* 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

```
* 前期环境准备
* 添加依赖
   <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
    </dependency>
     <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.10</version>
            <scope>provided</scope>
    </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.2.2.RELEASE</version>
```

```
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>5.2.2.RELEASE</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aspects</artifactId>
            <version>5.2.2.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjrt</artifactId>
            <version>1.9.5</version>
        </dependency>
        <dependency>
            <groupId>aopalliance</groupId>
            <artifactId>aopalliance</artifactId>
            <version>1.0</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.16</version>
        </dependency>
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.3.0</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>5.2.2.RELEASE</version>
        </dependency>
        <!--slf4j-->
        <dependency>
            <groupId>org.slf4j</groupId>
```

```
            <artifactId>slf4j-api</artifactId>
            <version>1.7.25</version>
        </dependency>
        <!--log4j-->
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
        </dependency>
        <!--slf4j到log4j-->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>1.7.25</version>
        </dependency>
        <!--Spring 和 MyBatis链接的jar-->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
            <version>1.3.2</version>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
            <version>1.1.21</version>
        </dependency>
* copy之前mybatis的代码，单元测试通过

* 案例一（XML配置）
 * spring-mybatis
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/aop
```

```
       http://www.springframework.org/schema/aop/spring-aop.xsd">
      <context:property-placeholder location="db.properties"/>
      <bean id="druid" class="com.alibaba.druid.pool.DruidDataSource">
          <property name="driverClassName" value="${lg.driver}"/>
          <property name="url" value="${lg.url}"/>
          <property name="username" value="${lg.username}"/>
          <property name="password" value="${lg.password}"/>
      </bean>
    <!--构建SqlSessionFactory:由于这里是与Spring结合的选择SqlSessionFactoryBean（
     <bean id="ssf" class="org.mybatis.spring.SqlSessionFactoryBean">
         <property name="dataSource" ref="druid"/>
         <property name="mapperLocations" value="classpath:com/lg/dao/*.xml"/>
         <property name="typeAliasesPackage" value="com.lg.bean"/>
     </bean>
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.lg.dao"/>
        <property name="sqlSessionFactoryBeanName" value="ssf"/>
    </bean>
</beans>
 * applicationContext
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/aop
       http://www.springframework.org/schema/aop/spring-aop.xsd">
    <import resource="spring-mybatis.xml"/>
    <context:component-scan base-package="com.lg"/>
</beans>
* Service
public interface CompanyService {
    /**
     * @param code
     * @return
     * 通过公司编号获取Company
```

```
     */
    Company getCompany(int code);
}
@Service
public class CompanyServiceImpl implements CompanyService {
    @Autowired
    private CompanyDao companyDao;
    @Override
    public Company getCompany(int code) {
        return companyDao.getCompany(code);
    }
}
* 单元测试
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath:applicationContext.xml")
public class AppTest
{
    @Autowired
    private CompanyService companyService;
    @Test
    public void test1(){
        Company company = companyService.getCompany(1001);
        System.out.println(company);
    }
}

* 零XML配置
@Configuration
@ComponentScan("com.lg")
@EnableAspectJAutoProxy
@Import(SMConfiguration.class)
public class SpringConfiguration {
}

@Configuration
@MapperScan("com.lg.dao")
@PropertySource("classpath:db.properties")
public class SMConfiguration {
    @Value("${lg.driver}")
    private String driver;
```

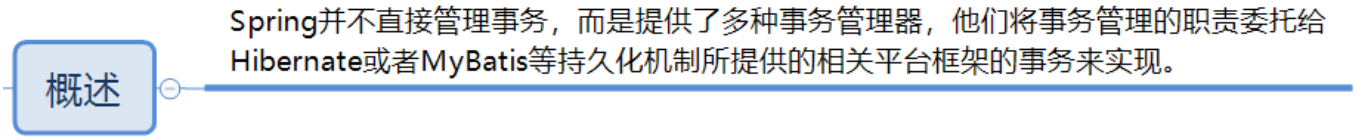
```
    @Value("${lg.url}")
    private String url;
    @Value("${lg.username}")
    private String username;
    @Value("${lg.password}")
    private String password;
    @Bean
    @Scope(ConfigurableBeanFactory.SCOPE_SINGLETON)
    public DataSource dataSource(){
        DruidDataSource ds=new DruidDataSource();
        ds.setDriverClassName(driver);
        ds.setUrl(url);
        ds.setUsername(username);
        ds.setPassword(password);
        return ds;
    }
    @Bean
    public SqlSessionFactoryBean sqlSessionFactoryBean(DataSource dataSource){
        SqlSessionFactoryBean sqlSessionFactoryBean=new SqlSessionFactoryBean()
        sqlSessionFactoryBean.setDataSource(dataSource);
        PathMatchingResourcePatternResolver resolver=new PathMatchingResourcePa
        try {
            Resource[] resources = resolver.getResources("classpath:com/lg/dao/
            sqlSessionFactoryBean.setMapperLocations(resources);
        } catch (IOException e) {
            e.printStackTrace();
        }
        sqlSessionFactoryBean.setTypeAliasesPackage("com.lg.bean");
        return sqlSessionFactoryBean;
    }
    // @MapperScan("com.lg.dao"):可以使用MapperScann注解代替
    /*
    @Bean
    public MapperScannerConfigurer mapperScannerConfigurer(){
        MapperScannerConfigurer mapperScannerConfigurer=new MapperScannerConfig
        mapperScannerConfigurer.setBasePackage("com.lg.dao");
        mapperScannerConfigurer.setSqlSessionFactoryBeanName("sqlSessionFactory
        return mapperScannerConfigurer;
    }
    */
```

```
}


* 单元测试
@RunWith(SpringJUnit4ClassRunner.class)
@SpringJUnitConfig(SpringConfiguration.class)
public class AppTest2
{
    @Autowired
    private CompanyService companyService;
    @Test
    public void test1(){
        Company company = companyService.getCompany(1001);
        System.out.println(company);
    }
}

```

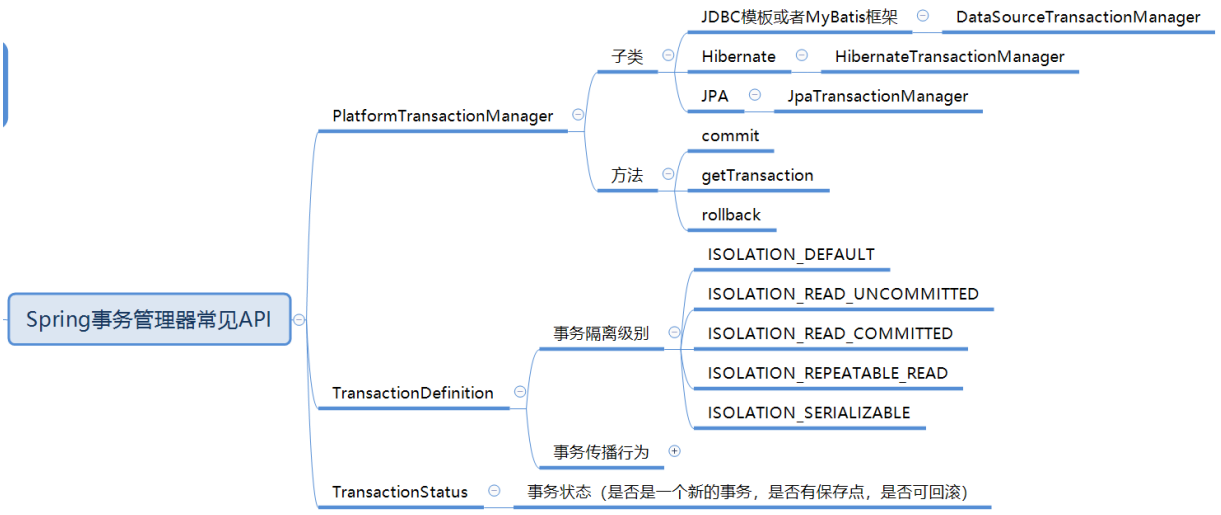
* 能够掌握Spring 事务管理

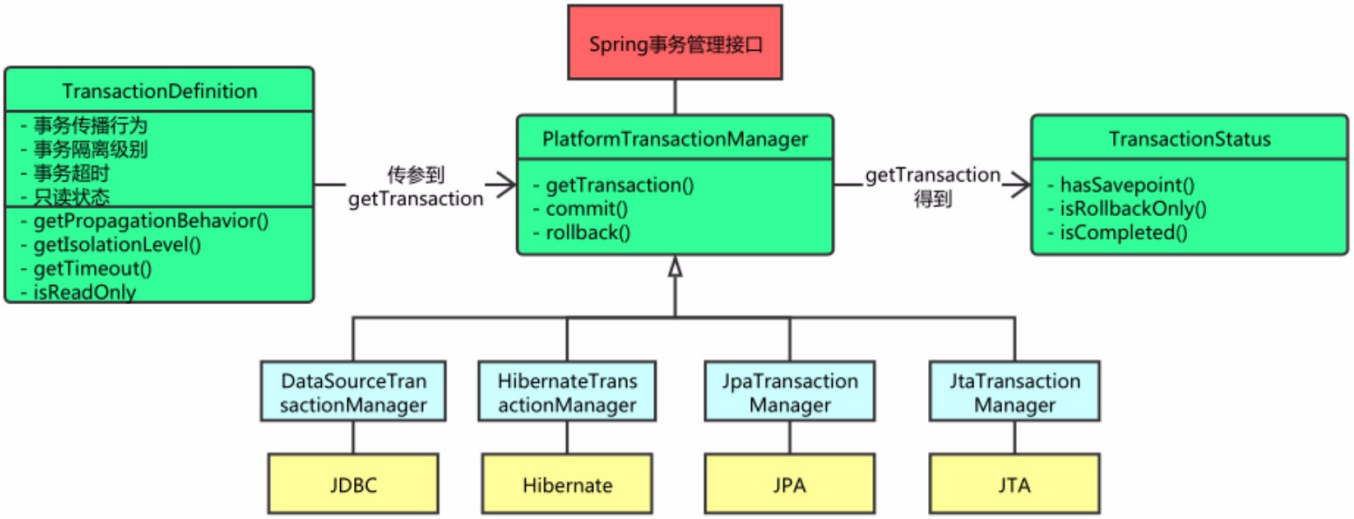
* 事务的回顾：08-Oracle高级新1

* Spring事务管理概述

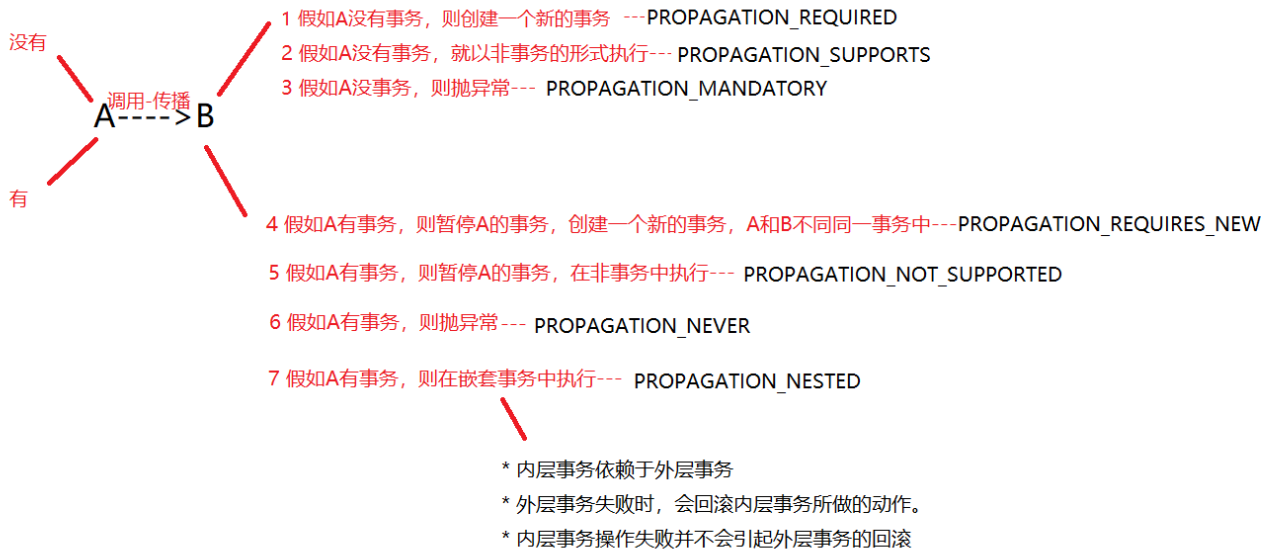


* Spring事务管理常见的API


Spring事务管理器常见API
PlatformTransactionManager
子类
JDBC模板或者MyBatis框架
DataSourceTransactionManager
Hibernate
HibernateTransactionManager
JPA
JpaTransactionManager
方法
commit
getTransaction
rollback
TransactionDefinition
事务隔离级别
ISOLATION_DEFAULT
ISOLATION_READ_UNCOMMITTED
ISOLATION_READ_COMMITTED
ISOLATION_REPEATABLE_READ
ISOLATION_SERIALIZABLE
事务传播行为
TransactionStatus
事务状态（是否是一个新的事务，是否有保存点，是否可回滚）


Spring事务管理接口
TransactionDefinition
- 事务传播行为
- 事务隔离级别
- 事务超时
- 只读状态
- getPropagationBehavior()
- getIsolationLevel()
- getTimeout()
- isReadOnly
传参到
getTransaction
PlatformTransactionManager
- getTransaction()
- commit()
- rollback()
getTransaction
得到
TransactionStatus
- hasSavepoint()
- isRollbackOnly()
- isCompleted()
DataSourceTran
sactionManager
HibernateTrans
actionManager
JpaTransaction
Manager
JtaTransaction
Manager
JDBC
Hibernate
JPA
JTA

* 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）
```

```
 UPDATE employee SET job='Java开发工程师',departid=1 WHERE empno='LG008';
 UPDATE department SET dnum=dnum-1 WHERE deptno=1;
 UPDATE department SET dnum=dnum+1 WHERE deptno=2;
 * dao
public interface EmployeeDao {
    @Update("UPDATE employee SET job=#{job},departid=#{ndid} WHERE empno=#{empr
    void updateEmployee(Map<String,Object> params);
}
public interface DepartmentDao {
    @Update("UPDATE department SET dnum=dnum+1 WHERE deptno=#{ndid}")
    void updateDepartmentByIncrease(Map<String,Object> params);
    @Update("UPDATE department SET dnum=dnum-1 WHERE deptno=#{odid}")
    void updateDepartmentByDecrease(Map<String,Object> params);
}
 * service
public interface TransferEmployeeService {
    void transferEmployee(Map<String,Object> params);
}
@Service
public class TransferEmployeeServiceImpl implements TransferEmployeeService {
    @Autowired
    private EmployeeDao employeeDao;
    @Autowired
    private DepartmentDao departmentDao;

    @Transactional
    @Override
    public void transferEmployee(Map<String, Object> params) {
        employeeDao.updateEmployee(params);
        // 待转部门加1
        departmentDao.updateDepartmentByIncrease(params);
//        int i=100/0;
        // 原部门减1
        departmentDao.updateDepartmentByDecrease(params);
    }
}
* 单元测试
 @RunWith(SpringJUnit4ClassRunner.class)
 @ContextConfiguration("classpath:applicationContext.xml")
public class AppTest
```

```
{
    @Autowired
    private CompanyService companyService;
    @Test
    public void test1(){
        Company company = companyService.getCompany(1001);
        System.out.println(company);
    }
}


* 配置（xml）
 * spring-mybatis.xml
  <bean id="transactionManager" class="org.springframework.jdbc.datasource.Data
        <property name="dataSource" ref="druid"/>
    </bean>
    <tx:advice id="txAspect" transaction-manager="transactionManager">
        <tx:attributes>
            <tx:method name="transferEmployee" propagation="REQUIRED"/>
        </tx:attributes>
    </tx:advice>
    <aop:config>
        <aop:advisor advice-ref="txAspect" pointcut="execution(* *.transferEmpl
    </aop:config>
  * 单元测试
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath:applicationContext.xml")
public class AppTest
{
    @Autowired
    private TransferEmployeeService transferEmployeeService;
    @Test
    public void test2(){
        Map<String,Object> params=new HashMap<>();
        params.put("empno","LG008");
        params.put("job","html讲师456");
        params.put("odid",2);// -1
        params.put("ndid",1);// +1
        transferEmployeeService.transferEmployee(params);
    }
```

```
}
* 配置（xml+注解）
  * 在xml中
  <bean id="transactionManager" class="org.springframework.jdbc.datasource.Data
        <property name="dataSource" ref="druid"/>
  </bean>
  <tx:annotation-driven transaction-manager="transactionManager"/>
  * 在方法中
    @Transactional
    @Override
    public void transferEmployee(Map<String, Object> params) {
        employeeDao.updateEmployee(params);
        // 待转部门加1
        departmentDao.updateDepartmentByIncrease(params);
//        int i=100/0;
        // 原部门减1
        departmentDao.updateDepartmentByDecrease(params);
    }
 * 继续通过加装xml形式测试
  * @ContextConfiguration("classpath:applicationContext.xml")

* 配置（注解）
 * 在SMConfiguration加上注解
  * @EnableTransactionManagement
  * 加上获取事务管理器的方法
 @Bean
 public TransactionManager transactionManager(DataSource dataSource){
      DataSourceTransactionManager transactionManager=new DataSourceTransactior
      transactionManager.setDataSource(dataSource);
      return transactionManager;
 }
 * 单元测试
@RunWith(SpringJUnit4ClassRunner.class)
@SpringJUnitConfig(SpringConfiguration.class)
public class AppTest2
{
    @Autowired
    private TransferEmployeeService transferEmployeeService;
    @Test
    public void test2(){
```

```
        Map<String,Object> params=new HashMap<>();
        params.put("empno","LG008");
        params.put("job","html工程师1000");
        params.put("odid",1);// -1
        params.put("ndid",2);// +1
        transferEmployeeService.transferEmployee(params);
    }
}

* 案例二（事务传播属性）
 * 前期准备
@Service
public class PropagationServiceImpl {
    @Autowired
    private EmployeeDao employeeDao;
    @Autowired
    private DepartmentDao departmentDao;
    @Transactional(propagation = Propagation.REQUIRED)
    public void serviceB(Map<String, Object> params){
        employeeDao.updateEmployee(params);
        // 待转部门加1
        departmentDao.updateDepartmentByIncrease(params);
        int i=100/0;
        // 原部门减1
        departmentDao.updateDepartmentByDecrease(params);
    }

    @Transactional(propagation = Propagation.REQUIRED)
    public void service(){
        Map<String,Object> paramsB=new HashMap<>();
        paramsB.put("empno","LG003");
        paramsB.put("job","html工程师88");
        paramsB.put("odid",2);// -1
        paramsB.put("ndid",3);// +1
        serviceB(paramsB);
    }

}

 * 测试：PROPAGATION_REQUIRED
```

```
  * 支持当前事务；如果不存在，则创建一个新事务

 * 测试：PROPAGATION_SUPPORTS
  * 支持当前事务；如果当前没有事务，就以非事务方式执行
   * 修改属性：@Transactional(propagation = Propagation.SUPPORTS)

* 测试：PROPAGATION_MANDATORY
  * 支持当前事务；如果没有当前事务，则引发异常
   * 修改属性：@Transactional(propagation = Propagation.MANDATORY)

* 单元测试代码
* 当前没有事务，调用ServiceB
    @Autowired
    private PropagationServiceImpl propagationService;
    @Test
    public void test3(){
        Map<String,Object> paramsB=new HashMap<>();
        paramsB.put("empno","LG003");
        paramsB.put("job","html工程师88");
        paramsB.put("odid",2);// -1
        paramsB.put("ndid",3);// +1
        propagationService.serviceB(paramsB);
    }
  * 当前有事务，调用ServiceB
     @Test
    public void test3(){
        propagationService.service();
    }
```