com.fr.report.fun.ActorProvider

-
-
-
-
-
-
-
-
-

3ActorProviderH5

ActorProvider.java

```
package com.fr.report.fun;

import com.fr.plugin.injectable.SpecialLevel;
import com.fr.report.stable.fun.Actor;
import com.fr.stable.fun.Level;

/**
 * @author : richie
 * @since : 8.0
 */
public interface ActorProvider extends Level{

    String XML_TAG = SpecialLevel.ActorProvider.getTagName();
    int CURRENT_LEVEL = 1;

    Actor[] createActor();

}
```

Actor.java

```
package com.fr.report.stable.fun;

import com.fr.base.BaseFormula;
import com.fr.base.Style;
import com.fr.base.chart.BaseChartPainter;
import com.fr.base.present.Present;
import com.fr.calculate.cell.BoxCEProvider;
import com.fr.general.DeclareRecordType;
import com.fr.json.JSONArray;
import com.fr.json.JSONException;
import com.fr.json.JSONObject;
import com.fr.main.TemplateWorkBook;
import com.fr.main.workbook.ResultWorkBook;
import com.fr.page.PageSetChainProvider;
import com.fr.plugin.injectable.SpecialLevel;
import com.fr.regist.FunctionPoint;
import com.fr.report.cell.CellElement;
import com.fr.report.cell.cellattr.core.attribute.CellElementAttribute;
import com.fr.report.cell.cellattr.core.attribute.OptionalAttribute;
import com.fr.report.core.block.PolyWorkSheetExecutor;
```

```
import com.fr.report.core.cal.BoxFactory;
import com.fr.report.core.cal.SE;
import com.fr.report.core.sheet.SheetExecutor;
import com.fr.report.core.sheet.WorkBookExecutor;
import com.fr.report.poly.PolyWorkSheet;
import com.fr.report.poly.ResultChartBlock;
import com.fr.report.poly.ResultECBlock;
import com.fr.report.web.ToolBarManager;
import com.fr.report.worksheet.AbstractResECWorkSheet;
import com.fr.script.Calculator;
import com.fr.stable.fun.Level;
import com.fr.stable.script.ExTool;
import com.fr.stable.web.Repository;
import com.fr.web.cache.ReportCache;
import com.fr.web.cache.ReportEntry;
import com.fr.web.core.ReportSessionIDInfor;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.Map;

/**
 * Created with IntelliJ IDEA.
 * User: richie
 * Date: 14-1-24
 * Time: 5:26
 *
 */
public interface Actor extends Level, OperatorBuilder {

    String XML_TAG = SpecialLevel.Actor.getTagName();
    int CURRENT_LEVEL = 1;

    /**
     *
     *
     * @return
     */
    String description();

    /**
     *
     *
     * @return truefalse
     */
    boolean canCalculateOnDemand();

    /**
     *
     *
     * @return truefalse
     */
    boolean willPreCalculate();

    /**
     *
     *
     * @return
     */
    BoxFactory createBoxFactory();

    /**
     *
     *
     * @return
     */
    ResultECBlock createResultECBlock();

    /**
     *
```

```
     *
     * @param se
     * @return
     */
    AbstractResECWorkSheet createResultECWorkSheet(SE se);

    /**
     *
     *
     * @param sheet
     * @param maxCount
     * @param activePoolCount
     * @param cacheFirst
     */
    void cacheCellElement(AbstractResECWorkSheet sheet, int maxCount, int activePoolCount, boolean cacheFirst);

    /**
     *
     *
     * @return truefalse
     */
    boolean considerBuildRelation();

    /**
     *
     *
     * @param se
     */
    void release(SheetExecutor se);

    /**
     *
     *
     * @param em
     * @return
     */
    OptionalAttribute cloneOptionalAttribute(OptionalAttribute em);

    /**
     *
     *
     * @return
     */
    CellElementAttribute createHyperCellAttr();

    /**
     *
     *
     * @return UI
     */
    CellElementAttribute createCellGUIAttr();

    /**
     *
     *
     * @return
     */
    CellElementAttribute createWidgetAttr();

    /**
     *
     *
     * @param exTool
     * @param calculator
     * @param curFa
     * @param oriFa
     * @param boxCe
     */
    void buildRelation(ExTool exTool, Calculator calculator, BaseFormula curFa, BaseFormula oriFa,
BoxCEProvider boxCe);
```

```
/**
 *
 *
 * @return
 */
boolean isDynamicJavaScript();

/**
 *
 *
 * @return truefalse
 */
boolean canBeUseForSchedule();

/**
 *
 * (, )
 *
 * @param ce
 * @param present
 * @param presentValue
 * @date 2014-9-24-8:49:38
 */
void present(CellElement ce, Present present, Object presentValue);

/**
 *
 *
 * @param ce
 * @param present
 * @param presentStyle
 */
void present(CellElement ce, Present present, Style presentStyle);

/**
 *
 *
 * @return truefalse
 */
boolean hasWidget();


/**
 *
 *
 * @param tpl
 * @param parameterMap
 * @param actor
 * @return
 */
PolyWorkSheetExecutor createPolySequenceExecutor(PolyWorkSheet tpl, Map parameterMap, Actor actor);

/**
 *
 *
 * @return
 */
FunctionPoint getBookFUNC();

/**
 *
 *
 * @param parameterMap
 * @return
 */
ResultWorkBook createResultBook(Map parameterMap);

/**
 *
 *
 * @param workBook
```

```
     * @param parameterMap
     * @return
     */
    ResultWorkBook executeWorkBook(TemplateWorkBook workBook, Map parameterMap);

    /**
     *
     *
     * @param workBook
     * @param parameterMap
     * @param sheetIndex   sheet
     * @return
     */
    ResultWorkBook executeWorkBook(TemplateWorkBook workBook, Map parameterMap, int sheetIndex);


    /**
     * sheet
     *
     * @param workBook
     * @param parameterMap
     * @return sheet
     */
    WorkBookExecutor createWorkBookExecutor(TemplateWorkBook workBook, Map parameterMap);

    /**
     *
     *
     * @return
     */
    DeclareRecordType getRecordType();

    /**
     *
     *
     * @return truefalse
     */
    boolean shouldRecord();

    /**
     *
     *
     * @param tpl
     * @return
     */
    long updateCacheTime(TemplateWorkBook tpl);

    /**
     *
     *
     * @param tpl
     * @param reportEntry
     * @param parameterMap
     * @param useCache
     * @return
     */
    ReportCache createReportCache(TemplateWorkBook tpl,
                                  ReportEntry reportEntry,
                                  Map parameterMap,
                                  boolean useCache);

    /**
     *
     *
     * @param tpl
     * @param tplPath
     * @param reportCache
     * @param parameterMap
     * @param sheetIndex   sheet
     * @return
     */
```

```
ResultWorkBook getResultBookFromCacheIfNeed(TemplateWorkBook tpl,
                                            String tplPath,
                                            ReportCache reportCache,
                                            Map<String, Object> parameterMap,
                                            int sheetIndex);

/**
 *
 *
 * @return truefalse
 */
boolean shouldNotBeScale();

/**
 *
 *
 * @param req HTTP
 * @return truefalse
 */
boolean isPageByPage(HttpServletRequest req);

/**
 *
 *
 * @param wb
 * @return
 */
PageSetChainProvider getPageSet(ResultWorkBook wb);

/**
 *
 *
 * @param repo
 * @return
 */
ToolBarManager[] toolbarManagers(Repository repo);

/**
 *
 *
 * @param repository
 * @return
 * @throws JSONException
 */
JSONObject panelConfig(Repository repository) throws JSONException;

/**
 * JavaScript
 *
 * @return
 */
String panelType();

/**
 * js
 *
 * @return js
 */
String mainJavaScriptPath();

/**
 * Session
 *
 * @param repository Session
 * @return
 */
int getReportCountInRepo(Repository repository);

/**
 * Session
 *
```

```
     * @param repository Session
     * @return
     */
    int calculateCurrentSheetIndex(Repository repository);

    /**
     * block
     *
     * @param chartPainter painter
     */
    ResultChartBlock getChartBlock4Ploy(BaseChartPainter chartPainter);

    /**
     *
     *
     * @return truefalse
     */
    boolean supportPolyExecute();

    /**
     *
     *
     * @param req            HTTP
     * @param sessionIDInfor
     * @return
     */
    Map<String, Object> createContext4Tpl(HttpServletRequest req, ReportSessionIDInfor sessionIDInfor);

    /**
     * html
     *
     * @param req            HTTP
     * @param res            HTTP
     * @param map4Tpl
     * @param sessionIDInfor
     */
    void flushHtml(HttpServletRequest req, HttpServletResponse res, Map<String, Object> map4Tpl,
ReportSessionIDInfor sessionIDInfor) throws IOException;


    /**
     * web
     *
     * @param req            HTTP
     * @param sessionIDInfor
     * @return web
     * @throws JSONException e
     */
    JSONObject createReportWebAttr4Mobile(HttpServletRequest req, ReportSessionIDInfor sessionIDInfor) throws
JSONException;

    /**
     * sheetsheet
     *
     * @param repository
     * @return sheet
     */
    JSONArray processMultipleSheet(Repository repository);

    /**
     * session
     *
     * @param repo
     * @param req
     * @param c
     * @param map
     * @param isdebug __isdebug__
     */
    void dealWithSessionInfo(Repository repo, HttpServletRequest req, Calculator c, Map<String, Object> map,
boolean isdebug);
```

```
    /**
     *
     */
    int getScheduleShowType();
}
```

WorkBookExecutor.java

```
package com.fr.report.core.sheet;

import com.fr.main.workbook.ResultWorkBook;
import com.fr.report.elementcase.ResultElementCase;
import com.fr.report.report.ResultReport;
import com.fr.report.report.TemplateReport;
import com.fr.report.stable.fun.Actor;

import java.util.Map;

/**
 * Created by richie on 16/5/11.
 *
 */
public interface WorkBookExecutor {

        /**
         *
         * @param currentIndex
         * @param report
         * @return
         */
        ResultReport execute(int currentIndex, TemplateReport report);

        /**
         *
         * @return
         */
        ResultWorkBook execute();

        /**
         *
         * @return
         */
        ResultWorkBook result();

        /**
         *
         * @return
         */
        Actor getExeType();


        /**
         * sheet
         * @param index
         * @param resEC
         */
        void addResult(int index, ResultReport resEC);

        /**
         *
         * @param index
         * @return
         */
        ResultElementCase getResultByIndex(int index);

        /**
         *
         *
         * @param parameterMap map
         *
         * @return
         *
         */
        ResultWorkBook initResultBook(Map<String, Object> parameterMap);
}
```

FR	8.0		
FR	9.0		
FR	10.0		

plugin.xml

```
<extra-report>
        <ActorProvider class="your class name"/>
</extra-report>
```

ActorFactoryactorActorFactoryActorProvidermap

ActorFactory.java

```
package com.fr.stable;

import com.fr.general.GeneralContext;
import com.fr.general.web.ParameterConstants;
import com.fr.plugin.context.PluginContext;
import com.fr.plugin.manage.PluginFilter;
import com.fr.plugin.observer.PluginEvent;
import com.fr.plugin.observer.PluginEventListener;
import com.fr.report.ExtraReportClassManager;
import com.fr.report.fun.ActorProvider;
import com.fr.report.stable.fun.Actor;
import com.fr.web.utils.WebUtils;

import javax.servlet.http.HttpServletRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReadWriteLock;
import java.util.concurrent.locks.ReentrantReadWriteLock;

/**
 * Created with IntelliJ IDEA.
 * User: richie
 * Date: 14-2-14
 * Time: 3:27
 */
public class ActorFactory {

    private static Map<String, Actor> actorMap = new HashMap<String, Actor>();

    private static Map<Actor, String> typeMap = new HashMap<Actor, String>();

    private static Map<String, Actor> reportMap = new HashMap<String, Actor>();

    private static Map<String, Actor> extraMap = new HashMap<String, Actor>();

    private static final ReadWriteLock LOCK = new ReentrantReadWriteLock();

    private static final Lock READ = LOCK.readLock();

    private static final Lock WRITE = LOCK.writeLock();

    static {
```

```
    GeneralContext.listenPluginRunningChanged(new PluginEventListener() {

        @Override
        public void on(PluginEvent event) {

            refresh();
        }
    }, new PluginFilter() {

        @Override
        public boolean accept(PluginContext context) {

            return context.contain(Actor.XML_TAG) || context.contain(ActorProvider.XML_TAG);
        }
    });
    refresh();
}

private static void refresh() {

    refreshExtra();
    merge();
}

private static synchronized void merge() {

    WRITE.lock();
    try {
        actorMap.clear();
        typeMap.clear();
        actorMap.putAll(reportMap);
        actorMap.putAll(extraMap);
        for (Map.Entry<String, Actor> entry : actorMap.entrySet()) {
            typeMap.put(entry.getValue(), entry.getKey());
        }
    } finally {
        WRITE.unlock();
    }
}

private synchronized static void refreshExtra() {

    extraMap.clear();
    Set<Actor> actors = ExtraReportClassManager.getInstance().getActors();
    for (Actor actor : actors) {
        extraMap.put(actor.panelType(), actor);
    }
    Set<ActorProvider> providers = ExtraReportClassManager.getInstance().getActorProviders();
    Actor[] providedActors;
    for (ActorProvider provider : providers) {
        providedActors = provider.createActor();
        if (providedActors != null) {
            for (Actor actor : providedActors) {
                extraMap.put(actor.panelType(), actor);
            }
        }
    }
}

private ActorFactory() {

}

/**
 * typeactoractorFactory
 *
 * @param type
 * @param actor Actor
 */
public synchronized static void registerActor(String type, Actor actor) {
```

```
    reportMap.put(type, actor);
    merge();
}

/**
 * Actor
 *
 * @param type
 */
public static Actor getActor(String type) {

    Actor actor;
    READ.lock();
    try {
        actor = actorMap.get(type);
        if (actor == null) {
            actor = actorMap.get(ActorConstants.TYPE_PAGE);
        }
    } finally {
        READ.unlock();
    }
    return actor;
}

public static String getType(Actor actor) {

    READ.lock();
    try {
        String type = typeMap.get(actor);
        if (type != null) {
            return type;
        }
    } finally {
        READ.unlock();
    }
    return ActorConstants.TYPE_PAGE;

}

public static Actor getActor(String type, boolean isMobile, boolean isNoPage) {

    if (ActorConstants.TYPE_VIEW.equalsIgnoreCase(type) && isMobile) {
        type = ActorConstants.TYPE_PAGE;
    }
    if (ActorConstants.TYPE_PAGE.equalsIgnoreCase(type) && isNoPage) {
        type = ActorConstants.TYPE_NO_PAGE;
    }
    return getActor(type);
}

/**
 * requestActor
 *
 * @param req http
 */
public static Actor getActor(HttpServletRequest req) {
    return getActor(req, null);
}

/**
 * requestActor
 *
 * @param req http
 */
public static Actor getActor(HttpServletRequest req, Actor defaultActor) {
    String op = WebUtils.getHTTPRequestParameter(req, ParameterConstants.OP);
    //apiop, op=getSessionID
    if (StringUtils.isEmpty(op) || actorMap.get(op) == null) {
        //  defaultActor
        // op=fs_mainActorFactory.getActor(req)  PageActor
        if (defaultActor != null) {
```

```
                return defaultActor;
            }
            op = ActorConstants.TYPE_PAGE;
        }
        Actor actor;
        READ.lock();
        try {
            actor = actorMap.get(op);
            if (actor == null) {
                throw new RuntimeException("Not support op:" + op + "@" + ActorFactory.class.getName());
            }
        } finally {
            READ.unlock();
        }
        return actor;
    }
}
```

ActorProviderActor

ActorPageActorWriteActorViewActor

WorkBookExecutor createWorkBookExecutor(TemplateWorkBook workBook, Map parameterMap)

panelType()urlop

WorkBookExecutorResultWorkBook execute();

ActorProvider

demodemo-actor-provider

demo-dynamic-sheet

open-JSD-8210