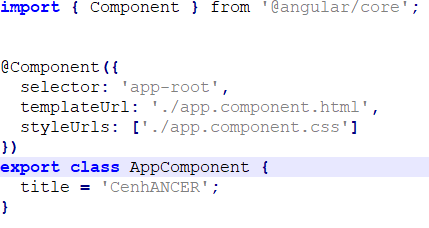
使用的angular2的前端模板作为网页搭建的，angular2自己就可以构建完整的前端框架以及动态的渲染，使用的是nodejs进行安装。首先需要安装nodejs，然后使用nodejs安装angular-cli，这个是angular2自己的一个管理工具。安装办法参考下面的网页（其实是我随便搜出来的）

<https://www.jianshu.com/p/ebf8064a2679>

安装完成之后是创建一个新的项目，ng new出来

比如我的项目中CEdb-portal\cedb\CEdb，这个CEdb就是ng new出来的，他会自动创建下面一堆文件配置的内容以及src和node\_modules文件夹。之后有需要安装的node模块会自动装进node\_modules。而src文件夹就是网页的内容

这里要提的是其中外层显示的页面是index.html。其中<app-root></app-root>则是网页启动后被填充的位置，之后angular的网页都是类似的填充格式。由<app-xxx></app-xxx>作为结构，另外一个组件绑定该接口进行填充。而<app-root></app-root>绑定的接口就是src\app\app.component.html。由app.component.ts进行绑定



可以看到app-root在这里进行了声明

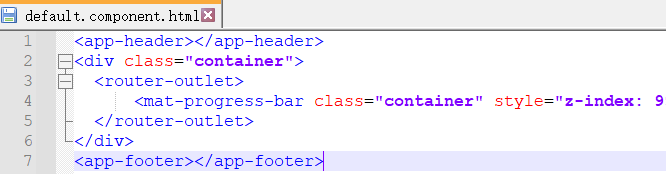
然后在app.component.html中，也可以看到<router-outlet></router-outlet>，这个是表示由路由进行分配填充，路由就是网址的意思，首先是baseurl + 网页的分url。比如www.google.com/search?q=angular2。www.google.com就是baseurl ，search就是分url，q就是url中放进的一个参数，angular2就是参数的值。？是分隔url与参数的符号

而angular2这里就是对分url对应的网页进行分配的，指定不同的url对应的不同功能组件。

路由的分配是在app-rounting中设置的



需要注意的是，本项目使用了一个比较特殊的方式这只路由，我使用了两层路由的重定向。这里把住页面重新定向到了default的组件里，再由default重新定向路由。所以可以直接看default组件



页面是首先固定注入了header和footer两个，然后router-outlet内就注入成路由分配的内容。这两个在share文件夹中。这里另外提一点，header和footer是在default.module中导入的share.module中声明导入的。每一个组件在使用时都是需要在module中声明的。

然后从app-routing导入到default时，路由分配是由default-routing进行分配的



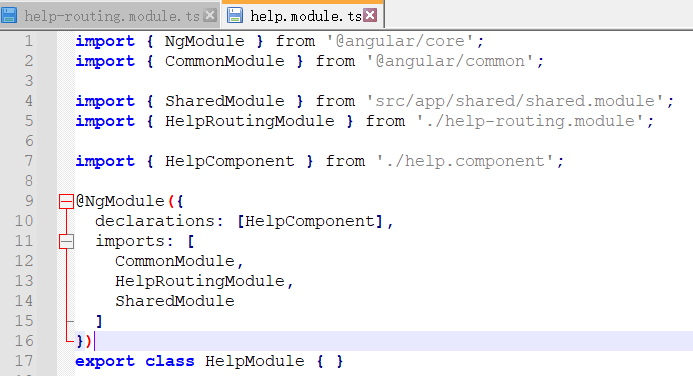
不同的path对应不同的组件。

完成路由的分配之后，就是对每一个组件进行完善。

所有的组件都存在了modules文件夹中。首先根据angular cli创建一个新组件

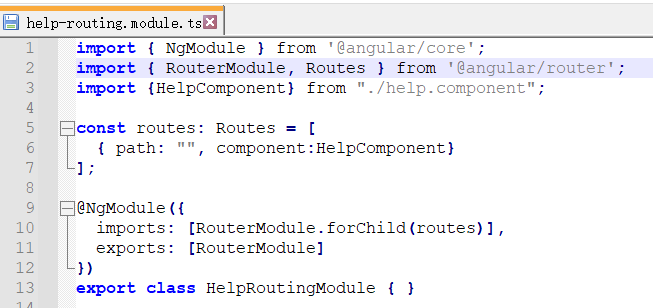
ng g c xxx -d 其中xxx是组件名称，-d是表示先看看该命令会做出哪些改动。看看情况是不是自己想要的，确定之后再去掉-d 运行ng g c xxx。另外，在最上层创建组件时需要加上--skip-import参数，因为cli默认会将组件导入到上一层的module中，但是由于我们是在default中用路径导入的，所以不用在app中声明，而在次一级的组件中则可以不用加这个参数。每个最上层的组件都需要创建一个module和routing.module。方法就是ng g m xxx --routing。具体命令可以百度。https://angular.cn/cli/generate#module-command

会继续创建default中指出的模块

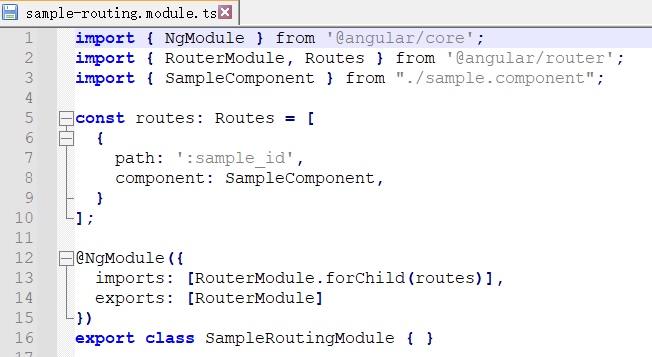


比如这里的help模块，对各个module进行导入。包括sharedmodule和helproutingmodule，以及组件本身进行声明。这样在default中就知道baseurl/help指向哪一个组件了。

而在helproutingmodule中，进一步规范了哪一个url是指向了什么组件。



这里再做一次routing分配的目的是为了方便有些网址需要顺便传参数，比如

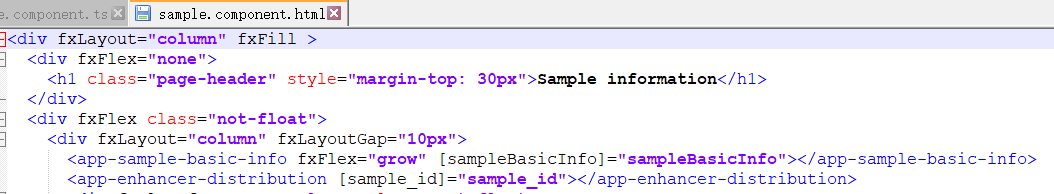


可以看到path里加了个:sample\_id。这样写法就可以是baseurl/sample/sample\_id。这样就可以在组件中直接获得sample\_id，进行页面渲染了。关于在url中进行传值的内容之后上面也提到过，也可以使用?的方式进行指定传参。也是可以的。

组件中的主要结构是component.html、component.css、component.ts。Module只是用来声明组件的位置。大部分的工作其实是在这三个文件中进行的。Html和css是这个组件的样子，ts是这个网页的功能。Html中就是正常的html语言，当然，在angular中，angular支持很多在html里的表达式，用来做传值使用，这个就比较细节了，具体麻烦自己查一下，最简单比如{{ value }}，这种双大括号可以用来传值，以及一些写法可以用来做大小或者字符相等的判断。Css样式文件，每一个组件都是使用了自己的样式文件，想要应用在全局的样式需要在scr\styles.css中设置，这样的好处就是样式之间不会相互干扰。

接着是ts部分，这部分的内容需要学习一点typescript的基础，主要用来实现页面的一些功能以及传值用，这部分使用的东西较多，还是需要自学的。待会只简单讲讲传值怎么实现。

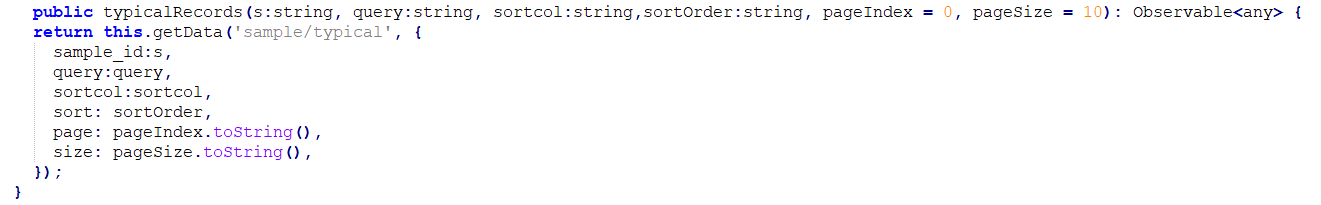
而在组件之内，也可以创建子组件，用子组件来填充。如下图，app-sample-basic-info就是子组件，在这里填充进去。子组件的声明在sample.module中。



这是angular部分的文件结构，还有部分细节，包括数据模式的定义文件等，都麻烦自己看看吧。

然后是传值的部分，这部分内容是和flask联系在一起的。由flask提供api接口，由angular进行请求，这样进行数据库的访问。

请求的部分我一般写成了api.service文件，这里我在BaseHttpService中定义过getData函数，所以是直接进行了使用，比如下图



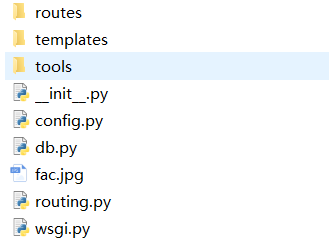
意思就是使用如下的参数对sample/typical的api进行访问。这里需要说的是，baseapi的定义在BaseHttpService中就写好了，所以实际请求的api是baseapi/sample/typical。

然后在实际脚本中使用typicalRecords函数就可以完成请求。同时，由于需要兼顾进度条的问题，所以我把函数又封装了一下，在datasource文件里，只是为了传值和解析返回值方便。

剩下，网页的表格呀，自己功能如何实现等问题，麻烦就自己百度解决，每个人想实现的功能不同，写法自然是不同的。

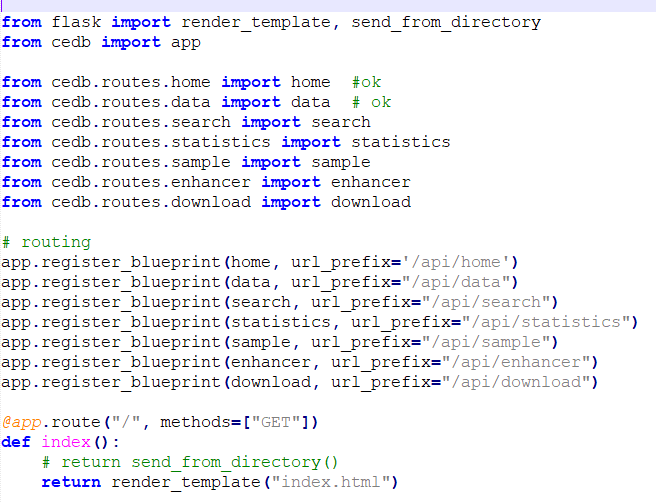
然后是后台的问题，本项目使用的后台是python的flask包，当然，用什么都可以，因为angular的网页架构不依赖于后台。后台仅仅是提供api接口，这代表什么东西提供接口都行。

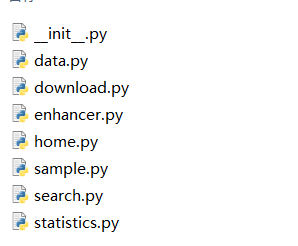
首先是flask的文件结构。

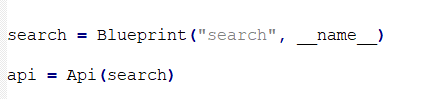


其中重要的是init，和routing文件。剩下的文件都是数据库的配置文件。

Init也是做模块导入的工作，照抄就行，routing文件则是实际提供路由的文件，他指示了哪一个模块提供哪一个api。例如search模块提供了/api/search的api。而search模块在routes文件夹下面

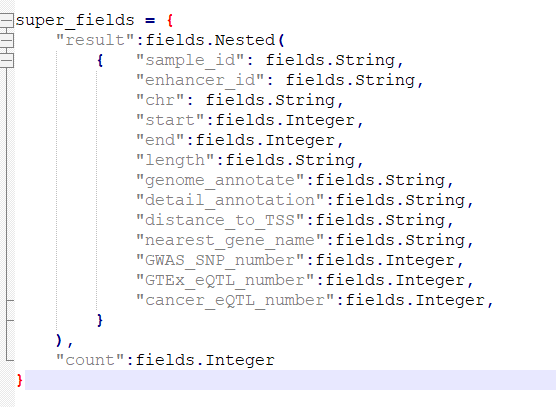






这部分就是在声明search模块时一个api接口。里面的字符和routing文件中对应

然后实际传值就两个部分，一个绑定数据格式，一个是mongodb访问数据





可以看到@marshal\_with(super\_fields)绑定了上面的数据格式，这么做是因为，在做完检索之后，可能会检索出其他不需要的字段，用这种方法可以直接去掉这些字段。同时也是为了规范数据格式。当然，如果数据在数据库中不好规定格式，这部分绑定也可以去掉。

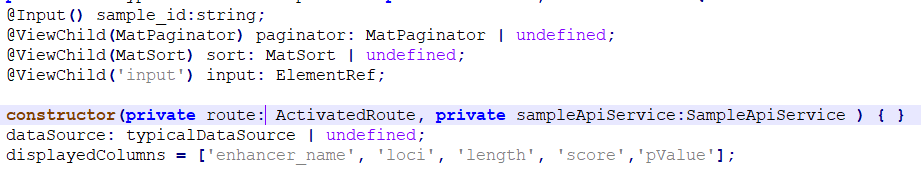
然后是parser获取参数。得到参数之后构建检索条件。然后再数据库中进行检索，检索完毕之后则传回值。同时api.add\_resource将该class绑定到super的api上，结合之前的api，整个api就是baseapi/sample/super了。这里提一下baseapi是自己设置的，多数情况下就是www.zijidewangzhi.com/api，就是这个。

之前提到angular的service脚本进行请求，flask返回请求结果到service，传到datasource进行结果处理，然后由组件进行结果展示。

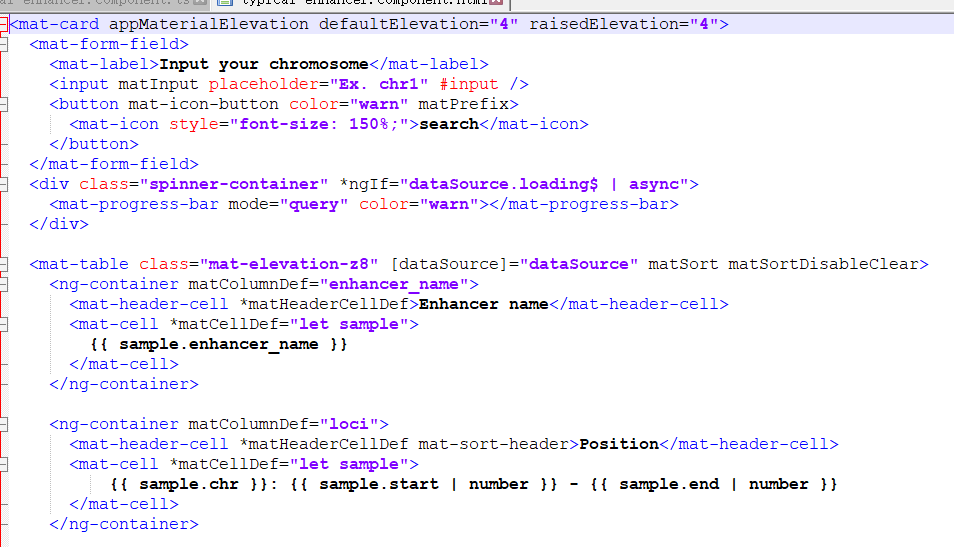


在组件中的代码大致就是如此。具体的表格渲染的部分是使用的angular自己的工具mat-table进行的。

需要配置的部分就是，声明一下表格的页码和排序。然后定义需要展示的列（displayedColumns）。



接着到html中写表格即可



结构如图，在ts文件中已经声明过dataSource，直接注入到mat-table中。\*matCellDef="let sample"声明将datasource的每一个结果赋值为sample的变量。然后开始写值。这样就完成了表格的渲染。

以上就是数据库的基本结构。

最后是后台数据库的问题，我个人使用mongodb数据库储存数据，因为方便python进行操作，当然也可以用mysql进行，这个python操作困难点，但是其他语言的接口应该写的也很好，如果不是使用python作为后台也可以考虑。Mongodb的用法相当简单，这个部分就不多讲了。