国际物流云商系统第十一天

一.回顾

出口报运的功能补充(批量数据更新)

购销合同打印的读程（14 个问题 ）

读程的两种方法？

为什么要查询购销合同下货物对象列表时，要考虑根据厂家名称进行排序，这样做可以减少回退次数，算法效率得到提高

任务调度 Quartz
Quartz+spring 整合实现定时任务调度

配置

1. 自定义的任务类
2. 定义任务对象(MethodInvokingJobDetailFacoryBean)
   注入： targetObject targetMethod
3. 定义触发器
   类：CronTriggerFactoryBean
   注入：jobDetail

4. 定义总调度器
   类： SchedulerFactoryBean
   注入： triggers (list 集合)

二.早期的图形报表实现

1.使用 Excel 实现图形报表

1.从数据库中统计出结果
2.打开 Excel,将数据复制进去，再插入图表就可以了

会龙	25901.4
光华	276643
升华	300
华艺	40112
南皮开发	48400
天顺	89864.8
宏艺	244497.6
平遥远江	121
平遥鸿艺	19318.4
文水志远	89136
晶晨	1400
民鑫	89007
汇越	6960
精艺	102216

2.使用 JFreeChart 实现图表制作

它是使用 java 开源方式写出来的一套专门制作图形的插件。
早期用的非常普遍。jFreeChart 的文档和服务是收费的。

1.导入 jar 包

Maven工程的坐标:

```
<!-- https://mvnrepository.com/artifact/jfree/jcommon -->
<dependency>
    <groupId>jfree</groupId>
    <artifactId>jcommon</artifactId>
    <version>1.0.16</version>
</dependency>
```

```
mvn install:install-file -DgroupId=jfree -DartifactId=jcommon -Dversion=1.0.16 -Dpackaging=jar -Dfile=jcommon-1.0.16.jar
```

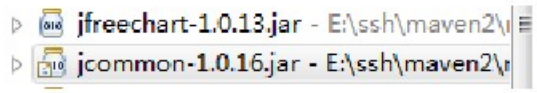
```
<dependency>
        <groupId>jfree</groupId>
        <artifactId>jfreechart</artifactId>
        <version>1.0.13</version>
</dependency>
```

```
mvn install:install-file -DgroupId=jfree -DartifactId=jfreechart -Dversion=1.0.13 -Dpackaging=jar -Dfile=jfreechart-1.0.13.jar
```

资料位置：国际物流云商系统项目-day11\资料\jfreechart
Pom.xml文件中引入;

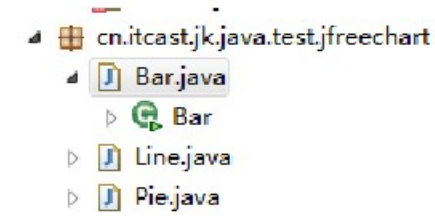
```
<!-- jfreechart -->
<dependency>
    <groupId>jfree</groupId>
    <artifactId>jcommon</artifactId>
    <version>1.0.16</version>
</dependency>

<dependency>
    <groupId>jfree</groupId>
    <artifactId>jfreechart</artifactId>
    <version>1.0.13</version>
</dependency>
<!-- end -->
```

Web 工程直接将这两个 jar 包放入 WEB-INF/lib 文件夹下

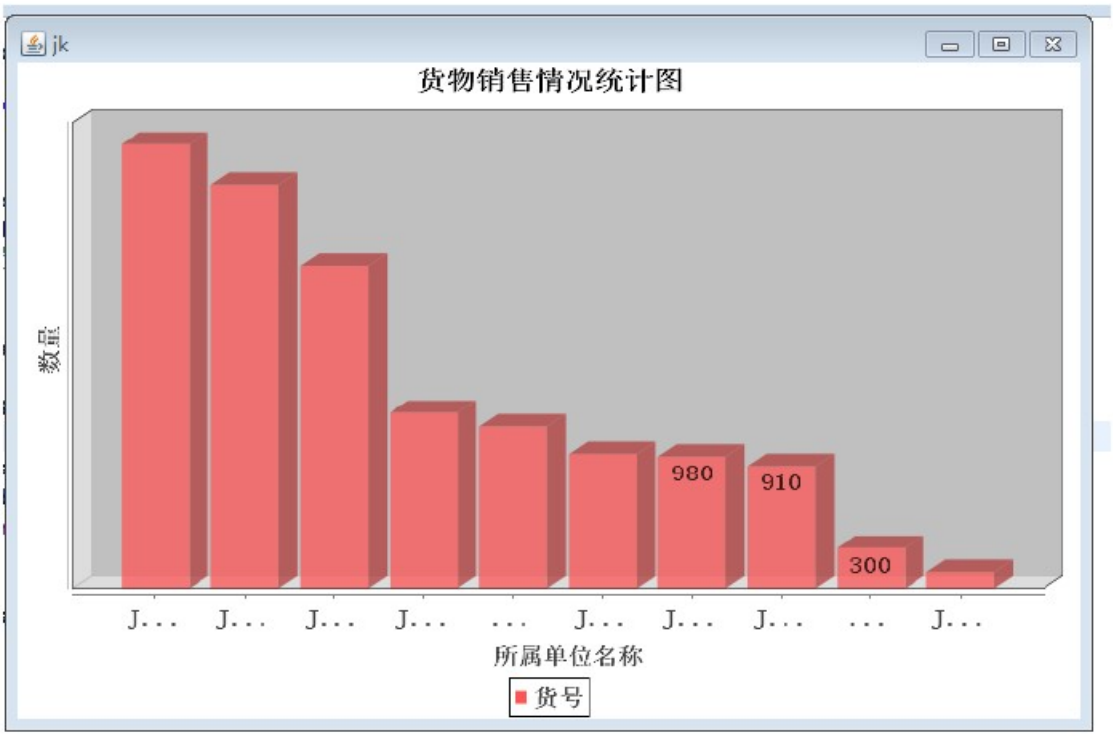


2.编写代码

参考



3.实现效果



3.第三方图形报表插件

功能强大 **bird**，润乾，数巨报表，水晶报表**(.net)**（收费）

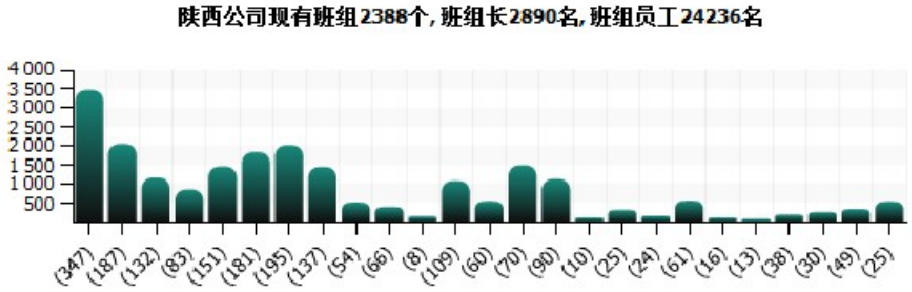
4.JS 占市场比例最大

AmCharts ,HighChart， eCharts

三.AmCharts 实现图形报表

1.AmCharts 老版本(了解)

amChart 实现图表制作，早期使用的是 XML 做为数据来源，新版本中进行调整，使用 JSON 数据作为数据来源。
效果图：



2.数据来源技术分析

从数据来源可以看出，将来要使用 amChart 制作图表时，主要就是把 data.xml 中的数据替换为从数据库中查询出来的数据就可以了，就会涉及到操作 XML 文档，常用的 XML 操作方式有哪些？（JDOM,DOM4J,SAX）,用得最多的是 DOM4J 实现 XML 文档操作。但是本次项目不使用这些技术，我只是把 XML 文档也当成是一个普通文件，所以就可以直接使用流的方式，将一个组织好的符合 XML 结构的字符串直接写入到 XML 文件中，这样就可以不用再去频繁操作 XML 文件的结点了。

四． DAO 的实现方式

统计查询时更适合使用 SQL 语句，所以该项目中就要支持 SQL 查询。
添加相关的操作 sql 语句的工具类 SqlDAO

1.项目的持久化方案介绍

本次项目中实现数据访问的 DAO 是两条腿走路，一条腿是 Spring Data Jpa,另一条腿 jdbc 操作 sql 语句。就要避免两腿打架。

Spring Data Jpa 的 Hibernate 实现进行 CUD 操作,同时会操作一级缓存。而在同时 jdbc 也对这条记录进行 CUD 操作，此时就会出现问题，因为 jdbc 是直接操作数据库的，不会及时更新 Hibernate 一级缓存，由此导致数据不一致。

Jdbc 使用 sql 语句只做查询！

问题的解决：jdbc 操作 sql 语句的机会只是查询。

因为操作 sql 语句时，使用的是 jdbcTemplate 技术，所以就需要配置它，操作 sql 语句的工具类还要注入 jdbcTemplate.

1.配置 jdbcTmplate(注入 dataSource)

2.配置 SqlDao 工具类（注入 jdbcTemplate）

加入最新配置 applicationContext.xml

```
<!--加入另一腿的配置-->
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="dataSource" ></property>
</bean>
<bean id="sqlDao" class="cn.itcast.common.springdao.SqlDao">
    <property name="jdbcTemplate" ref="jdbcTemplate" ></property>
</bean>
```

applicationContext.xml文件中加入的内容

2.SqlDao 工具的源码分析

```
@Repository
public class SqlDao {
    private static Logger log = Logger.getLogger(SqlDao.class);
    private UtilFuns utilFuns = new UtilFuns();

    @Autowired
    private JdbcTemplate jdbcTemplate;
```

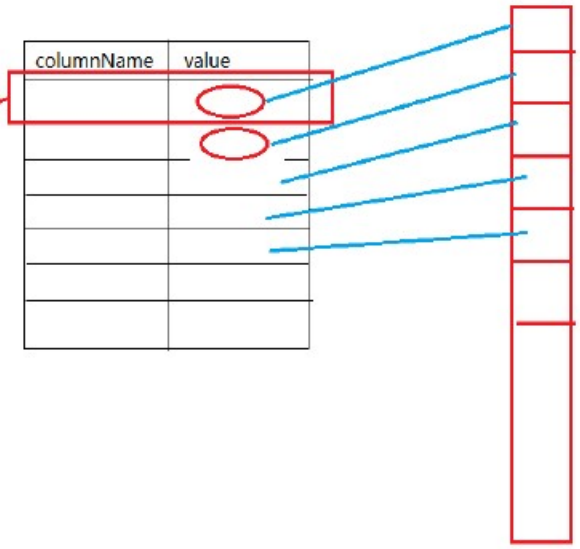
注入JdbcTemplate模板技术

重要方法分析

executeSQL 用于执行 sql 语句，并且能够将执行的结果由二维结构，转化为符合要求的一维结构，为后面 Action 实现结果处理提供了很多便利。

它的执行过程，主要是先将 sql 查询结果放入一个 Map 集合中，map 中存放的是每个列的列名及它的值，这要我们的 executeSQL 方法就可以根据特征来实现根据 key 得到每个 value。

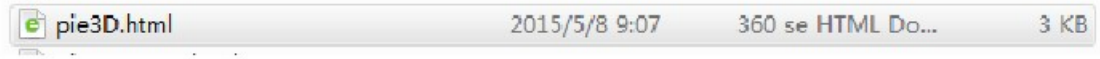
```
public List executeSQL(String sql){
    log.debug(sql);
    List<String> aList = new ArrayList();
    List jlist = jdbcTemplate.queryForList(sql);
    Iterator ite = jlist.iterator();
    while(ite.hasNext()){
        Map map = (Map)ite.next();
        for(Object o:map.keySet()){
            if(map.get(o.toString())==null){
                aList.add("");       //对象不存在时，写空串
            }else{
                aList.add(map.get(o.toString()).toString());
            }
        }
    }
    return aList;
}
```



五．生产厂家销售情况的统计

生产厂家销售统计主要是按货物的销量统计出各个生产厂家的销售额，再以图形报表的方式实现数据的显示，我们的原型主要是到 AmCharts 中找到对应的图形来显示，本次主要以饼图来实现数据显示。

从 AmCharts 提供的 demo 示例中，找到饼图



1.sql 语句的分析与构建

因为是各个不同的生产厂家，所以要进行分组统计，sql 语句如下：
select factory_name ,sum(amount) from contract_product_c group by factory_name

2.业务逻辑 Service 的实现

```
@Service
public class StatServiceImpl implements StatChartService {
    @Autowired
    private SqlDao sqlDao;

    public List genFactoryData() {
        String sql = "select factory_name ,sum(amount) from contract_product_c group by factory_name";
        return sqlDao.executeSQL(sql);
    }
```

3.编写 Action

运行界面，看菜单是否完整，数据库中可以添加内容

0	(NULL)	生产厂家销售情况	stat/statChartAction_factorysale
0	(NULL)	产品销售排行	stat/statChartAction_productsale
0	(NULL)	系统访问压力图	stat/statChartAction_onlineinfo

添加处理的 Action，并注入业务逻辑 service

```
@Namespace("/stat")
public class StatChartAction extends BaseAction {
    @Autowired
    private StatChartService statChartService;
```

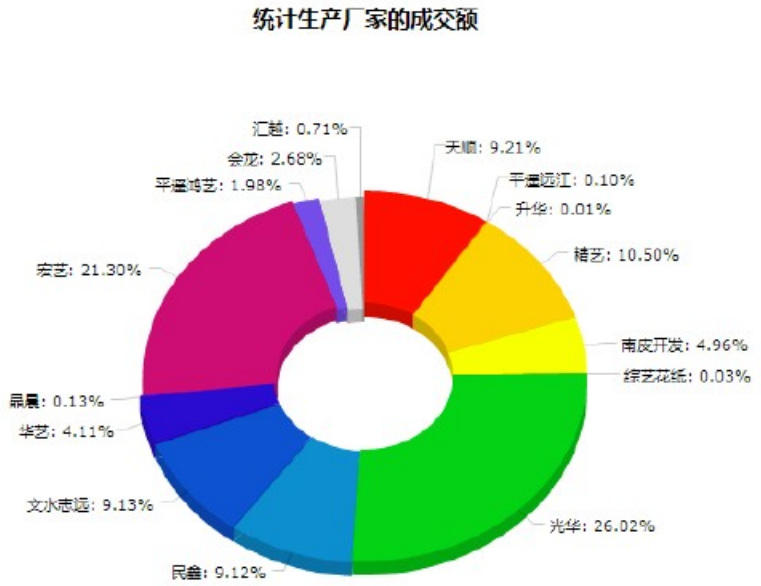
在 Action 中添加处理生产厂家销售额的方法，并使用 FastJSON 实现 JSON 数据的处理。

```
@Action(value = "statChartAction_factorysale", results = {
        @Result(name = "factorysale", location = "/WEB-INF/pages/stat/chart/factorySalePie.jsp") })
public String factorysale() throws Exception {
    // 1.使用sql得到统计数据
    List<String> factoryList = statChartService.genFactoryData();
    // 2.将工厂列表的数据组织成json串
    /**
     * [ { "country": "United States", "visits": 9252 }]
     */
    List<Map<String, Object>> factoryJsonList = new ArrayList<Map<String, Object>>();
    for (int i = 0; i < factoryList.size(); i++) {
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("factoryName", factoryList.get(i));
        i++;
        map.put("sales", factoryList.get(i));

        factoryJsonList.add(map);
    }
    // 3.使用FastJson生成串
    String jsonStr = JSON.toJSONString(factoryJsonList);

    super.put("pieChartData", jsonStr);
    return "factorysale";
}
```

4.显示饼图的 JSP 页面的处理

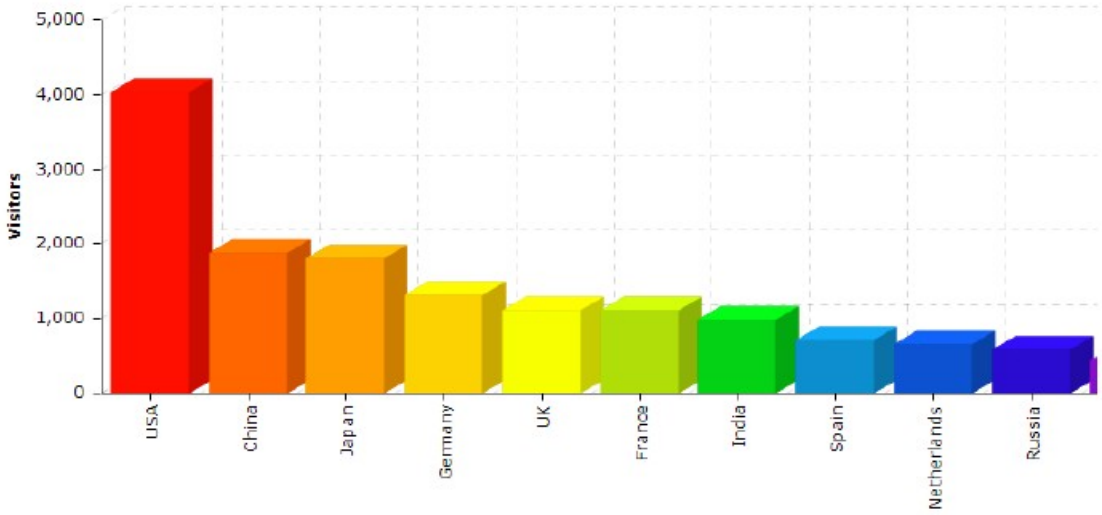
```
AmCharts.ready(function () {
    // PIE CHART
    chart = new AmCharts.AmPieChart();

    // title of the chart
    chart.addTitle("统计生产厂家的成交额", 16);

    chart.dataProvider = chartData;//====================
    chart.titleField = "factoryName";//================================
    chart.valueField = "sales";//==================================
    chart.sequencedAnimation = true;
    chart.startEffect = "elastic";
    chart.innerRadius = "40%";
    chart.startDuration = 5;
    chart.labelRadius = 15;
    chart.balloonText = "[[title]]<br><span style='font-size:14px'><b>[[value]]</b> ([[percents]]%)</span>";
    // the following two lines makes the chart 3D
    chart.depth3D = 10;
    chart.angle = 15;

    chart.creditsPosition = "top-right";

    // WRITE
    chart.write("chartdiv");
});
```

5.效果图展示



六.产品销售排行（统计前 15 名）

产品销售量要求按前 15 名来显示，并且以柱状图来展示结果。
为了显示的效果，可以到 AmCharts 中找到合适的图表进行改造。



1.查询的 SQL 语句构建

```
select * from (select product_no,sum(amount) from contract_product_c group by product_no
order by sum(amount) desc) where rownum<=15
```

2.实现查询的 Service

```
/**
 * 产品销里的前15名
 * select * from (select product_no,sum(amount) from contract_product_c group by product_
   where rownum<=15;
 */
public List genProductData() {
    String sql = "select * from (select product_no,sum(amount) from contract_product_c "
            + "group by product_no order by sum(amount) desc) where rownum<=15";
    return sqlDao.executeSQL(sql);
}
```

3.编写产品销售的 Action

```
/**
 * 产品列表
 */
@Action(value = "statChartAction_productsale", results = {
        @Result(name = "productsale", location = "/WEB-INF/pages/stat/chart/productSaleColumn.jsp") })
public String productsale() throws Exception {
    return "productsale";
}
```

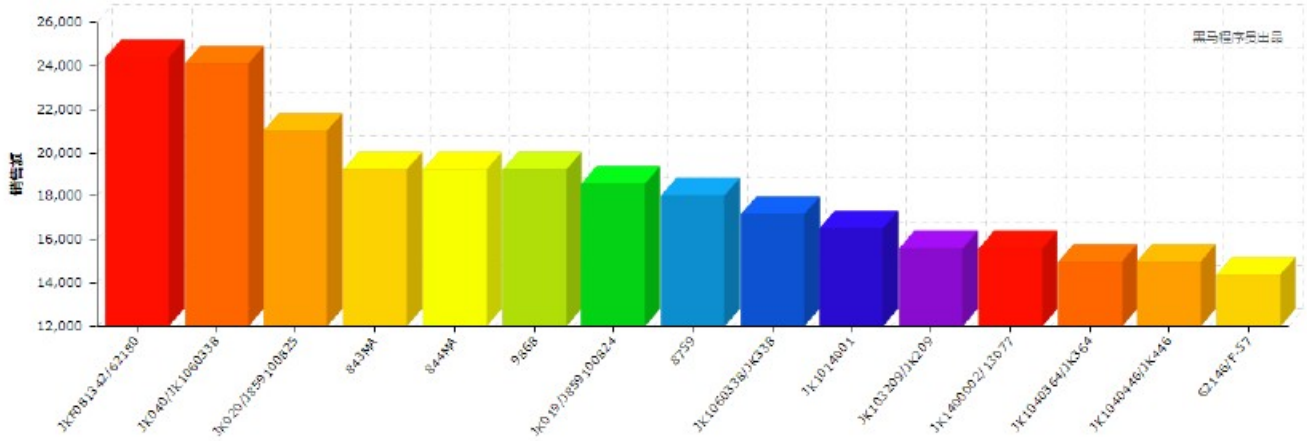
为了处理 Ajax 方式实现图形报表，在服务端编写产品销量的方法

```
@Action("statChartAction_genProductSale")
public String genProductSale() throws Exception {
    List<String> productList = statChartService.genProductData();
    String colors[] = { "#FF0F00", "#FF6600", "#FF9E01", "#FCD202", "#F8FF01", "#B0DE09", "#04D215", "#0D8ECF",
            "#0D52D1", "#2A0CD0", "#8A0CCF" };
    int j = 0;
    List<Map<String, Object>> productJsonList = new ArrayList<Map<String, Object>>();
    for (int i = 0; i < productList.size(); i++) {
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("productNo", productList.get(i));
        i++;
        map.put("saleAmount", productList.get(i));
        map.put("color", colors[j++]);
        if (j % 11 == 0) {
            j = 0;
        }
        productJsonList.add(map);
    }
    // 3.生成json串
    String productJson = JSON.toJSONString(productJsonList);

    // 4.使用HttpServletResponse输出
    HttpServletResponse response = ServletActionContext.getResponse();
    response.setContentType("application/json;charset=UTF-8");
    response.setHeader("Cache-Control", "no-cache");
    response.getWriter().write(productJson);
    return NONE;
}
```

4.JSP 页面发送 Ajax 请求，并显示柱状图

```
AmCharts.ready(function () {
    $.ajax(
          {
              url:"${pageContext.request.contextPath}/stat/statChartAction_genProductSale",
              type:"POST",
              dataType:"json",
              success:function(chartData){
                // SERIAL CHART
                    chart = new AmCharts.AmSerialChart();
                    chart.dataProvider = chartData;
                    chart.categoryField = "productNo";//====================================
                    // the following two lines makes chart 3D
                    chart.depth3D = 20;
                    chart.angle = 50;

                    // AXES
                    // category
                     var categoryAxis = chart.categoryAxis;
                    categoryAxis.labelRotation = 50;//============================
                    categoryAxis.dashLength = 5;
                    categoryAxis.gridPosition = "start";
```

5.效果图



七．在线人数统计的折线图

1.在线登录日志的收集

准备数据

online_info_t 表　　指的是一天的 **24** 小时

login_log_p 表　访问日志，当用户登录时就应当向这个表中添加一条记录

LOGIN_LOG_ID	LOGIN_NAME	IP_ADDRESS	LOGIN_TIME
2191758C-872C-4B0B-9FBF-85A6F10126FD	anonymous|	127.0.0.1	2012-11-27 10:55:03
21DE607B-0E68-4EE1-9C65-8D55526B1382	liuxiaoshan|刘晓山|刘晓山	192.168.1.107	2012-05-02 17:19:08
21F1656C-9B40-4B27-9A9D-460B3668D192	anonymous|	192.168.1.107	2012-11-08 09:12:08
220336B2-7FD2-458D-8426-5CA88D8868E2	001|调试	127.0.0.1	2012-11-27 11:24:25
228A0984-5BE4-4FFF-B5C4-A09ECB8A46F2	001|调试	192.168.1.100	2012-07-11 11:04:11
22DA6C12-23B0-4A34-928A-11732D9BD218	001|调试	192.168.1.106	2012-10-29 14:42:08
23661154-F280-43FC-BA18-C445FFBCF40D	001|调试	127.0.0.1	2013-03-09 23:30:16
237D9844-5C50-4843-AEDD-98DE912C08B5	001|调试	192.168.1.115	2012-01-31 13:54:07
2424F2EA-0768-45FF-B9B5-9446B6364EC8	001|调试	127.0.0.1	2013-03-09 23:30:31
24652510-1930-45D1-BAA5-5D4D2144F8D2	001|调试	192.168.1.115	2012-02-03 11:59:03
24894E43-794A-41A3-A012-8F0406BEAF94	anonymous|	192.168.1.250	2012-07-11 15:17:12
2492ABB0-20F4-400F-908F-1EA295674998	001|调试	192.168.1.115	2012-01-31 12:01:31
24B15961-1603-4152-A68B-F99301FE52E3	anonymous|席军	192.168.1.103	2011-12-05 10:06:14

数据来源，可以再次修改登录操作

获取相关数据，直接在 **login_log_p** 表中添加数据

ServletActionContext.getRequest().getRemoteAddr();//获取客户端 **ip** 地址

2.准备统计的 sql 语句

#14 点（范围）有多少人在线
select count(*) from login_log_p where to_char(login_time,'HH24')='14'
select substring(login_time,12,2) from login_log_p;

#要得到每个时间片段(1 小时就是一个片段)有多少人在线

```
select a.a1,b.c from
   (select * from online_info_t) a
left join
  (select to_char(login_time,'HH24') a1, count(*) c from
   login_log_p group by to_char(login_time,'HH24')) b
on (a.a1 = b.a1)
order by a.a1
```

3.编写 Service 方法

```
public List genOnlineInfo() {
    String sql = "select a.a1,b.c from "+
     "(select * from online_info_t) a "+
    "left join "+
             " (select to_char(login_time,'HH24') a1, count(*) c from login_log_p group by to_char(login_time,'HH24'))"+
             " b"+
    " on (a.a1 = b.a1)"+
    " order by a.a1";
    return sqlDao.executeSQL(sql);
}
```

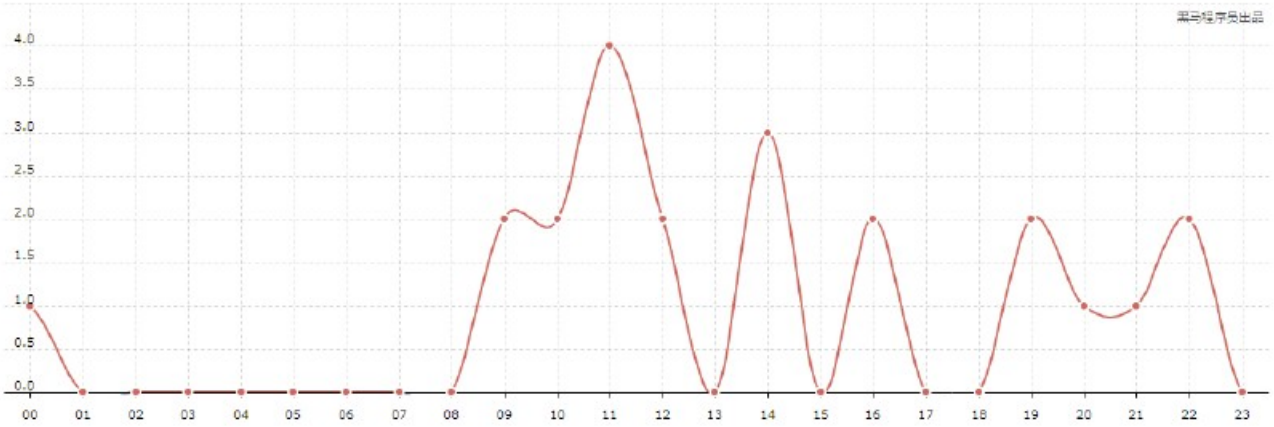
4.编写及配置 Action

```
@Action(value = "statChartAction_onlineinfo", results = {
        @Result(name = "onlineinfo", location = "/WEB-INF/pages/stat/chart/onlineinfolineSmooth.jsp") })
public String onlineinfo() throws Exception {
    // 1.加载数据
    List<String> onlineList = statChartService.genOnlineInfo();
    // 2.组织数据
    List<Map<String, Object>> onlineJsonList = new ArrayList<Map<String, Object>>();
    for (int i = 0; i < onlineList.size(); i++) {
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("hour", onlineList.get(i));
        i++;
        map.put("value", onlineList.get(i));
        onlineJsonList.add(map);
    }
    // 3.生成json串
    String onlineJson = JSON.toJSONString(onlineJsonList);

    //4.保存到值栈中
    super.put("onlineJson", onlineJson);

    return "onlineinfo";
}
```

5.JSP 页面编写

```
AmCharts.ready(function () {
    // SERIAL CHART
    chart = new AmCharts.AmSerialChart();

    chart.dataProvider = chartData;
    chart.marginLeft = 10;
    chart.categoryField = "hour";
```

6.实现效果图



八．eCharts 的介绍

1.eCharts 介绍

获取 ECharts

你可以通过以下几种方式获取 ECharts。

1. 从官网下载界面选择你需要的版本下载，根据开发者功能和体积上的需求，我们提供了不同打包的下载，如果你在体积上没有要求，可以直接下载完整版本。开发环境建议下载源代码版本，包含了常见的错误提示和警告。
2. 在 ECharts 的 GitHub 上下载最新的 `release` 版本，解压出来的文件夹里的 `dist` 目录里可以找到最新版本的 echarts 库。
3. 通过 npm 获取 echarts， `npm install echarts --save` ，详见"在 webpack 中使用 echarts"
4. cdn 引入，你可以在 cdnjs，npmcdn 或者国内的 bootcdn 上找到 ECharts 的最新版本。

系统首页

2.eCharts 的引入

引入 ECharts

ECharts 3 开始不再强制使用 AMD 的方式按需引入，代码里也不再内置 AMD 加载器。因此引入方式简单了很多，只需要像普通的 JavaScript 库一样用 script 标签引入。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <!-- 引入 ECharts 文件 -->
    <script src="echarts.min.js"></script>
</head>
</html>
```

绘制一个简单的图表

在绘图前我们需要为 ECharts 准备一个具备高宽的 DOM 容器。

```
<body>
    <!-- 为 ECharts 准备一个具备大小（宽高）的 DOM -->
    <div id="main" style="width: 600px;height:400px;"></div>
</body>
```

九. **eCharts** 的使用

然后就可以通过 echarts.init 方法初始化一个 echarts 实例并通过 setOption 方法生成一个简单的柱状图，下面是完整代码。

```
<!DOCTYPE html>
```

```
<html>
<head>
    <meta charset="utf-8">
    <title>ECharts</title>
    <!-- 引入 echarts.js -->
    <script src="echarts.min.js"></script>
</head>
<body>
    <!-- 为ECharts准备一个具备大小（宽高）的Dom -->
    <div id="main" style="width: 600px;height:400px;"></div>
    <script type="text/javascript">
        // 基于准备好的dom，初始化echarts实例
        var myChart = echarts.init(document.getElementById('main'));

        // 指定图表的配置项和数据
        var option = {
            title: {
                text: 'ECharts 入门示例'
            },
            tooltip: {},
            legend: {
                data:['销量']
            },
            xAxis: {
              data: ["衬衫","羊毛衫","雪纺衫","裤子","高跟鞋","袜子"]
            },
            yAxis: {},
            series: [{
                name: '销量',
                type: 'bar',
                data: [5, 20, 36, 10, 10, 20]
            }]
        };

        // 使用刚指定的配置项和数据显示图表。
        myChart.setOption(option);
    </script>
</body>
</html>
```

3.作业

使用 **eCharts** 实现图形报表