**基于内容的新闻推荐系统**

**详细设计文档**

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布  [ ] 正在修改 | 文件标识： |  |
| 当前版本： | 1.0 |
| 作 者： | wangjj |
| 日 期： | 2020-4-13 |

目 录

[1 环境搭建 4](#_Toc42451328)

[1.1 新建工程 4](#_Toc42451329)

[1.2 引入spring+springmvc 5](#_Toc42451330)

[1.3 引入mysql+mybatis 6](#_Toc42451331)

[2 主要技术实现 11](#_Toc42451332)

[2.1 算法实现 11](#_Toc42451333)

[2.2 计算TF 13](#_Toc42451334)

[3 数据库 16](#_Toc42451335)

[3.1 用户表(users) 16](#_Toc42451336)

[3.2 用户token表(users\_token) 17](#_Toc42451337)

[3.3 新闻表(news) 17](#_Toc42451338)

[3.4 新闻类别表(news\_type) 18](#_Toc42451339)

[4 主要功能 19](#_Toc42451340)

[4.1 用户管理 19](#_Toc42451341)

[4.1.1 注册、登录(管理员、普通用户) 19](#_Toc42451342)

[4.1.2 修改用户信息(管理员、普通用户) 19](#_Toc42451343)

[4.1.3 删除用户信息(管理员) 20](#_Toc42451344)

[4.1.4 获取用户信息(管理员、普通用户) 20](#_Toc42451345)

[4.1.5 管理员获取普通用户信息(管理员) 20](#_Toc42451346)

[4.1.6 获取用户信息列表(管理员) 20](#_Toc42451347)

[4.2 新闻管理 21](#_Toc42451348)

[4.2.1 添加新闻类别(管理员) 21](#_Toc42451349)

[4.2.2 获取新闻类别列表 21](#_Toc42451350)

[4.2.3 添加新闻(管理员) 22](#_Toc42451351)

[4.2.4 删除新闻(管理员) 22](#_Toc42451352)

[4.2.5 获取新闻列表(普通用户/管理员) 23](#_Toc42451353)

[4.2.6 获取新闻内容(普通用户/管理员) 23](#_Toc42451354)

[4.2.7 查看历史记录(普通用户/管理员) 24](#_Toc42451355)

[4.2.8 搜索新闻(普通用户/管理员) 24](#_Toc42451356)

[4.2.9 推荐新闻(普通用户/管理员) 24](#_Toc42451357)

[5 版本历史 25](#_Toc42451358)

# 1 环境搭建

开发框架：SSM(spring+springmvc+mybatis)

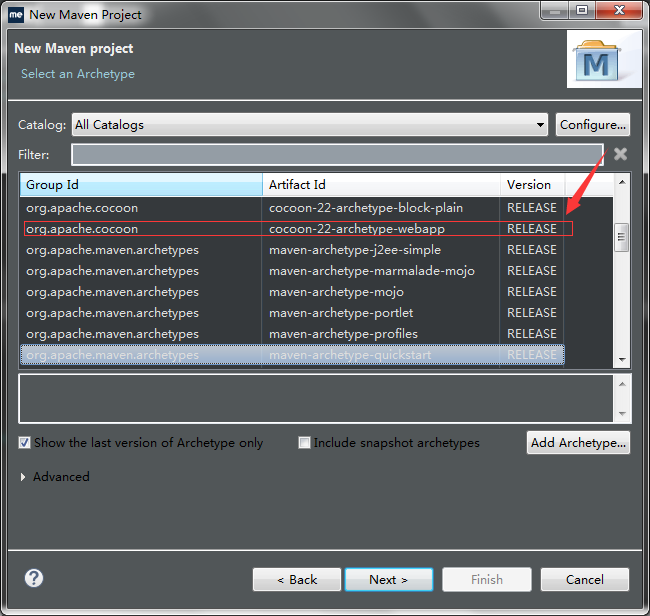
开发语音：Java、HTML5、JavaScript

开发工具：MyEclipse

软件依赖：tomcat8、mysql

## 1.1 新建工程

打开myeclispe，新建一个maven工程，选择weapp选项。



## 1.2 引入spring+springmvc

在pom.xml文件中引入如下依赖：

|  |
| --- |
| <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>4.3.18.RELEASE</version>  </dependency> |

然后在web.xm文件中加入以下配置：

|  |
| --- |
| <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext.xml</param-value>  </context-param>    <servlet>  <servlet-name>springmvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:springmvc-servlet.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>springmvc</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping> |

创建applicationContext.xml和springmvc-servlet.xml文件，在springmvc-servlet.xml中加入以下配置：

|  |
| --- |
| <!-- spring扫描的包 -->  <context:component-scan base-package="com.wjj"/>  <!-- DispatcherServlet不处理静态资源,交给服务器默认的servlet处理 -->  <mvc:default-servlet-handler />  <!-- 启用annotation -->  <mvc:annotation-driven /> |

到此，spring+springmvc配置基本完成。

## 1.3 引入mysql+mybatis

pom.xm依赖引入：

|  |
| --- |
| <!-- mysql 的驱动包-->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.12</version>  </dependency>    <!--logback 日志包-->  <dependency>  <groupId>ch.qos.logback</groupId>  <artifactId>logback-classic</artifactId>  <version>1.2.3</version>  </dependency>    <!-- mybatis jar 包-->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>3.4.5</version>  </dependency>    <!-- 数据库连接池 alibaba 的 druid -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid</artifactId>  <version>1.1.7</version>  </dependency>    <!-- mybatis与spring整合的jar包-->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis-spring</artifactId>  <version>1.3.1</version>  </dependency>    <!--spring管理的 jdbc ,以及事务相关的-->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>4.3.17.RELEASE</version>  </dependency> |

新建一个db.properties文件，内容如下：

|  |
| --- |
| db.driver=com.mysql.cj.jdbc.Driver  db.url=jdbc:mysql://localhost:3306/mybatis?serverTimezone=UTC  db.username=root  db.password=123456  max=20  min=10 |

再创建一个userMapper.xml文件，内容大致如下：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <mapper namespace="com.wjj.dao.UserDao">    <insert id="addUser" parameterType="com.wjj.entity.User">  insert into users(name,age) values(#{name}, #{age})  </insert>    <delete id="deleteUser" parameterType="int">  delete from users where id=#{id}  </delete>    <update id="updateUser" parameterType="com.wjj.entity.User">  update users set name=#{name},age=#{age} where id=#{id}  </update>    <select id="getUser" parameterType="int" resultType="com.wjj.entity.User">  select \* from users where id = #{id}  </select>    <select id="getAllUsers" resultType="com.wjj.entity.User">  select \* from users  </select>  </mapper> |

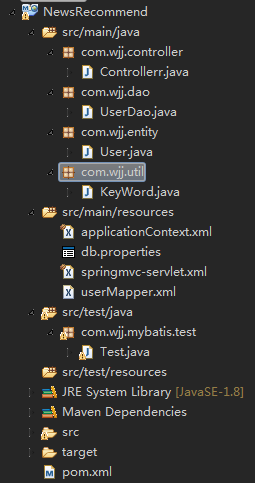
在com.wjj.dao包下创建一个UserDao接口：

|  |
| --- |
| package com.wjj.dao;  import java.util.List;  import com.wjj.entity.User;  public interface UserDao {  void addUser(User user);  void deleteUser(int id);  void updateUser(User user);  User getUser(int id);  List<User> getAllUsers();  } |

在applicationContext.xml文件中加入如下内容：

|  |
| --- |
| <context:property-placeholder location="classpath:db.properties"/>    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">  <property name="driverClassName" value="${db.driver}"/>  <property name="url" value="${db.url}"/>  <property name="username" value="${db.username}"/>  <property name="password" value="${db.password}"/>    <property name="maxActive" value="${max}"/>  <property name="minIdle" value="${min}"/>  </bean>    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  <property name="dataSource" ref="dataSource"/>  <property name="mapperLocations" value="classpath\*:\*Mapper.xml"/>  </bean>    <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">  <constructor-arg index="0" ref="sqlSessionFactory" />  </bean> |

整体结构如下：



操作mybatis示例：

|  |
| --- |
| ClassPathXmlApplicationContext context = new  ClassPathXmlApplicationContext("classpath:applicationContext.xml");  /\* 得到 SqlSession 对象\*/  SqlSession sqlSession = (SqlSession) context.getBean("sqlSession");    String statement = null;    //添加  statement = "com.wjj.dao.UserDao.addUser";  int insert = sqlSession.insert(statement, new User(-1, "yinfei", 28));  System.out.println(insert);    //删除  statement = "com.wjj.dao.UserDao.deleteUser";  int delete = sqlSession.delete(statement, 9);  System.out.println(delete);    //查询  statement = "com.wjj.dao.UserDao.getUser";  User user = sqlSession.selectOne(statement,1);  System.out.println(user);    //修改  statement = "com.wjj.dao.UserDao.updateUser";  int update = sqlSession.update(statement, new User(10, "yinfei", 30));  System.out.println(update);    //查询所有  statement = "com.wjj.dao.UserDao.getAllUsers";  List<User> list = sqlSession.selectList(statement);  System.out.println(list); |

到此，mybatis配置基本完成。

# 2 主要技术实现

## 2.1 算法实现

TF-IDF算法

其实这个是两个词的组合，可以拆分为TF和IDF。

TF（Term Frequency，缩写为TF）也就是词频啦，即一个词在文中出现的次数，统计出来就是词频TF，显而易见，一个词在文章中出现很多次，那么这个词肯定有着很大的作用，但是我们自己实践的话，肯定会看到你统计出来的TF 大都是一些这样的词：‘的’，‘是’这样的词，这样的词显然对我们的分析和统计没有什么帮助，反而有的时候会干扰我们的统计，当然我们需要把这些没有用的词给去掉，现在有很多可以去除这些词的方法，比如使用一些停用词的语料库等。

假设我们把它们都过滤掉了，只考虑剩下的有实际意义的词。这样又会遇到了另一个问题，我们可能发现"中国"、"蜜蜂"、"养殖"这三个词的出现次数一样多。这是不是意味着，作为关键词，它们的重要性是一样的？

显然不是这样。因为"中国"是很常见的词，相对而言，"蜜蜂"和"养殖"不那么常见。如果这三个词在一篇文章的出现次数一样多，有理由认为，"蜜蜂"和"养殖"的重要程度要大于"中国"，也就是说，在关键词排序上面，"蜜蜂"和"养殖"应该排在"中国"的前面。

所以，我们需要一个重要性调整系数，衡量一个词是不是常见词。如果某个词比较少见，但是它在这篇文章中多次出现，那么它很可能就反映了这篇文章的特性，正是我们所需要的关键词。

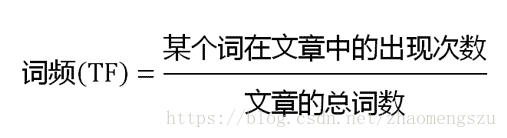
用统计学语言表达，就是在词频的基础上，要对每个词分配一个"重要性"权重。最常见的词（"的"、"是"、"在"）给予最小的权重，较常见的词（"中国"）给予较小的权重，较少见的词（"蜜蜂"、"养殖"）给予较大的权重。这个权重叫做"逆文档频率"（Inverse Document Frequency，缩写为IDF），它的大小与一个词的常见程度成反比。

知道了"词频"（TF）和"逆文档频率"（IDF）以后，将这两个值相乘，就得到了一个词的TF-IDF值。某个词对文章的重要性越高，它的TF-IDF值就越大。所以，排在最前面的几个词，就是这篇文章的关键词。

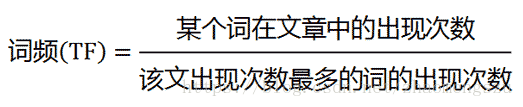
1.计算词频TF



考虑到文章有长短之分，为了便于不同文章的比较，进行"词频"标准化。

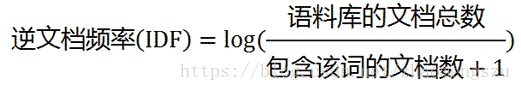


或者



2.计算逆文档频率IDF

需要一个语料库（corpus），用来模拟语言的使用环境。



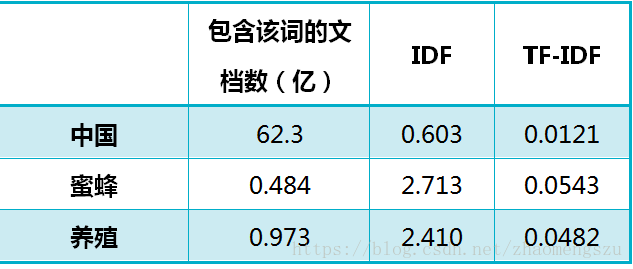
如果一个词越常见，那么分母就越大，逆文档频率就越小越接近0。分母之所以要加1，是为了避免分母为0（即所有文档都不包含该词）。log表示对得到的值取对数。

3.计算TF-IDF



可以看到，TF-IDF与一个词在文档中的出现次数成正比，与该词在整个语言中的出现次数成反比。所以，自动提取关键词的算法就很清楚了，就是计算出文档的每个词的TF-IDF值，然后按降序排列，取排在最前面的几个词。

还是以《中国的蜜蜂养殖》为例，假定该文长度为1000个词，"中国"、"蜜蜂"、"养殖"各出现20次，则这三个词的"词频"（TF）都为0.02。然后，搜索Google发现，包含"的"字的网页共有250亿张，假定这就是中文网页总数。包含"中国"的网页共有62.3亿张，包含"蜜蜂"的网页为0.484亿张，包含"养殖"的网页为0.973亿张。则它们的逆文档频率（IDF）和TF-IDF如下：



从上表可见，"蜜蜂"的TF-IDF值最高，"养殖"其次，"中国"最低。（如果还计算"的"字的TF-IDF，那将是一个极其接近0的值。）所以，如果只选择一个词，"蜜蜂"就是这篇文章的关键词。

除了自动提取关键词，TF-IDF算法还可以用于许多别的地方。比如，信息检索时，对于每个文档，都可以分别计算一组搜索词（"中国"、"蜜蜂"、"养殖"）的TF-IDF，将它们相加，就可以得到整个文档的TF-IDF。这个值最高的文档就是与搜索词最相关的文档。

TF-IDF算法的优点是简单快速，结果比较符合实际情况。缺点是，单纯以"词频"衡量一个词的重要性，不够全面，有时重要的词可能出现次数并不多。而且，这种算法无法体现词的位置信息，出现位置靠前的词与出现位置靠后的词，都被视为重要性相同，这是不正确的。（一种解决方法是，对全文的第一段和每一段的第一句话，给予较大的权重。）

## 2.2 计算TF

这里需要用到分词工具jieba。

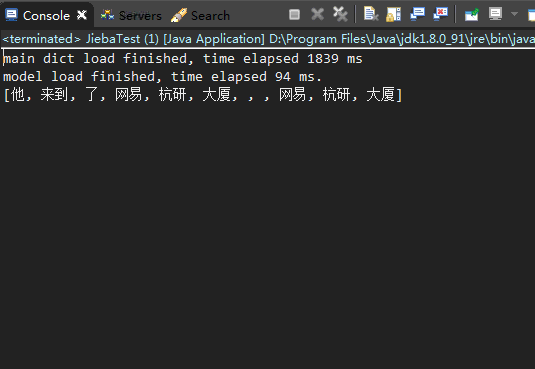
引入依赖：

|  |
| --- |
| <dependency>  <groupId>com.huaban</groupId>  <artifactId>jieba-analysis</artifactId>  <version>1.0.2</version>  </dependency> |

使用示例：

|  |
| --- |
| import com.huaban.analysis.jieba.JiebaSegmenter;  JiebaSegmenter segmenter = new JiebaSegmenter();  String sentences = "他来到了网易杭研大厦, 网易杭研大厦";  List<String> wordList = segmenter.sentenceProcess(sentences);  System.out.println(wordList); |

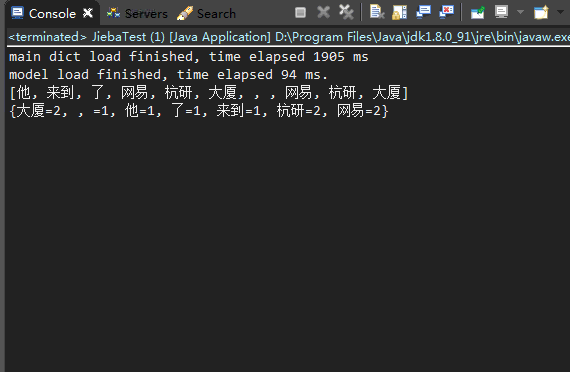
分词结果如下：



然后统计每个单词的出现次数：

|  |
| --- |
| Map<String,Integer> amountWord=new HashMap<String,Integer>();  for (String string : wordList) {  if(!amountWord.containsKey(string)){  amountWord.put(string,1);  }else{  amountWord.put(string, amountWord.get(string).intValue()+1);  }  }  System.out.println(amountWord); |

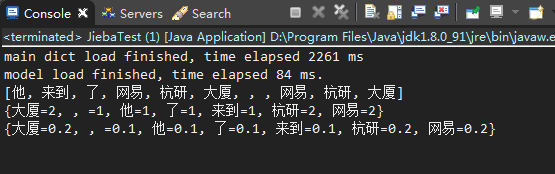
统计结果如下：



接下来需要计算词频TF：

|  |
| --- |
| //计算词频TF  Map<String, Double> tfWord=new HashMap<String,Double>();  for (Entry<String, Integer> string : amountWord.entrySet()) {  tfWord.put(string.getKey(), Double.valueOf(string.getValue())/wordList.size());  }  System.out.println(tfWord); |

统计结果如下：



# 3 数据库

数据库采用mysql。

数据库名：

news\_recommend

创建数据库：

|  |
| --- |
| CREATE DATABASE news\_recommend; |

## 3.1 用户表(users)

|  |  |  |
| --- | --- | --- |
| **字段** | **默认类型** | **备注** |
| id | int | 用户id，唯一，单调递增，**主键** |
| username | VARCHAR(30) | 用户名，注册时候填写的，登录的时候使用，**唯一索引** |
| password | VARCHAR(30) | 密码 |
| phonenum | VARCHAR(30) | 手机号 |
| email | VARCHAR(30) | 邮箱 |
| level | int | 用户等级。**索引。**  1：管理员  2：普通用户 |
| history\_news | TEXT | 历史浏览记录, 新闻id列表，用","分割，例如"2,3,11,15" |
| search\_words | TEXT | 搜索记录，字符串列表，用","分割，例如"信息,天天," |

创建表：

|  |
| --- |
| USE news\_recommend;  CREATE TABLE users (id INT PRIMARY KEY AUTO\_INCREMENT, username  VARCHAR(30), password VARCHAR(30), phonenum VARCHAR(30), email  VARCHAR(30), level INT, history\_news TEXT, search\_words TEXT);  ALTER TABLE users ADD UNIQUE (username); //创建唯一索引  ALTER TABLE users ADD INDEX level\_index (level); //创建普通索引  DESC users; //查看表的情况 |

## 3.2 用户token表(users\_token)

|  |  |  |
| --- | --- | --- |
| **字段** | **默认类型** | **备注** |
| id | int | 等同于上表中的用户id，**主键** |
| token | VARCHAR(30) | 管理员token，调用API时需要带上该参数，15分钟后过期，**唯一索引** |
| create\_time | DATETIME | 创建时间，用于使该条记录过期，**索引** |
| level | int | 用户等级。  1：管理员  2：普通用户 |

注：1、每次用户重新登录，需要修改token值。

2、用户使用token调用API时，token过期时间需要重新刷新成15分钟。

a、打开event\_scheduler：my.cnf中[mysqld]添加event\_scheduler=on #重启服务

b、创建事件：create event myevent on SCHEDULE every 5 second do delete from news\_recommend.users\_token where create\_time <(CURRENT\_TIMESTAMP() + INTERVAL -15 MINUTE);#删除15分钟前的数据

c、开启事件：alter event myevent on completion preserve enable;

d、关闭事件：alter event myevent on completion preserve disable;

e、查看事件：show events;

f、删除事件：drop event if exists myevent;

创建表：

|  |
| --- |
| CREATE TABLE users\_token (id INT PRIMARY KEY , token VARCHAR(30), create\_time DATETIME, level INT);  ALTER TABLE users\_token ADD UNIQUE (token); //创建唯一索引  ALTER TABLE users\_token ADD INDEX create\_time\_index (create\_time); //创建普通索引  DESC users\_token; //查看表的情况 |

## 3.3 新闻表(news)

|  |  |  |
| --- | --- | --- |
| **字段** | **默认类型** | **备注** |
| id | int | 新闻id，唯一，单调递增，**主键** |
| title | VARCHAR(30) | 新闻标题 |
| type | VARCHAR(30) | 新闻类别，例如：政治新闻、经济新闻、法律新闻、军事新闻、科技新闻、文教新闻、体育版新闻、社会新闻等 |
| label | VARCHAR(30) | 新闻标签，新增新闻时用户手动输入 |
| keyword | VARCHAR(30) | 关键字，系统自动从新闻内容中提取 |
| content | TEXT | 新闻内容,长文本格式，0~65535长度 |
| src | VARCHAR(100) | 新闻来源，一个超链接，连接到源地址，例如：https://new.qq.com/omn/20200415/20200415A05J6M00.html |

创建表：

|  |
| --- |
| USE news\_recommend;  CREATE TABLE news (id INT PRIMARY KEY AUTO\_INCREMENT, title VARCHAR(30), type VARCHAR(30), label VARCHAR(30), keyword VARCHAR(30), content TEXT, src VARCHAR(100));  ALTER TABLE news ADD INDEX type\_index (type); //创建普通索引  ALTER TABLE news ADD INDEX label\_index (label); //创建普通索引  ALTER TABLE news ADD INDEX keyword\_index (keyword); //创建普通索引  DESC news; //查看表的情况 |

## 3.4 新闻类别表(news\_type)

|  |  |  |
| --- | --- | --- |
| **字段** | **默认类型** | **备注** |
| id | int | 新闻类别id，唯一，单调递增，**主键** |
| type | VARCHAR(30) | 新闻类别，例如：政治新闻、经济新闻、法律新闻、军事新闻、科技新闻、文教新闻、体育版新闻、社会新闻等 |

创建表：

|  |
| --- |
| USE news\_recommend;  CREATE TABLE news\_type(id INT PRIMARY KEY AUTO\_INCREMENT,type VARCHAR(30));  DESC news\_type; //查看表的情况 |

# 4 主要功能

## 4.1 用户管理

### 4.1.1 注册、登录(管理员、普通用户)

注册：

<http://localhost:8080/NewsRecommend/userRegist?username=ee&password=bb&level=1>

参数：

username：用户名

password：密码

level：用户权限，1、管理员(只有一个)，2、普通用户

登录：

<http://localhost:8080/NewsRecommend/userLogin?username=ff&password=cc&level=1>

参数： 同上

返回值：

{ "retcode" : "200" , "token" : "bqP6PQHP4T" }

异常码：

|  |  |
| --- | --- |
| 错误码 | 说明 |
| 301 | 用户已存在（user is already existed） |
| 302 | 管理员已存在（manager is already existed） |
| 303 | 用户不存在（user is not existed） |
| 304 | 用户权限错误(user permission error) |
| 305 | 参数格式错误(invalid parm) |
| 306 | 密码错误(password incorrect) |

### 4.1.2 修改用户信息(管理员、普通用户)

<http://localhost:8080/NewsRecommend/userModify?token=ee&level1=1&username=a&level2=2&password=aa&phonenum=13015929018&email=123@qq.com>

参数：

token： 修改人的token

level1： 修改人的权限：1、管理员， 2、普通用户

username： 被修改人的用户名

level2： 被修改人的权限：1、管理员， 2、普通用户

password： 修改后的密码

phonenum： 修改后的手机号

email： 修改后的邮箱

异常码：

|  |  |
| --- | --- |
| 错误码 | 说明 |
| 307 | 无效的token(invalid token) |
|  |  |

### 4.1.3 删除用户信息(管理员)

<http://localhost:8080/NewsRecommend/userDelete?token=2BQg2SDUA&username=aa>

参数：

token： 管理员token

username： 普通用户用户名

### 4.1.4 获取用户信息(管理员、普通用户)

<http://localhost:8080/NewsRecommend/userGetInfo?token=2BQg2SDUA>

参数：

token： 用户token

返回值：

{"retcode":"200","password":"456","phonenum":"13015929018","email":"493589280@qq.com","username":"infi"}

### 4.1.5 管理员获取普通用户信息(管理员)

<http://localhost:8080/NewsRecommend/userGetInfoByAdmin?token=2BQg2SDUA&username=abc>

参数：

token： 管理员token

username： 用户名

返回值：

{"retcode":"200","password":"456","phonenum":"13015929018","email":"493589280@qq.com","username":"infi"}

### 4.1.6 获取用户信息列表(管理员)

<http://localhost:8080/NewsRecommend/userGetInfoList?token=2BQg2SDUA>

参数：

token： 管理员token

返回值：

{"userList":

[{"email":"493589280@qq.com","history\_news":"2,3,4,5,6,9,10,","id":1,"level":2,"password":"456","phonenum":"13015929018","search\_words":"天天向上,哈哈,","username":"infi"},

{"id":7,"level":2,"password":"123456","username":"zhouyinfei"},

{"id":9,"level":2,"password":"123","username":"wc"},

{"id":10,"level":2,"password":"111","username":"admin"}],

"retcode":"200"}

## 4.2 新闻管理

### 4.2.1 添加新闻类别(管理员)

<http://localhost:8080/NewsRecommend/newsTypeAdd?token=aaa&type=历史新闻>

参数：

token： 管理员token

type： 新闻类别

异常码：

|  |  |
| --- | --- |
| 错误码 | 说明 |
| 401 | 新闻类别已存在(news type is already existed) |
|  |  |

### 4.2.2 获取新闻类别列表

<http://localhost:8080/NewsRecommend/newsTypeGetList?token=aaa>

参数：

token： 管理员token

异常码：

|  |  |
| --- | --- |
| 错误码 | 说明 |
|  |  |
|  |  |

返回值：

{"newsTypeList":[{"id":2,"title":"八卦新闻"},{"id":3,"title":"法律新闻"}],"retcode":"200"}

newsTypeList： 新闻类型列表，包括id和名称

retcode： 错误码，200时表示正常

### 4.2.3 添加新闻(管理员)

http://localhost:8080/NewsRecommend/newsAdd?token=aaa&title=aa&type=历史新闻&label=aa&content=aa&src=aa

参数：

token： 管理员token

title： 新闻标题

type： 新闻类别

label： 新闻标签

content： 新闻内容

src： 新闻来源

异常码：

|  |  |
| --- | --- |
| 错误码 | 说明 |
| 402 | 新闻类别不存在(news type not exist) |
|  |  |

### 4.2.4 删除新闻(管理员)

<http://localhost:8080/NewsRecommend/newsDelete?token=aaa&id=3>

参数：

token： 管理员token

title： 新闻标题

type： 新闻类别

label： 新闻标签

content： 新闻内容

src： 新闻来源

异常码：

|  |  |
| --- | --- |
| 错误码 | 说明 |
| 403 | 新闻不存在(news not exist) |
|  |  |

### 4.2.5 获取新闻列表(普通用户/管理员)

<http://localhost:8080/NewsRecommend/newsGetList?token=aaa&type=aa>

参数：

token： 用户token

type： 新闻类别

返回值：

{"newsList":[{"id":2,"title":"aa"},{"id":3,"title":"aa"},{"id":4,"title":"aa"}],"retcode":"200"}

newsList： 新闻列表，包括id和标题

retcode： 错误码，200时表示正常

异常码：

|  |  |
| --- | --- |
| 错误码 | 说明 |
|  |  |
|  |  |

### 4.2.6 获取新闻内容(普通用户/管理员)

<http://localhost:8080/NewsRecommend/newsGet?token=aaa&id=aa>

参数：

token： 用户token

id： 新闻id

返回值：

{"news":{"content":"abcd","id":2,"keyword":"bb","label":"bb","src":"http://xxx","title":"aa","type":"aa"},"retcode":"200"}

异常码：

|  |  |
| --- | --- |
| 错误码 | 说明 |
|  |  |
|  |  |

### 4.2.7 查看历史记录(普通用户/管理员)

<http://localhost:8080/NewsRecommend/newsGetHistoryList?token=aaa>

参数：

token： 用户token

返回值

{"historyList":[{"id":2,"title":"aa"},{"id":4,"title":"aa"}],"retcode":"200"}

### 4.2.8 搜索新闻(普通用户/管理员)

<http://localhost:8080/NewsRecommend/newsSearch?token=aaa&keyword=aaa>

参数：

token： 用户token

keyword: 搜索的关键字

返回值

{"newsList":[{"id":2,"title":"天天向上"},{"id":3,"title":"天天想你"},{"id":4,"title":"天罡北斗"}],"retcode":"200"}

### 4.2.9 推荐新闻(普通用户/管理员)

根据搜索记录，推荐用户新闻

[http://localhost:8080/NewsRecommend/newsRecommend?token=aaa](http://localhost:8080/NewsRecommend/newsSearch?token=aaa)

参数：

token： 用户token

返回值

{"newsList":[{"id":2,"title":"天天向上"},{"id":3,"title":"天天想你"},{"id":4,"title":"天罡北斗"}],"retcode":"200"}

异常码：

|  |  |
| --- | --- |
| 错误码 | 说明 |
| 404 | 无搜索记录(no search history) |
|  |  |

# 5 版本历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| 1.0 | wjj |  | 2020-4-13 |  |
|  |  |  |  |  |
|  |  |  |  |  |