Direct-To-Web

Overview

There are two commonly referred to articles by Max Muller that were written several years ago. The first is located at Stepwise (or in the Wonder frameworks): http://www.stepwise.com/Articles/Technical/D2W/D2W1.html

There is a second part to the article linked in the Attachments tab above.

The most complete overview continues to be the Apple Documentation.

There is additional material in the two chapters in the Professional WebObjects 5.0 with Java book. This material is essential reading. There is a Case Study in the book that gives some hints at how powerful the rule structure can be (I think this may have been written by Max Muller as well). This case study is also a mini introduction to some of the Wonder frameworks (before they were open sourced). The application example in the Case Study chapter is also located in the Wonder frameworks (Wonder/Common/Examples/Wrox/NetstruxrCaseStudy/). You can get it working once you make sure that the EOModel is pointing to the right database. The important thing about the NetstruxrCaseStudy is that it uses D2W and gives examples of the types of rules you can add to your application. In particular, the concept of creating sub-tasks is covered.

Additional Comments on the referenced articles

The following notes were made as I was trying to learn how direct to web works. I hope they are useful. The whole process, including all of the trials and errors, looking in the wrong place for things, getting things to build correctly, and writing up this little summary (tasks which the reader will hopefully now be able to mostly avoid) took me about three hours. At the end, I feel like I have a fairly decent basic understanding of the beginnings of how to use direct to web (without knowing anything about what project wonder adds yet).

I would recommend beginning at http://www.stepwise.com/Articles/Technical/D2W/D2W1.html. This article is also in the D2WListExample project from project wonder (see below for how to get it from Eclipse) under Components/BestKeptSecret.wo/BestKeptSecret.html. The article is pretty good, but the first part was very confusing to me. Once I finally understood what was being talked about, I wrote a brief synopsis of the content, which makes more sense to me. That is shown below, but before you start reading, I highly recommend that you create an Eclipse workspace for holding project wonder projects and examples (so that you can easily reference the source). I just went ahead and got everything, as shown below:

First, I created a new workspace in Eclipse (File->SwitchWorkspace). I called it "CurrentWonder" since it would hold all the projects imported from CVS. The project referenced in the article (D2WListExample) already exists in Eclipse format under the Wonder repository, so you don't need to follow the link from the article to get the code (the linked code is in XCode format and rather old).

Checked out wonder
in eclipse, right clicked in package explorer
import
cvs -> Projects from CVS
host: wonder.cvs.sourceforge.net
repository path: /cvsroot/wonder
user: anonymous
Use an existing module (I selected all of the ones listed below)
common-> examples -> D2WComponentTour
frameworks -> all
ERDirectToWeb

By default, I had the eclipse compiler set to use Java 1.4.2. I seemed to have a lot of compile errors, and switching the eclipse compiler and runtime to 1.5 really seemed to help (look under Preferences->Java). Also, several of the projects apparently did not get everything down correctly when I checked them out, because they were missing pieces that the compiler was complaining about. I did an update (right click on the project in Package Explorer ~~-> Team -> Update) on the projects in question, and everything was better. Finally after all of that, I had to clean all of the projects to get the dependent projects to build correctly after the fixes to the ones that didn't get everything (Project~~>Clean->Clean all projects). Now I have a happy, cleanly building set of all the project wonder frameworks and examples.

OK, now we're ready!_*If_you_like,_read_my_synopsis_below_and_then_dive_into_the_first*_"Best_Kept_Secret"_article! After you've read it, open D2WListExample->Components->BestKeptSecret and read the next one.

In a component, you often want to display a list of EOs. You could create a table and display group, add headers, add a repetition, add functions to sort the columns, add edit buttons, etc., etc.. OR, you could use a reuseable component. Because there are so many configurable pieces to this particular component, using a standard component with bindings just isn't going to work. There are too many potential bindings. So Instead, you can use a special component called D2WList which takes the list of items you want to display ("list"), the name of the entity from your eomodel for the things in the list ("entityName"), and a tag that points to a ruleset about the display properties of this list, which you define in a different format elsewhere ("pageConfiguration"). If you look at MyD2WListPage2.wo in the D2WListExample project, you will see a webobjects tag named List1 of type D2WList (bindings shown below):

```
List1: D2WList { pageConfiguration = "ListMovies2"; list = session.movies; entityName = "Movie"; }
```

The "ruleset defined elsewhere" that I mentioned earlier is found in the direct to web model file in the Resources folder of the project: user.d2wmodel. You can open this file in RuleModeler (which can be found at http://wocommunity.org/documents/tools/), and set up various properties such as the template to use to display the list, whether each thing in the list is editable, which properties to display, etc. The "left hand side key" column shown in RuleModeler is the thing that lets the D2WList component find the rules for the list it is showing.

You can also just have a full page generated for you, rather than a subsection of a page that contains your list. This part of the article made better sense to me, so I'll refer any readers directly to it (everything from the section labeled "Dynamic page creation" down). Now that you have some context for what the article is helping you do, the first part hopefully will make more sense as well.

Quickstart Demo

Here is a quick way to get started with Wonder's Direct-to-Web extensions, using Xcode 2.3, WebObjects 5.3.1, and Mac OS 10.4.6

- Download and install the Wonder frameworks
- Follow the Project Wonder Quickstart with the following differences:
    - Add ERDirectToWeb.framework to your project
    - You will need to add an EOModel to your project, either one of your own, or the sample ones in JavaRealEstate.framework or JavaBusinessLogic.framework
    - Have DirectAction extend ERD2WDirectAction instead of ERXDirectAction. You'll need to import er.directtoweb.*;
- Now try some of the direct actions that ERD2WDirectAction provides by appending a pageConfiguration name to an entity name. So ListCustomer will use Wonder's List page configuration with the Customer entity. Some other page configurations not shown in the docs are:
    - EditList<EntityName>: this will show a list view of the entity but all of the fields are editable for all of the rows
    - ListGrouping<EntityName>: for this you will need to add a rule, for example: (pageConfiguration = 'ListGroupListingAddress') => groupingKey = "city" -- *the ListGrouping template does not appear to be grouping correctly with this configuration yet.*
    - ListXML<EntityName>: this will return the list in some funky XML format. The format of course can be changed to meet your needs. There is also a CSV format available, but you'll need to add a rule similar to (pageConfiguration like 'ListCSV*') => subTask = 'csv' and then ListCSVCustomer will output in pipe-delimited format. You can also add a rule like (pageConfiguration like '**PrinterFriendly**') => subTask = 'printerFriendly' and then ListPrinterFriendlyCustomer will show a list in a printer friendly format.
    - ListCompact<EntityName>, PickCompact<EntityName>, etc.: the compact format excludes the page wrapper and navigation bars
- To see some D2W debugging details, open the PageWrapper.wo template in your Xcode project (in the Components folder). Insert a custom component below the existing content (Dynamic->Custom...) with a class of ERD2WDebugFlags. Click OK and then save the file.
  For this new component to be visible your application has to be in development mode. This involves setting a property in the property file:
  er.extensions.ERXApplication.developmentMode=TRUE

Go back to the browser and refresh the screen, and you should see 2 new links for Log4J and D2W Info. Click the link for D2W Info, and it will show some details of what settings are active for the given components.
--My version of the sample Real Estate database has some bad data for Agents, so you may have trouble working with the Agent and AgentPhoto entities until you clean it up manually.

Related Links

- RuleModeler, a Rule Editor replacement