

1. Spring概念

1 spring是开源的轻量级框架

2 spring核心主要两部分：

(1) aop：面向切面编程，扩展功能不是修改源代码实现

(2) ioc：控制反转，

- 比如有一个类，在类里面有方法（不是静态的方法），调用类里面的方法，创建类的对象，使用对象调用方法，创建类对象的过程，需要new出来对象

- 把对象的创建不是通过new方式实现，而是交给spring配置创建类对象

3 spring是一站式框架

(1) spring在javaee三层结构中，每一层都提供不同的解决技术

- web层：springMVC

- service层：spring的ioc

- dao层：spring的jdbcTemplate

4 spring版本

(1) hibernate5.x

(2) spring4.x

2. Spring的ioc操作

1 把对象的创建交给spring进行管理

2 ioc操作两部分：

(1) ioc的配置文件方式

(2) ioc的注解方式

3. IOC底层原理

1 ioc底层原理使用技术

(1) xml配置文件

(2) dom4j解析xml

(3) 工厂设计模式

(4) 反射

2 画图分析ioc实现原理

ioc原理

```
public class UserService {
```

```
public class UserServlet {  
    //得到UserService的对象  
    //原始：new创建  
    UserFactory.getService();
```

第一步 创建xml配置文件，配置要创建对象类

```
<bean id="userService" class="cn.itcast.UserService"/>
```



第二步 创建工厂类，使用dom4j解析配置文件+反射




```
//返回UserService对象的方法  
public static UserService getService() {  
    //1 使用dom4j解析xml文件  
    //根据id值 userService，得到id值对应class属性值  
    String classValue = "class属性值";  
    //2 使用反射创建类对象  
    Class clazz = Class.forName(classValue);  
    //创建类对象  
    UserService service = clazz.newInstance();  
    return service;
```

4. IOC入门案例

第一步 导入jar包

(1) 解压资料zip文件

Jar特点：都有三个jar包

 spring-beans-4.2.4.RELEASE.jar
 spring-beans-4.2.4.RELEASE-javadoc.jar
 spring-beans-4.2.4.RELEASE-sources.jar



(2) 做spring最基本功能时候，导入四个核心的jar包就可以了

(3) 导入支持日志输出的jar包

commons-logging-1.2.jar

log4j-1.2.16.jar

spring-beans-4.2.4.RELEASE.jar

spring-context-4.2.4.RELEASE.jar

spring-core-4.2.4.RELEASE.jar

spring-expression-4.2.4.RELEASE.jar

第二步 创建类，在类里面创建方法

```
public class User {  
  
    public void add() {  
        System.out.println("add.....");  
    }  
  
    public static void main(String[] args) {  
        //原始做法  
        User user = new User();  
        user.add();  
    }  
}
```

第三步 创建spring配置文件，配置创建类

(1) spring核心配置文件名称和位置不是固定的

- 建议放到src下面，官方建议**applicationContext.xml**

(2) 引入schema约束

« docs ▶ spring-framework-reference ▶ html ▶

```
<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">
```

(3) 配置对象创建

```
<!-- ioc入门 -->  
<bean id="user" class="cn.itcast.ioc.User"></bean>
```

第四步 写代码测试对象创建

(1) 这段代码在测试中使用

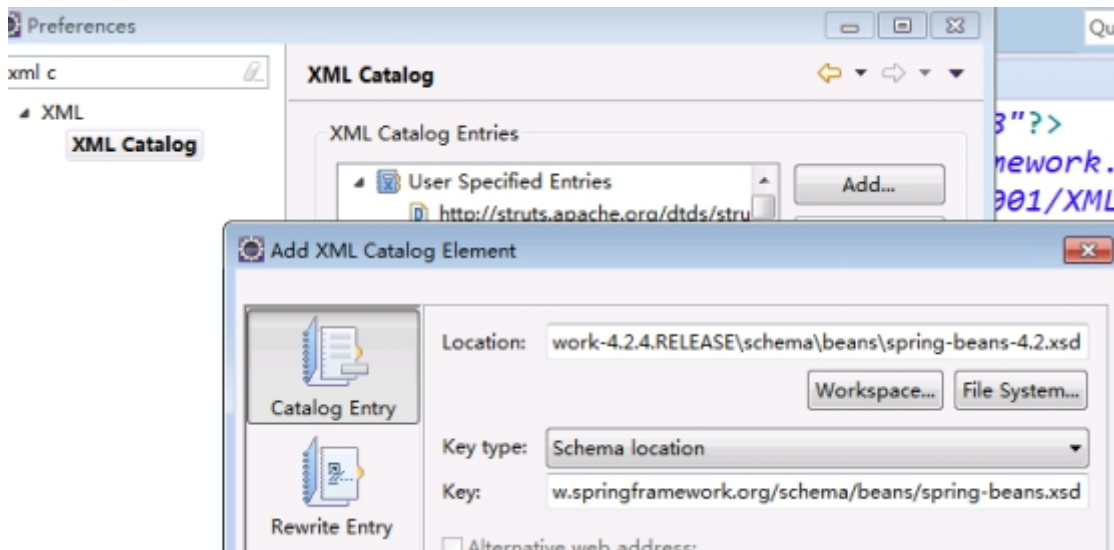
```
//1 加载spring配置文件，根据创建对象
ApplicationContext context =
    new ClassPathXmlApplicationContext("bean1.xml");
//2 得到配置创建的对象
User user = (User) context.getBean("user");
System.out.println(user);
user.add();
```

5. 配置文件没有提示问题

1 spring引入schema约束，把约束文件引入到eclipse中

(1) 复制约束路径

<http://www.springframework.org/schema/beans/spring-beans.xsd>



6. Spring的bean管理 (xml方式)

6-1. Bean实例化的方式

1 在spring里面通过配置文件创建对象

2 bean实例化三种方式实现

第一种 使用类的无参数构造创建 (重点)

```
<!-- ioc入门 -->
<bean id="user" class="cn.itcast.ioc.User"></bean>
```

类里面没有无参数的构造，出现异常

(3) name属性：功能和id属性一样的，id属性值不能包含特殊符号，但是在name属性值

里面可以包含特殊符号

(4) scope属性

- singleton：默认值，单例

```
scope="singleton">
```

```
cn.itcast.ioc.User@1be3a66
```

```
cn.itcast.ioc.User@1be3a66
```

- prototype：多例(Struts把Action类的创建交给Spring要用到多实例)

```
cn.itcast.ioc.User@e90eef
```

```
cn.itcast.ioc.User@d381e4
```

- request：创建对象把对象放到request域里面

- session：创建对象把对象放到session域里面

- globalSession：创建对象把对象放到globalSession里面

6-3. 属性注入介绍

1 创建对象时候，向类里面属性里面设置值

2 属性注入的方式介绍（三种方式）

(1) 使用set方法注入

(2) 使用有参数构造注入

(3) 使用接口注入

第一种 使用set方法注入

```
public class User {  
    private String name;  
    public void setName(String name) {  
        this.name = name;  
    }  
}  
  
User user = new User();  
user.setName("abcd");
```

第二种 有参数构造注入

```
public class User {  
    private String name;  
    public User(String name) {  
        this.name = name;  
    }  
}  
  
User user = new User("lucy");
```

第三种 使用接口注入

```
public interface Dao {  
    public void delete(String name);  
}  
  
public class DaoImpl implements Dao {  
    private String name;  
    public void delete(String name) {  
        this.name = name;  
    }  
}
```

3 在spring框架里面，支持前两种方式

(1) set方法注入 (重点)

(2) 有参数构造注入

6-4. 使用有参数构造注入属性

```
<!-- 使用有参数构造注入属性 -->
<bean id="demo" class="cn.itcast.property.PropertyDemo1">
    <!-- 使用有参数构造注入 -->
    <constructor-arg name="username" value="小王小马"></constructor-arg>
</bean>
```

```
private String username;

public PropertyDemo1(String username) {
    this.username = username;
}

public void test1() {
    System.out.println("demo1....."+username);
}
```

6-5. 使用set方法注入属性 (重点)

```
private String bookname;
//set方法
public void setBookname(String bookname) {
    this.bookname = bookname;
}
```

```
<!-- 使用set方法注入属性 -->
<bean id="book" class="cn.itcast.property.Book">
    <!-- 注入属性值
    name属性值: 类里面定义的属性名称
    value属性: 设置具体的值
    -->
    <property name="bookname" value="易经"></property>
</bean>
```

6-6. 注入对象类型属性 (重点)

1 创建service类和dao类

- (1) 在service得到dao对象
- 2 具体实现过程
 - (1) 在service里面把dao作为类型属性
 - (2) 生成dao类型属性的set方法

```
public class UserService {  
  
    //1 定义dao类型属性  
    private UserDao userDao;  
    //2 生成set方法  
    public void setUserDao(UserDao userDao) {  
        this.userDao = userDao;  
    }  
}
```

- (3) 配置文件中注入关系

```
<!-- 1配置service和dao对象 -->  
<bean id="userDao" class="cn.itcast.ioc.UserDao"></bean>  
  
<bean id="userService" class="cn.itcast.ioc.UserService">  
    <!-- 注入dao对象  
        name属性值: service类里面属性名称  
        现在不要写value属性, 因为刚才是字符串, 现在是对象  
        写ref属性: dao配置bean标签中id值  
    -->  
    <property name="userDao" ref="userDao"></property>  
</bean>
```

6-7. P名称空间注入

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

```
<!-- p名称空间注入 -->  
<bean id="person" class="cn.itcast.property.Person" p:pname="Lucy">  
    ...  
</bean>
```

6-8. 注入复杂类型属性

- 1 数组
- 2 list集合
- 3 map集合
- 4 properties类型


```

15     private String[] arrs;
16     private List<String> list;
17     private Map<String,String> map;
18     private Properties properties;
19
20     public void setArrs(String[] arrs) {
21         this.arrs = arrs;
22     }
23     public void setList(List<String> list) {
24         this.list = list;
25     }
26     public void setMap(Map<String, String> map) {
27         this.map = map;
28     }
29     public void setProperties(Properties properties) {
30         this.properties = properties;
31     }

```

<!-- 注入复杂类型属性值 -->

<bean id= "person" class= "cn.itcast.property.Person">

<!-- 数组 -->

<property name= "arrs">

<list>

<value>小王</value>

<value>小马</value>

<value>小宋</value>

</list>

</property>

<!-- list -->

<property name= "list">

<list>

<value>小奥</value>

<value>小金</value>

<value>小普</value>

</list>

</property>

<!-- map -->

<property name= "map">

<map>

<entry key= "aa" value= "lucy"></entry>

<entry key= "bb" value= "mary"></entry>

```

        <entry key= "cc" value= "tom"></entry>
    </map>
</property>
<!-- properties -->
<property name= "properties">
    <props>
        <prop key= "driverclass">com.mysql.jdbc.Driver</prop>
        <prop key= "username">root</prop>
    </props>
</property>
</bean>

```

7. IOC和DI区别

- (1) IOC: (Inversion of Control) 控制反转，把对象创建交给spring进行配置
- (2) DI: (Dependency Injection) 依赖注入，向类里面的属性中设置值
- (3) 关系：依赖注入不能单独存在，需要在ioc基础之上完成操作

8. Spring整合web项目原理

1 加载spring核心配置文件

```

//1 加载spring配置文件，根据创建对象
ApplicationContext context =
    new ClassPathXmlApplicationContext("bean1.xml");

```

- (1) new对象，功能可以实现，效率很低

2 实现思想：把加载配置文件和创建对象过程，在服务器启动时候完成

3 实现原理

- (1) ServletContext对象

- (2) 监听器

- (3) 具体使用：

- 在服务器启动时候，为每个项目创建一个ServletContext对象
- 在ServletContext对象创建时候，使用监听器可以具体到ServletContext对象在什么时候创建

- 使用监听器监听到ServletContext对象创建时候，加载spring配置文件，把配置文件bean对象创建
- 把创建出来的对象放到ServletContext域对象里面（setAttribute方法）
- 获取对象时候，到ServletContext域得到（getAttribute方法）