内大信件管理系统

GRASP 文档

指导教师：孟和吉雅
成员：王林 齐琦 张淑云 史泽弘

2018 年 6 月 5 日

一、 系统整体类图

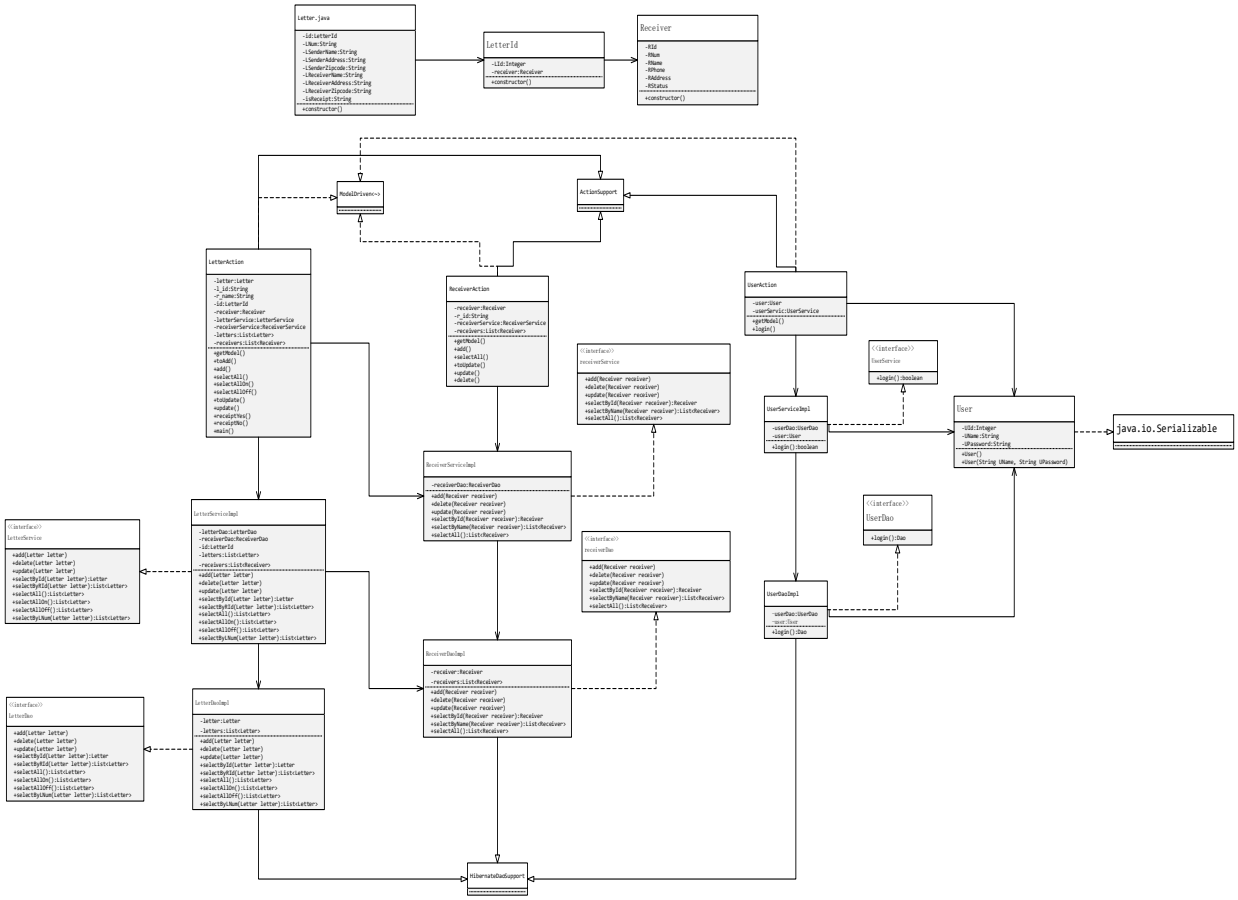

图 1. 整体类图

二、 GRASP 原则分析

1. 信息专家原则(information expert)

信息专家模式的本质指的是我们应该将职责委托给哪一个对象,这个职责可以是一个方法，也可以是一个算法或者其他内容。它是面向过程设计过程中最基本的原则。

在内大信件管理系统中，首先基于功能需求，我们设计了两个 until 类，共有两个

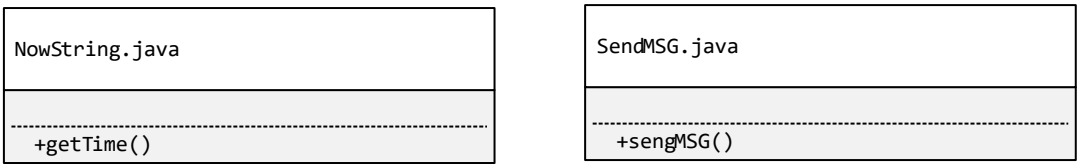

图 2.1.1　工具类图

其中，NowString 类负责获取当前时间，并将其转换为字符串输出，SendMSG 类负责发送信息到用户，该类调用了阿里云提供的三方包从而实现短信功能。

我们还基于信息专家原则，将数据库的每张表重新映射为一个个单独的实体类，

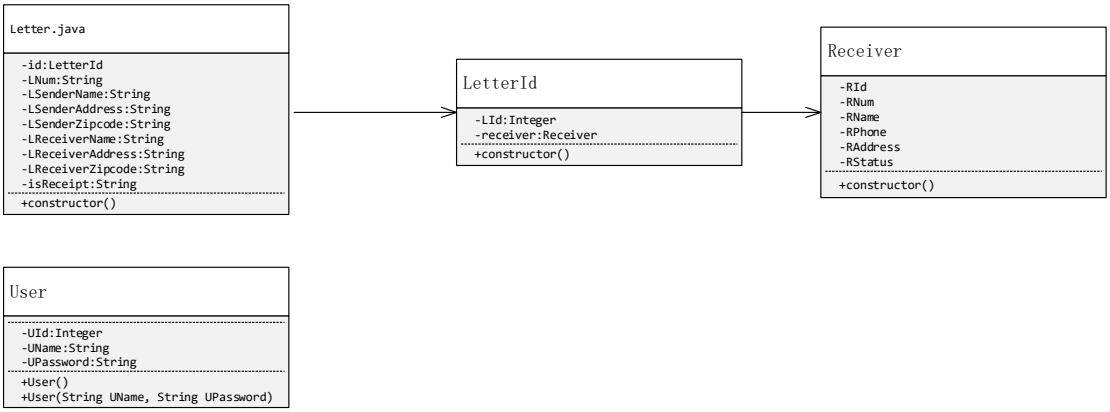

图 2.1.2 实体类图

其中，有实体类 Letter,User,Receiver，这三个对应数据库表 letter、user、receiver 三张。

2. 创造者原则(creator)

creator 原则的本质是创建类对象职责应该委托给那个对象，也就是谁应该负责产生某个类的实例。

对于建造者原则，它最核心的任务就是为程序提供众多的接口，程序中哪里需要用到了就在哪里去调用。在本系统中一共会在两个地方进行接口的声明，分别是数据交互层 Dao 和业务交互层 Service。

图 2.2.1 接口类图

主要类有 LetterService、ReceiverService、UserService 三个 service 层的接口，及 LetterDao、ReceiverDao、UserDao 三个 dao 层接口。接口具体提供方法及参数、返回值如图 2.2.1 所示。

3. 低耦合(Low coupling)

耦合是评价一个系统中各个元素之间连接或者依赖关系强弱的尺度。低耦合的原则是我们在设计系统的时候尽量降低系统中各个元素之间的耦合度，这样对于系统的理解和维护都有很大的益处。

对于低耦合，我们认为应该和高内聚一起来说，对于一个系统的设计即实现，高内聚低耦合始终贯穿整个设计及开发过程，在本系统中，我们采用的 mvc 三层架构就明显的体现了高内聚、低耦合的设计特性，通过设计 action、service、dao 的三层结构，更是将跟层级间的耦合性达到最大。

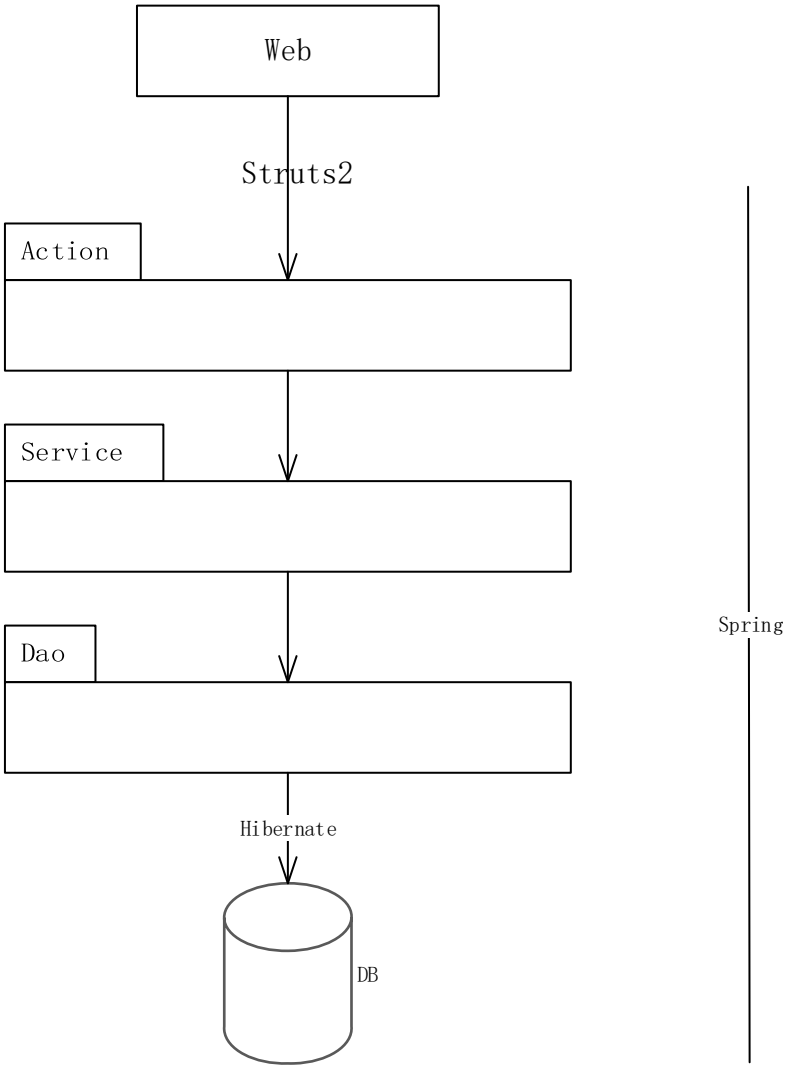

图 2.3.1 系统架构图

4. 高内聚(high cohesion)

• 内聚是评价一个对象的职责被关联的尺度或者强弱，也可以说是功能性内聚的职责。也就是功能性紧密的相关职责应该放在同一个类中，并共同完成有限的功能。这样做更加有利于对类的理解和重用，也可以降低类的维护成本。

在低耦合已经详细说明。

5. 控制器原则(controller)

控制器模式的实质是将一些系统事件的接受和处理委托给一个的对象 controller，这个对象可以是一个类，系统或者子系统，它不与 UI 进行交互，它只负责系统信息的接收和处理。

针对控制器模式，在本系统中，我们创建了 action 层，来全权处理业务逻辑，很好的体现控制机原则。

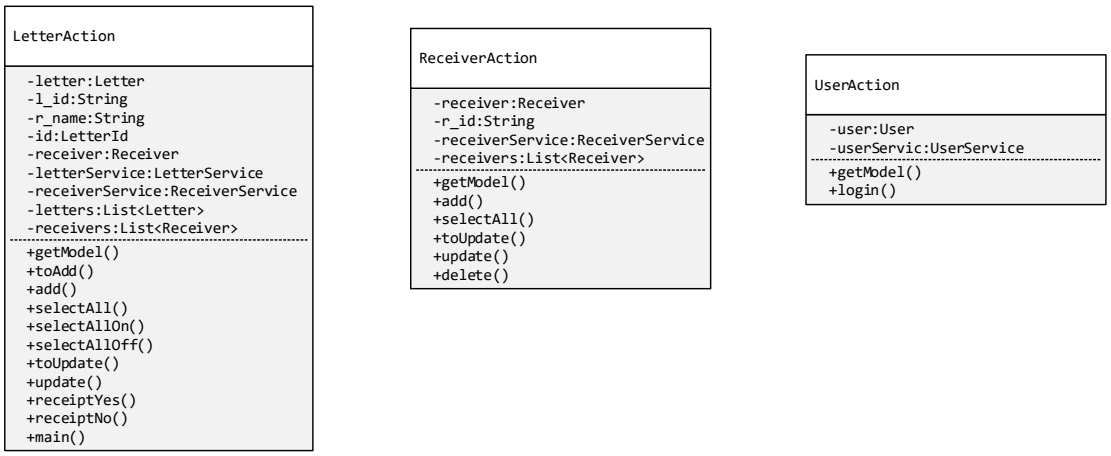

图 2.5.1 action 层全部类图

6. 多态原则(polymorphism)

多态原则与面向对象设计原则中的多态概念类似。

对与本系统，多态主要展现在一个一个方法的重写，主要存在一 antion 层，如图 2.5.1 和图图 6.1.1 可以看到我们继承了 ModelDriven 和 ActionSupport 两个类,并对其 getModel（）方法进行了重写，这就很好的体现了多台原则。

图 6.1.1 antion 继承的两个类

7. 纯虚构(pure Fabrication)

纯虚构原则与我们所说的纯虚函数类似。纯虚构的作用是用来解决高内聚和低耦合之间的矛盾的。高内聚低耦合是我们系统设计的终极目标，高内聚意味着我们要将类拆分成多个功能集中的类，但是拆分的多个类之间需要进行协作才能正常工作，这样又增加了类之间的耦合性。

在本系统中，纯虚构的使用有很多地方，如数据库连接部分，我们使用的 Hibernate 来连接数据库，这个类就是虚构的、不存在的类，还有我们在信息专家模式中设计的两个工具列，也是纯虚构的类。还有所有的 Service 层的类都是为了处理业务逻辑而设计的类，也是纯虚构的了类，还有 dao 层是为了和数据库交互而设计的类，也是不存在的类。

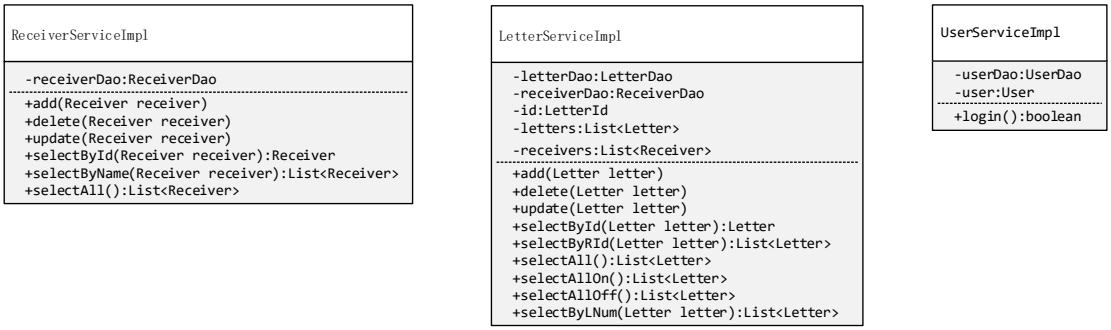

图 2.7.1 service 层实现类

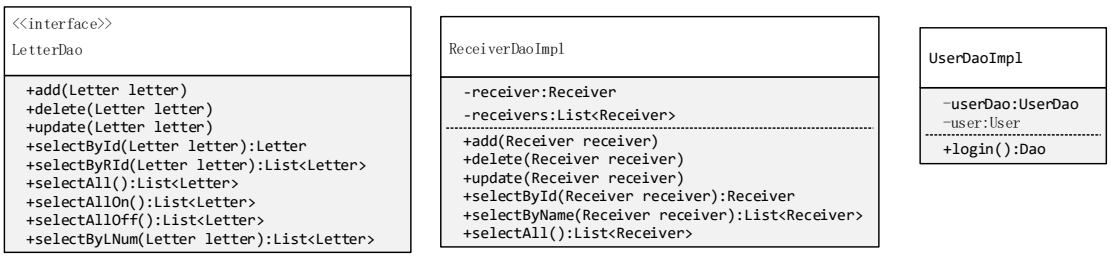

图 2.7.2 dao 层实现类

8. 中介模式(indirect)

中介模式的目的是为了避免两个对象之间产生直接耦合，降低对象之间的耦合度。解决方案是建立中间对象来协调两个对象之间的交互，避免耦合性过高。

中介模式在本系统中的应用点也很多。其中最为明显的是在实体类中—图 2.1.2 实体类图中，我们数据库只有三张表，我们却生成了四个实体类，其中额外生成的实体类也是纯虚构模式的产物，但他更加明显的体现了中介模式的理念。通过设计一个 LetterId 的实体类，来讲 letter 和 user 关联起来。

9. 受保护模式(protected variations)

受保护模式的实质与 OCP(开放闭合原则)类似，我们首先找到系统中不稳定的变化点，使用统一的接口封装起来，如果未来发生变化的时候，可以通过扩展接口来扩展新的功能，而不需要改变旧的代码。这样达到易于扩展的目的。

在本系统中受保护模式的设计主要体现在基于接口的个规范化开发中，我们在 dao 和 service 中均面向接口进行编程，从而很好的体现了受保护模式。

三、 使用总结

在一个系统的设计中，没有明确的规定是否使用以及如何使用某一原则，我们通常在开发中组合以及交替使用多个原则，来达到设计出一个健壮、可扩展性更高的系统。