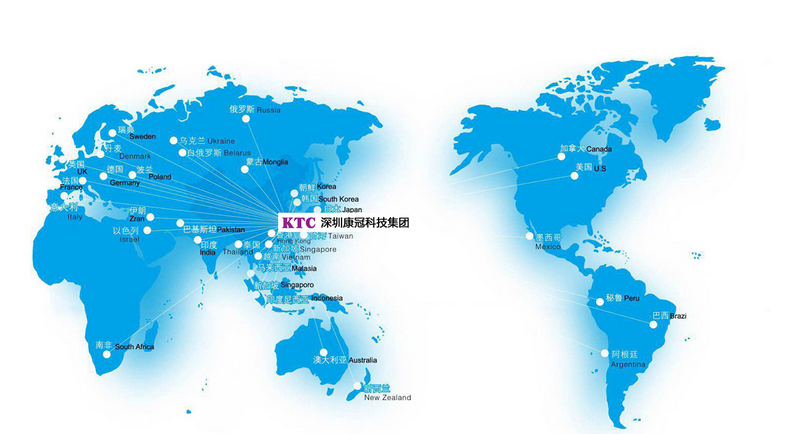
**KTC会议管理软件开发说明**



|  |  |  |
| --- | --- | --- |
| 文件状态：草稿  保密级别：绝密 | 文件标识： | KTC会议管理软件开发说明 |
| 当前版本： | 1.0 |
| 作 者： | 丁维龙 |
| 审 核： | 肖伟华 |
| 完成日期： | 2018年10月 |

**目录**

[1. 软件开发背景和使用需求](#_Toc713_WPSOffice_Level1) [1](#_Toc713_WPSOffice_Level1)

[1.1软件开发说明](#_Toc28037_WPSOffice_Level2) [1](#_Toc28037_WPSOffice_Level2)

[1.2软件使用环境](#_Toc20616_WPSOffice_Level2) [2](#_Toc20616_WPSOffice_Level2)

[1.3软件需求分析](#_Toc15496_WPSOffice_Level2) [2](#_Toc15496_WPSOffice_Level2)

[2. 软件整体开发](#_Toc28037_WPSOffice_Level1) [3](#_Toc28037_WPSOffice_Level1)

[2.1软件MVC框架结构](#_Toc14880_WPSOffice_Level2) [3](#_Toc14880_WPSOffice_Level2)

[2.2路由访问设置](#_Toc14980_WPSOffice_Level2) [5](#_Toc14980_WPSOffice_Level2)

[2.3控制器操作](#_Toc21508_WPSOffice_Level2) [6](#_Toc21508_WPSOffice_Level2)

[2.4视图显示](#_Toc25838_WPSOffice_Level2) [7](#_Toc25838_WPSOffice_Level2)

[2.5数据库模型](#_Toc20898_WPSOffice_Level2) [8](#_Toc20898_WPSOffice_Level2)

[3. 系统间通讯设计](#_Toc20616_WPSOffice_Level1) [11](#_Toc20616_WPSOffice_Level1)

[3.1 Web消息交互](#_Toc17953_WPSOffice_Level2) [11](#_Toc17953_WPSOffice_Level2)

[3.2 WebSocket](#_Toc11776_WPSOffice_Level2) [11](#_Toc11776_WPSOffice_Level2)

[3.3 SignalR](#_Toc1724_WPSOffice_Level2) [12](#_Toc1724_WPSOffice_Level2)

[4.界面显示控件UI交互](#_Toc15496_WPSOffice_Level1) [15](#_Toc15496_WPSOffice_Level1)

[END](#_Toc14880_WPSOffice_Level1) [19](#_Toc14880_WPSOffice_Level1)

**开发环境**  
计算机系统：Windows 7 SP1 64Bit  
开发语言：C# 3.0，基于.NET Framework 4.6

开发框架：ASP.NET MVC 5.0版+EF6  
开发IDE 环境：Visual Studio 2015专业版、ISS 7、Microsoft SQL Server2012

1. **软件开发背景和使用需求**

**1.1软件开发说明**

针对公司的会议室的使用情况出发，定制对应的Web端软件管理系统，实现前台对会议室快捷的安排管理。

采用B／S结构以ASP.NET MVC框架为基础构建系统表现层，以Entity Framework框架为基础构建数据访问层形成一个完整的系统架构。前端利用HTML+JavaScript(jquery)交互+Bootstrap做UI界面的显示操作。平板终端推送信息基于ASP.NET SignalR可以用于将任何种类的“实时”Web功能添加到您的ASP.NET应用程序。SingalR的基本通讯机制为服务端与客户端均建立一些操作函数，当客户端与服务端连接后，客户端可以调用服务端的函数，待服务端处理完毕后，可以再调用客户端的函数来更新客户端的相关信息，从而避免了客户端进行轮询服务端的完成状况。（其轮询机制封装在SingalR机制中了）

**1.2软件使用环境**

基于同一个局域网的环境中，各个设备间的网络连接畅通，保证对应的不同平台的数据传输的正常；平板端软件安装在安卓系统中，通过无线网进行连接，服务端部署windows系统中提供访问请求服务，客户端通过浏览器进行请求操作。

**1.3软件需求分析**

需要通过软件实现对会议室的快捷管理系统：利用操作的网页端控制和终端的显示，通过服务端的数据处理，基于MVC框架的布局设计。

操作客户端：提供给前台人员进行直接操作；

1. 提供安排相应的会议室和日程管理，会议室的内容有：对应的客户、对接业务、开会人数、时间、备注等信息；
2. 对未来半个月时间内进行会议的安排预约，可以记录预约的信息备注等；
3. 对过去一个月内的会议室安排使用信息进行查询，可以对某个会议室在那一天的时间内进行查询显示；
4. 接收终端的呼叫服务进行提示，确认收到信息；
5. 对会议室使用完毕后的状态进行清除，对会议室进行整理；

会议室显示终端：以平板形式放置于各会议室的门口显示信息、操作；

* 1. 显示当前会议室的使用状态、和必要的会议信息；
  2. 提供给开会人员进行呼叫服务内容：可通知前台提供茶水、咖啡、水果等服务；把对应的信息传给前台客户端进行通知确认；

数据服务端：记录保存客户端的操作信息，同时同步其他的客户端的显示情况；

1. 记录当前各个会议的使用信息情况，和客户端、终端的连接情况；
2. 保存会议室清除后的历史记录，提供后续的查询操作；
3. 能够对客户端连接后及时进行信息的一致同步显示；
4. 把客户端的安排设置信息及时通知终端进行显示；或把终端操作的反馈信息传送给对应前台进行确认安排；
5. **软件整体开发**

**2.1软件MVC框架结构**

MVC代表了模型-视图-控制器。MVC是一个架构良好，可测试性和易于维护的应用程序开发模式。基于MVC模式的应用程序包含：

* 模型：模型类代表了应用程序中的数据，为强制执行业务规则使用了数据逻辑验证。
* 视图：应用程序中的视图模板文件用来动态生成HTML响应。
* 控制器：控制器类处理传入的浏览器请求，获取模型数据，然后指定视图模板，将响应返回给浏览器。

**模 型**

封装应用程序的状态

响应状态查询

应用程序功能

通知视图变化

状态查询 状态改变

通知改变

控 制 器

定义应用程序功能

用户动作模式更新

选择响应的视图

**视 图**

解析模型

模型更新请求

发送用户输入

允许控制器选择视图

视图选择

用户请求

方法调用 事件

**MVC组件类型的关系和功能**

MVC框架项目文件目录结构及作用：

|  |  |
| --- | --- |
| **目录** | **作 用** |
| /Controllers | 用于保存处理URL请求的Controller控制器类 |
| /Models | 用于保存表示和操作数据及业务对应的类 |
| /Scripts | 用于保存JavaScript库文件和脚本(.js) |
| /Images | 用于保存网站使用的图片 |
| /Content | 用于保存CSS和其他站点的内容 |
| /App-Data | 用于存储读取/写入的数据文件 |
| /App-Start | 用于保存功能的配置代码，如路由、捆绑和Web API |
| /View | 用于保存负责显示输出的页面(HTML)的UI模版文件 |

**2.2路由访问设置**

ASP.NET MVC 根据传入的URL调用不同的控制器（还有不同的操作方法），ASP.NET MVC 默认的路由规则决定了调用那些代码：

/[Controller]/[ActionName]/[Parameters]

你可以在 App\_Start/RouteConfig.cs 文件中设置路由格式。

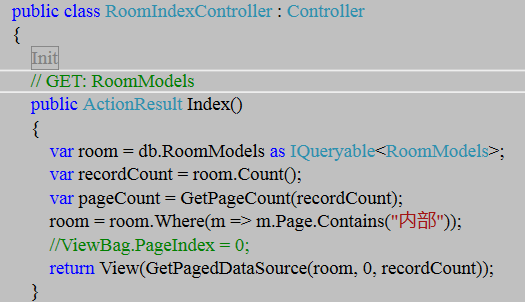


URL第一部分决定了要调用的控制器，/RoomIndex对应的是 RoomIndexController 类。第二部分决定了被调用的Action方法，因此 /RoomIndex/Index 引发调用了RoomIndexController 类的Index 方法。虽然我们浏览的地址是/RoomIndex，但是Index方法是默认的Action，所以Index方法会被调用。

提示：default中的默认值变量的名称必须和url中的名称相同，但是大小写不限制。

**2.3控制器操作**

Controller(控制器)在ASP.NET MVC中负责控制所有客户端与服务端的交互，并且负责协调Model与View之间数据传递，是ASP.NET MVC框架核心。Controller本身就是一个类(Class)，该类别有许多方法(Method)，这些方法中只要是公开方法(public method)就会被视为是一个动作(Action)或动作方法(Action Method)，只要动作存在，就可以通过该动作方法接收客户端传来的要求与决定响应的检视(View)。



Controller应具备如下几个基本条件：

1. Controller必须为公开类别；
2. Controller名称必须以Controller结尾；
3. 必须继承自ASP.NET MVC内建的Controller类别，或实现IController自定义类别;
4. 所有动作方法必须为公开方法，任何非公开的方法如声明为private或protected的方法都不会被视为一个动作方法；

控制器动作结果类型(ActionResult)：



控制器动作(具体的action)返回的结果叫做控制器动作结果，动作结果是控制器返回给浏览器请求的内容。

**2.4视图显示**

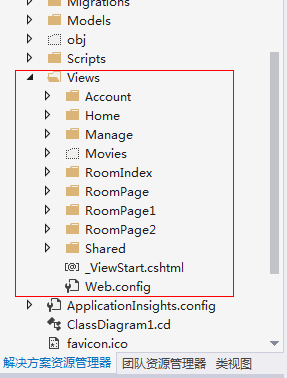
**在每一个控制器的View文件夹中，每一个操作方法都有一个同名的视图文件与其对应。**这就提供了视图与操作方法关联的基础。

1.ASP.NET MVC页面基本被放在Views文件夹下：

2.利用APS.NET MVC模板生成框架，Views文件夹下的默认页面为.cshtml页面；

3.ASP.NET MVC默认页面为Razor格式的页面，因此默认页面为.cshtml页面；

4.ASP.NET MVC中，支持静态html页面；

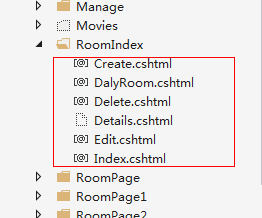


Account文件：包含用于注册登录用户账号的页面；

Home文件夹：存储首页和About页面信息；

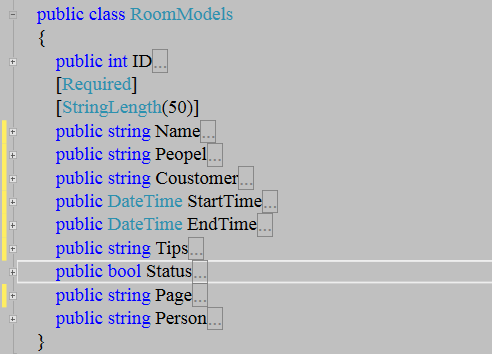
Share文件：存储控制器共享的视图，布局页面或模版页；

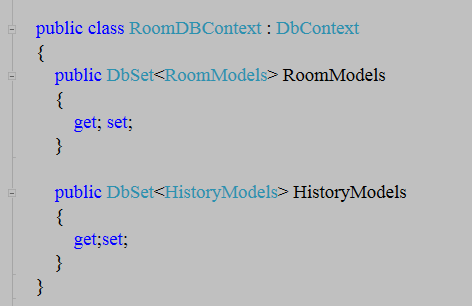
\_ViewStart.cshtml:程序开始执行页面；

ASP.NET MVC 提供的模板中，当创建View时，提供了六种模型，分别为： Create、Delete、Detail、Edit、Empty,Empty(不具有模型),List。基于该模型，我们可以轻松地创建具体实体的增删改查。

**2.5数据库模型**

ASP.NET MVC中的基架可以为应用程序的创建、读取、更新和删除（CRUB）功能生成所需要的样板代码。基架模版检测模型类的定义，然后生成控制器以及与该控制器关联的视图，有些情况下还会生成数据访问类。

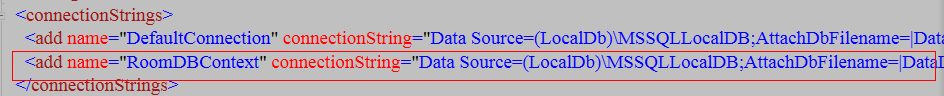


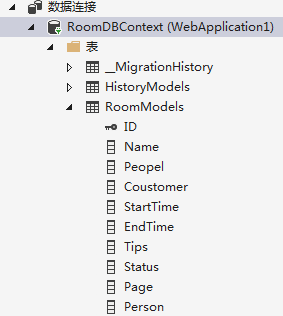
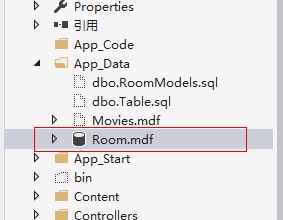
使用从EF的DbContext类派生出的一个类来访问数据库，该派生类具有一个或多个DbSet<T>类型的属性，类型DbSet<T>中的每一个T代表一个想要持久保存的对象。

RoomDBContext 类代表了Entity Framework 中 RoomModels 类的数据库上下文，用来处理获取、存储和更新数据库中的RoomModels 类的实例。RoomDBContext类继承自Entity Framework中提供的DbContext类。

新建的ASP.NET MVC5项目会自动包含对实体框架（EF）的引用。**EF是一个对象关系映射（object-relational mapping,ORM）框架，它不但知道如何在关系型数据库中保存.NET对象，而且还可以利用LINQ查询语句检索那些保存在关系型数据库中的.net对象。**

RoomDB是配置的数据库连接：

如果不配置具体的连接，EF将尝试连接SQL Server的LocalDB实例，并且查找与DbContext派生类名相同的数据库。如果EF能够连接上数据库服务器，但找不到数据库，那么框架会自动创建一个数据库。



1. **系统间通讯设计**

**3.1 Web消息交互**

常见的Web下的信息交互应用技术：

轮询（polling）：常见的一种实现数据交互的方式，开发人员控制客户端以一定时间间隔中向服务器发送Ajax查询请求大，但是也因此，当服务器端内容并没有显著变化时，这种连接方式将带来很多无效的请求，造成服务器资源损耗。适合并发量小，实时性要求低的应用模型，更像是定时任务。

长轮询（long polling）：长轮询是对轮询的改进，客户端通过请求连接到服务器，并保持一段时间的连接状态，直到消息更新或超时才返回Response并中止连接，可以有效减少无效请求的次数。

WebSocket：是HTML5提供的一种在单个 TCP 连接上进行全双工通讯的协议，目前chrome、Firefox、Opera、Safari等主流版本均支持，Internet Explorer从10开始支持；优点：服务器与客户端之间交换的数据包档头很小，节约带宽。全双工通信，服务器可以主动传送数据给客户端。

**3.2 WebSocket**

WebSocket本质上是一个基于TCP的持久化协议，相对于HTTP这种非持久的协议来说，它能够更好的节省服务器资源和带宽，并且真正实现实时通信。

WebSocket在实际运用时是在握手阶段从http请求升级上来的,具体的通讯连接的数据协议：

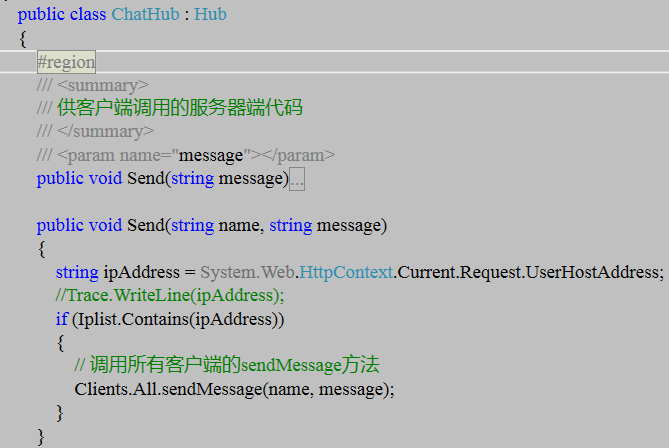


注意“1.”中的Upgrade:websocket和Connection: Upgrade这两个核心属性表示本次是一个特殊的http请求，目的就是要将浏览器端和服务器端的通讯协议从HTTP协议—升级—>WebSocket协议，其他属性都是客户端向服务器端提供的握手信息。

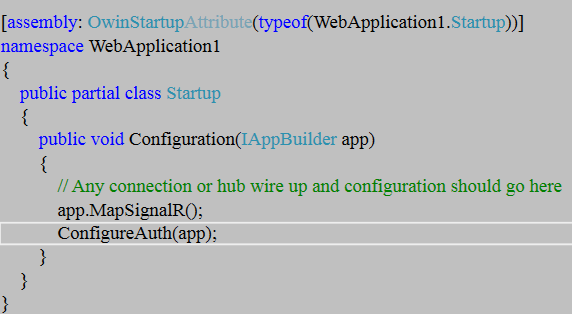
## 3.3 ****SignalR****

SignalR是一个.Net开源库，用于构建需要实时进行用户交互和数据更新的Web应用；SignalR简化了构建实时应用的过程，它包括了一个Asp .Net服务器端库和一个Js端库，集成了数种常见的消息传输方式，如long polling，WebSocket，并提供相应的Api供开发人员选择如何调用，帮助其可以简单快速地实现客户端与服务器端相互间的实时通信。

当环境条件合适时，**SignalR将WebSocket作为底层传输方式的优先实现**，当然，它也能很高效地回退到其他技术。同时，SignalR提供了非常良好的Api以供远程调用(RPC) 浏览器中的js代码。

服务端：

### **Startup启动类：**



前端JS实现：



注：SignalR基本适用于任何可以用上述技术实现的场合，但是对寄宿平台版本有要求。如.Net Framework 平台，SignalR库需要4.5及以上版本的支持

**4.界面显示控件UI交互**

主界面：



把各会议室在界面上直观显示，已经安排好的信息的或空闲的状态，界面中需要有所以的会议室的列表情况，同时需要把不同区域的会议室进行分开归类进行显示；



根据情况分为：内部、坂田、惠南、记录四个页面的显示。通过链接按键选择进行切换的显示。

会议室的安排状态分为：空闲中和开会中

每个会议室的显示布局在DIV模块中：里面有链接控件进行会议室状态设置操作,由跳转事件进行响应用户的操作状态

安排房间设置：



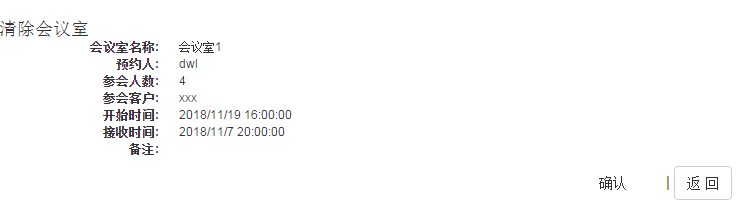
进入后进行当前会议室的信息安排编辑设置，保存后返回原页面显示对应的信息展示。

修改设置：



加入界面后对原有的会议信息进行修改设置，保存后返回原页面显示修改后的信息。

清空房间设置：



页面显示当前会议的信息，点击【确认】进行当前会议室的信息删除操作，如何返回原页面显示空闲的状态。

在对会议房间进行清空后，会把当前的信息记录到历史记录中，可以通过历史记录表查询进行查看。



**END**