课程目标:

1. 介绍Spring和搭建Spring的环境
2. **理解Spring IoC的原理 重点 难点**
3. **掌握Spring IoC的配置重点 难点**

编程思想:

面向过程->面向对象->面向接口->面向组件(Spring的IOC)->面向切面(Spring的Aop)->面向服务

1. 介绍spring
2. Spring是一个轻量级框架，Java EE的春天。

Spring框架可以理解:组装型框架。

2. spring目标:

使现有技术更加易用，推进编码的最佳实践.

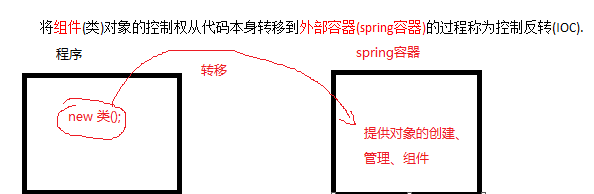
3. spring包含的内容:

IOC容器、AOP实现、数据访问支持(提供JdbcTemplate对象简化Jdbc操作、简化Mybatis操作)、web集成(springmvc支持)

1. Spring的IOC(控制反转)
2. 什么是IOC:

将组件(类)对象的控制权从代码本身转移到外部容器(spring容器)的过程称为控制反转(IOC).

注意;控制反转的目的是实现程序的解藕



1. 搭建spring的开发环境实现IOC的步骤:
   1. 导入spring框架依赖

|  |
| --- |
| <**properties**>  *<!--指定springmvc的版本-->* <**spring.version**>4.3.3.RELEASE</**spring.version**> </**properties**> <**dependencies**>  *<!--spring依赖 start-->* <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context-support</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-core</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-oxm</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-tx</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-jdbc</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-aop</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-test</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  *<!-- 引入Spring 依赖 end -->* </**dependencies**> |

* 1. 添加spring框架的配置文件(applicationContext.xml)并存于resources目录

|  |
| --- |
| *<?***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:p="http://www.springframework.org/schema/p"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="  http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-4.3.xsd  http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd"**> </**beans**> |

* 1. 在spring配置文件中使用bean标签创建对象(IOC实现)

|  |
| --- |
| *<!-- 使用bean标签添创建bean对象 <bean id="bean对象id" class="类的限定名"></bean> -->* <**bean id="student" class="com.Student"**></**bean**> |

3. 使用spring容器中的对象的步骤

|  |
| --- |
| *//1.创建spring容器* ApplicationContext ctx=**new** ClassPathXmlApplicationContext(**"applicationContext.xml"**); *//2.使用getBean方法获取容器中的对象 //ctx.getBean("bean对象的id值")* Student stu=(Student) ctx.getBean(**"stu"**); *//3.使用对象* stu.setXh(101); stu.setName(**"张三"**); stu.setSex(**"男"**); stu.show(); |

1. 依赖注入(理解:利用依赖注入给类的私有属性赋值)

目的:将组件对象的构建和使用分离

1. 依赖注入的方式

属性注入、构造注入、工厂注入….

1. 使用property标签进行属性注入

2.1 注入基本类型(包含String) 使用value属性

<**property name="属性名称" value="101"**></**property**>

2.2 注入对象类型 使用ref属性

<**property name="属性名称" ref="bean对象的id值"**></**property**>

|  |
| --- |
| 示例:  <!--使用bean标签创建类的对象(实现IOC)  <bean id="bean对象的名称" class="类的限定名称"></bean> --> <**bean id="stu" class="com.kgc.Student"**>  <!--使用property标签实现属性注入:实现类的私有 属性赋值-->  <**property name="xh" value="102"**></**property**>  <**property name="name" value="张三"**></**property**>  <**property name="sex" value="男"**></**property**>  <**property name="address" ref="address"**></**property**> </**bean**> <!--创建地址对象--> <**bean id="address" class="com.kgc.Address"**>  <**property name="bh" value="4300"**></**property**> <**property name="info" value="湖北武汉某区1090"**></**property**> </**bean**> |

1. 使用p命名空间注入(简化属性注入)

p:属性名=”值” 注入基本类型

p:属性名-ref=”bean对象的id值” 注入对象类型

|  |
| --- |
| *<!--p命名空间注入： 简化属性注入-->* <**bean id="student2" class="com.Student" p:xh="102" p:name="麻子" p:sex="女" p:address-ref="address"**> </**bean**> |

注意:使用p命名空间注入必需在beans标签中必需导入p命名空间

**xmlns:p="http://www.springframework.org/schema/p"**

1. 使用构造注入

注意:构造注入必需依赖于构造方法

示例:

|  |
| --- |
| <!--使用constructor-arg实现构造注入：实现学生属性的初始化-->  <**bean id="stu3" class="com.kgc.Student"**>  <!--  一个constructor-arg标签代表一个构造方法的参数  value属性给基本类型 ref属性指对象类型  index属性指定参数的顺序，不指定就按参数的顺序传递  -->  <**constructor-arg value="习大大" index="1"**></**constructor-arg**>  <**constructor-arg value="103" index="0"**></**constructor-arg**>  <**constructor-arg value="男" index="2"**></**constructor-arg**>  <**constructor-arg ref="address" index="3"**></**constructor-arg**> </**bean**>  学生类中添加构造方法:  注意：添加构造方法时一定添加默认构造  **public** Student(){} *//默认构造* **public** Student(Integer xh, String name, String sex, Address address) {  **this**.**xh** = xh;  **this**.**name** = name;  **this**.**sex** = sex;  **this**.**address** = address; } |

扩展:类与类之间的关系

1. 继承关系(纵向关系)
2. 利用类的属性实现:横向关系

Class Person{

Private Car car=new Car(); 1 对 1

Private List<Car> cars=new ArrayList<Car>; 1对多

}

Class Car{

Private List<Person> cars=new ArrayList< Person >; 1对多

}