Kapitola 1. Přednáška 6 - komplexní podpora webových aplikací. Webové rámce.

Obsah

Webové aplikační rámce

Osnova

- Webový rámec a jeho architektura
- Funkcionalita webových rámců
- Výstavba aplikací nad webovými rámci

Webový rámec

- Více či méně ucelené prostředí pro tvorbu (případně i nasazení) webových aplikací středního rozsahu
- Není to knihovna!
- Využívá "The Hollywood Principle": Do Not Call Us, We Will Call You!
- Rámec "funguje sám", my jej modifikujeme a doplňujeme
- Budeme diskutovat javové rámce

Třívrstvá architektura

Vrstvy "od uživatele dolů":

1. prezentační vrstva - ve webových aplikacích HTML, CSS, JavaScript,... na klientovi (prohlížeči): JSP, servlety, XSLT, JSTL, JSF, šablony...
2. aplikační logika - javové objekty, JavaBeans, EJB...
3. databázová vrstva - DBMS dostupný přes JDBC

Koncepce MVC

- Model - objekty/data a jejich obslužné metody realizující vlastní aplikační logiku
- View - pohled na data - prezentace, vstup od uživatele
- Controller - řídí tok zpracování - kdy se aktivuje jaká metoda modelu, jaký pohled se uplatní na výsledek,...

Přednosti MVC

- Nezávislost modelu na zbylých částech
- Jeho znovupoužitelnost
- Na modelu a pohledech lze pracovat do jisté míry nezávisle - jsou požadovány jiné schopnosti (programátor vs. designér)
- Controller mnohdy programovat ani nemusíme - díky (webovým) rámcům!

Javové webové aplikace

Javové webové aplikace

- Základem je Java Servlet API
- specifikuje Servlety, stránky JSP, knihovny značek, filtry, popisovače...
- Jeho implementace jsou volně dostupné a dobře prozkoumané

Javové webové aplikace (2)

Java Servlety:

- javové třídy rozšiřující typicky abstraktního předka javax.servlet.http.HttpServlet

stránky JSP:

- podobný princip jako ASP, PHP
- při požadavku se přeloží do servletu

Javové webové aplikace (3)

- knihovny značek (Tag Libraries): pro JSP lze zadefinovat vlastní XML značky obsluhované javovým kódem a ty na stránkách JSP používat
- filtry: než je HTTP požadavek předán servletu (JSP), může být filtrován, filtr je podobný servletu, lze filtrovat i vygenerovanou odpověď

Javové webové kontejnery

Kontejnery (Java Web Containers):

- servery (programy) hostující javové webové aplikace

Příklady (volně dostupné):

- Apache (Jakarta) Tomcat
- Jetty
- LWS, Bajie, ...

Rozdíl oproti aplikačním serverům:

- většinou nepodporují plnou specif. J2EE
- nehostují EJB

Funkcionalita webových rámců

Oddělení prezentační vrstvy

Oddělení prezentační vrstvy - většinou prostřednictvím jazyka šablon (template language) jako jsou:

- Velocity, WebMacro, FreeMarker, RIFE ...
- nebo JSTL (std. knihovna značek)
- moderně přes JSF (Java Server Faces)
- napodobují klasické "desktopové" GUI v prostředí webové aplikace
- případně transformace XSLT

Oddělení prezentační vrstvy (2)

Oddělení prezentační vrstvy

- Šablona Velocity - stránka jednoduchého weblogu

```
<html><head>
  <title>#showWebsiteTitle()</title>
  <style type="text/css">#includePage("_css")</style>
  #showRSSAutodiscoveryLink()
</head>
```

```
<body><div id="Content">
    <center>
      <h1>#showWebsiteTitle()</h1>
      <p class="descrip">#showWebsiteDescription()</p>
      #showWeblogCategoryChooser()<br>
    </center>
    #showWeblogEntries("_day" 15)
    <hr />
    #showReferers( 40 25 )
</div></body></html>
```

Oddělení prezentační vrstvy (3)

- Stránka využívající JSF

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-faces"
  prefix="s" %>
<f:use_faces>
  <s:form action="/listFlights">
  <h:input_text id="fromCity"
             valueRef="FlightSearchForm.fromCity"/>
  <h:input_text id="toCity"
             valueRef="FlightSearchForm.toCity"/>
<h:command_button id="submit" action="success" label="Submit" commandName="submit"
   <h:command_button id="reset" action="reset" label="Reset"
                     commandName="reset" />
   <s:errors/>
   </s:form>
</f:use_faces>
```

Funkcionalita webových rámců

- Správa zabezpečení - např. deklarativní nařízení přihlašování
- použití nastavení Java Security Manageru
- pak lze na serveru hostovat i nedůvěryhodný "zákaznický" kód
- lze také použít filtry (jakási vnitřní "autentizační proxy")
- Vše směřuje k deklarativní specifikaci zabezpečení - snadněji se udržuje

Správa zabezpečení - příklad

Příklad konfigurace Java Security Manageru (Apache Tomcat 5):

```
The permission granted to your JDBC driver
grant codeBase "jar:file:${catalina.home}/webapps/examples/WEB-INF/lib/driver.jar!
    permission java.net.SocketPermission "dbhost.mycompany.com:5432", connect";
};...
// These permissions apply to the container's core code, plus any additional libra
grant codeBase "file:${catalina.home}/server/-" {
        permission java.security.AllPermission;
};
```

Validace uživatelských vstupů

Řada rámců podporuje deklarativní specifikaci uživatelských vstupů

- datový typ, formát, přípustné rozmezí hodnot...
- a jejich dekódování
- znakové sady, různé národní zvyklosti

Pokročilé rámce:

- řízení toku při vyplňování formulářů (průvodci jako u desktopových GUI)
- automatické generování fyz. podoby form.

Příklad specifikace zpracování uživ. vstupů - rámec Struts

```
<form-validation>
   <formset>
      <form name="addSubjectForm">
         <field property="subjID"
                depends="required" page="1">
             <arg0 key="admin.subject.missing.ID"/>
         </field>
         <field property="subjName"
                depends="required" page="1">
             <arg0 key="admin.subject.missing.name"/>
         </field>
         <field property="groupID"
                depends="required" page="2">
             <arg0 key="admin.subject.missing.groupID"/>
         </field>
```

```
</form>
<form name="addTaskForm">...
```

Udržování informací o relaci

Mezi jednotlivými interakcemi s uživatelem je uchován stav dané relace -

- Překlenuje bezestavovost HTTP
- V javových aplikacích pomocí objektů javax.servlet.http.HttpSession

Řízení toku aplikace

- Rozsáhlejší aplikace mívají složitou navigační strukturu
- v MVC modelu řídí navigaci Controller
- navigace bývá v rámcích popisována deklarativně

Příklad řízení toku - Struts

```
<!-- Action Mapping Definition -->
<action-mappings>
  <!-- List Flights action -->
  <action path="/listFlights"
     type="foo.bar.FlightSearchAction"
     name="FlightSearchForm"
     scope="request"
     input="/faces/FlightSearch.jsp">
     <forward name="success" path="/faces/FlightList.jsp"/>
  </action>
</action-mappings>
```

Zotavení z chyb

- Opět deklarativně bývá popsáno řízení toku po vzniku chybového stavu (výjimky)
- Umožňuje bezpečně pokrýt i neočekávané chyby a zaznamenat je, poslat informaci správci...

Záznamy o událostech (logging)

Cíl:

- S minimálním "obtěžováním" programátora zajistit kontrolu nad chodem aplikace ve fázi tvorby, ladění i ostrého provozu

Java nabízí několik "logging API":

- od Java 1.4: balík java.util.logging
- kdekoli: balíky log4j
- zastřešující: Jakarta Commons Logging

Pomocí Aspect-oriented Programming lze logování řešit ještě méně invazivně

Internacionalizace a lokalizace

Rozlišujeme:

- Internationalization (i18n)
- Localization (l10n)

  Co znamenají národní zvyklosti:

- texty v nabídkách, ovládacích prvcích,...
- formáty výpisu data, času, měny, vícemístných a desetinných čísel
- lexikografické řazení

Internacionalizace a lokalizace - příklad

```
<%@ taglib uri="http://java.sun.com/jstl/fmt" prefix="fmt" %>
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
<c:if test="${lang==null}">
 <fmt:setBundle basename="com.heaton.bundles.Forum"
    var="lang" scope="session"/>
</c:if>
<c:if test="${param.lang!=null}">
 <fmt:setLocale value="${param.lang}"/>
 <fmt:setBundle basename="com.heaton.informit.I18NBundle"
    var="lang" scope="session"/>
 <c:redirect url="index.jsp"/>
</c:if>
<html><head><title>I18N Example</title></head>
<body>
    <h1><fmt:message key="login.pleaselogin" bundle="${lang}"/></h1>
```

```
    <form method=post action=main.jsp>
        <fmt:message key="login.uid" bundle="${lang}"/><input name=uid><br/>
        <fmt:message key="login.pwd" bundle="${lang}"/><input name=pwd><br/>
        <input type="submit" name="action" value="<fmt:message
        key="login.title" bundle="${login}"/>">
    </form>
    <h1><fmt:message key="login.language" bundle="${lang}"/></h1>
    <ul>
        <li><a href="index.jsp*lang=en">
        <fmt:message key="login.english" bundle="${lang}"/>(English)</li>
        <li><a href="index.jsp*lang=es">
        <fmt:message key="login.spanish" bundle="${lang}"/>(Spanish)</li>
    </ul>
</body></html>
```

Zajištění perzistence objektů

- Objektový model v paměti vs. data na discích (v databázi)
- Rámce do jisté míry vzájemné převody automatizují
- Většinou používají osvědčený produkt třetí strany - např. Hibernate, POJO

Škálovatelnost a distribuovanost

- Výkon aplikace nelze zvyšovat pouze hardwarovými prostředky nebo optimalizací
- Zejména dnes je levnějším řešením použít svazek (cluster) "běžně výkonných" počítačů namísto jednoho supervýkonného
- Více zatížená aplikace musí být schopna výkon škálovat

Škálovatelnost a distribuovanost

Řešením jsou tzv. aplikační servery

- poskytují middleware: komplexní prostředí pro běh rozsáhlých (podnikových) aplikací
- míří nad schopnosti běžných webových rámců - ty ale většinou podporují/využívají

Podpora vývojovými prostředími

- Eclipse (+Struts Studio) volně dostupný
- Borland JBuilder (+InternetBeans Express)

Přehled rámců

- Web Application Framework Research project - http://www.waferproject.org.

Shrnutí

- Webové rámce usnadňují tvorbu středně rozsáhlých aplikací s webovým rozhraním
- Realizují oddělení jednotlivých vrstev, jsou postavené na koncepci MVC
- Programátorovi nabízejí infrastrukturu, která "funguje sama" a on ji doplňuje
- Kromě toho poskytují řadu dalších drobných služeb