YmtdJavaBeans

JavaBeans

JavaBeans are the way to use Java objects from JSP pages in order to follow the MVC design pattern. The point of doing this is to implement something similar to the Pull methodology. For example:

```
<jsp:useBean id="name" scope="page|request|session|application"
             class="className" type="typeName">
```

Examining the syntax of the above code, the first thing that pops up right away is the use of the scope attribute. How many HTML designers understand the programming concepts of scope? It is safe to suggest that a good portion of web designers barely understand the concept of how a CGI works. By stating this, we are not trying to slight people. Instead, we are simply pointing out that design and software engineering are distinct skill sets. You wouldn't expect a Java programmer to select a print and web safe color palette, would you?

The common response to an argument like this is that the designers should simply ignore these tags and let others define and implement them. The problem with that is that you have now given them the power to accidentally wreck your entire application in such a way that it is very difficult to debug because a complex scope issue might not show up right away.

```
The Java code:

public class HelloBean {
  private String name = "World";

  public void setName(String name) {
    this.name = name;
  }

  public String getName() {
    return name;
  }
}

The JSP code:

<jsp:useBean id="hello" class="HelloBean">
  <jsp:setProperty name="hello" property="*" />
</jsp:useBean>

<HTML>
<HEAD><TITLE>Hello</TITLE></HEAD>
<BODY>
<H1>
Hello, <jsp:getProperty name="hello" property="name" />
</H1>
</BODY>
</HTML>
```

Above, we have a very simple example of using a bean in a page. Pass it some properties and then retrieve the results. This is the right way to do things when using JSP. However, if we look at an example of doing the same exact thing in Velocity, the extra amount of needless typing that one needs to perform to simply retrieve a property seems a just bit absurd. Of course there are always GUI based drag and drop tools to make typing a thing of the past. Really.

There are several commercial solutions available today which provide a nice drag and drop view for doing development with JSP and Struts. However many of these tools are still first generational tools. They typicially only address parts of the problem and require digging down into the nitty gritty stuff when things become difficult or even impossible to do with the GUI (anyone remember a product called Tango?). Often these tools also produce code that is not optimized for heavily hit sites and getting an existing application to scale sometimes requires a complete rewrite. Again, this is not our decision, it is yours. Another item to note here is that these are costly (>$1000/seat) development tools. In this .bomb economy, who really has the money to spend on these tools?

```
The Java code:

context.put ("hello", new HelloBean());

The Velocity code:

$hello.setName("*")
<HTML>
<HEAD><TITLE>Hello</TITLE></HEAD>
<BODY>
<H1>
Hello, $hello.Name
</H1>
</BODY>
</HTML>
```

The example shows the creation of the HelloBean object and then placing it into the Context. Then, during runtime execution of the template, that object is available as a $variable which uses the JavaBean specification to do introspection on the object. For example, Velocity uses Bean style introspection to permit the method call to be shortened from `$hello.getName()` to simply typing what is shown above.

When Velocity is combined with Turbine, the HelloBean object can be added into the Context as a configuration option or it can be added at any point of the processing. This is what provides the "scope" of the object in the Context.

Another "gotcha" with using JavaBeans in JSP is again quoted from Jason's book:

```
One thing to watch out for:  On some servers (including Tomcat 3.2) if
you have a bean with a scope of "session" or "application" and you
change the bean class implementation, you may get a ClassCastException
on a later request. This exception occurs because the generated servlet
code has to do a cast on the bean instance as it's retrieved from the
session or application, and the old bean type stored in the session or
application doesn't match the new bean type expected.  The simplest
solution is to restart the server.
```

You make the decision.

YmtdErrorHandling <- Previous | Next -> YmtdSampleApplication