**摘要:** 2014年3月，本人所在公司承担了一项农业系统平台的开发项目，该项目主要是实现农业系统各项内部业务，以及各项农业项目的审批工作，并提供外部用户通过web服务进行信息访问。我在该项目中担任系统架构设计师一职，负责系统的架构设计和软件开发的部分设计工作。本文以该农业系统平台的开发项目为例，主要论述了软件设计模式在该系统开发中的具体应用。在农产品标准化模块中，针对不同的农产品质量指标我们采用了责任链模式；在数据访问中我们采用了工厂模式，以实现对不同数据格式的转换；在验证码生成中我们采用了策略模式，以实现算法的灵活替换。通过使用这些设计模式，提高了软件的设计质量和开发效率，最终项目顺利上线，并获得用户一致好评。

2014年3月，本人所在公司承担了某市农委系统的系统平台的开发项目，该项目是农业系统的工作平台。不仅承担着农委系统的内部业务工作，包括：生产处、环能处、经管处、农村处、生态处等多个处室。通过实施该系统，可以实现不同处室的业务信息的共享和交流，消除信息孤岛，提高办事效率和质量。另一方面，通过这个平台，可以为农产品加工企业、合作社农户等涉农群体，提供信息公开、在线审批、政策查询、留言信箱、技术推广等农业服务，实现与农产品加工企业、合作社农户等社会群体的网上在线交互，提高服务三农的质量和水平。在该项目中，本人担任系统架构设计师，负责项目的架构设计以及软件开发的部分工作。  
由于传统的结构化的软件设计方法不符合面向对象的设计原则，无法很好的实现高内聚和低耦合的要求。模块之间过于紧密，给软件扩展和维护带来很多困难。在这种情况下，设计模式的出现和广泛应用给问题的解决提供了一种有效方法。通过利用设计模式，可以帮助开发者复用已有的设计方法，设计出结构合理、易于复用和可维护的软件，当用户需要发生改变时，可以通过修改少量代码或不修改原有代码即可满足新的需求，增强了系统的可修改性和稳定性，降低系统开发成本。  
一般而言，一个设计模式具有模式名称、适应场景、解决方案和效果四个方面的基本要素。设计模式依据其目的可以分为创建型、结构型、行为型三种类型。创建型模式，主要负责对象的创建工作，程序在确定需要创建对象时，可以获得更大的灵活性。常用的创建型设计模式有：单例模式、工厂方法、原型、构造器、抽象工厂等5种模式。结构型模式，负责处理类或对象之间的关系，用于构件结构更加复杂庞大的系统。常用的结构型设计模式有适配器、桥接模式、享元模式、组合模式、外观模式、代理模式等7种模式。行为型模式，主要任务是对类或对象如何交互以及为类和对象分配具体职责进行描述。常用的行为型模式有观察者、状态、策略模式、备忘录、命令、责任链、中介者等11种模式。这些设计方法都是经过反复使用的成熟方法，对优化软件结构，提高软件质量具有重要的指导意义。  
在农业信息平台的开发过程中，我们综合使用了多种设计模式，本文着重对责任链模式、工厂方法、策略模式等3种设计模式在该项目中的具体应用进行介绍。  
一、责任链模式  
我们在信息平台的开发过程中，需要完成对农产品质量进行标准化评选，从低到高评选无公害农产品、绿色产品、有机食品、地理标志认证4种认证，其中，无公害农产品的认定数量较多，标准较低，由农业生产处进行认定。在认定过程中，我们采用了责任链的设计模式。首先，定义了农产品对象fproducts,该对象中保存有农产质量的各项指标，包括水、空气、土壤等环境质量指标，及耕地净化、品种优质高抗、投入品无害化等生产技术。能够全面反映农产品质量水平。其次，我们定义了接口类deal，接口中持有一个农产品对象和自身的接口，以及处理函数processrequest。对外提供对农产品进行分类，并存入不同的信息数据库。随后，我们定义了无公害处理类、绿色食品、有机食品和地理标识4个实现类。对农产品对象fproducts的处理，按照由高到低的顺序，依次进行处理，直到符合某个标准为止，并完成信息处理，将对象信息按照审核的分档标准，存入信息库。通过这个方法，可以实现农产品对象，与处理方法的分离。  
二、工厂方法  
在农业产业化管理过程中，需要对各区市数据进行采集，由于不同类型的数据导入算法不同，在程序设计过程中，设计者需要定义若干类分别实现导入excel、xml、sqldata等类型的数据的算法，而且用户导入的数据类型存在不确定性，设计者无法确定应该实例化哪一个类。为解决这一问题，我们使用工厂方法模式。首先，定义一个数据访问接口类import。同时，针对不同的数据类型，还定义了ImportExcel、ImprotXML、ImportSQLDATA具体产品类，实现了import所声明的公共接口，其主要功能是封装了不同类型的数据导入到数据库的具体算法。Importcreator是抽象工厂类，持有一个接口产品类import的对象。Importexcelcreator、importXMLCreator、importSQLdata是具体工程类，主要功能是生产具体产品实体，直接在客户端的调用下创建产品实例。通过工厂方法模式的引入，可以有效解决客户需要变化对设计的影响，设计者无需知道那个子类被实例化，子类会根据具体情况自己决定实例化哪一个类，而且创建具体产品的细节也有着很好的封装，符合高内聚、低耦合的设计和原则。当需要在系统中添加新的产品时，也不需要修改抽象工厂和抽象产品的接口，以及其他具体工厂和具体产品，具有很好的的可扩展性。  
三、策略模式  
在系统的安全性方面，我们采用了用户名—密码—自动验证码相结合的办法，以保证系统访问安全性。根据验证码的使用环境，一般分为数字验证码、汉字验证码、英文验证码3种类型。而生成不同类型验证码的算法存在巨大差异，为此需要定义不同的生成验证码的算法。为解决此问题，可以利用策略模式将不同的算法封装起来，并使他们可以相互替换，使得算法独立于使用它的客户而变化。在设计策略模式中，我们定义了3个角色。环境角色：持有一个抽象策略角色StrategyVerifyCode接口的引用，并通过StrategyVerifyCode接口，来实现一个具体的策略算法。抽象策略角色：定义所有的具体策略类所需的统一访问接口；具体策略角色：包装了相关的算法或行为；在该项目中，我们按照数字、文字、字符3种类型，分别定义了shuzi\_verify、zifu\_verify、wenzi\_verify3个具体的策略类。通过使用策略算法，将生成验证码的算法封装在一个个独立的策略类中，用户可以根据自己的需求从不同策略中进行选择，有效的避免了使用条件转移语句不易维护的缺点。而且策略模式利用组合代替继承，将算法的实现与算法的选择分离开来，降低了程序之间的耦合度，增强了代码的可扩展性和可维护性。  
以上设计模式的选用基本达到了预期的效果。首先是，这些设计模式都是一些常用的设计方法，在架构设计师、系统分析师、开发人员之间，形成了良好的沟通桥梁，大家很容易进行交流和沟通。其次，在使用设计模式过程中，软件的开发效率较高，能够节省开发成本。最重要的是，这些模式都是一些经过反复使用的成熟设计方案，符合面向对象中设计规范，比如：面向接口编程、里氏替换原则、单一职责原则、依赖倒转等设计原则，最大限度的提高软件的标准化，为日后的系统维护打下了很好的基础。  
当然，我们在设计过程中，也存在一些问题和不足，不少开发人员在设计过程中，有时还是习惯于原有的设计方法，对模式的使用有些抵触。而且，这些设计模式在应用过程中，往往不是单独使用，需要对多个模式进行综合运用。这方面，我们还缺少相关的经验。所以，在以后的项目设计中，我们将继续应用各种设计模式，做到融会贯通，不拘一格的目标，争取能设计出更多的高质量软件项目。