Data Extraction: How to Get Your Data Out of Apache OFBiz

Gil Portenseigne - Néréide

Introduction

- Who am i ?
- User needs ?
- OFBiz tools ...
- Future, an idea of a new concept ?

What a user need ?

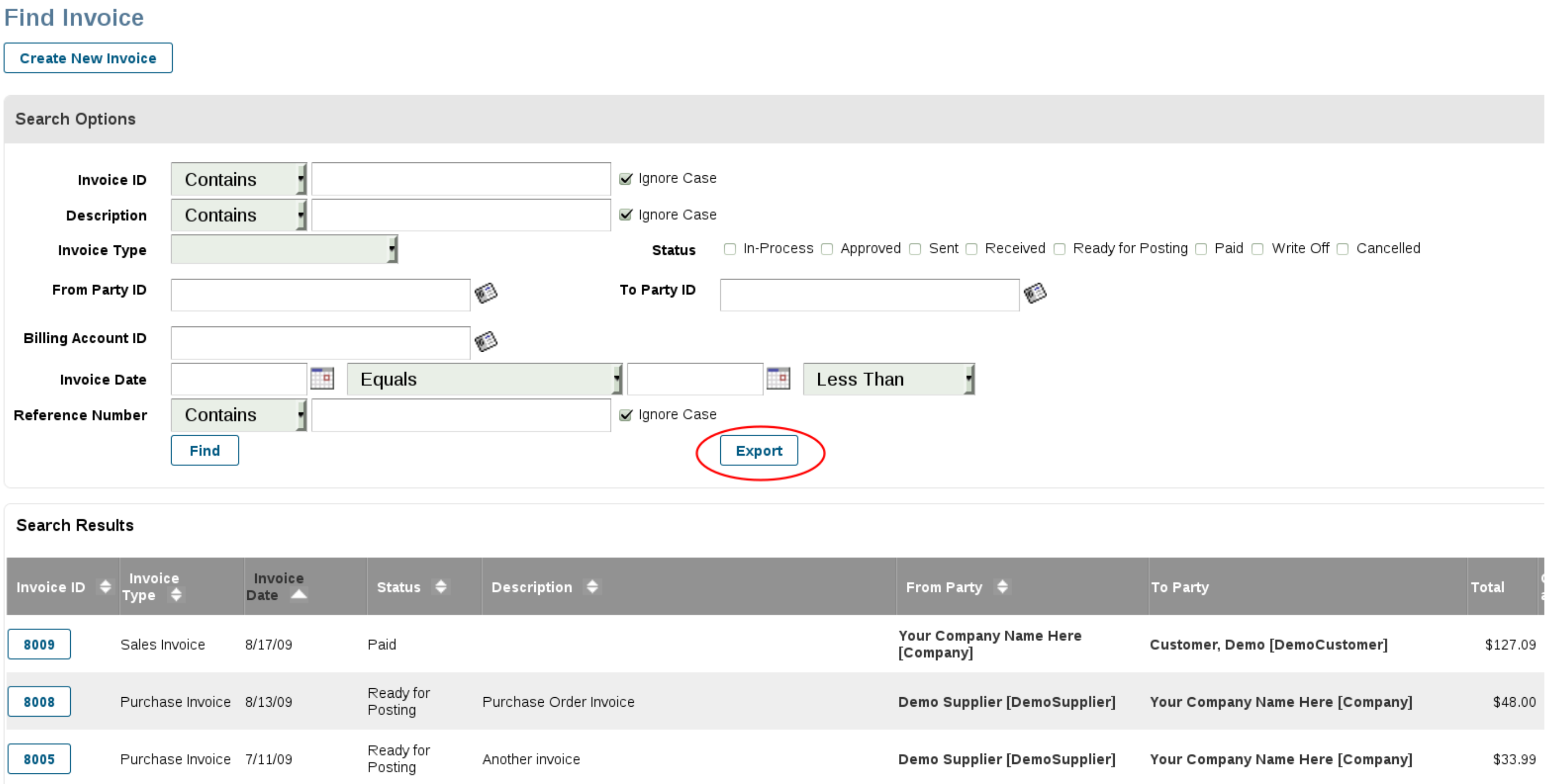
- Apache OFBiz : ERP !
  - Data manipulations to process order, invoicing...
- Data extraction for
  - Exchanges with other parties
  - Reports and statistics
- Won't speak about mass data extraction
  - Look for `DataFile` system in **OFBiz**

Daily data

- Basic needs for extracting daily use data !
  - Extract the list of all unpaid invoices for a customer ?
- Users globally start a process by searching an object in the database
  - Daily use of find/list wigdet
- OFBiz offers csv/xls handlers

Export principle :

Export into basic spreadsheet of list widgets :



Find Invoice

Create New Invoice

Search Options

Invoice ID	Contains	☑ Ignore Case	
Description	Contains	☑ Ignore Case	
Invoice Type		Status	□ In-Process □ Approved □ Sent □ Received □ Ready for Posting □ Paid □ Write Off □ Cancelled
From Party ID		To Party ID	
Billing Account ID			
Invoice Date	Equals	Less Than	
Reference Number	Contains	☑ Ignore Case	

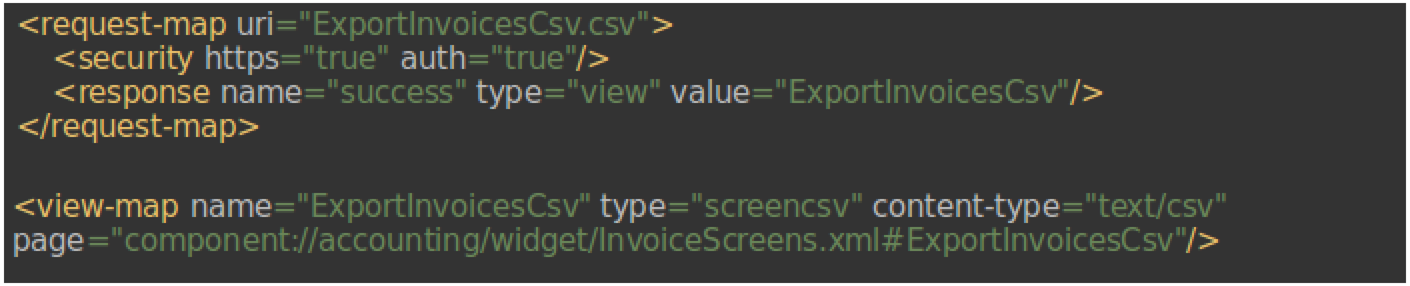
Find Export

Search Results

Invoice ID	Invoice Type	Invoice Date	Status	Description	From Party	To Party	Total
8009	Sales Invoice	8/17/09	Paid		Your Company Name Here [Company]	Customer, Demo [DemoCustomer]	$127.09
8008	Purchase Invoice	8/13/09	Ready for Posting	Purchase Order Invoice	Demo Supplier [DemoSupplier]	Your Company Name Here [Company]	$48.00
8005	Purchase Invoice	7/11/09	Ready for Posting	Another invoice	Demo Supplier [DemoSupplier]	Your Company Name Here [Company]	$33.99

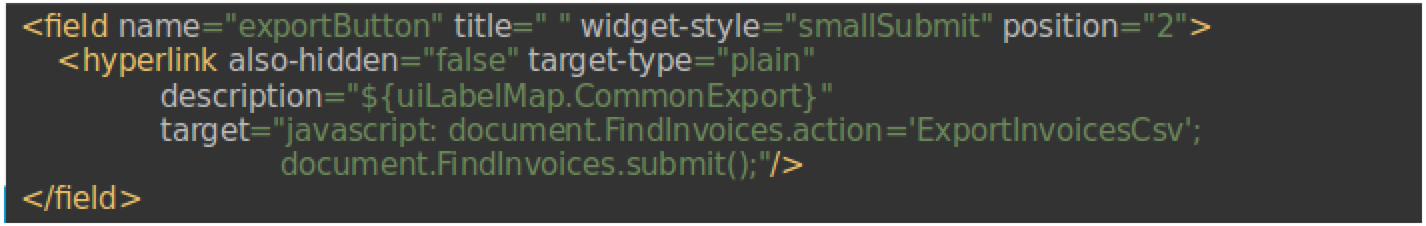
Implementation

Define request and view :



```
<request-map uri="ExportInvoicesCsv.csv">
   <security https="true" auth="true"/>
   <response name="success" type="view" value="ExportInvoicesCsv"/>
</request-map>

<view-map name="ExportInvoicesCsv" type="screencsv" content-type="text/csv"
page="component://accounting/widget/InvoiceScreens.xml#ExportInvoicesCsv"/>
```

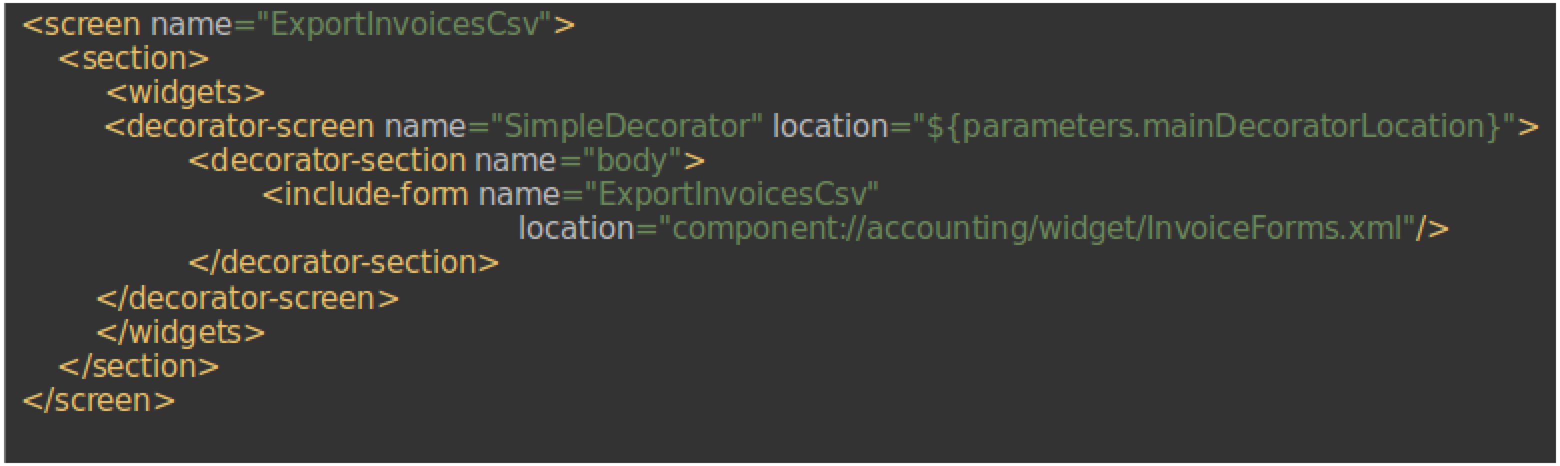
Add link to call the request :



```
<field name="exportButton" title=" " widget-style="smallSubmit" position="2">
   <hyperlink also-hidden="false" target-type="plain"
            description="${uiLabelMap.CommonExport}"
            target="javascript: document.FindInvoices.action='ExportInvoicesCsv';
                      document.FindInvoices.submit();"/>
</field>
```

Implementation

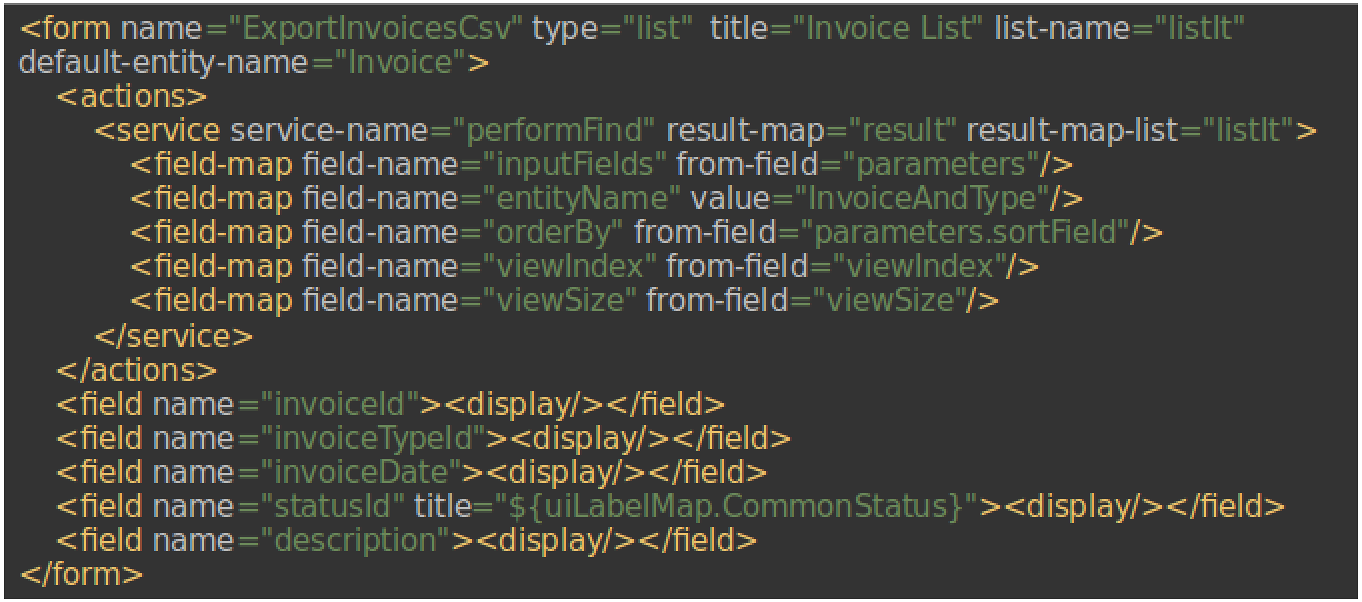
Define Basic Screen



```
<screen name="ExportInvoicesCsv">
   <section>
       <widgets>
       <decorator-screen name="SimpleDecorator" location="${parameters.mainDecoratorLocation}">
              <decorator-section name="body">
                    <include-form name="ExportInvoicesCsv"
                                         location="component://accounting/widget/InvoiceForms.xml"/>
              </decorator-section>
      </decorator-screen>
      </widgets>
   </section>
</screen>
```

Implementation

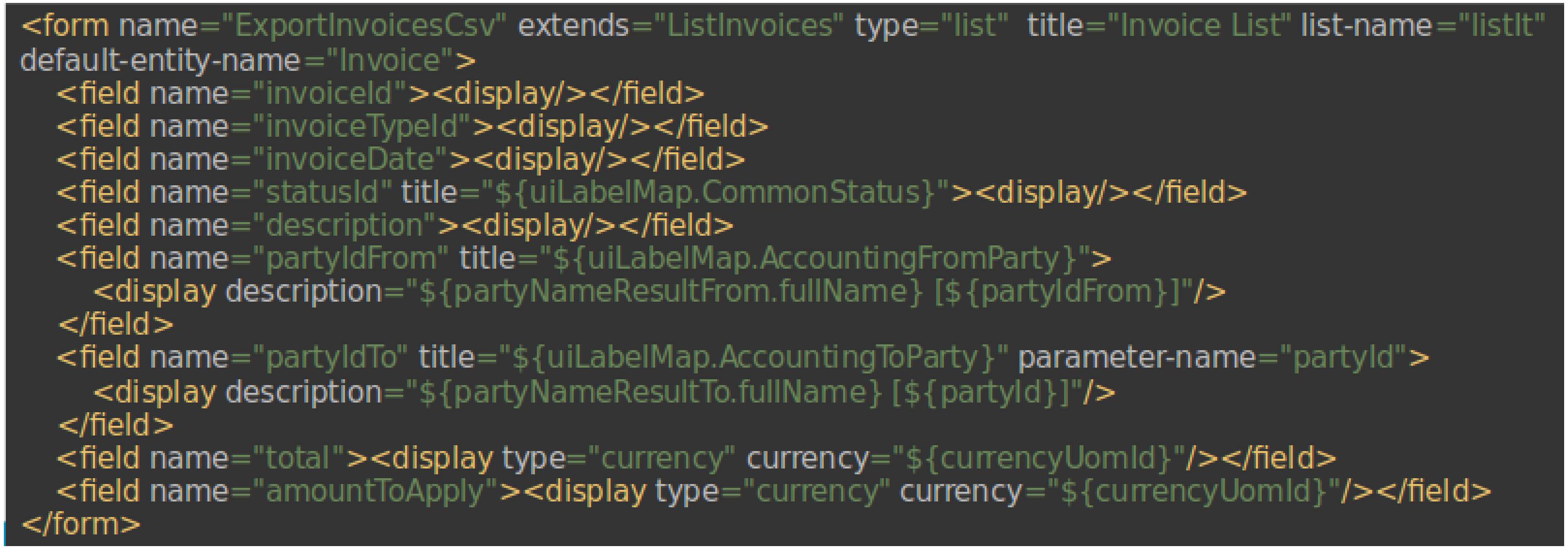
Create your Own list form



```
<form name="ExportInvoicesCsv" type="list"  title="Invoice List" list-name="listIt"
default-entity-name="Invoice">
   <actions>
      <service service-name="performFind" result-map="result" result-map-list="listIt">
         <field-map field-name="inputFields" from-field="parameters"/>
         <field-map field-name="entityName" value="InvoiceAndType"/>
         <field-map field-name="orderBy" from-field="parameters.sortField"/>
         <field-map field-name="viewIndex" from-field="viewIndex"/>
         <field-map field-name="viewSize" from-field="viewSize"/>
      </service>
   </actions>
   <field name="invoiceId"><display/></field>
   <field name="invoiceTypeId"><display/></field>
   <field name="invoiceDate"><display/></field>
   <field name="statusId" title="${uiLabelMap.CommonStatus}"><display/></field>
   <field name="description"><display/></field>
</form>
```

Implementation

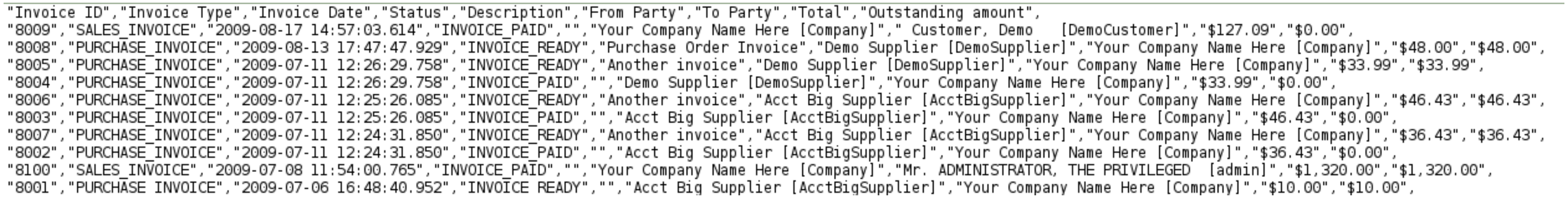
Better extend the existing form !



```
<form name="ExportInvoicesCsv" extends="ListInvoices" type="list"  title="Invoice List" list-name="listIt"
default-entity-name="Invoice">
   <field name="invoiceId"><display/></field>
   <field name="invoiceTypeId"><display/></field>
   <field name="invoiceDate"><display/></field>
   <field name="statusId" title="${uiLabelMap.CommonStatus}"><display/></field>
   <field name="description"><display/></field>
   <field name="partyIdFrom" title="${uiLabelMap.AccountingFromParty}">
      <display description="${partyNameResultFrom.fullName} [${partyIdFrom}]"/>
   </field>
   <field name="partyIdTo" title="${uiLabelMap.AccountingToParty}" parameter-name="partyId">
      <display description="${partyNameResultTo.fullName} [${partyId}]"/>
   </field>
   <field name="total"><display type="currency" currency="${currencyUomId}"/></field>
   <field name="amountToApply"><display type="currency" currency="${currencyUomId}"/></field>
</form>
```

Result

Data comma separated : (widget.properties)



```
"Invoice ID","Invoice Type","Invoice Date","Status","Description","From Party","To Party","Total","Outstanding amount",
"8009","SALES_INVOICE","2009-08-17 14:57:03.614","INVOICE_PAID","","Your Company Name Here [Company]"," Customer, Demo   [DemoCustomer]","$127.09","$0.00",
"8008","PURCHASE_INVOICE","2009-08-13 17:47:47.929","INVOICE_READY","Purchase Order Invoice","Demo Supplier [DemoSupplier]","Your Company Name Here [Company]","$48.00","$48.00",
"8005","PURCHASE_INVOICE","2009-07-11 12:26:29.758","INVOICE_READY","Another invoice","Demo Supplier [DemoSupplier]","Your Company Name Here [Company]","$33.99","$33.99",
"8004","PURCHASE_INVOICE","2009-07-11 12:26:29.758","INVOICE_PAID","","Demo Supplier [DemoSupplier]","Your Company Name Here [Company]","$33.99","$0.00",
"8006","PURCHASE_INVOICE","2009-07-11 12:25:26.085","INVOICE_READY","Another invoice","Acct Big Supplier [AcctBigSupplier]","Your Company Name Here [Company]","$46.43","$46.43",
"8003","PURCHASE_INVOICE","2009-07-11 12:25:26.085","INVOICE_PAID","","Acct Big Supplier [AcctBigSupplier]","Your Company Name Here [Company]","$46.43","$0.00",
"8007","PURCHASE_INVOICE","2009-07-11 12:24:31.850","INVOICE_READY","Another invoice","Acct Big Supplier [AcctBigSupplier]","Your Company Name Here [Company]","$36.43","$36.43",
"8002","PURCHASE_INVOICE","2009-07-11 12:24:31.850","INVOICE_PAID","","Acct Big Supplier [AcctBigSupplier]","Your Company Name Here [Company]","$36.43","$0.00",
"8100","SALES_INVOICE","2009-07-08 11:54:00.765","INVOICE_PAID","","Your Company Name Here [Company]","Mr. ADMINISTRATOR, THE PRIVILEGED  [admin]","$1,320.00","$1,320.00",
"8001","PURCHASE_INVOICE","2009-07-06 16:48:40.952","INVOICE_READY","","Acct Big Supplier [AcctBigSupplier]","Your Company Name Here [Company]","$10.00","$10.00",
```

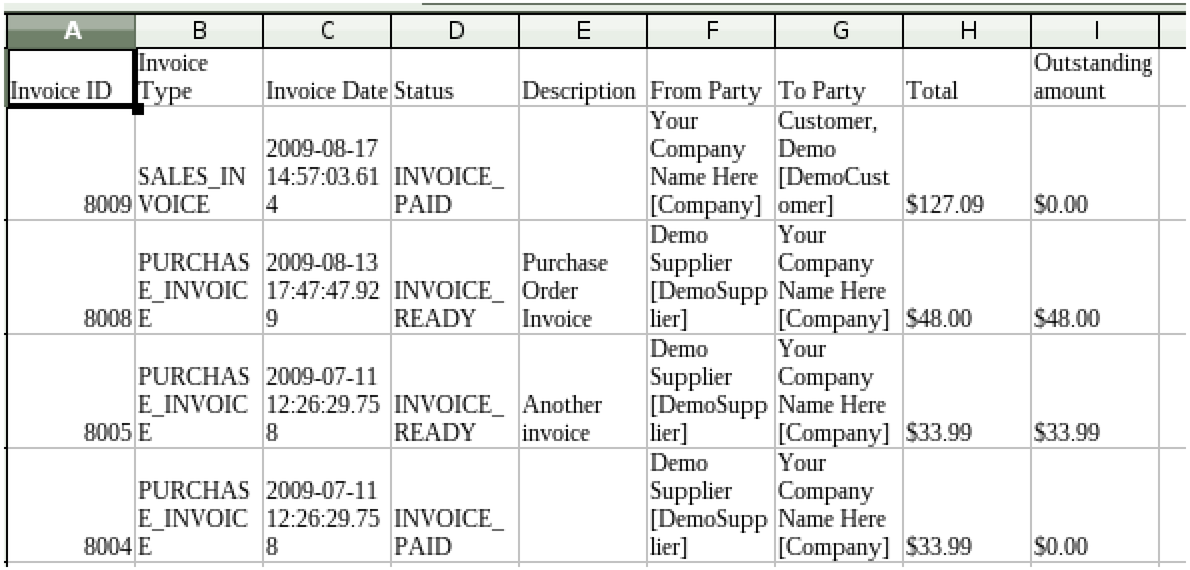
Pros & Cons :

+ Standard but very Basic, quick implementation

− Limited in data quantity (pagination)

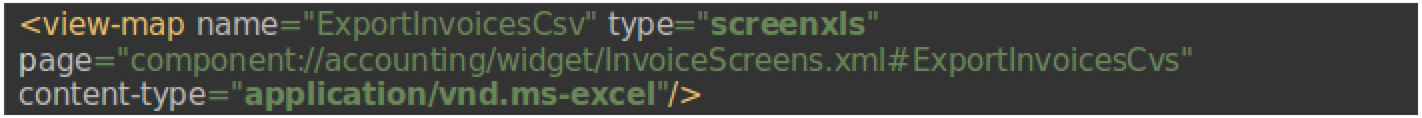
Alternative

Xls handler :



A	B	C	D	E	F	G	H	I
Invoice ID	Invoice Type	Invoice Date	Status	Description	From Party	To Party	Total	Outstanding amount
8009	SALES_INVOICE	2009-08-17 14:57:03.614	INVOICE_PAID		Your Company Name Here [Company]	Customer, Demo [DemoCustomer]	$127.09	$0.00
8008	PURCHASE_INVOICE	2009-08-13 17:47:47.929	INVOICE_READY	Purchase Order Invoice	Demo Supplier [DemoSupplier]	Your Company Name Here [Company]	$48.00	$48.00
8005	PURCHASE_INVOICE	2009-07-11 12:26:29.758	INVOICE_READY	Another invoice	Demo Supplier [DemoSupplier]	Your Company Name Here [Company]	$33.99	$33.99
8004	PURCHASE_INVOICE	2009-07-11 12:26:29.758	INVOICE_PAID		Demo Supplier [DemoSupplier]	Your Company Name Here [Company]	$33.99	$0.00

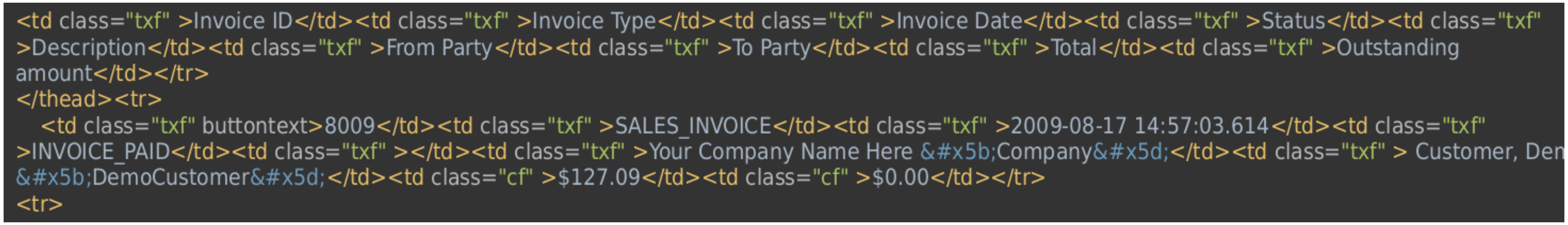
Same as csv with :



```
<view-map name="ExportInvoicesCsv" type="screenxls"
page="component://accounting/widget/InvoiceScreens.xml#ExportInvoicesCvs"
content-type="application/vnd.ms-excel"/>
```

How that works ?

Html generation with css :



```
<td class="txf" >Invoice ID</td><td class="txf" >Invoice Type</td><td class="txf" >Invoice Date</td><td class="txf" >Status</td><td class="txf"
>Description</td><td class="txf" >From Party</td><td class="txf" >To Party</td><td class="txf" >Total</td><td class="txf" >Outstanding
amount</td></tr>
</thead><tr>
   <td class="txf" buttontext>8009</td><td class="txf" >SALES_INVOICE</td><td class="txf" >2009-08-17 14:57:03.614</td><td class="txf"
>INVOICE_PAID</td><td class="txf" ></td><td class="txf" >Your Company Name Here &#x5b;Company&#x5d;</td><td class="txf" > Customer, Den
&#x5b;DemoCustomer&#x5d;</td><td class="cf" >$127.09</td><td class="cf" >$0.00</td></tr>
<tr>
```

Css Customizations :

framework/common/template/includes/Simple.xls.ftl

XLS extraction

Pro and cons :

+ Use of style and format

+ Customizable

− Multi language (Fr/En) data format

− Limited in data quantity (pagination)

Real Usage Purpose :

- Small data list extraction
- XLS for more advanced styles and Microsoft Office Users

What a user need ?

Reports and statistics

- To follow company activity and success
- To detect flaws and fix them

➡ Aggregated Data Extraction

Agregated data

These are specific needs, not existing widget screen

- Developper designed
  - POI
  - XSL/FO
- Dev/User designed
  - Birt

Apache POI

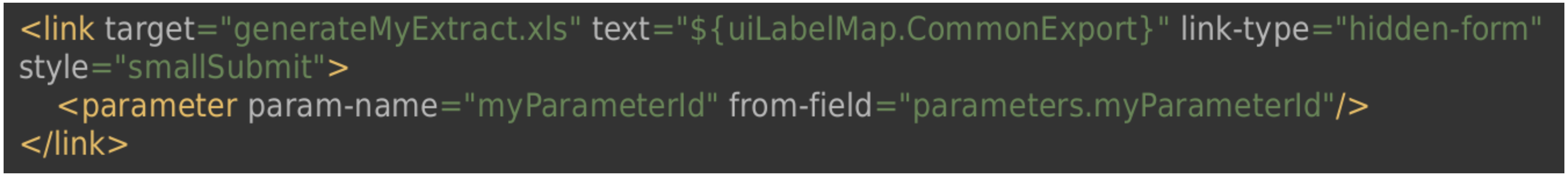
Advanced Spreadsheet design

- org.apache.poi Library to manipulate Microsoft Document
  - Xls parsing for importing data (productImportFromSpreadsheet)
  - Xls creation for exporting data
- Implementation/design via developer only
  - Java
  - Apache Groovy

Apache POI

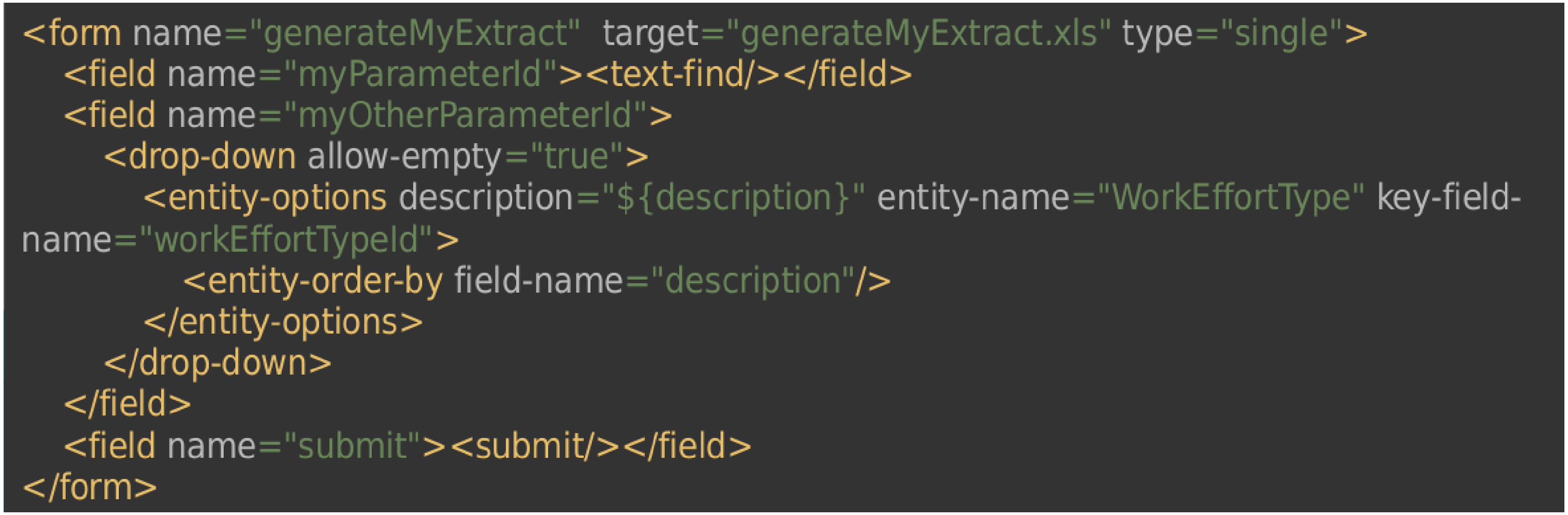
Implementation Basics in OFBiz

Request call via link :



```
<link target="generateMyExtract.xls" text="${uiLabelMap.CommonExport}" link-type="hidden-form"
style="smallSubmit">
   <parameter param-name="myParameterId" from-field="parameters.myParameterId"/>
</link>
```

Or form :

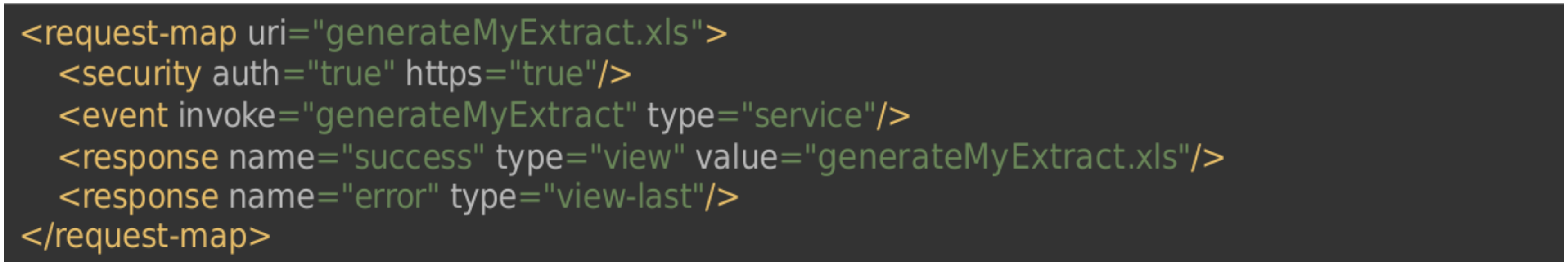


```
<form name="generateMyExtract"  target="generateMyExtract.xls" type="single">
    <field name="myParameterId"><text-find/></field>
    <field name="myOtherParameterId">
        <drop-down allow-empty="true">
            <entity-options description="${description}" entity-name="WorkEffortType" key-field-
name="workEffortTypeId">
                <entity-order-by field-name="description"/>
            </entity-options>
        </drop-down>
    </field>
    <field name="submit"><submit/></field>
</form>
```

Apache POI

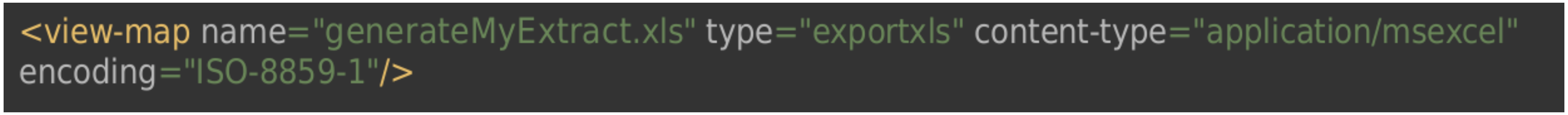
Implementation Basics in OFBiz

Request definition :



```
<request-map uri="generateMyExtract.xls">
   <security auth="true" https="true"/>
   <event invoke="generateMyExtract" type="service"/>
   <response name="success" type="view" value="generateMyExtract.xls"/>
   <response name="error" type="view-last"/>
</request-map>
```

View definition :

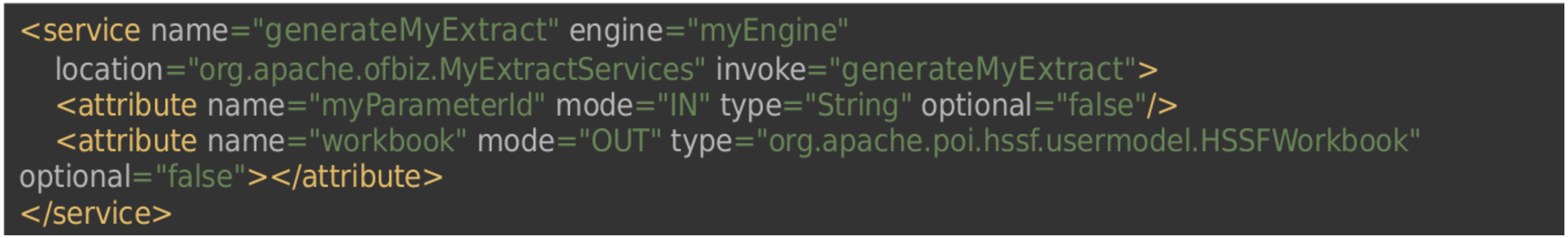


```
<view-map name="generateMyExtract.xls" type="exportxls" content-type="application/msexcel"
encoding="ISO-8859-1"/>
```

Apache POI

Implementation Basics in OFBiz

Service definition



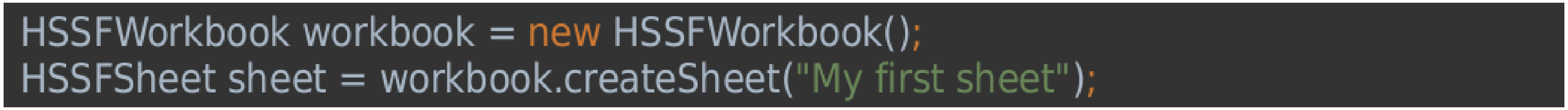
```
<service name="generateMyExtract" engine="myEngine"
   location="org.apache.ofbiz.MyExtractServices" invoke="generateMyExtract">
   <attribute name="myParameterId" mode="IN" type="String" optional="false"/>
   <attribute name="workbook" mode="OUT" type="org.apache.poi.hssf.usermodel.HSSFWorkbook"
optional="false"></attribute>
</service>
```

Engine available :

- Java
- Apache Groovy

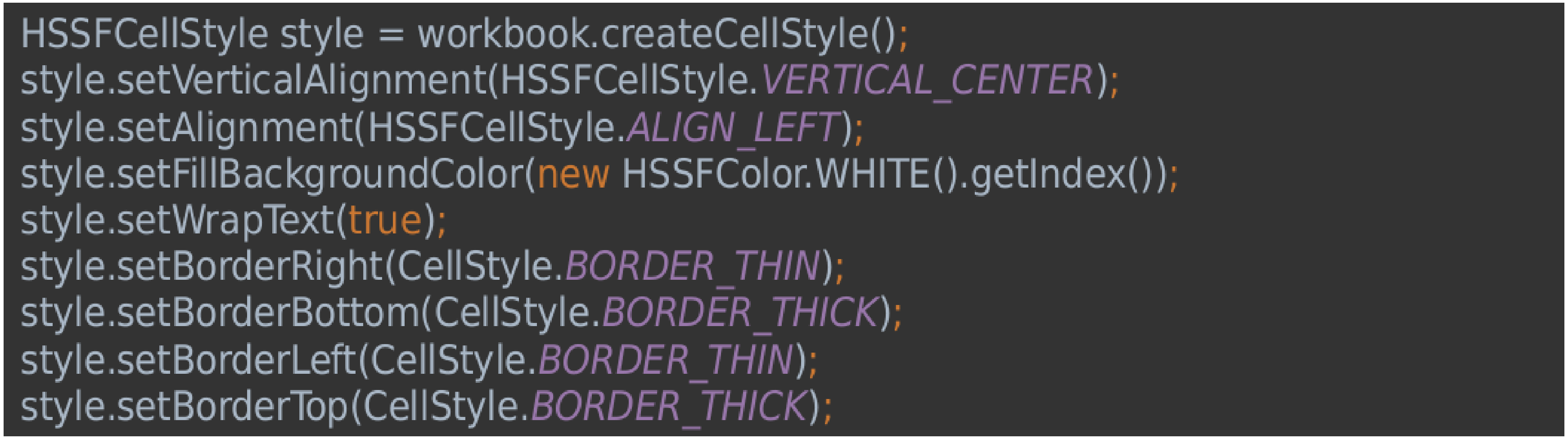
Apache POI

Workbook and sheet init :



```
HSSFWorkbook workbook = new HSSFWorkbook();
HSSFSheet sheet = workbook.createSheet("My first sheet");
```

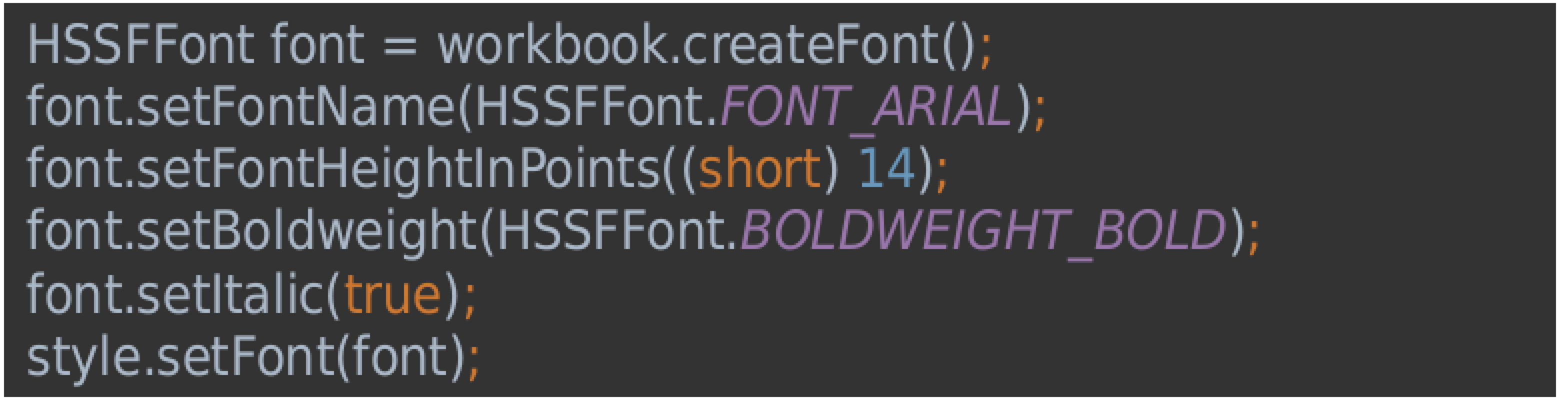
Defining Styles :



```
HSSFCellStyle style = workbook.createCellStyle();
style.setVerticalAlignment(HSSFCellStyle.VERTICAL_CENTER);
style.setAlignment(HSSFCellStyle.ALIGN_LEFT);
style.setFillBackgroundColor(new HSSFColor.WHITE().getIndex());
style.setWrapText(true);
style.setBorderRight(CellStyle.BORDER_THIN);
style.setBorderBottom(CellStyle.BORDER_THICK);
style.setBorderLeft(CellStyle.BORDER_THIN);
style.setBorderTop(CellStyle.BORDER_THICK);
```

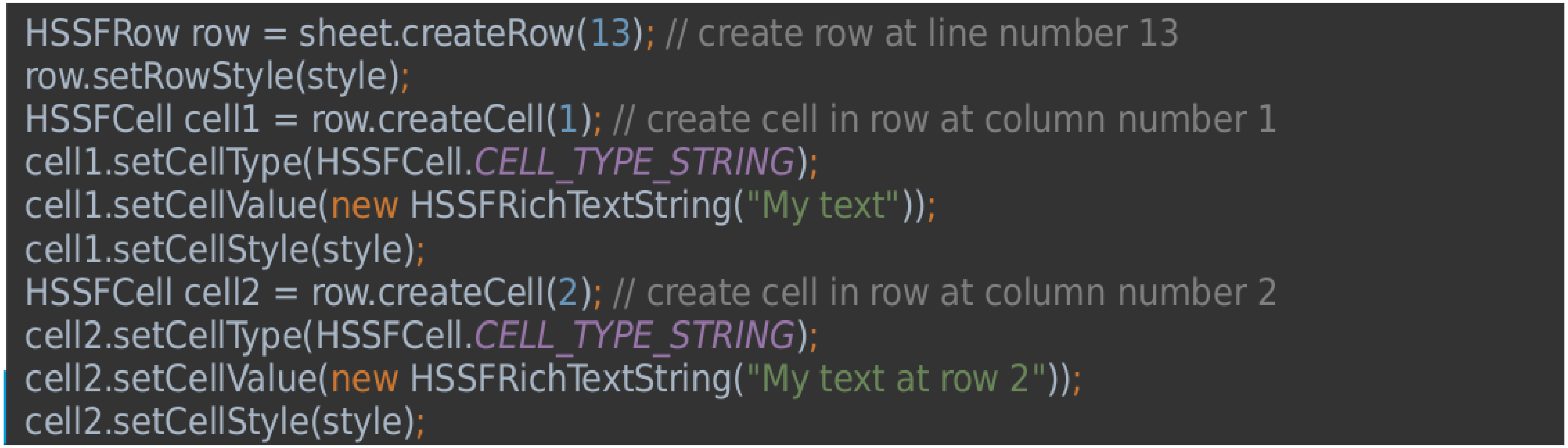
Apache POI

Defining Font :



```
HSSFFont font = workbook.createFont();
font.setFontName(HSSFFont.FONT_ARIAL);
font.setFontHeightInPoints((short) 14);
font.setBoldweight(HSSFFont.BOLDWEIGHT_BOLD);
font.setItalic(true);
style.setFont(font);
```

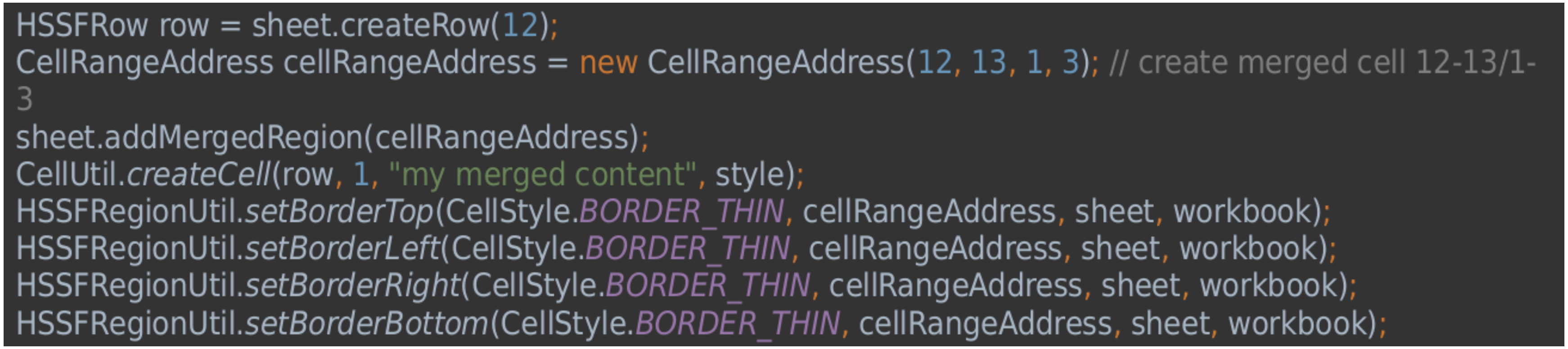
Cell creation and content management



```
HSSFRow row = sheet.createRow(13); // create row at line number 13
row.setRowStyle(style);
HSSFCell cell1 = row.createCell(1); // create cell in row at column number 1
cell1.setCellType(HSSFCell.CELL_TYPE_STRING);
cell1.setCellValue(new HSSFRichTextString("My text"));
cell1.setCellStyle(style);
HSSFCell cell2 = row.createCell(2); // create cell in row at column number 2
cell2.setCellType(HSSFCell.CELL_TYPE_STRING);
cell2.setCellValue(new HSSFRichTextString("My text at row 2"));
cell2.setCellStyle(style);
```

Apache POI

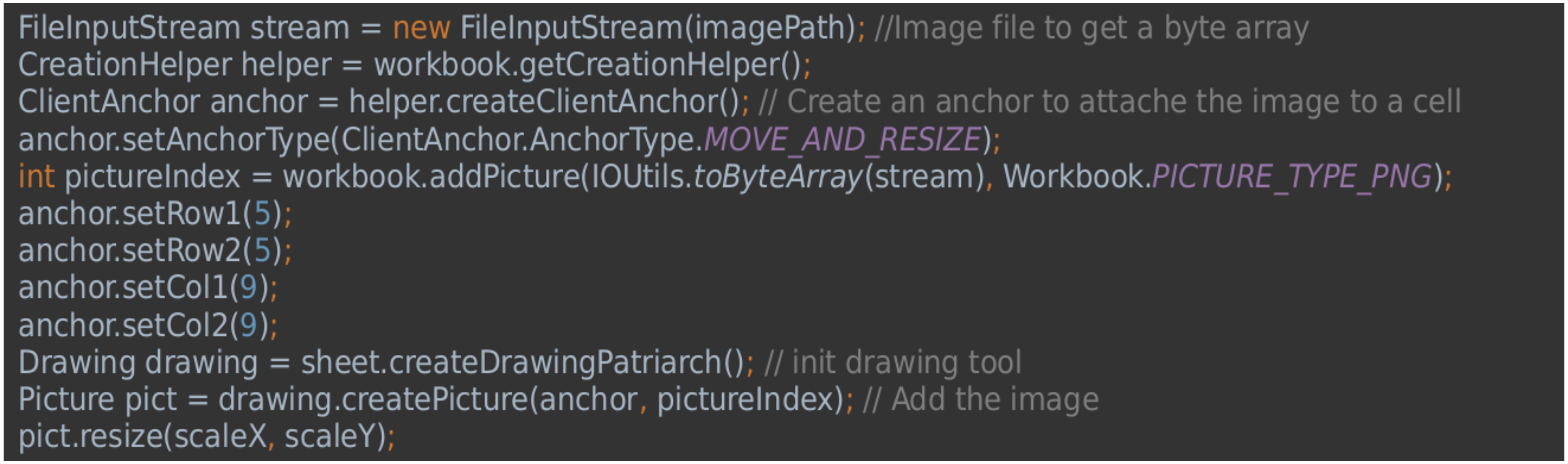
Merged Cell creation and content management



```
HSSFRow row = sheet.createRow(12);
CellRangeAddress cellRangeAddress = new CellRangeAddress(12, 13, 1, 3); // create merged cell 12-13/1-
3
sheet.addMergedRegion(cellRangeAddress);
CellUtil.createCell(row, 1, "my merged content", style);
HSSFRegionUtil.setBorderTop(CellStyle.BORDER_THIN, cellRangeAddress, sheet, workbook);
HSSFRegionUtil.setBorderLeft(CellStyle.BORDER_THIN, cellRangeAddress, sheet, workbook);
HSSFRegionUtil.setBorderRight(CellStyle.BORDER_THIN, cellRangeAddress, sheet, workbook);
HSSFRegionUtil.setBorderBottom(CellStyle.BORDER_THIN, cellRangeAddress, sheet, workbook);
```

Apache POI

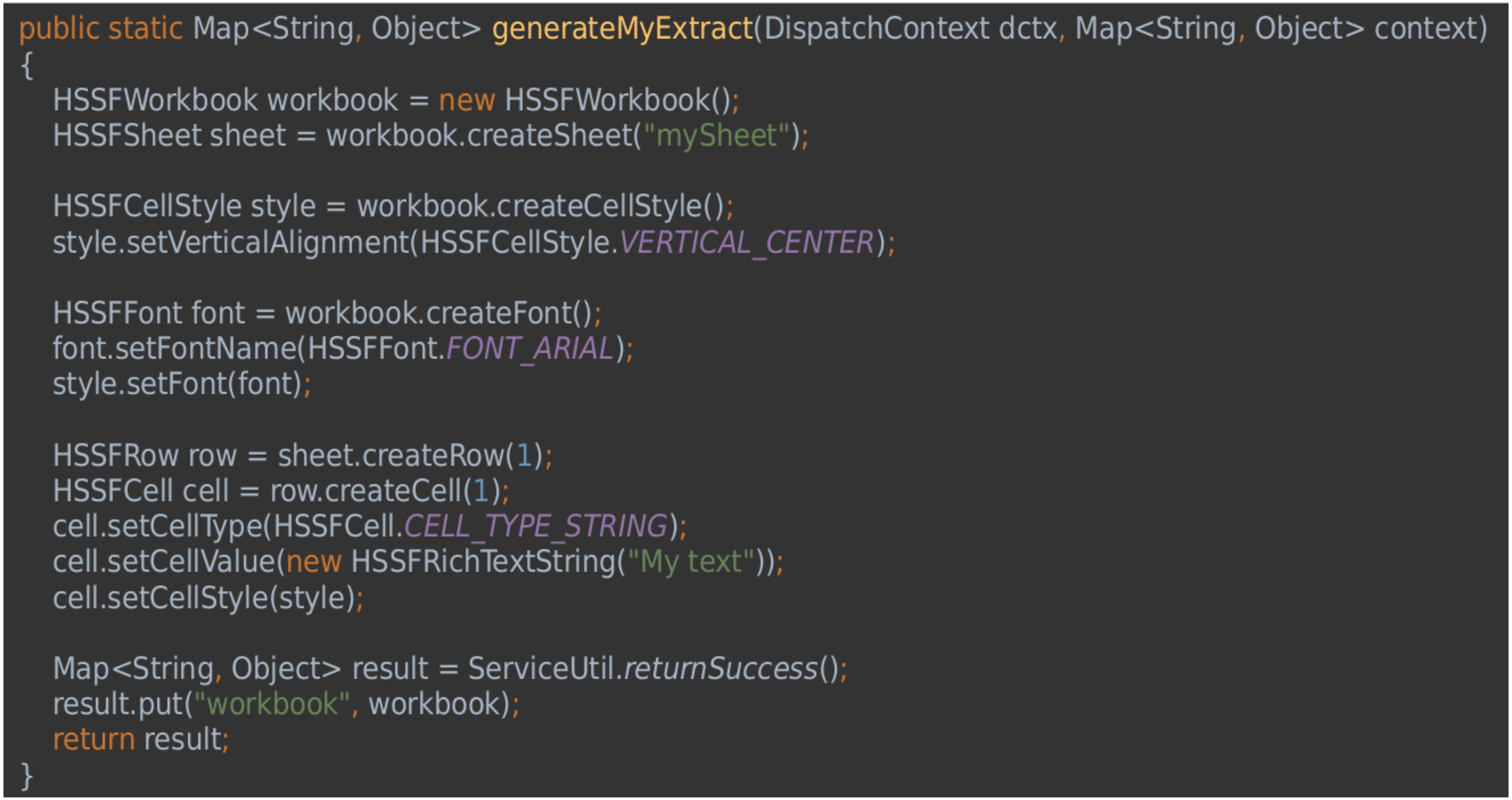
Image insertion :



```
FileInputStream stream = new FileInputStream(imagePath); //Image file to get a byte array
CreationHelper helper = workbook.getCreationHelper();
ClientAnchor anchor = helper.createClientAnchor(); // Create an anchor to attache the image to a cell
anchor.setAnchorType(ClientAnchor.AnchorType.MOVE_AND_RESIZE);
int pictureIndex = workbook.addPicture(IOUtils.toByteArray(stream), Workbook.PICTURE_TYPE_PNG);
anchor.setRow1(5);
anchor.setRow2(5);
anchor.setCol1(9);
anchor.setCol2(9);
Drawing drawing = sheet.createDrawingPatriarch(); // init drawing tool
Picture pict = drawing.createPicture(anchor, pictureIndex); // Add the image
pict.resize(scaleX, scaleY);
```

Apache POI

Service Structure



```
public static Map<String, Object> generateMyExtract(DispatchContext dctx, Map<String, Object> context)
{
   HSSFWorkbook workbook = new HSSFWorkbook();
   HSSFSheet sheet = workbook.createSheet("mySheet");

   HSSFCellStyle style = workbook.createCellStyle();
   style.setVerticalAlignment(HSSFCellStyle.VERTICAL_CENTER);

   HSSFFont font = workbook.createFont();
   font.setFontName(HSSFFont.FONT_ARIAL);
   style.setFont(font);

   HSSFRow row = sheet.createRow(1);
   HSSFCell cell = row.createCell(1);
   cell.setCellType(HSSFCell.CELL_TYPE_STRING);
   cell.setCellValue(new HSSFRichTextString("My text"));
   cell.setCellStyle(style);

   Map<String, Object> result = ServiceUtil.returnSuccess();
   result.put("workbook", workbook);
   return result;
}
```

Apache POI

Pro and cons:

+ Total control of xls design

− very verbose coding

− Performance issue for extracting lots of data (Not so true anymore)

Real purpose :

- Designed and aggregated data spreadsheet export (Report)

XSL-FO

Advanced PDF design

- Many example in OFBiz (invoice, order...)

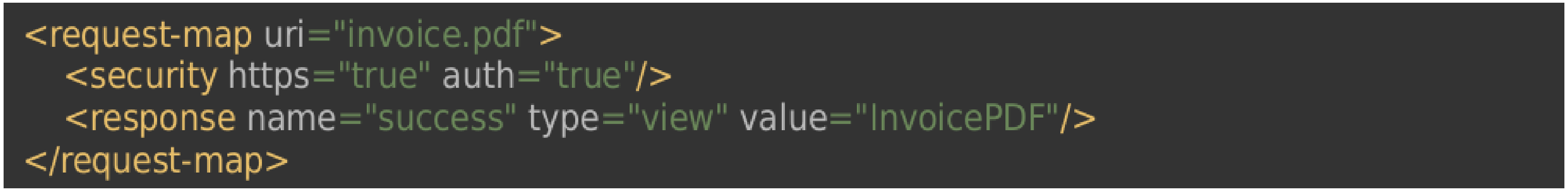
Use two Apache projects

- Apache Freemarker (incubator) : template engine
  http://freemarker.org/
- Apache FOP (Formating Object Processor) : print formatter
  http://xmlgraphics.apache.org/fop/

XSL-FO

Implementation Basics in OFBiz

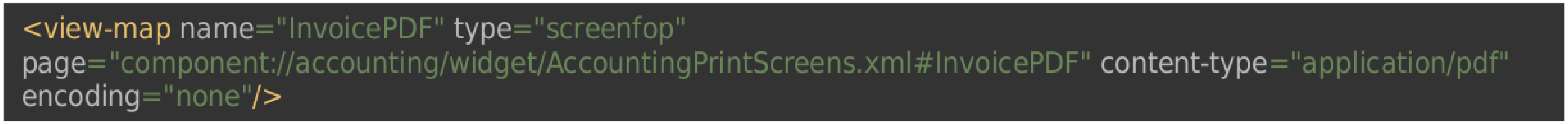
Controller :



```
<request-map uri="invoice.pdf">
    <security https="true" auth="true"/>
    <response name="success" type="view" value="InvoicePDF"/>
</request-map>
```

Screen manage data aggregation via actions...

Handler screenfop :

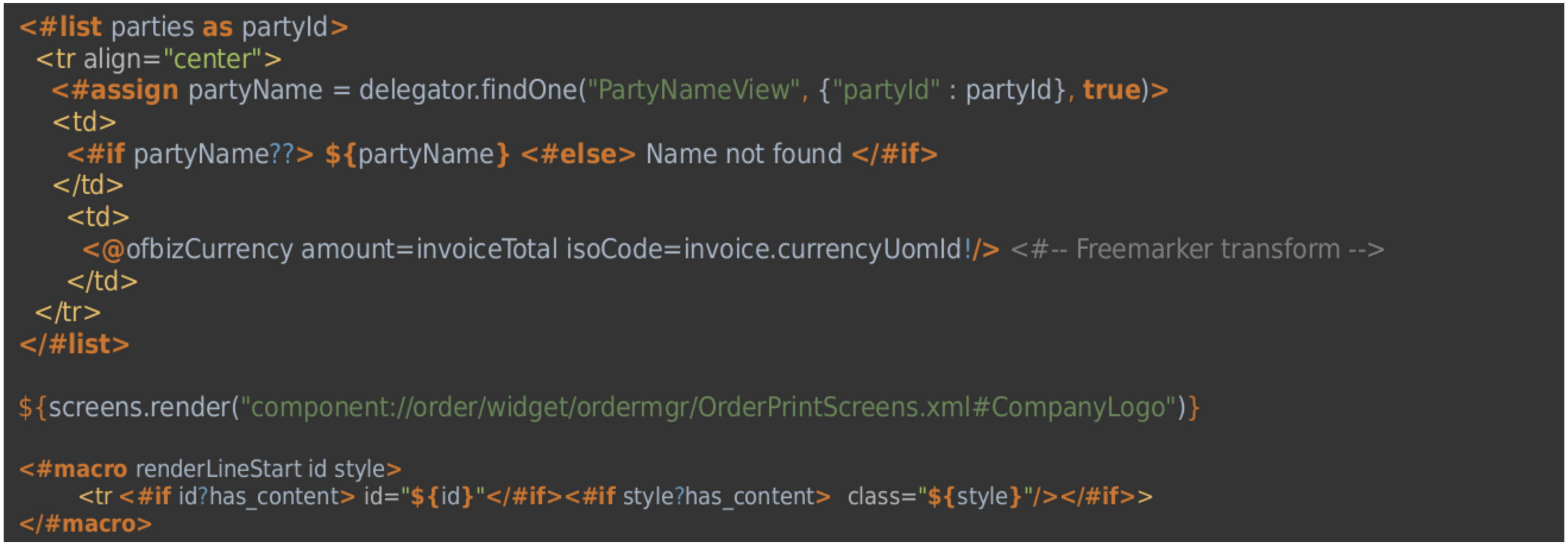


```
<view-map name="InvoicePDF" type="screenfop"
page="component://accounting/widget/AccountingPrintScreens.xml#InvoicePDF" content-type="application/pdf"
encoding="none"/>
```

XSL-FO

Implementation Basics in OFBiz

Freemarker

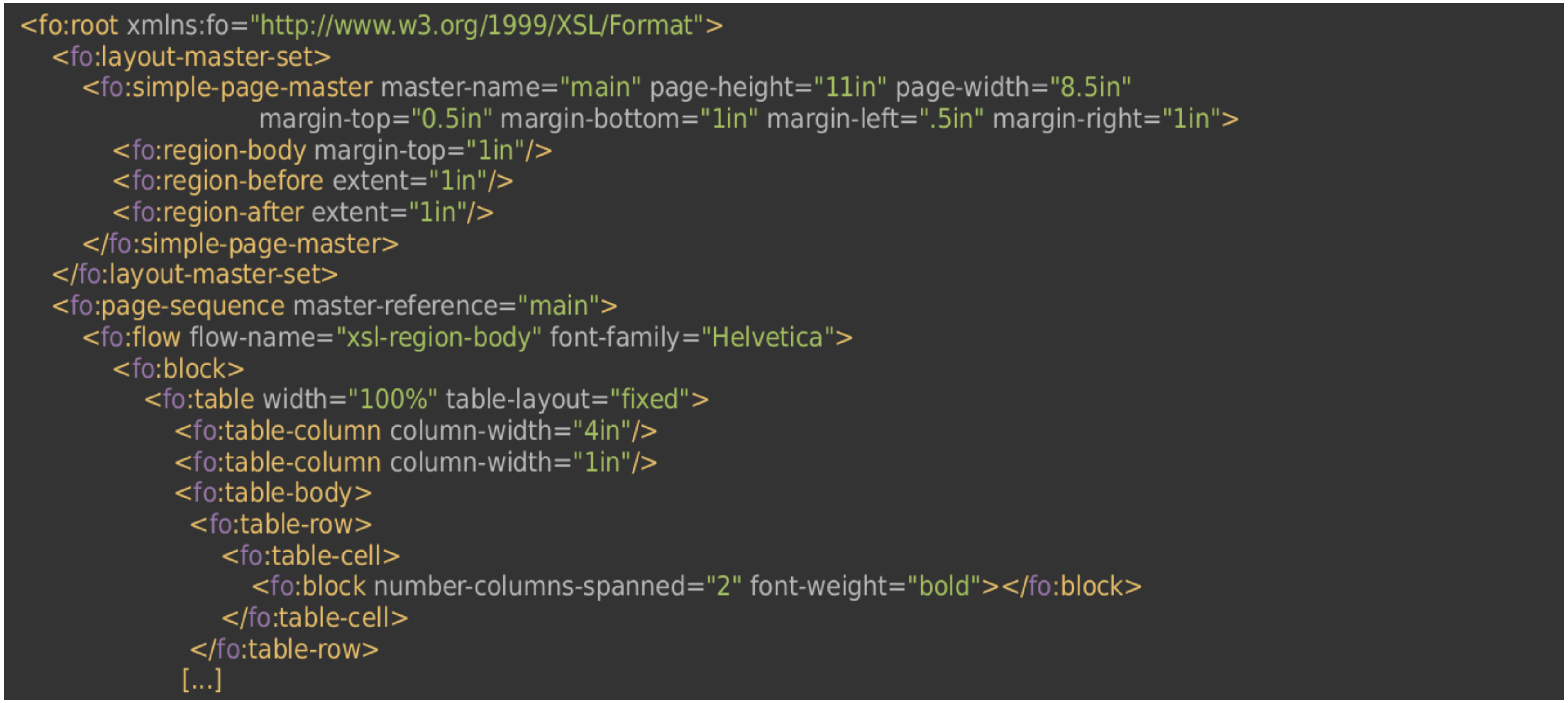


```
<#list parties as partyId>
 <tr align="center">
  <#assign partyName = delegator.findOne("PartyNameView", {"partyId" : partyId}, true)>
  <td>
   <#if partyName??> ${partyName} <#else> Name not found </#if>
  </td>
   <td>
    <@ofbizCurrency amount=invoiceTotal isoCode=invoice.currencyUomId!/> <#-- Freemarker transform -->
   </td>
 </tr>
</#list>

${screens.render("component://order/widget/ordermgr/OrderPrintScreens.xml#CompanyLogo")}

<#macro renderLineStart id style>
     <tr <#if id?has_content> id="${id}"</#if><#if style?has_content>  class="${style}"/></#if>>
</#macro>
```

XSL-FO

Implementation Basics in OFBiz

FOP :



```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="main" page-height="11in" page-width="8.5in"
                  margin-top="0.5in" margin-bottom="1in" margin-left=".5in" margin-right="1in">
      <fo:region-body margin-top="1in"/>
      <fo:region-before extent="1in"/>
      <fo:region-after extent="1in"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="main">
    <fo:flow flow-name="xsl-region-body" font-family="Helvetica">
      <fo:block>
        <fo:table width="100%" table-layout="fixed">
          <fo:table-column column-width="4in"/>
          <fo:table-column column-width="1in"/>
          <fo:table-body>
           <fo:table-row>
             <fo:table-cell>
               <fo:block number-columns-spanned="2" font-weight="bold"></fo:block>
             </fo:table-cell>
           </fo:table-row>
          [...]
```

XSL-FO

Pros and cons

+ Total control of pdf design
+ Modular with freemarker macro usage
− Very verbose coding

Real purpose :

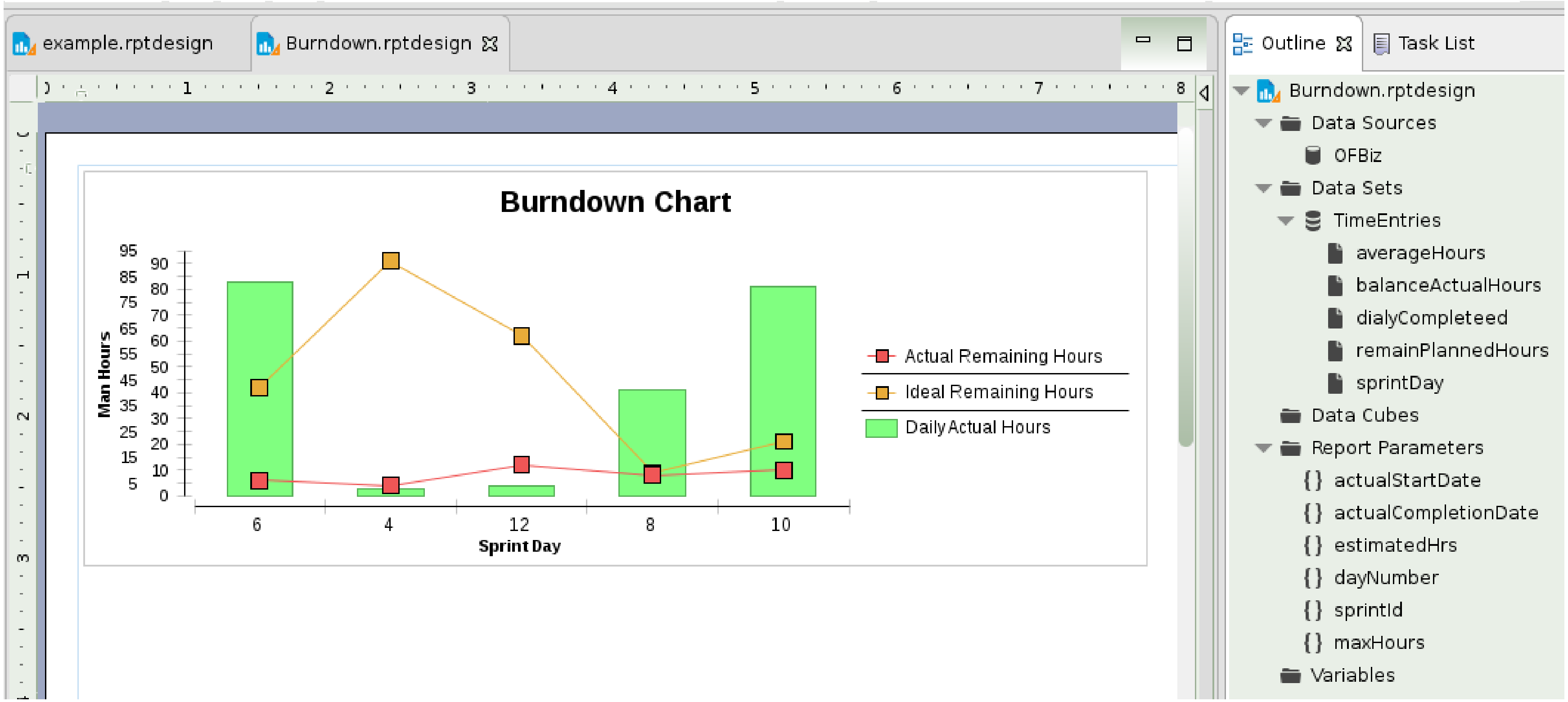
- Designed and aggregated data PDF export (Report)

Birt

Complex multitype design

- Html/pdf/xls/...
- Data Table, Charts ...

Eclipse Designer :



- Burndown.rptdesign
  - Data Sources
    - OFBiz
  - Data Sets
    - TimeEntries
      - averageHours
      - balanceActualHours
      - dialyCompleteed
      - remainPlannedHours
      - sprintDay
  - Data Cubes
  - Report Parameters
    - {} actualStartDate
    - {} actualCompletionDate
    - {} estimatedHrs
    - {} dayNumber
    - {} sprintId
    - {} maxHours
  - Variables

Birt

Pro and cons

+ Many export format

+ Designable by advanced users (for existing report)

− Heavy to implement/use (any feedback?)

− Eclipse dependant

Real purpose :

- Designed and aggregated multi-type data export
- Give some flexibility to advanced User

Future ?

- 😄 Paris Meeting 2016-08-10
  - Diverse interesting topics discussed
  - Data Extraction

Analysis and Proof Of Concept

The Idea

Thanks Taher for the Food for thought !

What is the most spread « language » : HTML

There must be out there some tools to convert HTML to PDF !

Why not using JavaScript ?

The idea was to find a Javascript system to easily convert html rendered screens to PDF !

Proof of concept

In practice :

What to convert from HTML ?

- Simple Tables :
    - jsPDF library with autoTable plugin
- Other Objects ? Not Sure

My little POC :

- JS export button in FindScreenDecorator
    - Extend with JQueryUI to select fields

Any Ideas ? Discussion is Open !