Direct To Web: Part 1

Direct To Web. What are some of the first words that come to your mind? Toy. Demo Ware. Not for real programers. Yeah I used to think that too. Then about five months ago I started working for Patrice Gautier (one of the original authors of DirectToWeb) at NetStruxr and everything changed. I have found DirectToWeb to be the most dynamic and flexible piece of software I have ever worked with, truly taking WebObjects design to another level. DirectToWeb is the best kept secret of WebObjects, which as we all can attest to is Apple's best kept secret.

So what is DirectToWeb? DirectToWeb originally started out as a way for non-technical developers to quickly produce an application and then mdify its look and feel at runtime through the use of a Java Applet named the Assistant. This probably sounds familiar. Instead what the team ended up developing was another layer onto of the WebObjects framework. A layer that abstracts and leverages WebObjects and EOF even further than you can possibly imagine. The only problem is that where as the rest of WebObjects has at least some documentation, DirectToWeb has extremely little. It is my hope that after three or four articles you will begin to glimpse the power beneath. In these articles we will start with the basics of DirectToWeb and work our way up. I thought I would first outline a few misconceptions that I had of DirectToWeb before I started working with it.

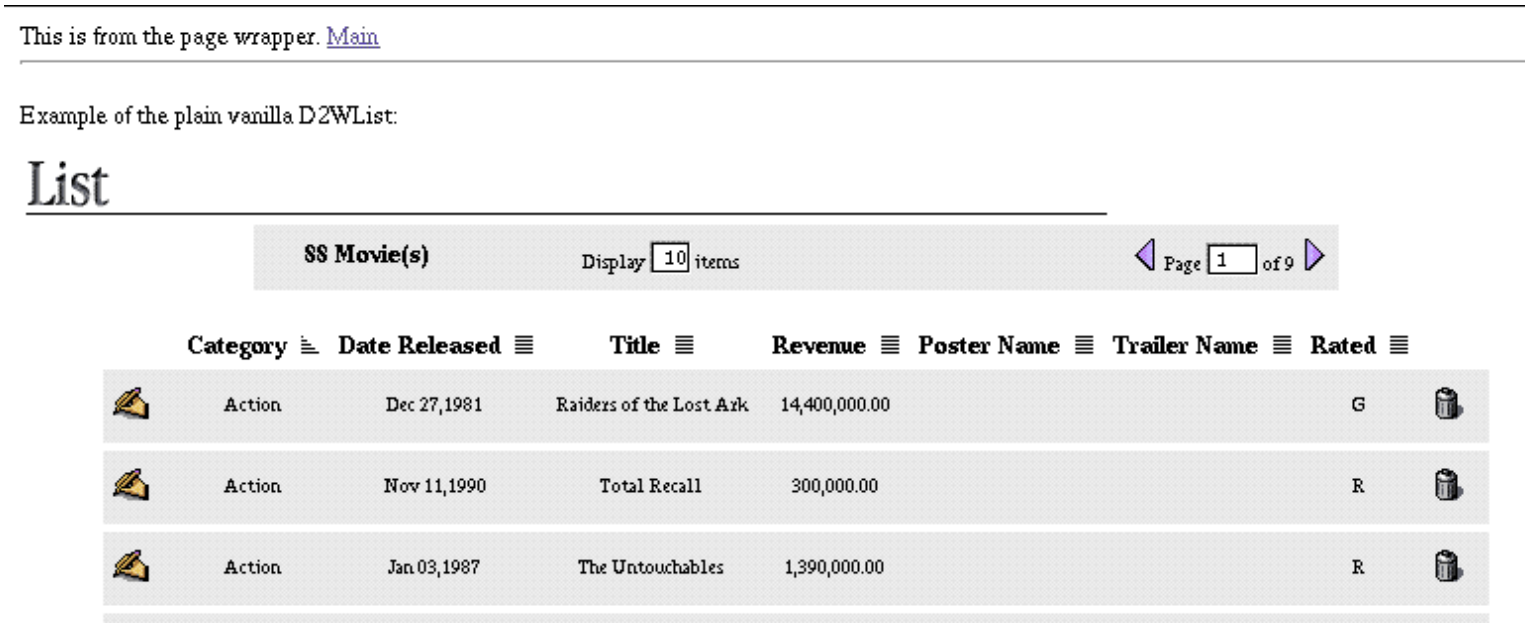
Common Misconceptions of DirectToWeb:

1) "Using DirectToWeb locks your whole app into being a 'DirectToWeb Application'." Not true at all. DirectToWeb does not change the way your app functions in the least. DirectToWeb is actually just a framework for creating dynamic pages on the fly based on a set of rules. To take advantage of DirectToWeb all that you have to do is add the DirectToWeb framework to your project and add a DirectToWeb rule file to your resources directory, just like an eomodel file (more on this in a bit).

2) "DirectToWeb will never look like my application. I've seen the three looks they have and that just won't work for me." Not true at all. By default DirectToWeb uses one of the built in looks or templates. But as you will soon see it is rather easy create your own template with your own site's look and then tell DirectToWeb to use that instead of one of the bundled templates. This is not the same as freezing a page (the act of turning a dynamic DirectToWeb page into a static regular WOComponent).

Simple DirectToWeb List Pages

Before you read any further you will probably want to download the sample project DirectToWebListExample which contains all of the examples discussed below. The eomodel used for this article is the standard movies eomodel, so if you have WebObjects installed on your machine you should be able to build the application and have it just work. Using list pages is a really simple way to get to know the DirectToWeb world. A list page does exactly what its name implies, it lists objects. In its simplest form, it takes an array of objects and an entity name and does the rest for you. This is shown in the WOComponent MyDirectToWebListPage1. You can see this page by clicking on the link "To List Page 1," from the main page of the example application Here all we have done is drop in a DirectToWebList component off of the DirectToWeb palette in WOBuilder and bound a list of Movie objects to the list binding and the string "Movie" to the entityName binding. This will spit out a plain vanilla DirectToWeb list page inside our standard WOComponent which will look like this:


This is from the page wrapper. Main

Example of the plain vanilla D2WList:

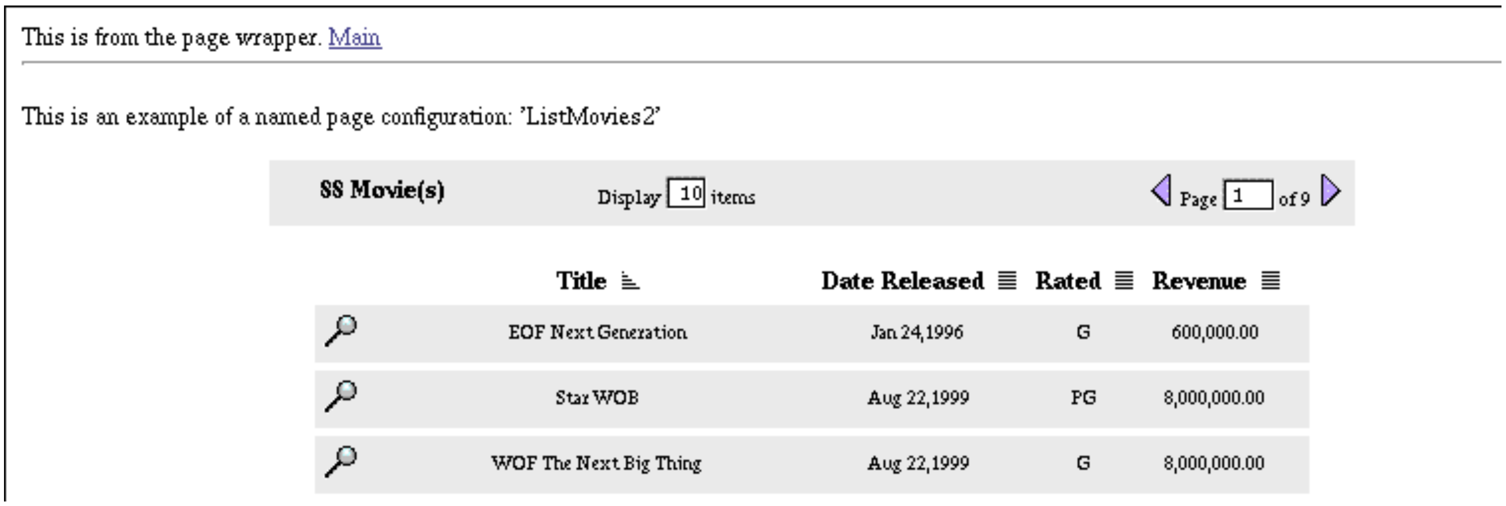
List

88 Movie(s) Display 10 items Page 1 of 9

	Category	Date Released	Title	Revenue	Poster Name	Trailer Name	Rated	
	Action	Dec 27,1981	Raiders of the Lost Ark	14,400,000.00			G	
	Action	Nov 11,1990	Total Recall	300,000.00			R	
	Action	Jan 03,1987	The Untouchables	1,390,000.00			R	

Notice that by default we have the ability to delete objects by clicking the trashcan on the right and to edit these objects by clicking on the edit button on the left. Also note that all of the attributes that are marked as class properties on the entity Movie are shown and that none of the relationships that are marked as class properties are shown. Now let's modify this behaviour to make the objects read only, get rid of the banner at the top and limit the display keys to title, dateReleased, rated and revenue. To do this we need to create a pageConfiguration or a set of rules that tells DirectToWeb how to render the page. The application that is used to edit and create rules is called RuleEditor and is located after the EOModeler icon in the drop down list under the WebObjects tab. RuleEditor edits and creates d2wmodel files which store all of the rules for a given application or framework. DirectToWeb also offers another means of editing rules called The Live Assistant. The Live Assistant is an applet that can be configured to startup when teh application is launched. The Live Assistant is a useful tool when getting started, but it hids alot of the power and flexibility that DirectToWeb offers. For this reason all the examples shown are done with the RuleEditor. At the moment DirectToWeb will only pick up model files that are named user.d2wmodel and d2w.d2wmodel, where typically the former resides in the resources directory of application projects and the later in the resources directory of framework projects, although nothing is stopping you from adding both a d2w and a user d2wmodel to a project.

To change the behaviour, we bind "ListMovies2" to the D2WList's binding pageConfiguration. The final product which is displayed in the WOComponent MyD2WListPage2. It should look something like this:


This is from the page wrapper. Main

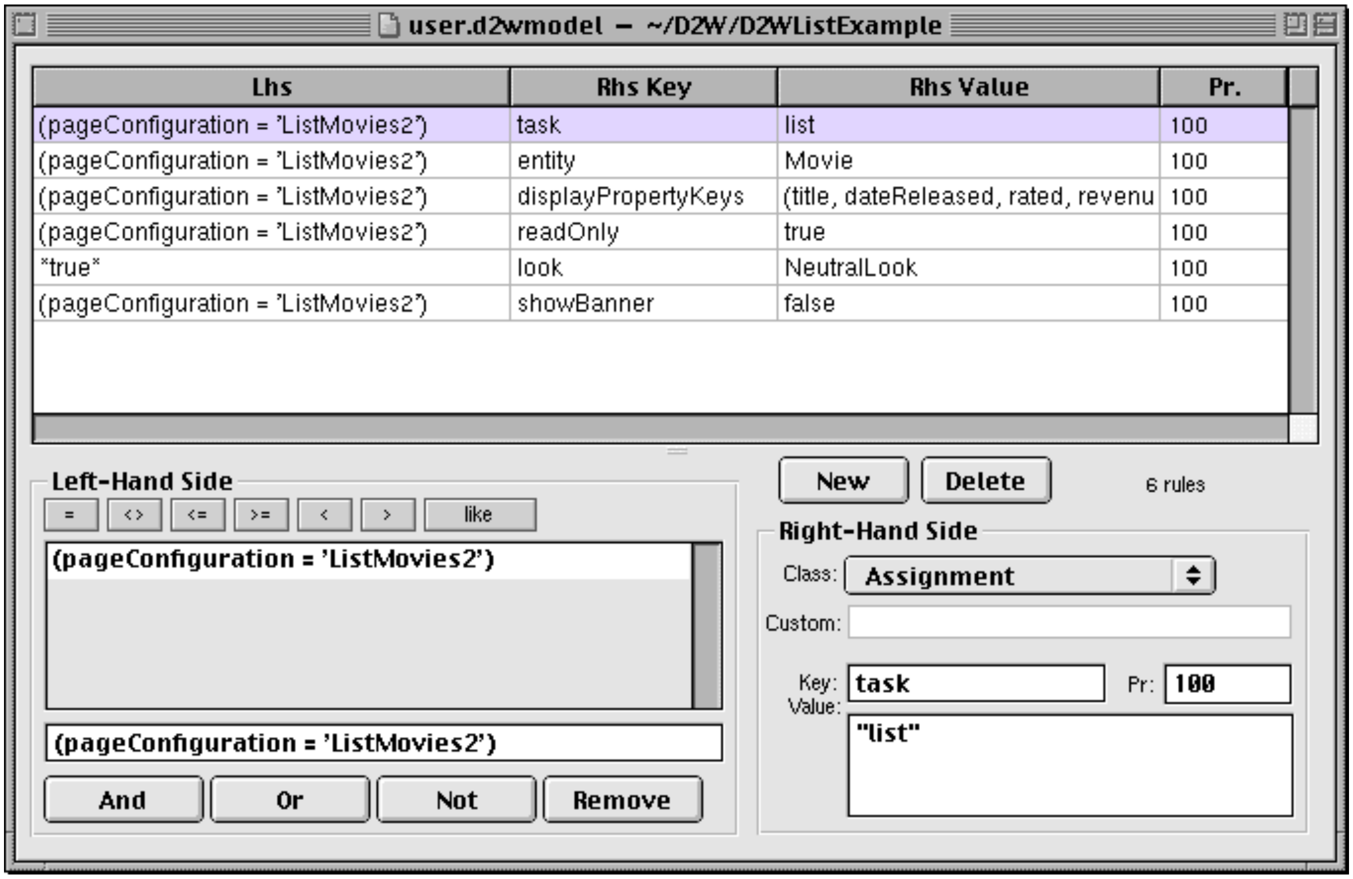
This is an example of a named page configuration: 'ListMovies2'

88 Movie(s) Display 10 items Page 1 of 9

	Title	Date Released	Rated	Revenue
	EOF Next Generation	Jan 24,1996	G	600,000.00
	Star WOB	Aug 22,1999	PG	8,000,000.00
	WOF The Next Big Thing	Aug 22,1999	G	8,000,000.00

Notice that now the banner and trashcan are gone and the editing icon has been replaced by an inspection icon (don't worry soon we will show how to use your own icons instead ;). To add the page configuration "ListMovies2" to the rules we first needed to add a user.d2wmodel to our project. To add a d2wmodel to our example project we open RuleEditor, selected new model and then chose the "Save As" option. We saved the file as user.d2wmodel in the project folder, then we chose "Add Files To Project" from inside Project Builder to add user.d2wmodel to the project's Resources.

Now for a quick'n dirty overview of the DirectToWeb rule system. DirectToWeb rules are very straightforward. A DirectToWeb rule is composed of four parts: the left hand side (lhs) or the condition in which the rule fires, the rigt hand side key (rhs), the right hand side value, and the priority. For example, the set of rules for the "ListMovies2" configuration is composed of five rules. If you open user.d2wmodel and select "Filter Rules" and type "ListMovies2" you should see six rules that look like this:


user.d2wmodel — ~/D2W/D2WListExample

Lhs	Rhs Key	Rhs Value	Pr.
(pageConfiguration = 'ListMovies2')	task	list	100
(pageConfiguration = 'ListMovies2')	entity	Movie	100
(pageConfiguration = 'ListMovies2')	displayPropertyKeys	(title, dateReleased, rated, revenu	100
(pageConfiguration = 'ListMovies2')	readOnly	true	100
true	look	NeutralLook	100
(pageConfiguration = 'ListMovies2')	showBanner	false	100

(pageConfiguration = 'ListMovies2')

(pageConfiguration = 'ListMovies2')

Class: Assignment
Custom:
Key: task
Pr: 100
Value: "list"

For now ignore the rule that has the Rhs key 'look'. These five rules tell DirectToWeb everything it needs to know about the pageConfiguration "ListMovies2". Look at the top most rule:

```
pageConfiguration = 'ListMovies2' => task = "list"
```

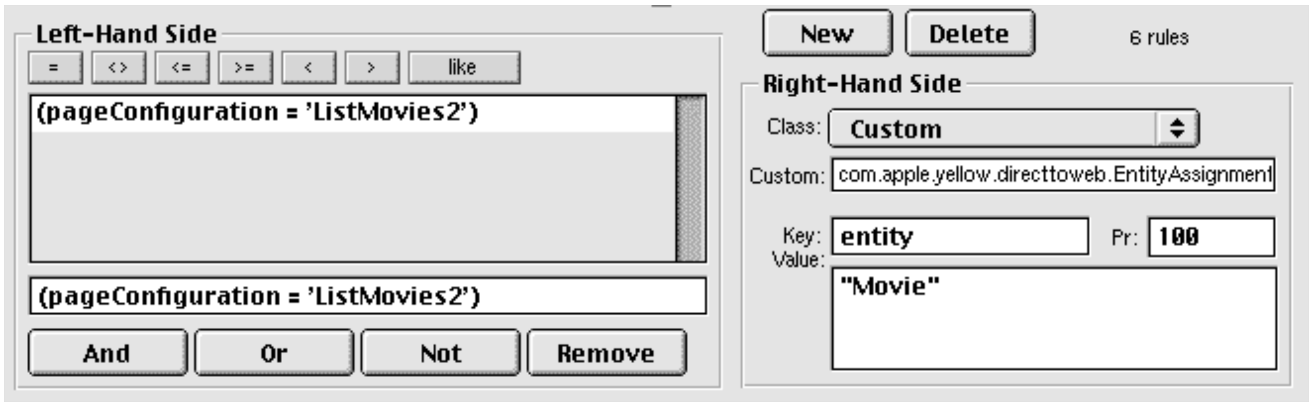
This says for this configuration the task is 'list'. By default, another rule will fire that will look like this:

```
task = 'list' => pageName = 'DefaultD2WListTemplateName'
```

In this way DirectToWeb will know which template to use for the list, in other words all the D2WList component really is is a WOSwitchComponent where the componentName is bound to the value pageName from the rules. Now click on the rule:

```
pageConfiguration = 'ListMovies2' => entity = "Movie"
```

It should look like this:



(pageConfiguration = 'ListMovies2')

(pageConfiguration = 'ListMovies2')

6 rules

Class: Custom

Custom: com.apple.yellow.directtoweb.EntityAssignment

Key: entity

Pr: 100

Value: "Movie"

Note that the type of class assignment isn't of the type Assignment, instead it says Custom with the string "com.apple.yellow.directtoweb.EntityAssignment" typed in the white box below. This will be covered in greater detail in another article, but for now just know that the only two special assignments are the EntityAssigment and the BooleanAssignment (click on the rule with the rhs readOnly to see the boolean assignment in action). In this way we can write rules that take advantage of the fact that the key entity represents an entity object. For instance, we could write the following rule if we wanted all objects whose entity is Movie to be readOnly by default:

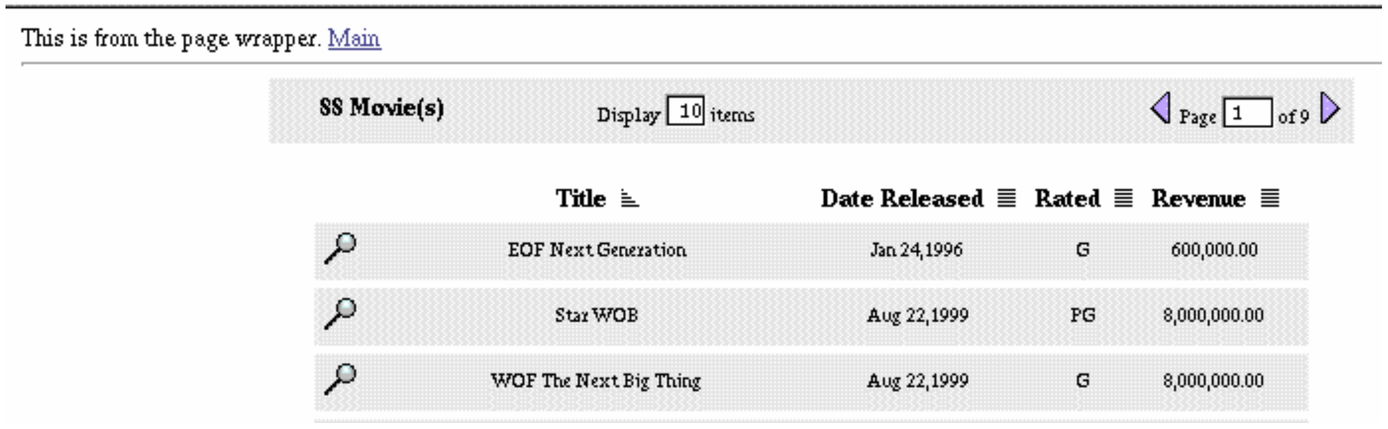
```
entity.name = 'Movie' => readOnly =
"true"(com.apple.yellow.directtoweb.BooleanAssignment)
```

Now look at this rule:

```
*true* => look = "NeutralLook"
```

This rule does not have a lhs component, so we say that this is a default rule, or in this case the default 'look'. This is the only rule that you absolutely must have in your rule file. Without this rule, DirectToWeb will throw an exception when you attempt to render a dynamic page, even if you are not using any of the built-in DirectToWeb templates.

Now what if we don't want to have to have WOComponents that we have to maintain, but instead we would like them to be created on the fly. This is really simple. From the Main page are two links "To List Page 3" and "To List Page 4". These produce almost the exact same pages as pages 1 and 2 except they are dynamicly constructed from the rule system. By clicking on "Link To Page 4" you should see a page that looks like this:



This is from the page wrapper. Main

88 Movie(s) Display 10 items Page 1 of 9

	Title	Date Released	Rated	Revenue
	EOF Next Generation	Jan 24,1996	G	600,000.00
	Star WOB	Aug 22,1999	PG	8,000,000.00
	WOF The Next Big Thing	Aug 22,1999	G	8,000,000.00

The only difference between the above page and "To List Page 2" is the return button at the bottom (not shown above) and hardcoded text "This is an example of ..." that was hardcoded into the WOComponent

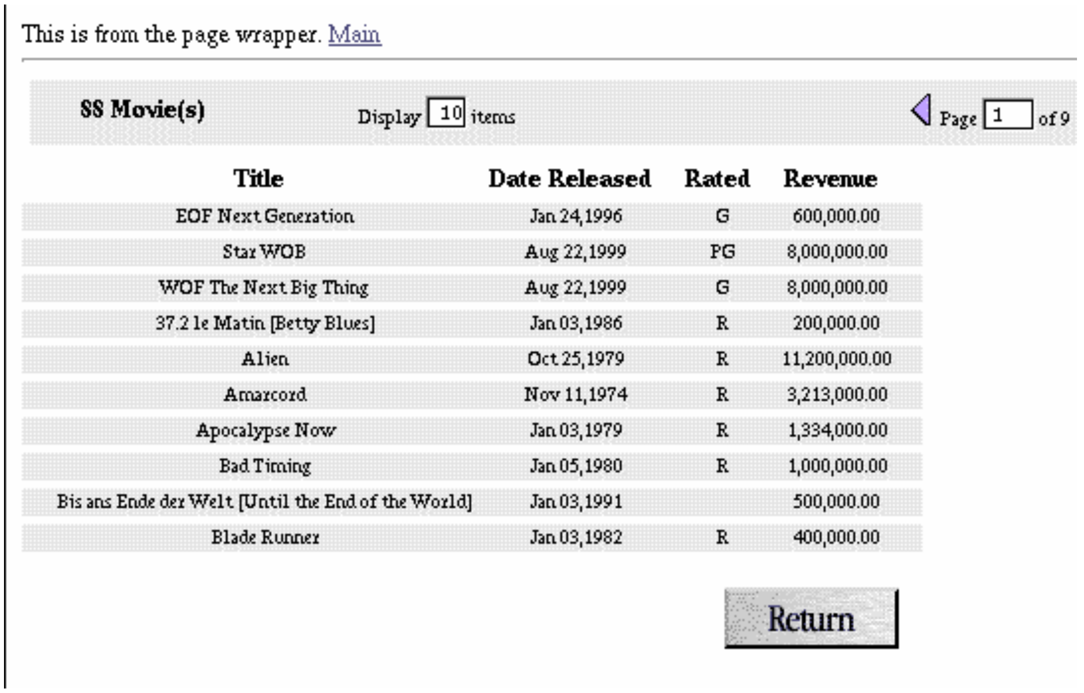
MyD2WListPage2. Also note that we still have the dynamic page being wrapped in our PageWrapper navigation component. The reason for this is that when DirectToWeb renders a top level dynamic page it will look for a component named PageWrapper in your project. If it finds one then it uses that to wrap the dynamic pages in when they are rendered, thus giving you the ability to wrap top level dynamic pages in your own look and feel. Note that only top level dynamic pages have the ability to pick up pageWrappers, embedded components like the D2WList do not display the pageWrapper. Want to use a different name than PageWrapper, say for situations where you don't want navigation shown or if you already have a navigation component named something else? Easy - add a rule with the lhs specified (pageConfiguration = 'APageConfig' or *true* to set the default) and the rhs equal to pageWrapperName = "MyPageWrapperComponentName." The method to return the above page is located in Main.java and looks like:

```
    public WOComponent myDirectToWebListPage4() {
        ListPageInterface lpi =
(ListPageInterface)DirectToWeb.factory().pageForConfigurationNamed("ListMovies2",
session());
        lpi.setNextPage(context().page());
        lpi.setDataSource(((Session)session()).movieArrayDataSource());
        return (WOComponent)lpi;
    }
```

The method movieArrayDataSource off of session simply returns an EOArrayDataSource that contains all of the Movie enterprise objects.

Custom List Templates

Now for the finale ... adding your own look and feel to a DirectToWeb List. Let's look at the result before explaining how to get there. By clicking on the link "To List Page 5" you should see a page that looks like this:


This is from the page wrapper. Main

88 Movie(s) Display 10 items Page 1 of 9

Title	Date Released	Rated	Revenue
EOF Next Generation	Jan 24,1996	G	600,000.00
Star WOB	Aug 22,1999	PG	8,000,000.00
WOF The Next Big Thing	Aug 22,1999	G	8,000,000.00
37.2 le Matin [Betty Blues]	Jan 03,1986	R	200,000.00
Alien	Oct 25,1979	R	11,200,000.00
Amarcord	Nov 11,1974	R	3,213,000.00
Apocalypse Now	Jan 03,1979	R	1,334,000.00
Bad Timing	Jan 05,1980	R	1,000,000.00
Bis ans Ende der Welt [Until the End of the World]	Jan 03,1991		500,000.00
Blade Runner	Jan 03,1982	R	400,000.00

What you are probably thinking now is, "This looks just like a DirectToWebList page!?!?! Where is our custom look?" Well you are right, this is mainly because I completely lack graphic design ability. However, the really neat part is that the entire look of this page is generated from the template named DemoListTemplate. Here you have access to all of the images, all of the colors, fonts, everything. In fact it is fairly trival to integrate a stylesheet into this list component. You can add your own navigation bar, or buttons, whatever you like. Now the cool part. If you are using this style of approach and you have say ten pages displaying different lists of objects, and you need to change the color of text or backgound, you can make that change in one place and have the effect uniform across all places where that list template is used.

Let's take a quick look at how we integrated DemoListTemplate into DirectToWeb. First, we subclassed com.apple.yellow.directtoweb.DirectToWebListPage with our own component called DemoListTemplate. Then, we added a new pageConfiguration "ListMovies3" and duplicated all of the rules from before (Note: not the most efficient number of rules if you look at the operators DirectToWeb has to offer, but I wanted to keep the rules simple), as well as adding two rules to the pageConfiguration:

```
pageConfiguration = 'ListMovies3' => pageName = "DemoListTemplate"
```

and

```
pageConfiguration = 'ListMovies3' => isEntityInspectable = "false"
(com.apple.yellow.directtoweb.BooleanAssignment)
```

We also added a default value to the rules:

```
*true* => isEntityInspectable = "true" (com.apple.yellow.directtoweb.BooleanAssignment)
```

We did this so that we will always get a Boolean value back from the rules when we ask for the key: "isEntityInspectable." Note, the rule system will be covered in far greater detail in another article. If you look at the code in DemoListTemplate.java we have only added two methods:

```
     public boolean showReturn() { return nextPage()!=null; }

     public boolean isEntityInspectable() {
          Integer
isEntityInspectable=(Integer)d2wContext().valueForKey("isEntityInspectable");
          return isEntityReadOnly() && (isEntityInspectable!=null &&
isEntityInspectable.intValue()!=0);
     }
```

The first method is bound to a conditional that displays the "Return" button. By doing this we now have the ability to not show the Return button if we do not set the nextPage on the ListPageInterface. The second is a hook into the DirectToWeb rules that asks for the value of "isEntityInspectable." If you delete the rule:

```
pageConfiguration = 'ListMovies3' => isEntityInspectable = "false"
(com.apple.yellow.directtoweb.BooleanAssignment)
```

then the inspection button will once again show up in the list.

Understand that we are still dealing with the very, very basics here and that we have just begun to scratch the surface of the flexibility offered by DirectToWeb. If you are not impressed with this simple example, please be patient. I want to lay a solid foundation that more complex articles can be built on later. As this is the first in what I hope will be many articles, I would appreciate any feedback you might have about layout and/or content of this article or suggestions for future articles. If you have any general questions about DirectToWeb Omni's webobjects-dev mailing list is a great resource.

Many thanks to Patrice Gautier who is an incredible teacher and if it wasn't for him I would still be approaching WebObjects design the "old" way. Also thanks to Patrice, Angela Muller and Travis Cripps for their valuable feedback.
