# 第一章 概述

Spring：出现是在2002左右，解决企业开发的难度-减轻对项目模块之间的管理，类和类之间的管理，帮助开发人员创建对象，管理对象之间的关系。spring核心技术ioc , aop。能实现模块之间类之网的解耦合。

依赖：类A中使用类B中的属性或者方法，叫做类A依赖类B（如果没有B，A就不能用）

框架怎么学：框架是一个软件，其他人写好的软件。

1. 知道框架能做什么，mybatis—访问数据库，对表中的数据执行增删改查
2. 框架的语法，框架要完成一个功能，需要一定的步骤支持的
3. 框架的内部实现，框架的内部怎么做。原理是什么？
4. 通过学习，可以实现一个框架。（写框架，实现框架）

# 第二章控制反转

Spring的第一个核心思想。IOC（控制反转），是一个理论，概念，思想。描述：把对象的创建，赋值，管理工作都交给代码之外的容器实现。也就是对象的创建是有其它外部资源完成。

控制：创建对象，对象的属性赋值，对象之间的关系管理。

反转：把原来的开发人员管理，创建对象的权限转移给代码之外的容器实现。由容器代替工作人员管理对象，创建对象，给属性赋值

正转：由开发人员在代码中，使用new构造方法创建对象，开发人员主动管理对象。

为什么要使用ioc：目的就是减少对代码的改动，也能实现不同的功能。实现解耦合。

Java中创建对象有哪些方式：

1.构造方法，new Student（）

2.反射

3.序列化

4.克隆

5.Ioc：容器创建对象

6.动态代理

IOC的体现：

Servlet 1：创建类继承HttpServelt

2：在web.xml中注册servlet，使用<servlet-name>myservlet</servlet-name>

<servlet-class>com.bjpwernode.controller.MyServlet</servlet-class>

3：没有创建servlet对象，没有MyServlet myservlet=new MyServlet()

4：Servlet是Tomcat服务器它不是你创建的。Tomcat也称为容器。

Tomcat作为容器：里面存放servlet对象，Listener，Filter对象

IOC的技术实现，

DI是ioc的技术实现，DI（Dependency Injection）是依赖注入，只需要在程序中提供要使用的对象名称就可以，至于对象如何在容器中创建，赋值，查找都由容器内部实现。

Spring是使用的di实现了ioc的功能，spring底层创建对象，使用的反射机制。

Spring是一个容器，管理对象，给属性赋值，底层是反射创建对象。

Junit：单元测试，一个工具类库，做测试方法使用的。

单元：指定的是方法，一个类中有很多方法，一个方法称为一个单元。

1. 需要加入junit依赖。

<dependency>  
*<!-- 单元测试-->* <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.11</version>  
 <scope>test</scope>  
 </dependency>

1. 创建测试作用的类：叫做测试类。

src/test/java目录中创建类

1. 创建测试方法
2. public方法
3. 没有返回值void
4. 方法名称自定义，建议名称是test+你要测试的方法名称
5. 方法没有参数
6. 方法上面加入@Test，这样的方法是可以单独执行，不需要使用main方法。

什么样的对象放入容器中：dao类，service类，controller类，工具类

Spring中的对象默认都是单例，在容器中叫此名称对象仅一个

1. 使用xml配置文件，使用bean标签
2. 注解

不放到spring容器中的对象：实体类对象，实体类数据来自于数据库的。

Servlet，listener，filter等。

核心技术：IOC（控制反转）：1.理论，思想，概念：知道开发人员在容器中，代码之外管理对象，给属性赋值，管理依赖。

2.IOC技术实现的DI（依赖注入）：开发人员在项目中只需要提供对象的名称，对象的创建查找赋值都由容器自己实现

3.spring使用的DI的技术，底层使用的是反射机制

4.DI给属性赋值：1set注入，2property方法赋值；引用类型注入属性名和ref的bean的id

2构造注入：constructor-arg的index和name进行对应构造方法的赋值

使用spring框架的步骤：加入依赖，创建类：接口、实现类、没有接口的类，创建spring的配置文件xml，使用<bean>声明对象，在使用容器中的对象，通过ApplicationContext接口和他的实现类ClassPathXmlApplcationContext的方法getBean（）就可以拿到容器。

1. 多个配置优势
   1. 每个文件的大小比一个文件要小很多。效率高
   2. 避免多人竞争带来的冲突

如果你的项目有多个模块（相关的功能在一起），一个模块一个配置文件。

多文件的分配方式

* 1. 按功能模块，一个模块一个配置文件。
  2. 按类的功能，数据库相关的配置一个文件配置文件，做事务的功能一个配置文件，做service功能的一个配置文件等

1. 基于注解的di：通过注解完成java对象创建，属性赋值。

使用注解的步骤：

1、加入maven的依赖spring-context，在你加入spring-context的同时，

，间接加入spring-aop的依赖。使用注解必须使用aop的依赖。

2、在类中加入spring 的注解（多个不同功能的注解）

3、在spring的配置文件中，加入一个组件扫描器的标签，说明注解在你的项目中的位置

学习的注解：

1、@Component，创建对象

2、@Respotory，创建dao对象，用来访问数据库的。

3、@Service，创建Service对象，处理业务逻辑的，可以有事务功能。

4、@Controller，创建控制器对象，接受请求，显示处理结果。

5、@Value，简单类型的属性赋值

6、@Autowired，spring框架中引用类型的赋值注解，支持byName，byTyp默认是byType

7、@Resource，jdk中的注解，使用自动注入给引用赋值，支持byName，byType，默认是byName

8、注解的使用步骤：

1、加入依赖：spring-context，简介加入spring-aop

2、在类中加入注解

3、在spring的配置文件中，加入组件扫描器的标签<context:component-scan base-package=’包名’>

Ioc能实现业务之间的解耦合，例如service和dao对象之间的解耦合。

=============================================================

# 第三章 AOP

1．动态代理

实现方式：JDK动态代理，使用JDK中的Proxy，Method，InvocationHanderl创建代理对象。JDK动态代理要求目标类必须实现接口

Cglib动态代理：第三方的工具库，创建代理对象，原理是继承。通过继承目标类，创建子类，子类就是代理对象。要求目标类不能是final的，方法也不能是final

1. 动态代理的作用：

在目标类源代码不改变的情况下增加功能

减少代码的重复

专注业务逻辑代码

解耦合，让你的业务功能和日志，事务非业务功能分离

1. AOP：面向切面编程，基于动态代理的，可以使用jdk，cglib两种代理方式。Aop就是动态代理的规范化，把动态代理的实现步骤，方式都定义好了，让开发人员用一种统一的方式，就用动态代理。
2. AOP(Aspect Orient Programming)：面向切面编程

Aspect：切面，给你的目标类增加的功能，就是切面。上面的日志、事务

切面的特点：一般是非业务方法，独立使用

Orient：面向

Programming：编程

怎么理解面向切面编程？

1）需要在分析项目功能时，找出切面。

2）合理的安排切面的执行时间（目标方法前，还是目标方法后）

3）合理的安全切面执行位置，在哪个类具体增强什么功能。

术语：

1）Aspect：切面，表示增强的功能，代码，完成某个功能，非业务功能，常见的切面功能有日志，事务，统计信息，参数检查，权限验证。

2）JoinPoint：连接点，连接业务和切面的位置。某类的业务方法

3）Pointout：切入点，指多个连接点方法的集合。多个方法

4）目标对象：给哪个类的方法增加功能，这个类就是目标对象

5）Advice：通知，通知表示切面功能执行的时间。

说一个切面有三个关键的要素：

1）切面的功能代码，切面干什么

2）切面的执行位置，使用Pointcut表示切面执行的位置

3）切面的执行时间，使用Advice表示时间，在目标方法之前还是目标方法之后

5、Aop的实现

Aop是一个规范，是对动态代理的一个规范化，一个标准。

Aop的技术实现框架：

1、spring：spring内部实现了aop规范，能做aop的工作

Spring主要在事务处理使用aop

项目开发中很少使用spring 的aop实现，因为spring的aop比较笨重。

2、aspectJ：一个开源的专门做aop的框架。Spring框架中集成了aspectJ框架，通过spring就能使用aspectJ的功能。

aspectJ框架实现aop的两种方式：

1. 使用xml文件配置：配置全局事务
2. 使用注解，我们在项目中要做aop的功能，一般都用注解，aspectJ有5个注解。

6.学习aspectJ框架的使用

1）切面的执行时间，这个执行时间在规范中他叫做Advice（通知，增强），在aspectJ框架中使用注解表示的。也可以使用xml配置文件中的标签

1）@Before

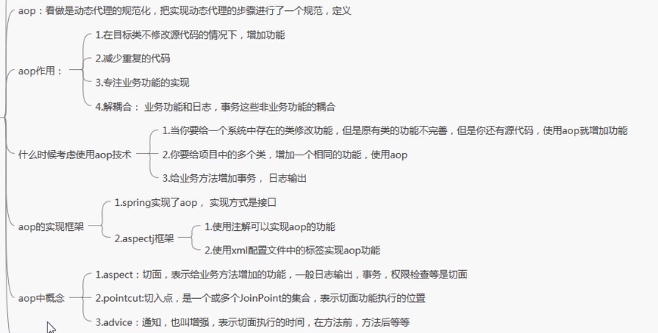
2）@AfterReturning

3）@Around

4）@AfterThrowing

5）@After

2）表示切面执行的位置，使用的是切入点表达式。



# 第四章集成Spring和MyBatis

用的技术：ioc。

为什么ioc：能把Mybatis和spring集成到一起，像一个框架，是因为ioc能创建对象。

可以吧Mybatis框架中的对象交给spring统一创建，开发人员从spring中获取对象。

开发人员就不用同时面对两个或多个框架，就面对一个spring框架

Mybatis使用步骤：对象

1. 定义dao接口，studentDao
2. 定义mapper文件 studentDao.xml
3. 定义mabatis的主配置文件 mybatis.xml文件
4. 创建dao 的代理对象，StudentDao dao=SqlSession.getMapper(StudentDao.class);

List<Student> students=dao.selectStudents();

要使用dao对象，需要使用getMapper()方法，

怎么能使用getMapper()方法，需要哪些条件

1. 获取SqlSession对象，需要使用SqlSessionFactory的openSession()方法。
2. 创建SqlSessionFactor对象。通过读取mybatis 的主配置文件，能创建SqlSessionFactory对象

需要SqlSessionFactory对象，使用Factory能获取SqlSession，有了SqlSession就能有dao，目的就是获取dao对象。Factory创建需要读取主配置文件。

主配置文件：

1. 数据库信息



1. Mapper文件的位置



通过以上的说明，我们需要让spring创建以下对象

1.独立的连接池类的对象，使用阿里的druid连接池

2.SqlSessionFactory对象

3.创建出dao对象

需要学习就是上面三个对象的创建语法，使用xml的bean标签。

复习Aop：

[1.@Before](mailto:1.@Before)：前置通知，在目标方法之前先执行切面的功能

[2.@AfterReturning](mailto:2.@AfterReturning)后置通知，在目标方法之后执行的，能够获取到目标方法的返回值

[3.@Around](mailto:3.@Around)环绕通知，在目标方法前和后都能增强功能，控制目标方法的访问，修改返回值

[4.@AterThrowing](mailto:4.@AterThrowing)异常通知，在目标方法抛出异常后执行的通知

[5.@After](mailto:5.@After)最终通知，总是会被执行的代码

[6.@Pointcut](mailto:6.@Pointcut)定义和管理切入点的辅助注解

7.代理的使用方法（1.如果目标有接口，框架使用jdk 的动态代理。如果目标类没有接口使用的是cglib动态代理。3.有接口也可以强制使用cglib动态代理。）

复习整合mybatis：

使用spring的ioc核心技术，把mybatis框架中使用的对象交给spring统一创建和管理。（Spring是容器，存放项目中要使用的各种对象，例如Service对象，Dao对象，工具类对象等等。）

1.交给spring的mybatis对象：1.数据源DataSource，使用阿里公司的Druid连接池

2.SqlSessionFactory独享，使用的SqlSessionFactoryBean在内部创建的SqlSessionFactory

3.Dao代理对象，使用的MapperScanConfigure，在这个类的内部，调用getMapper()，创建接口的Dao对象。

# Spring的事务处理

1. 什么是事务

讲mysql的时候，提出了事务，事务是指一组sql语句的集合，集合中有多条sql语句可能是insert，update，select，delete，我们希望这些多个sql语句都能成功，或者都失败，这些sql语句的执行是一直的，作为一个整体执行。

1. 在什么时候想到使用事务

当我的操作，涉及到多个表，或者是多个sql语句的insert，update，delete。需要保证这些语句都是成功才能完成我的功能，或者都失败，保证操作是符合要求的。

在java代码中写程序，控制事务，此时事务应该放在哪里呢？事务应该放在service类的业务方法上，因为业务方法会调用多个dao方法，执行多个sql语句

1. 通常使用JDBC访问数据库，还是mybatis访问数据库怎么处理事务

Jdbc访问数据库，处理事务 Connection conn；conn.commit()；conn.rollback();

Mybatis访问数据库，处理事务，SqlSession.commit()；SqlSession.rollback();

Hibernate访问数据库，处理事务，Session.commit()；Session.rollback()；

1. 3问题中事务的处理方式，有什么不足

1）不同的数据库访问技术，处理事务的对象，方法不同，需要了解不同数据库访问技术使用事务的原理。

2）掌握多种数据库中事务的处理逻辑。什么时候提交事务，什么时候回滚事务。

3）处理事务的多种方法。

总结：就是多种数据库的访问技术，有不同的事务处理的机制，对象，方法

1. 怎么解决不足

Spring提供一种处理事务的统一模型，能使用统一步骤，方式完成多种不同数据库访问技术的事务处理。

使用spring的事务处理机制，可以完成JDBC访问数据库的事务处理

使用spring的事务处理机制，可以完成mybatis访问数据的事务处理

使用spring的事务处理机制，可以完成hibernate访问数据库的事务处理

1. 处理事务，需要怎么做，做什么

Spring处理事务的模型，使用的步骤都是固定的。把事务使用的信息提供给spring就可以了。

1）事务内部提交，回滚事务，使用的事务管理器对象，代替你完成commit，rollback。事务管理器是一个接口和他的众多实现类。

接口：PlatformTransactionManager，定义了事务重要方法commit，rollback

实现类：spring把每一种数据库访问技术对应的事务处理类都创建好了。

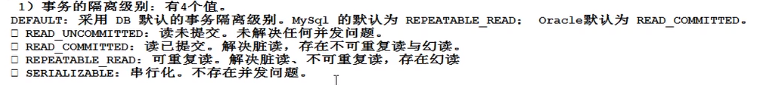
Mybatis访问数据库—spring创建好的DataSourceTransactionManager

Hibernate访问数据库—spring创建DataSourceTransactionManager

怎么使用：你需要告诉spring你用的哪种数据库访问技术，怎么告诉spring呢？声明数据库访问技术对于的事务管理器实现类，在spring的配置文件使用<bean>声明就可以了。例如：你要使用mybatis访问数据库，你应该在xml配置文件中

<bean id=”XXX” class=”…DataSourceTransactionManager”>

2）你的业务方法需要什么样的事务，说明需要事务的类型。

说明方法需要的事务：

2）事务的超时时间：表示一个方法最长的执行时间，如果方法执行时超过了，事务就会回滚。单位是秒，整数值，默认是-1。

3）事务的传播行为：控制业务方法是不是有事务的，是什么样的事务的。7个传播行为，表示你的业务方法调用时，业务在方法之间是如果使用的。

PROPAGATION\_REQUIRED

PROPAGATION\_REQUIRES\_NEW

PROPAGATION\_SUPPORTS

3）事务提交事务，回滚事务的时机

1）当你的业务方法，执行成功，没有异常抛出，当方法执行完毕，spring的方法执行后提交事务。事务管理器commit

2）当你的业务方法抛出运行异常，spring执行回滚，调用事务管理器的rollback运行时异常的定义：RuntimeException和他的子类都是运行时异常，例如NullPointException，NumberFormatException

3）当你的业务方法抛出非运行时异常，主要是受异常时，提交事务

受查异常：在你写代码中，必须处理的异常。例如IOExcetion，SQLException

**总结soring的事务**

1. 管理事务的是事务管理和他的实现类
2. Spring的事务是一个统一的模型

1）指定要使用的事务管理器实现类，使用<bean>

2）指定哪些类，哪些方法需要加入事务的功能

3）指定方法需要的隔离级别，传播行为，超时时间

你需要告诉spring，你的项目类的信息，方法的名称，方法的事务传播行为。

Spring框架中提供的事务处理方案

1. 适合中小项目使用的，注解方案。

spring框架自己用aop实现给业务方法增加事务的功能，使用@Transactional注解增加事务。Transactional注解是spring框架自己注解，放在public方法的上面，表示当前方法具有事务。可以给注解的属性赋值，表示具体的隔离级别，传播行为，异常信息等等。

使用@Transactional的步骤

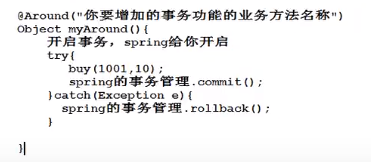
1. 需要声明事务管理器对象

<bean id=”XX” class=”DataSourceTransactionMannager”>

1. 开启事务注解驱动，告诉spring框架，我要使用注解的方式管理事务。

Spring使用aop机制，创建@Transactional所在的类代理对象，给方法加入事务的功能。Spring给业务加入事务：

在你的业务执行之前，先开启事务，在业务方法之后提交或回滚事务，使用aop的环绕通知

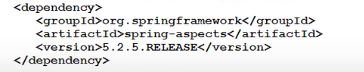


[3.在你的方法上面加入@Trancational](mailto:3.在你的方法上面加入@Trancational)注解

2.适合大型项目，有很多的类，方法，需要大量的配置事务，使用aspectj框架功能，在spring配置文件中声明类，方法需要的事务。这种方式业务方法和事务配置完全分离。

实现步骤：都是在xml配置文件中实现。

1）要使用的是aspectj框架，需要加依赖



2）声明事务管理器对象

<bean id=”XXX” class=”DataSourceTransactionManager”>

3）声明方法需要的事务类型（配置方法的事务属性【隔离级别，传播行为，超时】）

4）配置aop：指定哪些类要创建代理。

# 第六章：web项目中怎么使用容器对象。

1.之前做的是javase项目有main方法的，执行代码是执行main方法，

在main里面创建的容器对象

ApplicationContext ctx=new ClassPathXmlApplicationContext(“applicationContext.xml”);

2.web项目是在tomcat服务器上创建的，tomcat一启动，项目一直运行。

需求：

Web项目中容器对象只需要创建一次，把容器对象放到全局作用域ServletContext中。

怎么实现：

使用监听器 当全局作用域对象被创建时 创建容器 存入ServletContext

监听器的作用：

1）创建容器对象，执行ApplicationContext ctx=new ClassPathApplicationContext(“applicationContext.xml”)

2）把容器对象放入ServletContext， ServletContext.setAttribute(key,ctx)

监听器可以自己创建，也可以使用框架中创建好的ContextLoaderListener（知道用法）

private wabApplicationcontext context;

public interface webApplicationcontext extends ApplicationContext

ApplicationContext：javase项目中使用的容器对象

webApplicationcontext：web项目中使用的容器对象

把创建的容器对象，放入到全局作用域

