目录

[1 Spring框架 1](#_Toc491932538)

[1.1 Spring的相关概念 1](#_Toc491932539)

[1.2 Spring中ioc的底层原理 1](#_Toc491932540)

[1.3 Spring中ioc的入门案例 1](#_Toc491932541)

[1.4 Bean的三种实例化方式 1](#_Toc491932542)

[1.5 Bean标签的常用属性（id，class，name，scope） 1](#_Toc491932543)

[1.6 属性注入介绍（DI） 1](#_Toc491932544)

[1.7 注入对象类型的属性(ref关键字) 1](#_Toc491932545)

[1.8 P名称空间的注入 2](#_Toc491932546)

[1.9 Spring注入复杂数据 2](#_Toc491932547)

[1.10 IOC和DI的区别 3](#_Toc491932548)

[1.11 Spring整合web项目原理 3](#_Toc491932549)

[1.12 Spring的bean管理方式，注解方式 4](#_Toc491932550)

[1.12.1 注解开发准备工作 4](#_Toc491932551)

[1.12.2 注解创建对象 4](#_Toc491932552)

[1.12.3 注解设置属性 5](#_Toc491932553)

[1.12.4 注解和配置文件混合使用 5](#_Toc491932554)

[1.13 AOP的原理 5](#_Toc491932555)

# Junit

<http://www.yiibai.com/junit/what-is-junit.html>

# Spring框架

## Spring的相关概念

## Spring中ioc的底层原理

Xml配置文件 🡪 dom4j解析xml配置文件，利用反射机制创建类的实例

## Spring中ioc的入门案例

## Bean的三种实例化方式

1. 无参数构造函数实例化
2. 工厂类静态方法实例化
3. 工厂类非静态方法实例化

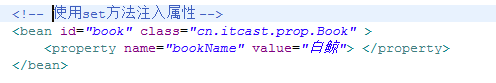
## Bean标签的常用属性（id，class，name，scope）

## 属性注入介绍（DI）

属性注入（设置）：由配置文件创建对象实例的时候，给实例中的属性赋值

Spring主要实现两种方式实现属性赋值

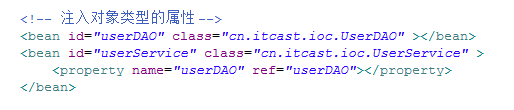
1. set方法设置（看到第13集）



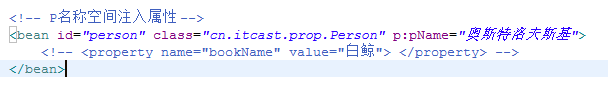
1. 带参数构造函数的属性设置（12-使用有参数构造注入属性.avi）

## 注入对象类型的属性(ref关键字)

1. 创建service类和dao类
2. 在service类中有dao对象



## P名称空间的注入



## Spring注入复杂数据

|  |
| --- |
| 1. 数组类型 2. List集合 3. Map集合 4. Properties类型 |

## IOC和DI的区别

|  |
| --- |
| 1. IOC控制反转，把对象的创建交给spring进行配置 2. DI，依赖注入， 向类里面的属性设置值 3. 关系，依赖注入是不能单独存在的，需要在ioc的基础之上完成。 |

## Spring整合web项目原理

|  |
| --- |
| 1. 加载核心配置文件     效率很低。   1. 把加载配置文件和创建对象的过程，在服务器启动的时候完成 2. 实现原理   ServelteContext对象  监听器监听serveletContext对象在何时创建  具体使用  ・在服务器启动的时候，会为每个项目创建一个serveletContext对象🡪  ・在serveletContext对象创建的时候，被监听器监听到 🡪  ・监听到之后，加载spring配置文件，创建对象 🡪  ・把创建出来的对象放到serveletContext域对象中(setAttribute) 🡪  ・获取对象的时候，到serveletContext域对象中获取(getAttribute) |

## Spring的bean管理方式，注解方式

|  |
| --- |
| 注解：代码里面的特殊标记  注解写法：  @注解名称(属性名称=属性值)  注解的应用地方：  类，方法，属性上面都可以使用注解 |

### 注解开发准备工作

|  |
| --- |
| * 导入spring基本的4个jar包 * 导入aop的jar包 * 创建类，方法   创建spring的配置文件，引入约束   * 开启注解扫描功能 |

### 注解创建对象

|  |
| --- |
| 在创建对象的类上面使用注解即可    Value属性可以省略，创建对象有四个注解   * @Component * @controller // 控制类注解 * @Service // 业务类注解 * @Repository // 持久层类注解   创建对象是单实例还是多实例 |

### 注解设置属性

|  |
| --- |
| 1. 创建service类，创建dao类，在service类中得到dao对象 2. 创建dao属性，不用set方法，采用autoWired，自动装载，根据类名去找 3. 创建dao属性，不用set方法，采用Resource注解，指定name |

### 注解和配置文件混合使用

05-spring的bean管理（注解和配置文件混合使用）.avi

|  |
| --- |
| 创建对象操作，都使用配置文件的方式    注入属性的操作，一般使用注解方式 |

## AOP的原理

|  |
| --- |
| Aop的概念  面向切面编程，扩展功能不是通过修改源代码实现，而是通过配置来实现  Aop采用横向抽取机制，取代了传统的纵向继承体系重复性代码（性能监视，事物管理，安全检查，日志等） |
| Aop原理  底层使用动态代理的方式来实现    有接口的情况，用jdk动态代理  ----------------------------------------------------------------------------------------------------------------------    没有接口的情况，用cglib动态代理 |
| Aop操作的术语  JoinPoint（连接点）：就是被拦截到的方法  Pointcut（切入点）：就是一系列JoinPoint的集合  Advice（通知/增加）：就是拦截到方法之后，做的相关操作（方法）  Aspect（切面）：就是Pointcut和Advice通知的结合 |
| Spring的aop操作  在spring框架中使用AspectJ实现aop操作  aspectJ不是spring的一部分，而是和spring一起使用，spring2.0以后增加了对aspectJ的支持  使用aspectJ进行aop操作，主要有两种方式   1. 基于aspectJ的xml的配置 2. 基于aspectJ的注解方式来实现 |

## AOP操作实例

|  |
| --- |
| 1. 需要导入aop相关的jar包 2. 添加相关约束 3. 使用表达式配置切入点 4. execution(访问修饰符[空格]增强方法全路径)   例1：execution(访问修饰符 cn.itcast.aop.Book.add(..))  例2：execution(\* cn.itcast.aop.Book.\*(..)) 对Book类中的所有方法进行增强  例3：execution(\* \*.\*(..)) 对所有类中的所有方法进行增强  例4：execution(\* save\*(..)) 匹配所有save开头的方法   1. 前置，后置，环绕 |

## Log4j介绍

|  |
| --- |
| 通过log4j可以看到程序运行过程中更详细的信息，log4j很常用  使用方法：   1. 导入log4j的jar包 2. 复制log4j的配置文件到src下面 |

## Spring整合web项目的演示

|  |
| --- |
| 问题  Action调用service，service再调dao。Action中会多次调用加载xml文件的方法 |
| 解决方案  Spring框架自带。只需要配置监听器就可以了。  要先导入spring整合web项目的jar包  Spring-web-xxx.jar  在web.xml中配置    指定要加载的spring配置文件的位置 |

## 基于注解的aspectj实现aop操作

|  |
| --- |
| 1. 创建类的对象 2. 在spring的核心配置文件中，开启aop扫描操作      1. 在增强类和被增强类上添加注解 |

# SpringMVC学习步骤

1. 搭建环境
2. 如何完成controller和viewer的映射
3. 如何把值传递给controller控制器
4. controller控制器如何把值传递给viewer
5. 异常处理
6. 页面标签
7. 文件上传
8. 深入源代码

## 框架流程

|  |
| --- |
| 1. 浏览器发起url请求给dispatcherservlet 2. Dispatcher Servlet调用Handler mapping 3. 通过Handler mapping把请求转交到对应的Controller 4. Controller调用Model层的业务处理之后，返回Model and logical view name给DispatcherServlet 5. DispatcherServlet再调用ViewResolver 6. ViewResolver解析返回的ModelAndView，返回View页面给客户端 |

## HelloWorld入门

|  |
| --- |
| 1. 导入相应的jar包 2. 配置web.xml文件 3. 配置\*-serverlet.xml（注意文件名的前半部分要和web.xml中配置的<servlet-name>同名）文件 4. 创建controller类 |

## 页面值传递

|  |
| --- |
| 试图把值传递给controller    页面侧，url直接传递 |
| Controller把值传递给视图    视图侧打印 |

## SpringMVC的rest简介

|  |
| --- |
| 通常如果要查询某个实例，网页url风格为：  <http://....?id=xxx>  如果采用rest风格则写成  <http://..../xxx> |

## SpringMVC简单用户管理

04.基于SpringMVC实现简单用户管理（1）.mp4

25：12