武汉纺织大学

Web应用开发课程设计

**基于Highcharts组件的数据可视化系统**

**学 院： 数学与计算机学院**

**班 级： 物联网11803**

**姓 名： 尹祁钰**

**学 号： 1804280304**

**指导老师： 聂刚**

**成 绩：**

**完成日期： 2020年12月15日**

目 录

[1 需求分析 1](#_Toc59106536)

[1.1主界面 1](#_Toc59106537)

[1.2获取图表 1](#_Toc59106538)

[2 系统设计 1](#_Toc59106539)

[2.1数据设计 1](#_Toc59106540)

[2.1.1 测试类 1](#_Toc59106541)

[2.1.2 条形图数据库 1](#_Toc59106542)

[2.1.3 饼图数据库 2](#_Toc59106543)

[2.1.4 折线图数据库 2](#_Toc59106544)

[2.2 ER图 3](#_Toc59106545)

[2.3 UML类图（Class Diagram） 3](#_Toc59106546)

[2.3.1 ContryWeather类 3](#_Toc59106547)

[2.3.2 Employee类 4](#_Toc59106548)

[2.3.2 Explore类 5](#_Toc59106549)

[2.3.3 Sale类 6](#_Toc59106550)

[2.4 UML时序图（Sequence Diagram） 8](#_Toc59106551)

[2.4.1 Dao层数据访问 8](#_Toc59106552)

[2.4.2 数据转换servlet层 8](#_Toc59106553)

[2.4.3 服务层CountryWeatherService 8](#_Toc59106554)

[2.4.4 测试类TestContryWeatherDao 8](#_Toc59106555)

[3 系统实现 9](#_Toc59106556)

[3.1 项目结构 9](#_Toc59106557)

[3.2 配置文件 9](#_Toc59106558)

[3.2.1 c3p0-config文件 9](#_Toc59106559)

[3.3 VO类CountryWeather.java 10](#_Toc59106560)

[3.4 DAO接口类IUserDAO.java 11](#_Toc59106561)

[3.5 测试类TestContryWeatherDao.java 11](#_Toc59106562)

[3.6 工具包Util 12](#_Toc59106563)

[3.6.1 DataSourceUtils.java 12](#_Toc59106564)

[3.7 服务层UserService.java 13](#_Toc59106565)

[3.8 控制层CountryWeatherServlet.java 13](#_Toc59106566)

[4 系统测试 14](#_Toc59106567)

[4.1Employe类的json格式 14](#_Toc59106568)

[4.2 Contryweather类的json格式 14](#_Toc59106569)

[4.3 Explore类的json格式 15](#_Toc59106570)

[4.4 Sale类的json格式 15](#_Toc59106571)

[4.5 系统主界面 15](#_Toc59106572)

[5 系统总结 16](#_Toc59106573)

# 1 需求分析

采用highcharts图标库计一个数据可视化系统，用户可通过直观的图表形式来查看可视化的数据，如条形图，柱状图，曲线图，数据均来源于数据库。使用bootstrap框架实现侧边导航栏。

## 1.1主界面

当程序运行时候，显示主界面，分别有四张图：条形图，柱状图，曲线图，饼图。将其显示到网页上，均采用hicharts图标库实现。网站使用bootstrap框架进行导航栏的设计，Bootstrap Affix 插件可以让一个<div>元素漂浮在网页上，可以让该<div>元素随屏滚动，或者固定在指定位置上。

## 1.2获取图表

进入网站主界面之后，每一张图表均从数据库查询对应的数据表，获取每个图标的数据，使用highcharts图表库显示到每个图表上。

# 2 系统设计

## 2.1数据设计

### 2.1.1 测试类

表格

描述已自动生成

### 2.1.2 条形图数据库

表格

描述已自动生成

### 2.1.3 饼图数据库

表格

描述已自动生成

### 2.1.4 折线图数据库

表格

描述已自动生成

## 2.2 ER图

图示

描述已自动生成

## 2.3 UML类图（Class Diagram）

### 2.3.1 ContryWeather类

对于各个城市天气显示模块，共设计如下6个类。

* Vo实体类CountryWeather：与数据库结构进行映射的类。主要由属性，setter, getter方法组成，VO类中的属性与表中的字段相对应，每一个VO类的对象都表示表中的每一条记录。
* DAO接口CountryWeatherDao：主要定义操作的接口，定义一系列数据库的原子性操作，例如增删改查（通常称为CRUD）等。创建CountryWeatherDao用于查询数据中所有的记录，使用DbUtils组件来实现
* service业务逻辑实现类CountryWeatherService：对于数据层的原子操作进行整合。还要负责数据库的打开与关闭（不管是否出异常，数据库都要关闭）
* servlet工厂类CountryWeatherServlet：创建CountryWeatherServlet，使用注解指定访问地址是：/CountryWeather。在Servlet调用业务层，将查询到的数据转换成JSON对象，打印到浏览器端。
* test测试类类TestContryWeatherDao：使用JUnit测试一下编写的Dao方法，同时测试一下JSON对象转换是否正确。这里使用flexjson组件进行转换。

各类的结构及类之间的关系如图所示：

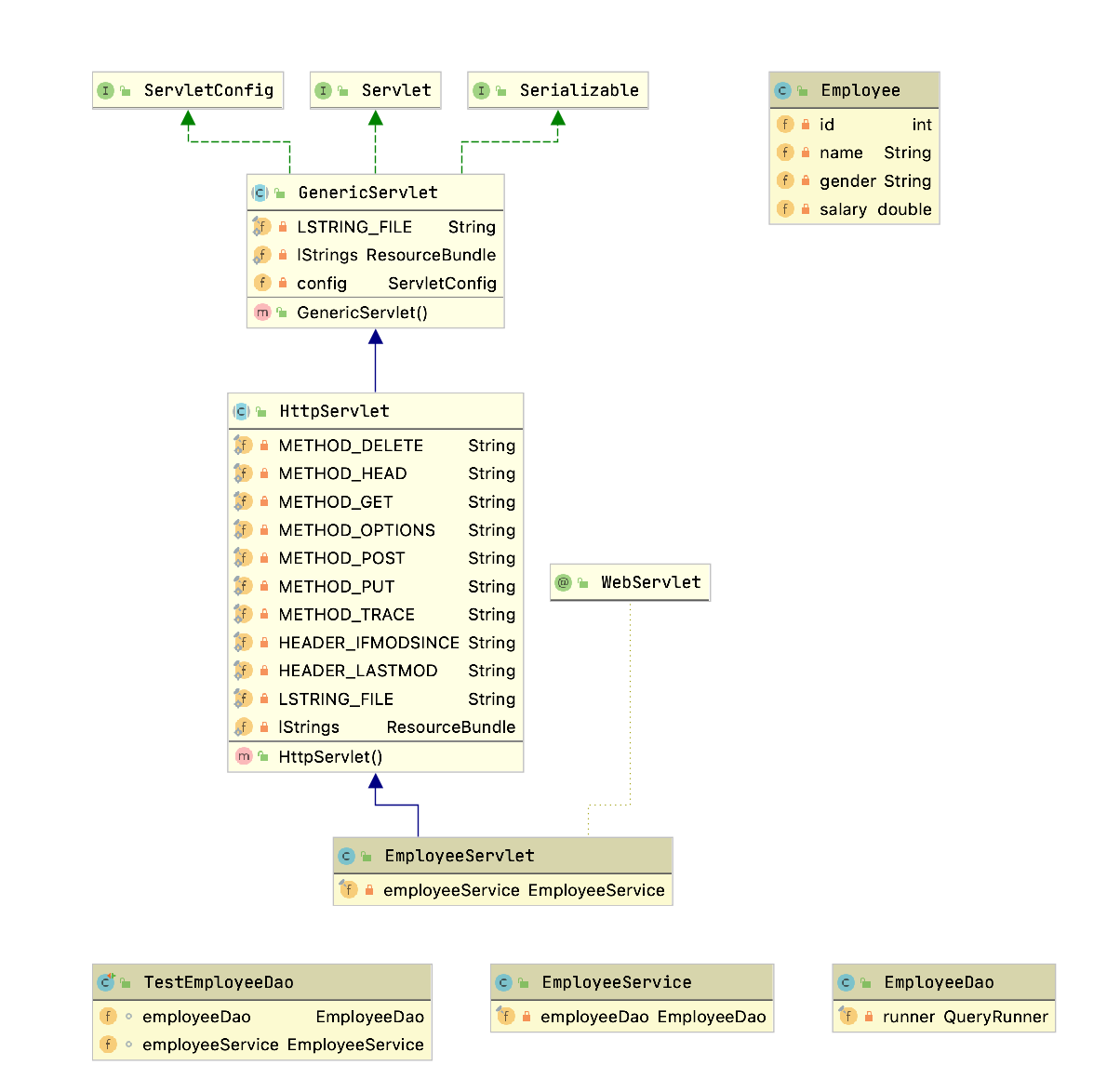
图片包含 图形用户界面

描述已自动生成

### 2.3.2 Employee类

* Vo实体类Employee：与数据库结构进行映射的类。主要由属性，setter, getter方法组成，VO类中的属性与表中的字段相对应，每一个VO类的对象都表示表中的每一条记录。
* DAO接口EmployeeDao：主要定义操作的接口，定义一系列数据库的原子性操作，例如增删改查（通常称为CRUD）等。创建EmployeeDao用于查询数据中所有的记录，使用DbUtils组件来实现
* service业务逻辑实现类EmployeeService：对于数据层的原子操作进行整合。还要负责数据库的打开与关闭（不管是否出异常，数据库都要关闭）
* servlet工厂类CountryWeatherServlet：创建EmployeeServlet，使用注解指定访问地址是：/employee。在Servlet调用业务层，将查询到的数据转换成JSON对象，打印到浏览器端。
* test测试类类TestEmployeeDao：使用JUnit测试一下编写的Dao方法，同时测试一下JSON对象转换是否正确。这里使用flexjson组件进行转换。

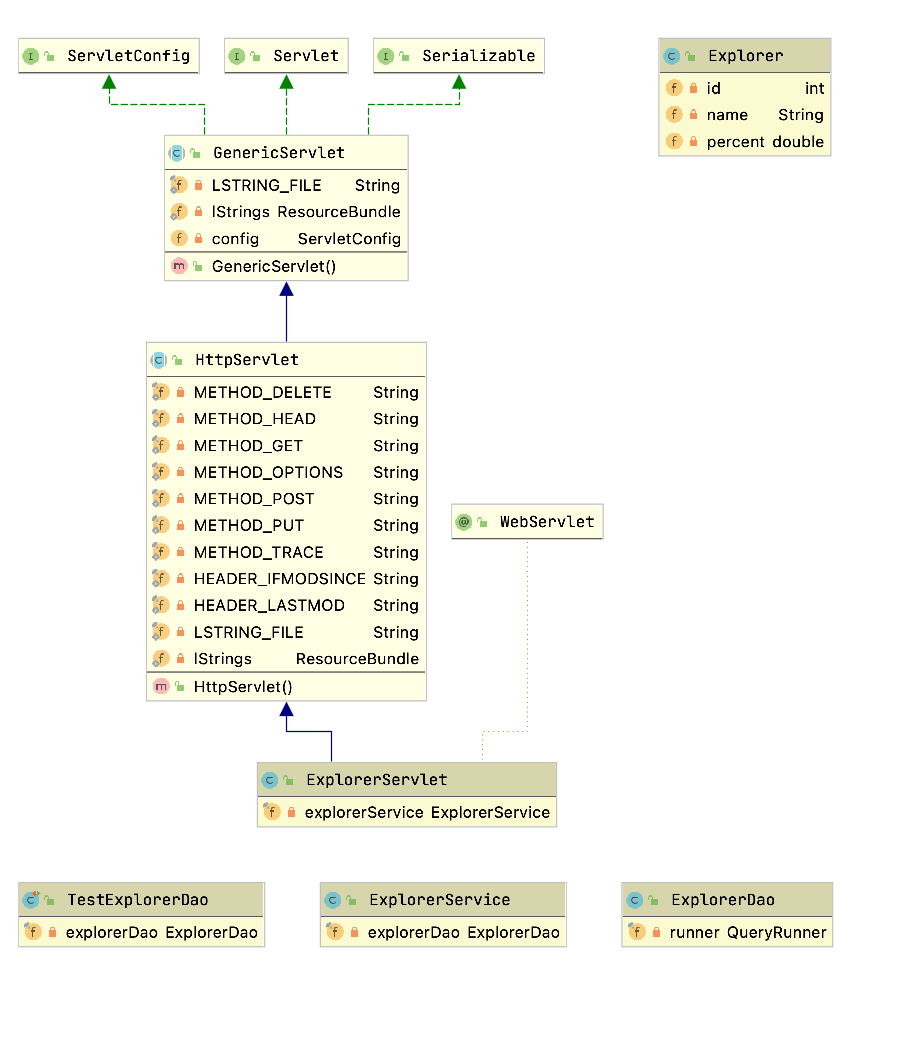
各类的结构及类之间的关系如图所示：



### 2.3.2 Explore类

* Vo实体类Explore：与数据库结构进行映射的类。主要由属性，setter, getter方法组成，VO类中的属性与表中的字段相对应，每一个VO类的对象都表示表中的每一条记录。
* DAO接口ExploreDao：主要定义操作的接口，定义一系列数据库的原子性操作，例如增删改查（通常称为CRUD）等。创建ExploreDao用于查询数据中所有的记录，使用DbUtils组件来实现
* service业务逻辑实现类ExploreService：对于数据层的原子操作进行整合。还要负责数据库的打开与关闭（不管是否出异常，数据库都要关闭）
* servlet工厂类ExploreServlet：创建ExploreServlet，使用注解指定访问地址是：/CountryWeather。在Servlet调用业务层，将查询到的数据转换成JSON对象，打印到浏览器端。
* test测试类类ExploreDao：使用JUnit测试一下编写的Dao方法，同时测试一下JSON对象转换是否正确。这里使用flexjson组件进行转换。

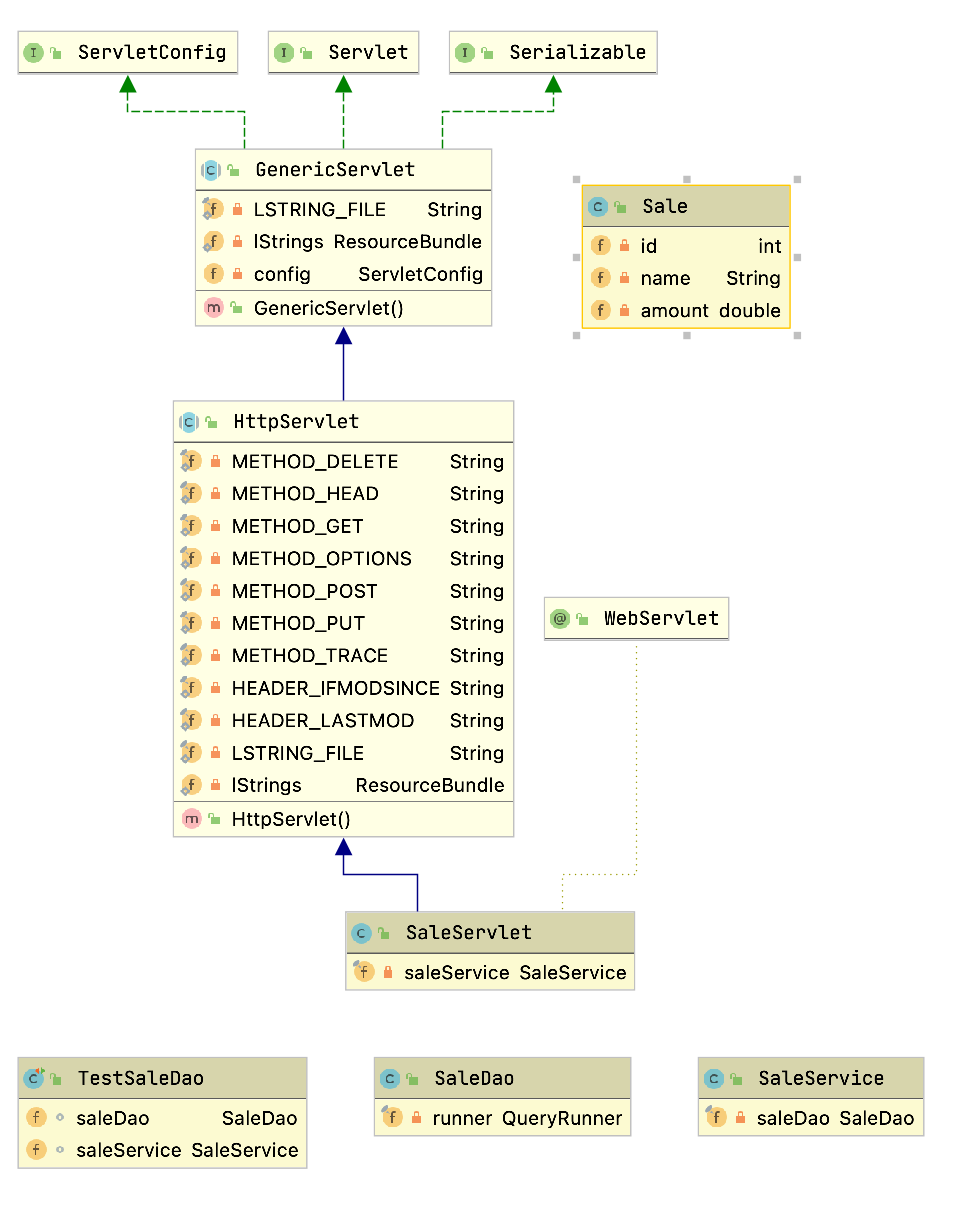
各类的结构及类之间的关系如图所示：



### 2.3.3 Sale类

* Vo实体类Sale：与数据库结构进行映射的类。主要由属性，setter, getter方法组成，VO类中的属性与表中的字段相对应，每一个VO类的对象都表示表中的每一条记录。
* DAO接口SaleDao：主要定义操作的接口，定义一系列数据库的原子性操作，例如增删改查（通常称为CRUD）等。创建SaleDao用于查询数据中所有的记录，使用DbUtils组件来实现
* service业务逻辑实现类SaleService：对于数据层的原子操作进行整合。还要负责数据库的打开与关闭（不管是否出异常，数据库都要关闭）
* servlet工厂类SaleServlet：创建CountryWeatherServlet，使用注解指定访问地址是：/CountryWeather。在Servlet调用业务层，将查询到的数据转换成JSON对象，打印到浏览器端。
* test测试类类TestSaleDao：使用JUnit测试一下编写的Dao方法，同时测试一下JSON对象转换是否正确。这里使用flexjson组件进行转换。

各类的结构及类之间的关系如图所示：



## 2.4 UML时序图（Sequence Diagram）

以CountryWeather类为例，其他类同理。

### 2.4.1 Dao层数据访问

图表, 图示, 箱线图

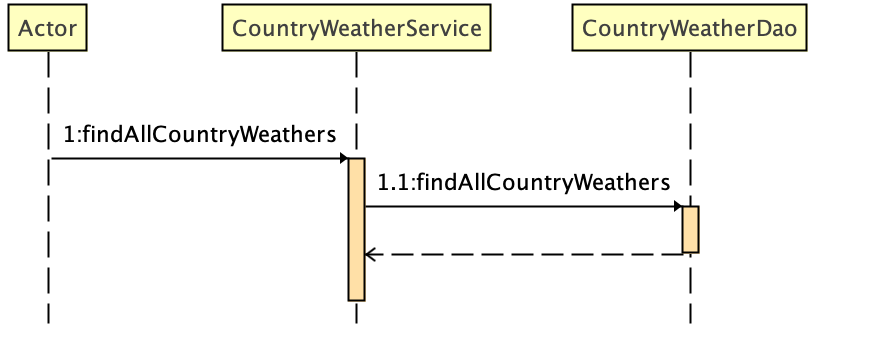
描述已自动生成

### 2.4.2 数据转换servlet层

图示

描述已自动生成

### 2.4.3 服务层CountryWeatherService



### 2.4.4 测试类TestContryWeatherDao

图示

描述已自动生成

# 3 系统实现

## 3.1 项目结构

图形用户界面, 应用程序

描述已自动生成

## 3.2 配置文件

在项目下新建Source Folder，名为resources

### 3.2.1 c3p0-config文件

C3P0是一个开源的JDBC连接池,它实现了数据源和JNDI绑定,支持JDBC3规范和JDBC2的标准扩展。目前使用它的开源项目有Hibernate、Spring等。

<?xml version="1.0" encoding="UTF-8"?>

<c3p0-config>

<!-- 默认配置 -->

<default-config>

<!-- 配置数据库连接信息 -->

<property name="user">root</property>

<property name="password">root</property>

<property name="jdbcUrl">jdbc:mysql://localhost:3306/test</property>

<property name="driverClass">com.mysql.jdbc.Driver</property>

</default-config>

</c3p0-config>

## 3.3 VO类CountryWeather.java

描述该文件作用，并对里面的关键代码进行解释

package com.yqy.vo;

public class CountryWeather {

private int id;

private String name;

private int month;

private double data;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getMonth() {

return month;

}

public void setMonth(int month) {

this.month = month;

}

public double getData() {

return data;

}

public void setData(double data) {

this.data = data;

}

@Override

public String toString() {

return "CountryWeather{" +

"id=" + id +

", name='" + name + '\'' +

", month=" + month +

", data=" + data +

'}';

}

}

## 3.4 DAO接口类IUserDAO.java

描述该文件作用，并对里面的关键代码进行解释

public class CountryWeatherDao {

private final QueryRunner runner = new QueryRunner(DataSourceUtils.getDataSource());

public List<CountryWeather> findAllCountryWeathers() {

try {

return runner.query("select \* from country\_weather", new BeanListHandler<>(CountryWeather.class));

} catch (SQLException e) {

e.printStackTrace();

throw new RuntimeException(e);

}

}

}

## 3.5 测试类TestContryWeatherDao.java

描述该文件作用，并对里面的关键信息进行解释

public class TestContryWeatherDao {

private final CountryWeatherDao countryWeatherDao = new CountryWeatherDao();

@Test

public void testFindall(){

List<CountryWeather>countryWeathers = countryWeatherDao.findAllCountryWeathers();

for (CountryWeather countryWeather:countryWeathers){

System.out.println(countryWeather);

}

}

@Test

public void testToJSON(){

List<CountryWeather>countryWeathers = countryWeatherDao.findAllCountryWeathers();

JSONSerializer serializer = new JSONSerializer();

serializer.exclude("class");

String serialize = serializer.serialize(countryWeathers);

System.out.println(serialize);

}

}

## 3.6 工具包Util

### 3.6.1 DataSourceUtils.java

C3P0创建数据源工具类

package com.itheima.utils;

import com.mchange.v2.c3p0.ComboPooledDataSource;

import javax.sql.DataSource;

/\*\*

\* 数据源的工具类

\*/

public class DataSourceUtils {

/\*\*

\* 创建私有静态数据源成员变量

\*/

private static ComboPooledDataSource ds = new ComboPooledDataSource();

/\*\*

\* 创建公有的得到数据源的方法

\*/

public static DataSource getDataSource() {

return ds;

}

}

## 3.7 服务层UserService.java

描述该文件作用，并对里面的关键代码进行解释

public class CountryWeatherService {

private final CountryWeatherDao countryWeatherDao = new CountryWeatherDao();

public List<CountryWeather> findAllCountryWeathers(){

return countryWeatherDao.findAllCountryWeathers();

}

}

## 3.8 控制层CountryWeatherServlet.java

这里对从数据库取出来的数据进行格式转化

@WebServlet("/CountryWeather")

public class CountryWeatherServlet extends HttpServlet {

private final CountryWeatherService countryWeatherService = new CountryWeatherService();

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("text/json;charset=utf-8");

PrintWriter out = response.getWriter();

List<CountryWeather> countryWeathers = countryWeatherService.findAllCountryWeathers();

List<Map<String,Object>> data = new ArrayList<>();

Set<String> names = new HashSet<>();

for(CountryWeather c : countryWeathers){

names.add(c.getName());

}

int k = 0;

for(String name : names){

Map<String,Object> map = new HashMap<>();

List<Object> ts = new ArrayList<>();

if (k > 0){

ts.clear();

k = 0;

}

for(CountryWeather c : countryWeathers){

if (name.equals(c.getName())){

ts.add(c.getData());

}

}

map.put("name",name);

map.put("data",ts);

data.add(map);

k++;

}

ObjectMapper objectMapper = new ObjectMapper();

String res = objectMapper.writeValueAsString(data);

out.print(res);

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

this.doPost(request,response);

}

}

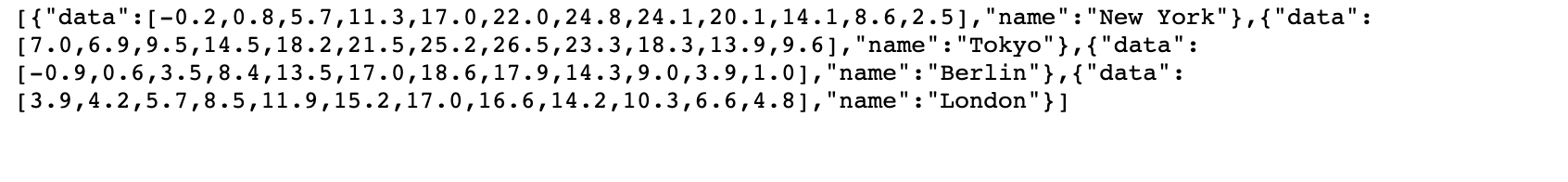
# 4 系统测试

## 4.1Employe类的json格式

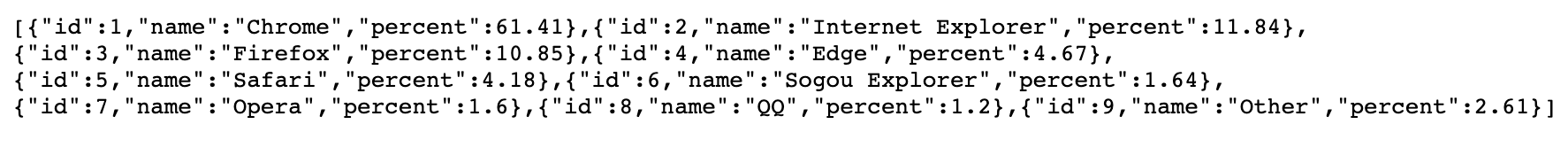
文本

描述已自动生成

## 4.2 Contryweather类的json格式



## 4.3 Explore类的json格式



## 4.4 Sale类的json格式

文本

描述已自动生成

## 4.5 系统主界面

图形用户界面, 图表, 条形图

描述已自动生成

图表, 条形图

描述已自动生成

图表

描述已自动生成

# 5 系统总结

很庆幸这次课设能拿到这个题目，刚好是我完全没有接触过的领域，大数据展示，也很庆幸能在有限的时间内，能了解到更多有关web图表库的使用，由于我的ssm框架并不是掌握的很熟练，于是采用了最基础的mvc架构来完成本次课设。这次课设主要的难点是对于数据的处理，将数据库中的数据转成每一个图表所需要的格式。有三种方式可以实现：1.通过servlet处理数据，2. 在dao层通过sql语句查询转格式，3。通过网页中js代码来将json格式的数据转成图表所需要的格式。这次系统采用了1和3方法来进行数据的处理。在我画折线图的时候从建数据库表就陷入了沉思，如何建表才能让后端的工作量和前端的工作量最小。Highcharts 包含标题（Title）、坐标轴（Axis）、数据列（Series）、数据提示框（Tooltip）、图例（Legend）、版权标签（Credits）等。大多数是指数据格式的问题，如果在后端生成json格式的时候就把问题都解决了，那么开发也会变得迎刃而解，当完成了系统后，仍觉得有所不足，在我去图书馆学习的时候，发现一楼大厅有一个大数据可视面板，上面能实时的显示数据的变化，但是我的数据是不可动的，所以希望后期能完善系统，做出一个可以实时监测的数据可视化面板。最后感谢聂老师倾心撰写的文档和激情昂扬的课堂，让我从一个懵懂无知的java小白摇身一变成为了也会用框架写出网站的入门选手。最后我想给这个学期的我打80分，在javaweb的学习上，对得起自己的努力，并且很开心能遇到一位这么负责，有能力，有技术的老师，还有20分，10分献未来，10分敬过往，最后希望自己随时保持90分的状态，60分的谦卑。我会在学习的路上不断的积累，不断夯实基础的同时摄取最细腻的技术，来创造无限的可能。