# Spring\_day01笔记

## 上次课内容回顾

* Struts2的拦截器
  + Struts2的拦截器概念
    - 拦截器：拦截对Action的访问，拦截到Action的具体的方法。
  + Struts2的执行流程
    - 请求-🡪核心过滤器🡪创建ActionProxy，调用proxy.execute方法。在这个内部🡪ActionInvocation.invoke()在这个方法内部，递归执行一组拦截器🡪Action🡪Result🡪拦截器后面的代码
  + Struts2的拦截器
  + CRM的登录案例：
    - 权限拦截器
* Struts2的标签库
  + 通用标签
    - if、elseif、else、iterator、property、date、debug
  + UI标签（数据回显）
    - 表单标签：

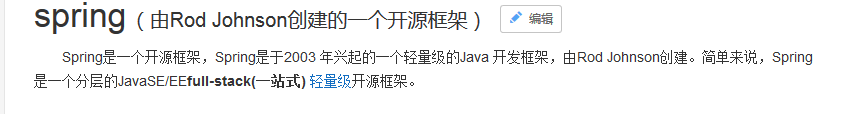
## Spring4学习路线

* Spring第一天：Spring的概述、SpringIOC入门（XML）、Spring的Bean管理、Spring属性注入
* Spring第二天：Spring的IOC的注解方式、Spring的AOP开发（XML）
* Spring第三天：Spring的AOP的注解开发、Spring的声明式事务、JdbcTemplate。
* Spring第四天：SSH的整合、HibernateTemplate的使用、OpenSessionInViewFilter的使用。

## Spring的概述

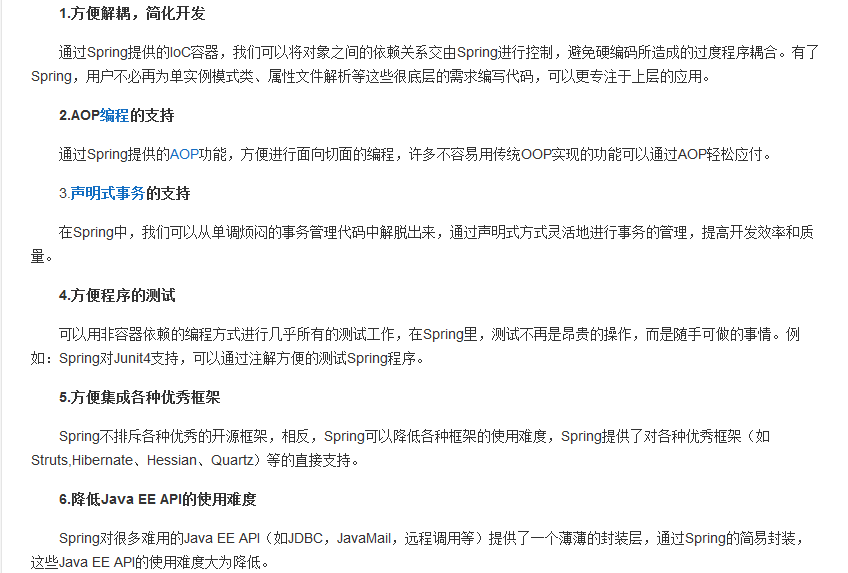
### Spring的概述

#### 什么是Spring



* Spring：SE/EE开发的**一站式**框架。
  + 一站式框架：有EE开发的每一层解决方案。
    - WEB层 ：SpringMVC
    - Service层 ：Spring的Bean管理，Spring声明式事务
    - DAO层 ：Spring的Jdbc模板，Spring的ORM模块

#### 为什么学习Spring



#### Spring的版本

Spring3.x和Spring4.x

### Spring的入门（IOC）

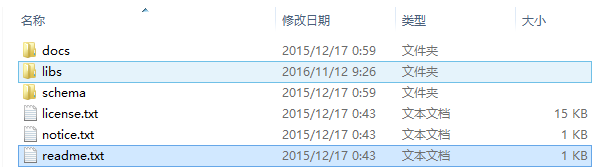
#### 什么IOC

* IOC: Inversion of Control(控制反转)。
  + 控制反转：将对象的创建权反转给（交给）Spring。

#### 下载Spring的开发包

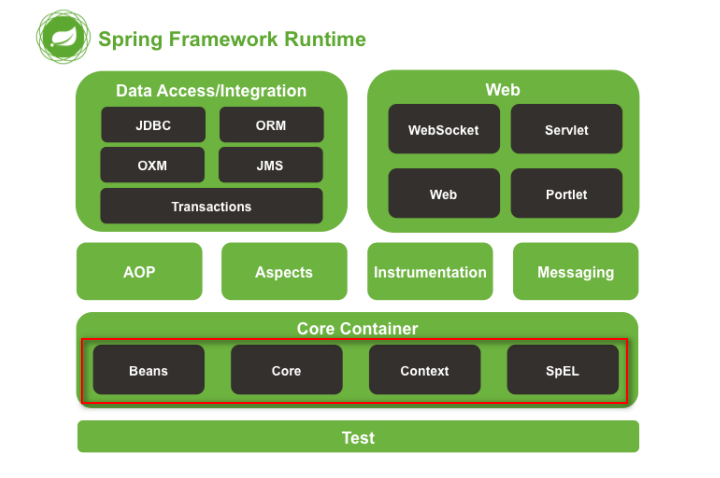
官网：<http://spring.io/>

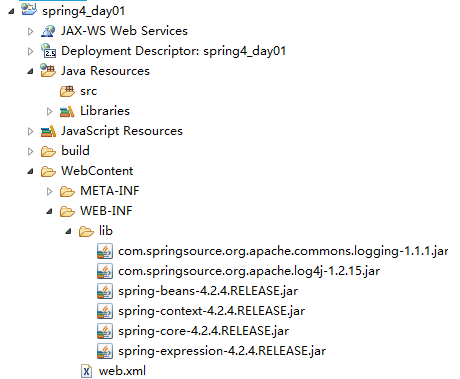
#### 解压Spring的开发包



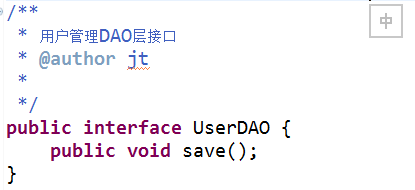
* docs ：Spring的开发规范和API
* libs ：Spring的开发的jar和源码
* schema ：Spring的配置文件的约束

#### 创建web项目，引入jar包



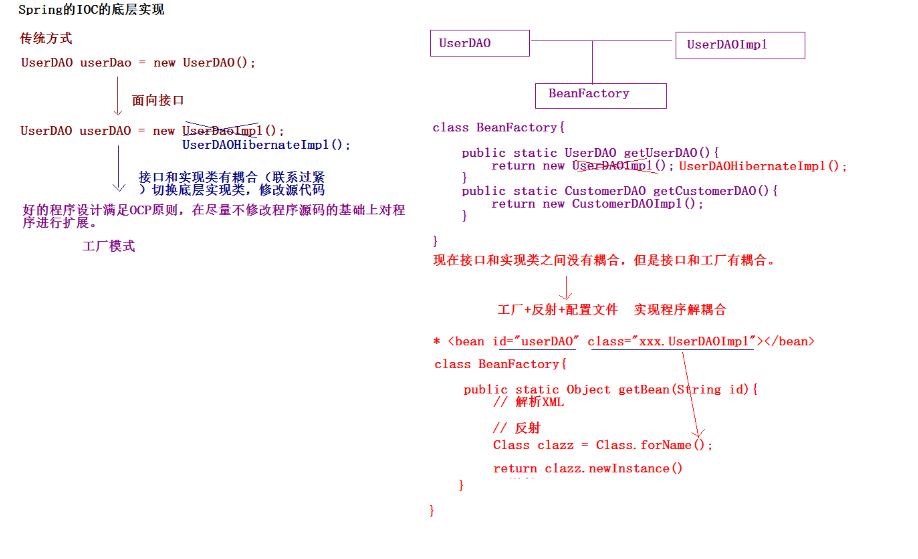


#### 创建接口和类



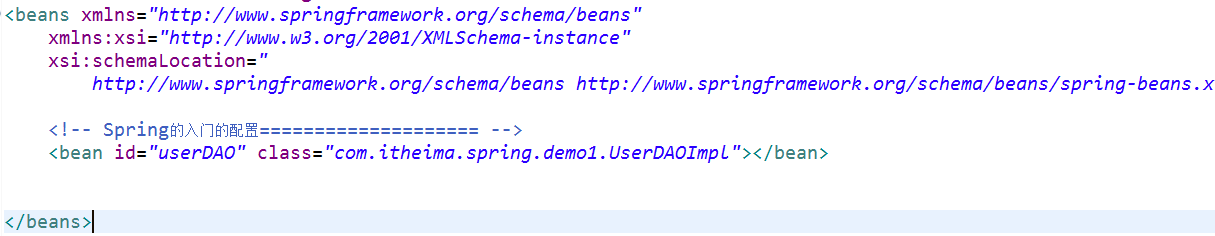


* 问题：
  + 如果底层的实现切换了，需要修改源代码，能不能不修改程序源代码对程序进行扩展？

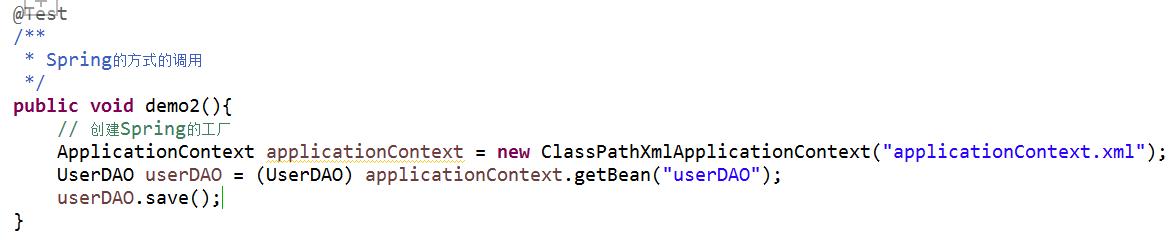


#### 将实现类交给Spring管理

在spring的解压路径下spring-framework-4.2.4.RELEASE\docs\spring-framework-reference\html\xsd-configuration.html



#### 编写测试类



#### IOC和DI（\*\*\*\*\*）

* IOC：控制反转，将对象的创建权反转给了Spring。
  + DI：依赖注入，前提必须有IOC的环境，Spring管理这个类的时候将类的依赖的属性注入（设置）进来。
* 面向对象的时候
  + 依赖

Class A{

}

Class B{

public void xxx(A a){

}

}

* + 继承:is a

Class A{

}

Class B extends A{

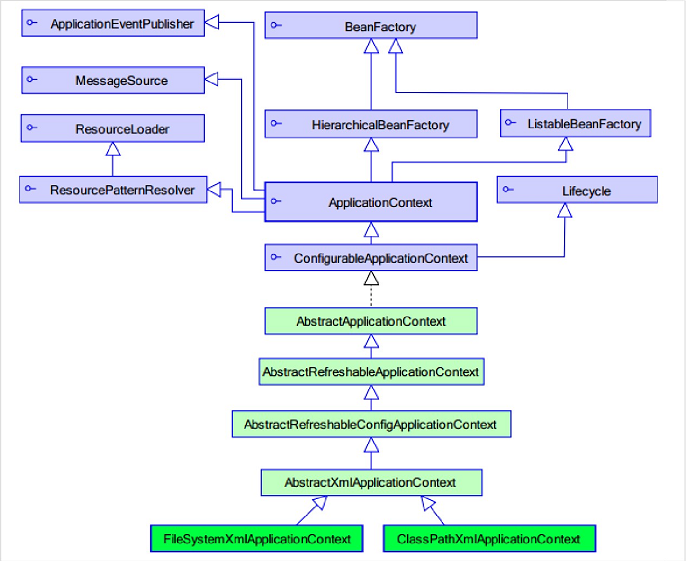
}

* + 聚合:has a

## Spring的工厂类

### Spring的工厂类

#### Spring工厂类的结构图



* ApplicationContext继承BeanFactory。

#### BeanFactory ：老版本的工厂类

* BeanFactory：调用getBean的时候，才会生成类的实例。

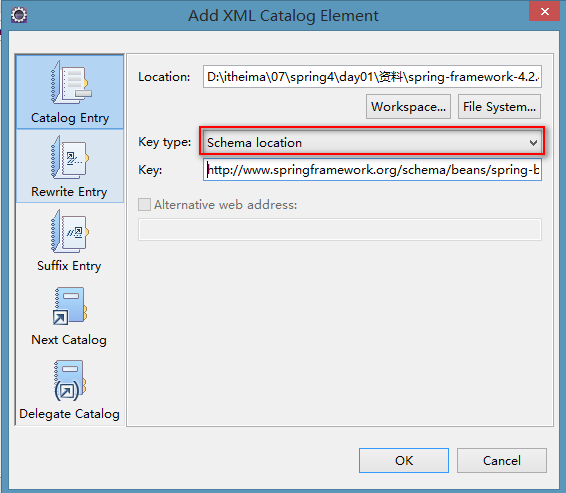
#### ApplicationContext ：新版本的工厂类

* ApplicationContext：加载配置文件的时候，就会将Spring管理的类都实例化。
* ApplicationContext有两个实现类
  + **ClassPathXmlApplicationContext ：加载类路径下的配置文件**
  + FileSystemXmlApplicationContext ：加载文件系统下的配置文件

## Spring的配置

### XML的提示配置

#### Schema的配置



### Bean的相关的配置

#### <bean>标签的id和name的配置

* id :使用了约束中的唯一约束。里面不能出现特殊字符的。
* name :没有使用约束中的唯一约束（理论上可以出现重复的，但是实际开发不能出现的）。里面可以出现特殊字符。
  + Spring和Struts1框架整合的时候
  + <bean name=”/user” class=””/>

#### Bean的生命周期的配置（了解）

* init-method :Bean被初始化的时候执行的方法
* destroy-method :Bean被销毁的时候执行的方法（Bean是单例创建，工厂关闭）

#### Bean的作用范围的配置（重点）

* scope ：Bean的作用范围
  + **singleton ：默认的，Spring会采用单例模式创建这个对象。**
  + **prototype ：多例模式。（Struts2和Spring整合一定会用到）**
  + request ：应用在web项目中，Spring创建这个类以后，将这个类存入到request范围中。
  + session ：应用在web项目中，Spring创建这个类以后，将这个类存入到session范围中。
  + globalsession ：应用在web项目中，必须在porlet环境下使用。但是如果没有这种环境，相对于session。

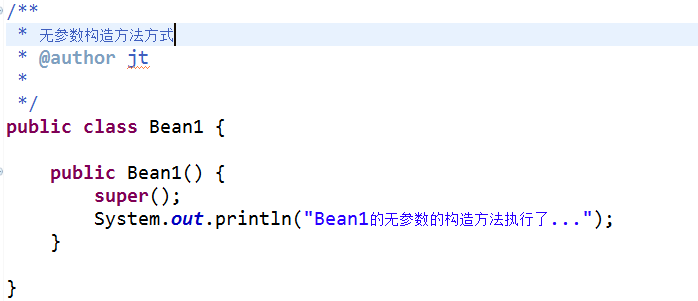
## Spring的Bean管理（XML方式）

### Spring的Bean的实例化方式（了解）

Bean已经都交给Spring管理，Spring创建这些类的时候，有几种方式：

#### 无参构造方法的方式（默认）

* 编写类

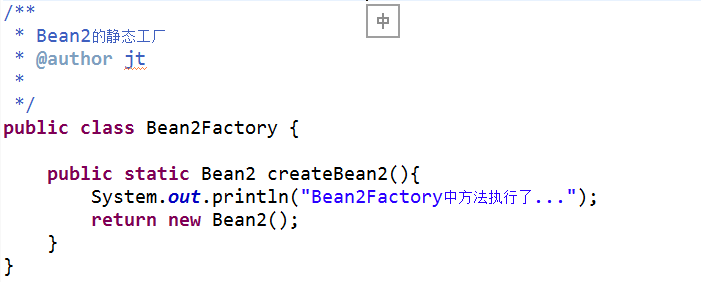


* 编写配置



#### 静态工厂实例化的方式

* 编写Bean2的静态工厂



* 配置

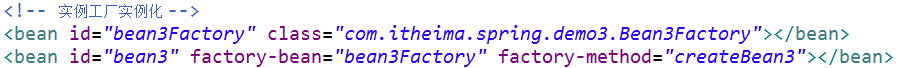


#### 实例工厂实例化的方式

* Bean3的实例工厂



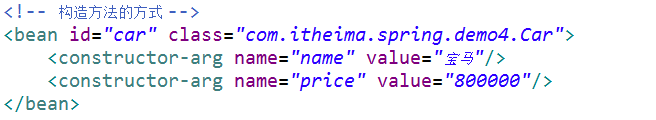
* 配置



### Spring的属性注入

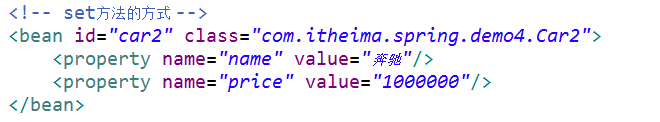
#### 构造方法的方式的属性注入

* 构造方法的属性注入

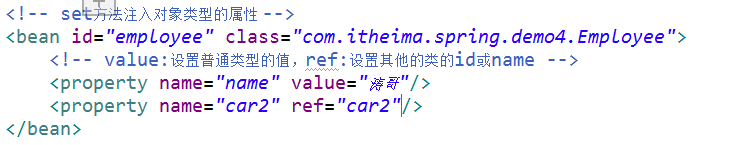


#### Set方法的方式的属性注入

* Set方法的属性注入

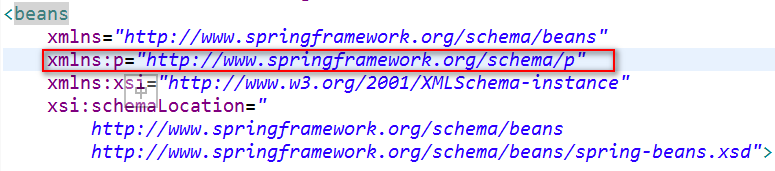


* Set方法设置对象类型的属性

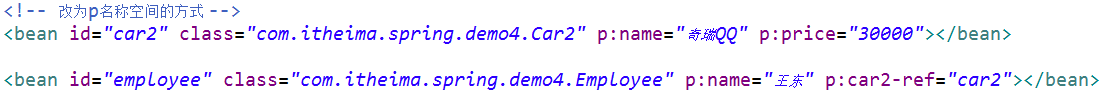


#### P名称空间的属性注入（Spring2.5以后）

* 通过引入p名称空间完成属性的注入：
  + 写法：
    - 普通属性 p:属性名=”值”
    - 对象属性 p:属性名-ref=”值”
* P名称空间的引入

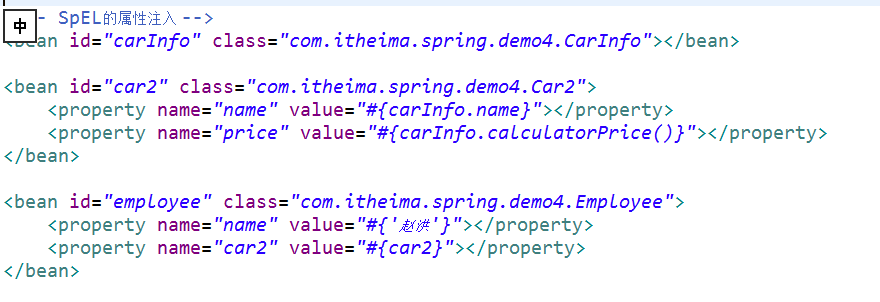


* 使用p名称空间



#### SpEL的属性注入（Spring3.0以后）

* SpEL：Spring Expression Language，Spring的表达式语言。
  + 语法：
    - #{SpEL}



### 集合类型属性注入(了解)

#### 配置

<!-- Spring的集合属性的注入============================ -->

<!-- 注入数组类型 -->

<bean id=*"collectionBean"* class=*"com.itheima.spring.demo5.CollectionBean"*>

<!-- 数组类型 -->

<property name=*"arrs"*>

<list>

<value>王东</value>

<value>赵洪</value>

<value>李冠希</value>

</list>

</property>

<!-- 注入list集合 -->

<property name=*"list"*>

<list>

<value>李兵</value>

<value>赵如何</value>

<value>邓凤</value>

</list>

</property>

<!-- 注入set集合 -->

<property name=*"set"*>

<set>

<value>aaa</value>

<value>bbb</value>

<value>ccc</value>

</set>

</property>

<!-- 注入Map集合 -->

<property name=*"map"*>

<map>

<entry key=*"aaa"* value=*"111"*/>

<entry key=*"bbb"* value=*"222"*/>

<entry key=*"ccc"* value=*"333"*/>

</map>

</property>

</bean>

## Spring的分模块开发的配置

### 分模块配置

#### 在加载配置文件的时候，加载多个



#### 在一个配置文件中引入多个配置文件



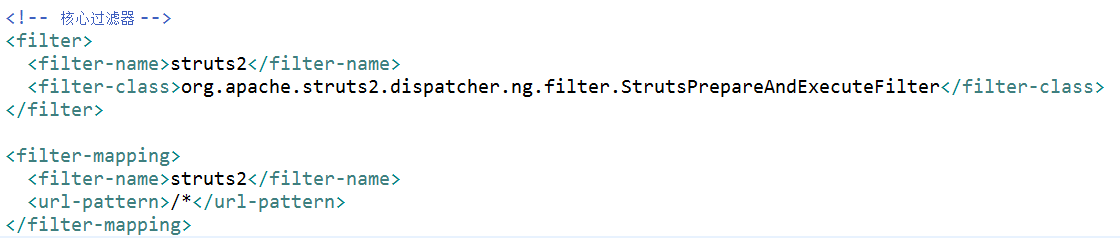
## CRM的综合案例

### 代码实现

#### 创建数据库和表

#### 创建web项目，引入jar包

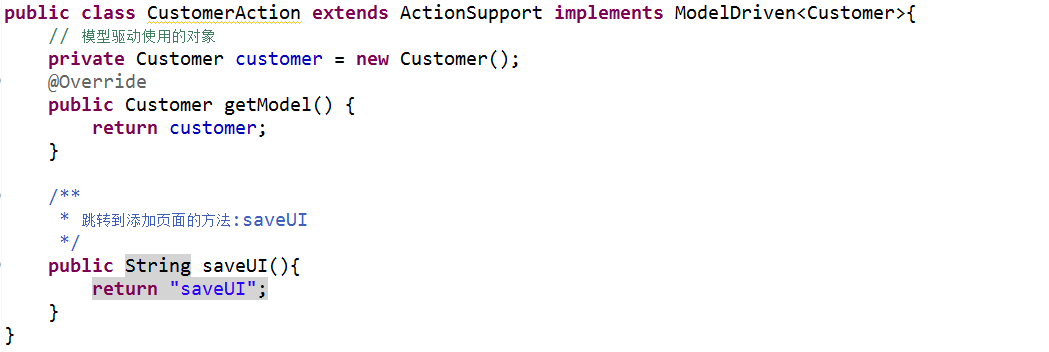
* 引入struts2的开发的jar包
* 引入struts2的配置文件
  + web.xml



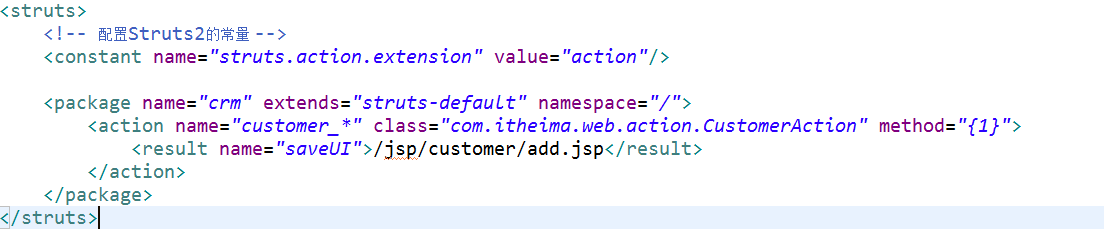
* + struts.xml

#### 引入页面

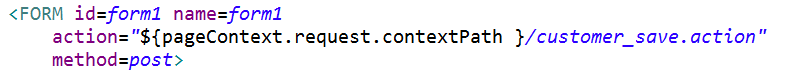
#### 编写Action类



#### 配置Action

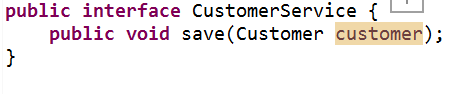


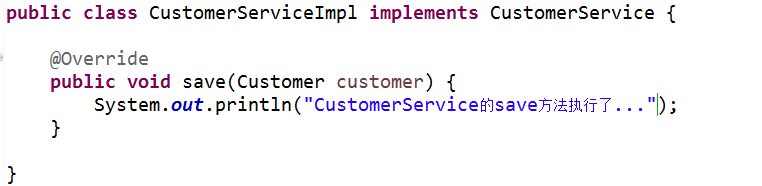
#### 修改页面提交到Action



#### 编写Action的save方法

* 不在Action中直接创建Service，将Service交给Spring管理。
* 创建Service接口和实现类

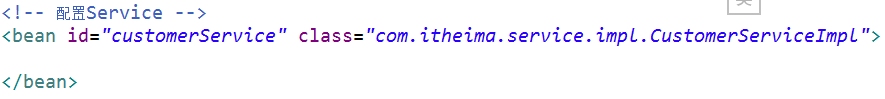




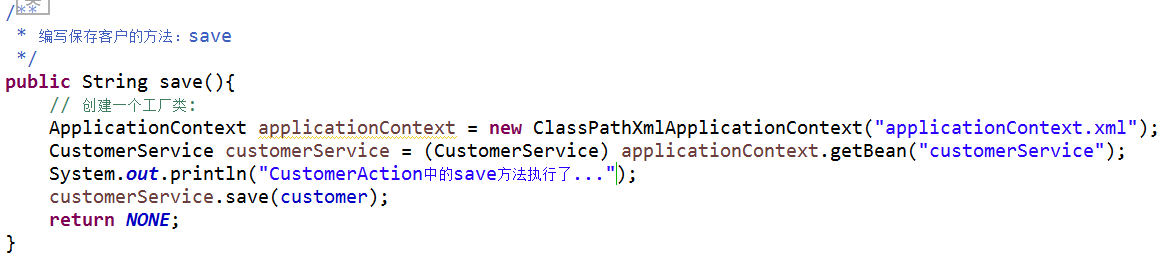
#### 引入Spring的环境

* 引入jar包
* 引入配置文件

#### 将Service交给Spring

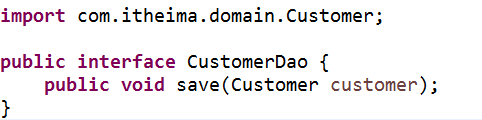


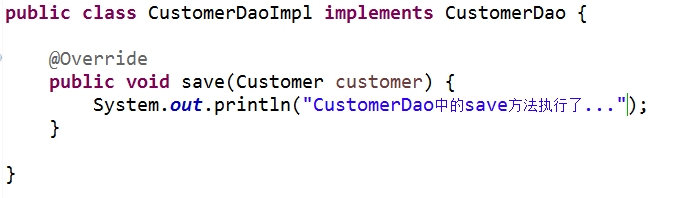
#### 在Action中调用Service



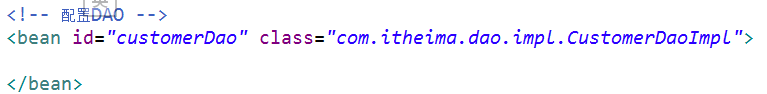
#### 编写DAO并且完成配置

* 编写DAO的接口和实现类

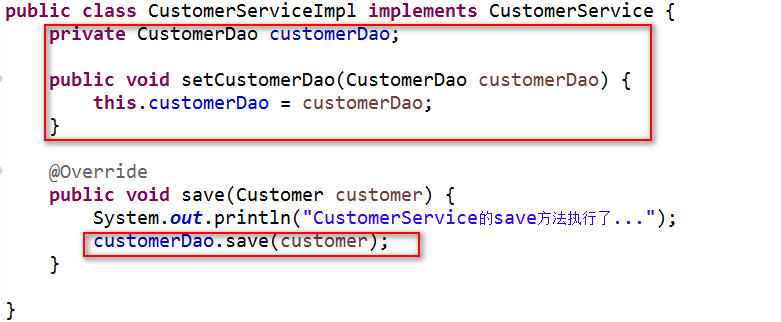


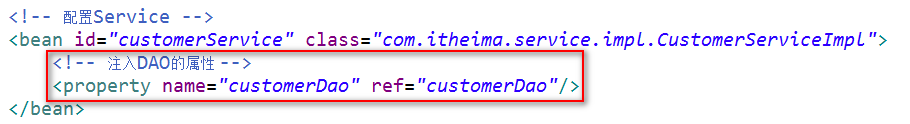


* 将DAO交给Spring管理



#### 在Service中使用DAO





### 问题描述

#### 程序问题

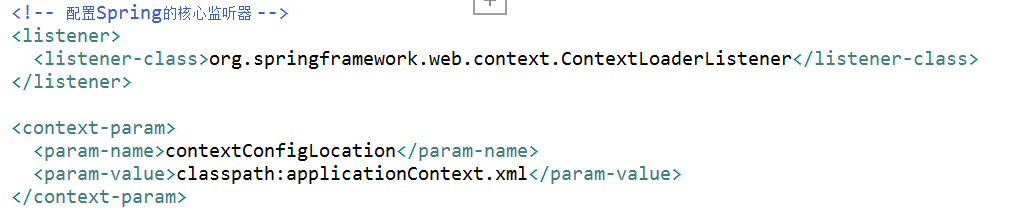
* 每次请求都会创建一个Spring的工厂，这样浪费服务器资源，应该一个项目只有一个Spring的工厂。
  + 在服务器启动的时候，创建一个Spring的工厂。
  + 创建完工厂，将这个工厂类保存到ServletContext中。
  + 每次使用的时候都从ServletContext中获取。
* **\*\*\*\*\*使用ServletContextListener**
  + **监听ServletContext对象的创建和销毁。**

#### 解决方案：使用Spring核心监听器ContextLoaderListener（整合web项目）

* 引入jar包

spring-web.jar

* 配置监听器



* 在Action中获取工厂

