

demo

FineReport

demo

```
plugins-calendarchart
--src
  --com.fr.plugins.calendarchart
  --custompie
    --ChartConfigPane.java
    --DemoChartsPie.java
    --demoChartsPieUI.java
    --PieChart.java
  --images
    --pie256.png
  --web
    --echarts.bridge.js
    --echarts.loader.js
    --echarts.min.js
    --EChartsFileLoader.java
--build.xml
--plugin.xml
--plugins-calendarchart.iml
```

1.custompie

2.images

3.web

js

4.build.xml

5.plugin.xml

6.plugins-calendarchart.iml

IEDA

demo

1.demo

1

-
-

js

2 plugin.xmlbuild.xml

2.

1) + jsjs

Charts

AbstractIndependentChartsProvider

AbstractJavaScriptFileHandler

Charts

toJSON()getChartData()jsonjson

getChartID():plugin.xml

writeXMLreadXML

```
public class YourCharts extends Charts {
    @Override

    public JSONObject toJSON(Repository repo) throws JSONException {

        //getChartData()jsonjson
        return null;
    }

    @Override

    public String getChartID() {

        //plugin.xmlplotID
        return yourPlotID;
    }

    @Override

    public void writeXML(XMLPrintWriter xmlPrintWriter) {

        //xml
    }

    @Override

    public void readXML(XMLableReader xmlableReader) {

        //writeXML
    }
}
```

AbstractIndependentChartsProvider

getChartName()

getChartTypes()

getRequiredJS()js

getWrapperName()JSdom

getChartImagePath()images

currentAPILevel()API

```

public class YourChartsProvider extends AbstractIndependentChartsProvider {

    @Override

    public String getChartName() {

        return "";
    }

    @Override

    public Chart[] getChartTypes() {

        //YourCharts
        return new YourCharts[]{new YourCharts()};
    }

    @Override

    public String[] getRequiredJS() {

        //jsjsecharts.js
        return new String[]{
            "/com/fr/plugins/xxx/web/echarts.bridge.js"
        };
    }

    @Override

    public String getWrapperName() {

        //JSdom
        return "EChartsFactory";
    }

    @Override

    public String getChartImagePath() {

        //
        return "com/fr/plugins/xxx/images/xxx.png";
    }

    @Override

    public int currentAPILevel() {

        //API
        return CURRENT_API_LEVEL;
    }
}

```

AbstractJavaScriptFileHandler

pathsForFiles()Echartsechartsjs

```

return new String[]{
    "/com/fr/plugin/parallelchart/web/echarts.loader.js",
    "/com/fr/plugin/parallelchart/web/echarts.min.js"
};

```

g2g2js

```

return new String[]{
    "/com/fr/plugin/parallelchart/web/g2.js"
};

```

encode()

```
public class EchartFileLoader extends AbstractJavaScriptFileHandler {  
    @Override  
  
    public String[] pathsForFiles() {  
        //js  
        return new String[]{  
            "/com/fr/plugin/parallelchart/web/echarts.loader.js",  
            "/com/fr/plugin/parallelchart/web/echarts.min.js"  
        };  
    }  
  
    @Override  
  
    public String encode() {  
        return EncodeConstants.ENCODING_UTF_8;  
    }  
}
```

2 + +js

ChartsConfigPaneYourChartsConfigPanepopulateupdateCharts

YourChartsConfigPane()

Populate():YourCharts

Update():YourCharts

```

public class YourChartsConfigPane extends ChartsConfigPane<YourCharts> {

    private ChartCollection chartCollection;
    private UITextField value;

    public YourChartsConfigPane() {

        //
        this.setLayout(new BorderLayout());
        northJpane.setLayout(new GridLayout(4, 1, 10,10));
        northJpane.add(nameUILabel);
        northJpane.add(value);
        northJpane.add(demoUILabel);
        northJpane.add(customTextArea);
        this.add(northJpane, BorderLayout.NORTH);
        this.add(centerJpane, BorderLayout.CENTER);
        this.setSize(200, 200);
        this.setVisible(true);

        //
        initAllListeners();
    }

    //
    class ColorEventListener implements ActionListener {

        @Override

        public void actionPerformed(ActionEvent e) {

            update(chartCollection);
        }
    }

    @Override

    //
    public Class<?extends Charts> accptType() {

        return YourCharts.class;
    }

    @Override

    public void populate(ChartCollection collection, YourCharts selectedChart) {

        chartCollection = collection;

        //YourCharts
        value.setText(selectedChart.getCustomData());
    }

    @Override

    public void update(ChartCollection collection, YourCharts selectedChart) {

        //YourCharts
        selectedChart.setCustomData(value.getText());
    }
}

```

ChartsYourChartswriteXMLreadXML

writeXML():XMLdemo

readXML():xmldemo

```
public class YourCharts extends Charts {

    //writeXMLreadXML
    @Override

    public void writeXML(XMLPrintWriter xmlPrintWriter) {

        //XML
        xmlPrintWriter.startTAG(TAG_NAME)
            .attr("custom", getCustomData())
            .end();
        writeDefinition(xmlPrintWriter);
    }

    @Override

    public void readXML(XMLLabelReader xmlLabelReader) {

        //XML
        if (xmlLabelReader.isChildNode()) {

            String tagName = xmlLabelReader.getTagName();
            if (tagName.equals("customChartDemo")) {
                setCustomData(xmlLabelReader.getAttrAsString("custom", "111"));
            }
        }
    }
}
```

3 ++js

3

AbstractTableDataContentPane

populateBean
updateBean
checkBoxUsebox
clearAllBoxListbox
refreshBoxListWithSelectTableDatabox

```
public class TableDataContentPane extends AbstractTableDataContentPane {

    private JComboBox dateComboBox;

    private JComboBox valueComboBox;

    public TableDataContentPane(ChartDataPane parent) {

        dateComboBox = new JComboBox();
        valueComboBox = new JComboBox();
        dateComboBox.setPreferredSize(new Dimension(100, 20));
        valueComboBox.setPreferredSize(new Dimension(100, 20));

        Component[][] components = new Component[][]{

            new Component[]{new UILabel("", SwingConstants.RIGHT), dateComboBox},

            new Component[]{new UILabel("", SwingConstants.RIGHT), valueComboBox}};

        double p = TableLayout.PREFERRED;
        double[] columnSize = {p, p};
        double[] rowSize = {p, p, p};

        JPanel panel = TableLayoutHelper.createTableLayoutPanel(components, rowSize, columnSize);
```

```

        setLayout(new BorderLayout());

        add(panel, BorderLayout.CENTER);
    }

    @Override

    public void populateBean(ChartCollection collection) {

        DefaultTableDataDefinition configuration = (DefaultTableDataDefinition)
            collection.getSelectedChart().getFilterDefinition();

        if (configuration == null) return;

        combineCustomEditValue(dateComboBox, configuration.getDate());
        combineCustomEditValue(valueComboBox, configuration.getValue());
    }

    @Override

    public void updateBean(ChartCollection ob) {

        DefaultTableDataDefinition myConfiguration = new DefaultTableDataDefinition();
        Object wname = dateComboBox.getSelectedItem();
        Object wvalue = valueComboBox.getSelectedItem();

        if (wname != null) {
            myConfiguration.setDate(wname.toString());
        }

        if (wvalue != null) {
            myConfiguration.setValue(wvalue.toString());
        }

        ob.getSelectedChart().setFilterDefinition(myConfiguration);
    }

    @Override

    public void clearAllBoxList() {

        dateComboBox.removeAll();
        valueComboBox.removeAll();
    }

    @Override

    protected void refreshBoxListWithSelectTableData(List columnNameList) {

        refreshBoxItems(dateComboBox, columnNameList);
        refreshBoxItems(valueComboBox, columnNameList)
    }
}

```

TableDataDefinition3

```

public class DefaultTableDataDefinition extends TableDataDefinition {

    public static final String XML_TAG = "DefaultTableDataDefinition";
    private String date;
    private String value;

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }
}

```

```

    }

    public String getValue() {
        return value;
    }

    public void setValue(String value) {
        this.value = value;
    }

    @Override

    public ChartData createChartData(DataModel resultSet, DataProcessor dataProcessor) {

        Map<String, String> data = new HashMap<>();

        try {

            int wordNameCol = DataCoreUtils.getColumnIndexByName(resultSet, getDate());
            int wordValueCol = DataCoreUtils.getColumnIndexByName(resultSet, getValue());

            Map<Object, List<Object>> map = new HashMap<>();

            for (int rowIndex = 0; rowIndex < resultSet.getRowCount(); rowIndex++) {

                Object wordName = resultSet.getValueAt(rowIndex, wordNameCol);
                Object wordValue = resultSet.getValueAt(rowIndex, wordValueCol);

                if (wordName != null && wordValue != null) {

                    if (!map.containsKey(wordName)) {
                        map.put(wordName, new ArrayList<>());
                    }

                    map.get(wordName).add(wordValue);

                }

            }

            for (Map.Entry<Object, List<Object>> entry : map.entrySet()){

                data.put(entry.getKey().toString(),
                    entry.getValue().get(0).toString());

            }

        } catch (Exception e) {

            FRLogger.getLogger().error(e.getMessage(), e);

        }

        return new TableDataContent(data);
    }

    public void writeXML(XMLPrintWriter writer) {

        writer.startTAG(XML_TAG)
            .attr("date", getDate())
            .attr("value", getValue());
        super.writeXML(writer);
        writer.end();
    }

    public void readXML(XMLeableReader reader) {

        super.readXML(reader);

        if (reader.isAttr()) {

            String tmpVal;

```



```

        if ((tmpVal = reader.getAttrAsString("date", null)) != null){
            this.setDate(tmpVal);
        }

        if ((tmpVal = reader.getAttrAsString("value", null)) != null){
            this.setValue(tmpVal);
        }
    }

    public boolean equals(Object ob) {

        return ob instanceof DefaultTableDataDefinition
        && ComparatorUtils.equals(((DefaultTableDataDefinition) ob).getDate(), this.getDate())
        && ComparatorUtils.equals(((DefaultTableDataDefinition) ob).getValue(), this.getValue())
        && super.equals(ob);
    }

    public Object clone() throws CloneNotSupportedException {

        DefaultTableDataDefinition cloned = (DefaultTableDataDefinition) super.clone();
        cloned.setDate(getDate());
        cloned.setValue(getValue());
        return cloned;
    }
}

```

NormalChartData2

```

public class TableDataContent extends NormalChartData {

    private final Map<String, String> data;

    TableDataContent(Map<String, String> data) {
        this.data = data;
    }

    public Map<String, String> getData() {
        return data;
    }
}

```

AbstractIndependentChartsUIOverride

```

@Override

public AbstractTableDataContentPane getTableDataSourcePane(Plot plot, ChartDataPane parent) {

    //
    return new YourDataContentPane(parent);
}

```

4JS

echarts.bridge.jsEChartsFactory

function

```
EChartsFactory = function(options, $dom) {  
  
    this.options = options;  
    this.$dom = $dom;  
    this.chartID = options.chartID;  
    this.autoRefreshTime = options.autoRefreshTime || 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);  
};
```

prototypeEchartsprototype

```

EChartsFactory.prototype = {

    constructor : EChartsFactory,

    inits : function() {

        // this.options.chartAttr
        var ct = this.options.chartAttr;

        //EchartsmyChart
        var myChart = echarts.init(this.$dom[0]);

        //data
        var data = ct.data;
        var max = 0;

        for (var i = 0; i < data.length; i += 1) {
            if (parseInt(max) < parseInt(data[i][1])) {
                max = data[i][1];
            }
        }

        var year = echarts.number.parseDate(data[0][0]).getFullYear();
        max = max / 20;

        //title
        var title = ct.title;

        //titletooltiplelegend
        option = {

        }

        this.newCharts.setOption(ct);
    },

    resize : function() {

        this.newCharts.resize();
    },

    refresh:function() {

    },

    refreshData:function(options){

    },

    //
    setData:function(options, aimation){

    }

};

```

AbstractLocaleFinderfind()

```

import com.fr.stable.fun.impl.AbstractLocaleFinder;

public class youclassname extends AbstractLocaleFinder {

    @Override

    public int currentAPILevel() {
        return CURRENT_LEVEL;
    }

    @Override

    public String find() {
        return "com/fr/plugin-XXX/locale/XXX";
    }
}

```

com/fr/plugin-XXX/locale/XXX.properties

Plugin-XXX=XX

Plugin-XXX_title=

....

plugin.xmlyouclassname

```

<extra-core>

    <LocaleFinder class="com.fr.plugin.xxx.youclassname"/>

</extra-core>

```

1JAVAIinter.getLocText("Plugin-XXX")XX

2JSFR.i18nText("Plugin-XXX ")XX