

JSPs

JSP/ASP/PHP Theory

CGI world

- CGI -- looks like code, possibly with some HTML thrown in
- Servlet -- same thing
- Flexible but difficult (expensive)

JSP/ASP/PHP world

- Have HTML, with little bits of "markup" thrown in to do a little database lookup or other small computation.
- HTML is easier to deal with for most purposes.
- No re-compile is required -- edit the HTML and go
- Works well if the problem is small -- simpler than writing (and understanding) a CGI
- Not appropriate for large computations.

Separate Computation and Presentation

1. Programming -- "business logic"

- Code, algorithms, database operations -- things that look like code

2. HTML -- "presentation"

- HTML, images, tables -- complex in its way, but it's not code

Result

- Keep code and presentation separate
- Servlets, CGI's, etc. for heavy computation
- JSP etc. for presentation -- JSPs etc. refer to servlet/CGI system for complex computations

Cheapness hypothesis

- Perhaps this allows you to have just 2 coders and 10 HTML people -- save money
- It's not clear that this is true -- the HTML stuff is also expensive.
- But perhaps it's still better to keep the two activities logically separate.

HTML markup technologies

- ASP -- Microsoft proprietary -- looks like visual basic
- PHP -- open source -- works well for connecting a web site to a database such as the open source MySQL
- JSP -- java answer to ASP -- uses Java/servlets for the back-end

JavaBean

- Simple object that collects some named properties
- Has a no-argument constructor
- The properties are accessible with get/set messages
- For a "foo" property, responds to getFoo(), setFoo()

MagicBean

```
// MagicBean.java
/*
  A simple bean that contains a single "magic" string.
*/
public class MagicBean {
    private String magic;

    public MagicBean(String string) {
        magic = string;
    }

    public MagicBean() {
        magic = "Woo Hoo";// default magic string
    }

    public String getMagic() {
        return(magic);
    }

    public void setMagic(String magic) {
        this.magic = magic;
    }
}
```

HelloJSP

1. Put an attribute on the request for the JSP to see -- a simple way to communicate to the JSP
`request.setAttribute("foo", "Binky");`
2. Put a bean on the request. The JSP can access the bean properties with the `useBean` tag below
`request.setAttribute("bean", bean);`
3. Use `rd.include` to paste in the output of `"/hello.jsp"`

HelloJSP Code

```
// HelloJSP.java
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloJSP extends HttpServlet
{

    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");

        // Generate the HTML part of the response
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
    }
}
```

```

String title = "Hello JSP";

out.println("<title>" + title + "</title>");
out.println("</head>");
out.println("<body bgcolor=white>");

out.println("<h1>" + title + "</h1>");

out.println("<p>Let's see what Mr. JSP has to contribute...");

// 0. There may already be form bindings on our request --
// the JSP can access those with request.getParameter("name")

// 1. Put an arbitrary binding on the request for the JSP
// to see -- JSP accesses with getAttribute()
request.setAttribute("foo", "Binky");

// 2. Put a bean on the request for the JSP to see
MagicBean bean = new MagicBean("Peanut butter sandwiches!");
request.setAttribute("bean", bean);

// 3. Paste in the output of the JSP
RequestDispatcher rd = getServletContext().
    getRequestDispatcher("/hello.jsp");
rd.include(request, response);

out.println("<hr>");
out.println("</body>");
out.println("</html>");

}
}

```

JSP

up in the /servlet/webapps/dir/ directory next to the index.html
 JSPs are picky about syntax -- don't forget the trailing .../> at the end of the tags

1. request.getParameter("db")

Pull a binding off the request
 Returns "null" if the binding is not present

2. request.getAttribute("foo")

Pull an attribute off the request (set earlier with a to setAttribute() in the servlet)

3. jsp:usebean

```
<jsp:usebean id="bean" scope="page" class="MagicBean" />
```

Pull in the bean, prep for getProperty

```
<jsp:getProperty name="bean" property="magic" />
```

Pull the given property out of the given bean

hello.jsp

```

<!-- hello.jsp -->
<%@ page language="java" %>
<hr>
<h3>JSP</h3>
<p>Pulling stuff off the request...

<ul>
  <li>The db parameter:<%= request.getParameter("db") %>
<!-- this only works if there is a ?db=xxx binding on the request -->

  <li>The foo attribute:<%= request.getAttribute("foo") %>
  <li>Remote host:<%= request.getRemoteHost() %>
</ul>
<p>
Behold -- I bring forth the magic property from the Magic Bean...

<jsp:usebean id="bean" scope="page" class="MagicBean" />

<table border=1>
<tr>
<td bgcolor=pink>
  <jsp:getProperty name="bean" property="magic" />
</td>
</tr>
</table>

```

