YmtdTaglibs

Taglibs

Taglibs are intended to be the savior for JSP. They are billed as the way to allow one to extend JSP so that there is a nice MVC abstraction while still adding functionality to the base "language". This is where Struts has concentrated a good portion of its efforts by providing a nice taglib library for people to use.

The advantage of using taglibs is that they allow you to extend the "language" syntax of JSP to provide the things that are missing from it, but are available in Java. In other words, instead of encouraging people to embed Java code within their pages, this has been now abstracted to encouraging people to embed XML tags in their page. How is this any better than what ColdFusion did with their product? JSP is on the cutting edge of re-inventing the broken wheel. Yea!

This is an example that shows how logic would be embedded into your JSP page. It is borrowed directly from the Struts documentation.

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>


<!-- Is the number guess right? -->
<logic:equal parameter="number" value="7">
  You guessed right! You win a high speed blender!
</logic:equal>


<!-- If the number guessed was wrong -->
<logic:notEqual parameter="number" value="7">
  <!-- Less Than -->
  <logic:lessThan parameter="number" value="7">
         A little higher...
  </logic:lessThan>
  <!-- Greater Than -->
  <logic:greaterThan parameter="number" value="7">
         A little lower...
  </logic:greaterThan>
</logic:notEqual>
```

The same example using Velocity would be written like this:

```
<!-- Is the number guess right? -->
#if ( $number == 7 )
  You guessed right! You win a high speed blender!
#end

<!-- If the number guessed was wrong -->
#if ( $number != 7 )
    <!-- Less Than -->
    #if ( $number < 7 )
         A little higher...
    <!-- Greater Than -->
    #elseif ( $number > 7 )
         A little lower...
    #end
#end
```

Of course this could be written even cleaner as:

```
#if($number == 7)
  You guessed right! You win a high speed blender!
#elseif( $number < 7 )
  lower
#else
  higher
#end
```

This really falls into a preferences situation. In other words, which syntax would someone prefer to use? It seems as though the amount of typing required to implement the taglibs approach would be a major deciding factor for many people. One reason is that the more typing that needs to be done, the more chances for errors. Lets continue with another example taken from the Struts documentation:

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

<%
java.util.ArrayList list = new java.util.ArrayList();
  list.add("First");
  list.add("Second");
  list.add("Third");
  list.add("Fourth");
  list.add("Fifth");
  pageContext.setAttribute("list", list, PageContext.PAGE_SCOPE);
%>

<logic:iterate id="myCollectionElement" name="list">
  Element Value: <bean:write name="myCollectionElement" /><br />
</logic:iterate>
```

It is clear from the Struts example above that the whole strict MVC model has been broken again because a call to `java.util.ArrayList` and creating an Object is embedding Java code within a template. Compound that with the idea that one needs to place that `ArrayList` into the `pageContext` is even more confusing. Not only that but the designer has to remember to use a bunch of cryptic code at the top of the file that makes references to some taglib document as well as declare a prefix attribute. Why do things need to be so overly complicated?

The same example using Velocity would be written like this:

```
#set ( $list = ["First", "Second", "Third", "Fourth", "Fifth"] )

#foreach ( $item in $list )
  Element Value: $item<br />
#end
```

Moving on with examples, we come back to the previous YmtdSampleApplication provided in Jason Hunter's book that was shown before. This time it has been implemented entirely within the Struts framework.

```
<%-- toolview-tag.jsp --%>

<%@ taglib uri="/WEB-INF/struts.tld" prefix="struts" %>

<%
  String title = "Tool Listing";
  String deck = "A list of content creation tools";
  String desc = "Without tools, people are nothing more than animals.";
%>

<%@ include file="/header.jsp" %>

<%@ page session="false" %>
<%@ page errorPage="/errorTaker.jsp" %>

<%-- Fetch the tools array as a request attribute --%>
<jsp:useBean id="tools" class="Tool[]" scope="request"/>

<struts:iterate id="tool" collection="<%= tools %>">
  <HR SIZE=2 ALIGN=LEFT>

  <H3>
  <%-- Automatically HTML-escapes values --%>
  <struts:property name="tool" property="name" />

  <% if (((Tool)tool).isNewWithin(45)) { %>
    <FONT COLOR=#FF0000><B> (New!) </B></FONT>
  <% } else if (((Tool)tool).isUpdatedWithin(45)) { %>
    <FONT COLOR=#FF0000><B> (Updated!) </B></FONT>
  <% } %>

  </H3>
  <A HREF="<struts:property name="tool" property="homeURL"/>">
           <struts:property name="tool" property="homeURL"/></A><BR>

  <%-- Assume don't want HTML in comments --%>
  <struts:property name="tool" property="comments" />

</struts:iterate>

<%@ include file="/footer.jsp" %>
```

At this point, we now have a combination of standard JSP tags as well as Struts specific tags. The use of Struts has appeared to clean things up significantly with regards to embedded scriptlets. Note that quite a few of JSP's warts are still shining through. Is it as easy to grok as the Velocity version?

Velocity comes in with a simpler soluton that is designed around the idea of a few core template tags. A scripting language is something like PHP which may require months of usage to completely master. A template language is something that can be mastered in just a few hours. This is what differentiates Velocity from being a scripting language vs. a template language.

For example, Velocity has the following core template tags (otherwise known as "directives"):

```
#if
#else
#elseif
#foreach
#set
#parse
#include
```

It is possible to add more #directives to Velocity either by adding them directly to the parser or adding them through an API. It is also possible to use a neat feature of Velocity called Velocimacro's which are documented [../user-guide.html#velocimacro here].

Some people suggest that the benefit to JSP and Struts is that it simply extends HTML, which is something that designers already understand. This is a powerful argument. However, the reality is that HTML is not a template language. In other words, there is no logic in HTML. For example, it is not possible to embed conditional statements (like in the very first example above) into HTML.

What this means is that Taglibs allow developers to infinitely extend HTML to become much much more than it previously was. Almost like inventing another scripting language. This is reminiscent the early browser wars where each company was implementing their own browser tags. Netscape went so far as to invent the <blink> tag. What did we learn from that? Is that really a good thing? Sure, standardizing taglibs is a great idea. Will it ever become widely adopted? We sure hope so.

One last point to make about using a HTML syntax is that this really pigeon holes JSP and Struts into simply being a tool for creating dynamic HTML/XML /WML (ie: tag markup) code. Velocity on the other hand is designed to take as input any type of text (Java, SQL, HTML, etc) and output anything as a result of running it through its processing engine.

You make the decision.

YmtdSampleApplication <- Previous | Next -> YmtdEmbeddedUsage