Spring Framework
Module 6 – ORM Support

Evgeniy Krivosheev
Vyacheslav Yakovenko
Last update: Feb, 2012

Contents

Spring :: Overview of ORM module

- org.springframework.orm
- org.springframework.orm.hibernate3
- org.springframework.orm.hibernate4
- org.springframework.orm.ibatis
- org.springframework.orm.jdo
- org.springframework.orm.jpa

Spring :: ORM: Overview of Deprecated API

- Spring Framework v.2.* supported ORM through XxxTemplate classes:
    - @Deprecated JpaTemplate, @Deprecated JpaCallback<T>;
    - @Deprecated JdoTemplate, @Deprecated JdoCallback<T>; ;
    - org.springframework.orm.hibernate3.HibernateTemplate ;
- Trend of Spring Framework v.3 is moving away from XxxTemplate and switching to the most native API, particular ORM:
    - JPA:
        - LocalEntityManagerFactoryBean
        - LocalContainerEntityManagerFactoryBean
    - Hibernate:
        - org.springframework.orm.hibernate3.LocalSessionFactoryBean
        - org.springframework.orm.hibernate4.LocalSessionFactoryBean

Spring :: Benefits of Working with ORM

- Easier testing;
- Exceptions handling;
- General resource management (DataSource, mappings);
- Integrated transaction management ;

Spring :: JPA + Hibernate

- Nowadays JPA is a commercial standard, while Hibernate (as of v.3.2) is a JPA implementation. Therefore, during the training we will examine this alternative: using Hibernate 4 as JPA 2.0 provider;
- Please note that next Spring Framework versions will not support JPA v.1.0;
- Besides, Spring Framework starting from v.3.0 doesn’t support Hibernate versions below 3.2;

Spring :: JPA Setup

Currently Spring offers three ways of setting up JPA EntityManagerFactory :

- Obtaining an EntityManagerFactory from JNDI;
- Using LocalEntityManagerFactoryBean:
  - Persistence.xml that is mandatory from JPA standard point of view is not required;
  - Used in simple applications and prototypes for testing;
- LocalContainerEntityManagerFactoryBean is a factory that gives full control:
  - Supports multiple persistence units;
  - Can be configured for various application servers (WebLogic, OC4J, GlassFish, Tomcat, Resin, JBoss)

Spring :: JPA Setup

Obtaining EntityManagerFactory from JNDI:

```
<jee:jndi-lookup id="myEmf"
     jndi-name="persistence/myPersistenceUnit"/>
```

Spring :: JPA Setup

Using LocalEntityManagerFactoryBean:

```
<bean id="myEmf"
   class="org.springframework.orm.jpa.LocalEntityManagerFactoryBean">
   <property name="persistenceUnitName"   value="myPersistenceUnit"/>
</bean>
```

Spring :: JPA Setup

LocalContainerEntityManagerFactoryBean is a factory that gives full control:

```
<bean id="myEmf"
   class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
   <property name="dataSource" ref="dataSource"/>
   <property name="loadTimeWeaver">
        <bean
   class="org.springframework.instrument.classloading.InstrumentationLoadTimeWeaver"/>
   </property>
   <property name="persistenceUnitName" value="persistenceUnitName" />
</bean>
```

Spring :: JPA Setup

If using Hibernate 4 as JPA provider, an additional configuration is needed.

application-context.xml:

```
<bean id="lcemf" class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="loadTimeWeaver">
           <bean class="org.springframework.instrument.classloading.InstrumentationLoadTimeWeaver" />
    </property>
    <property name="dataSource" ref="dataSource"></property>
    <property name="persistenceUnitName" value="springframework.lab.orm.jpa" />
    <property name="persistenceProviderClass"     value="org.hibernate.ejb.HibernatePersistence"/>
</bean>

<bean id="countryDao" class="lab.dao.jpa.CountryJpaDaoImpl" />
```

META-INF/persistence.xml:

```
<persistence>
    <persistence-unit name="springframework.lab.orm.jpa">
           <class>lab.model.Country</class>
           <properties>
                  <property name="hibernate.show_sql" value="true" />
                  <property name="hibernate.hbm2ddl.auto" value="create" />
           </properties>
    </persistence-unit>
</persistence>
```

Spring :: JPA, Query Example

Weaving example:

```
@Repository
public class CountryJpaDaoImpl {
  protected EntityManagerFactory emf;

  @PersistenceUnit
  public void setEntityManagerFactory(EntityManagerFactory emf) {
      this.emf = emf;
  }

  public List<Country> getAllCountries() {
      EntityManager em = emf.createEntityManager();
      return = em.createQuery("from Country", Country.class);
  }
}
```

Spring :: JPA, Query Example

Weaving is performed through method annotated as **@PersistenceUnit**:

- Spring calls **LocalContainerEntityManagerFactoryBean**;
- Obtains **EntityManagerFactory**;
- Using autoweaving mechanism injects into DAO implementation;
- When you have one **EntityManagerFactory** instance, call it for executing queries.

Exercises

№: 7 : Using ORM in Spring when handling data

– 45 min for practice;

– 15 min for discussion;

Any questions!?