Manuál pro Spring-based DAO vrstvu (webkiv-db) aplikací webu Katedry informatiky a výpočetní techniky

Radek Muzika, březen 2008
úpravy a doplňky Přemek Brada

Úvodem

Jádro aplikace je postaveno nad Spring Frameworkem (dále jen SF), ze kterého jsou využity funkce IOC kontejneru a pro přístup do databáze je využita knihovna Spring JDBC.

IOC kontejner – při použití SF v aplikaci jsou veškeré základní objekty, potřebné pro chod aplikace (dao třídy, řídící třídy,...) odkazovány z IOC kontejneru a jsou nazývány beans. Bean je objekt instanciován, sestaven a řízen IOC kontejnerem. Beans a závislosti mezi nimi jsou definovány pomocí konfiguračních metadat. Konfigurační metadata mohou být buď ve formě xml dokumentu, či od javy 1.5 i jako anotace (v xml je uvedeno, že se konfigurace načte z anotací). V redsys backendu je využito konfigurace pomocí anotací.

Spring JDBC – usnadňuje práci s databází a odstiňuje programátora od tak nebezpečných úkonů, jako je zavírání všech statements a connections.

Více informací o SF 2.5: http://static.springframework.org/spring/docs/2.5.x/reference/index.html

Obsah knihoven

cz.zcu.kiv.db.model – obsahuje doménový model

cz.zcu.kiv.db.dao – obsahuje rozhraní pro DAO třídy

cz.zcu.kiv.db.dao.jdbc – obsahuje implementaci dao rozhraní za použití Spring JDBC knihovny

Implementační DAO třídy jsou vyčleněny mimo do samostatného balíku (dao.jdbc), jelikož je možné provést implementaci i pro jiné způsoby přístupu do databáze, jako např. Hibernate (doménový model je navržen tak, aby vyhovoval i případné implementaci ORM). Tento balík dále obsahuje některé třídy přetěžující doménové objekty. Tyto třídy jsou nezbytné pro implementaci Spring JDBC nad doménovým modelem. Obsahují atributy odpovídající cizím klíčům v databázi a dále jsou zde přetížené některé metody, které zajišťují načtení dodatečných informací svázaných s doménovým objektem z databáze. V implementaci metod je použit „lazy-load“, tj. data jsou načtena z databáze, až když jsou zapotřebí.

Jak použít webkiv-db backend

1) do aplikace přidáme knihony
   - webkiv-db – knihovny redsys backendu
   - spring – základní knihovna SF
   - log4j – vyzadovano springem
   - mysql-connector – knihovna zprostředkovávající připojení do db
2) nakonfigurujeme web.xml (zadáme cestu ke spring konfiguračnímu souboru)

```
<context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/redsys-servlet.xml</param-value>
</context-param>
```

```
<listener>
<listener-class>
                org.springframework.web.context.ContextLoaderListener
        </listener-class>
</listener>
```

*Tento krok je svázaný s medodou získání aplikačního kontextu ze servlet kontextu.

3) upravit parametry pro připojení v jdbc.properties

```
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/redsys
jdbc.username=root
jdbc.password=
```

Použití v JSP

Abychom mohli přistupovat k beans, potřebujeme získat aplikační kontext. Pokud jsme v konfiguraci provedli krok 2, pak získáme kontext ze servlet kontextu zadáním

```
org.springframework.context.ApplicationContext ctx =
WebApplicationContextUtils.getWebApplicationContext(config.getServletContext());
```

Pokud jsme neprovedli krok 2, pak lze získat kontext ještě dvěma způsoby a to buď zadáním cesty ke konfiguračnímu souboru, nebo umístěním konf.souboru do classpath. Pak by se použilo metod

```
org.springframework.context.ApplicationContext ctx = new
org.springframework.context.support.FileSystemXmlApplicationContext( "redsys-
servlet.xml");

org.springframework.context.ApplicationContext ctx = new
org.springframework.context.support.ClassPathXmlApplicationContext( "redsys-servlet.xml");
```

Bean z kontextu získáme metodou

```
UdalostDao uDao = (UdalostDao) ctx.getBean("udalostDao");
```

Získáním dao objektu můžeme přistupovat pomocí metod objektu do nakonfigurované databáze.

```
List<Udalost> seznamUdalosti = uDao.getUdalosti();
pageContext.setAttribute("seznamUdalosti", seznamUdalosti);
```

A výsledky zobrazíme pomocí jstl.

```
<c:forEach items="${seznamUdalosti}" var="udalost">
        <c:out value="${udalost.nazevCz}" />
</c:forEach>
```