

### 全国疫情实时追踪

2020-5-6 23:13:57

#### 全国累计趋势

统计项目	数值
累计确诊	84406
现有疑似	5
累计治愈	79204
累计死亡	4643

#### 非湖北地区城市确诊TOP5

地区待确认	北京	温州
地区待确认	650	580

#### 全国新增趋势

统计项目	数值
新增确诊	84406
新增疑似	5
新增死亡	4643
新增治愈	79204

#### 地区待确认

#### 今日疫情热搜

## 所用技术

## 实现过程

## 实现过程

- 用webmagic爬取腾讯,百度疫情网站,获取数据
- 将返回的数据存储在mysql中
- 编写业务,在controller中调用业务
- 用ajax获取controller传来的数据

- 将返回的数据存储在mysql中
- 编写业务,在controller中调用业务
- 用ajax获取controller传来的数据

编写业务,在controller中调用业务

用ajax获取controller传来的数据

## 用ajax获取controller传来的数据

## 环境和软件

JDK1.8 ,Intellij IDEA 2020.1 x64, MySQL 5.5.40,node.js v12.16.2 ,Maven

## 依赖

## 依赖

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
```

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
```

```
<version>2.0.2.RELEASE</version>
</parent>

<properties>
  <java.version>1.8</java.version>
</properties>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.12</version>
  </dependency>
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.9.1</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
  </dependency>
  <dependency>
    <groupId>com.huaban</groupId>
    <artifactId>jieba-analysis</artifactId>
    <version>1.0.2</version>
  </dependency>
  <!--SpringMVC-->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
  </dependency>
  <dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
```

```

        <version>1.1.1</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
    <dependency>
        <groupId>us.codecraft</groupId>
        <artifactId>webmagic-core</artifactId>
        <version>0.7.3</version>
    </dependency>
    <dependency>
        <groupId>us.codecraft</groupId>
        <artifactId>webmagic-extension</artifactId>
        <version>0.7.3</version>
    </dependency>
    <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-lang3</artifactId>
    </dependency>
</dependencies>

```

## 数据库搭建

### #详情信息

```

CREATE TABLE `details` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `update_time` varchar(50) DEFAULT NULL COMMENT '数据最后更新时间',
  `province` varchar(50) DEFAULT NULL COMMENT '省',
  `city` varchar(50) DEFAULT NULL COMMENT '市',
  `confirm` int(11) DEFAULT NULL COMMENT '累计确诊',
  `confirm_add` int(11) DEFAULT NULL COMMENT '新增确诊',
  `heal` int(11) DEFAULT NULL COMMENT '累计治愈',
  `dead` int(11) DEFAULT NULL COMMENT '累计死亡',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1826 DEFAULT CHARSET=utf8mb4

```

### #历史信息

```

CREATE TABLE `history` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `ds` varchar(50) NOT NULL COMMENT '日期',
  `confirm` int(11) DEFAULT NULL COMMENT '累计确诊',
  `confirm_add` int(11) DEFAULT NULL COMMENT '当日新增确诊',
  `suspect` int(11) DEFAULT NULL COMMENT '剩余疑似',
  `suspect_add` int(11) DEFAULT NULL COMMENT '当日新增疑似',
  `heal` int(11) DEFAULT NULL COMMENT '累计治愈',
  `heal_add` int(11) DEFAULT NULL COMMENT '今日新增治愈',
  `dead` int(11) DEFAULT NULL COMMENT '累计死亡',
  `dead_add` int(11) DEFAULT NULL COMMENT '当日新增死亡',
  PRIMARY KEY (`id`)
)

```

```
) ENGINE=InnoDB AUTO_INCREMENT=108 DEFAULT CHARSET=utf8mb4
```

#热搜

Create Table

```
CREATE TABLE `hot` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `dt` varchar(50) DEFAULT NULL,  
  `content` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=1085 DEFAULT CHARSET=utf8mb4
```

## 爬虫源网址

腾讯

[https://view.inews.qq.com/g2/getOnsInfo?name=disease\\_h5](https://view.inews.qq.com/g2/getOnsInfo?name=disease_h5) [https://view.inews.qq.com/g2/getOnsInfo?name=disease\\_other](https://view.inews.qq.com/g2/getOnsInfo?name=disease_other)

百度

<https://voice.baidu.com/act/virussearch/virussearch>

## Application配置类

DB Configuration:

```
spring.datasource.driverClassName=com.mysql.jdbc.Driver  
spring.datasource.url=jdbc:mysql://localhost:3306/pro?  
serverTimezone=UTC&useUnicode=true&characterEncoding=utf-8  
spring.datasource.username=root  
spring.datasource.password=root
```

```
mybatis.type-aliases-package=CR553.Pojo  
mybatis.mapper-locations=classpath:Mapper/*.xml
```

```
server.port=8088
```

//关闭缓存

```
spring.thymeleaf.cache=false
```

## POJO,Mapper,Service

没啥特别的,基本的crud,直接上代码

POJO

```

@Data//依赖于lombok
@AllArgsConstructor
@NoArgsConstructor
public class Details {
    private Long id;
    private String update_time; //更新时间
    private String province; //省
    private String city; //市
    private Long confirm; //累计确诊
    private Long confirm_add; //今日新增确诊
    private Long heal; //治愈
    private Long dead; //死亡
}

```

```

@Data
@NoArgsConstructor
@AllArgsConstructor
public class History {
    private Long id;
    private String ds; //日期
    private Long confirm; //累计确诊
    private Long confirm_add; //今日累计确诊
    private Long suspect; //疑似
    private Long suspect_add; //今日新增疑似
    private Long heal; //治愈
    private Long heal_add; //当日新增治愈
    private Long dead; //死亡
    private Long dead_add; //今日新增死亡
}

```

```

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Hot {
    private Long id;
    private String content;
    private String dt;
}

```

mapper

```

@Mapper
public interface DetailsMapper {

    //存储
    void saveDetails(Details details);

    //更新
    void updateDetails(Details details);
}

```

```
//查找(省份和市名相同的)
List<Details> findDetails(Details details);

//查找省
List<String> findProvince();

//查找每个省的确诊人数
List<Integer> findProvinceValue();

//查找城市
List<String> findCity();

//查找每个城市的确诊人数
List<Long> findCityValue();

}
```

```
@Mapper
public interface HistoryMapper {

    //保存
    void saveHistory(History history);

    //更新
    void updateHistory(History history);

    //查找 日期相同的
    List<History> findHistory(History history);

    //查找今日数据
    History findToday();

    //返回每天历史累计数据
    List<History> findEachDayTotal();

    //返回每天历史增加数据
    List<History> findEachDayAdd();

}
```

```
@Mapper
public interface HotMapper {

    //保存
    void saveHot(Hot hot);

    List<Hot> findTopHot20();

}
```

serviceImp

```
@Service
public class DetailsServiceImpl implements DetailsService {

    @Autowired
    private DetailsMapper detailsMapper;

    //查找并更新
    @Override
    public void saveDetails(Details details) {
        List<Details> list = this.findDetails(details);

        if (list.size() == 0) {
            //没查到,新增
            this.detailsMapper.saveDetails(details);
        }else
        {
            //查到了,修改
            this.detailsMapper.updateDetails(details);
        }
    }

    @Override
    public void updateDetails(Details details) {
        this.updateDetails(details);
    }

    @Override
    public List<Details> findDetails(Details details) {
        List<Details> list = this.detailsMapper.findDetails(details);
        return list;
    }

    @Override
    public List<String> findProvince() {
        List<String> list = this.detailsMapper.findProvince();
        return list;
    }

    @Override
```

```

    public List<Integer> findProvinceValue() {
        List<Integer> list = this.detailsMapper.findProvinceValue();
        return list;
    }

    @Override
    public List<String> findCity() {
        List<String> city = this.detailsMapper.findCity();
        return city;
    }

    @Override
    public List<Long> findCityValue() {
        List<Long> cityValue = this.detailsMapper.findCityValue();
        return cityValue;
    }
}

```

```

@Service
public class HistoryServiceImpl implements HistoryService {

    @Autowired
    private HistoryMapper historyMapper;

    @Override
    public void saveHistory(History history) {
        List<History> list = this.historyMapper.findHistory(history);
        if(list.size()==0)
        {

            this.historyMapper.saveHistory(history);
        }else
        {
            this.historyMapper.updateHistory(history);
        }
    }

    @Override
    public void updateHistory(History history) {
        this.historyMapper.updateHistory(history);
    }

    @Override
    public List<History> findHistory(History history) {
        return this.historyMapper.findHistory(history);
    }

    @Override
    public History findToday() {
        return this.historyMapper.findToday();
    }
}

```



```

    }

    @Override
    public List<History> findEachDayTotal() {
        return this.historyMapper.findEachDayTotal();
    }

    @Override
    public List<History> findEachDayAdd() {
        return this.historyMapper.findEachDayAdd();
    }

}

```

```

@Service
public class HotServiceImpl implements HotService {

    @Autowired
    private HotMapper hotMapper;

    @Override
    public void saveHot(Hot hot) {
        this.hotMapper.saveHot(hot);
    }

    @Override
    public List<Hot> findTopHot20() {
        return this.hotMapper.findTopHot20();
    }

}

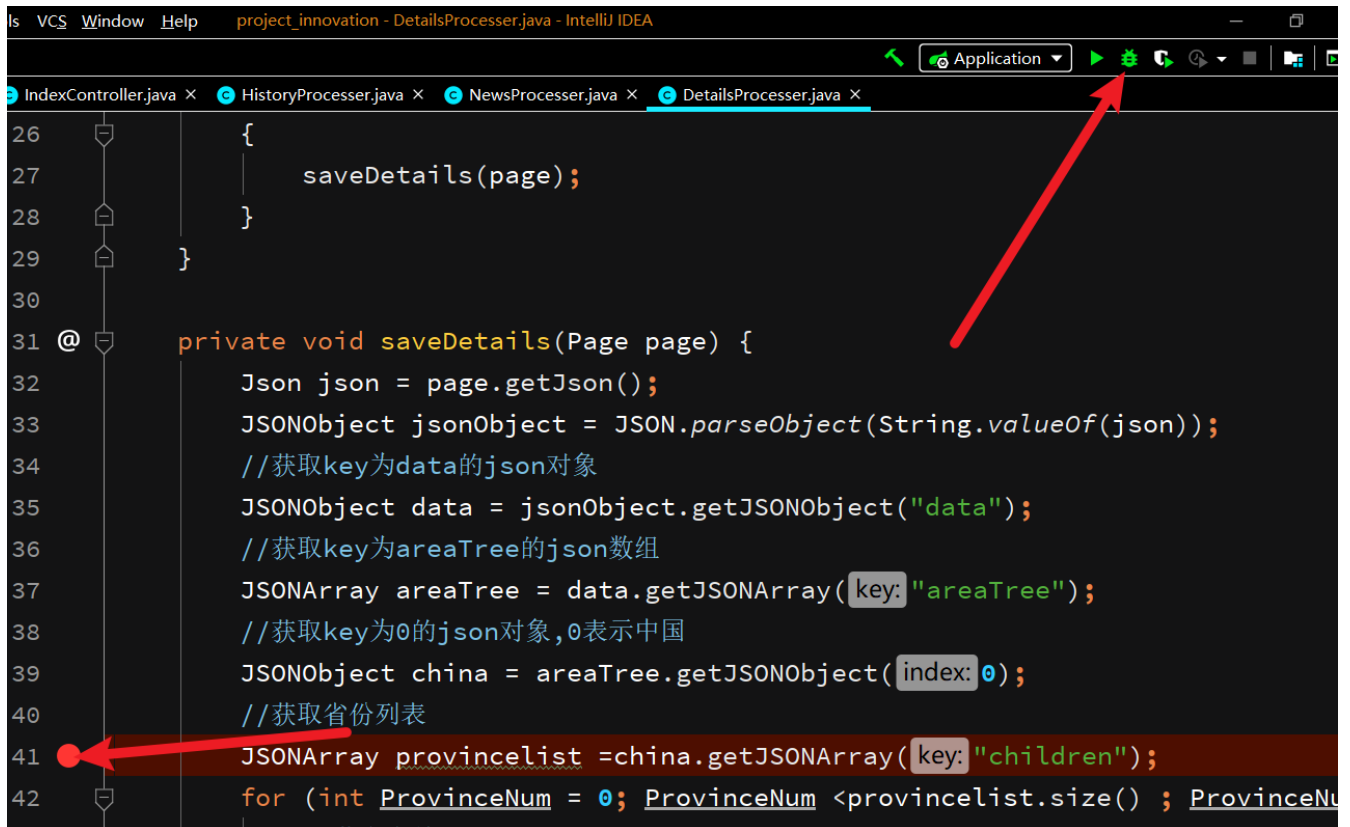
```

## Webmagic爬虫

---

可能就是在解析数据的时候要花一点点时间

这里是通过打个断点,debug慢慢分析的



```
26 {
27     saveDetails(page);
28 }
29 }
30
31 @private void saveDetails(Page page) {
32     Json json = page.getJson();
33     JSONObject jsonObject = JSON.parseObject(String.valueOf(json));
34     //获取key为data的json对象
35     JSONObject data = jsonObject.getJSONObject("data");
36     //获取key为areaTree的json数组
37     JSONArray areaTree = data.getJSONArray(key: "areaTree");
38     //获取key为0的json对象,0表示中国
39     JSONObject china = areaTree.getJSONObject(index: 0);
40     //获取省份列表
41     JSONArray provincelist = china.getJSONArray(key: "children");
42     for (int ProvinceNum = 0; ProvinceNum < provincelist.size(); ProvinceNum++) {
43         //获取省份
```

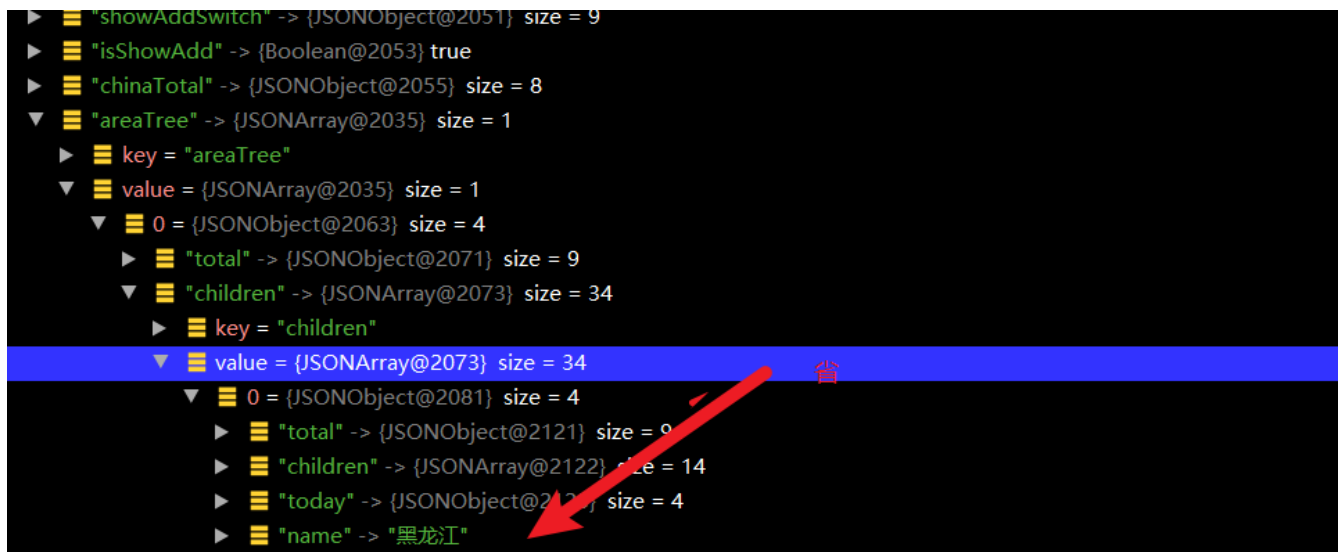
差不多就这样

先看details数据的解析



```
data = {JSONObject@7403} size = 6
  ▶ showAddSwitch -> {JSONObject@7479} size = 9
  ▶ isShowAdd -> {Boolean@7481} true
  ▶ chinaTotal -> {JSONObject@7483} size = 8
  ▶ areaTree -> {JSONArray@7464} size = 1
  ▶ chinaAdd -> {JSONObject@7486} size = 8
  ▶ lastUpdateTime -> "2020-05-06 18:16:12"
```

数据在这两个中获取



```
▶ showAddSwitch -> {JSONObject@2051} size = 9
▶ isShowAdd -> {Boolean@2053} true
▶ chinaTotal -> {JSONObject@2055} size = 8
▼ areaTree -> {JSONArray@2035} size = 1
  ▶ key = "areaTree"
  ▼ value = {JSONArray@2035} size = 1
    ▼ 0 = {JSONObject@2063} size = 4
      ▶ total -> {JSONObject@2071} size = 9
      ▼ children -> {JSONArray@2073} size = 34
        ▶ key = "children"
        ▼ value = {JSONArray@2073} size = 34
          ▼ 0 = {JSONObject@2081} size = 4
            ▶ total -> {JSONObject@2121} size = 9
            ▶ children -> {JSONArray@2122} size = 14
            ▶ today -> {JSONObject@2123} size = 4
            ▶ name -> "黑龙江"
```

```
▼ 0 = {JSONObject@2081} size = 4
  ▶ "total" -> {JSONObject@2121} size = 9
  ▼ "children" -> {JSONArray@2122} size = 14
    ▶ key = "children"
    ▼ value = {JSONArray@2122} size = 14
      ▶ 0 = {JSONObject@2127} size = 3
      ▼ 1 = {JSONObject@2128} size = 3
        ▶ "total" -> {JSONObject@2154} size = 9
        ▶ "today" -> {JSONObject@2155} size = 3
        ▶ "name" -> "哈尔滨市"
        ▶ 2 = {JSONObject@2129} size = 3
        ▶ 3 = {JSONObject@2130} size = 3
        ▶ 4 = {JSONObject@2131} size = 3
        ▶ 5 = {JSONObject@2132} size = 3
        ▶ 6 = {JSONObject@2133} size = 3
```

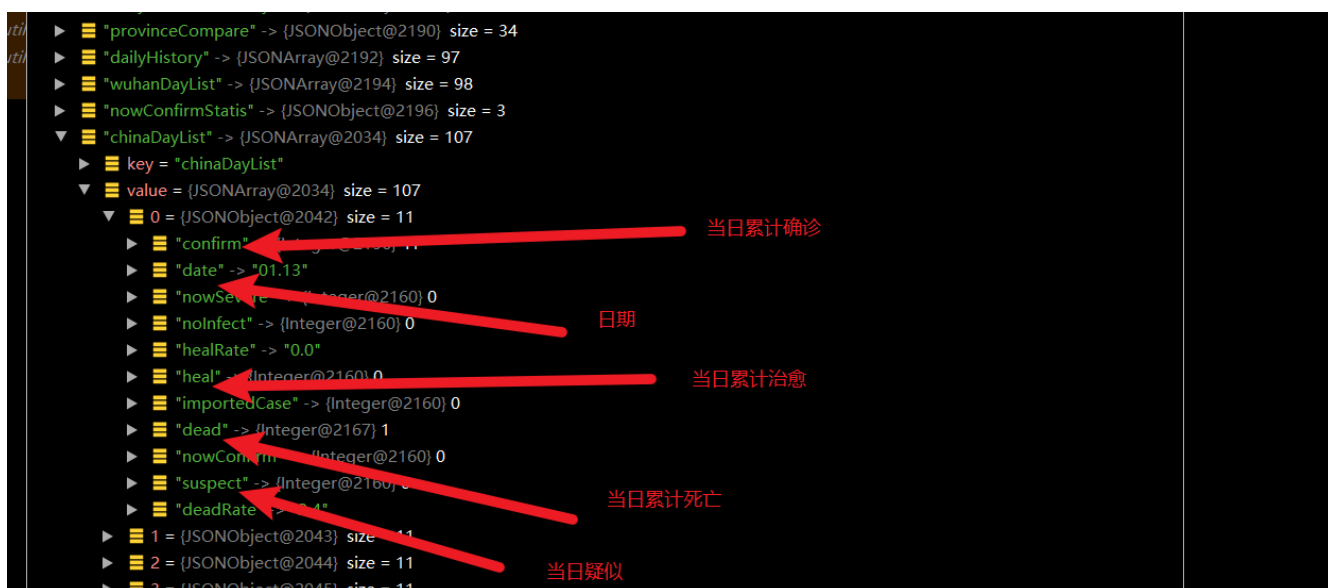
```
▶ key = "children"
▼ value = {JSONArray@2073} size = 34
  ▼ 0 = {JSONObject@2081} size = 4
    ▶ "total" -> {JSONObject@2121} size = 9
    ▼ "children" -> {JSONArray@2122} size = 14
      ▶ key = "children"
      ▼ value = {JSONArray@2122} size = 14
        ▼ 0 = {JSONObject@2127} size = 3
          ▼ "total" -> {JSONObject@2146} size = 9
            ▶ key = "total"
            ▼ value = {JSONObject@2146} size = 9
              ▶ "confirm" -> {Integer@2293} 386
              ▶ "showRate" -> {Boolean@2190} false
              ▶ "healRate" -> "30.05"
              ▶ "heal" -> {Integer@2295} 116
              ▶ "nowConfirm" -> {Integer@2296} 270
              ▶ "dead" -> {Integer@2172} 0
              ▶ "suspect" -> {Integer@2172} 0
              ▶ "deadRate" -> "0.00"
              ▶ "showHeal" -> {Boolean@2053} true
            ▶ "today" -> {JSONObject@2147} size = 3
              ▶ key = "today"
              ▼ value = {JSONObject@2147} size = 3
                ▶ "confirm" -> {Integer@2172} 0
                ▶ "isUpdated" -> {Boolean@2190} false
                ▶ "confirmCuts" -> {Integer@2172} 0
              ▶ "name" -> "境外输入"
            ▶ 1 = {JSONObject@2128} size = 3
            ▶ 2 = {JSONObject@2129} size = 3
            ▶ 3 = {JSONObject@2130} size = 3
```

本质上就是遍历一个双重for循环

history的

注意的也是两个json数据

```
▶ "chinaDayAddList" -> {JSONArray@7532} size = 108
▶ "articleList" -> {JSONArray@7551} size = 20
▶ "dailyNewAddHistory" -> {JSONArray@7553} size = 107
▶ "provinceCompare" -> {JSONObject@7555} size = 34
▶ "dailyHistory" -> {JSONArray@7557} size = 104
▶ "wuhanDayList" -> {JSONArray@7559} size = 98
▶ "nowConfirmStatis" -> {JSONObject@7561} size = 3
▶ "chinaDayList" -> {JSONArray@7563} size = 115
▶ "cityStatis" -> {JSONObject@7565} size = 3
```



这里要注意一下, chinaDayList 中的日期是从1月20号开始的, 而 chinaDayAddList 中的日期是从1月13号开始的, 一个for循环就可以搞定了.

热搜hot表数据很少, debug一下很容易发现

## 数据的保存

以details为例

把数据先保存在ResultItems中, 我这里用了随机id, 防止key相同

```

        UUID uuid = UUID.randomUUID();
        //保存结果
        page.putField(key: "details"+uuid,details);
    }

```

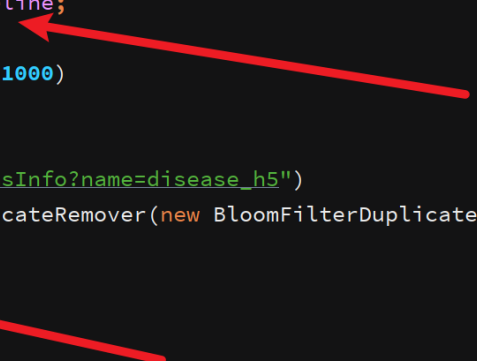
使用pipepel

```

@Autowired
private DetailsSpringDataPipeline detailsSpringDataPipeline;

@Scheduled(initialDelay = 1000, fixedDelay = 12*60 *60* 1000)
public void process(){
    Spider.create(new DetailsProcessor())
        .addUrl("https://view.inews.qq.com/g2/get0nsInfo?name=disease_h5")
        .setScheduler(new QueueScheduler().setDuplicateRemover(new BloomFilterDuplicateRemover(
        .addPipeline(detailsSpringDataPipeline)
        .thread(5)
        .run();
}

```



遍历ResultItems,调用detailsService方法保存数据到数据库

```

@Autowired
private DetailsService detailsService;

@Override
public void process(ResultItems resultItems, Task task) {

    Map<String, Object> all = resultItems.getAll();

    for (Map.Entry<String, Object> entry : all.entrySet()) {
        String key = entry.getKey();
        Details details = resultItems.get(key);
        this.detailsService.saveDetails(details);
    }
}

```

另外两个类似操作

## Controller+SQL+ajax

数据保存到数据库后,剩下就是发送数据了

整个数据的展示大体分为了六个部分,依次是l1,l2,c1,c2,r1,r2

c1

84406

累计确诊

5

剩余疑似

79204

累计治愈

4643

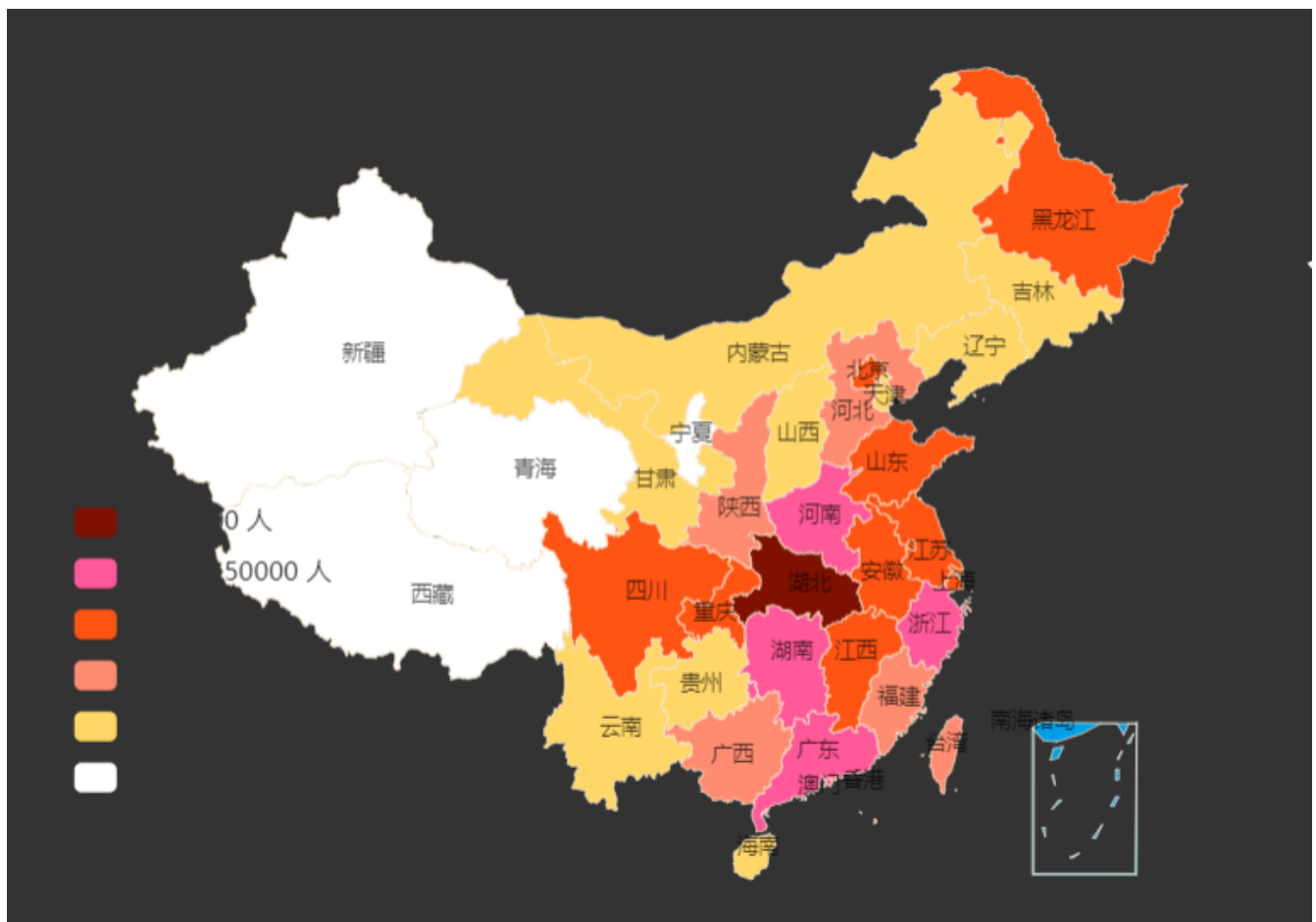
累计死亡

```
@RequestMapping("/c1")
@ResponseBody
public JSONObject getDateC1()
{
    History today = this.historyService.findToday();
    JSONObject json = new JSONObject();
    json.put("confirm",today.getConfirm());
    json.put("suspect",today.getSuspect());
    json.put("heal",today.getHeal());
    json.put("dead",today.getDead());
    return json;
}
```

```
<select id="findToday" resultType="History">
    SELECT * FROM history ORDER BY ds DESC LIMIT 1;
</select>
```

```
function get_c1_data(){
    $.ajax({
        url: "/c1",
        success:function (data) {
            $(".num h1").eq(0).text(data.confirm);
            $(".num h1").eq(1).text(data.suspect);
            $(".num h1").eq(2).text(data.heal);
            $(".num h1").eq(3).text(data.dead);
        }
    });
}
```

c2



```

@RequestMapping("/c2")
@ResponseBody
public JSONArray c2()
{
    JSONArray list = new JSONArray();
    List<String> province = this.detailsService.findProvince();
    List<Integer> provinceValue = this.detailsService.findProvinceValue();
    for (int i = 0; i < province.size() ; i++) {
        JSONObject js = new JSONObject();
        js.put("name",province.get(i));
        js.put("value",provinceValue.get(i));
        list.add(js);
    }
    return list;
}

```

```

<select id="findProvince" resultType="String">
    SELECT province FROM details
    WHERE update_time=(SELECT update_time FROM details
    ORDER BY update_time DESC LIMIT 1 )
    GROUP BY province;
</select>

```

```

<select id="findProvinceValue" resultType="Integer">
    SELECT SUM(confirm) FROM details
    WHERE update_time=(SELECT update_time FROM details
    ORDER BY update_time DESC LIMIT 1 )
    GROUP BY province;
</select>

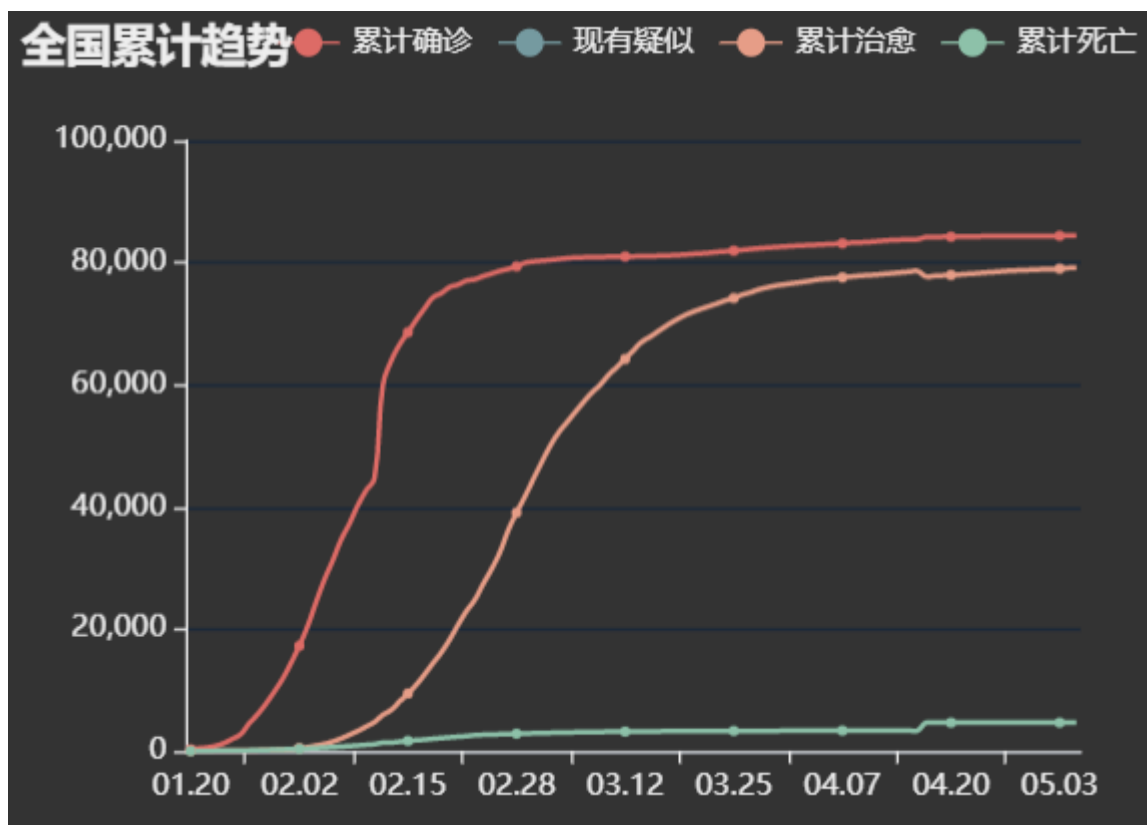
```



```
function get_c2_data() {
  $.ajax({
    url: "/c2",
    success: function (response) {
      ec_center2_option.series[0].data=response;
      //console.log(response);
      // console.log(ec_center2_option.series[0].data);
      ec_center2.setOption(ec_center2_option);
    }
  });
}
```

这里的两个sql其实是可以放一起,我这里多走了一步路

11



```

@RequestMapping("/l1")
@ResponseBody
public JSONObject l1()
{
    List<History> eachDayTotal = this.historyService.findEachDayTotal
    JSONObject json = new JSONObject();
    List<String> daylist=new ArrayList<>();
    List<Long> confirmlist=new ArrayList<>();
    List<Long> heallist=new ArrayList<>();
    List<Long> deadlist=new ArrayList<>();
    List<Long> suspectlist=new ArrayList<>();

    for (History history : eachDayTotal) {
        daylist.add(history.getDs());
        confirmlist.add(history.getConfirm());
        heallist.add(history.getHeal());
        deadlist.add(history.getDead());
        suspectlist.add(history.getSuspect());
    }

```

```

        json.put("day",daylist);
        json.put("confirm",confirmlist);
        json.put("heal",heallist);
        json.put("dead",deadlist);
        return json;
    }

```

```

<select id="findEachDayTotal" resultType="History">
    SELECT ds,confirm,suspect,heal,dead FROM history;
</select>

```

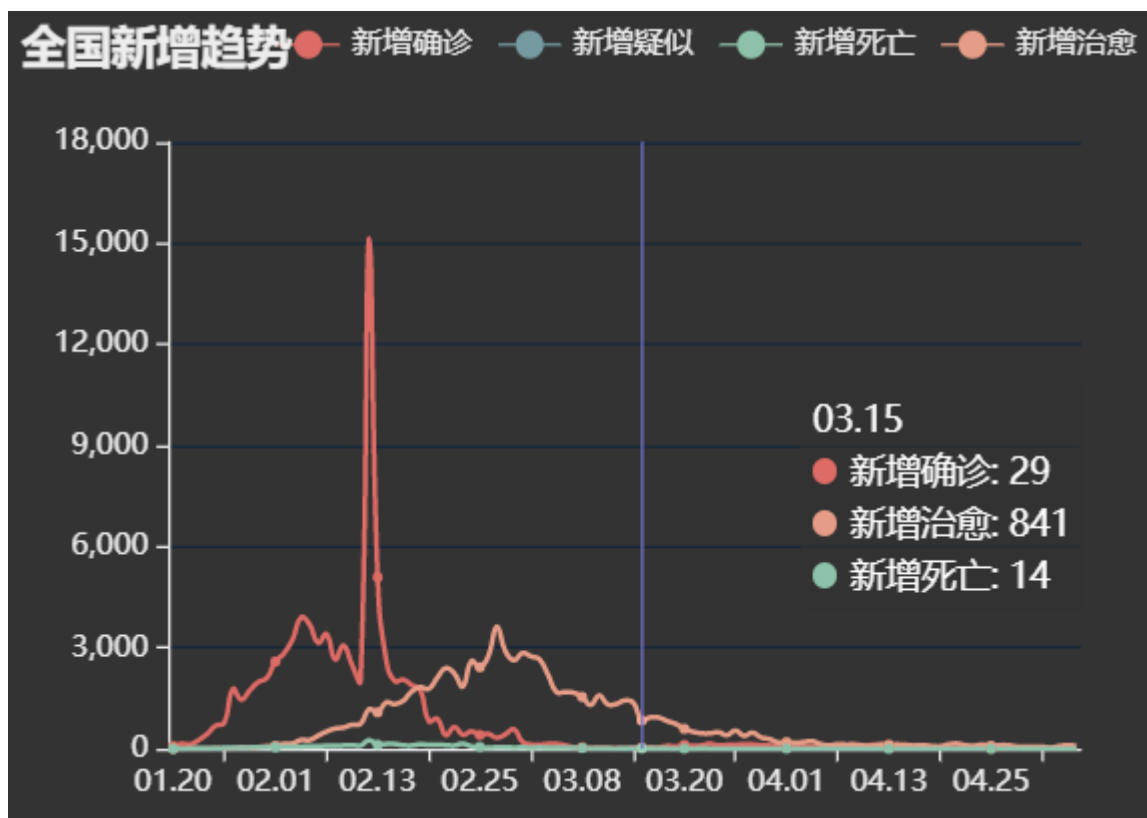
```

function get_l1_data() {
    $.ajax({
        url: "/l1",
        success: function (response) {
            // console.log(response);
            ec_left1_option.xAxis[0].data=response.day;
            ec_left1_option.series[0].data=response.confirm;
            ec_left1_option.series[1].data=response.suspect;
            ec_left1_option.series[2].data=response.heal;
            ec_left1_option.series[3].data=response.dead;

            ec_left1.setOption(ec_left1_option);
        }
    });
}

```

12



```

@RequestMapping("/l2")
@ResponseBody
public JSONObject l2()
{
    List<History> eachDayTotal = this.historyService.findEachDayAdd();
    JSONObject json = new JSONObject();
    List<String> daylist=new ArrayList<>();
    List<Long> confirmlist=new ArrayList<>();
    List<Long> heallist=new ArrayList<>();
    List<Long> deadlist=new ArrayList<>();
    List<Long> suspectlist=new ArrayList<>();

    for (History history : eachDayTotal) {
        daylist.add(history.getDs());
        confirmlist.add(history.getConfirm_add());
        heallist.add(history.getHeal_add());
        deadlist.add(history.getDead_add());
        suspectlist.add(history.getSuspect_add());
    }
}

```

```

        json.put("day",daylist);
        json.put("confirm",confirmlist);
        json.put("heal",heallist);
        json.put("dead",deadlist);
        return json;
    }
}

```

```

<select id="findEachDayAdd" resultType="History">
    SELECT ds,confirm_add,suspect_add,heal_add,dead_add FROM history;
</select>

```

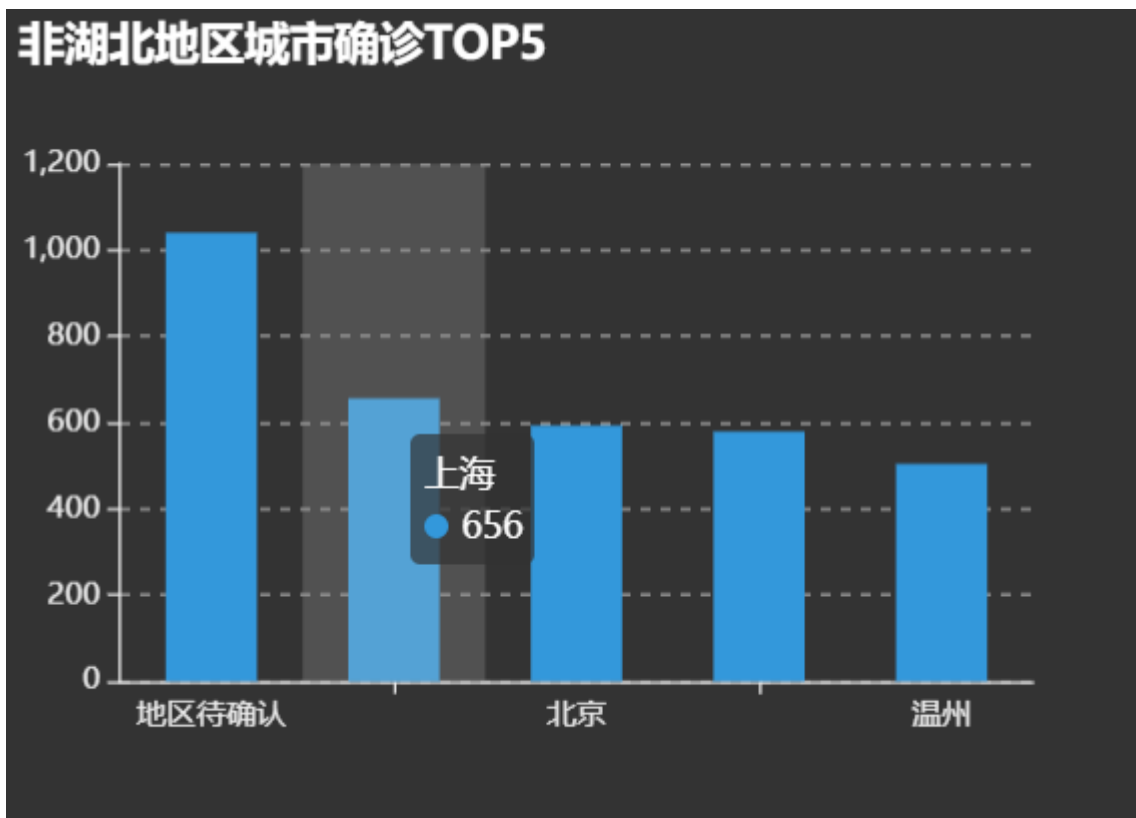
```

function get_l2_data() {
  $.ajax({
    url: "/l2",
    success: function (response) {
      // console.log(response);
      ec_left2_option.xAxis[0].data=response.day;
      ec_left2_option.series[0].data=response.confirm;
      ec_left2_option.series[1].data=response.suspect;
      ec_left2_option.series[2].data=response.heal;
      ec_left2_option.series[3].data=response.dead;

      ec_left2.setOption(ec_left2_option);
    }
  });
}

```

r1



```

@RequestMapping("/r1")
@ResponseBody
public JSONObject r1()
{
    JSONObject json = new JSONObject();
    List<String> city = this.detailsService.findCity();
    List<Long> cityValue = this.detailsService.findCityValue();
    json.put("city",city);
    json.put("cityValue",cityValue);
    return json;
}
}

```

```

<select id="findCity" resultType="String">
    SELECT city FROM
    (SELECT city,confirm FROM details
    WHERE update_time=(SELECT update_time FROM details ORDER BY update_time DESC LIMIT 1)
    AND province NOT IN("湖北","北京","上海","天津","重庆")
    UNION ALL
    SELECT province AS city ,SUM(confirm) AS confirm FROM details
    WHERE update_time=(SELECT update_time FROM details ORDER BY update_time DESC LIMIT 1)
    AND province IN("北京","上海","天津","重庆") GROUP BY province ) AS a
    ORDER BY confirm DESC LIMIT 5;
</select>

```

```

<select id="findCityValue" resultType="Long">
    SELECT confirm FROM
    (SELECT city,confirm FROM details
    WHERE update_time=(SELECT update_time FROM details ORDER BY update_time DESC LIMIT 1)
    AND province NOT IN("湖北","北京","上海","天津","重庆")
    UNION ALL
    SELECT province AS city ,SUM(confirm) AS confirm FROM details
    WHERE update_time=(SELECT update_time FROM details ORDER BY update_time DESC LIMIT 1)
    AND province IN("北京","上海","天津","重庆") GROUP BY province ) AS a
    ORDER BY confirm DESC LIMIT 5;
</select>

```

这两个sql其实是可以放在一起的,但是我的放一起只查出了一列数据,于是就多走了一步路

## r2

这里用到了jieba分词器,将分词后的关键字和数值返回给前端

[illegible]

```

@RequestMapping("/r2")
@ResponseBody
public JSONArray r2()
{
    //获取hot前20
    List<Hot> topHot20 = this.hotService.findTopHot20();
    //jieba分词
    JiebaSegmenter segmenter = new JiebaSegmenter();
    JSONArray list = new JSONArray();
    for (Hot hot : topHot20) {
        String content=hot.getContent();
        //获取数字
        String num = content.replaceAll(regex: "[\\u4e00-\\u9fa5]", replacement: "");
        //获取中文
        String msg= content.replaceAll(regex: "[0-9]", replacement: "");
        //分词
        List<String> strings = segmenter.sentenceProcess(msg);

        for (String string : strings) {
            Pattern pattern = Pattern.compile("[0-9]*");
            if(!pattern.matcher(string).matches())//不为数字
            {
                JSONObject jsonObject = new JSONObject();
                jsonObject.put("name",string);
                jsonObject.put("value",num);
                list.add(jsonObject);
            }
        }
    }
    return list;
}

```

```
<select id="findTopHot20"    resultType="Hot" >
    SELECT content FROM hot ORDER BY id DESC LIMIT 20;
</select>
```

```
function get_r2_data() {
    $.ajax({
        url: "/r2",
        success: function (response) {
            ec_right2_option.series[0].data= response;
            // console.log(response);
            ec_right2.setOption(ec_right2_option);
        }
    });
}
```

## 其他技术

地图和词云需要先引入相关js文件,相关文件都可以在echarts官网下载

<https://echarts.apache.org/zh/download-extension.html>

五分钟上手echarts

<https://echarts.apache.org/zh/tutorial.html#5%20%E5%88%86%E9%92%9F%E4%B8%8A%E6%89%8B%20ECharts>

简单实用jieba

<https://blog.csdn.net/wbcg111/article/details/53191721>

selenium基本实用

[https://blog.csdn.net/qg\\_22003641/article/details/79137327](https://blog.csdn.net/qg_22003641/article/details/79137327)

webmagic官方文档 <http://webmagic.io/docs/zh/>

## 总结

这个项目简单,基础,适合新手.这是我学完springboot 之后第一个小项目,做之前感觉无从下手,做完后又觉得没有什么,项目是参考b站上一个基于python实现的视频做的. 数据库的搭建,sql语句,都是模仿的.自己的部分主要是爬虫和业务的编写.尽管如此,我还是花了好几天时间,从一开始的懵逼到慢慢拨云见日,做完还是有一丢丢的成就感的.

获取资源的路有点曲折,那个视频拿资料要加vx,我一加,说要去他们的培训机构官网注册账号,当时嫌麻烦,没要



在视频的评论区有人先做出来了,基本和视频没差,他把链接发了到了评论(事实上他在好几个疫情可视化的视频下面都留了链接,我真的是服了),我顺着网站找到了他的博客,下面有人留言说可不可以要源码,他说先加vx.我兴冲冲的去加了,但是向他要的时候他说"**有偿!!!!**"

我理解但不认同这种做法,也幸亏他拒绝了我,浇灭了我想偷懒的心

在网上也很少搜到基于java实现的疫情可视化项目(其实主要是爬虫数据解析部分不明朗,大家好像都去用python了)

基于以上原因,还有我一路对白嫖过来的视频和资源提供者感激,我当时就决定自己把这种项目写出来,发到网上,供像我一样的萌新参考,学习.

第一次写,可能整理的不太详细,具体的就参照源码吧,. 基本的框架差不多就是这样,网站我会慢慢修改,逐渐脱离之前的布局(u1s1,真的有点丑),与此同时这也是一个融合和学习的好途径.

**开源是一种精神**