第二课：自定义报表宏

在自定义报表宏之前，建议对报表宏事件要有个基本的概念及了解。

也建议了解跳转规则向导，根据向导生成的宏脚本具有很大的参考意义，节省编写宏的时间。

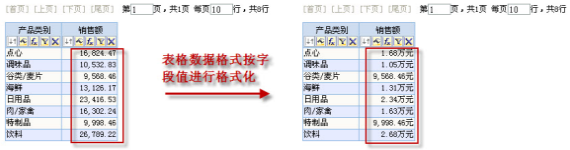
客户端模块

这个模块下的宏是在浏览器端运行，所以语法就是JavaScript，需要有前端开发的经验才可能灵活编写，调试也是使用浏览器前端调试即可。

注意：客户端宏主要影响浏览器端的展现效果，除了图形（直接加在图形对象上的部分宏功能可以对导出生效），其他报表的客户端宏都不会对导出有效果。

示例说明

在组合分析中，根据字段值的大小，使表格数据进行格式化。

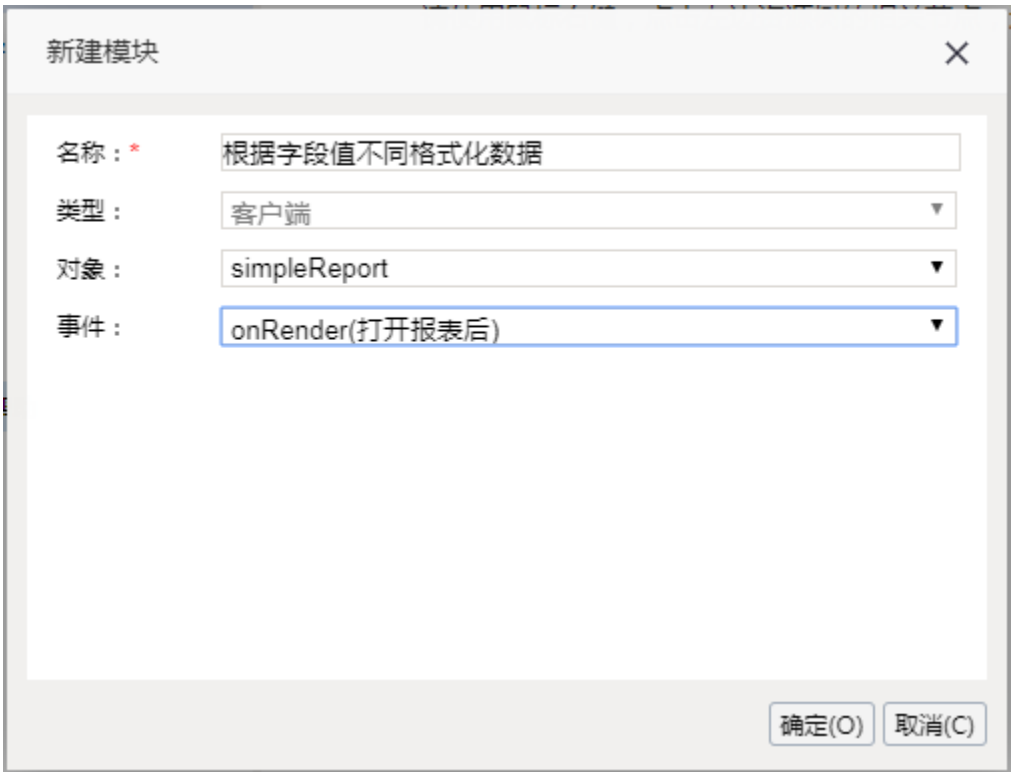


第一步：进入报表宏编辑界面

对灵活分析报表进行宏编辑，如图所示：

第二步：选择报表宏触发事件

1. 在"宏编辑"界面的左上角选择"客户端模块"，点击右键"新建模块"。
2. 选择"simpleReport"对象（当前报表对象），事件选择"onRenderTable"（表格刷新事件）。目的是报表的表格刷新完后触发我们编写的宏代码脚本，如图所示：



第三步：报表宏脚本编写

编写报表宏代码。

名称：根据字段值不同格式化 类型：ClientSide 对象：simpleReport 事件：onRender

```
function main(simpleReport, simpleReportContext) {
    //获取表头行数
    var headerRows = simpleReport.grid.getHeaderRows();
    //获取总行数
    var rows = simpleReport.grid.getRowCount();
    //取序号所在的列号
    var n = simpleReport.getFieldIndexByAlias("销售额");
    for (var i = headerRows; i < rows; i++) {
        //获取金额
        var cell1 = simpleReport.grid.getCell(i, n);
        var value1 = cell1.innerText;
        //把金额装换为浮点型
        var floatValue1 = parseFloat(value1.replace(/,/g, ""));
        if (floatValue1 > 10000) {
            //金额除以10000
            var money = floatValue1 / 10000;
            //金额保留两位小数并且后缀加万元
            money = money.toFixed(2) + '万元';
            //使用格式处理后的金额替换原来金额
            simpleReport.grid.getCell(i, n).innerText = money;
        } else {
            //直接加后缀元
            var money = value1 + '元';
            //使用格式处理后的金额替换原来金额
            simpleReport.grid.getCell(i, n).innerText = money;
        }
    }
}
```

宏代码示例

```
function main(simpleReport, simpleReportContext) {
    //
    var headerRows = simpleReport.grid.getHeaderRows();
    //
    var rows = simpleReport.grid.getRowCount();
    //
    var n = simpleReport.getFieldIndexByAlias("");
    for (var i = headerRows; i < rows; i++) {
        //
        var cell1 = simpleReport.grid.getCell(i, n);
        var value1 = cell1.innerText;
        //
        var floatValue1 = parseFloat(value1.replace(/,/g, ""));
        if (floatValue1 > 10000) {
            //10000
            var money = floatValue1 / 10000;
            //
            money = money.toFixed(2) + '';
            //
            simpleReport.grid.getCell(i, n).innerText = money;
        } else {
            //
            var money = value1 + '';
            //
            simpleReport.grid.getCell(i, n).innerText = money;
        }
    }
}
```

服务端模块

这个模块下的宏是在应用服务器上运行的，底层原理和自定义计划任务脚本一样的，都是使用了 Rhino 工具包，语法遵循Javascript 语法规范，能够引用 Java 类并创建 Java 对象来使用，详细的语法说明见自定义计划任务。

服务端宏因为在服务器上运行，对浏览器端展现和导出报表都是生效的，但并不是所有报表类型都有服务端宏（编辑对应报表宏时就可以看到有没有）。

示例说明

对多维分析增加一个汇总项。

商店	Gender	1997年1季		1997年2季	
		存储成本	店铺销售	存储成本	店铺销售
CA	F	7,156.39	17,928.37	7,663.24	19,146.41
	M	7,274.70	18,246.83	7,668.78	19,250.34
OR	F	7,835.22	19,602.14	6,305.07	15,812.72
	M	8,245.85	20,568.15	6,373.89	15,960.16
WA	F	12,830.16	32,267.72	12,341.27	30,898.01
	M	12,409.93	31,015.14	12,611.98	31,598.63

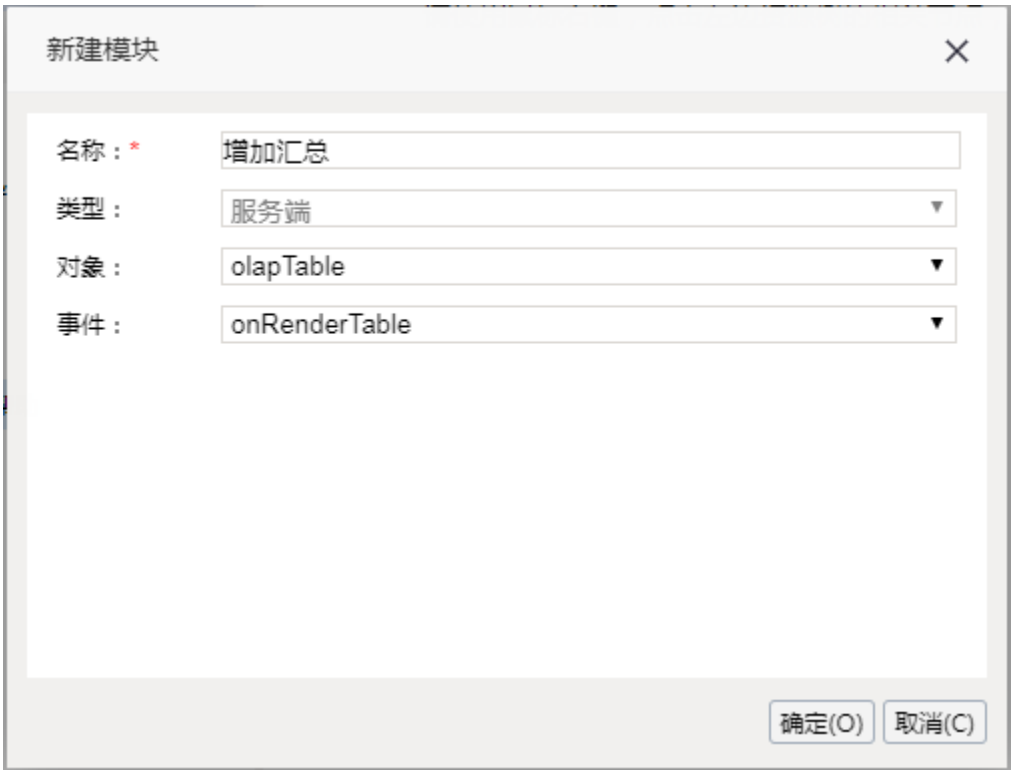
商店			Gender		1997年1季		1997年2季	
					存储成本	店铺销售	存储成本	店铺销售
All Stores	美国	CA	All Gender	F	7,156.39	17,928.37	7,663.24	19,146.41
				M	7,274.70	18,246.83	7,668.78	19,250.34
		OR	All Gender	F	7,835.22	19,602.14	6,305.07	15,812.72
				M	8,245.85	20,568.15	6,373.89	15,960.16
		WA	All Gender	F	12,830.16	32,267.72	12,341.27	30,898.01
				M	12,409.93	31,015.14	12,611.98	31,598.63
合计			合计		55752.24	139628.35	52964.22	132666.27

第一步：进入报表宏编辑界面

对多维分析报表进行宏编辑，如图所示：

第二步：选择报表宏触发事件

1. 在"宏编辑"界面的左上角选择"服务端模块"，点击右键"新建模块"。
2. 选择"olapTable"对象（当前报表对象），事件选择"onRenderTable"（表格刷新事件）。目的是报表的表格刷新完后触发我们编写的宏代码脚本，如图所示：



第三步：报表宏脚本编写

编写报表宏代码。

名称：增加汇总　类型：ServerSide　对象：olapTable　事件：onRenderTable

```
var jmUtils = use("system.utils.JMServerUtils");
function main(olapTable) {
    //获取表格总行数
    var rowsize = olapTable.getRowSize();
    //获取表格总列数
    var columnsize = olapTable.getColumnSize();
    //存放合计值
    var talValueList = [];
    //汇总
    for (var i = 0; i < rowsize; i++) {
        for (var j = 0; j < columnsize; j++) {
            if (talValueList[j] == null) talValueList[j] = 0;
            if (olapTable.getRow.getData()[j].getDouble() != null) {
                talValueList[j] += olapTable.getRow.getData()[j].getDouble().doubleValue();
            }
        }
    }
    //对单元格赋值
    var newRow = olapTable.appendRow();
    for (var j = 0; j < columnsize; j++) {
        //保留两位小数
        newRow.getData()[j].setValue(talValueList[j].toFixed(2));
    }
    var pos = newRow.getPosition();
    pos.getMembers()[0].setCaption("合计");
    pos.getMembers()[1].setCaption("合计");
}
```

宏代码示例

```
var jmUtils = use("system.utils.JMServerUtils");
function main(olapTable) {
    //
    var rowsize = olapTable.getRowSize();
    //
    var columnsize = olapTable.getColumnSize();
    //
    var talValueList = [];
    //
    for (var i = 0; i < rowsize; i++) {
        for (var j = 0; j < columnsize; j++) {
            if (talValueList[j] == null) talValueList[j] = 0;
            if (olapTable.getRow.getData()[j].getDouble() != null) {
                talValueList[j] += olapTable.getRow.getData()[j].getDouble().doubleValue();
            }
        }
    }
    //
    var newRow = olapTable.appendRow();
    for (var j = 0; j < columnsize; j++) {
        //
        newRow.getData()[j].setValue(talValueList[j].toFixed(2));
    }
    var pos = newRow.getPosition();
    pos.getMembers()[0].setCaption("");
    pos.getMembers()[1].setCaption("");
}
```