

NeurodisM 文档

NeurodisM 文档

项目技术栈

环境

前端

后台

快速上手

1. 安装Node.js和R

2. 启动后台服务

3. 前端开发

编译

4. 后台开发

5. 发布到服务器

项目文件结构

后台数据和API

数据库

后台API

nginx 反向代理

安装

添加配置

开发者

项目技术栈

环境

[Node.js](#): Javascript运行环境和包管理器**npm**

前端

[vue](#): 整个项目的组件框架

[vue-cli](#): 快速生成搭配Webpack构建工具的项目文件

[element-ui](#): UI组件库

[vue-data-tables](#): 基于vue2和element-ui的表格组件

[echarts](#): 数据可视化图表库

[vue-echarts-v3](#): 将echarts嵌入到vue组件

[cytoscape.js](#): 强大的网络图可视化库

后台

[express](#): 基于Node.js的简洁的web开发框架

[mongodb](#): 以文档形式存储的非关系型数据库

[mongoose](#): 基于Node.js的mongodb中间件，API简洁友好

[nginx](#): 后台服务器，在本项目中用来做反向代理

[R](#): 卡方检验计算

快速上手

1. 安装Node.js和R

安装Node.js后，需要先安装依赖组件

```
npm install express
npm install mongoose
npm install csvtojson
```

2. 启动后台服务

```
node server.js
```

当出现

```
server started
mongodb: coonnected
```

时，说明后台服务开启成功。此时网站<http://localhost:8088/neurodism>已经可用。

如果需要在关闭终端后仍保持服务运行，可以使用[pm2](#)来启动服务。

3. 前端开发

上面两步只是开启后台服务，网站使用的是编译后的前端静态资源文件。接下来开启前端开发环境。

```
cd neurodism
npm install # 只运行一次安装依赖
npm start
```

此时前端开发环境使用<http://localhost:8080>，修改代码保存后，网页将会自动热更新。

编译

开发代码需要编译成静态文件

```
npm run build
```

编译后的静态文件在 `./client` 里

4. 后台开发

安装**nodemon**并使用**nodemon**来启动后台服务以实现后台服务自动更新

```
npm install -g nodemon
nodemon server.js
```

5. 发布到服务器

只需上传如下文件到服务器上，并运行一、二步的代码即可在<http://ip:8088/neurodism>访问该项目。

服务器上请使用之前所述的pm2来启动后台服务。

```
server.js
neurodism
  client
  server
```

项目文件结构

server.js: 后台服务文件，监听端口 `8088`。使用 `neurodism/server` 里定义的静态资源重定向和API。

neurodism: 项目文件夹

- **client**: 编译结果
 - **static**: 静态文件资源
 - **index.html**: 入口网页，直接打开无效
- **server**: 后台文件
 - **data**: 后台静态数据
 - **ci.json**: 染色体数据文件
 - **dp.csv**: 疾病对的p-value值
 - **genes.csv**: 基因在染色体上的位置信息
 - **mut.csv**: 所有突变数据
 - 以上三个文件为建库数据
 - **r**: R 脚本
 - **p_value.R**: 计算P-Value
 - **api.js**: 项目使用的静态资源 `/client` 的重定向以及API
 - **create.js**: 构建MongoDB数据库的建库代码
 - **db.js**: 数据库连接以及数据格式申明
- **src**: 代码及前端静态资源
 - **assets**: 前端静态资源
 - **components**: **vue**组件库
 - **disease**: 疾病相关组件
 - **Disease.vue**: 疾病主页
 - **DisRelDiagram.vue**: 疾病关系网络图
 - **GoCChart.vue**: 与该疾病有关的基因显示在染色体上的图
 - **gene**: 基因相关组件
 - **Gene.vue**: 基因主页
 - **GeneRelDiagram.vue**: 基因关系网络图
 - **GeneRelDiagramCy.vue**: 使用Cytoscape.js生成的大规模基因关系网络图，用于主页展示
 - **MoGChart.vue**: 与该基因有关的突变显示在基因上的图

- **relation**: 关系相关组件
 - **Relation.vue**: 展示基因之间或疾病之间的关系数据
 - **DGRelation.vue**: 展示用户输入的疾病基因相关数据
 - **RelDiagram.vue**: 展示用户输入疾病与数据库中疾病的关系
- **search**: 搜索相关组件
 - **Input.vue**: 用于输入基因/疾病的自动完成输入框
 - **Search.vue**: 搜索组件
- **MainPage.vue**: 网站主页, 展示疾病关系网络图和基因关系网络图
- **Table.vue**: 表格组件, 使用**vue-data-tables**, 用于展示突变数据
- **router**: vue路由
 - **index.js**: 用于配置组件之间的路由
- **static**: 开发静态文件, 编译后会在 `/client/static` 中
- **App.vue**: 应用的布局以及固定的组件
- **main.js**: 项目主文件, 用于申明全局使用的组件和资源, 定义项目的入口组件
- **index.html**: 开发时的项目入口文件

其他文件为**vue-cli**初始化项目时生成的配置文件。

后台数据和API

数据库

使用**MongoDB**, 其存储格式类似于JSON。数据库文件如下如下 (数据格式参考 `db.js`)

words: 字典数据库。存储所有基因和疾病 (在**mut.csv**中出现的) 的名称和类型 (`'g'/'d'`)。

mutations: 存储**mut.csv**的信息, 其中 `pt` 代表疾病

genes: 存储基因位置信息

diseases: 存储疾病信息。 `d`: 疾病名称, `m`: 相关突变的数量, `g`: 相关基因列表

genePairs: 存储基因对的信息。 `g1/g2`: 基因名称, `d`: 共享的疾病

diseasePairs: 存储疾病对的信息。 `d1/d2`: 疾病的名称, `v`: 存储三个p-values, `s`: p-values的数量, `g`: 共享的基因列表, `m`: 共享的突变列表

后台API

`/api/dp/:name`: `:name` 为参数, 允许的参数为 `all` 或者 `disease`。 `all`: 返回所有疾病对数据, `disease`: 返回与指定疾病有关的所有疾病对

`/api/gp/:name`: `:name` 为参数, 允许的参数为 `all` 或者 `gene`。 `all`: 返回所有基因对数据, `gene`: 返回与指定基因有关的所有基因对

`/api/disease/:name`: `:name` 指定某个疾病。返回 `{dis: 该疾病在疾病库中的数据, muts: 与该疾病有关的突变库中的数据}`

`/api/gene/:name`: `:name` 指定某个基因。返回 `{gene: 该基因在基因库中的数据, muts: 与该基因有关的突变库中的数据}`

`/api/word/:type`: `:type` 为参数, 允许的参数为 `all`, `gene`, `disease`。 `all`: 返回所有字典, `gene/disease`: 返回指定类型的字典

`/api/chr/all`: 返回所有染色体的数据信息 (`data/ci.json`)

`/api/relation/:type/:n1/:n2`: 根据 `:type` 选择返回疾病对或者基因对的数据。

`/api/pvalue/:in`: 参数为一组基因数据, 返回由R脚本得到的P-Value列表。

以上所返回的数据的格式与数据库中格式一致 (参考 `db.js`)

nginx 反向代理

此步骤可忽略

上述后台服务启动的端口是 `8088`, 而HTTP的端口是 `80`, 为了在网址上去除 `8088`, 需要使用nginx进行反向代理。

安装

根据官方文档进行安装: [Installing nginx](#)

添加配置

```
cd /etc/nginx/conf.d
```

新建一个配置文件, 名称自定, 扩展名为 `.conf`, 输入如下内容

```
upstream nodejs {
    server 127.0.0.1:8088;
    keepalive 64;
}
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name yourdomain; # 将yourdomain改为你的域名
    location / {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_set_header X-Nginx-Proxy true;
        proxy_set_header Connection "";
        proxy_pass http://nodejs;
    }
}
```

重启nginx

```
service nginx restart
```

开发者

谢子聪

