Last Edited : Jul 16 Notebook

Report Abuse

```
Spring boot: @Transactional (Part-2)
                                     Hierarchy of Transaction Managers
                                                    <<interface>>
                                                 TransactionManager
                                                    extends
                                                    <<interface>>
                          3 Methods:

    getTransaction

                                            PlatformTransactionManager

    commit

    rollback

                                                       <<Abstract Class>>
       Provides default implementation
                                           Abstract Platform Transaction Manager\\
       of below methods:

    getTransaction

    commit

    rollback

                                                                             <<Concrete Class>>
                                                                                                          <<Concrete Class>>
                                              <<Concrete Class>>
            <<Concrete Class>>
                                            Hibernate
                                                                            JPA
          DataSource
           TransactionManager
                                            TransactionManager
                                                                            TransactionManager
                                                                                                          TransactionManager
                                                                                                           (Java Transaction API)
                                                                             (Java Persistence API)
           <<Concrete Class>>
          JDBC
           TransactionManager
        (Java Data Base Connectivity)
                                   These all manages LOCAL Transaction
                                                                                                       JTA manage Distributed Transactions
                                                       Transaction Management
                                                                                                                                Programmatic
          Transaction Management through
                                                                                                                   - Transaction Management through code
          Annotation
                                                                                                                   - Flexible but difficult to maintain.
 public class User {
                                                                                                                      why Flexible? Lets consider the below code:
   public void updateUser(){
                                                                                                                             Component
      System.out.println("UPDATE QUERY TO update the user db values");
                                                                                                                            public class User {
                                                                                                                               @Transactional
                                                                                                                               public void updateUser(){
  Here, based on underling DataSource used like JDBC or JPA etc.
                                                                                                                                 //1. update DB
  Spring boot will choose appropriate Transaction manager.
                                                                                                                                 //2. External API call
                                                                                                                                  //3. update DB
                                            Declarative
                       Transaction Management through
                       Annotation
    @Component
    public class User {
          @Transactional
          public void updateUser(){
                System.out.println("UPDATE QUERY TO update the user db values");
      Here, based on underling DataSource used like JDBC or JPA etc.
      Spring boot will choose appropriate Transaction manager.
  @Configuration
  public class AppConfig {
        @Bean
        public DataSource dataSource() {
              DriverManagerDataSource dataSource = new DriverManagerDataSource();
               dataSource.setDriverClassName("org.h2.Driver");
              dataSource.setUrl("jdbc:h2:mem:testdb");
              dataSource.setUsername("sa");
              dataSource.setPassword("");
              return dataSource;
        @Bean
        public PlatformTransactionManager userTransactionManager(DataSource dataSource) {
              return new DataSourceTransactionManager(dataSource);
  @Component
  public class UserDeclarative {
          @Transactional(transactionManager = "userTransactionManager")
          public void updateUserProgrammatic() {
                 //SOME DB OPERATIONS
                 System.out.println("Insert Query ran");
                 System.out.println("Update Query ran");
                                  2 ways to implement by Programmatic
                    Approach 1
                                                                                              Usage of TransactionTemplate
       public class AppConfig {
                                                                                 ublic class AppConfig {
        public DataSource dataSource() {
           OriverManagerDataSource dataSource = new DriverManagerOataSource();
           dataSource.setOriverClassName("org.h2.Oriver");
                                                                                   public DataSource dataSource() {
           dataSource.setUrl("jdbc:h2:mem:testdb");
                                                                                     DriverManagerOataSource dataSource = new DriverManagerOataSource();
           dataSource.setUsername("sa");
                                                                                     dataSource.setDriverClassName("org.h2.Driver");
           dataSource.setPassword("");
                                                                                     dataSource.setUrl("jdbc:h2:mem:testdb");
                                                                                     detaSource.setUsername("sa");
                                                                                     return dataSource;
         public PlatformTransactionManager userTransactionManager(DataSource dataSource) {
           return new DataSourceTransactionManager(dataSource);
                                                                                  public PlatformTransactionManager userTransactionManager(DataSource dataSource) {
                                                                                     return new DataSourceTransactionManager(dataSource);
public class UserProgrammaticApproach1 {
                                                                                  public TransactionTemplate transactionTemplate(PlatformTransactionManager userTransactionManager) {
                                                                                     return new TransactionTemplate(userTransactionManager);
  PlatformTransactionManager userTransactionManager;
   UserProgrammaticApproach1(PlatformTransactionManager userTransactionManager) {
     this.userTransactionManager = userTransactionManager;
                                                                              public class UserProgrammaticApproach2 {
                                                                                 TransactionTemplate transactionTemplate;
   public void updateUserProgrammatic(){
     TransactionStatus status = userTransactionManager.getTransaction( definition: null);
                                                                                 public UserProgrammaticApproach2(TransactionTemplate transactionTemplate){
                                                                                    this.transactionTemplate = transactionTemplate;
        //SOME INITIAL SET OF DB OPERATIONS
        System.out.println("Insert Query run1");
        System.out.println("Update Query run1");
                                                                                 public void updateUserProgrammatic(){
        userTransactionManager.commit(status);
                                                                                    TransactionCallback<TransactionStatus> dbOperationsTask = (TransactionStatus status) -> {
     cotch (Exception c) (
                                                                                       Syctom.out.println("Incort Query ran");
        userTransactionManager.rollback(status);
                                                                                       System.out.println("Update Query ran");
                                                                                       return status;
                                                                                    TransactionStatus status = transactionTemplate.execute(dbOperationsTask);
                                                            Now, lets see Propagation
      When we try to create a new Transaction, it first check the PROPAGATION value set, and this tell whether we
      have to create new transaction or not.
■ REQUIRED (default propagation):
   @Transactional(propagation=Propagation.REQUIRED)
        if(parent txn present)
               Use it;
        else
               Create new transaction;
REQUIRED_NEW:
   @Transactional(propagation=Propagation.REQUIRED_NEW)
        if(parent txn present)
               Suspend the parent txn;
               Create a new Txn and once finished;
               Resume the parent txn;
        else
               Create new transaction and execute the method;
SUPPORTS:
   @Transactional(propagation=Propagation.SUPPORTS)
         if(parent txn present)
               Use it;
         Else
               Execute the method without any transaction;
      ■ NOT_SUPPORTED:
            @Transactional(propagation=Propagation.NOT_SUPPORTED)
                      if(parent txn present)
                                 Suspend the parent txn;
                                 Execute the method without any transaction;
                                 Resume the parent txn;
                       else
                                 Execute the method without any transaction;
       MANDATORY:
           @Transactional(propagation=Propagation.MANDATORY)
                      if(parent txn present)
                                  Use it;
                       Else
                                 Throw exception;
       ■ NEVER:
           @Transactional(propagation=Propagation.MANDATORY)
                      if(parent txn present)
                                 Throw exception;
                       Else
                                 Execute the method without any transaction;
                                         Declarative way of usage:
                                                                         public class UserDAO {
oublic class UserDeclarative {
                                                                           @Transactional(propagation = Propagation.REQUIRED)
 UserDAO userDAOobj;
                                                                            * if(parent txn present)
 @Transectional
 public void updateUser() {
                                                                                  use it
   System.out.println("Is transaction active: * + TransactionSynchronizationManager.isActuelTransactionActive());
                                                                            * else
   System.out.grintln("Current transaction name: " + TransactionSynchronizationFanager.getCurrentTransactionName());
                                                                                 create new
   System.out.println("Some initial DB operation");
   userOADobj.dbOperationWithRequiredPropagation();
   System.out.println("Some final DB operation");
                                                                           public void dbOperationWithRequiredPropagation() {
 public void updateUserFromWonTransactionalMethod() {
                                                                              //EXECUTE DB QUERIES
   System.out.println("Is transaction active: " + TransactionSynchronizationManager.isActualTransactionActive());
                                                                              boolean isTransactionActive = TransactionSynchronizationManager.isActualTransactionActive();
   System.out.println("Current transaction name: " * TransactionSynchronizationHanager.getCurrentTransactionName());
   System.out.println("Some initial DB operation");
                                                                              String currentTransactionName = TransactionSynchronizationManager.getCurrentTransactionName();
   userGADobj.dbOperationWithRequiredPropagation();
                                                                              System.out.println("Some final OB operation");
                                                                              System.out.println("Propagation.REQUIRED: Is transaction active: " + isTransactionActive);
                                                                              System.out.println("Propagation.REQUIRED: Current transaction name: " + currentTransactionName);
                                                                              System.out.println("*********************************);
                                                           Output:
      Current transaction name: com.conceptandcoding.learningspringboot.TransactionManagement.UserDeclarative.updateUser
      Some initial DB operation
      *********
      Propagation.REQUIRED: Is parent transaction active: true
      Propagation.REQUIRED: Current transaction name: com.conceptandcoding.learningspringboot.TransactionManagement.UserDeclarative.updateUser
      *************
      Some final DB operation
      Is transaction active: false
      Current transaction name: null
      Some initial DB operation
      **********
      Propagation.REQUIRED: Is parent transaction active: true
      Propagation.REQUIRED: Current transaction name: com.conceptandcoding.learningspringboot.TransactionManagement.UserOAO.dbOperationWithRequiredPropagation
      ****************
      Some final DB operation
                                                                     Programmatic way of usage:
                                                                             (Approach 1)
 @Component
 public class UserDeclarative {
     @Autowired
     UserDAO userDAOobj;
     @Transactional
     public void updateUser() {
        System.out.println("Is transaction active: " + TransactionSynchronizationManager.isActualTransactionActive());
         System.out.println("Current transaction name: " + TransactionSynchronizationManager.getCurrentTransactionName());
         System.out.println("Some initial DB operation");
        userDAOobj.dbOperationWithRequiredPropagationUsingProgrammaticApproach1();
         System.out.println("Some final DB operation");
     public void updateUserFromNonTransactionalMethod() {
         System.out.println("Is transaction active: " + TransactionSynchronizationManager.isActualTransactionActive());
         System.out.println("Current transaction name: " + TransactionSynchronizationManager.getCurrentTransactionName());
         System.out.println("Some initial DB operation");
         userDAOobj.dbOperationWithRequiredPropagationUsingProgrammaticApproach1();
         System.out.println("Some final DB operation");
@Component
public class UserDAO {
   PlatformTransactionManager userTransactionManager;
   UserDAO(PlatformTransactionManager userTransactionManager) {
       this.userTransactionManager = userTransactionManager;
    * if(parent txn present)
             use it
    * else
   public void dbOperationWithRequiredPropagationUsingProgrammaticApproach1(){
       DefaultTransactionDefinition transactionDefinition = new DefaultTransactionDefinition();
       transactionDefinition.setName("Testing REQUIRED propagation");
       transactionDefinition.setPropagationBehavior(TransactionDefinition.PROPAGATION_REQUIRED);
       TransactionStatus status = userTransactionManager.getTransaction(transactionDefinition);
           //EXECUTE operation
           System.out.println("Propagation.REQUIRED: Is transaction active: " + TransactionSynchronizationManager.isActualTransactionActive());
           System.out.println("Propagation.REQUIRED: Current transaction name: " + TransactionSynchronizationManager.getCurrentTransactionName());
           userTransactionManager.commit(status);
       catch (Exception e) {
           userTransactionManager.rollback(status);
                               (Approach 2): using TransactionTemplate
                                                                  public class UserService {
public class AppConfig {
                                                                    @Autowired
 public DataSource dataSource() {
                                                                    UserDAO userDAOobj;
    DriverManagerDataSource dataSource = new DriverManagerDataSource();
    dataSource.setDriverClassName("org.h2.Driver");
                                                                     @Transactional
    dataSource.setUrl("jdbc:h2:mem:testdb");
                                                                    public void updateUser() {
    dataSource.setUsername("sa2");
    dataSource.setPassword("");
                                                                       System.out.println("Is transaction active: " + TransactionSynchronizationManager.isActualTransactionActive());
    return dataSource;
                                                                       System.out.println("Current transaction name: " + TransactionSynchronizationManager.getCurrentTransactionName());
                                                                       System.out.println("Some initial DB operation");
                                                                       userDAOobj.dbOperationWithRequiredPropagationUsingProgrammaticApproach2();
  public PlatformTransactionManager userTransactionManager(DataSource dataSource) {
    return new DataSourceTransactionHanager(dataSource);
                                                                       System.out.println("Some final DB operation");
                                                                     public void updateUserFromNonTransactionalMethod() {
  public TransactionTemplate transactionTemplate(PlatformTransactionManager userTransactionManager) {
                                                                       System.out.println("Is transaction active: " + TransactionSynchronizationManager.isActualTransactionActive());
    TransactionTemplate transactionTemplate = new TransactionTemplate(userTransactionManager);
    transactionTemplate.setPropagationBehavior(TransactionDefinition.PROPASATION_REQUIRED);
                                                                       System.out.println("Current transaction name: " + TransactionSynchronizationManager.getCurrentTransactionName());
    transactionTemplate.setName("TRANSACTION TEMPLATE REQUIRED PROPAGATION");
                                                                       System.out.println("Some initial DB operation");
    return transactionTemplate;
                                                                       userOAOobj.dbOperationWithRequiredPropagationUsingProgrammaticApproach2();
                                                                       System.out.println("Some final DB operation");
           public class UserDAD {
             TransactionTemplate transactionTemplate;
             UserDAD(TransactionTemplate transactionTemplate) {
               this.transactionTemplate = transactionTemplate;
             * if(parent txn present)
             * use it
              * else
             public void dbOperationWithRequiredPropagationUsingProgrammaticApproach2() {
                TransactionCallback<TransactionStatus> operations = (TransactionStatus status) -> {
                  System.out.println("Propagation.REQUIRED: Is transaction active: " + TransactionSynchronizationManager.isActualTransactionActive());
                  System.out.println("Propagation.REQUIRED: Current transaction name: " + TransactionSynchronizationManager.getCurrentTransactionName());
                  return status;
                TransactionStatus status = transactionTemplate.execute(operations);
                                                                            Output:
 Is transaction active: true
 Current transaction name: com.conceptandcoding.learningspringboot.TransactionManagement.UserDeclarative.updateUser
 Some initial DB operation
 **********
 Propagation.REQUIRED: Is transaction active: true
 Propagation.REQUIRED: Current transaction name: com.conceptandcoding.learningspringboot.TransactionManagement.UserDeclarative.updateUser
 ********
 Some final DB operation
 Is transaction active: false
 Current transaction name: null
 Some initial DB operation
 ********
 Propagation.REQUIRED: Is transaction active: true
```

Propagation.REQUIRED: Current transaction name: TRANSACTION TEMPLATE REQUIRED PROPAGATION

\*\*\*\*\*\*\*\*

Some final DB operation