# 大数据项目开发实战

## 课程介绍

主要内容：演练城建住房大数据分析平台开发过程

每节课：40分钟

课时：30个（讲解+练习）

## 项目演示

## 学习目标

* 了解项目开发流程
* 了解项目开发过程中涉及到的工具
* 掌握Python开发Spark算子
* 掌握HDFS的实际应用
* 掌握Spark算子的常见应用
* 掌握数据可视化的开发方法

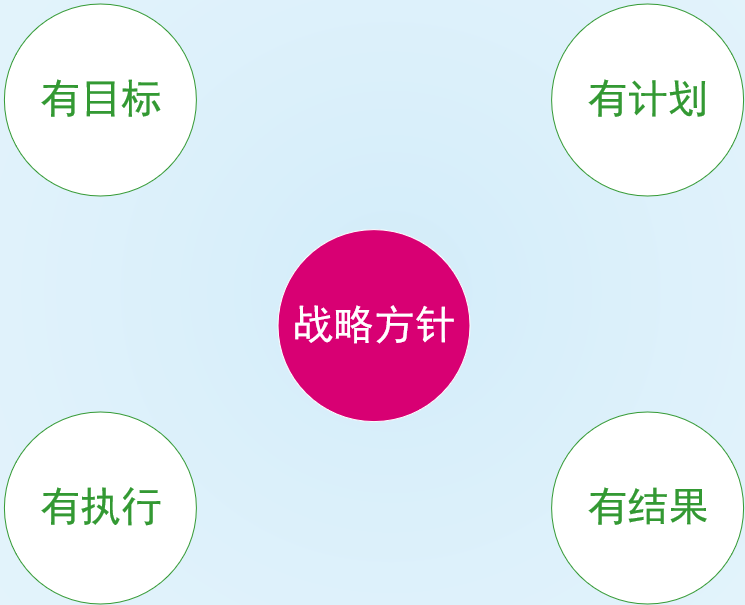
## 必备条件

完成一个项目

* 你不必拥有很强的技术能力
* 但是你应该有基本的学习能力，与干掉问题的决心
* 做出一个项目，干出一个毕业设计，我们要相信仅仅是时间问题
* 干项目，就是在干仗，要讲战略，要有战术，即便从零开始，也有赢的机会

我们的战略就是：要做一个四有青年

1



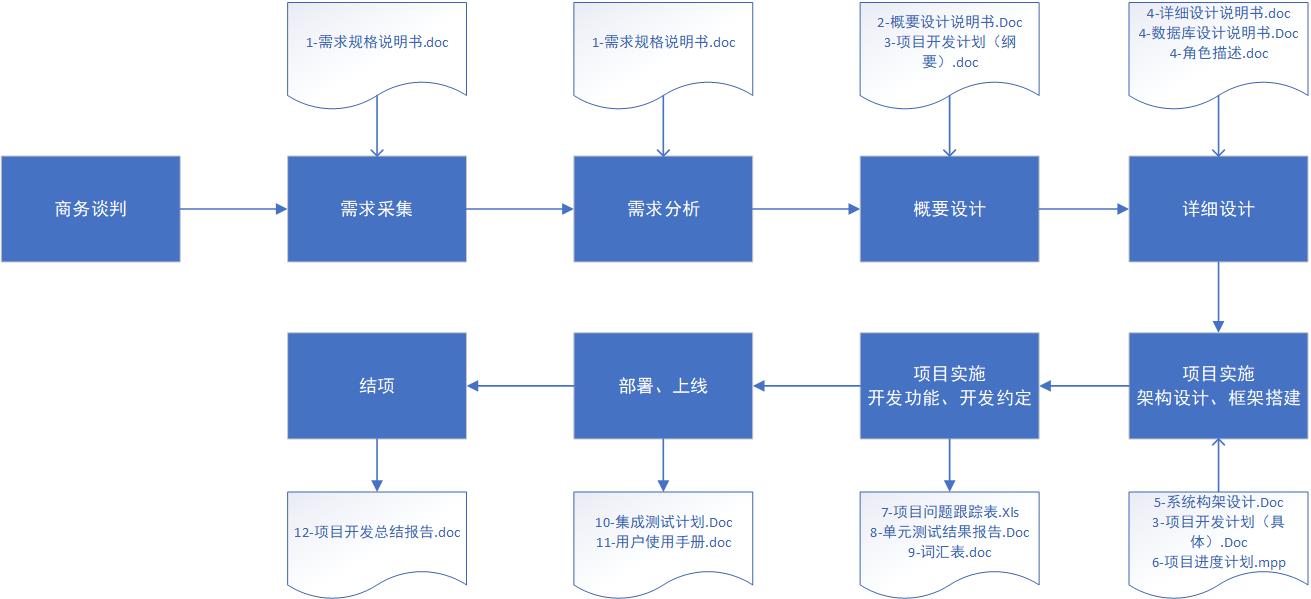
## 项目开发

### 商业项目整体开发流程

项目各阶段时间规划：4:3:2:1

* 4:40%做需求
* 3:30%做开发
* 2:20%做测试
* 1:10%做验收，并接受客户反馈

1



### 项目开发工具

1

xmind/mindmanager：思维导图，用于分析系统主题模块

powerdesigner/ Navicat Premium：建模工具，用于分析数据模型及相互之间的关系

axure：原型工具，用于绘制项目原型，快速与用户达成一致

pycharm：开发算子与可视化项目

git/svn：版本控制

mysql：结果数据存储

hdfs：原始数据存储

spark：数据分析

vmare(centos7.5)：虚拟机

xshell：操作工具

winscp/xftp：文件同步工具

### 大数据分析类项目开发流程

一个大数据分析开发项目，主要由业务分析、算法建模、数据采集、数据清洗、数据建模、数据存储、数据探索、特征提取、特征转换、数据分析、数据展示等步骤构成。

1

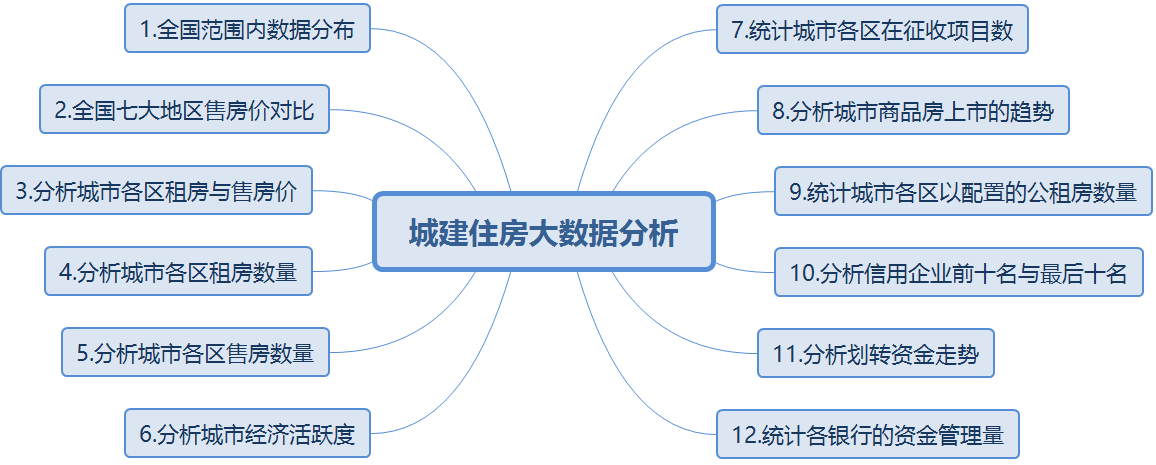
1. 业务分析：即需求分析，就是搞清楚，要做什么
2. 算法建模：根据业务，设计一个模型，可用伪代码表示。
3. 数据采集：根据业务目标，收集数据
4. 数据清洗：采集的数据存在空缺值、异常值、不符合规范的值，需要补充或者清除，原始文档无法查看需要转换或提取
5. 数据建模：根据算法或者业务来构造数据模型，即表和表之间的关系，目的是为了方便做计算
6. 数据存储：存储的方式有很多，csv、excel、txt、xml、parquet、mysql、mongodb、hdfs等，这需要视数据未来规划进行存储。比如纯粹存一下，仅供分析，不做其他处理，可以采用parquet、csv等，需要做联机事务处理的，对数据进行增删改查、就建议使用mysql，数据未来增长较大，可存到HDFS，若是需要做实时分析，则存到kafka、rabbitmq
7. 数据探索：这一步，主要是根据统计学原理来查看数据规律，比如分布状态、数据走势。还可以利用工具做数据清洗
8. 特征提取：比如身高、体重，这些都是描述事物的特征，按面向对象设计，称为属性。实际上，在做数据分析的时候，并不是全部特征都需要分析，比如股票那一天售价最高，那么就只有两个特征，时间+价格。特征提取，就是选择有效特征，来做分析。
9. 特征转换：将数据数字化、标准化、归一化有助于数据分析。比如，将特征性别设置为0/1，将成绩<60设为不及格，60-80设为中、80-90设为良、90以上为优
10. 数据分析：此时得到的数据就可以已经是干净的数据了，利用算法，展开分析
11. 数据展示：得到的结果还需要能够直观的观察，比如查看数据走势、数据分布等。利用可视化工具，将数据生动展示。至此，完整整个项目开发工作。

## 案例

### 需求模块

项目各模块如下图所示（xmind）

1



### 需求设计

1. 后端设计

1

1.全国范围内数据分布：

业务设计：分析数据集中的数据，主要集中在那些城市。对于写算子，就是按城市分组统计

2.全国七大地区售房价对比

业务设计：按华中、华北这样的区域维度，来做房价分析

3.分析城市各区租房与售房价

业务设计：分析租房与售房价，看性价比

4.分析城市各区租房数量

业务设计：分析各区域的租房市场

5.分析城市各区售房数量

6.分析城市经济活跃度（词云）

7.统计城市各区在征收项目数

8.分析城市商品房上市的趋势

9.统计城市各区以配置的公租房数量

10.分析信用企业前十名与最后十名（两个图）

11.分析划转资金走势

12.统计各银行的资金管理量

2. 前端设计（简化版/axure）

1

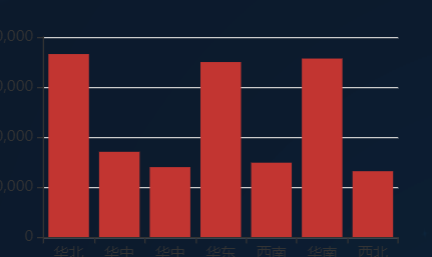
#### 1.全国范围内数据分布

效果图：



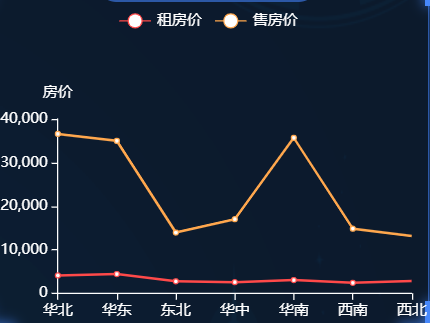
#### 2.全国七大地区售房价对比

效果图：



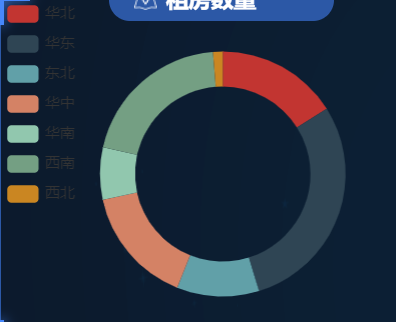
#### 3.分析城市各区租房与售房价

效果图：



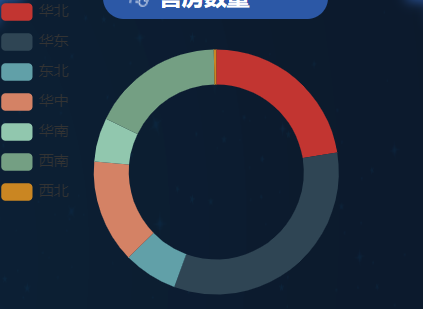
#### 4.分析城市各区租房数量

效果图：



#### 5.分析城市各区售房数量

效果图：



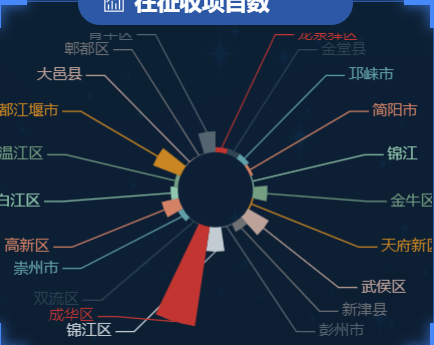
#### 6.分析城市经济活跃度（词云）

效果图：



#### 7.统计城市各区在征收项目数

效果图：



#### 8.分析城市商品房上市的趋势

效果图：



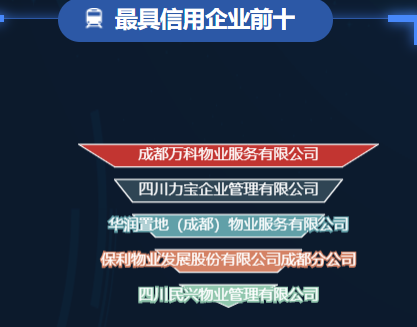
#### 9.统计城市各区以配置的公租房数量

效果图：



#### 10.分析信用企业前十名与最后十名

效果图：



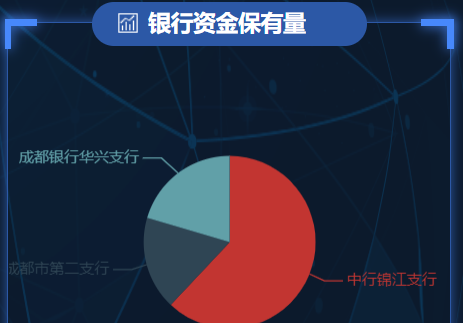
#### 11.分析划转资金走势

效果图：



#### 12.统计各银行的资金管理量

效果图：



### 数据集

所有数据集在 \城建住房大数据分析\城建住房爬虫、项目与数据\house\stock\_original\_data 目录下

1

#### 1.全国范围内数据分布：

ershoufang\_price.txt

#### 2.全国七大地区售房价对比

ershoufang\_price.txt

#### 3.分析城市各区租房与售房价

province.txt

#### 4.分析城市各区租房数量

zufang\_price.txt

#### 5.分析城市各区售房数量

zufang\_price.txt

#### 6.分析城市经济活跃度（词云）

ershoufang\_price.txt

#### 7.统计城市各区在征收项目数

guo\_you\_tu\_di\_fang\_wu\_zheng\_shou.CSV

#### 8.分析城市商品房上市的趋势

k\_fgj\_yxs\_xm\_1.txt

#### 9.统计城市各区以配置的公租房数量

gong\_zu\_fang.txt

#### 10.分析信用企业前十名与最后十名（两个图）

company.txt

#### 11.分析划转资金走势

bank\_trend.txt

#### 12.统计各银行的资金管理量

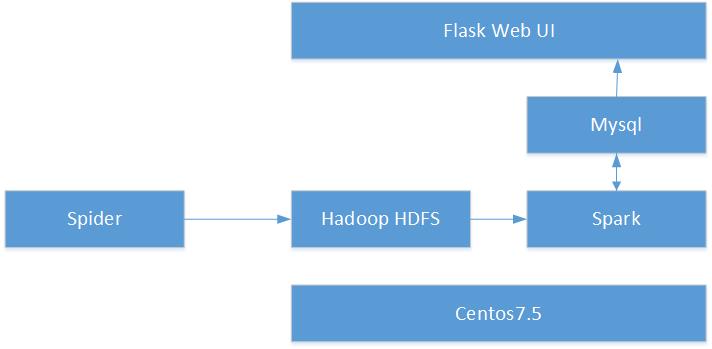
bank\_trend.txt

### 项目架构

#### 1.架构设计

系统架构如下：

* 底层基础平台：采用centos7.5
* Hadoop：仅作为数据存储
* Spark：作为计算引擎，从hadoop和Mysql取数据进行计算，然后将计算结果存入Mysql
* Flask Web UI：用于数据可视化

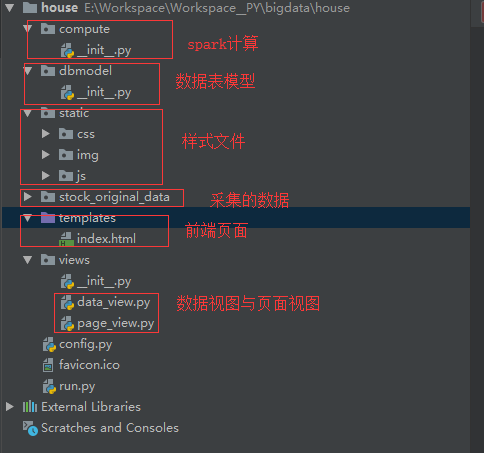


**注意：在本地部署项目时，创建house数据库，执行house.sql语句，初始化数据。爬虫采集的内容在文件中、数据分析的结果在Mysql数据库。**

#### 2.项目结构

创建项目，并重新编排架构，如下图所示。

1



### 算子

以下每个功能对应的算子。每个算子实现了具体的业务统计过程

1

#### 1.全国范围内数据分布：

city\_count.py

|  |
| --- |
| import os  from compute import MysqlConfig  from pyspark import Row  os.environ["PYSPARK\_PYTHON"] = r"D:\JavaRelease\Anaconda3\envs\flaskenv\python.exe"  # 华北地区  from pyspark.sql import SparkSession  spark = SparkSession \  .builder \  .appName("Python Spark SQL basic example") \  .config("spark.some.config.option", "some-value") \  .getOrCreate()  sc = spark.sparkContext  # 二手房  ershoufang\_rdd = sc.textFile("hdfs://localhost:9000/house/ershoufang\_price.txt")  def to\_pair(item):  tmp\_list = item.split(",")  # 二手房城市  ershoufang\_city = tmp\_list[0]  return ershoufang\_city, 1  pair\_rdd = ershoufang\_rdd.map(lambda x: to\_pair(x)).reduceByKey(lambda x, y: x + y).map(  lambda x: Row(city=x[0], count=x[1]))  schema\_ershoufang = spark.createDataFrame(pair\_rdd)  conn\_param = {}  conn\_param['user'] = MysqlConfig.MYSQL\_USER  conn\_param['password'] = MysqlConfig.MYSQL\_PWD  conn\_param['driver'] = MysqlConfig.MYSQL\_DRIVER  schema\_ershoufang.write.jdbc(MysqlConfig.MYSQL\_CONN, 'city\_count', 'overwrite', conn\_param)  print("执行完毕") |

#### 2.全国七大地区售房价对比

seven\_regions\_house\_price\_analysis.py

|  |
| --- |
| import os  from compute import MysqlConfig  from pyspark import Row  os.environ["PYSPARK\_PYTHON"] = r"D:\JavaRelease\Anaconda3\envs\flaskenv\python.exe"  # 华北地区  from pyspark.sql import SparkSession  huabei = ['北京', '天津', '河北', '山西', '内蒙古']  # 华东地区  huadong = ['上海', '江苏', '浙江', '山东', '安徽', '江西', '福建']  # 东北地区  dongbei = ['辽宁', '吉林', '黑龙江']  # 华中地区  huazhong = ['湖北', '湖南', '河南']  # 华南地区  huanan = ['广东', '广西', '海南']  # 西南地区  xinan = ['四川', '重庆', '贵州', '云南', '西藏']  # 西北地区  xibei = ['陕西', '甘肃', '新疆', '青海', '宁夏']  spark = SparkSession \  .builder \  .appName("Python Spark SQL basic example") \  .config("spark.some.config.option", "some-value") \  .getOrCreate()  sc = spark.sparkContext  # 城市  province\_list = []  f = open("hdfs://localhost:9000/house/province.txt", encoding="utf-8")  for i in f:  province\_list.append(i.strip())  # 二手房  ershoufang\_rdd = sc.textFile("hdfs://localhost:9000/house/ershoufang\_price.txt")  def group(item, province):  tmp\_list = item.split(",")  # 二手房城市  ershoufang\_city = tmp\_list[0]  result = list(filter(lambda p: ershoufang\_city in p, province))  if len(result) > 0:  # 根据二手房的城市找到对应的省份  tmp\_province = result[0].split(",")[0].replace("市", "").replace("省", "")  if tmp\_province in huabei:  return "华北", (1, int(tmp\_list[3]))  elif tmp\_province in huadong:  return "华东", (1, int(tmp\_list[3]))  elif tmp\_province in dongbei:  return "东北", (1, int(tmp\_list[3]))  elif tmp\_province in huazhong:  return "华中", (1, int(tmp\_list[3]))  elif tmp\_province in huanan:  return "华南", (1, int(tmp\_list[3]))  elif tmp\_province in xinan:  return "西南", (1, int(tmp\_list[3]))  elif tmp\_province in xibei:  return "西北", (1, int(tmp\_list[3]))  else:  return "", (1, 0)  else:  return "", (1, 0)  pair\_rdd = ershoufang\_rdd.map(lambda x: group(x, province\_list)).reduceByKey(  lambda x, y: (x[0] + y[0], x[1] + y[1])).map(lambda x: Row(city=x[0], price=x[1][1] / x[1][0]))  schema\_ershoufang = spark.createDataFrame(pair\_rdd)  conn\_param = {}  conn\_param['user'] = MysqlConfig.MYSQL\_USER  conn\_param['password'] = MysqlConfig.MYSQL\_PWD  conn\_param['driver'] = MysqlConfig.MYSQL\_DRIVER  schema\_ershoufang.write.jdbc(MysqlConfig.MYSQL\_CONN, 'serven\_area', 'overwrite', conn\_param)  print("执行完毕") |

#### 3.分析城市各区租房与售房价

zufang\_shoufang\_price\_compare.py

|  |
| --- |
| import os  from compute import MysqlConfig  from pyspark import Row  os.environ["PYSPARK\_PYTHON"] = r"D:\JavaRelease\Anaconda3\envs\flaskenv\python.exe"  # 华北地区  from pyspark.sql import SparkSession  huabei = ['北京', '天津', '河北', '山西', '内蒙古']  # 华东地区  huadong = ['上海', '江苏', '浙江', '山东', '安徽', '江西', '福建']  # 东北地区  dongbei = ['辽宁', '吉林', '黑龙江']  # 华中地区  huazhong = ['湖北', '湖南', '河南']  # 华南地区  huanan = ['广东', '广西', '海南']  # 西南地区  xinan = ['四川', '重庆', '贵州', '云南', '西藏']  # 西北地区  xibei = ['陕西', '甘肃', '新疆', '青海', '宁夏']  spark = SparkSession \  .builder \  .appName("Python Spark SQL basic example") \  .config("spark.some.config.option", "some-value") \  .getOrCreate()  sc = spark.sparkContext  # 城市  province\_list = []  f = open(r"E:\Workspace\Workspace\_\_PY\bigdata\house\stock\_original\_data\province.txt", encoding="utf-8")  for i in f:  province\_list.append(i.strip())  # 二手房  ershoufang\_rdd = sc.textFile("hdfs://localhost:9000/house/ershoufang\_price.txt")  # 租房  zufang\_rdd = sc.textFile("hdfs://localhost:9000/house/zufang\_price.txt")  def group(item, province):  tmp\_list = item.split(",")  # 二手房城市  ershoufang\_city = tmp\_list[0]  result = list(filter(lambda p: ershoufang\_city in p, province))  if len(result) > 0:  # 根据二手房的城市找到对应的省份  tmp\_province = result[0].split(",")[0].replace("市", "").replace("省", "")  price = tmp\_list[3]  if "-" in tmp\_list[3]:  price = tmp\_list[3].split("-")[0]  if tmp\_province in huabei:  return "华北", (1, int(price))  elif tmp\_province in huadong:  return "华东", (1, int(price))  elif tmp\_province in dongbei:  return "东北", (1, int(price))  elif tmp\_province in huazhong:  return "华中", (1, int(price))  elif tmp\_province in huanan:  return "华南", (1, int(price))  elif tmp\_province in xinan:  return "西南", (1, int(price))  elif tmp\_province in xibei:  return "西北", (1, int(price))  else:  return "", (1, 0)  else:  return "", (1, 0)  # 二手房键值对RDD  ershoufang\_pair\_rdd = ershoufang\_rdd.map(lambda x: group(x, province\_list)).reduceByKey(  lambda x, y: (x[0] + y[0], x[1] + y[1], "ershoufang"))  # 租房键值对RDD  # zufang\_pair\_rdd = zufang\_rdd.map(lambda x: group(x, province\_list)).collect()  zufang\_pair\_rdd = zufang\_rdd.map(lambda x: group(x, province\_list)).reduceByKey(  lambda x, y: (x[0] + y[0], x[1] + y[1], "zufang"))  union\_rdd = ershoufang\_pair\_rdd.union(zufang\_pair\_rdd).map(  lambda x: Row(city=x[0], price=x[1][1] / x[1][0], data\_type=x[1][2]))  schema\_ershoufang = spark.createDataFrame(union\_rdd)  conn\_param = {}  conn\_param['user'] = MysqlConfig.MYSQL\_USER  conn\_param['password'] = MysqlConfig.MYSQL\_PWD  conn\_param['driver'] = MysqlConfig.MYSQL\_DRIVER  schema\_ershoufang.write.jdbc(MysqlConfig.MYSQL\_CONN, 'zufang\_shoufang\_price\_compare', 'overwrite', conn\_param)  print("执行完毕") |

#### 4.分析城市各区租房数量

every\_regions\_zufang\_count.py

|  |
| --- |
| import os  from compute import MysqlConfig  from pyspark import Row  os.environ["PYSPARK\_PYTHON"] = r"D:\JavaRelease\Anaconda3\envs\flaskenv\python.exe"  # 华北地区  from pyspark.sql import SparkSession  huabei = ['北京', '天津', '河北', '山西', '内蒙古']  # 华东地区  huadong = ['上海', '江苏', '浙江', '山东', '安徽', '江西', '福建']  # 东北地区  dongbei = ['辽宁', '吉林', '黑龙江']  # 华中地区  huazhong = ['湖北', '湖南', '河南']  # 华南地区  huanan = ['广东', '广西', '海南']  # 西南地区  xinan = ['四川', '重庆', '贵州', '云南', '西藏']  # 西北地区  xibei = ['陕西', '甘肃', '新疆', '青海', '宁夏']  spark = SparkSession \  .builder \  .appName("Python Spark SQL basic example") \  .config("spark.some.config.option", "some-value") \  .getOrCreate()  sc = spark.sparkContext  # 城市  province\_list = []  f = open(r"E:\Workspace\Workspace\_\_PY\bigdata\house\stock\_original\_data\province.txt", encoding="utf-8")  for i in f:  province\_list.append(i.strip())  # 租房  ershoufang\_rdd = sc.textFile("hdfs://localhost:9000/house/zufang\_price.txt")  def group(item, province):  tmp\_list = item.split(",")  # 二手房城市  ershoufang\_city = tmp\_list[0]  result = list(filter(lambda p: ershoufang\_city in p, province))  if len(result) > 0:  # 根据二手房的城市找到对应的省份  tmp\_province = result[0].split(",")[0].replace("市", "").replace("省", "")  if tmp\_province in huabei:  return "华北", 1  elif tmp\_province in huadong:  return "华东", 1  elif tmp\_province in dongbei:  return "东北", 1  elif tmp\_province in huazhong:  return "华中", 1  elif tmp\_province in huanan:  return "华南", 1  elif tmp\_province in xinan:  return "西南", 1  elif tmp\_province in xibei:  return "西北", 1  else:  return "", 0  else:  return "", 0  pair\_rdd = ershoufang\_rdd.map(lambda x: group(x, province\_list)).reduceByKey(  lambda x, y: x + y).map(lambda x: Row(city=x[0], count=x[1]))  schema\_ershoufang = spark.createDataFrame(pair\_rdd)  conn\_param = {}  conn\_param['user'] = MysqlConfig.MYSQL\_USER  conn\_param['password'] = MysqlConfig.MYSQL\_PWD  conn\_param['driver'] = MysqlConfig.MYSQL\_DRIVER  schema\_ershoufang.write.jdbc(MysqlConfig.MYSQL\_CONN, 'every\_regions\_zufang\_count', 'overwrite', conn\_param)  print("执行完毕") |

#### 5.分析城市各区售房数量

every\_regions\_zufang\_count.py

|  |
| --- |
| import os  from compute import MysqlConfig  from pyspark import Row  os.environ["PYSPARK\_PYTHON"] = r"D:\JavaRelease\Anaconda3\envs\flaskenv\python.exe"  # 华北地区  from pyspark.sql import SparkSession  huabei = ['北京', '天津', '河北', '山西', '内蒙古']  # 华东地区  huadong = ['上海', '江苏', '浙江', '山东', '安徽', '江西', '福建']  # 东北地区  dongbei = ['辽宁', '吉林', '黑龙江']  # 华中地区  huazhong = ['湖北', '湖南', '河南']  # 华南地区  huanan = ['广东', '广西', '海南']  # 西南地区  xinan = ['四川', '重庆', '贵州', '云南', '西藏']  # 西北地区  xibei = ['陕西', '甘肃', '新疆', '青海', '宁夏']  spark = SparkSession \  .builder \  .appName("Python Spark SQL basic example") \  .config("spark.some.config.option", "some-value") \  .getOrCreate()  sc = spark.sparkContext  # 城市  province\_list = []  f = open(r"E:\Workspace\Workspace\_\_PY\bigdata\house\stock\_original\_data\province.txt", encoding="utf-8")  for i in f:  province\_list.append(i.strip())  # 租房  ershoufang\_rdd = sc.textFile("hdfs://localhost:9000/house/zufang\_price.txt")  def group(item, province):  tmp\_list = item.split(",")  # 二手房城市  ershoufang\_city = tmp\_list[0]  result = list(filter(lambda p: ershoufang\_city in p, province))  if len(result) > 0:  # 根据二手房的城市找到对应的省份  tmp\_province = result[0].split(",")[0].replace("市", "").replace("省", "")  if tmp\_province in huabei:  return "华北", 1  elif tmp\_province in huadong:  return "华东", 1  elif tmp\_province in dongbei:  return "东北", 1  elif tmp\_province in huazhong:  return "华中", 1  elif tmp\_province in huanan:  return "华南", 1  elif tmp\_province in xinan:  return "西南", 1  elif tmp\_province in xibei:  return "西北", 1  else:  return "", 0  else:  return "", 0  pair\_rdd = ershoufang\_rdd.map(lambda x: group(x, province\_list)).reduceByKey(  lambda x, y: x + y).map(lambda x: Row(city=x[0], count=x[1]))  schema\_ershoufang = spark.createDataFrame(pair\_rdd)  conn\_param = {}  conn\_param['user'] = MysqlConfig.MYSQL\_USER  conn\_param['password'] = MysqlConfig.MYSQL\_PWD  conn\_param['driver'] = MysqlConfig.MYSQL\_DRIVER  schema\_ershoufang.write.jdbc(MysqlConfig.MYSQL\_CONN, 'every\_regions\_zufang\_count', 'overwrite', conn\_param)  print("执行完毕") |

#### 6.分析城市经济活跃度（词云）

economic\_active.py

|  |
| --- |
| import os  from compute import MysqlConfig  from pyspark import Row  os.environ["PYSPARK\_PYTHON"] = r"D:\JavaRelease\Anaconda3\envs\flaskenv\python.exe"  # 华北地区  from pyspark.sql import SparkSession  spark = SparkSession \  .builder \  .appName("Python Spark SQL basic example") \  .config("spark.some.config.option", "some-value") \  .getOrCreate()  sc = spark.sparkContext  # 二手房  ershoufang\_rdd = sc.textFile("hdfs://localhost:9000/house/ershoufang\_price.txt")  def group(item):  tmp\_list = item.split(",")  # 二手房城市  ershoufang\_city = tmp\_list[0]  price = tmp\_list[3]  if "-" in tmp\_list[3]:  price = tmp\_list[3].split("-")[0]  return ershoufang\_city, (1, int(price))  # 二手房键值对RDD  ershoufang\_pair\_rdd = ershoufang\_rdd.map(lambda x: group(x)).reduceByKey(  lambda x, y: (x[0] + y[0], x[1] + y[1])).map(  lambda x: Row(city=x[0], count=x[1][0], price=x[1][1] / x[1][0]))  schema\_ershoufang\_df = spark.createDataFrame(ershoufang\_pair\_rdd)  schema\_ershoufang\_df.createGlobalTempView("zufang\_shoufang\_price\_compare")  last\_df = spark.sql(  "SELECT \* FROM global\_temp.zufang\_shoufang\_price\_compare order by count,price desc limit 10")  conn\_param = {}  conn\_param['user'] = MysqlConfig.MYSQL\_USER  conn\_param['password'] = MysqlConfig.MYSQL\_PWD  conn\_param['driver'] = MysqlConfig.MYSQL\_DRIVER  last\_df.write.jdbc(MysqlConfig.MYSQL\_CONN, 'economic\_active', 'overwrite', conn\_param)  print("执行完毕") |

#### 7.统计城市各区在征收项目数

guo\_you\_tu\_di\_fang\_wu\_zheng\_shou.py

|  |
| --- |
| import os  from compute import MysqlConfig  from pyspark import Row  os.environ["PYSPARK\_PYTHON"] = r"D:\JavaRelease\Anaconda3\envs\flaskenv\python.exe"  # 华北地区  from pyspark.sql import SparkSession  spark = SparkSession \  .builder \  .appName("Python Spark SQL basic example") \  .config("spark.some.config.option", "some-value") \  .getOrCreate()  sc = spark.sparkContext  # 二手房  df = spark.read.load(  r"E:\Workspace\Workspace\_\_PY\bigdata\house\stock\_original\_data\guo\_you\_tu\_di\_fang\_wu\_zheng\_shou.CSV",  format="csv", sep=",", inferSchema="true", header="true", encoding="gbk")  def group(item):  print(item[1])  return item[1], 1  df.createGlobalTempView("guo\_you\_tu\_di\_fang\_wu\_zheng\_shou")  fang\_wu\_zheng\_shou\_df = spark.sql("SELECT area,count(\*) as counter FROM global\_temp.guo\_you\_tu\_di\_fang\_wu\_zheng\_shou group by area")  conn\_param = {}  conn\_param['user'] = MysqlConfig.MYSQL\_USER  conn\_param['password'] = MysqlConfig.MYSQL\_PWD  conn\_param['driver'] = MysqlConfig.MYSQL\_DRIVER  fang\_wu\_zheng\_shou\_df.write.jdbc(MysqlConfig.MYSQL\_CONN, 'fang\_wu\_zheng\_shou', 'overwrite', conn\_param)  print("执行完毕") |

#### 8.分析城市商品房上市的趋势

yu\_shou\_trend.py

|  |
| --- |
| import datetime  import os  from compute import MysqlConfig  from pyspark import Row  from pyspark.sql.functions import udf  from pyspark.sql.types import StringType  os.environ["PYSPARK\_PYTHON"] = r"D:\JavaRelease\Anaconda3\envs\flaskenv\python.exe"  # 华北地区  from pyspark.sql import SparkSession  spark = SparkSession \  .builder \  .appName("Python Spark SQL basic example") \  .config("spark.some.config.option", "some-value") \  .getOrCreate()  sc = spark.sparkContext  # 土地房屋征收  tudixiangmu\_rdd = sc.textFile("hdfs://localhost:9000/house/k\_fgj\_yxs\_xm\_1.txt")  def to\_pair(item):  datas = item.split("\t")  year = datas[1].split("-")[0]  try:  if datas[0] is not None and len(datas[0])>0:  return year, float(datas[0])  else:  return year, 0  except Exception as e:  print("错误的：",datas[0])  return year, 0  pair\_rdd = tudixiangmu\_rdd.map(lambda x: to\_pair(x)).reduceByKey(  lambda x, y: x + y).map(lambda x: Row(year=x[0], area=x[1]))  schema\_ershoufang = spark.createDataFrame(pair\_rdd)  conn\_param = {}  conn\_param['user'] = MysqlConfig.MYSQL\_USER  conn\_param['password'] = MysqlConfig.MYSQL\_PWD  conn\_param['driver'] = MysqlConfig.MYSQL\_DRIVER  schema\_ershoufang.write.jdbc(MysqlConfig.MYSQL\_CONN, 'yu\_shou\_trend', 'overwrite', conn\_param)  print("执行完毕") |

#### 9.统计城市各区以配置的公租房数量

gong\_zu\_fang.py

|  |
| --- |
| import datetime  import os  from compute import MysqlConfig  from pyspark import Row  from pyspark.sql.functions import udf  from pyspark.sql.types import StringType  os.environ["PYSPARK\_PYTHON"] = r"D:\JavaRelease\Anaconda3\envs\flaskenv\python.exe"  # 华北地区  from pyspark.sql import SparkSession  spark = SparkSession \  .builder \  .appName("Python Spark SQL basic example") \  .config("spark.some.config.option", "some-value") \  .getOrCreate()  sc = spark.sparkContext  # 土地房屋征收  gong\_zu\_fang\_rdd = sc.textFile("hdfs://localhost:9000/house/gong\_zu\_fang.txt")  def to\_pair(item):  datas = item.split("\t")  try:  if datas[0] is not None and len(datas[0]) > 0:  return datas[0], (1, float(datas[1]))  else:  return datas[0], (1, 0)  except Exception as e:  print("错误的：", datas[0])  return datas[0], (1, 0)  pair\_rdd = gong\_zu\_fang\_rdd.map(lambda x: to\_pair(x)).reduceByKey(  lambda x, y: (x[0] + y[0], x[1] + y[1])).map(lambda x: (x[0], x[1][0], x[1][1] / x[1][0])).map(  lambda x: Row(location=x[0], counter=x[1], area=x[2]))  schema\_gong\_zufang = spark.createDataFrame(pair\_rdd)  conn\_param = {}  conn\_param['user'] = MysqlConfig.MYSQL\_USER  conn\_param['password'] = MysqlConfig.MYSQL\_PWD  conn\_param['driver'] = MysqlConfig.MYSQL\_DRIVER  schema\_gong\_zufang.write.jdbc(MysqlConfig.MYSQL\_CONN, 'gong\_zu\_fang', 'overwrite', conn\_param)  print("执行完毕") |

#### 10.分析信用企业前十名与最后十名（两个图）

company\_sorted.py

|  |
| --- |
| import os  from compute import MysqlConfig  from pyspark import Row  os.environ["PYSPARK\_PYTHON"] = r"D:\JavaRelease\Anaconda3\envs\flaskenv\python.exe"  # 华北地区  from pyspark.sql import SparkSession  spark = SparkSession \  .builder \  .appName("Python Spark SQL basic example") \  .config("spark.some.config.option", "some-value") \  .getOrCreate()  sc = spark.sparkContext  company\_rdd = sc.textFile("hdfs://localhost:9000/house/company.txt")  def to\_pair(item):  items = item.split("\t")  return items[0], float(items[1])  pair\_rdd = company\_rdd.map(lambda x: to\_pair(x)).reduceByKey(  lambda x, y: x + y).map(lambda x: Row(city=x[0], score=x[1]))  schema\_company = spark.createDataFrame(pair\_rdd)  conn\_param = {}  conn\_param['user'] = MysqlConfig.MYSQL\_USER  conn\_param['password'] = MysqlConfig.MYSQL\_PWD  conn\_param['driver'] = MysqlConfig.MYSQL\_DRIVER  schema\_company.write.jdbc(MysqlConfig.MYSQL\_CONN, 'company\_sorted', 'overwrite', conn\_param)  print("执行完毕") |

#### 11.分析划转资金走势

bank\_trend.py

|  |
| --- |
| import os  from compute import MysqlConfig  from pyspark import Row  os.environ["PYSPARK\_PYTHON"] = r"D:\JavaRelease\Anaconda3\envs\flaskenv\python.exe"  # 华北地区  from pyspark.sql import SparkSession  spark = SparkSession \  .builder \  .appName("Python Spark SQL basic example") \  .config("spark.some.config.option", "some-value") \  .getOrCreate()  sc = spark.sparkContext  company\_rdd = sc.textFile("hdfs://localhost:9000/house/bank\_trend.txt")  def to\_pair(item):  items = item.split("\t")  year = items[2].split("-")[0]  try:  amount = float(items[1])  except Exception as e:  amount = 0  return year, amount  pair\_rdd = company\_rdd.map(lambda x: to\_pair(x)).reduceByKey(  lambda x, y: x + y).map(lambda x: Row(year=x[0], amount=x[1]))  schema\_bank = spark.createDataFrame(pair\_rdd)  conn\_param = {}  conn\_param['user'] = MysqlConfig.MYSQL\_USER  conn\_param['password'] = MysqlConfig.MYSQL\_PWD  conn\_param['driver'] = MysqlConfig.MYSQL\_DRIVER  schema\_bank.write.jdbc(MysqlConfig.MYSQL\_CONN, 'bank\_trend', 'overwrite', conn\_param)  print("执行完毕") |

#### 12.统计各银行的资金管理量

bank\_amount.py

|  |
| --- |
| import os  from compute import MysqlConfig  from pyspark import Row  os.environ["PYSPARK\_PYTHON"] = r"D:\JavaRelease\Anaconda3\envs\flaskenv\python.exe"  # 华北地区  from pyspark.sql import SparkSession  spark = SparkSession \  .builder \  .appName("Python Spark SQL basic example") \  .config("spark.some.config.option", "some-value") \  .getOrCreate()  sc = spark.sparkContext  company\_rdd = sc.textFile("hdfs://localhost:9000/house/bank\_trend.txt")  def to\_pair(item):  items = item.split("\t")  company\_name = items[0]  try:  amount = float(items[1])  except Exception as e:  amount = 0  return company\_name, amount  pair\_rdd = company\_rdd.map(lambda x: to\_pair(x)).reduceByKey(  lambda x, y: x + y).map(lambda x: Row(company\_name=x[0], amount=x[1]))  schema\_bank = spark.createDataFrame(pair\_rdd)  conn\_param = {}  conn\_param['user'] = MysqlConfig.MYSQL\_USER  conn\_param['password'] = MysqlConfig.MYSQL\_PWD  conn\_param['driver'] = MysqlConfig.MYSQL\_DRIVER  schema\_bank.write.jdbc(MysqlConfig.MYSQL\_CONN, 'company\_amont', 'overwrite', conn\_param)  print("执行完毕") |

### 前端

以下每个功能前后端的实现

1

#### 1.全国范围内数据分布

1. 后端

|  |
| --- |
| @data.route('/getMap', methods=['GET'])  def get\_map():  data = db.session.query(CityCount).all()  view\_data = {}  view\_data["series"] = []  def build\_view\_data(item):  dic = {}  dic["value"] = item.count  dic["name"] = item.city  view\_data["series"].append([dic])  [build\_view\_data(item) for item in data]  return json.dumps(view\_data, ensure\_ascii=False) |

1. 前端

|  |
| --- |
| (function (func) {  $.ajax({  url: "/data/getMap",  type: "GET",  dataType: "json",  success: function (data) {  func(data);  }  });  })(function (data) {  // 基于准备好的dom，初始化echarts实例  var myChart = echarts.init(document.getElementById('chart\_map'));  var mapName = 'china'  /\*获取地图数据\*/  myChart.showLoading();  var mapFeatures = echarts.getMap(mapName).geoJson.features;  myChart.hideLoading();  var geoCoordMap = {  '福州': [119.4543, 25.9222],  '长春': [125.8154, 44.2584],  '重庆': [107.7539, 30.1904],  '西安': [109.1162, 34.2004],  '成都': [103.9526, 30.7617],  '常州': [119.4543, 31.5582],  '北京': [116.4551, 40.2539],  '北海': [109.314, 21.6211],  '海口': [110.3893, 19.8516],  '长沙': [113.019455, 28.200103],  '上海': [121.40, 31.73],  '内蒙古': [106.82, 39.67]  };  function get\_latitude\_longitude(item) {  return item["name"].indexOf(window.cityname) > -1;  }  var color = ['#c5f80e'];  var series = [];  var tmp\_data = data.series.map(function (dataItem) {  var cityname = dataItem[0].name;  window.cityname = cityname;  var tmp\_area = areas.filter(get\_latitude\_longitude)[0];  var data1 = [parseFloat(tmp\_area["longitude"]),parseFloat(tmp\_area["latitude"]), dataItem[0].value];  return {  name: cityname,  value: data1  };  });  series.push({  name: "数据量",  type: 'effectScatter',  coordinateSystem: 'geo',  zlevel: 2,  rippleEffect: {  brushType: 'stroke'  },  label: {  normal: {  show: true,  position: 'right',  formatter: '{b}'  }  },  symbolSize: function (val) {  return val[2] / 1000;  },  itemStyle: {  normal: {  color: color[0]  }  },  data: tmp\_data  });  option = {  tooltip: {  trigger: 'item'  },  geo: {  map: 'china',  label: {  emphasis: {  show: false  }  },  roam: true,  itemStyle: {  normal: {  borderColor: 'rgba(147, 235, 248, 1)',  borderWidth: 1,  areaColor: {  type: 'radial',  x: 0.5,  y: 0.5,  r: 0.8,  colorStops: [{  offset: 0,  color: 'rgba(175,238,238, 0)' // 0% 处的颜色  }, {  offset: 1,  color: 'rgba(47,79,79, .1)' // 100% 处的颜色  }],  globalCoord: false // 缺省为 false  },  shadowColor: 'rgba(128, 217, 248, 1)',  // shadowColor: 'rgba(255, 255, 255, 1)',  shadowOffsetX: -2,  shadowOffsetY: 2,  shadowBlur: 10  },  emphasis: {  areaColor: '#389BB7',  borderWidth: 0  }  }  },  series: series  };  // 使用刚指定的配置项和数据显示图表。  myChart.setOption(option);  window.addEventListener("resize", function () {  myChart.resize();  });  }); |

#### 2.全国七大地区售房价对比

1. 后端

|  |
| --- |
| @data.route('/getServenArea', methods=['GET'])  def get\_serven\_area():  data = db.session.query(ServenArea).all()  view\_data = {}  view\_data["series\_data"] = []  def build\_view\_data(item):  tmp\_dic = {}  tmp\_dic["price"] = item.price  tmp\_dic["city"] = item.city  view\_data["series\_data"].append(tmp\_dic)  [build\_view\_data(item) for item in data]  return json.dumps(view\_data, ensure\_ascii=False) |

1. 前端

|  |
| --- |
| (function (func) {  $.ajax({  url: "/data/getServenArea",  type: "GET",  dataType: "json",  success: function (data) {  func(data.series\_data);  }  });  })(function (data) {  var myChart = echarts.init(document.getElementById('serven\_area'), 'infographic');  var data1 = [];  var data2 = [];  $(data).each(function (k, v) {  data1.push(v.city);  data2.push(v.price);  });  option = {  tooltip: {  trigger: 'axis'  },  calculable: true,  xAxis: [  {  type: 'category',  data: data1  }  ],  yAxis: [  {  type: 'value'  }  ],  series: [  {  name: '房价',  type: 'bar',  data: data2  }  ]  };  myChart.setOption(option);  window.addEventListener("resize", function () {  myChart.resize();  });  }); |

#### 3.分析城市各区租房与售房价

1. 后端

|  |
| --- |
| @data.route('/getZufangShoufangPriceCompare', methods=['GET'])  def get\_zufang\_shoufang\_price\_compare():  data = db.session.query(ZufangShoufangPriceCompare).all()  view\_data = {}  view\_data["xAxis"] = ["华北", "华东", "东北", "华中", "华南", "西南", "西北"]  view\_data["series1"] = []  view\_data["series2"] = []  all\_area = ["华北", "华东", "东北", "华中", "华南", "西南", "西北"]  def build\_view\_data(item):  tmp\_dic = {}  tmp\_dic["price"] = item.price  tmp\_dic["city"] = item.city  if item.data\_type == "zufang":  view\_data["series1"].append(item.price)  else:  view\_data["series2"].append(item.price)  [build\_view\_data(item) for i in all\_area for item in data if item.city == i]  return json.dumps(view\_data, ensure\_ascii=False) |

1. 前端

|  |
| --- |
| (function (func) {  $.ajax({  url: "/data/getZufangShoufangPriceCompare",  type: "GET",  dataType: "json",  success: function (data) {  func(data);  }  });  })(function (data) {  var myChart = echarts.init(document.getElementById('chart\_3'), 'infographic');  myChart.clear();  var option = {  title: {  text: ''  },  tooltip: {  trigger: 'axis'  },  legend: {  data: ['租房价', '售房价'],  textStyle: {  color: '#fff'  },  top: '8%'  },  grid: {  top: '40%',  left: '3%',  right: '4%',  bottom: '3%',  containLabel: true  },  color: ['#FF4949', '#FFA74D', '#FFEA51', '#4BF0FF', '#44AFF0', '#4E82FF', '#584BFF', '#BE4DFF', '#F845F1'],  xAxis: {  type: 'category',  boundaryGap: false,  data: data.xAxis,  splitLine: {  show: false  },  axisLine: {  lineStyle: {  color: '#fff'  }  }  },  yAxis: {  name: '房价',  type: 'value',  splitLine: {  show: false  },  axisLine: {  lineStyle: {  color: '#fff'  }  }  },  series: [  {  name: '租房价',  type: 'line',  data: data.series1  },  {  name: '售房价',  type: 'line',  data: data.series2  }  ]  };  myChart.setOption(option);  window.addEventListener("resize", function () {  myChart.resize();  });  }); |

#### 4.分析城市各区租房数量

1. 后端

|  |
| --- |
| @data.route('/getEveryRegionsZufangCount', methods=['GET'])  def get\_every\_regions\_zufang\_count():  data = db.session.query(EveryRegionsZufangCount).all()  view\_data = {}  view\_data["xAxis"] = ["华北", "华东", "东北", "华中", "华南", "西南", "西北"]  view\_data["series1"] = []  all\_area = ["华北", "华东", "东北", "华中", "华南", "西南", "西北"]  def build\_view\_data(item):  tmp\_dic = {}  tmp\_dic["value"] = item.count  tmp\_dic["name"] = item.city  view\_data["series1"].append(tmp\_dic)  [build\_view\_data(item) for i in all\_area for item in data if item.city == i]  return json.dumps(view\_data, ensure\_ascii=False) |

1. 前端

|  |
| --- |
| (function (func) {  $.ajax({  url: "/data/getEveryRegionsZufangCount",  type: "GET",  dataType: "json",  success: function (data) {  func(data);  }  });  })(function (data) {  var myChart = echarts.init(document.getElementById('chart\_2'), 'infographic');  myChart.clear();  var option = {  tooltip: {  trigger: 'item',  formatter: "{a} <br/>{b} : {c} ({d}%)"  },  legend: {  orient: 'vertical',  x: 'left',  data: data.xAxis  },  calculable: true,  series: [  {  name: '租房数量',  type: 'pie',  radius: ['50%', '70%'],  itemStyle: {  normal: {  label: {  show: false  },  labelLine: {  show: false  }  },  emphasis: {  label: {  show: true,  position: 'center',  textStyle: {  fontSize: '30',  fontWeight: 'bold'  }  }  }  },  data: data.series1  }  ]  };  myChart.setOption(option);  window.addEventListener("resize", function () {  myChart.resize();  });  }); |

#### 5.分析城市各区售房数量

1. 后端

|  |
| --- |
| @data.route('/getEveryRegionsErshoufangCount', methods=['GET'])  def get\_every\_regions\_ershoufang\_count():  data = db.session.query(EveryRegionsErshoufangCount).all()  view\_data = {}  view\_data["xAxis"] = ["华北", "华东", "东北", "华中", "华南", "西南", "西北"]  view\_data["series1"] = []  all\_area = ["华北", "华东", "东北", "华中", "华南", "西南", "西北"]  def build\_view\_data(item):  tmp\_dic = {}  tmp\_dic["value"] = item.count  tmp\_dic["name"] = item.city  view\_data["series1"].append(tmp\_dic)  [build\_view\_data(item) for i in all\_area for item in data if item.city == i]  return json.dumps(view\_data, ensure\_ascii=False) |

1. 前端

|  |
| --- |
| (function (func) {  $.ajax({  url: "/data/getEveryRegionsErshoufangCount",  type: "GET",  dataType: "json",  success: function (data) {  func(data);  }  });  })(function (data) {  var myChart = echarts.init(document.getElementById('chart\_4'), 'infographic');  myChart.clear();  var option = {  tooltip: {  trigger: 'item',  formatter: "{a} <br/>{b} : {c} ({d}%)"  },  legend: {  orient: 'vertical',  x: 'left',  data: data.xAxis  },  calculable: true,  series: [  {  name: '售房数量',  type: 'pie',  radius: ['50%', '70%'],  itemStyle: {  normal: {  label: {  show: false  },  labelLine: {  show: false  }  },  emphasis: {  label: {  show: true,  position: 'center',  textStyle: {  fontSize: '30',  fontWeight: 'bold'  }  }  }  },  data: data.series1  }  ]  };  myChart.setOption(option);  window.addEventListener("resize", function () {  myChart.resize();  });  }); |

#### 6.分析城市经济活跃度（词云）

1. 后端

|  |
| --- |
| @data.route('/getEconomicActive', methods=['GET'])  def get\_economic\_active():  data = db.session.query(EconomicActive).all()  view\_data = {}  view\_data["series1"] = []  def build\_view\_data(item):  view\_data["series1"].append(item.city)  [build\_view\_data(item) for item in data]  return json.dumps(view\_data, ensure\_ascii=False) |

1. 前端

|  |
| --- |
| (function (func) {  $.ajax({  url: "/data/getEconomicActive",  type: "GET",  dataType: "json",  success: function (data) {  func(data);  }  });  })(function (data) {  var html = "";  var regBusiness = $("#regBusiness");  data.series1.forEach(function (item) {  html += '<li><a href="javascript:;">' + item + '</a></li>'  });  regBusiness.html(html);  textAnim();  function textAnim() {  try {  TagCanvas.Start('cloudCanvas', 'tags2', {  textFont: "Arial, Helvetica, sans-serif",  maxSpeed: 0.05,  minSpeed: 0.01,  textColour: '#41FDFE',  textHeight: 12,  outlineMethod: "colour",  fadeIn: 800,  outlineColour: "#41b1c3",  outlineOffset: 0,  depth: 0.97,  minBrightness: 0.2,  wheelZoom: false,  reverse: true,  shadowBlur: 2,  shuffleTags: true,  shadowOffset: [1, 1],  stretchX: 1.2,  initial: [0, 0.1],  clickToFront: 600,  maxSpeed: 0.01,  outlineDashSpeed: 0.5,  });  } catch (e) {  // something went wrong, hide the canvas container  document.getElementById('chart\_5').style.display = 'none';  }  }  }); |

#### 7.统计城市各区在征收项目数

1. 后端

|  |
| --- |
| @data.route('/getFangWuZhengShou', methods=['GET'])  def get\_fang\_wu\_zheng\_shou():  data = db.session.query(FangWuZhengShou).all()  view\_data = {}  view\_data["series1"] = []  def build\_view\_data(item):  tmp\_dic = {}  tmp\_dic["name"] = item.area  tmp\_dic["value"] = item.counter  view\_data["series1"].append(tmp\_dic)  [build\_view\_data(item) for item in data]  return json.dumps(view\_data, ensure\_ascii=False) |

1. 前端

|  |
| --- |
| (function (func) {  $.ajax({  url: "/data/getFangWuZhengShou",  type: "GET",  dataType: "json",  success: function (data) {  func(data);  }  });  })(function (data) {  var myChart = echarts.init(document.getElementById('chart\_10'), 'infographic');  myChart.clear();  var option = {  tooltip: {  trigger: 'item',  formatter: "{a} <br/>{b} : {c} ({d}%)"  },  calculable: true,  series: [  {  name: '征收数量',  type: 'pie',  radius: [30, 110],  roseType: 'area',  x: '50%', // for funnel  max: 40, // for funnel  sort: 'ascending', // for funnel  data: data.series1  }  ]  };  myChart.setOption(option);  window.addEventListener("resize", function () {  myChart.resize();  });  }); |

#### 8.分析城市商品房上市的趋势

1. 后端

|  |
| --- |
| @data.route('/getYuShouTrend', methods=['GET'])  def get\_yu\_shou\_trend():  data = db.session.query(YuShouTrend).order\_by(YuShouTrend.year.asc()).all()  view\_data = {}  view\_data["year"] = []  view\_data["area"] = []  def build\_view\_data(item):  view\_data["area"].append(item.area)  view\_data["year"].append(item.year)  [build\_view\_data(item) for item in data]  return json.dumps(view\_data, ensure\_ascii=False) |

1. 前端

|  |
| --- |
| (function (func) {  $.ajax({  url: "/data/getYuShouTrend",  type: "GET",  dataType: "json",  success: function (data) {  func(data);  }  });  })(function (data) {  var myChart = echarts.init(document.getElementById('chart\_11'), 'infographic');  var option = {  legend: {  data: ['上市面积'],  textStyle: {  color: '#fff'  },  top: '8%'  },  grid: {  top: '40%',  left: '3%',  right: '4%',  bottom: '3%',  containLabel: true  },  color: ['#FF4949', '#FFA74D', '#FFEA51', '#4BF0FF', '#44AFF0', '#4E82FF', '#584BFF', '#BE4DFF', '#F845F1'],  xAxis: {  type: 'category',  boundaryGap: false,  data: data.year,  splitLine: {  show: false  },  axisLine: {  lineStyle: {  color: '#fff'  }  }  },  yAxis: {  name: '房价',  type: 'value',  splitLine: {  show: false  },  axisLine: {  lineStyle: {  color: '#fff'  }  }  },  series: [  {  name: '上市面积',  type: 'line',  data: data.area  }  ]  };  myChart.setOption(option);  window.addEventListener("resize", function () {  myChart.resize();  });  }); |

#### 9.统计城市各区以配置的公租房数量

1. 后端

|  |
| --- |
| @data.route('/getGongZuFang', methods=['GET'])  def get\_gong\_zu\_fang():  data = db.session.query(GongZuFang).all()  view\_data = {}  view\_data["xAxis"] = []  view\_data["series1"] = []  view\_data["series2"] = []  def build\_view\_data(item):  view\_data["xAxis"].append(item.location)  view\_data["series1"].append(item.counter)  view\_data["series2"].append(item.area)  [build\_view\_data(item) for item in data]  return json.dumps(view\_data, ensure\_ascii=False) |

1. 前端

|  |
| --- |
| (function (func) {  $.ajax({  url: "/data/getGongZuFang",  type: "GET",  dataType: "json",  success: function (data) {  func(data);  }  });  })(function (data) {  var myChart = echarts.init(document.getElementById('chart\_12'), 'infographic');  var option = {  tooltip: {  trigger: 'axis'  },  calculable: true,  legend: {  data: ['公租房数量', '平均面积']  },  xAxis: [  {  type: 'category',  data: data.xAxis  }  ],  yAxis: [  {  type: 'value',  name: '数量',  axisLabel: {  formatter: '{value}'  }  },  {  type: 'value',  name: '面积',  axisLabel: {  formatter: '{value} '  }  }  ],  series: [  {  name: '公租房数量',  type: 'bar',  data: data.series1  },  {  name: '平均面积',  type: 'line',  yAxisIndex: 1,  data: data.series2  }  ]  };  myChart.setOption(option);  window.addEventListener("resize", function () {  myChart.resize();  });  }); |

#### 10. 1分析信用企业前十名

1. 后端

|  |
| --- |
| @data.route('/getCompanySorted', methods=['GET'])  def get\_company\_sorted():  order = request.args.get("order")  if order == "1":  data = db.session.query(CompanySorted).order\_by(CompanySorted.score.desc()).all()[0:5]  else:  data = db.session.query(CompanySorted).order\_by(CompanySorted.score.asc()).all()[0:5]  view\_data = {}  view\_data["series"] = []  def build\_view\_data(item):  view\_data["series"].append({"value": item.score, "name": item.city})  [build\_view\_data(item) for item in data]  return json.dumps(view\_data, ensure\_ascii=False) |

1. 前端

|  |
| --- |
| (function (func) {  $.ajax({  url: "/data/getCompanySorted",  type: "GET",  data: {"order": 1},  dataType: "json",  success: function (data) {  func(data);  }  });  })(function (data) {  var myChart = echarts.init(document.getElementById('chart\_13'), 'infographic');  var option = {  tooltip: {  trigger: 'item',  formatter: "{a} <br/>{b} : {c}%"  },  calculable: true,  series: [  {  name: '评分',  type: 'funnel',  x: '10%',  y: 60,  //x2: 80,  y2: 60,  width: '80%',  // height: {totalHeight} - y - y2,  min: 0,  max: 2500,  minSize: '0%',  maxSize: '100%',  sort: 'descending', // 'ascending', 'descending'  gap: 10,  itemStyle: {  normal: {  // color: 各异,  borderColor: '#fff',  borderWidth: 1,  label: {  show: true,  position: 'inside'  // textStyle: null // 默认使用全局文本样式，详见TEXTSTYLE  },  labelLine: {  show: false,  length: 10,  lineStyle: {  // color: 各异,  width: 1,  type: 'solid'  }  }  },  emphasis: {  // color: 各异,  borderColor: 'red',  borderWidth: 5,  label: {  show: true,  formatter: '{b}:{c}%',  textStyle: {  fontSize: 20  }  },  labelLine: {  show: true  }  }  },  data: data.series  }  ]  };  myChart.setOption(option);  window.addEventListener("resize", function () {  myChart.resize();  });  }); |

#### 10. 2最后十名

1. 后端

|  |
| --- |
| @data.route('/getCompanySorted', methods=['GET'])  def get\_company\_sorted():  order = request.args.get("order")  if order == "1":  data = db.session.query(CompanySorted).order\_by(CompanySorted.score.desc()).all()[0:5]  else:  data = db.session.query(CompanySorted).order\_by(CompanySorted.score.asc()).all()[0:5]  view\_data = {}  view\_data["series"] = []  def build\_view\_data(item):  view\_data["series"].append({"value": item.score, "name": item.city})  [build\_view\_data(item) for item in data]  return json.dumps(view\_data, ensure\_ascii=False) |

1. 前端

|  |
| --- |
| (function (func) {  $.ajax({  url: "/data/getCompanySorted",  type: "GET",  data: {"order": 0},  dataType: "json",  success: function (data) {  func(data);  }  });  })(function (data) {  var myChart = echarts.init(document.getElementById('chart\_14'), 'infographic');  var option = {  tooltip: {  trigger: 'item',  formatter: "{a} <br/>{b} : {c}%"  },  calculable: true,  series: [  {  name: '评分',  type: 'funnel',  x: '10%',  y: 60,  //x2: 80,  y2: 60,  width: '80%',  // height: {totalHeight} - y - y2,  min: 0,  max: 150,  minSize: '0%',  maxSize: '100%',  sort: 'ascending',  gap: 10,  itemStyle: {  normal: {  // color: 各异,  borderColor: '#fff',  borderWidth: 1,  label: {  show: true,  position: 'inside'  // textStyle: null  },  labelLine: {  show: false,  length: 10,  lineStyle: {  // color: 各异,  width: 1,  type: 'solid'  }  }  },  emphasis: {  // color: 各异,  borderColor: 'red',  borderWidth: 5,  label: {  show: true,  formatter: '{b}:{c}%',  textStyle: {  fontSize: 20  }  },  labelLine: {  show: true  }  }  },  data: data.series  }  ]  };  myChart.setOption(option);  window.addEventListener("resize", function () {  myChart.resize();  });  }); |

#### 11.分析划转资金走势

1. 后端

|  |
| --- |
| @data.route('/getBankTrend', methods=['GET'])  def get\_bank\_trend():  data = db.session.query(BankTrend).order\_by(BankTrend.year.asc()).all()  view\_data = {}  view\_data["year"] = []  view\_data["amount"] = []  def build\_view\_data(item):  view\_data["year"].append(item.year)  view\_data["amount"].append(item.amount)  [build\_view\_data(item) for item in data]  return json.dumps(view\_data, ensure\_ascii=False) |

1. 前端

|  |
| --- |
| (function (func) {  $.ajax({  url: "/data/getBankTrend",  type: "GET",  dataType: "json",  success: function (data) {  func(data);  }  });  })(function (data) {  var myChart = echarts.init(document.getElementById('chart\_15'), 'infographic');  var option = {  legend: {  data: ['管理资金'],  textStyle: {  color: '#fff'  },  top: '8%'  },  grid: {  top: '40%',  left: '3%',  right: '4%',  bottom: '3%',  containLabel: true  },  color: ['#FF4949', '#FFA74D', '#FFEA51', '#4BF0FF', '#44AFF0', '#4E82FF', '#584BFF', '#BE4DFF', '#F845F1'],  xAxis: {  type: 'category',  boundaryGap: false,  data: data.year,  splitLine: {  show: false  },  axisLine: {  lineStyle: {  color: '#fff'  }  }  },  yAxis: {  name: '管理资金',  type: 'value',  splitLine: {  show: false  },  axisLine: {  lineStyle: {  color: '#fff'  }  }  },  series: [  {  name: '管理资金',  type: 'line',  data: data.amount  }  ]  };  myChart.setOption(option);  window.addEventListener("resize", function () {  myChart.resize();  });  }); |

#### 12.统计各银行的资金管理量

1. 后端

|  |
| --- |
| @data.route('/getCompanyAmont', methods=['GET'])  def get\_company\_amont():  data = db.session.query(CompanyAmont).all()  view\_data = {}  view\_data["series"] = []  def build\_view\_data(item):  dic = {}  dic["value"] = item.amount  dic["name"] = item.company\_name  view\_data["series"].append(dic)  [build\_view\_data(item) for item in data]  return json.dumps(view\_data, ensure\_ascii=False) |

1. 前端

|  |
| --- |
| (function (func) {  $.ajax({  url: "/data/getCompanyAmont",  type: "GET",  dataType: "json",  success: function (data) {  func(data);  }  });  })(function (data) {  var myChart = echarts.init(document.getElementById('chart\_16'), 'infographic');  var option = {  tooltip: {  trigger: 'item',  formatter: "{a} <br/>{b} : {c} ({d}%)"  },  calculable: true,  series: [  {  name: '资金保有量',  type: 'pie',  radius: '55%',  center: ['50%', '60%'],  data: data.series  }  ]  };  myChart.setOption(option);  window.addEventListener("resize", function () {  myChart.resize();  });  }); |