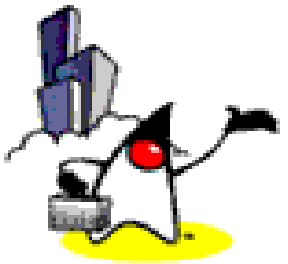
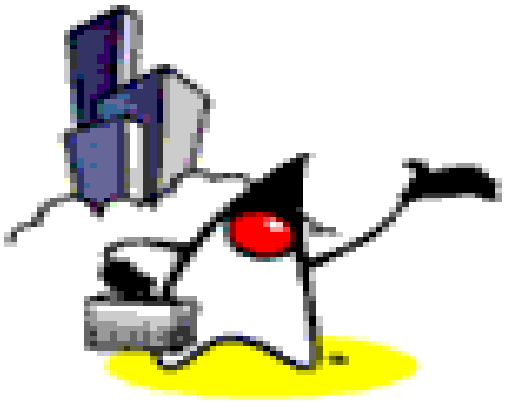
Spring Transaction



Topics

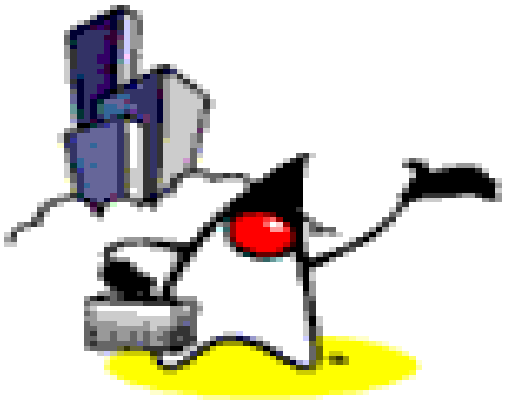
- Transaction types
- Isolation levels
- Propagation
- Transaction support in Spring
- Declarative transaction



Transaction Types

Transaction Types

- Local transaction
  - Specific to a single transactional resource (example: JDBC)
- Global transaction
  - Managed by container
  - Can span multiple transactional resources



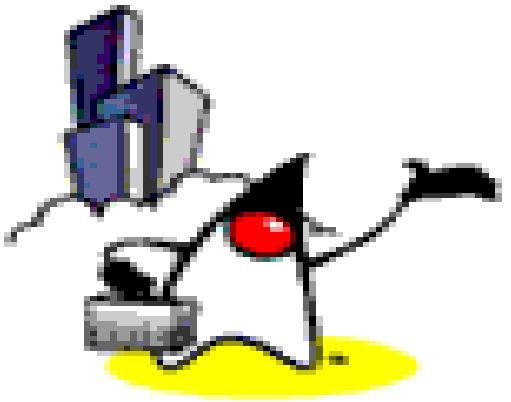
Transaction Isolation Levels

Transaction Isolation Levels

- You can specify per method

Transaction Isolation Levels

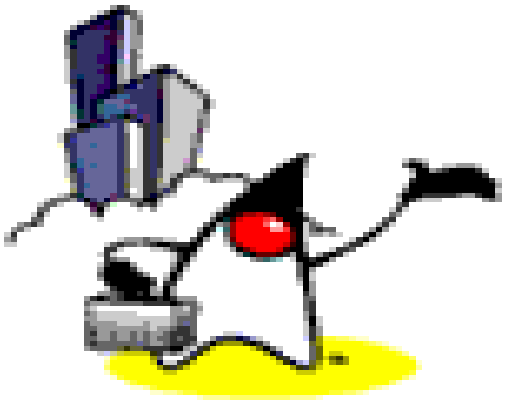
- ISOLATION_DEFAULT
- ISOLATION_READ_UNCOMMITTED
  - Dirty reads, non-repeatable reads and phantom reads can occur.
- ISOLATION_READ_COMMITTED
  - Dirty reads are prevented; non-repeatable reads and phantom reads can occur.
- ISOLATION_REPEATABLE_READ
  - Dirty reads and non-repeatable reads are prevented; phantom reads can occur.
- ISOLATION_SERIALIZABLE
  - Dirty reads, non-repeatable reads and phantom reads are prevented



Transaction Propagation

Transaction Propagation

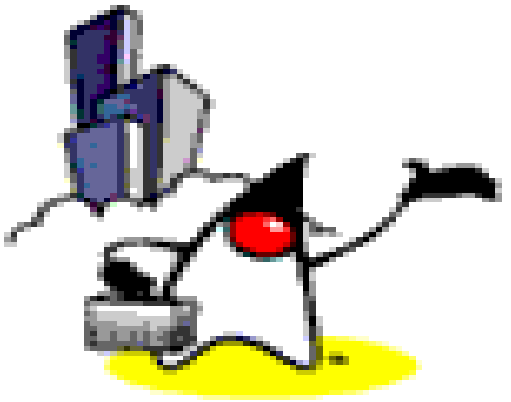
- PROPAGATION_REQUIRED
  - Support a current transaction, create a new one if none exists.
- PROPAGATION_SUPPORTS
  - Support a current transaction, execute non-transactionally if none exists.
- PROPAGATION_MANDATORY
  - Support a current transaction, throw an exception if none exist
- PROPAGATION_REQUIRES_NEW
  - Create a new transaction, suspend the current transaction if or exists.
- PROPAGATION_NOT_SUPPORTED
- PROPAGATION_NEVER
- PROPAGATION_NESTED



Transaction Support in Spring

Transaction Support in Spring

- Declarative transaction
- Programmatic transaction



Declarative Transaction

Declarative Transaction

- You can declaratively specify that a method on a bean has transactional properties
  - Spring handles the transactional behavior
- Built upon AOP
  - For intercepting calls to methods for performing transaction
- No need to modify the code
  - The code does not contain any transaction management code
  - Changing transactional properties is just changing the configuration file

Declarative Transaction

- A group of methods can be specified with a same transactional properties
  - wildcard
- Additional interceptors can be plugged in

Configuration of Declarative Transaction

```
<bean id="clinicTarget"
 class="org.springframework.samples.petclinic.hibernate.HibernateClinic">
    <property name="sessionFactory" ref="sessionFactory"/>
</bean>

<bean id="clinic"
 class="org.springframework.transaction.interceptor.TransactionProxyFact
 oryBean">
    <property name="transactionManager" ref="transactionManager"/>
    <property name="target" ref="clinicTarget"/>
    <property name="transactionAttributes">
       <props>
          <prop key="get*">PROPAGATION_REQUIRED,readOnly</prop>
          <prop key="find*">PROPAGATION_REQUIRED,readOnly</prop>
          <prop key="load*">PROPAGATION_REQUIRED,readOnly</prop>
          <prop key="store*">PROPAGATION_REQUIRED</prop>
       </props>
    </property>
 </bean>
```

Configuration of Declarative Transaction

```
  <!-- Transaction manager for a single Hibernate SessionFactory
       (alternative to JTA) -->
<bean id="transactionManager"
   class="org.springframework.orm.hibernate3.HibernateTransactionManager">
   <property name="sessionFactory" ref="sessionFactory"/>
</bean>

<!-- Transaction manager that delegates to JTA (for a transactional JNDI
 DataSource) -->
<!--
<bean id="transactionManager"
         class="org.springframework.transaction.jta.JtaTransactionManager"/>
-->
```

Business Logic Class

```
public class HibernateClinic extends HibernateDaoSupport implements Clinic {

  public Collection getVets() throws DataAccessException {
     return getHibernateTemplate().find("from Vet vet order by vet.lastName,
     vet.firstName");
  }

  public Collection getPetTypes() throws DataAccessException {
     return getHibernateTemplate().find("from PetType type order by type.name");
  }

  public Collection findOwners(String lastName) throws DataAccessException {
     return getHibernateTemplate().find("from Owner owner where owner.lastName
     like ?", lastName + "%");
  }
```

Thank You!