可修改性：

当需求发生变更或者环境发生变化时，系统是否可以快速得到调整，就体现出系统的可修改性。好的系统要求可以方便快速修改。

系统的任何部分都是可修改的，需要关注的是修改的成本。由此可见可修改性战术的目标是控制实现、测试和部署变更的时间和成本。

可修改性可以分为三大类：

(1)局部化修改一目标是减少由某个变更直接影响的模块的数量

1、维持语义的一致性。2、预期期望的变更。

(2)防止连锁反应一一目标是限制对局部化的模块的修改,以防止对某个模块的修改间接地影响到其他模块

1、信息隐藏 。2、维持现有的接口。添加接口。 3、限制通信路径。4、使用仲裁者。

(3)延迟绑定时间一一目标是控制部署时间并允许非开发人员进行修改

1、运行时注册 。2、配置文件。3、多态。4、组件更换。 5、遵守已定义的协议

其中，无论是局部化修改还是防止连锁反应，都是基于“高内聚，低耦合”思想。那么根据软件设计模式的设计原则来分析论坛系统：

（1）局部性变更应用：主要表现在类、函数、方法和接口的时候，实现“高内聚，低耦合”。

我们采用了Struts+Spring+Hibernate集成框架来进行系统的开发，集成SSH框架的系统从职责上分为四层：表示层、业务逻辑层、数据持久层和域模块层，以帮助开发人员在短期内搭建结构清晰、可复用性好、维护方便的Web应用程序。其中使用Struts作为系统的整体基础架构，负责MVC的分离，在Struts框架的模型部分，控制业务跳转，利用Hibernate框架对持久层提供支持，Spring做管理，管理struts和hibernate。

论坛系统开发时设计类、函数、接口的过程中，秉持着“高内聚，低耦合”的思想。

（2）连锁反应定义：我们平时编程，无论是写函数还是写类，都需要被其他类还是函数调用，修改此函数或类就会影响到调用他的函数，这就是连锁反应。

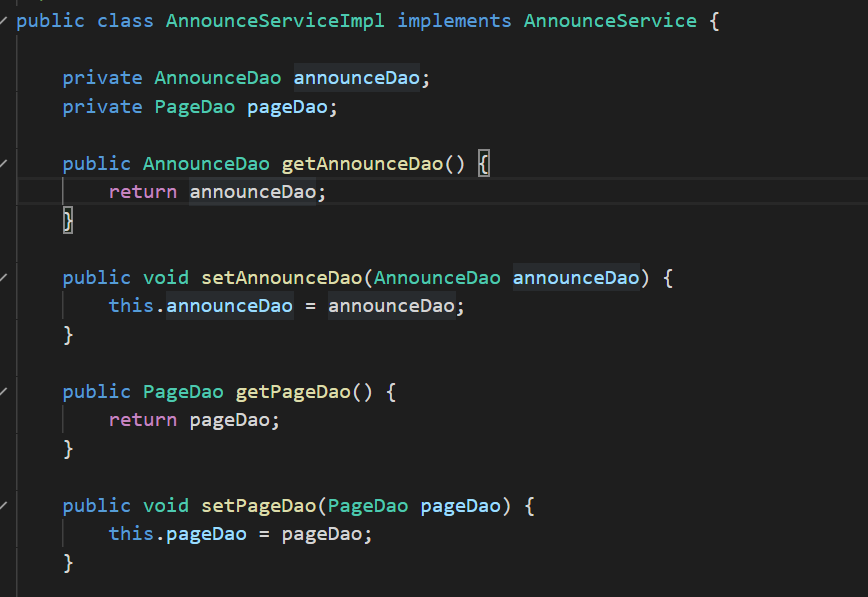
防止连锁反应应用：

1. 信息隐藏。信息隐藏就是把某个实体的责任分解为更小的部分，并选择使哪些信息成为公有的，哪些信息成为私有的。

在编程时，我们写类时，会开设私有成员，公有成员等。保证了信息隐藏。

例如：AnnounceServiceImpl具有私有成员变量，公有的成员函数。

public：可以被所有其他类所访问；private：只能被自己访问和修改



1. 维持现有的接口。该战术的模式包括添加接口、添加适配器、提供一个占位程序。

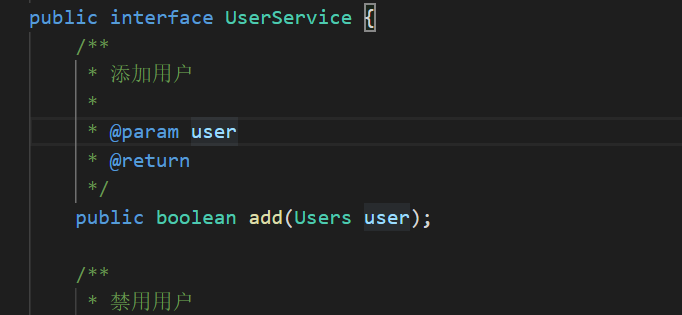
Java是面向对象的程序设计语言，所以我们经常要用到类，在编写程序时，我们采用了关键字**implements实现接口，进行接口的继承：**

关于接口的作用：接口是软件编程的规范，可以降低耦合性，换句话说，可以让某个模块或功能能够重复利用，这样只要写这功能一次代码就ok了。其他地方要用到的，全部用接口调用来实现

接口的优势：a .“接口+实现”最常见的优势就是实现类和接口分离，在更换实现类的时候，不用更换接口功能。b .这对于做单元测试也非常有帮助

例如该部分代码：

创建接口UserService：



类UserServiceImpl继承了接口UserService：



　3、限制通信路径。限制与一个给定的模块共享的模块，减少联系，一旦变更影响会小很多。

　4、使用仲裁者。插入仲裁者来管理依赖之间的关系. 数据库的使用，通过数据库来管理不同的数据信息。

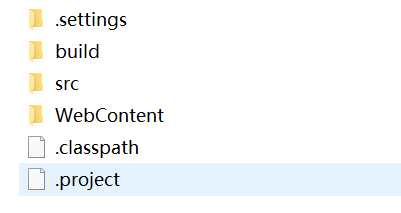
论坛系统通过mysql来管理不同的数据信息

（3）推迟绑定时间

　推迟绑定时间。将有可能的修改，尽量用配置文件，或者其他后期让非开发人员可调整的方式实现。

1、框架级可修改性：平台代码、应用代码、后台模块代码分离。

我们采用了Struts+Spring+Hibernate集成框架来进行系统的开发，优点是论坛系统前后端的代码是分离的，框架可修改性提高。



2、根据场景、调用后台模块出参入参可配原性修改，服务变更无需通过编码即可修改。

我们在做软件开发时，肯定会自觉的将需求抽象成软件的功能，这是毋庸置疑的，但是我们在开发前的架构设计阶段不单单要考虑软件的将要完成的功能，还要将软件的质量属性考虑进去，因为这将影响着一个软件的长期发展。