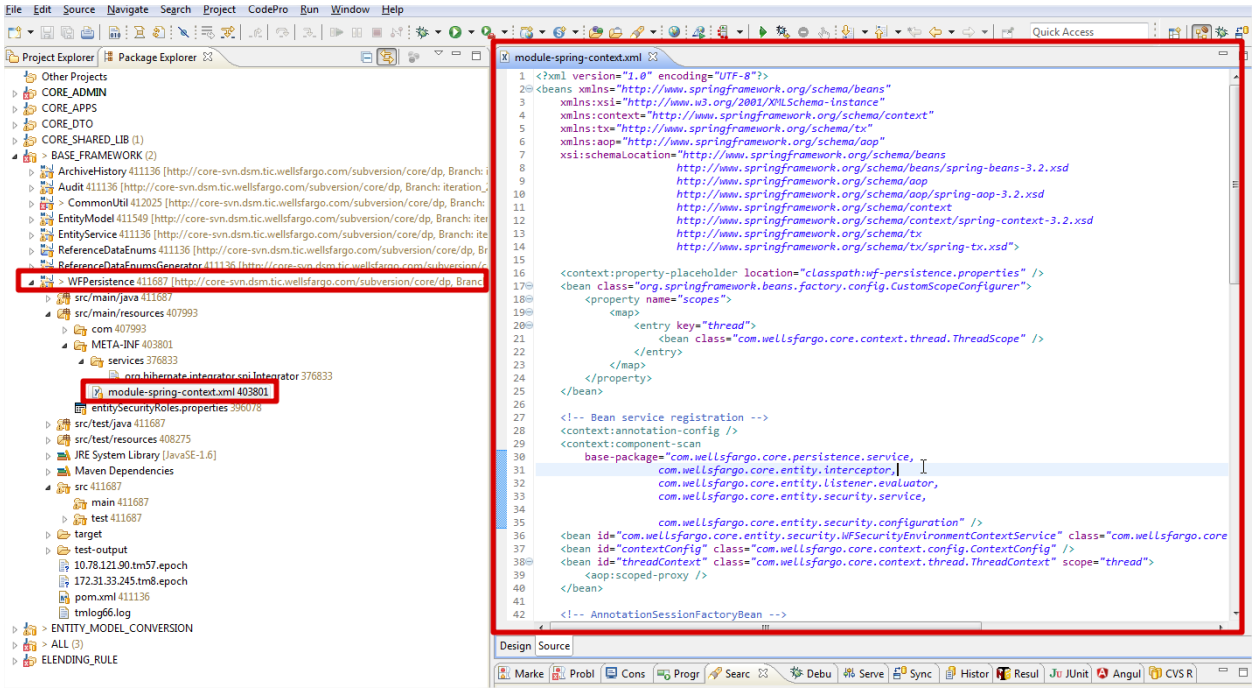
Team Learning Session 4 Meeting Notes: **TRANSACTION MANAGEMENT IN CORE**

TOPICS DISCUSSED:

S.NO	Topic
1	Configuration
2	Sample Snippet
3	Importance of “tx:annotation-driven”
4	Type of Transaction Manager being used
5	Other important info

Date	**05-27-2015**
Topics Covered by:	Sudipta Sarkar, Venkat M., Ramesh K.
Document Compiled by:	Krishna Bodduluri
Document Version	1.0

1. **What is the key configuration files?**
   Module-spring-context.xml



2. Example of Default Transaction from the Application?

```
@Transactional
public List<ProductRequest> getProductRequestsExceptHE(final String dealId) throws Throwable {
    return (List<ProductRequest>) persistenceService.findByNamedQuery("bpdHelperGetProductRequestsExceptHEByDealId", new String[] {"dealId"}, new Object[] {new Long(dealId)});
}
```

3. **What's the importance of "tx:annotation-driven" element from configuration file?**

It scans all beans in the application context and creates AOP interceptor for those which are annotated. This is done via the SpringTransactionAnnotationParser, which is used by TransactionInterceptor. Then whenever these beans are accessed, this advice is triggered and a transaction is started before the target method is executed, and committed after the execution.

module-spring-context.xml

```
68          </property>
69          <!-- <property name="configLocation" value="hibernate.cfg.xml" /> -->
70          <property name="hibernateProperties">
71              <props>
72                  <prop key="SESSION_FACTORY_ID">CORE</prop>
73                  <prop key="hibernate.hbm2ddl.auto">${hibernate.hbm2ddl.auto}</prop>
74                  <prop key="hibernate.dialect">${hibernate.dialect}</prop>
75                  <prop key="hibernate.cache.use_second_level_cache">${hibernate.cache.use_second_level_cache}</pro
76                  <prop key="hibernate.cache.use_query_cache">${hibernate.cache.use_query_cache}</prop>
77                  <prop key="net.sf.ehcache.configurationResourceName">${net.sf.ehcache.configurationResourceName}<
78                  <prop key="hibernate.cache.region.factory_class">${hibernate.cache.region.factory_class}</prop>
79                  <prop key="hibernate.show_sql">${hibernate.show_sql}</prop>
80                  <prop key="hibernate.jdbc.batch_size">${hibernate.jdbc.batch_size}</prop>
81                  <prop key="hibernate.default_fetch_size">${hibernate.default_fetch_size}</prop>
82                  <prop key="hibernate.default_batch_fetch_size">${hibernate.default_batch_fetch_size}</prop>
83
84                  <prop key="hibernate.transaction.factory_class">${hibernate.transaction.factory_class}</prop>
85                  <prop key="hibernate.transaction.jta.platform">${hibernate.transaction.jta.platform}</prop>
86                  <prop key="hibernate.current_session_context_class">${hibernate.current_session_context_class}</p
87                  <prop key="hibernate.id.new_generator_mappings">true</prop>
88                  <prop key="hibernate.id.optimizer.pooled.prefer_lo">true</prop>
89                  <prop key="hibernate.listeners.envers.autoRegister">false</prop>
90              </props>
91          </property>
92      </bean>
93
94      <bean id="hibernateConfiguration" factory-bean="&amp;sessionFactory" factory-method="getConfiguration" />
95
96      <!-- <bean id="txManager" class="org.springframework.orm.hibernate4.HibernateTransactionManager">
97          <property name="sessionFactory" ref="sessionFactory" />
98      </bean> -->
99      <bean id="transactionTemplate"  class="org.springframework.transaction.support.TransactionTemplate">
100         <property name="transactionManager" ref="transactionManager"></property>
101     </bean>
102     <bean id="jtaPlatformAdaptor" class="com.wellsfargo.core.persistence.transaction.JtaPlatformAdaptor">
103         <property name="jtaTransactionManager" ref="transactionManager" />
104     </bean>
105     <context:load-time-weaver weaver-class="com.wellsfargo.core.common.util.WfContextLoadTimeWeaver"/>
106     <tx:annotation-driven transaction-manager="transactionManager" mode="aspectj" />
107
108 </beans>
109
```

4. **What is the Transaction model we have in the application?**

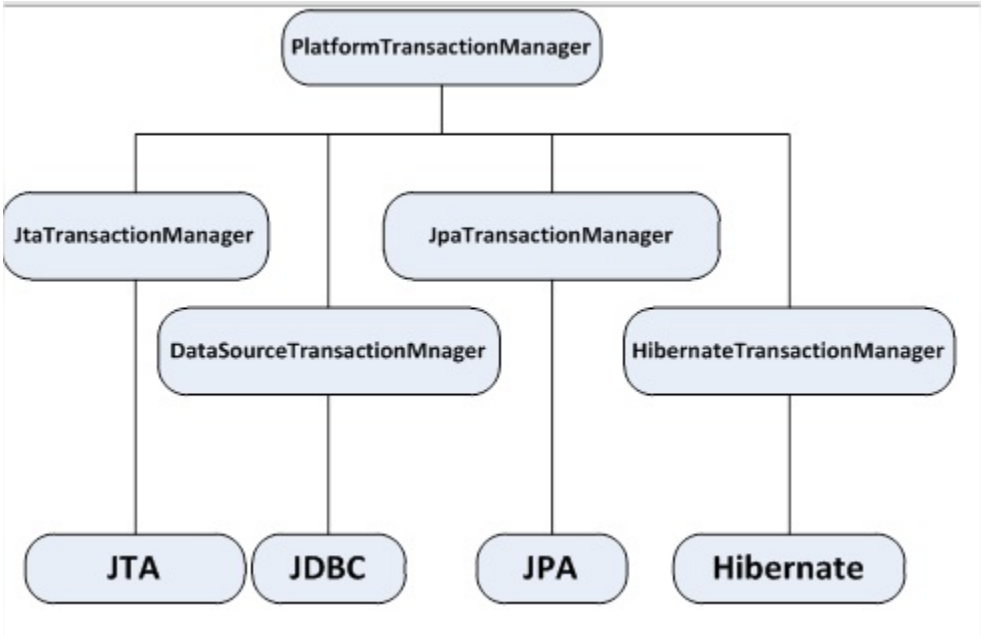
   We have the Transaction wrapped at Implementation classes of Business Service Layer. Calls to Entity Layer are being made from Business Service Layer, which is where all the Transactions get initialized as per configuration file.

5. **Are we using DMT or PMT?**

   We are using Container Managed Transaction. We declare the annotations and let the Container to take care of controlling the annotations

6. **Are we using nested transactions?**
   Yes. We are using nested Transactions. Method 1 will call Method 2 and Method 2 will call Method 3.If we initiate the transaction at Method 1, it will be a nested transaction. We are doing that on several methods in the application.

7. **What type of Transaction Manager are we using?**
   PlatformTransactionManager



8. **Are we wrapping key methods like save, update, delete automatically with in a transaction from configuration file or are we doing it manual?**
   No. There is no separate configuration or configuration file for handling the Transactions on key methods.

9. **Are we using any remote transactions too?**
   No. We don't have any applications the needs distributed transactions.

10. **In terms of locking from multiple sources, how are we controlling our Transactions?**
    If we take a Quartz job as an example, we have configured all the jobs in Serial mode. One job runs at a time. Transaction Lock acquired by whatever the current job is running. Next job starts only after the 1st job has finished its execution.

Here is the Wiki link to know more about Transactions:
http://t01drw2510.dsm.tic.wellsfargo.com:8090/display/CUF/Transaction?src=search