An Introduction to Apache Struts Development with Oracle JDeveloper 10*g*

An Oracle White Paper
December 2003

ORACLE

An Introduction to Apache Struts Development with Oracle JDeveloper 10*g*

An Introduction to Apache Struts Development with Oracle JDeveloper 10*g*

Oracle JDeveloper 10*g* can open and model Struts applications from any source, be it another IDE or a hand-coded Struts configuration

INTRODUCTION

This paper is concerned with the use of the Apache Struts Controller within web deployed thin client applications developed using Oracle JDeveloper 10*g* The term Controller in this context relates to the specific Model-View-Controller (abbreviated to MVC) design pattern or architecture. Although there is not space in this paper to cover MVC in detail[1], an overview is useful.

An Overview of the MVC Design Pattern

When we look at database-centric applications in general, and web-based thin-client applications in particular, we can see that the application has to undertake several distinct tasks:

- Data access
- Business logic implementation
- User interface display (presentation of data)
- User interaction
- Application (page) flow

The MVC architecture provides a way of compartmentalizing these tasks, working from the premise that activities such as data presentation should be separate from data access, so that we can, for instance, easily plug in a different data source to the application without having to rewrite the user interface. So, MVC is all about the logical separation of an application into three distinct layers, or functional areas in particular, called the Model, the View, and the Controller

[1] For more information on the MVC design pattern refer to the Sun blueprints at http://java.sun.com/blueprints/patterns/MVC.html
For a more detailed introduction to MVC as a whole see the article *Understanding MVC* on OTN: http://otn.oracle.com/oramag/webcolumns/2003/techarticles/mills_mvc.html

The Model

The Model is the repository for the application data and business logic. However, it is too simplistic to say that the Model is a representation of the database. Generally, with a database-based application, part of the Model's function is to retrieve data from, and persist data to, the database, but it is also responsible both for exposing the data in such a way that the View can access it and for implementing a business logic layer to validate and consume the data entered through the View

So, at the application level the Model acts as a validation and abstraction layer between the user interface that displays and the business data that is displayed. The database server itself is simply a persistence layer for the Model.

The View

The View is responsible for rendering the Model data,, in the common case with Struts, this would involve using a JSP page, although other view technologies such as the XML driven ADF UIX framework are also valid. The important factor here is that the View code itself does not hardcode application or navigation logic, although it may contain some logic to carry out tasks like conditional data display based on a user's role. When an end user carries out an action within the HTML page that is eventually rendered from the View, an event will be submitted to the Controller, and it will be up to the Controller to work out what to do next.

The Controller

So the Controller is, as the name suggests, the lynchpin of the whole pattern. Every user action carried out in the view is submitted through the Controller which, based on the contents of the request from the browser, combined with its programming or metadata, decides what to do next.

Controllers can be driven in several different ways. Some will use URL arguments to route requests to the correct piece of code, some will take into account the page that the request was submitted from, some will use hidden fields on the page submission to work out what to do. The MVC design pattern itself does not prescribe how the Controller should work, simply what its function is.

AN INTRODUCTION TO APACHE STRUTS

Apache Struts is a Controller implementation, although it goes a little further that as we shall see. The Controller portion of Struts is provided by an HTTP Servlet, that is driven through an XML configuration file (usually called struts-config.xml.).

For more information about the basics of the Struts framework, you can visit the Apache Struts website (http://jakarta.apache.org/struts).

Struts as a Controller

The Controller functionality of Struts revolves around the use of *Actions*. An Action is a place where the programmer can define code to process the input from a page and define a *Forward*, which tells the controller which page to display next. It will be used as the target of a submit on an HTML Form, and it will be provided with the contents of the page for processing.

A simple scenario using an Action would be a logon screen:

1. The Controller dispatches the logon page to the browser
2. The user fills in the logon information and presses the logon button to submit the page to the logon handling Action.
3. The logon Action is provided (by the Controller servlet) with a JavaBean containing the information entered by the user of the page. The code in the Action can then use this information to validate the user.
4. The Action completes returning a Forward destination to the Controller, in this case, back to the logon page if there was a problem, or on to the rest of the application if the logon was successful.
5. Finally the Controller dispatches the relevant page back to the browser to display.

The programmer can use these Actions to process input from the page and to define content for subsequent pages as well as simply directing the flow of pages to display. So it is in these Controller Actions that your "Model" interaction would take place (for instance inserting the data supplied by the user on a page into a database table)

The Struts XML configuration however, only supplies half of the page flow story, it defines the flow from Actions to other Actions and pages, however, it does not define the flow from pages to specific actions. That is a job for the portion of Struts that lives in the View layer.

Struts in the View layer

So to complete the Page flow story Struts has to also extend into the View layer by providing several functions which complement the controller functionality:

- JSP tags to aid page navigation within the context of the Struts servlet - Flow from the page to Controller Actions.
- Facilities for simplifying the transport of page content (specifically HTML Forms) via JavaBeans into the Controller for programmatic access by the Actions

- Data aware tags which co-operate with the JavaBean data transport mechanism that Struts uses binding that data into HTML input controls or as text[2]
- Utility tags for rendering nested data or looping through arrays of data.
- Internationalization of static strings

So building a Struts application is a combination of defining page flow in the XML metadata and the use of the specialized Struts tag libraries within the JSP pages of the application.

STRUTS IN ORACLE JDEVELOPER 10*G*

Previous versions of Oracle JDeveloper have provided help with the building of Struts applications on two fronts:

- A dialog based Struts configuration editor providing a structured interface to edit the struts-config XML metadata. This gives less experienced Struts users the ability to edit the metadata without having to understand the Struts XML definition directly. More experienced users can of course directly edit the XML or use a combination of direct editing and the dialogs as the two interfaces are synchronized.
- Data aware tags and specialized actions for integrating Struts applications with a model layer based on the Business Components for Java framework (BC4J). These specialized tags and actions make it much simpler to solve the problem of model layer integration for database enabled applications.

So let's look at how things have changed in Oracle JDeveloper 10*g* and how the productivity in both the areas of XML definition and model integration have been further simplified and made more productive.

The Struts Configuration Editor

The struts-config.xml file is of course the key to a Struts application. It defines a portion of the page flow, provides the abstractions for the view implementation and of course defines the portion of the controller logic that the developer codes in terms of the utilization of custom actions that the programmer has created.

Oracle JDeveloper 10*g* provides a primary view of the Struts metadata in the form of a page flow diagram. This makes it much simpler for the programmer to understand the flow logic that they are building as it is all laid out visually. Of course the struts-config.xml file does not tell the whole story, although it defines

[2] Some of the facilities covered by the Struts taglibs are capabilities shared by the Java Standard Tag Library, specifically accessing and iterating through data from Objects stored in the session. Struts tags and JSTL tags are often used in concert.

the flow from a specific Struts Action to further Actions or page displays it does not encode knowledge of the flow back from pages to those Controller Actions.

The Struts page flow representation within Oracle JDeveloper makes up for this deficiency by following the logical flow into the designated pages and parsing out the navigational information from within these pages to add to the diagram. This provides a much fuller picture of the application page flow as a whole.

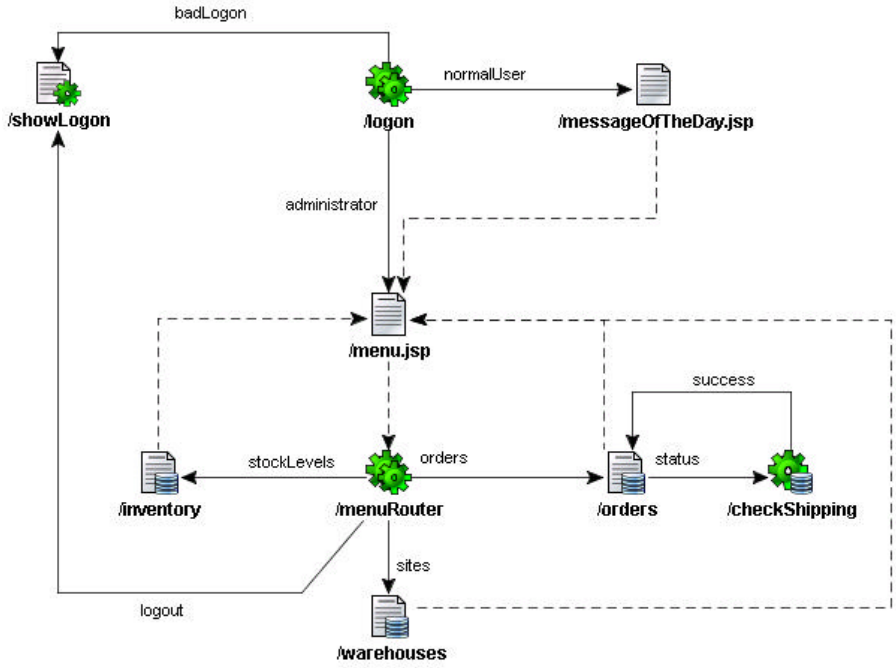

Figure 1: Struts Page flow in Oracle JDeveloper10*g*

The parsing logic understands both JSP and Struts JSP navigational tags such as <a>, <html:form> etc. as well as the navigational logic encoded into ADF UIX pages and, in the future, JavaServer Faces based pages.

Oracle JDeveloper 10g does not add any custom information to the struts-config.xml file to support it's diagramming capabilities. As a result you can take a Struts configuration created by Oracle JDeveloper 10*g* and use it in any Struts 1.1 editor, just as you can do the reverse with a configuration file from another source.

The Page Flow Diagram as a Workbench

It is important to understand that the page flow diagram that represents the Struts configuration and the Page encoded navigation logic is not simply a static rendition of the flow, it is a fully capable editor for the underlying metadata. So you can use the diagram to literally draw out your page flow by dragging the appropriate objects (Actions, Page Forwards, pages etc) from the component palette onto the drawing surface and linking them up with simple point and click

gestures. As you create Struts objects on the diagram, the Struts configuration file is fully synchronized with it.

Once the outline of the flow is on the diagram you can start to use it as a workbench to implement the various parts of the application. From any object on the diagram you can drill down to it's implementation, for instance, into the execute method of an Action or into the visual editors for a JSP or ADF UIX page, something we'll cover in a moment.

You can also use the diagram as a convenient place to generate Struts form beans for a particular action from an option available off of the context menu. And most important of all you can run the page flow right off of the diagram.

Alternate Editing of Struts Metadata

Of course a visual view of a page flow is nice to have, but that should never constrain programmers to only editing the metadata through that interface. Quite apart from personal preference for the use of text editing as apposed to diagramming, there are important issues of accessibility to address. Oracle JDeveloper 10*g* provides several further ways to manipulate this metadata in addition to the diagram and the Oracle JDeveloper 9.0.3 console style editor.

Editing the XML directly

As stated before, the struts-config.xml is edited through the diagram UI in a synchronized way, so diagram changes are automatically reflected in the xml. There is no generate step to go through. Conversely, as the struts-config xml is the metadata that defines the diagram, manual changes to the XML in the XML Code editor will be automatically rendered into the visual diagram. This has the benefit of enabling you to import any Struts configuration regardless of how it was authored and see a visual representation of it in Oracle JDeveloper.

Whilst in XML editing view, the XML that is entered will be automatically validated against the Struts DTD for you and code insight can help the programmer with the exact tag and attribute syntax.

Using the Structure Pane and the Property Inspector

In common with the rest of Oracle JDeveloper, The structure pane and property inspector can be used in concert to provide another way of viewing and editing an object such as the Struts configuration. In the Struts case, the structure pane provides a hierarchical view of the xml file and provides the mechanism for adding Actions, Beans etc into that XML tree. The attributes of the individual Struts elements selected in the Structure pane or on the diagram can then be edited through the synchronized property inspector, which again enforces the validation implied in the Struts DTD file.

The Property Inspector itself provides a series of built-in editors relevant to the properties being manipulated, for instance class browsers for setting the type and

className attributes and intelligent poplists for cross referencing Form Bean and Actions and so on.

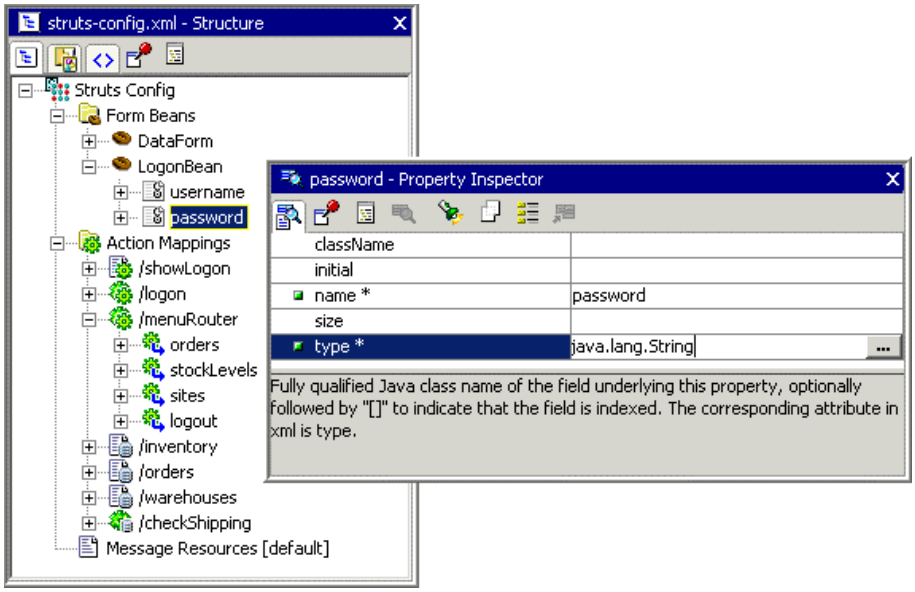

Figure 2: Editing the Struts configuration through the Structure Pane and Property Inspector

ADDED VALUE WITH DATA ACTIONS AND DATA PAGES

Struts as used within Oracle JDeveloper 10*g* is a pure 1.1 distribution direct from Apache with no extension or alteration of the Servlet itself. Where Oracle has added value is in the area of Model integration. Struts itself adds no explicit help with the integration with your database, other than providing the ability to define JDBC connections in metadata.

The Oracle Application Development Framework (ADF) is a runtime framework which allows the developer to increase design time productivity whilst utilizing the robust, scalable and flexible architecture of the J2EE platform.

Data Actions and Data Pages are specialized Struts actions, provided by ADF, which understand how to communicate with the model layer as defined in JSR 227 *Standard Data Binding & Data Access Facility for J2EE*[3]. As such, the Data Page can be used to display data bound pages that use the ADF model and Data Actions can be used to invoke business methods as part of a process flow. The data binding used by these actions removes much of the workload required to integrate Struts with various data sources such as Enterprise Java Beans(EJB), Web Services or databases.

Functionality provided by the Data Page and Data Action includes:

[3] http://www.jcp.org/en/jsr/detail?id=227

- Querying data from the Business Service – The Data Page action will obtain data from the Model and the underlying business service and make it available to the page using JSTL (Java Standard Tag Library) tags.
- Handling standard CRUD operations – A Data Page or a page that is managed by a Data Action will handle operations such as inserts, updates and deletes of data displayed on a submitted page and similarly handle other operations relating to a record set such as paging through the records, or executing methods.
- Business Service Methods in the Page flow – The model as defined by JSR 227 is not only concerned with the enabling pages for basic database interactions sourced from EJB, Toplink or ADF Business Components. The model can also be used to expose ad-hoc methods and record sets from diverse business services such as Web Services, or indeed any other type of Java Class. Such methods can be bound to the user interface on the page though a suitable control (for instance a button) or they may also be bound directly to a Data Action though a simple drag and drop operation. When such a bound Data Action is executed within the Struts page flow, the corresponding method on the business service will be called.

VISUAL EDITING OF STRUTS BASED PAGES

The Struts page flow diagram primarily operates on the Struts configuration XML file, but, as was said before, Struts also has tag libraries, which are used within the JSP views themselves, so how are these supported in Oracle JDeveloper 10*g*?

Within Oracle JDeveloper 10*g* there is a fully featured HTML/JSP editor which provides a range of views on the pages you are authoring, WYSIWYG visual views of the page, the page code with full code insight on the JSP markup and a structural view of the page which can be used in concert with the property inspector to edit the page also.

The JSP editor provides multiple groups of components for inclusion in the pages. The standard HTML objects are included of course, but also custom tag libraries such as the Struts and JSTL libraries are exposed.

So to place Struts tags into the page, it's as simple as selecting the relevant set of Struts tags (Struts HTML, Struts Bean etc.) in the component palette and dragging the required tag into the visual editor or code view. The editor already understands the Struts tags and can render them in a meaningful way rather than rendering them as generic "non standard" tags that other HTML editors may fall back to.

The visual editor can also be configured to pre-render tags as required. This is very useful in Struts page development where most static text on the page would

be held as a property in one of the message bundles associated with the project. This preview rendering feature means that <bean:write> tags can actually be rendered as the final display string rather than just as a tag or the resource key, making the page editing environment much more WYSIWYG.

The following figure shows a data bound page that uses a mixture of Struts and JSTL tags viewed in the visual editor. You an also see the structure of the page and the properties of the currently selected tag in the Structure Pane and Property Editor respectively.

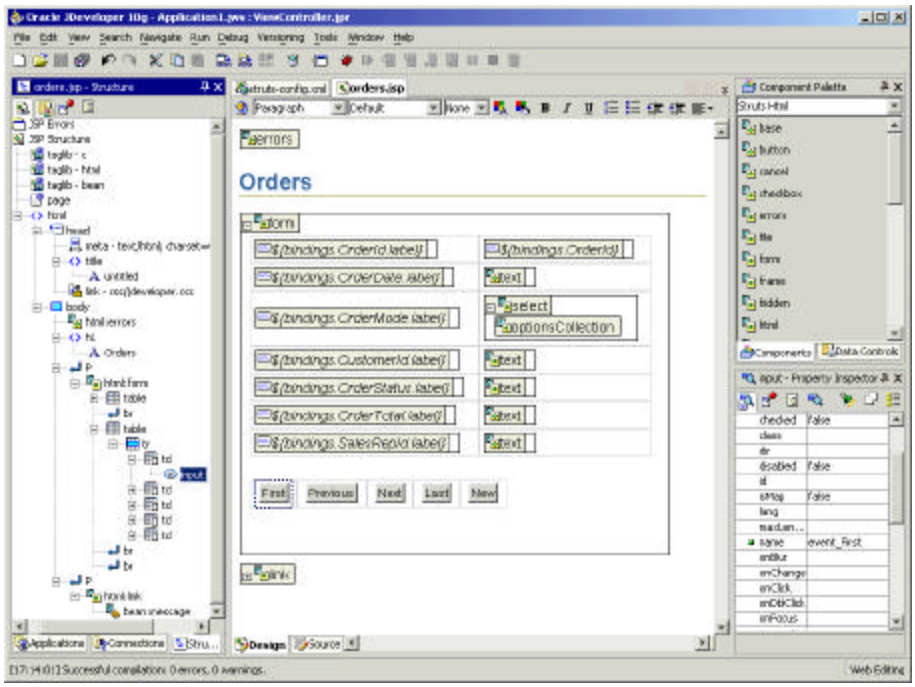

Figure 3: Visual editing of Struts based pages

CONCLUSION

Oracle JDeveloper 10*g* provides a cutting edge environment for building and maintaining Struts applications from any source. The IDE provides a set of intuitive visual tools to simplify the development process without sacrificing openness or flexibility.

The visualisation of page flow through the Struts diagram and the help that the JSP visual editors provide can greatly increase a programmers productivity and help them to overcome the initial barriers to learning and using Struts in their applications.

Oracle JDeveloper 10g also provides a huge value add for Struts developers in the new Data Action and Data Page Actions, which integrate Struts with the Model access standards proposed under the Java Community process by JSR 227. In doing so Oracle JDeveloper 10*g*, and the Oracle ADF framework, address one

of the major problems for Struts developers in simplifying the integration of Enterprise Java Beans, database access and other data sources into their applications.

An Introduction to Apache Struts and JDeveloper 10*g*
December 2003
Author: Duncan Mills
Version1.2

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com