Changing the Rules with Direct to Web

Changing the Rules with Direct to Web

by David Holt

Intent

This is a document that I have been working on for quite some time, but I have always felt that it wasn't complete enough to actually put it out in the wild. WOWODC 2009 has rejuvenated the WebObjects developer community's interest in D2W and in the Wonder additions to it (ERD2W). Now seems like an opportune time to continue the recent work by Ramsey Gurley, David LeBer and Ravi Mendis to promote and improve the D2W documentation and Look frameworks and further expose the outstanding work done by the Wonder team (especially Anjo Krank who, to his credit, has tried at every opportunity to get people to look at this stuff for years).

Given the relative lack of documentation for such a huge topic (and my relative lack of practical experience with the technology), the majority of this content has been culled and organized from almost 10 years of posts to the various WebObjects mailing lists. I have tried to put it in my own words, but if you think you recognize an old post about a given topic, you probably do. Unfortunately, the "official" documentation stopped long before the power of this technology was understood by WebObjects developers and the tool chain has changed significantly enough over the years that the original documentation and tutorials are a little hard to follow for someone starting out. With luck, this work will begin to address this issue for you. Please feel free to edit the content or post comments and questions here or on the lists. I look forward to a renewed dialog about D2W!

Introduction

Now, adding new features to my application is done by adding a few entries into a navigation list, (probably) writing some new rules and localizing some keys. That's it. For most of the stuff I get asked for, it takes more time to write the commit messages to my version control system than it does to actually implement the requested changes. The day I decide I revamp my interface I need to change about five templates and adjust one CSS file and I'm done. -Paraphrased from a post by Anjo Krank

What developer wouldn't want this kind of power at their fingertips?

What is DirectToWeb?

DirectToWeb describes a technology that STILL, some nine years after its introduction, points to a logical future for how data-driven web applications ought to be developed. DirectToWeb is an expert system applied to web applications. "The general architecture of an expert system involves two principal components: a problem dependent set of data declarations called the knowledge base or rule base, and a problem independent (although highly data structure dependent) program which is called the inference engine." http://en.wikipedia.org/wiki/Expert_systems The WebObjects developer that wishes to use DirectToWeb to create his application manipulates the rules in the rule base to result in the desired behaviour. The DirectToWeb frameworks contain the inference engine that is used to determine what rules will be fired.

David LeBer has now posted links to his entire presentation introducing D2W / ERD2W from WOWODC 2009.
See also What is Direct to Web? by Ramsey Gurley.

D2W as Expert System

General information about Expert Systems below has been taken from http://en.wikipedia.org/wiki/Expert_systems and http://en.wikipedia.org/wiki/Inference_engine

An inference rule is a statement that has two parts, an if-clause and a then-clause. In D2W this is referred to as left hand side (LHS) and right hand side (RHS). The LHS is used to determine whether something is true and the RHS assigns a value if the LHS condition is true. When more than one rule is true, the more specific rule (i.e. the one with more terms on the LHS) will fire. In the case of the same specificity, rule priorities are compared.

One advantage of inference rules over traditional programming is that the rules use logic and reasoning that is similar to how humans reason. The disadvantage in the case of D2W is that traditional programmers find it unfamiliar and perhaps a little threatening to think like this!

When a conclusion is drawn from the rule system and a rule is executed, it is possible to retrace the reasoning that the system used to fire that rule (there is a special debug flag in Wonder that allows you to explicitly follow the rule system step by step as it works through the logic and makes choices about the application's behaviour).

The engine that drives and evaluates all these rules in D2W is called the Rule System. Given that you'll be writing the rules that are going to be evaluated by the rule engine it may be helpful for you to know what the system is doing behind the scenes. In general, the Rule System is an inference engine with three states. The first state "matches rules" that meet the current conditions of the application. The rules that match are passed to the second state called "select rules" which determines which rules will be executed (or "fired" in D2W terminology). D2W uses heuristics to determine which rules fire. In some cases the stock D2W rule system has limitations (for example it had limited significant keys used to evaluate rules, so the programmer needed to add additional keys so that rules would fire correctly) that Wonder (ERD2W) has addressed using a more modern "select rules" process that removes the original limitations. The selected rules are passed to the third state "execute rules". The rules that execute will change the context of the application, whether through updates in the database or presenting a different UI to the user. This will cause the rule system to cycle back to the first state, ready to match the appropriate rules for the new context of the application. For much greater detail about the mechanics of the D2W rule system please see The D2W Rule System by Ramsey Gurley.

As an expert system that has been designed to capture the process of developing a data driven web application, DirectToWeb inherits the advantages and disadvantages of any expert system: http://en.wikipedia.org/wiki/Expert_systems#Advantages_and_disadvantages

Advantages:

- Provides consistent answers for repetitive decisions, processes and tasks

    - The normal WO developer's tasks involve programming similarly functioning components that are created to perform a common set of repetitive tasks on entities. D2W allows the developer to leverage the similarities and pulls out the common tasks (for example: query, list, inspect, edit, select, confirm) to put them in their own templates and apply them to your entities (defined by the EOModel).
- Holds and maintains significant levels of information
    - The D2W rules system knows the tasks that you will want to use in a typical application for a typical entity. It also knows when you need the behaviour for a given component to be different from the default because you have told it so.
- Encourages organizations to clarify the logic of their decision-making
    - Once the organization determines their business rules, the developer has a logical way to codify those rules into the rule base. D2W uses "rule models" that are edited using a specific tool for the job. Once the codified behaviour is observed in the application, the business logic can be verified by the organization and quickly changed if it is incorrect.
- Never "forgets" to ask a question, as a human mind
    - Once the rules are established, the rules are always considered to determine the "correct" behaviour for the given situation.

Disadvantages:

- Lacks common sense needed in some decision making
    - Not a severe shortcoming for WebObjects developer since the practical range of UI options for a finite set of data types is also finite
- Cannot make creative responses as human expert would in unusual circumstances
    - The development of a website doesn't often present unusual circumstances and when it does, traditional completely customized component-based WO development is always available as an option.
- Domain experts not always able to explain their logic and reasoning
    - This is not likely true of the process of developing WebObjects applications
- Errors may occur in the knowledge base, and lead to wrong decisions
    - This is true of any application! You have to test and correct errors.
- Cannot adapt to changing environments, unless knowledge base is changed
    - This is true of any WebObjects application, but based on Anjo's description at the top of this page, the scope of the work to change a D2W application is potentially much less.

Historical Context of D2W

D2W is a simple system that uses true/false logic to evaluate rules against the existing data and the continually updated D2WContext. The goal of D2W is not to provide sophisticated reasoning ability, but to provide the ability to describe in a declarative manner defaults for components, sizes, layout, etc., and to do so fast enough to be usable in a production system. In practice, D2W makes publishing a database on the web a trivial exercise by creating pages to query, list, display and edit objects. It does this by introspecting your data model and its context at runtime to tailor the display to the current condition.

D2W is not easily explained to those who have not used the technology. Part of the problem is that the target audience has changed over time. Initially it was designed to get beginning developers through the steep learning curve associated with WebObjects. The power lay in freezing auto-generated components and then modifying the resulting pages. This idea became closely tied to D2W, but it is actually for beginners and offers very limited (although interesting) capability compared to the potential. Anjo refers to using D2W at this level as "using the clicky thing", by which he refers to the WebAssistant, to make simple changes to your application's behaviour such as hiding, showing or reordering the display of your entities' attributes, or making changes to the page loaded for a given task/entity combination. So long as your application's requirements are simple, this may be as far as you need or want to take the D2W concept.

If you stop there, though, you are missing out on the vast potential that became apparent to the NeXT developers after working with the concept for awhile: the power of the D2W system lay in moving more of the information into the rules and away from frozen components and code. Unfortunately, developers who try to take an application through the transition from the WebAssistant modified behaviour to actually writing the rules that underly the assistant will find that the rules will no longer be "human-readable" because of the way that the assistant creates the rules. If you look at the HTML output of any common word processor compared to a hand-coded page, you'll have an idea what is in store.

What this really means is that there is yet another leap in the already steep learning curve of WebObjects to take D2W to its most advanced level. You should know going in that "in the real world you will need to invest a lot of time to get things working the way you want and maintain them until you understand what is happening in your application." The object of this document is to reduce the time you need to invest to understand the technology and the capabilities that it offers. If you decide to take this leap, your productivity gain by using D2W will more than offset any time investment you make to learn it. The first reason is that the D2W rule system allows you to create default rules and then specialize them. The second reason is that your application programming will begin to leverage Project Wonder which was created to expose the thinking that allows you to best exploit the power of D2W. If you really intend to learn D2W advanced features, you will be obligated to link in the Wonder frameworks to take advantage of its bug fixes to the stock WO frameworks and the vast number of additional components that you will get to use for free.

DirectToWeb in Practice

The DirectToWeb framework includes the inference engine and stock components designed to access information contained in the inference engine.
It is important to note right off that there is really no such thing as a "DirectToWeb Application" despite the common practice of calling the default projects created by using the D2W project template in Xcode "DirectToWeb" applications. In reality, the project templates that refer to "D2W Application" included in WOLips, are simply referring to a WebObjects application that has had the D2W related frameworks linked in by default. In fact, you can leverage D2W in any WO application by linking in the relevant frameworks and then using the components in your pages (or as your pages). See David LeBer's excellent screencast about adding Excel spreadsheet generation to your existing application by leveraging D2W: SCREENCAST: USING EREXCELLOOK

DirectToWeb does not generate code. It uses the inference engine combined with templates for pages and display components to determine behaviour and rendering. The application that is created by using the default D2W project template has very simple and generic behaviours and appearance. When customization is required, the developer can make changes at the rule level, the template (page) level, or at the component level. Learning the most appropriate place to intervene is the crux of the new learning curve that you have embarked on!

D2W dynamically generates web applications based on rules that are matched, evaluated and executed in a context. The context continuously tracks the state of the application and its related data. Rules do not have to perform simple assignments. They can perform more complex assignments using KVC or they can even use custom assignment subclasses.

Outline of Topics that I Intend to Cover

- Covering default behaviour of a D2W app
- How to change default behaviour by simple means
    - General rule concepts
    - Default rule writing conventions
    - Setting default property values in your application
    - Changing the task on entry to the application (by default it is the infamous "Query All" page that scares everyone)
    - Changing the PageWrapper that the page uses to construct its dynamic content
    - Changing the display of an entity's attributes and relationships (hide or show, order, choosing an appropriate display component)
    - Using existing templates and property level components to achieve the behaviour you're looking for
- How to change the default behaviour by more complex means
    - Creating custom UI and behaviour
    - Creating custom property level components
    - Creating custom page level templates
- Named page configurations
- Assignment classes
- Look Frameworks
- How to create your own Look Framework
- Using D2W components inside an existing application
- The importance of the the ERD2W debug flags and how to use them
- How to take advantage of -D2WTraceRuleFiringEnabled
- Advanced topics
    - BugTracker
    - Localization
    - Validation
    - Page delegates and changing application flow
    - ERXNavigationMenu
    - Reporting