Dian团队2017年冬招后台培训

主讲人——Dian573 单勇

从需求开始

> 设计一个通讯录

我们需要哪些基本功能

1.获取所有联系人
2.修改某个联系人的信息
3.添加联系人
4.删除联系人

MVC模型

后台中的MVC分层

具象	抽象	分层	意义
一本专门记录联系人的笔记本	数据库	Model层	单纯的数据操作
专人负责查找、更新、添加、删除联系人信息	基本操作的逻辑	Controller层	其他所有逻辑、参数验证等
前台窗口，负责传入命令和传出结果	后端对外暴露的接口	View层	单纯的路由操作

扩展：从移动端、Web前端、后端看MVC架构

设计的两个输出

- API文档
- 测试样例

Restful API设计

1. 每个URL都是一个资源，只能出现名词，不能出现动词

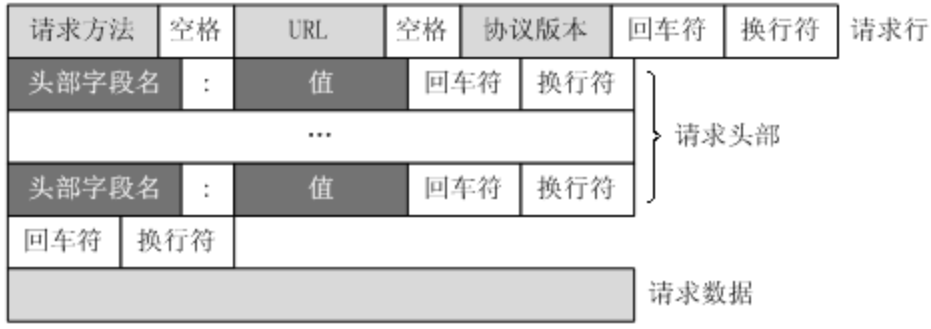
```
GET /zoos：列出所有动物园
POST /zoos：新建一个动物园
GET /zoos/ID：获取某个指定动物园的信息
PUT /zoos/ID：更新某个指定动物园的信息（提供该动物园的全部信息）
PATCH /zoos/ID：更新某个指定动物园的信息（提供该动物园的部分信息）
DELETE /zoos/ID：删除某个动物园
GET /zoos/ID/animals：列出某个指定动物园的所有动物
DELETE /zoos/ID/animals/ID：删除某个指定动物园的指定动物
```

(例子来源:阮一峰博客)

Restful API设计

2. HTTP Method的规范使用

3. 重新看HTTP协议




```
- body常用于POST、PUT等请求（需要携带结构化对象）
- headers常用于GET请求（类似于`baseurl?page=2&limit=3`）
```

（图片来自简书）

测试样例

1. 自动化测试
    - 可复现
    - 减少人力劳动
    - 易于集成
2. 测试驱动开发
    - 我们希望程序具有哪些功能
    - 我们希望程序能处理哪些错误
    - 验证接口输入输出的正确性

Nodejs+Express+MongoDB后台技术栈

为什么用Nodejs?

1. 基于Google V8 Engine
2. 高并发
3. 异步执行

Javascript ES6初探

Javascript广泛用于Web/Nodejs/移动开发，一种弱类型解释型动态语言，天生支持异步

1. 基本数据类型

```
//constant integer
const a = 1;
//string
let b = 'dian';
//object
let c = {
    hello: 'world'
};
//property reference
let d = c.hello;
//float
let e = 1.5;
let f = parseFloat(e);
// string template
let g = `cast e as integer => ${parseInt(e)}`;
```

Javascript ES6初探

2. 函数与函数式编程

```
//define a function
function welcome() {
    let msg = 'welcome to Dian!';
    console.info(msg);
    return msg;
}
//function is also variable type
let func = welcome;
func();

//functional operations
let list = [1, 2, 3];
list.map(x => x.toString());
list.filter(x => x !== 2);
```

Javascript ES6初探

3. try/catch
4. 箭头函数（Lambda Expression）

```
let func = (x) => {
    x++;
    console.info(x);
}
```

5. 新手常犯的一些错误
    - null, undefined与NaN的区别
    - `===` 与 `==`
    - 弱类型

异步编程模型

1. 何为异步？

```
setTimeout(() => console.info('hello world'), 3000);
```

异步编程模型

2. Promise

   - 链式调用，结构清晰

```
return new Promise((resolve, reject) => {
    promise1()
        .then(value => {
            return promise2(value);
        })
        .then(value => {
            resolve(value);
        })
        .catch(err => {
            reject(err);
        });
})
```

扩展：Promise小书

异步编程模型

3. Async/Await，也许是目前最先进的异步编程模型
    - 化异步为同步
    - 最早由Microsoft在C#中投入使用

```
return async () => {
    try {
        return await promise2(await promise1());
    } catch(err) {
        throw err;
    }
}
```

扩展：Async/Await替代Promise的6个理由

使用NPM

NPM是Nodejs Package Manager的简称，社区十分强大

基本使用

```
# 安装包常用 -g, --save, --save-dev
npm install -g pm2 mocha mongo-express
npm install --save debug
npm install --save-dev chai mocha
# 启动express app
npm start
```

Express和常用的包

`package.json`和`package-lock.json`

```
npm install
```

- express
- joi
- winston
- chai
- debug

打开代码

MongoDB

1. 类似JSON的组织结构
2. 一种NOSQL(Not Only SQL)

教程

Nodejs驱动的文档

扩展

- 一对多、多对多关系设计
  - http://www.cnblogs.com/lgxlsm/archive/2013/05/15/3080994.html
  - http://www.jianshu.com/p/bb0caddff60a
- 条件操作符、排序、分页
- 流式操作
- 索引
  - http://www.imooc.com/article/11725
- 缓存同步、原子性与一致性
  - https://coolshell.cn/articles/17416.html
  - https://www.zhihu.com/question/30272728
- 数据库伸缩与容灾设计
- 分布式数据库，Map/Reduce操作

Mocha自动化测试

```
# 本地测试
DEBUG=TEST NODE_ENV=development mocha tests/contacts_test.js
#  生产环境测试
DEBUG=TEST NODE_ENV=production mocha tests/contacts_test.js
```

文档

PM2进程管理器

```
pm2 start pm2_contacts.json
pm2 stop pm2_contacts.json
pm2 restart pm2_contacts.json
pm2 show 0
pm2 kill
pm2 monit
pm2 flush
```

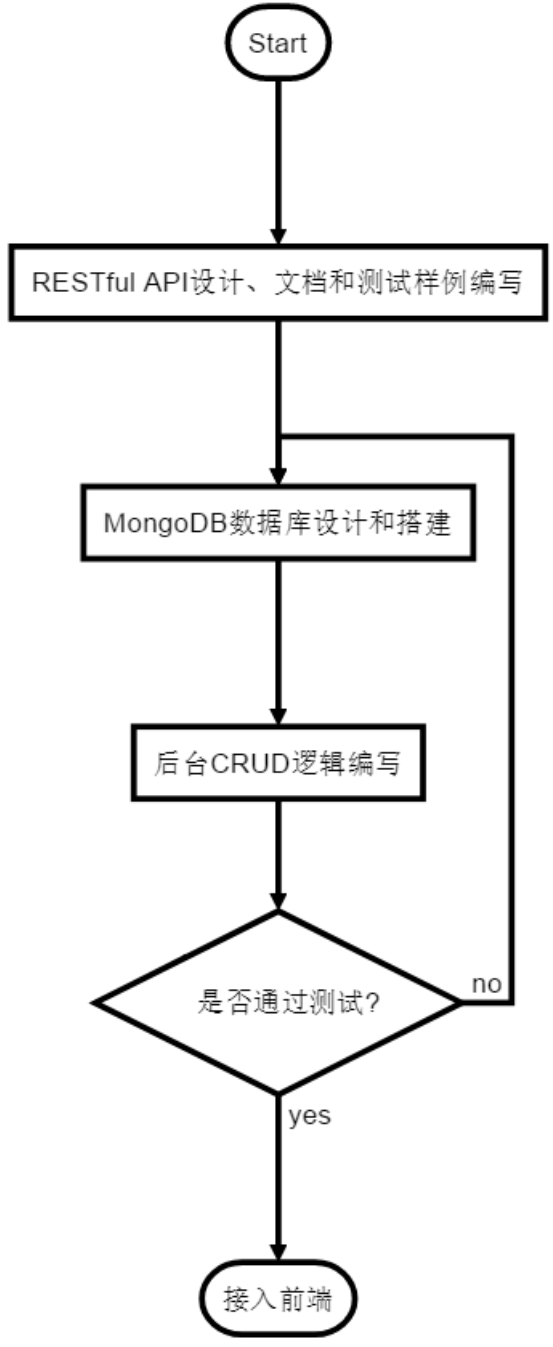
文档

部署至云端

1. 完成本地开发并通过本地测试
2. 同步代码至服务器并启动程序
3. 在服务器上通过本地测试
4. 在服务器的网络安全组中打开对应的端口
5. 在本机执行生产环境测试

作业要求

1. 运用TDD的开发模型，基于nodejs starter code完成通讯录程序的后端开发
2. 欢迎在群里讨论或者向我提问(严禁抄袭)
3. 作业以zip格式打包，命名为`2017冬招新人_姓名_后台作业.zip`，以附件的形式发送邮件至`yong_shan@foxmail.com`，邮件标题为`2017冬招新人_姓名_后台作业`
4. 截止时间：**2018年1月6日晚24:00**


Start
RESTful API设计、文档和测试样例编写
MongoDB数据库设计和搭建
后台CRUD逻辑编写
是否通过测试?
no
yes
接入前端

作业中应包含但不限如下要素：

1. `api_doc.md` 详尽的API文档
2. `tests/` 正确、完整的测试样例，包括本地测试和生产环境测试
3. 完整的代码实现，逻辑正确，可以通过所有测试
4. 将后端部署至服务器，使用pm2管理进程（为了方便前端访问，前端代码应请求服务器的后台接口）
5. 附加题：
    - 查询所有联系人的接口中加入分页机制
        - http://www.cnblogs.com/xiaolai/p/3401289.html
        - http://www.runoob.com/mongodb/mongodb-sort.html
        - http://mongodb.github.io/node-mongodb-native/3.0/reference/main/
    - 性能测试（分析结果放入 `README.md` 中）
        - vegeta
    - 良好的代码风格