

- \* 学习目标
  - \* 能够掌握SSM无配置的搭建
  - \* 能够掌握SSM综合之AOP日志
  - \* 能够理解Echarts的概述
  - \* 能够使用Echarts的饼图展现出员工男女比例
  - \* 能够使用Echarts的柱状图展现出员工男女比例
- 

- \* 能够掌握SSM无配置的搭建
- \* 复习SSM集成：[04-springmvc03](#)

```
1 * SSM无配置的搭建
2 * 核心配置类
3 * SpringConfiguration
4 @Configuration
5 @ComponentScan(value = "com.lg",
6     excludeFilters = {
7         @ComponentScan.Filter(type = FilterType.ANNOTATION,
8             classes = Controller.class),
9         @ComponentScan.Filter(type = FilterType.ASSIGNABLE_TYPE,
10             classes = SpringMVCConfig.class)})
11 @EnableAspectJAutoProxy
12 @Import(SMConfiguration.class)
13 public class SpringConfiguration {
14 }
15
16 * SMConfiguration
17 @Configuration
18 @EnableTransactionManagement
19 @MapperScan("com.lg.dao")
20 @PropertySource("classpath:db.properties")
21 public class SMConfiguration {
22     @Value("${lg.driver}")
```

```

23     private String driver;
24     @Value("${lg.url}")
25     private String url;
26     @Value("${lg.username}")
27     private String username;
28     @Value("${lg.password}")
29     private String password;
30     @Bean
31     @Scope(ConfigurableBeanFactory.SCOPE_SINGLETON)
32     public DataSource dataSource(){
33         DruidDataSource ds=new DruidDataSource();
34         ds.setDriverClassName(driver);
35         ds.setUrl(url);
36         ds.setUsername(username);
37         ds.setPassword(password);
38         return ds;
39     }
40     @Bean
41     public SqlSessionFactoryBean sqlSessionFactoryBean(DataSource dataSource){
42         SqlSessionFactoryBean sqlSessionFactoryBean=new SqlSessionFactoryBean()
43         sqlSessionFactoryBean.setDataSource(dataSource);
44         PathMatchingResourcePatternResolver resolver=new PathMatchingResourcePa
45         try {
46             Resource[] resources = resolver.getResources("classpath:com/lg/dao/
47             sqlSessionFactoryBean.setMapperLocations(resources);
48         } catch (IOException e) {
49             e.printStackTrace();
50         }
51         sqlSessionFactoryBean.setTypeAliasesPackage("com.lg.bean");
52         return sqlSessionFactoryBean;
53     }
54     @Bean
55     public TransactionManager transactionManager(DataSource dataSource){
56         DataSourceTransactionManager transactionManager=new DataSourceTransacti
57         transactionManager.setDataSource(dataSource);
58         return transactionManager;
59     }
60 }
61
62 * SpringMVCConfig

```

```

63 @Configuration
64 @ComponentScan(value = "com.lg", useDefaultFilters = false, includeFilters =
65     {@ComponentScan.Filter(type = FilterType.ANNOTATION, classes = Controller
66 @EnableAspectJAutoProxy
67 @EnableWebMvc
68 public class SpringMVCConfig implements WebMvcConfigurer{
69     /**
70     * 添加视图解析器
71     */
72     @Bean
73     public InternalResourceViewResolver internalResourceViewResolver() {
74         InternalResourceViewResolver internalResourceViewResolver =
75             new InternalResourceViewResolver();
76         internalResourceViewResolver.setPrefix("/WEB-INF/jsp/");
77         internalResourceViewResolver.setSuffix(".jsp");
78         return internalResourceViewResolver;
79     }
80
81     @Override
82     public void addResourceHandlers(ResourceHandlerRegistry registry) {
83         registry.addResourceHandler("/css/**").addResourceLocations("/css/");
84         registry.addResourceHandler("/image/**").addResourceLocations("/image/");
85         registry.addResourceHandler("/js/**").addResourceLocations("/js/");
86     }
87 }
88 * WebConfig
89
90 public class WebConfig
91     extends AbstractAnnotationConfigDispatcherServletInitializer {
92     /**
93     * @return
94     * spring 根容器
95     */
96     @Override
97     protected Class<?>[] getRootConfigClasses() {
98         return new Class[]{SpringConfiguration.class};
99     }
100
101     /**
102     * @return

```

```
103     * Spring mvc容器
104     */
105     @Override
106     protected Class<?>[] getServletConfigClasses() {
107         return new Class[]{SpringMVCCConfig.class};
108     }
109
110     //DispatcherServlet映射,从"/"开始
111     @Override
112     protected String[] getServletMappings() {
113         return new String[]{"/*"};
114     }
115 }
116 * 控制器
117 @Controller
118 @RequestMapping("/emp")
119 public class TransferEmployeeController {
120     @Autowired
121     private TransferEmployeeService transferEmployeeService;
122     @RequestMapping("/transfer")
123     public String transfer(String empno,String job,Integer odid,Integer ndid){
124         Map<String,Object> params=new HashMap<>();
125         params.put("empno",empno);
126         params.put("job",job);
127         params.put("odid",odid);
128         params.put("ndid",ndid);
129         transferEmployeeService.transferEmployee(params);
130         return "success";
131     }
132 }
133
134 * 单元测试
135 @Test
136 public void test17() throws Exception {
137     String result = mockMvc.perform(delete("/emp/transfer")
138         .param("empno","LG008")
139         .param("job","前端讲师")
140         .param("odid","2")
141         .param("ndid","1")).
142     andExpect(status().isOk()).
```

```

143         andDo(print()).andReturn().getResponse().getContentAsString();
144         System.out.println(result);
145     }

```

\* 能够掌握SSM综合之AOP日志

\* 复习AOP:[03-spring03](#)

\* 创建表syslog

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
* id	int	11		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	主键, 自动增长无意义
visitTime	timestamp		CURRENT_TIMESTAMP	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	访问时间
username	varchar	50		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	操作者用户名
ip	varchar	50		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	访问ip
url	varchar	50		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	访问资源url
executionTime	int	11		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	执行时长
method	varchar	100		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	访问方法
				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

```

1  * 案例:
2  * 表syslog
3  CREATE TABLE syslog (
4      id int(11) NOT NULL AUTO_INCREMENT COMMENT '主键,自动增长无意义',
5      visitTime timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
6      ON UPDATE CURRENT_TIMESTAMP COMMENT '访问时间',
7      username varchar(50) DEFAULT NULL COMMENT '操作者用户名',
8      ip varchar(50) DEFAULT NULL COMMENT '访问ip',
9      url varchar(50) DEFAULT NULL COMMENT '访问资源url',
10     executionTime int(11) DEFAULT NULL COMMENT '执行时长',
11     method varchar(100) DEFAULT NULL COMMENT '访问方法',
12     PRIMARY KEY (id)
13 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
14 * 实体类
15 @Data
16 @AllArgsConstructor
17 @NoArgsConstructor
18 public class SysLog {
19     private String id;
20     private Date visitTime;
21     private String visitTimeStr;
22     private String username;

```

```
23     private String ip;
24     private String url;
25     private Long executionTime;
26     private String method;
27 }
28 * dao
29 public interface SysLogDao {
30     @Select("select * from syslog")
31     @Results({ @Result(id=true,column="id",property="id"),
32                @Result(column="visitTime",property="visitTime"),
33                @Result(column="ip",property="ip"),
34                @Result(column="url",property="url"),
35                @Result(column="executionTime",property="executionTime"),
36                @Result(column="method",property="method"),
37                @Result(column="username",property="username") })
38     List<SysLog> findAll() throws Exception;
39
40     @Insert("insert into syslog(visitTime,username,ip,url,executionTime,method)
41            values(#{visitTime},#{username},#{ip},#{url},#{executionTime},#{met
42            void save(SysLog log) throws Exception;
43 }
44 * service
45 public interface SysLogService {
46     List<SysLog> findAll() throws Exception;
47     void save(SysLog log) throws Exception;
48 }
49 @Service
50 public class SysLogServiceImpl implements SysLogService {
51     @Autowired
52     private SysLogDao sysLogDao;
53     @Override
54     public List<SysLog> findAll() throws Exception {
55         return sysLogDao.findAll();
56     }
57     @Override
58     public void save(SysLog log) throws Exception {
59         sysLogDao.save(log);
60     }
61 }
62
```

```
63 * 忽略LogAop注解
64 @Retention(RetentionPolicy.RUNTIME)
65 @Target(ElementType.METHOD)
66 public @interface IgnoreLog {
67 }
68
69 * Controller
70 @RestController
71 @RequestMapping("/syslog")
72 public class SysLogController {
73     @Autowired
74     private SysLogService sysLogService;
75     @RequestMapping("/findLogs")
76     public SysLogVo findSysLog(){
77         List<SysLog> sysLogs = null;
78         SysLogVo sysLogVo=new SysLogVo();
79         try {
80             sysLogs = sysLogService.findAll();
81             sysLogVo.setCode("200");
82             sysLogVo.setMsg(sysLogs);
83         } catch (Exception e) {
84             sysLogVo.setCode("500");
85             sysLogVo.setMsg("服务器错误");
86         }
87         return sysLogVo;
88     }
89 }
90
91 * 切面
92 @Controller
93 @Aspect
94 public class LogAop {
95     @Autowired
96     private HttpServletRequest request;
97     @Autowired
98     private SysLogService sysLogService;
99
100     private Long startTime;// 访问时间
101     @Pointcut("execution(* com.lg.controller.*.*(..))")
102     public void pointcut(){
```

```

103     }
104     @Around("pointcut()")
105     public Object recordSysLog(ProceedingJoinPoint joinPoint) throws Throwable
106     {
107         startTime=System.currentTimeMillis();
108         // 执行业务方法
109         Object result = joinPoint.proceed();
110         try{
111             handle(joinPoint);
112         }catch (Exception e){
113             e.printStackTrace();
114         }
115         return result;
116     }
117
118     private void handle(ProceedingJoinPoint jp) throws Exception {
119         // 获取访问的类
120         Class executionClass=jp.getTarget().getClass();
121         // 获取访问的方法的名称
122         MethodSignature msig = (MethodSignature) jp.getSignature();
123         Method executionMethod=executionClass.getMethod(msig.getName(),msig.get
124         IgnoreLog ignoreLogAnnotation = executionMethod.getAnnotation(IgnoreLog
125         if(ignoreLogAnnotation!=null){
126             return;
127         }
128         //获取类上的@RequestMapping对象
129         Annotation classAnnotation = executionClass.getAnnotation(RequestMappi
130         RequestMapping methodAnnotation = executionMethod.getAnnotation(Request
131         if(classAnnotation==null && methodAnnotation==null){
132             return;
133         }
134         String url = null;
135         if(classAnnotation!=null){
136             RequestMapping classAnnotation1=(RequestMapping)classAnnotation;
137             if(methodAnnotation!=null){
138                 // 它的值应该是类上的@RequestMapping的value+方法上的 @RequestMapp
139                 url = classAnnotation1.value()[0] + methodAnnotation.value()[0]
140             }
141         }else{
142             if(methodAnnotation!=null){
143                 // 方法上的 @RequestMapping的value

```



```

143         url = methodAnnotation.value()[0];
144     }
145 }
146 SysLog sysLog = new SysLog();
147 // 获取访问时长
148 Long executionTime = System.currentTimeMillis() - startTime;
149 // 将sysLog对象属性封装
150 sysLog.setExecutionTime(executionTime);
151 sysLog.setUrl(url);
152 // 获取ip
153 String ip = request.getRemoteAddr();
154 sysLog.setIp(ip);
155 // 用户名从session中获取
156 Object username=request.getSession().getAttribute("username");
157 if(username!=null){
158     sysLog.setUsername((String) username);
159 }
160 sysLog.setMethod("[类名]" + executionClass.getName() + "[方法名]" + executionTime);
161 sysLog.setVisitTime(new Date(startTime));
162 // 调用Service, 调用dao将sysLog insert数据库
163 sysLogService.save(sysLog);
164 }
165
166 }
167
168
169 * 测试
170 @RequestMapping("/test15")
171 public String test15(){
172     System.out.println("hello test15");
173     return "success";
174 }
175 * http://localhost:8080/ssm02/test/test15
176 * 界面
177 <%@ page contentType="text/html; charset=UTF-8" language="java" isELIgnored="false"%>
178 <html>
179 <head>
180     <meta charset="UTF-8">
181     <title>日志列表</title>
182     <link rel="stylesheet" href="${pageContext.request.contextPath}/css/bootstrap.css">

```

```
183 <script src="${pageContext.request.contextPath}/js/jquery-3.3.1.js"></script>
184 <script src="${pageContext.request.contextPath}/js/popper.min.js"></script>
185 <script src="${pageContext.request.contextPath}/js/bootstrap.js"></script>
186 <script>
187     var url="/ssm02/syslog/findLogs";
188     $.ajax({
189         type:"POST",
190         url:url,
191         contentType:"application/json;charset=utf-8",
192         success:function(result,status){
193             var content = JSON.parse(JSON.stringify(result));
194             if(content.code==200){
195                 var tbody=document.getElementsByTagName("tbody")[0];
196                 tbody.innerHTML="";
197                 for(var index in content.msg){
198                     var syslog=content.msg[index];
199                     tbody.innerHTML+="|\n" +
200                         "            <td>"+syslog.visitTime+"</td>\n" +
201                         "            <td>"+syslog.ip+"</td>\n" +
202                         "            <td>"+syslog.url+"</td>\n" +
203                         "            <td>"+syslog.executionTime+"</td>\n" +
204                         "            <td>"+syslog.method+"</td>\n" +
205                         "            </tr>";
206                 }
207             }
208         },
209         error:function(result){
210             alert("请求数据失败")
211         }
212     });
213
214 </script>
215 </head>
216 <body>
217 <div class="container-fluid">
218     <h1>亮哥教育</h1>
219     <table class="table table-bordered table-striped">
220         <thead>
221         <tr>
222             <th>访问时间</th>
|  |

```

```

223         <th>访问ip</th>
224         <th>访问资源</th>
225         <th>执行时间</th>
226         <th>访问方法</th>
227     </tr>
228 </thead>
229 <tbody>
230 <tr>
231     <td>Jan 6, 2020 8:57:34 PM</td>
232     <td>0:0:0:0:0:0:1</td>
233     <td>/test/test15</td>
234     <td>6</td>
235     <td>[类名]com.lg.controller.TestController[方法名]test15</td>
236 </tr>
237 </tbody>
238 </table>
239 </div>
240 </body>
241 </html>

```

POST http://localhost:8080/ssm02/syslog/findLogs Params Send

New key	Value	Description
odid	2	
ndid	1	

Body Cookies (4) Headers (4) Test Results Status: 200 OK Time: 272 ms

Pretty Raw Preview JSON

```

1 {
2   "code": "200",
3   "msg": [
4     {
5       "id": "1",
6       "visitTime": "Jan 6, 2020 8:57:34 PM",
7       "ip": "0:0:0:0:0:0:1",
8       "url": "/test/test15",
9       "executionTime": 6,
10      "method": "[类名]com.lg.controller.TestController[方法名]test15"
11    },
12    {
13      "id": "2",
14      "visitTime": "Jan 6, 2020 8:59:19 PM",
15      "ip": "0:0:0:0:0:0:1",
16      "url": "/test/test15",
17      "executionTime": 4,
18      "method": "[类名]com.lg.controller.TestController[方法名]test15"
19    },
20    {
21      "id": "3",
22      "visitTime": "Jan 6, 2020 8:59:54 PM",
23      "ip": "0:0:0:0:0:0:1",
24      "url": "/test/test15",
25      "executionTime": 1,
26      "method": "[类名]com.lg.controller.TestController[方法名]test15"
27    }
28  ]
29 }

```

访问时间	访问ip	访问资源	执行时间	访问方法
Jan 6, 2020 8:57:34 PM	0:0:0:0:0:0:1	/test/test15	6	[类名]com.lg.controller.TestController[方法名]test15
Jan 6, 2020 8:59:19 PM	0:0:0:0:0:0:1	/test/test15	4	[类名]com.lg.controller.TestController[方法名]test15
Jan 6, 2020 8:59:54 PM	0:0:0:0:0:0:1	/test/test15	1	[类名]com.lg.controller.TestController[方法名]test15
Jan 6, 2020 9:00:57 PM	0:0:0:0:0:0:1	/test/test15	0	[类名]com.lg.controller.TestController[方法名]test15
Jan 6, 2020 9:10:39 PM	0:0:0:0:0:0:1	/emp/transfer	925	[类名]com.lg.controller.TransferEmployeeController[方法名]transfer
Jan 6, 2020 9:11:46 PM	0:0:0:0:0:0:1	/emp/transfer	205	[类名]com.lg.controller.TransferEmployeeController[方法名]transfer
Jan 6, 2020 9:16:10 PM	0:0:0:0:0:0:1	/test/test15	11	[类名]com.lg.controller.TestController[方法名]test15
Jan 6, 2020 9:23:24 PM	0:0:0:0:0:0:1	/emp/transfer	201	[类名]com.lg.controller.TransferEmployeeController[方法名]transfer
Jan 6, 2020 9:23:39 PM	0:0:0:0:0:0:1	/test/transfer	6	[类名]com.lg.controller.TestController[方法名]transfer
Jan 6, 2020 9:24:07 PM	0:0:0:0:0:0:1	/test/transfer	2	[类名]com.lg.controller.TestController[方法名]transfer
Jan 6, 2020 9:29:53 PM	0:0:0:0:0:0:1	/test/test15	6	[类名]com.lg.controller.TestController[方法名]test15
Jan 6, 2020 9:30:41 PM	0:0:0:0:0:0:1	/emp/transfer	15	[类名]com.lg.controller.TransferEmployeeController[方法名]transfer
Jan 6, 2020 9:32:42 PM	0:0:0:0:0:0:1	/test1	5	[类名]com.lg.controller.Test2Controller[方法名]test1

### \* 能够理解Echarts的概述

ECharts是一个免费的、功能强大的图表和可视化库，为您的商业产品添加直观、交互式 and 高度可定制的图表提供了一种简单的方法

#### 概述

它是用纯JavaScript编写的，基于zrender，这是一个全新的轻量级画布库

zrender 是一个轻量级的Canvas类库

官网 <https://www.echartsjs.com/zh/index.html>

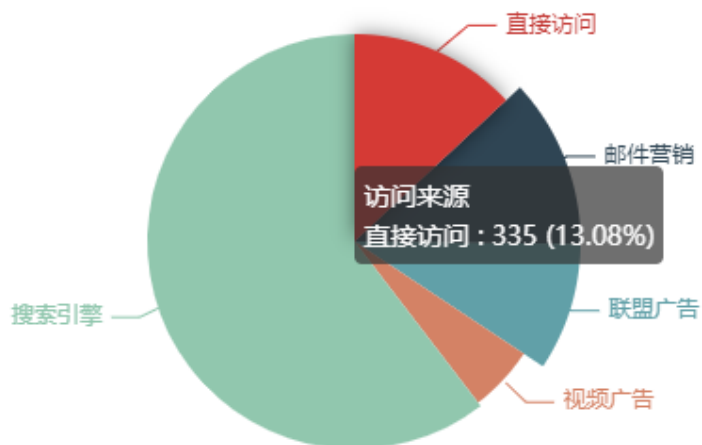
\* <https://www.echartsjs.com/zh/index.html>

### \* 能够使用Echarts的饼图展现出员工男女比例

- 直接访问
- 邮件营销
- 联盟广告
- 视频广告
- 搜索引擎

### 某站点用户访问来源

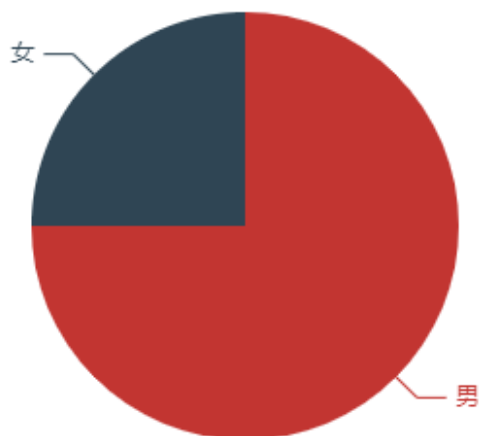
纯属虚构



- 男
- 女

### 员工性别统计

纯属虚构



- \* 案例一（官网例子）
- \* 下载echart依赖库
- \* 代码

```
4 <%@ page contentType="text/html; charset=UTF-8" language="java" isELIgnored="fa
5 <html>
6 <head>
7     <title>pie图演示</title>
8     <meta charset="utf-8">
9     <script src="${pageContext.request.contextPath}/js/jquery-3.3.1.js"></scrip
10 <!-- 引入 ECharts 文件 -->
11 <script src="${pageContext.request.contextPath}/js/echarts.js"></script>
12 <script>
13     $(function () {
14         // 基于准备好的dom，初始化echarts实例
15         var myChart = echarts.init(document.getElementById('main'));
16         // 指定图表的配置项和数据
17         var option = {
18             title: {
19                 text: '某站点用户访问来源',
20                 subttext: '纯属虚构',
21                 left: 'center'
22             },
23             tooltip: {
24                 trigger: 'item',
25                 formatter: '{a} <br/>{b} : {c} ({d}%)'
26             },
27             legend: {
28                 orient: 'vertical',
29                 left: 'left',
30                 data: ['直接访问', '邮件营销', '联盟广告', '视频广告', '搜索引擎']
31             },
32             series: [
33                 {
34                     name: '访问来源',
35                     type: 'pie',
36                     radius: '55%',
37                     center: ['50%', '60%'],
38                     data: [
39                         {value: 335, name: '直接访问'},
40                         {value: 310, name: '邮件营销'},
41                         {value: 234, name: '联盟广告'},
42                         {value: 135, name: '视频广告'},
43                         {value: 1548, name: '搜索引擎'}
```

```

44         ],
45         emphasis: {
46             itemStyle: {
47                 shadowBlur: 10,
48                 shadowOffsetX: 0,
49                 shadowColor: 'rgba(0, 0, 0, 0.5)'
50             }
51         }
52     }
53 ]
54 };
55 // 使用刚指定的配置项和数据显示图表。
56 myChart.setOption(option);
57 });
58 </script>
59 </head>
60 <body>
61     <!-- 为 ECharts 准备一个具备大小（宽高）的 DOM -->
62     <div id="main" style="width: 600px;height:400px;"></div>
63 </body>
64 </html>
65
66 * 案例二：（统计员工男女性别）
67 * 实体
68 @Data
69 @AllArgsConstructor
70 @NoArgsConstructor
71 public class ResResult {
72     private String code;
73     private Object msg;
74 }
75
76 * dao
77 public interface EmployeeDao {
78     @Select("SELECT COUNT(*) FROM employee WHERE sex=#{sex}")
79     Integer selectEmpCountBySex(String sex);
80 }
81 * service
82 public interface EmployeeService {
83     List<Integer> queryEmployeeSexCount();

```

```

84 }
85 @Service
86 public class EmployeeServiceImpl implements EmployeeService {
87     @Autowired
88     private EmployeeDao employeeDao;
89     @Transactional
90     @Override
91     public List<Integer> queryEmployeeSexCount() {
92         List<Integer> list=new ArrayList<>();
93         list.add(employeeDao.selectEmpCountBySex("男"));
94         list.add(employeeDao.selectEmpCountBySex("女"));
95         return list;
96     }
97 }
98
99 * Controller
100 @RestController
101 @RequestMapping("/emp")
102 public class EmployeeController {
103     @Autowired
104     private EmployeeService employeeService;
105     @RequestMapping("/sexCount")
106     public ResResult queryEmployeeSexCount(){
107         ResResult resResult=new ResResult();
108         List<Integer> list = employeeService.queryEmployeeSexCount();
109         resResult.setCode("200");
110         resResult.setMsg(list);
111         return resResult;
112     }
113 }
114 * 测试
115 @Test
116 public void test18() throws Exception {
117     String result = mockMvc.perform(delete("/emp/sexCount")).
118         andExpect(status().isOk()).
119         andDo(print()).andReturn().getResponse().getContentAsString();
120     System.out.println(result);
121 }
122 * 页面
123 <%@ page contentType="text/html; charset=UTF-8" language="java" isELIgnored="fa

```

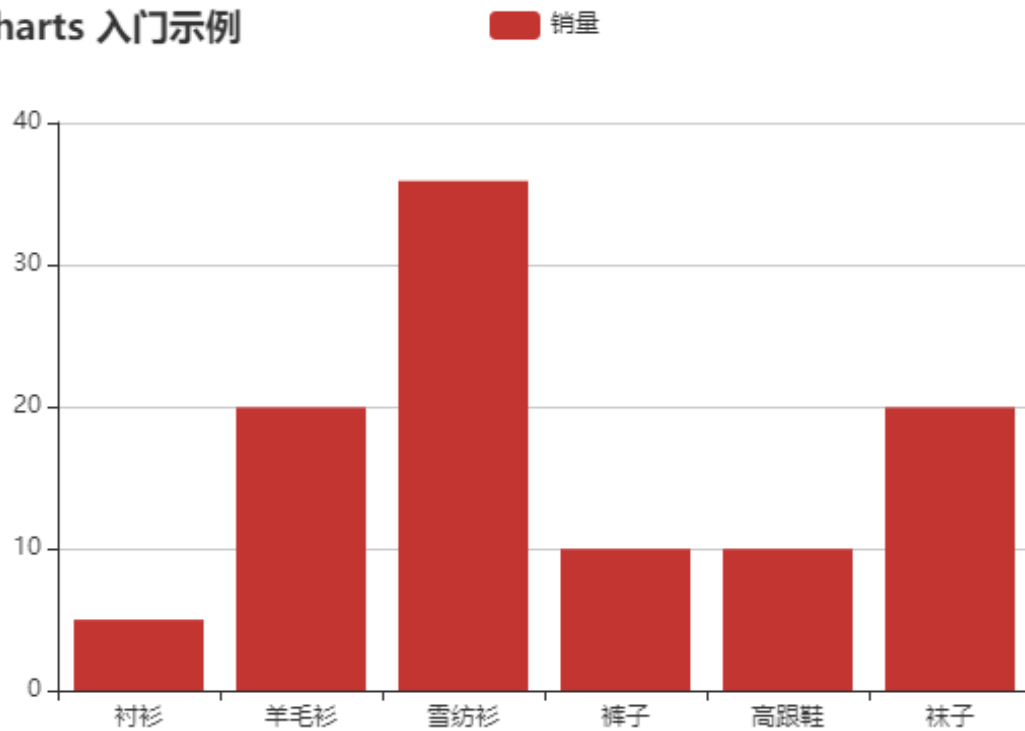


[illegible]

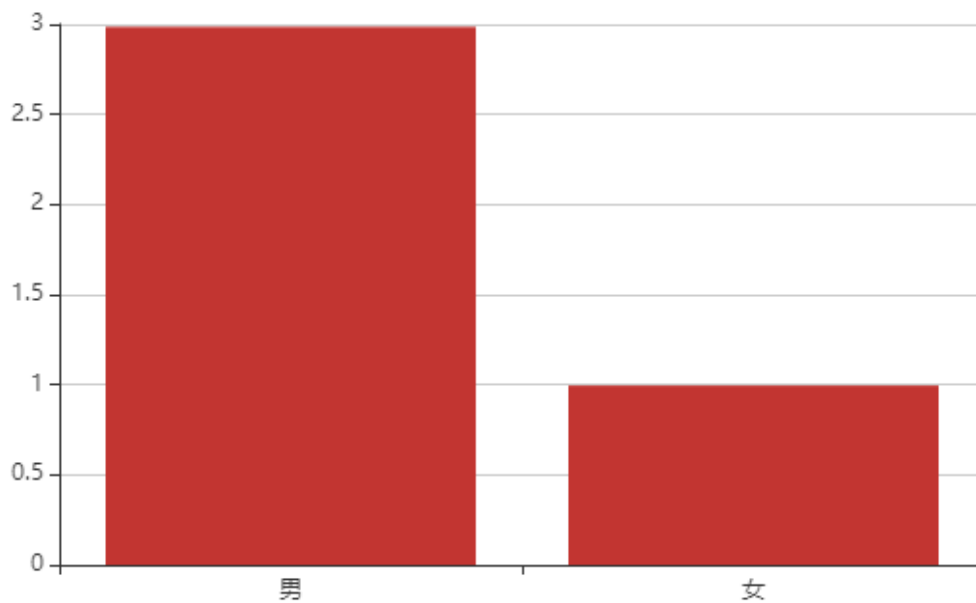
```
164         }
165     }
166 }
167 ]
168 };
169 // 使用刚指定的配置项和数据显示图表。
170 myChart.setOption(option);
171 var url="/ssm02/emp/sexCount";
172 $.ajax({
173     type:"POST",
174     url:url,
175     contentType:"application/json;charset=utf-8",
176     success:function(result,status){
177         var content = JSON.parse(JSON.stringify(result));
178         if(content.code==200){
179             // 填入数据
180             myChart.setOption({
181                 series: [{
182                     data: [
183                         {value: content.msg[0], name: '男'},
184                         {value: content.msg[1], name: '女'},
185                     ]
186                 }]
187             });
188         }
189     },
190     error:function(result){
191         alert("请求数据失败")
192     }
193 });
194 });
195 </script>
196 </head>
197 <body>
198     <!-- 为 ECharts 准备一个具备大小（宽高）的 DOM -->
199     <div id="main" style="width: 600px;height:400px;"></div>
200 </body>
201 </html>
202
```

\* 能够使用Echarts的柱状图展现出员工男女比例

## ECharts 入门示例



## 员工性别统计



```
1 * 案例一
2 <%@ page contentType="text/html; charset=UTF-8" language="java" isELIgnored="false"%>
3 <html>
4 <head>
5     <title>柱状图演示</title>
```

```

6    <meta charset="utf-8">
7    <script src="${pageContext.request.contextPath}/js/jquery-3.3.1.js"></script>
8    <!-- 引入 ECharts 文件 -->
9    <script src="${pageContext.request.contextPath}/js/echarts.js"></script>
10   <script>
11       $(function () {
12           // 基于准备好的dom，初始化echarts实例
13           var myChart = echarts.init(document.getElementById('main'));
14           // 指定图表的配置项和数据
15           var option = {
16               title: {
17                   text: 'ECharts 入门示例'
18               },
19               tooltip: {},
20               legend: {
21                   data: ['销量']
22               },
23               xAxis: {
24                   data: ["衬衫", "羊毛衫", "雪纺衫", "裤子", "高跟鞋", "袜子"]
25               },
26               yAxis: {},
27               series: [{
28                   name: '销量',
29                   type: 'bar',
30                   data: [5, 20, 36, 10, 10, 20]
31               }]
32           };
33           // 使用刚指定的配置项和数据显示图表。
34           myChart.setOption(option);
35       });
36   </script>
37 </head>
38 <body>
39     <!-- 为 ECharts 准备一个具备大小（宽高）的 DOM -->
40     <div id="main" style="width: 600px;height:400px;"></div>
41 </body>
42 </html>
43
44 * 案例二
45 <%@ page contentType="text/html; charset=UTF-8" language="java" isELIgnored="fa

```

```
46 <html>
47 <head>
48     <title>柱状图演示</title>
49     <meta charset="utf-8">
50     <script src="${pageContext.request.contextPath}/js/jquery-3.3.1.js"></script>
51     <!-- 引入 ECharts 文件 -->
52     <script src="${pageContext.request.contextPath}/js/echarts.js"></script>
53     <script>
54         $(function () {
55             // 基于准备好的dom，初始化echarts实例
56             var myChart = echarts.init(document.getElementById('main'));
57             // 指定图表的配置项和数据
58             var option = {
59                 title: {
60                     text: '员工性别统计'
61                 },
62                 tooltip: {},
63                 legend: {
64                     data: ['性别统计']
65                 },
66                 xAxis: {
67                     data: ["男","女"]
68                 },
69                 yAxis: {},
70                 series: [{
71                     name: '性别',
72                     type: 'bar',
73                     data: []
74                 }]
75             };
76             // 使用刚指定的配置项和数据显示图表。
77             myChart.setOption(option);
78             var url="/ssm02/emp/sexCount";
79             $.ajax({
80                 type:"POST",
81                 url:url,
82                 contentType:"application/json;charset=utf-8",
83                 success:function(result,status){
84                     var content = JSON.parse(JSON.stringify(result));
85                     if(content.code==200){
```

```
86         // 填入数据
87         myChart.setOption({
88             series: [{
89                 // 根据名字对应到相应的系列
90                 name: '性别',
91                 data: content.msg
92             }]
93         });
94     },
95     },
96     error:function(result){
97         alert("请求数据失败")
98     }
99 });
100 });
101 </script>
102 </head>
103 <body>
104     <!-- 为 ECharts 准备一个具备大小（宽高）的 DOM -->
105     <div id="main" style="width: 600px;height:400px;"></div>
106 </body>
107 </html>
```