

谨以此论文献给我敬爱的导师和家人

-----乔淑夷



# 基于 MVC 模式的 Web 前端框架关键技术研究实现

学位论文答辩日期： \_\_\_\_\_

指导教师签字： \_\_\_\_\_

答辩委员会成员签字： \_\_\_\_\_

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_



## 独 创 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的  
研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其  
他人已经发表或撰写过的研究成果，也不包含未获得  
(注：如没有其他需要特别声明的，本栏可空)或其他教育机构的学位或证书使  
用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明  
确的说明并表示谢意。

学位论文作者签名：                    签字日期：      年    月    日

-----

## 学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，并同意以下  
事项：

1、学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许  
论文被查阅和借阅。

2、学校可以将学位论文的全部或部分内容编入有关数据库进行检索，可以  
采用影印、缩印或扫描等复制手段保存、汇编学位论文。同时授权清华大学“中  
国学术期刊(光盘版)电子杂志社”用于出版和编入 CNKI《中国知识资源总库》，  
授权中国科学技术信息研究所将本学位论文收录到《中国学位论文全文数据库》。  
(保密的学位论文在解密后适用本授权书)

学位论文作者签名：

导师签字：

签字日期：      年    月    日

签字日期：      年    月    日



# 基于 MVC 模式的 Web 前端框架关键技术研究是实现

## 摘 要

随着互联网 Web2.0 时代的到来,各种 Web 应用大量涌现,Web 站点的前端产生了天翻地覆的变化,网页不再简单的显示基本的文字和图片,各种富媒体元素的使用使得网页的内容更加生动,网页的交互形式增强了 Web 应用的用户体验,Web 前端技术在应用开发过程中也得到了越来越广泛的使用。

Web 应用的普及使得用户对站点的期望越来越高,直接导致网站前端规模不断扩大,越来越多的前端脚本代码增加到应用系统中,前端开发出现了代码规模大、组织维护困难、代码重用性低、扩展性差等问题。建立合理的前端开发架构,并遵循开发标准和规范将越来越重要。

本文借鉴 MVC 模式的后端架构方式,通过分析传统前端开发和维护过程所面临的问题,结合现在流行的前端设计模式,依托 JavaScript、jQuery、DOM、Ajax、JSON 等前端技术,采用理论和实践相结合的方式,编码实现了基于 MVC 模式的前端框架。本文构建的前端框架使得前端开发遵循模块式的开发方式,分离了不同类型的逻辑,减少依赖关系,实现了前端代码的解耦、复用,降低了应用扩展和维护的复杂度,具体研究过程如下:

首先,按照分层架构的思想对前端框架各部分进行设计。前端框架由模型对象、集合对象、控制器对象、视图对象及异步请求对象五部分组成,对各个对象需要完成的功能进行了分析,然后对对象中涉及的属性和方法进行了设计。

其次,对前端框架中对象的初始化及包含的方法进行编码实现,本框架中的一个创新点是在视图对象中实现了模板引擎,将 JavaScript 代码和 HTML 代码片段分离,通过模板引擎实现页面视图的渲染,使得代码易于维护,同时提高了页面渲染效率。另一创新点是在异步请求对象中优化了 Ajax 请求调度方式,提出请求优先级队列及基于等待时间延长优先级提升的策略,减少用户请求的等待时间,防止用户请求长时间无法得到响应。

最后,将本文前端框架的模板引擎与目前流行的模板引擎的渲染效率进行测试对比,将采用 Ajax 异步请求调度策略的请求与普通 Ajax 请求的响应时间进行

比较,并在相同请求数的情况下比较本文前端框架与其他前端框架的请求响应完成时间,通过测试结果可以看出本文实现的前端框架在页面渲染效率及请求响应效率方面得到了较大的提高。然后通过前端框架在新闻管理系统中的应用描述了整个框架的流程及可行性。

**关键词：**MVC；前端框架；模板渲染；模块化设计



# **Research and Implementation of key technologies in Web front-end framework based on MVC Pattern**

## **Abstract**

With the Internet entering into the Web2.0 time, various Web browser applications have prompted a large number of emerging, the front-end of the website has undergone enormous changes, the webpage no longer just contains the single text and images, a variety of rich media contents make the web page more vivid, the interactive web forms strength the experience of users for web application, web front-end technology in the development process has also been more and more extensive utilization.

A result of the popularity of Web applications is users increase high expectations of the website, which makes that the website continuously extends its front-end scale, more and more front-end scripts are added to the system, and as a consequence, the front-end development codes emerge huge scale, complexity of organization and maintenance, low code reusability, and poor expandability. To develop a reasonable front-end development framework, and following the development of standards and specifications will become increasingly important.

In this paper, for reference of the MVC Pattern in back-end architecture, by analyzing the issues in traditional front-end development approach process encounters, combining with the existing front-end design patterns, relying on the front-end technology of JavaScript, jQuery, DOM, Ajax and JSON, combining theory and practice, we implement the front-end framework on the basis of MVC Pattern. The front-end framework established in the paper makes the front-end development follow the modular development approach, and separate different types of logic. Consequently, the framework decreases the interdependent relationship and achieves decoupling and reuse of the front-end codes, and then lowers the complexity of extension and maintenance in applications, and the specific research process is as

follows:

Firstly, we design the various parts of the front-end framework in accordance with the hierarchical structure theory. The framework consists of a model object, a collection object, a controller object, a view object and a asynchronous request object, and we analyze the function of each object needs to do, and then design the properties and methods the object involves.

Secondly, we encode and realize the initialization of the objects and the functions in objects. An innovative point is to achieve a template engine which separates the JavaScript code with the HTML code fragment. It makes easier to maintain the codes by adopting the template engine to implement page view rendering, and improves the efficiency of page rendering. Another innovation is optimizing the Ajax asynchronous request scheduling in the synchronous request object. We adopt the request priority queue and promoting the priority based on waiting time which reduces the user requests waiting time, and prevents user requests could not get the response for a long time.

Finally, we test and compare the efficiency of the template rendering engine in the paper with the current popular template engine, and the response time of applying the Ajax asynchronous request scheduling strategy with the common Ajax request, and the response time by using the front-end framework and the others in the same number of requests. The results shows that the efficiency of page rendering and page response under the front-end framework of the paper has been greatly improved. Then describe the process and the feasibility of the framework with the news management system application.

**Keywords: MVC Pattern; Front-end framework; Template rendering; Modular designing**

# 目 录

<b>1</b>	<b>绪论.....</b>	<b>1</b>
1.1	研究背景与意义.....	1
1.1.1	研究背景.....	1
1.1.2	研究意义.....	2
1.2	国内外发展现状.....	3
1.3	研究内容与思路.....	4
1.4	论文的组织结构.....	5
<b>2</b>	<b>Web 前端框架关键技术介绍.....</b>	<b>7</b>
2.1	模块化设计思想.....	7
2.2	MVC 设计模式 .....	7
2.3	REST 架构.....	9
2.4	WEB 前端相关技术.....	10
2.4.1	DOM 文档对象模型 .....	10
2.4.2	JavaScript 技术.....	11
2.4.3	jQuery 框架 .....	12
2.4.4	AJAX 技术 .....	14
2.4.5	JSON 数据格式 .....	15
2.5	本章小结.....	16
<b>3</b>	<b>基于 MVC 模式的前端框架设计.....</b>	<b>17</b>
3.1	前端 MVC 框架 FRONTBASE 设计 .....	17
3.2	前端 MVC 框架 FRONTBASE 对象 .....	18
3.2.1	模型对象.....	18
3.2.2	集合对象.....	19
3.2.3	控制器对象.....	20
3.2.4	视图对象.....	21
3.2.5	异步请求对象.....	22
3.3	前端 MVC 框架 FRONTBASE 工作流程 .....	22
3.4	本章小结.....	23
<b>4</b>	<b>基于 MVC 模式的前端框架实现.....</b>	<b>25</b>
4.1	模型层的实现.....	25
4.1.1	初始化模型.....	25

4.1.2	操作模型数据.....	26
4.1.3	初始化集合.....	28
4.1.4	操作集合模型.....	29
4.2	视图层的实现.....	30
4.2.1	创建视图.....	30
4.2.2	绑定事件.....	30
4.2.3	模板引擎.....	31
4.3	控制器的实现.....	34
4.3.1	事件管理器.....	34
4.3.2	导航管理器.....	36
4.4	服务器同步的实现.....	37
4.4.1	Ajax 请求调度算法.....	37
4.4.2	Ajax 异步请求.....	39
4.5	本章小结.....	40
<b>5</b>	<b>前端框架性能测试及在新闻管理系统中的应用 .....</b>	<b>41</b>
5.1	框架性能测试.....	41
5.1.1	模板引擎渲染效率测试.....	41
5.1.2	Ajax 请求调度效率测试.....	41
5.1.3	页面加载效率测试.....	42
5.2	新闻管理模块前端设计.....	42
5.3	新闻管理模块前端实现.....	43
5.3.1	新闻管理模块模型层实现.....	44
5.3.2	新闻管理模块视图层实现.....	45
5.3.3	新闻管理模块控制器实现.....	47
5.3.4	新闻管理模块效果图.....	48
5.4	本章小结.....	49
<b>6</b>	<b>总结与展望 .....</b>	<b>51</b>
6.1	工作总结.....	51
6.2	研究展望.....	51
	<b>参考文献.....</b>	<b>53</b>
	<b>致 谢.....</b>	<b>57</b>
	<b>个人简历、在学期间发表的学术论文与研究成果.....</b>	<b>59</b>

# 1 绪论

## 1.1 研究背景与意义

### 1.1.1 研究背景

Web 前端是用户通过浏览器浏览和使用的 Web 页面<sup>[1]</sup>，包括页面的整体风格、文字、图片、广告、flash 动画等，所涉及的内容主要包括 Web 页面的结构、Web 页面的外观视觉表现及 Web 层面的交互实现，

在 Web 前端发展初期，HTML 技术只能展示简单的页面，维护和更新相当麻烦。CSS2.0 所描述的信息结构能帮助软件开发人员分离页面表现和内容，使得站点易于构建和维护，以 CSS+DIV 为主要技术的页面重构技术开始崭露头角。随着 Ajax 的出现，Web2.0 的兴起及各种 Web 应用如视频网站、SNS、博客、微博等陆续出现，人们对 Web 应用交互复杂度和用户体验的需求不断提升，以 JavaScript 为标志的 Web 前端开发，对页面上的 DOM 操作、数据处理、数据交互等进行处理，前端开发逐渐进入高速发展的时期。

目前 B/S 架构下前端数据处理存在下面几个问题：

- 数据模型、业务模型不清晰；
- 代码重复，复用率低；
- HTML/CSS/JS 代码生命周期延长，导致代码可读性和可维护性降低；
- 难以满足不断变化的需求，特别是频繁变化的前端需求。

jQuery 在一定程度上改善了原生 JavaScript 代码编写繁琐复杂的问题<sup>[2]</sup>，但是，jQuery 主要功能是查找和操作 DOM 元素，在 Web 应用中可以作为底层库使用。随着用户体验需求的不断提高及团队开发人员的不断变化，如果没有一定的编码规范，每个工程师有自己的编程风格，应用代码依然杂乱无章，代码后期将会越来越复杂，维护越来越困难，同时对系统性能产生负面影响，页面加载速度慢，降低用户体验。所以，通过构建前端框架，遵循模块化和分层的设计思想，在框架的基础上实现统一编码，有利于提高系统的开发效率，同时有利于提高模块的复用性及可维护性。

本文实现的基于 MVC 模式的前端框架与其他同类框架分层相似，包括模型

层（数据处理逻辑部分）、视图层（显示数据部分）、控制器层（事务处理逻辑部分），实现前端页面的逻辑处理及数据展示分离<sup>[3]</sup>，达到模块复用，易于开发和维护的目的。同时，本框架具有自己的特点：1) 基于 jQuery 框架进行开发具有良好的浏览器兼容性；2) 采用优化的底层操作，具有较高的代码执行效率；3) 对用户请求及 DOM 渲染进行优化，提高请求响应速度及页面加载速度，从而提高用户体验。

### 1.1.2 研究意义

早期的 Web 前端功能简单，基本以页面作为工作单位，内容以浏览型为主，偶尔出现简单的表单操作，通过简单的 JavaScript 逻辑代码进行实现，基本不需要框架的支撑。随着 Ajax 应用的大量出现，人们可以在页面上进行比较复杂的操作，Web 前端功能增强，前端代码量迅速膨胀。此外，Web 应用性能优化黄金法则<sup>[4]</sup>中提到 80% 或以上的用户请求响应时间花费在前端程序性能上，即用户访问网站资源，花费时间最长的并不是后端应用程序处理以及数据库等消耗的时间，而是前端花费的时间（包括请求、网络传输、页面加载、渲染等）。如何合理有效的规范编码，提高代码的复用性和维护性，提高 Web 应用的性能，前端框架应运而生。

根据 Web 前端开发过程中出现的问题，前端架构的实现需要解决如下几个问题：

- 1) 解决在不使用框架的开发方式下编码方式不统一，模块性差、缺乏消息传递机制，全局变量滥用等问题<sup>[5]</sup>。
- 2) 减少不同模块之间代码的耦合，多个视图共享一个模型，同一视图调用多个模型，达到代码复用的目的。
- 3) 优化底层逻辑处理方法，提高系统性能。

与传统 Web 前端开发方式相比，本文实现的前端框架按照 MVC 模式进行模块式分层，采用统一的编码方式，最大程度上实现代码的复用，避免代码的冗余，提高了 B/S 架构系统前端的开发效率和维护效率。与现有的前端 MVC 框架相比，本文实现的前端框架对底层逻辑处理代码进行了优化，提高了页面响应效率，实现了良好的用户体验。

## 1.2 国内外发展现状

Web 页面动态交互的需求不断变化,使得页面中的数据处理代码和显示代码混杂在一起,如何有效地组织前端代码,使得 Web 应用前端结构化,易于阅读和维护,成为 Web 前端开发人员面临的难题。W3C 根据对页面中的代码进行分类,提出了内容(Content)-结构(Structure)-表现(Presentation)-行为(Behavior)相互分离的页面开发标准<sup>[6]</sup>。其中,网页中的文字、图片、视频等数据属于内容范畴,内容按照一定的结构组织,如页面的标题、作者、正文,表现指通过 CSS 样式设置内容的外观,行为是通过 JavaScript 实现内容的请求和更新。

通过 Web 页面开发标准可以有效的对页面代码进行分离,独立完成各个部分的开发,目前国内外比较流行的浏览器前端框架有 jQuery 框架、ExtJS 框架、Mootools 框架等。

jQuery 框架是一个高效简洁的 JavaScript 框架<sup>[7]</sup>,它使得用户更方便地选择页面中的元素,改变页面的内容,修改页面的外观,响应用户的页面操作,实现动画效果,并且方便地为网站提供 Ajax 交互。jQuery 框架语法简洁,具有完善的 CSS 选择器,使得 DOM 操作更加便捷。在用户可视化组件上, jQuery 提供了官方的 jQuery UI 插件,为开发者提供了创建丰富交互体验的 Web 应用程序的基础。由于设计思想追求简洁和高效<sup>[8]</sup>,没有进行面向对象方面的扩展,在开发 Web 应用时 JavaScript 代码中嵌套 HTML 代码片段,浏览器解析速度慢,模块复用性差。

ExtJS 框架是一个用 JavaScript 编写,基本与后台技术无关的前端框架<sup>[9]</sup>,用于在浏览器端创建丰富多彩的 Web 应用程序界面。利用 ExtJS 框架构建的 Web 应用将页面显示、数据管理及逻辑处理分离,视图创建并渲染 HTML 标签与 CSS 样式,以组件如网格或树的形式展示在页面上,模型以 JavaScript 对象的形式保存应用程序的数据域及相应数值,控制器实现视图中控件与模型之间交互行为的接口。ExtJS 框架采用面向对象与组件化的设计思想<sup>[10]</sup>,主要功能包括:对标准信息提示框的扩展、对标准表单组件的扩展、支持面板及各种布局方式,提供增强的事件机制、提供对 Ajax 功能的支持,具有结构完整的数据模型。但是,框架中使用了大量的 UI 组件,体积较大,导致页面加载速度慢。此外,在 ExtJs2.0 之后的版本都是收费的,阻碍了其推广和应用。

Mootools 框架是一个简洁、模块化，面向对象的开源 JavaScript Web 应用框架<sup>[11]</sup>。它为 Web 开发者提供了一个跨浏览器 JavaScript 的解决方案，在处理 HTML、CSS、JavaScript 的时候，它提供了一个比普通 JavaScript 更面向对象的 Document API<sup>[12]</sup>。其功能模块化，可自由选择使用哪些模块，使用时只需导入所使用的模块，语法简洁、直观，易于阅读和修改，特效比 jQuery 略强，但是，Mootools 框架修改了底层的一些类如 Array、String 等及扩展 DOM 的方式，不利于框架的扩展，同时 DOM 和 CSS 选择器没有 jQuery 强大。

本文实现的前端框架是在 jQuery 框架的基础上进行的二次开发，采用面向对象的设计思想，对 jQuery 方法进行封装和优化，按照 W3C 提出的 Web 页面开发标准，对前端逻辑交互进行分层，实现代码复用。

### 1.3 研究内容与思路

本文主要研究内容包括：

1) 分析 Web 前端的工作流程，在 jQuery 的基础上进行二次开发，实现了基于 MVC 模式的前端框架，该框架按照前端逻辑处理流程包括数据处理，事件处理及页面显示三部分，对每部分包含的对象类进行定义，并对对象中的属性和方法进行设计和实现。

2) 实现模板引擎，在传统的 Web 应用开发过程中，通过 JavaScript 渲染 DOM 时，采用字符串拼接创建 DOM，然后附加到父元素上，如果附加的 DOM 是动态易变的，需要在函数中写大量逻辑，导致代码复杂，页面渲染速度下降。本文实现的模板引擎将 JavaScript 处理代码和 HTML 代码片段进行分离，通过将业务数据和设置特定标记的 HTML 模板整合生成 HTML 字符串返回。模板引擎实现了动态数据和静态数据的分离及代码重用，简化了 Web 前端开发工作，同时提高了页面渲染的效率。

3) 对 Ajax 请求进行优化，提出 Ajax 连接请求快响应调度算法并在前端框架中进行应用。Ajax 异步请求的特性使得用户可以在页面上发送多个并发的请求，但是一些主流的浏览器如 IE、Firefox、Opera、Chrome 限制了同一时刻的请求最大并发连接数<sup>[13]</sup>，这样就会出现問題：当用户同时发起的连接请求数大于浏览器的并发连接限制数时，连接的请求可能被阻塞，其他未连接的请求一直不能



得到请求，页面停止响应，本文对 Ajax 请求进行优化，解决 Ajax 请求并发连接服务器快响应的问题。

## 1.4 论文的组织结构

本篇文章总共分为六部分，组织结构如下：

第一章是绪论，主要介绍了论文的研究背景、研究意义、国内外前端框架发展现状，对目前流行的前端框架 jQuery 框架、ExtJS 框架、Mootools 框架等进行了介绍，通过研究和分析前端开发中的问题及前端框架的现状，得出本文的主要研究内容。

第二章介绍 MVC 前端框架开发中使用的关键技术，阐述了 Web 前端模块化设计思想的重要性，介绍了 MVC 设计模式及 REST 架构风格，分析了前端开发中关键技术的使用方式，包括 DOM 文档对象模型、JavaScript 技术、jQuery 框架、Ajax 技术、JSON 数据交互方式。

第三章根据 MVC 模式对前端框架进行分层，前端框架由模型对象、集合对象、视图对象、控制器对象、异步请求对象五部分组成，对各个对象包含的属性和方法进行设计，并描述了基于本框架的工作流程。

第四章对本文前端框架进行实现，包括模型对象、集合对象、视图对象、控制器对象、异步请求对象的初始化及所包含的方法，对模板引擎流程进行分析，并对模板引擎进行编码实现，提出 Ajax 请求调度策略，并在异步请求对象中进行实现。

第五章对本文前端框架中模板引擎的渲染效率、Ajax 请求调度效率、页面加载效率进行测试，并在框架的基础上搭建新闻管理系统，验证框架的可行性。

第六章是总结与展望。总结本文完成的工作，指明本文前端框架实现的不足以及需要改进的方面。



## 2 Web 前端框架关键技术介绍

### 2.1 模块化设计思想

随着计算机技术的快速发展,程序设计方法经历了多次变革,起初是功能分解法,即首先定义各种功能,然后将功能分解为子功能,同时定义功能之间的接口,然后是结构化程序设计方法,其核心是模块化,是一种把功能模块分离的方法,现在广泛使用的是面向对象程序设计方法<sup>[14]</sup>,其重点是系统的数据结构,实现数据和操作的封装。软件规模和复杂度越大,软件架构就越复杂,而在模块化开发中封装模块内部实现细节,使得模块开发更重视架构的设计,标准化的架构具有更高的开发性和灵活性等优点,可以更好地支持模块的复用。

模块化开发使得模块的重用性增加,进而减少了开发人员的重复编码,同时也使得开发人员之间的分工更加明确清晰,开发人员只需要对自己负责的模块进行设计和实现,能够有效地提高开发效率和代码质量。

模块化的软件具有很多显而易见的好处。在开发期,一个模块化的设计有利于程序员实现,使其在实现过程中一直保持清晰的思路,减少潜伏的 BUG;而在维护期,维护人员只需对问题模块进行调试,而不会影响其他模块的使用。应用程序的更新粒度从整个应用程序细化为模块。此外,软件设计的模块化将不同功能的模块设计成小耦合度模块,使程序执行出错率降低,提高程序可靠性、复用率和可维护性<sup>[15]</sup>,降低代码占有空间,延长程序生命周期。

### 2.2 MVC 设计模式

在富浏览器应用中,由于 JavaScript 灵活性的特点使得应用程序前端变量的定义和函数的编码无法统一<sup>[16]</sup>,大量用户请求代码及视图渲染代码运行于浏览器端,使得浏览器解析效率降低,页面加载速度慢。然而,服务器端分层实现可重用性及可维护性的原则同样适用于浏览器端,浏览器端采用 MVC 模式分层有助于前端程序的开发与维护。

MVC 模式(Model-View-Controller)即:模型-视图-控制器,强调把复杂的 Web 应用系统分成模型、视图和控制器三个层次进行开发<sup>[17]</sup>,视图对应于应用

系统中的用户界面，负责接收用户提交的请求信息以及对后台处理的结果进行展示，从而满足用户的访问请求；模型对应于应用项目中的数据和业务逻辑处理，控制器对应于应用系统中的输入输出控制、用户请求处理及数据显示的同步，每个核心部件各自处理各自的业务，如此各层之间边界划分清晰，任务分配明确，各尽其职，降低了处理数据和显示数据间的耦合性<sup>[18]</sup>。MVC 模式是一种非常优秀的软件设计模式<sup>[19]</sup>，目前在 B/S 模式的 Web 程序设计中被广泛采用。

**模型（Model）：**在 Web 应用后端 MVC 框架设计中，模型层负责实现对用户请求数据的校验及对数据库数据进行持久化操作<sup>[20]</sup>，而在前端框架中模型用于封装用户请求数据，以某种数据格式如 XML 格式或 JSON 格式，发送给应用后端服务器处理程序，并接收服务器返回结果传递到视图层，业务模型是前端 MVC 框架的核心部分。不同视图可以接收从同一模型传递的数据，以不同的风格个性化地展示给用户，多个视图重用同一模型代码，使得开发人员减少代码的重写工作，提高代码的可重用性。

**视图（View）：**接收模型层返回的结果数据，通过 JSP 或 HTML 页面的形式将数据显示在浏览器中<sup>[21]</sup>。模型接收用户提交的数据请求，经过验证后交由模型业务逻辑层进行处理，然后把业务逻辑处理的数据结果以某种格式显示在页面上。当模型数据发生变化时，视图也会把这些改变反映到展示的页面上。视图的功能和目标是方便用户可以方便友善地与数据库数据进行交互。

**控制器（Controller）：**控制用户请求的流转，起到指挥官的作用<sup>[22]</sup>，它监听用户请求，并把请求交给相应的业务模型进行处理，然后选择对应的视图将模型业务处理后的结果数据显示出来。控制器不做任何输入输出的工作，只是协调各层各模块合理协同调度。模型层、视图层和控制器相互之间的调用关系如图 2-1 所示：

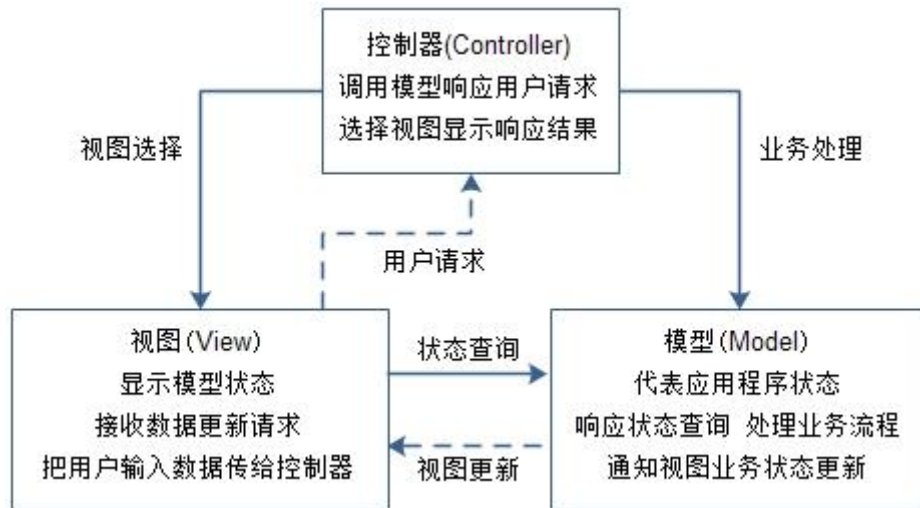


图 2-1 MVC 关系图

MVC 各层协作过程：

- 1) 用户对页面元素进行操作，页面元素状态发生改变。
- 2) 控制器监听到用户操作事件，调用相应的模型来处理这个请求。
- 3) 模型对请求数据进行序列化并向服务器发送请求，服务器完成业务逻辑处理，将最终的处理结果再返回模型。
- 4) 控制器把模型得到的数据返回给用户界面，最终结果就呈现在用户眼前。

## 2.3 REST 架构

REST (Representational State Transfer) 表述性状态转移是一种描述架构样式的网络系统。REST 架构是 Web 应用开发过程中的一种崭新的思维方式，根据需求设计合理的 URL<sup>[23]</sup>，通过 URL 来设计系统结构。符合 REST 风格的客户端与软件系统之间的连接是无状态保持、可分离的，请求双方的状态相互透明，每一次请求的发起与响应都必须包含当时的状态信息，一次服务结束双方失去连接。

应用程序提供的每一个资源都使用统一的 URI 进行标记，表面上每一段 URI 都定义为一个服务实体，URI 的设计以名词为核心，采用目录结构的方式<sup>[24]</sup>，能够反映资源的层次性和模块性。例如对于一个 blog 博客应用：URI 的定义可以表示成 `http://localhost:8080/blog`，`http://localhost:8080/blog/articles` 可以用于表示博客网站上发布的所有文章，`http://localhost:8080/blog/articles/1` 用于表示数据库中 id 为 1 的文章。

REST 基于 HTTP 网络协议，对资源的操作通过 HTTP 协议来进行<sup>[25]</sup>。HTTP

上的操作通常使用 POST、DELETE、GET、PUT 四种请求方式，分别对应资源的创建、删除、查询及更新操作<sup>[26]</sup>。目前，大部分 Web 应用系统中采用 POST 或 GET 完成资源的请求，忽略了 HTTP 最初的应用协议设计理念，REST 对 HTTP 协议的设计初衷作了诠释，对资源的增删查改操作与 HTTP 协议操作相对应：

- 1) 使用 HTTP POST 方法去创建资源
- 2) 使用 HTTP GET 方法去读取资源
- 3) 使用 HTTP PUT 方法去更新资源
- 4) 使用 HTTP DELETE 方法去删除资源

因此，REST 把 HTTP 对一个 URL 资源的操作限制在 GET、POST、PUT 和 DELETE 这四个方法之内。这种针对网络应用的设计和开发方式，可以降低开发的复杂性，提高系统的可伸缩性。

## 2.4 Web 前端相关技术

DOM 将页面中的元素和内容组织为一个易读的树状模型，用于获取和操作 HTML 文档中的任意元素或内容；JavaScript 是前端开发的基础，通过 DOM 访问和操作页面上的所有元素，完成用户请求与服务器交互；jQuery 框架对 JavaScript 方法进行封装，解决了不同浏览器解析 JavaScript 的兼容性问题，为前端开发提供了一个兼容主流浏览器的统一接口；AJAX 实现对页面中局部区域的动态刷新，使得用户能够以更好的方式获取最新的数据信息；JSON 用于定义前后端数据的请求响应格式，其数据储存量小且简洁易读。

### 2.4.1 DOM 文档对象模型

DOM 即文档对象模型，是一种描述结构化的文档与浏览器脚本之间访问和交互的 Web 标准。它实际上把浏览器支持的文档，如 XML 文档、HTML 文档、XHTML 文档当作对象来解析<sup>[27]</sup>。DOM 实际上是操作页面文件里面内容的 API，允许开发人员从文件中读取、查询、更改、添加和删除数据。DOM 对象对语言平台没有特别的要求，只要支持 DOM 的获取和解析，都能够用来操作文档。

DOM 是浏览器与页面内容结构之间沟通的接口，定义了一系列对象、方法和属性，用于访问、操作和创建文档中的内容、结构、样式以及行为<sup>[28]</sup>。每一个

网页元素都对应着一个对象。网页上的标签是一层层嵌套的，最外面的一层是<HTML>，文档对象模型也是一层层嵌套的，但是通常被理解成一棵树的形状，树根是 window 或 document 对象，相当于最外层的标签的外围，也就是整个文档。树根之下是子一级的对象，子对象也有它自己的子对象，除了根对象以外，所有的对象都有自己的父对象，同一对象的子对象之间就是兄弟的关系。通过 JavaScript 操作 DOM 对象，可以修改、增加、删除或重新排版页面上的元素，达到重构整个 HTML 页面的目的，图 2-2 为 HTML DOM 树结构。

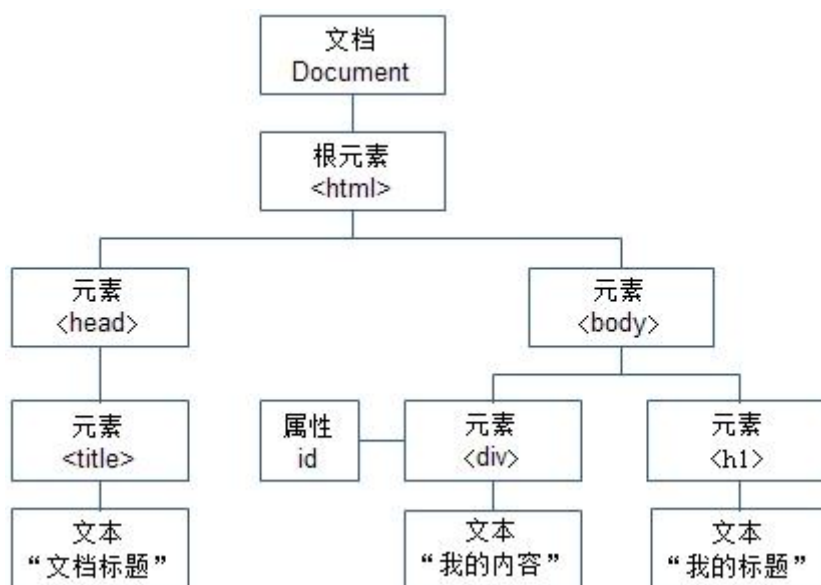


图 2-2 HTML DOM 树

在本文前端框架中，视图通过操作页面 DOM 节点对象，将从服务器返回的响应数据以特定的形式显示在页面上，完成用户数据展示和更新相关的功能。

## 2.4.2 JavaScript 技术

JavaScript 是一种基于客户端浏览器的网页脚本语言，它可以直接和 HTML 文档整合，其编译过程不需要专门的编译器，通过浏览器进行逐行解释和执行<sup>[29]</sup>。JavaScript 监听浏览器用户事件，调用对应的事件方法，接收返回结果并更新页面，实现 Web 应用的动态性和交互性，同时在浏览器端解析请求的结果数据，使得 Web 服务器的资源消耗得以降低。

JavaScript 的特点可以概括为：

1) 是一种跨平台解释性语言，语法基本结构形式与 Java 比较相似。

JavaScript 与 HTML 页面整合在一起，浏览器加载页面的过程中逐行解析

JavaScript 代码。

2) 是一种面向对象的语言，能运用其已经创建的对象，比如日期、数学函数等。

3) 是一种安全性语言，通过浏览器编译实现动态交互和信息浏览，用户不能对网络文档进行任意的修改和删除，从而可以有效地防止非法访问系统和丢失数据。

4) 是一种事件驱动脚本语言，事件指用户在页面进行某种操作时的动作，如点击鼠标，选择菜单等，事件驱动指发生事件时执行对应的响应脚本。

JavaScript 是 Web 应用前端开发中一项最基础的技术，使用 JavaScript 使得用户和信息之间不仅仅是浏览和显示的关系，用户参与到请求交互过程中，能够实时动态地获取交互信息。

### 2.4.3 jQuery 框架

jQuery 是一个免费开源并且跨浏览器的 JavaScript 框架，其核心设计理念是写更少的代码，做更多的事情（Write Less Do More）<sup>[30]</sup>。jQuery 框架改变了原生 JavaScript 代码的书写方式，降低了使用 JavaScript 操作网页的复杂度，提高了网页 JavaScript 开发效率，同时能将 JavaScript 代码和 HTML 代码完全分离，便于代码的维护和修改。jQuery 的优点可以概括为以下几点：

1) 轻量级脚本，较小的体积不仅在带宽上有较大的优势，同时代码高效简洁，具有强大高效的选择器，能够满足开发者对元素的分类选择需求，使得开发者更轻松地编写遍历操作 DOM 元素的代码。

2) 兼容浏览器，jQuery 具有高级的选择器，封装了 JavaScript 处理函数<sup>[31]</sup>，简化了 JavaScript 编程，开发人员不需要检查浏览器的类型，不用担忧浏览器的差异，可以专注于功能的开发和实现。

3) 有效支持 Ajax，对 Ajax 异步调用实现细节进行封装，提供 Ajax 请求和响应对象，为开发者提供完善的 API 调用接口函数<sup>[32]</sup>，使得 Ajax 的应用体验上了一个新的台阶。

jQuery 提供了一套易于使用的 API，这些 API 极大地简化了客户端浏览器编程过程中的许多方面，包括 HTML DOM 的遍历与操作，浏览器事件处理，Ajax 编程等。



## 1. HTML DOM 操作

jQuery 的选择器非常强大<sup>[33]</sup>，“\$”是 jQuery 核心函数 jQuery 的别名，它不仅支持 DOM 的选择方式，也支持 CSS 选择器，通过选择器使得操作 HTML 元素更加简单方便。

查找节点：var \$td = \$("tr td:eq(2)"); 查找<tr>里第三个<td>节点

创建节点：\$("ul").append(\$li\_1).append(\$li\_2); 创建两个元素节点

插入节点：\$("p").append("<b>Hello</b>"); 向元素的内部附加内容

删除节点：\$("ul li:eq(1)").remove(); 删除<ul>里第二个<li>节点

遍历节点：\$.each( [0,1,2], function(i, n){ alert( "Item #" + i + ": " + n ); });

## 2. 浏览器事件处理

jQuery 的事件模型统一了事件的属性和方法，可以在一个事件类型上添加多个事件处理函数，可以一次添加多个事件类型的事件处理函数，提供了统一的事件封装、绑定、执行、销毁机制。

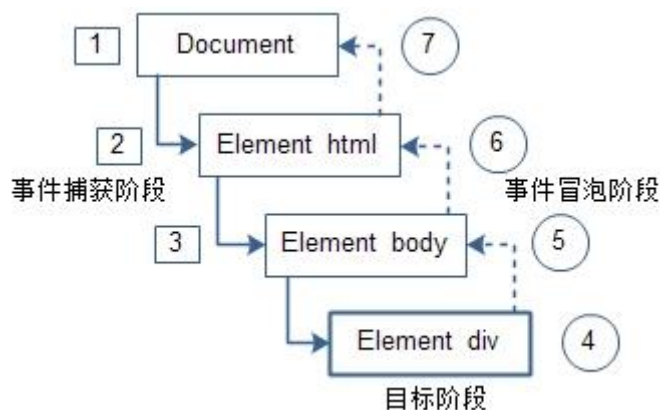


图 2-3 DOM 事件流程图

事件传送过程如图 2-3 所示，可以分为 3 个阶段<sup>[34]</sup>：

1) 事件捕捉阶段，事件对象从目标节点的祖先节点沿着 DOM 树向下传送，到达目标节点的父节点。例如，用户点击页面中的按钮，点击事件将从文档节点传送到 html 节点、body 节点以及包含该按钮的 div 节点。

2) 目标阶段，事件对象到达目标节点，目标节点的事件监听器在事件对象到达时进行处理。例如，用户点击页面中的按钮，那么该按钮就是目标节点。

3) 冒泡阶段，事件对象从目标节点开始，沿着 DOM 树向上传送，经过祖先节点到达文档节点。

jQuery 封装了 JavaScript 中的 addEventListener/attachEvent 事件处理方法，

构建了新的事件绑定方法 bind()、live()、delegate()、on()等。

### 3. Ajax 编程

jQuery 框架拥有完整的 Ajax 兼容套件,包括\$.ajax(options)实现把远程数据加载到 XMLHttpRequest 对象中, \$.get(url,data,callback,type) 使用 HTTP GET 来加载服务器数据, \$.post(url,data,callback,type) 使用 HTTP POST 来请求服务器数据<sup>[35]</sup>。通过 jQuery AJAX, 使用 HTTP GET 和 HTTP POST 请求, 可以从远程服务器请求 TXT、HTML、XML 或 JSON 数据, 允许用户在不刷新浏览器页面的情况下向服务器请求并加载数据。

基于 jQuery 框架能够方便地处理 HTML 文档、事件, 为网站提供 Ajax 交互, 本框架以 jQuery 作为基础, 减少了由于浏览器兼容性导致的工作量, 重点实现框架及优化框架性能。

#### 2.4.4 AJAX 技术

在传统互联网应用程序中, 用户在使用时的运行模式是: 用户提交请求信息, 等待服务器返回的响应, 页面进行刷新重新显示。传统的 Web 技术中刷新页面必须完全刷新, 在页面刷新之前, 用户不能对页面再执行任何操作。

Ajax 是异步的 JavaScript 和 XML, 其应用使得页面交互不再是“点击-等待-页面刷新”的访问模式, 可以在不刷新页面的情况下向后台服务器进行异步数据通信<sup>[36]</sup>, 当接收到服务器响应数据后对部分页面视图进行更新。

JavaScript 对象 XMLHttpRequest 是 Ajax 异步请求的核心<sup>[37]</sup>, 其请求过程是首先在浏览器端创建一个 XMLHttpRequest 对象, 由于不同浏览器之间存在差异, 所以创建 XMLHttpRequest 对象需要不同的方法。然后检查 XMLHttpRequest 对象的整体状态并且保证数据已经发送完毕, 根据服务器设定询问请求状态, 如果一切准备就绪, 执行 open()和 send()方法, open()方法中指定了向服务器提交数据的类型、请求的地址和参数及传输方式, send()方法用来向服务器发送请求。服务器将响应数据返回, XMLHttpRequest 对象获取数据并通过 DOM 节点树的属性和方法解析数据。Ajax 工作原理如图 2-4 所示:

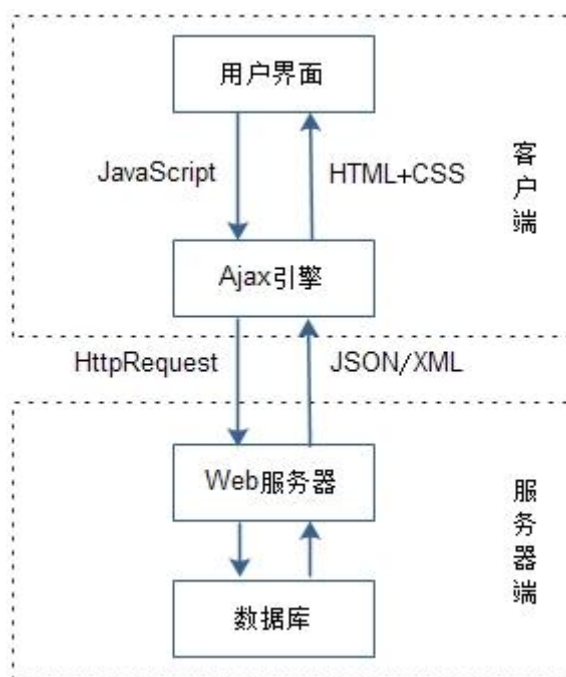


图 2-4 Ajax 工作原理

采用 Ajax 请求的应用程序具有以下优点：

- 浏览器和服务器之间减少了不必要的数据传输，降低了带宽的占用；
- Ajax 异步请求模式有助于提升用户体验；
- Ajax 引擎运行于浏览器端，承担了一部分本来由服务器完成的工作，降低了服务器处理负载。

本文前端框架采用 Ajax 引擎实现模型对象向服务器发送请求并接收服务器响应，降低了由于静态数据的传输产生的数据流量，同时用户请求之间不需要等待，提高页面体验效果。

#### 2.4.5 JSON 数据格式

JSON 即 JavaScript 对象表示法（JavaScript Object Notation），是存储和交换文本信息的格式，是基于 JavaScript 的一个子集，提供了一种具有嵌套数据元素的结构<sup>[38]</sup>。在 JSON 格式出现以前，XML 一直是前后台异步数据交换的主要数据格式，相比于 XML 严谨规范的编码格式，JSON 代码具有更简单、灵活可读性的特点。

在 Web 开发方面，JSON 采用“名称/值”键值对的形式表示一对数据<sup>[39]</sup>，一个 JSON 对象以“{”（左括号）开始，“}”（右括号）结束，其格式可表示为

{ "firstName": "Tom", "lastName": "Smith" }。不同的语言中, 这种形式被看作对象、字典、哈希表、关联数组或者有键列表, XML 使用重复的开始标记和结束标记包装数据值, 其格式表示为<result><node firstName="Tom" lastName="Smith"></node></result>, 与相应的 JSON 格式相比, 产生了两倍的元数据开销。在基于 MVC 的前端框架的实现上, 前后端采用 JSON 格式的数据, 方便对数据的处理同时减少了不必要的数据流量。

## 2.5 本章小结

本章中对本文所涉及到的 MVC 设计模式、REST 架构风格、DOM 文档对象模型、JavaScript 技术、jQuery 框架、AJAX 异步请求技术、JSON 数据交换技术分别进行了详细的介绍, 为前端 MVC 框架的开发提供技术支持。

### 3 基于 MVC 模式的前端框架设计

#### 3.1 前端 MVC 框架 Frontbase 设计

本文实现的前端框架以 Frontbase 命名，所有的属性和方法都定义在命名空间之内，遵循面向对象的封装性。Frontbase 框架将前端的交互按照功能划分为模型、控制器、视图三层，如图 3-1 所示。

模型层用来存放 Web 应用的所有数据对象，定义交互式数据对象及属性，负责数据对象的创建、销毁、验证和存储等。

控制器是连接视图层与模型层之间的桥梁<sup>[40]</sup>，控制器的功能是监听视图事件并创建对应的模型对象，将数据传送到数据模型，并接收模型对象的数据传送给视图对象。

视图层负责页面展示逻辑，接收模型数据并通过模板引擎将数据和 HTML 片段整合进行显示。

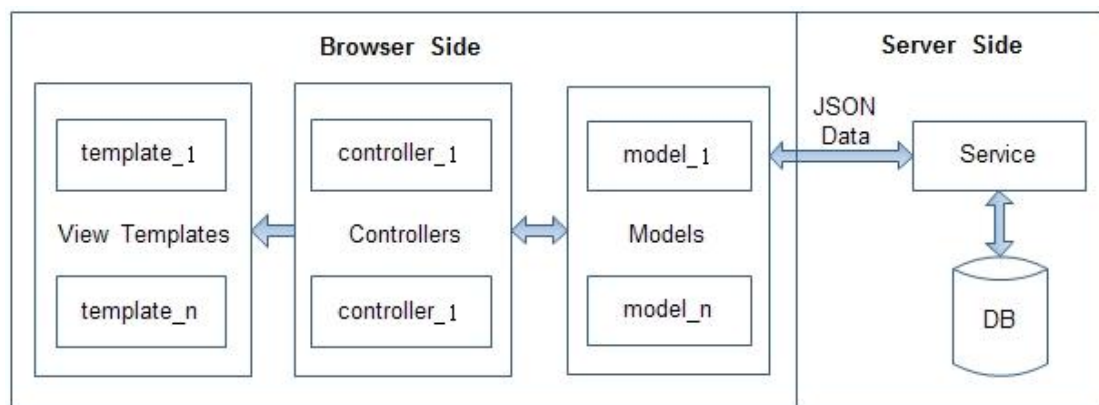


图 3-1 前端 MVC 架构图

当界面上的操作引起模型对象中属性的变化时，视图层的监听控制器接收变化数据，将数据发送给模型对象，模型对象向服务器发送请求，模型对象接收到服务器的返回数据后，通知视图重新渲染新的数据到页面<sup>[41]</sup>。在采用前端 MVC 框架的应用程序中，不需要从 DOM 中通过特殊的 id 来获取节点，或者手动更新 HTML 页面，在模型对象发生变化时，相应的视图对象会根据更新后的数据自动渲染页面。

## 3.2 前端 MVC 框架 Frontbase 对象

前端 MVC 架构 Frontbase 的设计主要涉及五个对象，如图 3-2 所示：

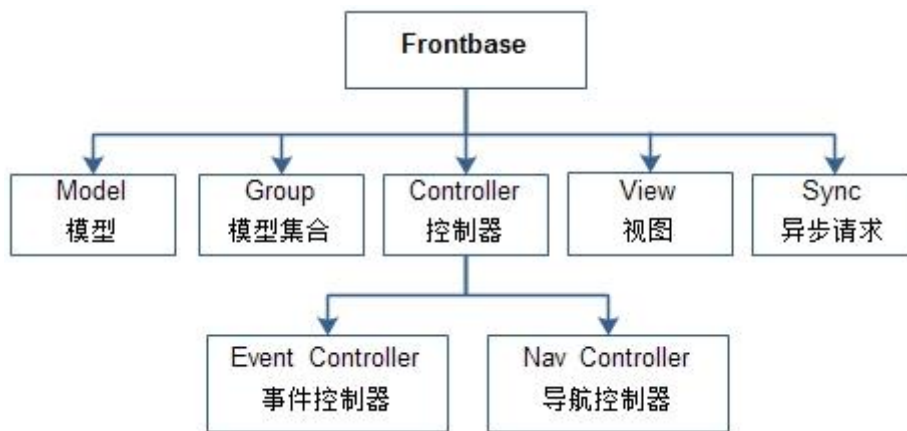


图 3-2 Frontbase 框架对象

Model 是模型对象，定义数据模型相关属性、构造函数及原型扩展；

Group 是模型对象的集合，定义模型集合构造函数及原型扩展；

Controller 是控制器对象，用于自定义事件，拥有 Model 对象的调度和 View 对象的控制；

View 是视图渲染对象，定义视图构造函数和原型扩展；

Sync 是异步请求对象，用于向服务器发送请求同步数据状态，建立与服务

### 3.2.1 模型对象

Model 是 Frontbase 架构中所有数据模型的基类，用于封装原始数据，并提供对数据进行操作的方法。Model 就像是映射的一个数据对象，它可以对应到数据库中的某一条记录，并通过操作对象，将数据自动同步到服务器数据库。每个模型对象都有一个唯一标识(id)，它与数据库中记录的 id 保持一致，同时，每个模型对象内部还会自动创建一个 cid 属性，用于标识每个模型。模型类包含一个 extend() 静态方法用于实现子类继承，在定义好一个模型类之后，通过 new 关键字创建该模型的实例对象，如果模型类在定义时设置了 defaults 默认数据，这些数据将被复制到每一个实例化的对象中。模型将原始数据存放在对象的 attributes 属性中，可以通过 attributes 属性直接获取和操作这些数据，对于实例化的模型对象，还可以通过 get() 和 escape() 方法获取模型中的数据，通过 set() 方法修改模

型中的数据,通过 `previous()`方法获取模型数据修改前的状态,通过 `unset()`或 `clear()`方法删除模型中的数据。图 3-3 为模型对象图

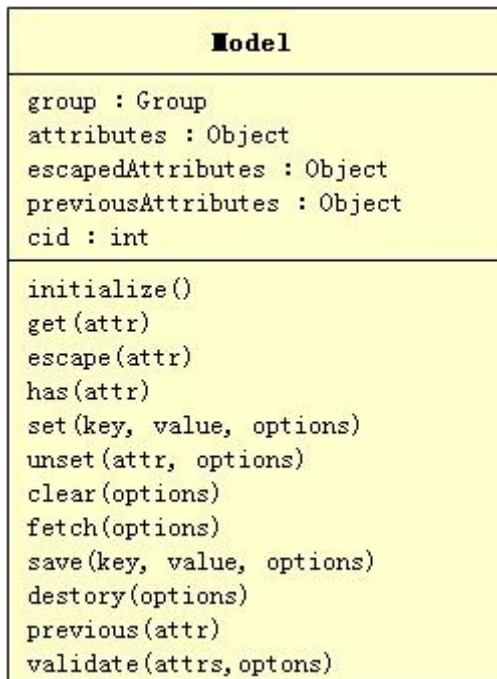


图 3-3 模型对象图

除了对模型数据的增删查改操作, **Model** 还提供了数据验证方法 `validate()`方法,该方法在模型数据发生改变之前被自动调用,当验证不通过时,会将模型对象和 `validate()`方法的返回值传递给 `error` 事件的监听函数。

### 3.2.2 集合对象

如果将模型对象比喻成数据库中的一条记录,那么 **Group** 就对应数据库中的一张数据表,它表示模型的集合类,用于存储和管理一系列相同类型的模型对象。对集合中的模型进行操作包括以下方法:

- 1) 动态地向集合中插入模型,包括 `add()`向集合中的指定位置插入模型, `push()`将模型追加到集合底部, `unshift()`将模型插入到集合头部;
- 2) 从集合中移除模型对象,包括 `remove()`从集合中移除指定的模型对象, `pop()`移除集合尾部的模型对象, `shift()`移除集合头部的模型对象;
- 3) 从集合中快速查找模型的方法,包括 `get()`根据模型的唯一标识查找模型对象, `getByCid()`根据模型的 `cid` 查找模型对象, `at()`查找集合中指定位置的模型对象, `where()`根据数据对集合的模型进行筛选。



集合对象图如图 3-4 所示：

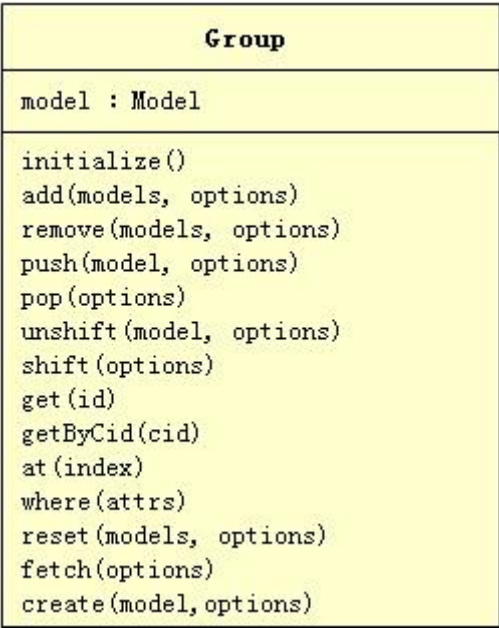


图 3-4 集合对象图

3.2.3 控制器对象

控制器对象为模型对象和视图对象之间的枢纽，包括两种不同类型的控制器，一种是自定义事件控制器，另一种是自定义导航控制器。自定义事件控制器提供了事件管理相关的方法，通过在对象中绑定 Events 相关方法，允许向对象添加、删除和触发自定义事件。自定义导航控制器允许定义导航规则，通过 URL 片段导航，并将每个导航对应到一个方法，当 URL 匹配某个导航时会自动执行该方法。

自定义事件控制器方法包括 on()将一个函数绑定到对象的某个事件中，off()函数移除对象某个事件中已绑定的函数，trigger()函数触发对象的某个事件，图 3-5 为事件控制器对象图



图 3-5 事件控制器对象图



在创建自定义导航控制器实例时，通过配置 `navs` 属性来设置某个导航规则对应的监听方法，主要方法包括 `nav()` 将一个导航规则绑定给一个监听事件，当 URL 片段匹配该规则时，会自动调用触发该事件，`bindNavs()` 解析当前实例定义的导航规则，并调用 `nav()` 方法将每一个规则绑定到相应的方法上，`navToRegExp()` 方法将字符串形式的导航规则转化为正则表达式对象，对字符串中特殊字符添加转义符，防止特殊字符在被转换为正则表达式后转变为元字符，返回的正则表达式根据 `navs` 字符串中的规则匹配对应的 URL 片段。图 3-6 为导航控制器对象图

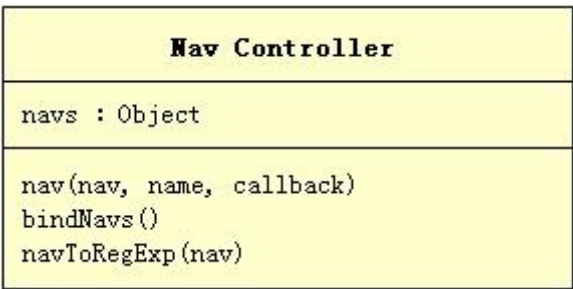


图 3-6 导航控制器对象图

3.2.4 视图对象

视图对象用于将从服务器获取的数据渲染到页面，以及管理页面事件和逻辑。视图对象继承自 `Frontbase.View`，并通过 `new` 关键字实例化一个视图对象，每个视图对象都会关联一个 DOM 对象，视图中所有操作都限定在这个 DOM 对象之内，便于视图界面的控制（如渲染、隐藏和移除等），同时能够提高查找视图内子元素的效率。DOM 对象通过 `tagName` 属性、`className` 属性或 `id` 属性进行选择，`tagName` 表示新标签的名称，`className` 表示标签的 `class` 样式属性，`id` 表示标签的 `id` 属性。视图对象通过 `render()` 方法处理渲染模板，实现渲染逻辑。视图对象图如图 3-7 所示：

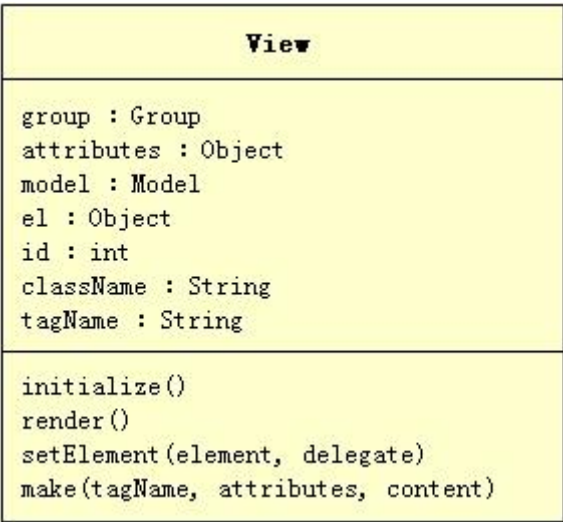


图 3-7 视图对象图

3.2.5 异步请求对象

异步请求对象用于模型和集合与服务器进行连接，实现与服务器的数据交互，其将 Ajax 异步请求方法封装为 sync()方法，通过\$.ajax 方法发送请求，服务器响应数据采用 JSON 的数据格式，图 3-8 为异步请求对象图。

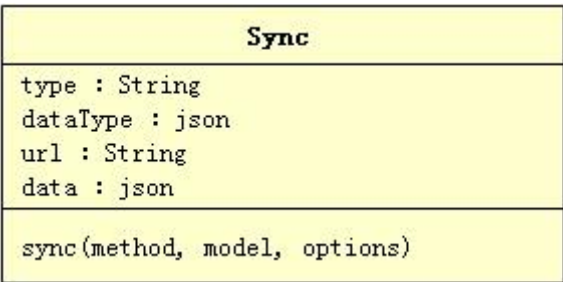


图 3-8 数据同步对象图

3.3 前端 MVC 框架 Frontbase 工作流程

前端框架工作流程为：

- 1) 用户行为导致页面导航 URL 发生变化，导航控制器 Nav Controller 根据 navs 属性中 URL 的配置信息匹配相应的事件处理方法；
- 2) 事件控制器捕捉用户行为；
- 3) 创建模型对象实例或集合对象实例；
- 4) 模型调用服务器异步请求类方法，向服务器发送请求数据；
- 5) 模型实例接收从服务器返回的数据，调用相应的视图对象进行视图更新；

6) 视图通过 template 加载模板引擎构建视图；

7) 视图对象渲染完毕后呈现给用户。

前端框架工作流程图如图 3-9 所示：

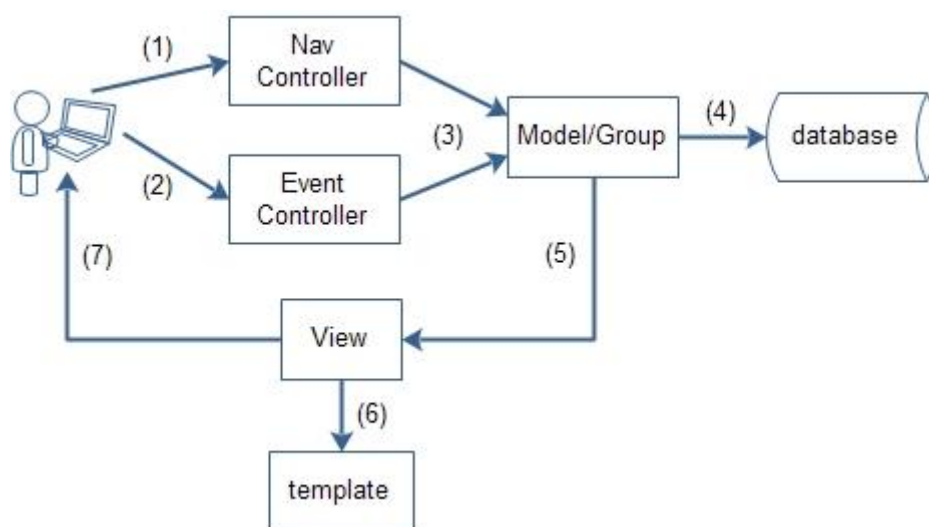


图 3-9 Frontbase 工作流程图

### 3.4 本章小结

本章描述了前端 MVC 框架中模型层、视图层、控制器层各个层的功能，并对前端框架中涉及对象的属性和方法进行详细设计，然后介绍了基于 MVC 前端框架的工作流程。



## 4 基于 MVC 模式的前端框架实现

### 4.1 模型层的实现

#### 4.1.1 初始化模型

模型对象结构代码如下：

```
var Model = Frontbase.Model = function(attributes, options){ }  
_.extend(Model.prototype, Events, {..})
```

Model 构造器是前端框架中所有数据对象模型的基类，用于创建一个数据模型，extend 方法为 Model 原型定义了一系列属性和方法。

Model 构造器中定义了一系列的属性，attributes 用于存储当前模型的 JSON 对象化数据，初始化时设置为模型默认数据与 attributes 参数合并后的数据，每个模型都配置了一个唯一的标识 cid，previousAttributes 存储模型数据的一个副本，在触发事件时获取模型数据被改变之前的状态，可以通过 previous 或 previousAttributes 方法获取上一个状态的数据。escapedAttributes 缓存对象缓存通过 escape 方法处理过的数据，changed 属性记录了每次调用 set 方法时被改变数据的 key 集合，group 属性显示指定模型所属的 Group 对象，在调用 Group 对象的 add、push 方法将模型添加到集合中的方法时，自动设置模型所属的 Group 对象。

创建数据模型代码如下：

```
var Student = Frontbase.Model.extend({ //定义 Student 模型类  
  defaults:{  
    name: " <a href=\"#\>Lucy</a>",  
    age: 20  
  }  
});  
var Bob = new Student ({ //实例化 Student 类  
  name: " <a href=\"#\>Bob</a> ",  
  age: 21  
});
```

### 4.1.2 操作模型数据

#### 1) 读取数据

获取模型数据的方法有两种，即 `get()`和 `escape()`。所有的模型数据都存储在 `attributes` 属性中，`get()`方法通过 `attributes[attr]`直接返回数据，`escape()`方法用于返回包含 HTML 字符的数据，将 HTML 字符串中的特殊字符如`&`、`<`、`>`等转换为 HTML 实体，防止跨站脚本攻击<sup>[42]</sup>。`escape()`方法处理的数据被缓存在 `escapeAttributes` 对象中，首先从缓存对象中查找数据，如果数据已经被缓存则直接返回，如果缓存对象中没有找到数据，则先调用 `get()`方法从模型中获取数据，将数据中的 HTML 特殊字符转换为 HTML 实体，并缓存到 `escapeAttributes` 对象中便于下次直接获取。

#### 2) 修改数据

`set()`方法用于修改模型中的数据，如果属性 `key` 值不存在，则将新的属性添加到模型，如果 `key` 值已经存在，则更新对应的 `value` 值。`set()`方法实现原理如图 4-1 所示：



图 4-1 `set()`方法执行顺序

`set()`方法中参数允许 `key-value` 对象形式，或通过 `key`、`value` 两个参数进行单独设置，如果 `key` 是一个对象，则认定为使用对象形式设置，第二个参数将被视为 `options` 参数；如果 `key`、`value` 两个参数单独设置，将数据放到 `attrs` 对象中统一处理，即 `attrs[key] = value`；如果被设置的数据对象属于 `Model` 类的一个实例，通过 `Model` 对象的 `attributes` 数据对象赋值给 `attrs`，表示为 `attrs =`

`attrs.attributes`。

对属性和属性值进行初始化后,调用 `validate()` 方法,验证属性值是否符合校验规则,如果验证不通过则停止执行,如果验证通过,遍历需要设置的属性数据对象,获取每个属性对应的属性值,如果当前属性在模型中不存在,或在 `options` 中指定了 `unset` 属性删除,则删除该属性被缓存在 `escapeAttributes` 中的数据,保证缓存中的数据与模型中的真实数据保持同步,如果模型中属性值与新的属性值不一致,表示该属性数据发生变化,在 `changed` 属性中记录当前属性已经发生变化的状态,同时更新模型属性值。

### 3) 删除数据

删除模型数据有两种方法 `unset()` 和 `clear()`。`unset()` 方法用于删除当前对象中指定的属性和数据,通过设置 `options.unset` 属性告知 `set()` 方法进行删除操作。`clear()` 方法用于清除模型中所有的属性和数据,首先克隆一个当前模型属性的副本 `clone(model.attributes)`,然后将 `unset` 设置为 `true`,调用 `set()` 方法执行删除操作。

### 4) 同步数据

模型对象中提供了三个方法用于和服务器保持数据同步,`save()` 方法用于将模型的数据保存到服务器,可能是增加一条新的数据,也可能是修改数据库现有的一条数据,取决于模型中是否存在唯一标识 `id`; `fetch()` 方法用于从服务器接口获取模型的默认数据, `destroy()` 方法用于将数据从服务器中删除。

模型通过 Ajax 交互与服务器进行通信,模型层同步方法通过 REST JSON API 异步将当前状态保存到服务器,每个 `create`、`read`、`update` 或 `delete` 活动均会与各种不同类型的 HTTP 请求相关联,模型请求方式 `method` 和服务器交互方法 `type` 的对应关系如下:

```
var methodMap = {  
    'create': 'POST',  
    'update': 'PUT',  
    'delete': 'DELETE',  
    'read': 'GET'  
}
```

`save()` 方法首先对需要保存到服务器的数据对象进行格式化,并保存到属性

attrs 中，和 set()方法设置参数的方法相同，将修改过最新的数据保存到模型中，便于在异步请求方法 sync()中获取模型数据保存到服务器；然后设置服务器响应成功后的回调函数，将最新的数据状态设置到模型中；最后调用 Frontbase.sync()方法将模型中的数据保存到服务器，如果当前模型是一个新建的模型，则向 sync()方法传递参数为 create，如果修改当前模型数据，则向 sync()方法传递参数为 update。

fetch()方法通过设置 method 参数为 read 类型，调用 Frontbase.sync()方法，从服务器获取数据，获取数据成功后调用自定义回调函数，在回调函数中通过 set()方法将服务器返回数据填充到模型中。

destroy()方法删除模型分为两种情况，如果模型是在浏览器端新建的，则直接从浏览器端删除，如果模型数据同时存在于服务器，则同时删除服务器端的数据。通过委托 HTTP DELETE 请求到 Frontbase.sync()销毁服务器上的模型，在模型上触发 destroy 事件，可以通过任意包含它的集合向上冒泡，从而删除集合中的模型，

#### 5) 验证数据

validate()方法用于模型数据验证，在 set()和 save()方法之间调用，该方法包含 2 个参数，分别为模型中的数据集合与配置对象，在模型中需要重写 validate()方法来包含模型的有效逻辑，如果验证通过则不返回任何数据，验证失败返回带有错误信息的数据。

### 4.1.3 初始化集合

集合块结构代码如下：

```
var Group = Frontbase.Group = function(models, options) {}  
_.extend(Group.prototype, Events, {..})
```

Group 集合构造器是前端框架所有集合对象的基类，在配置参数中设置集合的模型类，用于创建多个模型的集合，通过 extend 方法定义集合类原型方法。

创建集合代码如下：

```
var StudentList = Frontbase.Group.extend({  
    model: Student  
});
```



```
var Cidy = new Student ({name:" Cidy ", country: "America"});  
var Penny = new Student ({name:" Penny ", country: "America"});  
var Student List = new StudentList([Cidy, Penny]);
```

#### 4.1.4 操作集合模型

##### 1) 向集合中添加模型

Group 提供三个方法动态地向集合中插入模型。add()方法向集合中的指定位置插入模型，push()方法将模型追加到集合尾部，unshift()方法将模型插入到集合头部。

add()方法操作的一系列的模型对象必须是以数组的形式表示的，如果只传入一个模型，则将其转换为数组，首先遍历需要添加的模型列表，遍历过程中将数据对象转换为模型对象，建立模型与集合之间的引用，然后遍历需要添加的模型，将模型根据 cid 记录到 byCid 对象，根据 id 记录到 byId 对象，用于在调用 get() 和 getBycid()方法时作为索引，最后设置新模型列表插入到集合的位置，如果在 options 中设置了 at 参数，则在集合的 at 位置插入。

push()方法通过调用 add()方法将模型对象插入集合尾部，unshift()方法通过设置 at 属性值为 0，将模型对象插入到集合的第一个位置。

##### 2) 从集合中查找模型

byId 对象中的模型是根据 id 记录的，通过调用 get(id)方法可以查找 byId 对象中的模型，byCid 对象中的模型是根据 cid 记录的，通过调用 getByCid(cid)方法查找 byCid 对象中的数据模型。

##### 3) 删除集合中的模型

Group 提供三个方法用于从集合中移除模型对象，remove()从集合中移除指定的模型对象，pop()移除集合尾部的模型对象，shift()移除集合头部的模型对象。

remove()方法参数 models 必须是数组类型，遍历需要移除的模型列表，调用 getByCid()或 get()方法获取模型，在 byId 对象中移除模型的 id 引用，在 byCid 对象中移除模型的 cid 引用，调用数组的 indexOf()方法找到模型的位置，然后使用数组的 splice()方法将该模型从集合列表中移除，同时重置当前集合中模型的数量。

pop()方法首先获取集合中的最后一个模型 models[length-1]，然后调用

`remove()`方法移除模型。`shift()`方法和 `pop()`相反，首先获取集合的第一个模型 `models(0)`，然后调用 `remove()`方法移除模型。

#### 4) 从服务器获取集合数据

`fetch()`方法实现从服务器获取集合的初始化数据，定义属性 `group` 记录当前集合对象，调用 `Frontbase.sync` 方法发送请求从服务器获取数据，当从服务器请求数据成功时执行回调函数，如果 `options` 中设置 `add=true`，则调用 `add()`方法将数据追加到集合，否则遍历当前集合中的模型，依次删除并解除它们与集合的引用关系，将集合中的数据替换为服务器的返回数据。

## 4.2 视图层的实现

### 4.2.1 创建视图

视图对象负责一切和界面相关的工作，比如绑定 **HTML** 模板，绑定界面元素的事件，初始化渲染，模型值改变后的重新渲染和界面元素的销毁等。

视图对象创建与数据低耦合的界面控制实例，通过将视图的渲染方法绑定到数据模型的 `change` 事件，当数据发生变化时会通知视图进行渲染，视图对象中的 `el` 用于存储当前视图所需要操作的 **DOM** 最父层元素，这主要是为了提高元素的查找和操作效率，其优点包括：查找或操作元素时，将操作的范围限定在 `el` 元素内，不需要再整个文档树中搜索；在为元素绑定事件时，可以方便地将事件绑定到 `el` 元素或者是其子元素。

创建视图代码如下：

```
var StudentView = Frontbase.View.extend({
    el: "#list"
});
var studentView = new StudentView;
```

### 4.2.2 绑定事件

视图类中 `events` 属性表示用户事件列表，视图类在实例化的过程中，视图对象会自动解析 `events` 列表中的描述，使用 `jQuery` 获取选择器描述的 **DOM** 对象，并将事件处理函数绑定到事件名称中。

events 描述方式为：事件名称 选择器 ：事件处理对象。

events 中的事件是通过 jQuery 提供的 delegate()方法绑定到视图对象的 el 元素上，事件绑定在父层元素，然后在事件冒泡<sup>[43]</sup>过程中，通过检查目标子元素来触发事件。绑定事件代码如下：

```
delegateEvents : function(events) {  
    for(var key in events) {  
        var method = events[key];  
        if(!_isFunction(method))  
            method = this[events[key]];  
        var match = key.match(delegateEventSplitter);  
        var eventName = match[1], selector = match[2];  
        method = _.bind(method, this);  
        eventName += '.delegateEvents' + this.cid;  
        if(selector === "") {  
            this.$el.bind(eventName, method);  
        } else {  
            this.$el.delegate(selector, eventName, method);  
        }  
    }  
}
```

#### 4.2.3 模板引擎

Web 前端的 JavaScript 代码中经常会拼接 HTML 代码片段，既不容易维护，也不利于测试<sup>[44]</sup>，前端 MVC 框架在视图层通过 JavaScript 的替换方式实现模板引擎，将界面的 HTML 模板和服务器返回的 JSON 数据组合生成动态地 HTML 片段，也就是将 JSON 数据填充到 HTML 模板标签中，其实现对模板的编译，在首次渲染时会根据模板解析的结果生成一个用于拼接数据的函数，后续调用时直接使用这个函数而不需要再次解析模板。模板引擎工作原理如图 4-2 所示：



图 4-2 模板引擎工作原理

本框架实现了一个轻量级的模板引擎，其核心是模板解析函数 `template()`，其接收两个参数，第一个参数是用来解析模板的原始字符串，第二个是选填参数，用来立即呈现。将模板内容放到一个 `<script type="text/template">` 标签中，为了防止浏览器按照 JavaScript 代码解析，`type` 属性设置为 `text/template`，通过 `template` 模板函数可以解析三种模板标签：

- `<% %>` 用于包含 JavaScript 代码，这些代码将在渲染数据时被执行。
- `<%= %>` 用于输出数据，可以是一个变量、某个对象的属性、或函数调用（将输出函数的返回值）。
- `<%- %>` 用于输出数据，同时会将数据中包含的 HTML 字符转换为实体形式。

每个视图都有一个 `render` 方法，默认情况下没有任何操作，一旦视图需要重绘便会调 `render` 方法，不同的视图用不同功能的函数来覆盖函数，以处理模板渲染，并使用新的 `html` 来更新 `el` 选择器内容。将模板内容和需要填充的数据传递给 `template()` 方法，按图 4-3 所示流程进行处理：

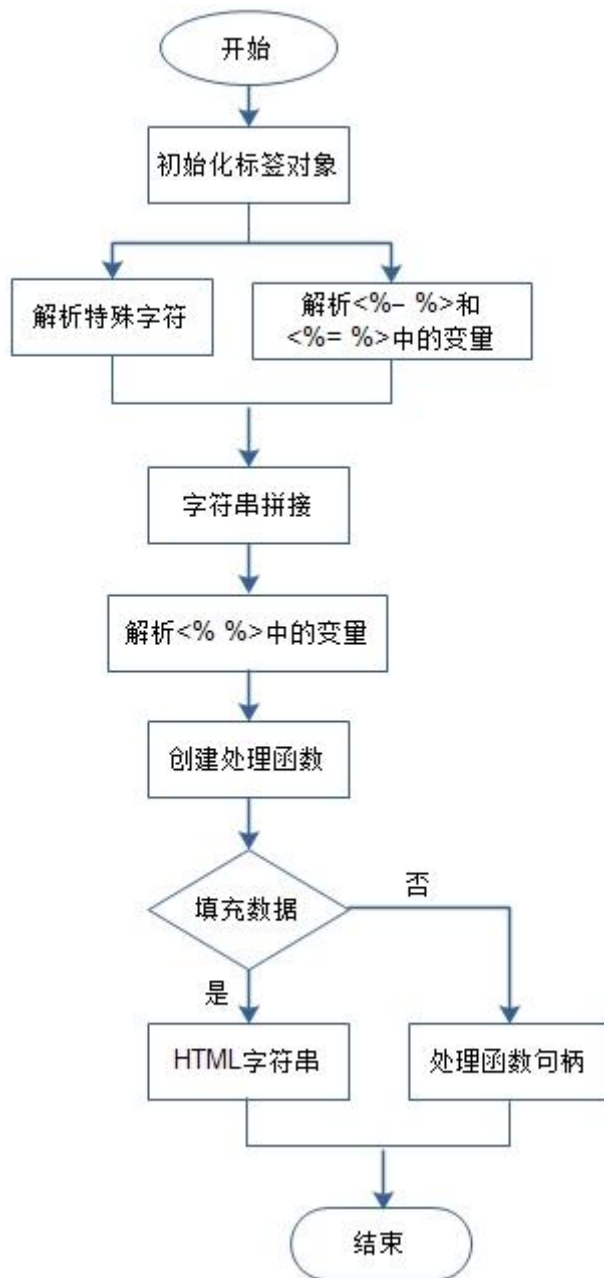


图 4-3 模板解析流程图

1) 通过正则表达式匹配 JavaScript 可执行代码的界定符，区分三种不同的模板逻辑语法标签，初始化模板标签对象。

2) 匹配模板中的特殊字符，包括单引号、换行符、回车符、制表符等，将模板中的特殊符号转换为字符串。

3) 根据模板标签对象进行模板标签的匹配，解析模板中<%- %>形式的标签，将变量中包含的 HTML 代码解析为 HTML 实体，解析模板中<%= %>形式的标签，将模板内容作为变量返回。

4) 将 2) 3) 中返回的字符串连接为新的字符串作为一个变量返回。

5) 根据模板标签对象匹配模板中<% %>形式的标签，由于<% %>存储了需要执行的 JavaScript 代码，能够被正常执行，直接作为新的一行 JavaScript 语法执行。

6) 创建一个处理函数，将 JavaScript 源码作为函数执行体，如果指定了模板的填充数据，则替换模板内容，返回替换后的结果，如果没有指定填充数据，则返回一个处理函数句柄，该函数用于接收数据替换到模板，如果程序中多次填充相同的模板，可以在第一次调用时不指定填充数据，在获得处理函数的引用后，再直接调用可以提高程序运行效率。

### 4.3 控制器的实现

前端框架中自定义事件控制器和自定义导航控制器是模型层和视图层之间的媒介，模型层改变时通知视图层更新，视图层改变时更新模型层。

#### 4.3.1 事件管理器

事件管理器是前端框架中所有其他模块的基类，为任意对象提供绑定、解析和触发自定义事件功能，无论是 Model、Group、View 还是导航管理器都继承了事件管理器的方法，事件管理的方法包括 on()、off()、trigger()，bind()方法和 on()方法相同，是绑定事件的别名，unbind()方法和 off()方法相同，是释放事件的别名。事件最基本的三个要素是事件的名称 events、回调函数 callback 和上下文 context，将自定义事件和回调函数绑定到当前对象。events 允许指定多个事件名称，通过空白字符进行分隔，回调函数中的上下文对象为指定的 context，如果没有设置 context 则上下文对象默认为当前绑定事件的对象。

事件绑定 on()方法首先通过事件名称正则表达式变量 eventSplitter 对事件名称进行解析，使用 split()方法将多个事件拆分成一个数组 events。每个模型都配置了 callbacks 变量，它是一个 key-value 键值对，其中 key 是事件名称，value 是一个对象数组，每个对象包含绑定在该事件上的一对 callback 和 context。on()方法通过定义变量 calls 记录当前对象中已绑定的事件与回调函数列表。然后循环事件名称列表，从头至尾将事件名称存放到 event 变量中，通过 list 变量记录单个 event 事件绑定的回调函数列表，函数列表通过多个对象的 next 属性进行链

式存储，回调函数列表链式存储格式如下：

```
{
  tail: {Object},
  next: {
    callback: {Function},
    context: {Object},
    next: {
      callback: {Function},
      context: {Object},
      next: {Object}
    }
  }
}
```

回调函数列表最顶层存储了一个 **tail** 对象，它存储了最后一次绑定回调事件的标识，通过 **tail** 标识，可以在遍历回调列表时得知已经到达最后一个回调函数。最后，定义存放本次回调函数相关信息的变量，重新组装当前事件的回调函数列表，按照回调函数绑定顺序排列将本次回调函数加入到函数列表中，方便进行方法链调用。

移除对象中已绑定的事件或回调函数，可以通过 **events**、**callback** 和 **context** 对需要删除的事件或回调函数进行过滤：

- 如果 **context** 为空，则移除所有的 **callback** 指定的函数；
- 如果 **callback** 为空，则移除事件中所有的回调函数；
- 如果 **events** 为空，但指定了 **callback** 或 **context**，则移除 **callback** 或 **context** 指定的回调函数；
- 如果没有传递任何参数，则移除对象中绑定的所有事件和回调函数。

遍历事件对象中的所有回调函数对象，根据参数中的回调函数和上下文，对回调函数进行过滤，将不符合过滤条件的回调函数重新绑定到事件中实现解除事件绑定方法。

**trigger()**用于触发已经定义的一个或多个事件，依次执行绑定的回调函数列表，其参数为所有事件数组。其实现通过两层嵌套循环，外层是循环需要触发的事件列表，内层循环是当前事件的所有回调函数列表，通过 **apply()**执行所有绑

定的函数，并将调用 `trigger()` 时的参数传递给回调函数。

#### 4.3.2 导航管理器

导航构造器中 `navs` 属性记录了导航规则与事件方法的绑定关系，当 URL 与某个规则匹配时，会自动调用关联的事件方法。解析和绑定导航规则 `bindNavs()` 方法实现流程如下：

1) 遍历 `navs` 属性配置，将导航规则通过数组的 `unshift()` 方法放置到一个新的数组中，按照[规则名称，绑定方法]的形式记录，实现倒序排列。这里将 `navs` 中的规则倒序排列，在后面调用 `nav()` 方法时会再次调用 `unshift` 将顺序倒过来，以保证最终的顺序是按照 `navs` 配置中定义的顺序来执行的。倒换两次顺序后，会重新恢复最初调用前的顺序，之所以这样做，是因为用户可以手动调用 `nav()` 方法动态添加导航规则，而手动添加的导航规则会被添加到列表的第一个，因此要在 `nav()` 方法中使用 `unshift` 来插入规则。

2) 循环 1) 中得到的 `navs` 导航规则，依次调用 `nav()` 方法，将规则名称绑定到具体的事件函数上。

`nav()` 将一个导航规则绑定给一个监听事件，当 URL 片段匹配该规则时，会自动调用触发该事件，它需要接收三个参数：规则名称 `nav`、事件函数名 `name`、事件函数对象 `callback`。导航规则名称是一个字符串，通过 `navToRegExp()` 方法将其转换为一个正则表达式，然后获取匹配到的规则中的参数，调用 `callback` 导航监听事件，并将参数传递给监听事件。

`navToRegExp()` 将字符串形式的导航规则转换为正则表达式对象，用于匹配 URL 片段，转换规则如下：

- 为字符串中特殊字符添加转义符，防止特殊字符在被转换为正则表达式后变成元字符（这些特殊字符包括 `-[{}() + ? . \ ^ $ # | \s`）；
- 将字符串中以 `/` 斜线为分隔的动态导航规则转换为 `([^\s/]+)`，在正则中表示以 `/` 斜线开头的多个字符；
- 将字符串中的 `*` 星号动态导航规则转换为 `(.*?)`，在正则中表示 0 或多个任意字符（例如这样的组合导航规则：`*list/:id`，将匹配 `booklist/12`，同时会将 `"book"` 和 `"12"` 作为参数传递给事件方法）。



## 4.4 服务器同步的实现

### 4.4.1 Ajax 请求调度算法

Ajax 请求是互不影响的，当用户发送一个请求，服务器处理该请求的过程中，用户不必等待响应结果，可以同时对面中其他元素进行操作，可以再次发送 Ajax 请求。用户同时或连续向服务器发送多个连接请求，如果连接请求过于频繁，服务器就会因为不断处理来自多个用户的请求而阻塞导致无数据返回，客户端将会一直等待服务器响应数据而导致无响应。为了避免服务器发生阻塞，一些主流的浏览器如 IE、Firefox、Opera、Chrome 都限制了同一时刻的最大并发连接数，当用户同时发起的连接请求数大于浏览器的并发连接限制数时，不同的请求发送顺序对应不同的用户等待时间，为了有效减少用户的请求等待时间，本框架通过设置请求优先级队列<sup>[45]</sup>的方式提出了 Ajax 连接请求快响应调度算法。

根据请求响应时间级别规定优先级顺序，0 表示请求优先级最高，3 表示请求优先级最低。

优先级为 0 的请求：页面远程加载所需文件的请求，主要是 CSS 样式布局文件及 JavaScript 事件处理文件。

优先级为 1 的请求：查询数据库初始化页面数据及其他数据查询

优先级为 2 的请求：更改数据库数据，如页面表单验证、提交

优先级为 3 的请求：页面重载方面的请求，如 URL 重定向、页面定期刷新

在 Ajax 连接请求快响应调度算法中设计一个请求优先级队列，两个目标响应队列，根据 Ajax 请求的优先级参数，将请求插入到请求队列的适当位置，高优先级的请求插入到低优先级请求之前，同优先级的请求按照队列的先进先出原则插入到同优先级请求之后。两个响应队列中的某个队列在进行 Ajax 请求时，另一个响应队列只能进行比其请求优先级高的 Ajax 请求，当不存在高优先级的请求时，才进行同级别的 Ajax 请求。由于优先级高的请求会首先被放入响应队列，较早插入到请求队列的低优先级请求可能一直处于请求队列的尾部而得不到响应，通过等待时间提升优先级的方法解决，请求等待超过 30s 没有被插入到响应队列的请求，其优先级将会被提升。

Ajax 连接请求快响应调度算法如下：

用一个数组表示请求优先级队列 reqList(req01...req0n, req11...req1n, req21...req2n, req31...req3n), 其中 reqX1...reqXn 代表四类不同优先级的请求, X 对应优先级 0-3, resList1(res11...req1n)、resList2(res21...req2n)是请求响应队列, 按照队列先进先出的策略可以同时向服务器发送 Ajax 请求。

表 4-1 Ajax 队列相关方法

方法	参数	描述
addAjaxItem()	options 请求数据 priority 请求优先级	请求优先级队列增加 Ajax 请求, 返回当前 AjaxItem 对象
changePriority()	ajaxItem AjaxItem 对象	改变 Ajax 请求对象的优先级
getAjaxItems()	priority 请求优先级	获取同一级别的 Ajax 请求
deleteAjax()	ajaxItem AjaxItem 对象	删除 Ajax 请求对象

表 4-1 列出了 Ajax 队列处理相关方法, 初始化页面时所有的 Ajax 请求根据 addAjaxItem()方法按照优先级和请求的顺序插入到请求队列 reqList, 请求调度的操作如图 4-4 所示:

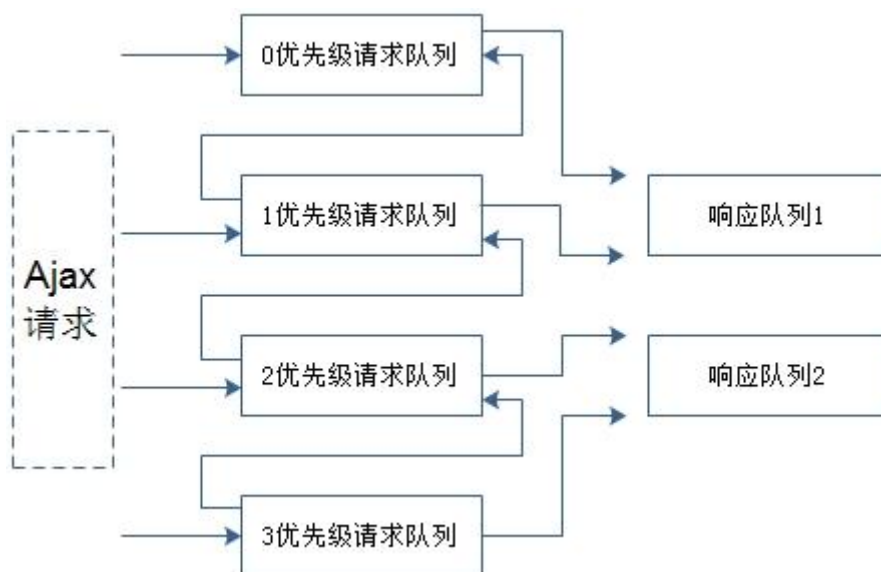


图 4-4 Ajax 请求调度流程

1) 调用 getAjaxItems(priority)方法获取每个优先级的请求, 将优先级为 0 的 Ajax 请求添加到 resList1 响应队列中, 如果没有优先级为 0 的请求, 则将优先级为 1 的请求添加到 resList2 响应列表中, 依次类推, 将前两个最高优先级的请求队列分别添加到两个响应队列中。

2) 当 resList1 响应队列完成 Ajax 请求时, 检测 resList2 响应队列的请求优

优先级，如果 resList2 中请求优先级为 2，在 resList1 响应队列中首先添加请求优先级为 0 的请求，如果没有优先级为 0 的请求，添加优先级为 1 的请求，照此方法将请求添加到响应队列中。

3) 当 resList2 完成请求时，按照 2)的策略选择响应高优先级的请求。

4) 当请求队列中 Ajax 请求等待时间超过 30s，则调用 changePriority()方法提升其优先级，重复 2) 3)操作。

5) 按上述步骤反复执行，直到 reqList 请求队列及 resList1、resList2 响应队列均为空，所有请求全部完成释放资源。

#### 4.4.2 Ajax 异步请求

前端框架中采用 Sync 对象向服务器发送请求，建立与服务器的连接。Sync 通过调用 Ajax 异步请求调度方法向服务器发送请求，服务器响应数据采用 JSON 数据格式返回。

异步方法 sync()需要接收三个参数，method 参数定义执行的插入、删除、查询、修改操作名称，model 参数是需要与服务器同步状态的模型对象，options 是请求数据，其执行流程如图 4-5 所示：

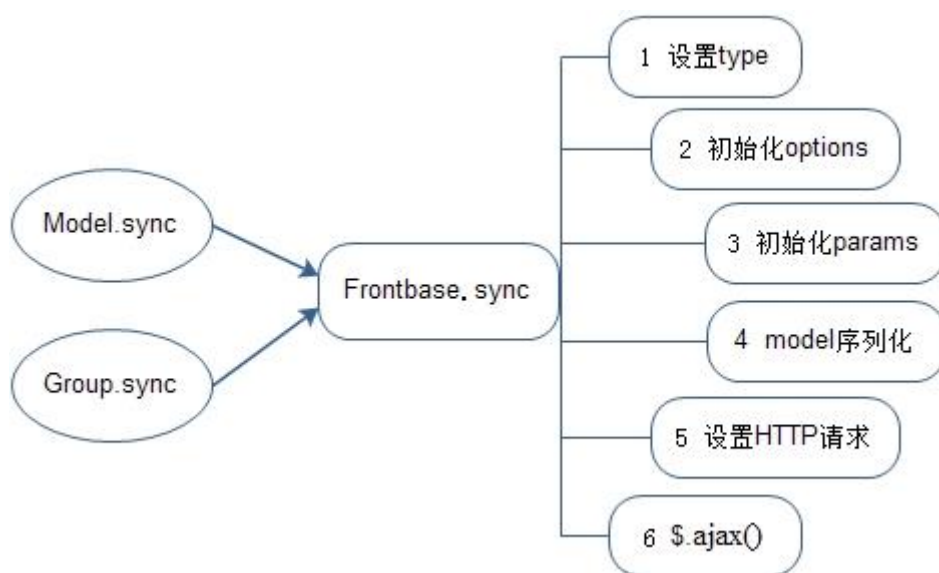


图 4-5 sync ()方法处理流程

1) 根据 method 参数从 methodMap 对象中获取与服务器交互的类型 type (POST、GET、PUT、DELETE);

2) 初始化请求数据 options;

3) `params` 作为请求参数对象传递给 Ajax 请求调度对象的 `getAjaxItems()` 方法, 初始化 `params`, 包括请求类型 `type`、数据交互默认格式 `dataType`、请求的 `url` 地址。如果在发送请求时没有在 `options` 中设置 `url`, 将通过模型对象的 `url` 属性获取 `url`;

4) 如果调用 `create` 和 `update` 方法, 调用模型的 `toJSON()` 方法将模型中的数据对象进行序列化;

5) 对于不支持 REST 请求方式的浏览器, 设置服务器的交互类型为 POST 请求;

6) 从 Ajax 请求调度对象的响应队列获取 Ajax 请求并调用 jQuery 框架的 `$.ajax()` 方法, 向服务器发送请求同步数据状态<sup>[46]</sup>。

## 4.5 本章小结

本章对前端框架中的模型层、视图层、控制器及服务器同步模块进行具体实现, 描述了模型数据的增删查改方法, 视图对象的创建及事件绑定, 事件控制器和导航控制器工作流程及模型数据与服务器同步的实现方法, 并对模板引擎的解析流程进行分析实现, 提出 Ajax 请求调度算法并在服务器同步对象中进行实现。

## 5 前端框架性能测试及在新闻管理系统中的应用

### 5.1 框架性能测试

#### 5.1.1 模板引擎渲染效率测试

图 5-1 为本框架中实现的模板引擎与现有模板引擎的渲染效率测试，横轴表示渲染时间（单位：毫秒），纵轴表示不同模板引擎，测试内容是将 100 条数据渲染 100 次所需的时间。

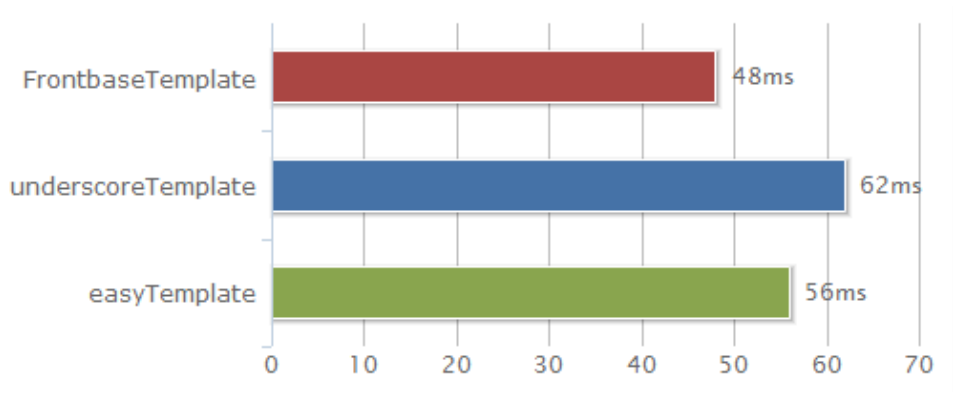


图 5-1 模板引擎渲染效率对比

#### 5.1.2 Ajax 请求调度效率测试

测试环境是在页面中发送 5 个优先级为 1 的请求，5 个优先级为 2 的请求，图 5-2 为采用 Ajax 优先级请求调度策略与同时发送 Ajax 请求的情况下比较优先级为 1 的请求得到响应所需时间。

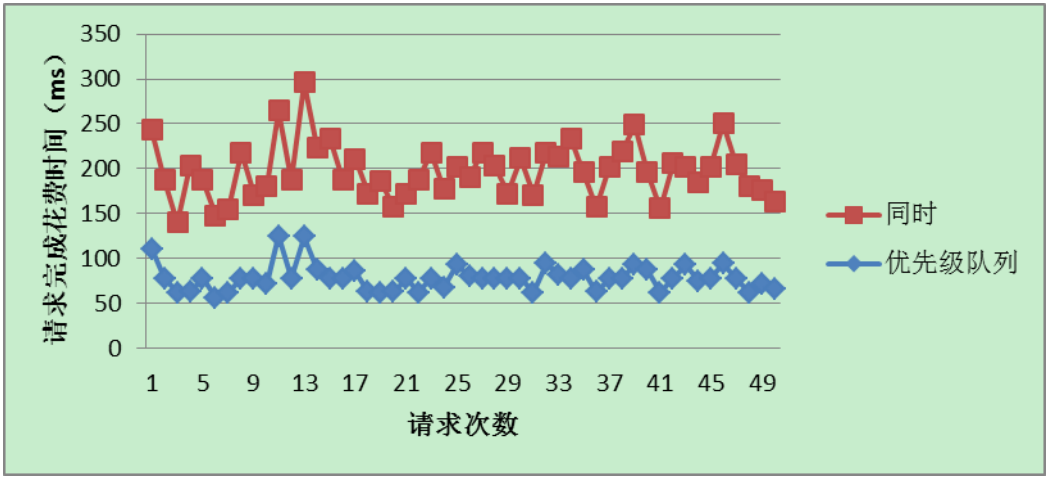


图 5-2 Ajax 请求调度效率对比

### 5.1.3 页面加载效率测试

根据性能响应时间 2-5-8 原则<sup>[47]</sup>,用户对页面的响应速度分为四个评价等级:小于 2 秒表明页面的响应很快;2-5 秒之间表明响应速度还可以,5-8 秒之间表明响应速度很慢,但是还可以接受,大于 8 秒后页面没有得到响应,用户可能会再次发送请求或离开站点。从用户行为来看,页面响应速度的提高有利于改善用户体验。

图 5-3 为基于本文实现的前端框架和 ExtJS 框架,向服务器发送 50 个请求,页面获取服务器响应并加载完成所需的时间对比。

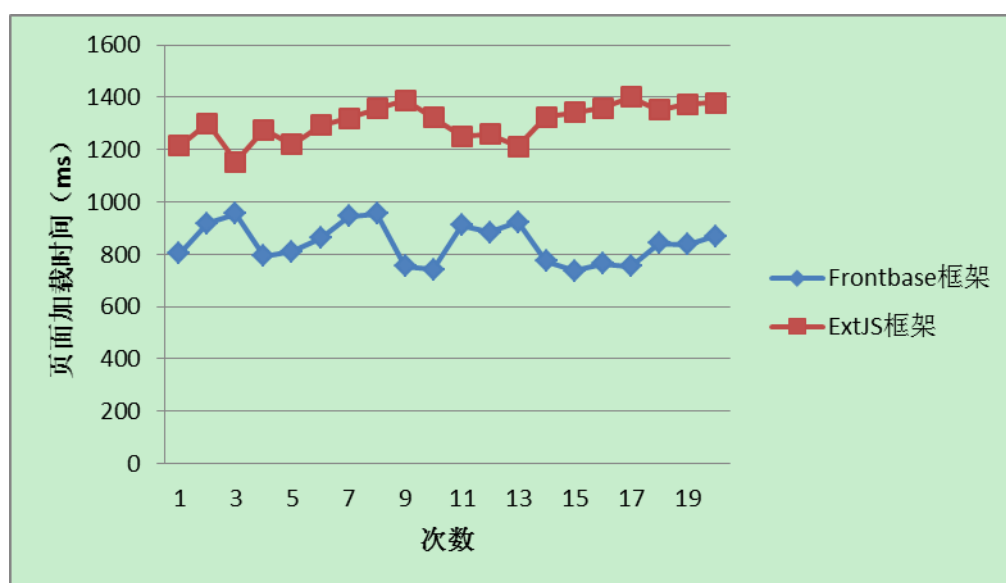


图 5-3 页面加载效率对比

## 5.2 新闻管理模块前端设计

新闻管理系统一般包括以下几个功能模块:用户管理模块实现对用户权限的管理,对不同身份的用户分配不同的权限;新闻管理模块实现对新闻的增加、删除、查询、修改功能;新闻检索模块采用分页的显示方式对已发布的新闻按照关键字进行查询;新闻审核模块实现用户对新闻的可见性功能,用户可以对审核通过的新闻进行搜索浏览。本文以新闻管理模块为例说明前端 MVC 框架 Frontbase 在实际系统开发中的应用。

根据新闻管理模块的前端功能需求,设计新闻管理模块的交互过程如下所示:

当用户点击增加新闻按钮时，页面呈现出新闻的详细属性信息，供用户编辑新闻所有的属性，同时提供给用户提交新闻的保存按钮，当用户点击保存按钮后，新闻列表动态更新。

选择新闻列表中的某一条新闻，点击编辑按钮时，页面呈现该条新闻的详细信息，用户可以查看并编辑这些信息，当用户点击保存按钮时，表单信息提交给后台，新闻列表动态更新，当用户点击删除按钮时，发送该条新闻的 ID 到服务器执行删除操作，服务器反馈成功后，新闻列表动态更新，该条新闻从新闻列表中移除。

根据新闻管理模块的交互过程，按照 MVC 架构对交互的时序设计如图 5-4：

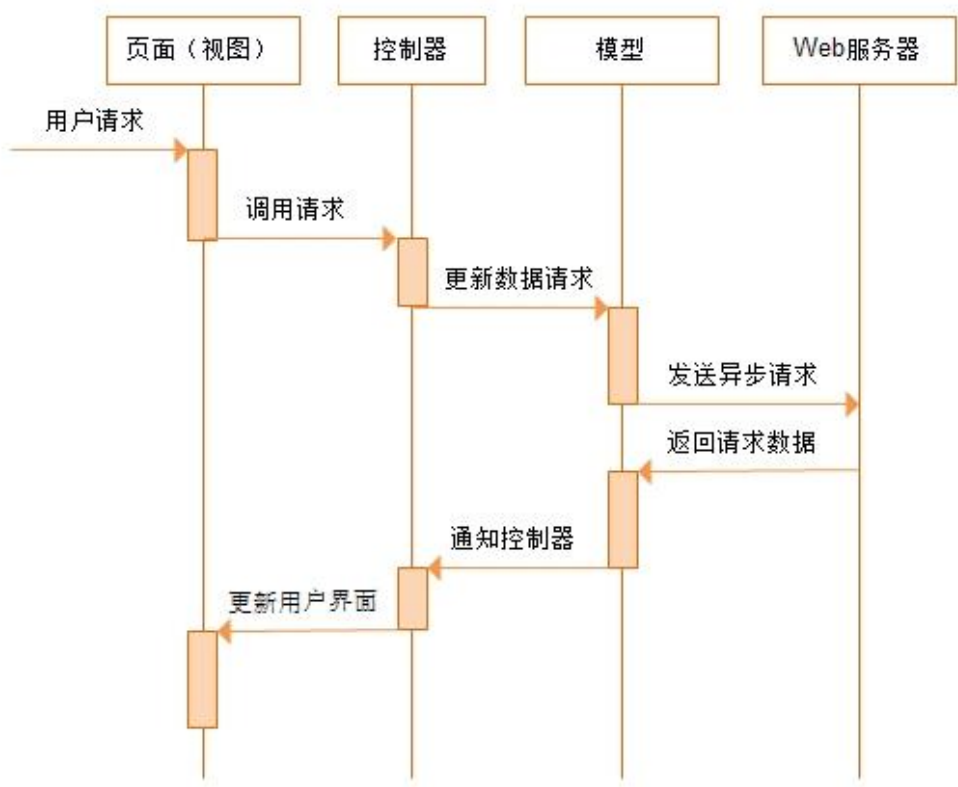


图 5-4 新闻管理交互时序图

控制器监听到页面的用户事件时，调用相应的模型层，模型向服务器发送请求数据，服务器响应请求后，模型层通知控制器把响应数据传递给视图，视图将控制器传递的数据更新或渲染到页面中。

5.3 新闻管理模块前端实现

新闻管理模块前端文件主要包括 models、views、template、lib 文件，models

文件夹中存放新闻管理模块的模型文件，views 文件夹存放视图文件，template 文件夹存放视图模板文件，lib 文件夹存放前端框架文件及 jQuery 文件。前端结构如图 5-5 所示

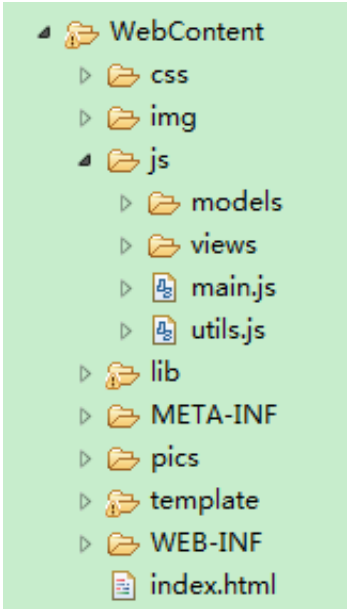


图 5-5 前端结构图

5.3.1 新闻管理模块模型层实现

模型层用于存储新闻数据对象，并与服务器进行同步，根据新闻管理系统设计 REST API 如表 5-1 所示：

表 5-1 新闻管理模块 REST API

HTTP 方法	URL	操作描述
GET	/Frontbase/news	检索所有的新闻列表
GET	/Frontbase/news/5	检索 id=5 的新闻信息
POST	/Frontbase/news	添加新的新闻
PUT	/Frontbase/news/5	更新 id=5 的新闻信息
DELETE	/Frontbase/news/5	删除 id=5 的新闻

创建单个新闻的模型对象 News 和新闻列表对象 NewsGroup, News 中 urlRoot 是 RESTful 服务请求的服务器地址，只有在检索或保存独立于集合的模型时才需要使用此属性。如果模型是集合的一部分，集合中定义的 URL 属性作为 RESTful API 检索、更新或删除数据的请求地址。



News 和 NewsGroup 模型代码如下：

```
window.News = Frontbase.Model.extend({
    urlRoot: "/Frontbase/news",
    validate: function(attrs){
        //数据验证
    }
});
window.NewsGroup = Frontbase.Group.extend({
    model: News,
    url: "/Frontbase/news"
});
```

### 5.3.2 新闻管理模块视图层实现

views 文件夹中的视图对象和 template 文件夹中的渲染模板文件名称是一一对应的，在项目启动时，根据视图对象的名称匹配相应的渲染模板。

当用户改变某条新闻的信息时，需要将对应的单条新闻视图自动重新呈现以反映这种变化，将视图绑定到模型的更改事件，然后在触发事件时执行视图渲染 render() 函数。

一个 NewsListItemView 对象对应 table 里面的一个 tr 元素，每次创建一个 NewsListItemView 对象的时候都会先调用 initialize() 方法，这个方法里面绑定的事件确保 tr 元素对应的 model 值每次发生改变或者被删除时都会同步到界面。也就是说每次操作界面对数据进行修改后都是先把当前的变更保存到 view 绑定的 model 对象里面，然后 model 里面的事件机制会自动触发一个 "change" 事件对界面进行修改。

单条新闻视图代码如下：

```
window.NewsListItemView = Frontbase.View.extend({
    tagName: "tr",
    initialize: function () {
        //每次更新模型后重新渲染
        this.model.bind("change", this.render, this);
        //每次删除模型后自动移除 UI
    }
});
```

```
        this.model.bind("change", this.remove, this);
    },
    render: function () {
        $(this.el).html(this.template(this.model.toJSON()));
    }
});
```

新闻列表视图在渲染的过程中，`render()`函数循环访问服务器返回的模型集合，调用单条新闻视图对象，实例化集合中的每条新闻，对每条新闻模型创建对应视图对象，然后以规定的形式展现在页面上。

`NewsView` 是新闻表单页面视图对象，用于显示单条新闻详细信息，添加新的新闻，删除某条新闻，`render()`函数将模型数据合并到从 `index.html` 检索的 `NewsView` 模板。新闻表单视图代码如下：

```
window.NewsView = Frontbase.View.extend({
    render: function () {
        $(this.el).html(this.template(this.model.toJSON()));
    },
    saveNews: function () { //增加/修改新闻
        this.model.save(null, {success: function (model) {self.render();}
        });
    },
    deleteNews: function () { //删除新闻
        this.model.destroy({success: function () {}
        });
    },
});
```

`saveNews()`通过调用模型的 `save()`方法实现新闻的增加。更改新闻信息包括两种方式更新模型，一种是实时的方法，控制器监听视图的 `change` 事件，当视图发生改变时，同步更新模型，另一种是等待用户点击保存按钮时根据页面表单值跟新模型，然后将这些更改发送到服务器。删除新闻通过 `deleteNews()`函数实现，调用模型的 `destroy()`方法将请求发送到服务器。

### 5.3.3 新闻管理模块控制器实现

新闻编辑页面事件监听配置：









```
events: {  
    "change"      : "change",  
    "click .save"  : "saveNews", //增加/修改新闻  
    "click .delete" : "deleteNews" //删除新闻  
},
```

新闻列表页面导航监听控制器，新闻列表和单条新闻的请求方式类似，都是通过 `fetch()` 方法向服务器发送请求，不同的是查询新闻列表方法创建的是新闻集合类的对象，查询单条新闻方法创建的是单个新闻的模型。增加新闻方法根据 `NewsView` 模型对象加载新闻表单视图模板。

列表页面控制器代码如下：

```
window.NewsController = Frontbase.Controller.extend({  
    navs: {  
        "" : "list", //新闻列表  
        "news/add" : "addNews", //增加新闻  
        "news/:id" : "newsDetails" //查看新闻  
    },  
    list: function(page) {  
        //创建列表集合对象实例，向服务器发送请求并将响应数据渲染到页面  
    },  
    newsDetails: function (id) {  
        //根据 id 创建模型对象实例，异步请求并渲染页面  
    },  
    addNews: function() {  
        //创建模型对象实例，并更新服务器数据  
    }  
})
```

5.3.4 新闻管理模块效果图

标题	发布时间	编辑
公益持续发展的动力 社会问题解决的路径	2014-02-20	 编辑
盘点“2013年社会组织十件大事”	2014-02-19	 编辑
微信也能投融资？	2014-02-18	 编辑
关于报名参加创业家年会的公告	2014-02-17	 编辑
大学生就业创业工作机制显成效	2014-02-16	 编辑
探索一种崭新的公益模式	2014-02-16	 编辑
公益组织如何规范运作	2014-02-15	 编辑
坚守的力量	2014-02-15	 编辑

1

2

3

图 5-6 新闻列表页

新闻标题:

盘点“2013年社会组织十件大事”

副标题:

2013社会组织十件大事

发布时间:

2014-02-19

所属栏目:

新闻资讯

发布人:

王芳

关键字:

社会组织

新闻内容:

坚持“突出重要性、注重导向性、兼顾全面性”的评选原则，对社会组织领域的重大方针政策、标志性事件、发展成果等进行集中展示，在扩大社会组织影响力、增进社会各界对社会组织了解、改善社会组织发展环境等方面取得了很好的效果。

保存

删除

图 5-7 新闻表单页面

## 5.4 本章小结

本章对模板引擎渲染效率、Ajax 请求调度效率及页面加载效率进行了对比和测试，并对前端 MVC 框架在新闻管理系统中的应用进行了实现，对新闻管理模块的前端进行设计，并对新闻管理模块中涉及的增加新闻、删除新闻、查询新闻、修改新闻模块的模型层、视图层及控制器的实现进行了详细的阐述，最后给出了主要的效果截图。



## 6 总结与展望

### 6.1 工作总结

早期 Web 页面采用 HTML 和简单的样式展示静态信息,页面交互功能较少,Web 的迅速发展使得 Web 页面不再仅限于展示静态信息,DHTML、CSS、JavaScript 等技术的出现使得页面交互受到越来越广泛地关注<sup>[48]</sup>。浏览器端页面代码规模不断扩大,页面中的 HTML、CSS、JavaScript 等代码相互交叉使用,代码重用性差。结构化 Web 前端程序,使得代码易于阅读和维护,提高页面响应效率,改善用户体验,是前端开发面临的重要问题。

本文介绍了前端框架开发过程中涉及的主要技术,并基于 MVC 设计模式实现了 Web 前端框架,建立各层之间的松耦合关系,设计并实现了模型层、视图层、控制器层的主要对象和方法。框架分为三层:

模型层:完成数据校验、数据处理,向服务器发送 Ajax 请求调用服务器逻辑,接收返回的数据,更新对应的视图。

视图层:绑定事件控制器,接收模型层返回的数据,并通过模板渲染更新页面。

控制器:监听视图元素的事件,根据事件调度执行模型的业务逻辑。

视图层实现了模板引擎,分离了前端 JavaScript 代码和 HTML 代码片段,提高了页面渲染效率和代码的复用性,在模型层提出 Ajax 请求调度算法并进行具体实现,提高了用户请求响应速度。

在前端 MVC 框架的基础上搭建新闻管理系统的前端应用,以新闻管理模块的增加、删除、修改、查询功能为例详细描述前端 MVC 框架的处理流程。

### 6.2 研究展望

目前企业级应用的前端开发大多停留在功能的实现,对前端代码的解耦和复用重视程度不够,代码复杂解析速度慢,用户体验不能得到很好的效果。本文实现的基于 MVC 模式的前端框架根据不同模块的功能进行分层,把对视觉界面、交互逻辑和数据处理分开,层次结构清晰,易于前端开发的扩展和复用,由于底

层采用 jQuery 作为基础开发框架，对浏览器兼容性具有良好的支持，使得用户体验得到了进一步提升。

但是，前端框架中还有以下几点不足与值得进一步研究的方面：

1) 前端框架中在模型层进行数据验证，没有与服务器建立有效的连接，模型层数据发送到服务器之后还需要进行验证。

2) 前端框架按照功能进行分层，视图层渲染的是基本的 HTML 和 CSS，在框架中集成相关的 UI 组件库，例如窗口插件、标签插件、按钮等，使得用户体验更加优化。

3) 随着前端技术的发展，前端安全问题也随之增多，如跨站脚本攻击<sup>[49]</sup>、跨站点伪造请求<sup>[50]</sup>，劫持 cookie 盗取用户数据<sup>[51]</sup>等。本框架实现的模板引擎对数据库中的特殊字符进行转义，可以防止跨站脚本攻击，对于其他安全问题还需要进一步研究。



## 参考文献

- [1] A. A. Andrews, J. Offutt, and R. T. Alexander. Testing web applications by modeling with FSMs[J]. *Software & Systems Modeling*, 2005, vol. 4, no. 3, pp. 326–345.
- [2] 赵增敏 and others. jQuery 全面提速[M]. 机械工业出版社, 2010.
- [3] A. Ranabahu and A. Sheth. Semantics centric solutions for application and data portability in cloud computing[C]. In: *Cloud Computing Technology and Science (CloudCom)*, 2010 IEEE Second International Conference, 2010, pp. 234–241.
- [4] S. Souders and 刘彦博. 高性能网站建设指南[M]. 北京: 电子工业出版社, 2008.
- [5] 曹刘阳, 李恒, 贾文副. 编写高质量代码[M]. 机械工业出版社, 2010.
- [6] B. Caldwell, W. Chisholm, J. Slatin, G. Vanderheiden, and J. White. Web content accessibility guidelines 2.0. W3C working draft, 2006, vol. 27.
- [7] 王保平. 对 JavaScript 框架的再思考[J]. *程序员*, 2008, no. 11, p. 24.
- [8] D. Esposito. A Little Insight of JavaScript Library[J]. *Programmer*, 2008, vol. 8, p. 55.
- [9] H. Yueshun and L. Min. Based on the ExtJS Technology and SSH Framework Authority Management Research[J]. In: *Software Engineering and Knowledge Engineering: Theory and Practice*, Springer, 2012, pp. 213–220.
- [10] 刘秀芹. 基于 STRUTS 的 EXT 技术应用研究[J]. *Software Guide*, 2009, vol. 8, no. 7.
- [11] L. M. Orchard, A. Pehlivanian, and 杨明军. JavaScript 框架高级编程: 应用 Prototype, YUI, Ext JS, Dojo, MooTools[M]. 清华大学出版社, 2011.
- [12] 裴生雷. Ajax+ JSP 开发模式的研究及应用[J]. *计算机技术与发展*, 2013, vol. 23, no. 1, pp. 242–245.
- [13] 浦云明, 范明红, 许明娜. Web 应用系统负载测试[J]. *计算机应用与软件*, 2009, vol. 26, no. 11, pp. 120–123.
- [14] 张仕. 基于面向对象软件的动态更新研究[D]. 上海交通大学, 2008.
- [15] 张治栋 and 韩康. 模块化: 系统结构与竞争优势[J]. *中国工业经济*, 2006, no. 3, pp. 92–99.
- [16] R. Chugh, J. A. Meister, R. Jhala, and S. Lerner. Staged information flow for JavaScript[J]. In: *ACM Sigplan Notices*, 2009, vol. 44, no. 6, pp. 50–62.
- [17] Y. Li and D. Cui. Improvement and Application of MVC Design Patterns [J]. *Computer Engineering*, 2005, vol. 9, p. 35.
- [18] 丁聪颖. 基于 J2EE MVC 的 webGIS 及其空间数据索引的研究 [D]. 上海交通大学, 2007.
- [19] 窦德聃. 基于 MVC 体系结构的 WEB 应用程序设计[D]. 北京: 中国地质大学, 2002.
- [20] 陈海洋. 基于 MVC 模式的 web 系统的解决方案[D]. 电子科技大学, 2007.

- [21] 吕海涛. 基于 XML 和 J2EE 架构的 MVC 视图层渲染引擎的设计与实现[D]. 中国海洋大学, 2007.
- [22] 陈玮 and 沈雷. 基于 MVC 模式的 WEB 应用框架[J]. 微计算机信息, 2009, no. 15, pp. 216–218.
- [23] R. L. Costello. Building web services the rest way. UR L: <http://www.xfront.com/REST-Web-Services.html>. Ultima Consulta, 2007, vol. 11, p. 2007.
- [24] 毛峰, 刘婷, 刘仁义, 刘南, 张丰. 基于 REST 面向资源的地理信息服务设计[J]. 计算机工程, 2011, vol. 37, no. 8.
- [25] J. M. Dodero and E. Ghiglione. ReST-based web access to learning design services[J]. Learning Technologies, IEEE Transactions, 2008, vol. 1, no. 3, pp. 190–195.
- [26] 牛磊. 基于 REST 的服务器框架研究与实现[D]. 北京邮电大学, 2010.
- [27] D. Boloker, R. Hosn, P. Jang, J. Kleindienst, T. Macek, S. Maes, T. Raman, L. Seredi, and others. Systems and methods for implementing modular DOM (Document Object Model)-based multi-modal browsers. Google Patents, 2001.
- [28] 李超. CSS 网站布局实录——基于 Web 标准的网站设计指南[M]. 北京: 科学出版社, 2006.
- [29] G. Richards, S. Lebresne, B. Burg, and J. Vitek. An analysis of the dynamic behavior of JavaScript programs. In: ACM Sigplan Notices, 2010, vol. 45, no. 6, pp. 1–12.
- [30] J. Resig and others. jquery: The write less, do more, javascript library. Dispon{\i}vel em <http://jquery.com/>, 2009, Acesso em, vol. 18, no. 04, p. 2009.
- [31] L. A. Meyerovich and B. Livshits. ConScript: Specifying and enforcing fine-grained security policies for Javascript in the browser[J]. In: Security and Privacy (SP), 2010 IEEE Symposium on, 2010, pp. 481–496.
- [32] R. M. Lerner. At the forge: jQuery. Linux Journal, vol. 2009, no. 178, p. 6, 2009.
- [33] 周玲余. 基于 jQuery 框架的页面前端特效的设计与实现[J]. 计算机与现代化, 2013, no. 1, pp. 61–63.
- [34] 程建. 嵌入式浏览器 DOM 研究与设计[D]. 电子科技大学, 2011.
- [35] W. Jiang, L. Zheng, and H. Qin. Design of network management system for optical terminal based on embedded web server[C]. In: Electric Information and Control Engineering (ICEICE), 2011 International Conference on, 2011, pp. 313–316.
- [36] 罗兵. 支持 AJAX 的互联网搜索引擎爬虫设计与实现[D]. 杭州: 浙江大学, 2007.
- [37] J. J. Garrett and others. Ajax: A new approach to web applications. 2005.
- [38] 赵金勇. JSON——开发 AJAX 程序首选数据传输格式. 中国科技论文在线, 2008.
- [39] G. Wang. Improving data transmission in web applications via the translation between XML and JSON[C]. In: Communications and Mobile Computing (CMC), 2011 Third International Conference on, 2011, pp. 182–185.
- [40] 邵刚. 基于 Spring 框架的 MVC 控制器的优化与改进[D]. 济南: 山东大学, 2011.

- [41] 郭丹丹. 基于 MVC 的前端开发研究与应用[D]. 北京邮电大学, 2012.
- [42] M. Martin and M. S. Lam. Automatic generation of XSS and SQL injection attacks with goal-directed model checking[C]. In: Proceedings of the 17th conference on Security symposium, 2008, pp. 31–43.
- [43] N. C. Zakas, 扎卡斯, and 欣, JavaScript 高级程序设计[M]. 人民邮电出版社, 2006.
- [44] 郭丹丹, 邓芳. Web 前端性能优化研究及其在网站足球彩票开发中的应用[D]. 北京邮电大学, 2011.
- [45] 李周志, 王晓东, 王真之, 冯志敏, and 周宇. 基于多优先级缓存队列的远程数据传输技术[J],” Computer Engineering, vol. 36, no. 18, 2010.
- [46] 黄竞. 基于 jQuery 框架的 Web 前端系统构建方法的研究与应用[D]. 北京邮电大学, 2013.
- [47] 芮素娟 and 丁晓明. Web 应用性能测试进展[J]. 计算机科学, 2006, vol. 33, no. 8, pp. 278–280.
- [48] 吕林涛, 万经华, 周红芳. 基于 AJAX 的 Web 无刷新页面快速更新数据方法[J]. 计算机应用研究, 2006, vol. 23, no. 11, pp. 199–200.
- [49] 邓巍巍. 跨站脚本攻击技术的成因与防御[J]. 信息安全, 2007, no. 9, p. 27.
- [50] 褚诚云. 跨站请求伪造攻击: CSRF 安全漏洞[J]. 程序员, 2009, no. 3, pp. 98–100.
- [51] 刘啸. 基于 Cookie 的 Session 渗透入侵分析及其安全模型研究[D]. 浙江大学, 2003.



## 致 谢

三年的研究生生活马上就要结束了，在完成毕业论文的过程中，许多人给予了我热心的帮助和指导，在此，我由衷的感谢他们。

首先，感谢我的导师魏志强教授。研究生期间，无论实验室课题的完成中还是学位论文的撰写中，魏老师都给予我许多帮助和指导。魏老师渊博的学识、严谨的学风和平易近人的高尚品德使我受益匪浅。在此，特向魏老师表示我最崇高的敬意。

感谢贾东宁老师和杨永全师兄，在我读研期间一直在研究项目上给予我最悉心的指导，无论在技术方面还是项目管理方面独到的见解和巧妙的思路带给我极大的启发，在学习工作中和论文撰写的过程中感谢你们对我的指导和鼓励。

感谢实验室的全体同学，感谢你们在三年时光中陪我一起学习，一起奋斗，使我的研究生生活充实而快乐，和你们在一起的奋斗时光是我宝贵的人生财富。

最后，感谢我的家人，你们对我无微不至的关怀和鼓励是我永远的精神支持，在我的求学道路上为我提供良好的学习条件，你们默默地付出和无私真挚的爱是我学习的最大动力，谢谢你们！



## 个人简历、在学期间发表的学术论文与研究成果

### 个人简历

1988 年 1 月 19 日出生于河北省保定市。

2007 年 9 月考入中国海洋大学信息科学与工程学院计算机科学与技术专业，2011 年 7 月本科毕业并获得工学学士学位。

2011 年 9 月考入中国海洋大学信息科学与工程学院计算机应用技术专业攻读硕士学位至今。

### 发表的学术论文

- [1] Shuyi Qiao, Zhiqiang Wei, Yongquan Yang. Research on Vegetable Supply Chain Traceability Model Based on Two-dimensional Barcode. 2013 Sixth International Symposium on Computational Intelligence and Design (ISCID2013).2013.6:317-320

### 研究成果

- [1]中国青年创业公共服务平台  
[2]有机蔬菜溯源平台