WEBOBJECTS TOOLS AND TECHNIQUES

Apple, NeXT, and the publishers have tried to make the information contained in this manual as accurate and reliable as possible, but assume no responsibility for errors or omissions. They disclaim any warranty of any kind, whether express or implied, as to any matter whatsoever relating to this manual, including without limitation the merchantability or fitness for any particular purpose. In no event shall they be liable for any indirect, special, incidental, or consequential damages arising out of purchase or use of this manual or the information contained herein. NeXT or Apple will from time to time revise the software described in this manual and reserves the right to make such changes without obligation to notify the purchaser.

This manual describes WebObjects 3.5

Writing: Ron Karr
Production: Gerri Gray
Art: Karin Stroud
With help from: Andy Belk, Craig Federighi, John Graziano, Ben Haller, Charles Lloyd, Jean Ostrem, Kelly Toshach, Greg Wilson

Contents

Table of Contents

Chapter 1

Setting Up WebObjects Applications

Introduction

To develop server-based WebObjects applications, you use two primary tools: Project Builder and WebObjects Builder. These tools help you perform the major tasks required to develop your application

Project Builder is an integrated software-development application. It contains a project browser, a code editor, build and debugging support, and many other features needed to develop an application. It helps you to:

- Create and manage your project.
- Write code to provide behavior in your application.
- Build and launch your application.

This chapter discusses the main features of Project Builder that you use when developing WebObjects applications. For more information on Project Builder's other features, refer to its on-line help.

WebObjects Builder is described in the chapters Editing With WebObjects Builder and Working With Dynamic Elements.

This document refers to example projects located in the /**Apple/Developer/Examples/WebObjects** directory.

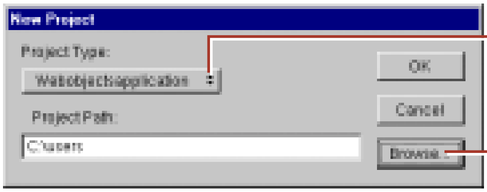
Creating WebObjects Application Projects

A WebObjects application project contains all the files needed to build and maintain your application. You use Project Builder to create a new project.

1. Launch Project Builder.

   On Rhapsody, choose Project Builder from the Apple menu under Developer Applications. On Windows NT, you can launch Project Builder from the WebObjects program group in the Start menu.

2. Choose Project ► New.



Project Type: Webobjectsapplication

Project Path: C:\users

Set project type here.

Click to choose directory in which to create your project.

The New Project panel has a Project Type pop-up list that lets you choose the type of project you want to create.

3. In the Project Type pop-up list, make sure Webobjectsapplication is selected.

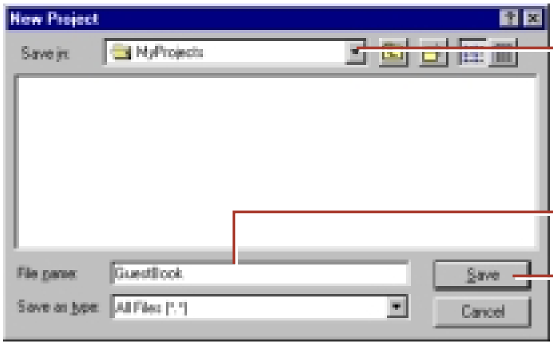
   Another type of project you may want to create is Webobjectsframework. See Frameworks for more information.

4. Click Browse to specify your project's location.

   Type your project's location and name directly in the Project Path text field.

   Note: On Mac OS X Server, the file browser and the Project Type pop-up list both appear on the New Project panel.

5. Navigate to the directory in which to create your project.



Save in: MyProjects

File name: GuestBook

Save as type: All Files (*.*)

Choose a directory here.

Type the project name here.

Click when finished.

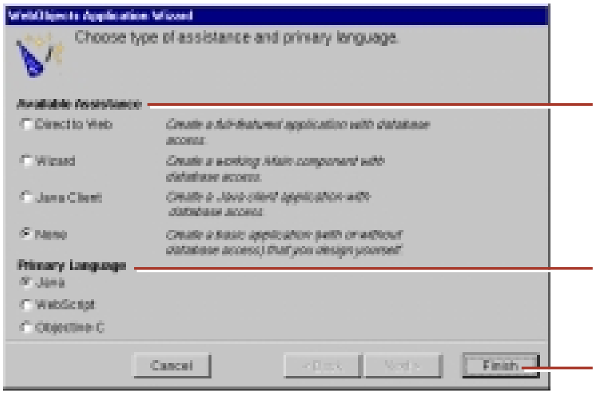
6. Type the name of the project you want to create in the File name field.

7. Click Save.

   The New Project panel now shows the path you specified.

8. Click OK.

   The WebObjects Application Wizard launches.



Choose type of assistance and primary language.

Available Assistance

○ Direct to Web — *Create a full-featured application with database access.*

○ Wizard — *Create a working Main component with database access.*

○ Java Client — *Create a Java client application with database access.*

● None — *Create a basic application (with or without database access) that you design yourself.*

Primary Language

● Java

○ WebScript

○ Objective-C

Choose level of assistance.

Choose programming language.

Click to proceed.

Choosing Assistance

If your application doesn’t access a database, choose None for Available Assistance.

If your application accesses a database, you also choose None if you want to develop the entire application yourself. However, you may wish to use one of the levels of assistance that WebObjects provides. Information about the Direct to Web option can be found in the Direct To Web chapter. Information about the Database Wizard can be found in “Creating a WebObjects Database Application” in *Getting Started With WebObjects*. Information about the Java Client option can be found in *Creating a Java Client WebObjects Application*.

Choosing the Programming Language

WebObjects supports three languages:

- Java
- Objective-C
- WebScript

Java and Objective-C are *compiled* languages. WebScript, which is based on Objective-C, is a *scripted* language. A scripted language allows you to make changes to your application without compiling. When you use compiled code, your application runs faster, but you must continually build your application before running it.

Java files have the extension **.java**, Objective-C files have the extension **.m**, and WebScript files have the extension **.wos**.

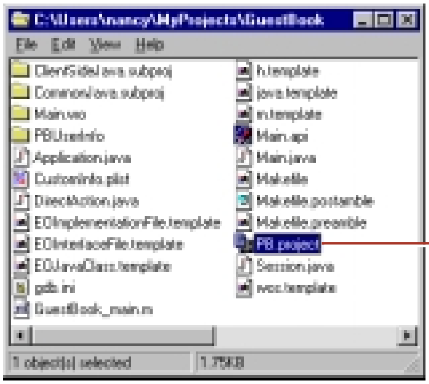
The language you choose in the Wizard applies to the following files:

- The Main *component*. A component in WebObjects represents a page in your application (or possibly part of a page). When you create your project, Project Builder provides you with an initial component called Main. The component's code file implements the behavior of the component.
- The *application* and *session* code files. Application code contains variables and methods that affect the entire application. Session code contains variables and methods that affect a single user's session.

If, for example, you specify Java as your primary language, the Wizard will create the files **Application.java**, **Session.java**, and **Main.java** for you. You can mix languages in a project by choosing a different language when you create individual components.

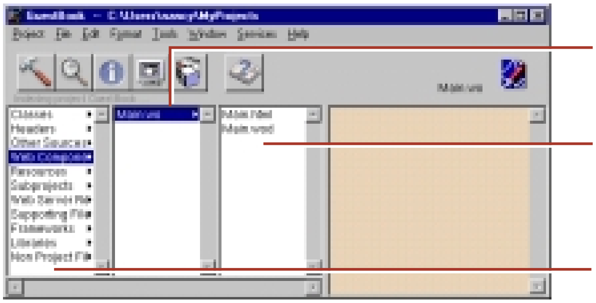
The Structure of a WebObjects Application Project

On disk, your project is a folder whose name is the project name. The project folder contains all the files in your project. The file **PB.project** is the *project file*. You can open a project by double-clicking this file.



Double-click here to open your project.

Project Builder displays a browser showing the contents of your project. The first column lists several categories of files that your project may contain. The following sections describe these categories.



Your project's components. Double-click to edit in WebObjects Builder.

Files in the selected component. Click to display their contents in Project Builder.

Categories ("suitcases") of project resources.

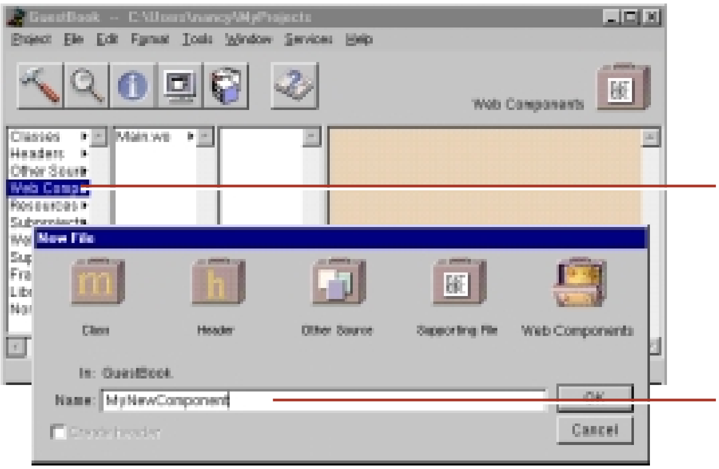
Adding or Deleting Items From a Project

As you work with your project, you'll use Project Builder to create new items (such as components or classes) or to add files (such as images) that you created with other programs to the project. For items in certain suitcases (such as subprojects and frameworks), there's a specific command to add them, discussed in the section about the suitcase. For other suitcases, you use the following procedures.

To create a new item of a particular kind:

1. Select the appropriate suitcase in the first column of the browser.

2. Choose File ► New in Project.



In: GuestBook

Name: MyNewComponent

Click here, then choose File->New in Project.

Enter new file name here.

The New File panel comes up with the suitcase you selected open by default.

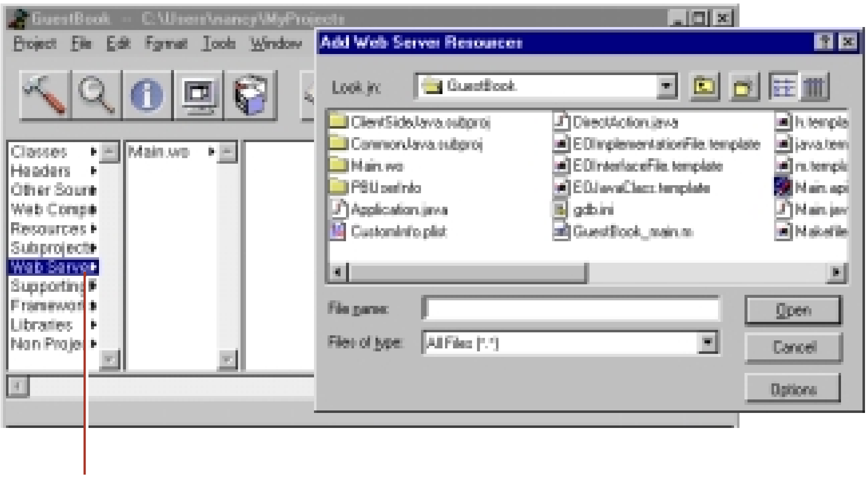
3. Type the name of the item and click OK.

   The new item is added to your project.

To add an existing item (for example, a component, a framework, or a source file) to a suitcase, first select the suitcase in the first column of the browser. Then either:

- Double-click the suitcase.
- Double-click the suitcase icon at the top right of the browser window.
- Choose Project ► Add Files.

A panel called Add *SuitcaseName* appears, allowing you to find an item to add to the suitcase.



Double-click to add existing files to this suitcase.

In addition, you can drag a file directly onto the suitcase icon in the browser, and the file is copied into the project and added to the suitcase.

To delete items from a project:

1. Select one or more items in the browser.

2. Choose Project ► Remove Files.

   A panel appears, giving you the option of removing the files from the project only or from the disk as well.

Web Components

A *component* represents a page, or part of a page, in your application. An application can have one or more components.

Every application starts with a component called Main, which is shown in the Web Components suitcase in the second column of the browser as **Main.wo**. All components have the **.wo** extension.

If you double-click a component, WebObjects Builder opens the component for editing. Editing With WebObjects Builder shows how to edit your component using WebObjects Builder.

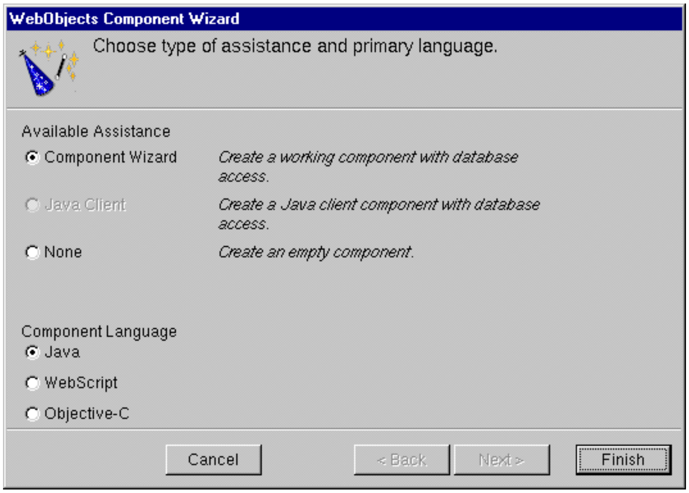
On disk, a component is represented as a folder with the **.wo** extension. Every component has several files that specify the component's look and behavior. The name of each one is the component's name followed by a specific file extension. These are the files in the Main component:

- **Main.html** is the HTML template for the component. This file contains HTML tags, just like any web page; in addition, it typically contains tags for dynamic WebObjects elements.
- **Main.wod** is the *declarations file* that specifies bindings between the dynamic elements and variables or methods in your code.
- **Main.woo** is used to store information about display groups (if your project accesses a database) and encodings for HTML templates. You should never edit this file (it does not appear in Project Builder's browser).

To create a new component:

1. With Web Components selected in the first column of the browser, choose File ► New in Project.
2. In the New File panel, type the name of your project and click OK.

   The WebObjects Component Wizard appears.



WebObjects Component Wizard

Choose type of assistance and primary language.

Available Assistance

● Component Wizard — *Create a working component with database access.*

○ Java Client — *Create a Java client component with database access.*

○ None — *Create an empty component.*

Component Language

● Java

○ WebScript

○ Objective-C

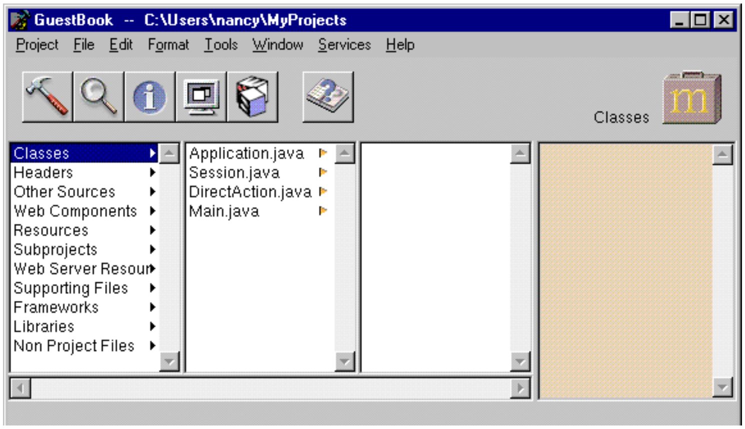
3. If you want the Wizard to assist you in creating a component with database access, choose Component Wizard from Available Assistance; otherwise choose None. See “Creating a WebObjects Database Application” in *Getting Started With WebObjects* for more information on using the Wizard with databases.

4. Specify the language for your component and click Finish.

Note: You can also create components in WebObjects Builder and save them into your project.

Classes

The Classes suitcase contains Java, Web Script and Objective-C classes. For example, if your application’s primary language is Java, this suitcase contains the **Application.java**, **Session.java**, **DirectAction.java**, and **Main.java** files. The files have the extension **.wos** if the primary language is Web Script and **.m** if the primary language is Objective-C. There is a class file for each component, as well as any other classes you add to the project.



You can specify that Java classes are client-side, server-side, or common classes. See Subprojects for more information on how to do this.

Headers

The Headers suitcase contains header files for projects that use Objective-C.

Other Sources

The Other Sources suitcase contains compiled code that doesn't belong to a particular class.

Resources

The Resources suitcase contains files that are needed by your application at run time, but which do not need to be in the web server's document root (and hence will not be accessible to users). It includes:

- Configuration files
- EOModel files
- API files containing the keys defined by a component (for example, **Main.api**) that other components can bind to (see Reusable Components) and the rules for binding the keys. WebObjects Builder uses these files to check if a reusable component is used correctly.

Web Server Resources

The Web Server Resources suitcase contains files, such as images and sounds that must be under the web server's document root at run time. When developing your application, you place these files in your project directory and add them to the project (see Adding or Deleting Items From a Project). When you build your project, Project Builder copies the files in this suitcase into the WebServerResources folder of your application wrapper (see The Application Wrapper).

Subprojects

A subproject has the same structure as a WebObjects Application project. You can use subprojects to divide large projects into manageable chunks.

When you create a new project, ProjectBuilder creates two subprojects (ClientSideJava and CommonJava) in your project folder. By default, they are not added to the Subprojects suitcase. If you need to use them, you must add them to the project. Then you can add your Java classes to the appropriate project as follows:

- Add server-side Java classes to your top-level project.
- Add client-side Java classes to the ClientSideJava subproject.

- Add Java classes that are common to both client and server to the CommonJava subproject.

Note: These subprojects have the makefile variables `JAVA_IS_CLIENT_SIDE` and `JAVA_IS_SERVER_SIDE` set in **Makefile.preamble** so that the appropriate Java code is generated when you build your project.

To create a subproject:

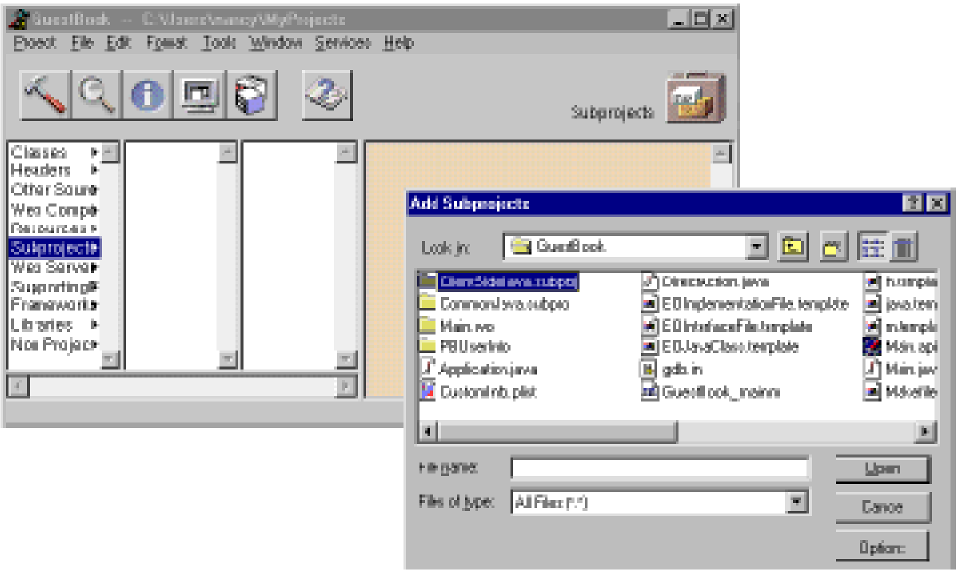
1. Choose Project ► New Subproject.

2. Specify the name of your subproject in the New Subproject panel, the type of project from the Type pull-down menu, and click OK.

   A subproject is created inside the project, with a similar structure to the top-level project. You can add items to the subproject in the same way that you add items to the top-level project.

To add an existing subproject (such as ClientSideJava or CommonJava) to your project:

1. Double-click Subprojects in the first column of the browser.



2. In the Add Subprojects panel, navigate to the directory of the subproject you want to add and click Open.

Supporting Files

The Supporting Files suitcase contains your project's **Makefile** (which you should not edit since this file is maintained by Project Builder), as well as **Makefile.preamble** and **Makefile.postamble**, which you can modify in order to customize the build. You can add other files your project may need (such as Read Me documents) to this suitcase; these files will become part of your application when you install it (see Installing Your Application.)

Frameworks

A *framework* is a collection of classes and resources that an application can use. By storing items such as components and images in frameworks, you can reuse them in multiple projects without having to create multiple copies.

Every WebObjects Application project includes several frameworks by default. When you build, your application links with these frameworks. They are:

- WebObjects: The basic WebObjects classes.
- WOExtensions: Extensions to the WebObjects framework.
- EOAccess: The Enterprise Objects Access Layer.
- EOControl: The Enterprise Objects Control Layer.
- Foundation: Basic object classes that most applications use (for example, strings, numbers, and arrays).

You can include additional system frameworks in your project if you need to. To add an existing framework to your project:

1. Double-click Frameworks in the first column of the browser.

2. In the Add Frameworks panel that appears, select a framework to add and click Open.

In addition, you can create your own frameworks in order to share WebObjects components and resources across multiple applications. To create a WebObjects Framework:

1. Choose Project ► New.

2. Select Webobjectsframework from the pop-up menu.

3. Select the path where you want to create the framework.

Once you have created a framework, you can add components, images, and other items to it in the same way that you would add them to a project. To make your framework accessible to other applications, you must install it (see Installing Your Application for more information). See Reusable Components for more information on using components that live in frameworks.

Libraries

The Libraries suitcase contains libraries that your application links to.

Non Project Files

The Non Project Files suitcase is used for files that you have opened that aren't part of the current project.

Opening an Existing Project

To open an existing project from Project Builder:

1. Choose Project ► Open.

2. In the Open Project panel, navigate to the project folder and click Open.

To open an existing project from the file system, double-click the **PB.project** file in the project directory. Project Builder launches (if it is not already running) and opens the project.

Editing Your Project's Source Files

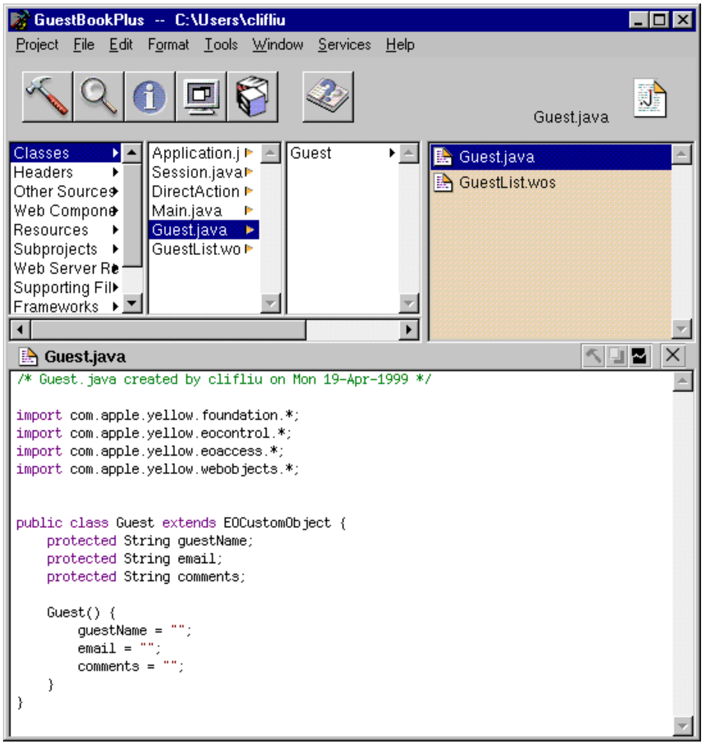
Every component in your project has a code file whose name is the name of the component followed by the appropriate extension (**.java** for Java, **.m** for Objective-C, and **.wos** for WebScript). Your project may use different languages for different components.

Each component's code specifies the component's behavior. Each component is actually a subclass of WOComponent. This class has standard methods (such as **awake** and **init**) that you may want to override (see *WebObjects Developer's Guide* for more information on these methods). You can also write your own methods and bind them to dynamic elements in your component (see Working With Dynamic Elements, as well as the *Dynamic Elements Reference*, for information on binding dynamic elements).

In addition to the component's code, each project has an *application code file* (**Application.java**, **Application.m**, or **Application.wos**) and a *session code file* (**Session.java**, **Session.m**, or **Session.wos**). These files implement your application's custom subclasses of WOApplication and WOSession.

When you first create your project using the Wizard, you specify the language you want to use (see Choosing the Programming Language). This language applies to the application and session code, as well as to the code for your initial component, Main. Other components may be written in different languages.

Regardless of the language you select, all source code for classes appear in the Classes suitcase. On disk, they are located at the top level of the project directory.



```
/* Guest.java created by cliflíu on Mon 19-Apr-1999 */

import com.apple.yellow.foundation.*;
import com.apple.yellow.eocontrol.*;
import com.apple.yellow.eoaccess.*;
import com.apple.yellow.webobjects.*;


public class Guest extends EOCustomObject {
    protected String guestName;
    protected String email;
    protected String comments;

    Guest() {
        guestName = "";
        email = "";
        comments = "";
    }
}
```

To save changes in your code, choose File ► Save.

Note: WebObjects Builder gets information from Project Builder about variables and methods in your code. If you add or delete a variable or method, WebObjects Builder doesn't get the updated information until you save the file.

Editing Your Component's HTML and Declarations Files

While you must use Project Builder to edit your components' code or script files, you typically use WebObjects Builder's graphical interface to generate the HTML and declarations files. You can, however, also edit these files using Project Builder.

To edit a component in WebObjects Builder:

1. Select Web Components in the first column of Project Builder's browser.

2. Double-click the component name (for example, **Main.wo**) in the second column.

   WebObjects Builder launches and opens your component in a window. See Editing With WebObjects Builder for information on using WebObjects Builder to edit your component.

To edit a component in Project Builder:

1. Select Web Components in the first column of Project Builder's browser.

2. Select the component you want to edit in the second column.

3. Select *ComponentName***.html** or *ComponentName***.wod** in the third column.

   The text of the file appears in the lower pane of the browser, where it can be edited.

4. Alternatively, you can double-click the file name or its icon at the top right of the browser, and the file opens in a separate window.

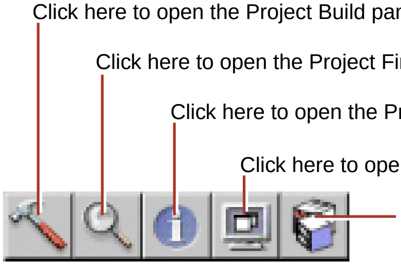
Building Your Application

You must build your application if your project contains any compiled code (Java or Objective-C). If your application uses WebScript only (and you do not modify any web server resources), you do not need to build. In this case, Project Builder runs a default executable (**WODefaultApp**) when you launch your application.

Once you have built your application, you do not need to rebuild unless you have made changes to your compiled code. You can make changes to your components (the **.html**, **.wod**, or **.wos** files) and test them without rebuilding.

For more information on how to build and run your application quickly, see Rapid Turnaround Mode.

Project Builder has a toolbar with buttons you use to build and launch your application.



Click here to open the Project Build panel.

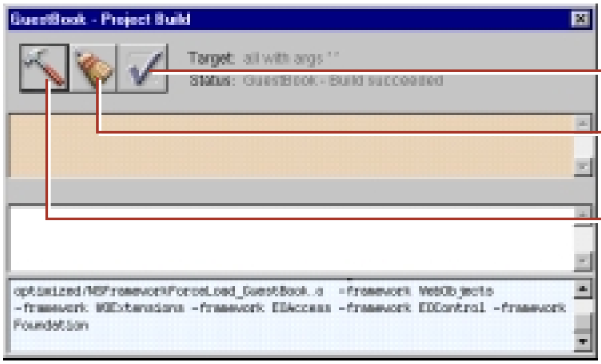
Click here to open the Project Find panel.

Click here to open the Project Inspector.

Click here to open the Launch panel.

Click here to open the Class Browser.

1. Click 🔨 in the toolbar to open the Project Build panel.
2. Click 🔨 in the Project Build panel to build your project.


GuestBook - Project Build
Target: all with args ""
Status: GuestBook - Build succeeded

```
optimized/WOFramework/ForceLoad_GuestBook.o  -framework WebObjects
-framework WOExtensions -framework EOAccess -framework EOControl -framework
Foundation
```

Click here to set build options.

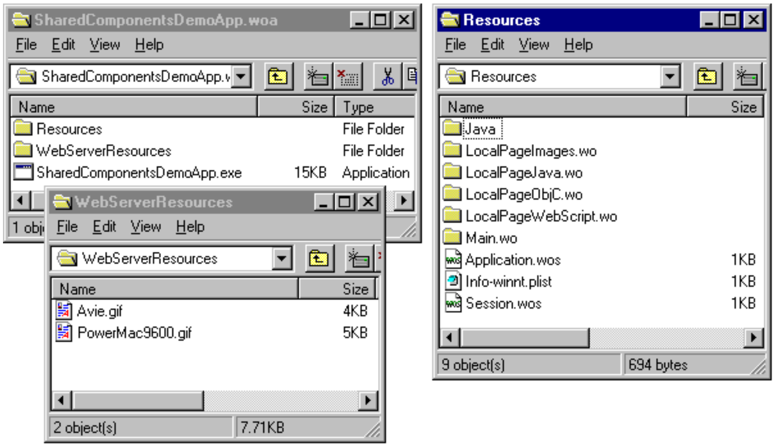
Click here to “clean” the project (delete derived files).

Click here to build your project.

The Project Build panel displays the commands that are being executed to build your project. If all goes well, it displays the status message “*Project Name* - Build succeeded”

The Application Wrapper

When you build your project, Project Builder creates an *application wrapper*, which is a folder whose name is the project name plus the extension **.woa**.



Name	Size	Type
Resources		File Folder
WebServerResources		File Folder
SharedComponentsDemoApp.exe	15KB	Application

Name	Size
Avie.gif	4KB
PowerMac9600.gif	5KB

Name	Size
Java	
LocalPageImages.wo	
LocalPageJava.wo	
LocalPageObjC.wo	
LocalPageWebScript.wo	
Main.wo	
Application.wos	1KB
Info-winnt.plist	1KB
Session.wos	1KB

The application wrapper has a structure similar to that of a framework. It consists of the following:

- The executable application.
- The application's resources.

  These include the application's components as well as other files that are needed by your application at run time.
- The application's web server resources.

When you build and install your application, Project Builder copies all the files from your Web Server Resources suitcase to a folder called WebServerResources inside the application wrapper. If you have client-side Java components in your project, these are also copied to the WebServerResources folder.

Launching Your Application

To launch your application:

1. Click 🖥 in the toolbar to open the Launch panel.
2. Click 🖥 in the Launch panel to launch your application.

When you launch your application, your machine's web browser is launched by default and it accesses your application. To turn off this feature:

1. Click ✓ to bring up the Launch Options panel.
2. Select Environment and Command-Line Arguments from the pop-up menus.
3. Enter `-browser OFF` as a command line option.

You can also launch your application directly from a command line. See *Serving WebObjects* for more information on command line options.

You can also launch your application by double-clicking its executable file. When you build your application, Project Builder creates an executable file (*ProjectName*.**exe** on Windows NT platforms) inside your application wrapper (**.woa**) directory.

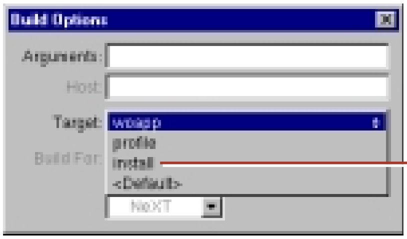
Installing Your Application

Some files in a web application (such as images and sounds) must be stored under the web server's document root in order for the server to access them. The remaining files (such as your components and source code) must be accessible to your application but not necessarily by the web server itself.

In previous versions of WebObjects, it was typical to store the entire project under the web server's document root. This practice has advantages for turnaround time during development. However, in deployment, it presents the possibility of allowing users access to your source code. WebObjects has a “split installation” feature that allows you to install only those files (such as images) that the web server must have access to under the document root. The remaining files can be stored elsewhere.

The same procedure applies to installing WebObjects applications and frameworks. To install:

1. Click ⓘ to open the Project Inspector.
2. Under “Install In:”, set the path where the application wrapper will be installed.
3. In Project Builder's Build panel, click ☑.
4. From the Target pop-up menu, choose **install**. (By default, the target is set to **woapp**.)



Select “Install.”

5. Click 🔨 in the Build panel to install your application.

   The full application wrapper is copied into the “Install In:” directory, and a wrapper containing only the Web Server Resources is copied into the document root.

See *Serving WebObjects* for more information about installing your application.

Rapid Turnaround Mode

With the exception of Objective-C and Java code, WebObjects is an interpreted environment. The HTML templates, declarations files, and WebScript files each represent interpreted languages. One of the main benefits of an interpreted environment is that you don't need to recompile every time you make a change to the project. The ability to test your changes without rebuilding the project is called "rapid turnaround" and, when using the rapid turnaround features, you're said to be in "rapid turnaround mode."

In WebObjects you can modify **.html**, **.wod**, and **.wos** files within application projects, framework projects, and subprojects of either applications or frameworks without rebuilding your project.

To support rapid turnaround, WebObjects must be able to locate the resources of your application and its associated frameworks within your system's projects rather than the built products (the **.woa** or **.framework** wrappers). To tell WebObjects where to look for your system's projects you must define the NSProjectSearchPath user default. Set this default to an array of paths where your projects may be found. (Relative paths are taken relative to the executable of your project.) The order of the entries in the array defines the order in which projects will be located. The default NSProjectSearchPath is ("../.."), which causes WebObjects to look in the directory where your application's project resides for any other applicable projects. For example, if your application's executable resides in:

```
c:\web\docroot\WebObjects\Projects\MyProject\MyProject.woa
```

then from the executable's directory, "../.." would point to:

```
c:\web\docroot\WebObjects\Projects
```

If you've set your project's "Build In" directory to something other than the default, "../.." isn't likely to be appropriate; you should set your NSProjectSearchPath to point to the directories where you keep your projects while you work on them.

When your application is starting up, pay close attention to those log messages which indicate that a given project is found and will be used instead of the built product. Many problems can be solved by understanding how to interpret this output. If no such log message is seen for a given project, it won't be possible to use rapid turnaround for that project. As well, if you have several projects with the same name in the same directory, a conflict will be reported. This often happens when you have several copies of the same project as backups in your project directory. For example, you might have:

```
c:\web\docroot\WebObjects\Projects\MyApp
c:\web\docroot\WebObjects\Projects\Copy of MyApp
c:\web\docroot\WebObjects\Projects\MyAppOld
```

Even though the folders containing the projects have different names, the **PB.project** files within them might be identical. WebObjects uses the PROJECTNAME attribute inside your project's **PB.project** file to determine the name of the project, not the name of the directory for the project. If this happens, you'll need to move the backups to another directory to avoid the conflict.

Rapid Turnaround and Direct Connect Mode

Direct connect mode allows you to test your application without involving a web server. This means that you don't have to install your WebServerResources under the document root of your web server. The result is that rapid turnaround is even more convenient when in direct connect mode because you don't need to rebuild to install WebServerResources changes to the document root.

Testing With a Web Server

When you're working in direct connect mode, few issues arise with respect to rapid turnaround. If your application has features which require a web server be used even for testing, however, there are a couple of things to know to make rapid turnaround work for you. Specifically, since you are relying on the web server to locate files within WebServerResources, you must follow these guidelines:

1. Your projects must reside somewhere below your web server's document root.

2. NSProjectSearchPath should point to all projects of interest.

3. For application projects, the WOApplicationBaseURL user default should specify the directory containing the application project. For example, if your application's project folder is:

   ```
   c:\web\docroot\WebObjects\MyApp
   ```

then the WOApplicationBaseURL user default must be "/WebObjects".

4. For framework projects, the WOFrameworksBaseURL user default should specify the directory containing all framework projects used by the application. For example, if your application uses MyFramework.framework and that project resides in:

   ```
   c:\web\docroot\WebObjects\Frameworks\MyFramework
   ```

   then the WOFrameworksBaseURL user default must be "/WebObjects/Frameworks".

Conveniently, the two examples above use the default locations for WOApplicationBaseURL and WOFrameworksBaseURL; if your projects reside in these default locations, you need only set NSProjectSearchPath.

Also, while it is possible to point WOApplicationBaseURL and WOFrameworksBaseURL to other locations, it is not recommended that WOFrameworksBaseURL be moved since all WebObjects applications use WOExtensions.framework, which resides in the default location. If you set WOFrameworksBaseURL to point elsewhere, one side effect will be that the images in the “Raised Exception” panel will not render.

Chapter 2

Editing With WebObjects Builder

Introduction

WebObjects Builder is an application that provides graphical tools for creating dynamic web pages (*components*). This chapter describes the basic procedures for creating your components' content with WebObjects Builder.

A web page consists of *elements*. WebObjects Builder allows you to add most of the common HTML elements to a component by using its graphical editing tools. You can type text directly into a component window and you can add additional HTML elements by using the buttons in the toolbar (or their menu equivalents).

In addition, WebObjects allows you to create *dynamic elements*, whose look and behavior are determined at run time. This chapter focuses on basic editing tasks and the use of standard HTML elements. The next chapter, Working With Dynamic Elements, provides more specifics on using dynamic elements.

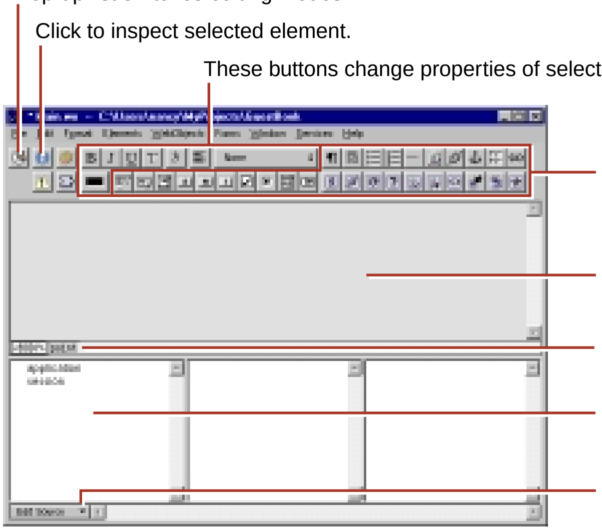
The Component Window

When you open a component, WebObjects Builder displays it in a *component window*. You create your component's appearance graphically in the upper pane of the component window. The browser at the bottom of the window (known as the *object browser)* displays variables and methods your component uses.

Pop-up list switches editing modes.

Click to inspect selected element.

These buttons change properties of selected elements, or text.



Click one of these buttons to create a specific element.

This window displays your component's elements graphically.

Click here to select another element in the hierarchy.

Object browser shows variables and methods in your application's code

Pull-down list lets you add variables, and methods to

The *path view* lies between the two panes. It displays the element path to the selected element. Any element can be contained in a hierarchy of several levels of elements and can in turn contain other elements. Here, the path view shows that the cursor is contained in the HTML <BODY> tag, which is at the top level of the hierarchy. You can click on an element in the path view to select it.

The toolbar at the top of the window contains buttons you use to create the content of your component. WebObjects Builder also has menu commands corresponding to these buttons.

Editing Views

WebObjects Builder allows you to view and edit your page in three ways: *layout view*, *preview view*, and *source view*. To switch between these views, use the pop-up list at the left of the toolbar.

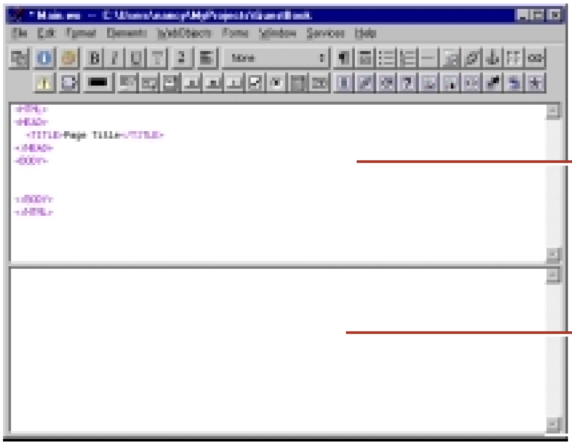
- The layout view shows a visual representation of your component, including its dynamic elements. The bottom pane, called the object browser, lists the variables and methods that are defined in your scripts or code files.

- The preview view shows a visual representation of your component similar to the layout view except that the elements are collapsed as much as possible and bindings are not displayed. This view displays as closely as possible what the user sees in the browser.

- The source view shows the text of your component's HTML template in the upper pane and the text of your declarations (**.wod**) file in the lower pane. In this view, you can enter any HTML code. For example, you can include HTML elements that are not directly supported by WebObjects Builder's layout tools. You can also add elements using the toolbar.

Source View

When you choose the source view, the text of your HTML template (*ComponentName***.html**) appears. When you add elements in the layout view, their corresponding HTML tags appear in this file.



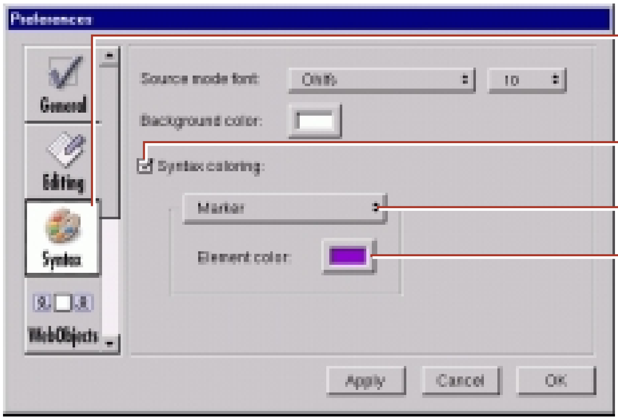
```
<HTML>
<HEAD>
  <TITLE>Page Title</TITLE>
</HEAD>
<BODY>


</BODY>
</HTML>
```

The HTML source for your component.

Information about bindings is displayed here.

As you can see, when you begin with a blank page, WebObjects Builder automatically inserts the necessary elements such as <HTML>, <HEAD>, and <BODY> for you. The tags and text display in different colors according to their type: errors are red, markers are purple, comments are grey, WebObjects are blue, and text is black. You can change these colors using the Preferences panel. Choose Edit ▶ Preferences (Edit ▶ Options on Windows NT) to bring up the panel and click Syntax to set the syntax coloring preferences.



Source mode font: Ohlfs 10

Background color:

☑ Syntax coloring:

Marker

Element color:

Click to set syntax coloring preferences.

Click to enable and disable syntax coloring.

Click to choose tag.

Click on border to choose color for the tag.

You select text elements as you would in most text-editing applications: by dragging, or by double-clicking words, or by Shift-clicking. The selected text appears shaded. When you triple-click on a tag, you select the tag, the matching tag, and the HTML in between. For example, if you triple-click on <TITLE> in a new page, WebObjects Builder selects <TITLE>, </TITLE>, and the

"Page Title" text. You can also drag the mouse after you triple-click, which selects the matching tag and the HTML in between for the tag the mouse is on as you drag it.

The bottom pane of the source editing window shows your declarations (**Main.wod**) file. When you bind variables and methods to your dynamic elements, this file stores the binding information. Normally, you don't edit this file directly. Chapter 3, Working With Dynamic Elements shows how you use WebObjects Builder to create bindings. Refer to the *WebObjects Developer's Guide* for more information on working with the declarations file.

Selecting Elements in the Layout and Preview Views

There are several operations you perform in WebObjects Builder that require you to select an element, such as copying, deleting, inspecting, or "wrapping" one element inside another.

You select text elements as you would in most text-editing applications: by dragging, or by double-clicking words, or by triple-clicking lines, or by Shift-clicking. The selected text appears shaded.

Some elements (such as text fields and text areas) can be selected simply by clicking them; they appear shaded.

Other elements (such as tables) require you to click outside the element and drag across it in order to select it.

To select a range of elements, drag across them, or press the Shift key while clicking at the end of the range.

You can also select elements in the path view by clicking the HTML tag corresponding to the element.

Context Menus

The elements in the edit window and in the path view have context menus that perform various operations on the element or its content. To access these menus, Control-click the element in Rhapsody or right-click the element in Windows NT.

Objects in the object browser also have context menus; to access them, Control-click the object in Rhapsody or right-click the object in Windows NT.

The WebObjects Builder Toolbar

At the left of the toolbar are five buttons:

The ✎ pop-up list allows you to switch editing views. See Editing Views.

The ℹ button brings up the Inspector window, which allows you to set various attributes of the currently selected element. The Inspector section describes each type of element in more detail.

The 🎨 button brings up the Palette window. See Palettes for more information on creating and using palettes.

The ⚠ button brings up the Validation window, which displays any binding validation errors on the page. See “Deleting Bindings” (page 77) for more information about binding validation.

The 🧩 button brings up the API editor, which allows you to set the binding specifications for a reusable component.

Entering Text

When you begin to edit a new component, the cursor (insertion point) appears at the upper left of the screen. You can begin typing text directly; the text appears in the default font and size. If you press Shift-Enter, a line break (<BR> element) is inserted after the line. If you want a paragraph element (<P>), press Enter. See Structure Elements for information on other types of text elements.

The top row of the toolbar contains a set of buttons that operate on the currently selected text. If no text is selected, they change the setting for any text typed in at the insertion point.

B *I* U T: In the layout view, these buttons allow you to toggle the style of the currently selected text. You can set any combination of bold (<B>), italics (<I>), underline (<U>) and typewriter (fixed-width) font (<TT>). In the source view, these buttons insert the code for the text style at the insertion point, or around selected text.

3 : This pop-up list allows you to set the font size of the currently selected text.

: This color well allows you to set the color of the currently selected text. To change the color, click on the border of the color well and select a color from the Colors panel. See Setting Colors for more information.

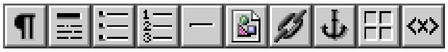
: This pop-up list allows you to center or justify text.

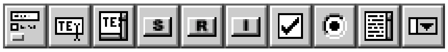
: This pop-up list allows you to choose the font of the currently selected text.

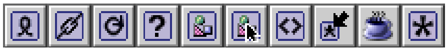
If you make a mistake, simply choose Undo from the Edit menu.

Creating Elements With the Toolbar

To create HTML elements, you use the buttons in the toolbar. There are three groups of buttons.

- **Structures** . Use these buttons to create paragraphs, lists, images, and other static HTML elements. See Structure Elements for more information.

- **Dynamic form elements** . Use these buttons to create form elements in which users enter information. WebObjects gives your application access to the data entered by users by allowing you to associate, or *bind*, these elements to variables in your application. See Creating Form-Based Dynamic Elements for more information.

- **Other WebObjects** . Use these buttons to create other dynamic elements, which you can bind to variables and methods in your program to control how they are displayed. Some of these (such as hyperlinks) have direct HTML equivalents. Others are *abstract dynamic elements*, such as repetitions and conditionals, which determine how many times an element is displayed or whether it is displayed at all. See Creating Other WebObjects for detailed information.

The general procedure for creating an HTML element is:

1. Place the cursor where you want the element to appear on the page.

2. Click the toolbar button representing the element you want.

   The element is placed at the cursor position.

3. Select the element (see Selecting Elements in the Layout and Preview Views). In most cases, the element is already selected when you create it.

4. Bring the Inspector to the front by clicking it. If it is not open, click ℹ.

   In the Inspector, you can set various properties of the element. For example, you can change a paragraph's type from plain to preformatted.

It's important to be aware of what happens when you have text or other elements selected and you create a new element:

- If the new element is a *container* element (that is, it can contain other elements), the selected elements are "wrapped" or contained inside the new element.

- If the new element cannot contain other elements (for example, a horizontal rule or image), the new element replaces the selection.

Menu Equivalents For Toolbar Commands

All the toolbar buttons have menu equivalents. This document refers to the toolbar buttons, but of course you can use the menu commands as well:

- The Edit menu contains Preview, Layout, and Source menu items for switching between editing views (or use the keyboard equivalents).

- The Format menu contains equivalents for the buttons that affect the selected text or element.

- The Elements menu contains equivalents for all the buttons that create static elements.

- The WebObjects menu contains equivalents for all the buttons that create dynamic elements that are not form elements.

- The Forms menu contains equivalents for all of the buttons that create dynamic form elements.

- The Window menu contains commands to open the Inspector, Palette, API Editor, Validation, and Color windows (and other commands).

Note: The menus contain many elements not available in the toolbar.

The Inspector

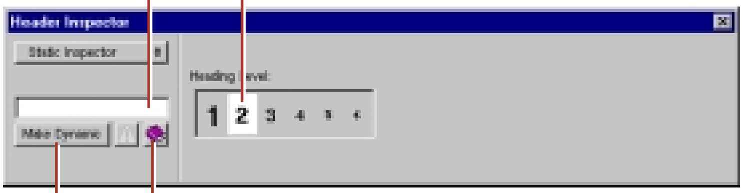
You use the Inspector to set HTML attributes of the elements in your component.

To open the Inspector, click ℹ. You can also Control-click the element on Mac OS X Server, or right-click the element on Windows and choose “Inspect” from the context menu that appears. Double-clicking many elements also opens the inspector.

The Inspector’s title and contents reflect the element you’ve selected in the component window. Each element has its own Inspector that allows you to set properties appropriate for the element. For example, the Heading Inspector shown here allows you to set the level of a heading element. Other elements have different properties that you can set.

This textfield contains the element’s HTML name attribute.

Click here to set the heading level.



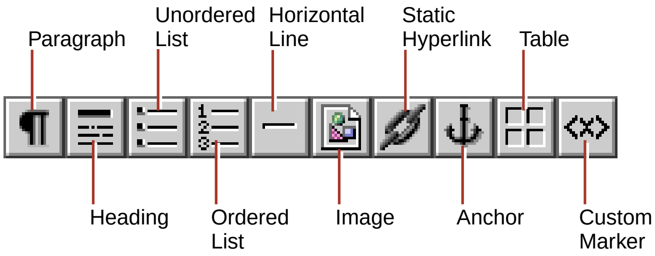
Click to view documentation about this element..

Click here to make static elements dynamic and vice versa.

The Make Dynamic button in the Inspector allows you to convert an HTML element into a dynamic WebObjects element. Dynamic elements have a Make Static button, which allows them to be converted to their static counterparts. This feature is discussed in more detail in Dynamic and Static Inspectors.

Structure Elements

The structures section of the toolbar looks like this:



The following sections describe the elements you can create with these buttons.

Paragraphs

Click ¶ to create a new paragraph. If there is a text selection, the entire selection becomes a paragraph.

You can use the Inspector to set the paragraph to one of the following tags:

- Plain (<P>)
- Preformatted (<PRE>)
- Address (<ADDRESS>)
- Block quote (<BLOCKQUOTE>)
- Division (<DIV>)

Headings

Click ≣ to create a heading. By default, an <H1> element is created. You can use the Inspector to change the level of the heading to between <H1> and <H6>. The heading elements are also available in the Elements menu.

Unordered Lists

Click ☰ to create a new unordered list (<UL>). If there is a selection, each line in the selection becomes a list item (<LI>). You can use the Inspector to change the list to an ordered list (<OL>). You can also change the bullet character.

When typing in a list:

- Press Enter to create a new list item.
- Press Shift-Enter to create a line break but no new list item.
- Press Tab to create a new list nested inside the original list.
- Press Shift-Tab to move the nesting back one level.

Ordered Lists

Click ☰ to create a new ordered list (<OL>). If there is a selection, each line in the selection becomes a list item (<LI>). You can use the Inspector to change the list to an unordered list (<UL>). You can also change the way in which lists appear; for example, displaying an ordered list in Roman numerals.

Typing in an ordered list is the same as typing in an unordered list (see Unordered Lists above.)

Horizontal Rule

Click — to create horizontal rule (<HR>) element. You can use the Inspector to vary its height and width, and whether it is displayed in as a flat or shaded line.

Images

Click 🖼 to add a static image (<IMG>). A Select Image panel appears, allowing you to select an image file to display at the insertion point. The Inspector allows you to change the image's properties, including its size, file path, and whether it uses an absolute or relative reference.

With static images, you must specify a known file path. You can also create a *dynamic image*, which refers to an image file that lives in your project or in a framework. See Dynamic Images for more information.

To set an image for the page background, see Setting Page Attributes.

Static Hyperlinks

There are two types of hyperlinks that you can use in a WebObjects application:

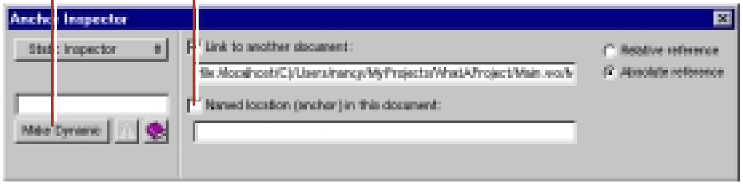
- A static hyperlink (which uses the HTML <A> tag), whose destination is constant.

- A dynamic hyperlink (WOHyperlink), whose destination can be specified at run time. See Dynamic Hyperlinks for more information about these.

To create a static hyperlink:

1. Click 🔗 on the toolbar.
2. Type the text that the hyperlink should contain. As you type, the text is underlined.

   Alternatively, you can select existing text and then click 🔗 to convert the text to a hyperlink.
3. Use the Inspector to set the destination of the link. You can also change the hyperlink to an anchor.



Click to make hyperlink dynamic.

Click to make this element an anchor that can be the destination of a link.

Anchor Inspector

Static Inspector

☑ Link to another document:

file://localhost/C|/Users/nancy/MyProjects/WhatAProject/Main.wo/

○ Relative reference

● Absolute reference

☐ Named location (anchor) in this document:

Note: While the destination of a static link cannot change, it’s possible to vary its text at run time by using a dynamic string (see Dynamic Strings) inside the hyperlink.

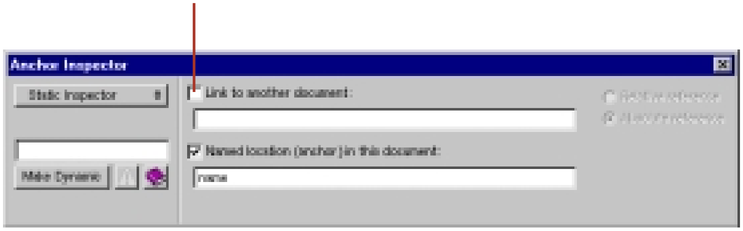
Anchors

An anchor (<A>) is a named location within a document that can be referenced by a hyperlink. Click ⚓ on the toolbar to create an anchor.

1. Type the text that the anchor should contain. As you type, the text is underlined.

   Alternatively, you can select existing text and then click ⚓ once to convert the text to an anchor.

2. Use the Inspector to set the name of the anchor. You can also change the anchor to a hyperlink.



Click to make this element a hyperlink.

Anchor Inspector

Static Inspector

□ Link to another document:

☑ Named location (anchor) in this document:

name

Make Dynamic

Note: While the anchor's name cannot change, it's possible to vary its text at run time by using a dynamic string (see Dynamic Strings) inside the anchor.

Tables

Click ⊞ to add create a table. See "Working With Tables" (page 47) for more information about creating tables.

Custom Marker

Not all legal HTML elements can be created directly using WebObjects Builder's buttons or menu commands. However, you can create any type of element using the custom tag.

To create an HTML element using a custom marker:

1. Place the cursor where you want the element.

2. Click <x>.

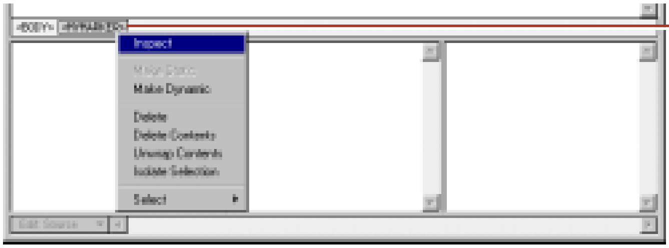
3. In the panel that appears, enter the tag's name in the "Tag to use:" field.

4. If the element doesn't require an end tag, uncheck "New marker is a container (allow children)."

5. Click OK.

   MYMARKER Custom Marker /MYMARKER appears in the component window. You can replace the text "Custom Marker" with the content of the element (if any).

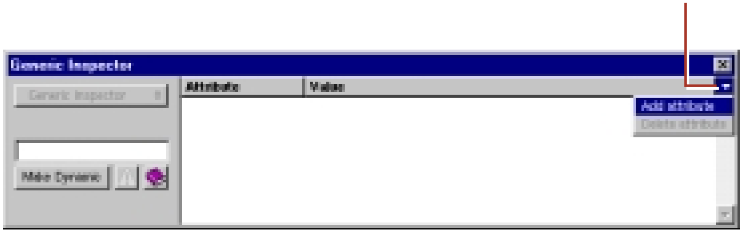
6. If the element has attributes you want to specify, Control-click the tag in the path view on Rhapsody (right-click on Windows NT) and choose Inspect.



Control-click on Rhapsody or right-click on Windows NT to inspect the tag.

In the Generic Inspector that appears, select Add Attribute from the attribute pull-down list (alternatively, you can press Enter.) Type the attribute's name, press Tab, and enter the attribute's value.

Click to add an element attribute.



You can also enter the source view and type the marker and its text directly.

Tip: To save a custom element so you can use it again, save it on a palette. See Palettes.

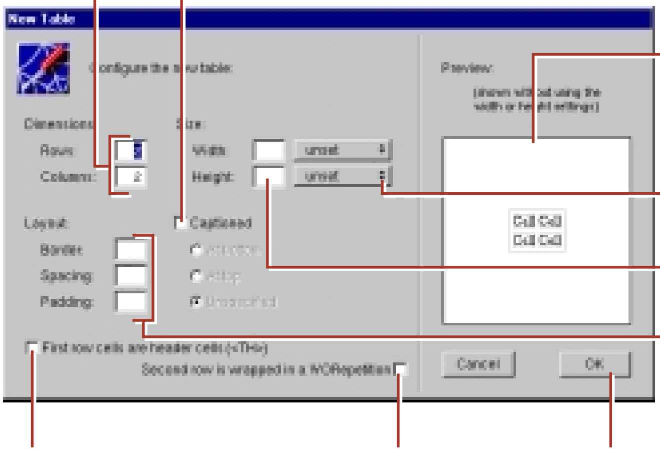
Working With Tables

Creating Tables

To create a table, click ⊞ from the toolbar. The table panel appears, which allows you to configure the table before you insert it.

Enter number of rows and columns here.

Click here to add a table caption.



This pane shows what your table will look like.

Choose pixels or percent here before you ...

... enter the table height here.

Enter table's BORDER, CELLPADDING, and CELLSPACING attribute values here.

Click here to use <TH> in the first row..

Click here to repeat the second row using a WORepetition.

Click here to insert table into your page.

To change the number of rows or columns in the table, use the Dimensions fields. When you press Enter, the preview pane shows what your table will look like.

To change the width of the table, select a unit for the width (pixels or percent) in the Width pop-up list. Enter the width in the Width field, which is now active. The preview pane is not affected by the table width setting.

To change the height of the table, select pixels in the Height pop-up list. Enter the height in the Height field, which is now active. The preview pane is affected by the table height setting.

To add a caption to the table, click the Captioned checkbox. You can choose whether you want the caption at the top, at the bottom, or wherever the user's browser decides to put it.

To change the table's border, spacing (corresponding to the HTML `CELLSPACING` tag), and padding (corresponding to the HTML `CELLPADDING` tag) enter the appropriate sizes (in pixels) in the Layout fields.

Sometimes the first row of a table contains captions for the columns. To create a table like this, click the checkbox marked "First row cells are header cells." The first row cells are wrapped in an HTML <TH> tag instead of a <TD> tag, which will cause the cells to be printed in bold in most browsers.

To create a table whose contents depends on a repetition, click the checkbox marked "Second row is wrapped in a WORepetition." The preview pane renders the repeated row in blue.

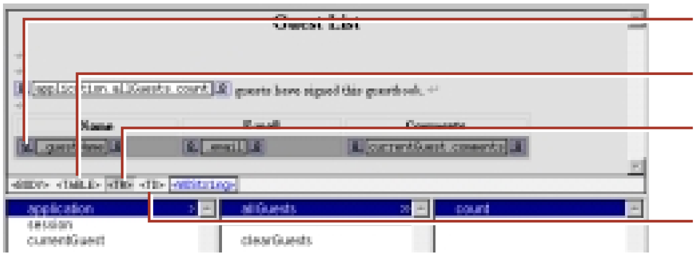
Click OK to place the table at the insertion point.

Inspecting and Making Selections

When you select the table or a part of it, the selection's HTML properties appear in the Inspector. You can use the Inspector to set these properties (for example, the width and height of a cell.)

Click in a cell to select it. To move to the next cell to the right (or the first cell of the next row if in the rightmost column), press Tab. If you reach the end of the table, pressing Tab will create a new row. Pressing Shift-Tab moves the opposite direction through the table.

An HTML table (<TABLE>) is a hierarchical structure, which contains rows (<TR>). Rows, in turn, contain cells (<TH> and<TD>). When you select a cell, the path view shows the path from the cell up to the page (<BODY>); you can inspect any element in the path by clicking its tag. For example, if you select a table cell, you can inspect the cell, the row (by clicking <TR> in the path view), or the table itself (by clicking <TABLE> in the path view).



First click within a cell.

Click here to select the entire table.

Click here to select the table row that contains the cell.

Click here to select the cell.

You can select multiple cells by

- clicking in a cell and dragging across the cells,
- selecting a cell and shift clicking in another cell, or
- command-clicking each cell on Rhapsody or control-clicking each cell on Windows NT.

The first two selection methods ensure that the selected cells form a contiguous region.

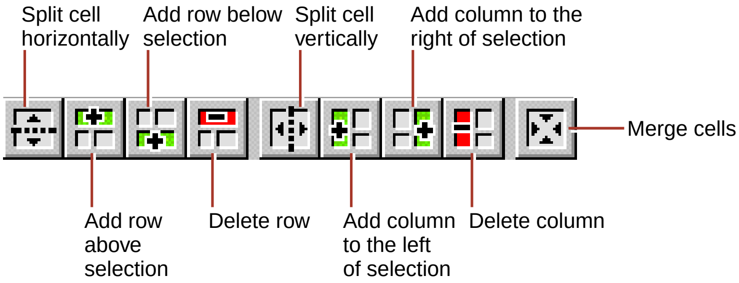
Note that selecting all of the cells in a row or table is not the same as selecting the row or table. To select a row, you need to select one or more cells in the row and click the <TR> element in the path view. To select the entire table, you need to select one or more cells in the table and click the <TABLE> element in the path view.

Modifying a Cell’s Contents

To change the contents of a cell, selecting it and type in it. You can also add an element by clicking its toolbar icon or selecting it using the menus.

Changing the Table Structure

To modify the table structure, you use buttons available in the table inspector, table row inspector, and table cell inspector.



To add a row, select a cell or row. Click in the inspector to add a row above your selection. Click to add a row below your selection.

To add a column, select a cell. Click in the inspector to add a column to the left of your selection. Click to add a column to the right of your selection.

To delete a row, select the row you want to delete (or a cell within that row) and click in the inspector.

To delete a column, select a cell within the column you want to delete and click in the inspector.

To split a cell, select it and click in the inspector to split it vertically. Click to split it horizontally.

To merge multiple cells, select a contiguous region of cells and click ▣ in the inspector. **Note:** This command isn't enabled unless the selected cells make up a group that could logically be merged into one cell.

To wrap an abstract dynamic element (conditional or repetition) around a selected row or cell, (see Repetitions) click the dynamic element's icon in the toolbar.

Sizing Tables

By default, the size of a table is determined by the contents of the table's cells. If you type text (or insert other elements) inside a table cell, the cell's width expands as necessary to fit the data. The width of any column, therefore, will be that of the widest cell in the column. **Note:** In WebObjects Builder, a cell does not resize until you have finished editing the cell and tabbed to another cell or moved out of the table. To update the cell immediately, press the Escape key.

If you want to set the size of a table or cell explicitly, use the Inspector:

- To set the width or height of a table, select the table and use the Table Inspector. You can enter values that correspond to HTML attributes controlling the size of the table.

- To set the width or height of a cell, select the cell and use the Table Data Inspector. Changing a cell's size affects the size of the column or row containing the cell.

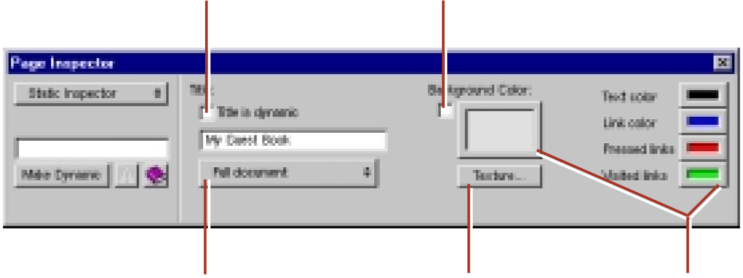
Warning: WebObjects Builder allows you to set sizing specifications that may be impossible, for example, percentages that add to more than 100% or cells in the same column with different widths. The way such tables appear depends on the browser.

Setting Page Attributes

The top level in the element hierarchy is always the page itself. To inspect a page's attributes, click the <BODY> tag in the path view. To inspect a page fragment (partial document), click the <DOCUMENT> tag in the path view. If necessary, click the Inspector button in the toolbar to display the Inspector.

Click here to make the title a dynamic string.

Enter the page title (or the binding if the title is dynamic)



Select Partial document if the component is designed for reuse within other components.

Click to choose a background image.

Click borders to open the Colors panel and set the color.

The Title text field allows you to set the title of the document. If you click the "Title is dynamic" checkbox, the title becomes a dynamic string whose value is determined at run time. You enter its binding in the Title field. See Dynamic Strings for more information.

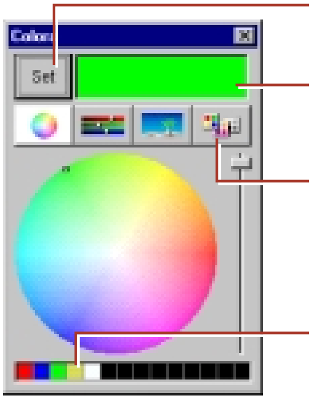
You can set the colors to be displayed for the page's background, text, or links by clicking in the border of the appropriate color well (or by clicking Colors). (See Setting Colors for more information on using the Colors panel.) To select an image to use as the page's background, click Textures.

Setting Colors

WebObjects Builder allows you to set the colors for a page's background, text, and hyperlinks.

To set the color of the selected text in the component window, click in the border of the color well ■ in the toolbar. To set other colors, use the Page Attributes Inspector.

Clicking the border of any color well brings up the Colors panel.



Click here to set the color of the selected text.

You can drag colors between this panel and any color well.

These buttons provide different ways to select colors.

Drag frequently used colors to these squares to save. Drag from squares to a color well to apply.

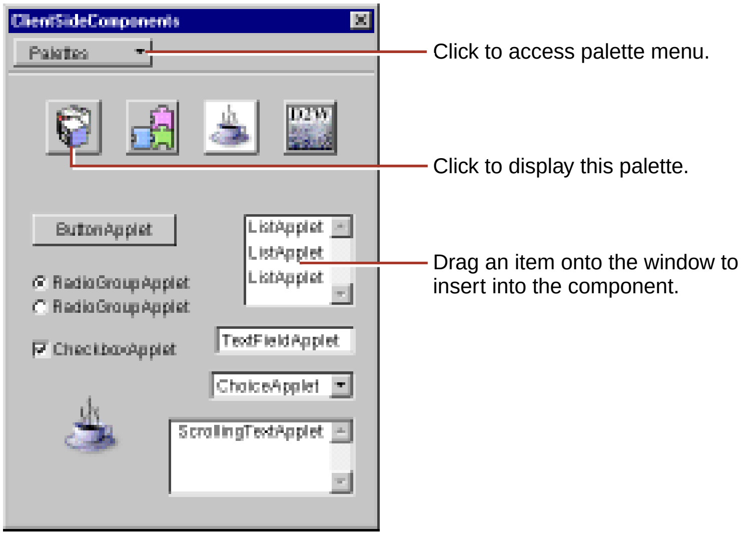
The Colors panel provides several methods of selecting colors. When you select a color, it appears in the currently selected color well.

You can drag colors from one color well to another, to the window at the top of the Colors panel, or to one of the squares at the bottom of the Colors panel to save it.

Palettes

A palette is a collection of resources (such as images, static or dynamic HTML elements, and components). You can drag elements from a palette to a component to use them. You can also drag elements from a component to a palette to store them.

Palettes appear in WebObjects Builder's palette window. To open the palette window, click 🎨 on the toolbar or choose Window ► Palette.



Click to access palette menu.

Click to display this palette.

Drag an item onto the window to insert into the component.

The Palettes pull-down list at the top of the palette window contains the palette commands. The icons below it show the available palettes. To select a palette, click its icon. Five pre-configured palettes are provided: components from the WOExtensions framework, pre-made components, Java client-side components, Direct to Web components, and JavaScript components.

You can create your own palettes to store frequently-used items, such as custom forms, tables, or images, and you can load palettes created by someone else.

To create a new palette, choose New Palette from the Palettes pull-down list. A panel appears, asking you to specify a location to save the palette. (A palette is represented on disk as a folder with the extension **.wbpalette**.) The palette appears in the palette window with the default palette icon 🎨. To change the palette's icon, see Changing a Palette Icon.

To add an existing palette to the palette window:

1. Choose Open Palette from the Palettes pull-down list.

2. Navigate to the palette’s location and click Open.

To remove a palette from the palette window:

1. Select the palette in the palette window.

2. Choose Close Palette from the Palettes pull-down list.

Note: You can’t close a pre-configured palette.

Creating and Using Palette Items

To add an item from a component to a palette:

1. Make the palette editable.

   If the palette’s background is gray, you can’t make any changes to it. To enable editing, choose Make Editable from the Palettes pull-down list. The palette's background changes to white and its title is appended with “Alt-drag to insert item.”

Note: You can’t make pre-configured palettes editable.

2. In the component window, select the element or elements that you want to add to the palette.

3. Hold down the Alt key and drag the selection to the palette.

   The cursor changes to html and displays in the palette when you are done dragging. You can change the title of the item by selecting its name and typing. To change the item's icon, see Changing a Palette Icon.

You can also add any item from the file system to a palette (including such things as a component, an image, or an EOModel). To do so:

1. Make the palette editable.

2. Locate the item in the file system.

3. Drag the item onto the palette.

   For example, to add a component to a palette, you would drag its **.wo** folder to the palette.

When you are done adding elements to your palette, choose Save Palette or Save Palette As from the Palettes pull-down list.

To copy an item from a palette to the component window:

1. Make sure the palette is not editable (if its background is white, choose Make Uneditable from the Palettes pull-down list).

   Note: If the palette is editable, you can drag the item to the window, but it will disappear from the palette.

2. Drag the item from the palette to the location in the component window where you want it to appear.

Changing a Palette Icon

You can replace the icon of any palette, or any item in a palette, with an image of your own choosing. To do so:

1. Open the palette window and select the palette whose icon you want to change.

2. Make the palette editable.

3. Drag an image from the file system onto the palette's icon.

   You can use any image file recognized by WebObjects Builder (such as a **.gif**, **.tif** or **.jpg** file) to change the icon of a palette or of any item in the palette.

4. Save the palette.

Chapter 3

Working With Dynamic Elements

Introduction to Dynamic Elements

A *dynamic element* is an element whose exact HTML representation isn't determined until run time. Dynamic elements are represented in the HTML template by the tag <WEBOBJECT>.

There are several types of dynamic elements that you can use in your WebObjects applications. Some of them (such as dynamic forms or images) have counterparts in standard HTML (<FORM> and <IMG>) and are always translated into those counterparts at run time. These are known as *concrete* dynamic elements. Others (such as conditionals and repetitions) don't translate directly into HTML but control the generation of other elements. These are known as *abstract* dynamic elements.

This chapter describes the techniques you use to add dynamic elements to your components and to bind them to variables and methods in your code. For more information on programming with dynamic elements, see "Dynamic Elements" in the *WebObjects Developer's Guide*. For details about specific dynamic elements, see the *Dynamic Elements Reference*.

Attributes

Every dynamic element has one or more *attributes*. These attributes are used for several purposes:

- Some attributes are used to determine the exact HTML to be generated when the element is displayed.

  For example, the **value** attribute of a dynamic string element (WOString) determines what text is generated in its place. At run time, WebObjects replaces the WOString with the value of the variable or method that is bound to it.

- Other attributes are used to capture information provided by users. In particular, form elements have attributes used for this purpose.

  For example, when the user submits a form, text typed by the user into a dynamic text area (WOText) inside the form is assigned to the variable bound to the **value** attribute of the text area.

- Other attributes are used to specify actions to be taken when an event occurs.

  For example, a dynamic hyperlink (WOHyperlink) has an **action** attribute that specifies an *action method* in the application that is executed when the user clicks the link.

The process of associating an attribute with a variable or method in your code is called *binding*. WebObjects Builder provides tools to make it easy for you to create bindings. Information about your bindings is stored in the declarations (**.wod**) file in your component.

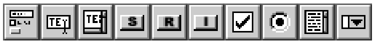
Most dynamic elements have a number of attributes that you can bind. Some are required and others are optional. For complete information about WebObjects dynamic elements and their attributes, see *Dynamic Elements Reference*.

Creating Dynamic Elements

There are several ways to add dynamic elements to your component.

Using the Toolbar

You create dynamic elements in the same way that you create other elements: by clicking buttons in the toolbar or using the menu commands. In WebObjects Builder, there are two toolbars that allow you to create dynamic elements:

- The Forms toolbar  allows you to create dynamic form elements. See Creating Form-Based Dynamic Elements for more detailed information about working with forms.
- The Other WebObjects toolbar allows you to create other types of dynamic elements. See Creating Other WebObjects for more details on each type of element.

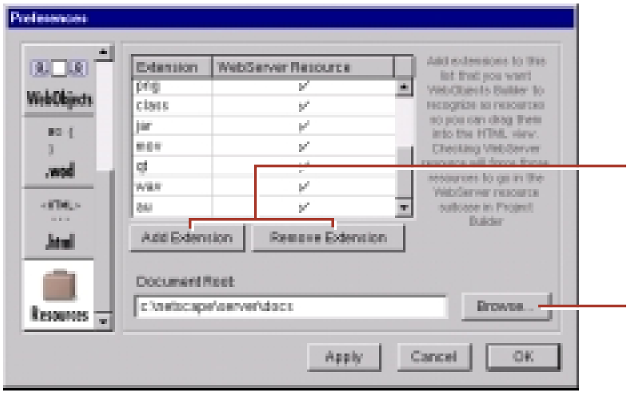
Dragging Elements into the Component Window

Some elements can be created by dragging an item from the file system into a component window. These include:

- Components (see Reusable Components)
- Client-side Java components (see WOApplets)
- Image files and image maps (see Dynamic Images)

In addition, you can also drag a model file (of type **.eomodeld**) into a component to create a variable of type WODisplayGroup (see Adding Display Groups).

Certain file types (such as **.gif**, **.jpeg**, **.tif**, **.eps**, and **.bmp**) are automatically recognized by WebObjects Builder. The Preferences Panel (which you display by choosing Tools ► Options) shows a list of file extensions that WebObjects Builder accepts. You can drag any item with one of those file extensions into a component window, and the item will be added to your project. You can add file types if you need them.



Preferences

Extension	WebServer Resource
png	✓
class	✓
jar	✓
mov	✓
qt	✓
wav	✓
au	✓

Add extensions to this list that you want WebObjects Builder to recognize as resources so you can drag them into the HTML view. Checking WebServer resource will force those resources to go in the WebServer resource suitcase in Project Builder

Document Root: c:\netscape\server\docs

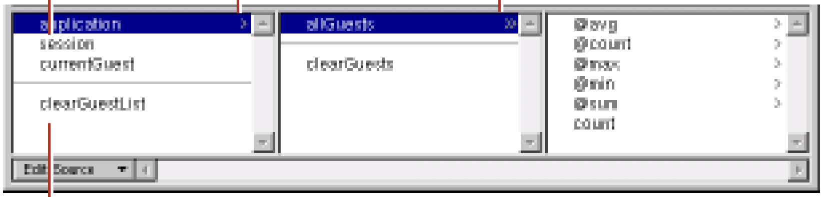
Click here to add or remove extensions that WebObjects Builder recognizes.

If you've changed your document root, click here to inform WebObjects Builder.

Note: Not all web browsers support **.tif**, **.eps** and **.bmp** images, so you probably want to convert them to **.gif** or **.jpeg** images.

The Object Browser

The bottom part of the component window is the *object browser*, which displays your application's variables and methods. This display provides a graphical method of binding objects in your code to dynamic elements in the component.



Keys (variables and methods) appear above the line.

Select key to display its keys and actions in next column (indicated by “>”.

“>>” indicates an array. Select it to show its keys (for example, count) in next column.

Actions appear below the line.

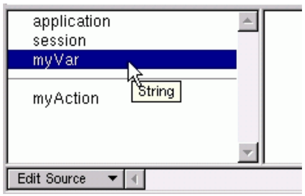
The first column of the object browser displays two types of objects:

- *Keys* are displayed above the horizontal line. A key can be either an instance variable or a method that returns a value.
- *Actions* are displayed below the line. An action (or action method) is a method that takes no parameters and returns a component (the next page to be displayed).

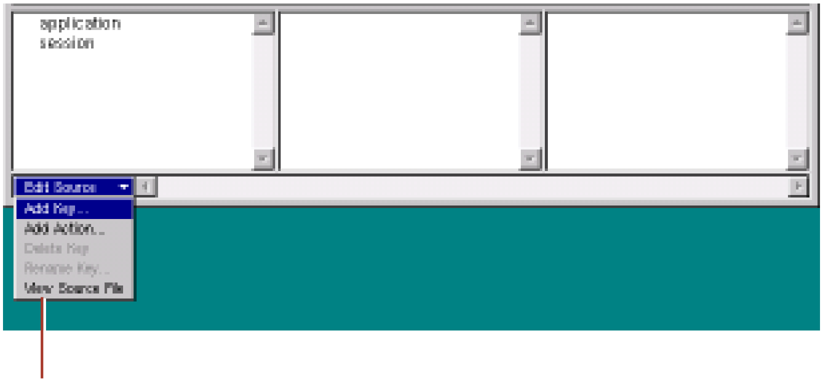
A “>” next to an object’s name in the browser indicates that it contains additional keys and actions, which are displayed in the next column when you select it.

In the figure, for example, the **application** object is selected, showing that there are keys and actions defined in the application code. One of these, **allGuests**, is an array (indicated by the “>>”), and the array’s **count** method is displayed in the next column.

Note: If you rest the mouse pointer on a key, WebObjects Builder displays its type.



When you create a new project, the only keys that appear in the object browser are **application** and **session** (unless you use the Wizard to create a database application). These are methods that allow you to access variables in your application and session code.



Choose a command from this menu to add objects to your source code or view the code.

There are several ways to add items to the object browser:

- Use Project Builder to add keys and actions to your component's source file.

  When you save changes to a source file, WebObjects Builder parses the file, detects items that have been added and deleted, and updates the object browser's display to reflect the changes. The source code can be written in any of the languages that WebObjects supports (Java, Objective-C, or WebScript).

- Use the menu at the bottom of the object browser to add items to your code directly from WebObjects Builder. See the next section, Working with Keys in WebObjects Builder, for more information.

- Drag a *model file*, an entity from Enterprise Objects Modeler, or a relationship from Enterprise Objects Modeler into the browser to create a *display group* variable. See Adding Display Groups for more information.

Working with Keys in WebObjects Builder

At the bottom of the object browser, there is a pull-down list called Edit Source File. It has five items:

- **Add Key** allows you to add a key (an instance variable or a method that returns a value) to your source file.

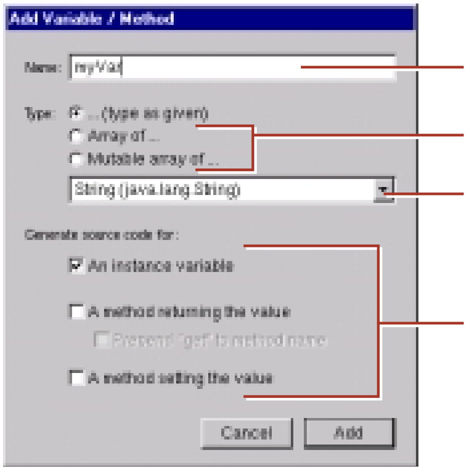
- **Add Action** allows you to add the template for an action (a method that takes no parameters and returns a component).

- **Delete Key** deletes a key from your source file by deleting its instance variable and accessor methods.

- **Rename Key** renames a key in your source file by renaming its instance variable and accessor methods.

- **View Source File** opens the source file in a Project Builder window.

When you choose Add Key, the following panel opens:



Name: myVar

Type: ● ...(type as given)
○ Array of ...
○ Mutable array of ...

String (java.lang.String)

Generate source code for :

☑ An instance variable

☐ A method returning the value

☐ Prepend "get" to method name

☐ A method setting the value

Type the key name here.

Click one of these buttons if your variable is an array.

Choose the key's type from this pop-up list.

Check one or more of these boxes.

In this panel, you specify:

- The name of the key.
- Its type.

  You can choose the type from the combo box or type it in directly. You can also use the radio buttons to specify whether the variable is an array.
- How the key is implemented.

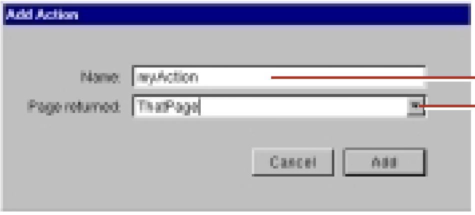
  The key can be an instance variable whose value is accessed directly, or a method that returns a value (not necessarily associated with an instance variable). You can also create a method that sets the value of an instance variable.

When you click Add, the key's name appears in the object browser (below **application** and **session**). To see what was added to your source code, choose View Source File from the Edit Source pull-down list in the object browser. You'll see something like the following:

```
protected String myVar;
public String myVar() {
    return myVar;
}
public void setMyVar(String newMyVar) {
    myVar = newMyVar;
}
```

The first line defines the instance variable. The first method returns its value. The second method sets its value.

When you choose Add Action, the following panel appears:



Name: myAction
Page returned: ThatPage

Enter the action method's name here.

Select response page name from combo box.

When you click Add, the following code is added to your source file:

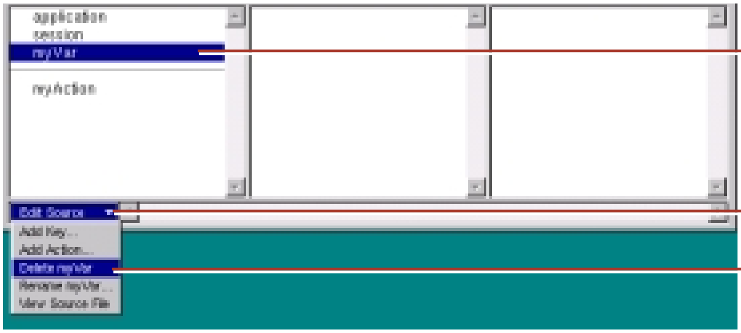
```
public ThatPage myAction()
{
    ThatPage nextPage = (ThatPage)pageWithName(“ThatPage”);

    // Initialize your component here

    return nextPage;
}
```

WebObjects Builder provides these ways to add variables and methods for your convenience. Of course, you can add variables and methods directly to your component’s code by editing them in Project Builder.

To delete a key, select it in the object browser and choose **Delete *Key*** from the Edit Source pull-down list. Any variables and methods associated with the key are deleted from the source file. You can restore the deleted key by selecting Undo from the Edit menu.



Click the key to select it.

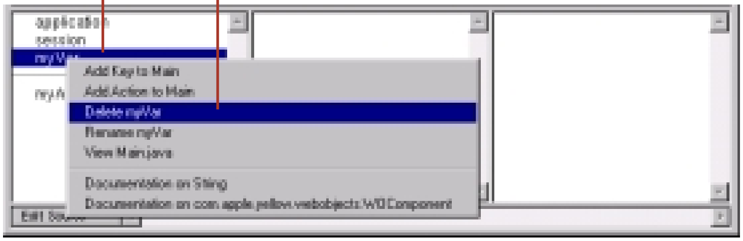
Click here to access the pull-down list.

Choose Delete key.

You can also delete the key with its context menu.

1. Control-click the key on Rhapsody or right-click the key on Windows NT

2. Choose Delete *key* from the menu that appears.



Control-click (Mac OS X Server) or right-click (Windows NT) the key to reveal the key's context menu.

Choose Delete *key*.

To rename a variable or method, select its key in the object browser and choose **Rename *Key*** from the Edit Source pull-down list. In the panel that appears, enter the new name for the key and click Rename. Any variables and methods associated with the key are renamed.

Alternatively, you can rename the key with its context menu.

1. Control-click the key on Rhapsody or right-click the key on Windows NT.

2. Choose Rename *key* from the context menu that appears.

3. Enter the new name in the panel that appears.

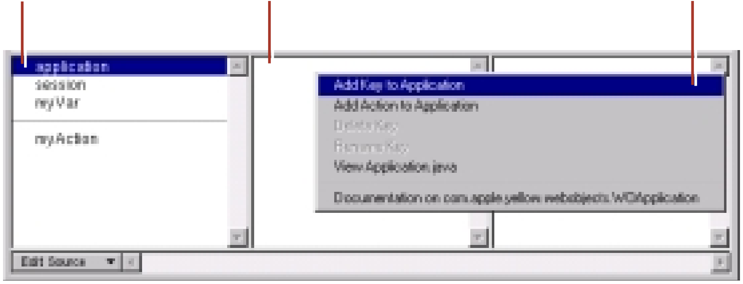
Working with Application and Session Variables

The Edit Source pull-down list applies to the keys and actions in the component's source file only. It does not manipulate the session or application source files. To add keys or actions to the application source file:

1. Select **application** from the object browser

2. Control-click (Rhapsody) or right-click (Windows NT) in the second column of the object browser.

3. Choose Add Key to Application or Add Action to Application from the menu that appears.



Click **application** to select the application source file.

Control-click (Mac OS X Server) or right-click (Windows NT) in this column to reveal the context menu.

Click Add Key to Application.

Similarly, to add keys or actions to the session source file, select **session** from the object browser and perform steps 2 and 3 above.

You can delete or rename an application or session key with the context menus. Control-click the key in the object browser (Rhapsody) or right-click the key (Windows NT) to access the context menu. Choose Delete *key* or Rename *key*.

Adding Display Groups

A *display group* is an important type of variable that you use in WebObjects applications that access databases. A display group is an object that can fetch, insert, delete, display, update and search records in a database.

This section describes the mechanics of adding display groups to a WebObjects project. For detailed information about display groups, see the WODisplayGroup class specification in the *WebObjects Class Reference*. To learn more about how to create a WebObjects database application, see “Creating a WebObjects Database Application” in *Getting Started With WebObjects*.

WebObjects applications access databases through the Enterprise Objects Framework, which represents database rows as *enterprise objects*. Enterprise object classes typically correspond to database tables, and an enterprise object instance corresponds to a single row or record in a table. For detailed information on enterprise objects, read the *Enterprise Objects Framework Tools and Techniques*.

In a database application, you use *entity-relationship models*. A model associates database columns with instance variables of objects. You create a model with the

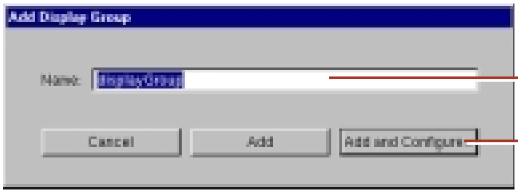
EOModeler application, or you can specify one when you use the Wizard to set up your application (when you add a model to your project, it is added to the Resources suitcase). A model is stored in a *model file*. For more information on creating models, see the chapter “Using EOModeler” in *Enterprise Objects Framework Developer’s Guide*.

A model contains *entities*, *attributes*, and *relationships*. An *entity* associates a database table with an enterprise object class. Display groups manage objects associated with a single entity. An *attribute* associates a database column with an instance variable. A *relationship* is a link between two entities that’s based on attributes of the entities.

If you used the Wizard to set up your application, a display group was set up for you based on the model you specified. There are several other ways to create a display group:

- Drag a model (a folder with the extension **.eomodeld**) from the file system into the object browser in your component window, or drag an entity from the EOModeler application into the object browser.

  When you do this, a panel asks you if you want to add the model to your project. If you reply Yes, the Add Display Group panel appears.



Enter the display group’s name.

Click here to add the display group and configure it.

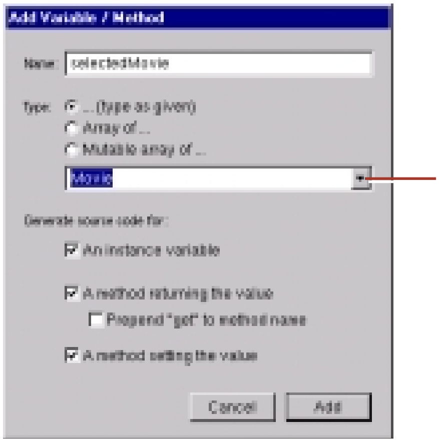
  It allows you to specify a name for your display group and decide if you want to simply add the display group, or configure it as well. Configuring the Display Group describes the configuration process.

- Use Add Key to define a variable of type WODisplayGroup, or declare the display group directly in your code:

```
protected WODisplayGroup myDisplayGroup; //this is a Java example
```

  When you add a display group this way, you are responsible for making sure your project contains the appropriate model file. (For example, once a model file has been added, you can create any number of display groups based on it). In addition, you need to configure the display group.

When you use the Add Key panel, you can create not only display group variables, but also enterprise objects associated with any of the entities in your project's models.



Add Variable / Method

Name: selectedMovie

Type: ● ... (type as given)
○ Array of ...
○ Mutable array of ...

Movie

Generate source code for:

☑ An instance variable

☑ A method returning the value

□ Prepend "get" to method name

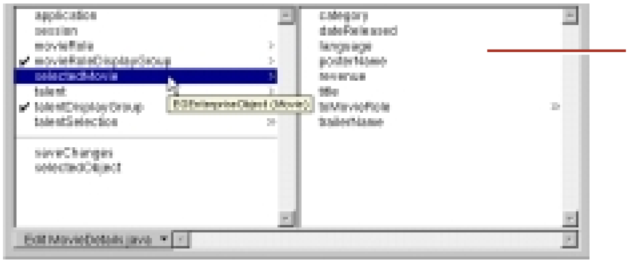
☑ A method setting the value

Choose an entity from the project's models to create an enterprise object based on that entity.

In the figure, if you choose the entity Movie as the variable's type, the following code gets added to your source file:

```
/** @TypeInfo Movie */
protected EOEnterpriseObject selectedMovie;
```

The variable **selectedMovie** is declared as type EOEnterpriseObject. The comment `/** @TypeInfo Movie */` is a *structured comment* that WebObjects Builder uses to identify the entity associated with the object (don't edit it). It is then able to display the attributes in the object browser as shown here:



The browser displays the attributes of the entity.

Configuring the Display Group

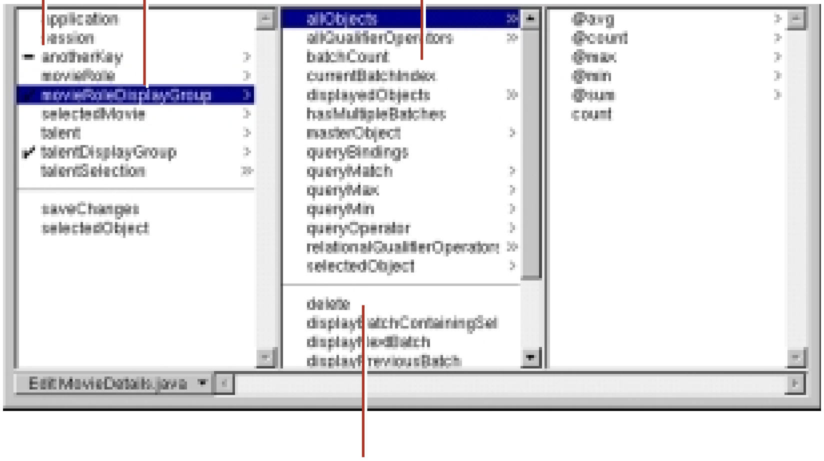
A display group must be configured in order for it to be created and initialized automatically when the component is initialized at run time. Display groups are instantiated from an archive file (with the extension **.woo**) that's stored in the component. You shouldn't edit **.woo** files by hand; they're maintained by WebObjects Builder.

In the object browser, ✓ means that the display group has been configured. A ⊟ means that it has not been configured, and so the variable isn't automatically created at run time. A configured display group shows its keys and actions in the second column of the object browser. You can bind them to elements in your program.

Unconfigured display group (double-click to configure).

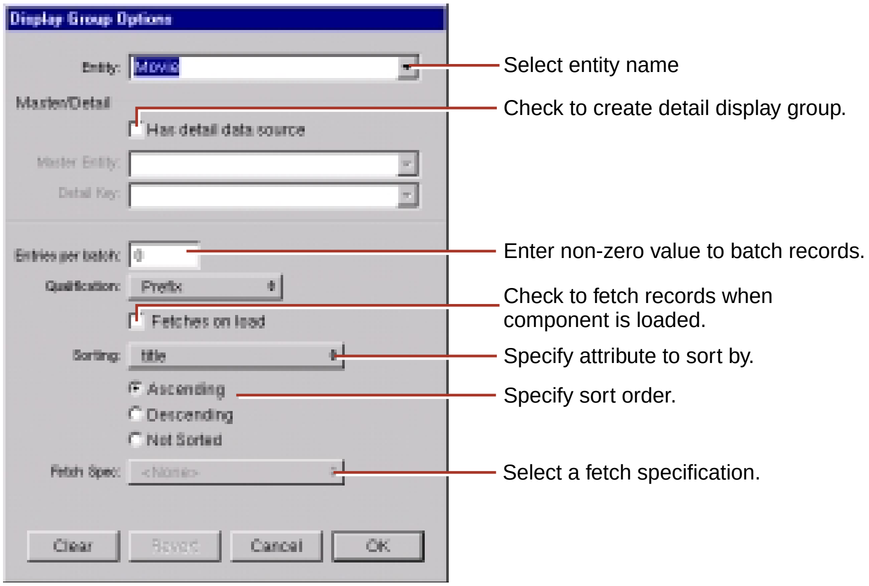
Configured display group

Display group's keys



Display group's actions.

To configure a display group (or change its configuration), double-click its name to open the Display Group Options panel.



Display Group Options

Entity: Movie

Master/Detail

□ Has detail data source

Master Entity:

Detail Key:

Entries per batch: 0

Qualification: Prefix

□ Fetches on load

Sorting: title

● Ascending

○ Descending

○ Not Sorted

Fetch Spec: <None>

Select entity name

Check to create detail display group.

Enter non-zero value to batch records.

Check to fetch records when component is loaded.

Specify attribute to sort by.

Specify sort order.

Select a fetch specification.

In this panel, you specify the following information:

- *Entity:* The Entity combo box contains entities from the models in your project. You can select one from the list or type the name.
- *Has detail data source:* Check this to create a detail display group. See Creating a Detail Display Group for more information.
- *Entries per batch:* Set a non-zero value here to specify the number of records to be displayed at once. When the value is zero, all records are displayed.
- *Qualification:* When displaying records according to a query, this setting determines whether to display records that begin with, end with, or contain the item specified.
- *Fetches on load*: When you check this option, the display group fetches all its objects as soon as the component is loaded into the application.
- *Sorting:* You select an attribute by which to sort your displayed objects from the pop-up list, and use the radio buttons to select the order of sorting.

- *Fetch Spec*: The Fetch Spec pop-up list contains all the fetch specification defined in the corresponding model file. A fetch spec is simply a predefined query.

Creating a Detail Display Group

While a display group manages objects associated with a single entity, you can access other kinds of objects through an entity's relationships. In a *master-detail* configuration, a master display group holds enterprise objects for the source of a relationship, while a detail display group holds records for the destination. As individual records are selected in the master display group, the detail display group gets a new set of enterprise objects to correspond to the selection in the master.

To create a detail display group, you can use the Display Group Options panel:

1. Check "Has detail data source."

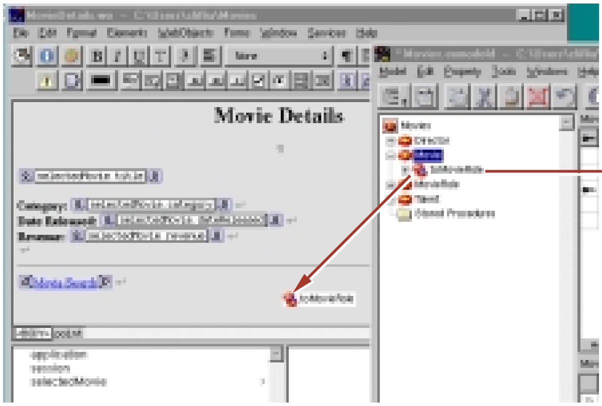
   The Master Entity pop-up list is enabled. It lists all entities in the models in your project.

2. Select the Master Entity from the pop-up list.

   The Detail Key pop-up list now contains the keys representing the master entity's relationships.

3. Select the Detail Key from the pop-up list.

You can also create a detail display group by dragging a to-many relationship from EOModeler into your component.



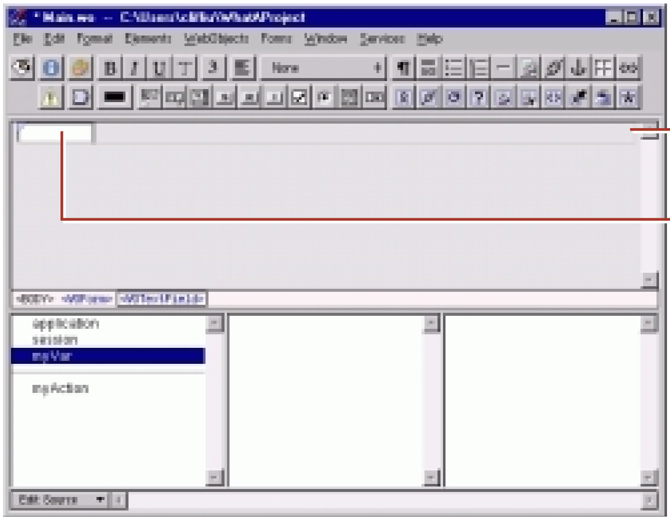
Drag relationship from EOModeler into component to create detail display group.

This example creates a detail display group based on the **toMovieRole** relationship, with **Movie** as the master entity and **MovieRole** as the detail entity.

As with other display groups, you can use the Display Group Options panel to immediately configure the newly created display group.

Binding Elements

This section discusses the basic techniques you use to bind elements. Further detail is presented in the sections that discuss specific dynamic elements.


File Edit Format Elements WebObjects Forms Window Services Help
None
<BODY> <WOForm> <WOTextField>
application
session
myVar
myAction
Edit Source

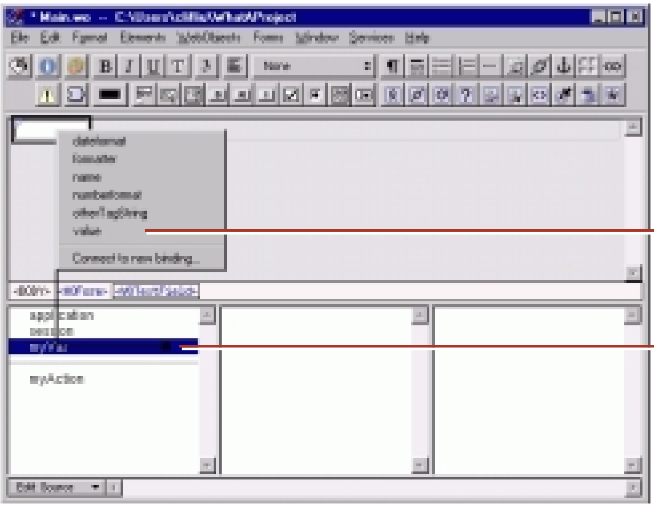
Form containing text field.

Dynamic text field.

In the figure, you have added a form (WOForm) containing a dynamic text field (WOTextField) to your component. Note the triangle in the top left corner, which distinguishes the dynamic text field from a static HTML text field. The long rectangle surrounding the text field represents the containing form.

To bind the text field to the variable **myVar**:

1. Press mouse down on **myVar** in the object browser and drag to inside the text field.



Click an attribute to complete the binding.

Bind by dragging from the key to the element.

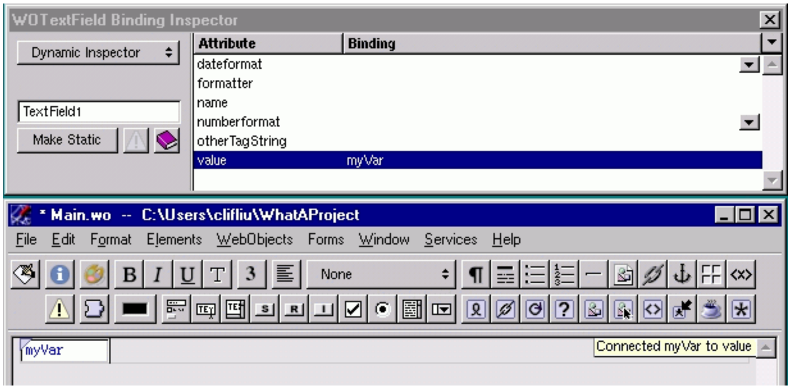
A black line appears as you drag, and a black border appears around the text field, indicating that you can bind to it.

2. Release the mouse button.

A menu containing the attributes for that element appears.

3. To complete the binding, click the attribute you want to bind (value in this case.)

In the inspector, which you can access by clicking ℹ in the toolbar, the name of the variable appears in the Binding column next to the attribute. Note that it also appears in blue inside the text field in the component window. Some (not all) dynamic elements display the binding for their default attribute inside the element itself.


WOTextField Binding Inspector
Dynamic Inspector
Make Static
* Main.wo -- C:\Users\clifliu\WhatAProject
File Edit Format Elements WebObjects Forms Window Services Help
None

TextField1

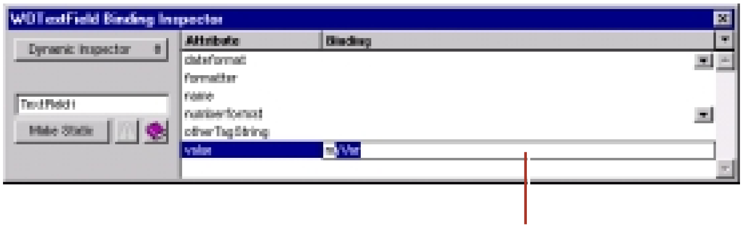
Attribute	Binding
dateformat	
formatter	
name	
numberformat	
otherTagString	
value	myVar

myVar

Connected myVar to value

You can also bind an element's attributes by typing in the Inspector directly. To do this:

1. Double-click in the binding column of the row for the attribute you want to set.



TextField1

Attribute	Binding
dateformat	
formatter	
name	
numberformat	
otherTagString	
value	myVar

Double-click here to get
a cursor, then type the binding.

   A cursor appears in the Binding column, allowing you to type.

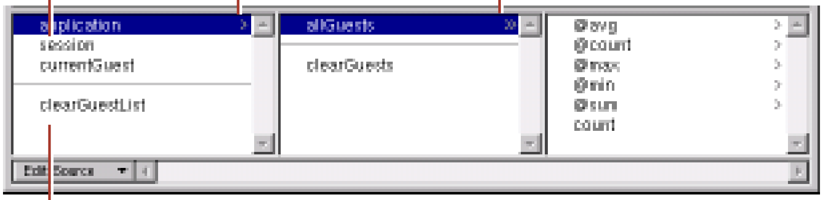
2. Type the binding in the text field. As you type, WebObjects Builder attempts to complete the binding using keys in the object browser.

3. When the desired key appears in the binding column, press Enter.

When entering bindings this way, the following rules apply:

- Constant strings (such as `"Joe"`) must be in double quotes.
- Variable and method names (such as **joe**) must not be in quotes.
- Symbolic constants (such as `YES` and `NO`) must not be in quotes.

- Keys must specify their full *key path*. For example, to bind the key that is selected in the following figure, you would type “a.a.c” to select `application.allGuests.count`.



Keys (variables and methods) appear above the line.

Select key to display its keys and actions in next column (indicated by “>”.

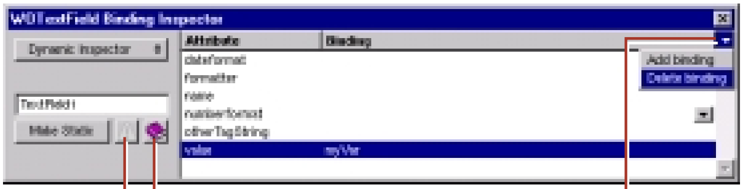
“>>” indicates an array. Select it to show its keys (for example, count) in next column.

Actions appear below the line.

Deleting Bindings

To delete a binding:

1. Open the Inspector by clicking ℹ.
2. Click the row containing the binding you wish to delete.
3. Click the pull-down list in the top-right corner of the inspector and choose Delete binding. You can also press the Delete key (Backspace on Windows NT) to delete a binding.



Click to see documentation on this dynamic element.

Click to find binding errors.

Click to delete or add attribute to element.

In the WOTextField example above, if you delete the **value** binding, the Inspector displays **value** in red. Each element has a binding syntax that must be followed; otherwise, WebObjects won't launch your application. The Inspector displays required bindings that have not been set and illegal bindings in red.

Other Binding Commands

The Inspector also allows you to:

- View documentation on an element by clicking 📕.

  The documentation is displayed in your web browser.

- Find binding errors by clicking ⚠ if the element has any binding errors.

  A window detailing the binding errors (that appear in red in the inspector) appears. This window shows all the binding errors in the component, highlighting the errors for the dynamic element you are inspecting.

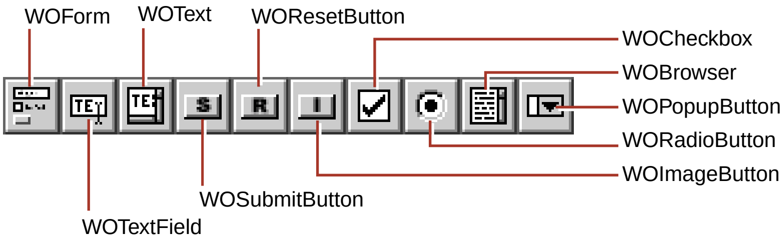
- Add a new attribute for this element by clicking the pull-down list in the top-right corner of the inspector and choosing Add binding.

  Typically, you don't add attributes to dynamic elements. For standard dynamic elements such as WOTextField or WOString, the attributes always appear in the inspector. For your own custom WebObjects, use the API editor (see Custom WebObjects).

Creating Form-Based Dynamic Elements

In HTML, a form is a container element (one that can contain other elements). Typically, forms contain input elements (such as text fields, radio buttons and checkboxes) to capture user information, a button or active image to submit the form data, as well as display elements such as text and images.

In WebObjects Builder, you create form elements by clicking one of the buttons in the Form Elements portion of the toolbar (or using their menu equivalents).



All the form elements you create in the toolbar are dynamic equivalents of standard HTML elements. You can convert any dynamic form element to its static equivalent (and vice versa) by using the Inspector (see Dynamic and Static Inspectors).

Most form elements have a **value** attribute that represents the information entered by the user. You bind this attribute to a key so that your application can work with it. Others, such as WOSubmitButton, WOImageButton, or WOForm itself, don't receive information but have an **action** attribute representing an action to be taken when the form is submitted. You bind form-based elements by the process described in Binding Elements.

Usually you create a WOForm element to contain other form elements, including buttons. The submit and reset buttons will apply to all other elements inside the same form.

Dynamic and Static Inspectors

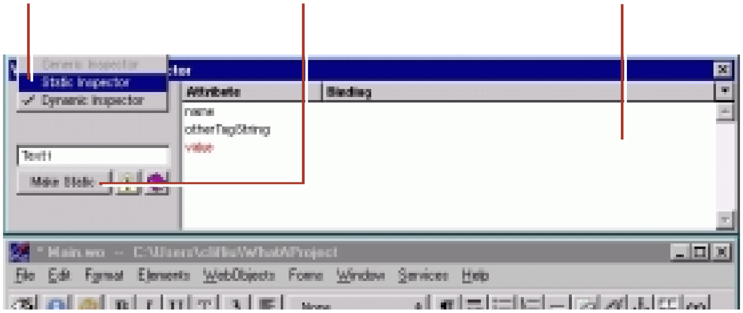
Most dynamic elements have static HTML counterparts (with the exception of abstract dynamic elements, such as: WOString, WORepetition, WOConditional, and WOCustom.) The Inspector for these elements has two states:

- The Dynamic Inspector, which you use to set the bindings for the element (see Binding Elements).
- The Static Inspector, which you use to set the HTML attributes for the element's static counterpart.

Select Static Inspector to edit HTML attributes.

Click here to convert WOText to a static text area.

Set bindings for the WOText element here.

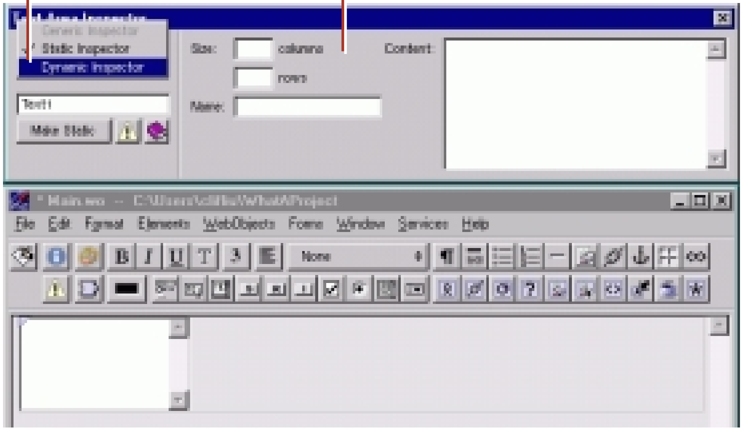


This example shows the Inspector for a dynamic text area element. It displays the bindable attributes for this element. If you select Static Inspector from the pop-up list, the Text Area Inspector appears. This is the same Inspector you would see for a static text area element (<TEXTAREA>) and allows you to set its HTML attributes (such as `COLS` or `ROWS`).

Note: You can also set the HTML attributes using the Dynamic Inspector. The Static Inspector is provided for convenience only.

Select Dynamic Inspector to set the WOText's bindings.

Set HTML attributes of text area here.



To switch back to the WOText Inspector, select Dynamic Inspector from the pop-up list.

In addition, you can convert any dynamic element into its static counterpart, or vice versa:

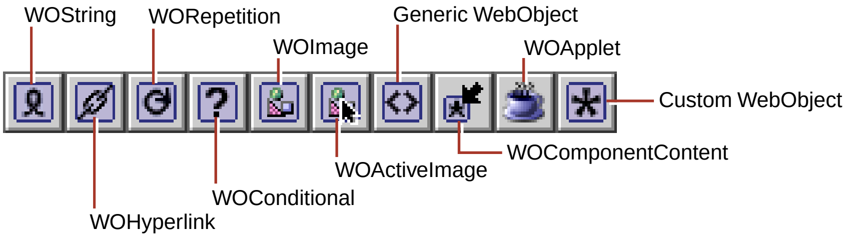
- When inspecting a dynamic element, if you click Make Static, the element becomes its static counterpart (if it has one), and the Static Inspector appears.
- When inspecting a static element, if you click Make Dynamic, the element becomes its dynamic counterpart. Both the Static and Dynamic Inspectors are now available.

The following table shows the dynamic counterpart for each static element.

Static Element	Dynamic Counterpart
Image	WOImage, WOActiveImage
Form	WOForm
Textfield	WOTextField
Text Area	WOText
Button	WOSubmitButton, WOResetButton, WOImageButton
Checkbox	WOCheckBox
Radio Button	WORadioButton
Select	WOBrowser, WOPopupButton
Hyperlink	WOHyperlink
Applet	WOApplet
Other	Generic WebObject

If you convert a static element to its dynamic counterpart by clicking Make Dynamic, and there is no direct counterpart, the element becomes a generic WebObject whose element name is the HTML tag for the static element (see Generic WebObjects).

Creating Other WebObjects



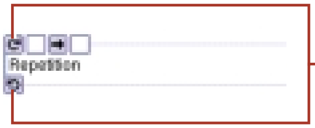
You use this toolbar to create all dynamic elements other than form-based elements. This section provides some general information about using these elements. Each element is described in more detail in its own section.

To create a dynamic element, you click its toolbar icon. One thing to be aware of is what happens when there are already elements selected when you create the element:

- Some dynamic elements (WOHyperlink, WOConditional, WORepetition, custom WebObjects and generic WebObjects) can contain other elements. In this case, the selected elements appear with the new element "wrapped" around it.

- Other dynamic elements (WOString, WOImage, WOActiveImage, and WOApplet) can't contain other elements. When you create one, it replaces whatever was selected.

The dynamic element types in the toolbar except for WOImage, WOActiveImage, and WOApplet display with a pair of icons surrounding the element (and possibly other icons in between). For example, when you create a repetition, it appears like this in the component window:



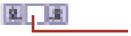
Drag from the object browser
to one of the outer icons
to bind (brings up an attribute menu)..

To bind a dynamic element, you drag from an item in the object browser to one of the outer icons. The attribute menu appears, allowing you to complete the binding. See Binding Elements for more information.

Dynamic Strings

A WOString element represents a dynamically generated string. You bind the **value** attribute of a WOString to a variable or method that returns a string at run time. A WOString is abstract in that it doesn't represent any specific element, but it can be contained in any other HTML element that can contain text.

WebObjects Builder provides a shortcut for binding the **value** attribute of commonly used elements such as WOString.



Drag to here to bind **value** attribute directly.

Instead of dragging to one of the icons, drag to the center binding box. The binding appears directly in the box, and the attribute menu doesn't appear.

Dynamic Hyperlinks

Dynamic hyperlinks (WOHyperlink) allow you to specify the link's destination at run time rather than at compile time. There are several ways to do this:

- You can specify the name of a page in your application as the destination of the link. To do this, bind the name to the WOHyperlink's **pageName** attribute. This is useful since pages in a WebObjects application don't have predictable URLs that you can specify in an HTML hyperlink.
- You can specify an action to be performed when the hyperlink is clicked by binding WOHyperlink's **action** attribute to an action method in your code. This method can perform any sort of action, as well as returning a page as the destination.
- You can also specify a URL as the destination by binding to the **href** attribute.

To create a dynamic hyperlink:

1. Click 🔗 in the toolbar.
2. Replace the word `Hyperlink` with the text of the link.
3. Create the element's bindings.

To learn how to create a static hyperlink, see Setting Page Attributes.

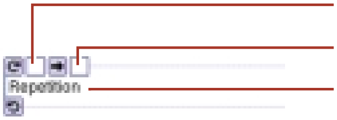
Repetitions

A repetition (WORepetition) is a container element that repeats its contents a certain number of times. It is like a loop in a structured programming language. Repetitions are one of the most important elements in WebObjects, since it is quite common for applications to display repeated data (often from databases) when the amount of data to be displayed isn't known until run time. Typically, a repetition is used to generate items in a list, multiple rows in a table, or multiple tables.

To create a repetition:

1. Click ↻.

   The repetition appears in the component window.



Drag to here to bind the **list** attribute directly.

Drag to here to bind the **item** attribute directly.

Add elements here.

2. Add elements inside the repetition (replacing the word "Repetition").

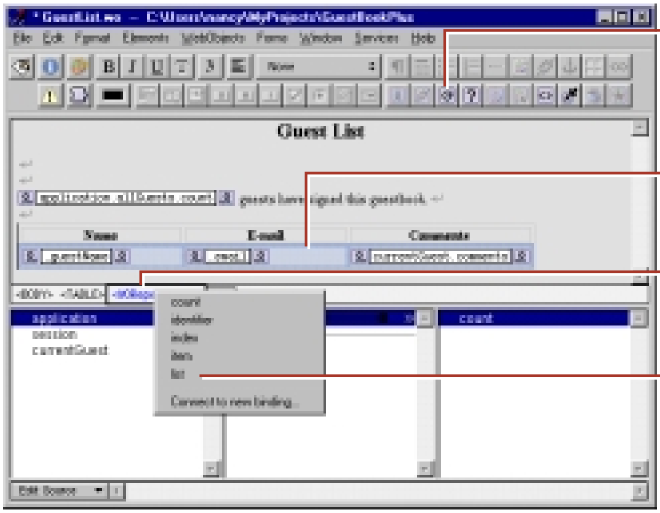
   A repetition can contain any other elements, either static HTML or dynamic WebObjects elements.

3. Alternatively, you can select existing elements, then click ↻ to wrap the repetition around the elements. This is necessary in some cases, such as wrapping a repetition around a table row.

You usually bind two attributes of a repetition: **list** and **item**. The **list** attribute must be bound to an array. WebObjects generates the elements in the repetition once for each item in the array. Each time through the array, the **item** attribute points to the current array object. Typically, you bind **item** to a variable and then use that variable in the bindings of the elements inside the repetition.

In addition, as with WOStrings, WebObjects Builder provides a shortcut for binding repetitions so that you don't have to use the Inspector. Drag to the first binding box to bind the **list** attribute; drag to the second box to bind the **item** attribute.

When you wrap a repetition around a table row, the repetition symbol doesn't appear by default (although you can turn on table tag rendering in the Editing Preferences panel). Instead, the row appears in blue. To bind the repetition, drag from the object browser to the <WORepetition> element in the path view. The attribute menu appears, allowing you to complete the binding as usual.



Select table row, then click here to create repetition around row.

Blue border and background means row is in a repetition.

Drag variable here to bind to repetition.

Click here to bind variable to the repetition's list attribute.

Note: You can also wrap a repetition around a single cell in a table. In addition, this same procedure of wrapping a repetition around a table row or cell also works for conditionals (see next section).

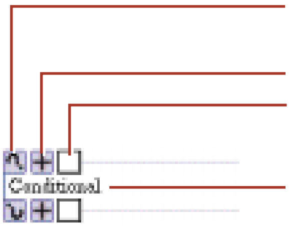
Conditionals

A conditional (WOConditional) is a dynamic container element that displays its contents only if a particular condition is true. WOConditional's main attribute is **condition**, which takes a Boolean value. If **condition** is true (1), the WOConditional's contents are displayed. If **condition** is false (0), the contents aren't displayed.

condition must be bound to a variable or a method that returns a boolean value. (WebScript and Objective-C use the constants **YES** and **NO**; Java uses **true** and **false**.) To bind **condition** (or any other attribute that takes a boolean) to a constant value, enter **YES** or **NO** in the bindings Inspector.

To create a conditional, click ? in the toolbar.

Note: Any selected elements will be contained within the conditional.



Drag from the object browser to here to bind (opens the attribute menu).

Click to set **negate** to true.

Drag from the object browser to this box to bind the condition attribute without opening the attribute menu. The binding appears inside the box.

Contents of the conditional display if condition is true (or negate is true and condition is false).

To bind to a conditional, click a variable or method and drag to one of the conditional's outer icons. The Inspector appears, displaying the bindings for the WOConditional, with the **condition** attribute selected by default. Complete the binding by clicking Connect, or choose a different attribute to bind.

There is a shortcut for binding the **condition** attribute similar to the WOString shortcut. Drag from a key in the object browser to the binding box in the conditional.

Sometimes, you want the equivalent of an "if-then-else" structure; that is, "if the condition is true, display this text; if not, display this other text." To accomplish this, you can use the **negate** attribute. If **negate** is true, then the contents of the conditional are displayed only if **condition** is false. To create an if-then-else structure, do the following:

1. Create two WOConditionals.

2. Bind the **condition** attribute of both of them to the same variable or method.

3. Click ⊞ on the second conditional to change its **negate** attribute to YES (true). The button changes to ⊟.

   By default, **negate** is false, so you do not explicitly need to bind the first conditional's **negate** attribute.

As with repetitions, you can "wrap" a conditional around a table row (see Repetitions). When you do this, the conditional symbol doesn't appear but the row appears with a blue border.

Dynamic Images

The elements WOImage and WOActiveImage are dynamic images. At run time, WOImage is rendered as a passive image and WOActiveImage as a mapped, active image. To create them, click 🖼 or 🖼 in the toolbar, respectively.

A static image element requires you to specify its pathname directly in the HTML. With dynamic images, you bind the **filename** attribute to specify the name of an image file in your project, or in a framework. You can bind this attribute to a key so that the filename is dynamically generated at run time.

You can also create a WOImage by dragging an image from the file system into your component (see Dragging Elements into the Component Window for more information). An alert appears, asking whether you want to add the image to the project (if it is not already in the project). If you do, the file is added to the Web Server Resources suitcase of your project.

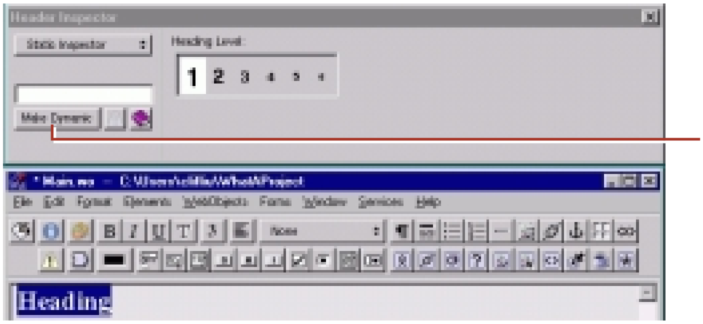
Generic WebObjects

You can use the generic WebObject element to create a dynamic version of any HTML element.

To create a dynamic version of a standard HTML element:

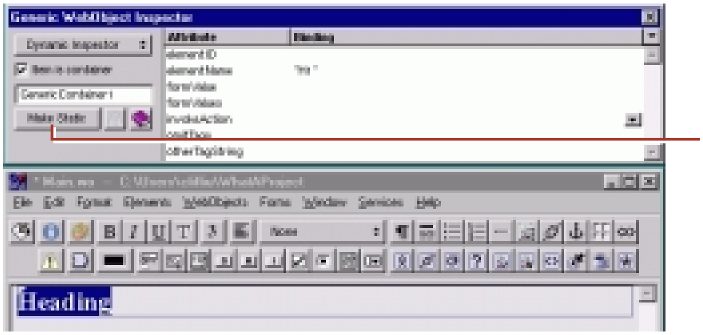
1. Create the element (say, a heading) and select it.

2. In the Inspector, click Make Dynamic.



Click here to convert the heading to a dynamic element.

If the element has no specific dynamic counterpart, it becomes a generic WebObject element.

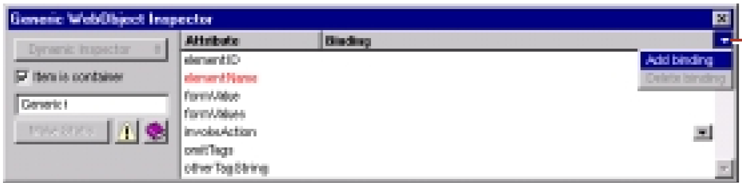


Click here to convert back to a heading.

To create a generic WebObject corresponding to any HTML element (even ones not supported directly by WebObjects Builder):

1. Click <> in the toolbar.

2. Bring up the inspector.



Click to add attributes.

A generic WebObject element has one required attribute, **elementName**, which specifies what type of element should be generated at run time.

For example, imagine that a future version of HTML adds a new container element, which you would like to generate dynamically in your component. You would:

3. Double click in the binding column next to **elementName**. Type the name of the container in quotes.

   If the name isn't in quotes, WebObjects assumes it is a binding that should be resolved at run time. You might use that technique if you wanted to choose the type of element programmatically rather than specifying it in advance.

4. Check "Element is container".

5. Use the Add binding item in the inspector's binding pull-down list to specify any additional properties of the element that don't appear in the inspector.

WOComponentContent

Like the WORepetition and WOConditional dynamic elements, reusable components can wrap HTML elements. When you are editing a reusable component, the wrapped HTML elements are collectively called the component content. Use the WOComponentContent dynamic element to refer to it (see Reusable Components).

To add the WOComponentContent, click in the toolbar.

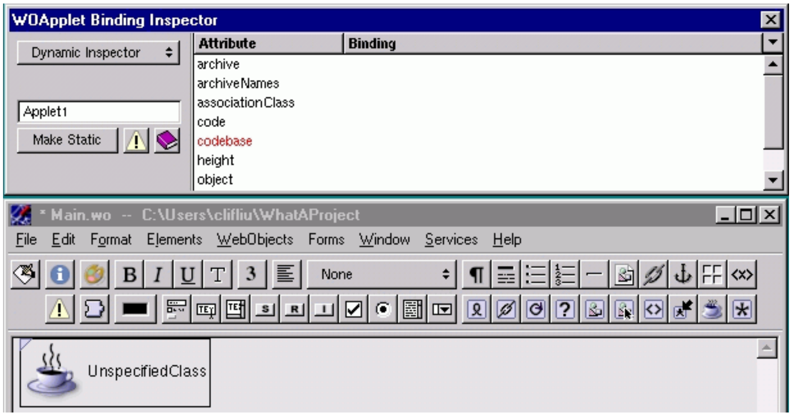
The WOComponentContent has no bindings. You can only have one WOComponentContent in a given component.

WOApplets

The WOApplet dynamic element represents a Java applet or client-side component. There are several ways to create a WOApplet. You can:

- Click ☕ in the toolbar.

  This creates a WOApplet, whose bindings you must set.



WOApplet Binding Inspector

Attribute	Binding
archive	
archiveNames	
associationClass	
code	
codebase	
height	
object	

- Drag a file of type **.class** into your component.

You are asked whether you want to add the **.class** file to your project. If you reply Yes, it is added to the Web Server Resources suitcase. A WOApplet appears in your component, with its **code** attribute set to the name of the file.

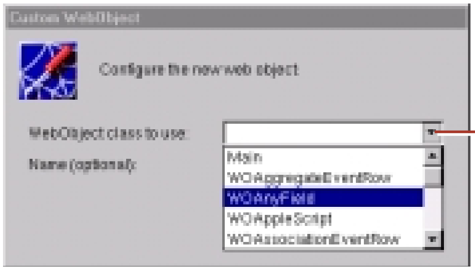
- Drag an element from the Client-Side Components palette to your component (see Palettes).

Custom WebObjects

You use custom WebObjects to implement reusable components (see Reusable Components for more details).

To create a custom WebObject:

1. Click ✱ in the toolbar.
2. In the panel that appears, specify the element class.



Custom WebObject

Configure the new web object

WebObject class to use:

Name (optional):

Select class name from the list or type it.

The WebObject Class combo box allows you to type the class name or select it from the components listed in the pull-down list. This lists all components that are in the current project and frameworks. For example, the WOAnyField and WOAppleScript components listed above are defined in the WOExtensions framework, which is included in your project by default.

If WebObjects Builder recognizes the element class, it automatically displays its attributes in the inspector. Otherwise, use the inspector to add them (see “Binding Elements” (page 74)).

The WOExtensions palette (see Palettes) contains several pre-defined custom WebObjects elements you can use in a component.

Reusable Components

One of the strengths of the WebObjects architecture is its support of reusable components. Any component that you define, whether it represents an entire page or part of a page, can be reused by any WebObjects application. A component can be used in multiple pages or even multiple times in the same page. Reusable components can be used for such items as headers, footers, and navigation bars.

When a reusable component is used inside another component, it is referred to as a *child component*; the containing component is called the *parent component*.

To reuse a component, you can either:

- Add the component to a framework and include the framework in any project that wants to use the component. The component is a *shared component* and doesn't need to be copied into each application that uses it.
- Add the component directly to your project (in the Web Components suitcase).

See Frameworks for information on creating frameworks and adding them to a project. To add a component directly to a project, you can:

- Drag a component (a folder with the **.wo** extension) from the file system onto a component window.

  You are asked whether you want to add the component to your project. If you respond Yes, the component is copied to the project and placed in the Web Components suitcase, along with all the other components.

  The child component then appears in the window at the insertion point.
- Use the toolbar to add a custom WebObject element (see Custom WebObjects) to your page, then use the Inspector to set its type to the name of the reusable component.
- Drag a component that has been stored on a palette to the component window (see Palettes).

A component that is designed for reuse can *export* keys and actions, which become attributes that the parent component can bind, just as it would set the attributes of any other dynamic element. When the component is added to a parent component, these attributes show up in the Custom WebObject Inspector. The attributes must be enumerated using the API Editor for the component.

The Inspector shows the child component's attributes. As with any other dynamic element, you can bind the child component's attributes to keys and actions in the parent component's code.

Note: When you create a component that is specifically designed to be used within other pages, specify "Partial document" in the Page Attributes Inspector pop-up list (see Setting Page Attributes). This way WebObjects Builder does not wrap <HTML>, <HEAD>, and <BODY> tags around your component.

For more information, see "Reusable Components" in the *WebObjects Developer's Guide*.

Chapter 4

Direct To Web

Direct to Web is a Java-based technology that provides a quick and easy method to create a web application that accesses a database. It lets you experiment and prototype, while also allowing you the flexibility to access the full power of WebObjects.

There are several stages you can go through, depending on your needs:

- First, you create a WebObjects project and specify a *model* to use. Direct to Web uses the model, which defines the mapping between your database and enterprise object classes, to generate an application that provides an interface to your database. This application consists of a set of pages that allow you to do queries on the entities in your database, display results, and add and delete records.

  A complete and correct model file with all the right relationships defined is key to creating a WebObjects application with Direct to Web.

- To change the way the pages are presented, you can use the Direct to Web Assistant (Web Assistant for short), which is a Java applet that runs in your web browser. For each page in your application, the Web Assistant allows you to specify which pages are shown, which properties are shown, how these properties are displayed, and the order in which they are listed. You can experiment with different configurations until you are satisfied, without writing any code.

- If you want to do customize beyond what the Web Assistant provides, you can "freeze" any or all of the pages in your application as WebObjects components. This gives you the full power of WebObjects: you can modify a component's layout using WebObjects Builder, and you can customize its behavior by writing Java code using Project Builder.

- You can also modify the templates which Direct to Web uses to generate its pages. These templates are WebObjects components that you can edit with WebObjects Builder. This way you can modify the "look" or style of the pages that Direct to Web generates.

You can also use Direct to Web in other types of WebObjects applications. Your application can take two approaches:

- Embedding Direct to Web components in your pages; these include query forms, lists, or edit/inspect forms.

- Linking to dynamically generated Direct to Web pages

This document describes the elements that make up a Direct to Web application, and shows you the steps you follow when creating and modifying an application. If you find that your application needs some specific behavior that Direct to Web does not provide, see *Developing WebObjects Applications with Direct to Web*. For more information about using Project Builder and WebObjects Builder to develop WebObjects applications, see *WebObjects Tools and Techniques* . For more information about using WebObjects with database applications, see “Creating a WebObjects Database Application” in *Getting Started With WebObjects*, as well as the *Enterprise Objects Framework Developer’s Guide*.

Creating a Direct to Web Project

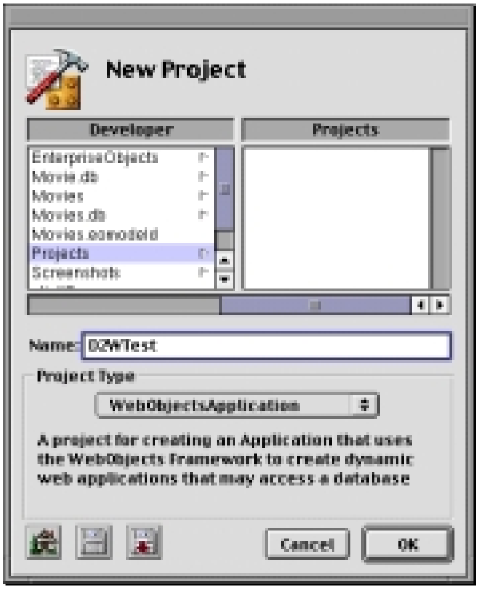
To create a Direct to Web application, begin by using Project Builder to create a WebObjects application project. Follow these steps:

1. Launch Project Builder.

2. Choose Project ► New.

3. In the New Project panel, choose the Webobjectsapplication project type from the pop-up list and specify the project path where you want to save the project.

Note: On Windows NT, the project type and project path appear on different panels.



New Project

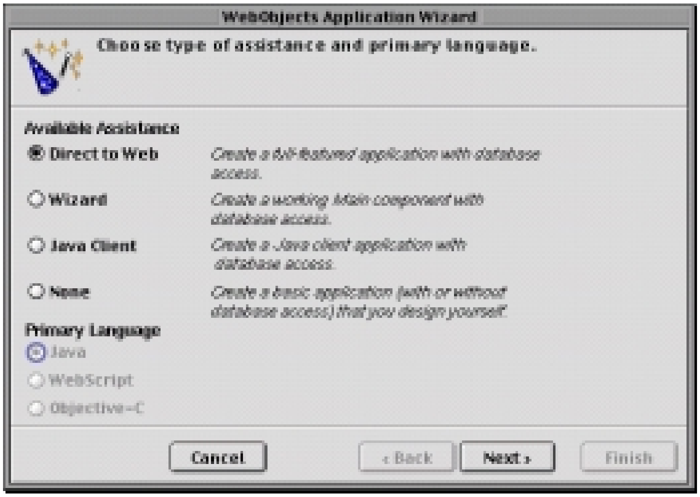
Name: D2WTest

Project Type: WebObjectsApplication

A project for creating an Application that uses the WebObjects Framework to create dynamic web applications that may access a database

4. Click OK.

   The first screen of the WebObjects application wizard appears.



WebObjects Application Wizard

Choo se type of assistance and primary language.

Available Assistance

● Direct to Web — *Create a full-featured application with database access.*

○ Wizard — *Create a working Main component with database access.*

○ Java Client — *Create a Java client application with database access.*

○ None — *Create a basic application (with or without database access) that you design yourself.*

Primary Language

● Java

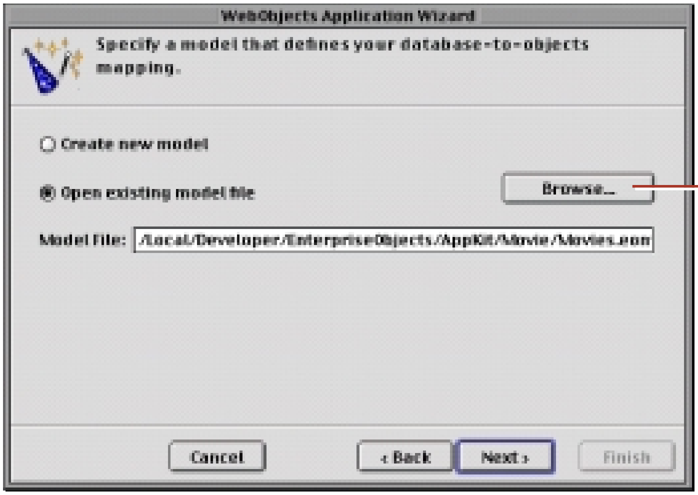
○ WebScript

○ Objective-C

5. Under Available Assistance, select Direct to Web.

You cannot select a language when the type of WebObjects application is Direct to Web; when you create a Direct to Web project, Java is the only available language.

6. Click Next.



Specify a model that defines your database-to-objects mapping.

○ Create new model

● Open existing model file

Model File: /Local/Developer/EnterpriseObjects/AppKit/Movie/Movies.eom

Click here to navigate to your model file.

7. Choose “Open existing model file.”

You can also create a new model file. If you choose “Create new model,” you are led through a series of screens that prompt you to create a new model. For more information about creating a new model file, see the chapter “Using EOModeler” in *Enterprise Objects Framework Developer’s Guide*.

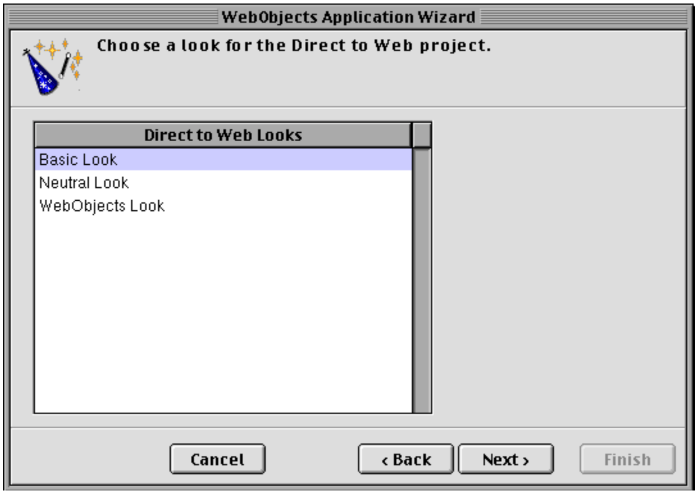
If the model you add to your project references entities in another model, you must add the other model to your project manually. The wizard doesn't include it automatically.

8. Click Browse, then navigate to the model file you want to use and select it.

If you are just exploring Direct to Web, you can use a model file from one the Enterprise Objects example projects, such as **Movies.eomodel** in the Movies project. This model is used throughout this document.

9. Click Next.

The next screen offers a selection of user-interface styles ("looks") for your Direct to Web application; see "The Different Looks for WebObjects Applications" (page 100) for more information. Click an item in the browser to select a look.



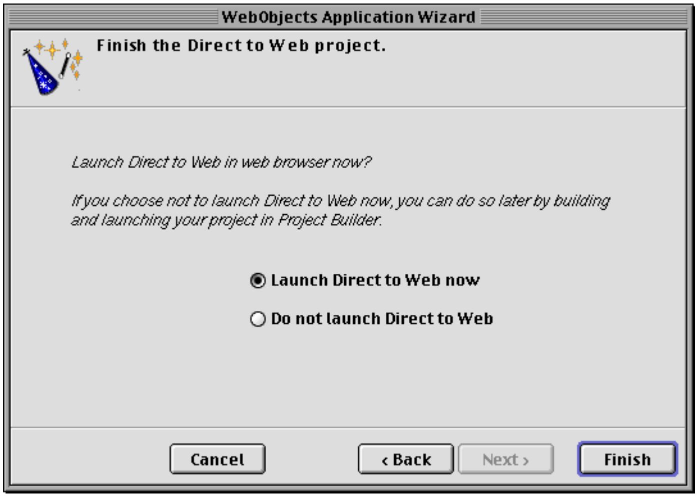
Choose a look for the Direct to Web project.

Direct to Web Looks

- Basic Look
- Neutral Look
- WebObjects Look

10. Click Next.

    The next screen asks if you would like to launch your application immediately. If you choose not to have the wizard launch your application, see "Using Your Direct to Web Application" (page 106), which tells you how to launch your WebObjects application and describes what you see when you launch it.



WebObjects Application Wizard

Finish the Direct to Web project.

Launch Direct to Web in web browser now?

If you choose not to launch Direct to Web now, you can do so later by building and launching your project in Project Builder.

● Launch Direct to Web now

○ Do not launch Direct to Web

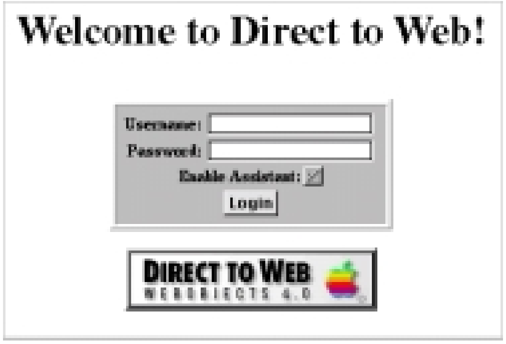
11. Click Finish to complete the WebObjects application wizard procedure.

The Different Looks for WebObjects Applications

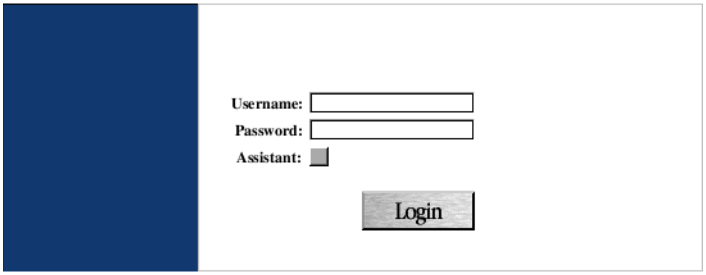
In this release, Direct to Web offers three different user-interface styles, or "looks," for WebObjects applications: Basic, Neutral, and WebObjects. Currently the only simple way to change the look of an application is to re-create a project using Project Builder and then redefine the project with the WebObjects application wizard. Therefore it is advisable to know which look you want in advance.

All the looks provide the same functions for the user. They only differ in the style and placement of their user-interface elements.The Neutral look and the WebObjects look are very similar but the Neutral look does not display the Apple logo, which makes it easier if you want to use your own logo.

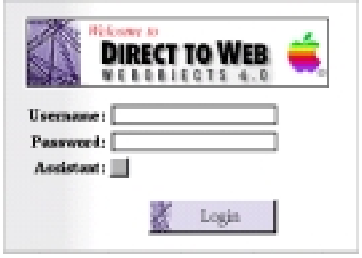
The login page for the Basic look has a panel-like submit form for the entry of user name and password:



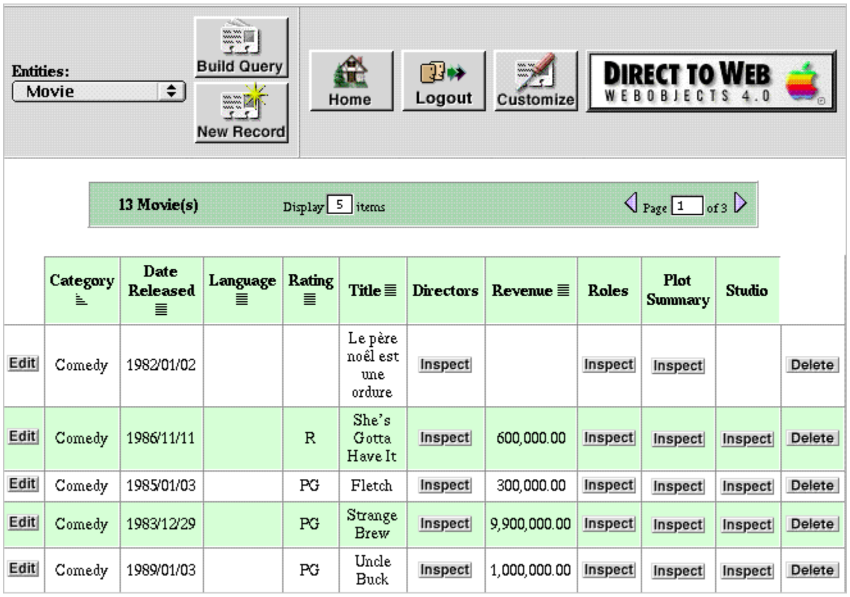
The login page for the Neutral look is much simpler:



This is the login page for the WebObjects look:



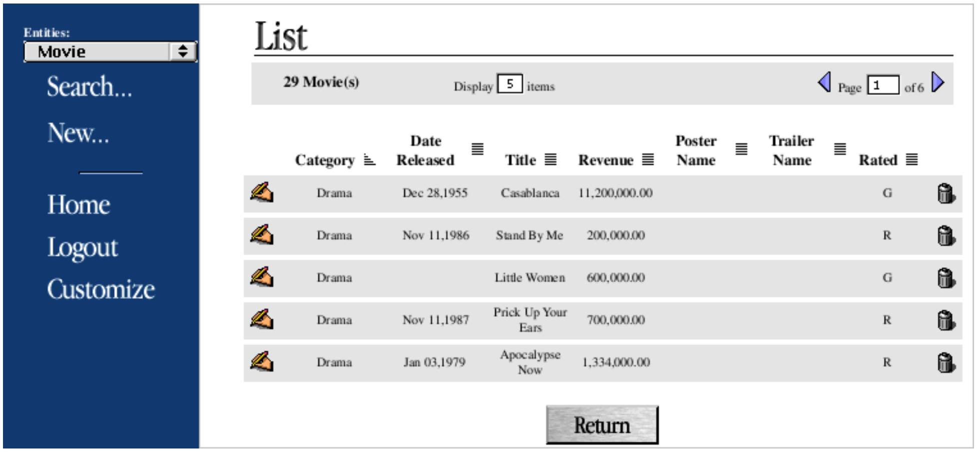
In the dynamically-generated pages (query, list, inspect, and so on), the Basic look differs from the Neutral and WebObjects looks even more strikingly. In the Basic look the control header runs across the top of the page whereas in the Neutral and WebObjects looks it appears on the left side of the page. In addition, the Basic look is more tabular while the Neutral and WebObjects looks tends to present records in visual “blocks.” For example, the following is an example of a list page in the Basic look:



13 Movie(s)

	Category	Date Released	Language	Rating	Title	Directors	Revenue	Roles	Plot Summary	Studio	
Edit	Comedy	1982/01/02			Le père noël est une ordure	Inspect		Inspect	Inspect		Delete
Edit	Comedy	1986/11/11		R	She's Gotta Have It	Inspect	600,000.00	Inspect	Inspect	Inspect	Delete
Edit	Comedy	1985/01/03		PG	Fletch	Inspect	300,000.00	Inspect	Inspect	Inspect	Delete
Edit	Comedy	1983/12/29		PG	Strange Brew	Inspect	9,900,000.00	Inspect	Inspect	Inspect	Delete
Edit	Comedy	1989/01/03		PG	Uncle Buck	Inspect	1,000,000.00	Inspect	Inspect	Inspect	Delete

The list page in the Neutral look appears like this:

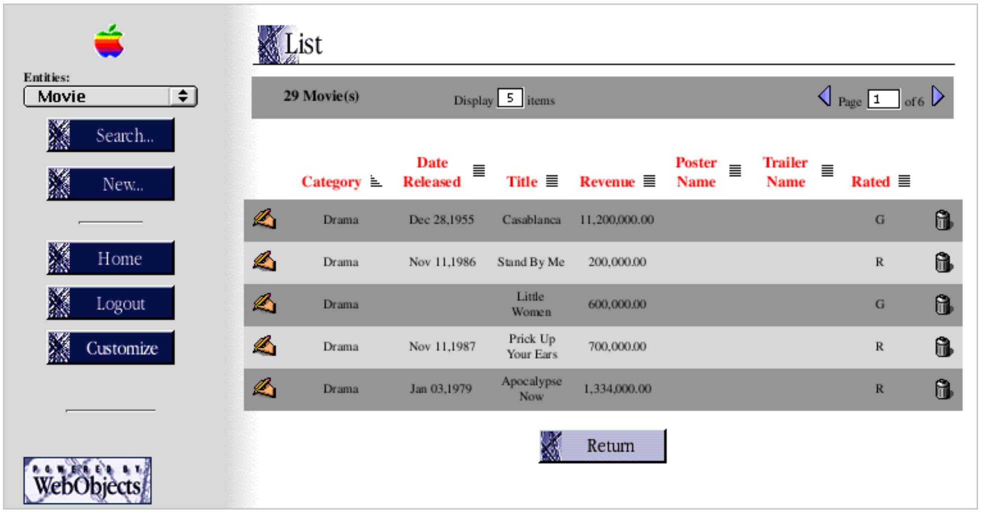


List

29 Movie(s)

Category	Date Released	Title	Revenue	Poster Name	Trailer Name	Rated
Drama	Dec 28,1955	Casablanca	11,200,000.00			G
Drama	Nov 11,1986	Stand By Me	200,000.00			R
Drama		Little Women	600,000.00			G
Drama	Nov 11,1987	Prick Up Your Ears	700,000.00			R
Drama	Jan 03,1979	Apocalypse Now	1,334,000.00			R

The following illustrates what a list page looks like in the WebObjects look.

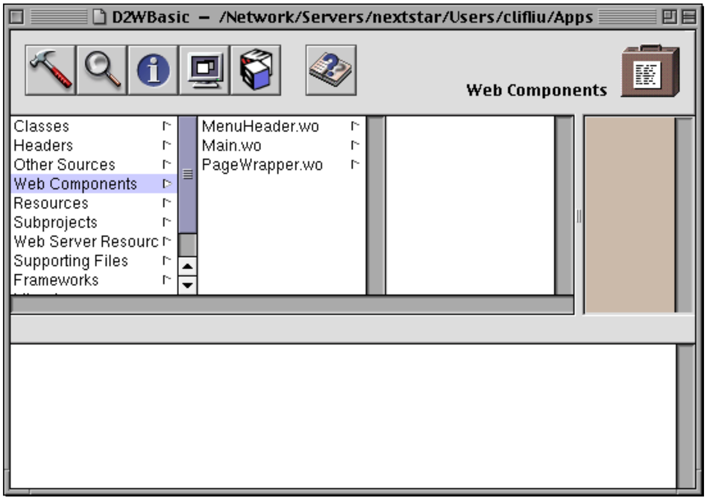


29 Movie(s) Display 5 items Page 1 of 6

	Category	Date Released	Title	Revenue	Poster Name	Trailer Name	Rated	
	Drama	Dec 28,1955	Casablanca	11,200,000.00			G	
	Drama	Nov 11,1986	Stand By Me	200,000.00			R	
	Drama		Little Women	600,000.00			G	
	Drama	Nov 11,1987	Prick Up Your Ears	700,000.00			R	
	Drama	Jan 03,1979	Apocalypse Now	1,334,000.00			R	

The Structure of a Direct to Web Project

A Direct to Web project has a structure similar to other WebObjects application projects. A newly created project contains three components:



- **MenuHeader.wo** is a reusable component that contains the header with the control buttons on the left side of each page (or the top of the page in the

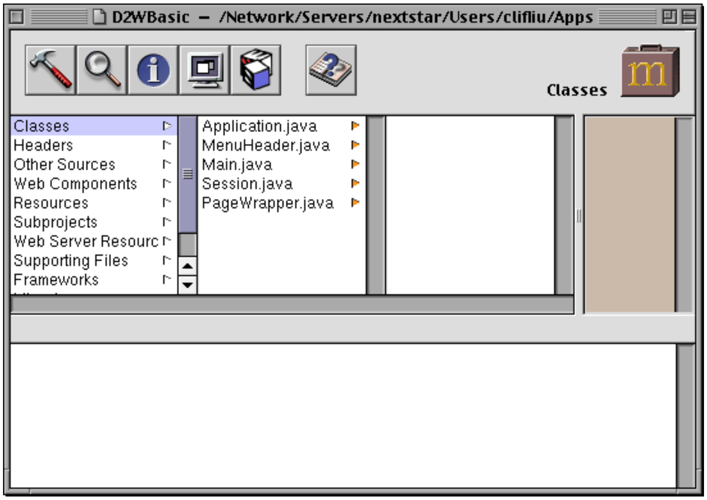
Basic look.) You can add text or other elements to this component if you choose.

- **Main.wo** is the main component, representing the login page of the application.

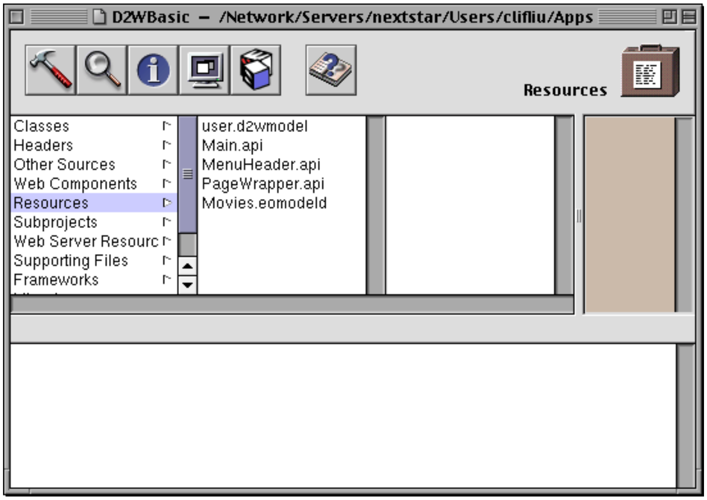
- **PageWrapper.wo** is a reusable component that wraps the content of the pages of the application (except for **Main.wo**). It contains a header, the menu header component (**MenuHeader.wo**), and footer text and elements common to these pages. If you want to customize the headers and footers for all pages of your application, you can add text or other elements to this component.

As you run your application, Direct to Web creates additional pages, using information in your model file and the settings specified in the Web Assistant. These pages do not show up as components in your project. Rather, Direct to Web creates them dynamically using a set of reusable components in the Direct to Web framework. However, you can save any page as a component or generate a user template. When you do that, you are then able to modify the component just as you would with any other WebObjects component. See "Generating Components" (page 132) and "User Templates" (page 135) for more information.

In your project's Classes suitcase, you'll see a Java file for each of the components, as well as the Session and Application objects. You can add code to any of these files to extend their functionality.



The Resources suitcase contains the model file you specified when you created the project (in this example, **Movies.eomodeld**). It also contains **user.d2wmodel**, which stores the preferences you have specified using the Web Assistant. Advanced users can edit this file; see *Developing WebObjects Applications With Direct to Web* for more information about the rule file. The Resources suitcase also holds files specifying the exported keys, both optional and required, for each type of component used in the application; these files have an extension of **.api**.



Using Your Direct to Web Application

Once you have created a Direct to Web application using Project Builder and the WebObjects application wizard, and have compiled the resulting project files, you can launch the application using Project Builder's Launch panel. The application pages are displayed in a web browser, where you can test the application's presentation of data and, with the Web Assistant enabled, modify the layout of that data.

Launching a Direct to Web Application

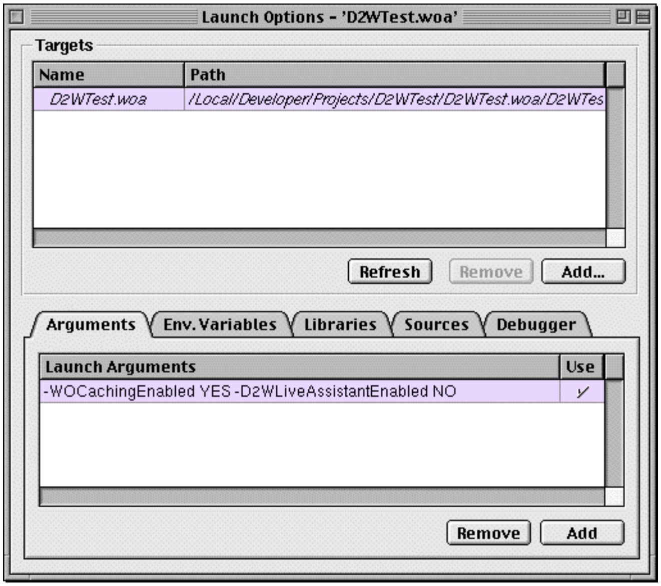
To launch your application from Project Builder:

1. Click 🖥 in the toolbar in Project Builder's main window to open the Launch panel.

2. Click 🖥 in the Launch panel to launch your application.

Before you launch the application you might want to set some command line options. For example, when running a Direct to Web Application for deployment, you should turn on caching and disable the Web Assistant (to prevent anyone from connecting to the application using the Web Assistant). To do this, set the `-WOCachingEnabled` and `-D2WWebAssistantEnabled` options, respectively:

1. Click ☑ to bring up the Launch Options panel.



Launch Options - 'D2WTest.woa'

Targets

Name	Path
D2WTest.woa	*/Local/Developer/Projects/D2WTest/D2WTest.woa/D2WTes*

Launch Arguments	Use
-WOCachingEnabled YES -D2WLiveAssistantEnabled NO	✓

2. Click the Arguments tab.

3. Click Add to create a new command line and type the entries as shown in the above example. If there is no checkmark under the Use column, double-click the line under Use to set it.

For other command-line options for WebObjects applications, such as `-WOPort`, see *Serving WebObjects*.

You can test the Direct to Web application using a web browser on a machine remote from the machine on which the application is running (that is, the server). When you launch the application, look in the console output, which is displayed in the Launch panel, for the line containing application's URL.

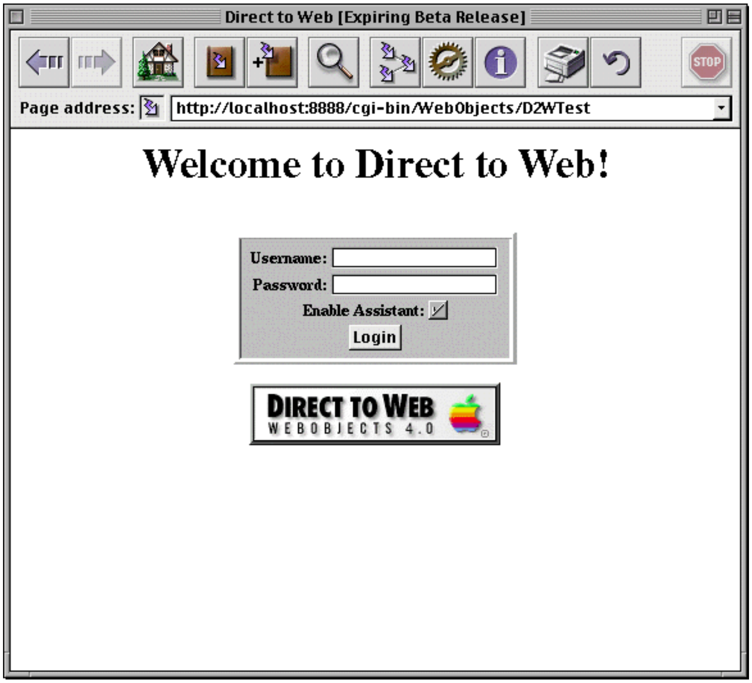
```
Jul 28 09:48:52 D2WTest[2777] Your application’s URL is:
http://localhost:1234/cgi-bin/WebObjects/D2WTest
```

Enter the URL in your browser, after substituting the host name of the server machine for "localhost". In fact, you can exclude every thing in the URL after the application port number. For example, if the server host name is "foobar" you would enter the following URL in the browser to load the WebObjects application:

```
http://foobar:1234/
```

The Login Page

When you launch your application, your web browser displays the Direct to Web login screen:


Direct to Web [Expiring Beta Release]
Page address:
http://localhost:8888/cgi-bin/WebObjects/D2WTest
STOP

Welcome to Direct to Web!

Username:
Password:
Enable Assistant: ☑
Login

DIRECT TO WEB
WEBOBJECTS 4.0

The login page is the default implementation of your Main component, **Main.wo**. It contains text fields to enter a name and password, as well as a submit button (Login) and an Enable Assistant checkbox. To go to the application's default first page, check Enable Assistant and click the Login button. You don't need to enter a name and password, because the default application provides no password-checking logic. If you don't check Enable Assistant before clicking the Login button, you won't have access to the Web Assistant.

You can modify the login page (**Main.wo**) to provide any behavior or appearance you like. For example, you can add your own password-checking logic.

Dynamically Generated Pages

Besides the login page, there are nine types of dynamically-generated pages (or reusable components) in a Direct to Web application:

- A *query-all page* that displays all entities that are currently exposed and lets users construct queries on the attributes (but not the relationships) of those entities; see "Query Pages" (page 110). The properties of this page cannot be customized.

- A *query page* that allows the user to construct a query for a particular entity; see "Query Pages" (page 110).

- A *list page* that displays one or more records of a particular entity in tabular form. List pages and select components are implemented with the same components; see "List Pages and Select Components" (page 113). The result of a query is always a list page.

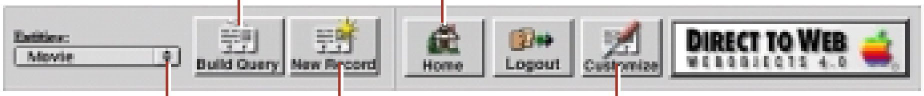
- An *inspect page* that displays a single record of a given entity. Inspect pages and edit pages are implemented with the same components; see "Inspect and Edit Pages" (page 115).

- An *edit page* that displays a single record of a given entity and also allows you to make changes to the record and save it to the database. Edit and inspect pages are implemented with the same components; see "Inspect and Edit Pages" (page 115).

- A *select component* that lets users select a record from a list, thereby adding it to a relationship or populating an edit component with it. List pages and select components are implemented with the same components; see "List Pages and Select Components" (page 113).

- A *confirm page* that prompts users to confirm that they want to delete records. The properties of this page cannot be customized.
- An *edit-relationship page* is a multiple component page for removing and adding objects to a relationship. See “Edit-Relationship Pages” (page 116).
- An *error page* for displaying information related to exceptions and other errors. The properties of this page cannot be customized.

All pages in your application contain the standard Direct to Web header (defined in **MenuHeader.wo**) at the top of the page. This header provides a number of controls, described in the following figure.



Click here to build a query for the selected entity.

Click here to return to the first page of your application.

Click here to activate the WebAssistant..

Click here to create a new record of the selected entity type.

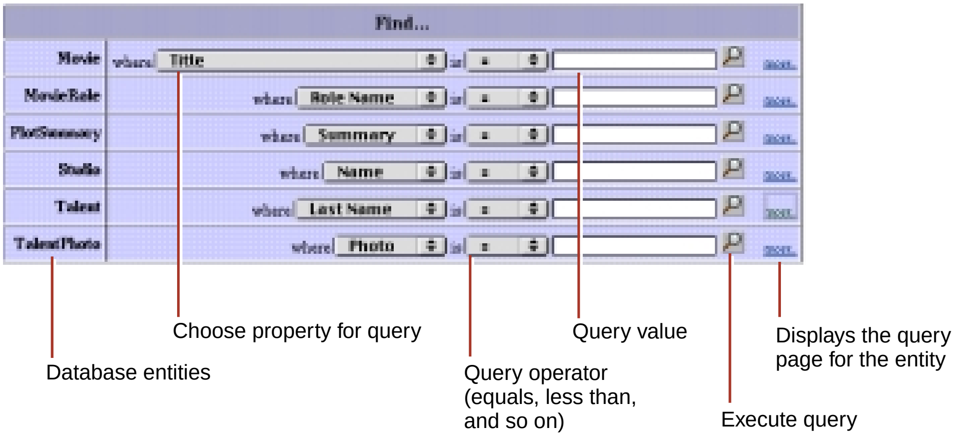
Select an entity to query on from this pop-up list.

For best results when navigating through a Direct to Web application, don’t use your web browser’s backtrack buttons. Instead:

- To return to the previous page from an edit or inspect page, click Cancel.
- To return to a query page from a list page, click Return.

Query Pages

Direct To Web has two kinds of pages for constructing queries on the properties of entities: a query-all page and a query page. When you log into a Direct To Web application, the query-all page is displayed first by default.



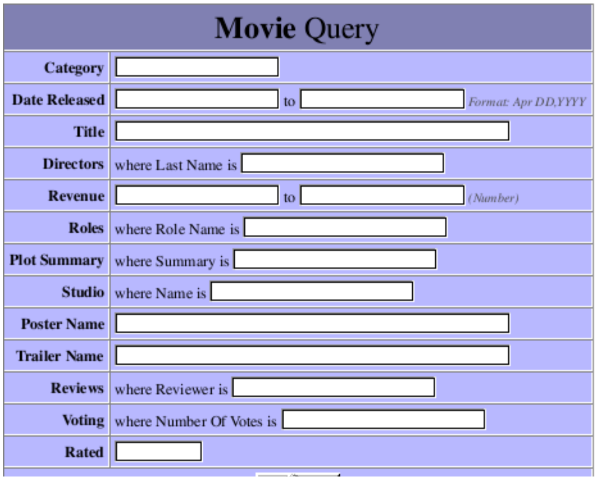
Find...

Movie	where Title	is =		more...
MovieRole	where Role Name	is =		more...
PlotSummary	where Summary	is =		more...
Studio	where Name	is =		more...
Talent	where Last Name	is =		more...
TalentPhoto	where Photo	is =		more...

Choose property for query
Query value
Displays the query page for the entity
Database entities
Query operator (equals, less than, and so on)
Execute query

The query-all page enables you to construct a query on an attribute of a particular entity (queries on relationships are not allowed). To use this page, select a property from an entity's pop-up list, specify the comparison operator, type the string to search on. and click the magnifying-glass button.

The query page, on the other hand, is tied to a particular entity but allows you to construct queries on relationships as well as attributes. The following example illustrates a query page:



Movie Query

Category	
Date Released	to Format: Apr DD,YYYY
Title	
Directors	where Last Name is
Revenue	to (Number)
Roles	where Role Name is
Plot Summary	where Summary is
Studio	where Name is
Poster Name	
Trailer Name	
Reviews	where Reviewer is
Voting	where Number Of Votes is
Rated	

The first column in the table lists the current entity's properties. The second column contains pop-up lists and text fields that let you enter values to construct queries on single and multiple properties. When you specify values for multiple properties, the query becomes the logical AND of the queries on the individual properties.

A property is either an *attribute* (a value stored directly in this entity's table) or a *relationship* (an association between this entity and another entity). For example, in the figure above, Title is an attribute and Studio is a relationship. You can use the Web Assistant to hide properties that you don't want users to see.

Note: Direct to Web only displays properties that are class properties. In addition, primary keys and attributes marked as the source of a relationship are hidden by default.

Properties are represented in various ways. For example, in the figure, you enter a single string value for Title, while you enter a range of values for Date Released. You can change the representation of most properties using the Web Assistant. In particular, you may want to change how relationships are shown, since by default, you query them by specifying an ID, which is something the user is unlikely to know. See "Changing How Properties Are Displayed" (page 126) for more information on the different ways of representing properties in your application's pages.

You can use initial characters and special characters in query fields for string searches. For example, you could enter "sh" in the Movie entity's Title to search for all movies that begin with those characters. You can also use the asterisk character to indicate "all occurrences." For instance, "*love" would return all movies that contain the substring "love".

In the Movie query, to get a list of all dramas released in the 1990's, you would:

1. Enter `Drama` in the Category field.
2. Enter `1980/1/1` and `1989/12/31` in the Date Released fields.
3. Click Query DB.

   The results are displayed in a list page; see "List Pages and Select Components" (page 113).

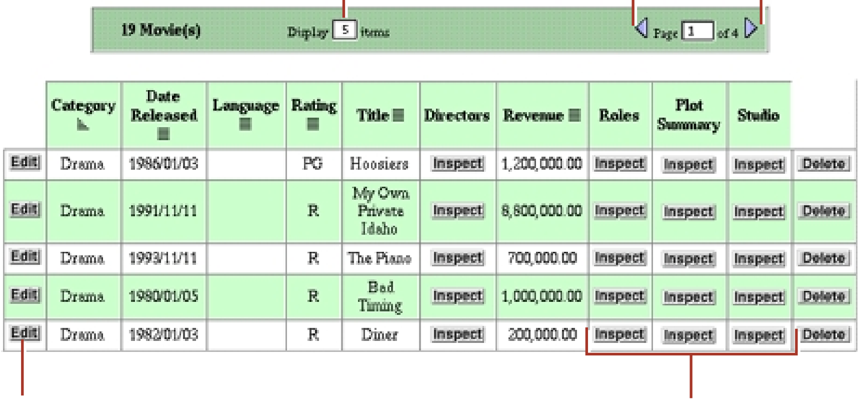
To clear the query page, click Build Query.

List Pages and Select Components

A list page displays a table showing multiple records of an entity. List pages are used to display the results of a query, or to show the records satisfying a to-many relationship in another list or inspect page.

Type a number here and press Enter to change batch size.

Click arrows to back and forward a batch, or enter a batch number.



19 Movie(s) Display 5 items ◁ Page 1 of 4 ▷

	Category ◣	Date Released ■	Language ■	Rating ■	Title ■	Directors	Revenue ■	Roles	Plot Summary	Studio	
Edit	Drama	1986/01/03		PG	Hoosiers	Inspect	1,200,000.00	Inspect	Inspect	Inspect	Delete
Edit	Drama	1991/11/11		R	My Own Private Idaho	Inspect	8,800,000.00	Inspect	Inspect	Inspect	Delete
Edit	Drama	1993/11/11		R	The Piano	Inspect	700,000.00	Inspect	Inspect	Inspect	Delete
Edit	Drama	1980/01/05		R	Bad Timing	Inspect	1,000,000.00	Inspect	Inspect	Inspect	Delete
Edit	Drama	1982/01/03		R	Diner	Inspect	200,000.00	Inspect	Inspect	Inspect	Delete

Click here to bring up an Edit page for the record shown in this row.

Click Inspect to display a list page showing the relationship's destination records.

Each row in the table represents a record. By default, a batch of ten records are shown in a page. To change the batch size, type a number in the "Display _ Items" field and press Return or Enter. To display additional records in either direction, click the triangle buttons or enter the page number you want to go to.

Each column in the list represents one of the entity's properties. By default, all properties are shown in alphabetical order. You can hide columns and change their order by using the Web Assistant; see "Customizing Your Application With the Web Assistant" (page 119).

The symbols to the right of attribute names represent their sort order:

- ◣: ascending order
- ≡: descending order
- ■: unsorted

To change the sort order for any attribute, click the title to cycle between ascending, descending, and unsorted. By default, the records are sorted in ascending order by the attribute in the first column. You can specify up to three columns to sort on; the last one specified becomes the primary sort key.

For properties that represent relationships, an Inspect button appears in the cell by default (DisplayToManyFault).

Note: By default, the list page does not display relationships (including the Inspect buttons). You can configure the list page to display relationships using the Web Assistant; see "Customizing Your Application With the Web Assistant" (page 119).

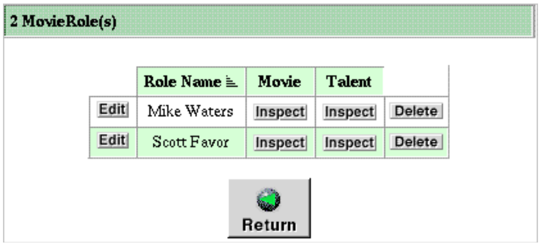
When you click the Inspect button one of two things happen, depending on the type of relationship:

- If it is a to-one relationship, an inspect page appears, showing the destination record.

  In the above example, the Movie entity's Studio relationship is a to-one relationship to the Studio entity. If you click the Inspect button, an inspect page appears for the Studio entity corresponding to the selected movie; see "Inspect and Edit Pages" (page 115).

- If it is a to-many relationship, another list page appears, showing all the destination records in the relationship.

  In the above example, the Movie entity's Roles relationship is a to-many relationship to the MovieRole entity. If you click the Inspect button, a list page appears, showing all the roles in the selected movie.

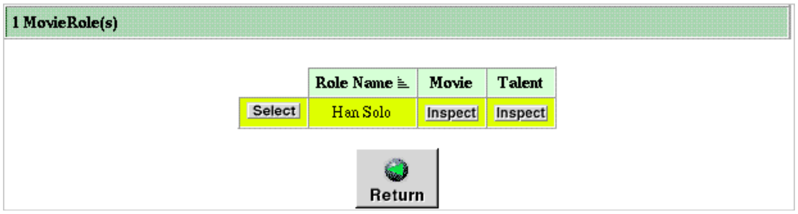


2 MovieRole(s)

	Role Name	Movie	Talent	
Edit	Mike Waters	Inspect	Inspect	Delete
Edit	Scott Favor	Inspect	Inspect	Delete

You can use the Web Assistant to display the related records directly in the table instead of with an Inspect button; see "Customizing Your Application With the Web Assistant" (page 119).

The select component looks a lot like the list page, but instead of the Edit button there is a Select button. The select component occurs in multiple-component pages. In the edit-relationship page you click Select to add a record to a to-many relationship or select a record for a to-one relationship. In the master-detail page you click Select to select a record to edit. A select component looks like this:



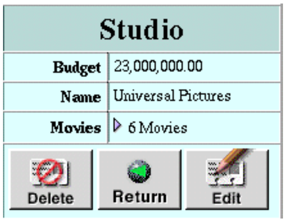
1 MovieRole(s)

	Role Name	Movie	Talent
Select	Han Solo	Inspect	Inspect

Inspect and Edit Pages

Inspect pages and edit pages display the data for a single record of an entity. An edit page allows you to make changes to the record and save the changes, while an inspect page is read-only.

An inspect page looks like this:



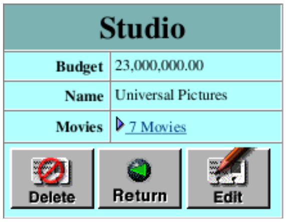
Studio

Budget	23,000,000.00
Name	Universal Pictures
Movies	▷ 6 Movies

Note the buttons at the bottom of the page:

- Delete allows you to delete the record from the database.
- Cancel takes you back to the page from which you accessed this inspect page.
- Edit brings up the equivalent edit page for this record, so that you can make changes. (However, if your application specifies a particular entity as read-only, you won't be able to edit it.)

Also note the Movies property in the example above. You click the triangle to display the movies of this studio in a list, browser, or table, as in the following example:

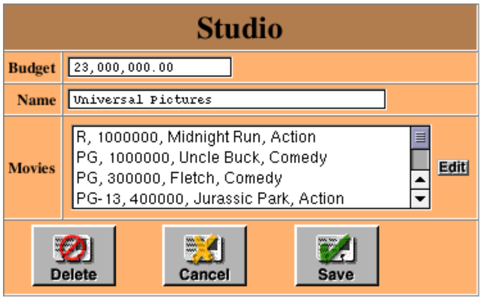


Studio

Budget	23,000,000.00
Name	Universal Pictures
Movies	▶ 7 Movies

This property is configured with the DisplayToManyTable component. For more on how this is done, see “Representation of Relationships” (page 128).

An edit page (or edit component) looks like this:



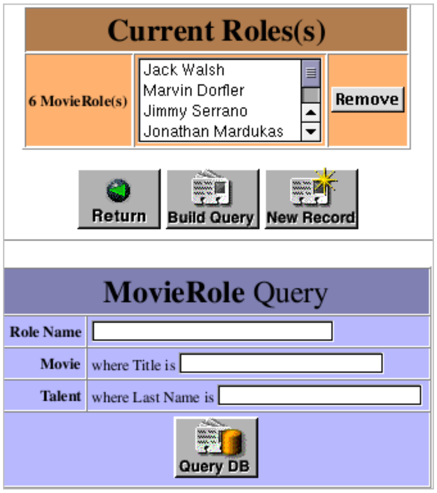
Studio

Budget	23,000,000.00
Name	Universal Pictures
Movies	R, 1000000, Midnight Run, Action PG, 1000000, Uncle Buck, Comedy PG, 300000, Fletch, Comedy PG-13, 400000, Jurassic Park, Action

It is similar to the inspect page, except that it has a Save button (for saving changes to the database) instead of an Edit button. If you click the Edit button next to the list of Movies, an edit-relationship page is displayed for editing the records in the to-many relationship. Edit components can occur in multiple-component pages, such as the master-detail page.

Edit-Relationship Pages

An edit-relationship page allows users to add records to a relationship and remove records from the relationship. Users typically come to these pages when they click an Edit button next to a relationship in an edit page. Edit-relationship pages consist of three separate components, of which two are shown at any one time. The first component lists the relationships of a particular property and contains several controls. In addition, a query component initially appears for locating another object to link to for that property. The third component, a select component, appears after you have specified a query and is discussed below.



Current Roles(s)

6 MovieRole(s)

Jack Walsh
Marvin Dorfler
Jimmy Serrano
Jonathan Mardukas

MovieRole Query

Role Name

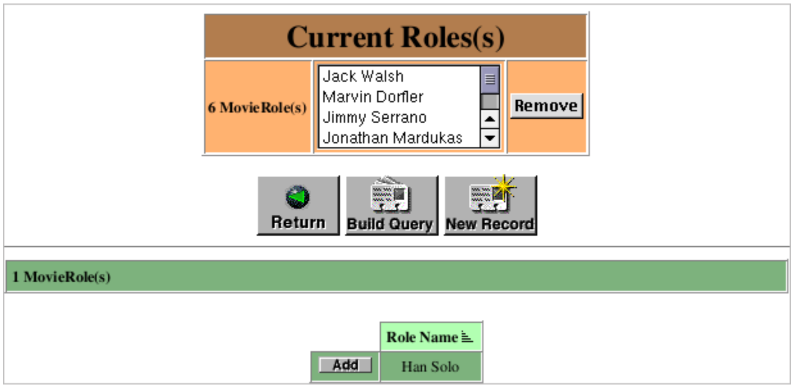
Movie where Title is

Talent where Last Name is

This user interface facilitates the following tasks:

- To remove a record from the property, select the key identifying the record in the browser and click Remove.
- To add a new record to the property, click New Record. An edit component appears underneath the list of relationships; fill out the fields of the edit component and click Save to add the new record to the database *and* the new relationship to the property above.
- To locate an existing record to add to the relationship, enter the properties to search on in the query component and click Query DB.

When a query is executed (assuming matching records are found) a select component replaces the query component.

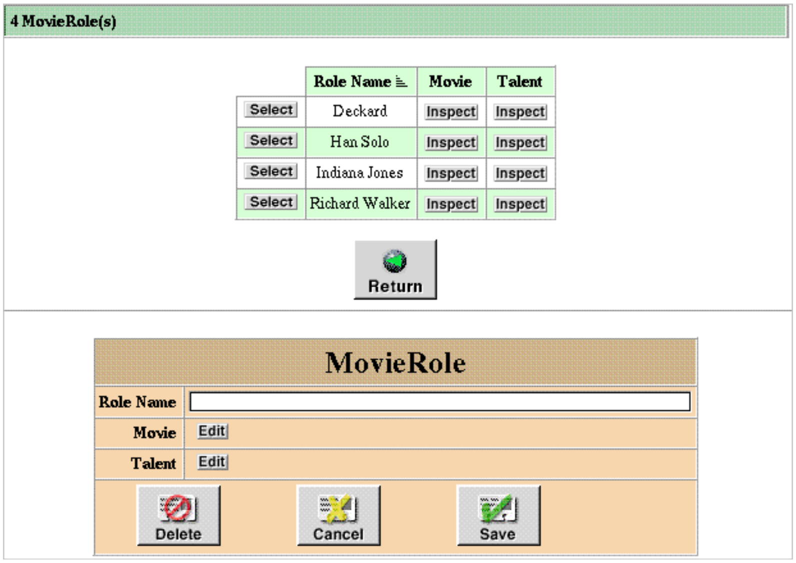


To add a listed record to the to-many relationship, click the Select button. To construct a new query, click the Build Query button.

When you have finished editing a relationship, click the Return button under the browser to return to the original edit page. You must click the Save button in this page to store the changed relationship in the database.

Master-Detail Pages

Master-detail pages put a select component and an edit component on the same page, thereby allowing users to select and edit records without having to go to another page. The following is an example of a master-detail page:



4 MovieRole(s)

	Role Name	Movie	Talent
Select	Deckard	Inspect	Inspect
Select	Han Solo	Inspect	Inspect
Select	Indiana Jones	Inspect	Inspect
Select	Richard Walker	Inspect	Inspect

MovieRole

Role Name	
Movie	Edit
Talent	Edit

To use a master-detail page, click Select next to a record in the list component. The record is displayed in an edit component. See “Inspect and Edit Pages” (page 115) for usage information.

The master-detail page does not appears under Tasks in the Web Assistant (expert mode). This is because it is defined as a type of list page (BASMasterDetailPage, NEUMasterDetailPage, or WOLMasterDetail page depending on the look) of the list task.

Customizing Your Application With the Web Assistant

The Web Assistant allows you to customize each page of your application. You can specify:

- Which entities of the model the application displays and, of these, which are read-only
- Global attributes of pages, such as style, color, and border thickness
- Which properties are displayed, and in what order

  By default, an entity’s properties are listed in alphabetical order. Often, you’ll want to change the order, as well as hiding some properties.

- How number and date strings should be represented
- How relationships should be represented

To activate the Web Assistant, click Customize in the Direct to Web header. A Java applet window appears showing the Web Assistant.

When you have activated the Web Assistant in your browser, a frame appears at the bottom of each page in your application in the browser (assuming it supports Java applets), containing a “Show WebAssistant” button and a status field. To bring the Web Assistant to the front, click the Show WebAssistant button (rather than clicking Customize again).

Running the Web Assistant With appletviewer

If you browser is incapable of running applets, or if you want to run the Web Assistant in a different machine from your browser, you can launch the Web Assistant using the Java program **appletviewer**. To do this:

1. Launch your application with the command-line option `D2WWebAssistantEnabled` set to `YES`.
2. In the console output look for a line similar to the following:

   ```
   Jul 23 10:29:48 D2WTest[527] Server-side WebAssistant launch line:
   appletviewer http://localhost:8888/cgi-
   bin/WebObjects/D2WTest.woa/wa/D2WActions/openWebAssistant
   ```

3. Open a shell such as provided by the Terminal application on Mac OS X Server systems or the Bourne Shell on Yellow Box for Windows systems.
4. Copy the string from “appletviewer” to “openWebAssistant” to the shell and press Return (or Enter).

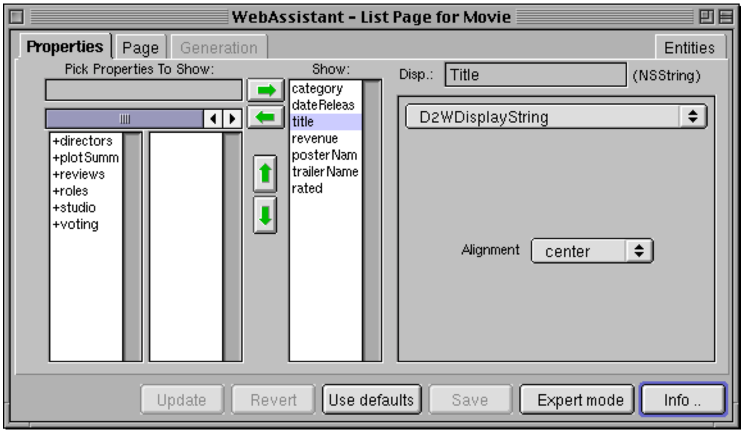
   If the port number is -1, look in the console output for the actual port number of the application and substitute that.

When you complete this procedure, the Web Assistant launches and is connected to your application. If you stop and restart the Direct to Web application, the Web Assistant will re-connect to it provided it is running on the same port.

A standalone Web Assistant has exactly the same functionality as one launched inside your browser. However, if the browser you are using is not Java-enabled, your pages are not automatically refreshed after you click Update. You must either click your browser's "reload" or "refresh" button or (when you are picking a new type of page, such as a MasterDetails page instead of a ListPage), you will have to re-navigate to the same page.

Web Assistant Overview

When the Web Assistant applet is launched, it appears in a window whose title indicates the current page and entity:



WebAssistant - List Page for Movie

Pick Properties To Show:
+directors
+plotSumm
+reviews
+roles
+studio
+voting

Show:
category
dateReleas
title
revenue
posterNam
trailerName
rated

Disp.: Title (NSString)

D2WDisplayString

Alignment center

The Web Assistant has four displays, each selectable by clicking a tab:

- **Properties**. Allows you to set which properties of an entity are shown in a page, the order in which they're displayed, and the display characteristics of properties (for example, color and alignment).

- **Page**. Allows you to customize global page properties, such as template, overall style, color, and border thickness.

- **Generation.** Only available in expert mode, this display allows you to generate templates and "freeze" customized pages as reusable components.

- **Entities**. Allows you to select which entities of the model are hidden, which are shown, and which are read-only.

The Web Assistant stays synchronous with your browser. When you navigate to a new page, it displays the settings for that page.

The Web Assistant has two modes, Standard mode and Expert mode. By default the Web Assistant opens in Standard mode, which lets you customize the current page in your application. When you customize a page in Standard mode, the changes apply to all occurrences of that page, and that page only. For example, if you change the order of properties in an edit page for the Movie entity, then any time a Movie edit page is displayed, those changes are in effect (provided you have clicked Update or Save). However, the changes don't apply to a Movie query, list, or inspect page; if you want to customize those in the same way, you must do so explicitly.

Using Web Assistant's Expert mode, you can customize any page in the application, regardless of whether it is currently displayed. Thus, by specifying the "*all*" setting in Expert mode, you could change all pages of a given entity at once. In addition, you can generate a template or "freeze" a page as a reusable component. For more information, see "Web Assistant Expert Mode" (page 130).

When you've made changes to a page, you can use the buttons at the bottom of the Web Assistant to apply them:

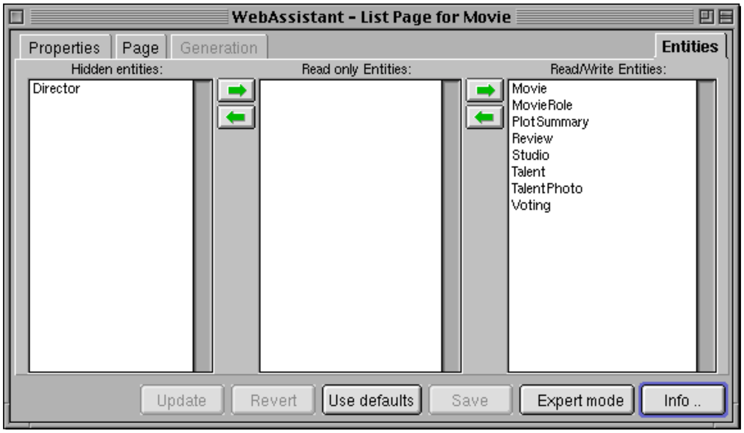
- *Update:* Sends your changes to the server. On some systems this causes the page to be refreshed in your browser.
- *Revert:* Causes all settings to revert to their last saved values.
- *Use Defaults:* reverts all settings to the values they had when the project was created.
- *Save:* Saves the changes to disk. You need to save your changes in order for them to persist beyond the current session.

The *Info* button displays a brief description of the currently selected Direct to Web component.

Restricting Access to Entities

The Entities display of the Web Assistant enables you to specify which entities of the database model appear in the application. Of those entities, it further allows you to specify which are read-only and which the user can write data to. Records from read-only entities are restricted from appearing in edit pages.

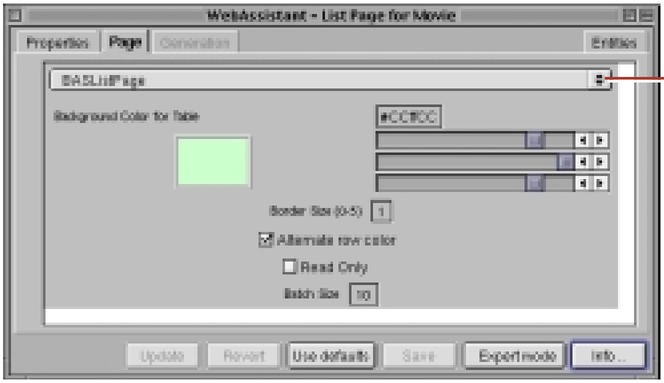
The user interface for accomplishing these tasks is simple, as the following example illustrates:



To specify an entity that shouldn't appear in the application, select it and use the arrow keys to move it to the Hidden Entities column. To specify an entity that should be read-only, select it and use the arrow keys to move it to the Read-only Entities column. You can also press Enter (or Return) to move selected entities right to left. By default, all entities initially appear in the Read/Write Entities column.

Customizing Pages

The Customize Page display of the Web Assistant enables you to set global attributes for the current page. These attributes include the page template, the color of the table, whether this color alternates with another color in lists, and the size of the border enclosing the page. The following is an example of the Customize Page display:



BASListPage

Background Color for Table #CCFFCC

Border Size (0-5) 1

☑ Alternate row color

□ Read Only

Batch Size 10

Click here to choose the template for this page.

- To change the template defining the page style, choose another template from the pop-up list.

- To change the thickness of the border around the page, type a number in the Border Size field, replacing the current number. (A border thickness of five pixels is the maximum allowed.)

- To change the color of the table, move the sliders to the right of the sample color. The color specification is RGB-based (that is, a specific mixture of red, green, and blue). The top slider manipulates red saturation, the middle slider is for green, and the bottom slider is for blue. The three pairs of hexadecimal digits after the number sign in the field represent (left to right) saturation levels of red, green, and blue.

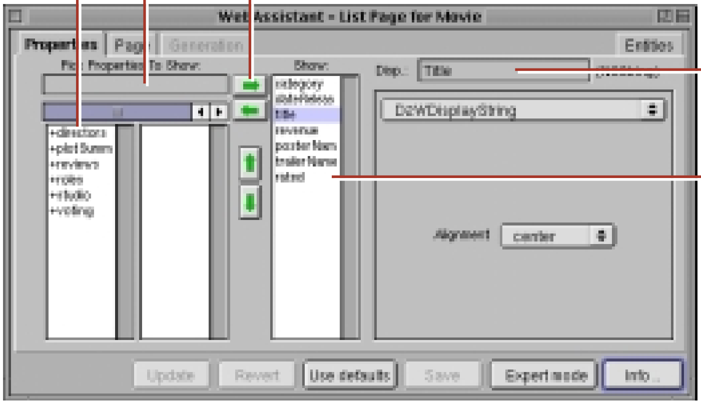
Setting Which Properties are Displayed

The Customize Properties display of the Web Assistant enables you to specify which properties of an entity appear in a page (or component) and the order in which these properties appear. Most of the user-interface elements for accomplishing these things are in the left half of the display as shown in the following example:

Choose a property to show with the key path browser...

Or type it into this field.

Use the arrow keys to show/hide properties or change their order.



Disp.: Title

Alignment: center

Enter a label for the property here.

These properties appear in the page.

The entity's properties (attributes and relationships) in the Show column are displayed in the page. To the left of the arrows is a key browser that shows relationships (which appear with a "+") and hidden attributes. You can click on a relationship to show its attributes and relationships in the next column of the key browser.

The Web Assistant displays the keys that can be found in the entity's source code. If you want to show a key or key path that doesn't appear, you can type in the text field.

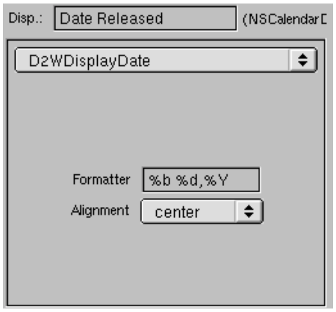
For each property, you can:

- Move it to the key browser by selecting it and clicking the left arrow. This hides the property. Similarly, if a property is hidden, you can show it by selecting it and clicking the right arrow.
- Move it up or down in the list by clicking the up and down arrows. This changes the order of appearance of the properties in the page (left to right or top to bottom, depending on the component).

You can also change the title for a property by editing the string in the Display (Disp.) field. This change only affects the way the property is labeled in the page, and has no effect on the actual property name.

Changing How Properties Are Displayed

You can use the Customize Properties display of the Web Assistant to specify various display characteristics of properties, such as formatting, color, alignment, and the representation of relationships. The fields and controls for setting these characteristics are on the right half of the display. Here is an example:



Disp.: Date Released (NSCalendarD

D2WDisplayDate

Formatter %b %d,%Y

Alignment center

Let's go over the various elements of this part of the user interface:

- At the top is the Display field, which holds the title of the property for the current page and entity. As discussed in "Setting Which Properties are Displayed" (page 124), you can edit this string.

- Next to the Display field, in parentheses, is its data type. The data type determines the set of display components available for use. You cannot edit this information directly (however, you can edit the model file, which specifies the data type, using EOModeler).

- The WOComponent group (or "box") contains a pop-up list showing the name of the component that is used to display the selected property in the current page. From this menu you can choose a different component to display the property. When you choose a display component, the set of controls and fields in the WOComponent group can change.

The items in the WOComponent pop-up list identify reusable components in the Direct to Web framework which are used to generate the pages you see in your application. Each property in a page of any type is initially shown in a default way for that type and is based on a certain component.

Textual Attributes and Formatting

The display components available for the currently selected property offer characteristics suitable to the data type and function of the attribute. A few examples might help to clarify this statement:

- If the data type of the attribute is an NSString (or String in Java) but it is a URL, then the DisplayHyperlink or DisplayMailTo components could be what you want.
- If the attribute is a date (NSCalendarDate), then you might choose the DisplayString component and provide format specifiers to have the date formatted in a certain way.
- Similarly, if the attribute is a currency value (NSNumber), you might want to use the DisplayNumber component and format the display of the attribute with two decimal positions and a leading dollar sign.
- If you want to highlight a certain column of values in a table by giving them a different color, then you could choose the DisplayStyledString component which lets you apply a color to a property.

You can click the Info button in the Web Assistant to get a short description of the currently selected display component.

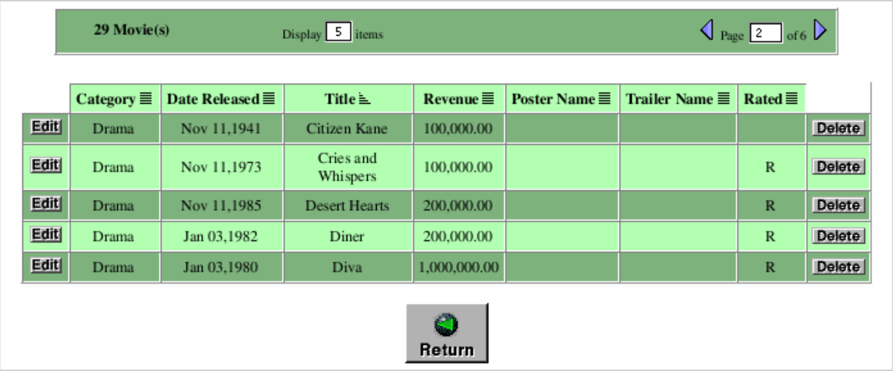
The three most common display characteristics for properties are alignment, formatting, and color. Each of these has their own controls or fields in the WOComponent group:

- **Alignment**. Choose Right, Center, or Left from the pop-up list to specify the alignment of text within a cell of a table.
- **Formatter**. You can have your application display some types of data, such as dates and numbers, as formatted strings. For example, the date “Sat 4 Jul 98” can be also represented as “July 4, 1998.” The number one thousand can be represented either as “1,000” or “1.000”, depending on the locale. There are different format specifiers for dates and numbers; check the reference documentation for the NSDateFormatter and NSNumberFormatter classes for details.
- **Color**. To change the color of text, either move the sliders to the right of the sample color or enter hexadecimal numbers in the field above the sliders. The color specification is RGB-based (that is, a specific mixture of red,

green, and blue). The top slider manipulates red saturation, the middle slider is for green, and the bottom slider is for blue. The three pairs of hexadecimal digits after the number sign in the field represent (left to right) saturation levels of red, green, and blue.

Representation of Relationships

Properties that are relationships (instead of attributes) have their own set of display components that you can use. Take the following list page as an example:

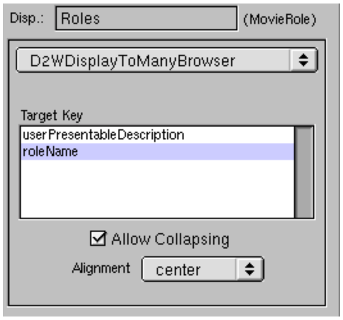


29 Movie(s) Display 5 items Page 2 of 6

	Category	Date Released	Title	Revenue	Poster Name	Trailer Name	Rated	
Edit	Drama	Nov 11,1941	Citizen Kane	100,000.00				Delete
Edit	Drama	Nov 11,1973	Cries and Whispers	100,000.00			R	Delete
Edit	Drama	Nov 11,1985	Desert Hearts	200,000.00			R	Delete
Edit	Drama	Jan 03,1982	Diner	200,000.00			R	Delete
Edit	Drama	Jan 03,1980	Diva	1,000,000.00			R	Delete

There are four relationships on this page. Two are to-one relationships (Studio and Plot Summary) and two are to-many relationships (Directors and Roles). By default, all to-many relationships are displayed using DisplayToManyFault, and to-one relationships use DisplayToOneFault. “Fault” indicates that the records in the relationship aren’t displayed until they are asked for; that is, until the user clicks Inspect. When you click Inspect, a list page appears, showing all the records in the relationship (such as all roles in the movie).

You can change the display component for the relationship to get a different presentation. Consider the Roles relationship in the Movie-List page example above. Using your browser, navigate to the list page for the Movie entity. Move the roles property to the Show list using the Web Assistant and select the DisplayToManyBrowser component from the WOComponent pop-up list. The right side of the Web Assistant should look similar to the following example:



Disp.: Roles (MovieRole)

D2WDisplayToManyBrowser

Target Key

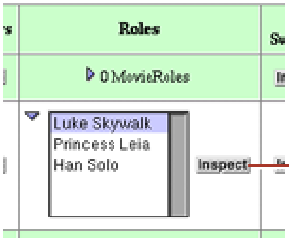
userPresentableDescription
roleName

☑ Allow Collapsing

Alignment center

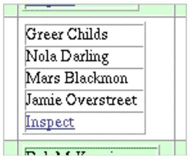
In addition to the Alignment pop-up list, the WOComponent group includes two controls specific to the display of relationships. The items in the Target Keys browser are selected attributes of the destination entity; these “target keys” refer to a string identifying a to-many relationship. In this case the Movie Roles entity has one target key, **roleName**. In addition, Direct to Web provides a default key called **userPresentableDescription**, which is usually a combination of the relationship’s keys, if there are multiple keys.

The Allow Collapsing checkbox, when checked, causes the relationship initially to be presented as a disclosure triangle followed by a number and the plural form of the display name for the destination entity (for example, “6 Movie Roles”). When the user clicks the triangle, the table cell expands to display the items in the form appropriate to the display component; in this case, a browser:



Click to display records in their own list page.

To get a better sense of the control you have over the presentation of relationships, set the display component for the Movie Roles relationship to DisplayToMany and uncheck the Allow Collapsing checkbox. When you update your browser, a cell in the Roles table should look similar to this:


Greer Childs
Nola Darling
Mars Blackmon
Jamie Overstreet

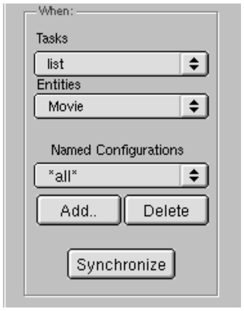
To-one relationships by their nature offer fewer possibilities for presentation.The DisplayToOneFault component presents an Inspect button which, when clicked, displays the relationship record in an inspect page. The other choice of component, DisplayToOne, displays the target key for the single destination record as a hyperlink which, when clicked, brings you to the same inspect page.

A note of caution: The type of display component appropriate to a relationship depends on the likely number of records in that relationship. For example, the Studio entity has a Movies to-many relationship; if some studios have produced hundreds of movies, it might make more sense to use DisplayToManyFault (that is, the Inspect button) rather than display the titles of all those movies in a cell in the table.

To find out more about a display component for a relationship, click the Info button in the Web Assistant after selecting the component.

Web Assistant Expert Mode

Expert mode is similar to standard mode, except that it allows you to make changes to any page in your application whether it is currently displayed in your browser or not. If you click the Expert mode button at the bottom of the Web Assistant, the window expands the following interface:

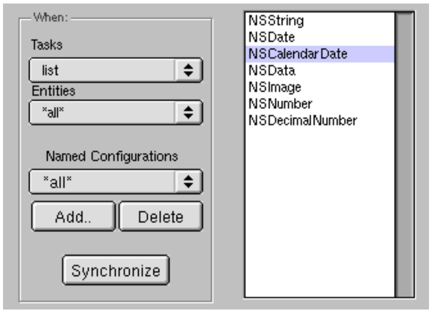

When:
Tasks: list
Entities: Movie
Named Configurations: *all*

The tasks pop-up list shows the types of pages available in Direct to Web. The Entities pop-up list shows the entities in the model.

Choosing a Page to Customize

To customize any page in your application, simply select the type of page and the entity. The figure above shows an example of choosing the inspect page for the Talent entity, making the Web Assistant focus on this page rather than the page currently showing in the browser.

If you select "*all*" under Tasks, any changes you make affect all customizable pages for the selected entity. If you select "*all*" under Entities, you'll see a list of data types that exist in the application, as shown in the following figure.


Tasks: list
Entities: *all*
Named Configurations: *all*

Any changes you make affect all occurrences of that type. For example, the figure shows NSCalendarDate selected. You can specify a formatter, and pick a component to use anywhere in the application that an NSCalendarDate object is displayed.

If you click Synchronize, the task and entity for the current browser page are selected in the Web Assistant.

You can also select the Page display of the Web Assistant while in Expert mode and change the underlying component, color, and border thickness of whatever page for whatever entity you select in the Tasks and Entities browsers.

Named Configurations

Once you have customized a page, you can capture the settings in a *named configuration*. Named configurations are used when you need more than one page for a particular task and entity. Consider a page that lists movies for a video rental store. A customer would want to see the names of the movies and plot summaries. A store clerk would want to see how many copies are available for rental and how long they can be rented. In addition, the customer should not be able to edit any information, while the store clerk might be able to edit some properties. To set up such a system, you create two named configurations for listing movies: one for the customer and the other for the store clerk.

Named configurations can only be displayed programmatically; the Web Assistant can edit named configurations but can't display the changes in your browser.

To save the settings of the current task and entity in a named configuration:

1. Click Add.

   A panel appears with a text field containing a default name for the configuration (the page name followed by the entity name).

2. Enter a new name for the configuration if you choose.

3. Click Ok.

To edit a named configuration, select it from the Named Configurations pop-up list. You can now change settings on the Properties and Page displays.

Note: When you edit a named configuration, the changes are not reflected in your browser. Named configurations can only be displayed programmatically.

To delete a named configuration, select it from the Named Configurations pop-up list and click Delete.

Generating Components

When you have worked with the Web Assistant and customized your pages to your liking, you may still want to add more features to your application. To do so, you can "freeze" a page; that is, save it as a WebObjects component. When you do this, the component becomes part of your project and is no longer created "on the fly" by Direct to Web. This has several advantages:

- You have complete control over the visual appearance of the page. You can add any static or dynamic HTML elements you like, using a tool such as WebObjects Builder.

- You can add functionality to the page by editing the component's Java code, as well as by editing the bindings of the page's dynamic elements.

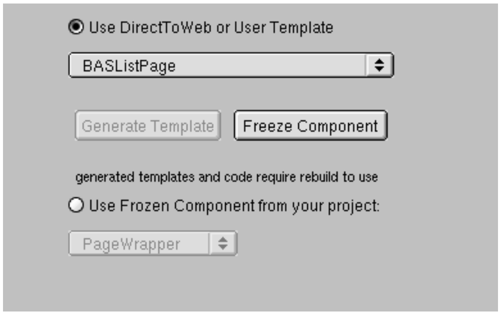
- Your application's performance improves because Direct to Web doesn't have to go through the process of creating the page "on the fly."

The main disadvantage of generating components is that you lose the ability to modify settings with the Web Assistant since the entity, property settings, and page configuration are stored directly in the generated component. To modify the page, you must edit the component or its corresponding **.java** file. Therefore, you should try to get your settings as close as possible to what you want before generating the component.

To generate a component:

1. Click the Expert Mode button at the bottom of the Web Assistant to enter Expert mode.

2. Click the Generate tab at the top of the Web Assistant.



● Use DirectToWeb or User Template

BASListPage

generated templates and code require rebuild to use

○ Use Frozen Component from your project:

PageWrapper

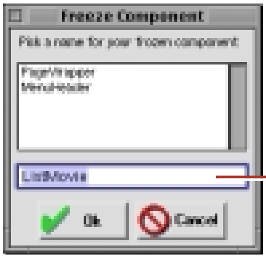
3. Select the task and entity corresponding to the page you want to generate.

   You can't select "*all*" to generate multiple components. You must generate the components one at a time.

4. In the Advanced Options group of controls, make sure the "Use DirectToWeb or User Template" radio button is selected.

5. Click Freeze Component.

The Freeze Component window appears. It contains a text field with a default name for your page (the page name followed by the entity name). You can edit the name if you choose.



Freeze Component

Pick a name for your frozen component:

PageWrapper
MenuHeader

ListMovie

Edit this field to change the component's name.

6. Click the Ok button.

   Direct to Web generates a component (with extension **.wo**) and a corresponding **.java** file and adds them to your project. You may have to wait a few moments for this process to complete. Your settings are automatically saved.

7. Rebuild your project.

If you decide not to use the frozen component and have Direct to Web build the page "on the fly," select the "Use DirectToWeb or User Template" radio button.

When you generate a page and click Update, the browser's current page doesn't reflect the changes. To use the new component, you must rebuild the application, relaunch it, and then navigate to a new instance of the page. For example, if the current page is a Movie query page, and you use the Web Assistant to freeze it, you must rebuild the project with the frozen component, then launch the application and navigate to a new instance of Movie query (by clicking Build Query); the new instance uses the frozen component.

The generated component is like any other WebObjects component. You can edit your component graphically using WebObjects Builder. You can also examine the HTML and bindings (**.wod** file) of the new component in Project Builder.

Direct to Web also generates Java code for your component, which you can modify appropriate to your needs. Each component implements an interface that is appropriate to the page: QueryPageInterface, ListPageInterface, InspectPageInterface, and EditPageInterface. For example, the **QueryMovieRole.java** file shown below implements the QueryPageInterface. For example, it contains an action method called **queryClicked** that returns a component when the Query DB button is clicked. (Note that the component's submit button is bound to **queryClicked** in **QueryMovieRole.wod**.)

User Templates

Sometimes you need to change the appearance of a Direct to Web page without freezing the component. You might want to change all of the pages for a particular task (list pages for example) without freezing components for every entity. Or you might want to use the Web Assistant to fine-tune pages having your custom appearance.

Direct to Web allows you to generate, modify, and use templates. Templates are WebObjects components that Direct to Web can use to generate pages (list pages for example) for any entity. Direct to Web provides a number of prebuilt templates from which you generate *user templates*. User templates appear together with the prebuilt templates in the Web Assistant and you can apply them to pages in your project.

The advantages of using templates are:

- You can use a template for any entity.
- You can modify the properties and page appearance with the Web Assistant.
- Since the template is a WebObjects component, you have control over its visual appearance. You can add any static or dynamic HTML elements you like, using a tool such as WebObjects Builder.
- You can add functionality to the template by editing the component's Java code, as well as by editing the bindings of the component's dynamic elements.

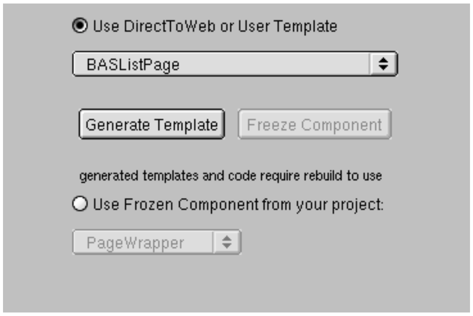
Templates are slower than frozen components since Direct to Web generates the pages that the user sees in the browser.

Generating a Templates

To generate a template:

1. Click the Expert Mode button at the bottom of the Web Assistant to enter Expert mode.

2. Click the Generate tab at the top of the Web Assistant.



● Use DirectToWeb or User Template

BASListPage

generated templates and code require rebuild to use

○ Use Frozen Component from your project:

PageWrapper

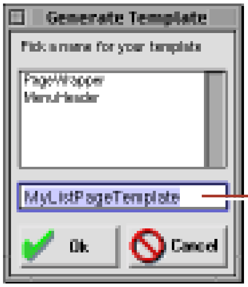
3. Select the task corresponding to the page you want to generate. In the Select "*all*" in the Entities pop-up list.

   You must select "*all*" for the entities because the template is independent of the entity. You cannot select "*all*" for the task to generate multiple templates. You must generate the templates one at a time.

4. Make sure the "Use DirectToWeb or User Template" radio button is selected.

5. Click Generate Template.

   The Generate Template window appears. It contains a text field with a default name for your template. You can edit the name if you choose.



Generate Template

Pick a name for your template

PageWrapper
MenuHeader

MyListPageTemplate

Edit this field to change the template's name.

6. Click the Ok button.

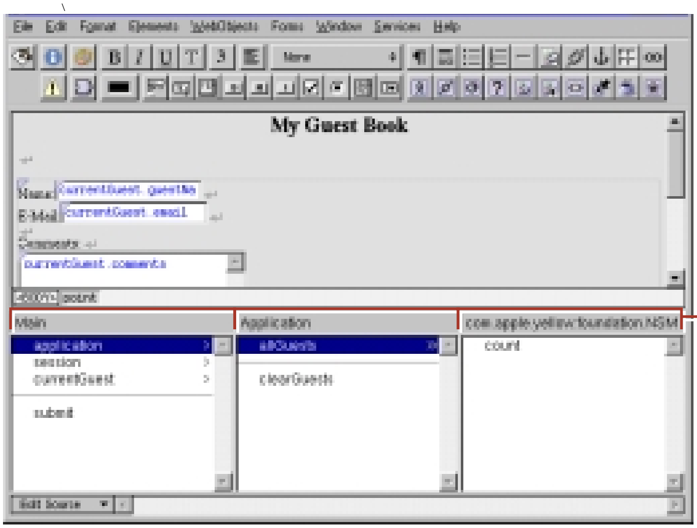
Direct to Web copies a component (with extension **.wo**) and a corresponding **.java** file from a predefined template and adds them to your project. You may have to wait a few moments for this process to complete. Your settings are automatically saved.

7. Rebuild your project.

After you generate the template and rebuild your project, you can use the Web Assistant to apply the template to a Direct to Web page. See "Customizing Pages" (page 123).

Erratum

The WebObjects Builder editing view layout has changed and longer matches the screenshots in this book. The object browser now has headings over the columns. Each heading specifies the class of the object selected in the previous column; this class defines the keys and actions listed in the column. In the example screenshot below, the user has selected the **application** key. The heading above the second column indicates that the **application** key refers to an object of the Application class that defines the **allGuests** key and the **clearGuests** action.



These headings have been added.