第4周YTM项目启动

NPM项目建设

我们学习了NPM的基本用法，即安装和介绍。在一些大型项目中，比如vue和其他开源前端模板，我们只需要通过两行命令:

```
git克隆xxx npm i
```

您可以直接快速安装和构建项目依赖项，并能够使用npm run dev等命令直接启动项目。

比如用直接启动一个项目，这是怎么做到的？

节点项目的构建依赖于一个名为package.json的文件，npm执行的大部分任务都与之相关。

Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.http://nodejs.cn/learn/the-package-json-guide·http://nodejs.cn/learn/the-package-lock-json-file

任务1

创建一个名为YTM-KOA的项目，临时引入axios、KOA、koa-jwt、音乐元数据、jpeg-js依赖项，然后使用npm开始运行脚本的命令:

```
节点main.js
```

在ytm-koa文件夹中，创建一个新文件main.js，并编写以下内容:

```
const Koa = require('koa'); const app
= new Koa();

app.use(async ctx => {
```

```
  ctx.body = 'Hello World';
});

app.listen(3000);
```

运行以下代码

```
npm i npm

start
```

项目成功启动并访问localhost:3000查看Hello World。

本地模拟环境构建

由于实习环境无法提供数据库，我们用本地的方式模拟了mongoDB数据库，mongo的基本格式其实是Json。

需要实现一个libraryInit (path)方法，功能是在APP启动时读取/Library下的所有音频文件(用户需要提前放一些MP3文件，数量在100个左右)，通过music-metadata library获取文件的标签信息，生成index.json文件存储在/Library中，格式如下:

```
    "track_id": "",
    "title": "", "artist":
    ["", ""], "album": "",
    "album_id": "",
    "genre": "",
    "copyright": "",
    "length" : "",
```

```
    "曲目编号" : 0,“质量” :

    "标准","文件" : "文件路

    径"

}
```

其中track_id的生成采用将艺人、片头、专辑三个字符串串联起来，用MD5加密取前16个字符的方法。track_id是曲目在曲库中的唯一索引值，因此不同的歌曲不能重复。(即track_id在数据库中不是严格唯一的，后期会有多个音质文件对应同一首歌)

Error! Hyperlink reference not valid.有关信息对应的标签密钥的更多信息，请参考https://www.npmjs.com/package/music-metadata。

此外，如果有专辑插图图像，则标记中的imageBuffer存储在/Library/cover/<album_id>.jpg注意:有时imageBuffer是PNG，需要使用jpeg-js库进行编码保存，质量为100。

这个JSON文件是连续的，存储在一个单独的行结构中，即每行一个上述对象。

每个文件的数据库创建应该是异步和并行的，使用各种方法将并行线程控制在有限的数量内。具体的软件行为是通过walk等方法建立MP3文件列表后，在n线程的线程池中对列表中的每个文件执行一个类似于indexCreate (filePath)的异步方法。该函数负责读取文件的id3Tag，获取上述信息，写入JSON变量，将imageBuffer转储到文件中，然后阻止写入/Library/index.json(即，如果index.json正在被另一个线程使用，则该函数应等待，直到它可以访问该文件)在创建每个文件后，在控制台中记录以下结构:

```
索引已创建:<曲目标识> <文件>
```

这个线程池n中的线程数应该是通过os模块获取CPU的信息来确定的。同时写一个libraryLoad (filePath)方法，直接读取index.json加载到内存，返回object。

编写一个libraryUpdate (lib，filePath)方法来更新index.json文件，该文件包含:

1. 比较和添加未编制索引的MP3文件

2. 删除在本地搜索中找不到相应文件的条目

Ps:这个库对于用户的索引track_id是唯一的。但是，在文件级别，文件路径是唯一的索引值。以上函数名是自己拟的。

libraryLoad(filePath)和libraryUpdate(path)应该占用文件，直到运行进程结束，然后释放该占用。所以现在逻辑非常清晰:在' npm start '启动项目后，首先尝试' libraryLoad(filePath)'，如果文件不存在就运行' libraryInit(path)'。如果文件存在，请挂载。然后((lib) => { libraryUpdate(lib，filePath) })'，完成索引库的加载和刷新。

“云”上的数据

在本项目中，我们使用非关系数据库mongoDB来构建整个系统的数据库系统。

mongo的基本概念和介绍:

Error! Hyperlink reference not valid.https://www.runoob.com/mongodb/mongodb-intro.html

Error! Hyperlink reference not valid.在node中，我们使用mongoose库来处理Node与MongoDB的交互。事实上，Node原生集成了对MongoDB的支持。js，但是mongoose中间件更容易使用。猫鼬中文手册:http://mongoosejs.net/docs/index.html

在本地安装MongoDB库之后，我们需要4个数据库:用户、库、播放列表、历史，用于后续项目。

其中，图书馆的数据结构前面已经指出，总馆藏命名为index，用户的私人图书馆馆藏命名为u_\<uid\ >，结构如下:

```
{
    " type ":"曲目/专辑/播放列表"，" id": "id "

      ，

     "添加日期":
}
```

用户的数据结构如下，不需要管理集合:

{

```
            " uid ":" " ,

            "名称": "" ,

            "秘密": "" ,

            " subscribe": "Premium " ，"

            subscribe_expired ": ，

        "最后一次登录": ，

        “正在播放”:“machine_id”
}
```

播放列表分为总索引集合和单列表集合，结构如下:

总集合被命名为索引

```
{

    “PID”:“id” ,

    "作者":" uid " ,

    "名称": "" ,

    "描述": "" ,

    【补充】:0

    “喜欢”:0 ，
```

```
“共享”:0，

“玩过”:0，

“公”:真的，

“图像”:“路径”，

"类型":"播放列表/专辑"，

"最新更新":

}
```

单列表集合是< pid >:

```
{
    " tid": "track_id "，"
    order": 0 // order
}
```

同时，用Mongose对Mongose数据库的操作替换前面构建的库的所有文件处理相关函数，将index.json迁移到MongoDB，但保留封面中的内容，仍然以本地文件路径索引的形式存储。

在向MongoDB等数据库写入数据时，可以忽略阻塞问题，中间件和数据库引擎会自行处理并发问题。