DIRECT TO WEB

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: Patrice Gautier, Eric Bailey, Craig Federighi, and Kelly Toshach

Table of Contents

Direct To Web 5

Chapter 1

Direct To Web

Direct to Web is a technology that provides a quick and easy method of creating 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.

- To change the way the pages are presented, you can use the WebAssistant, which is a Java applet that runs in your web browser. For each page in your application, you can specify which properties are shown, how they 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 further customization beyond what the WebAssistant 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.

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. See *WebObjects Tools and Techniques* for more information on using Project Builder and WebObjects Builder to develop WebObjects applications. 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.

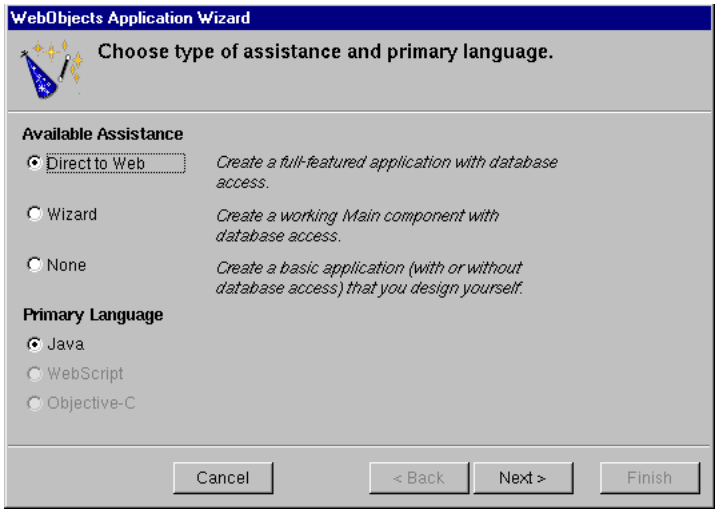
2. Choose Project► New.

3. In the New Project panel, specify the project path where you want to save the project.

   As with other Web Objects projects, during development you typically save your project in the WebObjects subdirectory of your web server's document root.

4. Click OK.

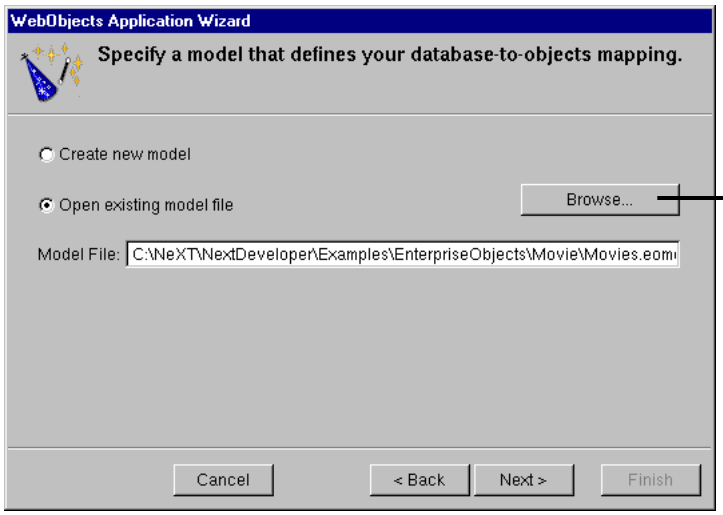
   The WebObjects application wizard screen appears.



5. Under Available Assistance, choose Direct to Web.

   Note: When you create a Direct to Web project, Java is the only available language.

6. Click Next.



7. Choose "Open existing model file."

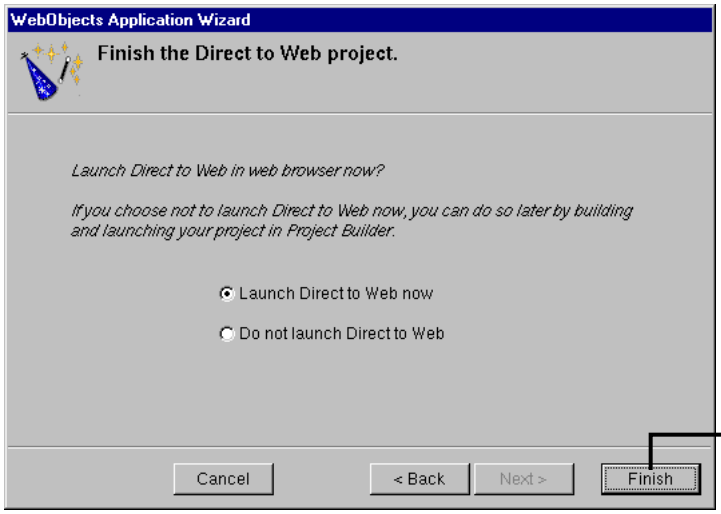
   You can also create a new model file. If you choose "Create new model," you are led through a series of screens that prompt you to create a new model. For more information about creating a new model file, see the chapter "Using EOModeler" in *Enterprise Objects Framework Developer's Guide*.

   Note: A complete and correct model file with all the right relationships defined is key to using Direct to Web.

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

9. Click Next.

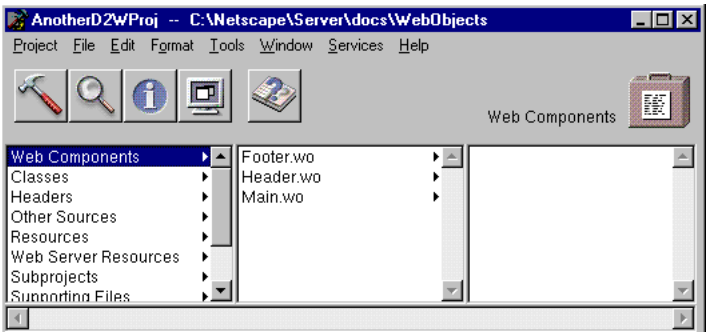
   The next screen asks you whether you want to launch your application now.



If you click “Launch Direct to Web now”, the application launches and, by default, your web browser displays the Direct to Web login page. If you don’t launch it now, you can launch it later from Project Builder, in the same way you would launch any other project. “Using Your Direct to Web Application” (page 12) describes what you see when you launch your application.

The Structure of a Direct to Web Project

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



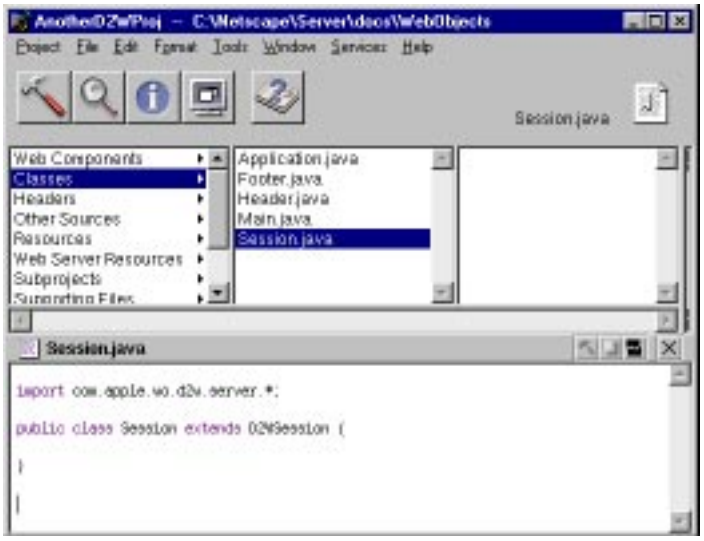
- **Main.wo** is the main component, representing the login page of the application.

- **Header.wo** is a reusable component that is inserted at the beginning of each page in the application (other than Main). It contains some control buttons that are displayed at the top of each page. If you choose, you can add text or other elements to the Header component.

- **Footer.wo** is a reusable component that is inserted at the end of each page in the application. By default, it is blank; you can add elements to it if you choose.

As you run your application, Direct to Web creates additional pages, using information in your model file and settings specified in the WebAssistant. 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 have the option of saving any page as a component. 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 27) 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. See "Modifying Your Application's Code" (page 32) for more information on the Direct to Web API.



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 is used to store the preferences you have changed using the WebAssistant (you should never need to edit this file directly).

Note: If your model references entities in another model, you must add the other model to your project manually. It doesn't get included automatically.

Using Your Direct to Web Application

To launch your application from Project Builder:

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

For best performance when running a Direct to Web Application, turn on caching by using the `-c` command line option. To do so:

1. Click  to bring up the Launch Options panel.
2. Select Environment and Command-Line Arguments from the pop-up menus.
3. Click Add to create a new command line and type `-c`.

The Login Page

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

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 two submit buttons. To continue with the application, click one of the submit buttons. (You don't need to enter a name and password, because the default application provides no password-checking logic.) They both take you to the application's default first page. If you use the Login button, you won't have access to the WebAssistant.

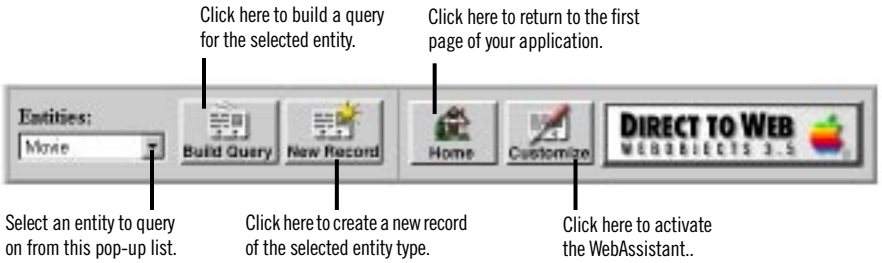
You can modify this page to provide any behavior or appearance you like. For example, you can add your own password-checking logic if necessary. See "Modifying Your Application's Code" (page 32) for more information.

Besides the login page, there are four types of pages in a Direct to Web application:

- A *query page* that allows the user to construct a query for a particular entity; see "Query Pages" (page 14).
- A *list page* that displays one or more records of a particular entity in tabular form; see "List Pages" (page 16). The result of a query is always a list page.
- An *inspect page* that displays a single record of a given entity; see "Inspect and Edit Pages" (page 18).
- 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; see "Inspect and Edit Pages" (page 18).

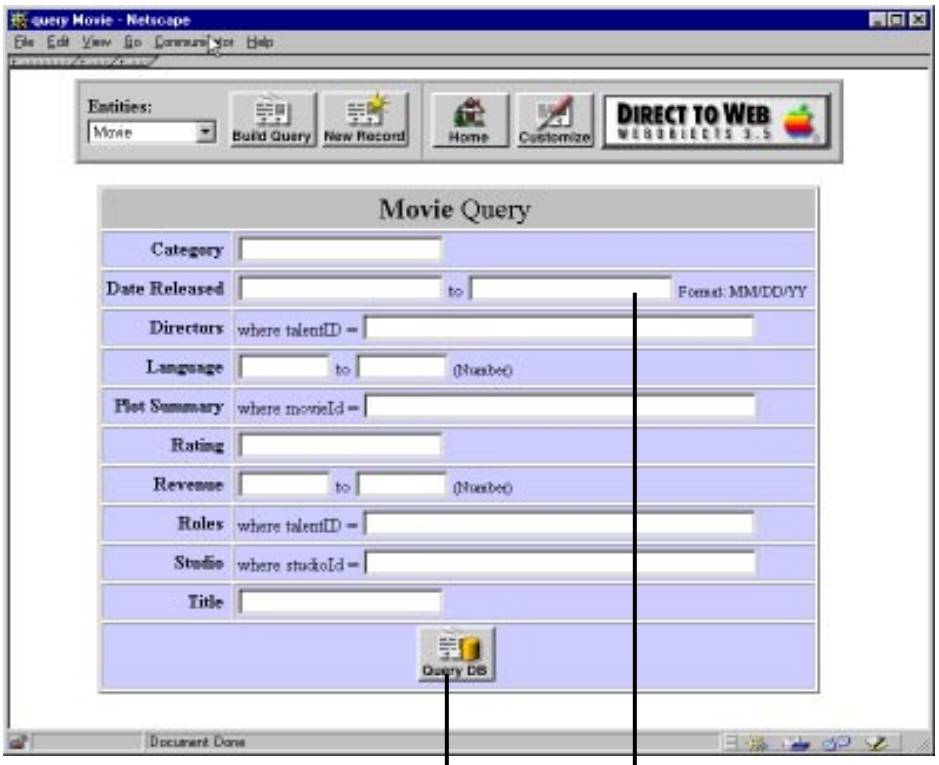
When you click one of the login buttons, by default a query page for the first entity in your model is displayed.

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



Query Pages

The query page for an entity is shown as a two-column table.



The first column in the table lists the entity's properties. The second column contains one or more text fields that let you enter values to query on each property.

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, Title is an attribute and Studio is a relationship. You can use the WebAssistant 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.

Note that properties are represented in different 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 WebAssistant. 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 "The Direct to Web Components" (page 22) for more information on the different ways of representing properties in your application's pages.

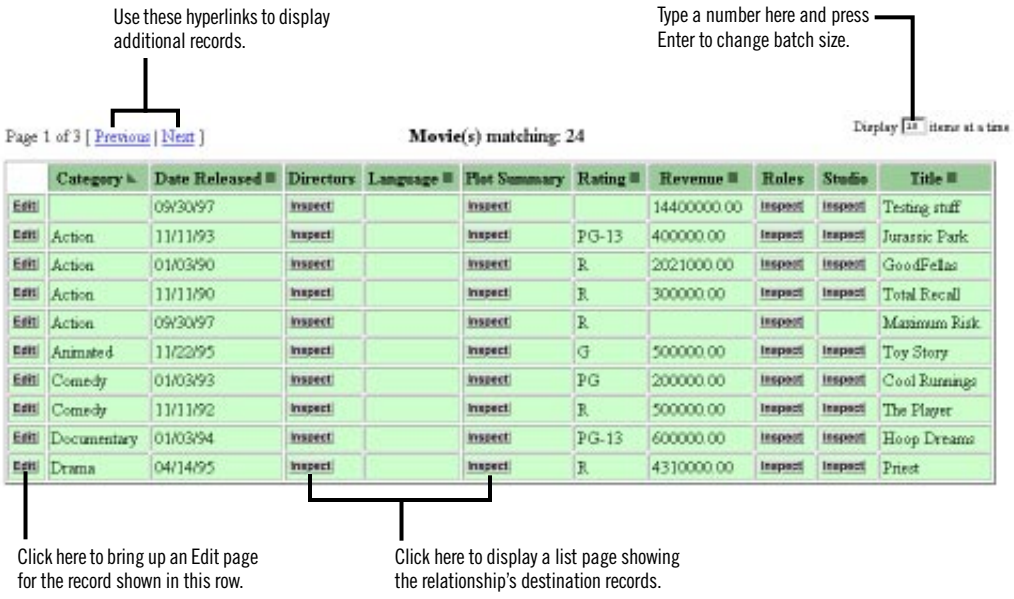
In the Movie query, to get a list of all dramas released in the 1990's, you would:

1. Enter `Drama` in the Category field.
2. Enter `1/1/90` and `12/31/99` in the Date Released fields.
3. Click Query DB.

   The results are displayed in a list page; see "List Pages" (page 16).

List Pages

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 satisyfing a to-many relationship in another list or inspect page.



	Category	Date Released	Directors	Language	Plot Summary	Rating	Revenue	Roles	Studio	Title
Edit		09/30/97	inspect		inspect		14400000.00	inspect	inspect	Testing stuff
Edit	Action	11/11/93	inspect		inspect	PG-13	400000.00	inspect	inspect	Jurassic Park
Edit	Action	01/03/90	inspect		inspect	R	2021000.00	inspect	inspect	GoodFellas
Edit	Action	11/11/90	inspect		inspect	R	300000.00	inspect	inspect	Total Recall
Edit	Action	09/30/97	inspect		inspect	R		inspect		Maximum Risk
Edit	Animated	11/22/95	inspect		inspect	G	500000.00	inspect	inspect	Toy Story
Edit	Comedy	01/03/93	inspect		inspect	PG	200000.00	inspect	inspect	Cool Runnings
Edit	Comedy	11/11/92	inspect		inspect	R	500000.00	inspect	inspect	The Player
Edit	Documentary	01/03/94	inspect		inspect	PG-13	600000.00	inspect	inspect	Hoop Dreams
Edit	Drama	04/14/95	inspect		inspect	R	4310000.00	inspect	inspect	Priest

Each row in the table represents a record. By default, up to ten records are shown in a page. To display additional records, use the Next and Previous hyperlinks above the table. To change the batch size to a number other than ten, type it in the text field and press Enter.

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 WebAssistant; see "Customizing Your Application With WebAssistant" (page 19).

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.

Note: 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. When you click the Inspect button:

- 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 18).

- 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.

MovieRole(s) matching: 6

	Movie	Role Name	Talent
Edit	Inspect	James Conway	Inspect
Edit	Inspect	Joe Buddha	Inspect
Edit	Inspect	Tommy DeVito	Inspect
Edit	Inspect	Karen Hill	Inspect
Edit	Inspect	Paul Cicero	Inspect
Edit	Inspect	Henry Hill	Inspect

You can use the WebAssistant to display the related records directly in the table instead of with an Inspect button; see "Customizing Your Application With WebAssistant" (page 19).

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:

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.

Note: If your model specifies a particular entity as read-only, you won't be able to edit it in a Direct to Web application.

An edit page looks like this:

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.

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, select the entity in the Entities pop-up list and click Build Query.

Customizing Your Application With WebAssistant

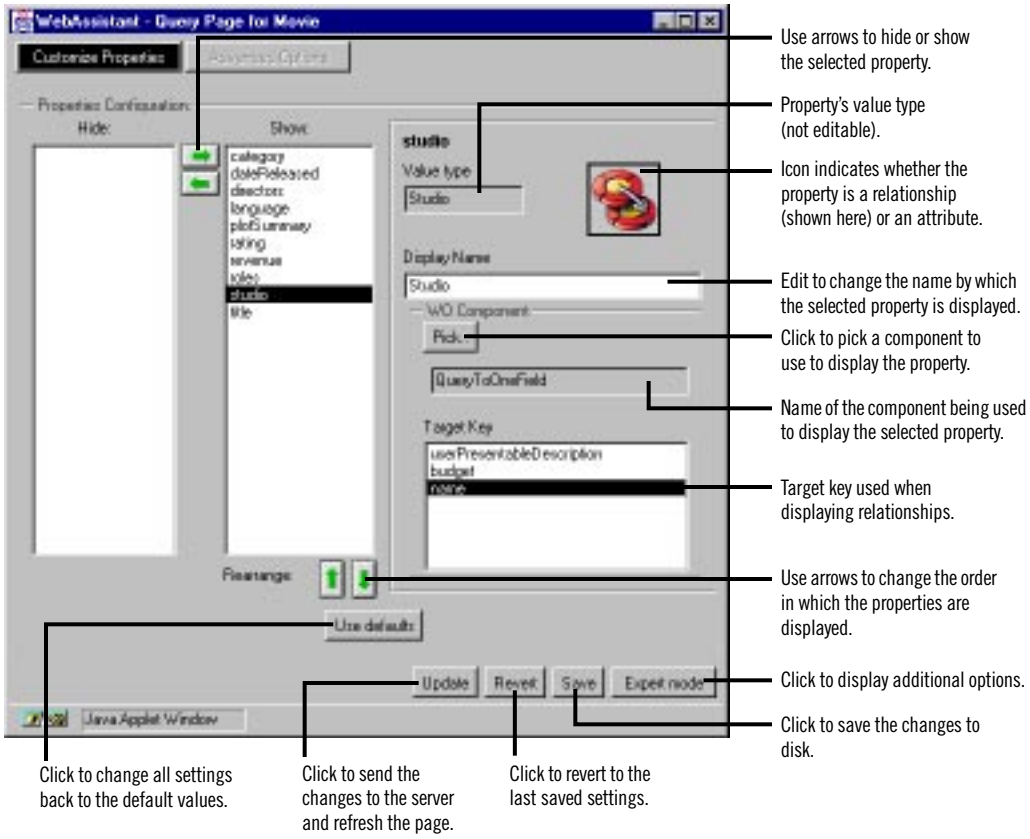
The WebAssistant allows you to customize each page of your application. You can specify:

- 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 WebAssistant, click Customize in the Direct to Web header. A Java applet window appears showing the WebAssistant. This window lets you customize the current page in your application. (Using the Expert mode, you can customize any page in the application, regardless of whether it is currently displayed; see "WebAssistant Expert Mode" (page 25).

When you customize a page using the WebAssistant, 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. 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. Or, you can use the "*all*" setting in Expert mode to change all four pages at once; see "WebAssistant Expert Mode" (page 25).



All the entity's properties (attributes and relationships) are listed in the Show column, in the order in which they are displayed in the page. Properties in the Hide column are not displayed in the page. For each property, you can:

- Move it to the Hide column it by double-clicking it or by selecting it and clicking the left arrow. Likewise, if a property is hidden, you can show it by double-clicking it or by selecting it and clicking the right arrow.

- Move it up or down in the list by clicking the Rearrange up and down arrows. This changes the order of appearance of the properties in the page.
- Change its name by editing the Display Name field.

  Note: This change only affects the way the entity is labeled in the page, and has no effect on the actual entity name.

The "Value type" field shows the data type of the selected property. You can't edit this field.

The icon in the upper right of the window shows whether the selected property is an attribute  or a relationship  .

The WOComponent box shows the name of the component that is being used to display the property in the page. In many cases (when the Pick button is enabled), you can choose a different component to display the property. In the above example, the **studio** entity uses the QueryToOneField component. If you click the Pick button, a new applet window appears, allowing you to select a different component.

Choose a component to display the selected property, then click OK.

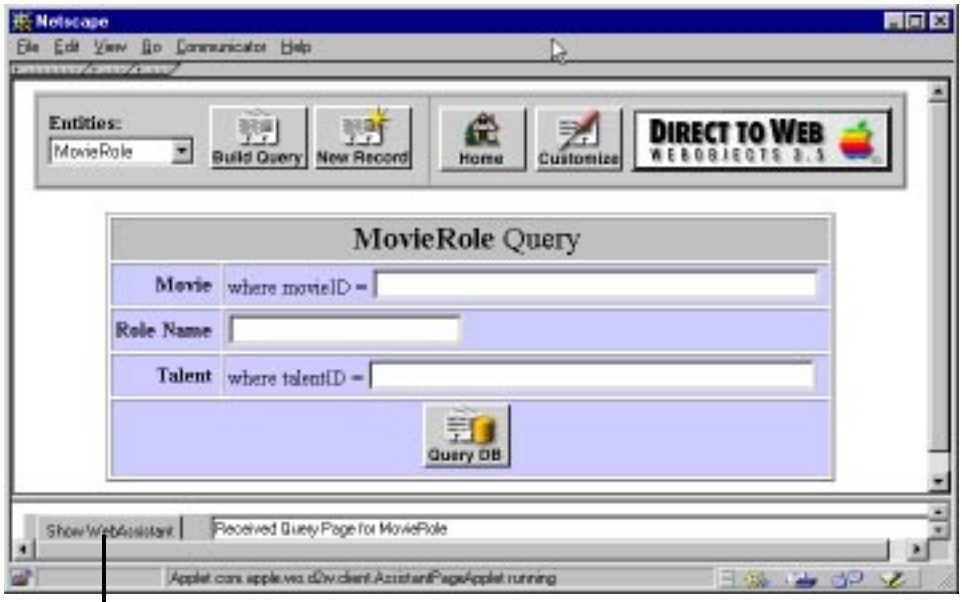
These are reusable components found in the Direct to Web Framework; see "The Direct to Web Components" (page 22) for more information on the available components.

When you've made your changes to the page, you can use the buttons at the bottom of the WebAssistant to apply them:

- *Update:* Sends your changes to the server and causes the page to be refreshed.
- *Revert:* Causes the settings to revert to their last saved values.
- *Save:* Saves the changes to disk. You need to save your changes in order for them to persist beyond the current session.

- *Use Defaults:* reverts all settings to the values they had when the project was created.

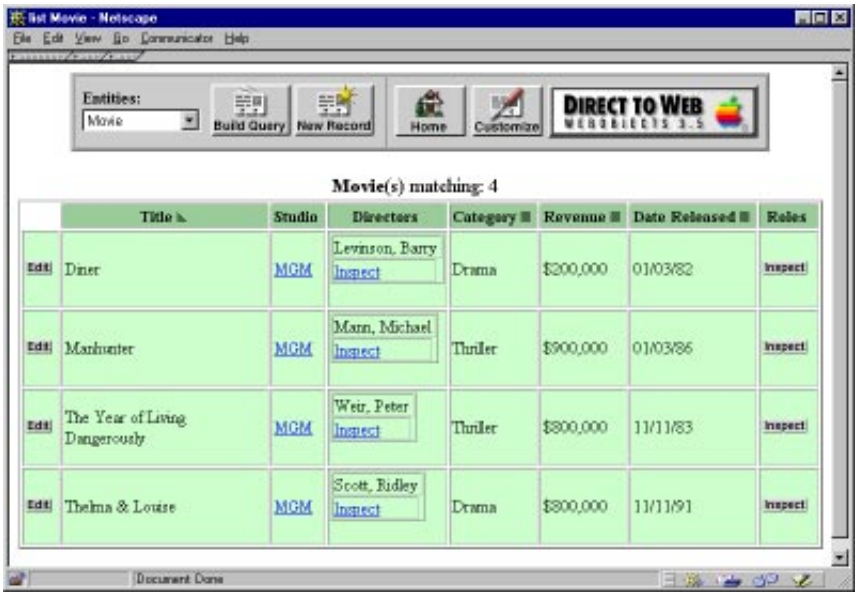
Note that when you have activated the WebAssistant, a frame appears at the bottom of each page in your application, containing a "Show WebAssistant" button and a status field. To bring the WebAssistant to the front, click the Show WebAssistant button (rather than clicking Customize again).



The Direct to Web Components

The Direct to Web Framework consists of a number of components that are used to generate the pages you see in your application. Each property in a query, list, edit, or inspect page is shown in a default way.

For example, the figure below shows a list page for the Movie entity. It illustrates some of the components that can be used to display properties.



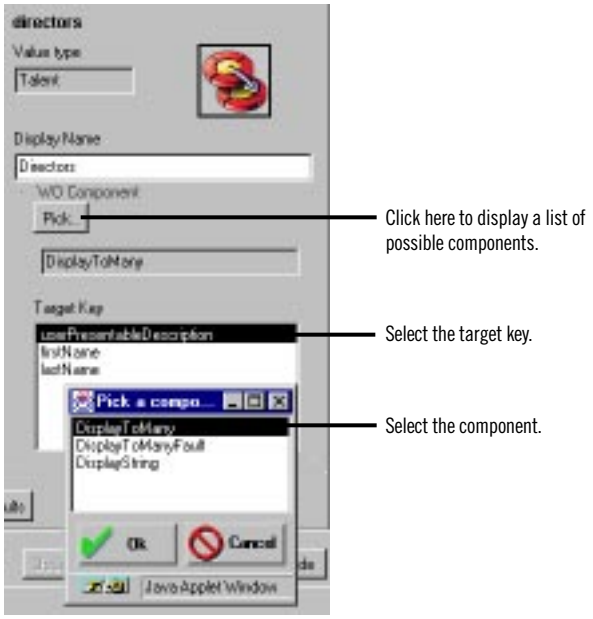
The Studio, Directors, and Roles properties are relationships:

- Roles is a to-many relationship that is displayed using the DisplayToManyFault component. “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 (that is, all roles in the movie).

  By default, all to-many relationships are displayed using DisplayToManyFault, and to-one relationships use DisplayToOneFault.

- The Directors relationship is also a to-many relationship (because a movie can possibly have more than one director). To display it as shown in the figure, you would select Directors in the WebAssistant and click Pick to display a list of components to use.

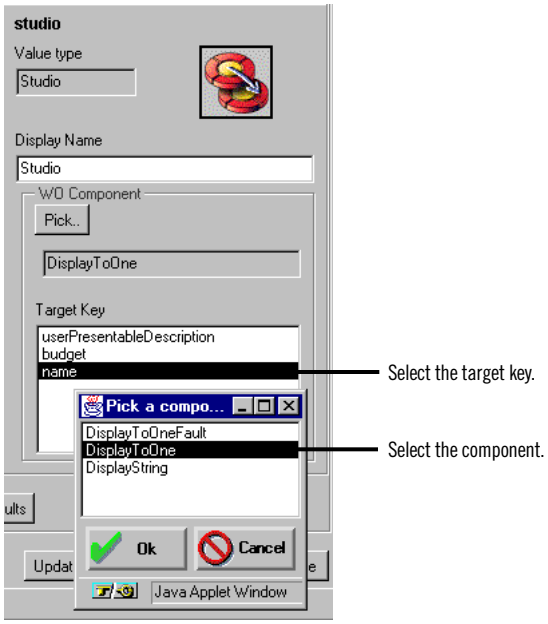
  Note: If Pick is disabled, there is no choice of components for the selected property.



In this example, the DisplayToMany component is used to display the directors. This component shows all the records in the relationship, identified by a target key, which you also select from a list. The target keys for Directors are **firstName** and **lastName**. In addition, Direct to Web provides a default key called **userPresentableDescription**, which is usually a combination of the relationship's keys. The example shown in the list page above uses this key to show both the last and first names of the directors.

Note: In the example shown, there happens to be only one director per movie.

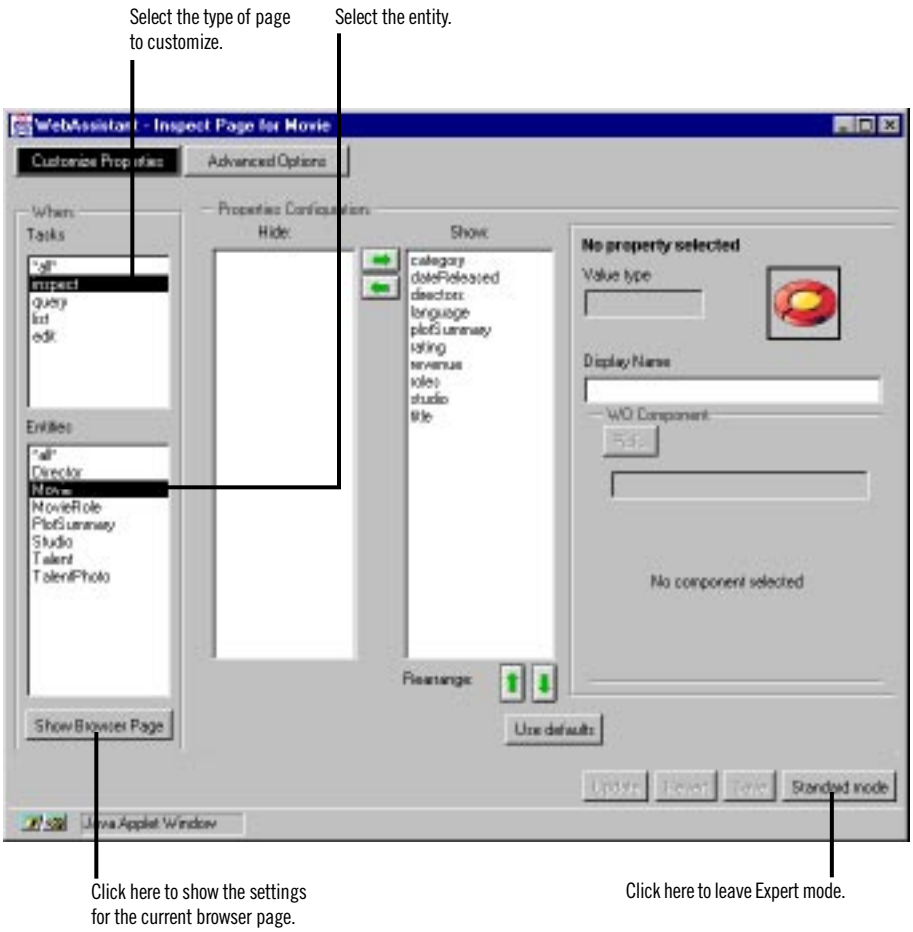
- The Studio relationship is a to-one relationship. Therefore, there is only one target record in the relationship. As shown in the figure below, it uses the DisplayToOne component with the **name** target key. Therefore, in the list page, the studio's name appears in the Studio column, as a hyperlink that you can click to bring up an inspect page for the studio.



WebAssistant Expert Mode

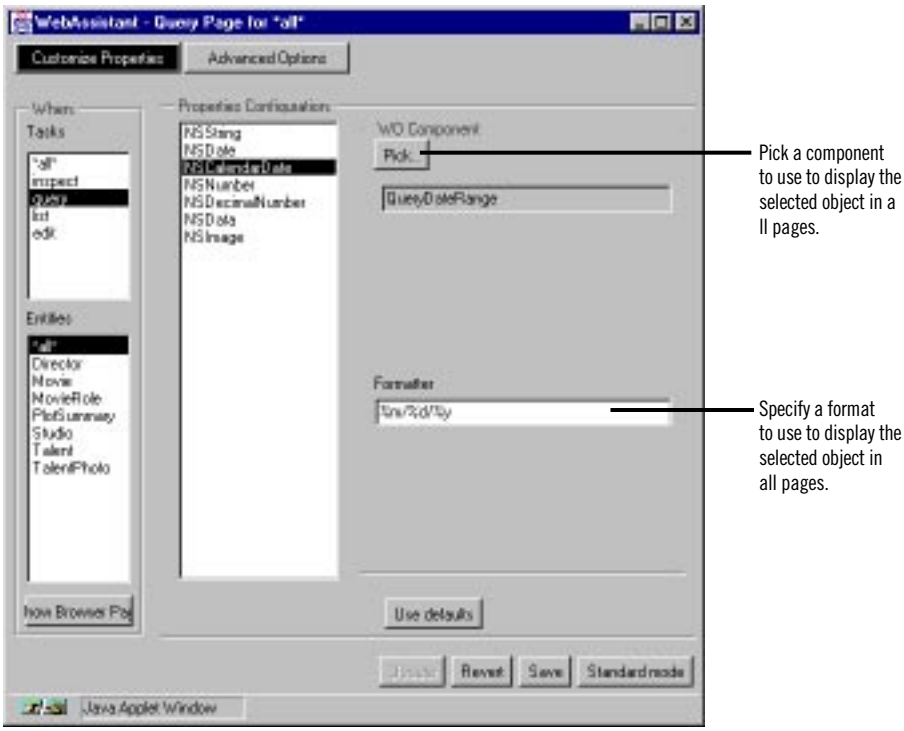
Expert mode is similar to standard mode, except that it allows you to make changes to any page in your application regardless of whether it is currently displayed in your browser. If you click the Expert mode button at the bottom of the WebAssistant, the window expands to show two additional lists:

- Tasks shows the four types of pages available in Direct to Web.
- Entities shows all the entities in the model.



To customize any page in your application, simply select the type of page and the entity. The figure above shows an example of customizing the inspect page for the Movie entity, regardless of what page is showing in the browser.

If you select "*all*" under Tasks, any changes you make will affect all four 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. 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 Show Browser Page, the task and entity for the current browser page are selected in the WebAssistant.

Generating Components

When you have worked with the WebAssistant 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:

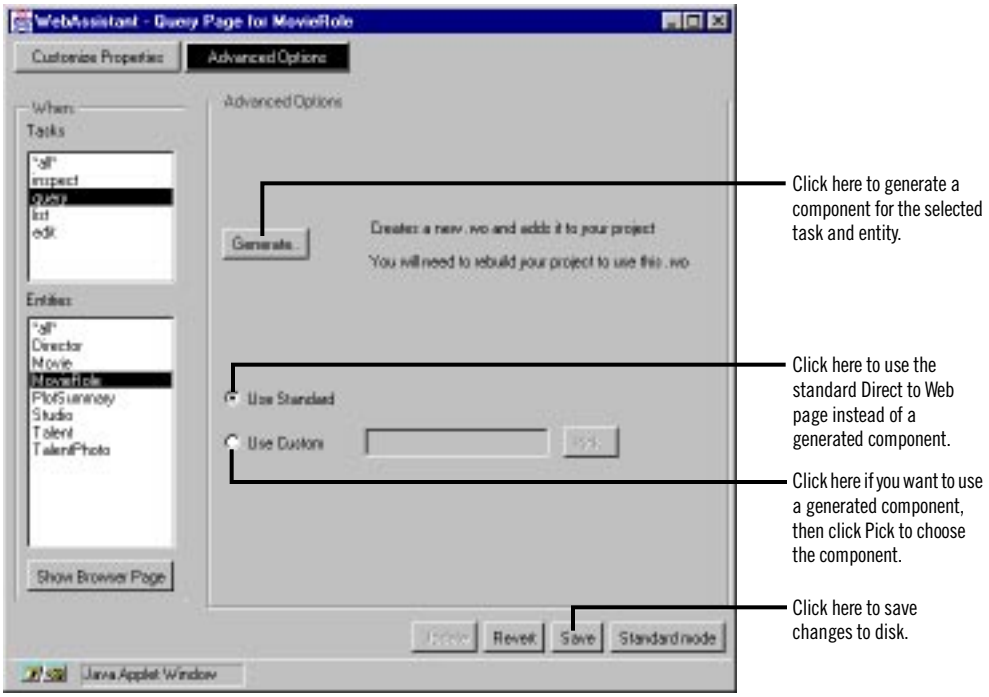
- You have complete control over the visual appearance of the page. You can add any static or dynamic HTML elements you like, using a tool such as WebObjects Builder.
- You can add functionality to the page by editing the component's Java code, as well as by editing the bindings of the page's dynamic 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 WebAssistant. Therefore, you should try to get your settings as close as possible to what you want before generating the component.

To save a page as a component:

1. Click Advanced Options at the top of the WebAssistant.

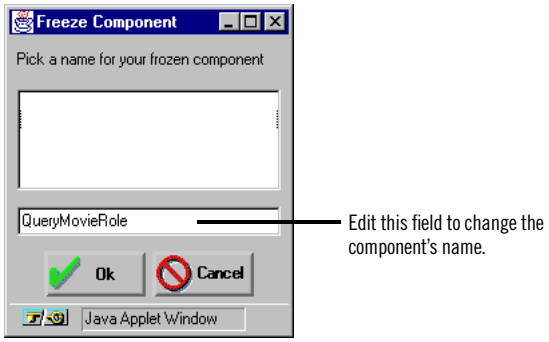


2. Select the task and entity corresponding to the page you want to generate.

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

3. Click Generate.

   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.



4. Click OK.

   Direct to Web generates a component and adds it to your project. (You may have to wait a few moments for this process to complete.)

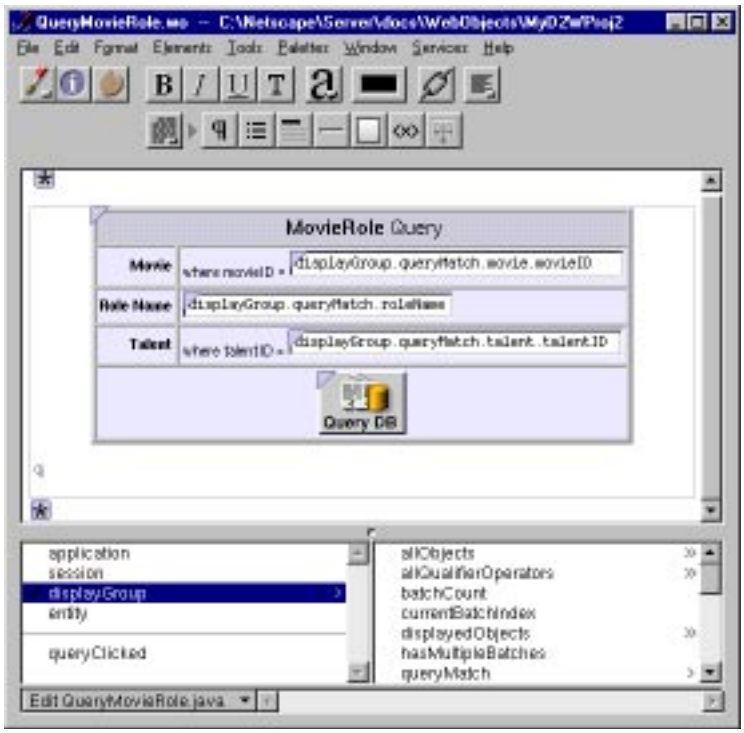
5. Click Save to save the changes to disk.

   If you don't click Save, the component will be generated, but Direct to Web won't automatically use the generated component the next time you run the project. To use the generated component, click Use Custom in the Advanced Options screen, then click Pick to choose the component.

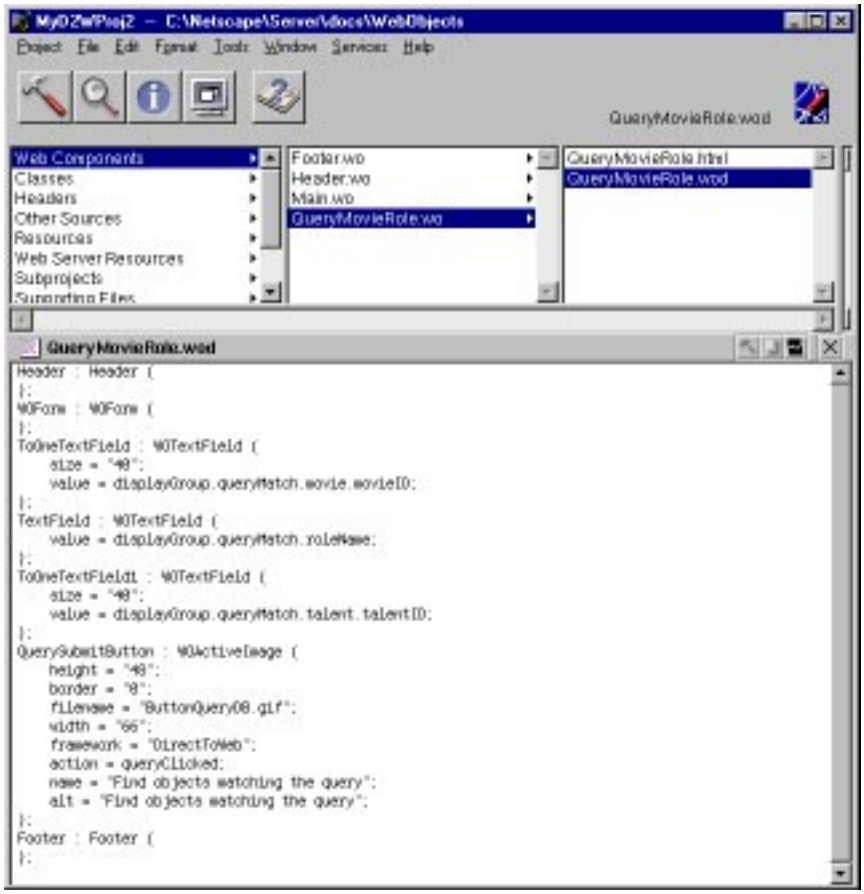
To "un-freeze" a component, click Use Standard.

Note: 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 navigate to a new instance of the page. For example, if the current page is a Movie query page, and you use the WebAssistant to freeze it, you must navigate to a new instance of Movie query (by clicking Build Query); the new instance uses the frozen component.

You can edit your component graphically using WebObjects Builder.

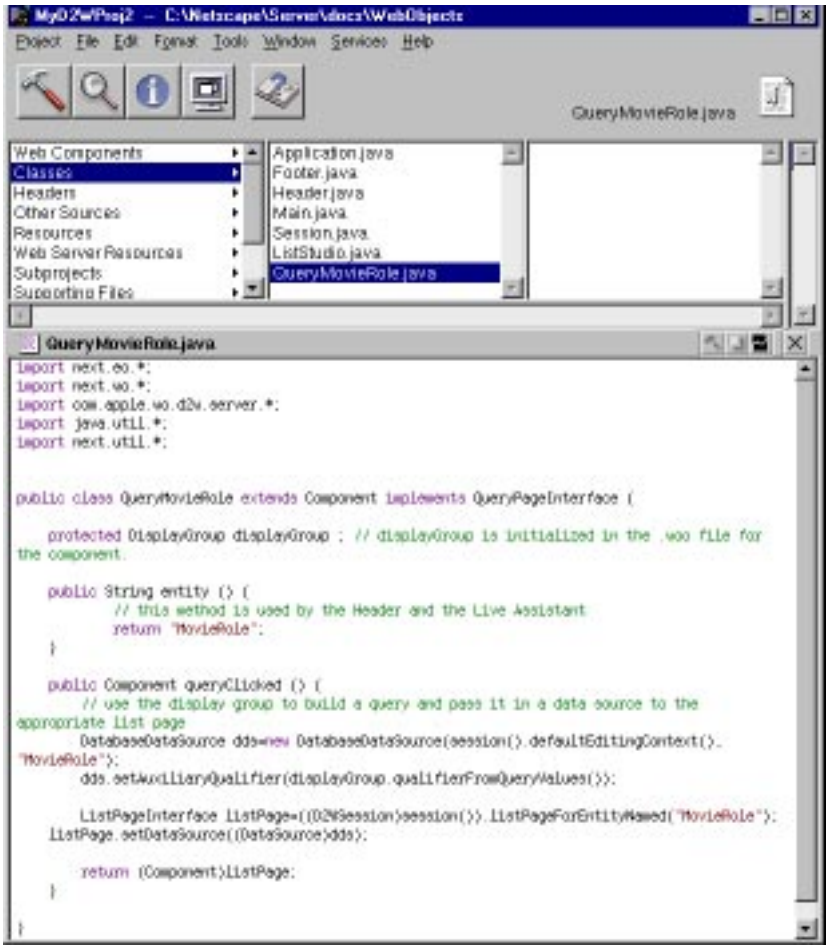


You can also examine the new component in Project Builder.



Direct to Web generates Java code for your component. 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**.)

You can modify the code in the generated routines if you choose.



To make use of the new component, you must rebuild and relaunch your application.

Modifying Your Application's Code

You can modify your application's code just as you would in any other WebObjects application. In addition, there is an API for you to use specifically in Direct to Web applications. This consists of a set of methods defined in the D2WSession class.

You can use the following methods to return a specific component of a given class, given its name:

```
QueryPageInterface queryPageForEntityNamed (String entity);
ListPageInterface listPageForEntityNamed (String entity);
EditPageInterface editPageForEntityNamed (String entity);
InspectPageInterface inspectPageForEntityNamed (String entity);
```

There are three other methods provided by Direct to Web that you may want to override:

```
public Component defaultPage();
public Component defaultPageWithAssistant();
public Component transitionToWebAssistant();
```

These methods have the following functionality:

- **defaultPage** returns the application's default page. Unless you override this method, the default page is the query page for the first entity in the model (alphabetically).
- **defaultPageWithAssistant** returns the default page with the Customize button in the header, so that the WebAssistant is available.
- **transitionToWebAssistant** returns a new page with two frames: the top frame contains the last page accessed, and the bottom frame contains the WebAssistant.

The following example shows how you can use these routines:

If you examine your application's Main component in WebObjects Builder, you'll notice that it contains a login form with text fields to enter a name and password, as well as two submit buttons.

The text fields are bound to the variables **username** and **password** (declared in **Main.java**). The Login button is bound to **session.defaultPage**, and LoginWithWebAssistant is bound to **session.defaultPageWithAssistant**. You can override these methods to change the behavior of the buttons. For example, you may wish to add password-checking code (the default implementation doesn't have any). You also might want to begin with an entity that's not the first one in the list.

The following code in **Session.java** overrides **session.defaultPage** to change the default page to the query page for the Movie entity. It also changes the application's default timeout value.

Session.java

```
import com.apple.wo.d2w.server.*;
import next.wo.*;

public class Session extends D2WSession {

    public Session()
    {
        setTimeOut(1800);       // set timeout to 30 minutes
    }

    public Component defaultPage()
    {
        return (Component)queryPageForEntityNamed("Movie");  // start with Movie query page
    }

}
```

Note: When you override **defaultPage**, this also takes care of the case where you use the LoginWithWebAssistant button, because the implementation of **defaultPageWithAssistant** calls **defaultPage**, which is now overridden by your code.