JSP/SERVLETS 2

S. Rosmorduc

Varia

POST, GET and Redirect

- Remainder : Correct usage of GET and POST : getting information vs. sending information
- Possible problem of robots :
- **BAD** : delete
- **better :**

```
<form action="delete" method= "POST">
<input type= "hidden" name= "id" value="6"/>
<input type="submit"/>
</form>
```

POST and redirect

- Problem with POST: reloading a POST page sends the request twice. E.g. : sends an order for a book twice.
- Solution: prevents the reloading by sending the user to a new page, using REDIRECT
- Mechanism of redirect :
  - special *http response* which asks the client to **visit a new page**
  - visiting the new page only displays data. It doesn't perform any operation.

POST and redirect: practical example

```
class addMessageServlet {
  public doPost(.......) {
      ... get the message content from request...
      ... create the message....
      Message m= myForum.createMessage(title, text);
      response.sendRedirect("showMessage?id="+ m.getId());
    }
}
```

- If the new message has 3 for id, the user browser will go to page showMessage?id=3
- This page can be reloaded without any problem.

Beans

Bean

- beans allow to pass information between JSP and Servlets
- A bean is an object of a class which:
  - has a default constructor ;
  - ***has properties manipulated throught accessors;***
  - implements **Serializable** (useful for session and application beans);
- a bean has a **name** and a **scope**;

Scope

- the **scope** describes the lifespan of the bean:
- **request**: the beans is forgotten when the request has been processed (communication between servlet and jsp) ;
- **session**: the lifespan of the bean is a particular user session, typically ended *when the user closes his browser.*
- **application**: the beans lives as long as the application is running (it disappears, for instance, if the server stops).
- **Beans are used for temporary data. Long term storage is done with databases or files.**

Request Beans

- Permettent à une Servlet d'envoyer des données à une JSP.

Servlet side:

```
String beanName=  "myBean";
Object beanValue= message; // (or anything)
request.setAttribute(beanName, beanValue);
```

JSP side:

Optional declaration:

```
<jsp:useBean id="myBean" class="String" scope="request"/>
```

With this declaration, the bean can then be used as a variable in Java code

or **direct use with the expression language:**

```
.... ${myBean} ...
```

Session Beans

- Allows to keep data as long as the user is connected (distroyed when the user closes his browser, or after a fixed idle time)
- session data are typically kept in memory;
- use: user login, basket for e-commerce;
- keeps the state of the application for the current user
- session beans need a session to be created.

Session beans, Servlet side

```
@WebServlet(name = "DemoCompteur", urlPatterns = {"/compteur"})
public class DemoCompteur extends HttpServlet {
    @Override
    protected void doGet(.....) {
        HttpSession session = req.getSession();
        Counter cpt= (Counter) session.getAttribute("counter");
        if (cpt== null) {
            cpt= new Counter();
            session.setAttribute("counter", cpt);
        }
        cpt.increase();
        req.getRequestDispatcher("/WEB-INF/jsp/counter.jsp")
              .forward(req, resp);
    }
  }
```

- getSession() : creates a session if no session exists
- there is a variant with a boolean argument: only creates the session if the argument is *true*.

Session beans, the bean

```
public class Counter implements Serializable {
    private int value = 0;
    public void increase() {
        value++;
    }
    public int getValue() {
        return value;
    }
}
```

Session Bean, JSP side

```
<!DOCTYPE html>
<html>
    <head>
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Counter ${counter.value} </h1>
    </body>
</html>
```

Application beans

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

    @Override
    protected void doGet(.........) {
        Counter cpt= (Counter)
            request.getServletContext().getAttribute("cptApp");
        if (cpt== null) {
            req.getServletContext()
                .setAttribute("cptApp", new Counter());
        }
        ...
    }
}
```

- lifespan : as long as the application runs
- use: to share data between all users; probably read-only
- used in multi-tasking environment: be careful !!!
- (actually, it's also true of session beans)

Let's remove Java code in JSP!

Expression Language

- In JSPs, written between ${...}
- expression ${a.b} has different meanings, depending on « a » :
- if a is an object, ${a.b} is a.getB()
- if a is a Map, ${a.b} is a.get("b");
- if a is an array, ${a.b} is a[b] : ${t.3} means t[3]
- «.» can be cascaded :

```
${client.address.town.name}
```

means

```
client.getAddress().getTown().getName()
```

Expression language (2)

- Expression language can be used in computations:

```
Full price : ${order.totalPrice *
1.196}
```

Expression Language : Operators

- empty : true if a bean is not defined, or is «empty» (meaning depends on the bean type)
- `${empty user}`: true if bean «user» is defined
- and, or, not : boolean operators:
- `${not empty user}`: true if bean «user» is defined and is not empty (for collections or strings)
- arithmetic operators : +, -, *, /, %
- comparisons: eq (equals), ne (not equals), lt (less than), gt (greater than), le (lesser or equals), ge (greater or equals)

Expression language: predefined variables

- **param**: parameters values for mono-valued parameters. E.g. `${param.nom}`
- **paramValues:** for multi-valued parameters (lists, check boxes...). E.g.

```
${param.sports}
```

- **cookie:** allows access to cookies values.

Java Standard Tags Library (JSTL)

- Special tags which can replace most of java code in JSPs;
- works hand in hand with the expression language;
- To use: write

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

  In the beginning of the JSP (done automatically by netbeans);
- Possibily add library JSTL 1.1 to your project.

Comparison

WITHOUT JSTL

```
<% for (int i=0; i < facture.getLignes().size(); i++) {
      siteMarchand.dto.LigneFacture ligne=
            (siteMarchand.dto.LigneFacture)facture.getLignes().get(i);
%>
  <li> <%= ligne.getArticle().getDesignation() %>,
      qté : <%= ligne.getQuantite() %>
<%
      }
%>
```

WITH JSTL

```
<c:forEach var="ligne" items="${facture.lignes}">
      <li> ${ligne.article.designation}, qté : ${ligne.quantite}</li>
</c:forEach>
```

Loops : forEach

- attributes:
  - **items :** collection to loop through
  - **var:** variable which will loop through the collection

```
<ul>
<c:forEach var="s" items="${students}">
   <li> ${s.name} ${s.surname}</li>
</c:forEach>
</ul>
```

ForEach

- Many possibilities : loop from the 4th element (starts with 1 (!))

```
<c:forEach var="e" items="${promotion}" begin="4">
    ${e}
</c:forEach>
```

- Loop a map :

```
<c:forEach var="a" items="${param}">
    <li> ${a.key} : ${a.value}</li>
</c:forEach>
```

- Loop with numeric indexes:

```
<c:forEach var="i" begin="0" end="12" step="3">
    ${i}
</c:forEach>
```

Conditional constructs: c:if

- no else !!!!

```
<!--
  Text shown only if bean «catalogue» is
  defined and not empty:
-->
<c:if test="${not empty catalogue}">
   Here is the catalogue :
          ...
</c:if>
```

Conditional constructs: choose/when/otherwise

- Allows multiple tests.

```
<c:choose>
  <c:when test="${group eq 'admin'}"> ..... </c:when>
  <c:when test="${group eq 'user'}"> ..... </c:when>
  <c:when test="${group eq 'guest'}">  ..... </c:when>
  <c:otherwise> Sorry, can’t connect. </c:otherwise>
</c:choose>
```

Protection vs. Xss

- Why: avoid *Javascript Injection*
- *Example:*

  ```
  <h1> Message </h1> <p> ${message}</p>
  ```

  is dangerous. If *message* is:

  ```
  <SCRIPT language="Javascript">
          document.location.href="http://pirate.com"
  </SCRIPT>
  ```

  javascript is loaded, executed, and the user is sent to pirat-site.com.:

- Called: XSS (Cross Site Scripting).
- solution : replace < and > by < and >

Protection vs. Xss

- Use tag <c:out value="..."/>
- The code would be:

```
<h1> Message </h1>

<p> <c:out value="${message}"/></p>
```

- output:

```
<h1> Message </h1>

&lt;SCRIPT language=&#034;Javascript&#034;&gt;

 document.location.href=&#034;http://pirate.com&#034; ;

&lt;/SCRIPT&gt;
```

- Ugly, but harmless.