spring 概念

spring是开源轻量级框架

spring核心：aop，ioc

aop:面向切面编程，扩展功能而不是修改源代码实习那个。

ioc:控制反转

比如有一个类，类里面有方法（不是静态的），调用类里面的方法，创建累的对象，使用对象调用的方法，创建对象的过程，需要new出来。

而spring将对象的创建不是通过new来实现，而是交给spring配置创建类对象。

spring是一个一站式的框架框架

spring在Javaee三层结构中，每一层提供了不同的技术

web层：springMVC

service层：spring的ioc

dao层：spring的jdbcTemplate

Spring的ioc操作

1.把对象的创建交给spring进行管理

2.ioc操作两个部分

ioc的配置文件方式

ioc的注解方式

ioc地层原理

ioc地层实用的技术

xml配置文件

dom4j解决xml

工厂设计模式

反射

ioc底层实现

```
public class user{
      public void add(){
            ......
      }
}

//servlet 调用User类里面的方法

User user = new User();
user.add();

*缺点耦合度太高
```

使用工厂模式解耦

```
public class UserService{
      public void add(){
            ...
      }
}

public class userServlet{
    UserService u =
}   Factory.getService()

//解决方法，创建工厂
public class Factory{
 //提供返回UserServlet对象的方法
 public static UserService getService(){
return new UserService();
}
```

servlet与工厂耦合

ioc原理

```
public class UserService{

}

public class UserServlet{
  //得到UserServlet对象
  //原始：new创建

}
```

第一步 创建xml配置文件，配置创建类对象

```
<bean id="userService" class="类路径" />
```

第二步 创建工厂类，使用dom4j解析配置文件+反射创建雷对象

射创建对象

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

降低类之间的耦合度

ioc入门案例

第一步 导入jar包

spring beans,spring centext, spring core,spring,spring-expression,commons-logging,log4j

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) {
  User u = new User();
  u.add();
 }
}
```

第三步 创建spring配置文件，配置创建类。

创建spring配置文件官网建议applicationCentext.xml(可以任意命名)。

引入约束：

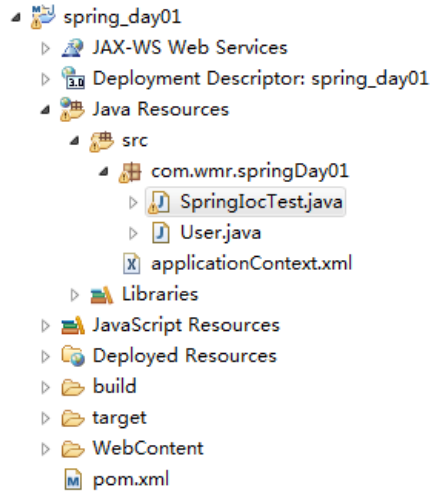
```
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="
      http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">

</beans>
```

配置对象：

```
<bean id="user" class="spring_dat01.User"></bean>
```

第四步 写代码测试对象创建

项目目录：



测试代码：

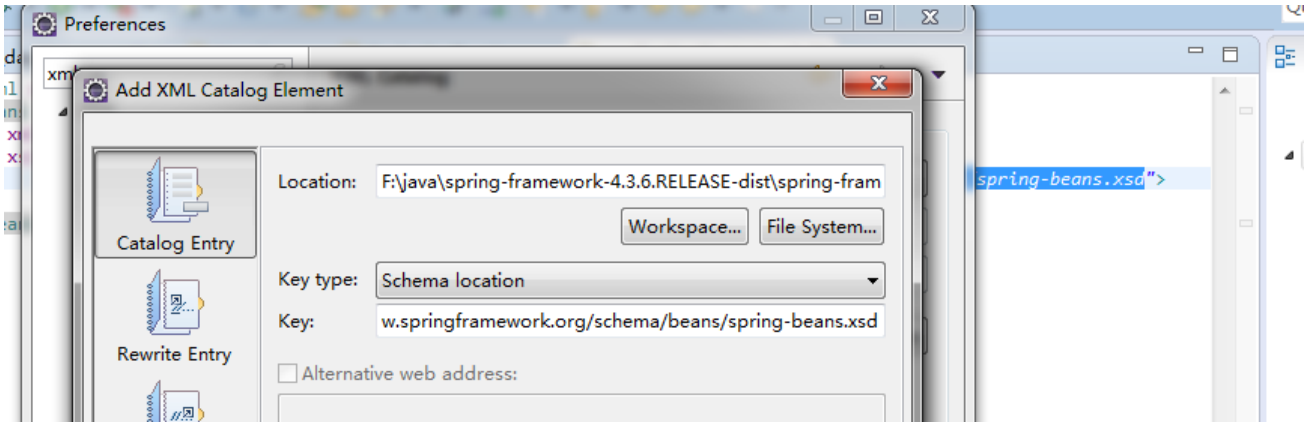
```
public class SpringIocTest {
 @Test
 public void test() {
  //加载spring配置文件
  ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

  //创建对象实例
  User user = (User) context.getBean("user");
  System.out.println(user);
  user.add();
 }
}
```

配置文件没有提示问题

1.spring引入schema约束，吧约束文件引入到eclipse中

复制schema类路径：http://www.springframework.org/schema/beans/spring-beans.xsd

在eclipse-> windows->perfereces->中搜索xml catalog中添加一个xml catelog entries



spring 的bean管理

1.spring bean的三种实例化方式

1）使用类的无参构造方式

```
public class User {
 private String name;
 public User() {
```

```
}
public User(String name){
 super();
 this.name = name;
}
public void add(){
 System.out.println("add.....!");
}
}
```

2）使用静态工厂创建（少用）

1）创建静态工厂，返回实例对象。

代码：

```
public class Bean2 {
 public void add(){
  System.out.println("bean2.....!");
 }
}
public class Bean2Factory {
 public static Bean2 getBean2(){
  return new Bean2();
 }
}
```

配置文件

```
<bean id="bean2" class="com.wmr.springDay01.beans.Bean2Factory" factory-method="getBean2"></bean>
```

测试

```
@Test
 public void bea2Test() {
  //加载spring配置文件
  ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

  //创建对象实例
  ((Bean2)context.getBean("bean2")).add();
 }
```

3）使用事例工厂创建（少用）

1）创建不是静态的方法，返回实例对象。

配置文件

```
<bean id="bean2Factory2" class="com.wmr.springDay01.beans.Bean2Factory2"></bean>
    <bean id="bean2" factory-bean="bean2Factory2" factory-method="getBean2"></bean>
```

实例工厂与静态工厂主要在spring配置文件的配置上不同实际应用相差不大。

bean标签属性

1. id属性

   id属性值，不能包含中文，特殊符号

   根据id值获取配置对象

2. class属性

   创建对象的路径

3. name属性

   功能和id相同，id属性值不能包含特殊符号，name中可以包含特殊符号

4. scope属性

bean的作用方位

singleton:单例的(默认值)

prototype：多实例的

request：创建对象放到request域中(不用)

session:创建对象放到session域中(不用)

globalSession:创建对象放到globalSession域中(不用)

属性注入

创建对象是可以向对象的属性设置值

属性注入的三种方式

1）set方式注入（重点）

```
<bean id="prop2" class="com.wmr.springDay01.beans.ProptitiesTest2">
 <!-- 注入属性值 -->
 <property name="name" value="prop2Name"/>
</bean>
```

2）有参构造注入

配置文件

```
<!-- 有参构造 -->
<bean id="prop1" class="com.wmr.springDay01.beans.ProptitiesTest">
 <constructor-arg name="name" value="test1111"></constructor-arg>
</bean>
```

spring中只支持以上两种注入方式。

已下是Java中属性赋值的方法

第一种实行set方式注入

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

User user = new User();
user.setName("");
```

第二种有参构造注入

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

第三种使用接口注入

```
public interface Dao{
      public void delete(String name);
}

public class DaoImp implements Dao{
      priate String name;
      pulic void delete(String name);
this.name;
}
```

注入对象类型属性（重点）

1）创建service类和dao类

将dao类当成一个service类的属性

配置文件：

```
<!-- 注入对象类型的属性 -->
<!-- 配置service和dao的对象 -->
<bean id="dao" class="com.wmr.springDay01.beans.UserDao"></bean>
<bean id="userService" class="com.wmr.springDay01.beans.UserService">
 <!-- 注入dao对象 -->
 <!-- name属性值：service类里面的属性名
 ref值：dao配置bean标签中的id
 -->
 <property name="dao" ref="dao"></property>
</bean>
```

P名称空间注入

```
xmlns:p="http://www.springframework.org/schema/p"
<!-- p空间属性注入 -->
    <bean id="p" class="com.wmr.springDay01.beans.Person" p:pname="personName"></bean>
```

复杂数据注入：

数组，集合，Map,Properties等

```
<!-- 复杂类型属性值注入 -->
    <bean id="p2" class="com.wmr.springDay01.beans.Person2">
     <!-- 数组 -->
     <property name="argss">
      <list>
       <value>aa</value>
       <value>ab</value>
       <value>ac</value>
      </list>
     </property>
     <!-- list -->
     <property name="lit">
      <list>
       <value>la</value>
       <value>lb</value>
       <value>lc</value>
      </list>
     </property>
     <!-- map -->
     <property name="map">
      <map>
       <entry key="ma" value="ma"></entry>
       <entry key="mb" value="mb"></entry>
       <entry key="mc" value="mc"></entry>
      </map>
     </property>
     <property name="props">
      <props>
       <prop key="driverclass">com.mysql.jdbc.Driver</prop>
       <prop key="username">root</prop>
      </props>
     </property>
    </bean>
```

IOC和DI的区别：

1）IOC控制反转

2）DI依赖注入，向类里面的属性设置值

3）关系：依赖注入，不能单独存在，需要在IOC的基础上完成

spring整合web项目原理

1.加载spring核心配置文件

ApplicationContext context = new ClassPathApplicationContext("applicationContext.cml");

1).new 对象，功能可以实现，但性能上耗损太大效率低。

2.是想思想，把加载配置文件对象的过程，在服务器启动时完成。

3.实现原理

1）ServletContext对象

2）监听器

3）具体使用

在服务器启动的时候，会为每个项目创建servletContext对象，在servletContext被创建的时候，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对象创建的时候，健在spring配置文件，把配置文件的对象创建

把创建出来的对象放到servletContext域对象里卖弄（setAttribute）

获取对象的时候，到servletContext域得到（getAtrribute方法）