

Hibernate & Spring

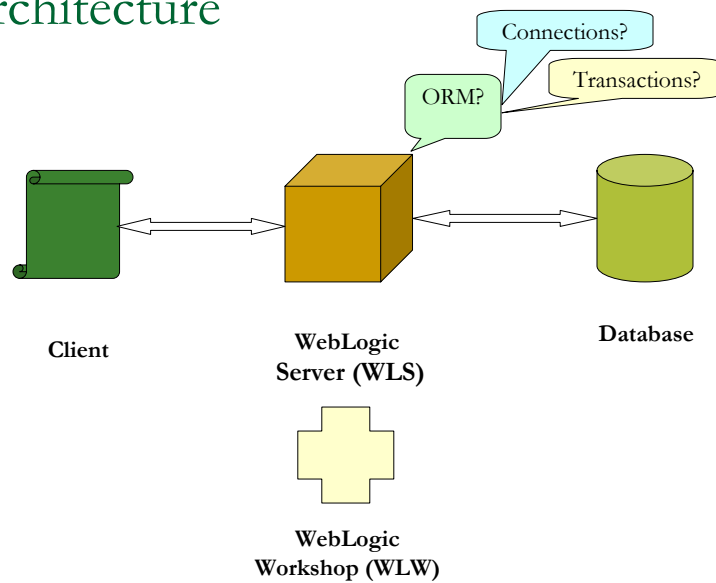
Prakash Malani

Email: prakash_at_malani_dot_org

Blog: <http://dev2dev.bea.com/blog/pmalani/>

Web: <http://www.bartssandbox.com/>

Architecture



Create Database & Populate

- When I create a WLW domain, an instance of PointBase is also created
 - I login to PointBase console (default username is weblogic and password is weblogic)
 - I create a user named sem with password sem
 - I login in as sem user
 - I create a schema named sem
 - I execute the sem.sql script to create a sem_person table and insert a row
-

WebLogic Workshop Domain

- Run config.py WLST script to configure the following:
 - JDBC Connection Pool
 - JDBC Data Source
 - JMS Server
 - JMS Connection Factory
 - JMS Queue
-

Hibernate

Configuration & Usage

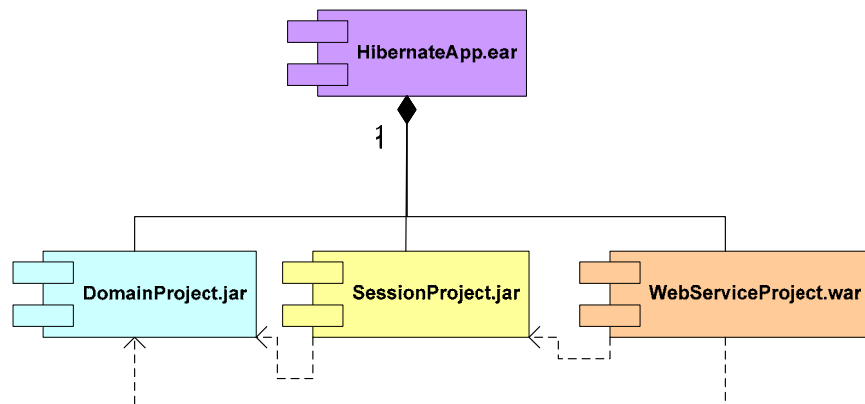
Hibernate Overview

- Map classes (tables) and attributes (columns)
 - Map relationships, cardinality, and navigability
 - Map inheritance
 - Navigation and query
 - Transaction Management
 - Performance (caching, field-groups, lazy-loading)
 - Locking and concurrency
 - Native SQL support
-

Incorporate Hibernate

- Locate the jar files mentioned here:
file:///C:/hibernate-3.0.5/doc/reference/en/html_single/index.html#3rdpartylibs under ...\\hibernate\\lib folder
- Drag-and-drop the hibernate3.jar plus the above jar files into Libraries folder in WLW or under APP-INF/lib folder on the disk

WebLogic Workshop (WLW) Application Component View



Hibernate Configuration

- There are lots of different ways to configure Hibernate
 - I will highlight our choices
 - I will store the main Hibernate configuration in hibernate.cfg.xml file
-

Hibernate Connection Configuration

- DriverManager
 - C3PO (and other connection pool implementations)
 - I am going to use WebLogic connection pool and datasource by configuring connection.datasource property
-

Hibernate Programmatic Transaction Configuration

- Use JDBC API directly
- Use JTA API directly (Client as well as BMTD)
- Hibernate Transaction API
 - Hibernate Transaction API provides identical and portable programming model to do transactions over JDBC or JTA
 - To use JDBC configure `hibernate.transaction.factory_class` as `JDBCTransactionFactory` (**default**)
 - To use JTA configure `hibernate.transaction.factory_class` as `JTATransactionFactory`

Programmatic Transaction Pseudo-code

```
Session aSession = aFactory.openSession();
Transaction t = null;
try {
    t = aSession.beginTransaction();
    // do work
    t.commit();
} catch (RuntimeException e) {
    if (t != null) t.rollback();
    throw e;
} finally {
    aSession.close();
}
```

Hibernate Declarative Transaction Configuration

- JTA with CMTD
- No need to use any transaction API
- Transactional boundaries defined in the deployment descriptors
- To use JTA configure `hibernate.transaction.factory_class` as `CMTTransactionFactory`
- Opening and closing of Session is still required (See Spring's `HibernateTemplate`)
- Example: `DomainManagerBean.getPerson()`

Hibernate & WebLogic JTA

- To configure Hibernate with WebLogic JTA configure the `transaction.manager_lookup_class` property to be WebLogic-specific value of `org.hibernate.transaction.WeblogicTransactionManagerLookup`

Hibernate 3 & JTA

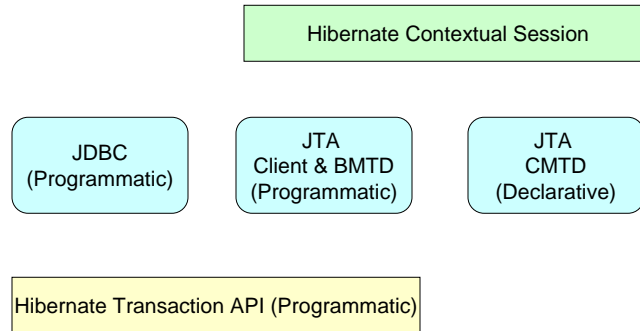
- Hibernate 3 introduces context session
 - The session is obtained using `factory.getCurrentSession()`
 - No need to open and close session as they are also driven by JTA transaction boundaries
 - Much simpler programming model
 - Example: `DomainManagerBean.getPerson3()`
 - Example: Compare `getPerson()` with `getPerson3()`

Declarative Transaction & Contextual Session Pseudo-code

```
Session aSession
    = aFactory.getCurrentSession();

// do work
```

Hibernate Transaction Option Summary



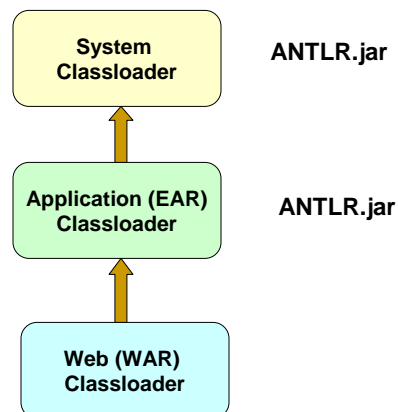
Obtaining the SessionFactory

- Setting the value for `setting_factory_name` property makes an instance of `SessionFactory` available in JNDI with the specified value
 - Example: `DomainManagerBean.getPerson2()`
- Manually bind the instance to JNDI via a start-up class or a start-up servlet
 - However, this did not work for me in a reliable way
- I just added an accessor for the `SessionFactory` in the `HibernateUtil` class
 - Example: `DomainManagerBean.getPerson3()`

Implement using Hibernate

- Implement domain class as POJO
 - Write the test.domain.Person.java class
 - Write the corresponding Person.hbm.xml file
 - Add Person.hbm.xml as mapping-resource to hibernate.cfg.xml
- Implement the Stateless Session Bean (SLSB) with CMTD
 - Example: DomainManagerBean.getPerson3()
 - Example: DomainManagerBean.getPersons()

Oops... The Server Crashed!!!



- The Hibernate 3 query uses ANTLR
- However, ANTLR does not use context class loader
- The class is not found causing a fatal crash
- One option is to use Hibernate 2 query parser
- The other option is to add the patched version of the antlr-2.7.5H3.jar to the PRE_CLASSPATH

Spring Framework

Configuration & Usage

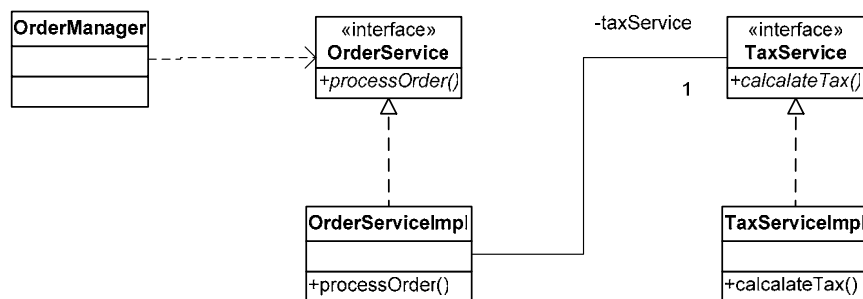
Spring Framework Concepts

- Dependency Injection
- Template Method Pattern
- Aspect Oriented Programming
- Encourages good programming practices like programming to interfaces
- Better exception handling (from checked to unchecked, more exception types, etc.)

Incorporate Spring

- Drag-and-drop the spring.jar into the Libraries folder in WLW or under APP-INF/lib folder on the disk
- I will do the Spring configuration in applicationContext.xml file

Dependency Injection

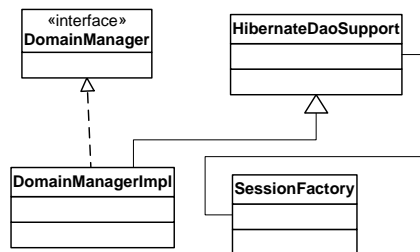


Dependency Injection (DI) Notes

- DI makes most sense for service / manager type objects (not domain type objects)
- Interface and implementation separation
- TaxService is injected into OrderServiceImpl
- OrderService (and its client like OrderManager) have no dependency on TaxService (or TaxServiceImpl)
- If I need a bean, the preferable way is for the bean to be injected
 - However, I can directly access the bean from the context (see OrderManager.jws)

Hibernate Data Access Object (DAO)

- Spring provides HibernateDaoSupport abstract class
- The SessionFactory class is injected into our DAO
- Example: bean id sessionFactory in applicationContext.xml
- Note: The normal way to create beans is with constructor. However, this example illustrates using a factory method (e.g. bean id sessionFactory)



Configure Hibernate with Spring

- I can configure Hibernate directly inside the applicationContext.xml
- Therefore, I can remove the need for hibernate.cfg.xml file
- The process and method of configuration is exactly identical
- Unlike hibernate.cfg.xml, the properties are prefixed with hibernate.
- Note: This example illustrates how bean's can be defined by looking them up in JNDI (e.g. bean id semJDBCDataSource)

Spring's HibernateTemplate

- (If I don't use the Hibernate 3 contextual session) I need to open and close the Session from the factory
- GoF **template method pattern** is an ideal pattern that allows us to change the algorithm based on inheritance
- Template method pattern applies very well to creating a template to handle resource allocation and de-allocation while leaving the **meat** to sub-classes (a code-smell known as TCFTC or TCTCFTC)
- HibernateTemplate is Spring's template for Hibernate Sessions
 - In fact, Spring provides such templates for many other resources (JDBC, JMS, JDO, etc.)
 - The **meat** is specified to the HibernateTemplate in terms of HibernateCallback implementation (Example: DomainManagerImpl.getPerson4())
 - However, HibernateTemplate provides utility methods for many single step methods such as get, find, etc. so I don't have to provide an instance of HibernateCallback (Example: DomainManagerImpl.getPerson3())
 - The easiest way to obtain an instance of HibernateTemplate is the **accessor method on the HibernateDaoSupport**

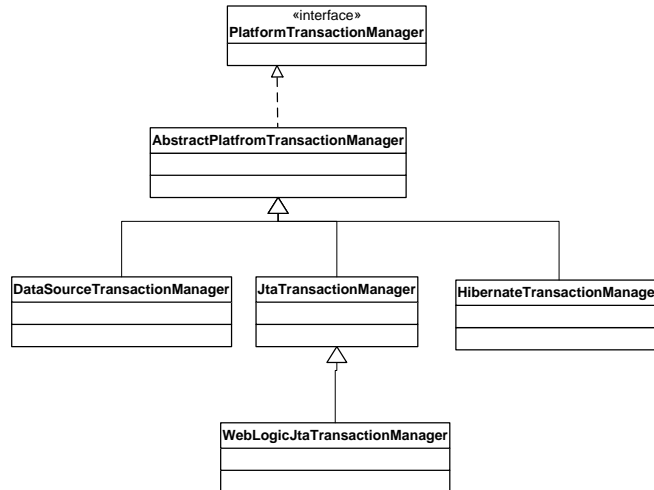
Spring & Hibernate Contextual Session

- I use Hibernate contextual session using `getSessionFactory().getCurrentSession()` method (Example: `DomainManagerImpl.getPerson2()`)
- I don't need to use `HibernateTemplate`

Spring Transaction Options

- Programmatic Options
 - Use JDBC API directly
 - Use JTA API directly (Client and BMTD)
 - Use Hibernate Transaction API directly
 - Use Spring Transaction API directly
 - Use `PlatformTransactionManager` interface directly (actually any implementation)
 - The `PlatformTransactionManager` class is very similar to `javax.transaction.UserTransaction`
 - Use `TransactionTemplate` on `PlatformTransactionManager` interface
- Declarative Options
 - Use JTA API (EJB CMTD)
 - Use Spring declarative transactions (similar to EJB CMTD, but more flexible) implemented using AOP
 - Use `PlatformTransactionManager` interface (actually any implementation)

Spring Transaction Manager



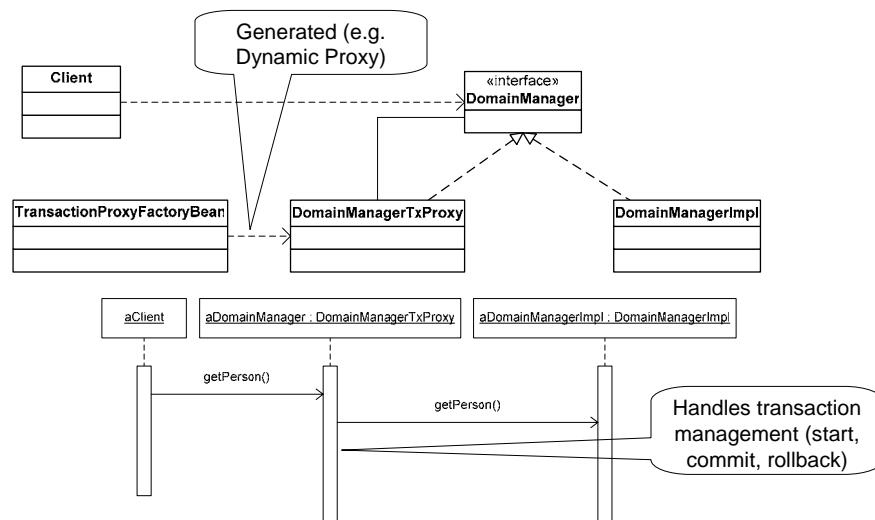
Spring Transaction Manager Configuration

- Use JDBC API
 - Configure transaction manager bean as `DataSourceTransactionManager`
 - Set the datasource as the property
- Use JTA API
 - Configure transaction manager bean as `JtaTransactionManager`
 - I use `WebLogicJtaTransactionManager` instead of `JtaTransactionManager` as it provides better integration with `WebLogic`
- Use Hibernate Transaction API
 - Configure transaction manager bean as `HibernateTransactionManager`
 - Set the `SessionFactory` as the property

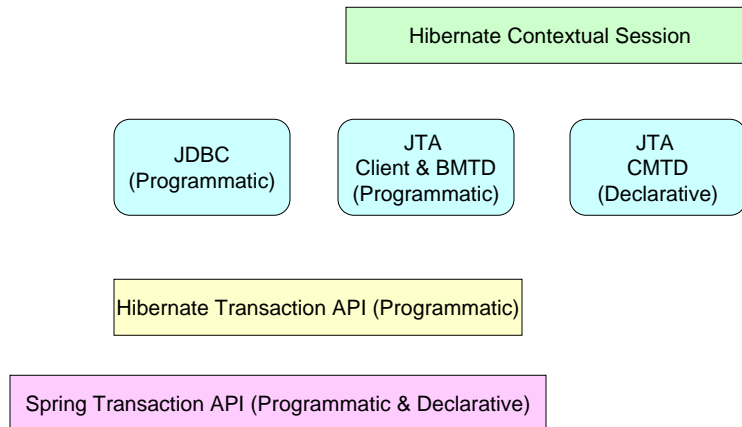
Declarative Transaction Configuration in Spring

- Define the transaction manager bean
 - Example: bean id txManager
- Define a bean of type TransactionProxyFactoryBean
 - Example: bean id **domainManagerTx**
- This bean handles transactions and delegates to a simple POJO (another bean)
 - Example: bean id domainManagerInline
- The other bean is specified using the target property
- Transaction attributes are specified for the bean
- The transaction attributes are identical to EJB CMTD transaction attributes
- There is one additional attribute that supports nested transactions

Transaction Proxy



Spring Transaction Option Summary



Futures

- BEA is offering support for Spring in WebLogic 9
- This article describes WebLogic and Spring integration in detail:
http://dev2dev.bea.com/pub/a/2005/09/spring_integration_weblogic_server.html
- The integration is available here:
<http://commerce.bea.com/showproduct.jsp?family=SPRING&major=1.2.5&minor=0>

References

- Spring Quick Start Tutorial (http://www.sourcebeat.com/docs/Spring%20Live/Rev_10/Spring%20Live_SampleChapter.pdf)
- Implementing Transaction Suspension in Spring (<http://dev2dev.bea.com/lpt/a/40>)
- I'm having problems with WebLogic Server! (<http://www.hibernate.org/120.html#A8>)
- WebLogic Server Application Classloading (<http://e-docs.bea.com/wls/docs81/programming/classloading.html>)
- Strategies for WebLogic Domain Configuration (<http://groups.yahoo.com/group/bartssandbox/files/WebLogicDomainConfigurationOptions.pdf>)

My Blog

- Configuring WebLogic 8.1 and Hibernate 3.0 (http://dev2dev.bea.com/blog/pmalani/archive/2005/07/configuring_web.html)
- Configuring WebLogic 8.1 and Hibernate 3.0 – Part II (http://dev2dev.bea.com/blog/pmalani/archive/2005/07/configuring_web_1.html)
- WebLogic and Spring – Part III (http://dev2dev.bea.com/blog/pmalani/archive/2005/08/weblogic_and_sp.html)

Questions?
