Aprameyah
Technologies Pvt. Ltd.

Spring-Hibernate Integration

Aprameyah Technologies Pvt. Ltd.

Integrate Hibernate With Spring

- The Hibernate configuration can be defined in spring's context xml

- SessionFactory can use any pooled database connection bean available in spring

- DAO classes can get access to SessionFactory via dependency injection

Aprameyah
Technologies Pvt. Ltd.

Choose A Datasource

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
      destroy-method="close">
    <property name=“driverClassName" value="org.apache.derby.jdbc.ClientDriver"/>
    <property name=“url" value=“jdbc:derby://localhost:1527/…” />
    <property name="username" value="APP" />
    <property name="password" value="app" />
</bean>
```

This can also be a Appserver controlled JNDI based connection pool. Mandatory for JTA transactions

Aprameyah
Technologies Pvt. Ltd.

Configuring Session Factory

```
<bean id="mySessionFactory"
class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
   <property name="dataSource" ref="dataSource"/>
   <property name="mappingResources">
      <list>
         <value>Users.hbm.xml</value>
         <value>Orders.hbm.xml</value>
      </list>
   </property>
   <property name="hibernateProperties">
      <value>
         hibernate.dialect=org.hibernate.dialect.DerbyDialect
         hibernate.show_sql=true
      </value>
   </property>
</bean>
```

Aprameyah
Technologies Pvt. Ltd.

Changes to DaoImpl

```
public class UserDaoImpl extends HibernateDaoSupport implements UserDao{
    public List<User> getAllUsers() throws Exception {

        Session ses = getSessionFactory().openSession();
        List<User> users = ses.createQuery("select u from User u").list();
        ses.close();
        return users;
    }
}

<bean id="userDao"
        class="com.mydomain.service.UserDaoImpl">
    <property name="sessionFactory" ref="mySessionFactory" />
</bean>
```

Aprameyah
Technologies Pvt. Ltd.

Transactions With Spring

```
<bean id="txManager"
class="org.springframework.orm.hibernate4.HibernateTransactionManager">
   <property name="sessionFactory" ref="mySessionFactory" />
</bean>

<tx:advice id="txAdvice" transaction-manager="txManager">
     <tx:attributes>
      <tx:method name="*" propagation="REQUIRED"/>
   </tx:attributes>
</tx:advice>

<aop:config>
   <aop:pointcut id="userOperations" expression="execution(*
com.mydomain.service.UserManagerImpl.*(..))"/>
   <aop:advisor advice-ref="txAdvice" pointcut-ref="userOperations"/>
</aop:config>
```

Aprameyah
Technologies Pvt. Ltd.

Lets Try It

- Create a method “generateOrder” that takes a user id and creates a record in the Order table. Throw an exception if User name is “admin” because admin cannot have orders

- Create a method “generateOrders” in UserManager that generates an order for each user in the database with the help of above method.

Obtaining Connections For Managed Transactions

```
Session ses = getSessionFactory().getCurrentSession();
List<User> users = ses.createQuery("select u from User u").list();
return users;
```

Aprameyah
Technologies Pvt. Ltd.

Propagation Configuration

- Change propagation configuration so that we succeed in creating orders for other users even if there is failure for one

```
<tx:advice id="txAdvice" transaction-manager="txManager">
        <tx:attributes>
         <tx:method name="generateOrder" propagation="REQUIRES_NEW"/>
            <tx:method name="*" propagation="REQUIRED"/>
        </tx:attributes>
    </tx:advice>
```