基于Spark技术的气温预测分析系统

实现文档

组长：王昱

组员：赵康辉 林小倩

武熠彬 张正阳

目录

[1 引言 2](#_Toc45520898)

[1.1 编写目的 2](#_Toc45520899)

[1.2 项目背景 2](#_Toc45520900)

[2 总体设计 4](#_Toc45520901)

[2.1 需求概述 4](#_Toc45520902)

[2.2 软件结构 4](#_Toc45520903)

[4 程序描述 5](#_Toc45520904)

[4.1 登录模块 5](#_Toc45520905)

[4.1.1前端实现 5](#_Toc45520906)

[4.1.2服务器端实现 5](#_Toc45520907)

[4.1.3数据交互实现 5](#_Toc45520908)

[4.2 用户管理模块 5](#_Toc45520909)

[4.2.1前端实现 5](#_Toc45520910)

[4.3 天气预测模块 5](#_Toc45520911)

[4.3.1前端实现 5](#_Toc45520912)

[4.3.2服务器端实现 5](#_Toc45520913)

[4.3.3数据交互实现 6](#_Toc45520914)

[4.3.4算法实现 6](#_Toc45520915)

[4.4 数据库实现 9](#_Toc45520916)

[4.4.1数据库连接实现 9](#_Toc45520917)

[4.4.2数据库设计 9](#_Toc45520918)

# 1 引言

## 1.1 编写目的

编写系统详细实现说明书是软件开发过程必不可少的部分，其目的是为了使开发人员在完成概要系统实现说明书的基础上完成概要设计规定的各项模块的具体实现。

作为这次小学期的结业大作业，详细系统实现说明书同时也是为了更好的让老师了解项目开发的信息与具体过程。

## 1.2 项目背景

软件名称：基于Spark技术的气温预测分析系统

软件开发平台：pycharm、eclipse、idea

软件开发语言：python、java、html、css、js

本项目的提出是基于小学期老师的要求，经过小组成员的多次讨论，经过多方面综合考虑最终决定的，该项目结合了Spark、ARIMA模型、springboot框架、vue框架、数据库等多方面知识，最终形成产品。

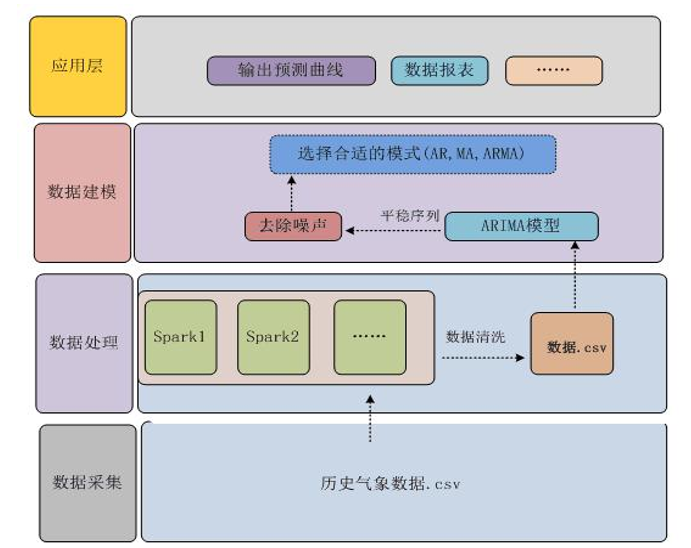
该软件运行于Windows系统平台，通过网页形式发布。

# 2 总体设计

## 2.1 需求概述

天气预测的主要内容是一个地区或城市未来一段时期内的阴晴雨雪、最高最低气温、风向和风力及特殊的灾害性天气。往小了说，具体到个人，主要是为了预防疾病，尤其是感冒，关注天气变化，加减衣服。往大了说，关乎到农业的生产发展，提前了解到天气可以更好的帮助植物定期调整种植方式。因此我们可以看出天气预测的意义重大，应用广泛，我们打算通过使用历史天气数据的方式构建模型，来实现对未来天气的预测，但是鉴于预测天气需要考虑气压、气温、风速、风向、湿度等诸多因素难以实现，我们仅针对最低温和最高温来对中国各大城市进行天气预测。

## 2.2 软件结构



# 4 程序描述

## 4.1 登录模块

### 4.1.1前端实现

登录界面由输入框和按钮组成。输出登陆成功。其他使用html、css、js进行美化和控制。

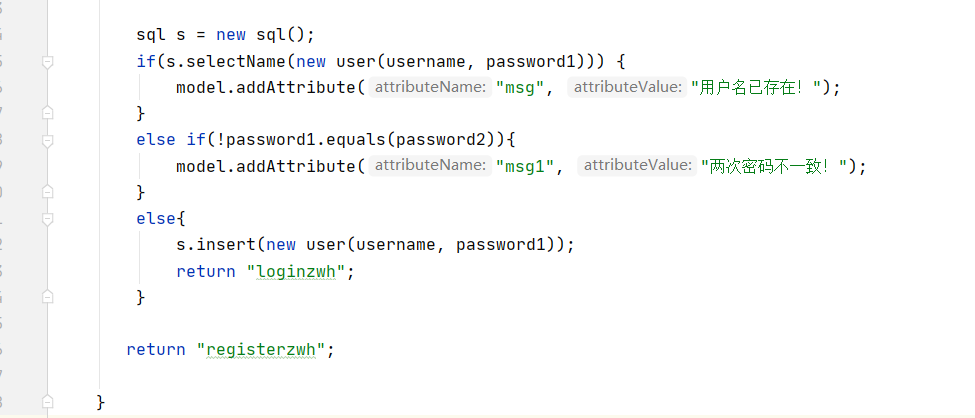
### 4.1.2服务器端实现

对比信息函数以及调用函数部分

### 3784d682bf9c1b6b327b561f92e89fc33ad74b2cf8371c92cea1ad719200bb

添加用户函数以及调用函数部分

### 0cb7cb6f50a6778903a558b97d110f0



### 4.1.3数据交互实现

**1）前端对用户名与密码的传输**

## 通过表单来实现用户名与密码的传输，“登录”的input类型为submmit，在点击后即可实现form的action内路由的发送；

<form id="login-form" action="/canilogin">

<input type="text" data-placeholder="用户名或邮箱账号" id="username" name="account" data-type="email" data-loginname="login" data-required="true" class="username" placeholder="用户名或邮箱账号" style="width:340px">

<input data-placeholder="输入密码" id="password" name="password" maxlength="50" data-required="true" class="passwd" type="password" autocomplete="new-password" data-max-length="50" tabindex="2" spellcheck="false" placeholder="输入密码" style="width:340px">

1. **后端对用户名与密码的接收**

**后端在接收到对应的路由后，通过@RequestParam + 想要获取的控件名来获得其相应的值**

ublic String search(@RequestParam("name")String username, @RequestParam("Password") String password1, @RequestParam("mainPassword") String password2){  
 setUsername(username);  
 setPassword(password1);  
 setEmail(password2);

## 4.2 用户管理模块

### 4.2.1前端实现

用户管理功能，实现增删改查。其他使用html、css和java script进行美化和控制。

## 4.3 天气预测模块

### 4.3.1前端实现

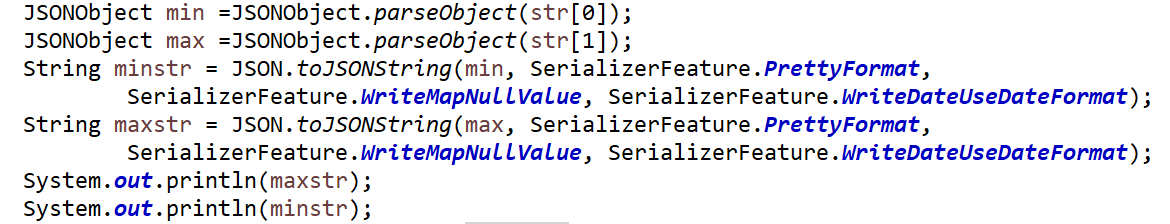
选择日期，和城市，显示天气的折线图。折线图具体使用echarts插件，通过读取json文件获取数据最后生成折线图。其他使用html、css和java script进行美化和控制

### 4.3.2服务器端实现

执行python脚本文件，生成预测数据，利用str[0]、str[1]分别接收最低温和最高温



将数据格式转换为json格式（未使用该部分代码）



### 4.3.3数据交互实现

1. **前端Json文件的传输**
2. **用Jquery内的￥加上相应的控件类型与名称即可获得对应的selcector内的选择，通过val（）函数即可将其转换成Json格式的文件**

var username = $('input[name="name"]').val();  
var password = $('input[name="mainPassword"]').val();  
var email = $('input[name="email"]').val();  
var objects= {"username":username, "password":password,"email":email};

1. **Axios来规定传输的方式、路由、数据以及数据类型等**

xios({  
  
 method: 'post',  
  
 url: '/leileleile',  
  
 data:objects  
  
});

1. **后端对Json文件的接收**

**通过@RequestBody与HttpServletResponse来进行Json文件的捕获与读入**

@RequestMapping(value = "/leileleile")  
public String search(@RequestBody adduserController sub, HttpServletResponse response){  
  
 setUsername(sub.username);  
 setPassword(sub.password);  
 setEmail(sub.email);  
  
 System.*out*.println(getUsername());  
 System.*out*.println(getPassword());  
 System.*out*.println(getEmail());  
  
 return "predict";  
}

### 4.3.4算法实现

1. **获取数据**

**从美国noaa网站上下载全国各个省会城市的气温数据（csv格式）。**

1. **清洗数据**

**通过clean（）函数读取数据，clean\_appointed\_data(area,month**

**,day,type)调用clean函数,并且依参数清洗数据并写入csv文档。具体代码如下：**

def clean(area):  
 # 读取数据  
 data\_raw = pd.read\_csv('D:\\Python Code\\PycharmProjects\\weatherDataAnalysis\\mod\_timeseries\\origin\_data\\' + area + '.csv',  
 encoding='utf-8')  
 data\_raw['date'] = data\_raw['DATE'].apply(parser.parse) # 把DATE列解析成日期格式  
 data\_raw['tmax'] = data\_raw['TMAX'].astype(float) # 变量转换为float  
 data\_raw['tmin'] = data\_raw['TMIN'].astype(float)  
 # 得到数据后,过滤空值  
 data = data\_raw.loc[:, ['date', 'tmax', 'tmin']] # 筛选出需要的三个列  
 # 过滤空值  
 data = data[(pd.Series.notnull(data['tmax'])) & (pd.Series.notnull(data['tmin']))]  
 return data  
# 筛选出1981到2012的日期  
def init(area):  
 dta = clean(area)  
 dta = dta[(dta['date'] >= datetime(1981, 1, 1)) & (dta['date'] <= datetime(2012, 12, 31))]  
 return dta  
def clean\_appointed\_day(area, month, day):  
 data = init(area)  
 temp = "date.dt.day ==" + str(day) + " & date.dt.month==" + str(month)  
 data.query(temp, inplace=True) # 得到历史上每年month.day的数据  
 filename = "..\\mod\_timeseries\\washed\_data\\washed\_data.csv"  
 data.to\_csv(filename, index=None)

1. **确定参数**

**通过get\_p\_d\_q()函数获得该组数据集应该使用的pdq参数。实现方法为将pq分别从2到7取值建模，跳过报错的模型（报错说明参数不匹配），最后通过比较bic，确定最佳pq值。**

# 返回pdq 由于事先跑了一遍都是原数据最优，故直接采用d=0，但pq值不同  
def get\_p\_d\_q(dta):  
 min\_bic = 10000  
 min\_p = 0  
 min\_q = 0  
 for p in range(2, 5):  
 for q in range(2, 5):  
 try:  
 arma\_mod = sm.tsa.ARMA(dta, (p, q)).fit(disp=False)  
 # print(p, q, arma\_mod.bic)  
 if arma\_mod.bic < min\_bic:  
 min\_bic = arma\_mod.bic  
 min\_p = p  
 min\_q = q  
 except:  
 pass  
 # print('跳过')  
 # print(min\_p, min\_q, min\_bic)  
  
 return min\_p, 0, min\_q

1. **建模**

**在获得相应pq值后，直接调用引入包的函数生成arima模型。**

arma\_mod = sm.tsa.ARMA(dta, (p, q)).fit(disp=False)

1. **预测一周天气**

**通过输入的相应城市找到数据源文件，清洗后建模预测未来10年数据，选取年份为2020年的预测并返回，重复7次。（由于数据集完整部分只到2012年，故只能这样获取2020年天气，后果是预测准确性降低）**

def get\_predict(area, month, day, type):  
 clean\_appointed\_day(area, month, day)  
 # 读文件 初始化  
 data = read\_file(area, month, day)  
 datatype = type  
 dta = data[datatype]  
 dta\_year = data['date']  
  
 # 得到开始年份和结束年份  
 begin\_year = dta\_year[0:1].dt.year  
 end\_year = dta\_year[-1:].dt.year  
 # print(begin\_year.values[0], end\_year.values[0])  
  
 # 设置数据类型  
 dta = np.array(dta, dtype=np.float)  
 # 转换为Series类型的一位数组  
 dta = pd.Series(dta)  
 # 改索引为年份  
 dta.index = pd.Index(sm.tsa.datetools.dates\_from\_range(str(begin\_year.values[0]), str(end\_year.values[0])))  
  
 # 获得pdq  
 temp = readPQ('pq\\'+area + '-pq-' + type + '.txt')  
 p = int(temp[str(month) + '-' + str(day)]) // 10  
 q = int(temp[str(month) + '-' + str(day)]) % 10  
 arma\_mod = sm.tsa.ARMA(dta, (p, q)).fit(disp=False)  
  
 # 未来10年同一天的预测数据  
 predict\_year = 10  
 predict\_end\_year = end\_year.values[0] + predict\_year  
 predict\_dta = arma\_mod.predict(str(end\_year.values[0]), str(predict\_end\_year), dynamic=True)  
 return predict\_dta[8]

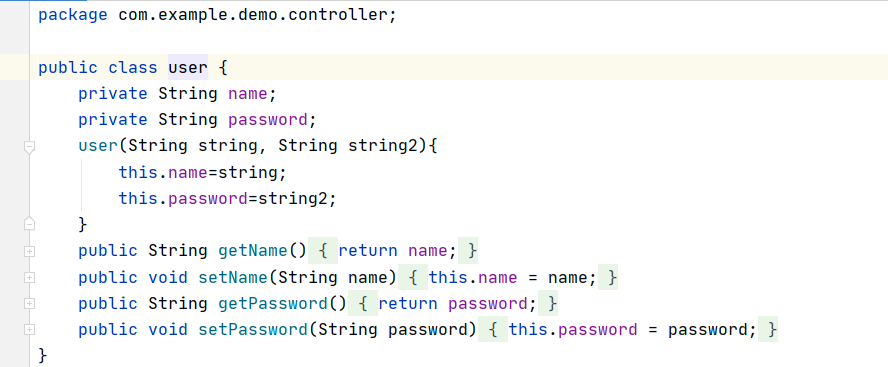
1. **转换json格式**

**引入json包，直接调用dump函数转换成json格式并写入文件。**

def json\_transfer(temp, type):  
 with open("predict\_file\\" + type + "-record.json", "w") as f:  
 j = json.dumps(temp)  
 f.write(j)  
 # print("加载入文件完成...")  
 f.close()

## 4.4 数据库实现

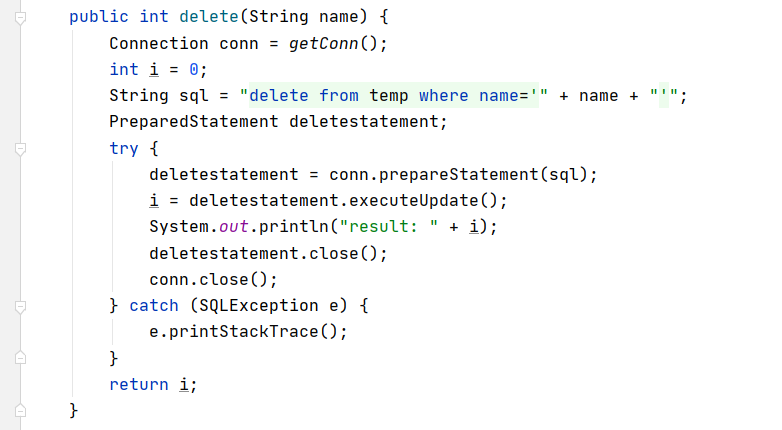
接收数据信息



数据库的增添操作



数据库的删除操作



数据库的查找操作（用户名和密码）

