武汉纺织大学

Web应用开发课程设计

**使用highchart组件生成规范图表**

**学 院： 数学与计算机学院**

**班 级： 软件11801**

**姓 名： 胡亚州**

**学 号： 1802220124**

**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](#_Toc4582)

[1.1生成、刷新条形图 1](#_Toc6313)

[1.2生成、刷新折线图 1](#_Toc32295)

[1.3生成、刷新饼图 1](#_Toc14791)

[2 系统设计 1](#_Toc30285)

[2.1用户用例图 1](#_Toc6011)

[2.2 ER图 2](#_Toc218)

[2.3 UML类图（Class Diagram） 2](#_Toc10870)

[2.4 UML时序图（Sequence Diagram） 4](#_Toc22317)

[2.4.1 系统登录 4](#_Toc1110)

[2.4.2 \*\*\*\*\*\*模块 4](#_Toc4896)

[2.4.3 \*\*\*\*\*\*模块 4](#_Toc21348)

[2.5 UML活动图（Activity Diagram） 4](#_Toc32357)

[2.5.1 登陆 4](#_Toc25831)

[2.5.2 收银 5](#_Toc14846)

[2.5.3 \*\*\*模块 6](#_Toc9041)

[2.5.4 \*\*\*\*模块 6](#_Toc26181)

[3 系统实现 7](#_Toc13971)

[3.1 项目结构 7](#_Toc30810)

[3.2 配置文件 7](#_Toc24669)

[3.2.1 jdbc.properties文件 7](#_Toc4093)

[3.2.2 log4j.properties文件 7](#_Toc15126)

[3.2.3 mybatis-config.xml文件 8](#_Toc30991)

[3.3 VO类User.java 8](#_Toc31178)

[3.4 DAO接口类IUserDAO.java 9](#_Toc7765)

[3.5 接口映射文件UserMapper.xml 9](#_Toc30826)

[3.6 工具包Util 9](#_Toc7997)

[3.6.1 MybatisUtils.java 9](#_Toc9988)

[3.6.2 MD5Util.java 10](#_Toc2207)

[3.7 服务层UserService.java 11](#_Toc28650)

[3.8 用户界面Driver.java 12](#_Toc28898)

[3.9 \*\*\*\*\*\*.java 12](#_Toc30686)

[4 系统测试 12](#_Toc31210)

[5 系统总结 12](#_Toc20663)

# 1 需求分析

设计一个从数据库获取数据并生成规范格式的图表的网页系统（ChartDisplay系统），实现“生成、刷新条形图(按销售额从大到小排列)”、“生成、刷新折线图”、“生成、刷新饼图”等功能，具体要求如下：

## 1.1生成、刷新条形图**(按销售额从大到小排列)**

当程序运行时，在网页上生成一个条形图，数据来自本地数据库；

图表右上方显示一个按钮，点击按钮，刷新饼图。

## 1.2生成、刷新折线图

当程序运行时，在网页上生成一个条形图，数据来自本地数据库；

图表右上方显示一个按钮，点击按钮，刷新饼图。

## 1.3生成、刷新饼图

当程序运行时，在网页上生成一个饼图，数据来自本地数据库；

图表右上方显示一个按钮，点击按钮，刷新饼图。

# **2 系统设计**

## 2.1用户用例图

系统分为访问者，管理员两个角色，各角色的用户用例如下图2-1所示：

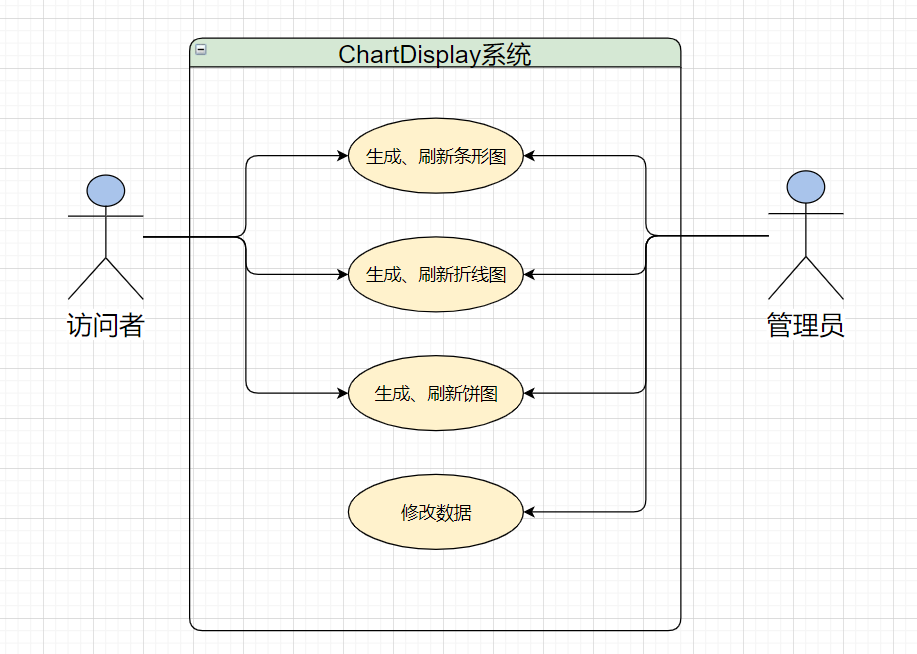


图2-1 用户用例图

## 2.2 UML类图（Class Diagram）

各类的结构及类之间的关系如图2-2-1、2-2-2、2-2-3所示：

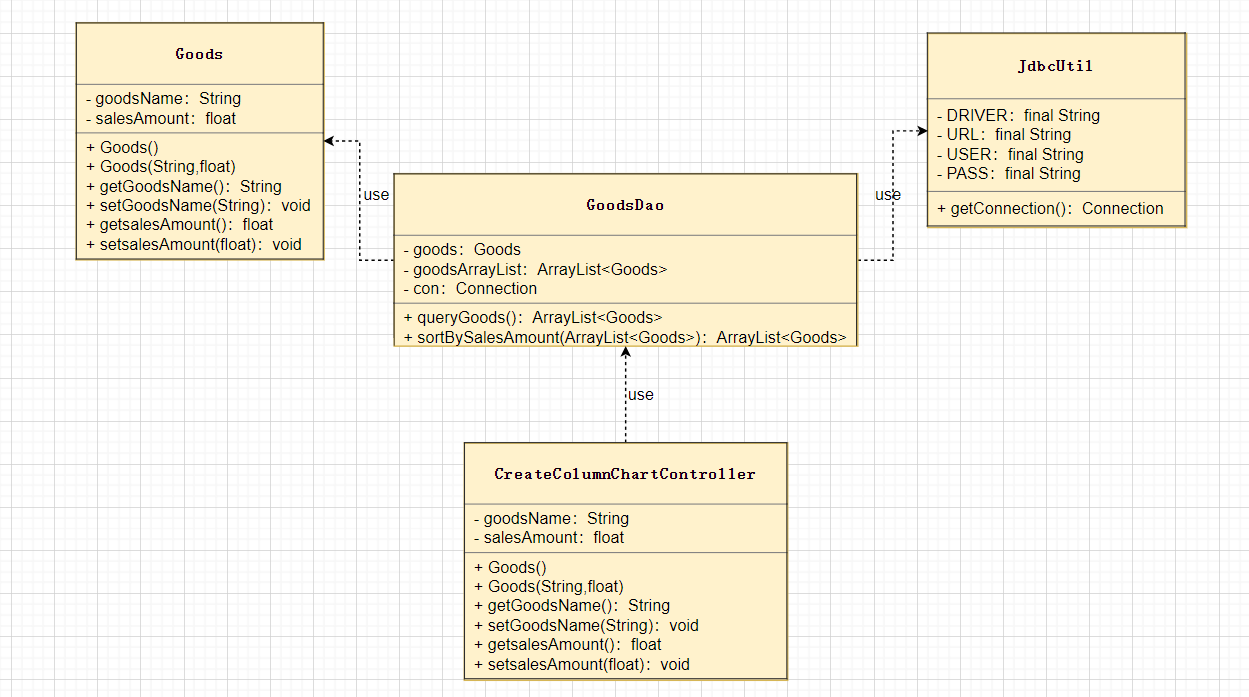


图2-2-1 功能1类图

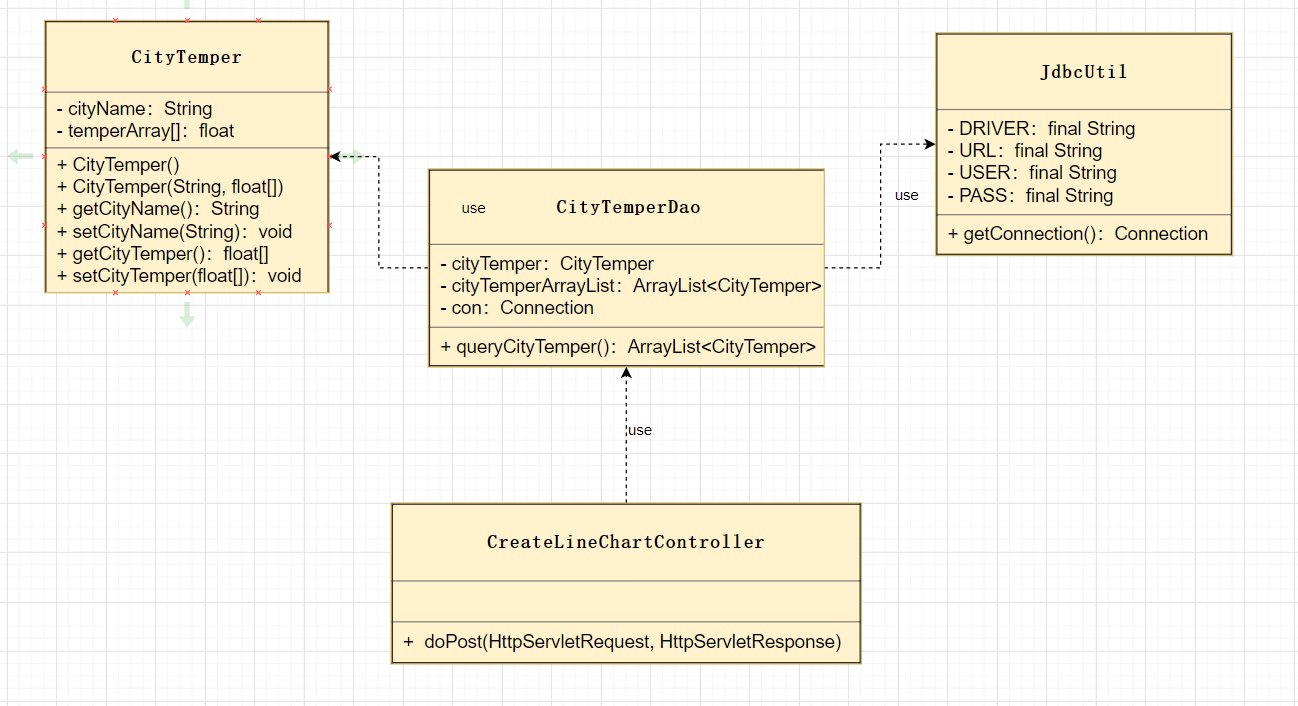


图2-2-2 功能2类图

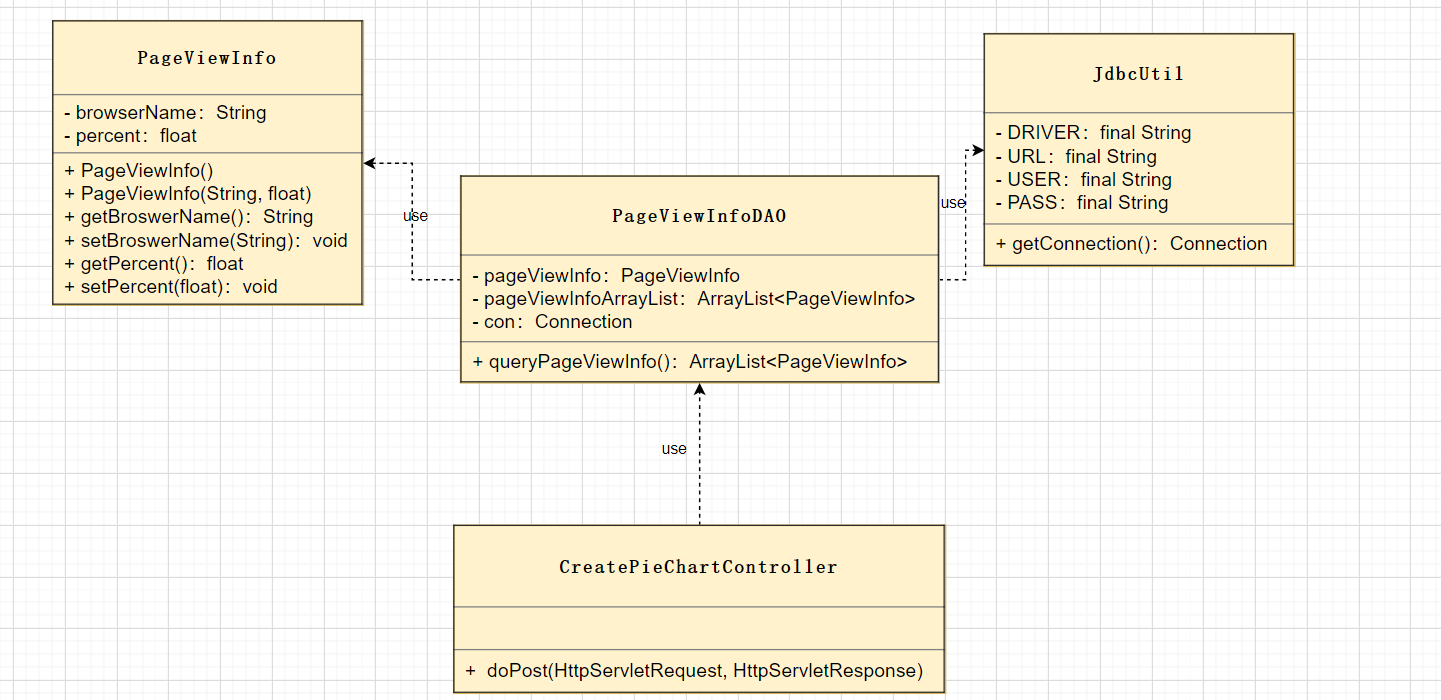
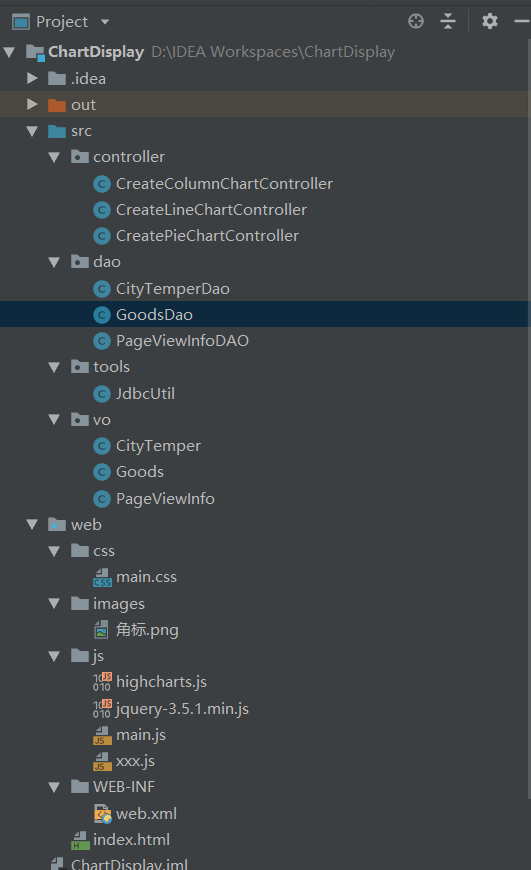


图2-2-3 功能3类图

# **3 系统实现**

## 3.1 项目结构



## 3.2 配置文件

web.xml文件：

<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"  
 version="4.0">  
</web-app>

## 3.3 VO类

**3.3.1 Goods.java**

商品信息（商品名、销售额）类

public class Goods {  
  
 private String goodsName; //商品名  
 private float salesAmount; //销售额  
  
 public Goods() {  
 }  
  
 public Goods(String goodsName, float salesAmount) {  
 this.goodsName = goodsName;  
 this.salesAmount = salesAmount;  
 }  
  
 public String getGoodsName() {  
 return goodsName;  
 }  
  
 public void setGoodsName(String goodsName) {  
 this.goodsName = goodsName;  
 }  
  
 public float getsalesAmount() {  
 return salesAmount;  
 }  
  
 public void setsalesAmount(float salesAmount) {  
 this.salesAmount = salesAmount;  
 }

**3.3.2 CityTemper.java**

城市气温类（城市名、温度字符串）

public class CityTemper {  
 private String cityName;  
 private float temperArray[];  
  
 public CityTemper() {  
 }  
  
 public CityTemper(String cityName, float[] cityTemper) {  
 this.cityName = cityName;  
 this.temperArray = cityTemper;  
 }  
  
 public String getCityName() {  
 return cityName;  
 }  
  
 public void setCityName(String cityName) {  
 this.cityName = cityName;  
 }  
  
 public float[] getCityTemper() {  
 return temperArray;  
 }  
  
 public void setCityTemper(float[] cityTemper) {  
 this.temperArray = cityTemper;  
 }  
}

**3.3.3 PageViewInfo.java**

浏览量信息类（浏览器名、浏览量占比）

public class PageViewInfo {  
 private String browserName;  
 private float percent;  
  
 public PageViewInfo() {  
 }  
  
 public PageViewInfo(String broswerName, float percent) {  
 this.browserName = broswerName;  
 this.percent = percent;  
 }  
  
 public String getBroswerName() {  
 return browserName;  
 }  
  
 public void setBroswerName(String broswerName) {  
 this.browserName = broswerName;  
 }  
  
 public float getPercent() {  
 return percent;  
 }  
  
 public void setPercent(float percent) {  
 this.percent = percent;  
 }  
}

## 3.4 DAO类

### **3.4.1 GoodsDao.java**

从数据库获取商品信息，并按照商品的销售额从大到小排列在list中。

public class GoodsDao {  
 public ArrayList<Goods> queryGoods() {  
 Goods goods = null;  
 ArrayList<Goods> goodsArrayList = new ArrayList<>();  
 Connection con = JdbcUtil.*getConnection*();  
  
 String sql = "select \* from t\_goods";  
 PreparedStatement ps = null;  
 try {  
 ps = con.prepareStatement(sql);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 ResultSet rs = null;  
 try {  
 rs = ps.executeQuery();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 try {  
 while (rs.next()) {  
 String goodsName = rs.getString("goodsName");  
 float salesAmount = (float) rs.getInt("salesAmount") / 10000;  
 goods = new Goods(goodsName, salesAmount);  
 goodsArrayList.add(goods);  
 }  
 goodsArrayList = sortBySalesAmount(goodsArrayList);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
  
 JdbcUtil.*closeAll*(rs, ps, con);  
  
 return goodsArrayList;  
 }  
  
  
 public ArrayList<Goods> sortBySalesAmount(ArrayList<Goods> list) {  
 for (int i = 0; i < list.size() - 1; i++) { //0~n-1  
 for (int j = 0; j < list.size() - 1 - i; j++) {  
 if (list.get(j + 1).getsalesAmount() > list.get(j).getsalesAmount()) {  
 Goods temp = list.get(j);  
 list.set(j,list.get(j+1));  
 list.set(j+1,temp);  
 }  
 }  
 }  
 return list;  
 }  
  
}

**3.4.2 CityTemperDao.java**

从数据库获取城市气温信息

public class CityTemperDao {  
 public ArrayList<CityTemper> queryCityTemper(){  
 CityTemper cityTemper = null;  
 ArrayList<CityTemper> cityTemperArrayList = new ArrayList<>();  
 Connection con = JdbcUtil.*getConnection*();  
  
 String sql = "select \* from t\_city\_temper";  
 PreparedStatement ps = null;  
 try {  
 ps = con.prepareStatement(sql);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 ResultSet rs = null;  
 try {  
 rs = ps.executeQuery();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 try {  
 while (rs.next()) {  
 String cityName = rs.getString("cityName");  
 String[] temperStrArray = rs.getString("temperStr").split(",");  
 //把气温字符串数组 转化成 气温浮点数数组  
 float[] temperArray = new float[12];  
 //System.out.println(temperStrArray.length);  
 for (int i = 0; i < temperStrArray.length; i++) {  
 temperArray[i] = Float.*parseFloat*(temperStrArray[i]);  
 }  
 //float[] temperArray2 = {1,2,3}; 测试用例  
 cityTemper = new CityTemper(cityName,temperArray);  
 cityTemperArrayList.add(cityTemper);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
  
 JdbcUtil.*closeAll*(rs, ps, con);  
  
 return cityTemperArrayList;  
  
  
 }  
}

**3.4.3 PageViewInfoDao.java**

从数据库获取浏览量信息

public class PageViewInfoDAO {  
 public ArrayList<PageViewInfo> queryPageViewInfo() {  
 PageViewInfo pageViewInfo = null;  
 ArrayList<PageViewInfo> pageViewInfoArrayList = new ArrayList<>();  
 Connection con = JdbcUtil.*getConnection*();  
  
 String sql = "select \* from t\_pageview\_info";  
 PreparedStatement ps = null;  
 try {  
 ps = con.prepareStatement(sql);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 ResultSet rs = null;  
 try {  
 rs = ps.executeQuery();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 try {  
 while (rs.next()) {  
 String browserName = rs.getString("browserName");  
 float percent = rs.getFloat("percent");  
 pageViewInfo = new PageViewInfo(browserName, percent);  
 pageViewInfoArrayList.add(pageViewInfo);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
  
 JdbcUtil.*closeAll*(rs, ps, con);  
  
 return pageViewInfoArrayList;  
 }  
}

## 3.6 工具包 JdbcUtil.java

连接数据库

public class JdbcUtil {  
 private static final String *DRIVER* = "com.mysql.jdbc.Driver";  
 private static final String *URL* = "jdbc:mysql://localhost:3306/chartdisplay?useUnicode=true&characterEncoding=utf-8";  
 private static final String *USER* = "root";  
 public static final String *PASS* = "410966";  
  
 public static Connection getConnection() {  
 try {  
 Class.*forName*(*DRIVER*);  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 Connection con = null;  
 try {  
 con = DriverManager.*getConnection*(*URL*, *USER*, *PASS*);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return con;  
 }  
  
 public static void closeAll(ResultSet rs, PreparedStatement pst ,Connection con) {  
 if(rs != null) {  
 try {  
 rs.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 if(pst != null) {  
 try {  
 pst.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 if(con != null) {  
 try {  
 con.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

## 3.7 服务层

**3.7.1 CreateColumnChartController**

把获取到的商品信息的list集合转成json串

@WebServlet(urlPatterns = "/creatColumnChart.do")  
public class CreateColumnChartController extends HttpServlet {  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 String jsonStr;  
 GoodsDao goodsDao = new GoodsDao();  
 ArrayList<Goods> goodsArrayList = goodsDao.queryGoods();  
 jsonStr = new Gson().toJson(goodsArrayList);  
  
 response.setContentType("text/html;charset=utf-8");  
 PrintWriter out = response.getWriter();  
  
 System.*out*.print("1.条形图信息：");  
 System.*out*.println(jsonStr);  
 System.*out*.println();  
  
 out.print(jsonStr);  
 out.flush();  
 out.close();  
 }  
}

**3.7.2 CreateLineChartController**

把获取到的城市气温信息的list集合转成json串

@WebServlet(urlPatterns = "/createLineChart.do")  
public class CreateLineChartController extends HttpServlet {  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 String jsonStr;  
 CityTemperDao cityTemperDao = new CityTemperDao();  
 ArrayList<CityTemper> cityTemperArrayList = cityTemperDao.queryCityTemper();  
 jsonStr = new Gson().toJson(cityTemperArrayList);  
  
 response.setContentType("text/html;charset=utf-8");  
 PrintWriter out = response.getWriter();  
  
 System.*out*.print("2.折线图信息：");  
 System.*out*.println(jsonStr);  
 System.*out*.println();  
  
 out.print(jsonStr);  
 out.flush();  
 out.close();  
 }  
}

**3.7.3 CreatePieChartController**

把获取到的浏览量信息的list集合转成json串

@WebServlet(urlPatterns = "/createPieChart.do")  
public class CreatePieChartController extends HttpServlet {  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 String jsonStr;  
 PageViewInfoDAO pageViewInfoDAO = new PageViewInfoDAO();  
 ArrayList<PageViewInfo> pageViewInfoArrayList = pageViewInfoDAO.queryPageViewInfo();  
  
 //转成json  
 jsonStr = new Gson().toJson(pageViewInfoArrayList);  
  
 response.setContentType("text/html;charset=utf-8");  
 PrintWriter out = response.getWriter();  
  
 System.*out*.print("3.饼图信息：");  
 System.*out*.println(jsonStr);  
 System.*out*.println();  
  
 out.print(jsonStr);  
 out.flush();  
 out.close();  
 }  
}

## 3.8 用户界面

### **3.8.1 index.html**

网页文件

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8">  
 <title>PZisGuard的主页</title>  
 <script src="js/jquery-3.5.1.min.js"></script>  
 <script src="js/highcharts.js"></script>  
 <script src="js/main.js"></script>  
 <link href="css/main.css" rel="stylesheet" type="text/css"/>  
</head>  
  
<body>  
 <!-- 图表容器 DOM -->  
 <div class="div1">  
 <img src="images/角标.png" alt="Study Website" width="170px" >  
 </div>  
 <div><font size="8px" color="greenyellow">Charts Display</font></div>  
 <br /><div class="line1"/><br />  
   
 <div><button class="updata" onclick="createColumnChart()">刷&ensp;&ensp;新</button><br/></div>  
 <div id="container1" style="min-width:400px;height:600px;"></div>  
   
 <br /><div class="line2"/><br />  
   
 <div><button class="updata" onclick="createLineChart()">刷&ensp;&ensp;新</button><br/></div>  
 <div id="container2" style="margin-left: 15%;width: 70%;height:600px"></div>  
   
 <br /><div class="line2"/><br />  
   
 <div><button class="updata" onclick="createPieChart()">刷&ensp;&ensp;新</button><br/></div>  
 <div id="container3" style="height: 600px;"></div>  
   
</body>  
</html>

**3.8.2 main.js**

处理后端数据并生成图表

**var** xmlHttpRequest;  
//创建XMLHTTPRequest对象  
**function** createXMLRequest() {  
 **if** (window.XMLHttpRequest) {  
 xmlHttpRequest = **new** XMLHttpRequest();  
 } **else** {  
 xmlHttpRequest = **new** ActiveXObject("Microsoft.XMLHTTP");  
 }  
}  
  
**function** createColumnChart(){  
 $.ajax({  
 type: 'post',//HTTP请求类型  
 url: "creatColumnChart.do",  
 data: {},  
 dataType: "json",  
 success: **function** (response) {  
 **var** xDate = [];  
 **var** yDate = [];  
 **for**(**var** i=0;i<response.length;i++){  
 xDate.push(response[i].goodsName);  
 yDate.push(response[i].salesAmount);  
 }  
 // 图表配置  
 **var** options = {  
 chart: {  
 type: 'column',  
 inverted: **true**, //调换xy轴  
 colorByPoint: **true**,  
 animation: **false** },  
 title: {  
 text: '按商品销售统计', // 标题  
 style:{  
 color: '#ff0000',  
 fontSize: "28px",  
 fontWeight: "blod"  
 }  
 },  
 xAxis: {  
 categories: xDate,// x 轴分类  
 labels:{  
 style:{  
 fontSize: "15px"  
 }  
 }  
 },  
 yAxis: {  
 title: {  
 text: '销售金额（万元）'// y 轴标题  
 },  
 style:{  
 fontSize:"16px"  
 }  
 },  
 plotOptions: {  
 column: {  
 colorByPoint:**true** }  
 },  
 credits:{  
 text: 'www.PZisGuard.com',  
 href: 'http://localhost:8080/ChartDisplay/',  
 style:{  
 fontSize:"20px",  
 color:"blue"  
 }  
 },  
 series: [{  
 name: '',  
 data: yDate  
 }]  
 }  
 Highcharts.chart('container1', options);  
 }  
 });  
}  
  
  
**function** createLineChart(){  
 $.ajax({  
 type: 'post',//HTTP请求类型  
 url: "createLineChart.do",  
 data: {},  
 dataType: "json",  
 success: **function** (response) {  
 //图表配置  
 **var** options = {  
 chart: {  
 type: 'line'  
 },  
 title: {  
 text: '不同城市的月平均气温',  
 x: -20,  
 style: {  
 color: '#5555ff',  
 fontSize: "30px",  
 fontWeight: "blod"  
 }  
 },  
 subtitle: {  
 text: '数据来源: WorldClimate.com',  
 x: -20,  
 style: {  
 color: '#000000',  
 fontSize: "24px"  
 }  
 },  
  
 xAxis: {  
 categories: ['一月', '二月', '三月', '四月', '五月', '六月', '七月', '八月', '九月', '十月', '十一月', '十二月'],  
 crosshair: {  
 width: 3,  
 color: 'green'  
 },  
 labels: {  
 rotation: -30 //倾斜的角度  
 }  
 },  
 yAxis: {  
 title: {  
 text: '温度 (°C)',  
 style: {  
 fontSize: "20px"  
 }  
 },  
 crosshair: {  
 width: 3,  
 color: 'green'  
 },  
 plotLines: [{  
 value: 0,  
 width: 1,  
 color: '#ff0000'  
 }]  
 },  
 tooltip: {  
 valueSuffix: '°C'  
 },  
 legend: {  
 layout: 'vertical',  
 align: 'right',  
 verticalAlign: 'middle',  
 borderWidth: 0  
 },  
 credits:{  
 enabled: **false** // 禁用版权信息  
 },  
 //数据列  
 series: []  
 }  
 **for**(**var** i=0;i<response.length;i++){  
 **var** data = {"name": response[i].cityName,"data": response[i].temperArray};  
 options.series.push(data);  
 }  
 Highcharts.chart('container2',options);  
 }  
 });  
}  
  
**function** createPieChart(){  
 $.ajax({  
 type: 'post',//HTTP请求类型  
 url: "createPieChart.do",  
 data: {},  
 dataType: "json",  
 success: **function** (response) {  
 //图表配置  
 **var** options = {  
 chart: {  
 type: 'pie'  
 },  
 title: {  
 text: '2014某网站各浏览器浏览量占比',  
 style: {  
 fontSize: "30px"  
 }  
 },  
 tooltip: {  
 headerFormat: '{series.name}<br>',  
 pointFormat: '{point.name}: <b>{point.percentage:.1f}%</b>',  
 style: {  
 fontSize: "17px"  
 }  
 },  
 plotOptions: {  
 pie: {  
 dataLabels: {  
 formatter: **function** () {  
 **return this**.point.name + "：" + **this**.y + "%";  
 },  
 distance: 60,  
 enabled: **true**,  
 style: {  
 fontWeight: 'bold',  
 fontSize: "23px"  
 }  
 }  
 }  
 },  
 series: []  
 }  
 **var** dataArray = **new** Array();  
 **for**(**var** i=0;i<response.length;i++){  
 **var** data = [response[i].browserName,response[i].percent];  
 dataArray.push(data);  
 }  
 **var** se = {"name": "浏览量占比","data": dataArray};  
 options.series.push(se);  
 Highcharts.chart('container3',options);  
 }  
 });   
}  
  
**function** createCharts(){ //先初始化表格，即第一次生成  
 createColumnChart();  
 createLineChart();  
 createPieChart();  
 //setTimeout('createCharts()',10000); //10秒刷新一次  
}  
  
  
$(document).ready(**function** () {  
 createCharts();  
});

**3.8.3 main.css**

网页的样式文件

.line1{  
 width:100%;  
 border-top:6px black dashed;  
}  
  
.line2{  
 width:100%;  
 border-top:2px black dashed;  
}  
  
.div1  
{  
 top: 5%;  
 left: 10%;  
}  
  
.updata{  
 width:110px;  
 height:45px;  
 background-color:blue;  
 color: white;font-size: 24px;  
 cursor: pointer;  
 margin-left: 90%;  
 margin-top: 1%;  
}

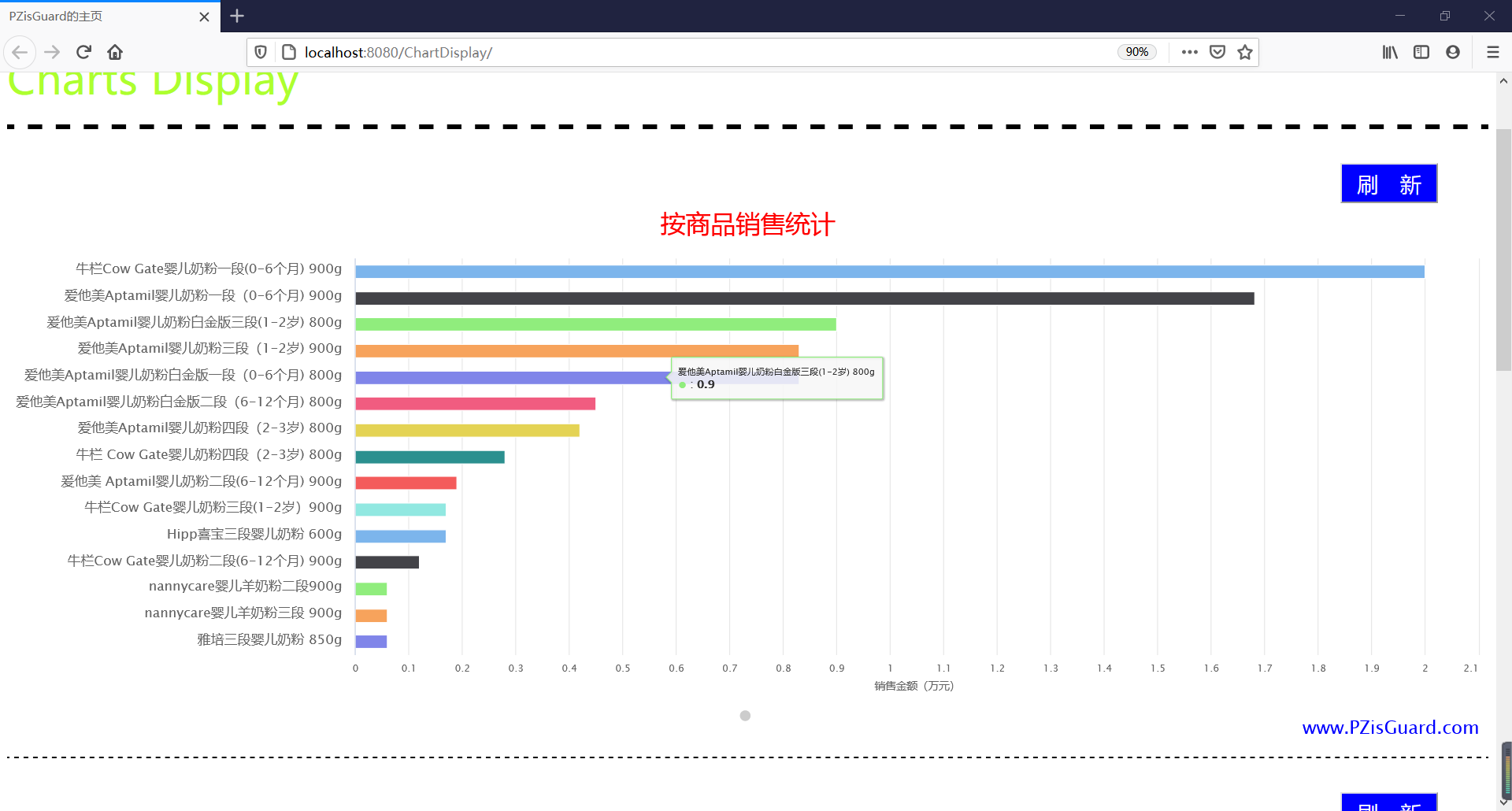
**3.8.4 jquery-3.5.1.min.js**

**3.8.5 highchart.js**

# **4 系统测试**

**4.1 生成、刷新条形图(按销售额从大到小排列)**

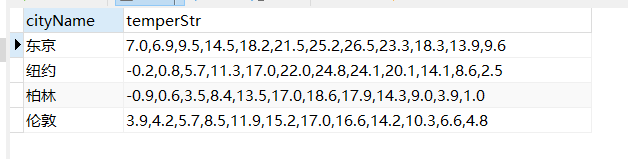


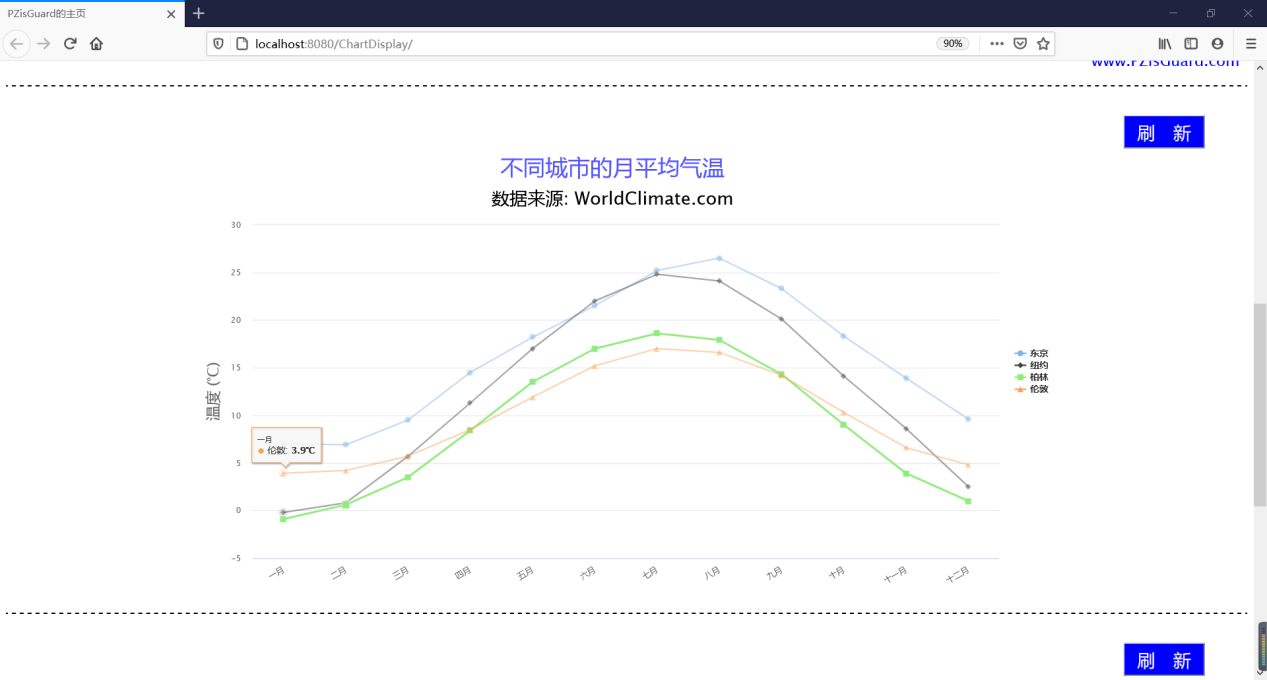


更新数据库后、点击“刷新”按钮：



**4.2 生成、刷新折线图**

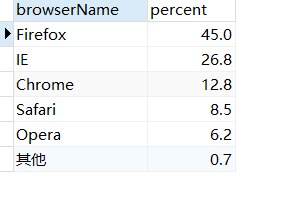


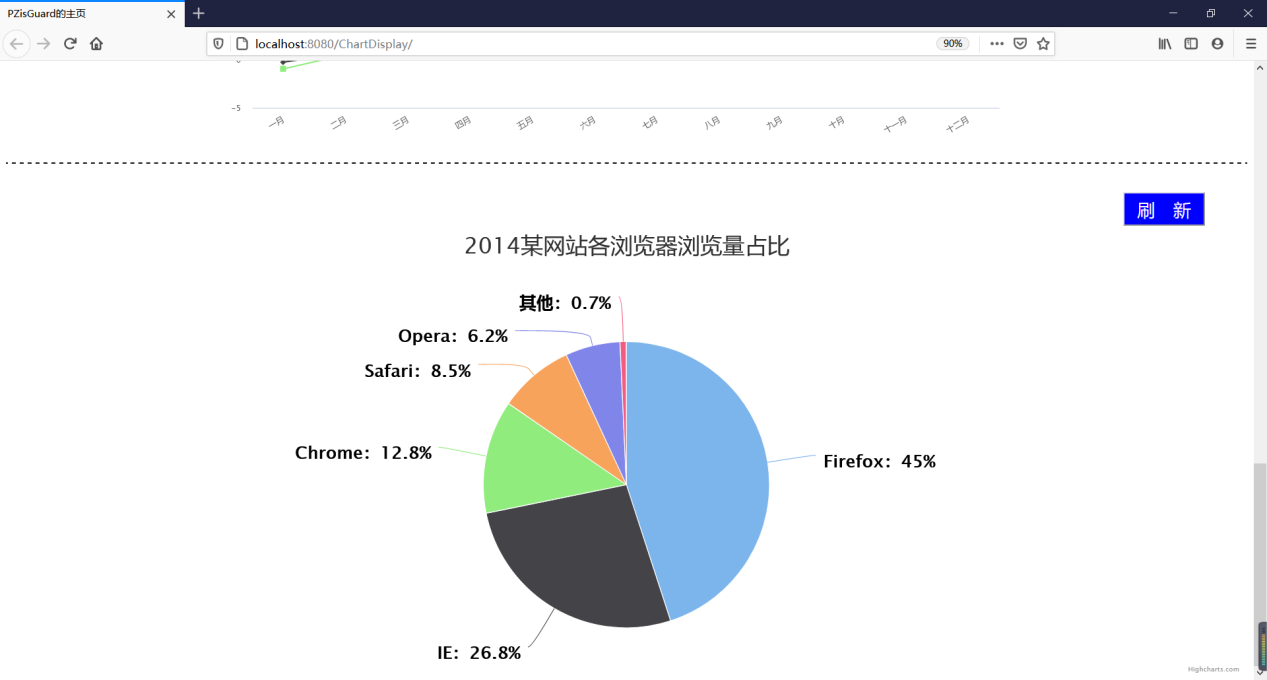


更新数据库后、点击“刷新”按钮：

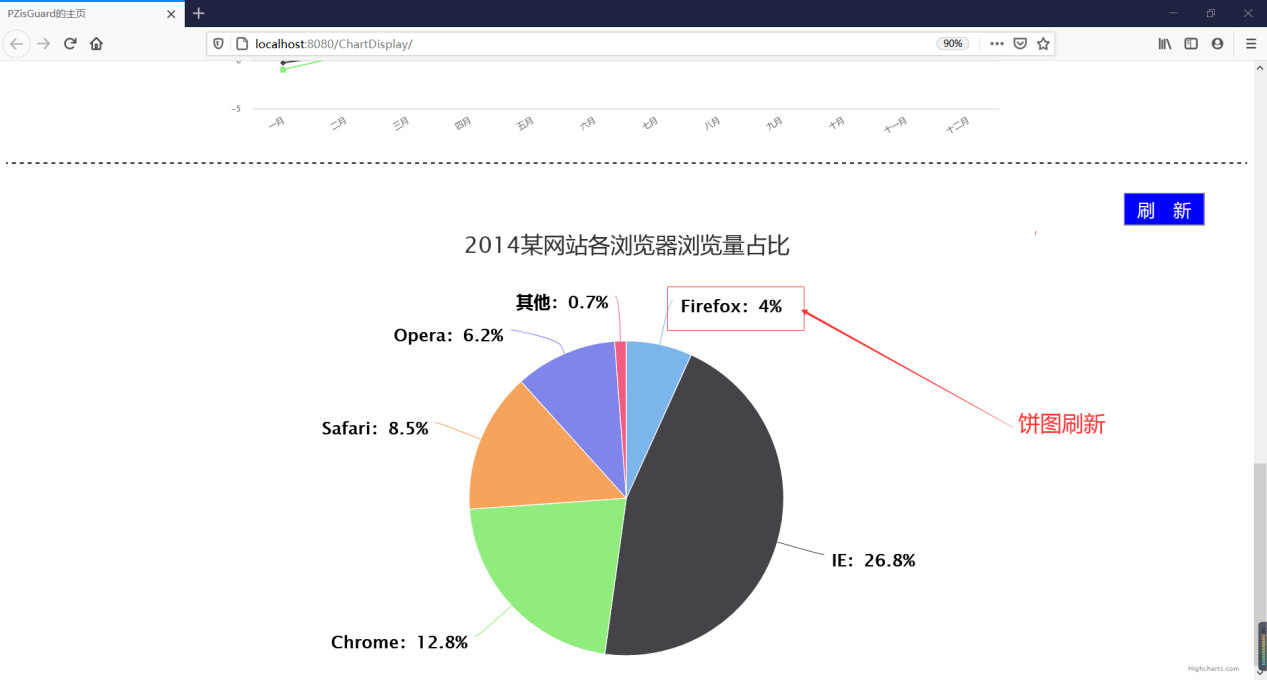


**4.3 生成、刷新饼图**





更新数据库后、点击“刷新”按钮：



# **5 系统总结**

主要问题为后端取到数据库文件后数据的格式化处理、以及前端对于后端传来的json数据的处理。后端Servlet从数据库取到数据，转化成对应的vo类并压入相应的list中，然后转成json串；前端通过ajax获取后端Servlet的数据，并处理成符合highchart的图表配置中数据列可使用的规范形式，再以此创建图表。

系统存在的不足：访问者需要手动点击按钮来进行刷新，理想化的该系统应该是自动刷新图表。

课程收获：跟着聂老师又学习了一门必须掌握的课程，学习到了关于java web应用开发的诸多知识，也学习到了在以后面试中需要注意的一些内容，以一些及IT企业的故事。

课程建议：老师依旧像上学年教授java应用开发时一样的认真负责，学生尚且想不到什么改进建议，便不弄虚作假。

发展方向：软件工程师

自评成绩：100