 摘要 通用报表系统设计运用模型-视图-控制器设计模式构造客户端报表视图与报表数据间的协作模型，将报表框架与报表数据分离，用户能根据处理需要自定义报表式样和指定数据源，系统自动生成所需要的报表。本文给出它们各自在报表系统设计中的应用范例。 关键字 报表系统；设计模式；数据异构；多窗口支持；模型-视图-控制器 1 引言 传统报表系统，通常是针对某个商业领域使用，其使用的报表格式往往在设计时由设计人员已经定做成模版的形式存储在模版库中，用户使用时直接从模版库读取，处理方式也仅限于该领域内；虽然在一定程度上该解决方法带来了一定管理上的便宜，可对于现代企业用户来说，报表格式单一已经局限了报表只能作为记账簿来使用，使得报表的可塑性差，后期维护艰难，难以适应生产过程的多样性和变化性，无法满足大型企业不断扩充的适应性、智能型的要求，特别是当业务领域变化时，原有的报表系统往往很难支持新业务数据的管理分析，要进行大量的重新开发工作。 在本通用报表的设计中，将报表框架与报表数据分离开，用户即能根据自己的需要随意绘制表格，又能按照业务要求自主的选择数据来源。一旦框架和数据来源定义完毕，系统可以自动生成所需要的报表。为了经济的达到方便使用的目的，必须吸收先进的软件开发思想，采用优秀的软件开发方法以提高软件质量和软件的重用性，其中提高软件的重用性是减少开发成本的关键。 本文主要介绍MVC设计模式

在通用报表系统开发中的应用，给出了具体问题相应的解决办法，提高了软件的通用性和扩展性。 2 设计模式 设计模式是设计面向对象软件的过程中记录的知识和经验，用一系列类结构和对象来具体描述其含义。设计模式的目的就是复用这些面向对象设计的解决方案，根据具体应用完成具体的设计以及便于这些抽象解决方案的积累和交流。与不使用设计模式的软件系统相比，一个大量使用设计模式的软件系统的对象建模更加合理，对象间的耦合度更小，效率、可靠性、可升级性、并发性、平行性和分布性更高，更能获得高层次的设计复用和代码复用。 设计模式概念最先来自于城市建筑专家对建筑模式的定义“ 每一个模式描述了在人们周围不断反复发生的问题，以及该问题的解决方案的核心。这样，你就能一次又一次的使用该方案而不必做重复劳动 ”。这种建筑上的模式思想在面向对象的设计模式中同样适用，模式的核心就在于提供了相关问题的解决方案。设计模式确定了所包含的类和实例，它们的角色、协作方式以及职责分配。它通过刻画部件静态和动态结构及其之间的合作关系，成功地应用于解决商业数据处理、电子通信、图形用户界面、数据库、分布式通信软件等软件构造中的问题。 一般而言，设计模式有4个要素：①模式名称：用来描述问题、解决方案和效果。②问题：描述可以在什么时候使用设计模式。③解决方案：描述了设计模式的组成部分，它们之间的相互关系及各自的职责和协作方式。④效果：描述了模式应用的效果及使用模式应该权衡的问题。一个设计模式命名抽象确定了一个通用设计结构的主要方面，这些设计结构能用来构造可重用的面向对象设计。 我们在报表系统中主要使用了模型-视图-

控制器设计模式(MVC)、观察者（Observer）、适配器模式（Adapter）以及桥接(Bridge)这几种设计模式。 3 设计模式的应用 3.1 模型-视图-控制器（MVC） 报表系统中为了方便用户对数据的分析和使用，同一业务数据常常需要多种视图呈现，即一个表格对象和一个柱状图对象可使用不同的表示形式描述同一个应用数据对象的消息。表格对象和柱状对象并不知道对方的存在，这样使用户可以根据需要单独复用表格或柱状图；当用户改变比表格中的信息时，柱状图能立即反映这一变化，这一行为意味着表格和柱状图都依赖于数据对象。早期的图形化程序设计常常围绕着事件驱动的用户界面来组织，这样的直接后果就是数据处理、程序功能与显示代码完全纠结在一起。大型的图形化程序中一个数据通常对应多种表示与处理方式，把特定界面绑定到应用程序上严重降低了程序的灵活性，使得一个很小的改动也牵扯到大量的代码，增加了程序开发与维护的工作量。20世纪70年代，MVC模式在small talk 80的GUI设计中被提出，并且描述了不同部分的对象之间的通信方式，使它们不必卷入彼此的数据模型开发方法中，使程序结构变得清晰而灵活。 MVC模式包括三个部分：模型（Model）、视图（View）和控制器（Controller）,分别对应于内部数据、数据表示和输入输出控制部分。模型是与问题相关数据的逻辑抽象，代表对象的内在属性，是整个模型的核心。它采用面向对象的方法，将问题领域中的对象抽象为应用程序对象，在这些抽象的对象中封装了对象的属性和这些对象所隐含的逻辑。视图是模型的外在表现，一个模型可以对应一个或者多个视图，如图形用户界面视图、命令行视图、API视图；或按使用者分类：新用户视图、熟

练用户视图等。视图具有与外界交互的功能，是应用系统与外界的接口：一方面它为外界提供输入手段，并触发应用逻辑运行；另一方面，它又将逻辑运行的结果以某种形式显示给外界。控制器是模型与视图的联系纽带，控制器提取通过视图传输进来的外部信息，并将用户与View的交互转换为基于应用程序行为的标准业务事件，再将标准业务事件解析为Model应执行的动作（包括激活业务逻辑或改变Model的状态）。同时，模型的更新与修改也将通过控制器来通知视图，从而保持各个视图与模型的一致性。 实现MVC模式时面对的主要问题是Model和View的关系，在设计模式中的Observer模式很好的描述了如何建立这种关系。这一模式中关键的对象是目标（subject）和观察者(observer)。一个目标可以有多个依赖它的观察者；一旦目标发生变化，所有依赖它的观察者都得到通知，并做出响应，即每个观察者都将查询目标进行更新，以保证和目标的状态同步。这种模式允许我们独立的改变目标和观察者；用户可以单独复用目标对象而无需同时复用其观察者，反之亦然。这种模式可以在不改动目标和其他观察者的前提下增加观察者。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com