Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

Adv. Web Technology
3) Java Server Pages

Emmanuel Benoist

Fall Term 2016-17

Computer Science Division

Presentation of the Course I

- Introduction
  - what is JSP?
  - Translation
  - Execution
- Basics of the language
  - Objects and Scope
  - Directives and Actions
  - Scripting elements
- Expression Language
- Tag Handlers

Introduction

what is JSP?

What is JSP?

- **Servlet:**
  - Write HTML inside a Java program
  - A lot of

```
out.println("<a href="/servlet/MyClass>Some Text↘
→</a>");
```

- **Java Server pages**
  - Write insert Java inside an HTML Page (like in PHP)
  - The corresponding Servlet is automatically generated

Example of a JSP 1.2 page

- **The NumberGess example from Tomcat**

```
<%@ page import = "num.NumberGuessBean" %>

<jsp:useBean id="numguess" class="num.NumberGuessBean" scope="session" />
<jsp:setProperty name="numguess" property="*" />

<html>
<head><title>Number Guess</title></head>
<body bgcolor="white">
<font size=4>
<% if (numguess.getSuccess()) { %>
  Congratulations! You got it.
  And after just <%= numguess.getNumGuesses() %> tries.<p>
  <% numguess.reset(); %>
  Care to <a href="numguess.jsp">try again</a>?
<% } else if (numguess.getNumGuesses() == 0) { %>
  Welcome to the Number Guess game.<p>
  I'm thinking of a number between 1 and 100.<p>
  <form method=get>
  What's your guess? <input type=text name=guess>
  <input type=submit value="Submit">
  </form>
<% } else { %>
  Good guess, but nope. Try <b><%= numguess.getHint() %></b>.
  You have made <%= numguess.getNumGuesses() %> guesses.<p>
  I'm thinking of a number between 1 and 100.<p>
  <form method=get>
  What's your guess? <input type=text name=guess>
  <input type=submit value="Submit">
  </form>
<% } %>
```

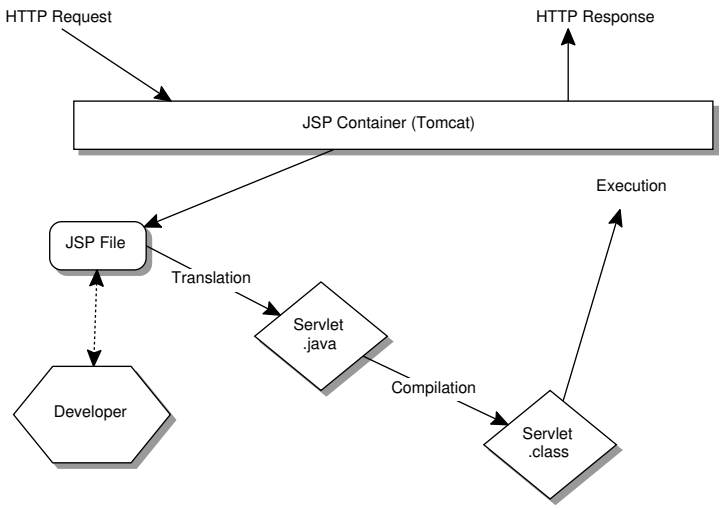

Example of a JSP 2 page

- **The Book example from Tomcat**

```
<%@ taglib prefix="my" uri="/WEB-INF/jsp2/jsp2-example-taglib.tld" %>
<html>
  <head>
    <title>JSP 2.0 Examples - Book SimpleTag Handler</title>
  </head>
  <body>
    <h1>JSP 2.0 Examples - Book SimpleTag Handler</h1>
    <hr>
   <br>
    <b><u>Result:</u></b><br>
    <my:findBook var="book" />
    <table border="1">
        <thead>
        <td><b>Field</b></td>
        <td><b>Value</b></td>
        <td><b>Capitalized</b></td>
    </thead>
    <tr>
        <td>Title</td>
        <td>${book.title}</td>
        <td>${my:caps(book.title)}</td>
    </tr>
    <tr>
        <td>Author</td>
        <td>${book.author}</td>
        <td>${my:caps(book.author)}</td>
    </tr>
    <tr>
        <td>ISBN</td>
        <td>${book.isbn}</td>
```

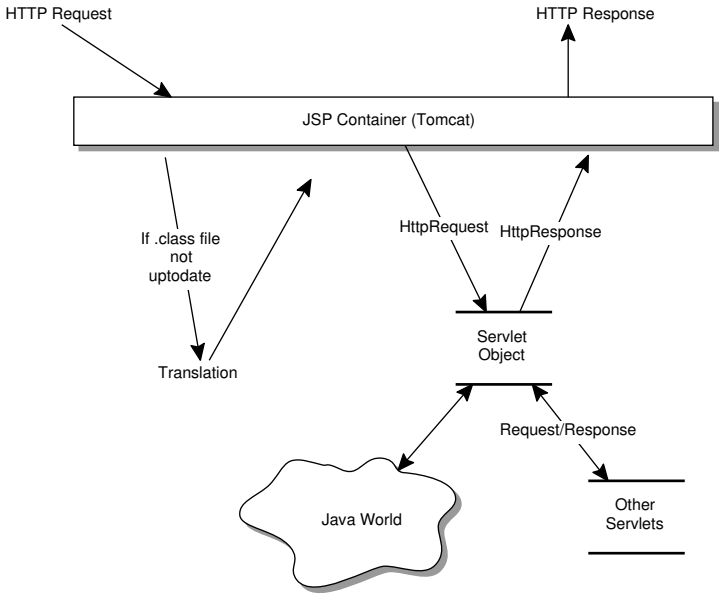

Translation

Translation



Execution

Execution



Basics of the language

Objects and Scope

Objects and Scope

- **A JSP Page can create and/or access some java objects**
  - Some are created implicitly
  - Some are created explicitely through actions
  - Some can be created directly using scripting code (less common)
- **Every object has a scope:**
  - page can be accessed from within the page
  - `request` from every page processing this request
  - `session` from every page processing a request in the current session
  - `application` from everywhere in the application (not user dependent)

Directives and Actions

Directives and Actions

Two types of elements in a JSP page

- **Directives**
  - Global information,
  - valid independent of any specific request
  - Instructions for compilation/translation
  - A syntax of the form:

    <%@ directive . . . %>
- **Actions**
  - Dependent on the details of the specific request
  - Information on the request processing phase
  - The XML syntax for actions:

    <mytag attr1="attribute␣value" . . .> body </mytag>

    or (if no body)

    <mytag attr1="attribute␣value" . . . />

Directives I

- **Syntax**

  <%@ directive attribute="value" attr2="val2"%>

- **The page directive**
  Information or compilation instruction for the page

  ```
  <%@ page import="java.util.*, myPackage.MyClass"
          contentType="text/plain"
          info="message"%>
  ```

- **The include directive**
  Insert the content of a file in the page

  <%@ include file="relative url" %>

Directives II

- **The TagLib directive**
  The taglib directive enables you to create your own custom tags

  ```
  <%@ taglib uri="http://www.bigbank.com/spTags" prefix↘
  →="accttags" %>
  <accttags:specialTag attribute="value"> ... </accttag:↘
  →specialTag>
  ```


Scripting elements

Scripting elements (JSP1.2)

- **Declarations**
  Used to declare constructs that are available to all other scripting elements

  ```
  <%! int i= 0; %>
  <%! int a, b; double c;%>
  <%! Circle a = new Circle(2.0);%>
  ```

- **Scriptlet**
  Program fragments, can also be used to do things like iterations and conditional execution of other elements

  ```
  <% String name = null;
     if (request.getParameter("name")==null){ %>
        . . .
  <% } %>
  ```

Scripting elements (JSP1.2) II

- **Expressions**
  - Complete expressions that get evaluated at response time
  - Commonly the result is converted into a string and then inserted into the output

```
<%= Math.sqrt(2)%>
<%= items[i] %>
<%= a + b + c %>
<%= new java.util.Date()%>
```


Expression Language

Expression Language

- **Include some results inside the page**
  - Not in Java
  - Can access to java objects
  - Syntax out of Java
  - Expression `${ valid expression language expression}`
  - Is inserted in the jsp file

```
<tr>
   <td>\${1}</td>
   <td>${1}</td>
 </tr>
 <tr>
   <td>\${1 + 2}</td>
   <td>${1 + 2}</td>
 </tr>
 <tr>
   <td>\${1.2 + 2.3}</td>
   <td>${1.2 + 2.3}</td>
 </tr>
 <tr>
   <td>\${1.2E4 + 1.4}</td>
```


Expression Language

- **Basic arithmetic**
  - `${1.2 + 2.3}`
  - `${1.2E4 + 1.4}`
  - `${10 mod 4}`
  - `${(1==2) ? 3 : 4}`
- **Basic comparisons**
  - `${1 < 2}`
  - `${1 lt 2}`
  - `${4.0 ge 3}`
  - `${4 <= 3}`
  - also `==`, `!=`, `gt`, `le`

Implicite Objects

- `pageContext` - the PageContext object
- `pageScope` - a Map that maps page-scoped attribute names to their values
- `requestScope` - a Map that maps request-scoped attribute names to their values
- `sessionScope` - a Map that maps session-scoped attribute names to their values
- `applicationScope` - a Map that maps application-scoped attribute names to their values
- `param` - a Map that maps parameter names to a single String parameter value
- `paramValues` - a Map that maps parameter names to a String[] of all values for that parameter
- `header` - a Map that maps header names to a single String header value
- `headerValues` - a Map that maps header names to a String[] of all values for that header

Forms

- **Implicit Object** `param`

```
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/↘
→functions" %>
...
<form action="implicit-objects.jsp" method="GET">
          foo = <input type="text" name="foo" value="${↘
          →fn:escapeXml(param["foo"])}">
          <input type="submit">
</form>
<table>
   <tr>
         <td>The foo parameter (escaped from XML tags)</↘
         →td>
         <td>${fn:escapeXml(param["foo"])} </td>
    </tr>
</table
```


Functions

- **Simple functions**
  - Functions are defined by tag libraries
  - They are implemented by a Java programmer as static methods.

```
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/↘
→functions" %>
<%@ taglib prefix="my" %uri="http://tomcat.apache.org/jsp2↘
→-example-taglib"%>
...
Print parameter foo: ${fn:escapeXml(param["foo"])} <br↘
→>
Reverse foo: ${my:reverse(fn:escapeXml(param["foo"]))}<br>
Reverse x2 : ${my:reverse(my:reverse(fn:escapeXml(param["foo"↘
→"])))}<br>
Count vowels ${my:countVowels(fn:escapeXml(param["foo"]))}
```


The corresponding java

```
package jsp2.examples.el;
import java.util.Locale;
public class Functions {
    public static String reverse( String text ) {
        return new StringBuilder( text ).reverse().toString();
    }
    public static int numVowels( String text ) {
        String vowels = "aeiouAEIOU";
        int result = 0;
        for( int i = 0; i < text.length(); i++ ) {
            if( vowels.indexOf( text.charAt( i ) ) != -1 ) {
                result++;
            }
        }
        return result;
    }
}
```

Tag Handlers

Hello World

- **In the JSP file**

```
<%@ taglib prefix="mytag" uri="/WEB-INF/jsp2/jsp2-example-taglib.tld" %>
...
<mytag:helloWorld/>
```

- **The corresponding Java file**

```
package jsp2.examples.simpletag;
import java.io.IOException;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.SimpleTagSupport;
/**
 * SimpleTag handler that prints "Hello, world!"
 */
public class HelloWorldSimpleTag extends SimpleTagSupport {
    @Override
    public void doTag() throws JspException, IOException {
        getJspContext().getOut().write( "Hello, world!" );
    }
}
```


The repeat tag

- **In the JSP**

```
<%@ taglib prefix="mytag" uri="/WEB-INF/jsp2/jsp2-example-taglib.tld" %>
...
 <mytag:repeat num="5">
      Invocation ${count} of 5<br>
  </mytag:repeat>
```

- **In java**

```
public class RepeatSimpleTag extends SimpleTagSupport {
    private int num;
    @Override
    public void doTag() throws JspException, IOException {
        for (int i=0; i<num; i++) {
            getJspContext().setAttribute("count", String.valueOf( i + 1 ) );
            getJspBody().invoke(null);
        }
    }
    public void setNum(int num) { this.num = num; } }
```


Tag file (JSP, no java)

- **The JSP file**

```
<%@ taglib prefix="tags" tagdir="/WEB-INF/tags" %>
<html>
...
    <h2>Products</h2>
    <tags:displayProducts>
      <jsp:attribute name="normalPrice">
        Item: ${name}<br/>
        Price: ${price}
      </jsp:attribute>
      <jsp:attribute name="onSale">
        Item: ${name}<br/>
        <font color="red"><strike>Was: ${origPrice}</strike></font><br/>
        <b>Now: ${salePrice}</b>
      </jsp:attribute>
    </tags:displayProducts>
    <hr>
    <h2>Products (Same tag, alternate style)</h2>
    <tags:displayProducts>
      <jsp:attribute name="normalPrice">
        <b>${name}</b> @ ${price} ea.
      </jsp:attribute>
      <jsp:attribute name="onSale">
        <b>${name}</b> @ ${salePrice} ea. (was: ${origPrice})
      </jsp:attribute>
    </tags:displayProducts>
```

Example II

- **The Tag file (displayProducts.tag)**

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ attribute name="normalPrice" fragment="true" %>
<%@ attribute name="onSale" fragment="true" %>
<%@ variable name-given="name" %>
<%@ variable name-given="price" %>
<%@ variable name-given="origPrice" %>
<%@ variable name-given="salePrice" %>

<table border="1">
  <tr>
    <td>
      <c:set var="name" value="Hand-held Color PDA"/>
      <c:set var="price" value="$298.86"/>
      <jsp:invoke fragment="normalPrice"/>
    </td>
    <td>
      <c:set var="name" value="4-Pack 150 Watt Light Bulbs"/>
      <c:set var="origPrice" value="$2.98"/>
      <c:set var="salePrice" value="$2.32"/>
      <jsp:invoke fragment="onSale"/>
    </td>
    <td>
      <c:set var="name" value="Digital Cellular Phone"/>
      <c:set var="price" value="$68.74"/>
      <jsp:invoke fragment="normalPrice"/>
    </td>
    <td>
      <c:set var="name" value="Baby Grand Piano"/>
      <c:set var="price" value="$10,800.00"/>
      <jsp:invoke fragment="normalPrice"/>
```


Conclusion

- **Writing Java inside JSP is prohibited**
  - It means mearging of Layer "Presentation" and layer "Business Logic"
- **Writing your own Java Libs is useless**
  - There are existing frameworks providing usefull taglibs and libraries
  - Reuse of such tags saves a lot of time (and money)
- **Example of Framework: Java Server Faces**
  - Standard developed by Sun together with a standard implementation
  - Other implementations are possible (MyFaces for example)

Source

- **JSP examples from Tomcat**
  `http://localhost:8080/examples/jsp/`
  Once Tomcat is installed