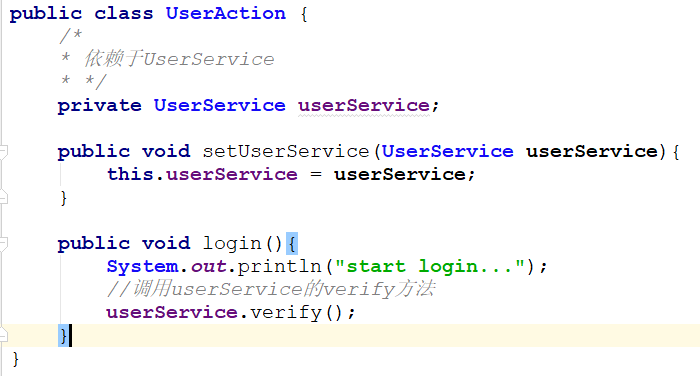
Spring框架最核心的就是容器这个概念。Spring有一个对象叫ApplicationContext，即应用上下文。它本质上就是一个容器，这个容器装配着所有的对象，意思就是容器会自动帮我们生成这些对象，并添加到容器中，我们需要哪个对象直接去容器中根据对象id获取就可以了。我们使用一个对象时不再是自己去用A a = new A()这种方式了，而是通过A a = ApplicationContext.getBean(“beanName”);这种方式来获取

先来看Spring的这个容器是如何装配对象的。在Spring中，一个对象叫一个Bean，为什么叫Bean可能是因为Caffee Bean吧。

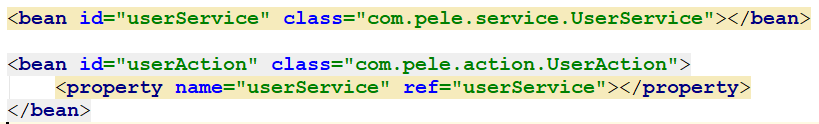
Spring最初装配Bean的方式就是通过解析xml配置文件来完成的，即我们在xml文件中配置我们所需要的Bean，然后Spring就可以解析xml并生成这些Bean了。先看Spring的简单的xml配置模板。

先假设我们现在定义了两个对象。两个对象关系如下：

UserAction对象和UserService对象。而UserAction依赖于UserService，它在login()这个方法里面需要用到UserService对象的方法。依赖注入(DI)的依赖就是指的这个依赖关系。



然后我们使用Spring容器来完成这两个Bean的装配。而不是自己去new 这两个对象。Xml关键配置如下:



配置比较容易理解，在userAction这个Bean中使用<property>这个标签来表示有一个属性是userService，引用了上面配置的userService。

然后就可以通过Spring解析这个xml文件来装配Bean了。简单的主程序如下：

假设这个xml命名为beans.xml文件

**public** s**tatic void** main(String[] args){

//解析beans.xml文件，创建一个容器，该容器会自动组装xml配置的那些bean

ApplicationContext apc = **new** ClassPathXmlApplicationContext("beans.xml");

//通过id获取对应的bean

UserAction action=(UserAction) apc.getBean("userAction");

//调用UserAction的login方法

action.login();

}

这样最基本的Spring的使用就完成了。在这个过程中，其实我们已经使用IOC(换做DI这个说法更容易理解)的功能。UserAction依赖于UserService，以前我们会自己在UserAction中去new一个Userservice对象，然后使用它。但是现在，我们把它们都交给了Spring容器，Spring容器会在生成UserAction这个对象时，将UserAction依赖的UserService对象自动注入到UserAction中，这就是所谓的依赖注入（DI）。