**1.爬虫代码**

1. from pyquery import PyQuery as pq
2. from selenium import webdriver
3. from time import sleep
4. import csv
5. import random
6. import multiprocessing
7. BASE\_url = 'https://www.zhipin.com/web/geek/job?query=%E5%B5%8C%E5%85%A5%E5%BC%8F&city=100010000'
8. TOTAL\_PAGE = 10
9. driver = webdriver.Edge()
10. def scrape\_url(page):
11. url\_join = f'&page={page}'
12. url = BASE\_url + url\_join
13. yield url
14. def parse\_html(url,num):
15. driver.get(url)
16. sleep(num)
17. html = driver.page\_source
18. doc = pq(html)
19. items = doc('.job-card-wrapper').items()
20. with open('boss\_5.csv', 'a', encoding='utf-8', newline='') as csvfile:
21. writer = csv.writer(csvfile)
22. for i, item in enumerate(items):
23. row = []
24. row.append(i+1)
25. row.append(item.find('.job-name').text())
26. row.append(item.find('.job-area').text())
27. row.append(item.find('.salary').text())
28. row.append(item.find('.degree').text())
29. row.append(item.find('.experien').text())
30. row.append(item.find('.company-name').text())
31. company\_tags = item.find('.company-tag-list li')
32. if len(company\_tags) == 3:
33. row.append(company\_tags.eq(0).text())
34. row.append(company\_tags.eq(1).text())
35. row.append(company\_tags.eq(2).text())
36. elif len(company\_tags) == 2:
37. row.append(company\_tags.eq(0).text())
38. row.append("未融资")
39. row.append(company\_tags.eq(1).text())
40. else:
41. row.append("")
42. row.append("")
43. row.append("")
44. writer.writerow(row)
45. print('scrape successfully!'+url)
46. def main(page):
47. urls = scrape\_url(page)
48. for url in urls:
49. parse\_html(url,8)
50. if \_\_name\_\_ == '\_\_main\_\_':
51. urls = scrape\_url(1)
52. for url in urls:
53. parse\_html(url, 90)
54. for i in range(1, TOTAL\_PAGE + 1):
55. main(i)
56. driver.quit()

**2.CSV文件中的数据转换并实时推送到Apache Kafka消息队列中**

1. */\*\**
2. \* 1.csvFileReader.java：读取CSV文件并提供数据行
3. \*/
4. package com.zq;
5. import com.csvreader.CsvReader;
6. import java.io.IOException;
7. import java.nio.charset.Charset;
8. import java.util.NoSuchElementException;
9. import java.util.function.Supplier;
10. public class csvFileReader implements Supplier<EIcsv> {
11. private final String filePath;
12. private CsvReader csvReader;
13. public csvFileReader(String filePath) throws IOException {
14. this.filePath = filePath;
15. try {
16. csvReader = new CsvReader(filePath, ',', Charset.forName("GBK"));
17. csvReader.readHeaders();
18. } catch (IOException e) {
19. throw new IOException("Error reading TaxiRecords from file: " + filePath, e);
20. }
21. }
22. @Override
23. public EIcsv get() {
24. EIcsv eIcsv = null;
25. try{
26. if(csvReader.readRecord()) {
27. csvReader.getRawRecord();
28. eIcsv = new EIcsv(
29. Long.valueOf(csvReader.get(0)),
30. csvReader.get(1),
31. csvReader.get(2),
32. Long.valueOf(csvReader.get(3)),
33. csvReader.get(4),
34. csvReader.get(5),
35. csvReader.get(6),
36. csvReader.get(7),
37. csvReader.get(8),
38. csvReader.get(9),
39. csvReader.get(10),
40. Long.valueOf(csvReader.get(11)));
41. }
42. } catch (IOException e) {
43. throw new NoSuchElementException("IOException from " + filePath);
44. }
45. if (null==eIcsv) {
46. throw new NoSuchElementException("All records read from " + filePath);
47. }
48. return eIcsv;
49. }
50. }
51. */\*\**
52. \*2.EIcsv.java：处理和转换从csvFileReader读取的CSV数据
53. \*/
54. package com.zq;
55. import com.fasterxml.jackson.annotation.JsonFormat;
56. public class EIcsv {
57. @JsonFormat
58. private long num;
59. @JsonFormat
60. private String work\_name;
61. @JsonFormat
62. private String location;
63. @JsonFormat
64. private long salary;
65. @JsonFormat
66. private String work\_year;
67. @JsonFormat
68. private String educate;
69. @JsonFormat
70. private String company\_name;
71. @JsonFormat
72. private String company\_type;
73. @JsonFormat
74. private String financing\_conditions;
75. @JsonFormat
76. private String company\_size;
77. @JsonFormat
78. private String city;
79. @JsonFormat
80. private long nums;
81. public EIcsv() {
82. }
83. public EIcsv(long num, String work\_name, String location, long salary, String work\_year, String educate,
84. String company\_name, String company\_type, String financing\_conditions, String company\_size,
85. String city, long nums) {
86. this.num = num;
87. this.work\_name = work\_name;
88. this.location = location;
89. this.salary = salary;
90. this.work\_year = work\_year;
91. this.educate = educate;
92. this.company\_name = company\_name;
93. this.company\_type = company\_type;
94. this.financing\_conditions = financing\_conditions;
95. this.company\_size = company\_size;
96. this.city = city;
97. this.nums = nums;
98. }
99. }
100. */\*\**
101. \*3.JsonSerializer.java：将数据对象序列化为JSON格式以发送至Kafka
102. \*/
103. package com.zq;
104. import com.fasterxml.jackson.core.JsonProcessingException;
105. import com.fasterxml.jackson.databind.ObjectMapper;
106. public class JsonSerializer<T> {
107. private final ObjectMapper jsonMapper = new ObjectMapper();
108. public String toJSONString(T r) {
109. try {
110. return jsonMapper.writeValueAsString(r);
111. } catch (JsonProcessingException e) {
112. throw new IllegalArgumentException("Could not serialize record: " + r, e);
113. }
114. }
115. public byte[] toJSONBytes(T r){
116. try {
117. return jsonMapper.writeValueAsBytes(r);
118. } catch (JsonProcessingException e) {
119. throw new IllegalArgumentException("Could not serialize record: " + r, e);
120. }
121. }
122. }
123. */\*\**
124. \* 4.KafkaProducer.java：配置和管理与Kafka集群的数据发布交互
125. \*/
126. package com.zq;
127. import org.apache.kafka.clients.producer.ProducerConfig;
128. import org.apache.kafka.clients.producer.ProducerRecord;
129. import org.apache.kafka.common.serialization.ByteArraySerializer;
130. import java.nio.charset.StandardCharsets;
131. import java.util.Properties;
132. import java.util.function.Consumer;
133. public class KafkaProducer implements Consumer<EIcsv> {
134. private final String topic;
135. private final org.apache.kafka.clients.producer.KafkaProducer<byte[], byte[]> producer;
136. private final JsonSerializer<EIcsv> serializer;
137. public KafkaProducer(String kafkaTopic, String kafkaBrokers) {
138. this.topic = kafkaTopic;
139. this.producer = new org.apache.kafka.clients.producer.KafkaProducer<>(createKafkaProperties(kafkaBrokers));
140. this.serializer = new JsonSerializer<>();
141. }
142. @Override
143. public void accept(EIcsv record) {
144. *// 将对象序列化成byte数组*
145. byte[] data = serializer.toJSONBytes(record);
146. *// 封装*
147. ProducerRecord<byte[], byte[]> kafkaRecord = new ProducerRecord<>(topic, data);
148. *// 发送*
149. producer.send(kafkaRecord);
150. *// 通过sleep控制消息的速度，请依据自身kafka配置以及flink服务器配置来调整*
151. try {
152. Thread.sleep(500);
153. }catch(InterruptedException e){
154. e.printStackTrace();
155. }
156. }
157. */\*\**
158. \* kafka配置
159. \* @param brokers The brokers to connect to.
160. \* @return A Kafka producer configuration.
161. \*/
162. private static Properties createKafkaProperties(String brokers) {
163. Properties kafkaProps = new Properties();
164. kafkaProps.put(ProducerConfig.BOOTSTRAP\_SERVERS\_CONFIG, brokers);
165. kafkaProps.put(ProducerConfig.KEY\_SERIALIZER\_CLASS\_CONFIG, ByteArraySerializer.class.getCanonicalName());
166. kafkaProps.put(ProducerConfig.VALUE\_SERIALIZER\_CLASS\_CONFIG, ByteArraySerializer.class.getCanonicalName());
167. return kafkaProps;
168. }
169. }
170. */\*\**
171. \* 5.SendMessageApplication.java：整合各组件，实现CSV数据到Kafka主题的迁移流程
172. \*/
173. package com.zq;
174. import java.util.stream.Stream;
175. public class SendMessageApplication {
176. public static void main(String[] args) throws Exception {
177. *// 文件地址*
178. String filePath = "C:\\Users\\ZQ\\Desktop\\EI\_2024-02-21.csv";
179. *// kafka topic*
180. String topic = "events";
181. *// kafka borker地址*
182. String broker = "192.168.10.102:9092";
183. Stream.generate(new csvFileReader(filePath))
184. .sequential()
185. .forEachOrdered(new KafkaProducer(topic, broker));
186. }
187. }

**3.通过SparkStreaming实时处理并分析就业信息流数据**

1. */\**
2. \* (1)DwdDauApp.scala：实现每日活跃用户（DAU）的计算逻辑并写入数据仓库明细层（DWD）的应用模块
3. \*/
4. package com.zq.realtime.app
5. import java.text.SimpleDateFormat
6. import java.time.{LocalDate, Period}
7. import java.{lang, util}
8. import java.util.Date
9. import com.alibaba.fastjson.{JSON, JSONObject}
10. import com.zq.realtime.bean.{DauInfo, PageLog}
11. import com.zq.realtime.util.{MyBeanUtils, MyEsUtils, MyKafkaUtils, MyOffsetsUtils, MyRedisUtils}
12. import org.apache.kafka.clients.consumer.ConsumerRecord
13. import org.apache.kafka.common.TopicPartition
14. import org.apache.spark.SparkConf
15. import org.apache.spark.streaming.dstream.{DStream, InputDStream}
16. import org.apache.spark.streaming.kafka010.{HasOffsetRanges, OffsetRange}
17. import org.apache.spark.streaming.{Seconds, StreamingContext}
18. import redis.clients.jedis.{Jedis, Pipeline}
19. import scala.collection.mutable.ListBuffer
20. object DwdDauApp {
21. def main(args: Array[String]): Unit = {
22. *//0.还原状态*
23. revertState()
24. *//1. 准备实时环境*
25. val sparkConf: SparkConf = new SparkConf().setAppName("dwd\_dau\_app").setMaster("local[4]")
26. val ssc: StreamingContext = new StreamingContext(sparkConf,Seconds(5))
27. *//2. 从Redis中读取offset*
28. val topicName : String = "DWD\_EVENTS\_TOPIC"
29. val groupId : String = "DWD\_DAU\_GROUP"
30. val offsets: Map[TopicPartition, Long] = MyOffsetsUtils.readOffset(topicName, groupId)
31. *//3. 从Kafka中消费数据*
32. var kafkaDStream: InputDStream[ConsumerRecord[String, String]] = null
33. if(offsets != null && offsets.nonEmpty){
34. kafkaDStream = MyKafkaUtils.getKafkaDStream(ssc,topicName,groupId,offsets)
35. }else {
36. kafkaDStream = MyKafkaUtils.getKafkaDStream(ssc,topicName,groupId)
37. }
38. *//4. 提取offset结束点*
39. var offsetRanges: Array[OffsetRange] = null
40. val offsetRangesDStream: DStream[ConsumerRecord[String, String]] = kafkaDStream.transform(
41. rdd => {
42. offsetRanges = rdd.asInstanceOf[HasOffsetRanges].offsetRanges
43. rdd
44. }
45. )
46. *//5. 处理数据*
47. *// 5.1 转换结构*
48. val pageLogDStream: DStream[PageLog] = offsetRangesDStream.map(
49. consumerRecord => {
50. val value: String = consumerRecord.value()
51. val pageLog: PageLog = JSON.parseObject(value, classOf[PageLog])
52. pageLog
53. }
54. )
55. pageLogDStream.foreachRDD(
56. rdd => {
57. rdd.foreachPartition(
58. dauInfoIter => {
59. val docs: List[(String, PageLog)] =
60. dauInfoIter.map( dauInfo=> (dauInfo.num , dauInfo)).toList
61. if(docs.size >  0 ){
62. *// 索引名*
63. val indexName : String = s"ei\_dau\_info"
64. *//写入到ES中*
65. MyEsUtils.bulkSave(indexName , docs)
66. }
67. }
68. )
69. *//提交offset*
70. MyOffsetsUtils.saveOffset(topicName, groupId , offsetRanges)
71. }
72. )
73. ssc.start()
74. ssc.awaitTermination()
75. }
76. def revertState(): Unit ={
77. *//从ES中查询到所有的mid*
78. val date: LocalDate = LocalDate.now()
79. val indexName : String = s"ei\_dau\_info"
80. val fieldName : String = "num"
81. val nums: List[ String ] = MyEsUtils.searchField(indexName , fieldName)
82. *//删除redis中记录的状态（所有的mid）*
83. val jedis: Jedis = MyRedisUtils.getJedisFromPool()
84. val redisDauKey : String = s"DAU:$date"
85. jedis.del(redisDauKey)
86. *//将从ES中查询到的mid覆盖到Redis中*
87. if(nums != null && nums.size > 0 ){
88. */\*for (mid <- mids) {*
89. jedis.sadd(redisDauKey , mid )
90. }\*/
91. val pipeline: Pipeline = jedis.pipelined()
92. for (num <- nums) {
93. pipeline.sadd(redisDauKey , num )  *//不会直接到redis执行*
94. }
95. pipeline.sync()  *// 到redis执行*
96. }
97. jedis.close()
98. }
99. }
100. */\**
101. \* (2)OdsBaseLogApp.scala：负责处理原始日志数据，进行基础转换与加载至ODS层的应用程序定义
102. \*/
103. package com.zq.realtime.app
104. import com.alibaba.fastjson.serializer.SerializeConfig
105. import com.alibaba.fastjson.{JSON, JSONArray, JSONObject}
106. import com.zq.realtime.bean.PageLog
107. import com.zq.realtime.util.{MyKafkaUtils, MyOffsetsUtils}
108. import org.apache.kafka.clients.consumer.ConsumerRecord
109. import org.apache.kafka.common.TopicPartition
110. import org.apache.spark.SparkConf
111. import org.apache.spark.streaming.dstream.{DStream, InputDStream}
112. import org.apache.spark.streaming.kafka010.{HasOffsetRanges, OffsetRange}
113. import org.apache.spark.streaming.{Seconds, StreamingContext}
114. object OdsBaseLogApp {
115. def main(args: Array[String]): Unit = {
116. *//1. 准备实时环境*
117. *//TODO 注意并行度与Kafka中topic的分区个数的对应关系*
118. val sparkConf: SparkConf = new SparkConf().setAppName("ods\_events\_app").setMaster("local[4]")
119. val ssc: StreamingContext = new StreamingContext(sparkConf, Seconds(5))
120. *//2. 从kafka中消费数据*
121. val topicName: String = "events" *//对应生成器配置中的主题名*
122. val groupId: String = "ODS\_EVENTS\_GROUP"
123. *//TODO  从Redis中读取offset， 指定offset进行消费*
124. val offsets: Map[TopicPartition, Long] = MyOffsetsUtils.readOffset(topicName, groupId)
125. var kafkaDStream: InputDStream[ConsumerRecord[String, String]] = null
126. if (offsets != null && offsets.nonEmpty) {
127. *//指定offset进行消费*
128. kafkaDStream =
129. MyKafkaUtils.getKafkaDStream(ssc, topicName, groupId, offsets)
130. } else {
131. *//默认offset进行消费*
132. kafkaDStream =
133. MyKafkaUtils.getKafkaDStream(ssc, topicName, groupId)
134. }
135. *// TODO 补充: 从当前消费到的数据中提取offsets , 不对流中的数据做任何处理.*
136. var offsetRanges: Array[OffsetRange] = null
137. val offsetRangesDStream: DStream[ConsumerRecord[String, String]] = kafkaDStream.transform(
138. rdd => {
139. offsetRanges = rdd.asInstanceOf[HasOffsetRanges].offsetRanges *// 在哪里执行? driver*
140. rdd
141. }
142. )
143. val jsonObjDStream: DStream[JSONObject] = offsetRangesDStream.map(
144. consumerRecord => {
145. *//获取ConsumerRecord中的value,value就是日志数据*
146. val log: String = consumerRecord.value()
147. *//转换成Json对象*
148. val jsonObj: JSONObject = JSON.parseObject(log)
149. *//返回*
150. jsonObj
151. }
152. )
153. *//     jsonObjDStream.print(1000)*
154. *//3.2 分流*
155. val DWD\_EVENTS\_TOPIC: String = "DWD\_EVENTS\_TOPIC"
156. jsonObjDStream.foreachRDD(
157. rdd => {
158. *// 按分区执行*
159. rdd.foreachPartition(
160. jsonObjIter => {
161. for (jsonObj <- jsonObjIter) {
162. *//分流过程*
163. val num: String = jsonObj.getString("num")
164. val work\_name: String = jsonObj.getString("work\_name")
165. val location: String = jsonObj.getString("location")
166. val salary: String = jsonObj.getString("salary")
167. val work\_year: String = jsonObj.getString("work\_year")
168. val educate: String = jsonObj.getString("educate")
169. val company\_name: String = jsonObj.getString("company\_name")
170. val company\_type: String = jsonObj.getString("company\_type")
171. val financing\_conditions: String = jsonObj.getString("financing\_conditions")
172. val company\_size: String = jsonObj.getString("company\_size")
173. val city: String = jsonObj.getString("city")
174. val nums: String = jsonObj.getString("nums")
175. *//封装成PageLog*
176. var pageLog =
177. PageLog(num, work\_name, location, salary, work\_year, educate, company\_name, company\_type, financing\_conditions, company\_size, city, nums)
178. *//发送到DWD\_PAGE\_LOG\_TOPIC*
179. MyKafkaUtils.send(DWD\_EVENTS\_TOPIC, JSON.toJSONString(pageLog, new SerializeConfig(true)))
180. }
181. *// foreachPartition里面:  Executor段执行， 每批次每分区执行一次*
182. *//刷写Kafka*
183. MyKafkaUtils.flush()
184. }
185. )
186. MyOffsetsUtils.saveOffset(topicName, groupId, offsetRanges)
187. }
188. )
189. ssc.start()
190. ssc.awaitTermination()
191. }
192. }
193. */\**
194. \* (3)DauInfo.scala：定义存储活跃用户信息的数据模型或案例类
195. \*/
196. package com.zq.realtime.bean
197. case class DauInfo(
198. var num:String,
199. var work\_name:String,
200. var location:String,
201. var salary:String,
202. var work\_year:String,
203. var educate:String,
204. var company\_name:String,
205. var company\_type:String,
206. var financing\_conditions:String,
207. var company\_size:String,
208. var city:String,
209. var nums:String,
210. ) {
211. def this(){
212. this(null,null,null,null,null,null,null,null,null,null ,null ,null)
213. }
214. }
215. */\**
216. \* (4)PageLog.scala：描述页面访问日志结构及相关属性的实体类
217. \*/
218. package com.zq.realtime.bean
219. case class PageLog(
220. num :String,
221. work\_name:String,
222. location:String,
223. salary:String,
224. work\_year:String,
225. educate:String,
226. company\_name:String,
227. company\_type:String,
228. financing\_conditions : String ,
229. company\_size:String ,
230. city:String,
231. nums:String,
232. ) {
233. }
234. */\**
235. \* (5)MyBeanUtils.scala：提供对象属性操作的工具方法集合
236. \*/
237. package com.zq.realtime.util
238. import java.lang.reflect.{Field, Method, Modifier}
239. import com.zq.realtime.bean.{DauInfo, PageLog}
240. import scala.util.control.Breaks
241. */\*\**
242. \* 实现对象属性拷贝.
243. \*/
244. object MyBeanUtils {
245. def main(args: Array[String]): Unit = {
246. val pageLog: PageLog =
247. PageLog(null ,null ,null , null ,null ,null ,null ,null ,null ,null ,null ,null)
248. val dauInfo: DauInfo = new DauInfo()
249. println("拷贝前: " + dauInfo)
250. copyProperties(pageLog,dauInfo)
251. println("拷贝后: " + dauInfo)
252. }
253. */\*\**
254. \* 将srcObj中属性的值拷贝到destObj对应的属性上.
255. \*/
256. def copyProperties(srcObj : AnyRef , destObj: AnyRef): Unit ={
257. if(srcObj == null || destObj == null ){
258. return
259. }
260. *//获取到srcObj中所有的属性*
261. val srcFields: Array[Field] = srcObj.getClass.getDeclaredFields
262. *//处理每个属性的拷贝*
263. for (srcField <- srcFields) {
264. Breaks.breakable{
265. *//get / set*
266. *// Scala会自动为类中的属性提供get、 set方法*
267. *// get : fieldname()*
268. *// set : fieldname\_$eq(参数类型)*
269. *//getMethodName*
270. var getMethodName : String = srcField.getName
271. *//setMethodName*
272. var setMethodName : String = srcField.getName+"\_$eq"
273. *//从srcObj中获取get方法对象，*
274. val getMethod: Method = srcObj.getClass.getDeclaredMethod(getMethodName)
275. *//从destObj中获取set方法对象*
276. *// String name;*
277. *// getName()*
278. *// setName(String name ){ this.name = name }*
279. val setMethod: Method =
280. try{
281. destObj.getClass.getDeclaredMethod(setMethodName, srcField.getType)
282. }catch{
283. *// NoSuchMethodException*
284. case ex : Exception =>  Breaks.break()
285. }
286. *//忽略val属性*
287. val destField: Field = destObj.getClass.getDeclaredField(srcField.getName)
288. if(destField.getModifiers.equals(Modifier.FINAL)){
289. Breaks.break()
290. }
291. *//调用get方法获取到srcObj属性的值， 再调用set方法将获取到的属性值赋值给destObj的属性*
292. setMethod.invoke(destObj, getMethod.invoke(srcObj))
293. }
294. }
295. }
296. }
297. */\**
298. \* (6)MyConfig.scala：封装项目配置参数和全局设置的模块
299. \*/
300. package com.zq.realtime.util
301. */\*\**
302. \* 配置类
303. \*/
304. object MyConfig {
305. val KAFKA\_BOOTSTRAP\_SERVERS : String = "kafka.bootstrap-servers"
306. val REDIS\_HOST : String = "redis.host"
307. val REDIS\_PORT : String = "redis.port"
308. val ES\_HOST : String = "es.host"
309. val ES\_PORT : String = "es.port"
310. }
311. */\**
312. \* (7)MyEsUtils.scala：集成Elasticsearch的工具类，包含索引、查询等相关操作
313. \*/
314. package com.zq.realtime.util
315. import java.util
316. import com.alibaba.fastjson.JSON
317. import com.alibaba.fastjson.serializer.SerializeConfig
318. import org.apache.http.HttpHost
319. import org.elasticsearch.action.bulk.BulkRequest
320. import org.elasticsearch.action.index.IndexRequest
321. import org.elasticsearch.action.search.{SearchRequest, SearchResponse}
322. import org.elasticsearch.client.indices.GetIndexRequest
323. import org.elasticsearch.client.{RequestOptions, RestClient, RestClientBuilder, RestHighLevelClient}
324. import org.elasticsearch.common.xcontent.XContentType
325. import org.elasticsearch.search.SearchHit
326. import org.elasticsearch.search.builder.SearchSourceBuilder
327. import scala.collection.mutable.ListBuffer
328. object MyEsUtils {
329. */\*\*客户端对象\*/*
330. val esClient : RestHighLevelClient = build()
331. */\*\*创建ES客户端对象\*/*
332. def build(): RestHighLevelClient ={
333. val host: String = MyPropsUtils(MyConfig.ES\_HOST)
334. val port: String = MyPropsUtils(MyConfig.ES\_PORT)
335. val restClientBuilder: RestClientBuilder = RestClient.builder(new HttpHost(host,port.toInt))
336. val client: RestHighLevelClient = new RestHighLevelClient(restClientBuilder)
337. client
338. }
339. */\*\*关闭ES对象\*/*
340. def close(): Unit ={
341. if(esClient != null ) esClient.close()
342. }
343. def bulkSave(indexName : String , docs :  List[(String , AnyRef)]  ): Unit ={
344. val bulkRequest: BulkRequest = new BulkRequest(indexName)
345. for ((docId, docObj) <- docs) {
346. val indexRequest: IndexRequest = new IndexRequest()
347. val dataJson: String = JSON.toJSONString(docObj , new SerializeConfig(true))
348. indexRequest.source(dataJson , XContentType.JSON)
349. indexRequest.id(docId)
350. bulkRequest.add(indexRequest)
351. }
352. esClient.bulk(bulkRequest ,RequestOptions.DEFAULT)
353. }
354. */\*\**
355. \* 查询指定的字段
356. \*/
357. def searchField(indexName: String, fieldName: String): List[String] = {
358. *//判断索引是否存在*
359. val getIndexRequest: GetIndexRequest = new GetIndexRequest(indexName)
360. val isExists: Boolean =
361. esClient.indices().exists(getIndexRequest,RequestOptions.DEFAULT)
362. if(!isExists){
363. return null
364. }
365. *//正常从ES中提取指定的字段*
366. val mids: ListBuffer[String] = ListBuffer[String]()
367. val searchRequest: SearchRequest = new SearchRequest(indexName)
368. val searchSourceBuilder: SearchSourceBuilder = new SearchSourceBuilder()
369. searchSourceBuilder.fetchSource(fieldName,null).size(100000)
370. searchRequest.source(searchSourceBuilder)
371. val searchResponse: SearchResponse =
372. esClient.search(searchRequest , RequestOptions.DEFAULT)
373. val hits: Array[SearchHit] = searchResponse.getHits.getHits
374. for (hit <- hits) {
375. val sourceMap: util.Map[String, AnyRef] = hit.getSourceAsMap
376. val mid: String = sourceMap.get(fieldName).toString
377. mids.append(mid)
378. }
379. mids.toList
380. }
381. def main(args: Array[String]): Unit = {
382. println(searchField("gmall\_dau\_info\_1018\_2022-03-28", "mid"))
383. }
384. }
385. */\**
386. \* (8)MyKafkaUtils.scala：实现与Apache Kafka连接和消息处理的实用工具类
387. \*/
388. package com.zq.realtime.util
389. import java.util
390. import org.apache.kafka.clients.consumer
391. import org.apache.kafka.clients.consumer.{ConsumerConfig, ConsumerRecord}
392. import org.apache.kafka.clients.producer.{KafkaProducer, ProducerConfig, ProducerRecord}
393. import org.apache.kafka.common.TopicPartition
394. import org.apache.spark.streaming.StreamingContext
395. import org.apache.spark.streaming.dstream.InputDStream
396. import org.apache.spark.streaming.kafka010.{ConsumerStrategies, KafkaUtils, LocationStrategies}
397. import scala.collection.mutable
398. private val consumerConfigs: mutable.Map[String, Object] = mutable.Map[String, Object](
399. *// kafka集群位置*
400. *//ConsumerConfig.BOOTSTRAP\_SERVERS\_CONFIG -> "hadoop102:9092,hadoop103:9092,hadoop104:9092",*
401. *//ConsumerConfig.BOOTSTRAP\_SERVERS\_CONFIG -> MyPropsUtils("kafka.bootstrap-servers"),*
402. ConsumerConfig.BOOTSTRAP\_SERVERS\_CONFIG -> MyPropsUtils(MyConfig.KAFKA\_BOOTSTRAP\_SERVERS),
403. *// kv反序列化器*
404. ConsumerConfig.KEY\_DESERIALIZER\_CLASS\_CONFIG -> "org.apache.kafka.common.serialization.StringDeserializer",
405. ConsumerConfig.VALUE\_DESERIALIZER\_CLASS\_CONFIG -> "org.apache.kafka.common.serialization.StringDeserializer",
406. *// groupId*
407. *// offset提交  自动 手动*
408. ConsumerConfig.ENABLE\_AUTO\_COMMIT\_CONFIG -> "true",
409. *//自动提交的时间间隔*
410. *//ConsumerConfig.AUTO\_COMMIT\_INTERVAL\_MS\_CONFIG*
411. *// offset重置  "latest"  "earliest"*
412. ConsumerConfig.AUTO\_OFFSET\_RESET\_CONFIG -> "latest"
413. *// .....*
414. )
415. */\*\**
416. \* 基于SparkStreaming消费 ,获取到KafkaDStream , 使用默认的offset
417. \*/
418. def getKafkaDStream(ssc: StreamingContext, topic: String, groupId: String) = {
419. consumerConfigs.put(ConsumerConfig.GROUP\_ID\_CONFIG, groupId)
420. val kafkaDStream: InputDStream[ConsumerRecord[String, String]] = KafkaUtils.createDirectStream(ssc,
421. LocationStrategies.PreferConsistent,
422. ConsumerStrategies.Subscribe[String, String](Array(topic), consumerConfigs))
423. kafkaDStream
424. }
425. */\*\**
426. \* 基于SparkStreaming消费 ,获取到KafkaDStream , 使用指定的offset
427. \*/
428. def getKafkaDStream(ssc: StreamingContext, topic: String, groupId: String, offsets: Map[TopicPartition, Long]) = {
429. consumerConfigs.put(ConsumerConfig.GROUP\_ID\_CONFIG, groupId)
430. val kafkaDStream: InputDStream[ConsumerRecord[String, String]] = KafkaUtils.createDirectStream(ssc,
431. LocationStrategies.PreferConsistent,
432. ConsumerStrategies.Subscribe[String, String](Array(topic), consumerConfigs, offsets))
433. kafkaDStream
434. }
435. */\*\**
436. \* 生产者对象
437. \*/
438. val producer: KafkaProducer[String, String] = createProducer()
439. */\*\**
440. \* 创建生产者对象
441. \*/
442. def createProducer(): KafkaProducer[String, String] = {
443. val producerConfigs: util.HashMap[String, AnyRef] = new util.HashMap[String, AnyRef]
444. *//生产者配置类 ProducerConfig*
445. *//kafka集群位置*
446. producerConfigs.put(ProducerConfig.BOOTSTRAP\_SERVERS\_CONFIG, MyPropsUtils(MyConfig.KAFKA\_BOOTSTRAP\_SERVERS))
447. *//kv序列化器*
448. producerConfigs.put(ProducerConfig.KEY\_SERIALIZER\_CLASS\_CONFIG, "org.apache.kafka.common.serialization.StringSerializer")
449. producerConfigs.put(ProducerConfig.VALUE\_SERIALIZER\_CLASS\_CONFIG, "org.apache.kafka.common.serialization.StringSerializer")
450. *//acks*
451. producerConfigs.put(ProducerConfig.ACKS\_CONFIG, "all")
452. *//batch.size  16kb*
453. *//linger.ms   0*
454. *//retries*
455. *//幂等配置*
456. producerConfigs.put(ProducerConfig.ENABLE\_IDEMPOTENCE\_CONFIG, "true")
457. val producer: KafkaProducer[String, String] = new KafkaProducer[String, String](producerConfigs)
458. producer
459. }
460. */\*\**
461. \* 生产（按照默认的黏性分区策略）
462. \*/
463. def send(topic: String, msg: String): Unit = {
464. producer.send(new ProducerRecord[String, String](topic, msg))
465. }
466. */\*\**
467. \* 生产（按照key进行分区）
468. \*/
469. def send(topic: String, key: String, msg: String): Unit = {
470. producer.send(new ProducerRecord[String, String](topic, key, msg))
471. }
472. */\*\**
473. \* 关闭生产者对象
474. \*/
475. def close(): Unit = {
476. if (producer != null) producer.close()
477. }
478. def flush(): Unit = {
479. producer.flush()
480. }
481. }
482. */\**
483. \* (9)MyOffsetsUtils.scala：管理Kafka消费者偏移量的工具类，用于消费进度跟踪
484. \*/
485. package com.zq.realtime.util
486. import java.util
487. import org.apache.kafka.common.TopicPartition
488. import org.apache.spark.streaming.kafka010.OffsetRange
489. import redis.clients.jedis.Jedis
490. import scala.collection.mutable
491. def saveOffset( topic : String , groupId : String , offsetRanges: Array[OffsetRange]  ): Unit ={
492. if(offsetRanges != null && offsetRanges.length > 0){
493. val offsets: util.HashMap[String, String] = new util.HashMap[String,String]()
494. for (offsetRange <- offsetRanges) {
495. val partition: Int = offsetRange.partition
496. val endOffset: Long = offsetRange.untilOffset
497. offsets.put(partition.toString,endOffset.toString)
498. }
499. println("提交offset: " + offsets)
500. *//往redis中存*
501. val jedis: Jedis = MyRedisUtils.getJedisFromPool()
502. val redisKey : String = s"offsets:$topic:$groupId"
503. jedis.hset(redisKey , offsets)
504. jedis.close()
505. }
506. }
507. */\*\**
508. \* 从Redis中读取存储的offset
509. \*
510. \* 问题:
511. \*    如何让SparkStreaming通过指定的offset进行消费?
512. \*
513. \*    SparkStreaming要求的offset的格式是什么?
514. \*                Map[TopicPartition ,Long  ]
515. \*/
516. def readOffset(topic: String, groupId : String ):  Map[TopicPartition ,Long  ] ={
517. val jedis: Jedis = MyRedisUtils.getJedisFromPool()
518. val redisKey : String = s"offsets:$topic:$groupId"
519. val offsets: util.Map[String, String] = jedis.hgetAll(redisKey)
520. println("读取到offset: " + offsets)
521. val results: mutable.Map[TopicPartition, Long] = mutable.Map[TopicPartition ,Long ]()
522. *//将java的map转换成scala的map进行迭代*
523. import scala.collection.JavaConverters.\_
524. for ((partition,offset) <- offsets.asScala) {
525. val tp: TopicPartition = new TopicPartition(topic,partition.toInt)
526. results.put(tp, offset.toLong)
527. }
528. jedis.close()
529. results.toMap
530. }
531. }
532. */\**
533. \* (10)MyPropsUtils.scala：提供读取和操作Properties配置文件的方法集
534. \*/
535. package com.zq.realtime.util
536. import java.util.ResourceBundle
537. */\*\**
538. \* 配置文件解析类
539. \*/
540. object MyPropsUtils {
541. private val bundle: ResourceBundle = ResourceBundle.getBundle("config")
542. def apply(propsKey: String): String = {
543. bundle.getString(propsKey)
544. }
545. def main(args: Array[String]): Unit = {
546. println(MyPropsUtils("kafka.bootstrap-servers"))
547. }
548. }
549. */\**
550. \* (11)MyPropsUtils.scala：提供读取和操作Properties配置文件的方法集
551. \*/
552. package com.zq.realtime.util
553. import redis.clients.jedis.{Jedis, JedisPool, JedisPoolConfig}
554. */\*\**
555. \* Redis工具类，用于获取Jedis连接， 操作Redis
556. \*/
557. object MyRedisUtils {
558. var jedisPool : JedisPool = null
559. def getJedisFromPool(): Jedis ={
560. if(jedisPool == null ){
561. *//创建连接池对象*
562. *//连接池配置*
563. val jedisPoolConfig = new JedisPoolConfig()
564. jedisPoolConfig.setMaxTotal(100) *//最大连接数*
565. jedisPoolConfig.setMaxIdle(20) *//最大空闲*
566. jedisPoolConfig.setMinIdle(20) *//最小空闲*
567. jedisPoolConfig.setBlockWhenExhausted(true) *//忙碌时是否等待*
568. jedisPoolConfig.setMaxWaitMillis(5000) *//忙碌时等待时长 毫秒*
569. jedisPoolConfig.setTestOnBorrow(true) *//每次获得连接的进行测试*
570. val host: String = MyPropsUtils(MyConfig.REDIS\_HOST)
571. val port: String = MyPropsUtils(MyConfig.REDIS\_PORT)
572. jedisPool = new JedisPool(jedisPoolConfig,host,port.toInt)
573. }
574. jedisPool.getResource
575. }
576. }

**4.前后端交互API（提供实时数据推送服务，使后端能即时向前端更新信息）**

1. */\*\**
2. \* (1)CompanyType.java: 定义公司类型的枚举或实体类，用于区分不同类型的公司
3. \*/
4. package com.zq.ei.publisherrealtime.bean;
5. public class CompanyType {
6. private String name;
7. private int value;
8. *// 构造方法*
9. public CompanyType(String name, int value) {
10. this.name = name;
11. this.value = value;
12. }
13. *// Getter和Setter方法*
14. public String getName() {
15. return name;
16. }
17. public void setName(String name) {
18. this.name = name;
19. }
20. public int getValue() {
21. return value;
22. }
23. public void setValue(int value) {
24. this.value = value;
25. }
26. }
27. */\*\**
28. \* (2)PublisherController.java: 提供RESTful API接口，处理与实时发布相关的HTTP请求和响应
29. \*/
30. package com.zq.ei.publisherrealtime.controller;
31. import com.zq.ei.publisherrealtime.service.CompanyTypeService;
32. import com.zq.ei.publisherrealtime.service.PublisherService;
33. import org.springframework.beans.factory.annotation.Autowired;
34. import org.springframework.stereotype.Controller;
35. import org.springframework.web.bind.annotation.GetMapping;
36. import org.springframework.web.bind.annotation.RequestParam;
37. import org.springframework.web.bind.annotation.RestController;
38. import java.util.Map;
39. */\*\**
40. \* 控制层
41. \*/
42. @RestController
43. public class PublisherController {
44. @Autowired
45. PublisherService publisherService;
46. @Autowired
47. private CompanyTypeService companyTypeService;
48. *// bar\_horizontal接口*
49. @GetMapping("dauRealtime")
50. public String dauRealtime(){
51. String results = publisherService.doDauRealtime();
52. return results;
53. }
54. *// bar\_markline接口*
55. @GetMapping("dauRealtime2")
56. public String getCompanyTypes() throws Exception {
57. return companyTypeService.getCompanyTypesAsJson();
58. }
59. *// bar\_ROA接口*
60. @GetMapping("dauRealtime3")
61. public String dauRealtime3(){
62. String results = publisherService.doDauRealtime3();
63. return results;
64. }
65. }
66. */\*\**
67. \* (3)PublisherMapper.java: 实现数据模型对象（POJO）与数据库表结构之间的映射转换
68. \*/
69. package com.zq.ei.publisherrealtime.mapper;
70. import java.util.Map;
71. public interface PublisherMapper {
72. String searchDau();
73. String searchDau3();
74. }
75. */\*\**
76. \* (4)PublisherServiceImpl.java: 实现PublisherService接口，提供关于发布者管理的具体业务逻辑服务
77. \*/
78. package com.zq.ei.publisherrealtime.service.impl;
79. import com.zq.ei.publisherrealtime.mapper.PublisherMapper;
80. import com.zq.ei.publisherrealtime.service.PublisherService;
81. import org.springframework.beans.factory.annotation.Autowired;
82. import org.springframework.stereotype.Service;
83. import java.util.Map;
84. */\*\**
85. \* 业务层
86. \*/
87. @Service
88. public class PublisherServiceImpl implements PublisherService {
89. @Autowired
90. PublisherMapper publisherMapper;
91. @Override
92. public String doDauRealtime() {
93. *//业务处理*
94. String dauResults = publisherMapper.searchDau();
95. return dauResults;
96. }
97. @Override
98. public String doDauRealtime3() {
99. *//业务处理*
100. String dauResults = publisherMapper.searchDau3();
101. return dauResults;
102. }
103. }
104. */\*\**
105. \* (5)CompanyTypeService.java: 定义和实现对公司类型服务的业务逻辑操作
106. \*/
107. package com.zq.ei.publisherrealtime.service;
108. import com.zq.ei.publisherrealtime.bean.CompanyType;
109. import org.springframework.stereotype.Service;
110. import com.fasterxml.jackson.core.JsonProcessingException;
111. import com.fasterxml.jackson.databind.ObjectMapper;
112. import java.util.ArrayList;
113. import java.util.List;
114. @Service
115. public class CompanyTypeService {
116. public String getCompanyTypesAsJson() throws JsonProcessingException {
117. List<CompanyType> companyTypes = new ArrayList<>();
118. companyTypes.add(new CompanyType("外企", 23962));
119. companyTypes.add(new CompanyType("私营", 21119));
120. companyTypes.add(new CompanyType("中外合办", 19924));
121. companyTypes.add(new CompanyType("国企", 19791));
122. companyTypes.add(new CompanyType("其它", 17073 ));
123. *// 添加更多CompanyType对象...*
124. ObjectMapper objectMapper = new ObjectMapper();
125. return objectMapper.writeValueAsString(companyTypes);
126. }
127. }
128. */\*\**
129. \* (6)PublisherService.java: 定义对外发布的服务接口，可能包含创建、更新、删除和查询实时发布内容的方法
130. \*/
131. package com.zq.ei.publisherrealtime.service;
132. import java.util.Map;
133. public interface PublisherService {
134. String doDauRealtime();
135. String doDauRealtime3();
136. }
137. */\*\**
138. \* (7)CorsConfig.java: 配置跨域资源共享设置，允许前端应用从不同源访问API资源
139. \*/
140. package com.zq.ei.publisherrealtime;
141. import org.springframework.context.annotation.Configuration;
142. import org.springframework.web.servlet.config.annotation.CorsRegistry;
143. import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
144. *// 前后端跨域问题配置类*
145. @Configuration
146. public class CorsConfig implements WebMvcConfigurer {
148. @Override
149. public void addCorsMappings(CorsRegistry registry) {
150. registry.addMapping("/\*\*")
151. *//是否发送Cookie*
152. .allowCredentials(true)
153. *//放行哪些原始域*
154. .allowedOriginPatterns("\*")
155. .allowedMethods(new String[]{"GET", "POST", "PUT", "DELETE"})
156. .allowedHeaders("\*")
157. .exposedHeaders("\*");
158. }
159. }
160. */\*\**
161. \* (8)PublisherRealtimeApplication.java: 应用程序主入口点，配置Spring Boot等框架启动项及初始化服务
162. \*/
163. package com.zq.ei.publisherrealtime;
164. import org.springframework.boot.SpringApplication;
165. import org.springframework.boot.autoconfigure.SpringBootApplication;
166. @SpringBootApplication
167. public class PublisherRealtimeApplication {
168. public static void main(String[] args) {
169. SpringApplication.run(PublisherRealtimeApplication.class, args);
170. }
171. }

**5.主页代码**

1. *<!-- 主页代码 -->*
2. <html>
3. <head>
4. <title>大数据就业信息可视化平台</title>
5. <meta charset="utf-8">
6. <meta name="viewport" content="width=device-width, initial-scale=1">
7. <style>
8. body {
9. color: aliceblue;
10. }
11. button {
12. background-color: aqua;
13. }
14. h2 {
15. text-align: center;
16. }
17. .grid-container {
18. display: grid;
19. */\* 6列，定义列宽 \*/*
20. grid-template-columns: 14% 14.5% 20% 20% 14.5% 14%;
21. */\* auto: 它用于自动设置行的高度，即取决于行中容器和内容的大小。 \*/*
22. grid-template-rows: 10% 25% 30% 30%;
23. grid-gap: 10px;
24. */\* background-color: #2196F3; \*/*
25. padding: 0;
26. width: 100%;
27. height: 100%;
28. }
29. #lo\_0 {
30. grid-area: 1 / 1 / 2 / 7;
31. background-image: url(img/header/15.png);
32. background-size: 100%;
33. background-repeat: no-repeat;
34. }
35. #lo\_5 {
36. grid-area: 3 / 1 / 4 / 3;
37. }
38. #lo\_6 {
39. grid-area: 3 / 5 / 4 / 7;
40. }
41. #lo\_7 {
42. grid-area: 4 / 1 / 5 / 3;
43. }
44. #lo\_8 {
45. grid-area: 2 / 3 / 4 / 5;
46. }
47. #lo\_9 {
48. grid-area: 4 / 3 / 5 / 5;
49. }
50. #lo\_10 {
51. grid-area: 4 / 5 / 5 / 7;
52. }
53. #lo\_1,
54. #lo\_2,
55. #lo\_3,
56. #lo\_4,
57. #lo\_5,
58. #lo\_6,
59. #lo\_7,
60. #lo\_8,
61. #lo\_9,
62. #lo\_10 {
63. */\* 使用border效果 \*/*
64. border: 6px solid transparent;
65. border-image: url(img/46.png);
66. border-image-slice: 6;
67. border-image-repeat: stretch;
68. */\* border: 5px solid blueviolet; \*/*
69. text-align: center;
70. padding: 0;
71. font-size: 30px;
72. }
73. </style>
74. *<!-- 官方库依赖 -->*
75. <script type="text/javascript" src="js/jquery/jquery-3.6.0.min.js"></script>
76. <script type="text/javascript" src="js/echarts/echarts.min.js"></script>
77. </head>
78. <body style="background-image: url(img/bg-9.png);  background-size: 100% ;  background-repeat: no-repeat;">
79. <div class="grid-container">
80. <div id="lo\_0">
81. <h2>大数据就业信息可视化平台</h2>
82. </div>
83. <div id="lo\_1">
84. </div>
85. <div id="lo\_2">
86. </div>
87. <div id="lo\_3">
88. </div>
89. <div id="lo\_4">
90. </div>
91. <div id="lo\_5">
92. </div>
93. <div id="lo\_6">
94. </div>
95. <div id="lo\_7">
96. </div>
97. <div id="lo\_8">
98. <div style="height: 10%;">
99. <button
100. onclick="async\_echart\_china('container\_8', 'map\_china\_map/map\_china\_map.json', 'confirmAdd')">新增就业岗位</button>
101. <button
102. onclick="async\_echart\_china('container\_8', 'map\_china\_map/map\_china\_map.json', 'confirm')">累计就业岗位</button>
103. <button
104. onclick="async\_echart\_china('container\_8', 'map\_china\_map/map\_china\_map.json', 'nowConfirm')">现有就业岗位</button>
105. </div>
106. <div id="container\_8" style="height: 90%;"></div>
107. </div>
108. <div id="lo\_9">9</div>
109. <div id="lo\_10">10</div>
110. </div>
111. *<!-- 鼠标右键切换主题 theme [1/2] -->*
112. *<!-- style="width: 200px;" 必须写在html，不能写在css文件中 -->*
113. <ul id="right\_menu" style="width: 200px;">
114. <li><img src="img/drop-down.png"> 主题列表</li>
115. <li>infographic</li>
116. <li>macarons</li>
117. <li>roma</li>
118. <li>shine</li>
119. <li>walden</li>
120. <li>westeros</li>
121. <li>wonderland</li>
122. <li>vintage</li>
123. <li>purple-passion</li>
124. <li>chalk</li>
125. <li>dark</li>
126. <li>essos</li>
127. </ul>
128. </body>
129. *<!-- 用户自定义函数依赖 -->*
130. <script type="text/javascript" src="js/udf/base.js"></script>
131. *<!-- 鼠标右键切换主题 theme [2/2] -->*
132. <script type="text/javascript" src="theme/js/switch\_theme.js"></script>
133. <script type="text/javascript" src="pie\_source/loader.js"></script>
134. <script type="text/javascript" src="line\_visualMap/loader.js"></script>
135. <script type="text/javascript" src="line\_area\_chart/loader.js"></script>
136. <script type="text/javascript" src="map\_china\_map/loader.js"></script>
137. <script type="text/javascript" src="bar\_stacked/loader.js"></script>
138. <script type="text/javascript" src="funnel/loader.js"></script>
139. <script type="text/javascript" src="bar\_markLine/loader.js"></script>
140. <script type="text/javascript" src="pie\_source\_rose/loader.js"></script>
141. <script type="text/javascript" src="bar\_ROA/loader.js"></script>
142. <script type="text/javascript" src="bar\_horizontal/loader.js"></script>
143. </html>
144. /\*\*
145. \* right\_menu.css
146. \*/
147. #right\_menu {
148. /\* list-style: none; \*/
149. background: rgb(143, 181, 238);
150. /\* border: solid 1px rgb(196, 222, 236); \*/
151. margin: 10px;
152. padding: 10px;
153. position: absolute;
154. /\* left: 0px; \*/
155. top: 10px;
156. /\* 默认不显示 \*/
157. display: none;
158. }
159. #right\_menu li {
160. border-bottom: dotted 1px rgb(74, 107, 129);
161. }

**6.各城市招牌数量页面代码**

1. <!-- 各城市招牌数量页面代码 -->
2. <html>
3. <head>
4. <title>大数据就业信息可视化平台</title>
5. <meta charset="utf-8">
6. <meta name="viewport" content="width=device-width, initial-scale=1">
7. </head>
8. <link href="css/bootstrap-combined.min.css" rel="stylesheet">
9. <link href="css/layoutit.css" rel="stylesheet">
10. <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
11. </script>
12. <!-- 官方库依赖 -->
13. <script type="text/javascript" src="js/bootstrap/bootstrap.min.js"></script>
14. <script type="text/javascript" src="js/jquery/jquery-3.6.0.min.js"></script>
15. <script type="text/javascript" src="js/vue/vue.js"></script>
16. <script type="text/javascript" src="js/vue/dv.js"></script>
17. <script type="text/javascript" src="js/echarts/echarts.min.js"></script>
18. <body style="background-color:rgb(11, 18, 39)">
19. <div id="vue\_app" class="container-fluid">
20. <div id="container\_b6" style="height: 100%; width: 100%;">
21. </div>
22. </div>
23. </body>
24. <!-- 用户自定义函数依赖 -->
25. <script type="text/javascript" src="js/udf/base.js"></script>
26. <script type="text/javascript" src="bar\_horizontal/loader.js"></script>
27. */\*\**
28. \* bar\_horizontal.js
29. \*/
30. function init\_echart\_bar\_horizontal(container) {
31. var chartDom = document.getElementById(container);
32. var myChart = echarts.init(chartDom, window.gTheme);
33. var option = {
34. title: {
35. text: "各个城市招聘数量",
36. *// top: "5%",*
37. *// left: "2%",*
38. textStyle: {
39. *// color: "#3690be",*
40. color: "orange",
41. fontSize: "12",
42. },
43. },
44. tooltip: {
45. trigger: "item",
46. formatter: "{b}: {c} 个",
47. position: function (p) {
48. *//其中p为当前鼠标的位置*
49. return [p[0] + 10, p[1] - 10];
50. },
51. },
52. grid: {
53. left: "5%",
54. *//   top: "30%",*
55. *//   right: "5%",*
56. bottom: "2%",
57. containLabel: true,
58. },
59. xAxis: [
60. {
61. name: "个",
62. type: "value",
63. min: 0,
64. max: 100,
65. nameLocation: "start",
66. nameTextStyle: {
67. color: "#3690be",
68. *//fontSize: 14,*
69. },
70. axisLabel: {
71. textStyle: {
72. color: "rgba(255,255,255,.8)",
73. *//fontSize: 14,*
74. },
75. },
76. axisLine: {
77. lineStyle: {
78. color: "rgba(255,255,255,.8)",
79. },
80. },
81. splitLine: {
82. lineStyle: {
83. color: "rgba(255,255,255,.1)",
84. },
85. },
86. },
87. ],
88. yAxis: {
89. *// name: "城市",*
90. type: "category",
91. data: [],
92. axisLabel: {
93. textStyle: {
94. color: "rgba(255,255,255,.8)",
95. *//fontSize: 14,*
96. },
97. },
98. axisLine: {
99. lineStyle: {
100. color: "rgba(255,255,255,.8)",
101. },
102. },
103. splitLine: {
104. lineStyle: {
105. color: "rgba(255,255,255,.1)",
106. },
107. },
108. },
109. series: [
110. {
111. *// name: "个",*
112. type: "bar",
113. xAxisIndex: 0,
114. label: {
115. show: true,
116. },
117. emphasis: {
118. focus: "series",
119. },
120. data: [],
121. },
122. ],
123. };
124. *// 使用刚指定的配置项和数据显示图表。*
125. myChart.setOption(option);
126. window.addEventListener("resize", function () {
127. myChart.resize();
128. });
129. }
130. *// function async\_echart\_bar\_horizontal(container, filename) {*
131. *//   $.getJSON(filename).done(function (data) {*
132. *//     var myChart = echarts.init(*
133. *//       document.getElementById(container),*
134. *//       window.gTheme*
135. *//     );*
136. *//*
137. *//     myChart.setOption({*
138. *//       yAxis: { data: data["yAxis"]["data"] },*
139. *//       series: [{ data: data["series"][0]["data"] }],*
140. *//     });*
141. *//   }); //end $.getJSON}*
142. *// }*
143. *// ajax读取后端数据，传递参数为容器元素位置与后端接口url*
144. function async\_echart\_bar\_horizontal(container, url) {
145. *// 确保容器元素存在*
146. var containerElement = document.getElementById(container);
147. if (!containerElement) {
148. console.error('Container element not found:', container);
149. return;
150. }
151. $.ajax({
152. url: url, *// 后端接口的URL*
153. dataType: 'json', *// 期望返回的数据类型为JSON*
154. success: function (data) {
155. *// 初始化ECharts实例*
156. var myChart = echarts.init(containerElement, window.gTheme);
157. *// 使用从后端接口获取的数据设置图表选项*
158. *// 确保数据结构与ECharts兼容*
159. if (data && data.yAxis && data.yAxis.data && data.series && data.series[0] && data.series[0].data) {
160. myChart.setOption({
161. yAxis: {
162. data: data.yAxis.data
163. },
164. series: [{
165. data: data.series[0].data
166. }]
167. });
168. } else {
169. console.error('Invalid data structure received from server:', data);
170. }
171. },
172. error: function (jqXHR, textStatus, errorThrown) {
173. *// 处理请求失败的情况*
174. console.error("Error fetching data from server: " + textStatus, errorThrown);
175. }
176. }); *// end $.ajax*
177. }
178. */\*\**
179. \* loader.js
180. \*/
181. *// 加载主题，必须在echarts.init之前。*
182. var gTheme = getQueryVariable("theme");
183. themeUrl = "theme/js/" + gTheme + ".js";
184. addScript(themeUrl);
185. *// 加载依赖文件*
186. var path\_bar\_horizontal = "bar\_horizontal/";
187. addScript(path\_bar\_horizontal + "bar\_horizontal.js");
188. *// 更新图表数据*
189. $(document).ready(function () {
190. var container = "lo\_1";
191. init\_echart\_bar\_horizontal(container);
192. *// 定时1s执行数据更新函数*
193. setInterval(function () {
194. async\_echart\_bar\_horizontal(
195. container,
196. "http://localhost:8080/dauRealtime"
197. );
198. }, 1000);
199. });

**7.平均薪资待遇最高的行业页面代码**

1. <!-- 平均薪资待遇最高的行业页面代码 -->
2. <html>
3. <head>
4. <title>YYDataV 数据可视化案例</title>
5. <meta charset="utf-8">
6. <meta name="viewport" content="width=device-width, initial-scale=1">
7. </head>
8. <link href="css/bootstrap-combined.min.css" rel="stylesheet">
9. <link href="css/layoutit.css" rel="stylesheet">
10. <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
11. </script>
12. <!-- 官方库依赖 -->
13. <script type="text/javascript" src="js/bootstrap/bootstrap.min.js"></script>
14. <script type="text/javascript" src="js/jquery/jquery-3.6.0.min.js"></script>
15. <script type="text/javascript" src="js/vue/vue.js"></script>
16. <script type="text/javascript" src="js/vue/dv.js"></script>
17. <script type="text/javascript" src="js/echarts/echarts.min.js"></script>
18. <body style="background-color:rgb(11, 18, 39)">
19. <div id="vue\_app" class="container-fluid">
20. <div id="container" style="height: 100%; width: 100%;">
21. </div>
22. </div>
23. </body>
24. <!-- 用户自定义函数依赖 -->
25. <script type="text/javascript" src="js/udf/base.js"></script>
26. <script type="text/javascript" src="bar\_markLine/loader.js"></script>
27. */\*\**
28. \* bar\_markLine.js
29. \*/
30. function init\_echart\_bar\_age(container) {
31. *// 基于准备好的dom，初始化echarts实例*
32. var myChart = echarts.init(document.getElementById(container), gTheme);
33. option = {
34. title: {
35. text: "平均薪资待遇最高的行业",
36. *//top: "5%",*
37. *//left: "2%",*
38. textStyle: {
39. *// color: "#17c0ff",*
40. color: "orange",
41. fontSize: "12",
42. },
43. },
44. tooltip: {
45. trigger: "item",
46. formatter: "{a} <br/>{b}: {c} ({d}%)",
47. position: function (p) {
48. *//其中p为当前鼠标的位置*
49. return [p[0] + 10, p[1] - 10];
50. },
51. },
52. grid: {
53. left: "3%",
54. right: "15%",
55. bottom: "3%",
56. top: "30%",
57. containLabel: true,
58. },
59. xAxis: {
60. type: "category",
61. data: [],
62. axisLabel: {
63. textStyle: {
64. color: "rgba(255,255,255,.8)",
65. *//fontSize: 14,*
66. },
67. },
68. axisLine: {
69. lineStyle: {
70. color: "rgba(255,255,255,.2)",
71. },
72. },
73. splitLine: {
74. lineStyle: {
75. color: "rgba(255,255,255,.1)",
76. },
77. },
78. },
79. yAxis: {
80. type: "value",
81. data: [],
82. axisLabel: {
83. textStyle: {
84. color: "rgba(255,255,255,.8)",
85. *//fontSize: 14,*
86. },
87. },
88. axisLine: {
89. lineStyle: {
90. color: "rgba(255,255,255,.2)",
91. },
92. },
93. splitLine: {
94. lineStyle: {
95. color: "rgba(255,255,255,.1)",
96. },
97. },
98. },
99. series: [
100. {
101. name: "平均薪资",
102. type: "bar",
103. *// stack: "total",*
104. *// label: {*
105. *//   show: true,*
106. *// },*
107. markLine: {
108. *// 图形是否不响应和触发鼠标事件*
109. silent: true,
110. lineStyle: {
111. color: "red",
112. },
113. label: {
114. textStyle: {
115. color: "rgba(255,255,255,.8)",
116. *//fontSize: 14,*
117. },
118. },
119. data: [
120. {
121. yAxis: 4000,
122. },
123. {
124. yAxis: 5000,
125. },
126. ],
127. },
128. },
129. ],
130. };
131. *// 使用刚指定的配置项和数据显示图表。*
132. myChart.setOption(option);
133. window.addEventListener("resize", function () {
134. myChart.resize();
135. });
136. }
137. function getKeys(dataList) {
138. var keys = [];
139. var len = dataList.length;
140. for (var i = 0; i < len; i++) keys.push(dataList[i].name);
141. return keys;
142. }
143. function async\_echart\_bar\_age(container, filename) {
144. $.getJSON(filename).done(function (data) {
145. var myChart = echarts.init(document.getElementById(container));
146. myChart.setOption({
147. xAxis: { data: getKeys(data) },
148. series: [{ data: data }],
149. });
150. }); *//end $.getJSON*
151. }
152. *// ajax读取后端数据，传递参数为容器元素位置与后端接口url*
153. function async\_echart\_bar\_horizontal(container, url) {
154. *// 确保容器元素存在*
155. var containerElement = document.getElementById(container);
156. if (!containerElement) {
157. console.error('Container element not found:', container);
158. return;
159. }
160. $.ajax({
161. url: url, *// 后端接口的URL*
162. dataType: 'char', *// 期望返回的数据类型为JSON*
163. success: function (data) {
164. *// 初始化ECharts实例*
165. var myChart = echarts.init(containerElement, window.gTheme);
166. *// 使用从后端接口获取的数据设置图表选项*
167. *// 确保数据结构与ECharts兼容*
168. if (data) {
169. myChart.setOption({
170. xAxis: { data: getKeys(data) },
171. series: [{ data: data }],
172. });
173. } else {
174. console.error('Invalid data structure received from server:', data);
175. }
176. },
177. error: function (jqXHR, textStatus, errorThrown) {
178. *// 处理请求失败的情况*
179. console.error("Error fetching data from server: " + textStatus, errorThrown);
180. }
181. }); *// end $.ajax*
182. }
183. */\*\**
184. \* loader.js
185. \*/
186. *// 加载主题，必须在echarts.init之前。*
187. var gTheme = getQueryVariable("theme");
188. themeUrl = "theme/js/" + gTheme + ".js";
189. addScript(themeUrl);
190. *// 加载依赖文件*
191. var path\_bar\_age = "bar\_markLine/";
192. addScript(path\_bar\_age + "bar\_markLine.js");
193. $(document).ready(function () {
194. var container = "lo\_10";
195. init\_echart\_bar\_age(container);
196. *// 定时1s执行数据更新函数*
197. setInterval(function () {
198. async\_echart\_bar\_age(container, path\_bar\_age + "bar\_markLine.json");
199. }, 1000);
200. });

**8.全国在招岗位规模页面代码**

1. <!-- 全国在招岗位规模页面代码 -->
2. <html>
3. <head>
4. <title>YYDataV 数据可视化案例</title>
5. <meta charset="utf-8">
6. <meta name="viewport" content="width=device-width, initial-scale=1">
7. </head>
8. <link href="css/bootstrap-combined.min.css" rel="stylesheet">
9. <link href="css/layoutit.css" rel="stylesheet">
10. <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
11. </script>
12. <!-- 官方库依赖 -->
13. <script type="text/javascript" src="js/bootstrap/bootstrap.min.js"></script>
14. <script type="text/javascript" src="js/jquery/jquery-3.6.0.min.js"></script>
15. <script type="text/javascript" src="js/vue/vue.js"></script>
16. <script type="text/javascript" src="js/vue/dv.js"></script>
17. <script type="text/javascript" src="js/echarts/echarts.min.js"></script>
18. <body style="background-color:rgb(11, 18, 39)">
19. <div id="vue\_app" class="container-fluid">
20. <div id="container\_b1" style="height: 100%; width: 100%;">
21. </div>
22. </div>
23. </body>
24. <!-- 用户自定义函数依赖 -->
25. <script type="text/javascript" src="js/udf/base.js"></script>
26. <script type="text/javascript" src="bar\_ROA/loader.js"></script>
27. */\*\**
28. \* bar\_ROA.js
29. \*/
30. function init\_echart\_bar\_ROA(container) {
31. var chartDom = document.getElementById(container);
32. var myChart = echarts.init(chartDom, window.gTheme);
33. var option;
34. option = {
35. title: {
36. text: "全国在招公司规模",
37. *//top: "5%",*
38. *//left: "2%",*
39. textStyle: {
40. *// color: "#00ffff",*
41. color: "orange",
42. fontSize: "12",
43. },
44. },
45. grid: {
46. left: "3%",
47. right: "4%",
48. bottom: "10%",
49. top: "30%",
50. containLabel: true,
51. },
52. tooltip: {
53. trigger: "item",
54. formatter: "{b}: {c} ",
55. position: function (p) {
56. *//其中p为当前鼠标的位置*
57. return [p[0] + 10, p[1] - 10];
58. },
59. },
60. xAxis: {
61. name: "名称",
62. type: "category",
63. nameTextStyle: {
64. color: "rgba(255,255,255,.8)",
65. *// fontSize: 12,*
66. },
67. data: [],
68. axisLabel: {
69. textStyle: {
70. color: "rgba(255,255,255,.8)",
71. *// fontSize: 14,*
72. },
73. },
74. axisLine: {
75. lineStyle: {
76. color: "rgba(255,255,255,.2)",
77. },
78. },
79. splitLine: {
80. lineStyle: {
81. color: "rgba(255,255,255,.1)",
82. },
83. },
84. },
85. yAxis: {
86. name: "数量（个）",
87. type: "value",
88. nameTextStyle: {
89. color: "rgba(255,255,255,.8)",
90. *// fontSize: 12,*
91. },
92. axisLabel: {
93. textStyle: {
94. color: "rgba(255,255,255,.8)",
95. *// fontSize: 14,*
96. },
97. },
98. axisLine: {
99. lineStyle: {
100. color: "rgba(255,255,255,.2)",
101. },
102. },
103. splitLine: {
104. lineStyle: {
105. color: "rgba(255,255,255,.1)",
106. },
107. },
108. },
109. series: [
110. {
111. data: [],
112. type: "bar",
113. showBackground: true,
114. backgroundStyle: {
115. color: "rgba(180, 180, 180, 0.2)",
116. },
117. },
118. ],
119. };
120. option && myChart.setOption(option);
121. window.addEventListener("resize", function () {
122. myChart.resize();
123. });
124. }
125. function async\_echart\_bar\_ROA(container, filename) {
126. $.getJSON(filename).done(function (data) {
127. var myChart = echarts.init(
128. document.getElementById(container),
129. window.gTheme
130. );
131. myChart.setOption(data);
132. }); *//end $.getJSON*
133. }
134. *// // ajax读取后端数据，传递参数为容器元素位置与后端接口url*
135. *// function async\_echart\_bar\_horizontal(container, url) {*
136. *//   // 确保容器元素存在*
137. *//   var containerElement = document.getElementById(container);*
138. *//   if (!containerElement) {*
139. *//     console.error('Container element not found:', container);*
140. *//     return;*
141. *//   }*
142. *//*
143. *//   $.ajax({*
144. *//     url: url, // 后端接口的URL*
145. *//     dataType: 'json', // 期望返回的数据类型为JSON*
146. *//     success: function (data) {*
147. *//       // 初始化ECharts实例*
148. *//       var myChart = echarts.init(containerElement, window.gTheme);*
149. *//*
150. *//       // 使用从后端接口获取的数据设置图表选项*
151. *//       // 确保数据结构与ECharts兼容*
152. *//       if (data) {*
153. *//         myChart.setOption(*
154. *//           data*
155. *//         );*
156. *//       } else {*
157. *//         console.error('Invalid data structure received from server:', data);*
158. *//       }*
159. *//     },*
160. *//     error: function (jqXHR, textStatus, errorThrown) {*
161. *//       // 处理请求失败的情况*
162. *//       console.error("Error fetching data from server: " + textStatus, errorThrown);*
163. *//     }*
164. *//   }); // end $.ajax*
165. *// }*
166. */\*\**
167. \* loader.js
168. \*/
169. *// 加载主题，必须在echarts.init之前。*
170. var gTheme = getQueryVariable("theme");
171. themeUrl = "theme/js/" + gTheme + ".js";
172. addScript(themeUrl);
173. *// 加载依赖文件*
174. var path\_bar\_ROA = "bar\_ROA/";
175. addScript(path\_bar\_ROA + "bar\_ROA.js");
176. $(document).ready(function () {
177. var container = "lo\_6";
178. init\_echart\_bar\_ROA(container);
179. *// 定时1s执行数据更新函数*
180. setInterval(function () {
181. async\_echart\_bar\_ROA(container, path\_bar\_ROA + "bar\_ROA.json");
182. }, 1000);
183. });

**9.近期招牌公司产权情况页面代码**

1. <!-- 近期招牌公司产权情况页面代码 -->
2. <html>
3. <head>
4. <title>YYDataV 数据可视化案例</title>
5. <meta charset="utf-8">
6. <meta name="viewport" content="width=device-width, initial-scale=1">
7. </head>
8. <link href="css/bootstrap-combined.min.css" rel="stylesheet">
9. <link href="css/layoutit.css" rel="stylesheet">
10. <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
11. </script>
12. <!-- 官方库依赖 -->
13. <script type="text/javascript" src="js/bootstrap/bootstrap.min.js"></script>
14. <script type="text/javascript" src="js/jquery/jquery-3.6.0.min.js"></script>
15. <script type="text/javascript" src="js/vue/vue.js"></script>
16. <script type="text/javascript" src="js/vue/dv.js"></script>
17. <script type="text/javascript" src="js/echarts/echarts.min.js"></script>
18. <body style="background-color:rgb(11, 18, 39)">
19. <div id="vue\_app" class="container-fluid">
20. <div id="container\_b3" style="height: 100%; width: 100%;">
21. </div>
22. </div>
23. </body>
24. <!-- 用户自定义函数依赖 -->
25. <script type="text/javascript" src="js/udf/base.js"></script>
26. <script type="text/javascript" src="bar\_stacked/loader.js"></script>
27. */\*\**
28. \* bar\_stacked.js
29. \*/
30. function init\_echart\_bar\_stacked(container) {
31. var chartDom = document.getElementById(container);
32. var myChart = echarts.init(chartDom, window.gTheme);
33. var option;
34. option = {
35. title: {
36. text: "近期招聘公司产权情况",
37. top: 0,
38. left: "left",
39. textStyle: {
40. *// color: "#17c0ff",*
41. color: "orange",
42. fontSize: "12",
43. },
44. },
45. tooltip: {
46. trigger: "axis",
47. axisPointer: {
48. *// Use axis to trigger tooltip*
49. type: "shadow", *// 'shadow' as default; can also be 'line' or 'shadow'*
50. },
51. },
52. legend: {
53. data: ["未融资", "天使轮", "A轮", "A轮以上", "上市"],
54. textStyle: {
55. color: "rgba(255, 255, 255, 0.8)",
56. },
57. *// top: "20%",*
58. *// orient:"vertical"*
59. },
60. grid: {
61. left: "3%",
62. right: "4%",
63. top: "35%",
64. bottom: "10%",
65. containLabel: true,
66. },
67. xAxis: {
68. type: "value",
69. axisLabel: {
70. textStyle: {
71. color: "rgba(255,255,255,.8)",
72. *// fontSize: 14,*
73. },
74. },
75. axisLine: {
76. lineStyle: {
77. color: "rgba(255,255,255,.2)",
78. },
79. },
80. splitLine: {
81. lineStyle: {
82. color: "rgba(255,255,255,.1)",
83. },
84. },
85. },
86. yAxis: {
87. type: "category",
88. data: ["01.11", "01.12", "01.13", "01.14"],
89. axisLabel: {
90. textStyle: {
91. color: "rgba(255,255,255,.8)",
92. *// fontSize: 14,*
93. },
94. },
95. axisLine: {
96. lineStyle: {
97. color: "rgba(255,255,255,.2)",
98. },
99. },
100. splitLine: {
101. lineStyle: {
102. color: "rgba(255,255,255,.1)",
103. },
104. },
105. },
106. series: [
107. {
108. name: "未融资",
109. type: "bar",
110. stack: "total",
111. label: {
112. show: true,
113. },
114. emphasis: {
115. focus: "series",
116. },
117. data: [],
118. },
119. {
120. name: "天使轮",
121. type: "bar",
122. stack: "total",
123. label: {
124. show: true,
125. },
126. emphasis: {
127. focus: "series",
128. },
129. data: [],
130. },
131. {
132. name: "A轮",
133. type: "bar",
134. stack: "total",
135. label: {
136. show: true,
137. },
138. emphasis: {
139. focus: "series",
140. },
141. data: [],
142. },
143. {
144. name: "A轮以上",
145. type: "bar",
146. stack: "total",
147. label: {
148. show: true,
149. },
150. emphasis: {
151. focus: "series",
152. },
153. data: [],
154. },
155. {
156. name: "上市",
157. type: "bar",
158. stack: "total",
159. label: {
160. show: true,
161. },
162. emphasis: {
163. focus: "series",
164. },
165. data: [],
166. },
167. ],
168. };
169. option && myChart.setOption(option);
170. window.addEventListener("resize", function () {
171. myChart.resize();
172. });
173. }
174. function async\_echart\_bar\_stacked(container, filename) {
175. $.getJSON(filename).done(function (data) {
176. var myChart = echarts.init(document.getElementById(container));
177. myChart.setOption({
178. series: [
179. { data: data[0] },
180. { data: data[1] },
181. { data: data[2] },
182. { data: data[3] },
183. { data: data[4] },
184. ],
185. });
186. }); *//end $.getJSON*
187. }
188. *// ajax读取后端数据，传递参数为容器元素位置与后端接口url*
189. function async\_echart\_bar\_horizontal(container, url) {
190. *// 确保容器元素存在*
191. var containerElement = document.getElementById(container);
192. if (!containerElement) {
193. console.error('Container element not found:', container);
194. return;
195. }
196. $.ajax({
197. url: url, *// 后端接口的URL*
198. dataType: 'json', *// 期望返回的数据类型为JSON*
199. success: function (data) {
200. *// 初始化ECharts实例*
201. var myChart = echarts.init(containerElement, window.gTheme);
202. *// 使用从后端接口获取的数据设置图表选项*
203. *// 确保数据结构与ECharts兼容*
204. if (data && data.yAxis && data.yAxis.data && data.series && data.series[0] && data.series[0].data) {
205. myChart.setOption({
206. yAxis: {
207. data: data.yAxis.data
208. },
209. series: [{
210. data: data.series[0].data
211. }]
212. });
213. } else {
214. console.error('Invalid data structure received from server:', data);
215. }
216. },
217. error: function (jqXHR, textStatus, errorThrown) {
218. *// 处理请求失败的情况*
219. console.error("Error fetching data from server: " + textStatus, errorThrown);
220. }
221. }); *// end $.ajax*
222. }
223. */\*\**
224. \* loader.js
225. \*/
226. *// 加载主题，必须在echarts.init之前。*
227. var gTheme = getQueryVariable("theme");
228. themeUrl = "theme/js/" + gTheme + ".js";
229. addScript(themeUrl);
230. *// 加载依赖文件*
231. var path\_bar\_stacked = "bar\_stacked/";
232. addScript(path\_bar\_stacked + "bar\_stacked.js");
233. $(document).ready(function () {
234. var container = "lo\_9";
235. init\_echart\_bar\_stacked(container);
236. *// 定时1s执行数据更新函数*
237. setInterval(function () {
238. async\_echart\_bar\_stacked(container, path\_bar\_stacked + "bar\_stacked.json");
239. }, 1000);
240. });

**10.热门岗位薪资待遇时事检测页面代码**

1. <!-- 热门岗位薪资待遇时事检测页面代码 -->
2. <html>
3. <head>
4. <title>YYDataV 数据可视化案例</title>
5. <meta charset="utf-8">
6. <meta name="viewport" content="width=device-width, initial-scale=1">
7. </head>
8. <link href="css/bootstrap-combined.min.css" rel="stylesheet">
9. <link href="css/layoutit.css" rel="stylesheet">
10. <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
11. </script>
12. <!-- 官方库依赖 -->
13. <script type="text/javascript" src="js/bootstrap/bootstrap.min.js"></script>
14. <script type="text/javascript" src="js/jquery/jquery-3.6.0.min.js"></script>
15. <script type="text/javascript" src="js/vue/vue.js"></script>
16. <script type="text/javascript" src="js/vue/dv.js"></script>
17. <script type="text/javascript" src="js/echarts/echarts.min.js"></script>
18. <body style="background-color:rgb(11, 18, 39)">
19. <div id="vue\_app" class="container-fluid">
20. <div id="container\_0" style="height: 100%; width: 100%;">
21. </div>
22. </div>
23. </body>
24. <!-- 用户自定义函数依赖 -->
25. <script type="text/javascript" src="js/udf/base.js"></script>
26. <script type="text/javascript" src="line\_visualMap/loader.js"></script>
27. */\*\**
28. \* line\_visualMap.js
29. \*/
30. function init\_echart\_line\_visualMap(container) {
31. *// 基于准备好的dom，初始化echarts实例*
32. var myChart = echarts.init(document.getElementById(container), gTheme);
33. option = {
34. title: {
35. text: "热门岗位薪资待遇实时监测",
36. *// top: 0,*
37. *// left: "center",*
38. textStyle: {
39. *// color: "#17c0ff",*
40. color: "orange",
41. fontSize: "12",
42. },
43. },
44. tooltip: {
45. trigger: "item",
46. formatter: "{a} <br/>{b}: {c} ({d}%)",
47. position: function (p) {
48. *//其中p为当前鼠标的位置*
49. return [p[0] + 10, p[1] - 10];
50. },
51. },
52. grid: {
53. left: "3%",
54. right: "3%",
55. bottom: "3%",
56. top: "25%",
57. containLabel: true,
58. },
59. xAxis: {
60. name: "名称",
61. type: "category",
62. data: [],
63. axisLabel: {
64. textStyle: {
65. color: "rgba(255,255,255,.8)",
66. *//fontSize: 14,*
67. },
68. *// formatter: "{value}%",*
69. },
70. axisLine: {
71. lineStyle: {
72. color: "rgba(255,255,255,.2)",
73. },
74. },
75. splitLine: {
76. lineStyle: {
77. color: "rgba(255,255,255,.1)",
78. },
79. },
80. },
81. yAxis: {
82. name: "元",
83. type: "value",
84. data: [],
85. axisLabel: {
86. textStyle: {
87. color: "rgba(255,255,255,.8)",
88. *//fontSize: 14,*
89. },
90. formatter: "{value}",
91. },
92. axisLine: {
93. lineStyle: {
94. color: "rgba(255,255,255,.2)",
95. },
96. },
97. splitLine: {
98. lineStyle: {
99. color: "rgba(255,255,255,.1)",
100. },
101. },
102. },
103. visualMap: {
104. top: "top",
105. left: "right",
106. textStyle: {
107. color: "rgba(255,255,255,.8)",
108. *//fontSize: 14,*
109. },
110. pieces: [
111. {
112. gt: 0,
113. lte: 100,
114. color: "#FF0000",
115. },
116. {
117. gt: 100,
118. lte: 800,
119. color: "#FFA500",
120. },
121. {
122. gt: 800,
123. lte: 900,
124. color: "#2E8B57",
125. },
126. ],
127. },
128. series: [
129. {
130. name: "该薪资平均分布",
131. type: "line",
132. *// stack: "total",*
133. *// label: {*
134. *//   show: true,*
135. *// },*
136. *// 使用系统函数*
137. markPoint: {
138. label: {
139. textStyle: {
140. color: "rgba(255,255,255,.8)",
141. *//fontSize: 14,*
142. },
143. },
144. data: [
145. { type: "max", name: "Max" },
146. { type: "min", name: "Min" },
147. ],
148. },
149. markLine: {
150. data: [{ type: "average", name: "Avg" }],
151. },
152. *// 使用刚指定的配置项和数据显示图表。*
153. myChart.setOption(option);
154. window.addEventListener("resize", function () {
155. myChart.resize();
156. });
157. }
158. function getKeys(dataList) {
159. var keys = [];
160. var len = dataList.length;
161. for (var i = 0; i < len; i++) keys.push(dataList[i].name);
162. return keys;
163. }
164. function async\_echart\_line\_visualMap(container, filename) {
165. $.getJSON(filename).done(function (data) {
166. var myChart = echarts.init(document.getElementById(container));
167. myChart.setOption({
168. xAxis: { data: getKeys(data) },
169. series: [{ data: data }],
170. });
171. }); *//end $.getJSON*
172. }
173. */\*\**
174. \* loader.js
175. \*/
176. *// 加载主题，必须在echarts.init之前。*
177. var gTheme = getQueryVariable("theme");
178. themeUrl = "theme/js/" + gTheme + ".js";
179. addScript(themeUrl);
180. *// 加载依赖文件*
181. var path\_line\_visualMap = "line\_visualMap/";
182. addScript(path\_line\_visualMap + "line\_visualMap.js");
183. $(document).ready(function () {
184. var container = "lo\_5";
185. init\_echart\_line\_visualMap(container);
186. *// 定时1s执行数据更新函数*
187. setInterval(function () {
188. async\_echart\_line\_visualMap(
189. container,
190. path\_line\_visualMap + "line\_visualMap.json"
191. );
192. }, 1000);
193. });

**11.在招岗位工作经验要求分析页面代码**

1. <!-- 在招岗位工作经验要求分析页面代码 -->
2. <html>
3. <head>
4. <title>【27】数据可视化大屏：基于 Echarts + Flask 实现的 32:9 超宽屏可视化范例 - 监控指挥中心</title>
5. <meta charset="utf-8">
6. <meta name="viewport" content="width=device-width, initial-scale=1">
7. </head>
8. <link href="css/bootstrap-combined.min.css" rel="stylesheet">
9. <link href="css/layoutit.css" rel="stylesheet">
10. <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
11. </script>
12. <!-- 官方库依赖 -->
13. <script type="text/javascript" src="js/bootstrap/bootstrap.min.js"></script>
14. <script type="text/javascript" src="js/jquery/jquery-3.6.0.min.js"></script>
15. <script type="text/javascript" src="js/vue/vue.js"></script>
16. <script type="text/javascript" src="js/vue/dv.js"></script>
17. <script type="text/javascript" src="js/echarts/echarts.min.js"></script>
18. <body style="background-color:rgb(11, 18, 39)">
19. <div id="vue\_app" class="container-fluid">
20. <div class="row" style="height: 100%;">
21. <div class="col-6">
22. <div id="container\_l1" style="height: 100%; "></div>
23. </div>
24. <div class="col-6">
25. <div id="container\_l2" style="height: 100%; "></div>
26. </div>
27. </div>
28. </div>
29. </body>
30. <script>
31. var vpp = new Vue({
32. el: "#vue\_app",
33. data() { }
34. })
35. </script>
36. <!-- 用户自定义函数依赖 -->
37. <script type="text/javascript" src="js/udf/base.js"></script>
38. <script type="text/javascript" src="pie\_source/loader.js"></script>
39. */\*\**
40. \* pie\_source.js
41. \*/
42. function init\_echart\_pie\_source(container) {
43. var myChart1 = echarts.init(
44. document.getElementById(container),
45. window.gTheme
46. );
47. option = {
48. title: {
49. text: "在招岗位经验要求分析",
50. top: 0,
51. left: "center",
52. textStyle: {
53. *// color: "#17c0ff",*
54. color: "orange",
55. fontSize: "12",
56. },
57. },
58. tooltip: {
59. trigger: "item",
60. formatter: "{a} <br/>{b} : {c} ({d}%)",
61. },
62. legend: {
63. *// orient: 'vertical',*
64. *// left: 'right',*
65. top: "75%",
66. itemWidth: 10,
67. itemHeight: 10,
68. data: [],
69. textStyle: {
70. color: "rgba(255,255,255,.5)",
71. *//fontSize: "10",*
72. },
73. },
74. series: [
75. {
76. type: "pie",
77. radius: "50%",
78. center: ["50%", "40%"],
79. *// 环形图的组成部分不显示文字*
80. label: { show: false },
81. data: [],
82. emphasis: {
83. itemStyle: {
84. shadowBlur: 10,
85. shadowOffsetX: 0,
86. shadowColor: "rgba(0, 0, 0, 0.5)",
87. },
88. },
89. },
90. ],
91. };
92. *// 使用刚指定的配置项和数据显示图表。*
93. myChart1.setOption(option);
94. *// 必须*
95. window.addEventListener("resize", function () {
96. myChart1.resize();
97. });
98. }
99. function async\_echart\_pie\_source(container, filename) {
100. *// 异步加载数据*
101. $.getJSON(filename).done(function (data) {
102. var myChart = echarts.init(document.getElementById(container));
103. var legendData = [];
104. for (var i = 0; i < data.length; i++) {
105. legendData.push(data[i].name);
106. }
107. myChart.setOption({
108. legend: { data: legendData },
109. series: [
110. {
111. data: data,
112. },
113. ],
114. });
115. }); *//end $.getJSON*
116. }
117. *// ajax读取后端数据，传递参数为容器元素位置与后端接口url*
118. function async\_echart\_bar\_horizontal(container, url) {
119. *// 确保容器元素存在*
120. var containerElement = document.getElementById(container);
121. if (!containerElement) {
122. console.error('Container element not found:', container);
123. return;
124. }
125. $.ajax({
126. url: url, *// 后端接口的URL*
127. dataType: 'json', *// 期望返回的数据类型为JSON*
128. success: function (data) {
129. *// 初始化ECharts实例*
130. var myChart = echarts.init(containerElement, window.gTheme);
131. *// 使用从后端接口获取的数据设置图表选项*
132. *// 确保数据结构与ECharts兼容*
133. if (data && data.yAxis && data.yAxis.data && data.series && data.series[0] && data.series[0].data) {
134. myChart.setOption({
135. yAxis: {
136. data: data.yAxis.data
137. },
138. series: [{
139. data: data.series[0].data
140. }]
141. });
142. } else {
143. console.error('Invalid data structure received from server:', data);
144. }
145. },
146. error: function (jqXHR, textStatus, errorThrown) {
147. *// 处理请求失败的情况*
148. console.error("Error fetching data from server: " + textStatus, errorThrown);
149. }
150. }); *// end $.ajax*
151. }
152. */\*\**
153. \* loader.js
154. \*/
155. *// 加载主题，必须在echarts.init之前。*
156. var gTheme = getQueryVariable("theme");
157. themeUrl = "theme/js/" + gTheme + ".js";
158. addScript(themeUrl);
159. *// 加载依赖文件*
160. var pie\_source = "pie\_source/";
161. addScript(pie\_source + "pie\_source.js");
162. $(document).ready(function () {
163. var container = "lo\_3";
164. init\_echart\_pie\_source(container);
165. *// 定时1s执行数据更新函数*
166. setInterval(function () {
167. async\_echart\_pie\_source(container, pie\_source + "pie\_source.json");
168. }, 1000);
169. });

**12.在招岗位学历要求分析页面代码**

1. <!-- 在招岗位学历要求分析页面代码 -->
2. <html>
3. <head>
4. <title>YYDataV 数据可视化案例</title>
5. <meta charset="utf-8">
6. <meta name="viewport" content="width=device-width, initial-scale=1">
7. </head>
8. <link href="css/bootstrap-combined.min.css" rel="stylesheet">
9. <link href="css/layoutit.css" rel="stylesheet">
10. <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
11. </script>
12. <!-- 官方库依赖 -->
13. <script type="text/javascript" src="js/bootstrap/bootstrap.min.js"></script>
14. <script type="text/javascript" src="js/jquery/jquery-3.6.0.min.js"></script>
15. <script type="text/javascript" src="js/vue/vue.js"></script>
16. <script type="text/javascript" src="js/vue/dv.js"></script>
17. <script type="text/javascript" src="js/echarts/echarts.min.js"></script>
18. <body style="background-color:rgb(11, 18, 39)">
19. <div id="vue\_app" class="container-fluid">
20. <div class="row" style="height: 100%;">
21. <div class="col-6">
22. <div id="container\_l1" style="height: 100%; "></div>
23. </div>
24. <div class="col-6">
25. <div id="container\_l2" style="height: 100%; "></div>
26. </div>
27. </div>
28. </div>
29. </body>
30. <script>
31. var vpp = new Vue({
32. el: "#vue\_app",
33. data() { }
34. })
35. </script>
36. <!-- 用户自定义函数依赖 -->
37. <script type="text/javascript" src="js/udf/base.js"></script>
38. <script type="text/javascript" src="pie\_source\_rose/loader.js"></script>
39. */\*\**
40. \* pie\_source\_rose.js
41. \*/
42. function init\_echart\_pie\_source\_rose(container) {
43. var myChart1 = echarts.init(
44. document.getElementById(container),
45. window.gTheme
46. );
47. option = {
48. title: {
49. text: "在招岗位学历要求分析",
50. *//top: "5%",*
51. *//left: "2%",*
52. textStyle: {
53. *// color: "#17c0ff",*
54. color: "orange",
55. fontSize: "12",
56. },
57. },
58. tooltip: {
59. trigger: "item",
60. formatter: "{a} <br/>{b} : {c} ({d}%)",
61. },
62. legend: {
63. orient: "vertical",
64. top: "75%",
65. itemWidth: 10,
66. itemHeight: 10,
67. data: [],
68. textStyle: {
69. color: "rgba(255,255,255,.5)",
70. *////fontSize: "10",*
71. },
72. },
73. series: [
74. {
75. type: "pie",
76. roseType: "radius",
77. radius: ["5%", "50%"],
78. center: ["50%", "40%"],
79. *// 环形图的组成部分不显示文字*
80. label: { show: false },
81. data: [],
82. emphasis: {
83. itemStyle: {
84. shadowBlur: 10,
85. shadowOffsetX: 0,
86. shadowColor: "rgba(0, 0, 0, 0.5)",
87. },
88. },
89. },
90. ],
91. };
92. *// 使用刚指定的配置项和数据显示图表。*
93. myChart1.setOption(option);
94. *// 必须*
95. window.addEventListener("resize", function () {
96. myChart1.resize();
97. });
98. }
99. function async\_echart\_pie\_source\_rose(container, filename) {
100. *// 异步加载数据*
101. $.getJSON(filename).done(function (data) {
102. var myChart = echarts.init(document.getElementById(container));
103. var legendData = [];
104. for (var i = 0; i < data.length; i++) {
105. legendData.push(data[i].name);
106. }
107. myChart.setOption({
108. legend: { data: legendData },
109. series: [
110. {
111. data: data,
112. },
113. ],
114. });
115. }); *//end $.getJSON*
116. }
117. *// ajax读取后端数据，传递参数为容器元素位置与后端接口url*
118. function async\_echart\_bar\_horizontal(container, url) {
119. *// 确保容器元素存在*
120. var containerElement = document.getElementById(container);
121. if (!containerElement) {
122. console.error('Container element not found:', container);
123. return;
124. }
125. $.ajax({
126. url: url, *// 后端接口的URL*
127. dataType: 'json', *// 期望返回的数据类型为JSON*
128. success: function (data) {
129. *// 初始化ECharts实例*
130. var myChart = echarts.init(containerElement, window.gTheme);
131. *// 使用从后端接口获取的数据设置图表选项*
132. *// 确保数据结构与ECharts兼容*
133. if (data && data.yAxis && data.yAxis.data && data.series && data.series[0] && data.series[0].data) {
134. myChart.setOption({
135. yAxis: {
136. data: data.yAxis.data
137. },
138. series: [{
139. data: data.series[0].data
140. }]
141. });
142. } else {
143. console.error('Invalid data structure received from server:', data);
144. }
145. },
146. error: function (jqXHR, textStatus, errorThrown) {
147. *// 处理请求失败的情况*
148. console.error("Error fetching data from server: " + textStatus, errorThrown);
149. }
150. }); *// end $.ajax*
151. }
152. */\*\**
153. \* loader.js
154. \*/
155. *// 加载主题，必须在echarts.init之前。*
156. var gTheme = getQueryVariable("theme");
157. themeUrl = "theme/js/" + gTheme + ".js";
158. addScript(themeUrl);
159. *// 加载依赖文件*
160. var pie\_source\_rose = "pie\_source\_rose/";
161. addScript(pie\_source\_rose + "pie\_source\_rose.js");
162. $(document).ready(function () {
163. var container = "lo\_4";
164. init\_echart\_pie\_source\_rose(container);
165. *// 定时1s执行数据更新函数*
166. setInterval(function () {
167. async\_echart\_pie\_source\_rose(
168. container,
169. pie\_source\_rose + "pie\_source\_rose.json"
170. );
171. }, 1000);
172. });

**13.城市线级招牌数量分析页面代码**

1. */\*\**
2. \* 城市线级招牌数量分析代码
3. \*/
4. *// 加载主题，必须在echarts.init之前。*
5. var gTheme = getQueryVariable("theme");
6. themeUrl = "theme/js/" + gTheme + ".js";
7. addScript(themeUrl);
8. *// 加载依赖文件*
9. var path\_funnel = "funnel/";
10. addScript(path\_funnel + "funnel.js");
11. $(document).ready(function () {
12. var container = "lo\_2";
13. init\_echart\_funnel(container);
14. *// 定时1s执行数据更新函数*
15. setInterval(function () {
16. async\_echart\_funnel(container, path\_funnel + "funnel.json");
17. }, 1000);
18. });
19. function init\_echart\_funnel(container) {
20. var chartDom = document.getElementById(container);
21. var myChart = echarts.init(chartDom, window.gTheme);
22. var option = {
23. title: {
24. text: "城市线级招聘数量分析",
25. *//top: "5%",*
26. *//left: "2%",*
27. textStyle: {
28. *// color: "#17c0ff",*
29. color: "orange",
30. fontSize: "12",
31. },
32. },
33. tooltip: {
34. trigger: "item",
35. formatter: "{b} : {c}",
36. },
37. series: [
38. {
39. type: "funnel",
40. left: "15%",
41. top: "20%",
42. width: "55%",
43. height: "65%",
44. sort: "ascending",
45. *// 坐标轴显示的字样式*
46. label: {
47. position: "right",
48. textStyle: {
49. color: "rgba(255, 255, 255, 0.8)",
50. },
51. },
52. data: [],
53. },
54. ],
55. };
56. window.addEventListener("resize", function () {
57. myChart.resize();
58. });
59. myChart.setOption(option);
60. }
61. function async\_echart\_funnel(container, filename) {
62. $.getJSON(filename).done(function (data) {
63. var myChart = echarts.init(document.getElementById(container));
64. myChart.setOption({
65. series: [{ data: data }],
66. });
67. }); *//end $.getJSON*
68. }
69. *// ajax读取后端数据，传递参数为容器元素位置与后端接口url*
70. function async\_echart\_bar\_horizontal(container, url) {
71. *// 确保容器元素存在*
72. var containerElement = document.getElementById(container);
73. if (!containerElement) {
74. console.error('Container element not found:', container);
75. return;
76. }
77. $.ajax({
78. url: url, *// 后端接口的URL*
79. dataType: 'json', *// 期望返回的数据类型为JSON*
80. success: function (data) {
81. *// 初始化ECharts实例*
82. var myChart = echarts.init(containerElement, window.gTheme);
83. *// 使用从后端接口获取的数据设置图表选项*
84. *// 确保数据结构与ECharts兼容*
85. if (data && data.yAxis && data.yAxis.data && data.series && data.series[0] && data.series[0].data) {
86. myChart.setOption({
87. yAxis: {
88. data: data.yAxis.data
89. },
90. series: [{
91. data: data.series[0].data
92. }]
93. });
94. } else {
95. console.error('Invalid data structure received from server:', data);
96. }
97. },
98. error: function (jqXHR, textStatus, errorThrown) {
99. *// 处理请求失败的情况*
100. console.error("Error fetching data from server: " + textStatus, errorThrown);
101. }
102. }); *// end $.ajax*
103. }

**14.全国就业岗位分析页面代码**

1. /\*\*
2. \* 全国就业岗位分析代码
3. \*/
4. // 加载主题，必须在echarts.init之前。
5. var gTheme = getQueryVariable("theme");
6. themeUrl = "theme/js/" + gTheme + ".js";
7. addScript(themeUrl);
8. // 加载地图依赖文件
9. var path\_map\_china\_map = "map\_china\_map/";
10. addScript(path\_map\_china\_map + "bmap.min.js");
11. addScript(path\_map\_china\_map + "china.js");
12. addScript(path\_map\_china\_map + "init.js");
13. // event事件，自动高亮地图各地区
14. // 高亮表盘事件
15. function emphasisData(container) {
16. var currentIndex = -1;
17. setInterval(function () {
18. var myChart = echarts.init(document.getElementById(container));
19. var dataLen = 0;
20. try {
21. dataLen = myChart.getOption().dataset[0]["source"].length;
22. } catch {
23. try {
24. dataLen = myChart.getOption().series[0]["data"].length;
25. } catch {
26. return;
27. }
28. }
29. // 取消之前高亮的图形
30. myChart.dispatchAction({
31. type: "downplay",
32. seriesIndex: 0,
33. dataIndex: currentIndex,
34. });
35. currentIndex = (currentIndex + 1) % dataLen;
36. // 高亮当前图形
37. myChart.dispatchAction({
38. type: "highlight",
39. seriesIndex: 0,
40. dataIndex: currentIndex,
41. });
42. // 显示 tooltip
43. myChart.dispatchAction({
44. type: "showTip",
45. seriesIndex: 0,
46. dataIndex: currentIndex,
47. });
48. }, 1000);
49. }
50. $(document).ready(function () {
51. var container = "container\_8";
52. init\_echart\_map\_china\_map(container);
53. async\_echart\_china(
54. container,
55. path\_map\_china\_map + "map\_china\_map.json",
56. "confirm"
57. );
58. // 定时5min执行数据更新函数
59. setInterval(function () {
60. async\_echart\_china(
61. container,
62. path\_map\_china\_map + "map\_china\_map.json",
63. "confirm"
64. );
65. }, 300000);
66. emphasisData(container);
67. });
68. /\*
69. \* Licensed to the Apache Software Foundation (ASF) under one
70. \* or more contributor license agreements.  See the NOTICE file
71. \* distributed with this work for additional information
72. \* regarding copyright ownership.  The ASF licenses this file
73. \* to you under the Apache License, Version 2.0 (the
74. \* "License"); you may not use this file except in compliance
75. \* with the License.  You may obtain a copy of the License at
76. \*
77. \*   http://www.apache.org/licenses/LICENSE-2.0
78. \*
79. \* Unless required by applicable law or agreed to in writing,
80. \* software distributed under the License is distributed on an
81. \* "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
82. \* KIND, either express or implied.  See the License for the
83. \* specific language governing permissions and limitations
84. \* under the License.
85. \*/
86. !function(e,t){"object"==typeof exports&&"undefined"!=typeof module?t(exports,require("echarts")):"function"==typeof define&&define.amd?define(["exports","echarts"],t):t((e=e||self).bmap={},e.echarts)}(this,function(e,d){"use strict";function l(e,t){this.\_bmap=e,this.dimensions=["lng","lat"],this.\_mapOffset=[0,0],this.\_api=t,this.\_projection=new BMap.MercatorProjection}function t(a,r){return r=r||[0,0],d.util.map([0,1],function(e){var t=r[e],o=a[e]/2,n=[],i=[];return n[e]=t-o,i[e]=t+o,n[1-e]=i[1-e]=r[1-e],Math.abs(this.dataToPoint(n)[e]-this.dataToPoint(i)[e])},this)}var c;function f(e){for(var t in e)if(e.hasOwnProperty(t))return;return 1}l.prototype.dimensions=["lng","lat"],l.prototype.setZoom=function(e){this.\_zoom=e},l.prototype.setCenter=function(e){this.\_center=this.\_projection.lngLatToPoint(new BMap.Point(e[0],e[1]))},l.prototype.setMapOffset=function(e){this.\_mapOffset=e},l.prototype.getBMap=function(){return this.\_bmap},l.prototype.dataToPoint=function(e){var t=new BMap.Point(e[0],e[1]),e=this.\_bmap.pointToOverlayPixel(t),t=this.\_mapOffset;return[e.x-t[0],e.y-t[1]]},l.prototype.pointToData=function(e){var t=this.\_mapOffset;return[(e=this.\_bmap.overlayPixelToPoint({x:e[0]+t[0],y:e[1]+t[1]})).lng,e.lat]},l.prototype.getViewRect=function(){var e=this.\_api;return new d.graphic.BoundingRect(0,0,e.getWidth(),e.getHeight())},l.prototype.getRoamTransform=function(){return d.matrix.create()},l.prototype.prepareCustoms=function(){var e=this.getViewRect();return{coordSys:{type:"bmap",x:e.x,y:e.y,width:e.width,height:e.height},api:{coord:d.util.bind(this.dataToPoint,this),size:d.util.bind(t,this)}}},l.dimensions=l.prototype.dimensions,l.create=function(e,p){var s,m=p.getDom();e.eachComponent("bmap",function(e){var t,o=p.getZr().painter,n=o.getViewportRoot();if("undefined"==typeof BMap)throw new Error("BMap api is not loaded");function i(e){this.\_root=e}if(c=c||((i.prototype=new BMap.Overlay).initialize=function(e){return e.getPanes().labelPane.appendChild(this.\_root),this.\_root},i.prototype.draw=function(){},i),s)throw new Error("Only one bmap component can exist");e.\_\_bmap||((a=m.querySelector(".ec-extension-bmap"))&&(n.style.left="0px",n.style.top="0px",m.removeChild(a)),(a=document.createElement("div")).className="ec-extension-bmap",a.style.cssText="position:absolute;width:100%;height:100%",m.appendChild(a),(r=e.get("mapOptions"))&&delete(r=d.util.clone(r)).mapType,t=e.\_\_bmap=new BMap.Map(a,r),a=new c(n),t.addOverlay(a),o.getViewportRootOffset=function(){return{offsetLeft:0,offsetTop:0}}),t=e.\_\_bmap;var a,r=e.get("center"),n=e.get("zoom");r&&n&&(a=t.getCenter(),o=t.getZoom(),e.centerOrZoomChanged([a.lng,a.lat],o)&&(o=new BMap.Point(r[0],r[1]),t.centerAndZoom(o,n))),(s=new l(t,p)).setMapOffset(e.\_\_mapOffset||[0,0]),s.setZoom(n),s.setCenter(r),e.coordinateSystem=s}),e.eachSeries(function(e){"bmap"===e.get("coordinateSystem")&&(e.coordinateSystem=s)})},d.extendComponentModel({type:"bmap",getBMap:function(){return this.\_\_bmap},setCenterAndZoom:function(e,t){this.option.center=e,this.option.zoom=t},centerOrZoomChanged:function(e,t){var o,n=this.option;return o=e,e=n.center,!(o&&e&&o[0]===e[0]&&o[1]===e[1]&&t===n.zoom)},defaultOption:{center:[104.114129,37.550339],zoom:5,mapStyle:{},mapStyleV2:{},mapOptions:{},roam:!1}}),d.extendComponentView({type:"bmap",render:function(r,e,p){var s=!0,t=r.getBMap(),m=p.getZr().painter.getViewportRoot(),l=r.coordinateSystem,o=function(e,t){var o,n,i,a;s||(a=m.parentNode.parentNode.parentNode,o=[-parseInt(a.style.left,10)||0,-parseInt(a.style.top,10)||0],n=m.style,i=o[0]+"px",a=o[1]+"px",n.left!==i&&(n.left=i),n.top!==a&&(n.top=a),l.setMapOffset(o),r.\_\_mapOffset=o,p.dispatchAction({type:"bmapRoam",animation:{duration:0}}))};function n(){s||p.dispatchAction({type:"bmapRoam",animation:{duration:0}})}t.removeEventListener("moving",this.\_oldMoveHandler),t.removeEventListener("moveend",this.\_oldMoveHandler),t.removeEventListener("zoomend",this.\_oldZoomEndHandler),t.addEventListener("moving",o),t.addEventListener("moveend",o),t.addEventListener("zoomend",n),this.\_oldMoveHandler=o,this.\_oldZoomEndHandler=n;var i=r.get("roam");i&&"scale"!==i?t.enableDragging():t.disableDragging(),i&&"move"!==i?(t.enableScrollWheelZoom(),t.enableDoubleClickZoom(),t.enablePinchToZoom()):(t.disableScrollWheelZoom(),t.disableDoubleClickZoom(),t.disablePinchToZoom());var a=r.\_\_mapStyle,o=r.get("mapStyle")||{},i=JSON.stringify(o);JSON.stringify(a)!==i&&(f(o)||t.setMapStyle(d.util.clone(o)),r.\_\_mapStyle=JSON.parse(i));a=r.\_\_mapStyle2,o=r.get("mapStyleV2")||{},i=JSON.stringify(o);JSON.stringify(a)!==i&&(f(o)||t.setMapStyleV2(d.util.clone(o)),r.\_\_mapStyle2=JSON.parse(i)),s=!1}}),d.registerCoordinateSystem("bmap",l),d.registerAction({type:"bmapRoam",event:"bmapRoam",update:"updateLayout"},function(e,t){t.eachComponent("bmap",function(e){var t=e.getBMap(),o=t.getCenter();e.setCenterAndZoom([o.lng,o.lat],t.getZoom())})});e.version="1.0.0",Object.defineProperty(e,"\_\_esModule",{value:!0})});
87. (function (root, factory) {
88. if (typeof define === 'function' && define.amd) {
89. // AMD. Register as an anonymous module.
90. define(['exports', 'echarts'], factory);
91. } else if (typeof exports === 'object' && typeof exports.nodeName !== 'string') {
92. // CommonJS
93. factory(exports, require('echarts'));
94. } else {
95. // Browser globals
96. factory({}, root.echarts);
97. }
98. }(this, function (exports, echarts) {
99. var log = function (msg) {
100. if (typeof console !== 'undefined') {
101. console && console.error && console.error(msg);
102. }
103. }
104. if (!echarts) {
105. log('ECharts is not Loaded');
106. return;
107. }
108. if (!echarts.registerMap) {
109. log('ECharts Map is not loaded')
110. return;
111. }
112. }));
113. function init\_echart\_map\_china\_map(container) {
114. // 基于准备好的dom，初始化echarts实例
115. var myChart = echarts.init(document.getElementById(container), window.gTheme);
116. var data = [];
117. var geoCoordMap = {};
118. option = {
119. title: {
120. text: "全国就业岗位分析",
121. top: "2%",
122. left: "center",
123. textStyle: {
124. color: "hsl(200, 86%, 48%)",
125. // color: "orange",
126. fontSize: "20",
127. },
128. },
129. tooltip: {
130. trigger: "item",
131. formatter: function (params) {
132. console.log(params);
133. value = 0;
134. if (params.value) value = params.value;
135. return params.seriesName + "<br>" + params.name + " : " + value + "万";
136. },
137. },
138. visualMap: {
139. type: "piecewise",
140. splitNumber: 5,
141. pieces: [
142. { gt: 10000 },
143. { gt: 1000, lte: 9999 },
144. { gt: 500, lte: 999 },
145. { gt: 100, lte: 499 },
146. { gt: 10, lte: 99 },
147. { gt: 1, lte: 9 },
148. { lte: 0 },
149. ],
150. textStyle: {
151. color: "#fff",
152. },
153. top: "bottom",
154. },
155. geo: [
156. {
157. map: "china",
158. layoutCenter: ["50%", "50%"],
159. zoom: 1,
160. roam: true,
161. // 地图放大或缩小的尺寸
162. layoutSize: "100%",
163. selectedMode: "single",
164. label: {
165. emphasis: {
166. show: false,
167. },
168. },
169. // 地图区域的多边形 图形样式。
170. itemStyle: {
171. // 图形阴影颜色。支持的格式同color。
172. shadowColor: "black",
173. // 图形阴影的模糊大小。
174. shadowBlur: 10,
175. // 阴影水平方向上的偏移距离。
176. shadowOffsetX: 2,
177. // 阴影垂直方向上的偏移距离。
178. shadowOffsetY: 2,
179. },
180. // 高亮状态下的多边形和标签样式。
181. emphasis: {
182. itemStyle: {
183. borderWidth: 3,
184. borderColor: "purple",
185. // areaColor: "#ffdead",
186. },
187. },
188. },
189. ],
190. series: {
191. name: "累计就业岗位",
192. type: "map",
193. coordinateSystem: "geo",
194. geoIndex: 0,
195. zoom: 1.2,
196. roam: true,
197. },
198. };
199. myChart.setOption(option);
200. window.addEventListener("resize", function () {
201. myChart.resize();
202. });
203. }
204. // 地图加载数据
205. function async\_echart\_china(container, filename, section) {
206. $.getJSON(filename).done(function (data) {
207. var myChart = echarts.init(document.getElementById(container));
208. series = "";
209. data\_section = [];
210. if (section == "confirmAdd") {
211. series = "新增就业岗位";
212. data\_section = data.map((item) => {
213. return { name: item.name, value: item.confirmAdd };
214. });
215. }
216. if (section == "confirm") {
217. series = "累计就业岗位";
218. data\_section = data.map((item) => {
219. return { name: item.name, value: item.confirm };
220. });
221. }
222. if (section == "nowConfirm") {
223. series = "现有就业岗位";
224. data\_section = data.map((item) => {
225. return { name: item.name, value: item.nowConfirm };
226. });
227. }
228. myChart.setOption({
229. series: {
230. name: series,
231. data: data\_section,
232. },
233. });
234. });
235. }
236. // ajax读取后端数据，传递参数为容器元素位置与后端接口url
237. function async\_echart\_bar\_horizontal(container, url) {
238. // 确保容器元素存在
239. var containerElement = document.getElementById(container);
240. if (!containerElement) {
241. console.error('Container element not found:', container);
242. return;
243. }
244. $.ajax({
245. url: url, // 后端接口的URL
246. dataType: 'json', // 期望返回的数据类型为JSON
247. success: function (data) {
248. // 初始化ECharts实例
249. var myChart = echarts.init(containerElement, window.gTheme);
250. // 使用从后端接口获取的数据设置图表选项
251. // 确保数据结构与ECharts兼容
252. if (data && data.yAxis && data.yAxis.data && data.series && data.series[0] && data.series[0].data) {
253. myChart.setOption({
254. yAxis: {
255. data: data.yAxis.data
256. },
257. series: [{
258. data: data.series[0].data
259. }]
260. });
261. } else {
262. console.error('Invalid data structure received from server:', data);
263. }
264. },
265. error: function (jqXHR, textStatus, errorThrown) {
266. // 处理请求失败的情况
267. console.error("Error fetching data from server: " + textStatus, errorThrown);
268. }
269. }); // end $.ajax
270. }

**15.城市行业岗位类型统计页面代码**

1. <!-- 城市行业岗位类型统计页面代码 -->
2. <html>
3. <head>
4. <title>YYDataV 数据可视化案例</title>
5. <meta charset="utf-8">
6. <meta name="viewport" content="width=device-width, initial-scale=1">
7. </head>
8. <link href="css/bootstrap-combined.min.css" rel="stylesheet">
9. <link href="css/layoutit.css" rel="stylesheet">
10. <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
11. </script>
12. <!-- 官方库依赖 -->
13. <script type="text/javascript" src="js/bootstrap/bootstrap.min.js"></script>
14. <script type="text/javascript" src="js/jquery/jquery-3.6.0.min.js"></script>
15. <script type="text/javascript" src="js/vue/vue.js"></script>
16. <script type="text/javascript" src="js/vue/dv.js"></script>
17. <script type="text/javascript" src="js/echarts/echarts.min.js"></script>
18. <body style=" background-color:rgb(11, 18, 39)">
19. <div id="vue\_app" class="container-fluid">
20. <div id="container\_l7" style="height: 100%; width: 100%;">
21. </div>
22. </div>
23. </body>
24. <script>
25. var vpp = new Vue({
26. el: "#vue\_app",
27. data() { }
28. })
29. </script>
30. <!-- 用户自定义函数依赖 -->
31. <script type="text/javascript" src="js/udf/base.js"></script>
32. <script type="text/javascript" src="line\_area\_chart/loader.js"></script>
33. */\*\**
34. \* line\_area\_chart.js
35. \*/
36. function init\_echart\_line\_area\_chart(container) {
37. var chartDom = document.getElementById(container);
38. var myChart = echarts.init(chartDom, window.gTheme);
39. var option;
40. option = {
41. title: {
42. text: "城市行业岗位类型统计",
43. *// top: 0,*
44. *// left: "center",*
45. textStyle: {
46. *// color: "#17c0ff",*
47. color: "orange",
48. fontSize: "12",
49. },
50. },
51. grid: {
52. left: "3%",
53. right: "3%",
54. bottom: "3%",
55. top: "25%",
56. containLabel: true,
57. },
58. tooltip: {
59. trigger: "axis",
60. axisPointer: {
61. *// Use axis to trigger tooltip*
62. type: "shadow", *// 'shadow' as default; can also be 'line' or 'shadow'*
63. },
64. },
65. xAxis: {
66. type: "category",
67. boundaryGap: false,
68. axisLabel: {
69. textStyle: {
70. color: "rgba(255,255,255,.8)",
71. *// fontSize: 14,*
72. },
73. },
74. axisLine: {
75. lineStyle: {
76. color: "rgba(255,255,255,.2)",
77. },
78. },
79. splitLine: {
80. lineStyle: {
81. color: "rgba(255,255,255,.1)",
82. },
83. },
84. },
85. yAxis: {
86. type: "value",
87. axisLabel: {
88. textStyle: {
89. color: "rgba(255,255,255,.8)",
90. *// fontSize: 14,*
91. },
92. },
93. axisLine: {
94. lineStyle: {
95. color: "rgba(255,255,255,.2)",
96. },
97. },
98. splitLine: {
99. lineStyle: {
100. color: "rgba(255,255,255,.1)",
101. },
102. },
103. },
104. series: [
105. {
106. data: [],
107. type: "line",
108. areaStyle: {},
109. },
110. ],
111. };
112. option && myChart.setOption(option);
113. window.addEventListener("resize", function () {
114. myChart.resize();
115. });
116. }
117. function async\_echart\_line\_area\_chart(container, filename) {
118. $.getJSON(filename).done(function (data) {
119. var myChart = echarts.init(document.getElementById(container));
120. myChart.setOption({
121. xAxis: data["xAxis"],
122. series: data["series"],
123. });
124. }); *//end $.getJSON*
125. }
126. *// ajax读取后端数据，传递参数为容器元素位置与后端接口url*
127. function async\_echart\_bar\_horizontal(container, url) {
128. *// 确保容器元素存在*
129. var containerElement = document.getElementById(container);
130. if (!containerElement) {
131. console.error('Container element not found:', container);
132. return;
133. }
134. $.ajax({
135. url: url, *// 后端接口的URL*
136. dataType: 'json', *// 期望返回的数据类型为JSON*
137. success: function (data) {
138. *// 初始化ECharts实例*
139. var myChart = echarts.init(containerElement, window.gTheme);
140. *// 使用从后端接口获取的数据设置图表选项*
141. *// 确保数据结构与ECharts兼容*
142. if (data && data.yAxis && data.yAxis.data && data.series && data.series[0] && data.series[0].data) {
143. myChart.setOption({
144. yAxis: {
145. data: data.yAxis.data
146. },
147. series: [{
148. data: data.series[0].data
149. }]
150. });
151. } else {
152. console.error('Invalid data structure received from server:', data);
153. }
154. },
155. error: function (jqXHR, textStatus, errorThrown) {
156. *// 处理请求失败的情况*
157. console.error("Error fetching data from server: " + textStatus, errorThrown);
158. }
159. }); *// end $.ajax*
160. }
161. */\*\**
162. \* loader.js
163. \*/
164. *// 加载主题，必须在echarts.init之前。*
165. var gTheme = getQueryVariable("theme");
166. themeUrl = "" + "theme/js/" + gTheme + ".js";
167. addScript(themeUrl);
168. *// 加载依赖文件*
169. var line\_area\_chart = "" + "line\_area\_chart/";
170. addScript(line\_area\_chart + "line\_area\_chart.js");
171. $(document).ready(function () {
172. var container = "lo\_7";
173. init\_echart\_line\_area\_chart(container);
174. *// 定时1s执行数据更新函数*
175. setInterval(function () {
176. async\_echart\_line\_area\_chart(
177. container,
178. line\_area\_chart + "line\_area\_chart.json"
179. );
180. }, 1000);
181. });

**16.省份动态轮播展示功能代码**

1. */\*\**
2. \* 省份动态轮播展示功能代码
3. \*/
4. package com.datav.bigscreen.controller;
5. import org.springframework.web.bind.annotation.RequestMapping;
6. import org.springframework.web.bind.annotation.RestController;
7. import org.springframework.web.bind.annotation.PathVariable;
8. import org.springframework.web.servlet.support.ServletUriComponentsBuilder;
9. import org.springframework.web.util.UriComponentsBuilder;
10. import java.io.\*;
11. */\*\**
12. \* 返回json格式
13. \* @author user
14. \*
15. \*/
16. @RestController
17. @RequestMapping(value = {"/map\_china\_map", "/line\_area\_chart", "/pie\_source"})
18. public class Process {
19. @RequestMapping("/{filename}.json")
20. public String json(@PathVariable("filename") String filename) throws Exception {
21. UriComponentsBuilder builder = ServletUriComponentsBuilder.fromCurrentRequest();
22. String requestedValue = builder.buildAndExpand().getPath();
23. System.out.println(requestedValue);
24. System.out.println(filename);
25. ChangeJSON("./src/main/resources/static" + requestedValue);
26. String jsonStr = readJSON("./src/main/resources/static" + requestedValue);
27. System.out.println(jsonStr);
28. return jsonStr;
29. }
30. public void ChangeJSON(String filename) throws Exception {
31. System.out.println("正在访问:" + filename);
32. if (filename.contains("map\_china\_map.json"))
33. {
34. int max=800, min=300;
35. String objectStr = String.format("[{\"name\": \"上海\", \"confirmAdd\": %d, \"confirm\": %d, \"nowConfirm\": %d}, {\"name\": \"云南\", \"confirmAdd\": %d, \"confirm\": %d, \"nowConfirm\": %d}, {\"name\": \"内蒙古\", \"confirmAdd\": %d, \"confirm\": %d, \"nowConfirm\": %d}, {\"name\": \"北京\", \"confirmAdd\": 373, \"confirm\": 568, \"nowConfirm\": 600}, {\"name\": \"吉林\", \"confirmAdd\": 646, \"confirm\": 396, \"nowConfirm\": 512}, {\"name\": \"四川\", \"confirmAdd\": 749, \"confirm\": 455, \"nowConfirm\": 146}, {\"name\": \"天津\", \"confirmAdd\": 233, \"confirm\": 596, \"nowConfirm\": 878}, {\"name\": \"安徽\", \"confirmAdd\": 595, \"confirm\": 637, \"nowConfirm\": 746}, {\"name\": \"山东\", \"confirmAdd\": 148, \"confirm\": 201, \"nowConfirm\": 850}, {\"name\": \"山西\", \"confirmAdd\": 211, \"confirm\": 481, \"nowConfirm\": 228}, {\"name\": \"广东\", \"confirmAdd\": 136, \"confirm\": 360, \"nowConfirm\": 191}, {\"name\": \"广西\", \"confirmAdd\": 387, \"confirm\": 319, \"nowConfirm\": 324}, {\"name\": \"江苏\", \"confirmAdd\": 870, \"confirm\": 573, \"nowConfirm\": 435}, {\"name\": \"江西\", \"confirmAdd\": 576, \"confirm\": 323, \"nowConfirm\": 124}, {\"name\": \"河北\", \"confirmAdd\": 405, \"confirm\": 542, \"nowConfirm\": 878}, {\"name\": \"河南\", \"confirmAdd\": 210, \"confirm\": 550, \"nowConfirm\": 188}, {\"name\": \"浙江\", \"confirmAdd\": 336, \"confirm\": 549, \"nowConfirm\": 764}, {\"name\": \"海南\", \"confirmAdd\": 758, \"confirm\": 411, \"nowConfirm\": 251}, {\"name\": \"湖北\", \"confirmAdd\": 814, \"confirm\": 264, \"nowConfirm\": 113}, {\"name\": \"湖南\", \"confirmAdd\": 872, \"confirm\": 337, \"nowConfirm\": 181}, {\"name\": \"福建\", \"confirmAdd\": 378, \"confirm\": 749, \"nowConfirm\": 741}, {\"name\": \"贵州\", \"confirmAdd\": 823, \"confirm\": 100, \"nowConfirm\": 603}, {\"name\": \"辽宁\", \"confirmAdd\": 173, \"confirm\": 140, \"nowConfirm\": 589}, {\"name\": \"陕西\", \"confirmAdd\": 546, \"confirm\": 206, \"nowConfirm\": 855}, {\"name\": \"青海\", \"confirmAdd\": 688, \"confirm\": 614, \"nowConfirm\": 480}, {\"name\": \"黑龙江\", \"confirmAdd\": 193, \"confirm\": 220, \"nowConfirm\": 666}, {\"name\": \"新疆\", \"confirmAdd\": 514, \"confirm\": 164, \"nowConfirm\": 113}, {\"name\": \"西藏\", \"confirmAdd\": 214, \"confirm\": 64, \"nowConfirm\": 13}, {\"name\": \"甘肃\", \"confirmAdd\": 414, \"confirm\": 174, \"nowConfirm\": 133}]",
36. (int) (Math.random()\*(max-min)+min),
37. (int) (Math.random()\*(max-min)+min),
38. (int) (Math.random()\*(max-min)+min),
39. (int) (Math.random()\*(max-min)+min),
40. (int) (Math.random()\*(max-min)+min),
41. (int) (Math.random()\*(max-min)+min),
42. (int) (Math.random()\*(max-min)+min),
43. (int) (Math.random()\*(max-min)+min),
44. (int) (Math.random()\*(max-min)+min));
45. System.out.println(objectStr);
46. StringWrite(filename, objectStr);
47. }else if (filename.contains("line\_area\_chart.json"))
48. {
49. int max=1000, min=100;
50. String objectStr = String.format("{\"xAxis\": {\"data\": [\"企业服务\", \"数据服务\", \"电子商务\", \"培训机构\", \"生活服务\", \"半导体\"]}, \"series\": [{\"data\": [%d, %d, %d, %d, %d, %d]}]}",
51. (int)(Math.random()\*(max-min)+min),
52. (int)(Math.random()\*(max-min)+min),
53. (int)(Math.random()\*(max-min)+min),
54. (int)(Math.random()\*(max-min)+min),
55. (int)(Math.random()\*(max-min)+min),
56. (int)(Math.random()\*(max-min)+min),
57. (int)(Math.random()\*(max-min)+min),
58. (int)(Math.random()\*(max-min)+min),
59. (int)(Math.random()\*(max-min)+min),
60. (int)(Math.random()\*(max-min)+min),
61. (int)(Math.random()\*(max-min)+min),
62. (int)(Math.random()\*(max-min)+min));
63. System.out.println(objectStr);
64. StringWrite(filename, objectStr);
65. }else if (filename.contains("pie\_source.json"))
66. {
67. int max=1000, min=100;
68. String objectStr = String.format("[{\"name\": \"1-3年\", \"value\": %d}, {\"name\": \"经验不限\", \"value\": %d}, {\"name\": \"7年以上\", \"value\": %d}, {\"name\": \"5-7年\", \"value\": %d}]",
69. (int)(Math.random()\*(max-min)+min),
70. (int)(Math.random()\*(max-min)+min),
71. (int)(Math.random()\*(max-min)+min),
72. (int)(Math.random()\*(max-min)+min));
73. System.out.println(objectStr);
74. StringWrite(filename, objectStr);
75. }
76. *//其它json请自行增加相关代码*
77. }
78. public void StringWrite(String filename, String content) throws Exception {
79. File jsonFile = new File(filename);
80. FileOutputStream outputStream = new FileOutputStream(jsonFile);
81. BufferedWriter bw = null;
82. try {
83. bw = new BufferedWriter(new OutputStreamWriter(outputStream, "UTF-8"));
84. bw.write("");
85. bw.write(content);
86. bw.close();
87. } catch (IOException e) {
88. e.printStackTrace();
89. } finally {
90. try {
91. bw.close();
92. } catch (IOException e) {
93. e.printStackTrace();
94. }
95. }
96. }
97. public static String readJSON(String fileName) throws Exception {
98. BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream(fileName)));
99. StringBuilder sb = new StringBuilder();
100. String line = "";
101. while ((line = reader.readLine()) != null) {
102. sb.append(line);
103. }
104. String result = sb.toString().replaceAll("\r\n", "").replaceAll(" +", "");
105. return result;
106. }
107. }