FineReportAPI

1chart

```
public abstract class ChartConfigPane <T extends Charts> extends ChartsConfigPane<T> {
    private JPanel northJpane = new JPanel();
    private JPanel centerJpane = new JPanel();
    private UILabel nameUILabel = new UILabel("");
    private UILabel demoUILabel = new UILabel("");
    private UILabel refreshTimeUILabel = new UILabel("");
    /**
     *
     *
     * UIObserver
     * UI***
     */
    //UI
    protected UITextField value = new UITextField();
    protected UISpinner autoRefreshTime = new UISpinner(0, Integer.MAX_VALUE, 1, 0);
    //
    protected CustomTextfield customTextArea = new CustomTextfield();
    protected ChartCollection chartCollection;
    public ChartConfigPane() //
    {
        this.setLayout(new BorderLayout());
        northJpane.setLayout(new GridLayout(4, 1, 10,10));
        northJpane.setLayout(new GridLayout(6, 1, 10,10));
        northJpane.add(nameUILabel);
        northJpane.add(value);
        northJpane.add(demoUILabel);
        northJpane.add(customTextArea);
        northJpane.add(refreshTimeUILabel);
        northJpane.add(autoRefreshTime);
        northJpane.setBorder(BorderFactory.createEmptyBorder(10,15,10,15));
        this.add(northJpane, BorderLayout.NORTH);
        this.add(centerJpane, BorderLayout.CENTER);
        this.setSize(200, 200);
        this.setVisible(true);
        //
        initAllListeners();
    }

  @Override
    protected void populate(ChartCollection collection, PieChart selectedChart) {
        this.chartCollection=collection;
        value.setText(selectedChart.getChartTitle());
        customTextArea.setText(selectedChart.getChartDescribe());
        autoRefreshTime.setValue(selectedChart.getRefreshTime());
    }
    @Override
    protected void update(ChartCollection collection, PieChart selectedChart) {
        selectedChart.setChartTitle(value.getText());
        selectedChart.setChartDescribe(customTextArea.getText());
        selectedChart.setRefreshTime((int) autoRefreshTime.getValue());
    }
}
```

2chart

chart

```
public class PieChart extends Charts<NormalChartData> {
    private String chartTitle = "demo";
    private String chartDescribe = "";
    private int refreshTime = 0;
    private static final NumberFormat format = new DecimalFormat("##%");

    public String getChartTitle() {
        return chartTitle;
```

```
}
public void setChartTitle(String chartTitle) {
    this.chartTitle = chartTitle;
}
public String getChartDescribe() {
    return chartDescribe;
}
public void setChartDescribe(String chartDescribe) {
    this.chartDescribe = chartDescribe;
}
@Override
public int getRefreshTime() {
    return refreshTime;
}
public void setRefreshTime(int refreshTime) {
    this.refreshTime = refreshTime;
}
/**
 * Object
 * @param ob  Object
 * @return boolean
 *
 */
public boolean equals(Object ob) {
    return ob instanceof Chart && super.equals(ob)
            && ComparatorUtils.equals(((PieChart) ob).getChartTitle(), chartTitle)
            && ComparatorUtils.equals(((PieChart) ob).getChartDescribe(), chartDescribe)
            && ComparatorUtils.equals(((PieChart) ob).getRefreshTime(), refreshTime)
            ;
}
@Override
public void writeXML(XMLPrintWriter xmlPrintWriter) {
    super.writeXML(xmlPrintWriter);
    xmlPrintWriter.startTAG("customChartDemo")
            .attr("chartTitle", getChartTitle())
            .attr("chartDescribe", getChartDescribe())
            .attr("refreshTime", getRefreshTime())
            .end();
}
@Override
public void readXML(XMLableReader xmLableReader) {
    super.readXML(xmLableReader);
    if (xmLableReader.isChildNode()) {
        String tagName = xmLableReader.getTagName();
        if (tagName.equals("customChartDemo")) {
            setChartTitle(xmLableReader.getAttrAsString("chartTitle", ""));
            setChartDescribe(xmLableReader.getAttrAsString("chartDescribe", ""));
            setRefreshTime(xmLableReader.getAttrAsInt("refreshTime", 0));
        }
    }
}
/**
 *
 * getChartData()
 */
@Override
//tojsongetChartData()json
public JSONObject toJSON(Repository repo) throws JSONException {
    JSONObject jsonObject = JSONObject.create();
    jsonObject.put("series", seriesJSONArray())
              .put("title", JSONObject.create().put("text", getChartTitle()));
    jsonObject.put("refreshTime", refreshTime);
    return jsonObject;
}
private JSONArray seriesJSONArray() throws JSONException {
    NormalChartData normalChartData = getChartData();
    int seriesLen = normalChartData.getSeriesCount();
    String radius = format.format(1.0 / seriesLen);
    //
    JSONArray series = JSONArray.create();
    for (int index = 0; index < seriesLen; index++) {
```

```
            JSONObject wrapper = JSONObject.create()
                    .put("type", "pie")
                    .put("data", pointJSONArray(index)) //data
                    .put("name", normalChartData.getSeries_array()[index].toString());//
            if (seriesLen > 1) {
                JSONArray center = JSONArray.create()
                        .put(format.format((float) index / (seriesLen + 1) + 0.20))
                        .put("55%");
                wrapper.put("radius", radius)
                        .put("center", center);
            }
            series.put(wrapper);
        }
        return series;
    }
    private JSONArray pointJSONArray(int index) throws JSONException {
        NormalChartData normalChartData = getChartData();
        int categoriesLen = normalChartData.getCategoryLabelCount();
        JSONArray point = JSONArray.create();
        for (int i = 0; i < categoriesLen; i++) {
            JSONObject item = JSONObject.create();
            //name
            String name = normalChartData.getCategory_array()[i].toString();
            item.put("name", name);
            if (normalChartData.getSeries_value_2D().length > 0) {
                //data
                item.put("value", normalChartData.getSeries_value_2D()[index][i]);
            }
            point.put(item);
        }
        return point;
    }
    @Override
    public String getChartID() {
        return "ChartsPieWithCommenDataPane";
    }

}
```

ajax

ajax

```
/**
 * Created by richie on 16/1/29.
 */
EChartsFactory = function(options, $dom) {
    debugger;
    this.options = options;
    this.$dom = $dom;
    this.chartID = options.chartID;
    this.autoRefreshTime = options.chartAttr.refreshTime || 0;
    this.width = options.width || $dom.width();// dom.
    this.height = options.height || $dom.height();
    this.sheetIndex = options.sheetIndex || 0;
    this.ecName = options.ecName || '';
    FR.Chart.WebUtils._installChart(this, this.chartID);
};
EChartsFactory.prototype = {
    constructor : EChartsFactory,
    inits : function() {
        debugger;
```

```
        // this.options.chartAttr
        var ct = this.options.chartAttr;
        // function clear(str){return str.replace(/\s/g, '');}
        // var cleanct = clear(ct);
        // console.log(cleanct);
        // var json = JSON.parse(cleanct);
        this.newCharts = echarts.init(this.$dom[0], EChartsTheme[ct.theme]);
        this.newCharts.setOption(ct);
        this._autoRefresh(this.autoRefreshTime);
    },

    //
    _autoRefresh: function (time) {
        if (time < 1) {
            return;
        }
        var self = this;
        setInterval(function () {
            var ID = FR.cjkEncode(self.chartID);
            var width = self.width || 0;
            var height = self.height || 0;
            var sheetIndex = self.sheetIndex;
            var ecName = self.ecName;
            if (FR.servletURL) {
                FR.ajax({
                    type: 'GET',
                    url: FR.servletURL + '?op=chartauto',
                    data: {
                        cmd: 'chart_auto_refresh',
                        sessionID: FR.SessionMgr.getSessionID(),
                        chartID: ID,
                        chartWidth: width,
                        chartHeight: height,
                        sheetIndex: sheetIndex,
                        ecName: ecName,
                        __time: new Date().getTime()
                    },
                    dataType: 'json',
                    success: function (chartRelateJS) {
                        var attrList = chartRelateJS.relateChartList;
                        for (var i = 0, len = attrList.length; i < len; i++) {
                            var chartID = FR.Chart.WebUtils._getChartIDAndIndex(attrList[i].id);
                            var chartSet = FR.ChartManager[chartID[0]];
                            if (chartSet && chartSet[chartID[1]]) {
                                var chartAttr = attrList[i].chartAttr;
                                self._setAutoRefreshData(chartSet[chartID[1]], chartAttr);
                            }
                        }
                    }
                });
            }
        }, time * 1000);
    },
    resize : function() {
        this.newCharts.resize();
    },
    refresh:function() {
    },
    refreshData:function(options){
    },
    //
    setData:function(options, aimation){
    }
};
```