Java Server Pages

Sesje, cookies, znaczniki

Sesje

- Problem: http jest bezstanowe
- Za każdym razem klient nawiązuje połączenie na nowo
- Zwykle konieczne jest zapamiętywanie co klient robił wcześniej
- Rozwiązanie:
  - Klient dostaje id sesji
  - Z każdym połączeniem wysyła swoje id
  - Serwer zachowuje dane klienta z numerem sesji

Session

- Zapamiętuje informacje o użytkowniku

```
<%
session.setAttribute("login", "Paweł");
%>
```

```
<%
String login = (String)session.getAttribute("login");
%>
```

Obiekt Session

- Metody:
  - getId()
  - getCreationTime()
  - getMaxInactiveInterval()
  - setMaxInactiveInterval(interval)
  - isNew()
  - invalidate()

Cookies

- Podobne do sesji ale cała informacja zapisana jest na komputerze klienta
- Zwykle trwałość większa niż sesji
- Mechanizm sesji może używać cookie do zapamiętania id

Obiekt Cookie

- Cookie[] cookies = request.getCookies()
- Cookie cookie = cookies[0];
- Metody:
  - getName()
  - getValue()
  - getMaxAge()
  - getComment()

Problemy ze skrypletami

- Mieszanie kodu HTML, Java i JavaScript powoduje nieczytelność
- Logika jest rozdzielona na różne strony
- Debugowanie jest trudne
- W rozbudowanych aplikacjach:
  - HTML developers – interfejs użytkownika
  - Java developers – logika aplikacji

Przykład strony JSP

```
<body>

<h1>Edycja odczynnika <%=request.getParameter("cas")%></h1>
<input type=button value='< powrót do karty'
    onClick='document.location="view.jsp?cas=<%=request.getParameter("cas")%>"'>
<font size=11>
<%
cas.CasEdit casedit=new cas.CasEdit(request.getParameter("cas"));
if(request.getParameter("delete")!=null) {
  casedit.delete(request.getParameter("oid"));
  %><SCRIPT language='JavaScript'>
    document.location='edit.jsp?cas=<%=request.getParameter("cas")%>&x=x';</SCRIPT><%
  }
if(request.getParameter("update")!=null && request.getParameter("sbm")!=null) {
  casedit.update(request.getParameter("oid"),request.getParameter("type"),
                request.getParameter("newdata"),request.getParameter("newdata2"),session);
  out.println(casedit.getTable("",request.getParameter("newtype"),session));
  }
else {
  if(request.getParameter("anuluj")!=null)
    out.println(casedit.getTable("",request.getParameter("newtype"),session));
  else

    out.println(casedit.getTable(request.getParameter("oid"),request.getParameter("newtype"),
    session));
}
%>
</body>
```

Użycie tagów

- Cel: uniknięcie kodu w Javie na stronie HTML
- Otwieranie i zamykanie tagu
  - <some.tag>
  - </some.tag>
- Tag pojedynczy
  - <some.tag/>
- Tagi mogą być:
  - predefinowane w specyfikacji JSP: <jsp:...>
  - ładowane z bibliotek

Tagi predefinowane

- <jsp:include page="page.jsp"/>
- <jsp:forward page="page.jsp"/>
- <jsp:param name="..." value="..."/>
- <jsp:useBean>
- <jsp:setProperty>
- <jsp:getProperty>

Biblioteki tagów

- Przykładowe biblioteki
  - http://java.sun.com/products/jsp/jstl/
  - http://jakarta.apache.org/taglibs/
  - http://jsptags.com/tags/
- Włączanie biblioteki do aplikacji
  - WEB-INF/lib/xyz.jar
  - WEB-INF/*.tld
- Załadowanie definicji (*.tld) na stronie JSP:
  - <% @taglib uri="<address>" prefix="xyz" %>
- Użycie tagu:
  - <xyz:newtag>...</xyz:newtag>

JSTL Java Standard Tag Library

- Biblioteka Sun'a
- Różne implementacje
  - najpopularniejsza: jakarta-taglibs
- Usprawnienia na stronach JSP bez użycia skrypletów (kodu w Javie)
- Ładowanie JSTL (jakarta):
  - wgrać pliki standard.jar i jstl.jar do katalogu WEB-INF/lib

Elementy JSTL

- core <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
  - variables, loops
- fmt <%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
  - formatting dates, numbers and money
- sql <%@taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
  - handling databases: connections, queries
- XML <%@taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x"%>
  - operations on XML documents
- functions <%@taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>
  - different converting functions

Expression Language

- ${expression}
- Uproszczony dostęp do:
  - standardowych obiektów i właściwości
  - beanów
  - obiektów typu Map, List czy Array
- Włączone do JSP 2.0

Core JSTL

- c:set
- c:out
- c:if
- c:forEach
- c:choose, c:when

c:set i c:out

- ustawianie i czytanie

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
<c:set var="name" value="Pawel"/>
...
My name is <c:out value="${name}"/>
</body>

</html>
```

c:if tag

- <c:if test="${some test}">
  - jakiś kod
- </c:if>

```
<c:if test="${x>5}">
x is greater than 5!
</c:if>
```

Zakresy widoczności obiektów

- page
  - tylko na stronie
- request
  - dla następnej strony przetwarzającej parametry z aktualnej
- session
  - dla wszystkich stron w ramach sesji
- application
  - dla całej aplikacji

Zakres (scope) zmiennych

- Domyślnie: request (inne: session, application, page)

```
<c:if test="${empty sessionScope.x}">
    <c:set var="x" value="1" scope="session"/>
</c:if>

<p>Current value:
<c:out value="${x}" /></p>

<c:set var="x" value="${x + 1}" scope="session"/>
```

c:choose tag

```
<p>
<c:choose>
    <c:when test="${x> 0 and x<= 5}">
            "x" value in range (0, 5>
    </c:when>
    <c:when test="${x> 5 and x <= 10}">
            "x" value in range (5, 10>
    </c:when>
    <c:otherwise>
            "x" value over 10 or below 0!
    </c:otherwise>
</c:choose>
</p>
```

c:foreach tag

Kod z użyciem JSTL:

```
<ul>
        <c:forEach var="i" begin="1" end="10">
                <li><c:out value="${i}"/>
        </c:forEach>
</ul>
```

To samo jako scriplet:

```
<ul>
<%
        for(int i=1; i<=10; i++) {
                out.println("<LI>"+i);
        }
%>
</ul>
```

Inne przykłady c:forEach

- <c:forEach var="i" begin="0" end="1000" step="100">
- <c:forEach var="word" items="${words}">
- <c:forEach var="country" items="Australia,Canada,Japan,Philippines,USA">
- <c:forTokens var="color" items="(red (orange) yellow)(green)((blue) violet)" delims="()">

Moduł Functions

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

<c:set var="str" value="Here we have a simple text"/>
<p>Text: ${str}
<p>To upper case: ${fn:toUpperCase(str)}
<p>Text length: ${fn:length(str)}</p>
<p>Does it contain "have" word?: ${fn:contains(str,"have")}</p>
<p>Number of words (space separated): ${fn:length(fn:split(str," "))}</p>
<p>Letter "s" in at position: ${fn:indexOf(str, "s")}</p>
<p>Replace "s" with "%": ${fn:replace(str, "s", "#")}</p>
```

Funkcje formatujące

- <fmt:formatNumber>
- <fmt:parseNumber>
- <fmt:formatDate>
- <fmt:parseDate>
- <fmt:setLocale>
- <fmt:bundle>, <fmt:setBundle>
- <fmt:message>
- <fmt:param>

Funkcje modułu sql

```
<sql:setDataSource
driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/mydb"
user="" password=""/>

<sql:query var="results" sql="SELECT * FROM car"/>

<c:forEach var="row" items="${results.rows}">
    <c:out value="${row.make}"/>
    <c:out value="${row.model}"/>
    <br/>
</c:forEach>
```

Uniwersalny showRS

```
<table border="1"> <tr>
 <%-- Get the column names for the header of the table --%>
 <c:forEach var="columnName" items="${results.columnNames}">
  <th><c:out value="${columnName}"/></th>
 </c:forEach>
 <%-- Get the value of each column while iterating over rows --%>
 <c:forEach var="row" items="${results.rows}">
  <tr>
   <c:forEach var="column" items="${row}">
    <td><c:out value="${column}"/></td>
   </c:forEach>
 </c:forEach>
</table>
```

JavaBeans

- Tag na stronie
  - <jsp:useBean id="credit" class="bank.Kredyt"/>
- Jeśli obiekt dostępny – używa go
- Jeśli obiektu nie ma – tworzy go
- Uzycie pól beana
  - <jsp:getProperty name="credit" property="rata"/>
- Ustawianie nowych wartości
  - <jsp:setProperty name="credit" property="kwota" value="1000"/>

Kredyt jako bean

- Pola z właściwościami
- Metody do pobierania i zapisywania (gettery i settery)
- Pusty konstruktor

Klasa Kredyt

```
public class Kredyt {
    double procent;
    double kwota;
    double lat;

    public double getProcent() {
        return procent;
    }

    public void setProcent(double procent) {
        this.procent = procent;
    }
...
```

Klasa Kredyt cd

```
public double getKwota() { return kwota;}

public void setKwota(double kwota) { this.kwota = kwota; }

public double getLat() { return lat; }

public void setLat(double lat) { this.lat = lat; }

public double getRata() {
    double rata = kwota * (procent/12)/
                  (1-(1/Math.pow(1.0+procent/12,lat*12)));
    return rata;
}
//nie ma setRata() !!!
}
```

Formatka

```
<form action="result.jsp">
kwota: <input type="text" name="kwota"><br/>
ile lat: <input type="text" name="lat"><br/>
procent: <input type="text" name="procent"><br/>
<input type="submit"/>
</form>
```

Skryplet z konstruktorem

```
<%
pl.edu.swsim.Kredyt kredyt = new pl.edu.swsim.Kredyt(
    Double.parseDouble(request.getParameter("kwota")),
    Double.parseDouble(request.getParameter("lat"))
    Double.parseDouble(request.getParameter("procent"))
);
out.write("<p>Rata wynosi: "+kredyt.getRata()+"</p>");
%>
```

Skryplet z setterami

```
<%
pl.edu.swsim.Kredyt kredyt = new pl.edu.swsim.Kredyt();
kredyt.setKwota(Double.parseDouble(
                        request.getParameter("kwota")));
kredyt.setLat(Double.parseDouble(
                        request.getParameter("lat")));
kredyt.setProcent(Double.parseDouble(
                        request.getParameter("procent")));
out.write("<p>Rata wynosi: "+kredyt.getRata()+"</p>");
%>
```

Użycie beana

```
<jsp:useBean id="credit" class="pl.edu.swsim.Kredyt">
 <jsp:setProperty name="credit" property="kwota"
    value=' <%=Double.parseDouble(
                        request.getParameter("kwota"))%>' />
 <jsp:setProperty name="credit" property="procent"
    value=' <%=Double.parseDouble(
                        request.getParameter("procent"))%>' />
 <jsp:setProperty name="credit" property="lat"
    value='<%=Double.parseDouble(
                        request.getParameter("lat"))%>'/>
</jsp:useBean>
```

Trochę prościej

```
<jsp:useBean id="credit" class="pl.edu.swsim.Kredyt">
 <jsp:setProperty name="credit" property="kwota" param="kwota"/>
 <jsp:setProperty name="credit" property="procent" param="procent"/>
 <jsp:setProperty name="credit" property="lat" param="lat"/>
</jsp:useBean>
```

Rata: <jsp:getProperty name="credit" property="rata"/>

Jeszcze prościej

```
<jsp:useBean id="credit" class="pl.edu.swsim.Kredyt">
 <jsp:setProperty name="credit" property="*"/>
</jsp:useBean>
```

Rata: <jsp:getProperty name="credit" property="rata"/>

Może być niebezpieczne! Co jeśli ktoś poda parametr rata a rata będzie miała settera setRata()?

Najprościej

```
<jsp:useBean id="credit" class="pl.edu.swsim.Kredyt"/>
<jsp:setProperty name="credit" property="*"/>

Rata: ${credit.rata}
```

Tablica kredytów

```
<jsp:useBean id="credit" class="pl.edu.swsim.Kredyt">
    <jsp:setProperty name="credit" property="*"/>
</jsp:useBean>
<ul>
<c:forEach var="i" begin="1" end="10">
    <li>
    <jsp:setProperty name="credit" property="lat" value="${i}"/>
    <c:out value="${i}"/> -
    <jsp:getProperty name="credit" property="rata"/>
</c:forEach>
</ul>
```

Tablica kredytów

```
<jsp:useBean id="credit" class="pl.edu.swsim.Kredyt">
    <jsp:setProperty name="credit" property="*"/>
</jsp:useBean>
<ul>
<c:forEach var="i" begin="1" end="10">
    <li>
    <jsp:setProperty name="credit" property="lat" value="${i}"/>
    <c:out value="${i}"/> -
    <jsp:getProperty name="credit" property="rata"/>
</c:forEach>
</ul>
```

Tablica kredytów

```
<jsp:useBean id="credit" class="pl.edu.swsim.Kredyt">
    <jsp:setProperty name="credit" property="*"/>
</jsp:useBean>
<ul>
<c:forEach var="i" begin="1" end="10">
    <li>
    <jsp:setProperty name="credit" property="lat" value="${i}"/>
    ${i} - ${credit.rata}
</c:forEach>
</ul>
```

Podsumowanie

- W JSP można używać tagów
- Jest kilka użytecznych bibliotek tagów
- Główna idea: bez kodu w Javie na stronie JSP!
- Problem:
  - obsługa wyjątków, kontrola parametrów
- Krok dalej:
  - sprawdzenie parametrów i przygotowanie danych w klasach Javy
  - wyświetlanie przygotowanych danych za pomocą JSP