#### 开发手册

**注意：这个文档中出现的命令，如果没有特殊指明运行目录，则全部是在annotationTool2/目录下执行的。**

### 目录说明

1. src/：后端服务器的全部代码在这个目录中
2. views/：网页前端的全部代码在这个目录中
3. build/,config/：编译前端目录所需的配置文件，一般不修改
4. dist/：前端文件生成的结果目录，每次编译网页前端会将生成的文件放在这个目录中
5. node\_modules/：依赖目录，网页前端和后端服务器所依赖的库都会放在这个目录中

### 设计概览

##### 前端：

网页前端使用Webpack技术将多个文件合并成少量几个文件，入口为views/main.js。

前端使用Vue作为整体框架构建单页应用，views/router目录下存放了前端的路由，views/pages目录下存放了每个页面，views/components目录下存放一些共用的组件。

当修改了前端的文件后，首先需要使用*语法检查命令*检查所写的代码有没有明显的错误，然后*前端编译命令*将前端文件编译。（在使用Docker和不使用Docker的情况下的命令有所不同，可在“命令列表”章节查看具体的命令）

##### 后端：

后端服务器的全部代码文件都在src/目录下，入口为src/app.js，所有的配置均由src/config.js文件载入，其他文件使用src/config.js提供的数据。后端服务器使用Node.js作为编程语言，使用Koa作为HTTP Server框架，使用mongodb作为数据库（使用mongoose库操作mongodb）。

src/controllers/目录下的文件是每个API接口的控制逻辑，src/models/目录下是数据库模型，src/services/目录下是一些公用的功能。

### 数据库设计

标注系统支持多个数据集，因此src/models/dataset.js文件描述每个数据集的属性，包括名称（title），描述（description）和是否被删除（is\_deleted）等。

但是每个数据集有哪些文本则是由src/models/dataset\_item.js描述，每个dataset\_item表示一个文本，每个文本由id和content组成，并通过dataset属性表示属于哪个数据集。

每个数据集可以关联多个任务，任务在src/models/task.js文件中表示。**每个任务属于且只属于一个数据集，每个数据集可以有0个或多个任务**。

src/models/task.js记录了每个任务的基本信息和一些标注信息，包括名称（title），实体标签列表（tags）以及这些实体标签列表如何分栏（tag\_splits），还有关系标签列表（relation\_tags）等。

但src/models/task.js只记录了任务的基本信息，具体的标注内容是由src/models/task\_item.js记录的。task\_item关联了task和dataset\_item，每个task\_item表示“一个任务对于一个数据文本的标注结果”，每当用户完成对一个文本的标注，就会创建一个对应的task\_item。

task\_item储存了用户对每个文本的标注结果，主要包括实体标注（tags）和关系标注（relation\_tags），具体的储存格式可查看src/models/task\_item.js文件的注释。

### 控制逻辑

在src/controllers/目录下的文件储存了每个API的控制逻辑，具体解释如下：

1. src/controllers/index.js：主要包括登入和登出的逻辑
2. src/controllers/dataset.js：主要包括创建和删除数据集，创建和删除任务
3. src/controllers/machine.js：曾经主要用于主动学习功能，现在基本上用不了了
4. src/controllers/task.js：后端服务器最复杂的内容部分，主要包括设置task的实体标签和关系标签等。注：**一些以download\_task开头的API现在已经不可用了，请使用download\_task\_triple\_std\_v2下载标注结果**。每个API的控制逻辑可结合assert语句进行理解。

在修改了src/目录下的文件后，首先需要使用*语法检查命令*检查所写的代码有没有明显的错误，然后可以使用*运行服务器命令*进行调试。

### 命令列表

**注意：这个文档中出现的命令，如果没有特殊指明运行目录，则全部是在annotationTool2/目录下执行的。**

##### Docker环境：

1. *语法检查命令*：sudo docker-compose run --rm web npm run lintfix
2. *前端编译命令*：./scripts.sh compile
3. *运行服务器命令*：sudo docker-compose run --rm -p 8800:8800 web node src/app.js

##### 非Docker环境：

1. *语法检查命令*：npm run lintfix
2. *前端编译命令*：npm run build
3. *运行服务器命令*：node src/app.js