

硕 士 研 究 生 读 书 报 告



题目  MVC架构分布研究综述

作者姓名 黄闯

作者学号 21860451

指导教师  才振功

学科专业 计算机技术

所在学院 工程师学院

提交日期 2019年1月10日

Anomaly detection algorithm for isolated forest

A Dissertation Submitted to

Zhejiang University

in partial fulfillment of the requirements for

the degree of

Master of Engineering

Major Subject: Software Engineering

Advisor: Cai zhen gong

By

Huang Chuang

Zhejiang University, P.R. China

2019

**摘要**

MVC架构自上世纪70年代构想以来，在80年代得到实现，在今天，得到了广泛的应用，其对web开发过程中的模块化，让开发者和设计者可以并行开发，使应用程序得以快速开发；而且目前众多的前端框架大多都是以MVC架构为基础，包括python的DJANGO ，java的spring，ruby的Rail,Struts,ASP .NET 等等；随着需求的深入，目前MVC架构在服务端和客户端进行了更加细致的部署，得以让服务商更好的为客户服务，并且降低服务商的宽带成本。本论文通过对部分外文文献的阅读和学习，旨在阐述MVC架构的产生、发展、以及各模块的作用，另外对于MVC架构在服务端和客户端部署的几种情况进行优劣势分析。

**关键字**：web应用，MVC，架构，服务端，客户端

**Abstract**

Since its conception in the 1970s, the MVC architecture has been implemented in the 1980s. Today, it has been widely used. Its modularity in the web development process allows developers and designers to develop in parallel and make applications faster. Development; and many of the current front-end frameworks are based on the MVC architecture, including Python's DJANGO, java's spring, ruby's Rail, Struts, ASP.NET, etc.; as the requirements deepen, the current MVC architecture is on the server side. A more detailed deployment with the client allows the service provider to better serve the customer and reduce the broadband cost of the service provider. Through the reading and learning of some foreign literatures, this thesis aims to expound the generation, development and function of each module of MVC architecture. In addition, the advantages and disadvantages of MVC architecture in server and client deployment are analyzed.

Key words:Web application, MVC, architecture, server, client

## 1 引言

MVC架构的出现改变了以往web应用程序的开发方式，也改变了web应用程序的各方面的性能；在以往针对一个web应用程序开发，所有的内容混杂在一起，在今天是不可想象的，而在开发过程中，各方面的工作分配往往无法分离，开发速度慢。

在MVC架构得以在web应用程序开发上应用之后，不仅仅提高了开发速度，更加提高了web应用系统的质量，其可移植性，可修改性，可扩展性等都得到了大幅度的提高，而且使维护方面的工作更加容易进行。

但是，针对一个应用系统，不同的系统其需求偏重不同，对于MVC架构的模块、视图、控制的部署也关系到整个系统的性能，演化出不同的方案，每种方案都有自身的优点和缺点，我们都加以深入的阐述。

这篇文章的结构如下：

第二部分主要讲MVC的产生和由来；第三部分说明MVC有那几个部分，以及每个部分的作用；第四部分讲MVC在客户端以及服务端部署的几种方案以及优劣势；第五部分是结论内容。

## 2 MVC起源

MVC最早起源于1970年时的构想，1985年在开发Xerox PARC 项目时的得以实现,在1988年作为Smalltalk 编程环境中的一种架构交互应用模块样式被引用；作为架构交互软件系统一种有用的设计模式，也被称为展示-抽象-控制（即PAC） 设计模式，核心思想是从基本的数据表示中分离出用户接口；经典的MVC设计模式事实上应用于低水平的用户交互，例如键盘对于键的敲击还有鼠标按钮的点击，同时伴随这处理用户交互的控制器，包含着应用的用户界面[2][4][5]。

这个模型应用部分包含视图应用展示，还有改变信息对用户交互做出反应，PAC模式只是简单的把应用的信息从用户界面中去耦，MVC架构更进一步，视图，控制器与模型相结合，应用数据被称为抽象部件，控制部件负责交互去耦后的模型与视图。

早期的图形化程序设计都是围绕事件驱动的用户界面来组织，直接后果就是数据处理，程序功能以及显示代码等部分都放在一起，没有把各个部分分离出来，MVC做到了这一点，所以说MVC的好处总结精炼一点就是高内聚，低耦合。

这也是一种面向对象开发思想，让不同的功能模块互相干扰的最少，这样修改起来就方便多了。

这种图形化程序中的一个数据通常对应多种表示与处理方式，把数据处理，程序功能以及显示代码放在一起降低了程序的灵活性，一个很小的改动就会牵涉到大量的代码，增加了程序开发与维护的工作量，不仅仅开发缓慢，而且需要添加新的功能点的时候，会很麻烦，而且结构混乱，MVC解决了此类问题[6]。

这个从软件架构上来讲，就是造成了不可修改性降低，以及可移植性降低，还有可扩展性也一般。

MVC的优点可以列举如下：

1. 利用MVC和PAC设计模式能够更方便的开发应用和维护应用；
2. 应用的视图可以快速更改还没有改变数据结构和商业逻辑；
3. 应用可以更容易的维护不同的界面，比如多语言，不同用户许可设置；
4. 促进了代码的重复利用，帮助开发这聚焦于一些单一的应用方面；
5. 分离数据和其表示，便于添加或者删除一个用户视图，并且用户视图的更新可以在程序动态执行中进行；
6. 模型和视图能够单独的开发(即业务逻辑和用户表现的分离)，增加了程序的可维护性、可扩展性并使测试变得更为容易；
7. 另一方面，将控制逻辑和表现界面分离，允许程序能够在运行 时根据工作流，用户习惯或者模型状态来动态选择不同的用户界面；
8. 控制器的最大的优点是可以联接不同的模型和视图完成来自需求用户的请求，为构造应用程序提供强有力的手段

## 3 MVC模式三个部分

### 3.1 模型

模型是系统的一部分，管理着所有和数据相关的任务，包括验证，进程状态和控制，数据源架构。模型极大的降低了开发者需要写的代码的复杂程度。

模型这一部分的职责就是应用的商业逻辑，封装数据处理的方法，制作一个可重用的类库。通常，一个模型和数据抽象，验证以及授权一起创建的。

模型是由一些类构成的，这些类定义了一些需要定义的域，这些对象属于存储在数据库经常多次封装，而且需要包含用来操作数据和强制商业规则的代码。

模型主要是用来处理数据存取抽象和验证，具备一些和不同数据源交互的方法[7]。

瘦模型，就是说模型保持足够简单，只需要封装和实际生命对象需要数据处理的模块，瘦模型往往和胖控制器搭配，这样的控制器包含很多应用所需要的数据处理方法，这种方法，这个模型就变的高可用在应用和很多开发者的工作之间，这些工作主要是针对控制器的。

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，采用面向对象的方法，将物理世界的对象抽象为应用程序抽象，在这些抽象的对象中封装了对象的属性和对这些对象所隐含的逻辑[1]。

模型的作用

1. 抽象应用程序的功能，封装程序数据的结构及其操作；
2. 接受controller提供对程序功能的访问；
3. 接受view的数据查询请求；
4. 当数据有变化时，同时对此数据感兴趣的view

### 3.2 视图

视图的职责在于图形化用户界面管理，这意味着所有应用需要的表格，按钮，图表元素，所有HTML元素；视图还可以用来产生订阅内容针对聚合器以及FLash展示。

通过从应用逻辑分离应用设计，我们极大的将少在展示时候的错误风险，当设计者决定选择改变logo或者表格界面的时候，同时，开发者的工作也极大减少了，因为他不再需要看HTML代码元素，设计元素以及图表元素。[4]

视图层，一般来说可以叫做网页设计或者模版，它控制着数据展示的方法，以及用户如何与它交互，也提供从用户汇聚数据的方法，这种技术被用于视图主要是HTML，CSS，JavaScript

作为一种通用的规则，视图不再包含属于应用逻辑的元素，为了让设计者能够更加容易的工作，这也意味着逻辑快需要保持最小。

今天，大多数网页应用框架用的模版引擎都是自动拼写检查功能，为了保持HTML代码最小以及降低拼写错误风险，这些发生器，通常用来制作复杂的网页局部功能，比如布局，表格，列表，菜单等等[5]。

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视图等等。

视图是用来与外界进行交互的，是应用系统与外界的接口，一方面为外界提供输入手段，触发应用逻辑运行，另外一方面把活动运行的结果以某种形式显示给外界。

视图的作用

1. 对数据的表现部分进行抽象；
2. 把数据展现给用户，获得用户输入；
3. 将用户输入转发给controller；
4. 当接受到来自model的数据已经更新的通知后，更新显示信息；

### 3.3 控制器

控制器负责事件处理，这些事件会被激发要么是用户和应用的交互，要么是系统进程；一个控制器接受请求和准备数据做出回应；控制器的职责还包括确立回应的形式，控制器和模型进行交互，为了检索需要的数据，以及产生视图，

这个过程也被称为一个活动或者一个动作，当请求到达服务器的时候，基于URL信息，MVC框架会把这个请求丢给控制器的一个方法，

控制器绑定所有的应用逻辑和视图展示，它的职责是从视图进行数据检索，确立执行路径对这个应用，控制器接受模型功能，编译数据检索命令以让视图能够展示[6]。

控制器还能够应对错误处理在用户请求过程中的问题。

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，将用户与view的交互转换为基于应用程序行为的我标准业务事件，再将事件解析为model应该执行的动作，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[3]。

1. 抽象用户交互和应用程序行为的映射
2. 将用户输入翻译成应用程序的动作，转发给model
3. 根据用户输入和model对程序动作的输出，选择适当的view来展现数据

## 4  MVC 和 web应用程序

web应用程序的发展是受想要对交叉关注点模块化以及降低web设计者和程序员之间的依赖所驱动。

传统的web应用程序，大多数是一个HTTP 路径对一个确定的服务端资源，请求参数被服务器处理，传输给web应用，产生一个HTTP回应。

大规模的web应用程序，数据是动态的以及至少涉及两门编程语言，展示是用客户端语言描述，功能专门用服务端语言实现。

在传统的web应用程序开发中，模型，视图和控制器分布横贯服务端和客户端在整个网络上。

针对MVC在服务端和客户端的分布，针对旅店预定应用程序，进行展开说明，

预定流程如下：

1. 日期选择和房间数目；
2. 选择一个房间类型；
3. 支付信息汇总；
4. 确定页面；

### 4.1 服务端MVC

所谓服务端MVC就是说，MVC结构全部分布在服务端，所有的计算和操作都是在服务端完成的。

在这种MVC分布设计中，我们可以注意到当用户在客户端，浏览器提供一个界面可视化和超文本导航，这种MVC设计模式创建了两种循环。

第一种就是

User—>Controller—>View—>User

这种用户与应用程序的交互方式，请求信息发送给控制器，然后回应信息把信息发送给视图层，视图层更新信息然后把结果发给用户。

第二种是：

User—>Controller—>Model—>View—>User

在这种循环中，用户和应用程序交互，然后信息发送给控制器，控制器更新模型的状态，最后视图更新与模型进行匹配，然后把信息发给了用户[8]。

这种MVC分布模式的选择，是非常有用的，当客户端对应用程序处理能力需求不高，或者应用程序只需要在低水平的用户交互展示信息[5]。

但是在平时一定要小心应用这种模式，因为这种模式无法更新部分视图，需要创建一个新的视图在服务端，把这个视图发送给用户。

由于客户端与服务端的交互必须性，增加了应用程序的宽带使用。

这种模式会导致服务端一直处于忙的状态，不仅仅服务页面，而且还要做表格，验证，产生UI等[6]。

这种模式，必须检查从客户端得到的两种信息，所有形式输入被验证，用户选择是可行的；如果用户数量比较多的时候，客户端与服务端的交互会急剧增加，由于出现错误导致服务无法使用以及宽带费用的增加，会导致公司出现大的问题。

### 4.2 客户端与服务端的混合模式

这种模式正是为了解决4.1出现的问题，由于用户交互和服务重载。

用这种方法，设计者选择用客户端用浏览器支持的一些客户端语言去处理一些对于数据的验证，让客户端利用一些脚本语言去处理验证，是比较容易的，但是主要的问题是由浏览器兼容性产生的，实现一些UI元素，不同的浏览器往往不同，而且甚至取决于操作系统，浏览器供应商不会执行相同标准，他们还有他们自身的bug。

这种情况被解决利用客户端系统探测，浏览器依赖验证功能，这种解决方法增加了发送给客户端的页面重量，浪费了网络资源。

但是利用这种混合分布开发方法可以解决问题。

这种方法就是把控制层与视图层分布在服务端和客户端。

这种混合MVC模式，比如单一的服务端模式，显的更加复杂，由于赋予了客户端更多的功能。

这种模式有几种业务循环

第一种：

User—>Controller(client)—>View(Client)—>User

这种与应用程序的交互方式，只需要在客户端完成，所以页面只需要由服务端发送给客户端一次，用户可以与界面快速的交互，由于它不需要服务端的操作[6]。

在客户端的控制层必须有能力决定被用户交互所激发请求操作处于什么范围以内，如果请求要求需要改变模型，则会把请求发送给服务端的控制层，但是如果影响只在展示界面，视图层更新，只需要发给客户端的就可以了。

第二种循环是：

User—>Controller(client)—>Controller(Server)—>View(Server)—>View(Client)—>User

这种与应用程序交互的方式，这种活动通常是由客户端的控制层决定的，发送给服务端的控制层，然后再转发给服务端的视图层，最后发给客户端，但是这种情况往往并不常发生，因为服务端的视图发生改变往往伴随着模型发生了改变[6]。

第三种循环：

User—>Controller(client)—>Controller(Server)—>Model—>View(Server)—>View(Client)—>User

这种模式，正是控制层改变了模型层，然后需要对视图层做出改变。

这种混合设计模式，是由目前几种框架产生的。

但是这种框架也存在很大的问题：

限制了对异步数据的检索，没有页面更新；

由于模型在服务端，限制提供高交互水平服务；

数据操作困难，因为没有任何模型层在客户端，商业逻辑或者数据是不可用的在客户端[7]；

另外，额外的交互成本在控制层和视图层之间；

这种贯穿服务端和客户端的交互成本很高，时间变长，复杂性高。

### 4.3 客户端全MVC模式

为了解决上面的问题，就在客户端也部署了模型层，虽然说这种部署方式，能够解决部分问题，但是不能解决所有问题；对服务端的请求过程是缓慢的，由于需要对所有的部件进行沟通；在客户端的模型层非常难以维护，由于每个客户端平台都需要部署代码，而不同的操作系统，浏览器，装置需要不同的代码，但是这种模式，现在在一些框架中得到运用，比如Ajax。

## 5 总结

MVC作为一种解决web应用开发速度与便捷性的解决方案，在今天已经发展了将近五十年了，在众多的web应用开发框架中得到了实现，而我们作为web开发者或者计算机技术学习者，对于MVC的发展了解非常有其，必要性；MVC在开发和维护以及提高系统性能方面虽然解决了一些问题，但是伴随着MVC分布的结构不同，也带来了其他相对应的问题，没有什么技术能够解决所有问题，我们所需要的是因地制宜，制定合适的方案，把MVC三部分结构的分布进行适当调整。

## 参考文献

[1]. Leff A, Rayfield J T. Web-application development using the model/view/controller design pattern[C]//Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International. IEEE, 2001: 118-127.

[2]. Pop D P, Altar A. Designing an MVC model for rapid web application development[J]. Procedia Engineering, 2014, 69: 1172-1179.

[3]. Grove R F, Ozkan E. The MVC-web Design Pattern[C]//WEBIST. 2011: 127-130.

[4]. Selfa D M, Carrillo M, Boone M D R. A database and web application based on MVC architecture[C]//Electronics, Communications and Computers, 2006. CONIELECOMP 2006. 16th International Conference on. IEEE, 2006: 48-48.

[5]. Wojciechowski J, Sakowicz B, Dura K, et al. MVC model, struts framework and file upload issues in web applications based on J2EE platform[C]//Modern Problems of Radio Engineering, Telecommunications and Computer Science, 2004. Proceedings of the International Conference. IEEE, 2004: 342-345.

[6]. Morales-Chaparro R, Linaje M, Preciado J C, et al. MVC web design patterns and rich internet applications[J]. Proceedings of the Jornadas de Ingenierıa del Software y Bases de Datos, 2007: 39-46.

[7]. Cui W, Huang L, Liang L J, et al. The research of PHP development framework based on MVC pattern[C]//2009 Fourth International Conference on Computer Sciences and Convergence Information Technology. IEEE, 2009: 947-949.

[8]. Deacon J. Model-view-controller (MVC) architecture[J]. Online][Citado em: 10 de março de 2006.] http://www. jdl. co. uk/briefings/MVC. pdf, 2009.