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，使用工具和目录结构。

## 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 | |
| service | 实现模拟服务的功能代码 | |
| 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。

## 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（interf）、模拟数据（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。
* 一个接口包含多条模拟数据。

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响应数据结构模板等。

四个部分包含关系如下图：



## 数据库设计

数据库的设计以模块的划分为依据，第二章第四节说明了数据库的存放位置为项目下的sqlite文件夹，下面配合存储目录说明存储的整体结构。

若存在数据，sqlite文件夹目录结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| sqlite |  |  |  |
|  | *project.db* |  | 保存项目基本信息和配置信息的数据库 |
|  | pro-demo1 |  | 项目pro-demo1的专属文件夹 |
|  |  | *storage.db* | 保存项目pro-demo1的模块和接口的基本信息和配置信息，数据信息的数据库 |
|  |  | *demo1.json* | 某条数据信息保存的模拟响应数据 |
|  |  | *……* | 其余数据信息保存的模拟响应数据 |
|  | …… |  | 其余项目的文件夹 |

注：斜体表示为文件

以上目录存储的规则如下：

* project.db保存项目基本信息表（project）和项目配置信息表（pro\_config）。
* pro-demo1等专属项目的文件夹的名称根据项目名称确定。
* storage.db保存模块基本信息表（modul）、模块配置信息表（mod\_config）、接口基本信息表（interf）、接口配置信息表（int\_config）和数据表（intdata）。
* demo1.json等属于某条模拟数据保存的模拟响应数据文件，根据数据id和用户自定义名称命名，即“id\_custom1.json”。

因为公司现有项目都只以json格式进行数据传送，其他格式几乎用不到，所以暂时只支持json格式文件。

可看出针对每个项目都会创建一个文件夹来保存它的模块、接口和数据等信息。这样设计一方面方便对项目下的模拟数据进行存储；另一方面是为以后的上传下载功能做的准备，上传和下载时，只需保证项目基本信息不冲突，则可将此项目的文件夹内容全部上传。

下面分别为项目、模块、接口、模拟数据相关数据表的字段结构进行说明。

### 项目

项目相关的表格有项目基本信息表（project）和项目配置信息表（pro\_config），存在于project.db数据库下。两个数据表的字段设计如下：

项目基本信息表 project

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 说明 |
| id | INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL | 唯一标识 |
| name | TEXT NOT NULL | 项目名称 |
| create\_time | DATE NOT NULL | 创建时间 |
| modify\_time | DATE | 修改时间，只指项目基本信息被修改的时间 |
| path | TEXT | 项目默认的路径前缀 |
| port | NUMBER NOT NULL | 项目启动模拟服务时的端口 |
| status | TEXT NOT NULL | 表示项目是否被启动的状态 |
| comment | TEXT | 备注信息 |

还需满足一下规则：

* name不可重复。
* status为启动状态的项目port不可重复。

项目配置信息表 pro\_config

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 说明 |
| id | INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL | 唯一标识 |
| name | TEXT NOT NULL | 配置模板名称 |
| type | TEXT NOT NULL | 是何种响应模板，正确或者错误的 |
| content | TEXT NOT NULL | 模板内容 |
| parent | INTEGER NOT NULL | 所属项目的id |

* name 不可重复。

### 模块

模块相关的数据表有模块基本信息表（modul）和模块配置信息表（mod\_config），存在于所属项目文件夹下的storage.db数据库中。两个数据表的字段设计如下：

模块基本信息表 modul

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 说明 |
| id | INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL | 唯一标识 |
| name | TEXT NOT NULL | 模块名称 |
| create\_time | DATE NOT NULL | 创建时间 |
| modify\_time | DATE | 修改时间，只指模块基本信息被修改的时间 |
| path | TEXT NOT NULL | 模块默认的路径前缀 |
| parent | INTEGER | 所属模块id，若不存在，表示直接属于项目 |
| comment | TEXT | 备注信息 |

* name不可重复。
* parent相同的模块，path不能相同。

模块配置信息表 mod\_config

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 说明 |
| id | INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL | 唯一标识 |
| name | TEXT NOT NULL | 配置模板名称 |
| type | TEXT NOT NULL | 是何种响应模板，正确或者错误的 |
| content | TEXT NOT NULL | 模板内容 |
| parent | INTEGER NOT NULL | 所属模块的id |

* name不可重复。

### 接口

接口相关的数据表有接口基本信息表（interf）和接口配置信息表（int\_config），存在于所属项目文件夹下的storage.db数据库中。两个数据表的字段设计如下：

接口基本信息表 interf

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 说明 |
| Id | INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL | 唯一标识 |
| create\_time | DATE NOT NULL | 创建时间 |
| modify\_time | DATE | 修改时间，只指接口基本信息被修改的时间 |
| path | TEXT NOT NULL | 接口默认的路径前缀 |
| full\_path | TEXT NOT NULL | 接口最后的完整路径，应该是项目前缀+模板前缀+接口路径 |
| method | TEXT NOT NULL | 此接口的请求方法POST,GET,PUT,DELETE等 |
| parent | INTEGER | 所属模块id，若不存在，表示直接属于项目 |
| comment | TEXT | 备注信息 |

* parent相同的接口，path和method必须有一个不同。
* path中可带花括号如“/pathdemo/{id}”，花括号部分表示会变动的参数，id则是参数名。

接口配置信息表 int\_config

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 说明 |
| id | INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL | 唯一标识 |
| name | TEXT NOT NULL | 配置模板名称 |
| type | TEXT NOT NULL | 是何种响应模板，正确或者错误的 |
| content | TEXT NOT NULL | 模板内容 |
| parent | INTEGER NOT NULL | 所属接口的id |

* name不可重复。

### 模拟数据

模拟数据相关的数据表只有模拟数据信息表（intdata），存在于所属项目文件夹下的storage.db数据库中。字段设计如下：

模拟数据信息表 intdata

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 说明 |
| id | INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL | 唯一标识 |
| name | TEXT NOT NULL | 模拟数据名称 |
| code | NUMBER NOT NULL | 相应的code码，400或200等 |
| create\_time | DATE NOT NULL | 创建时间 |
| modify\_time | DATE | 修改时间，只指接口基本信息被修改的时间 |
| type | TEXT NOT NULL | 保存模拟数据的类型（text,file） |
| content | TEXT NOT NULL | 保存的模拟数据，type为text直接保存字符串；type为file保存文件路径 |
| status | BOOLEAN NOT NULL | true表示当前接口的响应数据 |
| parent | INTEGER NOT NULL | 所属接口id |
| comment | TEXT | 备注信息 |

* parent相同的模拟数据，name不可重复。
* parent相同的模拟数据，只能有一条的status为true。

## 功能设计

此项目主要分为项目信息的管理功能和实现模拟服务的功能，下面分别对这两部分的功能的设计进行详细说明。

### 项目信息管理

项目信息的管理依然分为项目、模块、接口和模拟数据来设计，需分别实现对四个模块的增、删、改、查功能，以及在保存前的规则验证。

由第二章第四节可知，项目下的db文件夹为保存与数据库交互的代码文件，db文件夹下代码结构设计如下：

|  |  |  |
| --- | --- | --- |
| db |  | |
|  | dbutil.js | 连接数据库、执行数据库语句、公共验证方法等 |
| project.js | 项目的增、删、改、查以及相关验证等 |
| modul.js | 模块的增、删、改、查以及相关验证等 |
| interf.js | 接口的增、删、改、查以及相关验证等 |
| intdata.js | 数据的增、删、改、查以及相关验证等 |
| config.js | 配置信息的增、删、改、查以及相关验证等 |

#### 公共方法

与数据库相关的公共方法的代码在dbutil.js中，方法直接绑定在window的dbutil属性中，即window.dbutil。所需公共方法如下：

* getDB

功能：连接数据库并得到操作对象

参数：path：要打开的数据库的路径

返回值：Database，数据库操作对象

* getRootDB

功能：专门连接保存项目信息的project.db数据库并得到操作对象

参数：N/A

返回值：Database，数据库操作对象

* getRootDB

功能：连接保存专属于某个项目的storage.db数据库并得到操作对象

参数：name：项目名称

返回值：Database，数据库操作对象

* sql

功能：执行sql语句的统一方法

参数：db：要操作的数据库对象；fun：要执行的方法

返回值：无

* validName

功能：验证名称是否已存在

参数

type：要验证何种表的名称（project,modul,intdata,config）；

paramArr：做验证所需的参数组成的数组，如当前名称，所属接口id

返回值：Promise 异步执行方法的实例对象

说明：存在项目名不可重复、模块名不可重复、同接口下的模拟数据名不能重复、配置信息表中，parent相同的项的name不能重复

* validPath

功能：验证路径是否重复

参数

type：要验证何种表的路径（modul,interf）；

paramArr：做验证所需的参数组成的数组，如当前路径，所属模块id

返回值：Promise 异步执行方法的实例对象

说明：存在parent相同的模块路径path不能相同、parent相同的接口路径path不能相同

#### 项目

项目与数据库相关的代码在project.js中，方法绑定在window的projectdb属性中，所需方法如下：

* getAllProject

功能：得到所有的项目基本信息

参数：无

返回值：Promise 异步执行方法的实例对象

* getStartProject

功能：得到所有状态为启动的项目基本信息

参数：无

返回值：Promise 异步执行方法的实例对象

* getProjectById

功能：得到指定项目信息

参数：id：项目id

返回值：Promise 异步执行方法的实例对象

* add

功能：添加项目

参数：data：项目基本信息对象

返回值：Promise 异步执行方法的实例对象

说明：在添加前，先验证项目名name，重复的不可添加；在项目添加成功后，还需进行项目一些初始化的操作，即创建专属于项目的文件夹和stroage.db数据库，并在数据库中创建modul、mod\_config、interf、int\_config、intdata表

* delete

功能：删除项目

参数：id：项目id

返回值：Promise 异步执行方法的实例对象

说明：在删除项目之前，检查项目是否启动，启动的不可删除；然后清理项目的专属文件夹和配置信息，即删除以项目名命名的文件夹，删除此项目下的配置信息

* uqdate

功能：修改项目基本信息

参数：id：项目id；data：项目基本信息数据

返回值：Promise 异步执行方法的实例对象

说明：修改成功后，需修改专属文件夹名称

* validPort

功能：验证项目端口是否相同

参数：port：当前端口

返回值：Promise 异步执行方法的实例对象

说明：验证已启动的项目中是否存在此端口

#### 模块

模块与数据库相关的代码在modul.js中，方法绑定在window的moduldb属性中，所需方法如下：

* getModulByParentId

功能：得到parent相同的模块基本信息

参数：parent：所属模块id，不存在表示直接属于项目

返回值：Promise 异步执行方法的实例对象

说明：若parent不存在，说明获取直接属于项目的模块信息

* getModulById

功能：得到指定模块信息

参数：id：模块id

返回值：Promise 异步执行方法的实例对象

* add

功能：添加模块

参数

data：模块基本信息对象；

parent：欲添加到的模块id，不存在表示直接属于项目

返回值：Promise 异步执行方法的实例对象

说明：添加模块前先检查名称是否重复，重复不可添加；再检查同parent下的path是否重复，重复不可添加

* delete

功能：删除模块

参数：id：模块id

返回值：Promise 异步执行方法的实例对象

说明：在删除模块之前，需先删除属于此模块的模块信息、接口信息、配置信息等，且是嵌套删除

* uqdate

功能：修改模块基本信息

参数：id：模块id；data：模块基本信息

返回值：Promise 异步执行方法的实例对象

说明：修改前，先检查名称name是否重复，再检查同parent下的path是否重复，重复不可更改；修改成功后，若修改了path或所属模块parent，需此模块下及其子模块下的接口interf的全路径full\_path

#### 接口

接口与数据库相关的代码在interf.js中，方法绑定在window的interfdb属性中，所需方法如下：

* getInterfByParentId

功能：得到parent相同的接口基本信息

参数：parent：所属模块id，不能存在表示直接属于项目

返回值：Promise 异步执行方法的实例对象

说明：若parent不存在，说明获取直接属于项目的接口信息

* getInterfById

功能：得到指定接口信息

参数：id：接口id

返回值：Promise 异步执行方法的实例对象

* add

功能：添加接口

参数

data：接口基本信息对象；

parent：欲添加到的模块id，不存在表示直接属于项目

返回值：Promise 异步执行方法的实例对象

说明：添加模块前先检查同parent下的path和method是否重复，重复不可添加

* delete

功能：删除接口

参数：id：接口id

返回值：Promise 异步执行方法的实例对象

说明：在删除接口之前，需先删除属于此接口的配置信息和模拟数据信息

* uqdate

功能：修改接口基本信息

参数：id：接口id；data：接口基本信息

返回值：Promise 异步执行方法的实例对象

说明：修改前，先检查同parent下的path和method是否重复，重复不可更改；修改成功后，若修改了path或所属模块parent，需修改此接口的全路径full\_path

#### 模拟数据

模拟数据与数据库相关的代码在intdata.js中，方法绑定在window的intdatadb属性中，所需方法如下：

* getIntdataByParentId

功能：得到parent相同的模拟数据信息

参数：parent：所属接口id；name：所属项目名称

返回值：Promise 异步执行方法的实例对象

* getIntdataById

功能：得到指定模拟数据信息

参数：id：模拟数据id；name：所属项目名称

返回值：Promise 异步执行方法的实例对象

* add

功能：添加模拟数据

参数

data：模拟数据信息对象；

parent：欲添加到的接口id ；

name：所属项目名称

返回值：Promise 异步执行方法的实例对象

说明：添加模拟数据前先检查同接口下的名称是否重复，重复不可添加；若添加的模拟数据status为true，将同接口下的其他模拟数据的status置为false

* delete

功能：删除模拟数据

参数：id：模拟数据id；name：所属项目名称

返回值：Promise 异步执行方法的实例对象

* uqdate

功能：修改模拟数据信息

参数：id：模拟数据id；data：模块基本信息；name：所属项目名称

返回值：Promise 异步执行方法的实例对象

说明：修改前，先检查同接口下名称name是否重复，重复不可更改；若此条数据的status为true，将同接口下的其余模拟数据的status置为false

#### 配置

配置与数据库相关的代码在config.js中，方法绑定在window的configdb属性中，所需方法如下：

* getConfigByParentId

功能：得到parent相同的接口基本信息

参数

parent：配置信息所属对象的id；

type：所属对象类型（project,modul,interf）；

name：所属项目名称，不存在说明获取的是项目配置信息

返回值：Promise 异步执行方法的实例对象

* getconfigById

功能：得到指定接口信息

参数

id：配置信息id；

type：所属对象类型（project,modul,interf）；

name：所属项目名称，不存在说明获取的是项目配置信息

返回值：Promise 异步执行方法的实例对象

* add

功能：添加配置信息

参数

data：配置信息对象；

parent：欲添加到的所属对象的id；

type：所属对象类型（project,modul,interf）；

name：所属项目名称，不存在说明是添加的是项目的配置信息

返回值：Promise 异步执行方法的实例对象

说明：添加前先检查同parent下配置信息，name是否重复，重复不可添加

* delete

功能：删除配置信息

参数

id：配置信息id

type：所属对象类型（project,modul,interf）；

name：所属项目名称，不存在说明是删除的是项目的配置信息

返回值：Promise 异步执行方法的实例对象

* uqdate

功能：修改配置信息

参数

id：接口id；

data：配置信息对象；

parent：修改配置信息所属对象的id；

type：所属对象类型（project,modul,interf）；

name：所属项目名称，不存在说明是添加的是项目的配置信息

返回值：Promise 异步执行方法的实例对象

说明：修改前，先检查同parent下的配置信息的name是否相同，若相同不可修改

### 模拟服务

模拟服务的实现可分为两方面，一是控制一个Http服务的启动和停止，二是让次服务响应的请求和保存的项目模拟数据对应上。

由第二章第四节可知，项目下的service文件夹下保存的是实现模拟服务功能的代码文件，service文件夹下代码结构设计如下：

|  |  |  |
| --- | --- | --- |
| service |  | |
|  | control.js | 控制管理http服务的代码 |
| analog.js | 模拟服务根据请求选择相应响应数据的代码 |

#### 控制服务

服务的管理除了启动、停止，还需保存管理已启动的服务，在control.js存在一下数据和方法：

* services

键值对象类型，用于保存当前已启动的项目。以项目名称为key值，启动的Htttp服务对象为实际值。

* start

功能：启动项目

参数：data：项目信息

返回值：Promise 异步执行方法的实例对象

说明

启动项目前，先验证项目状态status是否是启动状态，是的话不能启动；

然后验证port是否被占用，被占用不能启动；

将项目状态更新为启动；

启动后将Http服务对象以项目名称为key值存入services中；

将此Http服务的请求响应都带入当前项目id

* stop

功能：停止项目

参数：data：项目信息

返回值：Promise 异步执行方法的实例对象

说明：根据项目名称从services中获取Http服务对象，停止服务成功后将项目状态更新为停止；然后将此项目的服务对象从services中移除

#### 响应数据

模拟数据的响应就是根据请求信息和在项目启动时带入的项目名称，找到此条请求需要响应的那条模拟数据。此部分代码保存在analog.js中，方法如下：

* getData

功能：根据请求信息得到模拟数据的方法

参数

request：Http服务请求对象；

response：Http服务响应对象；

id：项目的id

返回值：无

说明：此方法为所有请求的响应入口，内部调用解析请求的方法得到应该想要的模拟数据放入response中返回响应

* getRootDB

功能：得到保存项目信息的数据库project.db

参数：无

返回值：Database项目信息数据库对象

* getProjectDB

功能：得到此专属项目的stroage.db数据库

参数：name：项目名称

返回值：Datebase 项目专属数据库对象

* getInterf

功能：获取对应项目下与请求匹配的接口

参数：path：请求路径；method：请求类型（POST、GET等）

返回值：Promise 异步执行方法的实例对象

说明：首先用path与interf表中的full\_path进行完全匹配查询；匹配不成功，则根据path层层筛选出所属模块，然后获取此模块下对应method且路劲中存在“{}”的接口，将full\_path转换为相应正则与path匹配；若还是匹配不到，则在异步返回的方法中传入空

* getIntData

功能：获取接口下被选定作为响应的模拟数据

参数：id：接口id

返回值：Promise 异步执行方法的实例对象

说明：查询此接口下status为true的模拟数据

## 页面设计

此章节将详细说明页面排版设计、交互设计以及样式编写设计。由第二张第四节可知项目下的src文件夹存储页面相关的代码，其中app文件夹保存的是页面框架和交互相关代码，styles保存的是页面样式代码。祥见以下各小节。

### 排版设计

页面排版依然按项目、模块、接口和模拟数据的划分来设计，页面实现的代码在app文件夹下，目录划分如下：

|  |  |  |
| --- | --- | --- |
| app |  |  |
|  | component | 页面实现自定义组件，包含列表基本框、配置信息框、输入框等 |
|  | project | 实现项目页面代码 |
|  | modul | 实现模块页面代码 |
|  | interf | 实现接口页面代码 |
|  | intdata | 实现模拟数据页面代码 |
|  | position | 实现展示当前位置页面代码 |

下面分别画出对项目、模块、接口和模拟数据的排版设计的简易图，并直接在图上进行说明。

#### 项目



项目详细信息图

项目添加图

项目修改图

#### 模块

模块列表图

模块详细信息图

模块添加图

模块修改图

#### 接口

接口列表图

接口详细信息图

接口添加图

接口修改图

#### 模拟数据

模拟数据列表图

模拟数据详细信息图

模拟数据添加图

模拟数据修改图

### 交互设计

此项目中与用户的交互可分为列表交互、表单交互、页面切换交互三个方面。列表交互和表单交互以项目为例说明交互的设计规则，页面切换交互以各部分的切换说明。

#### 列表交互

图

#### 表单交互

图

#### 页面切换交互

图

### 样式设计

由第二章第四节可知，样式的定义都在项目下src文件夹下的styles文件夹下，styles文件夹下是分类定义的样式文件，最后都被引入src下styles.styl文件内被使用。styles文件夹的目录结构如下：

|  |  |  |
| --- | --- | --- |
| styles |  |  |
|  | common.styl | 定义用到的公共样式，如全局字体、间距以及盒模型公共布局定义等 |
|  | define.styl | 定义样式用到的公共变量和方法，如统一border、background等 |
|  | arrow.styl | 定义项目中用到的各式各样的箭头图标样式 |
|  | button.styl | 定义项目中按钮的样式 |
|  | input.styl | 定义项目中输入框的样式 |
|  | select.styl | 定义项目中下拉框的样式 |
|  | form.styl | 定义项目中表单的样式 |
|  | list.styl | 定义项目中列表框的样式 |
|  | pop.styl | 定义项目中弹框样式 |

## 数据管理设计

redux是一个前端页面数据管理的工具，其关键在于提供良好数据结构的定义方式，严格控制数据的修改权限，提供方便的数据改变响应订阅方式，实现对前端数据的管理。

由第二章第四节可知，项目的src文件的control文件夹保存的是页面数据管理的代码，页面数据的定义也是按项目、模块、接口和模拟数据的结构划分，control文件夹的目录结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| control |  | | |
|  | common | 公共数据管理定义的代码 | |
|  | config | 配置信息数据管理定义的代码 | |
|  |  | *config.model.ts* | 单条配置信息的数据结构定义 |
|  | modul | 模块数据管理定义的代码 | |
|  | interf | 接口数据管理定义的代码 | |
|  | intdata | 数据管理定义的代码 | |
|  | project | 项目数据管理定义的代码 | |
|  |  | *project.action.ts* | 定义对项目数据执行操作的方法 |
|  |  | *project.model.ts* | 定义单条项目项目信息的数据结构 |
|  |  | *project.reducer.ts* | 定义项目要保存的相关数据和操作方法的实现 |
|  |  | *project.service.ts* | 定义响应页面时要对数据库执行的操作方法 |
|  | *app.reducer.ts* | 合并对所有数据操作的实现 | |
|  | *app.store.ts* | 合并所有定义的数据和定义的操作 | |

注：斜体字为文件

以上只以项目为例展示了文件夹内会存在的文件，其他文件夹与项目的类似。但配置信息下列出了config.mode.ts文件，表示对单条配置信息的定义，配置信息无其他定义，因为配置信息都依赖其他模块存在，已在其他模块中进行管理。单条配置信息的定义与数据库中配置信息表的定义类似，请参考第三章第二节。

下面说明各个部分对数据结构和操作方法的定义。

### 公共数据

公共模块数据的管理不需要model和service部分，因为公共数据基本数据组成单位，也不用与数据库交互。公共管理数据包含两项定义

**click**：对页面点击事件的管理，内不记录点击次数和当前点击状态。用于控制页面下拉列表点击消失等响应而设计。

**position**：记录当前的位置信息，如当前在何项目何模块。

公共模块的数据所需以下方法

* 添加点击次数：每点击一次，增加一次点击记录，并设置点击状态为激活，用于记录用户的实际点击操作
* 设置点击状态：设置点击状态，不增加次数，自定义设置点击状态
* 添加位置信息：在用户切换到子模块或子页面时，增加一个位置信息
* 删除位置信息：在用户返回上一个模块时，删除一个位置信息
* 重置位置信息：在用户直接从子模块跳出时，重置位置信息

### 项目数据

project.model.ts中对单个项目数据结构的定义，和数据库project表中的类似，只是还另外加了config项的数组，用于保存项目的配置信息。

项目数据首先定义项目键值对数据，以项目id为key值，项目信息为实际值得数据类型ProjectsEntities。然后定义项目管理数据如下

**ids**：所有项目id组成的数组

**entities**：所有项目的键值对数据

**currentProjectId**：当前被选中的项目id

项目管理数据需要以下操作

* 设置所有项目：将从数据查询得到的项目数组，转换为以上定义的项目管理数据类型；并根据项目id分别查询各个项目的配置信息存入config中
* 设置被选项目：响应页面选中项目的事件，设置当前被选中的项目id

响应页面操作并与数据库交互的方法存在project.service.ts中，存在以下方法

getAllProjects

功能：得到所有项目信息

参数：reject：在成功查询到项目信息后的回调方法

add

功能：添加项目

参数：data：项目数据对象；reject：数据库正确响应回调方法

update

功能：修改项目

参数：data：项目数据对象；reject：数据库正确响应回调方法

delete

功能：删除项目

参数：id：项目id；reject：数据库成功响应的回调方法

### 模块信息

modul.model.ts中对单个模块数据结构的定义，和数据库modul表中的类似，只是还另外加了config项的数组，用于保存模块的配置信息。

模块数据首先定义项目键值对数据，以模块id为key值，模块信息为实际值得数据类型ModulsEntities。然后定义模块管理数据如下

**ids**：所有模块id组成的数组

**entities**：所有模块的键值对数据

**currentModulId**：当前被选中的模块id

模块管理数据需要以下操作

* 设置所有模块：将根据项目名称从数据查询得到的模块数组转换为以上定义的模块管理数据类型；并根据项目id分别查询各个项目的配置信息存入config中
* 设置被选模块：响应页面选中模块的事件，设置当前被选中的项目id

其余类似项目数据管理的设计，暂时不再一一赘述。

### 接口信息

与项目数据管理设计类似，暂时不再一一赘述。

### 模拟数据信息

与项目数据管理设计类似，暂时不再一一赘述。