Servlety a JSP

Petr Adámek, petr.adamek@ibacz.eu

Úvod

Rekapitulace vstupních znalostí

- Standardy
- Nástroje (Běhové prostředí, nástroje pro vývoj)
- Servlety
- JSP
- JSP značky
- EL (Expression Language)
- Internacionalizace a lokalizace
- Připojení k databázi

MVC

- Architektura MVC
- Best practices
- Proč webové frameworky

© 2011 IBA CZ, s.r.o.

Webové aplikace v Javě

Podpora pro vývoj webových aplikací v Javě

- Součást platformy Java EE
- Aktuální verze je Java EE 6 (JSR 316)
- Platforma Java je pro tvorbu serverové části webových aplikací poměrně vhodná, oproti např. PHP umožňuje psát snadněji udržovatelné aplikace, dobře škáluje, má skvěle zvládnutou internacionalizaci.

Klíčové specifikace

- Java Servlet 3.0 (JSR 315)
- JSP 2.2 (JSR 245)
- EL 2.2 (JSR 245)
- JSF 2.0 (JSR 314)
- JSTL 1.2 (JSR 052)

© 2011 IBA CZ, s.r.o.

Nástroje

Běhové prostředí

- Webové aplikace v Javě vyžadují běhové prostředí.
- Hlavní komponentou je tzv. *Servlet kontejner.*
- Servlet kontejner může být součástí plnohodnotného aplikačního serveru (např. Websphere AS, WebLogic AS, Glassfish, JBoss), nebo může být dodáván v podstatě samostatně pouze s nejnutnějšími dalšími komponentami (např. Tomcat, Jetty).
- Servlet kontejner je obvykle dodáván i s vestavěným webovým serverem, nicméně je možné jej použít i s jiným webovým serverem (např. Apache).
- Aplikace se do běhového prostředí instalují ve formě archivů war.

Vývojové prostředí

- NetBeans IDE
- IntelliJ IDEA
- Eclipse
- Rational Application Developer

© 2011 IBA CZ, s.r.o.

Demonstrační příklad

Vývojové prostředí

- Kde stáhnout (http://www.netbeans.org/)
- Jakou edici (buď *Java EE* nebo *all*)
- Jakou verzi (buď 6.9.1 nebo 7.0.1 – obě podporují Java EE 6)

Aplikační servery/web kontejnery

- Glassfish (pro Java EE 6 verze 3.0 nebo vyšší)
- Tomcat (pro Java EE 6 verze 7.0 nebo vyšší)
- NetBeans podporují i další, nicméně Glasfish a Tomcat jsou přímo přibaleny.

Postup

- Nainstalujeme NetBeans
- Vytvoříme nový projekt typu webová aplikace
- Jako server vybereme Tomcat a jako verzi Java EE vybereme *Java EE 6 Web*.

© 2011 IBA CZ, s.r.o.

Servlety

Servlet

- Základní komponenta pro tvorbu webových aplikací v Javě
- Rozšiřuje `javax.servlet.http.HttpServlet`
- Označená anotací `@WebServlet` (dříve se příslušné informace uváděly do tzv. deployment deskriptoru)
- Zpracovává HTTP požadavky od klienta a generuje odpovědi

© 2011 IBA CZ, s.r.o.

Demonstrační příklad

Postup

- Vytvoříme nový servlet
- Vysvětlíme si jeho jednotlivé části
- Upravíme kód servletu tak, aby generoval jednoduchý formulář a zpracovával výsledek
- Podíváme se na
  - HttpServletRequest
  - HttpServletResponse
  - HttpSession
  - RequestDispatcher
  - Parametry
  - Atributy
- Ukážeme si, jak vypadá deployment deskriptor (který je od verze Java EE 6 nepovinný)

© 2011 IBA CZ, s.r.o.

Java Server Pages

- Servlety jsou mocný nástroj, ale pro vytváření převážně textového obsahu se nehodí.
- Proto existují tzv. Java Server Pages, které fungují na podobném principu jako PHP – jde o textový dokument (obvykle ve formátu HTML), do kterého můžeme doplňovat tzv. Scriplety, tj. fragmenty kódu v jazyce Java.
- JSP stránky jsou ve skutečnosti také překládány do podoby servletů, provádí to ale zcela transparentně servletový kontejner.

Demonstrační příklad

- Vytvoříme si ukázkovou JSP stránku a vložíme do ní scriplet.

© 2011 IBA CZ, s.r.o.

Knihovny JSP značek

JSP značky

- Používání scripletů výrazně snižuje čitelnost stránek a velmi komplikuje údržbu. Naštěstí existuje alternativa v podobě tzv. JSP značek.
- Značky můžeme vytvářet sami, nebo používat již hotové knihovny značek.
- Existuje standardní knihovna značek JSTL (JSP Standard Tag Library).

Expression Language

- Jazyk, který je možné používat v kombinaci s JSP značkami (viz příklad).

Demonstrační příklad

- Nahradíme scriplety za JSP značky z JSTL a výrazy v EL.

© 2011 IBA CZ, s.r.o.

Servlety versus JSP

Servlety

- Mohou generovat jakýkoliv obsah (textový i binární)
- Nevhodný pro dlouhé fragmenty textového obsahu

JSP

- Mohou generovat pouze textový obsah
- Přehledné (zejména díky JSP značkám a EL)
- Nevhodné pro výkonný kód

Závěr

- JSP stránky se hodí pro generování HTML stránek
- Servlety se hodí pro binární obsah (např. obrázky) nebo pro zpracování požadavků (např. formulářových dat apod.)

© 2011 IBA CZ, s.r.o.

Internacionalizace a lokalizace

Základní třídy

- java.util.Locale
- java.util.ResourceBundle
- java.text.Format
- java.text.MessageFormat

Servlety

- Locale l = request.getLocale()
- Enumeration<Locale> en = request.getLocales()

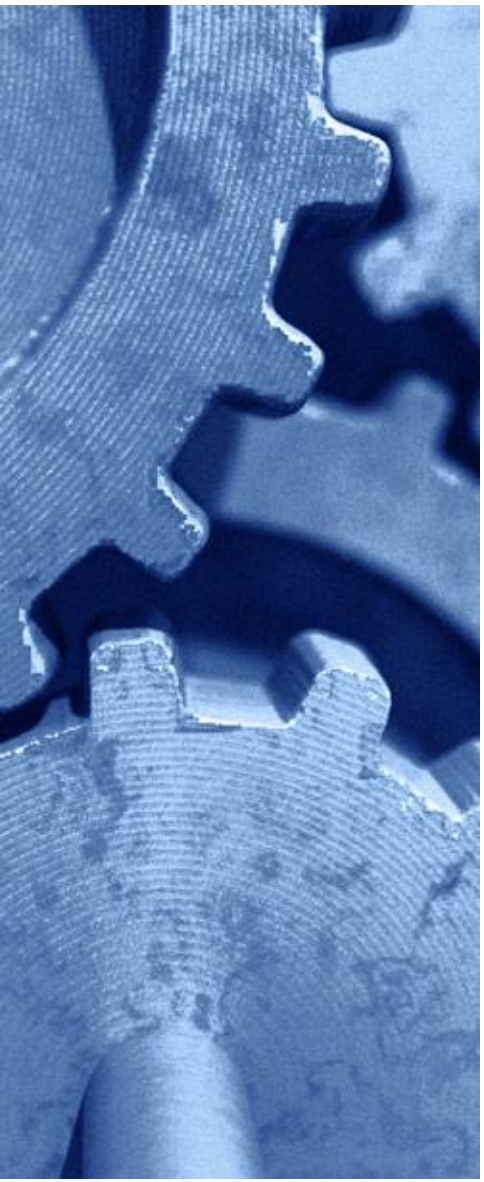
JSP a JSTL

K lokalizaci JSP stránek se hodí knihovna JSTL:

```
<%@ taglib prefix="f" uri="http://java.sun.com/jsp/jstl/fmt" %>
<f:setBundle basename=“Bundle" />
<f:message key=“now_is" />: <f:formatDate value="${now}"
    type="both" dateStyle="full" timeStyle="full"
    timeZone="${timezone}" />
```


© 2011 IBA CZ, s.r.o.

Připojení k databázi



Princip

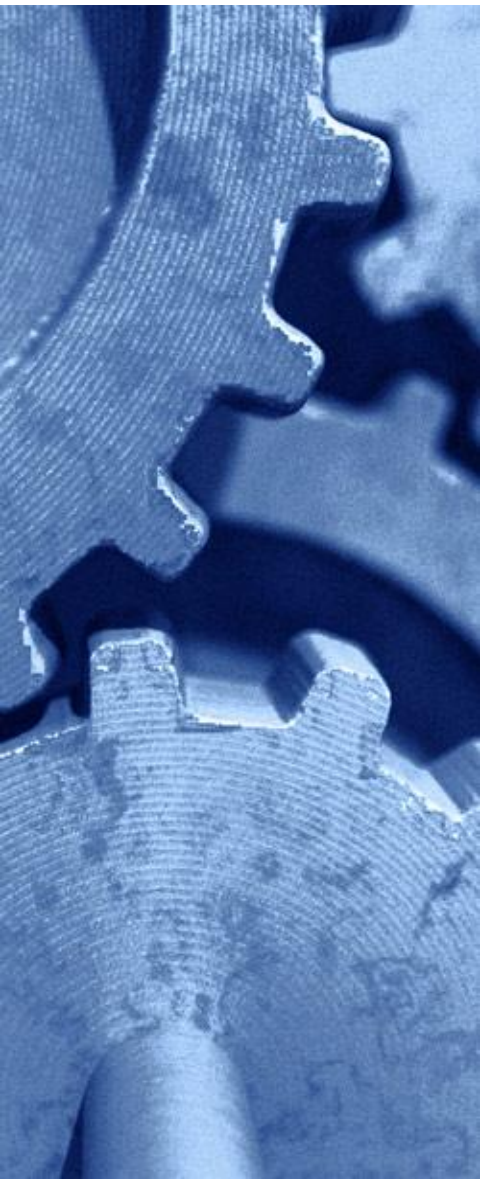
- Pro přístup k databázi se používá JDBC
- Je potřeba JDBC driver (např. derbyClient.jar)
- Parametry připojení se uvádí ve formě tzv. jdbc url: jdbc:derby://localhost:1527/ContactManager
- Za konfiguraci připojení k databázi nezodpovídá aplikace, ale servlet kontejner nebo aplikační server
- Připojení k databázi je kontejnerem poskytováno prostřednictvím rozhraní DataSource buď pomocí Dependency Injection nebo pomocí JNDI.

Postup nastavení u kontejneru Tomcat

- Konfiguraci připojení k databázi nastavit v context.xml
- Zkopírovat JDBC driver do příslušného adresáře Tomcatu (pokud spouštíme Tomcat 7.0.20 z prostředí NetBeans 7.0, jde o adresář ~/.netbeans/7.0/apache-tomcat-7.0.20_base/nblib)

© 2011 IBA CZ, s.r.o.

Připojení k databázi - příklad



META-INF/context.xml:

```
<Context antiJARLocking="true" path="/ContactManager">
<Resource name="jdbc/contactmgr"
    auth="Container" type="javax.sql.DataSource"
    driverClassName="org.apache.derby.jdbc.ClientDriver"
    username="app" password="app"
    url="jdbc:derby://localhost:1527/ContactManager"/>
</Context>
```

Přístup ze servletu prostřednictvím Dependency Injection

```
@Resource(name="jdbc/contactmgr")
private DataSource dataSource;
```

nebo

```
@Resource(name="jdbc/contactmgr")
void setDataSource(DataSource dataSource) {
    // ... Zde použít dataSource dle potřeby
}
```


© 2011 IBA CZ, s.r.o.

Architektura MVC

Prezentační vrstva je rozdělena na tři komponenty

- Model – reprezentuje data, s nimiž pracujeme
- View – je zodpovědná za zobrazení dat uživateli
- Controller – zpracovává vstup od uživatele a je zodpovědná za řízení celého procesu

Proč MVC

- Minimalizace závislostí
- Znovupoužitelnost
- Snadná změna nebo výměna kterékoliv komponenty s minimálním vlivem na ostatní komponenty
- Snadná údržba
- Za různé komponenty mohou mít zodpovědnost různí členové vývojového týmu
- Možnost využití různých technologií pro view vrstvu (JSP, XSLT, Šablony, PDF, apod.)

© 2011 IBA CZ, s.r.o.

Demonstrační příklad: Databáze kontaktů

Model = ContactManagerBackend

- Contact – reprezentuje entitu Kontakt
- ContactService – umožňuje kontakty ukládat do databáze a načítat je z databáze

View = JSP stránky

- listContacts.jsp – zobrazuje seznam kontaktů
- addContact.jsp – zobrazuje formulář pro přidání kontaktu

Controller = ActionServlet

- Při zobrazování dat tato data načte z databáze prostřednictvím ContactService a zpřístupní je jako atribut pro příslušnou JSP stránku
- Při odeslání formuláře pro přidání nového kontaktu načte z formuláře zadané hodnoty a prostřednictví ContactService je uloží do databáze
- Řídí page flow (určuje která stránka se v kterém okamžiku zobrazí)

© 2011 IBA CZ, s.r.o.

Best practices

Při vývoji webových aplikací dodržujte tyto zásady

- Používejte architekturu MVC a důsledně oddělujte aplikační vrstvu, prezentační vrstvu a kontrolér. Výrazně si tím usnadníte pozdější údržbu kódu.
- Používejte vhodný MVC framework, ušetří vám spoustu práce.
- Nikdy nepoužívejte scriplety, dejte přednost JSP značkám (např. JSTL) a EL.

Další zdroje informací

- http://kore.fi.muni.cz/wiki/index.php/Kategorie:Webov%C3%A9_aplikace

© 2011 IBA CZ, s.r.o.

Proč webové frameworky

Každá aplikace vyžaduje řešit standardní problémy

- Implementace architektury MVC
- Dekompozice aplikace s ohledem na udržovatelnost
- Propojení formulářů s modelem
- Validace vstupních dat, obsluha chyb.
- Internacionalizace a lokalizace
- Šablony rozložení (layout), skládání stránek
- Page flow
- atd.

Pokud použijeme webový framework

- Nemusíme opakovaně psát kód pro řešení výše zmíněných problémů a můžeme se soustředit na vlastní problémovou doménu
- Nemusíme znovuvynalézat kolo
- Většinu věcí řešíme *deklarativním způsobem*

Nedává žádný smysl psát webovou aplikaci bez použití vhodného webového frameworku

© 2011 IBA CZ, s.r.o.

Kontakty

IBA CZ

Petr Adámek
University Relations
petr.adamek@ibacz.eu

IBA CZ, s.r.o.
Petržílkova 2565/23
158 00 Praha 5

IBA CZ Development Center
Křenová 72
602 00 Brno

Tel.: (+420) 543 426 800
http://www.ibacz.eu/

© 2011 IBA CZ, s.r.o.