Spring-Hibernate Training Program
Session 9

BoTree Consulting

Aspect Oriented Programming

- OOP – Unit of Modularity is Class
- AOP – Unit of Modularity is Aspect
- Crosscutting Concerns
- Non Invasive
- Declaritive AOP

AOP Concepts

- Aspect – modulatization of a crosscutting concern
- Join Point – always represents a *method execution*
- Advice – action taken by an *Aspect* at a *Join Point*
- Pointcut – a predicate that matches *Join Points*
- Target Object – also known as *Advised Object*
- AOP Proxy – proxy of the *Target Object*
- Weaving - linking aspects with other application types or objects to create an advised object

Why use Spring Transaction?

- Declarative
- Programatic
- Managed Resources
- Non Invasive

Spring Transaction

- Core APIs
  - Platform Transaction Manager Interface
    - TransactionStatus getTransaction(TransactionDefinition definition) throws TransactionException
    - commit (TransactionStatus status) throws TransactionException
    - rollback (TransactionStatus status) throws TransactionException
  - TransactionDefinition
    - Isolation
    - Propagation
    - Timeout
    - Read-only status

TransactionDefinition - Isolation

- Isolation
  - Dirty Reads – able to read uncommitted data
  - Non Repeatable Reads – second read, different data
  - Phantom Reads – second read, lost data
  - DEFAULT
  - READ_COMMITTED – dirty reads prevented
  - READ_UNCOMMITTED – all can occur
  - REPEATABLE_READ – phantom reads can occur
  - SERIALIZABLE – all prevented

TransactionDefinition - Propogation

- Propogation
  - PROPOGATION_REQUIRED – support current, create new
  - PROPOGATION_SUPPORTS – support current only
  - PROPOGATION_MANDATORY – support curent only
  - PROPOGATION_REQUIRES_NEW – suspend current, create new
  - PROPOGATION_NOT_SUPPORTED – suspend current, use none
  - PROPOGATION_NEVER – exception if transaction exists
  - PROPOGATION_NESTED – create withing existing, not supported by HibernateTransactionManager

Transaction Status

- hasSavePoint()
- isCompleted()
- isNewTransaction()
- isRollbackOnly()
- setRollbackOnly()

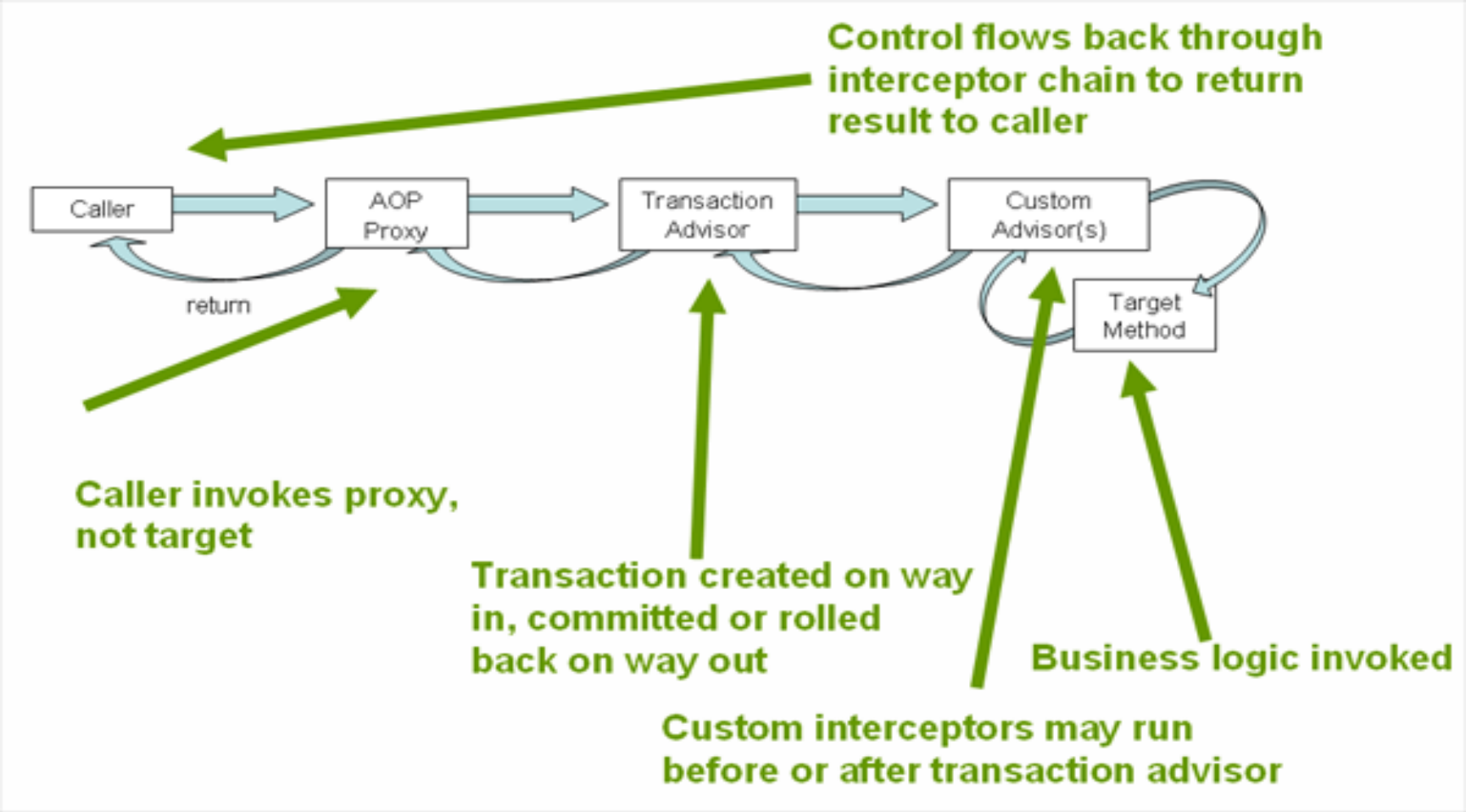
AbstractPlarformTransactionManager

- Workflow Handling
  - determines if there is an existing transaction
  - applies the appropriate propagation behavior
  - suspends and resumes transactions if necessary
  - checks the rollback-only flag on commit
  - applies the appropriate modification on rollback (actual rollback or setting rollback-only)
  - triggers registered synchronization callbacks (if transaction synchronization is active)

Transaction Synchronization

- Transaction Synchronization
  - High-level synchronization approach
    - using Spring's highest level template based persistence integration APIs OR use navtive ORM APIs with transaction- aware factory beans or proxies for managing the native resource factories.
    - relieve application code from handling resource creation and reuse, cleanup, optional transaction synchronization of the resources, and exception mapping
  - Low-level synchronization approach
    - using SessionFactoryUtils for Hibernate to get Spring managed instances
    - application code to deal directly with the resource types of the native persistence APIs

Spring's Declarative Transaction Impl



Hibernate Transaction Manager

org.springframework.orm.hibernate3

Class HibernateTransactionManager

java.lang.Object
  └ org.springframework.transaction.support.AbstractPlatformTransactionManager
    └ org.springframework.orm.hibernate3.HibernateTransactionManager

All Implemented Interfaces:

Serializable, BeanFactoryAware, InitializingBean, PlatformTransactionManager

@Transactional

- Can be placed
  - an interface definition
  - a method of an interface
  - before a definition
  - before a public method definition
  - Spring recommends using @Transactional only for concrete classes.

XML Configuration

```
<!-- enable the configuration of transactional behavior
based on annotations →

   <tx:annotation-driven transaction-
   manager="txManager"/>

<!-- a PlatformTransactionManager is still required →

<bean id="txManager"

   class="org.springframework.orm.hibernate3.HibernateTr
   ansactionManager">

<!-- (this dependency is defined somewhere else) -->

   <property name="dataSource" ref="dataSource"/>

</bean>
```

End of
Spring – Hibernate Training Program
Session 9