前端预测试工具设计文档

# 前言

公司现有项目都是通过Nginx以前后端分离的方式实现，前端通过Http请求与后台通信，需做好接口和数据的对接。但在实际开发过程中，会因为沟通不及时、开发进度不一致或人员疏忽等原因导致接口对接失败，影响开发进度。

前端预测试工具则提供了管理接口的平台，同时提供给前端开发人员进行实际请求测试的功能，是一个给前后端开发对接的辅助工具，以提高开发效率和质量。

# 概要设计

此章节介绍了预测试工具的简要交互流程，实现框架，使用工具信息。

## 交互流程

预测试工具的设计思路是，由人工录入项目接口等信息，然后根据录入的信息启动一个模拟后台响应的http服务。前端代码完成后，修改Nginx配置的服务为此模拟服务，重启Nginx后则可进行联调。交互流程如下：



预测试工具是针对录入数据和启动模拟服务进行开发的项目。

## 实现框架

此项目使用Sqlite+Nodejs+Electron+Angularjs的框架实现，这些工具的使用版本和一些使用的注意事项在下一节中介绍。

Sqlite用于保存录入的项目和接口信息。

Nodejs为应用层，用于整个项目功能的实现，如将数据存入数据库或存为文件、生成并启动模拟的Http服务、模拟服务中与录入数据的对应响应等。

Electron作为与用户交互页面的容器，代替了浏览器的功能，并且可直接运行Nodejs代码。

Angularjs为页面代码的实现框架，负责展现内容和与用户交互。

框架结构如下图：



以上实现，类似于一个本地应用。此外，可在下一版本中加入一个微服务模块，用于上传和下载录入的项目接口信息，实现与别人的信息共享，并且可加入对录入的信息变更提醒的功能。

加入微服务后的框架如下图：



## 使用工具

开发需用到以下工具：

**Webstorm**：webstorm2016.3，代码编写工具。

**Nodejs**：v6.11.1，用于应用层代码编写；自带npm v3.10.10，依赖包管理工具。

**Angularjs**：v4.0.0，npm install @angular/cli@4.0.0 –save，页面代码编写框架。

**Redux**：v3.6.0，npm install redux@3.6.0 –save，页面数据管理工具。

**Webpack**：前端代码打包工具。下载好Angularjs后，执行ng new –ng4 project-demo 创建项目，进入project-demo下，执行npm install。Angularjs生成的项目有自动配置好的Webpack配置，直接使用即可。（project-demo是举例的项目名称）

注：修改Webpack的配置项为target： ‘node-webkit’。在project-demo\node\_modules\@angular\cli\models\webpack-configs\common.js中getCommonConfig方法return的数据项中修改。

**Electron**：v1.4.13，npm install electron@1.4.13 --save，npm install electron-cli –g，使用Electron必须的依赖包；npm install electron-packager，electron的打包工具。

**Sqlite**：v3.1.8，npm install sqlite@3.1.8 –save，使用sqlite必须的依赖包。

注：用electron打开使用sqlite的代码时，会遇到Cannot find module ‘..\project\_demo\node\_modules\sqlite3\lib\binding\electron-v1.4-win32-x64\node\_sqlite3.node’的问题，执行以下步骤解决

cd \node\_modules\sqlite3

npm install nan –save

npm run prepublish

node-gyp configure –module\_name=node\_sqlite3 –module\_path=../lib/binding/electron-v1.4-win32-x64

node-gyp rebuild –target=1.4.13 –arch=x64 –target\_platform=win32 –dist-url=https://atom.io/download/electron/ --module\_name=node\_sqlite3 --module\_path=../lib/binding/electron-v1.4-win32-x64

具体可参考<http://blog.csdn.net/lord_L/article/details/53996828>

开发中遇到的其他问题还需根据实际情况解决。

## 目录结构

项目目录结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| project |  | | |
|  | db | 与数据库连接并实现增删改查的代码 | |
| dist | Webpack将页面相关代码打包合并后的文件 | |
| sqlite | 保存数据库数据的文件夹 | |
| src | 开发页面代码文件夹 | |
|  | app | 页面按模块实现的代码 |
| control | 控制页面数据的代码，调用与数据库连接的代码 |
| styles | 按模块分写的样式文件 |
| *index.html* | 入口页面 |
| *com-util.ts* | 公共方法 |
| *styles.styl* | 样式合并入口 |
| *DateFormat.ts* | 日期处理 |
| …… | 其余依赖框架自动生成的必须文件或文件夹 |
| …… | 依赖框架生成的必须的配置文件等 | |
| 注：斜体表示为文件 | | | |

# 详细设计

此章先从模块的角度对项目的设计进行了详细说明，然后分别对数据库、应用层、视图层、数据管理方面的设计进行了详细说明，最后对下一版本欲实现的工具做了简要的设计介绍。

## 模块设计

在项目中，将录入信息分为四个部分进行管理：项目（project）、模块（modul；module在es6语法中有特定含义，所以此处用modul，并非拼写错误）、接口（interface）、数据（intdata）。四个部分依次为包含关系。规则如下：

* 一个项目包含多个模块；一个项目也可直接包含多个接口。
* 一个模块包含多个接口；一个模块也可包含多个模块。
* 一个接口包含多条数据。

此外，针对项目、模块和接口都可存在多条配置信息，配置信息是预设置的请求数据结构等。

四个部分包含关系如下图：



## 数据库设计

说明如何存储，数据结构等

## 功能设计

与数据库的交互

启动模拟服务的实现

## 页面设计

说明页面排版和交互等

## 数据管理设计

说明如果保存并响应数据的更改