSP target, or you will have problems deploying to the server. If you are using a version of EAServer that is different from the version supplied with the PowerBuilder installation, you should replace these JAR files with the *easclient.jar* and *easj2ee.jar* files from the *Sybase\EAServer\Java\Lib* directory.

## Troubleshooting JSP Web services

When you run the JSP Web Services Proxy wizard, you can encounter problems if you select invalid WSDL files.

Using the wrong WSDL file

If you use a WSDL file created by the EAServer Web Services Toolkit, be aware that the toolkit creates two different WSDL files: an interface (or abstract) file and an implementation file. In the JSP Web Services Proxy wizard, you must select the implementation file rather than the interface file to access the Web services described by these files.

The WSDL files that you select must conform to version 1.1 of the WSDL specification found on the W3C Web site at <http://www.w3.org/TR/wsdl>.

Class not found error

If you get a class not found error at runtime (typically a provider not found error or a `java.lang.NoClassDefFoundError` error), make sure all the required JAR files are in your JSP server’s class path. You can find a description of these required files in “Files added by the wizard and files required by the server” on page 156.

You can copy these files from the *Sybase\Shared\PowerBuilder\WEB-INF\lib* directory to a directory in your JSP server class path. The required files are installed by default with EAServer in the server’s class path. However, if they are missing, you can copy them to the *EAServer\java\lib* directory. If EAServer is already running, you must shut down and restart EAServer to make sure they are included in the class path.



Nonspecific runtime error

Some browsers do not display complete error information returned from the JSP server. If you get a nonspecific runtime error, you may want to test the same JSP in a different browser.

## **Additional resources for JSPs and Web services**

Quick reference cards for JSPs, as well as a link to JSP specifications, are available on the Sun Microsystems Web site at <http://java.sun.com/products/jsp/docs.html>.

Links to tutorials for various Web service technologies are available on the XMethods Web site at <http://www.xmethods.com>.

A link to the Web Service Toolkit User's Guide is available on the Sybase Web site at <http://sybooks.sybase.com/nav/detail.do?docset=1254>.



# Index

## Numerics

- 4GL Web pages
  - about 163
  - adding content to pages 169
  - classes and objects 138
  - disabling 191
  - displaying runtime errors 255
  - displaying trace messages 257
  - EAServer integration 174
  - error reporting 165
  - page request processing 189
  - parameter binding 172
  - troubleshooting 255
  - variables 173
  - wizard 44

## A

- absolute positioning 67
- ActiveX controls
  - installed components 108
  - not marked as safe 109
- adding
  - elements from System Tree 52
  - styles 91
- Advanced tab 55
- application logic in JSPs 3
- application object, JSP 123
- application scope, JSP 124
- applications, developing 19
- authentication methods, JSP applications 249

## B

- binding retrieval arguments
  - constants 225

- control values 225
- JavaScript expressions 226
- page parameters 227
- variables 229
- buttons 120

## C

- cascading style sheets, CSS2 86
- class path
  - custom tag library search path 80, 113
  - Java 112
- classes
  - for 4GL Web pages 138
  - for JSP object model 7
- command line deployment
  - JSP targets 20
  - macros 21
- comments
  - in JSPs 121
  - METADATA tags 210
- components
  - inserting 109, 110
  - values 110
- config object, JSP 123
- configuration
  - deployment 23
- controls
  - binding 179
  - server scriptable 178
- create
  - HTML page 44
  - Web target 22
- custom deployment, JSP targets 20
- custom tag libraries
  - about 3
  - listing on Components tab 108
- custom tag libraries for JSP 79
- custom tags and JSPs 2

## D

data  
  binding 181  
  binding to Web DataWindow 224  
  displaying 145

databases  
  accessing 139  
  connection profiles 139  
  connections 145  
  display query results 147  
  handling errors 146  
  profiles in a script 139  
  SQL queries 147  
  using with Web DataWindow DTC 139, 216

declarations in a JSP 121

deployment  
  by server type 237  
  configuration 23  
  JSP command line 20  
  JSP targets 20  
  local configuration 238  
  target configuration 238  
  to EAServer 20  
  to Enterprise Portal 251  
  to Tomcat 20  
  Web targets 233, 237

description property, for JSP targets 241

design-time controls  
  inserting 111  
  METADATA tags 119  
  Web DataWindow DTC 203

directives, in JSPs 2, 76

display, element properties 54

distributable property, for JSP targets 242

document, HTML 44

DTC *see also* design-time controls 111

## E

EAServer  
  adding methods to scripts 188  
  components 153  
  integrating with Web applications 150  
  JSP deployment 20

  list of servers 108  
  variables 189

editing  
  frames 98  
  options 49

editor  
  configuring 47  
  HTML editor 37  
  integrated Script editor 116  
  standalone Web Script editor 116  
  Style Sheet editor 87

EJB local references property, for JSP targets 251

EJB references property, for JSP targets 250

elements, HTML 53

Enterprise Portal 251

environment  
  settings 47  
  Web delivery 15

environment properties, for JSP targets 250

error handling, for JSP targets 80

error pages  
  in JSP targets 81  
  property for JSP targets 245

errors  
  4GL Web pages 255

events  
  4GL Web Pages 185  
  scripting 124  
  server 184

exception object, JSP 123

## F

fallback, JSP tag 74

files  
  importing 26  
  types 26

filter mappings property, for JSP targets 243

filters property, for JSP targets 242

fonts 61

form field 64

format  
  character 61  
  paragraph style 51

- Source view 47
- tips 51
- forms
  - inserting on page 64
  - submitting from 4GL pages 198
- forward, JSP tag 73
- frames 98

## G

- getProperty, JSP tag 73

## H

- headings, paragraph styles 57
- HTML
  - common tagging 57
  - editing 37
  - modifying 111
  - SCRIPT tag 118
- HTML editor
  - formatting 47
  - insert form field 64
  - page title 46
  - starting 43
  - switching views 42
- HTTP (Hypertext Transfer Protocol)
  - requests and responses, JSPs 2
- hyperlink
  - creating 63
  - using images 105

## I

- image
  - as hyperlink 105
  - creating maps, client side 106
  - creating maps, server side 107
  - inserting 103
  - setting height and width 105
- import, files to Web target 26

- include
  - JSP directive 77
  - JSP tag 73
- initialization parameters property, for JSP targets 244
- inline styles 54, 84
- insert
  - HTML tables 65
  - special symbols 62
- InstaCode
  - about 127
  - examples 128
- Internet Explorer, warning 109

## J

- J2EE, JSP support 1, 27
- Java
  - applets and Java Beans 108
  - class path 112
- JavaBeans
  - listing in System Tree 112
  - using in JSP targets 3
- JavaScript expressions, binding to retrieval arguments 226
- JSP (JavaServer Pages)
  - application logic 3
  - application object 123
  - comments 121
  - custom tag libraries 3, 79
  - custom tags 2
  - declarations 121
  - directives 2, 76
  - error handling 80
  - error pages 81
  - handling requests and responses 2
  - implicit objects 123
  - include directive 77
  - overview 2
  - page directive 77
  - scope 124
  - scripting elements 2, 121
  - scriptlets 121
  - standard tags 2, 72
  - taglib directive 77

## Index

- translating to a servlet class 2
- using JavaBeans in 3
- Web application development and 1
- JSP targets, working with 155
- JSP Web Service Proxy wizard 155
- JSP Web Target wizard 20

## L

- links
  - anchors 62
  - binding types 195
  - using Web DataWindow DTC 230
- listener property, for JSP targets 243
- lists
  - definition-style 60
  - paragraph styles 59
  - types of 59
- load on start-up property, for JSP targets 244
- login config property, for JSP targets 248
- login variables, EA Server 177

## M

- macros, JSP command line deployment 21
- menus, for formatting 49
- METADATA tags 119
- mime mapping property, for JSP targets 244
- multimedia 107

## O

- object model
  - JSP 7
  - Web Target object model 5, 213
- objects 211
  - 4GL Web page 138
  - array indexes 121
  - assigning IDs 120, 125
  - editing in scripts 125
  - implicit 122
  - in Web Target object model 137
  - inserting from System Tree 127

- session 143
- supporting Web DataWindows 137
- Web target 136
- options, system settings 43
- out object, JSP 123

## P

- page directive for a JSP 77
- page object for a JSP 123
- page properties, title 46
- page request processing 189
- page scope for JSP 124
- Page view 37
  - about 37
  - add page title 46
  - edit 49
- pageContext object for a JSP 123
- parameters
  - 4GL Web pages 169
  - passing with hyperlinks 196
- params, JSP tag 73
- plugin, JSP tag 73
- plug-ins
  - Internet Explorer 108
  - Netscape 108
- position, absolute on page 67
- properties
  - JSP targets 239
  - Web pages 46
  - Web targets 22

## Q

- Quick Web Page wizard 44

## R

- redirection
  - hyperlink 193
  - managing server redirection 200
- redo changes 56
- request object, JSP 123

- request scope, JSP 124
- requests and responses, JSPs 2
- resource environment references properties, for JSP targets 247
- resource references properties, for JSP targets 246
- response object, JSP 123
- retrieval arguments, Web DataWindow 224
- runtime errors, displaying 255

## S

- Script editor
  - about 88
  - formatting 130
  - integrated 116
  - object list boxes 120
  - standalone 116
- scripting elements in JSPs 2, 121
- scriptlets, JSP 121
- scripts
  - accessing EAServer components 188
  - adding to 4GL pages 186
  - assigning object IDs 125
  - creating 125
  - creating as separate file 126
  - disabling server scripting for a control 182
  - editing 116
  - editing HTML 116
  - editing server scripts 183
  - external files 119
  - file extensions 116
  - getting values using 145
  - inline event handlers 117
  - read-only 119
  - saving 117
  - server-side 118, 135
  - session variables 143
  - supported languages 117
  - tips and techniques 131
  - using external code 129
  - using Page view 131
  - writing 115, 126
- security constraints property, for JSP targets 247
- security roles property, for JSP targets 249
- server types 15

- servers
  - application 133
  - application and transaction 133
  - deployment 237
  - EAServer components 175
  - session variables 143
  - supported 15
  - transaction 150
  - Web Data Window 206
- servlet class, translating JSPs 2
- session
  - JSP object 123
  - scope in JSP 124
- session timeout property, for JSP targets 242
- setProperty, JSP tag 73
- Source view
  - about 40
  - edit 49
  - format 47
- special symbols, inserting 62
- SSL (Secure Sockets Layer), Web application client 249
- standard tags in JSPs 2, 72
- style sheets
  - about 83
  - removing elements 98
- styles
  - adding 55
  - cascading style sheets 84
  - changing 55, 93
  - defining for elements 94
  - inline 54
- system options 14, 43
- System Tree
  - Components page 108
  - drag and drop from 51
  - inserting attributes from 53
  - Language page 51

## T

- tables 65
- taglib directive, JSP 77
- Taglib property, for JSP targets 245
- title, page property 46

## Index

Tomcat, JSP deployment 20  
toolbars 12  
tracing, 4GL Web pages 257  
troubleshooting  
    4GL Web pages 255  
    JSP targets 259

## U

undo changes 56  
URLs 131  
useBean, JSP tag 73

## V

variables  
    EAServer login 177  
    page and session 173  
video, adding to Web page 107  
view  
    Page 37, 67  
    Preview 41  
    Source 40

## W

warning, ActiveX controls 109  
Web application, JSP security constraints 247  
Web DataWindow  
    benefits of DTC 207  
    classes 213  
    creating a DataWindow object 213  
    design-time control (DTC) 208  
    editing 223  
    environment 206  
    implementation 203  
    links to other pages 231  
    page parameters 227  
    presentation 223  
    wizard 44  
Web pages  
    adding 4GL capability 167  
    adding Web DataWindow 208

    creating dynamic 135  
    managing data 140  
    page navigation 193  
    parameters and variables 140  
    passing parameters 141  
    *see also* 4GL Web pages 169  
    specifying target pages 195  
    templates 134  
    title 46  
    types 134  
    wizard 44  
Web services  
    custom tags 160  
    troubleshooting 259  
    using the wizard 155  
Web Target object model 5, 213  
Web targets  
    building 234  
    creating 22  
    deployment 233  
    introduction 5  
    running 255  
web.xml file, JSP targets 239  
welcome file list property, for JSP targets 245  
wizards  
    4GL JSP Page 44, 165  
    JSP DataWindow Page 44  
    JSP Page 44  
    JSP Web Service Proxy 155  
    JSP Web Target 20  
    Quick JSP Page 44  
    Table 65