**iQueue软件设计文档**

***医院排队系统***

***SpringMVC+mySql +WPF***

***作者：霍兆亨 14331110***

1. **架构设计**
2. **前端采用WPF进行页面开发。**
3. **后端采用SpringMVC+Mysql架设服务器**

**【前端】文件结构与接口技术说明**

**采用MVVM三层架构，文件结构：**

1. **四个wpf页面文件和他们对应的cs文件，一个用于登陆，一个用于控制排队，另外两个是添加信息的弹窗。**

**Init\_Login是我们的登陆界面，Init\_Login.xaml是定义了view， Init\_Login.xaml.cs是model和view\_model，这是wpf规定的架构设计。**



**登陆界面。**



**控制排队界面。**



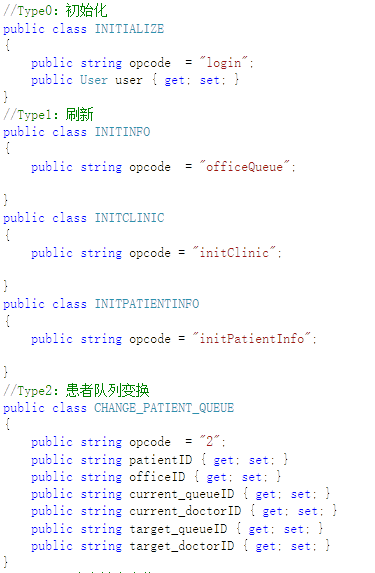
**添加**

**可扩展性：添加新的页面即创建新的xaml**

**可维护性：每个页面都有独立的代码，而且路径依赖清晰**

1. **两个库函数文件，information.cs一个负责定义model，networkWorker.cs是客户端网络接口。**

**OO设计：model定义了病人，医生，诊室等信息，主要用于缓存后台发过来的信息和显示，全部信息都是string类型存储，便于发送。而且所有发送的数据包都是OO，如下：**



**每一个发出去的数据包在前端都是一个类。**

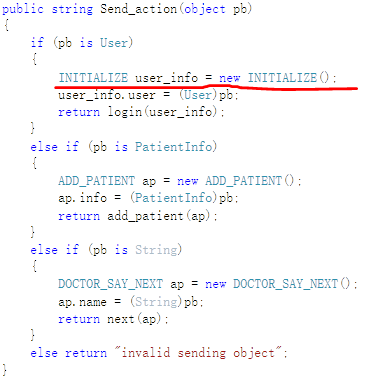
**客户端网络接口采用http协议**

**以登陆为例，我们来看看前端到底发生了什么。**

**当点击登录按钮之后，按钮绑定的函数会被执行：**



**OO设计是其中创建了个对象叫User，它有username和password。接着这个对象会被传入到login函数中，接着被打包成INITIALIZE类（添加一个opcode项）。**

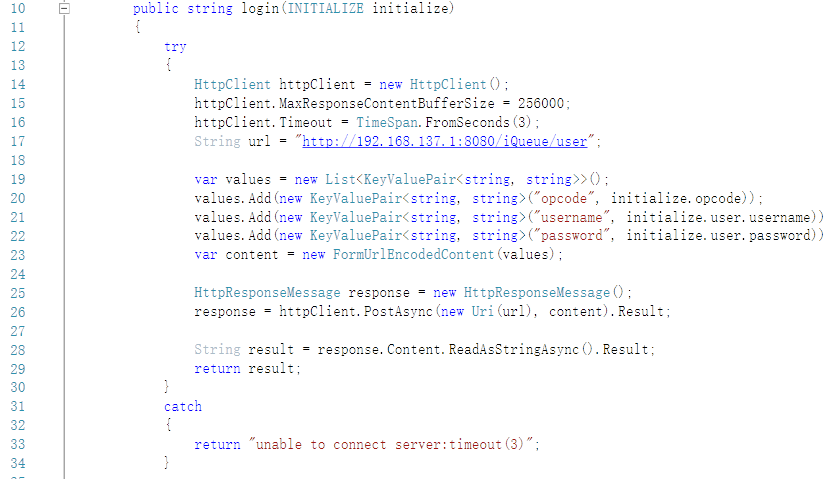


**最后通过键值对发出去。**

**url是http://192.168.137.1:8080/iQueue/user，Json格式是：**

{"opcode" :"0","User":{"username":"Henry","password":"123456"}}

服务器**超时设定3秒钟，超时返回超时信息，否则返回服务器发过来的字符串。**



**假设客户端成功连接服务端并且接受到数据返回给点击函数，点击函数紧接着解析发送JSON字符串，包依赖*Newtonsoft.js***



**对传回来的Json[“status”]的值进行判断，如果是”success”就接着解析。否则弹框显示用户名错误等等。至此一个登陆请求完成。然后跳转。其余请求原理类似。请求一览：**

# iQueue function

- /clinicDetail

> 点击诊室的列表，查看诊室详情 坐诊医生 起始时间 信息简介等

- /user

> 负责用户注册和登录 冲突检查 前端会进行正则过滤

- /getClinicList

> 进入科室大厅 获取所有诊室列表

- /add

> 添加病人信息 如卡号 ID号以及相应的体征信息

- /initPatientInfo

> 获取所有病人信息 包括体征

- /officeQueue

> 获取科室下的三个队列 首诊 2诊和分诊

- /updateOfficeInfo

> 更新科室信息，

- /updateClinicInfo

> 更新诊室信息 对应clinicDetails

- /checkPatientQueue

> 查看病人所在队列情况

- /deleteOvertime

> 删除超出一声坐诊时间的挂号病人

- /choiceDoctor

> 选择医生 进入相应的其候诊队列

**可扩展性：只需添加新的请求url即可，可扩展性强。**

**可维护性：后端代码对前端隐藏，只需要测试接口即可，维护简单。**

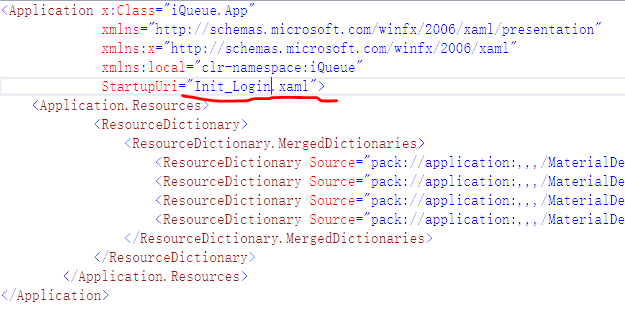
1. **一个全局资源文件app.xaml，定义全局资源，是应用的声明起始点。**

**App.xaml是应用的声明起始点。在VS新建一个WPF应用，就能自动生成一个App.xaml，同时包含了后台代码文件App.xaml.cs。这两个文件都是局部类，和Window类非常相似，让你能够使用标记语言和后台代码。**

**App.xaml.cs扩展了应用类（此处未改动），它是WPF窗口应用的中心类。.NET首先进入这个类的起始指令，从这里启动预想的窗口或者网页。同时这里订阅了重要的应用事件，如应用启动、未处理的异常等等。**

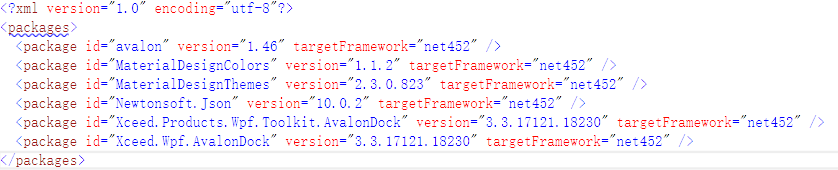
**App.xaml最常使用的特性是定义全局资源，它们可能会在整个应用里面被使用或者访问，如全局样式。**

**刚开始打开程序，编译器会查看app.xaml下的StartupUri来确定初始界面：**



***第一个打开的页面是Init\_login***

**下面标记了一些ResourceDictionary，代表着引用的外部包的路径，包的安装右键项目选择“管理Nuget程序包…”。**



***用的包有好几个，主要是json解析的包Newtonsoft.js***

**UI包是AvalonDock（用来实现选项卡）和MaterialDesignColors（输入框点击动画）。**

**可扩展性：新的包可以背很容易的添加**

**可维护性：包的维护由包提供者提供**

**【后端】服务器架构&数据库技术说明**

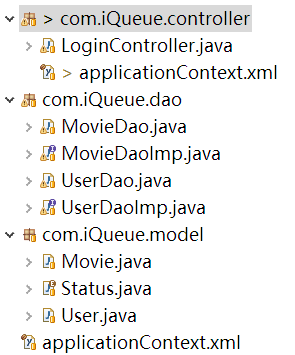
1. **服务器架构说明**

**Spring 框架提供了构建 Web 应用程序的全功能 MVC 模块。使用 Spring 可插入的 MVC 架构，可以选择是使用内置的 Spring Web 框架还是 Struts 这样的 Web 框架。通过策略接口，Spring 框架是高度可配置的，而且包含多种视图技术。**

**Spring MVC框架的特点**

* **清晰的角色划分：控制器（controller）、验证器（validator）、 命令对象（command object）、表单对象（form object）、模型对象（model object）、 Servlet分发器（DispatcherServlet）、 处理器映射（handler mapping）、视图解析器（view resolver）等等。 每一个角色都可以由一个专门的对象来实现。**

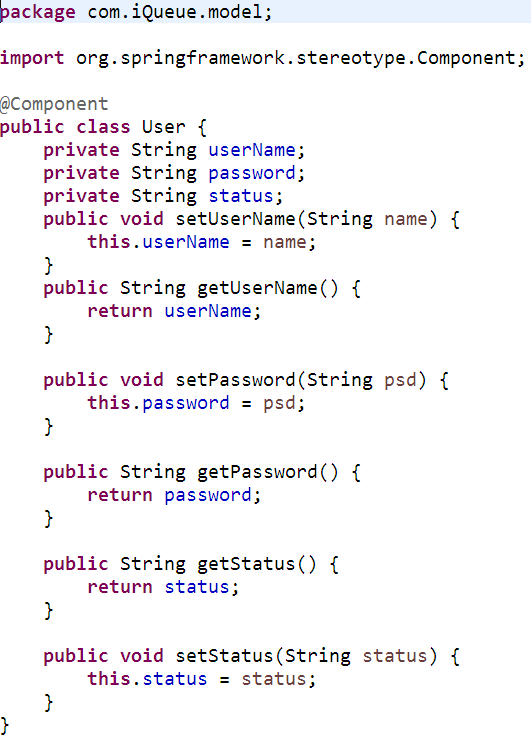
**具体代码结构：**



**MVC+java经典架构**

**上图可以观察到，我们将C层全部放到Controller包当中，数据库操作放到DAO(Data Access Object) 中，M层全部放到model包当中。**

**以User为例：**



***Model***



***Dao***



***Controller（登陆请求处理）***

**可扩展性：添加新的请求就是往controller中添加新的操作，添加新的model就是在model包里添加新的class，数据库操作更不用说，在DAO中添加新的SQL语句以响应不同请求即可。**

**可维护性：代码结构清晰，使用框架，一旦出现错误使用System.out.println()打印错误，方便寻找错误源。**

* **强大而直接的配置方式：将框架类和应用程序类都能作为JavaBean配置，支持跨多个context的引用。**

**具体代码结构：**

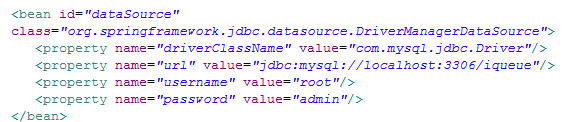


*Application.xml中定义了JavaBean*

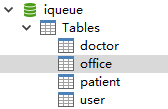
**2. 数据库说明**

1. 在文件中dataSource的bean中定义了数据库的接口为

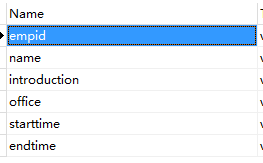
Localhost:3306/iqueue，用户名为root，密码为admin，若有需要，自行更改



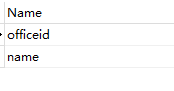
2. 数据库中有四个表



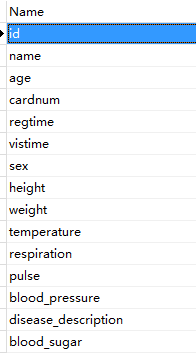
doctor的列：



office的列：



patient的列：



user的列

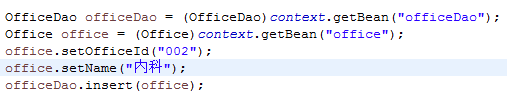


**Dao类函数接口说明**

以Office为例

1. 插入信息：

Insert(Office office)

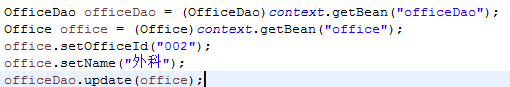


运行之后数据库中成功插入信息



2. 更改信息：

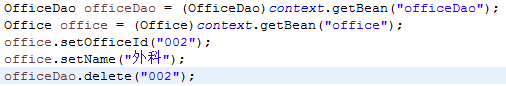
update(Office office)，根据officeId更新信息，在这里更新了id为002的Office信息





3. 删除信息：

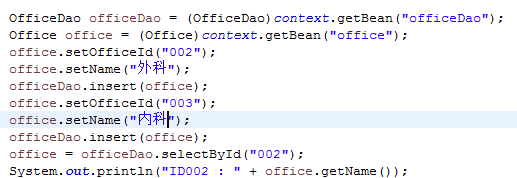
delete(String id)，在这删掉了id为002的信息





4. 查找信息

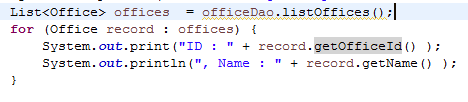
selectById(String id)，这里先插入两条信息，再查找id为002的信息，输出name

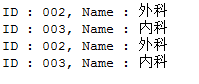




5. 获取所有Office

listOffices()，其他获取也同理，例如要获取所有User，则是listUsers()，获取Patient则是listPatients()





**可扩展性：与mysql原生相同。**

**可维护性：观察数据库采用GUI，容易观察容易维护。**

**二． 部署**

**服务器与客户端均部署到本地。**

**三． 模块划分**

**功能模块请参考需求文档，在此不再赘述。**