COMP201 Java Programming

Part III: Advanced Features

Topic 16: JavaServer Pages (JSP)

Servlets and JavaServer Pages (JSP) 1.0:

A Tutorial

http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/Servlet-Tutorial-Intro.html

Objective and Outline

- Objective:
  - Introduction to JSP
- Outline:
  - Introduction and overview
  - JSP constructs
    - JSP scripting elements
    - JSP directives
    - JSP actions
  - JSP and Beans
    - Beans
    - Using beans in JSP

Introduction and Overview

- Server-side java:
  - Scheme 1:
    - HTML files placed at location for web pages
    - Servlets placed at special location for servlets
    - Call servlets from HTML files
  - Scheme 2:
    - JSP: HTML + servlet codes + jsp tags
    - Placed at location for web pages
    - Converted to normal servlets when first accessed
      - Might take some time
      - Good idea for programmer to access it for the first time

Introduction and Overview

- Example: Hello.jsp

```
<HTML>
<HEAD><TITLE>JSP Test</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">

<H1>JSP Test</H1>
Time: <%= new java.util.Date() %>

</BODY>
</HTML>
```

  - **<H1>JSP Test</H1>: normal HTML**
  - <%, %>: special JSP tags
  - **new java.util.Date()**: java code

Introduction and Overview

- Ingredients of a JSP
  - Regular HTML
    - Simply "passed through" to the client by the servlet created to handle the page.
  - JSP constructs
    - **Scripting elements** let you specify Java code that will become part of the resultant servlet,
    - **Directives** let you control the overall structure of the servlet, and
    - **Actions** let you specify existing components that should be used, and otherwise control the behavior of the JSP engine
      - JavaBeans: a type of components frequently used in JSP

Outline

- Outline:
  - Introduction and overview
  - JSP constructs
    - JSP scripting elements
    - JSP directives
    - JSP actions
  - JSP and Beans
    - Beans
    - Using beans in JSP

JSP Scripting Elements

- JSP converted to Servlet at first access
- JSP scripting elements let you insert Java codes into the servlet
  - **Expressions**:
    - Form **<%=** expression **%>**
    - Evaluated and inserted into the output
  - **Scriptlets**
    - Form **<%** code **%>**
    - Inserted into the servlet's service method
  - **Declarations:**
    - Form **<%!** code **%>**
    - Inserted into the body

JSP Scripting Elements

- JSP Expressions:
  - Form: **<%=** expression **%>**
  - Example

```
Time: <%= new java.util.Date() %>
```

  - **Processing**
    - **Evaluated, converted to a string, and inserted in the page.**
    - **At run-time (when the page is requested)**

JSP Scripting Elements

- JSP Expressions:
  - Several variables predefined to simply jsp expressions
    - **request**, the HttpServletRequest;
    - **response**, the HttpServletResponse;
    - **session**, the HttpSession associated with the request (if any);
    - **out**, the PrintWriter (a buffered version of type JspWriter) used to send output to the client.
  - Example:

```
Your hostname: <%= request.getRemoteHost() %>
```

Script.jsp

JSP Scripting Elements

- **JSP Scriptlets**
  - Form: **<%** code **%>**
  - **Example:**

```
<% String queryData = request.getQueryString();
out.println("Attached GET data: " + queryData); %>
```

  - Inserted into the servlet's service method EXACTLY as written
  - Can access the same predefined variables as JSP expressions

Script.jsp

JSP Scripting Elements

- **JSP Declarations:**
  - Form: **<%!** code **%>**
  - Example: **<%! private int accessCount = 0; %>**
  - Inserted into the main body of the servlet class (outside of the service method processing the request)
  - Normally used in conjunction with JSP expressions or scriptlets.

<%! private int accessCount = 0; %> Example:

```
Accesses to page since server reboot:
 <%= ++accessCount %>
```

Script.jsp

Outline

- Outline:
  - Introduction and overview
  - JSP constructs
    - JSP scripting elements
    - JSP directives
    - JSP actions
  - JSP and Beans
    - Beans
    - Using beans in JSP

JSP Directives

- Affect the overall structure of the servlet class.
  - Form: `<%@ directive attribute1="value1" attribute2="value2" ... AttributeN="valueN" %>`
- Two commonly used types of directives
  - Page directives

    `<%@ page import="java.util.*" %>`
  - Include directives

    `<%@ include file="/navbar.html" %>`

JSP Directives

- Examples of Page directives
  - <%@ page import="java.util.*" %>
    <%@ page language="java“ import="java.util.*" %>
  - <%@ page contentType="text/plain" %>
    - Same as : <% response.setContentType("text/plain"); %>
  - <%@ page session="true"%>
  - Try: Remove the page directive in Script.jsp and see what happens

JSP Directives

- **Include Directive**
  - lets you include files at the time the JSP page is translated into a servlet.
  - Form: `<%@ include file="relative url" %>`
  - **Example situation where it is useful:**
    - **Same navigation bar at bottom of many pages.**
    - **Usage**
      - **Keep content of the navigation bar in an URL**
      - **Include the URL in all the pages**

Outline

- Outline:
  - Introduction and overview
  - JSP constructs
    - JSP scripting elements
    - JSP directives
    - JSP actions
  - JSP and Beans
    - Beans
    - Using beans in JSP

JSP Actions

- JSP *actions* control the behavior of the servlet engine. Let one
  - Dynamically insert a file
  - Forward the user to another page
  - Reuse JavaBeans components
  - ..

JSP Actions

- The include action
  - Form:

    ```
    <jsp:include page="relative URL" flush="true" />
    ```

  - Inserts the file at the time the page is requested.
    - Differs from the include directive, which inserts file at the time the JSP page is translated into a servlet.
  - Example: IncludeAction.jsp

JSP Actions

- The forward action:
  - Form: <jsp:forward page="relative URL" />
    <jsp:forward page="<%= someJavaExpression %>" />
  - Forward to the page specified.
  - Example: ForwardAction.jsp
- Several actions related to reuse of JavaBeans components
  - Discuss next

Outline

- Outline:
  - Introduction and overview
  - JSP constructs
    - JSP scripting elements
    - JSP directives
    - JSP actions
  - JSP and Beans
    - Beans
    - Using beans in JSP

JSP and JavaBeans

- Beans
  - Objects of Java classes that follow a set of simple naming and design conventions
    - Outlined by the JavaBeans specification
  - Beans are Java objects
    - Other classes can access them and their methods
    - One can access them from jsp using scripting elements.
  - Beans are special Java objects
    - Can be accessed using JSP actions.
    - Can be manipulated in a builder tool
    - Why interesting?
      - Programmers provide beans and documentations
      - Users do not have to know Java well to use the beans.

JSP and JavaBeans

- Naming conventions:
  - Class name:
    - Often include the word Bean in class name, such as UserBean
  - Constructor:
    - Must implement a constructor that takes no arguments
      - Note that if no constructor is provided, a default no-argument constructor will be provided.

JSP and JavaBeans

- Naming conventions: Methods
  - Semantically, a bean consists of a collection of properties
  - The signature for property access (getter and setter) methods

```
public void setPropertyName(PropertyType value);
public PropertyType getProperty
```

  - Example:
    - Property called **rank**:

```
public void setRank(String rank);
public String getRank();
```

    - Property called **age**:

```
public void setAge(int age);
public int getAge();
```


JSP and JavaBeans

- Properties should not be confused with instance variables
  - Even though instance variables are often mapped directly to property names,
  - Properties of a Bean are not required to correspond directly with instance variables.
  - Example:

```
public class DiceBean {
  private java.util.Random rand;
  public DiceBean() { rand = new Random();    }

  public int getDieRoll() {
        // return a number between 1 and 6
        return rand.nextInt(6) + 1;    }

  public int getDiceRoll() {
        // return a number between 2 and 12
        return getDieRoll() + getDieRoll();    }
}
```

JSP and JavaBeans

- Property name conventions
  - Begin with a lowercase letter
  - Uppercase the first letter of each word, except the first one, in the property name.
  - Examples: **`firstName, lastName`**
- Corresponding setter and getter methods:
  - **`setFirstName, setLastName`**
  - **`getFirstName, getLastName`**
  - Note the case difference between the property names and their access method

JSP and JavaBeans

- Indexed properties
  - Properties whose values are sets
  - Example: contacts --- set of all people in contact
  - Conventions:

```
public PropertyType[] getProperty()
public PropertyType getProperty(int index)

public void setProperty(int index, PropertyType value)
public void setProperty(PropertyType[])

public int getPropertySize()
```

JSP and JavaBeans

- Bean with indexed properties

```
import java.util.*;
public class StatBean {
   private double[] numbers;
   public StatBean() {numbers = new double[0];    }
   public double getAverage() {..}
   public double getSum() { .. }
   public double[] getNumbers()
   {        return numbers;   }
   public double getNumbers(int index)
   {       return numbers[index];    }
   public void setNumbers(double[] numbers)
   {       this.numbers = numbers;    }
   public void setNumbers(int index, double value)
   {       numbers[index] = value;    }
   public int getNumbersSize()
   {       return numbers.length;    }
}
```


JSP and JavaBeans

- Boolean Properties
  - Properties that are either true or false
  - Setter/getter methods conventions

```
public boolean isProperty();
public boolean isEnabled();
public boolean isAuthorized();

public void setProperty(boolean b);
public void setEnabled(boolean b);
public void setAuthorized(boolean b);
```

Outline

- Outline:
  - Introduction and overview
  - JSP constructs
    - JSP scripting elements
    - JSP directives
    - JSP actions
  - JSP and Beans
    - Beans
    - Using beans in JSP

Using Beans in JSP

- JSP actions for using beans:
  - jsp:useBean
    - Find or instantiate a JavaBean.
  - jsp:setProperty
    - Set the property of a JavaBean.
    - Call a setter method
  - jsp:getProperty
    - Get the property of a JavaBean into the output.
    - Call a getter method

Using Beans in JSP

- Example: The bean

```
package jspBean201;

public class SimpleBean {
  private String message = "No message specified";

  public String getMessage() {
    return(message);
  }
  public void setMessage(String message) {
    this.message = message;
  }
}
```

- Compile with javac and place in regular classpath
  - In Tomcat, same location as servlets. (can be different on other web servers)

Using Beans in JSP

- Use SimpleBean in jsp: ReuseBean.jsp
  - Find and instantiate bean

```
<jsp:useBean id="test" class="jspBean201.SimpleBean" />
```

  - **Set property**

```
<jsp:setProperty name="test" property="message" value="Hello WWW"/>
```

  - Get property: call the getMessage method and insert what it returns to web page

```
<H1>Message: <I>
 <jsp:getProperty name="test" property="message" />
</I></H1>
```

Using Beans in JSP

- The jsp:useBean action:
  - Format
    - Simple format: <jsp:useBean …/>
    - Container format: body portion executed only when bean first instantiated

      <jsp:useBean ...>
      Body
      </jsp:useBean>

Using Beans in JSP

- The jsp:useBean action:
  - Attributes:

```
<jsp:useBean id=.." scope="..", type="..", beanName="..", class=".." />
```

    - Scope: Indicates the context in which the bean should be made available
      - page (default): available only in current page
      - request, available only to current request
      - session, available only during the life of the current HttpSession
      - Application, available to all pages that share the same ServletContext
    - id: Gives a name to the variable that will reference the bean
      - New bean not instantiated if previous bean with same id and scope exists.
    - class: Designates the full package name of the bean.
    - type and beanName: can be used to replace the class attribute

Using Beans in JSP

- The jsp:setProperty action:
  - Forms:

```
<jsp:setProperty name=".." property=".." value=".." />
<jsp:setProperty name="." property=".." param=".." />
```

  - If the value attribute is used
    - String values are automatically converted to numbers, boolean, Boolean, byte, Byte, char, and Character
  - If the param attribute is used
    - No conversion

Using Beans in JSP/Example

- Calendar
  - Allow user to keep track of appointments online
    - login.html: for login
    - JSP papes: under **`tomcat/webapps/examples/jsp/cal`**
      - cal1.jsp: jsp for visualize appointments
      - cal2.sp: jsp for editing appointments
    - **Beans: under `tomcat/webapps/examples/WEB-INF/classes/cal`**
      - **`TableBean`**
      - **`Entries`**
      - **`Entry`**

Using Beans in JSP/Example

- cal1.jsp
  - Called from:
    - login.html:
      - Parameters: “name”, “email”
    - cal1.jsp
      - Parameters: “date”, with possible values “prev” or “next”
      - Note: This following is never activated

        <FORM METHOD=POST ACTION=cal1.jsp>

        …

        </FORM>
    - cal2.jsp
      - Pameters: “date”, “time”, “description”

Using Beans in JSP/Example

- cal1.jsp
  - It handles requests using TableBean

```
<%@ page language="java" import="cal.*" %>
<jsp:useBean id="table" scope="session" class="cal.TableBean" />

<%
   table.processRequest(request);
   if (table.getProcessError() == false) {
%>
```

Using Beans in JSP/Example

- cal1.jsp
  - How does TableBean handle requests?

```
public void processRequest (HttpServletRequest request) {
  // Request from login.html
  // Get the name and e-mail.
  this.processError = false;
  if (name == null)
    setName(request.getParameter ("name"));
  if (email == null)
  setEmail(request.getParameter ("email"));

  if (name == null || email == null)
  { this.processError = true; return; }
```


Using Beans in JSP/Example

- cal1.jsp
  - How does TableBean handle requests?

```
// Request from cal1.jsp
// Get the date.
String dateR = request.getParameter ("date");
if (dateR == null)
  date = JspCal.getCurrentDate ();
else if (dateR.equalsIgnoreCase("next"))
  date = JspCal.getNextDate ();
else if (dateR.equalsIgnoreCase("prev"))
  date = JspCal.getPrevDate ();
```

Using Beans in JSP/Example

- cal1.jsp
  - How does TableBean handle requests?

```
// Request from cal2.jsp
// make sure entry exist
entries = (Entries) table.get (date);
if (entries == null) {
  entries = new Entries ();
  table.put (date, entries);
}
// If time is provided add the event.
String time = request.getParameter("time");
if (time != null) entries.processRequest (request, time);}
```

//Entry update done in the Entries class.

Using Beans in JSP/Example

- cal2.jsp
  - Called from
    - Cal1.jsp
      - Parameters: “time”
    - It uses the parameter to call cal1.jsp
      - The call uses “date”, “time”, “description”
      - “date” from the TableBean
      - “description” from form (user input)