|  |
| --- |
| 浙江网新恒天软件有限公司 |
| MyHengtain 前端架构设计 |
| V1.0 |
|  |
| **ADM** |
| **7/1/2013** |

|  |
| --- |
| 本文档主要阐述了新版MyHengtian前端体系的架构设计，其包括前端技术JavaScript，CSS和HTML的整体架构设计方式。 |

目录

[1概述 3](#_Toc360536764)

[2 框架选择 4](#_Toc360536765)

[2.1 JavaScript MVC 4](#_Toc360536766)

[3前端架构设计 6](#_Toc360536767)

[3.1 架构总览 6](#_Toc360536768)

[3.2 模块化 7](#_Toc360536769)

[3.3 层次化 8](#_Toc360536770)

[3.4 抽象化 9](#_Toc360536771)

[4总结 15](#_Toc360536772)

# 1概述

新版MyHengtian（以下简称MyHengtian）的前端需要完全推翻老版的设计，一改其呆板，全静态，无交互的状态，取而代之要求一款操作流畅且富于用户体验的WebOS.

为了制作这一款产品，我们进行了审慎的考虑，经过对主流富客户端的技术的对比，最终选择了HTML5为技术基准，同时考虑到MyHengtian将是一款单页应用程序，而JavaScript的轻量级MVC框架非常合适开发此类应用，所以MyHengtian将以JavaScript的MVC设计为基础来进行架构。

使用HTML5技术开发应用有不可或缺的3大前端要素，他们是HTML5, CSS3以及JavaScript。到目前为止，经过对代码行数统计，各要素占比如下：

# 2 框架选择

MyHengtian前端开发框架的选择直接决定了后续架构的方式。经过分析和比对，我们选择jQuery为DOM基础操作框架，EmberJS为MVC框架。

## 2.1 JavaScript MVC

作为构建应用的基础，MVC框架必须具备一些重要的功能。比如，视图绑定、双向绑定、筛选、可计算属性（computed property）、脏属性（dirty attribute）、表单验证，等等。

* **可观察对象**：可以被监听是否发生变化的对象。
* **路由**：把变化通过浏览器URL的参数反映出来，并监听这些变化以便执行相应的操作。
* **视图绑定**：在视图中使用可观察对象，让视图随着可观察对象的变化而自动刷新。
* **双向绑定**：让视图也能把变化（如表单输入）自动推送到可观察对象。
* **部分视图**：包含其他视图的视图。
* **筛选列表视图**：用于显示根据某些条件筛选出来的对象的视图。

我们选择了业界使用较多的几款框架进行了对比：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 问题 | Angular | Ember | CanJS | Backbone |
| 可观察对像（observable） | **√** | **√** | **√** | **√** |
| 路由（routing） | **√** | **√** | **√** | **√** |
| 视图绑定（view binding） | **√** | **√** | **√** |  |
| 双向绑定（two way binding） | **√** | **√** |  |  |
| 部分视图（partial view） | **√** | **√** | **√** |  |
| 筛选列表视图（filtered list view） |  | **√** | **√** |  |

EmberJS对MVC框架所要求具备的各项特性支持度最好。Ember的开发效率首屈一指。它有很多强制性约束，可以帮助我们自动完成的事很多。而开发人员要做的，就是学习和应用这些约定，Ember会自动处理到位。

# 3前端架构设计

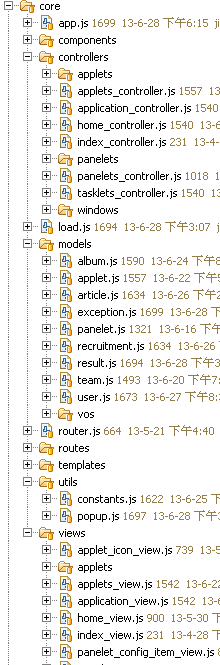
## 3.1 架构总览



MyHengtian的前端使用RequireJS进行脚本的模块化开发，脚本之间通过require方式来申明互相之间的依赖关系。CSS样式基于bootstrap进行开发，充分利用了CSS3的特性，同时又兼容IE等低端浏览器。JavaScript框架选用EmberJS，可以使整个应用具备动态绑定，路由，可观察对象等高级特性。EmberJS在应用初始化之时将加载必要的脚本模块,MyHengtian使用了Ember-Rest模块，该模块能够以面向对象的方式对后端资源发起RESTful格式的请求，后端Spring进行必要地业务处理，调用API组件向统一的MIS RESTful发起请求获取相应的数据返回给前端。Ember-Rest将反序列化这些数据并生成Model对象填充controller, controller为view的渲染提供了基准的数据支撑，最后view通过Handbar模板文件创建视图组件插入页面的HTML文档完成整个了渲染过程。

## 3.2 模块化

MyHengtian的前端采用模块化的方式进行开发，这些模块化的脚本在开发环境中将采用动态加载的方式进行加载并解析执行。模块的组织按照EmberJS对Scripts的结构要求进行分层，下面是MyHengtian前端脚本的目录结构的一部分：

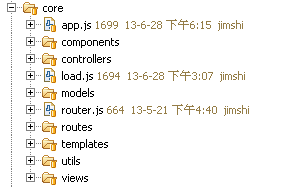


其中每一个JS文件就是一个模块脚本，在开发的过程中这些模块脚本无需在HTML文档中预先进行申明，可以在使用时从服务器端动态加载。当然，在正式的生产环境中，为了减少HTTP请求的数量，我们需要对这些模块脚本进行压缩合并。以下是一个RequireJS的代码样例：

|  |
| --- |
| define([  'app/core/app'  ],**function**(  Core  ){  "use strict";    **return** Core.Album = Ember.Resource.extend({  resourceUrl: '/albums',  resourceName: 'album',  resourceProperties: [  'id',  'urls',  'description']  });  }); |

## 3.3 层次化

由于我们采用了JavaScript MVC的开发方式，整个MyHengtian应用的前端JavaScript代码结构分层非常清晰，各个层次分工明确，非常适合大团队协作编程。MyHengtian的前端JS代码结构如下：

s

* app.js脚本为整个MyHengtian工程的主文件，其中定义了核心App对象，以及整个应用的核心名字空间Core.
* load.js 为模块加载脚本，用来加载所有核心代码需要的依赖模块。
* router.js路由脚本定义了MyHengtian核心功能中路径片段与主页面之间的路由关系以及层次。
* models层定义了数据模型，其中包含Applet, Panelet等WebOS页面要素的各种抽象以及具体的业务模型和传输对象。
* controllers层定义了数据控制器，为views层提供数据容器，同时也定义一系列针对数据以及后端的调用操作。
* routes层定义了路由控制器，当应用页面的路径发生变动时，路由就会被重新定义以及生产，同时相应的控制器和视图也会重新产生，这个时候可以对该页面所要求的数据进行刷新或准备。
* templates 层定义了handlebar模板，用于视图HTML的布局以及CSS3样式套用。Handlebar模板作为view层不可缺少的依赖，自身拥有丰富的模板语言。
* utils层定义了常量和工具类。
* views层定义了各个业务场景下不同的视图片段，任何jQuery插件的定义和使用都可以在view类中完成，同时view类也可以定义事件监听器响应各类DOM事件。

## 3.4 抽象化

为了能够使MyHengtian成为一个典型的WebOS框架，我们对WebOS的UI要素进行了抽象化，使其能够尽量泛化通用。MyHengtian在UI结构上一共由2个页面组成：

* 信息面板页。



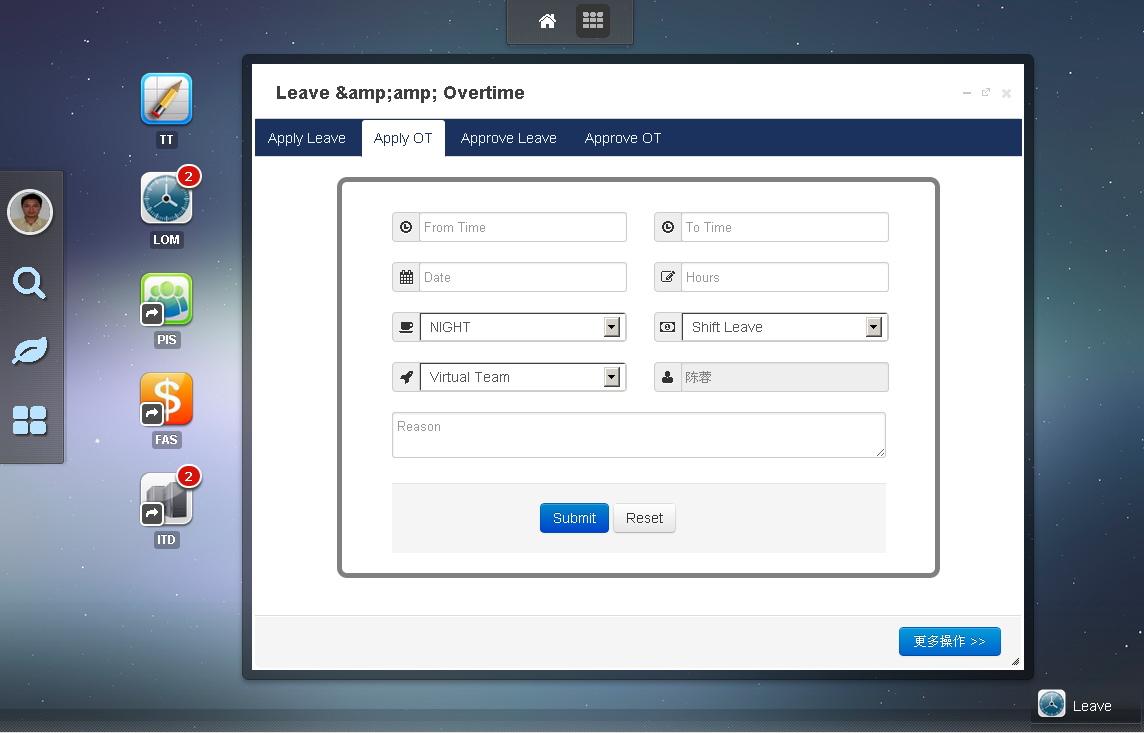
信息面板页由多个面板块组成，该页会根据当前登录用户的角色和权限状态来提供不同的面板集以供用户选择。MyHengtian对面板对象进行了一般抽象，定义模型为Panelet.以下是Panelet 的具体代码：

|  |
| --- |
| define([  'app/core/app'  ],**function**(  Core  ){  "use strict";    **return** Core.Panelet = Ember.Resource.extend({  resourceUrl: '/panelets',  resourceName: 'panelet',  selected: **true**,  resourceProperties: [  'code',  'title',  'size',  'url',  'type',  'selected',  'index']  });    }); |

其中，size属性定义了面板的尺寸（2 column 或者1 column）,title定义了面板的标题，code定义了面板的编号，type定义了面板的类型(embedded为嵌入类型，external为外链类型)，index为面板的默认排序号（越小排序越靠前），selected定义该面板是否被选中显示在页面上。当面板为external类型时，MyHengtian将为其创建一个默认iframe内部视图，通过其url属性链接过去显示；当为embedded类型时，需要为不同的面板自定义开发内部视图。以下是一个典型的文章列表内部视图：

|  |
| --- |
| define([  'app/core/app'  ],**function**(  Core  ){  "use strict";    **return** Core.ArticlesPaneletView = Ember.View.extend({    classNames: [''],    templateName: 'panelets/articles',    init: **function**() {  **this**.\_super();  },    didInsertElement: **function**(){  **this**.\_super();  **this**.get('controller').send('relayout');  },    willDestroyElement: **function**(){  **this**.\_super();  }  });  }); |

* 小应用列表页



小应用列表页展现了当前登录用户能够访问的MIS子系统列表，UI表现形式为一个个小应用图标，通过点击能进入具体的子系统。MyHengtian也对小应用对象进行了一般抽象，使其能够适用于通用的WebOS场景，为此抽象出的模型为Applet。以下是Applet的具体定义

|  |
| --- |
| define([  'app/core/app'  ],**function**(  Core  ){  "use strict";    **return** Core.Applet = Ember.Resource.extend({    resourceUrl: '/applets',  resourceName: 'applet',  running: **false**,  resourceProperties: [  'code',  'name',  'abbr',  'desc',  'width',  'height',  'running',  'minimized',  'notiCount',  'type',  'url'],    icon: **function**(){  **var** code = **this**.get('code');  **return** 'img/applet-'+code+'.png';  }.property('code'),    url: **function**(){  **var** code = **this**.get('code');  **return** code+'app';  }.property('code'),    isShortcut: **function**(){  **var** type = **this**.get('type');  **return** type === Core.Constants.APPLET\_TYPE\_SHORTCUT;  }.property('type')  });    }); |

Applet中，code表示具体的应用编号，abbr表示应用的名字缩写，title是显示在窗口上方的应用标题，notiCount是应用内的消息通知数，type表示应用的类型（emebedded表示内嵌类型需要自定义开发，external表示外链类型并通过iframe url打开子窗口，shortcut表示为快捷类型通过url属性直接在新的浏览器窗口打开）。MyHengtian创建了自己的窗口组件，通过继承该组件来实现不同的子系统快捷窗口，以下为一个典型的子系统窗口实现：

|  |
| --- |
| define([  'app/core/app'  ],**function**(  Core  ){  "use strict";    **return** Core.TaskTrackingView = Ember.View.extend({    classNames: ['applet-window-inner-content'],    templateName: 'applets/task\_tracking',    init: **function**() {  **this**.\_super();  },    didInsertElement: **function**(){  **this**.\_super();  },    willDestroyElement: **function**(){  **this**.\_super();  }  });  }); |

# 4总结

MyHengtian的前端使用了最新的JavaScript MVC的开发方式，可以说是开创性的。这中间我们遇到过一些弯路，踩过很多坑，但是始终对MyHengtian前端抱以产品化的开发态度，对UI元素精益求精。我们通过合理的架构以及不断的开发凝练，我们的目标是使MyHengtian的前端成为WebOS类型应用的开发典范，其他项目组可以直接基于该前端代码二次开发类似的应用，从而提高公司前端的整体开发效率。