**厦门工学院毕业论文（设计）外文文献翻译**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 学院 |  |  | | | | 学院 |  | | | |
|  | 级 |  | | | | 专业 |  | | 班 |  | |
| 学生姓名： | |  | | 学号： |  | 指导教师： | | |  | | |

MVC 设计模式

作者：Ralph F. Grove

1.1 介绍

模型-视图-控制器(Modle-View-Controller，MVC) 设计模式被一些web应用框架作为基础架构，例如ASP.NET，Rails，以及Struts。MVC 最初是在施乐帕克研究中心(Goldberg和Robson，1985) 开发的Smalltalk编程环境中实现的。为了适应web框架，MVC已经演变成了另一种方式，最终成为一种不同于其他任何设计模式，也与原始的Smaltalk完全不同的模式的实现。

本文的第一个目标是介绍MVC 设计模式，其中包括它的原始形态(第2节) 以及现代众所周知的用于web 应用框架的变更后的形态(第3节) 。第二个目标是对这个模式演变后发生的变化进行评估，同时呈现演变后版本的有效性(第3节) 。最后，我们提出了一个标准的MVC-Web 设计模式的描述，用于反映目前在web 框架中模式的使用，同时又能保持原始的MVC 中令人满意的特性。

基于MVC 的web 应用框架的修订版本已经被提出了(Chun, Yanhua, 和Hanhong, 2003) (Barrett和Delaney, 2004)。但是，本文并没有提出新的MVC 架构，而是分析和记录了MVC 模式从Smalltalk 到适应web 框架的演变。

2. SMALLTALK 中的MVC

MVC 设计模式是随着Smalltalk 的编程环境而引入的，从此我们可以以模块化的方式来构建交互式应用程序(Krasner和Pope, 1988)。正如这个名称所暗示的一样，MVC 设计模式的功能可以分解为三大部分。

模型(model)组件封装了应用程序的特定域的结构和功能，其本质就是包括了应用程序的状态以及改变这种状态的操作。模型还保持着对视图和控制器组件的依赖，当应用程序的状态发生变化时它会有通知。这种行为是观察者模式下的一个实例(Gamma, Helm, Johnson和Vlissides, 1995)。视图(view)组件通过图形用户界面将信息呈现给用户。应用程序中不同的操作会有多个视图，不同的视图呈现给多个用户。视图也有可能是分层的，它由一些更小的(子视图) 元素构成。当视图中包含的信息被更新时(通过对信息做出响应的模型组件) 视图会得到模型的通知，然后视图会查询模型以获得它所要呈现的信息。控制器(controller)组件通过用户界面响应用户的操作，它负责将事件传递给模型然后执行操作。控制器与视图是一一对应的存在的，多层次的视图也因此在相应的控制器之间复制。当控制器接受到输入信号时，它首先将其传送到活动的子控制器，因此输入信号首先会被最低层级的控制器处理。

用户的输入和输出形成了MVC的一个隐含的第四个组件。Smalltalk系统是基于图形显示和标准的用户输入设备，主要是键盘和鼠标。用户菜单被认为是一种虚拟类型的设备，它主要用于传送输入信号给控制器层，就跟键盘和鼠标一样。虽然菜单是在用户图形界面(GUI)中实现的，但是它们不被认为是视图组件。

MVC设计模式的主要优点是将关注点分离和由此产生的模块化。这种设计将用户界面的呈现与用户输入的操作隔离了，同时也将这两部分与应用程序的状态和事件处理过程隔离了。这就使得当你修改或替换某一个组件时，无需修改甚至无需解会其他部分。它也可以通过为新的接口介质添加一个视图/控制器的组合，或者通过独立于其他组件为模型添加新的功能而增加其可扩展性。

The MVC-Web Design Pattern

Ralph F. Grove

http://www.docin.com/p-1731117135.html

**1. INTRODUCTION**

*1.1. Basic Model*

The Model-View-Controller (MVC) design pattern is cited as the basis for the architecture of several web application frameworks, such as ASP .Net, Rails, and Struts. The MVC pattern was originally implemented in the Smalltalk-80 programming environment developed at Xerox PARC (Goldberg and Robson, 1985). As it has been adapted for web frameworks the MVC pattern has evolved in different ways, resulting in implementations that differ significantly from each other and from the original Smalltalk implementation.

The first goal of this paper is to present the MVC design pattern, both in its original form (section 2) and the variations currently used in well-known web application frameworks (section 3). Second, we present an evaluation of the changes in the pattern as it has evolved and the effectiveness of the evolved version (section 3). Finally, we propose a standard MVC-Web design pattern description that reflects the current use of the pattern in web frameworks while maintaining the original desirable qualities of MVC.

Revisions of the MVC-based web application framework design have been proposed (Chun, Yanhua, and Hanhong, 2003) (Barrett and Delaney, 2004). This paper, however, does not propose a new MVC architecture, rather it analyzes and documents the evolution of the MVC pattern as it was adapted from Smalltalk to web frameworks.

**2. SMALLTALK MVC**

The MVC design pattern was introduced with the Smalltalk programming environment as a way to structure interactive applications in a modular fashion (Krasner and Pope, 1988). As the name implies, the MVC design pattern decomposes functionality into three major components.

The model component encapsulates the domain-specific structure and functionality of the application. This essentially includes the state of the application and operations that can change state. The model also maintains dependencies of view and controller components, which it notifies in the event of changes in state. This behavior is an instance of the Observer pattern (Gamma, Helm, Johnson and Vlissides, 1995). The view component presents information to the user through a graphical user interface. There may be multiple views of different types operating within the application, presenting different views to multiple users. Views may also be hierarchical, constructed from smaller (subview) elements. When information contained in a view is updated (by a model component that is responsible for that information) the view is notified by the model and then the view may query the model to obtain information that it needs to present. The controller component responds to user actions via the user interface. It is responsible for passing transactions to the model for execution. Controllers exist in a one-to-one correspondence with views. The hierarchy of views is therefore also replicated among the corresponding controllers. When a controller receives input, it yields to its active subcontrollers first, so that input is processed at the lowest levels of the controller hierarchy first.

User input and output devices form an implicit fourth component of the MVC pattern. The Smalltalk system was based on a graphical display and standard user input devices, primarily a keyboard and mouse. User menus were also considered to be a type of virtual device that transmitted input to the controller hierarchy just as the keyboard or mouse did. Though menus were implemented in the GUI, they were not considered as view components.

The primary benefit of the MVC design pattern is separation of concerns and the resulting modularity. The design isolates user interface presentation from user input handling, and isolates both of these from application state and transaction processing. This makes it possible to modify or replace one component without needing to modify or even understand the others. It also facilitates extensibility by making it possible to add a view/controller pair for a new interface medium, or to add new functionality to the model independently of the other components.