**Spring学习**

Ioc和DI的概念：

将原有的项目中手动通过new关键字创建对象权利交给spring，由工厂去创建对象，spring框架不仅要创建对象还要在创建对象的同时维护类与类之间的依赖关系（组建于组建的依赖关系），以此在IOC的基础上引出了DI的概念。

Spring工厂的细节：

<bean id=”UserDao” class=”com.UserDaoImpl”></bean>

中的class底层写法为：

Class<?> aclass= Class.forname(“com.UserDaoImpl”);

prodectDao o=aclass.newInstance();

o.save();

System.out.println(o);

1. 面向接口编程可以进行多态和程序的扩展

缺点：接口和实现类有解耦合（联系过紧），修改源代码。

UserDaoImpl

UserDao

UserDao userDao=new UserDaoImpl();

//如果需要调用实现类如果是Hibernate实现

UserDao userDao=new UserDaoHibernateImpl();

//修改的源代码过大

1. 一个好的程序设计需要满足OCP原则，在尽量不修改程序源代码的基础上对程序进行扩展。
2. 因此 需要引出工厂模式

工厂模式中有: 工厂方法(Factory Method) [抽象工厂](https://baike.baidu.com/item/%E6%8A%BD%E8%B1%A1%E5%B7%A5%E5%8E%82)(Abstract Factory).

**简单工厂的优点/缺点：**

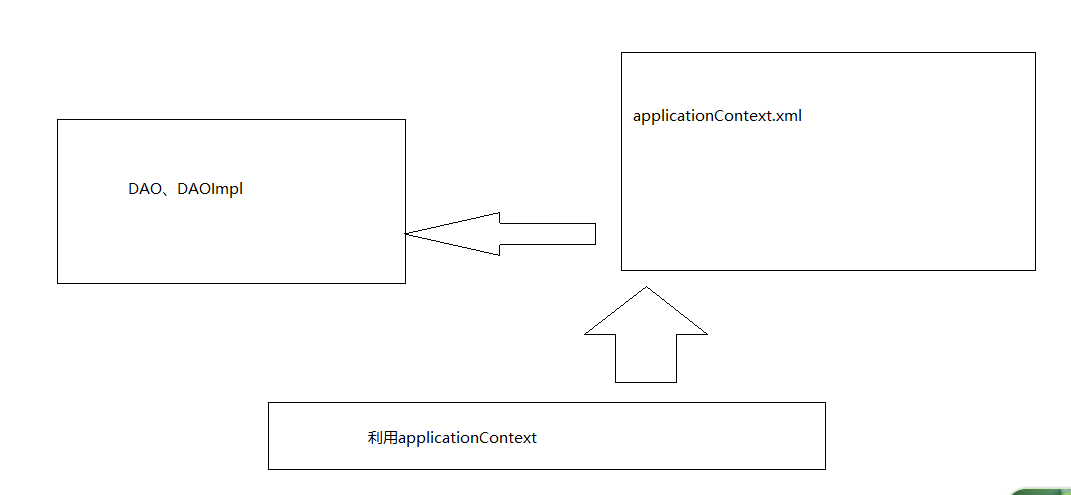
* 优点：简单工厂模式能够根据外界给定的信息，决定究竟应该创建哪个具体类的对象。明确区分了各自的职责和权力，有利于整个软件体系结构的优化。
* 缺点：很明显工厂类集中了所有实例的创建逻辑，容易违反GRASPR的高内聚的责任分配原则

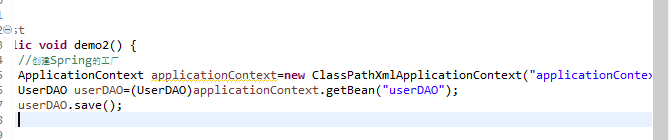
**抽象工厂的优点/缺点：**

* 优点：
  1. 抽象工厂模式隔离了具体类的生产，使得客户并不需要知道什么被创建。
  2. 当一个产品族中的多个对象被设计成一起工作时，它能保证客户端始终只使用同一个产品族中的对象。
  3. 增加新的具体工厂和产品族很方便，无须修改已有系统，符合“开闭原则”。
* 缺点：增加新的产品等级结构很复杂，需要修改抽象工厂和所有的具体工厂类，对“开闭原则”的支持呈现倾斜性。(不过说这个缺点好像有点吹毛求疵了)

现在接口和实现类之间没有耦合，但是接口和工厂有耦合。

工厂+反射+配置文件 实现程序解耦集合





#### <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。