狂神说Spring02：快速上手Spring

秦疆 狂神说 2020-04-16

狂神说Spring系列连载课程，通俗易懂，基于Spring最新版本，欢迎各位狂粉转发关注学习。禁止随意转载，转载记住贴出B站视频链接及公众号链接！

狂神说Spring01：概述及IOC理论推导

上一期中我们理解了IOC的基本思想，我们现在来看下Spring的应用：

HelloSpring

> 导入Jar包

注 : spring 需要导入commons-logging进行日志记录 . 我们利用maven , 他会自动下载对应的依赖项 .

```
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-webmvc</artifactId>
   <version>5.1.10.RELEASE</version>
</dependency>
```

> 编写代码

1、编写一个Hello实体类

```
public class Hello {
   private String name;

   public String getName() {
       return name;
  }
   public void setName(String name) {
       this.name = name;
  }

   public void show(){
       System.out.println("Hello,"+ name );
  }
}
```

2、编写我们的spring文件 , 这里我们命名为beans.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">

   <!--bean就是java对象 , 由Spring创建和管理-->
   <bean id="hello" class="com.kuang.pojo.Hello">
       <property name="name" value="Spring"/>
   </bean>

</beans>
```

3、我们可以去进行测试了 .

```java
@Test
public void test(){
   //解析beans.xml文件 , 生成管理相应的Bean对象
   ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
   //getBean : 参数即为spring配置文件中bean的id .
   Hello hello = (Hello) context.getBean("hello");
   hello.show();
}
```

> 思考

- Hello 对象是谁创建的？【hello 对象是由Spring创建的
- Hello 对象的属性是怎么设置的？ hello 对象的属性是由Spring容器设置的

这个过程就叫控制反转 :

- 控制 : 谁来控制对象的创建 , 传统应用程序的对象是由程序本身控制创建的 , 使用Spring后 , 对象是由Spring来创建的
- 反转 : 程序本身不创建对象 , 而变成被动的接收对象 .

依赖注入 : 就是利用set方法来进行注入的.

IOC是一种编程思想，由主动的编程变成被动的接收

可以通过newClassPathXmlApplicationContext去浏览一下底层源码 .

> 修改案例一

我们在案例一中， 新增一个Spring配置文件beans.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
       xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="MysqlImpl" class="com.kuang.dao.impl.UserDaoMySqlImpl"/>
    <bean id="OracleImpl" class="com.kuang.dao.impl.UserDaoOracleImpl"/>

    <bean id="ServiceImpl" class="com.kuang.service.impl.UserServiceImpl">
        <!--注意: 这里的name并不是属性 , 而是set方法后面的那部分 , 首字母小写-->
        <!--引用另外一个bean , 不是用value 而是用 ref-->
        <property name="userDao" ref="OracleImpl"/>
    </bean>

</beans>
```

测试!

```
@Test
public void test2(){
    ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
    UserServiceImpl serviceImpl = (UserServiceImpl) context.getBean("ServiceImpl");
    serviceImpl.getUser();
}
```

OK , 到了现在 , 我们彻底不用再程序中去改动了 , 要实现不同的操作 , 只需要在xml配置文件中进行修改 , 所谓的IoC,一句话搞定 : 对象由Spring 来创建 , 管理 , 装配 !

IOC创建对象方式

> 通过无参构造方法来创建

1、User.java

```
public class User {

    private String name;

    public User() {
        System.out.println("user无参构造方法");
    }

    public void setName(String name) {
        this.name = name;
    }

    public void show(){
        System.out.println("name="+ name );
```

```
  }

}
```

2、beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="user" class="com.kuang.pojo.User">
        <property name="name" value="kuangshen"/>
    </bean>

</beans>
```

3、测试类

```
@Test
public void test(){
    ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
    //在执行getBean的时候, user已经创建好了 , 通过无参构造
    User user = (User) context.getBean("user");
    //调用对象的方法 .
    user.show();
}
```

结果可以发现，在调用show方法之前，User对象已经通过无参构造初始化了！

> 通过有参构造方法来创建

1、UserT . java

```
public class UserT {

    private String name;

    public UserT(String name) {
        this.name = name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void show(){
        System.out.println("name="+ name );
    }
```

```
}
```

2、beans.xml 有三种方式编写

```
<!-- 第一种根据index参数下标设置 -->
<bean id="userT" class="com.kuang.pojo.UserT">
   <!-- index指构造方法 , 下标从0开始 -->
   <constructor-arg index="0" value="kuangshen2"/>
</bean>
```

```
<!-- 第二种根据参数名字设置 -->
<bean id="userT" class="com.kuang.pojo.UserT">
   <!-- name指参数名 -->
   <constructor-arg name="name" value="kuangshen2"/>
</bean>
```

```
<!-- 第三种根据参数类型设置 -->
<bean id="userT" class="com.kuang.pojo.UserT">
   <constructor-arg type="java.lang.String" value="kuangshen2"/>
</bean>
```

3、测试

```
@Test
public void testT(){
   ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
   UserT user = (UserT) context.getBean("userT");
   user.show();
}
```

结论：在配置文件加载的时候。其中管理的对象都已经初始化了！

Spring配置

> 别名

alias 设置别名 , 为bean设置别名 , 可以设置多个别名

```
<!--设置别名：在获取Bean的时候可以使用别名获取-->
<alias name="userT" alias="userNew"/>
```

> Bean的配置

```
<!--bean就是java对象,由Spring创建和管理-->

<!--
   id 是bean的标识符,要唯一,如果没有配置id,name就是默认标识符
   如果配置id,又配置了name,那么name是别名
   name可以设置多个别名,可以用逗号,分号,空格隔开
```

```
   如果不配置id和name,可以根据applicationContext.getBean(.class)获取对象;

class是bean的全限定名=包名+类名
-->
<bean id="hello" name="hello2 h2,h3;h4" class="com.kuang.pojo.Hello">
   <property name="name" value="Spring"/>
</bean>
```

import

团队的合作通过import来实现 .

```
<import resource="{path}/beans.xml"/>
```

小狂神温馨提示

到了这里，就算入门Spring了，认真体会它的好处吧！

end

视频同步更新

如果觉得帮助到了您，不妨赞赏支持一下吧！

“赠人玫瑰，手有余香”

狂神说 的赞赏码

狂神说