# com.fr.schedule.extension.report.job.output.formula.extract.OutputFormulaExtractorProvider

1.

2.sheet report0~A1block0~A1

OutputFormulaProvider

OutputFormulaExtractorProvider

**OutputFormulaExtractorProvider.java**

```
package com.fr.schedule.extension.report.job.output.formula.extract;

import com.fr.schedule.base.bean.output.BaseOutputAction;
import com.fr.stable.fun.mark.Mutable;

import java.util.Map;
import java.util.regex.Pattern;

/**
 * @author Cloud.Liu
 * @version 10.0
 * Created by Cloud.Liu on 2019/9/2
 */
public interface OutputFormulaExtractorProvider<T extends BaseOutputAction> extends Mutable {

    String XML_TAG = "OutputFormulaExtractorProvider";

    int CURRENT_LEVEL = 1;

    /**
     * BaseOutputAction
     * @return BaseOutputAction
     */
    String getActionClassName();

    /**
     * BaseOutputActionpatternkeyFormulavaluemap
     * @param t BaseOutputAction
     * @param pattern
     * @param map map
     */
    void addFormulaToMap(T t, Pattern pattern, Map<String, Object> map);
}
```

| | | | |
|---|---|---|---|
| FR | 10.0 | | |
| BI | 5.1 | | |
| BI | 5.1.2 | | |
| BI | 5.1.3 | | |

**plugin.xml**

```
<extra-decision>
        <OutputFormulaExtractorProvider class="your class name"/>
</extra-decision>
```

**ReportCalculationJob.java**

```
package com.fr.schedule.extension.report.job.execute;
```

```
...

public abstract class ReportCalculationJob extends BaseCalculationJob {

    ...

    @Override
    public Pair<CalculationResult, List<String>> calculation() throws Exception {

        Reportlet reportlet = ReportHelper.generateReportlet(this.templatePath);
        List<String> attachFileList = new ArrayList<String>();
        if (recordList.isEmpty()) {
            throw new EmptyRecordListException("Empty record list, output action will not be executed");
        }
        for (Map<String, Object> parameterMap : this.recordList) {
            String sessionId = null;
            try {
                //session
                this.addUserParam(parameterMap);
                // fr_locale
                DecisionTemplateUtils.preDealWithFRLocale(parameterMap);
                addFormulaToParameterMap(ScheduleParameterUtils.POLY_BLOCK_FORMULA_REGEX, parameterMap);
                String actorName = ScheduleTaskShowTypeFactory.fromInteger(this.getScheduleTask().
getShowType()).getActorName();
                ReportSessionIDInfor sessionIDInfor = new ReportSessionIDInfor(parameterMap, this.templatePath,
ActorFactory.getActor(actorName));
                sessionIDInfor.updateTableDataSource();
                sessionId = SessionPoolManager.addSessionIDInfor(sessionIDInfor);
                parameterMap.put(SessionInfo.SESSIONID.toString(), sessionId);
                TemplateWorkBook wbTpl = this.getTemplateWorkBook(reportlet, parameterMap);
                if (wbTpl == null) {
                    return new Pair<>(CalculationResult.ERROR, attachFileList);
                }
                CalculationResult calculationResult = calculateAndGenerateAttachment(parameterMap, wbTpl,
attachFileList);
                if (CalculationResult.SUCCESS != calculationResult) {
                    return new Pair<>(calculationResult, attachFileList);
                }
            } finally {
                if (sessionId != null) {
                    SessionPoolManager.closeSession(sessionId);
                }
            }
        }
        return new Pair<>(CalculationResult.SUCCESS, attachFileList);
    }

    ...

    /**
     * regexFormulaparameterMap
     * @param regex
     * @param parameterMap Map
     */
    public void addFormulaToParameterMap(Pattern regex, Map<String, Object> parameterMap) {

        ScheduleOutput scheduleOutput = this.getScheduleTask().getScheduleOutput();

        //
        String fileName = scheduleOutput.getBaseName();
        ScheduleParameterUtils.addFormulaToMap(fileName, regex, parameterMap);

        //
        List<BaseOutputAction> actionList = scheduleOutput.getOutputActionList();
        for (BaseOutputAction action : actionList) {
            OutputFormulaExtractorProvider provider = OutputFormulaExtractBox.KEY.fromActionClassName(action.
getActionName());
            if (provider != null) {
                provider.addFormulaToMap(action, regex, parameterMap);
            }
```

```
        }
    }

    ...
}
```

**OutputFormulaExtractBox.java**

```
package com.fr.schedule.extension.report.job.output.formula.extract;

import com.fr.decision.ExtraDecisionClassManager;
import com.fr.event.Event;
import com.fr.event.EventDispatcher;
import com.fr.event.Listener;
import com.fr.plugin.context.PluginContext;
import com.fr.plugin.manage.PluginFilter;
import com.fr.plugin.observer.PluginEventType;
import com.fr.schedule.extension.report.job.output.formula.extract.impl.EmailOutputFormulaExtractor;
import com.fr.schedule.extension.report.job.output.formula.extract.impl.MobileMsgOutputFormulaExtractor;
import com.fr.schedule.extension.report.job.output.formula.extract.impl.SmsOutputFormulaExtractor;
import com.fr.schedule.extension.report.job.output.formula.extract.impl.SystemMsgOutputFormulaExtractor;
import com.fr.stable.Filter;

import java.util.Map;
import java.util.Set;
import java.util.concurrent.ConcurrentHashMap;

/**
 * @author Cloud.Liu
 * @version 10.0
 * Created by Cloud.Liu on 2019/9/2
 */
public class OutputFormulaExtractBox {

    public static final OutputFormulaExtractBox KEY = new OutputFormulaExtractBox();

    private Map<String, OutputFormulaExtractorProvider> map = new ConcurrentHashMap<>();

    private OutputFormulaExtractBox() {
        registerInternal();
        listenPlugin();
    }

    private void registerInternal() {
        register(new EmailOutputFormulaExtractor());
        register(new SystemMsgOutputFormulaExtractor());
        register(new MobileMsgOutputFormulaExtractor());
        register(new SmsOutputFormulaExtractor());
    }

    public void register(OutputFormulaExtractorProvider provider) {
        if (!map.containsKey(provider.getActionClassName())) {
            map.put(provider.getActionClassName(), provider);
        }
    }

    private void remove(OutputFormulaExtractorProvider provider) {
        if (provider != null) {
            map.remove(provider.getActionClassName());
        }
    }

    public OutputFormulaExtractorProvider fromActionClassName(String className) {
        return map.containsKey(className) ? map.get(className) : null;
    }

    private void listenPlugin() {
```

```
        Set<OutputFormulaExtractorProvider> providers = ExtraDecisionClassManager.getInstance().getArray
(OutputFormulaExtractorProvider.XML_TAG);
        for (OutputFormulaExtractorProvider provider : providers) {
            register(provider);
        }
        Filter<PluginContext> filter = new PluginFilter() {

            @Override
            public boolean accept(PluginContext pluginContext) {

                return pluginContext.contain(OutputFormulaExtractorProvider.XML_TAG);
            }
        };
        EventDispatcher.listen(PluginEventType.AfterRun, new Listener<PluginContext>() {

            @Override
            public void on(Event event, PluginContext param) {

                Set<OutputFormulaExtractorProvider> set = param.getRuntime().get(OutputFormulaExtractorProvider.
XML_TAG);
                for (OutputFormulaExtractorProvider provider : set) {
                    register(provider);
                }
            }
        }, filter);

        EventDispatcher.listen(PluginEventType.BeforeStop, new Listener<PluginContext>() {

            @Override
            public void on(Event event, PluginContext param) {

                Set<OutputFormulaExtractorProvider> set = param.getRuntime().get(OutputFormulaExtractorProvider.
XML_TAG);
                for (OutputFormulaExtractorProvider provider : set) {
                    remove(provider);
                }
            }
        }, filter);
    }
}
```

OutputFormulaExtractBoxOutputFormulaExtractorProviderBaseOutputAction

EmailOutputFormulaExtractorSystemMsgOutputFormulaExtractorMobileMsgOutputFormulaExtractorSmsOutputFormulaExtractor

addFormulaToMap

```
ScheduleParameterUtils.addFormulaToMap(outputAction.getSubject(), pattern, map);
ScheduleParameterUtils.addFormulaToMap(outputAction.getBodyContent(), pattern, map);
```

demodemo-output-formula-extractor-provider