spring入门

依赖jar包：基础jar包自需要4个

appliactionContext.xml(spring2.0以前规范用的dtd文件，spring2.5以后使用xsd文件约束xml文件中跟元素和子元素的顺序。)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:context="http://www.springframework.org/schema/context"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"
 xmlns:tx="http://www.springframework.org/schema/tx" xmlns:p="http://www.springframework.org/schema/p"
 xmlns:util="http://www.springframework.org/schema/util" xmlns:jdbc="http://www.springframework.org/schema/jdbc"
 xmlns:cache="http://www.springframework.org/schema/cache"
 xmlns:context.="http://www.springframework.org/schema/util"
 xsi:schemaLocation="
           http://www.springframework.org/schema/context
           http://www.springframework.org/schema/context/spring-context.xsd
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/tx
           http://www.springframework.org/schema/tx/spring-tx.xsd
           http://www.springframework.org/schema/jdbc
           http://www.springframework.org/schema/jdbc/spring-jdbc-3.1.xsd
           http://www.springframework.org/schema/cache
           http://www.springframework.org/schema/cache/spring-cache-3.1.xsd
           http://www.springframework.org/schema/aop
           http://www.springframework.org/schema/aop/spring-aop.xsd
           http://www.springframework.org/schema/util
           http://www.springframework.org/schema/util/spring-util.xsd">

 <bean id="user" class="com.my_spring.pojo.User">
 <property name="userName" value="张三"></property>
 </bean>
</beans>
```

User.java(pojo)

```
package com.my_spring.pojo;

public class User {
 private Integer id;
 private String userName;
 private String password;
 public Integer getId() {
  return id;
 }
 public void setId(Integer id) {
  this.id = id;
 }
 public String getUserName() {
  return userName;
 }
 public void setUserName(String userName) {
  this.userName = userName;
 }
 public String getPassword() {
  return password;
 }
 public void setPassword(String password) {
  this.password = password;
 }

}
```

测试类

```
package com.my_spring.server;

import org.junit.Before;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.my_spring.pojo.User;
/**
 *
 * @ClassName:  UserTest
 * @Description:测试类（查询出spring注入的值）
 * @author: 杨旭东
 * @date:   2017年9月5日 上午11:40:06
 *
 * @Tel: 15903444833
 * @Copyright: @2017
 */
public class UserTest {
 private ApplicationContext applicationContext = null;
 @Before
 public void setUp() throws Exception {
  String resouce = "classpath:applicationContext.xml";
  System.out.println("xxxxxxxxxxx");
  applicationContext = new ClassPathXmlApplicationContext(resouce);
 }

  /**
  *
  * @Title: test1
  * @Description: ioc下的资源调度
  * @param:
  * @return: void
  * @throws
  */
 @Test
 public void test1() {
  User user = (User) applicationContext.getBean("user");
  System.out.println(user.getUserName());
 }
 /**
  *
  * @Title: test2
  * @Description: 传统资源调度
  * @param:
  * @return: void
  * @throws
  */
 @Test
 public void test2() {
  User user = new User();
  user.setUserName("王五");
  System.out.println(user.getUserName());
 }

}
```

通过上面的案例我们能发现，资源进过spring托管后，资源的获取方式由传统的主动获取变更到被动的获取。在测试类中user对象不再是new出来而是

通过spring直接获取即可，new的过程spring就帮我们操作了。释放了程序员的压力。而这种被动获取的机制就是IOC。控制反转。