



Java Web: JSP & JSTL

CS544: Enterprise Architecture

Overview

- In this lecture will look at some of the most common Java technologies for dynamic html generation.
- We will follow the chronological timeline of different technologies developed for this purpose, from servlets, to JSP, to JSTL and EL – the nature of life is to grow

Servlets

Other technologies
built on top

```
public class ServletDemo extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException {
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Hello Example</title>");
        out.println("</p>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello Servlet</h1>");
        out.println("<p>Current date & time on server:</p>");
        Date now = new Date();
        if (now.after(new Date(0))) {
            out.println(now);
        }
        out.println("</body>");
        out.println("</html>");
    }
}
```

Web.xml

Inside project's
/WEB-INF/

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
```

```
  <servlet>
    <servlet-name>Servlet Demo</servlet-name>
    <servlet-class>demo.ServletDemo</servlet-class>
  </servlet>
```

```
  <servlet-mapping>
    <servlet-name>Servlet Demo</servlet-name>
    <url-pattern>/servlet</url-pattern>
  </servlet-mapping>
</web-app>
```

Can also be done
with @WebServlet

JSTL and EL Example

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ page import="java.util.Date"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>JSP Hello Example</title>
  </head>
  <body>
    <h1>Hello JSP</h1>
    <p>Current date & time on server:
      <c:set var="old" value="<%= new Date(0)%>" scope="page"/>
      <c:set var="now" value="<%= new Date()%>" scope="page"/>
      <c:if test="${now gt old}">
        ${now}
      </c:if>
    </p>
  </body>
</html>
```

This is a Model1 page
(everything in one)

Model2 (MVC) is much
preferred

JSP & JSTL:

JAVA SERVER PAGES

Java Server Pages (JSP)

```
<%@ page import="java.util.Date"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>JSP Hello Example</title>
</head>
<body>
<h1>Hello JSP</h1>
<p>Current date & time on server:
    <!-- Date now = new Date(); -->
    <% if (now.after(new Date(0))) { %>
        <%= now %>
    <% } // end if %>
</p>
</body>
</html>
```

Compiled into servlet
when run first time

This is a Model1 page
(everything in one)

Model2 (MVC) is much
preferred

Controller Servlet

```
@WebServlet(name = "Hello", urlPatterns = {"/Hello"})
public class Hello extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        request.setAttribute("now", new Date());
        request.setAttribute("epoch", new Date(0));

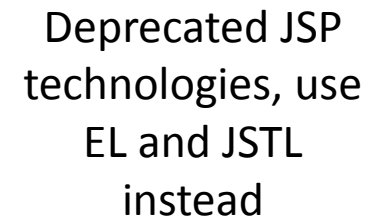
        ServletContext context = getServletContext();
        String jsp = "/hello.jsp";
        RequestDispatcher dispatcher = context.getRequestDispatcher(jsp);
        dispatcher.forward(request, response);
    }
}
```


View JSP

```
<%@page import="java.util.Date"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello JSP</h1>

        <jsp:useBean id="now" class="java.util.Date" scope="request" />
        <jsp:useBean id="epoch" class="java.util.Date" scope="request" />
        <% if (now.after(epoch)) {%>
            <%= now %>
        <% } //end if %>

    </body>
</html>
```



Deprecated JSP
technologies, use
EL and JSTL
instead

Elements in a JSP Page

- JPS Pages can have:
 - Directive elements
 - Scripting elements (**deprecated**)
 - Action elements (**deprecated**)
- Modern JSP uses:
 - Expression Language (EL)
 - JSP Standard Tag Library (JSTL)

Directive Elements

`<%@ directive attributes %>`

- XML compliant alternative:

`<jsp:directive attributes />`

- Examples:

`<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>`

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

`<%@ page import="java.util.Date"%>`

Forward and Include Actions

- Include action is at runtime

```
<jsp:include page="header.jsp">
```

As Opposed to the
include directive which
is at compile time

- Forward gives control to another resource:

```
<jsp:forward page="login.jsp" />
```

Stays within same
request

JSP & JSTL:

EL & JSTL

EL and JSTL

- Newer alternative to scriptlets
 - Expression Language (EL)
 - JSP Standard Tag Library (JSTL)

Expressions Language

- `${expression}`

- `${object.value}`

Uses `getValue()`
method

Easier for non Java
People?

- Supports Operators:

- Arithmetic: `+`, `-`, `*`, `/` (div), `%` (mod)
- Relational: `==` (eq), `!=` (ne), `<` (lt), `>` (gt), `<=` (le), `>=` (ge)
- Logical: `&&` (and), `||` (or), `!` (not)
- Other: `()`, empty, `[]`

EL Scopes

- When you use a variable in an EL expression it looks for a key by that name in the scopes:
 - pageScope
 - requestScope
 - sessionScope
 - applicationScope
- You cannot access 'local' variables declared in scriptlets or `<%! declarations %>`

JSP Standard Tag Library (JSTL)

- Better alternative to scriptlets
- Extendable (create your own tags)
- Standard tags libraries for:

| Area | URI | Prefix |
|-------------------------------|---|--------|
| Core | http://java.sun.com/jsp/jstl/core | c |
| XML Processing | http://java.sun.com/jsp/jstl/xml | x |
| I18N & Formatting | http://java.sun.com/jsp/jstl/fmt | fmt |
| Database access | http://java.sun.com/jsp/jstl/sql | sql |
| String manipulation functions | http://java.sun.com/jsp/jstl/functions | fn |

JSTL and EL Example

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ page import="java.util.Date"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>JSP Hello Example</title>
  </head>
  <body>
    <h1>Hello JSP</h1>
    <p>Current date & time on server:
      <c:set var="old" value="<%= new Date(0)%>" scope="page"/>
      <c:set var="now" value="<%= new Date()%>" scope="page"/>
      <c:if test="${now gt old}">
        ${now}
      </c:if>
    </p>
  </body>
</html>
```

This is a Model1 page
(everything in one)

Model2 (MVC) is much
preferred

Controller Servlet

```
@WebServlet(name = "Hello", urlPatterns = {"/Hello"})
public class Hello extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        request.setAttribute("now", new Date());
        request.setAttribute("epoch", new Date(0));

        ServletContext context = getServletContext();
        String jsp = "/hello.jsp";
        RequestDispatcher dispatcher = context.getRequestDispatcher(jsp);
        dispatcher.forward(request, response);
    }
}
```

JSP with EL and JSTL

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

JSTL taglib

```
<%@page import="java.util.Date"%>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
    <title>JSP Page</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1>Hello JSP</h1>
```

```
    <c:if test="${now gt old}">
```

```
      ${now}
```

```
    </c:if>
```

EL and JSTL

```
  </body>
```

```
</html>
```

Core Tags

| Tag | Description |
|---------------|---|
| <c:out> | Output result of evaluating an expression |
| <c:set> | Sets a key in one of the scopes |
| <c:remove> | Removes a key in one of the scopes |
| <c:catch> | Catches any java.lang.Throwable inside it |
| <c:if> | Executes content if test is true |
| <c:choose> | For a multi condition if (if/else) |
| <c:when> | Specify one (or more) conditions inside a <c:choose> |
| <c:otherwise> | Specify what should happen if none of the <c:when> are true |
| <c:import> | Ability to import data from a url (does http request) |
| <c:forEach> | Basic Foreach |
| <c:forTokens> | Iterates over tokens separated by the supplied delimiter |
| <c:redirect> | Redirects to a URL (HTTP 3xx) |
| <c:url> | Encodes a URL with optional parameters |
| <c:param> | To provide parameters to the url and import tags |

Formatting Tags

| Tag | Description |
|-----------------------|---|
| <fmt:message> | Gets message from bundle (for internationalization) |
| <fmt:param> | To give parameter to <fmt:message> |
| <fmt:bundle> | Specify a resource bundle to use |
| <fmt:setLocale> | Sets the locale |
| <fmt:requestEncoding> | Sets the request's character encoding |
| <fmt:timeZone> | Specifies the timezone for formatting |
| <fmt:setTimeZone> | Stores the specified timezone into a variable |
| <fmt:formatNumber> | Format a number based on the set locale |
| <fmt:parseNumber> | Parses a string of a (number, percentage, currency) |
| <fmt:formatDate> | Formats dates and times in a locale sensitive way |
| <fmt:parseDate> | Parses string representations of times and dates |

Spring Form Tag Lib

```
<form:form commandName="user">  
  <table>  
    <tr>  
      <td>First Name:</td>  
      <td><form:input path="firstName" /></td>  
    </tr>  
    <tr>  
      <td>Last Name:</td>  
      <td><form:input path="lastName" /></td>  
    </tr>  
    <tr>  
      <td colspan="2">  
        <input type="submit" value="Save Changes" />  
      </td>  
    </tr>  
  </table>  
</form:form>
```

Object placed in the SpringMVC Model

Path attribute specifies properties on user object

Spring From Tags

| Tag | Description |
|--------------------|---|
| <form:form> | Parent element for all other spring form elements |
| <form:input> | Typical input element (now also supports HTML5 types) |
| <form:checkbox> | Checkbox element |
| <form:radiobutton> | Radio button element |
| <form:password> | Password element, optional showPassword attribute |
| <form:select> | Dropdown |
| <form:option> | You can specify a single option in the dropdown |
| <form:options> | Or specify a list of items |
| <form:textarea> | Textarea |
| <form:hidden> | Hidden field |
| <form:errors> | To show validation errors |

Spring Tags

- There are also spring tags mostly used for working with variables inside contexts
 - Putting a variable inside HTML, JavaScript, URL

```
<%@taglib prefix="spring" uri="http://www.springframework.org/tags" %>
```

```
...
```

```
<spring:url value="/something" var="url" htmlEscape="true"/>
```

```
<a href="${url}">...</a>
```

```
<!-- alternate way without spring tags -->
```

```
<c:url value="/something" var="url"/>
```

```
<a href="<c:out value='${url}'/>">...</a>
```

<http://docs.spring.io/spring/docs/current/spring-framework-reference/html/spring-tld.html>

JSP & JSTL

DEPRECATED JSP

Scripting Elements deprecated

- Examples of these 3 on the following slides
 - Declarations
 - Used to declare variables (in the 'method' scope)
 - Scriptlets
 - Contain Java Code (perform actions)
 - Expressions
 - Evaluate expressions

Declaration Example deprecated

```
<%! int a = 5 %>
```

```
<%! List<Item> items = new ArrayList() %>
```

- Alternate XML form:

```
<jsp:declaration>int a = 5</jsp:declaration>
```

```
<jsp:declaration><![CDATA[
```

```
    List<Item> items = new ArrayList()
```

```
]]></jsp:declaration>
```

Scriptlet Examples

deprecated

```
<% for (Item item : items) { %>
    <p> do something with the item </p>
<% } // end for each %>
```

- Alternate XML syntax:

```
<jsp:scriptlet>for (Item item : items){</jsp:scriptlet>
    <p> do something with the item </p>
<jsp:scriptlet>} // end for each</jsp:scriptlet>
```

Expression Examples deprecated

```
<%= item.getPrice() * 0.85 %>
```

- Alternate XML Syntax:

```
<jsp:expression>
```

```
    item.getPrice() * 0.85
```

```
</jsp:expression>
```

Action Elements

deprecated

- Pure XML alternative to scriptlets

page, request,
session, application

```
<jsp:useBean id="now" class="java.util.Date" scope="request" />  
<jsp:getProperty name="now" property="time" />  
<jps:setProperty name="now" property="time" value="0" />
```

The EL replaces these

Active Learning

- What is wrong with the following?

```
<%! Date now = new Date(); %>  
${now}
```

- Why are scriptlets deprecated?

Summary

- Java Web Technologies consist of:
 - Servlets: Classes that handle request / response
 - JSP: HTML like pages compiled into servlets
 - EL: Expression Language to easily insert values
 - JSTL: Standard Tag Libraries for common actions
- The more modern parts are geared more and more towards separation of concerns

Main Point

- It is important to know the basics of Servlets, JSP, JSTL and EL in order to make modern (web) applications with Java. Although there are many different parts, they all have the same basis (web development)
- *Science of Consciousness*: Harmony exists in diversity; in cosmic consciousness we still experience all the different parts, but at the same time permanently experience the bliss at the basis.