(1)

com.fr.report.fun.ExportOperateProvider		
com.fr.report.fun.ExportExtensionProcessor		
com.fr.report.fun.FormatActionProvider	format	
com.fr.form.stable.FormExportProcessor		
com.fr.report.fun.ExcelExportAppProvider	excel	
com.fr.io.exporter.PDFExporterCreator	PDF	
com.fr.stable.fun.ExcelExportCellValueProvider	excel	
com.fr.report.fun.CommentExcelProcessor	excel2007sheet	
com.fr.design.fun.ToolbarItemProvider		
com.fr.report.fun.ExtensionButtonProvider		
com.fr.stable.fun.LocaleFinder		
JSCSS	JS/CSS	
web	web	

88

1.

2.web

3./

ExportService.java

```
package com.fr.web.core.reserve;

...

public class ExportService extends NoOPService {

    ...

    public String actionOP() {
        return "export";
    }

    /**
     *   hugh:
     *
     * @param req       http
     * @param res       http
     * @param sessionID ID
     * @throws Exception
     */
    public void process(HttpServletRequest req, HttpServletResponse res, String sessionID) throws Exception {
        ...
        dealWithExport(req, res, sessionID, false);
    }

    /**
     *
     *
     * @param req       http
     * @param res       http
     * @param sessionID ID
     * @param isEmbbed
     * @throws Exception
     */
    public static void dealWithExport(HttpServletRequest req,
                                      HttpServletResponse res, String sessionID, boolean isEmbbed) throws
Exception {
        ...

        ExportExtensionProcessor processor = getExportExtensionProcessor();
                //hugh: ExportExtensionProcessorfileName
        String fileName = processor.fileName(req, sessionIDInfor);

        String format = WebUtils.getHTTPRequestParameter(req, "format");
                //hugh: ExportExtensionProcessorcreateCollection
        ExportCollection exportCollection = processor.createCollection(req, res, sessionIDInfor, format,
fileName, isEmbbed);

        exportCollection.doExport(req, res, sessionIDInfor, format);

        ...
    }

    private static ExportExtensionProcessor getExportExtensionProcessor() {
                //hugh:DefaultExportExtension
                //ExportExtensionProcessor
                //DefaultExportExtensionDefaultExportExtension
                //
        ExportExtensionProcessor processor = ExtraReportClassManager.getInstance().getSingle
(ExportExtensionProcessor.MARK_TAG);
        if (processor == null) {
            processor = new DefaultExportExtension();
        }
        return processor;
    }

    ...

}
```

ExportExtensionProcessorDefaultExportExtension

DefaultExportExtension.java

```
package com.fr.web.core.reserve;

...

public class DefaultExportExtension extends AbstractExportExtension {

    public String fileName(HttpServletRequest req, TemplateSessionIDInfo sessionIDInfor) throws Exception {
        ...
        //hugh:ExportExtensionProcessorfileName
    }

    public ExportCollection createCollection(HttpServletRequest req, HttpServletResponse res,
                                             ReportSessionIDInfor sessionIDInfor, String format,
                                             String fileName, boolean isEmbed) throws Exception {
              //hugh:ExportOperateProvideroperateExportFactory
        Operate operate = ExportFactory.getOperate(format.toLowerCase());
        if (operate != null) {
            operate.setContent(req, res, sessionIDInfor, fileName, isEmbed);
                        //hughExportOperateProviderOperate
            return operate.newExportCollection(req, res, sessionIDInfor, fileName);
        }
        return ExportCollection.create();
    }
}
```

ExportFactory.java

```
package com.fr.web.core.reserve;

...

public class ExportFactory {

    private static final String FORMAT_EXCEL = "excel";
    ...
    private static final String FORMAT_PDF = "pdf";
    ...

    private final static Map<String, Operate> OPERATE_MAP = new HashMap<String, Operate>();

    static {
                //hugh:
        OPERATE_MAP.put(FORMAT_PDF, new DefaultOperate() {
                        ...
                        private AppExporter getPDFExporter(HttpServletRequest req, SessionProvider
sessionIDInfor) {
                ...
                //,PDFExporterCreator
                return PDFExporterFactory.getPDFExporter(isPDFPrint);
            }
        });

        OPERATE_MAP.put(FORMAT_EXCEL, new ExcelOperate());
        ...
                //hugh:OPERATE_MAP
        Set<ExportOperateProvider> providers = ExtraReportClassManager.getInstance().getArray
(ExportOperateProvider.MARK_STRING);
        for (ExportOperateProvider provider : providers) {
            OPERATE_MAP.put(provider.markType(), provider.operate());
        }
    }

    /**
     *
     *
     * @param format
     * @return
     */
    public static Operate getOperate(String format) {
        return OPERATE_MAP.get(format);
    }
}
```

ExportCollection#doExport(req, res, sessionIDInfor, format);

ExportCollectionAppExporter

```
ExportCollection#doExport(HttpServletRequest req, HttpServletResponse res, ReportSessionIDInfor sessionIDInfor,
String format)throws Exception {
        ...
        try {
            //
            LogUtils.exportAndLogRecordType(exporter, outputStream, new ReportRepositoryDeal(req,
sessionIDInfor), recordType);
        } catch (Exception e) {
            ...
        }
        ...
    }
}

LogUtils.exportAndLogRecordType(AppExporter exporter, OutputStream out,ReportRepositoryDeal repo, RecordType
exportType) throws Exception {
        ...
        exportAndLogRecordType(exporter, out, repo, exportType, sessionIDInfor,
                sessionIDInfor.getRelativePath(), sessionIDInfor.getFitBook2Show(),
                new PageSetCreator() {
                    public PageSetProvider createPageSet() {
                        return sessionIDInfor.getPrintPreviewPageSet4Traversing();
                    }
                });
}

LogUtils.exportAndLogRecordType(AppExporter exporter, OutputStream out, ReportRepositoryDeal repo, RecordType
exportType, SessionProvider sessionIDInfor, String bookPath,
                                            ResultWorkBook book, PageSetCreator pageSet) {
        ...
        try {
            ...
            export(exporter, out, repo, sessionID, bookPath, book, pageSet, sessionManager, sheets, exportType);
            ...
        } catch (Exception e) {
            throw SessionLocalManager.createLogPackedException(e);
        }
}

LogUtils.export(AppExporter exporter, OutputStream out, ReportRepositoryDeal repo, String sessionID, String
bookPath,
                               ResultWorkBook book, PageSetCreator pageSet, ExportSessionManager
sessionManager, int[] sheets, RecordType exportType) throws Exception {
        ...
        try {
                        //hugh:
            exporter.export(out, book, pageSet, repo, sheets);
        } finally {
            sessionManager.removeExportSession(sessionID, exportType.getTypeString());
        }
}
```

AppExporterOperateExportCollectionexcel

ExcelOperate.java

```
package com.fr.web.core.reserve;

...

public class ExcelOperate extends DefaultOperate {

    ...

    public void setContent(HttpServletRequest req, HttpServletResponse res, SessionProvider sessionIDInfor,
String fileName, boolean isEmbed) {
        //hugh:
    }


    public ExportCollection newExportCollection(HttpServletRequest req, HttpServletResponse res,
SessionProvider sessionIDInfor, String fileName) {
        return createExcelExportCollection(req, res, sessionIDInfor, fileName);
    }


    ...

    public ExportCollection createExcelExportCollection(HttpServletRequest req, HttpServletResponse res,
                                                        SessionProvider sessionIDInfor, String fileName) {

        ExcelExportType exportType = createExcelExportType(req, sessionIDInfor);

                //hugh:excelExcelExportAppProvider#newLargeDataExportCollection
        if (ExportConstants.TYPE_LARGEDATA_PAGE.equalsIgnoreCase(exportType.getExportType())) {
            Set<ExcelExportAppProvider> providers = ExtraReportClassManager.getInstance().getArray
(ExcelExportAppProvider.MARK_STRING);
            for (ExcelExportAppProvider provider : providers) {
                if (provider.exportType().equalsIgnoreCase(exportType.getExportType())) {
                    ExportCollection collection = provider.newLargeDataExportCollection(req, res,
sessionIDInfor, fileName, exportType);
                    if (collection != null) {
                        return collection;
                    }
                }
            }
            return createLargeDataExportCollection(req, res, sessionIDInfor, fileName, exportType);
        } else {
            ...
            AppExporter<Boolean> exporter = createExcelExporter(collection, exportType, sessionIDInfor);
            ...
            return collection;
        }
    }

    public AppExporter<Boolean> createExcelExporter(ExportCollection collection, ExcelExportType exportType,
SessionProvider sessionIDInfor) {
        AppExporter<Boolean> exporter;
                //hugh:excel,ExcelExportAppProvider#newAppExporter
        Set<ExcelExportAppProvider> providers = ExtraReportClassManager.getInstance().getArray
(ExcelExportAppProvider.MARK_STRING);
        for (ExcelExportAppProvider provider : providers) {
            if (provider.exportType().equalsIgnoreCase(exportType.getExportType())) {
                return provider.newAppExporter(collection, exportType, sessionIDInfor);
            }
        }
        //hugh:excel,ExcelExportAppProviderdemo
                ...
        return exporter;
    }
}
```

AbstractExcelExporter.java

```
package com.fr.io.exporter;

...

public abstract class AbstractExcelExporter<T> extends AbstractAppExporter<T> {
        ...

        public Object evalCellValue(CellElement cellElement, boolean exHiddenRow, boolean exHiddenColumn,
                        List hssfCellList, POICellAction hssfCell, Calculator cal, Style style, List
hssfCellFormulaList,
                        CellGUIAttr cellGUIAttr, DynamicUnitList rowHeightList, DynamicUnitList
columnWidthList, int column,
                        int row, int columnSpan, int rowSpan, POIWorkbookAction wb) {
                //hugh:ExcelExportCellValueProviderexcel
                //ExcelExporterExcelExportCellValueProvider
                //ExcelExporterAbstractExcelExporterexcelAbstractExcelExporter
                ...
                return getValueFromExcelExportCellValueProvider(cellElement, cal, value);
        }

        //
        private Object getValueFromExcelExportCellValueProvider(CellElement cellElement, Calculator cal, Object
value) {
                Set<ExcelExportCellValueProvider> providers = ExtraClassManager.getInstance().getArray
(ExcelExportCellValueProvider.XML_TAG);
                for (ExcelExportCellValueProvider provider : providers) {
                        value = provider.getCellValue(cellElement, value, cal);
                }
                return value;
        }

        ...

}
```

ExcelExporterexcel2007StreamExcel2007Exporter

ExcelExporter.java

```
package com.fr.io.exporter;

...
public class ExcelExporter extends AbstractExcelExporter<Boolean> {

        ...

        public void export(OutputStream out, ResultWorkBook workBook, PageSetCreator pageSet,
ReportRepositoryDeal repo, int[] sheets) throws Exception {
                export(out, workBook, true, sheets);
        }


        public void export(OutputStream out, ResultWorkBook book) throws Exception {
                export(out, book, false);
        }


    public void export(OutputStream out, ResultWorkBook book, boolean reUse) throws Exception {
        export(out, book, reUse, null);
    }


    public void export(OutputStream out, ResultWorkBook book, boolean reUse, int[] sheets) throws Exception {
        ...
        ResultWorkBook book2Export = removeUselessSheet(book, sheets, book);
                try {
                        if (checkExcelExportVersion()) {
                                PerformanceManager.getRuntimeMonitor().setCurrentSessionStatus(ReportStatus.
EXPORT_EXCEL_2007);
                                exportFor2007(out, book2Export);
                                return;
                        }
                        PerformanceManager.getRuntimeMonitor().setCurrentSessionStatus(ReportStatus.
EXPORT_EXCEL_2003);
                        exportFor2003(out, book2Export, reUse, false);
                }finally {
                        PerformanceManager.getRuntimeMonitor().setCurrentSessionStatus(ReportStatus.COMPLETE);
                }
        }

        protected void exportFor2007(OutputStream out, ResultWorkBook book) throws Exception {
                //hugh:2007ReportUtils.getPaperSettingListFromWorkBookPageExcel2007Exporter
PageExcel2007Exporter#export(OutputStream out, ResultWorkBook book)
                //PageExcel2007ExporterStreamExcel2007Exporter2007sheet
                getExporterFor2007(
                        ReportUtils.getPaperSettingListFromWorkBook(book)
                        ).export(out, book);
                return;
        }

    protected void exportFor2003(OutputStream out, ResultWorkBook book, boolean reUse, boolean layerEngine)
throws Exception {
            ...
                //hugh:2003
        exportBook(book, new HssfWorkbookWrapper(hssfWorkbook), hssfCellList,
                hssfCellFormulaList, reportList, reUse);

        ...
    }
        ...

}
```

StreamExcel2007Exporter.java

```
package com.fr.io.exporter.excel.stream;
...
public class StreamExcel2007Exporter<T> extends AbstractExcelExporter<T> {

        ...

        /**
         * Export report,Execute.
         *
         * @param report
         * @param exportAttr
         * @param sheetName sheet
         * @param wb poiexcelWorkbook
         * @param xssfCellList poiexcelList
         * @param xssfCellFormulaList poiexcelList
         * @param reportIndex Index
         */
        protected void innerExportReport(Report report, ReportExportAttr exportAttr, String sheetName,
                                                                                SXSSFWorkbook wb, List xssfCellList,
List xssfCellFormulaList, int reportIndex) throws Exception {
                ExcelExportAttr eea = exportAttr == null ? new ExcelExportAttr() : exportAttr.
getExcelExportAttr();
                //hugh
                StreamExcelReportExporter exporter = new StreamExcelReportExporter((ElementCase)report,
sheetName, wb, xssfCellList, xssfCellFormulaList, eea, reportIndex, paperSettingList, columnRowPostileMaps,
this);
                exporter.export();
        }
        ...
}
```

StreamExcelReportExporter.java

```
package com.fr.io.exporter.excel.stream;

...
public class StreamExcelReportExporter {
    ...

    public void export() throws Exception {
                //hugh:StreamExcelReportExporter
        ...
        CommentExcelProcessor processor = ExtraReportClassManager.getInstance().getSingle(CommentExcelProcessor.
MARK_STRING);
        if (processor != null) {
            if (patr == null) {
                patr = xssfSheet.createDrawingPatriarch();
            }
                        //hugh:excel2007sheet
            processor.addCellComment(this.xssfSheet, this.report, patr);
        }
    }
        ...

}
```

cpt86FormExportProcessorFormatActionProvider

(format)

ReportletDealWith.java

```
package com.fr.web.core.reserve;

...

public class ReportletDealWith {

    ...

    /**
     *
     *
     * @param req    http
     * @param res    http
     * @param weblet
     * @throws Exception e
     */
    public static void dealWithReportlet(HttpServletRequest req, HttpServletResponse res, Weblet weblet) throws
Exception {
        ...
                //hugh:viewlet=xxx.cpt&format=xxx
        if (WebUtils.getHTTPRequestParameter(req, "format") != null) {
            turnToExportWithSessionKept(req, res, sessionID);
        } else {
            ...
        }
    }


    public static void turnToExportWithSessionKept(HttpServletRequest req, HttpServletResponse res, String
sessionID) throws Exception {
        ExportSessionKeeper.getInstance().keepAlive(sessionID);
        turnToExport(req, res, sessionID);
        ExportSessionKeeper.getInstance().close(sessionID);
    }


    /**
     *
     *
     * @param req       request
     * @param res       response
     * @param sessionID session id
     * @throws Exception e
     */
    public static void turnToExport(HttpServletRequest req, HttpServletResponse res, String sessionID) throws
Exception {
        // ,PDF, PDF_Activex, Excel..
        String format = WebUtils.getHTTPRequestParameter(req, "format");
                //FormatActionFactoryFormatActionProviderFormatActionProvider
        FormatActionProvider requestProcessor = FormatActionFactory.getReqProcessor(format);
        if (requestProcessor != null) {
            requestProcessor.doAction(req, res);
        } else {
            String embedParameter = WebUtils.getHTTPRequestParameter(req, ParameterConstants.EXPORT_PDF_EMBED);
            boolean embed = "true".equals(embedParameter); // true,
                        //hughFormatActionProviderExportService
                        //FormatActionProvider
            ExportService.dealWithExport(req, res, sessionID, embed);
        }
    }
        ...
}
```

FormExportProcessor

4.DesignReportExportTypeIDEAPDFPDFExporterCreatorexcelCommentExcelProcessorExcelExportCellValueProvider

5.

FormatActionProviderExportExtensionProcessorExportOperateProvider cptFormExportProcessor

FormatActionProviderExportExtensionProcessorExportExtensionProcessorExportOperateProvider

cpt

1.ProviderProcessor

2.

3.FormatActionProviderExportExtensionProcessorExportOperateProvider

4.FormExportProcessor

5.ExportOperateProviderExportOperateProvider

UI