Spring-001-认识框架

Spring全家桶： spring, springmvc, spring boot, spring cloud

Spring：出现时在2002左右， 解决企业开发的难度。减轻对项目模块之前的管理，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，类之间的解耦合。

依赖：class A中使用class B的属性或者方法，叫做class A依赖class B。

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，框架内部怎么做，原理是什么。----提升
4. 通过学习，可以实现一个框架。

Spring的核心功能---IOC

Ioc：控制反转，是一个理论，概念，思想。

描述的：把对象的创建，赋值，管理工作都交给代码之外的容器实现，也就是对象的创建时有其他外部资源完成。

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。由容器代替开发人员管理对象。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。

Public static void main(String args[]) {

Student student = new Student(); // 在代码中，创建对象。--正转

}

容器： 是一个服务器软件，一个框架(spring)

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服务器创建的。Tomcat也称为容器。

Tomcat作为容器，里面存放的有Servlet对象，Listener, Filter对象

Ioc的技术实现

DI是ioc的技术实现，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，使用的是反射机制。

Spring是一个容器，管理对象，给属性赋值，底层是反射创建的对象。

Junit：单元测试，一个工具类库，做测试方法使用的。

单元：指的是方法，一个类中有很多方法，一个方法称为单元

使用单元测试

1. 需要加入junit依赖。

<dependency>

<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 + 你要测试方法名称
5. 方法没有参数
6. 方法的上面加入@Test，这样的方法是可以单独执行的。不用使用main方法
7. 多个配置文件优势
8. 每个文件的大小比一个文件要小很多。效率高
9. 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。

学生考勤模块一个配置文件，张三

学生成绩一个配置文件， 李四

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配置文件，做事务的功能一个配置文件，锁service功能的一个配置文件等。
3. 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springcontext的同时，简介加入spring-aop的依赖。使用注解必须使用spring-aop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

2. @Respotory

3. @Service

4. @Controller

5. @Vaule

6. @Autowired

7. @Resource

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, InvocationHandler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。子类就是代理兑现个，要求目标类不能是final的，方法也不能是final的

2. 动态代理的作用：

1）在目标类源代码不改变的情况下，增加功能。

2）减少代码的重复

3）专注业务逻辑代码

4）解耦合，让你的业务功能和日志，事务非业务功能分离。

3. Aop: 面向切面变成，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，使用动态代理。

4. 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，像上面用的日志，事务都是切面。

切面的特点：一般都是非业务方法，独立使用的。

Orient：面向，对等。

Programming：编程。

怎么理解面向切面编程？

1. 需要分析项目功能时，找出切面。
2. 合理的安排切面的执行时间（在目标方法前，还是目标方法后）
3. 合理的安排切面执行的位置，在哪个类，哪个方法增加增强功能

术语：

1. Aspect: 切面，表示增强的功能，就是一堆代码，完成某一个功能，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。

1. JoinPoint：连接点，连接业务方法和切面的位置。就是某个类中的业务方法。
2. PointCut: 切入点，指多个连接点方法的集合。多个方法
3. 目标对象：给哪个类的方法增加功能，这个类就是目标对象
4. Advice：通知，表示切面功能执行的时机。

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Adivce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，是动态的一个规范化，一个标准

Aop的技术实现框架：

1. Spring: spring在内部实现了aop规范，能做aop的工作。

Spring主要在事务处理时使用aop。

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

1. 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有两种方式：

1. 使用xml的配置文件：配置全局事务
2. 使用注解，我们在项目中要做aop功能，一般都使用注解，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叫做Advice（通知，增强）

在aspcetj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：把mybatis框架和spring集成在一起，像一个框架一样使用

用的技术是 ioc

为什么ioc：能把mybatis和spring集成在一起，像一个框架，是因为ioc能创建对象。

可以把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了，就面对一个spring。

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mybatis的主配置文件mybatis.xml
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SqlSessionFactory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

我们会使用独立的连接池类替换mybatis默认自己带的，把连接池类也交给spring创建。

主配置文件：

1. 数据库信息

<environment id="mydev">

<!--

transactionManager : mybatis的事务类型

type：JDBC（表示使用jdbc中的Connection对象的commit, rollback做事务处理)

-->

<transactionManager type="JDBC"/>

<!--

dataSource：表示数据源，连接数据库的

type: 表示数据源类型， POOLED表示使用连接池

-->

<dataSource type="POOLED">

<!--

driver, user, username, password是固定的，不能自定义

-->

<!-- 数据库的驱动类名 -->

<property name="driver" value="com.mysql.jdbc.Driver"/>

<!-- 连接数据库的url字符串 -->

<property name="url" value="jdbc:mysql://localhost:3306/ssm?characterEncoding=utf8"/>

<!-- 访问数据库的用户名 -->

<property name="username" value="root"/>

<!-- 密码 -->

<property name="password" value="123456"/>

</dataSource>

</environment>

1. Mapper文件的位置

<mappers>

<mapper resource="org/example/dao/StudentDao.xml"/>

</mappers>

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。

Spring的事务处理

回答问题

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都能成功，或者都失败，这些sql语句的执行是一致的，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，或者都是失败，保证操作时符合要求的。

在java代码中写程序，控制事务，此时事务应该放在哪里呢？

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？
2. 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的原理
3. 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
4. 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提供了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mybatis访问数据库的事务处理

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处理事务的魔心个，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

事务管理器是一个借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象的事务处理类都创建好了。

Mybatis访问数据库----spring创建好的是DataSourceTransactionManager

Hibernate访问数据库----spring创建的是HibernateTransactionManager

怎么使用：你需要告诉spring你用的是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应的事务管理器实现类，在spring的配置文件中使用<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=”…DataSourceTransationManager”/>

1. 你的业务需要什么样的事务，说明需要事务的类型。

说明方法需要的事务：

1. 事务的隔离级别：有4个值。

DEFAULT：采用DB默认的事务隔离级别。Mysql默认为REPEATABLE\_READ; Oracle默认为READ\_COMMITTED。

READ\_UNCOMMITTED: 读未提交。未解决任何并发问题。

READ\_COMMITED: 读已提交。解决脏读，存在不可重复读和幻读。

REPEATABLE\_READ: 可重复读。解决脏读、不能重复读，存在幻读。

SERIALIZABLE：串行化，不存在并发问题。

1. 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时间，事务就回滚。单位是秒，整数值，默认是-1.
2. 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，事务在方法之间是如何使用的。

PROPAGATION\_REQUIRED

PROPAGATION\_REQUIRES\_NEW

PROPAGATION\_SUPPORTS

以上三个需要掌握

PROPAGATION\_MANDATORY

PROPAGATION\_NESTED

PROPAGATION\_NEVER

PROPAGATION\_NOT\_SUPPORTED

1. 事务提交事务，回滚事务的时机
2. 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在方法执行后，提交事务。事务管理器commit
3. 当你的业务方法抛出运行时异常或ERROR，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

1. 当你的业务方法抛出非运行时异常，主要是受查异常时，提交事务

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

总结spring的事务：

1. 管理事务的是事务管理和他的实现类
2. Spring的事务是一个统一模型
3. 指定要使用的事务管理器实现类，使用<bean>
4. 指定哪些类，哪些方法需要加入事务的功能
5. 指定方法需要的隔离级别，传播行为，超时

你需要告诉spring，你的项目中类信息，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@Transaction注解增加事务。

@Transaction注解是spring框架自己的注解，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。

使用@Transaction的步骤：

1. 需要声明事务管理器的对象：

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

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@Transaction所在的类代理对象，给方法加入事务的功能。

Spring给业务方法加入事务：

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

@Around(“你要增加的事务功能的业务方法名称”)

Object myAround() {

开启事务，spring给你开启

try {

buy(1001, 10);

spring的事务管理器.commit();

} catch(Exception e) {

Spring的事务管理器.rollback();

}

}

1. 在你的方法的上面加入@Transactional
2. 适合大型项目，有很多类，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. 需要使用的是aspectj框架，需要加入依赖

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>5.2.5.RELEASE</version>

</dependency>

1. 声明事务管理器对象

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

1. 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】）
2. 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. 做的是javase项目有main方法的，执行方法是执行main方法的，

在main里面创建的容器对象

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

2. web项目是在tomcat服务器上运行的。Tomcat一启动，项目一直运行的。

需求：

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

怎么实现：

使用监听器，当全局作用于对象被创建时，创建容器，存入ServletContext中

监听器作用：

1. 创建容器对象，执行ApplicationContext ctx = ClassPathXmlApplicationContext(“applicationContext.xml”);
2. 把容器对象放入到ServletContext, ServletContext.setAttribute(key, ctx)

监听器可以自己创建，也可以使用框架中提供好的ContextLoaderListener