基于 NodeJS & MongoDB 的拉丁培训管理系统设计、实现

目录

1. 前言

虽然此文的题目是跟舞蹈相关的，但是内容本身却是纯粹的技术讨论，而拉丁舞只是个“引子”或者“噱头”。

之所以用 Node 开发这个系统一方面固然是简化拉丁舞培训报名， 最重要的还是因为最近学了 Nodejs 和 MongoDB，而学习的最好方式莫过于实践， 而且用 Node 做这个内部需求也不会有业务上的压力，比较适合技术积累。

目前应用已经初具雏形，仅供测试，感兴趣的同学可以通过：http://10.19.70.7:3000/ 访问，这个是系统的测试环境，管理员权限是开放的。如果您发现什么问题可以向我反馈。

2. 概述

本文主要介绍如何利用 NodeJS 以及相关模块和 MongoDB 等来构建一个虽小却“五脏俱全”的完整 Web 应用——拉丁培训管理系统。

NodeJS 基于 Chrome V8 Javascript 引擎，由于其具有事件驱动、非阻塞 I/O 模型、支持异步回调等特性使其非常适合用于构建快速、可扩展、轻量、高效的数据密集型实时应用。

先说说整个系统的实现方案：

- 前端： **FDEV4/jQuery**
- 后台： **NodeJS**
- Web Framework： **Express**
- Html 渲染模板引擎： **Jade/markdown**
- 数据持久化存储： **MongoDB ***
- 数据库驱动： **MongoSkin**(基于 node-mongodb-native)
- Flow Control： **Step**
- Node 应用持续可用性保证： **Forever**
- 静态资源提供/请求压缩、访客日志记录等： **Nginx**
- 部署环境： **Mac / Ubuntu**

3. 从一个基本的应用开始

这部分的内容需要达到的一个目标是能够利用 Node 和 Express Web Framework 搭建一个最简单的 Web 应用，可以：

1. 提供静态资源路由：加载相应的 js、css；
2. 利用 Jade 模板引擎渲染出一个 Html 页面；

3. 连接并读取 MongoDB 数据库，将查询的数据与模板一起合成相应的 Html 页面，在浏览器中显示出来；

4. 用户填写相应的表单数据，并将数据提交给相应的 Node Controller 加工处理后存入数据库；

这些是完成一个 web 应用最基本的操作，我们先要完成以上基本功能而后在此基础上进一步的完善和扩展，最终实现更为复杂的系统功能。

3.1 环境准备

工欲善其事，必先利其器。首先要配置好开发环境，这个过程相对来说还是比较简单的：

1. **NodeJS.** 首先安装 NodeJS：可以在 http://nodejs.org/ 上下载最新的安装文件进行安装。这个过程应该会比较顺利的。在 Ubuntu 下安装步骤：

    1. sudo apt-get remove nodejs #卸载旧版本 Node，可选

        (如果通过 source 安装 node 可以直接在 source 目录执行 sudo make uninstall 就可以了)

    2. wget http://nodejs.org/dist/v0.6.14/node-v0.6.14.tar.gz
    3. tar -xzvf node-v0.6.14.tar.gz; cd node-v0.6.14/
    4. ./configure
    5. make
    6. sudo make install

        node –v #可以通过此命令查看是否安装成功及 Node 的版本号

2. **NPM**(Node Package Manager). NPM 默认会随同 NodeJS 一起安装的。如果没有,可以参考这里的说明进行安装 :http://npmjs.org/doc/README.html 。在 Unix like 系统上通过 curl http://npmjs.org/install.sh | sh 命令应该就可以搞定的。

3. **MongoDB.** 由于需要将数据进行持久化存储所以还需要一个数据库。对于后台工程师来说采用 Oracle 或者 Mysql 可能是很自然的想法，不过我个人觉得采用 MongoDB 跟 Nodejs 搭配是非常完美的，这俩简直天生一对儿。MongoDB 天生对 JS 开发人员友好，而且性能、稳定性、扩展性都很好，是一个非常有潜力的 Nosql 解决方案。具体安装可以参考：http://www.mongodb.org/display/DOCS/Quickstart

    在 Ubuntu 下可以采用如下方式安装：

    1. sudo vi /etc/apt/sources.list

        加上这个软件源:

        deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen

    2. sudo apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10

3. sudo apt-get update

   sudo apt-cache search mongodb

4. sudo apt-get install mongodb-10gen

   安装好后配置文件路径： /etc/mongodb.conf

   mongod –version #可以通过此命令检查是否成功及 mongodb 版本号

4. **Express.** NodeJS 虽然很适合构建快速、可扩展的 web 应用，不过单纯利用 NodeJS 来构建 Web 应用还是有很多事情需要做的，重造轮子的成本很大，而且带来的后续可维护性、性能、安全性等都会大打折扣。所以推荐使用 Express web framework 来帮助搭建应用的基本框架。Express 具体使用指南参考：http://expressjs.com/guide.html

   Express 的安装：$ npm install express –g

   在 express 安装完成后通过简单的命令即可生成一个基本的 web application：

   1. **Create The App：** $ express /github/latinode && cd /github/latinode

   该步骤会在 latinode 目录下生成如下目录结构：

```
├── app.js            // web 应用启动文件
├── package.json      // Json 格式的应用描述信息：应用名、版本、作者、依赖、license 等
├── public            // 静态资源存放目录
│   ├── images        // 静态图片存放目录
│   ├── javascripts   // 前台 JS 存放路径
│   └── stylesheets   // 样式文件存放路径
│       └── style.css
├── routes            // 后台 Nodejs 代码存放目录
│   └── index.js      // index 首页相关代码
└── views             // 前端 Jade 模板存放路径
    ├── index.jade    // 首页模板对应 Jade 文件
    └── layout.jade   // 公用 layout 对应 jade 模板
```

   2. **Install Dependencies:** $ npm install –d #该命令会在 latinode 应用里的 node_modules 里安装 express module。

   3. **Start the Server:** $ node app.js

   安装好依赖，启动应用后可以访问：http://localhost:3000/ 即可看到页面输出：

   Express

   Welcome to Express

看到这个信息说明应用启动成功了。还是很顺利哈。

至此我们已经搭建好了一个完整的最基本的 web 应用：可以加载静态资源、可以渲染出 html 页面。是不是很容易？哈哈。不过终究还是缺了点什么？嗯，一个真正的应用没有数据哪行！？所以接下来才正式切入主题——打通数据库。

是的，要让数据流动起来，从数据库经由 MongoSkin 驱动流向 Node Controller，经过必要的加工处理，放入模板引擎，与 Jade 模板合成对应的 html，发送给客户端，在用户浏览器里展现出来。用户输入必要的信息，经过前端 JS 验证，然后由 express 路由给指定的 Node Controller 逻辑，这里可以进行后端验证、权限鉴别、其他处理等然后存入 MongoDB 数据库。因此我们要安装 Mongoskin：

5. **MongoSkin.** 通过 NodeJS 连接 MongoDB 数据库需要相应的驱动。node-mongodb-native 是最强大的 MongoDB 驱动之一，不过这个驱动用起来不太方便，回调太多。Mongoskin 是基于 node-mongodb-native 的，并在其上进一步封装使其更易于使用。封装后的使用语法类似 mongo shell，而且像 node-mongodb-native 一样强大，另外还支持 javascript 方法绑定。可以像使用 MVC Model 一样使用 Mongoskin 实例。

   MongoSkin 安装方法:

   在 node 应用目录/github/latinode 内执行：npm install mongoskin

   这样会在 node 应用目录 node_modules 里面安装 mongoskin 模块。

   其他相关细节可以参考：https://github.com/guileen/node-mongoskin

接下来看看如何来完成对数据库的操作——

3.2 通过 MongoSkin 操作数据库

3.2.1 创建测试数据

为了测试利用 Node 读取 MongoDB 数据是否成功，我们需要先向数据库中插入测试数据，这一步可以通过 mongo shell 来完成：

1. 安装完 mongoDB 后通过 mongod 命令启动数据库(如果在 ubuntu 上通过前文所述方法安装则会自动启动 mongod，可以通过 ps aux | grep mongo 看是否有相应进程)，Mongod 会默认监听来自 27017 端口的数据库连接请求；
2. 通过 mongo 命令启动 mongo shell，Mongo Shell 是一个完整的 Javascript 解释器，我们可以在其中执行各种 js 脚本语句，同时也可以通过该终端对 mongoDB 数据库进行操作；
3. 在数据库中插入数据：
   a) 默认情况下启动 mongo shell 后会连接 test 数据库，可以切换到 latin 数据库：执行 use latin 命令即可.注意一开始虽然没有 latin db，但是当执行 use latin 命令时会根据情况自动创建 latin db。可以通过 db.getName()命令显示当前数据库名。

b) 执行命令 db.latin.insert({'dancerID': '29411', "dancerName" : "M.J."});

c) 该命令会在数据库中创建一个名字为 latin 的 collection（相当于传统数据库的表 table），然后在该 collection 中新建一个 document（相当于传统数据库中的记录 row）。Document 的内容查询出来后为一个 json 对象。不过存储的是 Bson 格式的 document ,即 json-like Document 序列化后对其进行二进制编码存储。

d) 我们可以通过 db.latin.find();命令来验证之前的数据插入是否成功。

说明：MongoDB 不是此处的重点，先完成最基本的操作即可，后面会再进一步加以说明。

3.2.2 通过 Get 请求读取 MongoDB 数据

要想从数据库中读取数据先要通过 MongoSkin 连接数据库。这个过程相对来说还是挺简单的。

1. 由于之前已经为应用安装了 mongoskin 模块 ,我们可以在 routes 目录里面添加一个文件 database.js, 内容如下：

```
var db = exports.db = require('mongoskin').db('localhost:27017/latin');
// 基本 dancer 操作 DAO 接口
var dancerDAO = exports.dancerDAO = {
  /**
   * 根据 dancerID 查询其基本会员信息
   * @param dancerID          待查询的会员的 dancerID
   * @param fn                fn 为执行查询成功后的回调
   */
  findDancerByID: function(dancerID, fn){
    this.findOne({'dancerID':dancerID}, fn);
  }
}
```

2. Collection 操作绑定

使用 MongoSkin 的一个便利之处就是可以将 Collection 和其对应的操作进行绑定。这样一来 DAO 接口的逻辑就可以复用，不必重复编写相同的数据库读写代码逻辑，维护起来也容易很多。由于数据库操作是后面很多逻辑的基础，所以这个绑定操作也应当在读写数据库之前进行，放在 app.js 里面是比较合适的，可以在 app.js 中加入如下代码：

```
var    db   = require("./routes/database.js").db,
dancerOp    = require("./routes/database.js").dancerDAO;
```

在应用 configure（app.configure()）之后，Routes 初始化代码：

```
app.get('/', routes.index);
```

之前加入 Collection 绑定逻辑：

```
db.bind("latin", dancerOp);
```

进行如上绑定后就可以在其他 routes 逻辑中通过：db.latin.findDancerByID('29411',function(){});来查询数据了。

3. Node index 逻辑中读取数据

将 index.js 中代码改成如下：

```
var db = require("./database.js").db;
exports.index = function(req, res){
  db.latin.findDancerByID('29411', function(err, result) {

    if (err) throw err;
    if (!!result){
      res.render('index',
       {   title:   'Express',
         dancer: result
       });
    }
  });
};
```

通过以上逻辑将会从数据库中查询满足 dancerID 为 29411 的数据，然后将其放入 result 变量里面。

4. 模板数据合成展示：

修改 index 对应的模板文件：index.jade，添加如下内容：

```
h1= title
p Welcome to #{title}
p dancerID: #{dancer.dancerID}
p dancerName: #{dancer.dancerName}
```

Jade 模板引擎会用 title 和 dancer 中的数据替换 index.jade 中的相应变量，合成最终将在浏览器中展示的 html，最后合成出来的 html 代码如下：

```
<body><h1>Express</h1><p>Welcome to Express</p><p>dancerID:
29411</p><p>dancerName: MJ</p></body>
```

这样一来我们就将数据从数据库中取出来进行展示了。生成出来的 html 是去除空格后的压缩代码，阅读起来不太方便，不过倒是相当节约带宽的。注意 Jade 模板语言的具体详情现在不做描述，后面会进一步说明。

现在我们已经可以从数据库中读取数据并显示了，不过这里的逻辑很简单，而且不足的是 dancerID 也是写死的，一种很自然的想法是通过“/user/29411”…“/user/29555”等 URL 来访问对应的会员信息。

这个需要用到 express 的 Routing 功能。稍后再说。

接下来说说如何将用户提交的表单数据保存到数据库。

3.2.3 通过 Post 请求将数据存入 MongoDB

1. 首先需要一个提交数据的表单，可以直接对 index.jade 进行修改如下：

```
h1= title
p Welcome to #{title}
p dancerID: #{dancer.dancerID}
p dancerName: #{dancer.dancerName}
form(name="updateForm", id="updateForm", method="post", action="/update")
  table.apply-table-a
    tbody
      tr.em
        th
          label 工 号：
        td
          input#dancerID.comm-input(type="text", name="dancerID" )
      tr
        th
          label 姓 名：
        td
          input#dancerName.comm-input(type="text", name="dancerName")

  button(class="comm-button", id="update-btn")  提 交
```

注意：空格缩进对于 Jade 模板很重要，缩进代表了 DOM 结构的层级关系。子元素的缩进要深一级。

2. 接下来我们需要修改 database.js，在其中添加一个方法:

```
updateDancerByID: function(dancerID, dancerName, fn){
  this.update({dancerID:dancerID}, {dancerName:dancerName}, {upsert:true}, fn);
}
```

在 mongoDB 语法中有一种特殊的更新叫 upsert，该操作如果没有找到满足条件的文档就会以该条件和更新的文档为基础创建一个新的文档，如果找到了满足条件的文档就会正常更新。如此同一套代码

既可以创建又可以更新文档。Update 的第三个参数为 true 表示这是一个 upsert 操作。注意在 upsert 参数传递上 mongoskin 与 mongoshell 有些差别。

3. 添加用户提交表单数据处理逻辑：

由于在前面 index.jade 里面我们的表单提交到的 action 是 update， 所以我们可以在 routes/index.js 中添加该数据保存逻辑，新增代码如下：

```
exports.update = function(req, res){
  var dancerID    = req.body.dancerID,
    dancerName    = req.body.dancerName;   // 表单提交的数据可以通过 req.body 取得
  // 简单起见，此处忽略表单数据校验逻辑
  db.latin.updateDancerByID(dancerID, dancerName, function(err) {
    if (err) throw err;
    console.log('Dancer Updated With ID:',dancerID,' DancerName:', dancerName);
    // 表单提交成功后返回首页
    res.redirect('back');
  });
};
```

4. 在 app.js 里配置 post 请求的路由信息，修改 app.js 在 get 请求下添加新的路由如下：

```
// Routes
app.get('/', routes.index);
 app.post('/update', routes.update);
```

5. 重启应用，现在再来试试提交表单，数据应该可以成功保存了。可以通过 mongo shell 执行查询：

```
db.latin.find({dancerID:'29411'});
```

来查看表单提交操作执行后数据的变化情况。

从 updateDancerByID 方法我们不难发现目前只能修改用户 dancerName 信息，而 dancerID 是不能修改的，除非该会员不存在，则执行新增操作。

到此为止我们已经完成了 mongodb 数据的读取和保存。主干流程算是通了。不过一个实际的应用要远比这个复杂的多 好在基本原理上是相同的。而且迄今为止我们所掌握的 MongoDB、MongoSkin、Jade、express 技能等都是九牛一毛。NodeJS 的水还是很深的，而且涉及到的周边模块也很多，为了完成一个更为复杂的“过得去”的应用我们还需要了解更多……。

4. 拉丁培训管理系统需求概述

这个培训管理系统的功能远比上面的例子要复杂，细节上要考虑的也相当多。不过使用 NodeJS 开发的好处是前端、后端都用 JS，而且接口可以自己定义，沟通成本很小，同一个功能实现的方式可能很多，全在自己发挥。

先说说这个系统的基本功能：

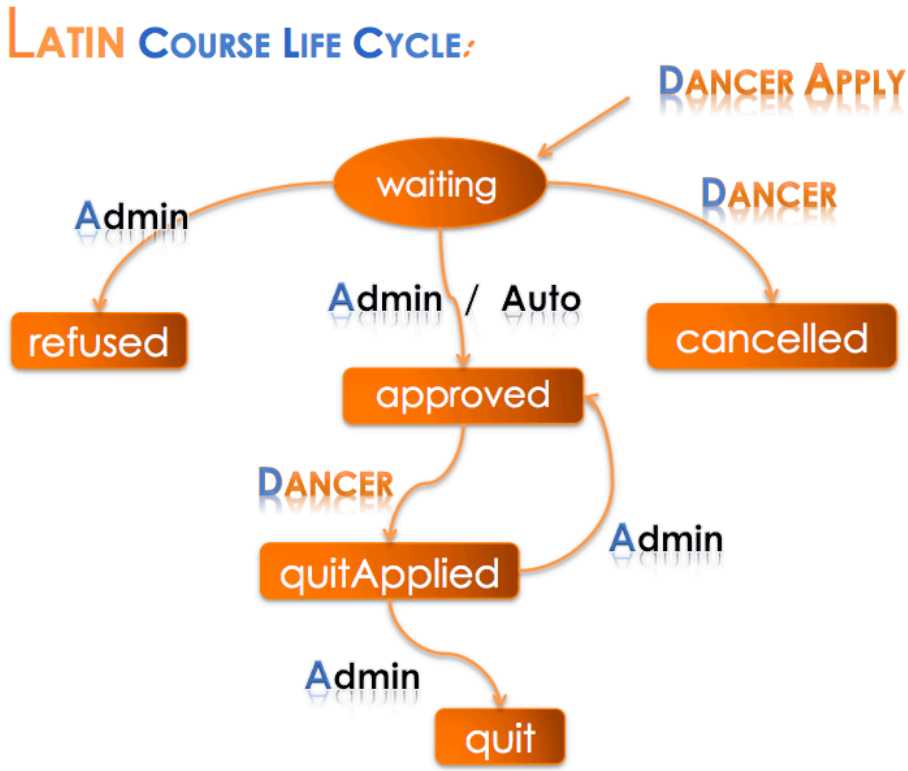
目前已经实现的功能大体上分两部分——用户前台和管理员后台。

4.1 用户前台功能描述

1. 实现**网上报名。**
   a) 新用户：填写基本信息（姓名、工号、旺旺、支付宝账号、分机、公司邮箱等）；选择培训课程；提交报名；系统存入数据库。
   b) 老用户：根据用户填写的工号**自动获取显示其个人信息**；然后选择相应课程提交报名；也可以修改个人信息。
2. 用户可以查看个人信息及报名情况等。
3. 报名页显示**当前开设课程实时信息**（舞种、额定人数，已申请人数、报名成功人数等）。
4. 新开课程配置简化 :开设新培训课程时代码逻辑不需要变更 ,只需要更改配置即可迅速满足要求。
5. 申请报名后需管理员手工审核报名是否通过，也可以根据一定规则由系统进行自动审核。
6. 管理员审核通过前用户可取消课程报名，审核通过后用户可申请退课。
7. 退课需管理员审核：可以拒绝退课；或者线下退费、退课。
8. 用户可以**查询报名情况：**根据工号、部门、性别、课程、报名状态、缴费状态等进行组合查询。
9. 查询结果分页展示，点击表格头部时可以对某些字段进行排序。

4.2 管理员后台功能描述

- **课程生命周期管理：**



1. 新申请报名课程状态为待审核----------(waiting)；
2. 管理员审核前用户可以自助取消----------(cancelled)；
3. 管理员可以审核拒绝----------(refused)；
4. 或者审核通过则报名成功----------(approved)；
5. 报名成功后会员可以申请退课----------(quitApplied)；
6. 退课需要管理员审核，审核拒绝则回到报名成功----------(approved)；
7. 退课审核通过则处于退课成功----------(quit)
8. 对于课程生命周期状态转换系统会验证其是否满足前置条件 :比如 waiting 状态不能直接转为 quit，但是可以转换为 cancelled，只有 waiting 或者 quiteApplied 状态的可以转换为 approved，而 approved 不能转为 waiting ,只有 quitApplied 的课程可以转为 quit，并且需要先退费等。

- **课程缴费状态设置：**

为了简化逻辑，课程的缴费状态跟其生命周期是分开的：

i. 新报名的会员其缴费状态为未缴费。

ii. 报名成功的会员在缴费后管理员可以将其设为已缴费。

iii. 申请退课的会员管理员可以线下退费并修改其状态为未缴费，然后退课。

- **管理员修改会员信息：**

  会员可以修改自身基本信息 ,管理员可以修改其基本及高级属性比如 :level、vip 等级、blacklist 状态、lock 状态等。

5. 系统的具体设计实现

5.1 Document 模型设计

首先说说会员模型设计。如果用传统的 Mysql 数据库来实现系统所需功能，由于会员和课程之间是多对多的关系，所以除了需要建会员表、课程表之外还需要增加一个培训表，把会员和课程关联起来，如此以来系统就相对复杂化了，而且为了保证操作的原子性可能还需要引入事务。

采用 MongoDB 后就完全不同了，由于 MongoDB 是 schema free 的 Nosql 数据库，在 MongoDB 中集合(collection)相当于传统数据库的表(table)，文档(document)相当于传统数据库的记录(row)，而且文档里面可以嵌套文档。这样以来可以在会员文档里面嵌入其参加的舞蹈课程信息文档，如此一个文档就能很自然地表达出会员和课程的关系了。具体文档模型如下：

dancer:

```
{
    "_id" : ObjectId("4f5c6c80359dda5b98000034"),     // mongodb 自动生成的
    "dancerID" : "29411",                    // 工号，唯一
    "dancerName" : "MJ",                     // 姓名
    "department" : "tech",                   // 部门
    "email" : "latin@alibaba-inc.com",       // 公司邮箱
    "extNumber" : "76211",                   // 分机号码
    "gender" : "male",                       // 性别
    "wangWang" : "hustcer",                  // 旺旺号码
    "alipayID" : "hustcer@gmail.com",        // 支付宝账号
    level:5,                                 // 舞蹈水平 9 is the highest level.
    vip:5,                                   // 根据上课积极程度以及参加演出次数确定,最高为 5
    performance:[],                          // 参加的演出信息
    ps:"",                                   // 个人备注、个性签名之类
    blacklist:true,                          // 加入黑名单的会员不能报名参加培训
    locked:true,                             // 个人信息锁定后不能修改
    dancerDNA: 110101,                       // 以后可以考虑利用二进制标志位来定义用户是否具备某些属性
    password:password,                       // 用户密码，暂不支持

    "courses" : [
        {
            "courseVal" : "13RI",        // 前面的数字代表期次，接下来的字母表示舞种，尾字母表示课程等级
            "gmtPayChanged" :       ISODate("2012-03-11T21:44:54.017Z"), // 缴费状态改变时间
```

```
            "gmtStatusChanged" :    ISODate("2012-03-17T06:23:17.858Z"), //  课程状态改变时间
            "applyTime" :           ISODate("2012-03-17T06:23:17.858Z"), //  课程申请时间
            "paid" :  100.00,       //  如果已缴费则为缴费金额
            "status" : "approved"   // status: waiting, cancelled, approved,refused,quitApplied, quit
        },
        {
            "courseVal" : "13CE",
            "gmtPayChanged" :       ISODate("2012-03-17T06:23:53.124Z"),
            "gmtStatusChanged" :    ISODate("2012-03-18T08:00:29.880Z"),
            "applyTime" :           ISODate("2012-03-17T06:23:17.858Z"), //  课程申请时间
            "paid" :        0.00,
            "status" :      "approved"
        }
    ],
    "gmtCreated" : ISODate("2012-03-11T09:12:32.878Z"),
    "gmtModified" : ISODate("2012-03-18T08:00:29.880Z")
}
```

需要说明的是：

1. courses 是一个数组，而数组里面的每一个元素又是一个内嵌文档，表示会员参与的舞蹈培训信息。
2. courseVal 的命名遵循特定规则 :比如"13RI"其中 13 代表培训的期次 ;R 代表舞种为伦巴(Rumba)，其他还有 C(恰恰，ChaCha)、S(桑巴、Samba)、J(牛仔，Jive)、P(斗牛，Paso doble)；I 代表课程等级为中级(intermediate), E 代表基础班：Elementary，A 代表高级班 Advanced；
3. 会员文档会有创建和修改时间属性，课程会有缴费状态变更时间、课程状态变化时间和报名申请时间属性，以跟进信息变更。

在建立以上文档模型后，由于用户和管理员的所有操作最终都要反映到数据库的数据变化上，所以我们还需要掌握基本的 MongoDB 数据库操作技能。

这部分内容比较多 ,还没来得及整理。后面其他内容抽空补上 ,感兴趣的同学可以保持关注。

Thanks For Reading !