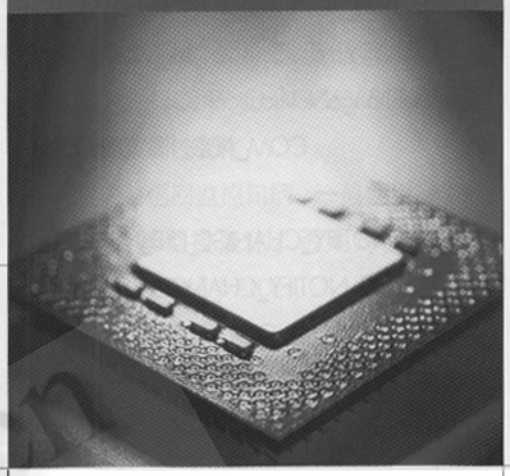
王晓楠 （江苏 常熟理工学院 215500）

MVC的设计和实现

Design and Implementation of MVC Architecture



摘 要： 本文根据个人的经验和实践，详细讨论并分析了如何设计和实现MVC结构。

关键词： 模型 视图 控制器

1 前言

用户界面，特别是图形用户界面，承担着向用户显示问题模型和与用户进行操作和I/O交互的作用。用户希望保持交互操作界面的相对稳定，但更希望根据需要改变和调整显示的内容和形式。例如，要求支持不同的界面标准或得到不同的显示效果，适应不同的操作需求。这就要求界面结构能够在不改变软件的功能和模型情况下，支持用户对界面构成的调整。

要做到这一点，从界面构成的角度看，困难在于：在满足对界面要求的同时，如何使软件的计算模型独立于界面的构成。模型-视图-控制（MVC：Model-View-Controller）就是这样的一种交互界面的结构组织模型。

2 什么是MVC

MVC由Trygve Reenskaug提出，首先被应用在SmallTalk-80环境中，是许多交互和界面系统的构成基础，Microsoft的MFC基础类也遵循了MVC的思想。

对于界面设计可变性的需求，MVC把交互系统的组成分解成模型、视图、控制三种部件。

模型部件独立于外在显示内容和形式，是软件所处理的问题逻辑的内在抽象，它封装了问题的核心数据、逻辑和功能的计算关系，独立于具体的界面表达和I/O操作；视图部件把表示模型数据及逻辑关系和状态的信息以特定形式展示给用户，它从模型获得显示信息，对于相同的信息可以有多个不同的显示形式或视图；控制部件处理用户与软件的交互操作，其职责是决定软件的控制流程，确保用户界面与模型间的对应联系，它接受用户的输入，将输入反馈给模型，进而实现对模型的计算控制，它是使模型和视图协调工作的部件。

模型、视图与控制器的分离，使得一个模型可以具有多个显示视图。如果用户通过某个视图的控制器改变了模型的数据，所有其他依赖于这些数据的视图都应反映出这些变化。因此，无论何时发生了何种数据变化，控制器都会将变化通知所有的视图，导致显示的更新。

MVC模型的最重要特点就是多个视图能共享一个模型，无论用户想要Flash界面或是WAP界面，用一个模型就能处理它们，由于模型返回的数据没有进行格式化，所以同样的模型构件能被不同界面使用，例如，很多数据可以用HTML来表示，同样，它们也可以用Flash和WAP来表示；除此之外，因为模型是自包含的，并且与控制器和视图相分离，所以很容易改变应用程序的数据层和业务规则，如果想把数据库从MySQL移植到Oracle，或者改变基于RDBMS数据源到LDAP，只需改变模型即可，而无须修改控制器和视图，一旦正确地实现了模型，不管数据来自数据库或是LDAP服务器，视图将会正确地显示它们。

这种模型的另外一个特点就是控制器可以联接不同的模型和视图去完成用户的需求，这样控制器可以为构造应用程序提供强有力的手段。给定一些可重用的模型和视图，控制器可以根据用户的需求选择模型进行处理，然后选择视图将处理结果显示给用户。

控制器处理所有和输入相关的数据；视图处理所有和输出相关的数据；模型和视图与控制器分开，向它们提供业务逻辑服务，例如：访问数据库或者其他访问介质，控制器把接收到的请求或者数据传送到模型去处理，而视图从模型中读取处理后的结果把其以不同的形式显示出来。

由于运用MVC的应用程序的三个部件是相互对立的，改变其中一个都不会影响其他两个，所以依据这种设计思想能构造良好的松偶合的构件。

3 MVC应用

这部分介绍MVC如何在网络中的应用，如图1所示，它描述了对于一个瘦客户网络应用程序，是如何应用MVC体系结构的。

图1是一个简单的应用MVC模型的瘦客户网络应用程序的结构图。从图中可以看出，

整个系统的结构分为两个部分，一部分是客户端，另一部分是服务器。客户端不运行任何和应用程序相关的代码（这就是它为什么被称为瘦客户的原因），它所需要的工具就是网络浏览器，通过网络浏览器，用户可以向服务器提出各种请求，包括显示网页、查询信息等请求，然后将服务器返回的请求结果再通过网络浏览器显示出来以满足自己的需求；而服务器和客户端不同，整个系统的代码都运行在服务器端，而我们所说的MVC体系结构也是在服务器端实现的。在服务器端，我们应用典型的三层结构，即表示层、业务层和存储层（由于篇幅有限，这里不对三层结构给以详细的介绍），而MVC的三个组成部分：视图、控制器和模型则分布在这个三层结构中，其中，视图和控制器属于表示层，而模型属于业务层。那么，MVC的三个组成部分是如何进行交互的呢?

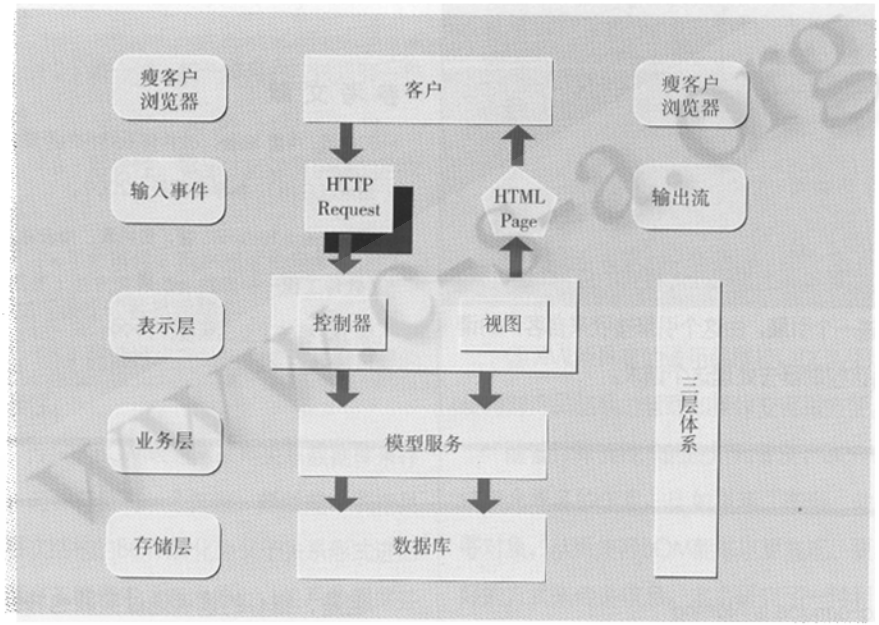


图1　MVC体系在网络中的应用

当客户端发送一个请求到服务器时，MVC中的控制器负责接收这个请求，并将请求中包含的参数传送给模型，同时，调用模型中的服务来处理这个请求，并根据模型的处理结果选择一个合适的视图，由这个视图创建一个HTML页面返回给用户，如果有必要，视图可以调用模型服务来获取处理请求的结果数据，以便组建HTML页面供用户查看。例如，当控制器接收到一个用户登陆请求时，它首先将用户名称和密码传送给模型去处理，而模型会检查传送来的用户名称和密码是否合法（可以通过查询数据库完成），并返回成功与否的标志。控制器根据模型返回的成功与否的标志来选取不同的视图，如果失败，则选取登陆失败视图，由该视图组建一个HTML页面，提示用户输入的用户名称或者密码无效；如果成功，则选取登陆成功视图，由该视图组建一个HTML页面，由于该页面需要显示该用户的某些个人信息，所以，该视图调用模型的服务来得到该用户的相关信息以完成页面的组建，并将其发送给用户。

4　MVC设计与实现

这部分将描述如何设计和实现MVC在瘦客户网络应用程序中的应用。

为了实现图1所描述的MVC网络应用程序，我们用下面的UML类图来说明其具体的设计和实现过程，如图2所示。

我们把用户请求封装到一个类HttpRequest中，这个类中的数据成员包含请求参数，例如：用户名称和密码等；控制器被封装在类Controller中，在这个类中，包括处理用户请求的成员函数，例如：判断用户的请求是否合法或者选取合适的视图等；模型被封装在类Model中，在这个类中包括一些业务处理逻辑的成员函数，例如：根据控制器传送的用户信息判断该用户是否合法等；视图被封装在类View中，这个类包含一些和显示相关的成员函数，例如：建立一个HTML页面或者将该页面传送到客户端等。

图2是一个抽象的MVC网络应用模型，我们可以利用它来实现一个具体的实例，即用户登陆的过程：服务器接收到一个登陆请求对象（类LoginHttpRequest的实例，而类LoginHttpRequest是类HttpRequest的派生类），在这个实例中包含两个数据成员：用户名称和密码；登陆请求控制器对象（类LoginController的实例，而类LoginController是类Controller的派生类）接收这个登陆请求对象，并将其参数传送给登陆模型对象（类LoginModel的实例，而类LoginModel是类Model的派生类），而这个登陆模型对象会查看数据库以便确认此用户名称和密码是否有效，并返回成功（True）或者失败（False），

登陆控制器对象根据登陆模型返回的成功与否结果来判断选取哪种视图，如果是失败，就选取类登陆失败视图（类LoginFailView的实例，而类LoginFailView是类View的派生类），如果成功，就选取登陆成功视图（类LoginSuccessView的实例，而类LoginSuccessView是类View的派生类），并调用该对象的方法BuildPage()创建HTML网页，然后，调用Print()方法请该页面发送到客户端，在成功的情况下，登陆成功视图对象可以调用登陆模型对象中的方法以便获取某些组建网页所需要的信息，例如：该用户的Email地址或者真实姓名等等。

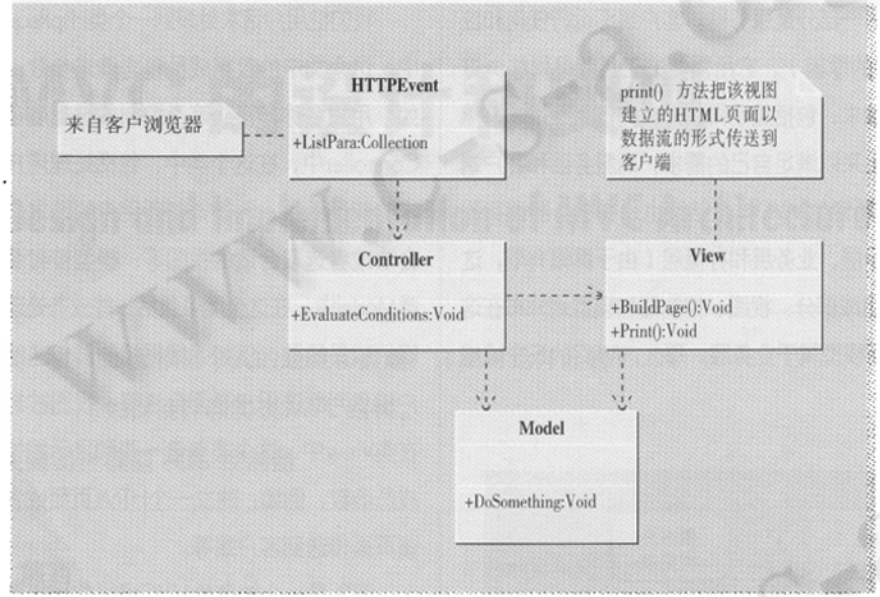


图 2　实现 MVC 的类图

通过以上的叙述，我们不难知道，MVC结构的实现实际上是Factory模式的应用。

对于大型网站，我们还可以扩展MVC结构，引进一个引擎，由这个引擎接收来自客户的请求，并根据请求的类型选取一个合适的控制器，由该控制器去处理这个请求。

不难看出，这种扩展的MVC结构是Abstract Factory模式的应用。

5 结束语

MVC结构是创建软件的很好途径，它所提倡的一些原则，例如：内容和显示互相分离；隔离模型、视图和控制器的构件等，会使应用程序的构架更健壮，更具扩展性，同时，它也会使软件在代码重用和结构方面上一个新的台阶。

参 考 文 献

1 万建成、卢雷 编著，《软件体系结构的原理、组成与应用》，科学出版社，2002。

2 [美]Roger S.Pressman 著，黄柏素、梅宏译，《软件工程——实践者的研究方法（第四版）》，机械工业出版社，1999。