JavaServer Pages

Úvod

- Cíl JSP
  - dát HTML autorům k dispozici možnosti servletů bez znalosti Javy
  - umožnit oddělení prezentace a aplikační logiky
    - prezentace: JSP
    - aplikační logika: obyč Java, servlety, Spring, EJB

- Jak JSP pracuje
  - HTML s doplněnými JSP značkami (a kusy Java kódu)
    - navenek stejná myšlenka jako PHP, ASP, …
  - kontejner interně přeloží JSP stránku na servlet
    - první zobrazení
    - předkompilování
  - servlet je spuštěn při dotazu klienta na JSP stránku

```
<p>Tahle JSP stránka říká: <cite>
<%
 Date d = new Date();
 out.println("dnes je: " + d + ".");
%>
</cite>. Je z ní možno přejít na
<a href="hello">Hello World servlet</a>.

</body>
```

Základní prvky

- JSP syntaxe
  - *<% … %>* označují kód
- Druhy JSP bloků
  - *directivy* – ovlivňují překlad
  - *deklarace* – deklarují Java elem.
  - *skriptlety* – provádí kód
  - *výrazy* – pro jednoduché echo
  - *action elements* – std akce
- Komentáře
  - <%-- text komentáře --%>

```
<%@ page import="java.util.*" %>
<%! Date epoch = new Date(0); %>

<%
Date today = new Date();
long seconds = today.getTime() -
    epoch.getTime();

out.println("<p>Today is: <em>" +
 today.toString() + "</em>.</p>");
%>

<p>It is <%=seconds/1000 %> secs
    since the Epoch.</p>

<jsp:include page="/footer.html" />
```

Direktivy <%@

- Říkají, jak má být stránka překládána
  - konfigurace a obsah generovaného servletu
  - direktiva: *name* a *attribute*="*value*" páry
    - každý atribut <= 1x
- *page* [contentType] [import] [buffer] [session] [errorPage]
  - *import* – Java třída nebo package; 0..N výskytů direktivy
  - *buffer* – default 8192 bytes; automatický flush
  - *session* – boolean, default "true"; doporučeno vypnout pro non-session JSP
  - *errorPage* – URL zobrazené při neodchycené výjimce
- *include* file
  - *file* – relativní URL zdroje, který má být zakompilován
- *taglib* uri prefix
  - deklaruje použitou knihovnu značek

Výrazy <%=

- Pouze výpis hodnoty konvertované na řetězec
  - objekt musí být znám (deklarován a inicializován)
  - bez středníku!

Deklarace <%!

- Deklarují kód globální pro generovaný servlet
  - tj. na úrovni třídy, nikoli uvnitř metody *service()*
    - » rozdíl od zbytku JSP
  - atributy třídy, metody
- Pozor
  - hodnoty atributů jsou inicializovány pouze jednou (servlet startup)

- Inicializační a finalizační kód
  - přepsat metody
    - *public void jspInit()*
    - *public void jspDestroy()*
  - definované v *javax.servlet.jsp.JspPage*
  - default implementace je závislá na kontejneru

Scriptlety <%

- Kód prováděný v servisní metodě
  - libovolná Java
  - lze používat implicitní objekty (viz dále)

- Pozor
  - kód z JSP deklarací je na „globální" úrovni servletu
  - kód scriptletu je lokální uvnitř metody *service()*
- Správný návrh
  - cíl: oddělit prezentaci a aplikační logiku
  - vypisovat HTML uvnitř scriptletu je čirá hloupost
  - co nejméně kódu ⇒ použít JavaBeans, knihovny značek

Implicitní objekty

- Reprezentace rozsahů platnosti
  - » použití analogické jako v servletu
- Standardní stránka
  - *request*, *response*, *out*
    - jako v servletu, out = response output stream
  - *page*, *session*, *application*
    - první zpřístupňuje konfiguraci servletu via *getServletConfig()* methodu
    - *javax.servlet.http.HttpSession* interface
    - *javax.servlet.ServletContext* interface, přístup ke kontextu servletu
- Chybová stránka
  - » deklarovaná <%@ page isError="true" %>
  - *exception*
    - *java.lang.Throwable* instance, obvykle *javax.servlet.jsp.JspException*

Standardní akce <jsp:action

- Zapouzdřují základní obecnou funkcionalitu
  - definovány standardem; syntax:

```
<jsp:action {attr="val"}*>
  { <jsp:param name="..." value="..." /> }*
</jsp:action>
```

- *useBean id="..." class|type="..." [scope="..."]*
  - *id* identifikuje instanci v JSP, *scope* ∈ {page,request,session,application}
  - type umožňuje zúžit typ (třídu)
    - » možno vložit tělo elementu: inicializační kód a/nebo hodnoty vlastností
- *set/getProperty name="..." property="..." [value="..."]*
  - použití *param* místo *value* v *setProperty* zpřístupňuje data dotazu
- *include/forward page="..."*
  - vložení/přesměrování na relativní URI s aktuálním kontextem
    - » tj. ovlivňuje <@...
    - » možno použít atribut *flush*: odešle dosavadní obsah
  - *<jsp:param ...>* v těle nastavuje parametry URI

Knihovny značek

- Důležitý mechanismus pro uživatelské rozšiřování funkčnosti JSP
  - abstrakce aplikační logiky a dat
  - reprezentovaná formou značek JSP akcí
  - napomáhá odstranění aplikační logiky (scriptlety) z JSP

- Životní cyklus značek
  - programátor vytvoří „taglib"
    - Java handlery pro značky
    - XML tag library descriptor (TLD)
  - autor JSP
    - umístí knihovnu do webové aplikace
    - použije značky ve stránce
  - kontejner zpracuje JSP
    - validace obsahu – značky mohou definovat validační metody
    - překlad do odpovídajícího kódu
  - kód handlerů značek je proveden vygenerovaným servletem

Známé knihovny:

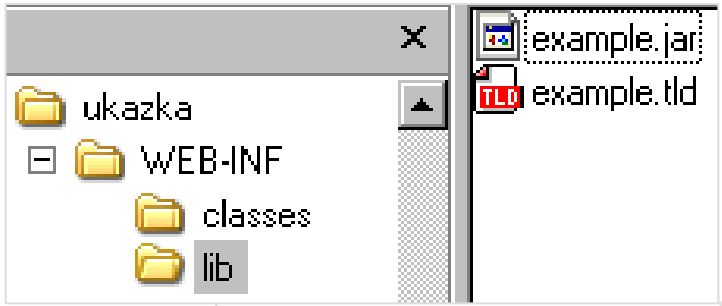
- JSTL
- Jakarta taglibs (logování, SQL, …)
- Struts
- …

Stránka s Custom Tags

```
<%@ taglib
  uri="http://example.org/example-taglib"
  prefix="eg" %>

<h3>Seznam studentů</h3>
<ul>
<eg:studentlist subject="PIA">
<li><%= student %></li>
</eg:studentlist>
</ul>

<jsp:include page="/include/footer.jsp" />
```



web.xml

```
<taglib>
   <taglib-uri>http://example.org/
example-taglib</taglib-uri>
   <taglib-location>/WEB-INF/lib/
example.tld</taglib-location>
</taglib>
```

Jazyk pro výrazy (EL)

- Cíl: snadný přístup k proměnným/datům včetně polí
  - alternativa k *<jsp:useBean … >*
  - od JSP 2.0, původně součást JSTL
  - funguje pouze když web.xml je dle specifikace Servlet 2.4 (XSD)
- Syntax: *${scope.var op var}* → hodnota
  - možno použít v textu, atributech značek
  - lze zakázat přes web.xml
- Příklady
  - *<c:if test="${applicationScope.loggedUsers > 0}"> … </c:if>*
  - *Máte ${cart.numberOfItems} položek v košíku …*
  - *Stránka vygenerována na ${header["host"]}*

EL operátory a objekty

- Operátory
  - std aritmetické a logické, ternární
  - procházení polí/kontejnerů
    - » velmi flexibilní (*null* handling) a polymorfní
    - » syntaxe: jak *map["key"]* tak i *map.key*
  - není možnost přiřazení

- Implicitní objekty
  - *pageContext* → *servletContext*, *session*, *request*
  - *param*, *header*, *cookie* – mapy
  - *initParam*, [page|request|session|application]*Scope* – mapy

JSP Standard Tag Library (JSTL)

- JSP 1.2 doplněk, JSR 52, od JSP 2.0 součást
- Core, obecná funkčnost
  - řízení toku výpočtu
  - i18n, formátování textu
  - podpora SQL, XML
- Jazyk pro výrazy

- Knihovny (pro *<%@taglib uri=... >*)
  - core: *http://java.sun.com/jstl/core* (prefix *c*)
  - text fmt, i18n: *http://java.sun.com/jstl/fmt* (prefix *fmt*)

Prvky JSTL Core

- Podmínky
  - c:if, c:choose/when/otherwise
- Cykly
  - c:forEach, c:forTokens
- Odkazy na zdroje
  - c:import, c:url, c:redirect
    - » přístup k externím zdrojům, přepisování pro relativní URI
- Lokalizace
  - fmt:message, fmt:formatDate
    - → vezme se obsah resource bundle {"myErrorMsg", "Stala se chyba"}
- XML
  - x:parse, x:set, x:out; x:forEach, x:transform

Příklad použití JSTL

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
...
<h2><c:out value="${pageScope.heading}"/></h2>

<table border="1">
<c:forEach var="customer" items="${customers}" varStatus="status">

  <c:if test="${status.first}">
    <tr><th>#</th><th><fmt:msg key="custHeadText" /></th></tr>
  </c:if>

  <tr><td>${status.index}</td><td>${customer.name}</td></tr>

</c:forEach>
</table>

<p><fmt:message key="custCountText">
      <fmt:param value="${fn:length(customers)}">
   </fmt:message></p>
```

Nejde ${customers.size} – není JavaBean

Počet zákazníků

#	Název
1	BEA Systems
1	RedHat
1	SoftEU
1	CIV ZČU

Celkem máme 4 zákazníky.