YmtdSayingHello

Saying Hello

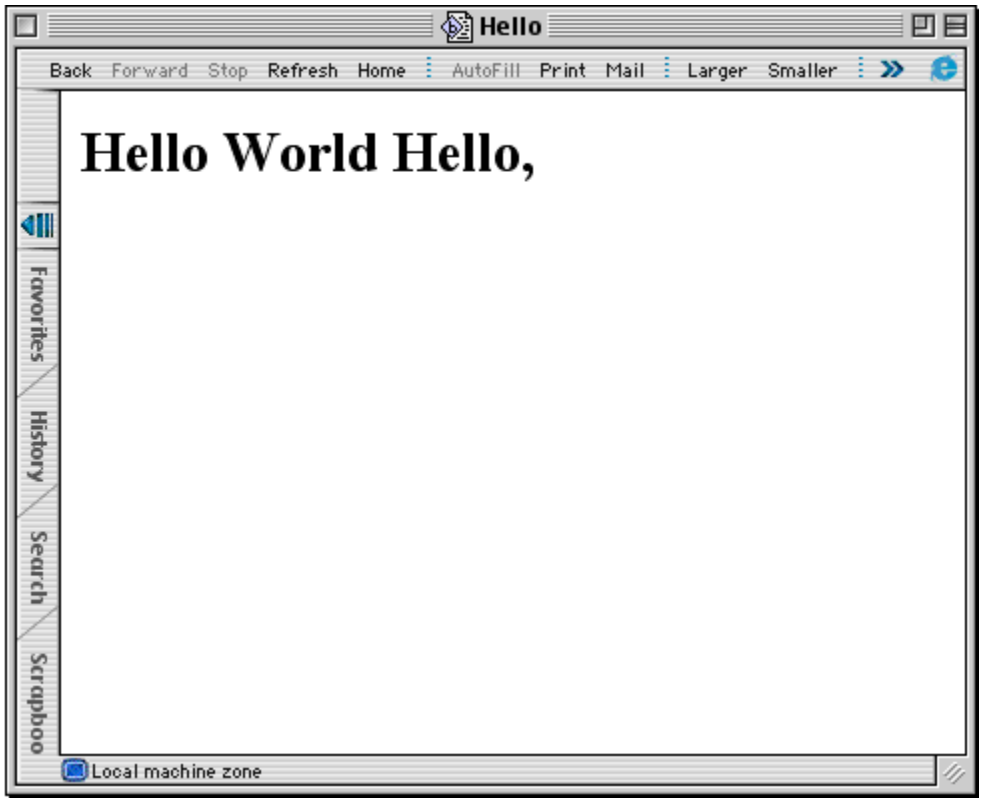
Ok, lets start off with some easy examples. These examples really do not even touch on the basics of correct design. However, they still make good examples because correct design is often harder than showing a simple example. We will show better examples further along in this essay.

For the first example, we show that there are two different approaches of doing the same exact thing using both JSP and Velocity. This is an example of printing out a parameter with JSP:

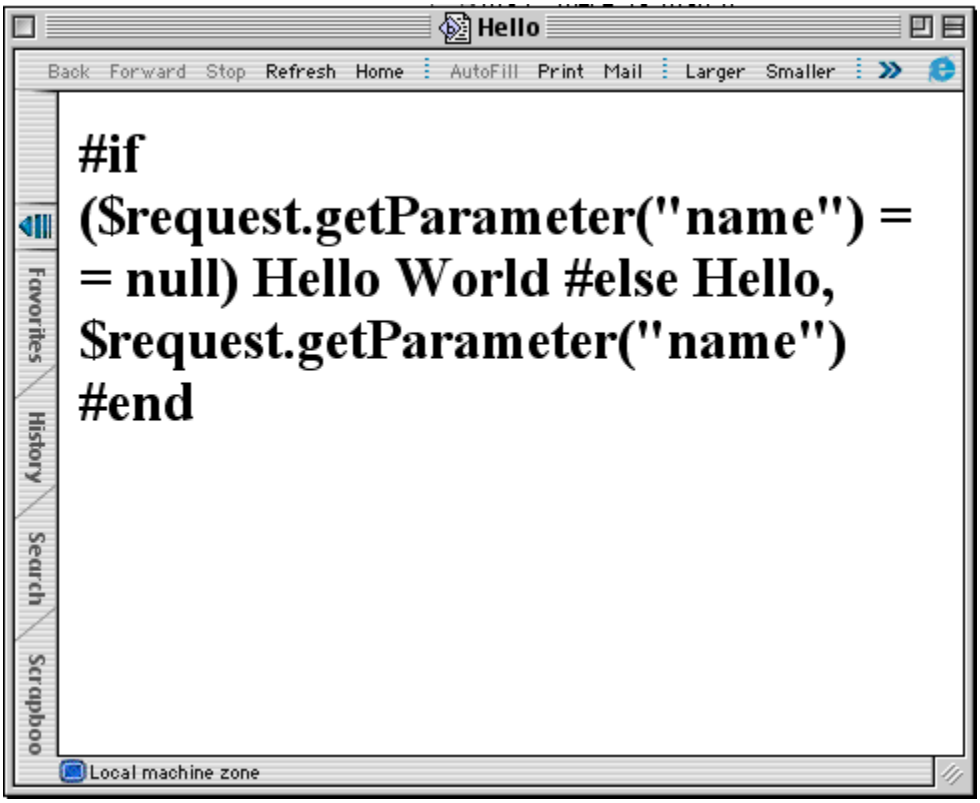
```
<html>
<head><title>Hello</title></head>
<body>
<h1>
<%
if (request.getParameter("name") == null) {
   out.println("Hello World");
}
else {
  out.println("Hello, " + request.getParameter("name"));
}
%>
</h1>
</body></html>
```

This is an example of doing the same thing with Velocity:

```
<html>
<head><title>Hello</title></head>
<body>
<h1>
#if ($request.getParameter("name") == null)
   Hello World
#else
   Hello, $request.getParameter("name")
#end
</h1>
</body></html>
```



Hello JSP Screen shot



Hello Velocity Screen shot

These are screen shots showing the above example when loaded directly into the browser. They demonstrate the idea that one cannot easily look at what the code is doing in a browser when using JSP.

The primary difference between the two is the way that output is performed. With JSP, one needs to embed "code" within `<% %>` tags and for Velocity, one does not need to embed "code" within tags. One can simply use the Velocity Template Language (VTL) directly in any portion of the template.

The benefit (and detriment) of the embedded code is that the JSP code within a page will not show up when the file is simply loaded into the browser. On the other hand, there might be a case where one may desire it to show up (for example, in debugging).

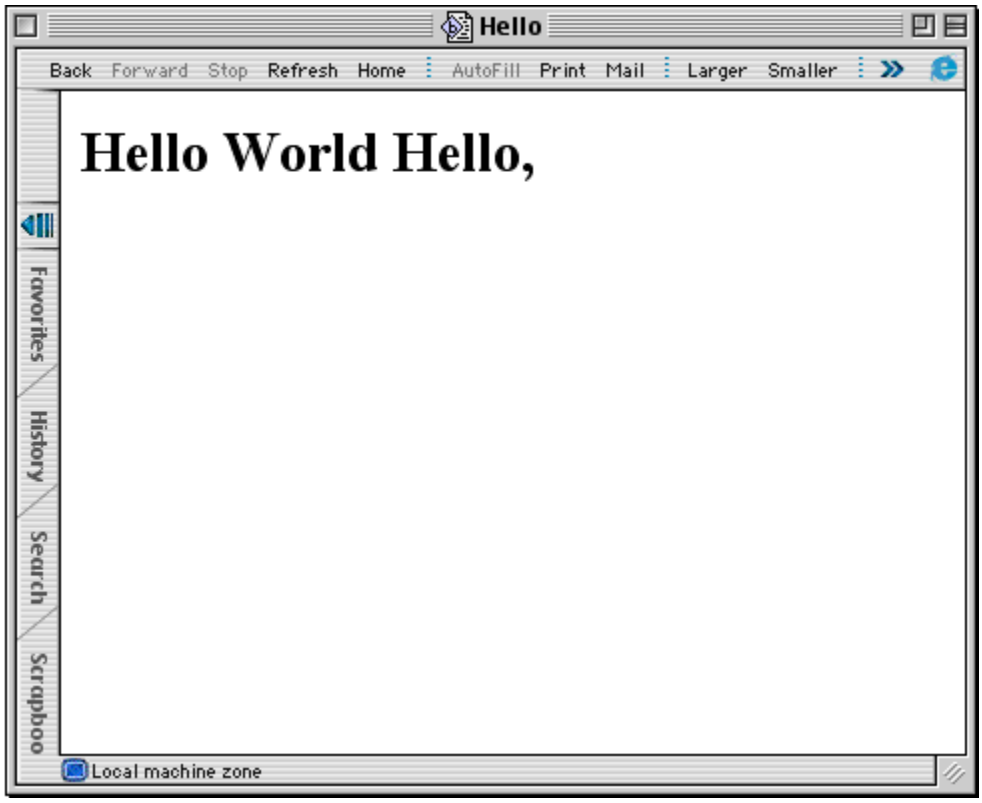
Another issue with JSP is the fact that even the most basic examples start to blow the whole MVC paradigm right out of the water. The reason is that embedding HTML code within Java code is a bad design decision because it makes it more difficult to modify the look and feel of an application at a later date. It also destroys the concept of MVC separation where the View (ie: HTML) display of the page is separated from the Model and Controller. For example, if just the word "Hello" needed to be in bold, we would need to embed `<b> </b>` tags into the `out.println()` statement.

Of course, *people in the know*, would recommend that we write JSP like this:

```
<html>
<head><title>Hello</title></head>
<body>
<h1>
<% if (request.getParameter("name") == null) %>
          Hello World
<% else %>
          Hello, <% request.getParameter("name"); %>
</h1>
</body></html>
```

Or with Struts:

```
<html>
<head><title>Hello</title></head>
<body>
<h1>
<logic:notPresent parameter="name">
          Hello World
</logic:notPresent>
<logic:present parameter="name">
          <bean:parameter id="name" name="name"/>
          Hello, <bean:write name="name"/>
</logic:present>
</h1>
</body></html>
```



Hello JSP Screen shot

This is the new JSP screen shot showing the above example when loaded directly into the browser.

The point that needs to be made is that in order to make JSP "pure", one really needs to jump through hoops. The example above looks very similar to the Velocity example. However, one still needs to embed the necessary
<% %> tags everywhere. More typing == more chances for mistakes! There is also a bit of a disconnect as to when the ";" needs to be added and when it does not. With Velocity, the rule is that you place a # before a directive and a $ before a member in the Context.

As you can see from the example image, there is now a bit more information in the displayed page, except that it is also missing all of the logic which was used to build the page. If one views the equivalent Velocity template directly in the browser, without it being rendered first, all of the logic in the template remains visible. The advantage is that it is easier to debug problems.

You make the decision.

YouMakeTheDecision <- Previous | Next -> YmtdGeneration