HelloSpring

1 HelloSpring

- 编写一个类

```java
package com.kuang.pojo;

import java.io.Serializable;

public class Hello implements Serializable {
    private String str;

    public String getStr() {
        return str;
    }

    public void setStr(String str) {
        this.str = str;
    }

    @Override
    public String toString() {
        return "Hello{" +
                "str='" + str + '\'' +
                '}';
    }
}
```

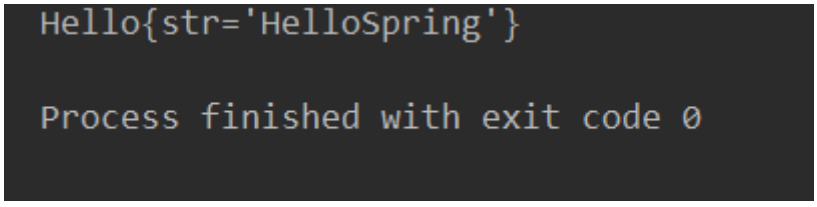
- 编写配置文件（这里先起名为beans.xml）

```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
        https://www.springframework.org/schema/beans/spring-beans.xsd">

     <!--使用Spring来创建对象
          在Spring中，这些都称为Bean
          bean = 对象
          class=Hello ===》new Hello();

          类型  变量名 = new 类型();
          Hello hello = new Hello();

          id ===> 变量名
          class ===> new的对象
          property ===> 相当于给对象中的属性设置一个值（利用set注入，pojo中去掉set会报错）
          -->
    <bean id="hello" class="com.kuang.pojo.Hello">
        <property name="str" value="HelloSpring"/>
    </bean>
```

```
</beans>
```

- 运行测试

```
package com.kuang.test;

import com.kuang.pojo.Hello;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class TestHello {
    @Test
    public void testHello() {
        //获取Spring的上下文对象
        ApplicationContext context = new
ClassPathXmlApplicationContext("beans.xml");

        //我们的对象现在都在Spring中管理，要使用直接在里面取出来即可
        Hello hello = (Hello) context.getBean("hello");

        System.out.println(hello.toString());


    }

}
```



因为，前面的property给Hello类的属性str赋值为“HelloSpring”，所以这里打印出来的str的属性值。

2 之前的那个例子

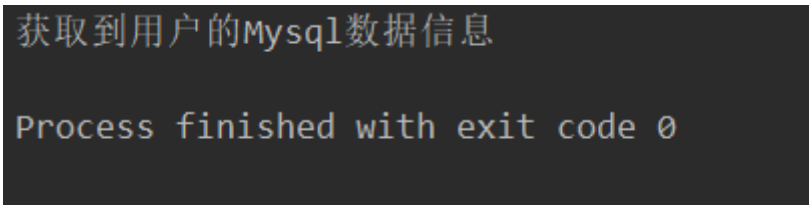
- bean配置

```
<?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
        https://www.springframework.org/schema/beans/spring-beans.xsd">

    <!--都没有属性，不需要property进行注入-->
    <bean id="mysqlImpl" class="com.kuang.daoImpl.UserDaoMySqlImpl"/>
    <bean id="oracleImpl" class="com.kuang.daoImpl.UserDaoOracleImpl"/>

    <bean id="userServiceImpl"
class="com.kuang.serviceImpl.UserServiceImpl">
        <!--
        ref --> 引用Spring容器中创建好的对象
        value --> 具体的值，基本数据类型
        -->
```

```
        <property name="userDao" ref="mysqlImpl"/>
    </bean>

</beans>
```

- 运行测试

```
@Test
public void testSpring() {
    //拿到Spring的容器
    ApplicationContext context = new
ClassPathXmlApplicationContext("beans.xml");

    UserServiceImpl userServiceImpl = (UserServiceImpl)
context.getBean("userServiceImpl");
    userServiceImpl.getUser();

}
```



因为配置的ref指向mysqlImpl对应的bean。

- 因此，我们可以根据需求修改配置文件即可，而不用动到源代码。修改配置文件就相对比较容易。

3 总结

以上的过程就叫控制反转 :

- 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 .

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 !