

* 学习目标

* 能够掌握Spring 集成MyBatis

- * dao对象放到IOC容器

- * 自学MyBatisPlus

* 能够掌握Spring 事务管理

- * 会用事务：XML，注解（那个方法，添加事务管理）

- * 事务传播行为

* 回顾

- * Spring IOC，AOP

* 能够掌握Spring 集成MyBatis

- * Spring集成MyBatis：主要把要配置MyBatis的Dao对象注入到SpringIOC中

```
1 * 前期环境准备
2 * 添加依赖
3     <dependency>
4         <groupId>junit</groupId>
5         <artifactId>junit</artifactId>
6         <version>4.12</version>
7         <scope>test</scope>
8     </dependency>
9     <dependency>
10         <groupId>org.projectlombok</groupId>
11         <artifactId>lombok</artifactId>
12         <version>1.18.10</version>
13         <scope>provided</scope>
14     </dependency>
15     <dependency>
16         <groupId>org.springframework</groupId>
17         <artifactId>spring-context</artifactId>
18         <version>5.2.2.RELEASE</version>
```

```
19     </dependency>
20     <dependency>
21         <groupId>org.springframework</groupId>
22         <artifactId>spring-test</artifactId>
23         <version>5.2.2.RELEASE</version>
24         <scope>test</scope>
25     </dependency>
26     <dependency>
27         <groupId>org.springframework</groupId>
28         <artifactId>spring-aspects</artifactId>
29         <version>5.2.2.RELEASE</version>
30     </dependency>
31     <dependency>
32         <groupId>org.aspectj</groupId>
33         <artifactId>aspectjrt</artifactId>
34         <version>1.9.5</version>
35     </dependency>
36     <dependency>
37         <groupId>aopalliance</groupId>
38         <artifactId>aopalliance</artifactId>
39         <version>1.0</version>
40     </dependency>
41     <dependency>
42         <groupId>mysql</groupId>
43         <artifactId>mysql-connector-java</artifactId>
44         <version>5.1.16</version>
45     </dependency>
46     <dependency>
47         <groupId>org.mybatis</groupId>
48         <artifactId>mybatis</artifactId>
49         <version>3.3.0</version>
50     </dependency>
51     <dependency>
52         <groupId>org.springframework</groupId>
53         <artifactId>spring-tx</artifactId>
54         <version>5.2.2.RELEASE</version>
55     </dependency>
56     <!--slf4j-->
57     <dependency>
58         <groupId>org.slf4j</groupId>
```

```

59         <artifactId>slf4j-api</artifactId>
60         <version>1.7.25</version>
61     </dependency>
62     <!--log4j-->
63     <dependency>
64         <groupId>log4j</groupId>
65         <artifactId>log4j</artifactId>
66         <version>1.2.17</version>
67     </dependency>
68     <!--slf4j到log4j-->
69     <dependency>
70         <groupId>org.slf4j</groupId>
71         <artifactId>slf4j-log4j12</artifactId>
72         <version>1.7.25</version>
73     </dependency>
74     <!--Spring 和 MyBatis链接的jar-->
75     <dependency>
76         <groupId>org.mybatis</groupId>
77         <artifactId>mybatis-spring</artifactId>
78         <version>1.3.2</version>
79     </dependency>
80     <dependency>
81         <groupId>com.alibaba</groupId>
82         <artifactId>druid</artifactId>
83         <version>1.1.21</version>
84     </dependency>
85 * copy之前mybatis的代码，单元测试通过
86
87 * 案例一（XML配置）
88 * spring-mybatis
89 <?xml version="1.0" encoding="UTF-8"?>
90 <beans xmlns="http://www.springframework.org/schema/beans"
91     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
92     xmlns:context="http://www.springframework.org/schema/context"
93     xmlns:aop="http://www.springframework.org/schema/aop"
94     xsi:schemaLocation="http://www.springframework.org/schema/beans
95     http://www.springframework.org/schema/beans/spring-beans.xsd
96     http://www.springframework.org/schema/context
97     http://www.springframework.org/schema/context/spring-context.xsd
98     http://www.springframework.org/schema/aop

```

```

99     http://www.springframework.org/schema/aop/spring-aop.xsd">
100 <context:property-placeholder location="db.properties"/>
101 <bean id="druid" class="com.alibaba.druid.pool.DruidDataSource">
102     <property name="driverClassName" value="${lg.driver}"/>
103     <property name="url" value="${lg.url}"/>
104     <property name="username" value="${lg.username}"/>
105     <property name="password" value="${lg.password}"/>
106 </bean>
107 <!--构建SqlSessionFactory:由于这里是与Spring结合的选择SqlSessionFactoryBean (
108     <bean id="ssf" class="org.mybatis.spring.SqlSessionFactoryBean">
109         <property name="dataSource" ref="druid"/>
110         <property name="mapperLocations" value="classpath:com/lg/dao/*.xml"/>
111         <property name="typeAliasesPackage" value="com.lg.bean"/>
112     </bean>
113     <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
114         <property name="basePackage" value="com.lg.dao"/>
115         <property name="sqlSessionFactoryBeanName" value="ssf"/>
116     </bean>
117 </beans>
118 * applicationContext
119 <?xml version="1.0" encoding="UTF-8"?>
120 <beans xmlns="http://www.springframework.org/schema/beans"
121     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
122     xmlns:context="http://www.springframework.org/schema/context"
123     xmlns:aop="http://www.springframework.org/schema/aop"
124     xsi:schemaLocation="http://www.springframework.org/schema/beans
125     http://www.springframework.org/schema/beans/spring-beans.xsd
126     http://www.springframework.org/schema/context
127     http://www.springframework.org/schema/context/spring-context.xsd
128     http://www.springframework.org/schema/aop
129     http://www.springframework.org/schema/aop/spring-aop.xsd">
130     <import resource="spring-mybatis.xml"/>
131     <context:component-scan base-package="com.lg"/>
132 </beans>
133 * Service
134 public interface CompanyService {
135     /**
136     * @param code
137     * @return
138     * 通过公司编号获取Company

```

```
139      */
140      Company getCompany(int code);
141  }
142  @Service
143  public class CompanyServiceImpl implements CompanyService {
144      @Autowired
145      private CompanyDao companyDao;
146      @Override
147      public Company getCompany(int code) {
148          return companyDao.getCompany(code);
149      }
150  }
151  * 单元测试
152  @RunWith(SpringJUnit4ClassRunner.class)
153  @ContextConfiguration("classpath:applicationContext.xml")
154  public class AppTest
155  {
156      @Autowired
157      private CompanyService companyService;
158      @Test
159      public void test1(){
160          Company company = companyService.getCompany(1001);
161          System.out.println(company);
162      }
163  }
164
165  * 零XML配置
166  @Configuration
167  @ComponentScan("com.lg")
168  @EnableAspectJAutoProxy
169  @Import(SMConfiguration.class)
170  public class SpringConfiguration {
171  }
172
173  @Configuration
174  @MapperScan("com.lg.dao")
175  @PropertySource("classpath:db.properties")
176  public class SMConfiguration {
177      @Value("${lg.driver}")
178      private String driver;
```

```

179     @Value("${lg.url}")
180     private String url;
181     @Value("${lg.username}")
182     private String username;
183     @Value("${lg.password}")
184     private String password;
185     @Bean
186     @Scope(ConfigurableBeanFactory.SCOPE_SINGLETON)
187     public DataSource dataSource(){
188         DruidDataSource ds=new DruidDataSource();
189         ds.setDriverClassName(driver);
190         ds.setUrl(url);
191         ds.setUsername(username);
192         ds.setPassword(password);
193         return ds;
194     }
195     @Bean
196     public SqlSessionFactoryBean sqlSessionFactoryBean(DataSource dataSource){
197         SqlSessionFactoryBean sqlSessionFactoryBean=new SqlSessionFactoryBean();
198         sqlSessionFactoryBean.setDataSource(dataSource);
199         PathMatchingResourcePatternResolver resolver=new PathMatchingResourcePa
200         try {
201             Resource[] resources = resolver.getResources("classpath:com/lg/dao/
202             sqlSessionFactoryBean.setMapperLocations(resources);
203         } catch (IOException e) {
204             e.printStackTrace();
205         }
206         sqlSessionFactoryBean.setTypeAliasesPackage("com.lg.bean");
207         return sqlSessionFactoryBean;
208     }
209     // @MapperScan("com.lg.dao"): 可以使用MapperScann注解代替
210     /*
211     @Bean
212     public MapperScannerConfigurer mapperScannerConfigurer(){
213         MapperScannerConfigurer mapperScannerConfigurer=new MapperScannerConfig
214         mapperScannerConfigurer.setBasePackage("com.lg.dao");
215         mapperScannerConfigurer.setSqlSessionFactoryBeanName("sqlSessionFactory
216         return mapperScannerConfigurer;
217     }
218     */

```

```
219 }
220
221
222 * 单元测试
223 @RunWith(SpringJUnit4ClassRunner.class)
224 @SpringJUnitConfig(SpringConfiguration.class)
225 public class AppTest2
226 {
227     @Autowired
228     private CompanyService companyService;
229     @Test
230     public void test1(){
231         Company company = companyService.getCompany(1001);
232         System.out.println(company);
233     }
234 }
235
```

* 能够掌握Spring 事务管理

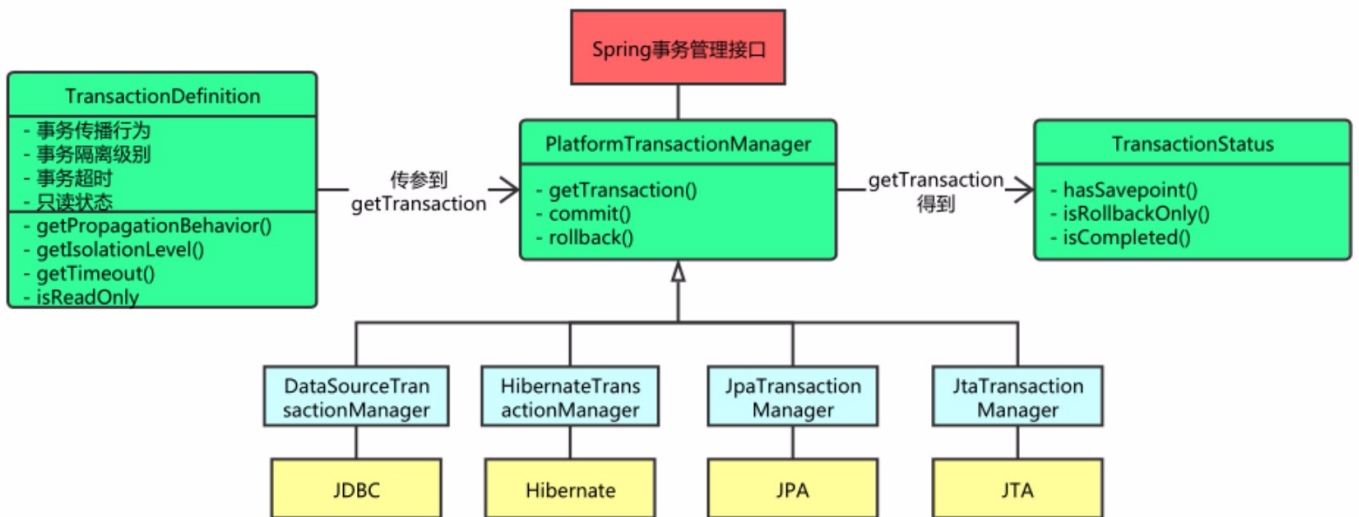
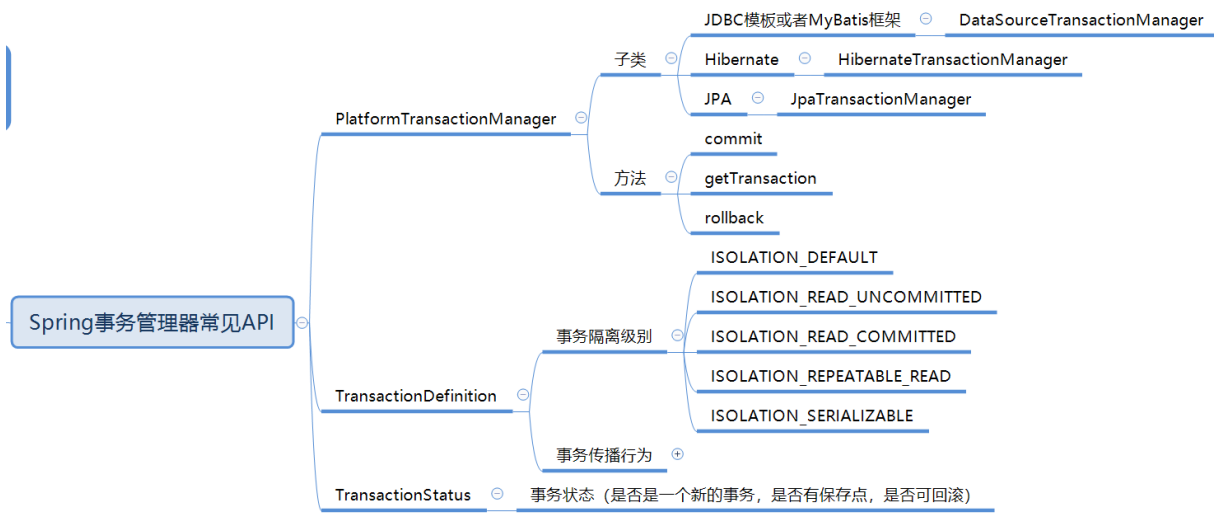
* 事务的回顾：[08-Oracle高级新1](#)

* Spring事务管理概述

概述

Spring并不直接管理事务，而是提供了多种事务管理器，他们将事务管理的职责委托给Hibernate或者MyBatis等持久化机制所提供的相关平台框架的事务来实现。

* Spring事务管理常见的API



* 事务传播特性



- 案例一（员工转部门）：
- * 给部门添加字段，部门人数dnum
- * sql语句（要么一起成功，要么一起失败）

```

4 UPDATE employee SET job='Java开发工程师',departid=1 WHERE empno='LG008';
5 UPDATE department SET dnum=dnum-1 WHERE deptno=1;
6 UPDATE department SET dnum=dnum+1 WHERE deptno=2;
7 * dao
8 public interface EmployeeDao {
9     @Update("UPDATE employee SET job=#{job},departid=#{ndid} WHERE empno=#{empno}")
10    void updateEmployee(Map<String,Object> params);
11 }
12 public interface DepartmentDao {
13     @Update("UPDATE department SET dnum=dnum+1 WHERE deptno=#{ndid}")
14    void updateDepartmentByIncrease(Map<String,Object> params);
15     @Update("UPDATE department SET dnum=dnum-1 WHERE deptno=#{odid}")
16    void updateDepartmentByDecrease(Map<String,Object> params);
17 }
18 * service
19 public interface TransferEmployeeService {
20    void transferEmployee(Map<String,Object> params);
21 }
22 @Service
23 public class TransferEmployeeServiceImpl implements TransferEmployeeService {
24     @Autowired
25     private EmployeeDao employeeDao;
26     @Autowired
27     private DepartmentDao departmentDao;
28
29     @Transactional
30     @Override
31     public void transferEmployee(Map<String, Object> params) {
32         employeeDao.updateEmployee(params);
33         // 待转部门加1
34         departmentDao.updateDepartmentByIncrease(params);
35         //     int i=100/0;
36         // 原部门减1
37         departmentDao.updateDepartmentByDecrease(params);
38     }
39 }
40 * 单元测试
41 @RunWith(SpringJUnit4ClassRunner.class)
42 @ContextConfiguration("classpath:applicationContext.xml")
43 public class AppTest

```

```

44 {
45     @Autowired
46     private CompanyService companyService;
47     @Test
48     public void test1(){
49         Company company = companyService.getCompany(1001);
50         System.out.println(company);
51     }
52 }
53
54
55 * 配置 (xml)
56 * spring-mybatis.xml
57 <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
58     <property name="dataSource" ref="druid"/>
59 </bean>
60 <tx:advice id="txAspect" transaction-manager="transactionManager">
61     <tx:attributes>
62         <tx:method name="transferEmployee" propagation="REQUIRED"/>
63     </tx:attributes>
64 </tx:advice>
65 <aop:config>
66     <aop:advisor advice-ref="txAspect" pointcut="execution(* *.transferEmployee)*">
67 </aop:config>
68 * 单元测试
69 @RunWith(SpringJUnit4ClassRunner.class)
70 @ContextConfiguration("classpath:applicationContext.xml")
71 public class AppTest
72 {
73     @Autowired
74     private TransferEmployeeService transferEmployeeService;
75     @Test
76     public void test2(){
77         Map<String, Object> params = new HashMap<>();
78         params.put("empno", "LG008");
79         params.put("job", "html讲师456");
80         params.put("odid", 2); // -1
81         params.put("ndid", 1); // +1
82         transferEmployeeService.transferEmployee(params);
83     }

```

```

84 }
85 * 配置（xml+注解）
86 * 在xml中
87 <bean id="transactionManager" class="org.springframework.jdbc.datasource.Data
88     <property name="dataSource" ref="druid"/>
89 </bean>
90 <tx:annotation-driven transaction-manager="transactionManager"/>
91 * 在方法中
92 @Transactional
93 @Override
94 public void transferEmployee(Map<String, Object> params) {
95     employeeDao.updateEmployee(params);
96     // 待转部门加1
97     departmentDao.updateDepartmentByIncrease(params);
98 //     int i=100/0;
99     // 原部门减1
100     departmentDao.updateDepartmentByDecrease(params);
101 }
102 * 继续通过加装xml形式测试
103 * @ContextConfiguration("classpath:applicationContext.xml")
104
105 * 配置（注解）
106 * 在SMConfiguration加上注解
107 * @EnableTransactionManagement
108 * 加上获取事务管理器的方法
109 @Bean
110 public TransactionManager transactionManager(DataSource dataSource){
111     DataSourceTransactionManager transactionManager=new DataSourceTransaction
112     transactionManager.setDataSource(dataSource);
113     return transactionManager;
114 }
115 * 单元测试
116 @RunWith(SpringJUnit4ClassRunner.class)
117 @SpringJUnitConfig(SpringConfiguration.class)
118 public class AppTest2
119 {
120     @Autowired
121     private TransferEmployeeService transferEmployeeService;
122     @Test
123     public void test2(){

```

```
124     Map<String,Object> params=new HashMap<>();
125     params.put("empno","LG008");
126     params.put("job","html工程师1000");
127     params.put("odid",1);// -1
128     params.put("ndid",2);// +1
129     transferEmployeeService.transferEmployee(params);
130 }
131 }
132
133 * 案例二（事务传播属性）
134 * 前期准备
135 @Service
136 public class PropagationServiceImpl {
137     @Autowired
138     private EmployeeDao employeeDao;
139     @Autowired
140     private DepartmentDao departmentDao;
141     @Transactional(propagation = Propagation.REQUIRED)
142     public void serviceB(Map<String, Object> params){
143         employeeDao.updateEmployee(params);
144         // 待转部门加1
145         departmentDao.updateDepartmentByIncrease(params);
146         int i=100/0;
147         // 原部门减1
148         departmentDao.updateDepartmentByDecrease(params);
149     }
150
151     @Transactional(propagation = Propagation.REQUIRED)
152     public void service(){
153         Map<String,Object> paramsB=new HashMap<>();
154         paramsB.put("empno","LG003");
155         paramsB.put("job","html工程师88");
156         paramsB.put("odid",2);// -1
157         paramsB.put("ndid",3);// +1
158         serviceB(paramsB);
159     }
160
161 }
162
163 * 测试：PROPAGATION_REQUIRED
```

```
164 * 支持当前事务；如果不存在，则创建一个新事务
165
166 * 测试：PROPAGATION_SUPPORTS
167 * 支持当前事务；如果当前没有事务，就以非事务方式执行
168 * 修改属性：@Transactional(propagation = Propagation.SUPPORTS)
169
170 * 测试：PROPAGATION_MANDATORY
171 * 支持当前事务；如果没有当前事务，则引发异常
172 * 修改属性：@Transactional(propagation = Propagation.MANDATORY)
173
174 * 单元测试代码
175 * 当前没有事务，调用ServiceB
176     @Autowired
177     private PropagationServiceImpl propagationService;
178     @Test
179     public void test3(){
180         Map<String,Object> paramsB=new HashMap<>();
181         paramsB.put("empno","LG003");
182         paramsB.put("job","html工程师88");
183         paramsB.put("odid",2);// -1
184         paramsB.put("ndid",3);// +1
185         propagationService.serviceB(paramsB);
186     }
187 * 当前有事务，调用ServiceB
188     @Test
189     public void test3(){
190         propagationService.service();
191     }
```