J4L FO Designer 2.16

Table of contents

1. Introduction....5
    Requirements....5
    Setup & startup....6
    Delivered files....9
2. The user interface....10
    The properties view....11
    XML Schema folder....13
    The XML test files folder....14
    The editor view....16
    The toolbox bar....20
    Working with the database....21
    Versioning system....24
    SVN plugin....26
    The XPath editor....29
    The help system....31
3. Working with the user interface....32
    Creating, deleting, importing and exporting....32
    Operations on files....36
    Selecting objects....37
    Operation on objects....37
    Operations on areas....38
    The preferences dialog....39
    The local history....41
4. Structure of the template....42
    Page masters....42
    Areas....45
    How to analyze the report execution....51
    Background colors and images....55
    The background area....57
    Attachment to the PDF file....58
    Report‘s metadata....59
    Decimal numbers formatting....61
    Columns and rows markers....62
    Drawing lines with columns and rows markers....65
    Properties of the objects....65
    Properties of the template....65
    Properties of the areas....67
    Properties of a text field....68
    Properties of a memo field....69
    Properties of a combo box....69
    Properties of a picture....69
    Properties of a line....70
    Properties of a www links....70
    Properties of a free code objects....71
    Properties of a table frame....74
5. Executing the template to create PDF files....77
    FOP web server (servlet)....77
    Java objects to PDF conversion....79
    Sending the PDF as an email attachment....80
    Adding a digital signature to the PDF file....80
6. Learn by doing: tutorial....83
    The examples in the database....83
    Hands on example....84
7. The invoice IDOC example explained....91

how to work without a schema file....93
Use of second level detail areas....94
Use of conditions in the XPath....95
Use of combo boxes....96
8."Group by" example....97
Group footers....98
9. The two columns example....100
10. The Barcodes and dynamic images example explained....101
11. The chart example explained....106
12. The running totals example explained....108
13. PDF interactive forms....111
Introduction....111
Form objects....111
Text field....112
Combo box....112
Checkbox....112
Button....113
Filling in the form....113
Form submission....115
Input form example....116
Form‘s internal structure....119
14. Flavours....122
15. Multilanguage support....124
16. The base template....127
17. Long texts and HTML fields....129
The Letter example....130
The HTML fields....132
18. The Memo field explained....135
19. Advanced functionalities....137
Understanding the generated code for areas....137
Introduction to Variables and Keys....139
Adding your own code....142
Markers (variable data in page header and footer)....144
Advanced examples....147
Using third party XLT templates....149
20. The inhouse Premium and Cloud J4L FOP Servers....152
Sending the PDF per email....154
PDF Protection....156
Creation of Excel files....157
21. Migration of existing XSL-FO files....158
Limitations....158
Migration step by step....159
22. FAQS....165
How to prevent NaN values in numeric fields....165
How to enable the designer console....165
How to debug Oracle APEX requests....165
How to enable Oracle APEX logging on Glassfish....166
How to create a user defined xpath function....167
How to add new fonts to J4L FO Designer....169
How to add new fonts to Oracle ORDS....172
How does the designer deal with namespaces....173
How does support for international character sets work....173
How to add my own xslfo attributes to the fields....173
How to add page numbers and page total count....174
23. Shortcuts....175
24. Troubleshooting....177
Permission error while creating a PDF or saving a report....177

25. Third party licenses.....178
26. Contact.....178

1. Introduction

The J4L FO Designer is a graphical tool for designing XSL-FO documents. XSL-FO is a language for formatting XML data and one of the most popular uses is for converting "technical" XML files to user friendly PDF files.

This tool uses the **Apache FOP** package for executing the XSL-FO documents and therefore it is specially suitable for Java environments.

The FO Designer will help you in the task of creating PDF files from XML files. There is no need for you to learn the XSL-FO language. Even if you are familiar with the XSL-FO language, the designer provides you with a user friendly graphical tool for increased productivity

Furthermore the designer provides an integrated environment for:

- Designing the PDF files in a WYSIWYG manner.
- Loading and modifying XML files for the testing.
- Running tests and checking the output with one click.

When working with the designer you will be handling up to 6 different files:

1. The input XML file (also called ***XML document*** in this guide) which needs to be converted to
   PDF
2. The XML ***schema file***, with extension *.xsd (optional) which defines the structure of the XML document. In the schema file you have a list of all possible elements contained in the input XML document. So, from the schema you can select fields to be placed on the PDF file.
3. The XRE file (also called document ***template or report*** in this guide) which stores the layout of the desired output, this is what you see in the designer and is used at **design time only**.
4. The ***XSL-FO document***. The XSL-FO output is generated by the designer and **used at runtime** by Apache FOP to convert your XML file to PDF.
5. The **FO output** (FO tab) which is the result of applying the text XML file to the ***XSL-FO document***..
6. The output ***PDF file***.

Requirements

FO Designer requires:

- Java 1.8 or later (included)
- Adobe reader to view the generated PDF files
- Apache FOP (included in the delivery). See also third party **licenses section**.
- J4L FOP Server (*http://www.java4less.com/apache/fop.php*) *or* J4L WAR file deployed on your web server **IF you use any of these proprietary extensions***:*
  1. charts
  2. barcodes
  3. generate interactive PDF forms
  4. use digital signature
  5. J4L XSL functions in the Xpaths

Setup & startup

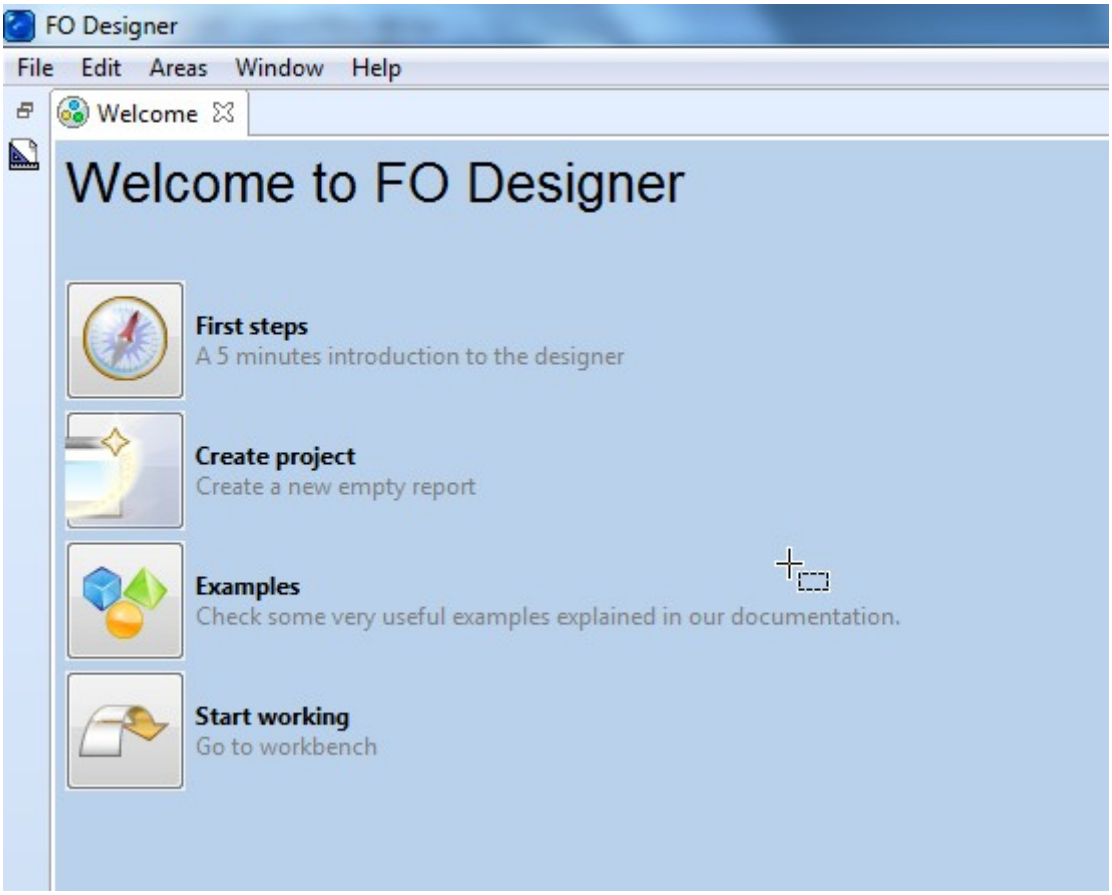
Installation

Unzip the downloaded ZIP file in a empty directory.

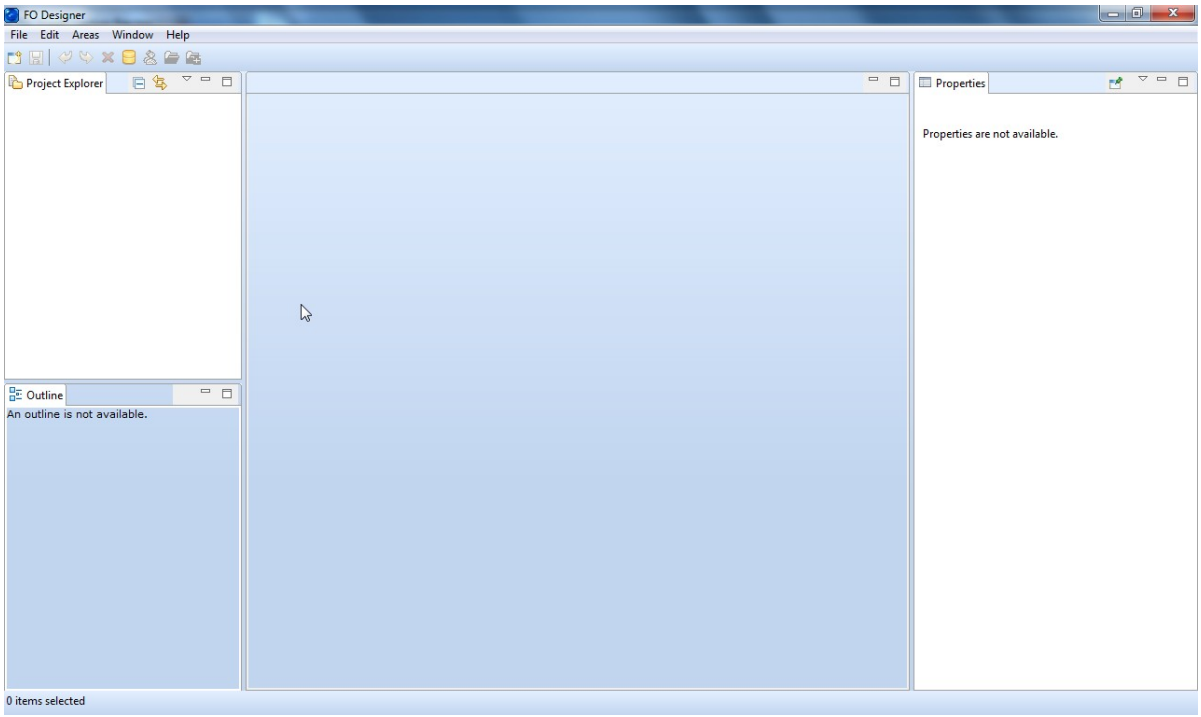
Start up

Start the file fodesigner.exe, the welcome page will be displayed. After start the following directories will be created:

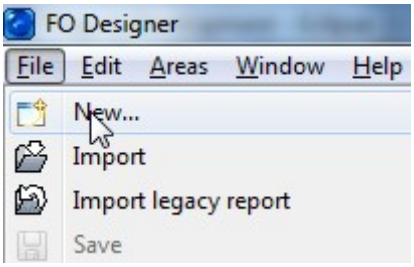
- *UserHomeDirectory/FO/db :* the database that will be copied from the installation *directory/db.*
- *UserHomeDirectory/FO/Workspace:* this is the working directory where all the projects and files will be stored.



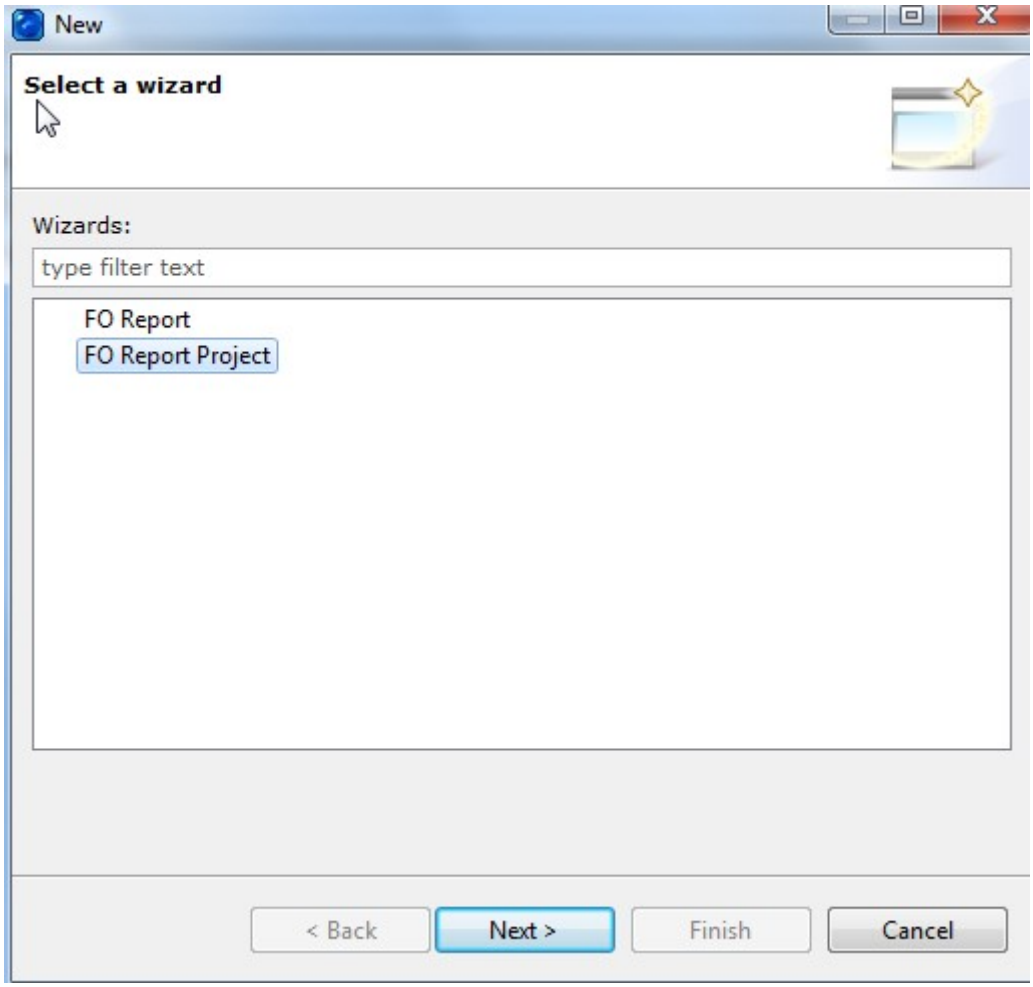
Click on the „Start working“ icon to open the workbench:



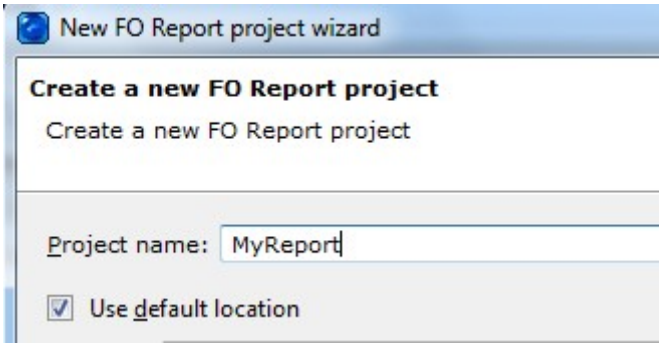
Creating the first report: Use the File→New menu item



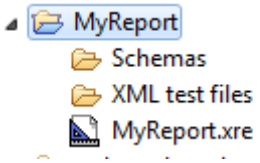
Select „FO Report Project“



Enter a report name:



and finish:



the created project will consist of the report file (xre) and 2 folders:

- the Schemas folder where you can load the XML schema
- and the „XML test files“ where you can copy files for testing your report. Files can be copied from windows explorer into the project using *copy* and *paste*.

Note the project name and the template file (xre) which always have the same name.

Delivered files

The delivery includes the following files and directories:

- **db: database directory.**
- **Examples included in the database**: several examples including schema, xml files and template for the designer
  - **employees**: example for the **tutorial section** in this guide, including grouping and **cover page example**.
  - **apex**: a simple Oracle APEX example
  - **apex_order:** a purchase order document based on Oracle APEX
  - **idoc_invoice:** SAP IDOC invoice example, **explained later** in this directory. o **xcbl_order:** reference example for this guide.
  -
  - **chart:** example report with charts.
  - **form:** interactive form
  - **barcodes:** example report with barcodes.
  - **JavaClass_to_PDF**: shows how to convert java classes to PDF
- **certs:** certificate files for testing the digital signature
- **web:** file for deployment to web server as servlet

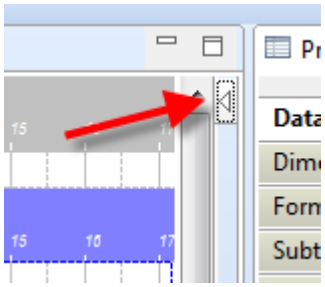
2. The user interface

The main window of the designer is divided in 3 areas:

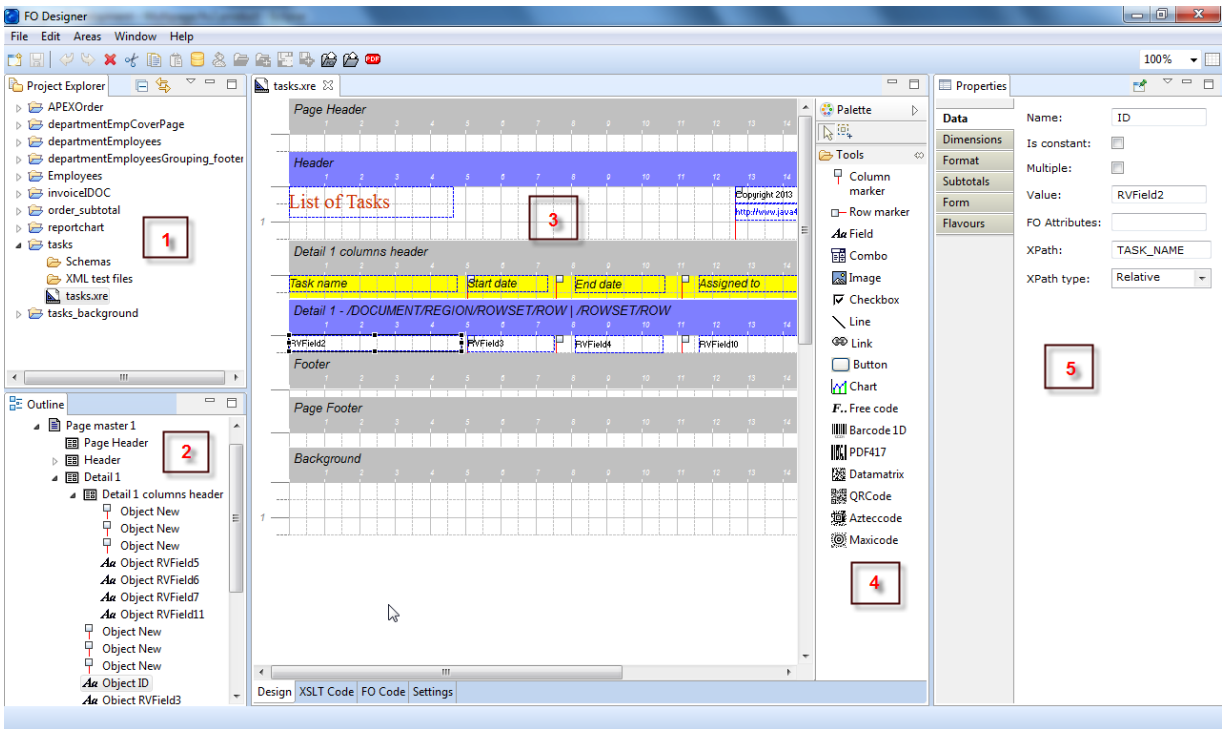
1. On the left side there are 2 views:
    - View 1 (left-top) is the Project explorer where the reports and the test files are located
    - View 2 (left-bottom) is the object outline of the currently open report. The outline provides an overview of the structure and content of the report.

2. On the middle view (3) is located the report editor. Next to it (4) is the tools palette.

    Note if the palette is not visible you need to click on this icon:

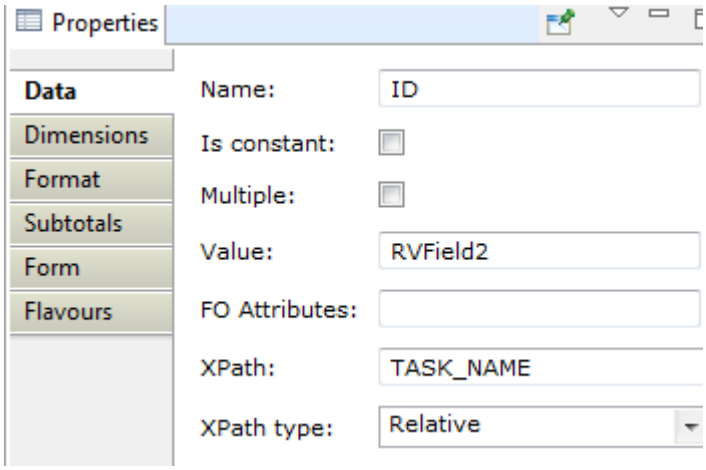


3. On the right view (4) the properties of the selected object will be displayed.



The properties view

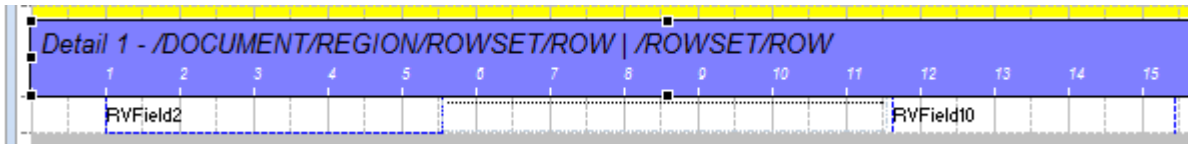
The properties view on the right side of the window will display the properties of the selected object grouped by section.



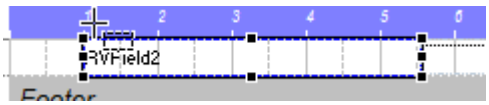
The selected object can be:

- the report
- the master
- the area
- the report fields

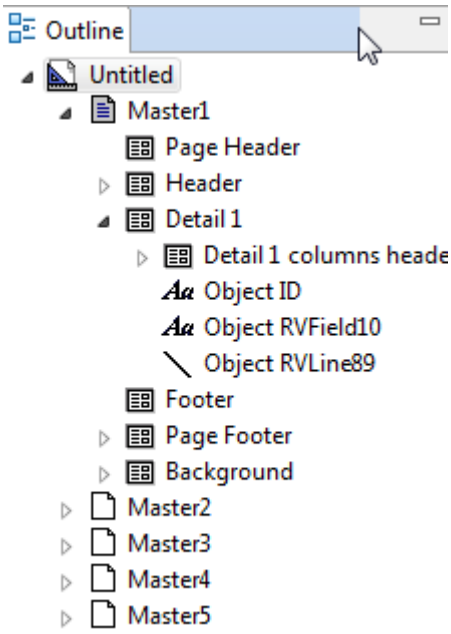
Areas and fields can be selected in the editor. For example you select an area by selecting the ruler (blue or grey header) or the (white) area background. You can also select the area in the outline view.



you select an object and view the properties by click on the field:

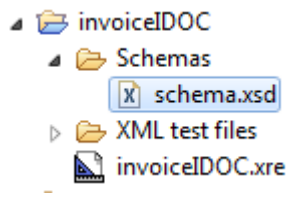


Report and master properties can only be displayed by selecting them **in the Outline view**.



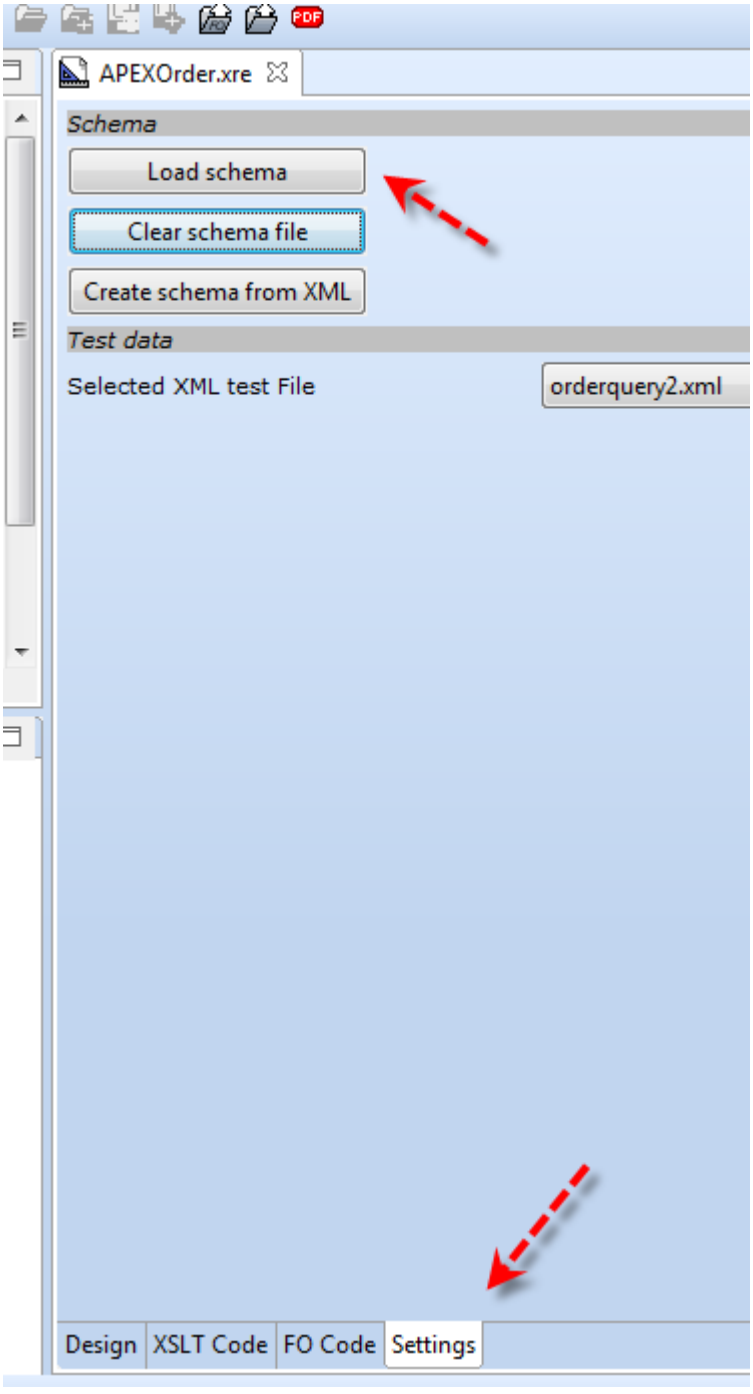
XML Schema folder

In the schema folder of the project



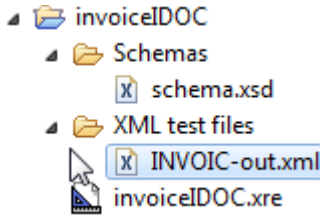
you can load the schema for the report. In the XML schema you have a tree structure which contains all possible elements of the input XML document.The schema file will be needed for the XPATH editor.

You need to load the schema into the folder by using the Settings tab of the editor and the „load schema“ button.



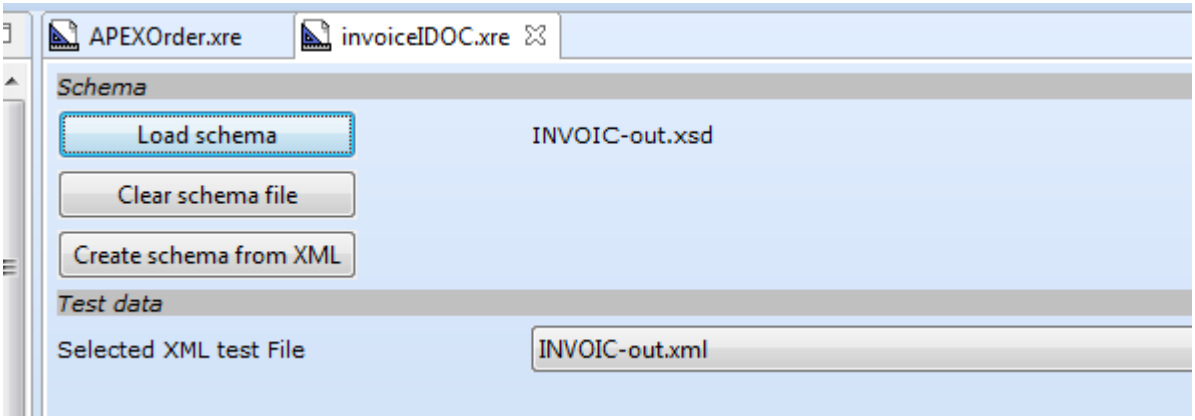
The XML test files folder

This folder contains the XML file that can be used for testing the report

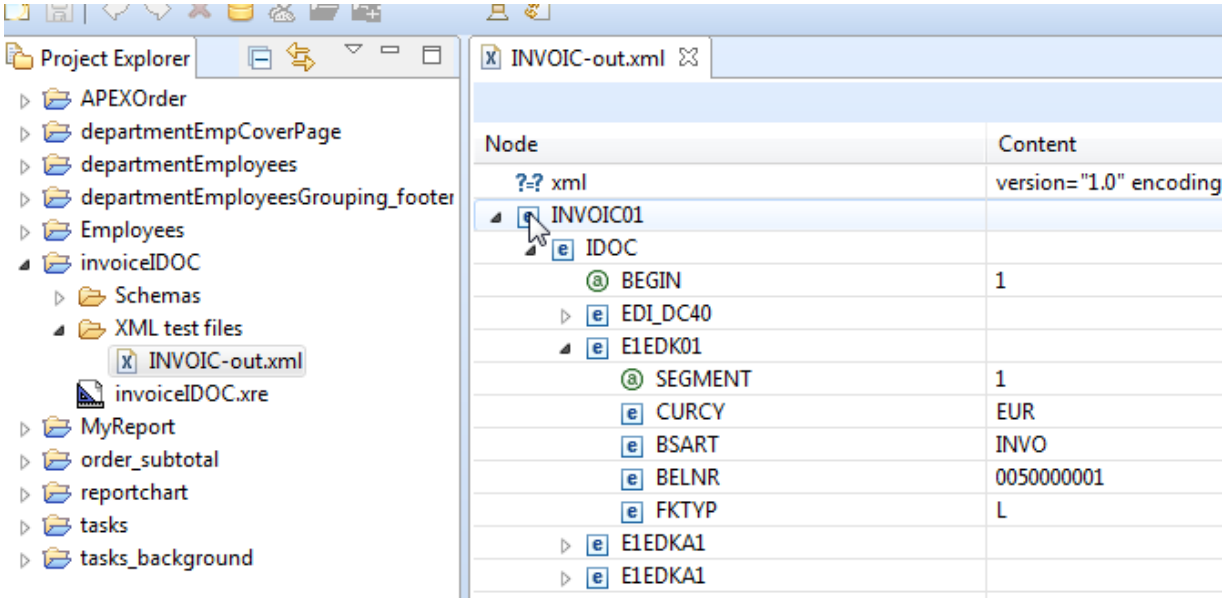


there can be one or several file in the folder. You can add files in windows by using *copy and paste* from windows explorer.

In order to use a file for testing the report you need to select the file in the report editor, setting tab.



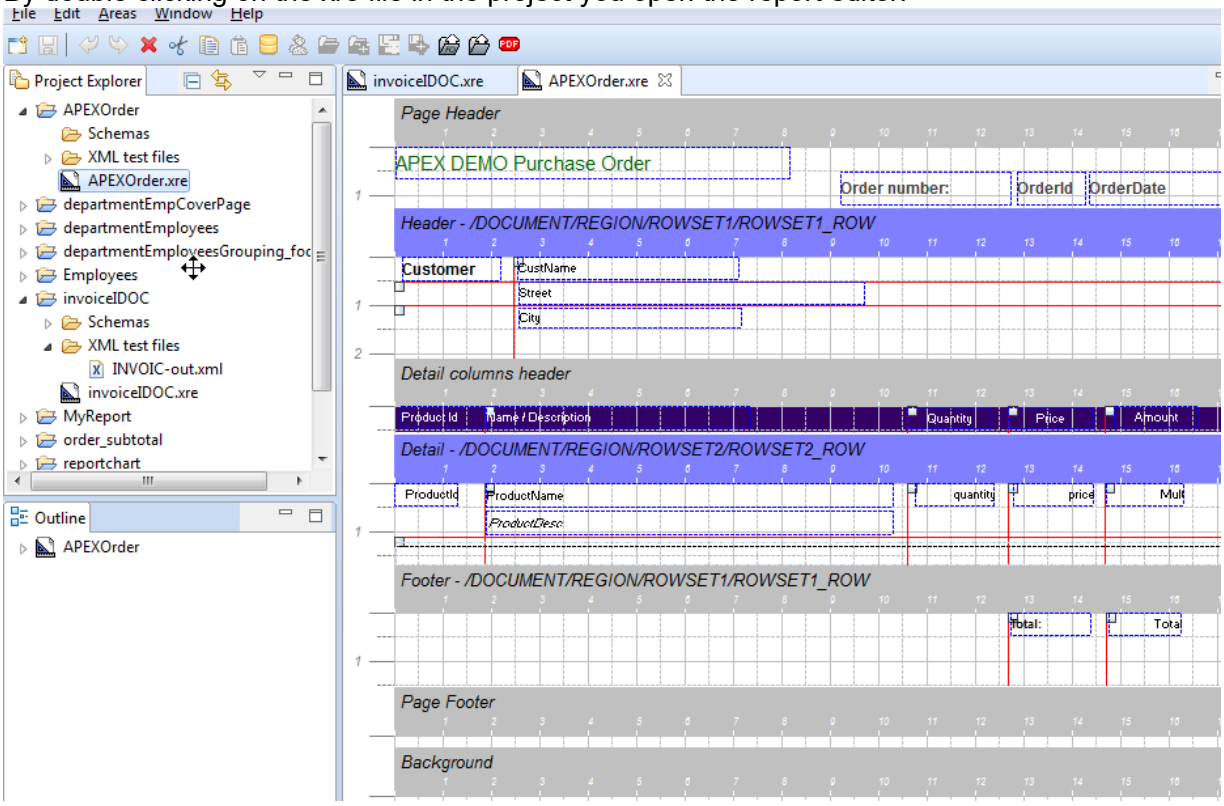
Loaded files can be edited by double clicking on them:



Note however, the designer will not check the validity of the XML nodes you create or import, that is, you can create nodes that are not part of the schema or you can create them in the wrong sequence or position.

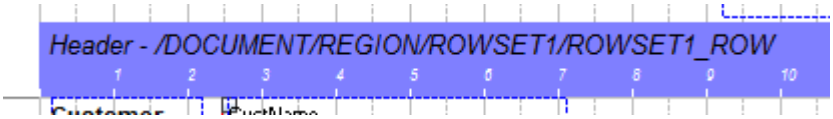
The editor view

By double clicking on the xre file in the project you open the report editor:

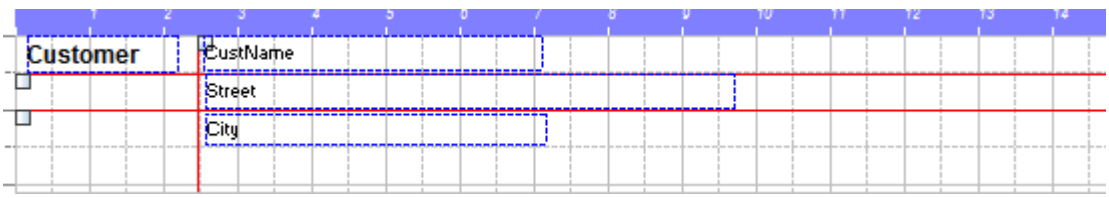


1. The ruler region. Click on this to display the properties of the area in the properties view.

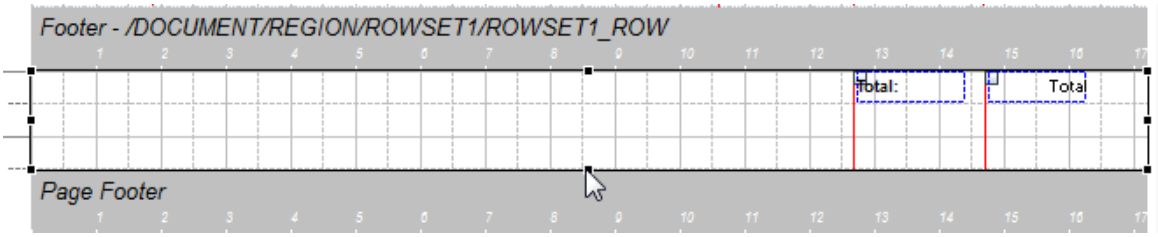
   The ruler contains the name of the area and the XML node element that has been assigned to the area:



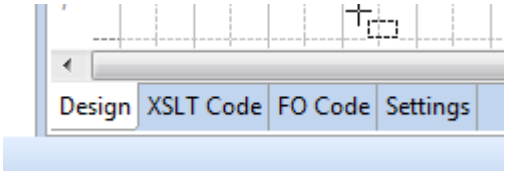
2. The work panel itself. In this panel you place the fields and other objects of the template. In order to add an object you select one of the tools from the palette and drag it the position of the work area where you want to add the object.



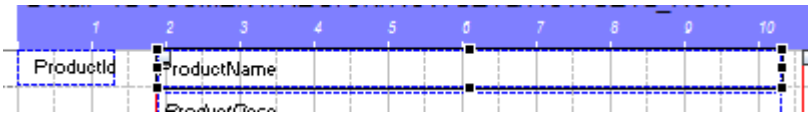
4. To resize an area (the height of it) you can click on the area background and move the area anchors up and down.



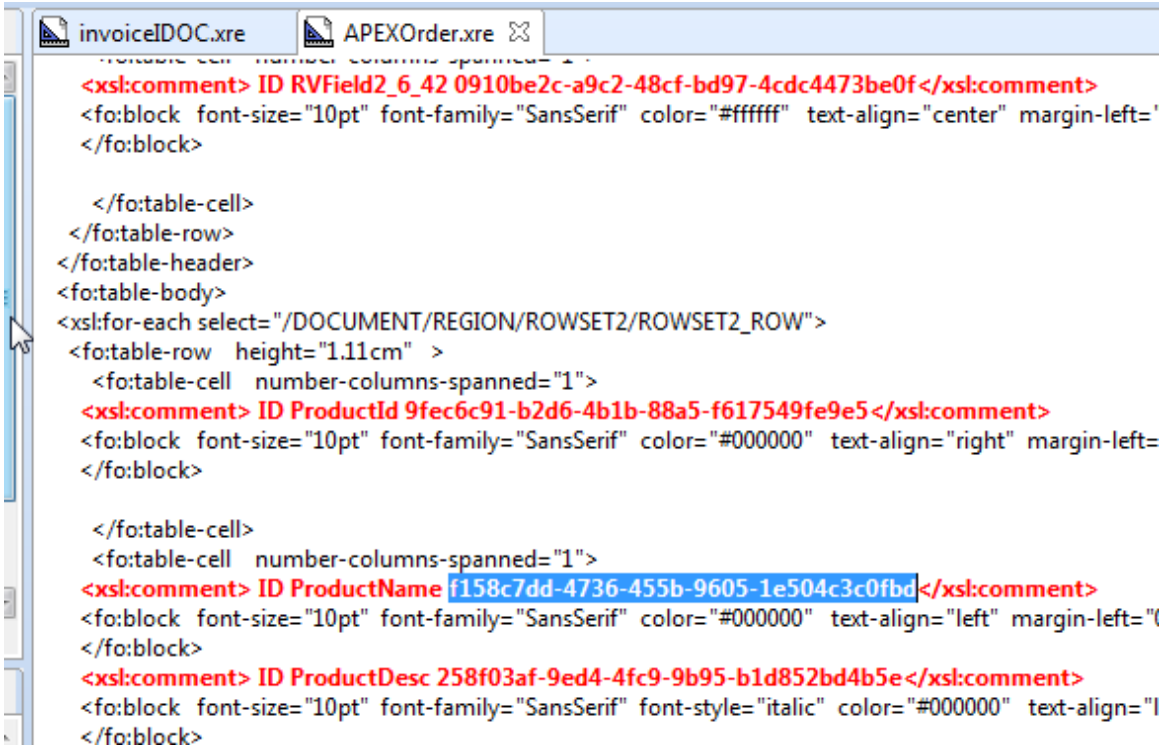
5. The editor has 3 additions tabs
   - the **XSLT Code**: will display the generated xsl code. This is the code that will be exported.
   - the **FO code**: will display the generated FO (code). That is the result of applying the XML test file to the XSLT code. Here you can see the code with the test data instead of Xpaths expressions.
   - The **settings** tab is used to select the schema file and the active test file.



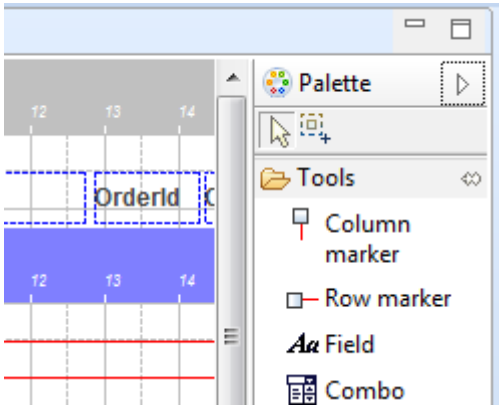
Both the XSLT code and the FO Code will always hightight the currently selected object so you can check the generated code. In the example below the object „ProductName“ has been selected.



Selected code:



6. the tools palette next to the editor contains all the objects that can be placed in the report



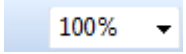
these are:

Column marker	Column marker
Row marker	Row marker
Field	text field. Drag this field to the area in order to create a new output text field .
Memo	Memo field for rich text content
Combo	combobox object. Select this tool to add a combobox. A combobox is a list of pairs key/value. The value in the input XML document will be used as key to find the value in the list. The value will be output in the PDF document.
Image	image object. Select this tool to add an image to the template.
Line	line object. Select this tool to add an horizontal line.
Box	Border box. Create a box object
Table	Table frame around existing fields
Barcode 1D	barcode object. Select this tool to add a 1D or 2D barcode. Note the designer includes the evaluation version of the barcoding components. This component requires the suite or enterprise license.
Chart	chart object. Select this tool to add a chart to the report. Note the designer includes the evaluation version of the charting component. This component requires the suite or enterprise license.
Link	Link object. Use this object to add www links to your report. The value field can be a constant value or a XPath expression, it represents the www link. The "Link label" field will be the displayed value in the report, if empty the http link will be used.

	As example *tasks.xre*.
F.. Free code	Free code object. Use this object to place you own XSL-FO in the output of the report.
Checkbox	Check box for input forms
Button	action button for input forms

The toolbox bar

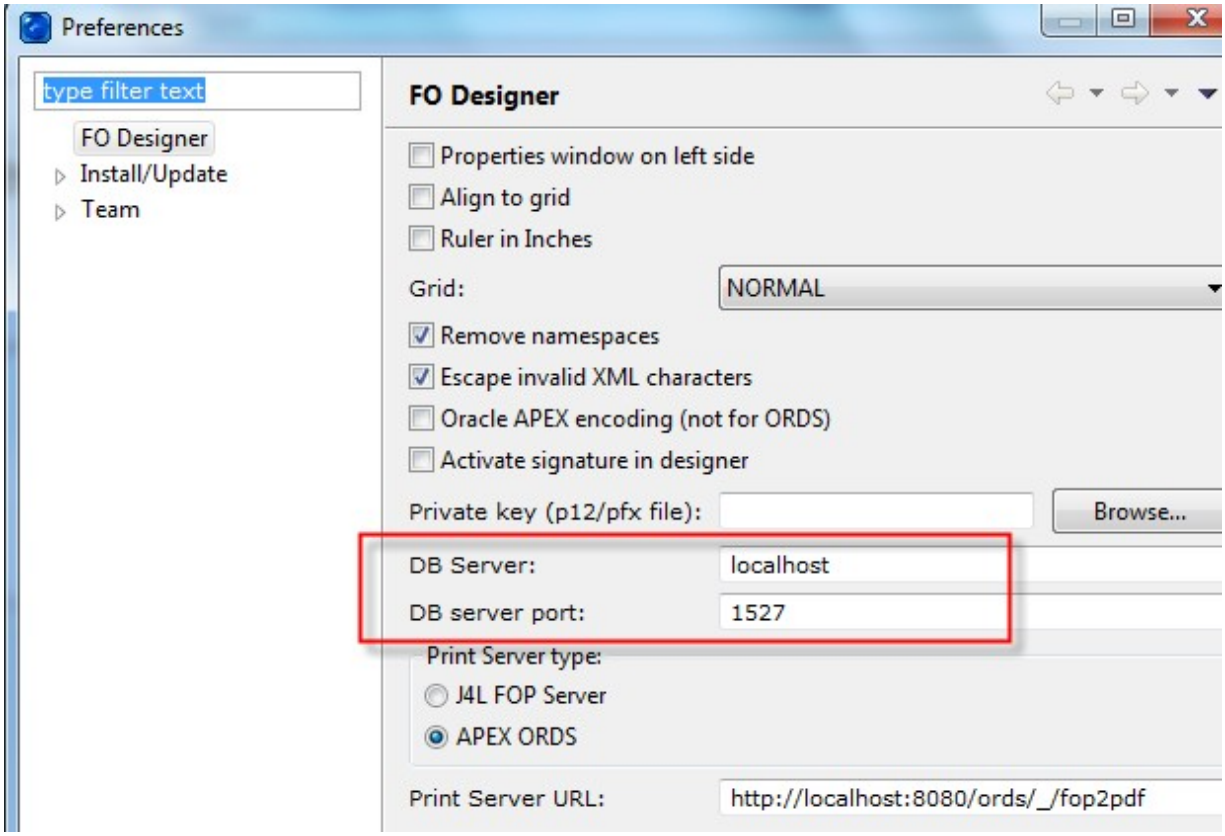
The actions available in the tool bar are:

	Create new report project
	Save report
	Undo and redo actions
	Delete, cut, copy and paste report objects
	Open database
	Database user administration (enterprise license only)
	Open report from database
	Open previous version of report from the database
	this button generates the XSL-FO output for the document. This output can then be used in your runtime environment, using Apache FOP. Note this button might be disabled in the evaluation version.
	save template to database (overwrite current version).
	save template to database and create new version this button generates a PDF file using the current template. This button requires that you have loaded a test XML document in the test data panel. After generating the PDF document you have the option to open it manually or let the designer open the PDF document for you
	Use this button to analyze the creation of the PDF file. This is a tool to assist debugging the PDF generation by tracing step by step the evaluation of the Xpath expressions in the report. See the „Structure of the template“ chapter for a detailed description of this functionality.
	this button generates a PDF file using the current template. This button requires that you have loaded a test XML document in the setting tab. After generating the PDF document it will be opened using the system‘s default reader.
	Generate the Excel file locally
	Generate the PDF file using a remote FOP server (J4L FOP Server or APEX Listener/Rest Data Services). See preferences window.
	Generate the Excel file using the J4L Premium and Cloud FOP Server
	Zoom editor work area
	Toogle report‘s grid

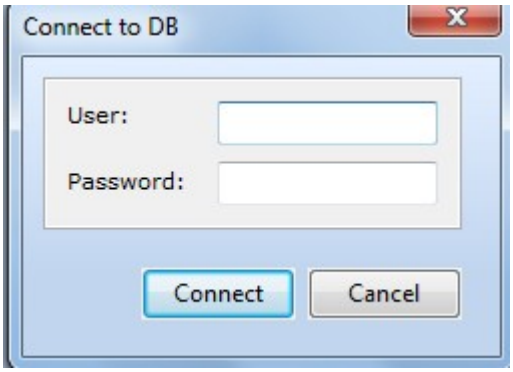
Working with the database

You can work with the project on the file system however you can also use the database. If you are working in a enterprise environment it can be useful using a central database.

By default FO Designer will use the local database (db subdirectory of the installation) however for team work you would probably use a central database (installed as part of the J4L FO Server). This can be setup it the Windows→preferences dialog. A central database can be set in the enterprise version (license) only.

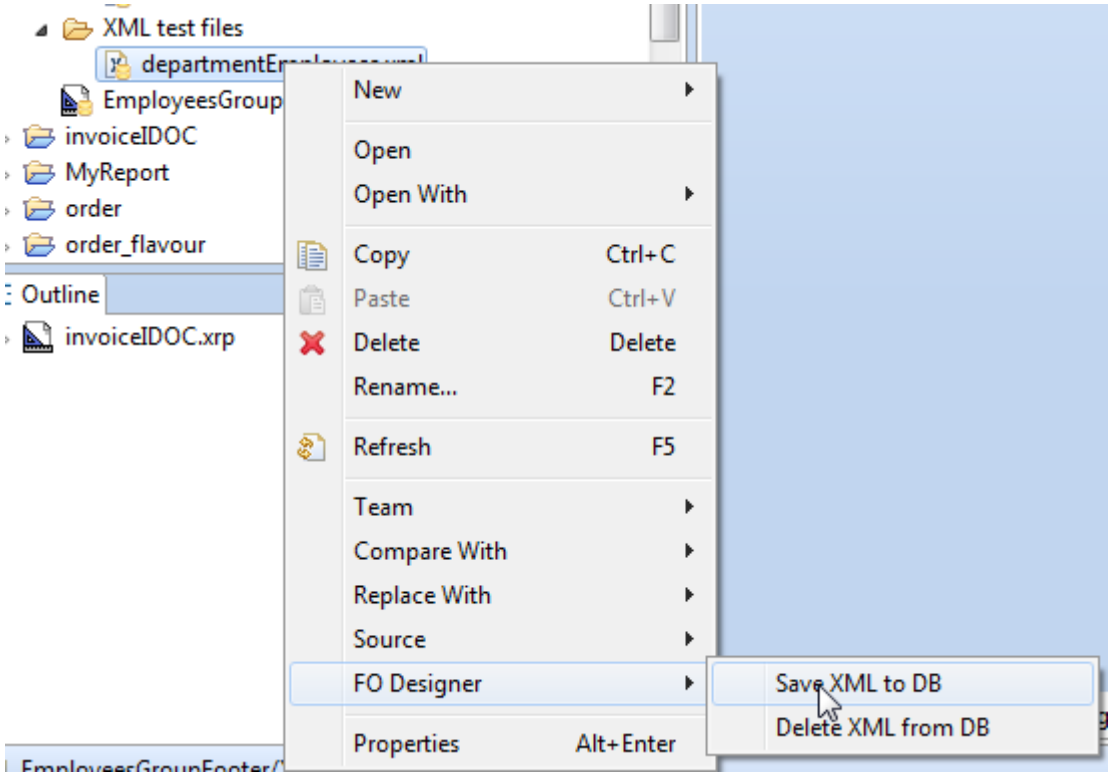


use the button for connecting to the database, the default user and password for the local installation are *admin/admin*

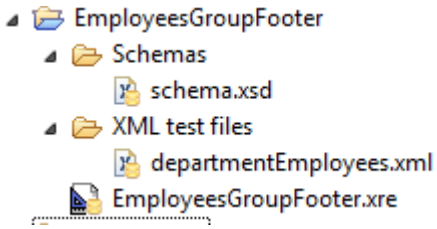


Once connected to the database you will be able save your projects:

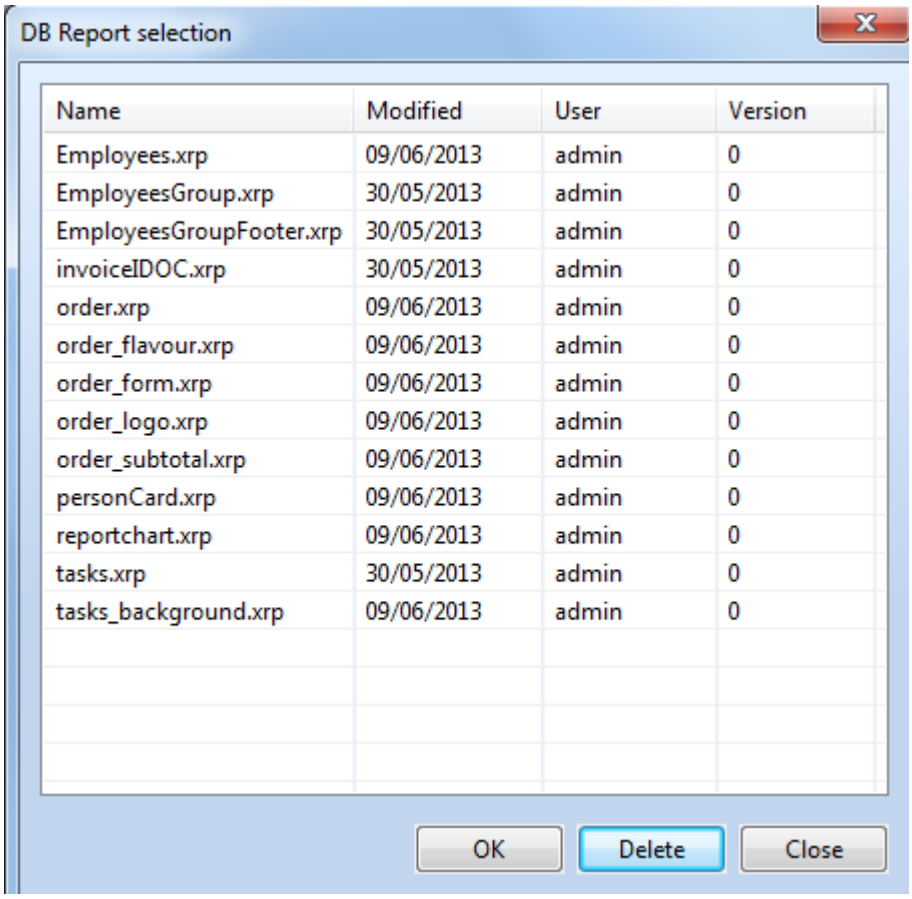
- The xre file will be saved in the database
- together with the currently selected schema (in the report`s settings page)
- XML test files must be saved separately using the context menu:



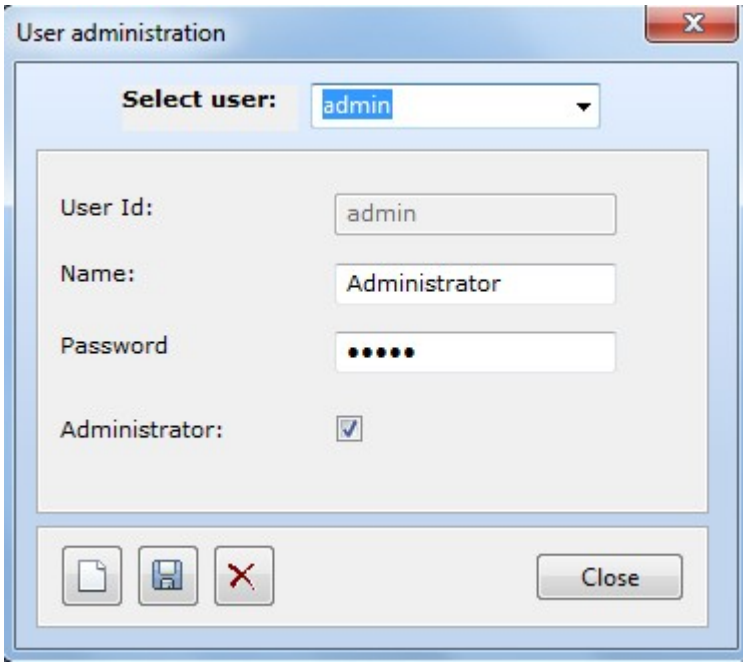
Note files saved to the database have the database icon, the update of the icon is however not immediate :



The button should be used to retrieve projects in the database , the following screenshot shows the open report dialog:



Finally the user administration button can be used in a team work environment

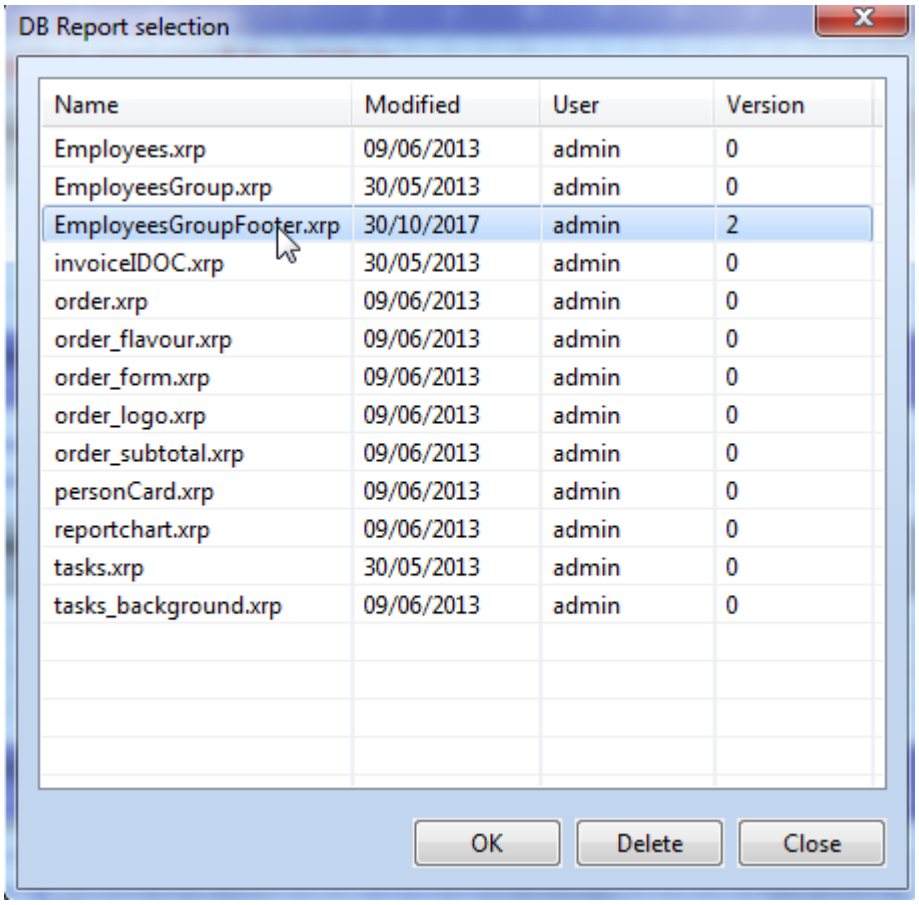


Versioning system

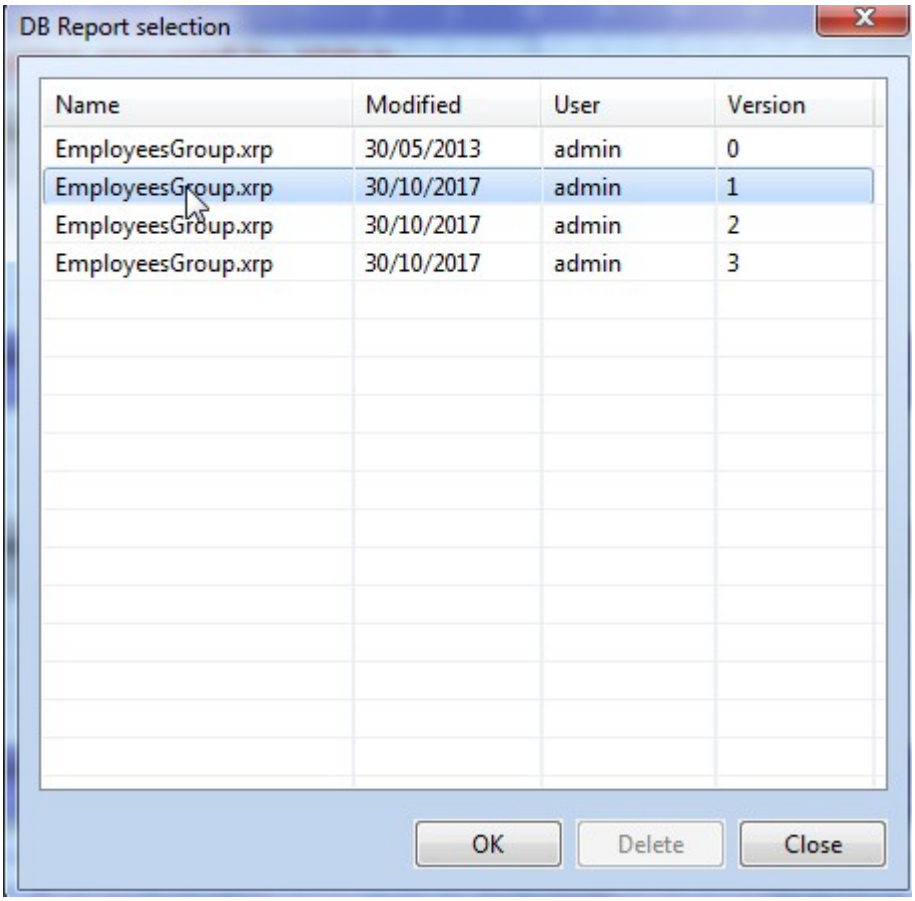
The designer can keep track of the different versions of your templates by using a versioning system. Furthermore it keeps track of the modification date and the user that made the changes.

- You use this button to save your current template to the database and overwrite the current version.
- You use this other button to save the template to database creating a new version.

Whenever you want to open a template, the system always proposes the last version. As in this example it proposes version 2 for the EmployeesGroupFooter template.



You can however open an previous version **of the currently open template** using the button . In the example below you would be opening version 1

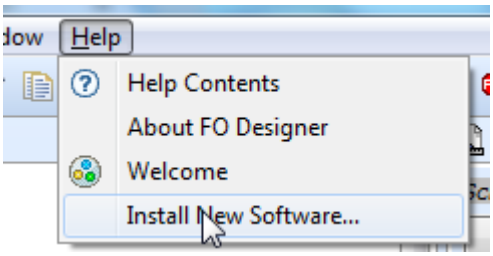


SVN plugin

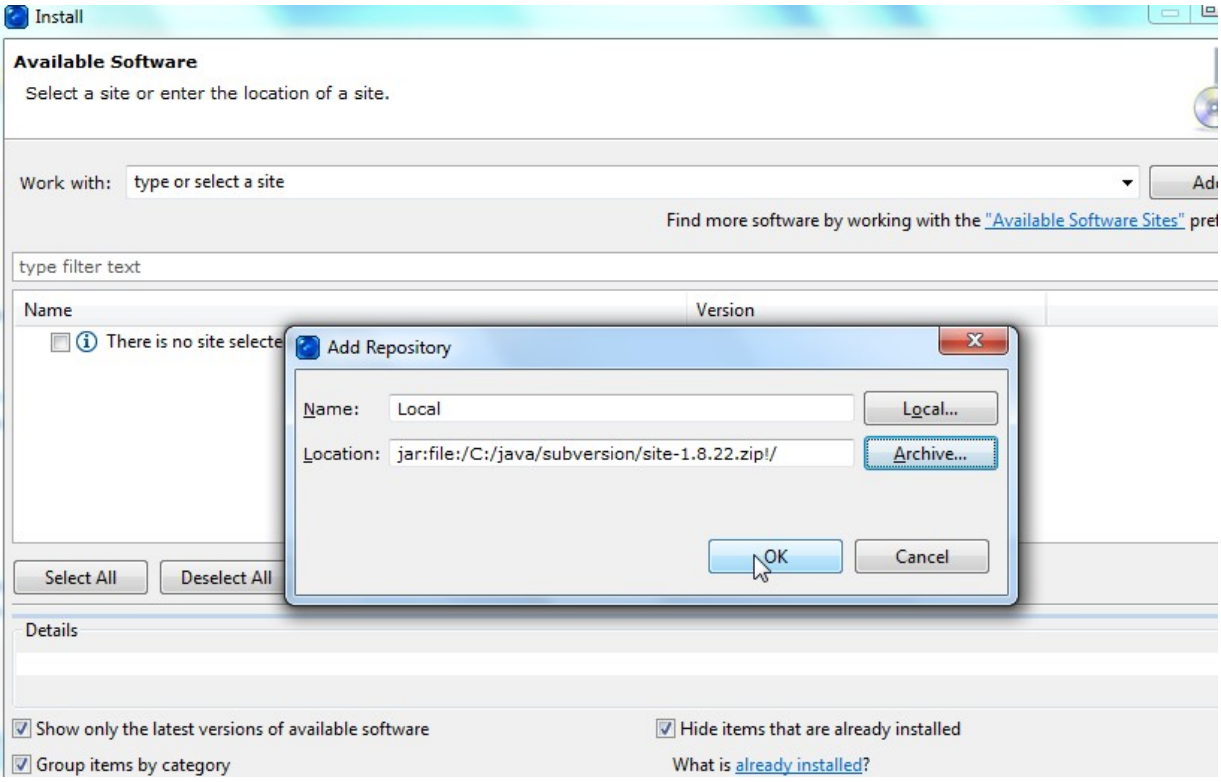
A more powerful alternative to our central database is using Apache Subversion (https://subversion.apache.org/) which is a generic tool for code management. If your enterprise is already using subversion, this section will show how to install the plugin for the FO Designer.

1. Download the file site-1.8.22.zip from https://dl.bintray.com/subclipse/archive/release/zipped/

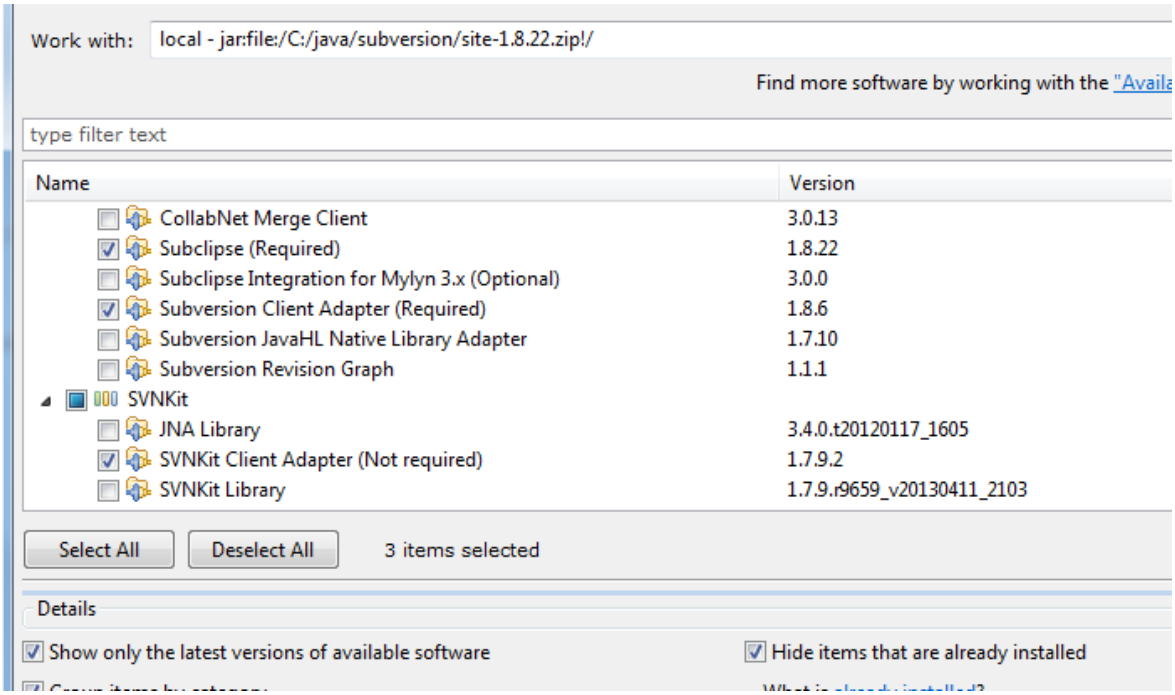
2. Select Help→Install new software



3. Select „Add“ and create a new repository. The location should be the downloaded zip file (use the archive button)

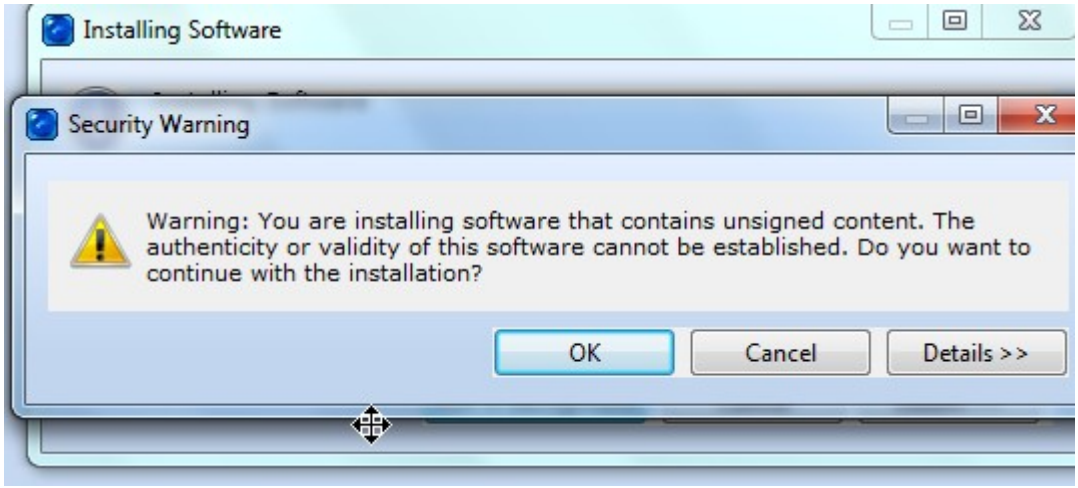


4. Make sure you select these items:

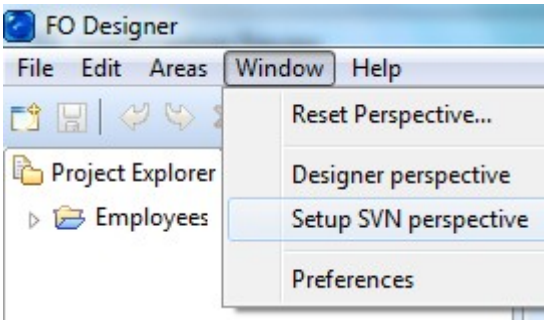


5. Click *next*, accept the license and finish

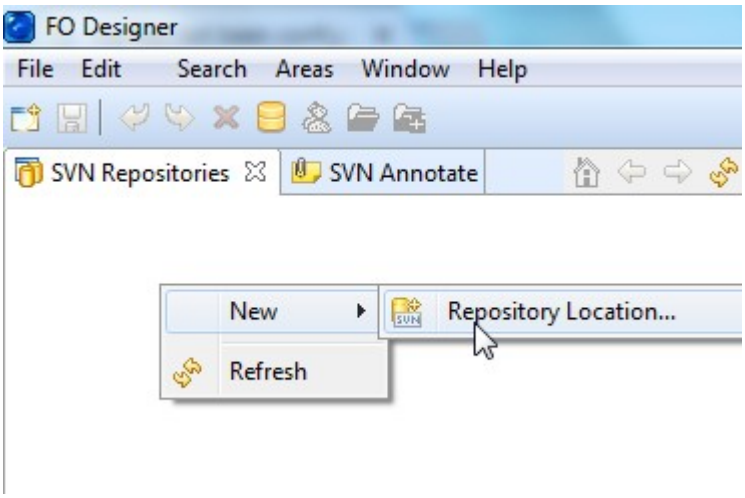
6. Accept also the security warning

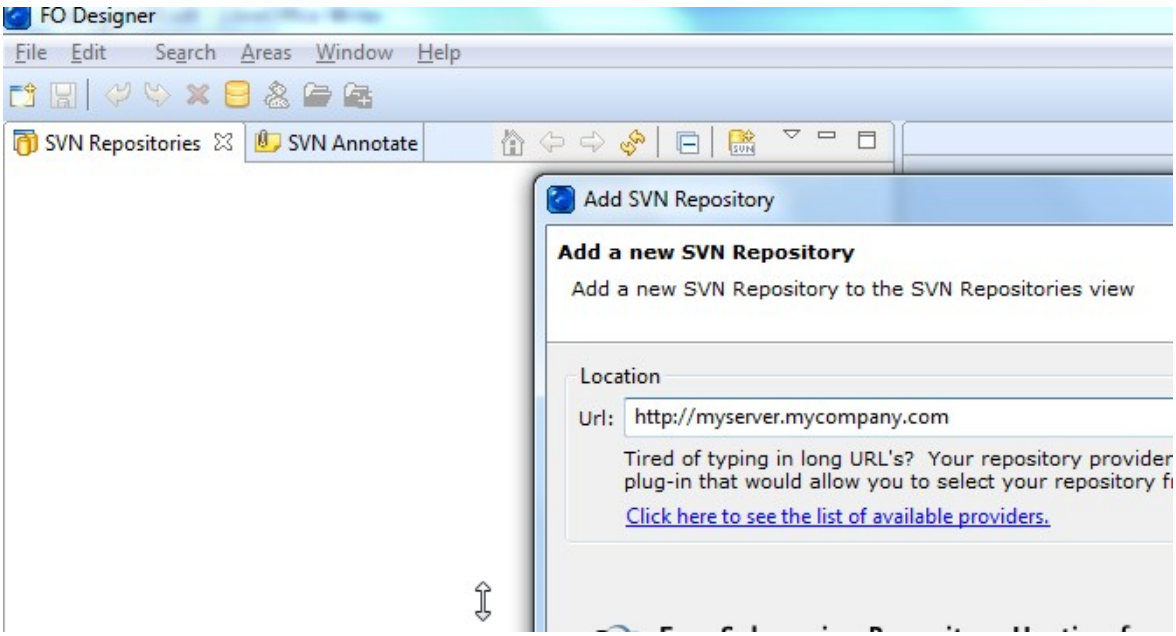


7. After restart there will be a new item in the windows menu:



8. In the SVN setup perspective the link to the central repository can be added (Contact your system administrator for details).



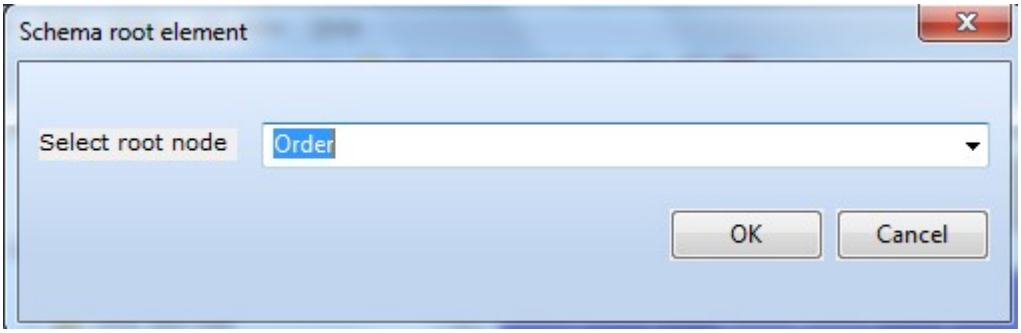


Once added all SVN functionalities will be available. Some tutorials and videos are available under http://www.eclipse.org/subversive/documentation/gettingStarted.php

The XPath editor

The xpath editor is used for selecting XML nodes in several object properties in the designer. The XPath editor does not only allow the selection of an XML element but it also lists a set of functions that can be applied to the XML elements.

When you use a schema file for the firs time, the system will ask you to select the root element are below:

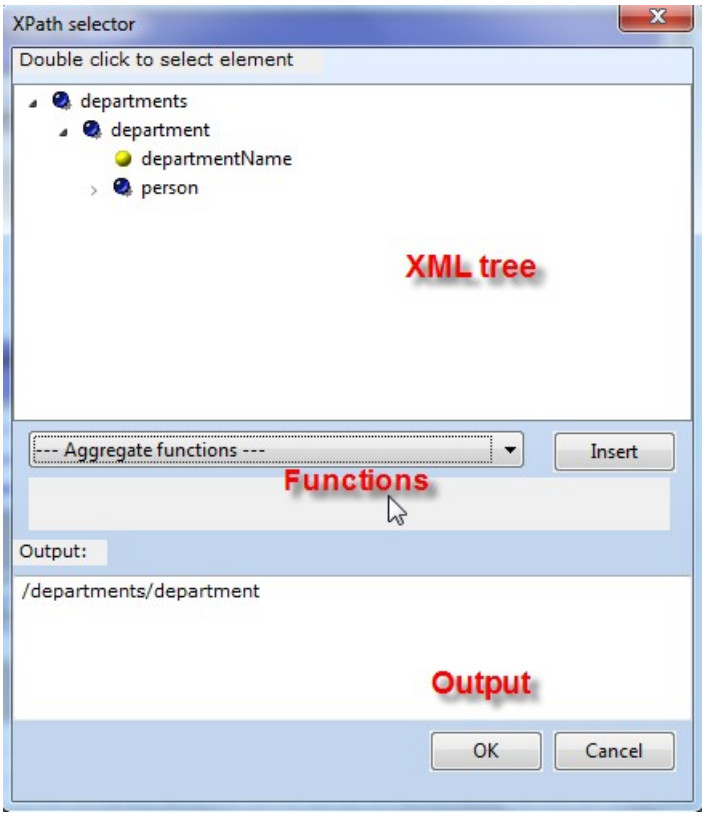


The Editor has 3 parts:

1. on the top you can select the XML node, double click on it to select it and add it to the output.
2. In the middle of the window you can select one of the available functions. The function will be added to the output only if you click the *Insert* button.

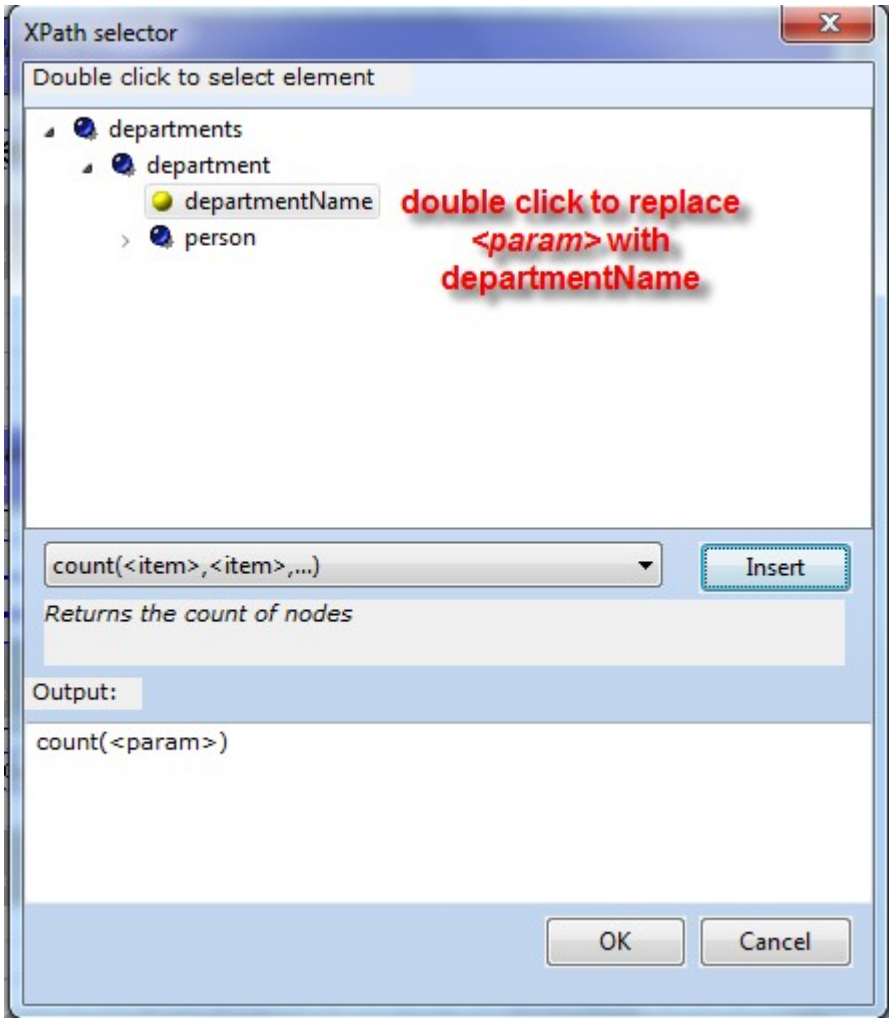
   Note: Most functions are standard XSL-FO and Apache FOP functions, other however are specific to the J4L Designer, those are the ones with the prefix *j4lext*. If you use any of these *j4lext* functions you must also use our runtime module to create the PDF files. It is also possible to **create your own functions, see FAQ section**.

3. At the bottom the output Xpath will be displayed.



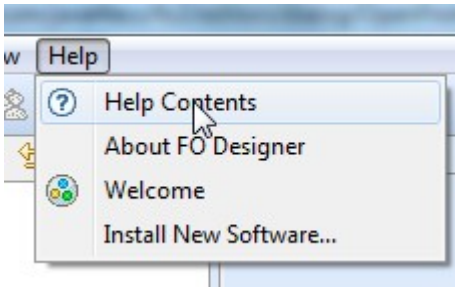
When you select an XML node or you insert a function, there are 3 possible situations:

1. If the output is empty, the node or function will be added to the output field.
2. If the output is not empty but there is at least one <param> string in the output, the left most <param> will be replaced with the selected node or function.
3. If no <param> value exists, the editor will ask you if you want to overwrite the content of the output field.

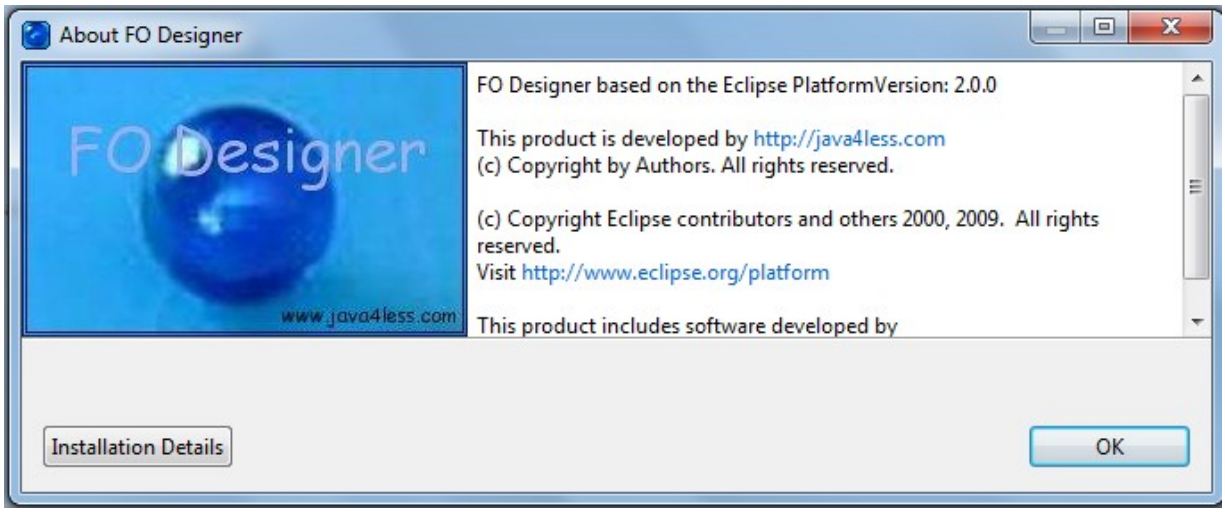


The help system

The Help menu can be used for displaying the products documentation (Help Contents) or product version (About FO Designer item)



Product version and license:

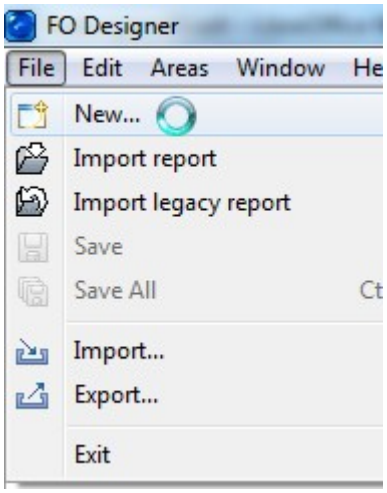


3. Working with the user interface

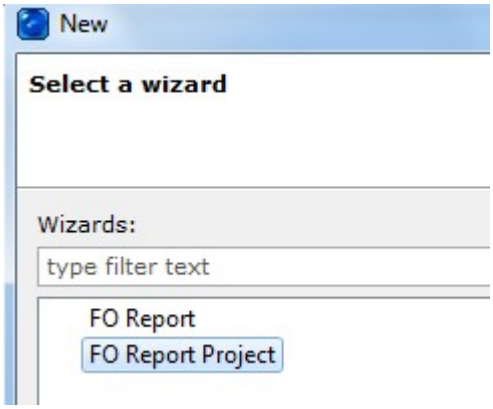
This section highlights some common operations performed while developing a template:

Creating, deleting, importing and exporting

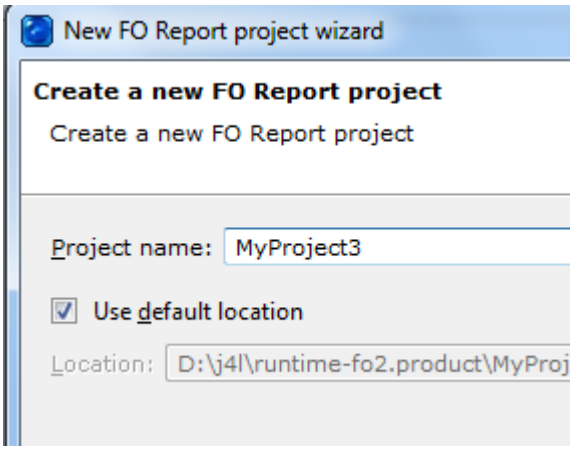
In the *File* menu you can find all the operations to be performed on projects, these are:



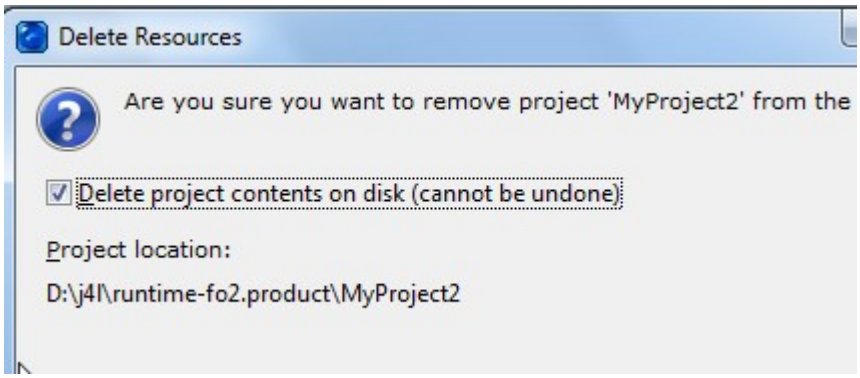
1. Create a new project. Use the „*New*..“ item. Select new *FO Report Project* and follow the steps in the wizard.



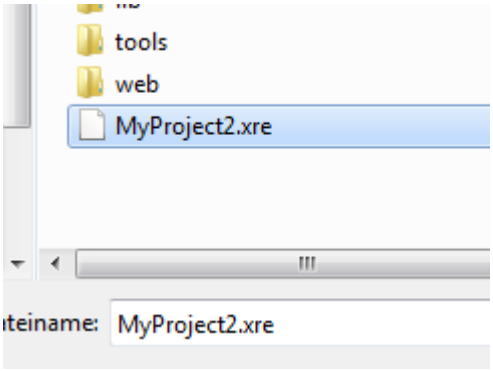
Enter a project name and press finish



2. Deleting a project: press *DEL* on the project folder, make sure you select the „*Delete project contents on disk*“ checkbox



3. Importing a report. Use this to import single report files. A new project will be created using the file name.

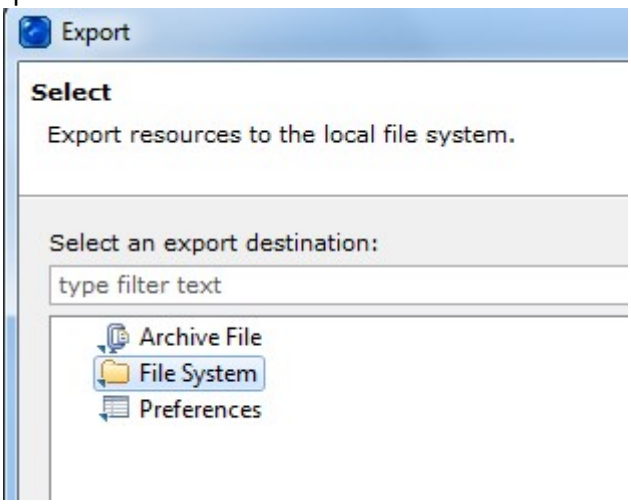


4. Import a legacy report. These are ***.xrp*** files used in FO Designer 1.*. Same as above but using legacy files instead.

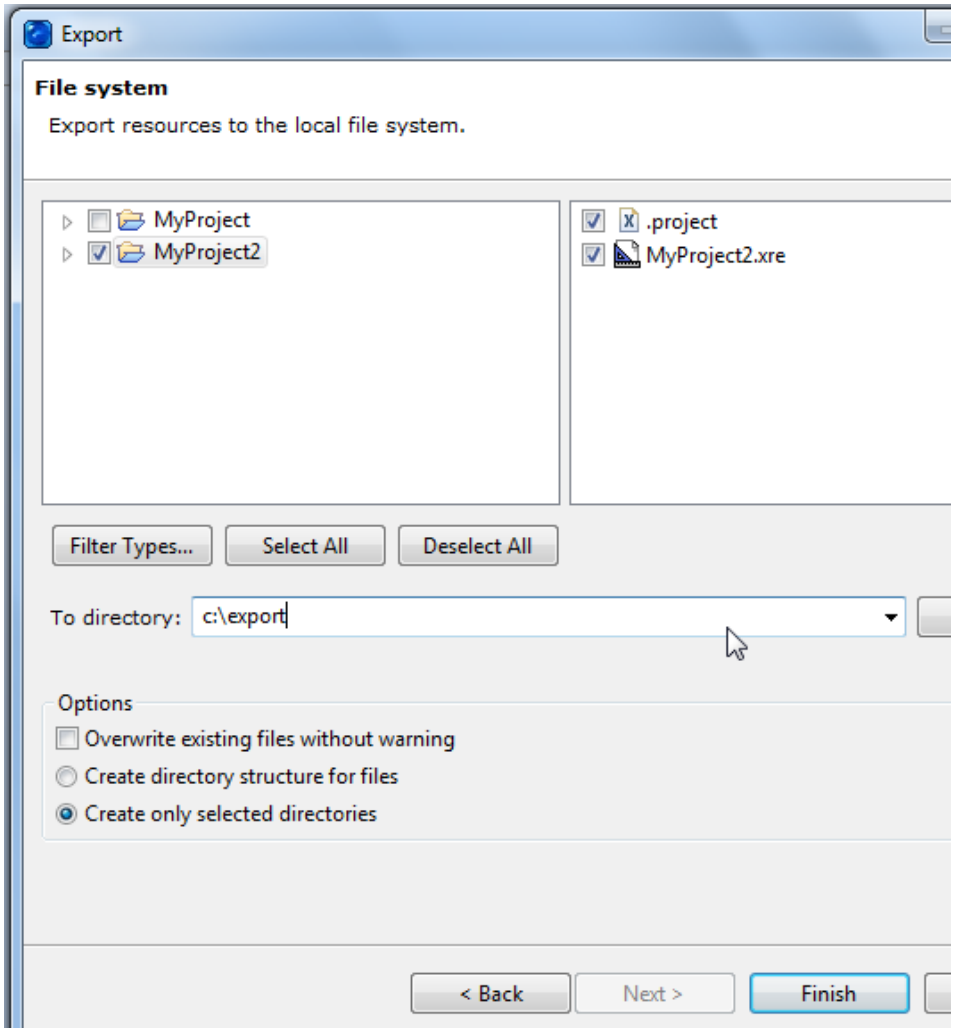
5. Save reports to the file system

6. Export and import complete projects (including XML and schema files).

   Select *export* and *File system.* If you select *„Archive File“* the project will be exported to a zip file.



Last, select the objects to be exported and *press finish*:



Operations on files

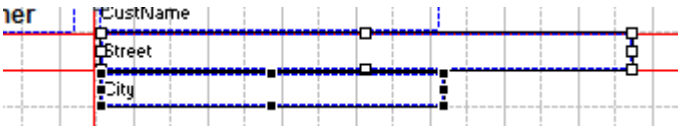
You may use the context menu to delete, copy or rename files or whole projects however make sure that:

- If you copy a whole project make sure you rename the xre to the same name as the project name. Basically **make sure the xre file has always the same name as the project**.
- You many have **only one xre file per project**. If you need to copy the xre file, copy the whole project instead.

Selecting objects

Objects in the areas can be selected in 3 ways:

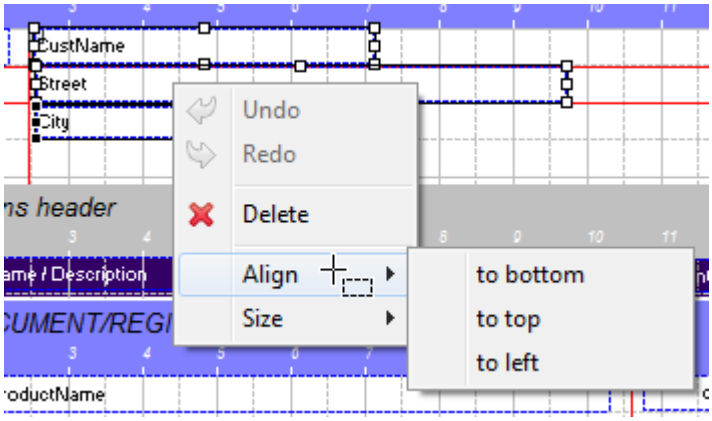
- Single object selection: clicking on the object (the properties of the object will be shown automatically
- Multiple object selection: click on several objects while holding the SHIFT key pressed.



The selected object can then be resized or moved all together using the mouse.

Operation on objects

There are other operations you can perform on the selected object by using the context menu (mouse‘s right click)

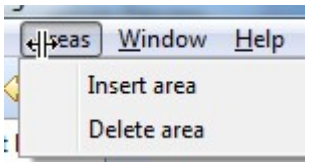


The operations are not explained here in detail since they are self-explanatory and quite standard in many windows programs. Additionally to the operations listed the following operations can be performed by using the keyboard:

- copy: *control key + C*
- cut: *control key + X*
- paste: *control key + V*
- delete*: <del> key*

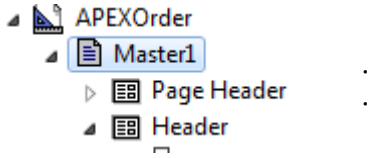
Operations on areas

The *areas* item in the main menu allows the following operations:

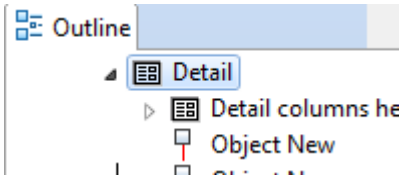


Both operations work using the Outline view.

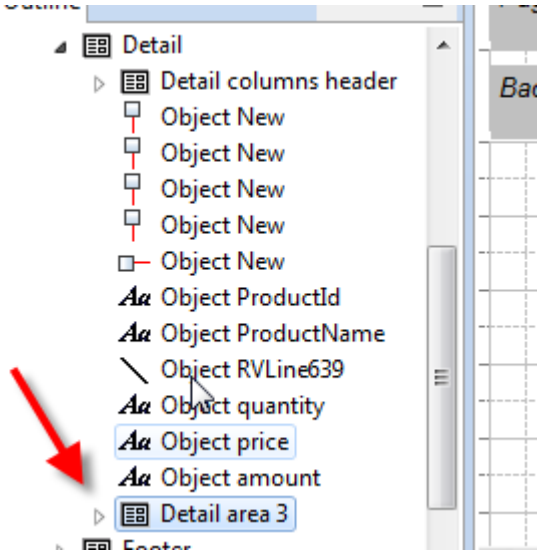
- **Inserting an first level area:** select the master on the outline view. The new area will be added as first level area.



- **Inserting an area below an existing area:** select the parent area in the outline, for example „Detail“

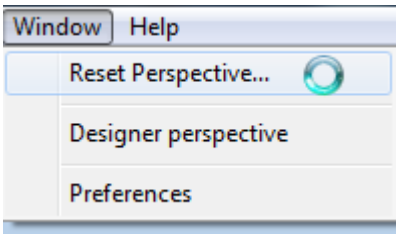


the new area will be added below (as a subarea). In this example „Detail area 3“ has been added below „Detail“

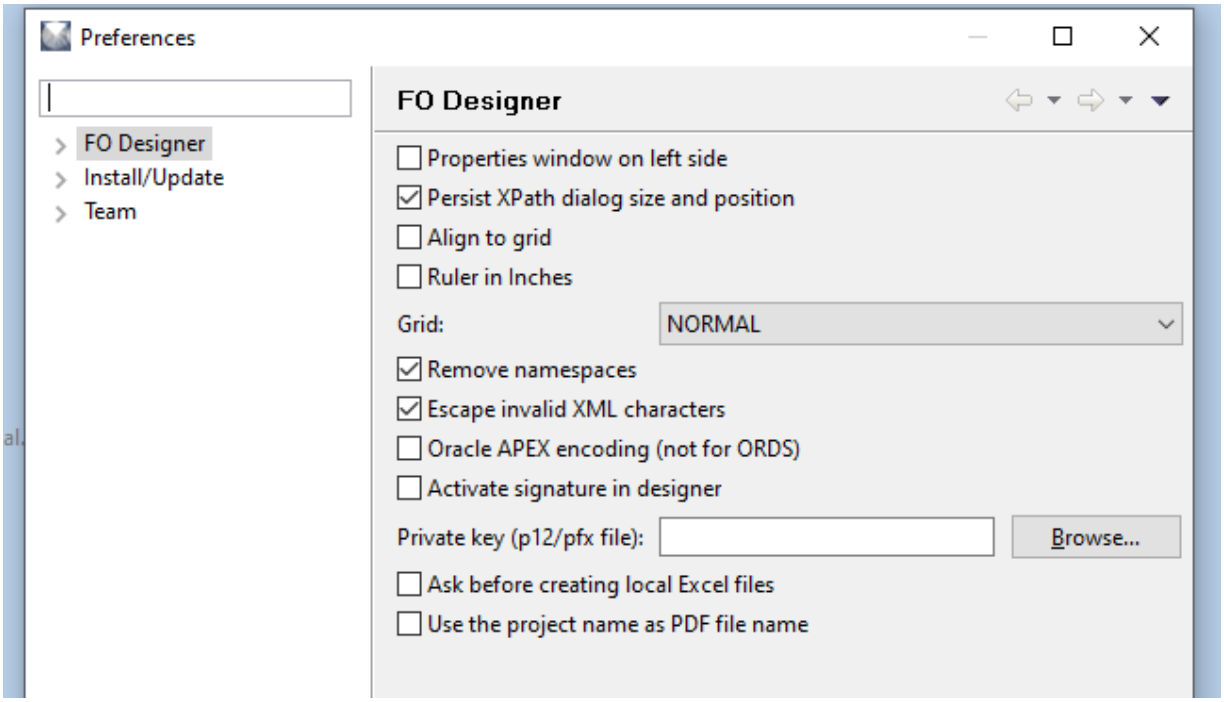


The preferences dialog

In the windows menu there are 3 items (4 if you install the SVN plugin):



- the reset perspective will replace all the view in the original position. This is useful if you have closed or moved some views by mistake and want to return to the original window layout.
- The Designer perspective will open the editor in case the Welcome or SVN setup perspectives are open.
- Preferences: in this dialog the settings described below are available



The settings are:

- **Properties windows on left side**: this is useful if you are working with small screens (laptop). In this case the properties of the object will be shown on the same position as the project explorer (as a new tab), saving therefore display space.
- **Persist XPath dialog size and position:** if marked the desinger will remember previous size and position of the dialog.
- **Align to grid**: align always new or moved object to the grid
- **Grid**: normal or fine
- **XML namespace processing setup.** Remove namespaces from XML test file before generting the PDF
- **Oracle APEX encoding setup:** select this option if you are working with Oracle APEX. Do not use however this option for the Oracle APEX Listener (ORDS) as print server.
- **Activate digital signature.** See section below.
- **Ask before creating local files:** self explaning
- **Use project name as PDF file name:** if not setm the designer will always generate a report.pdf file
- **Local history page, Limit history size:** set a limit on the number of past reports (history) to be kept.
- **Local history page, Days to keep files:** older copies will be deleted.
- **Local history page, Maximum entries per file:** maximum number of past reports to be kept.
- **Local history page, Maximum file size:** larger reports will not be kept.
- **Mail Page, Smtp Host and port** (In the Mail subpage): Setup for sending the PDF reports from the designer (see cloud server section)
- **Mail Page, From email:** sender email
- **Mail Page, User and password:** smtp server login data
- **Server Page, DB Server and DB Server port (**in the Server subpage).this setup is required when working in teams and there is a central J4L FOP Server together with a central database.
- **Server Page, Printer server type:** setup for remote PDF generation. Select one of the supported servers. Make sure you select the right server since when exporting the xsl-fo file the designer will check if the functionalities in the report are compatible with your target server.
- **Server Page, Printer server URL**:
  for Oracle Apex ORDS 5. use*: *http://host:port/ords/_/fop2pdf* or *http://host:port/apex/_/fop2pdf*
  for J4L FOP Server use: http://host:port/J4LFOPServer/Apex
  for J4L Could Server use: https://apex-reports/print/trial
-

  Note: for Apex ORDS to work you need to navigate to the /apex or /ords directory , locate the *defaults.xml* file and add the following code:

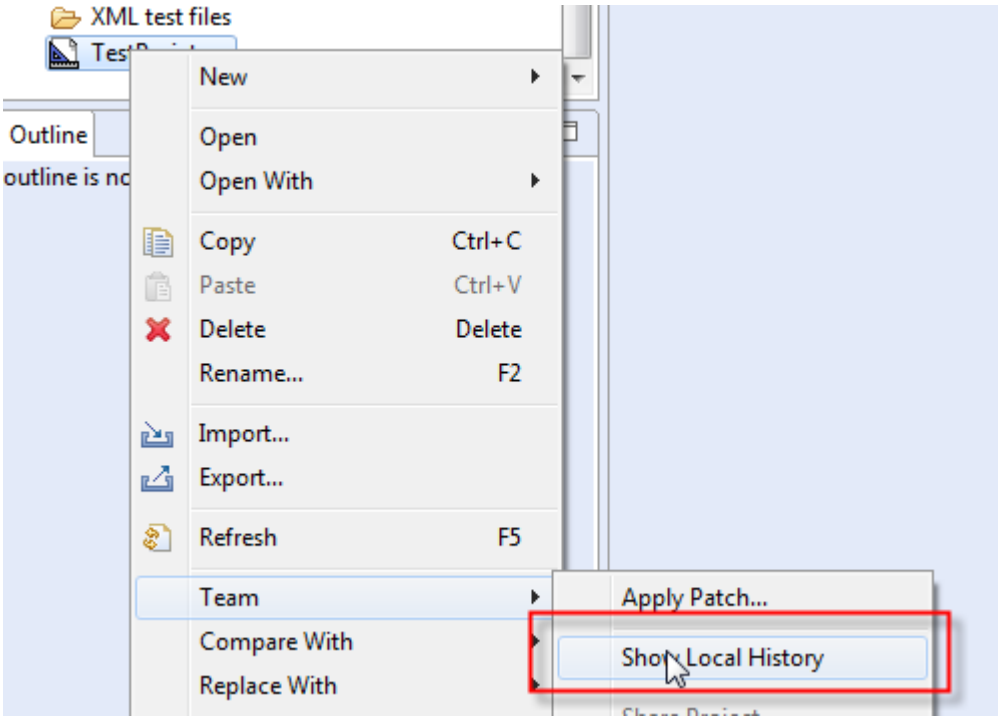
  <entry key="misc.enableOldFOP">true</entry>

- **Server Page, Connection timeout (secs):** timeout when connecting to remote servers for FOP generation
- **Server Page, Response timeout (secs):** timeout for the response from the remote FOP server.
- **Server Page, Ask before using the cloud server for PDF:** self-explaining
- **Server Page, Ask before using the cloud server for Excel:** self-explaining

The local history

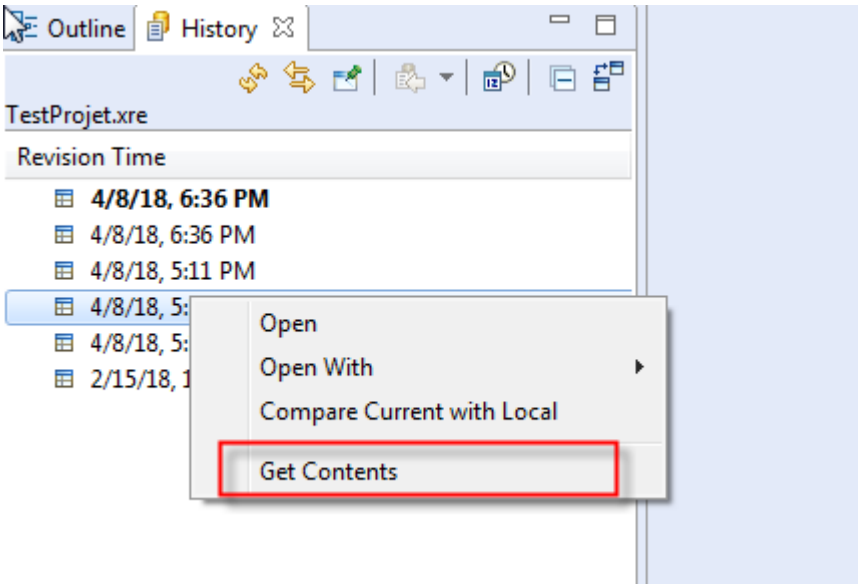
The designer will keep a copy of the previous report each time your click on the save button. The settings in the preferences dialog will determine how many and how long the copies will be kept.

In order to recover a previous version of the report, **first close the report** then select the „*Show local history*“ in the context menu.



The local history window will open. You can use the context menu „*Get Contents*“ to **recover a previous version (make it the new current version) and replace the current one**.

After getting a previous version you can open the report as usual, you will be now working with a copy of a past version.



4. Structure of the template

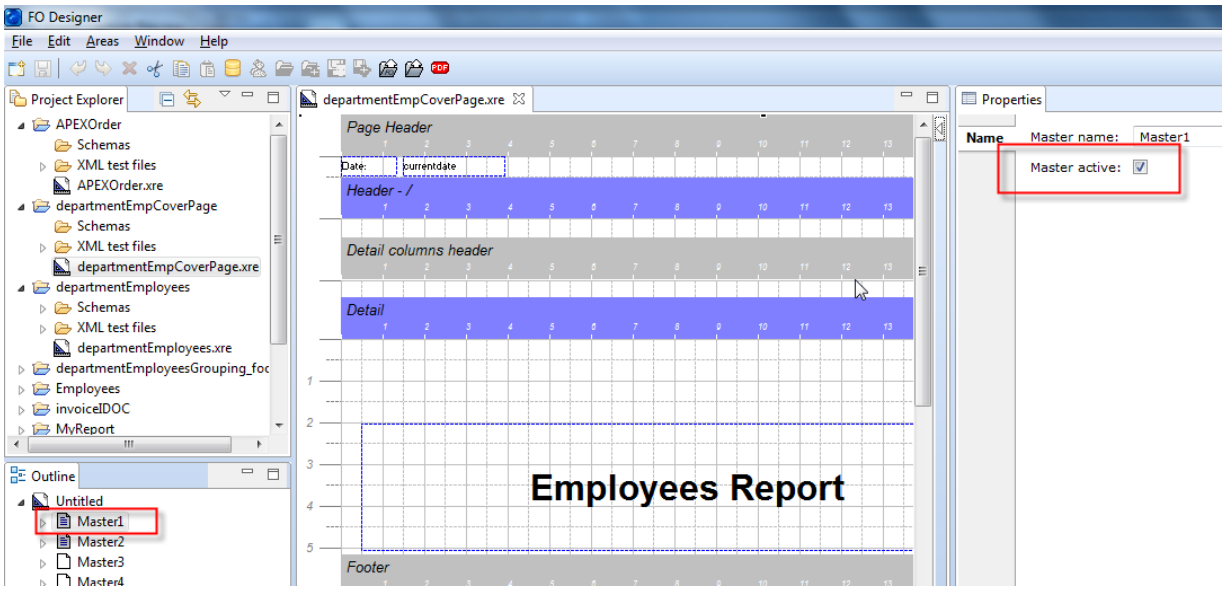
This section contains very valuable information to understand how the XML to PDF conversion takes place. First of all we will describe when and how areas are generated, then we will explain how the layout of the fields in the areas occur and last we will explain the meaning of the properties of each object involved in the template.

Page masters

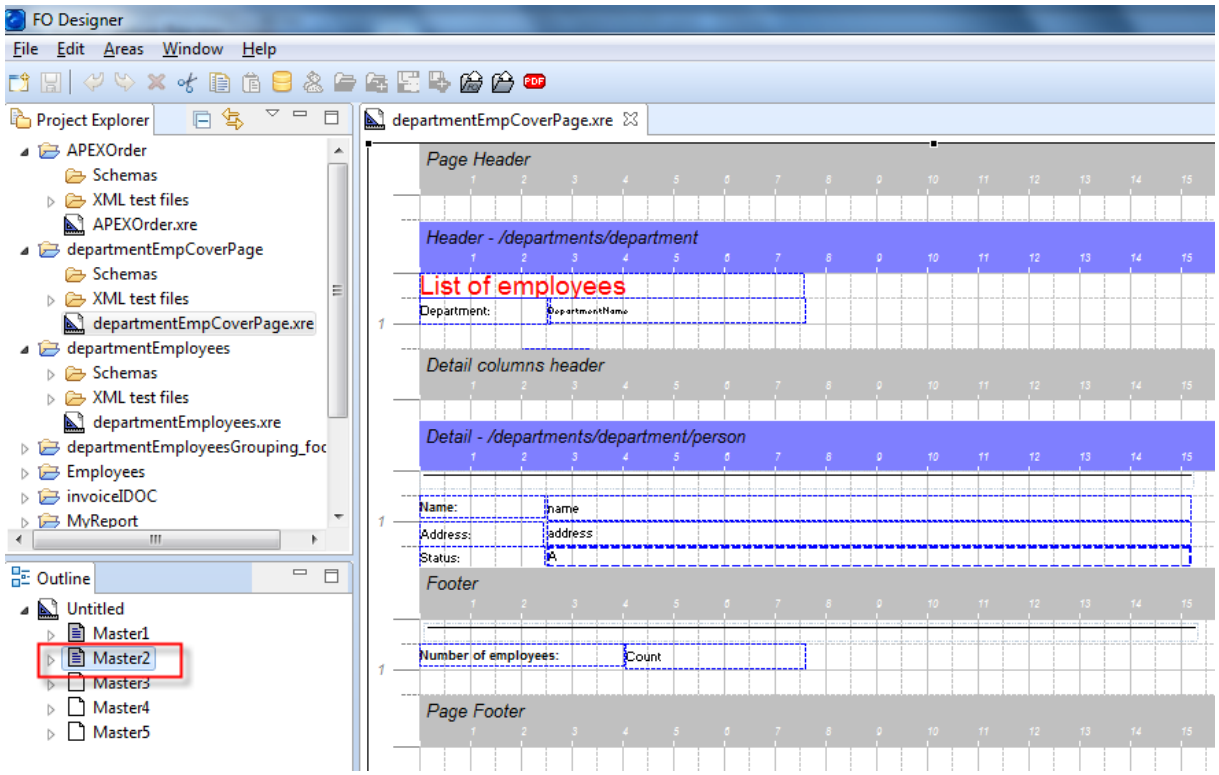
In most cases your templates will have only one page master, that is, all pages look the same, they just contain different data. A page master describes the layout of a type of page and the areas it contains, including page header and page footer. What to do however if you want to have no page header and no page footer in the first page? in that case the first page would be using a different page master. This is for example the case when you want to have a cover page followed by the regular pages.

The example located in *departmentEmpCoverPage.xre* contains such a template.

Each template can have up to 5 page masters that can be selected from the Outline view. **Note the page master 1 must always be active but master 2 to 5 must be explicitly be activated,** see "Master active" property in the screenshot below.



In this example the first master contains just the title of the report as a cover page, while the master 2 contains the layout of the pages that contains the employee data.



Note the XML schema and test data are common to all page masters. All page masters share the same source data (XML data), they just create different pages with different layouts.

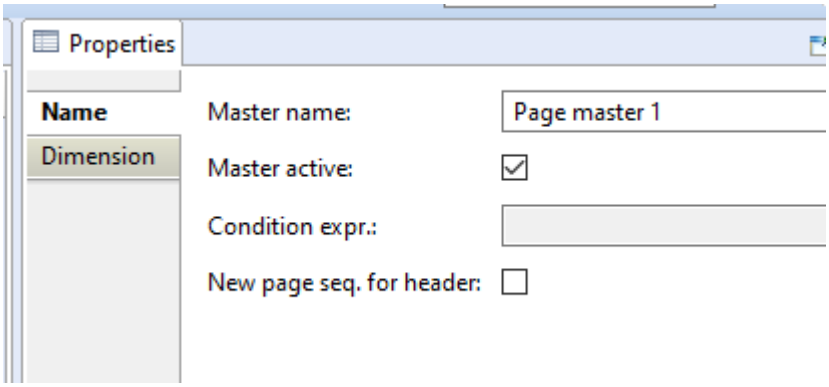
If you run the provided template and generate the PDF, you will see the output is a PDF file with 3 pages, where the first page is the report cover page.

Properties of the page master

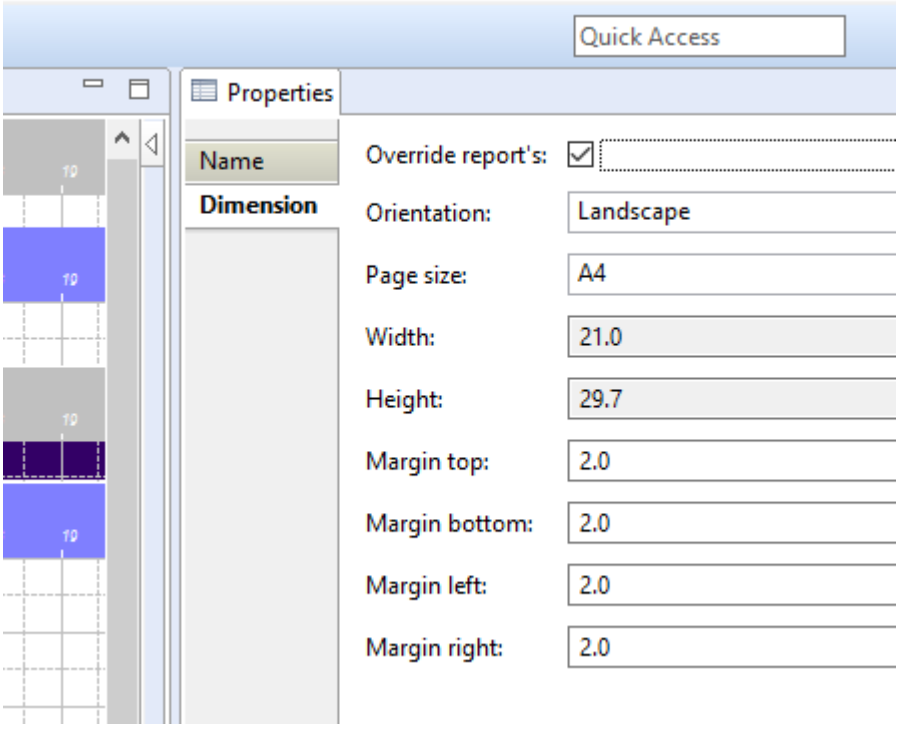
Each page master has the follwing properties

In the name section

- **the name**
- **the activation flag**
- **the condition**. This is an Xpath expresion to make masters optional
- **New page seq. for header.** If set it will create a new page-sequence element in the XSL-FO for each element in the Report's header node. This results on the page number being reset to 1. This is useful to create batch of documents within one PDF file, where each document has a page number starting with 1.



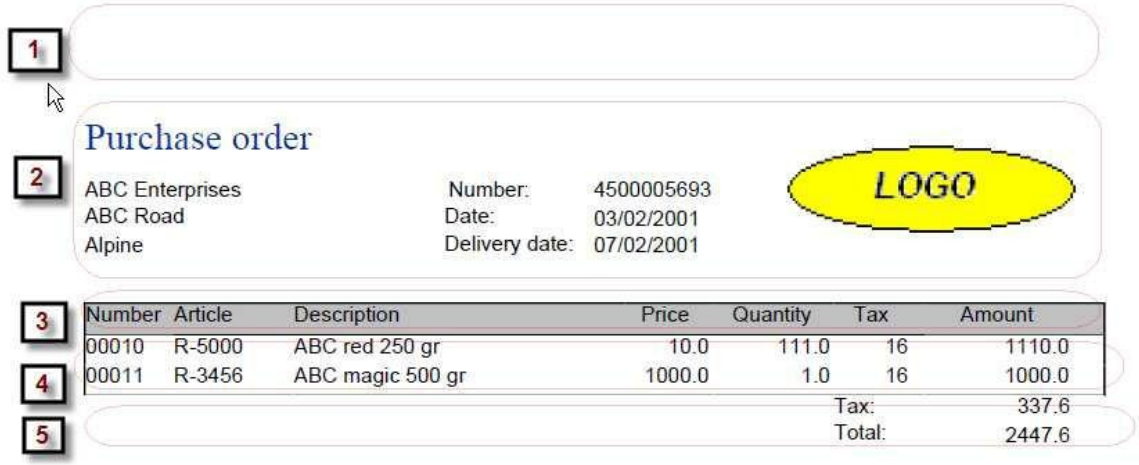
In the Dimensions section you can define a different page size and orientation for the master which overrides the report settings in the report properties.



Areas

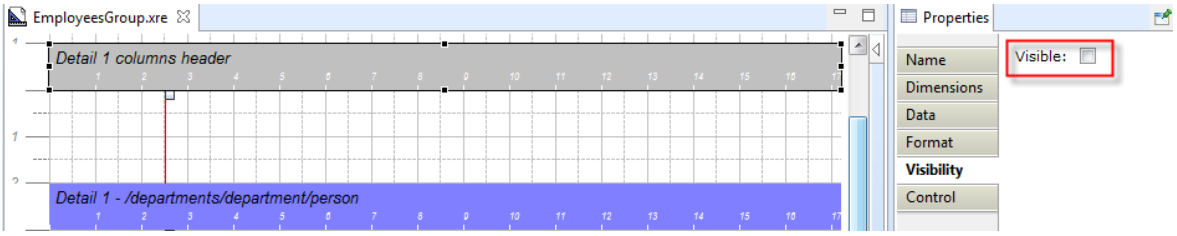
The following screenshot shows a PDF file which can be split in 5 areas:

1. The header of the page (in this case it is empty).
2. The header of the document (in this case the header of a purchase order).
3. The header of the detail area, it contain the labels of the columns in the detail area.
4. The detail area, in this case 2 repetitions (lines) of the detail area.
5. The footer of the document (commonly used for showing totals).
6. Additionally there is a page footer (not shown in this screenshot) which can contain, for example, the page number.

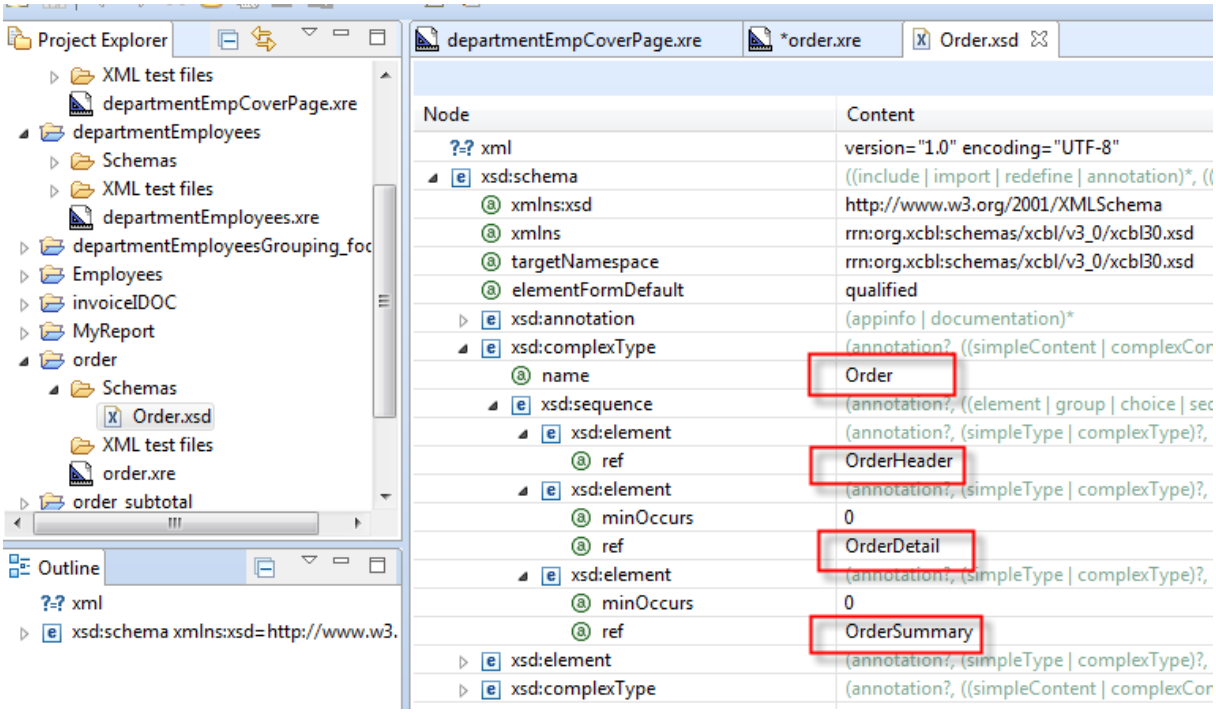


In the same way a normal business document contains this kind of areas the template you create in the FO designer contains also the following types of area:

- One template **header and footer**. Each PDF file will normally contain only one document header and one document footer, however it is possible to have several headers in one
- PDF file, for example if you are printing several invoices in one PDF file.
- One **page header and footer**. These areas are printer in each page.
- **detail** areas (optional).
  - Area can have subareas. Each time an area is generated (added to the PDF file), the dependent (sub)areas will be generated. The template Header is the first area generated , afterwards all first level areas (those that are right below the master in the outline view)
  - the **XML node** property specifies when the area must be generated. The XML node property contains an XPath expression which points to an element in the input XML file. The area will be generated if the element exists in the XML file, if several elements exist, several repetitions of the areas will be generated. If the XML Node property is
- left empty, the area will be generated only once.
- Detail areas may have a detail area header (as in area number 3 in the screenshot above), this header area however can be removed from the PDF output (see Visible checkbox below)

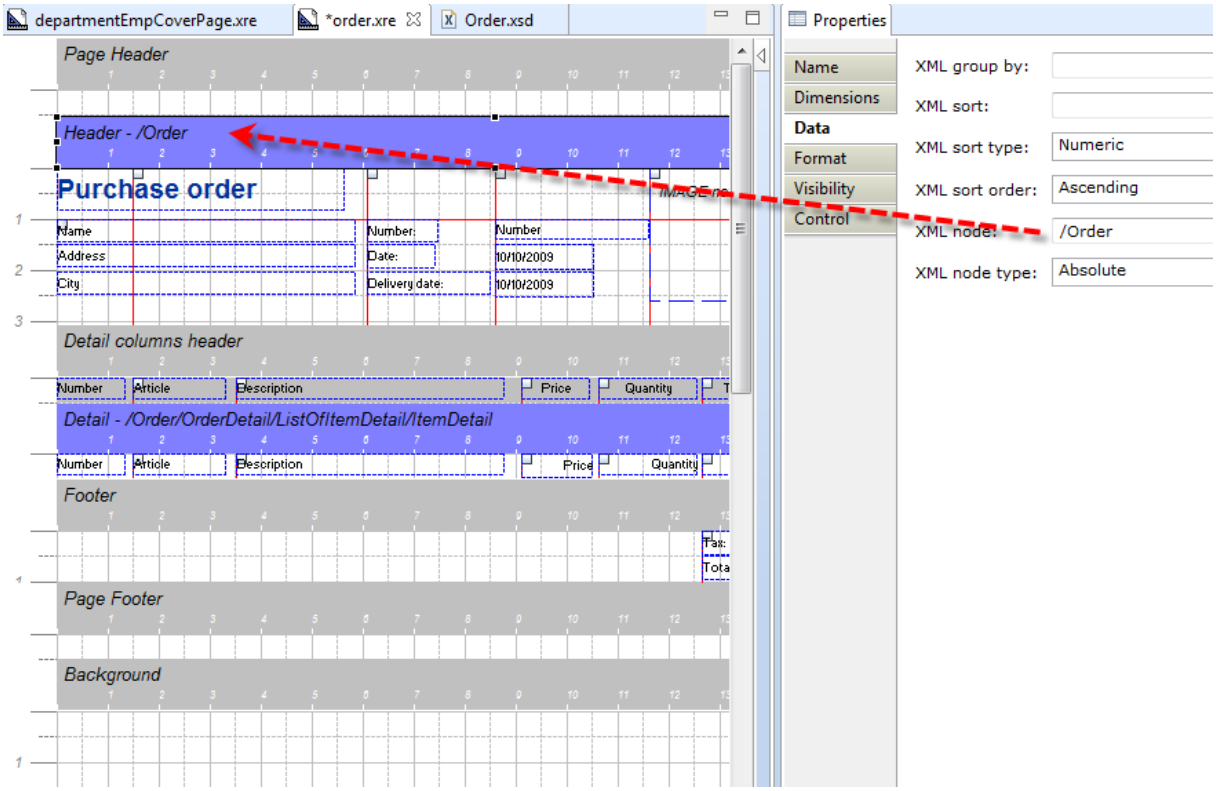


The delivery includes a file called *order.xre* which is the template used to create the above PDF file. This template uses a XML purchase order (using the **xcbl** schema) as input file. The following screenshot provides and overview of the schema:



The root element is called *Order*, under this root element there is a *OrderHeader* element, a *OrderDetail* element and a *OrderSummary* element.

In the following screenshot you see an overview of the template file.

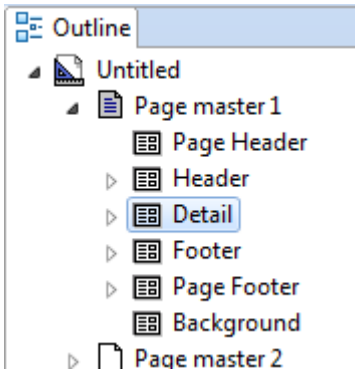


It contains the 6 areas we mentioned above:

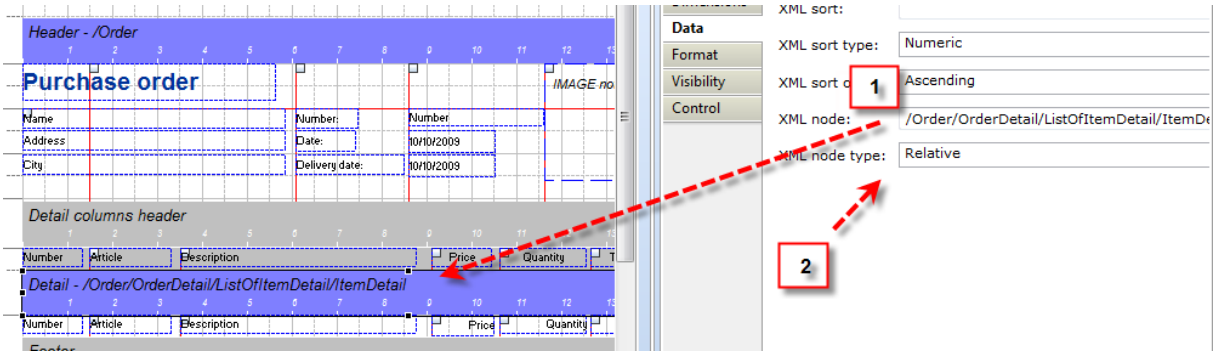
- The document header (contains the purchase order header) and footer (contains the total values)
- The page header (is empty) and footer (contains the page number)
- The detail area (for the articles) and the columns header of the detail area.

When the template is executed:

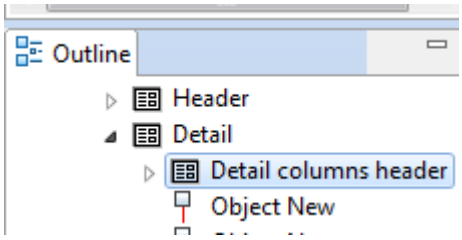
1. The page header and footer will be added to each page
2. The Header area will be generated. If the ***XML node*** property is empty, the area will be generated only once, if it contains a XPath expression as in this case, it will be generated as often as elements are return by the XPath expression. In the example the header will be created once for each */Order* element in the input XML file.
3. For each instance of the header , the dependent areas will be generated. These will be all first level areas (the ones right below the master in the outline view). In this case the Detail area.

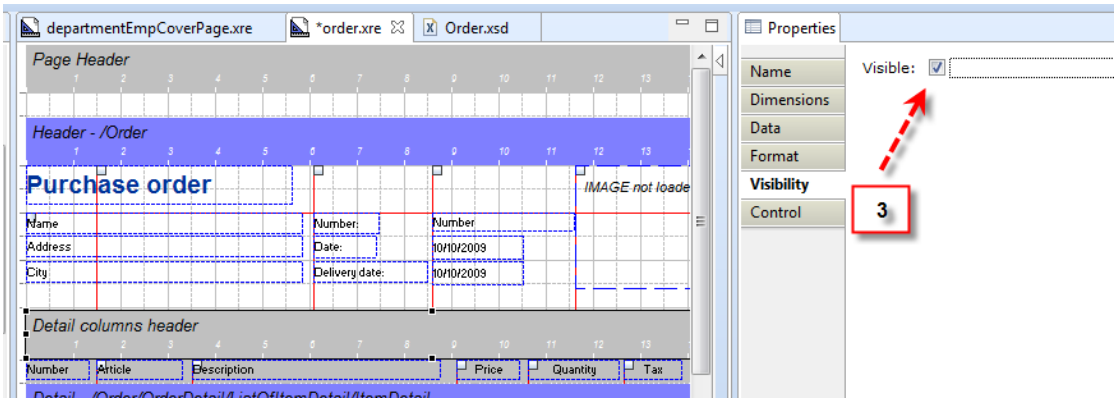


so we know the detail area will be generated after the header area , but how often? this on the ***XML node*** property (see arrow 1 in the screenshot). In this example we will create one repetition (line of the detail area) for each *ItemDetail* node in the input XML file. In other words, we will create one detail line for each item in the purchase order.

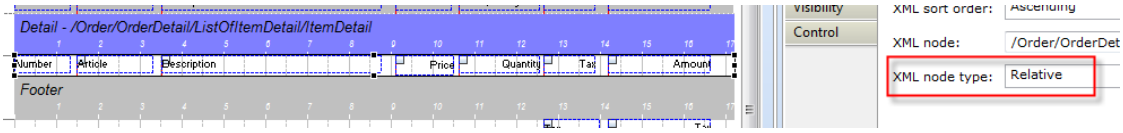


Last, as you see in the ***Visible*** property in the „Detail Columns header“ this header which will be generated before the detail area and contain the labels of the columns.



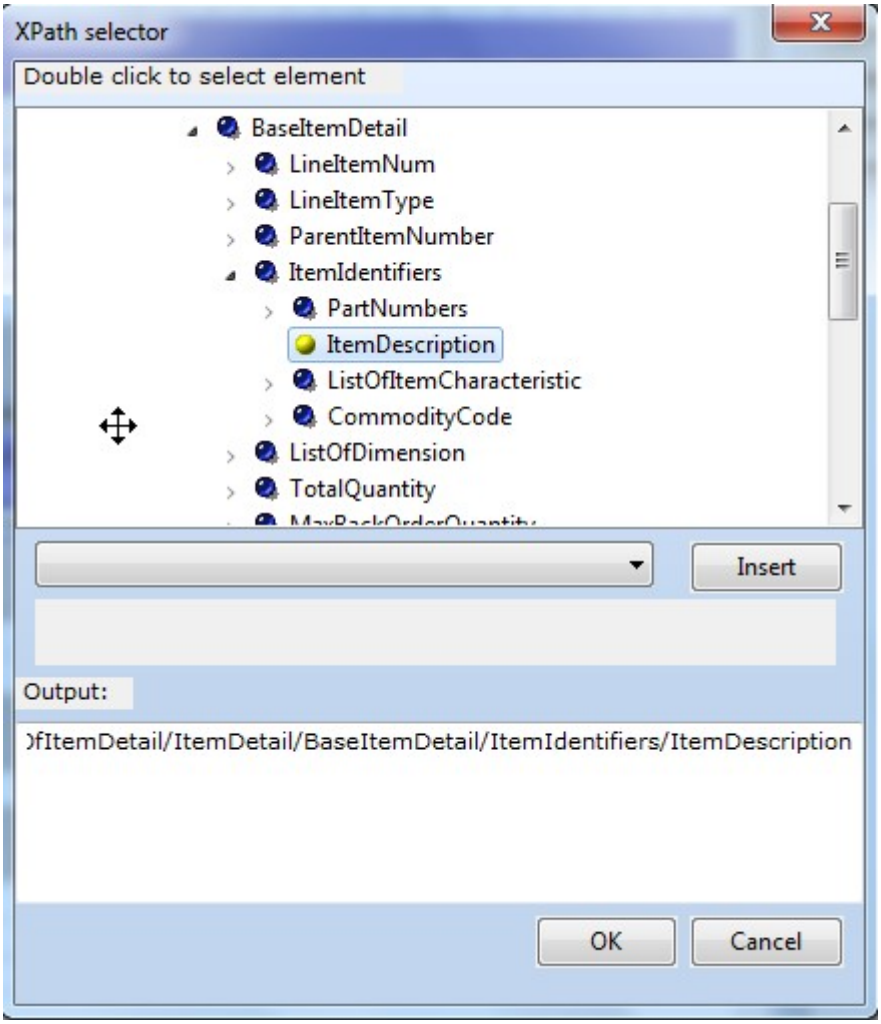


In this example each XML input file can contain only 1 purchase order, however let's suppose it does contain two */Order* elements. In this case we would get two instances of the header area and of course we want the items belonging to the first order to be selected when the first header has been generated and the items belonging to the second order to be selected after the second header has been generated. That is why the ***XML node type*** property of the detail area has been set to *relative.*



That means, select only the *ItemDetail* elements which belong to the */Order* element being generated. If you for example set that value to *absolute*, you would be selecting all items in the XML file, not only those belonging to the current order.

The same logic applies to the individual fields. Each field has a ***XPath*** property and ***XPath type*** property:

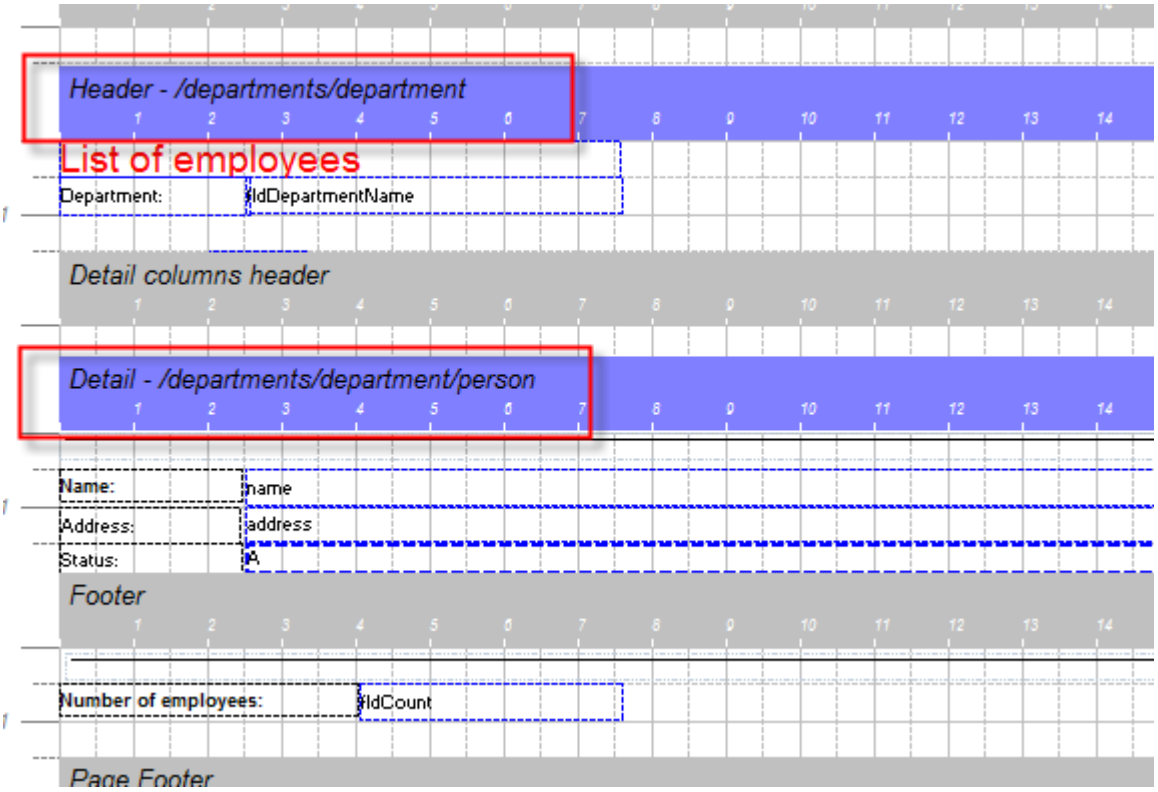


In the screenshot above you can see the value for the description field will be selected from the *ItemDescription* element which is located somewhere below the *ItemDetail* element (which is the *XML Node* of the area). Since we want to select the item description of the current *ItemDetail* node, we set the *XPath type* to relative.

How to analyze the report execution

The magnifying glass in the toolbar will trigger the creation of the PDF file and will trace the execution of the XSLT engine.

Let's run this feature for the *departmentEmployees.xre* report. This report will create a new report for each department (note the */departments/department* node in the Header area), and will list all the employees of the department (*/departments/department/person* node selection).



The input test file is:

departmentEmployees.xml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<departments>
<department>
  <departmentName>R&amp;D</departmentName>
  <person>..
  <person>..
  <person>..
  <person>..
</department>
<department>
  <departmentName>Sales</departmentName>
  <person>..
  <person>..

</department>
</departments>
```

If we click on the PDF button the result will be as below since the test XML file contains 2 departments with 4 and 2 employees each.

List of employees

Department: R&D

Name:	John Schmidt
Address:	Red street 3
Status:	Active
Name:	Paul Bones
Address:	White street 5
Status:	Active
Name:	Mark Mayer
Address:	Blue street 5
Status:	Active
Name:	Janet Black
Address:	Black street 8
Status:	Inactive

Number of employees: 4

List of employees

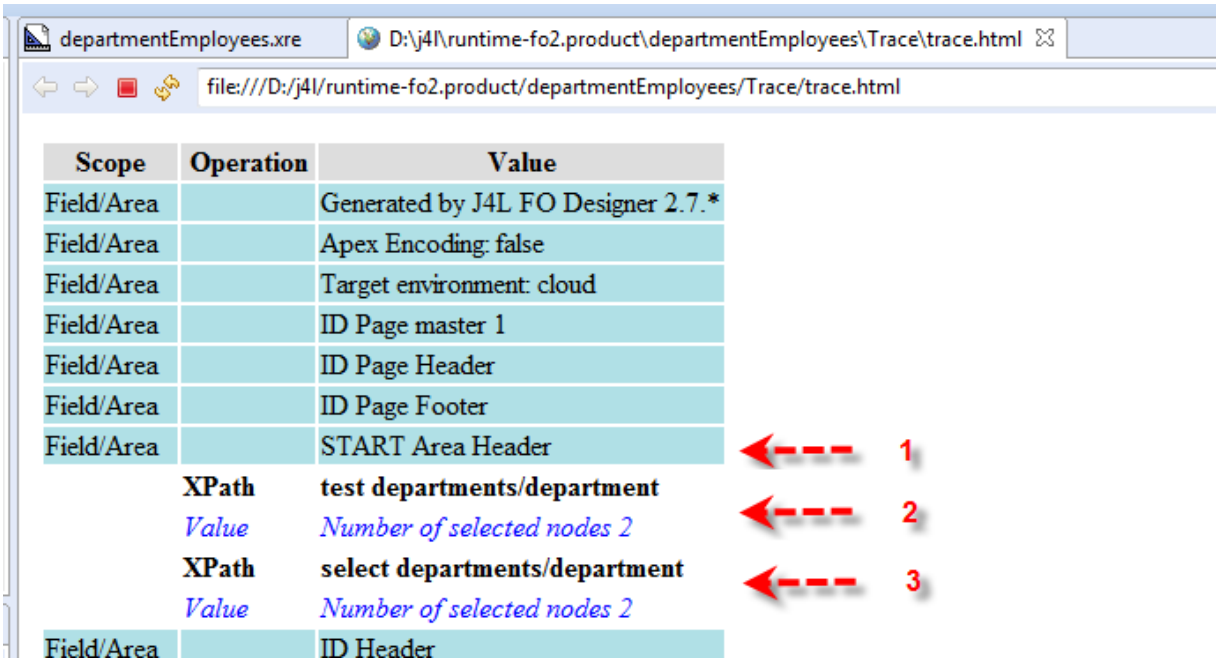
Department: Sales

Name:	Juan Gómez
Address:	Green street 3
Status:	Active
Name:	Juliet Bones
Address:	White street 5
Status:	Active

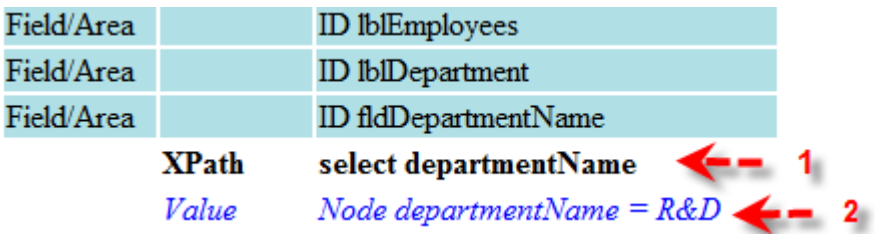
Number of employees: 2

Click now on the Analyze button, a new window will open.

- First the „Area Header“ will be generated (step 1), but only if there are „departments“ in the input XML file (step 2, runs a **test** on Xpath *departments/department)*. Since step 2 determines there are departments, it will select them (step 3), which returns 2 departments. This means for each one of these departments a new report header area will be generated.



- Next you see the **2 constant labels** „lblEmployees“ and „lblDepartment“ (the constant value itself is not visible in the trace) and the **one variable field** „fldDepartment“ begin generated. In step 1 the executed Xpath expression for the variable field is visible, in step 2 the result of the Xpath will be displayed (in this case the value „R & D“)



the result of the trace above is:

- Now the detail area will be generated. That is, the list of employees belonging to the **first** department. As in the case of the report header, there is first a **test** to check whether the area will be generated at all. In this case, the detail area will be generated 4 times.

Field/Area		START Area Detail
	XPath	**test person**
	Value	*Number of selected nodes 4*
Field/Area		ID Detail
	XPath	**select person**
	Value	*Number of selected nodes 4*

- For each *person* (detail area), a comment „repetition of Detail“ can be found in the trace.

Field/Area		repetition of Detail
Field/Area		ID RVLine216
Field/Area		ID lblName
Field/Area		ID fldName
	XPath	**select name**
	Value	*Node name = John Schmidt*
Field/Area		ID lblAddress
Field/Area		ID fldAddress
	XPath	**select address**
	Value	*Node address = Red street 3*

- Further down in the trace the second repetition of the Header (second department node) can be found. The second department name „Sales“ has been generated, a new detail area will be created, this time with 2 employees.

Field/Area		repetition of Header
Field/Area		ID Header
Field/Area		ID lblEmployees
Field/Area		ID lblDepartment
Field/Area		ID fldDepartmentName
	XPath	**select departmentName**
	Value	*Node departmentName = Sales*
Field/Area		START Area Detail
	XPath	**test person**
	Value	*Number of selected nodes 2*
Field/Area		ID Detail
	XPath	**select person**
	Value	*Number of selected nodes 2*
Field/Area		repetition of Detail
Field/Area		ID RVLine216
Field/Area		ID lblName

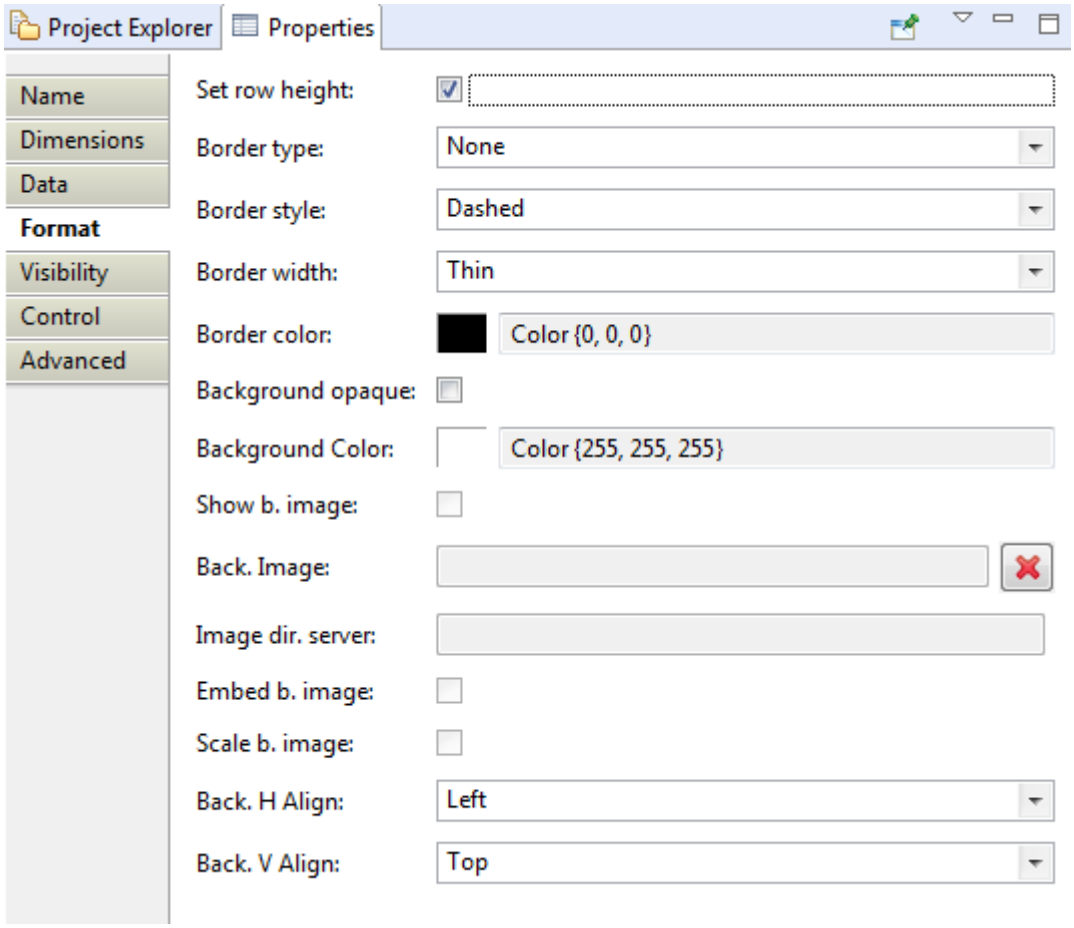
Background colors and images

Areas can have background color or background images, the available options are:

1. Set the opaque property to *Yes* and select a background color
2. or set the opaque property to *No* and set a background image. The background image can be align horizontally and vertically by using the *Background align* properties. Select the image in the local system (*back image* field), for later execution on the server, fill in the „*Image dir. server*“.

   You can use the „show b. Image“ flag for showing the image in the designer, if you remove this flag the image will still be created in the output but not shown in the designer

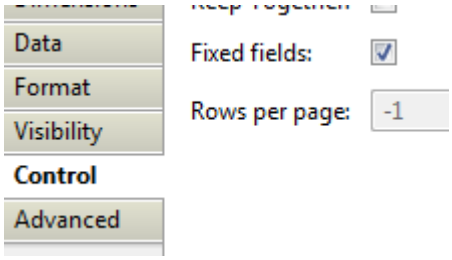
3. The embed image flag will include the image in the xslfo file as base64 resource so no image file will be required at runtime, Finally the scale flag will scale the image in a uniform way (i.e. without changing the aspect ratio)



The background image example

The *BackgroundAlign.xre* example shows how to use a background image and align some data fields to the background contents.

In order to align the field to the background the following flag must be set:

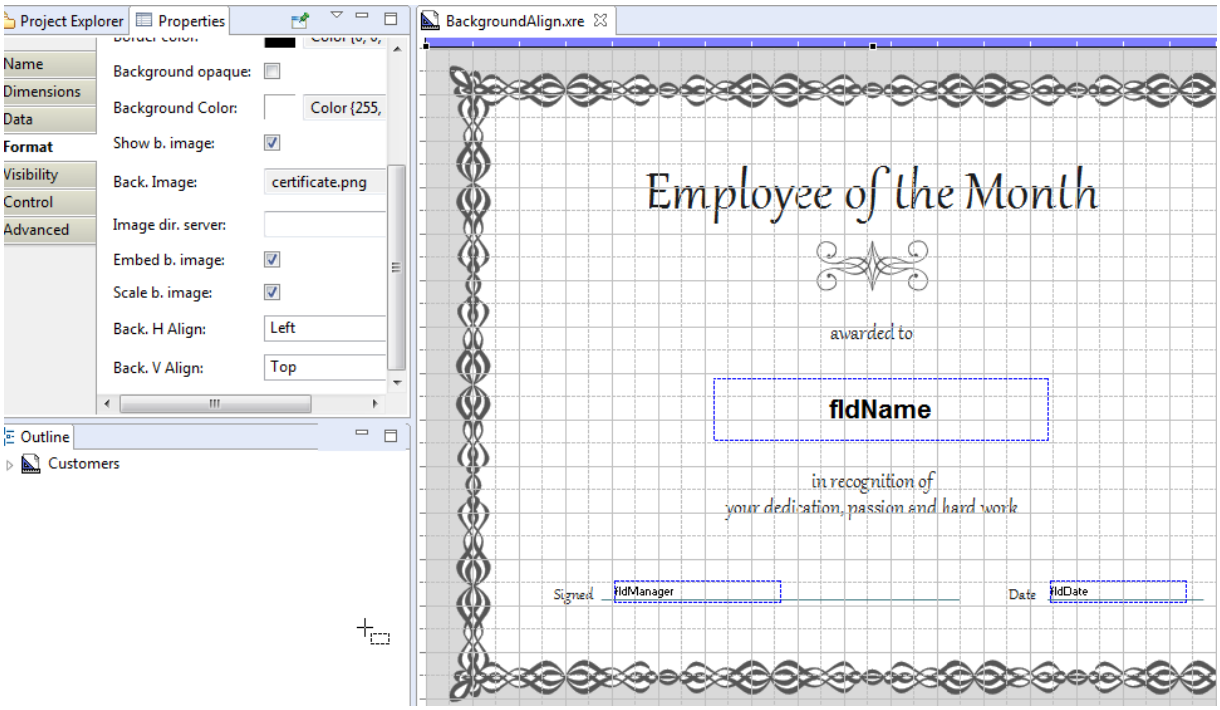


This makes sure the designer does not create a table like layout for the area. Instead only one table cell will be created and the fields will be positioned at the same absolute position as in the designer.

Note this means fields may not grow since they could overlap other fields placed at fixed positions.

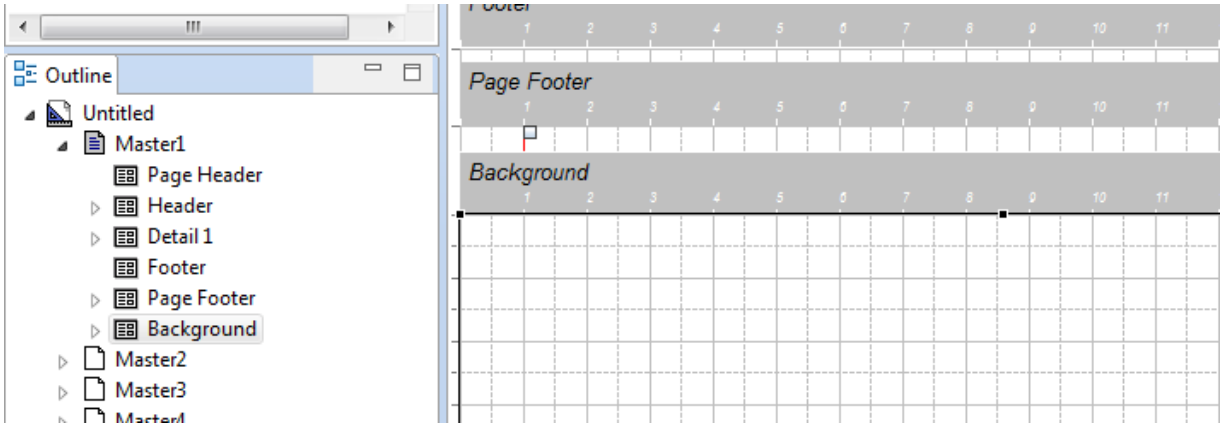
The embed setting will include the image within the xslfo file so there is no need of having the image in a separate image file.

Finally the scale image flag will scale the image to fill the area. It is however recommended to provide at image with the correct size since scaling can distort the image. The image must be approximately 37,5 per CM (or 96 pixels per inch)



The background area

The background area contains objects that will be placed on each page at a fixed location.
The background area is located as last area in the Master object in the outline view:



as with header areas you can remove the area from the output using the visibility flag in the area properties view.

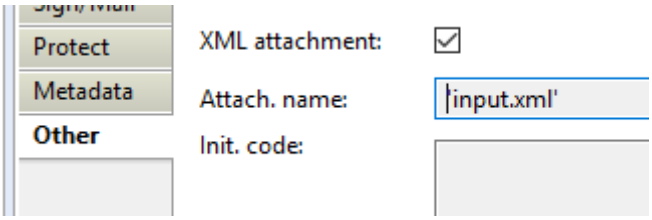
If you use the background area, make sure regular areas have the opaque property set to *false* otherwise they will be overlap the background area.

You can see an example of a background area in *tasks_background.xre*.

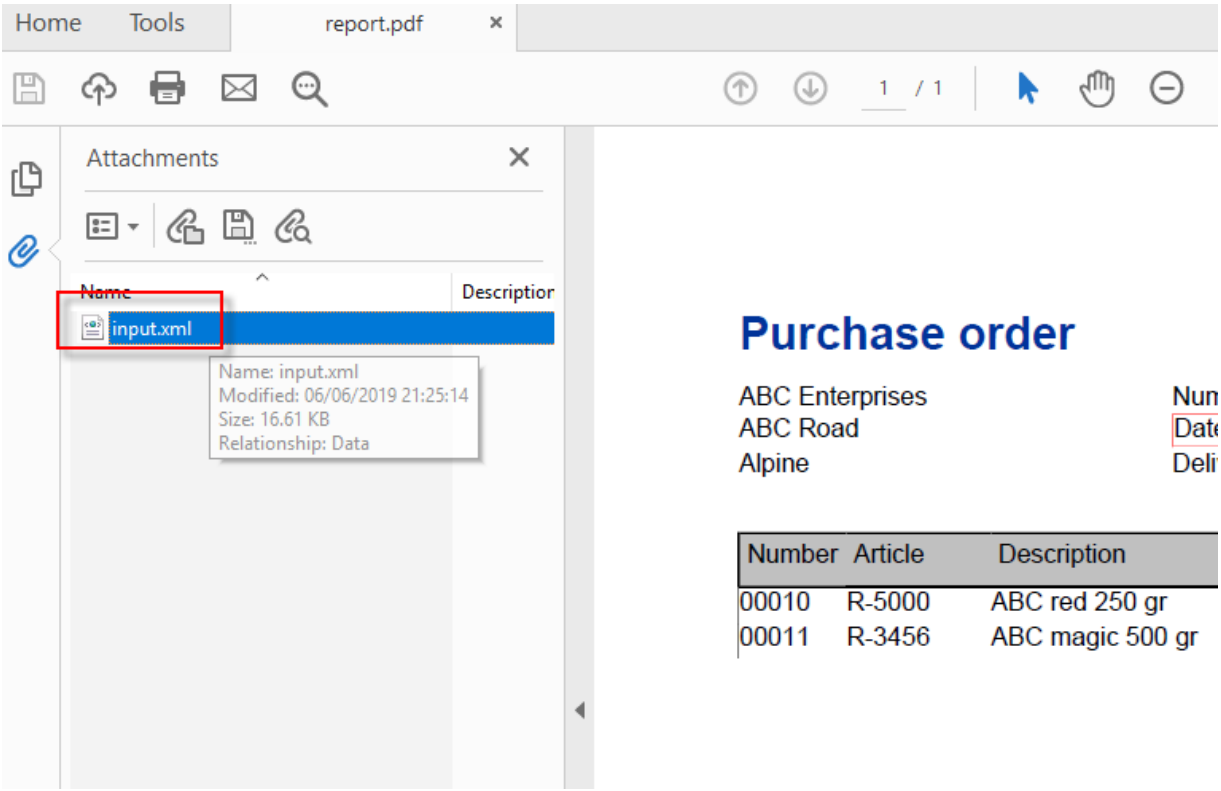
Attachment to the PDF file

It is possible to add the original input XML file to the PDF file as an attachment. In the report's properties that are two entries:

- XML attachment. Set this flag to add the input XML to the PDF
- Attach. Name: enter here a Xpath or constant value. If empty the attachment name will be generated at runtime.

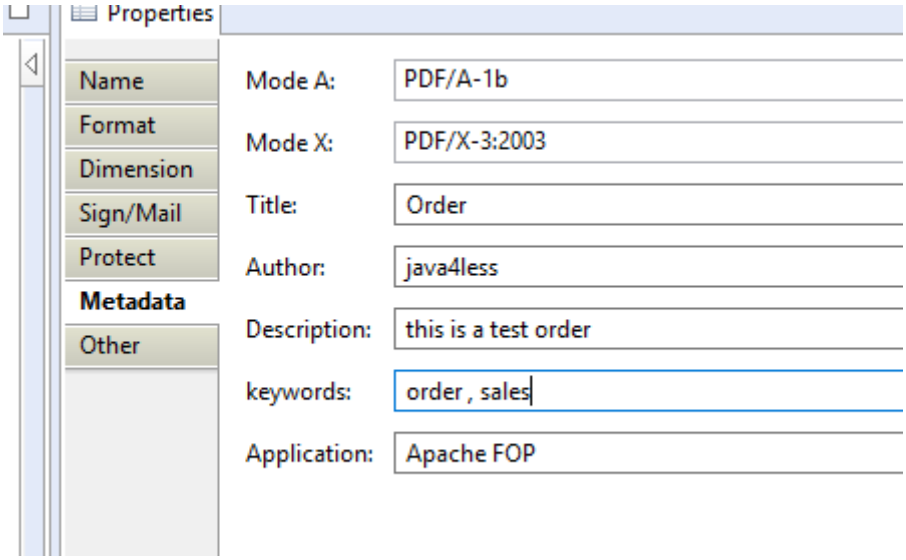


After setting the properties as above the xml file will be available as an attachment

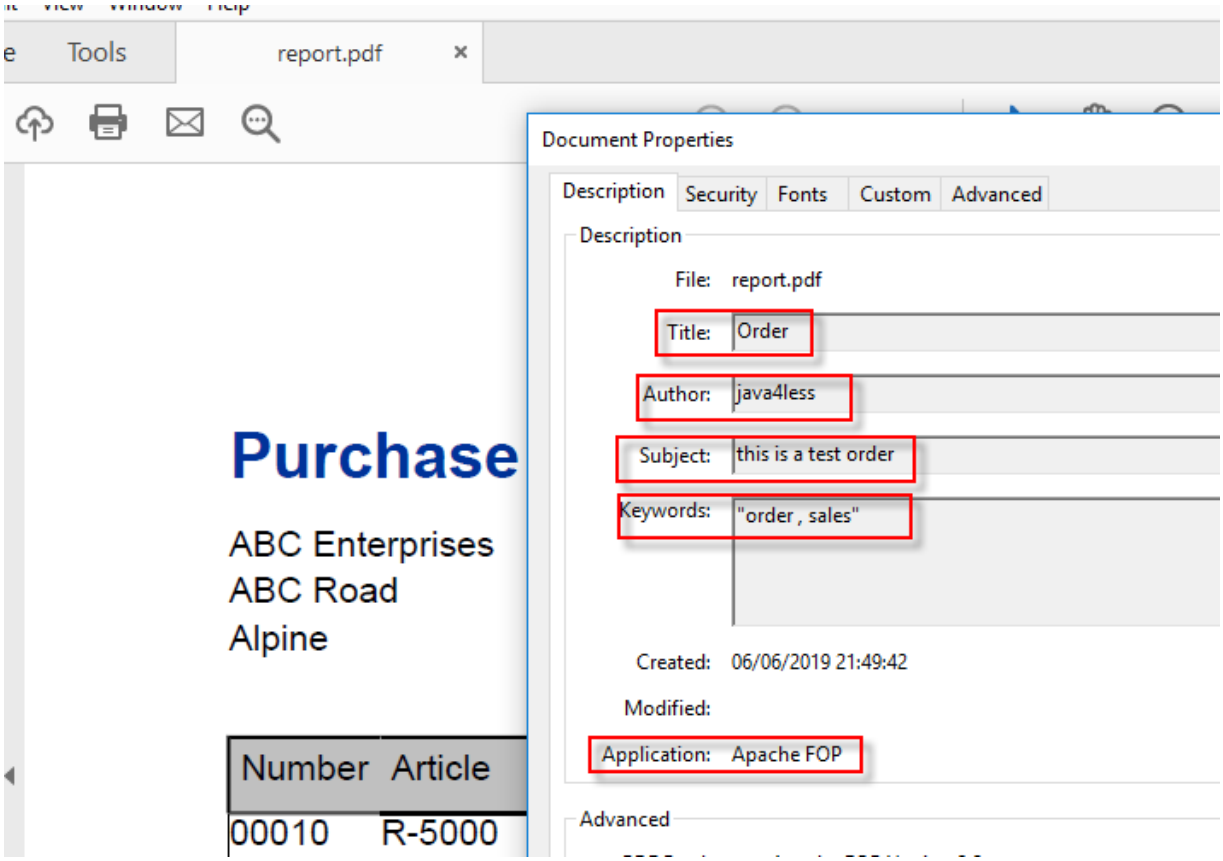


Report‘s metadata

The metadata tab in the properties can be used to set the PDF metadata information as below



The result will be visible in the File – properties menu of the PDF viewer

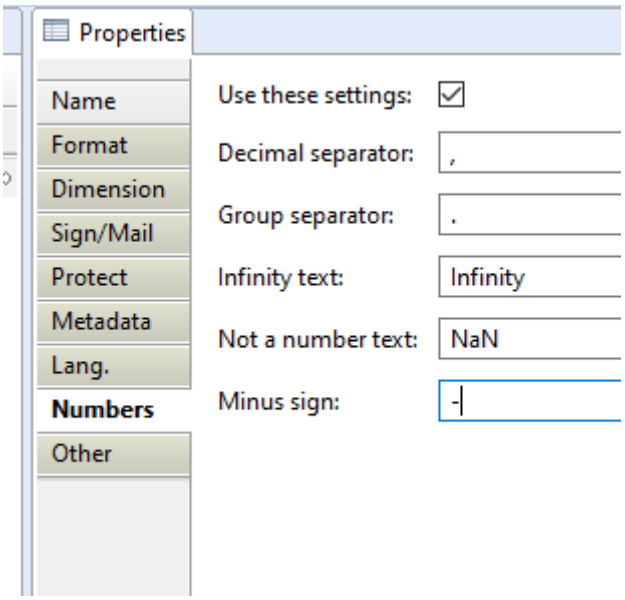


The two PDF modes properties can be set to make sure the PDF is complaint with the following standards:

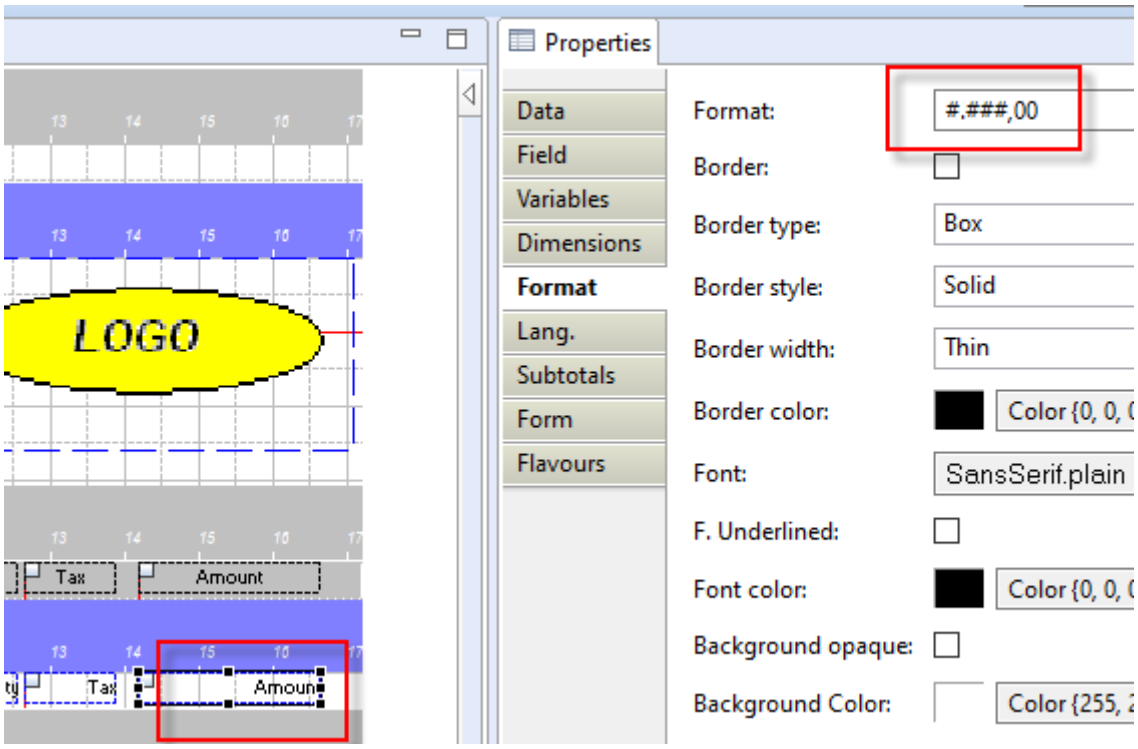
- **PDF/A-1b** : more information at https://en.wikipedia.org/wiki/PDF/A
- **PDF/X-3:2003** : more information at https://en.wikipedia.org/wiki/PDF/X

Decimal numbers formatting

In the report‘s properties , N*umbers* tab, the format of the decimal numbers can be defined. Set the flag at the top if you want to override your platform‘s default settings.

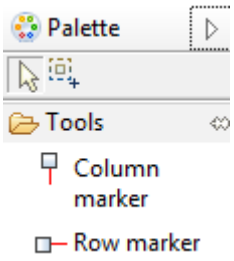


Beware that the entered values in „*decimal separator*“ and „*group separator*“ **need to match** the ones used in the format setting of the fields.

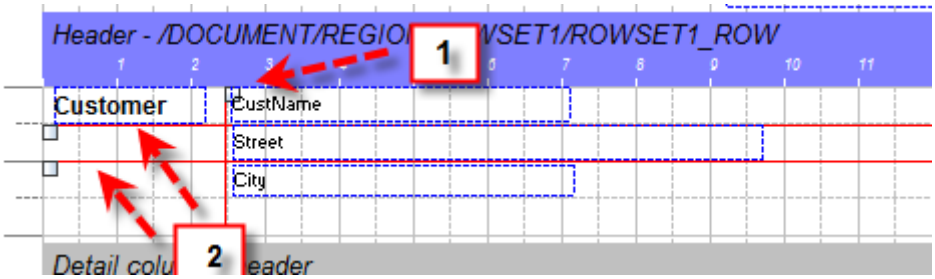


Columns and rows markers

FO designer will always internally create a XSL-FO table (similar to a HTML table) and place each field or image in one cell in the table. In most cases this is done automatically however there are some situations where you explicitly have to define the columns and rows of the area as explained below.



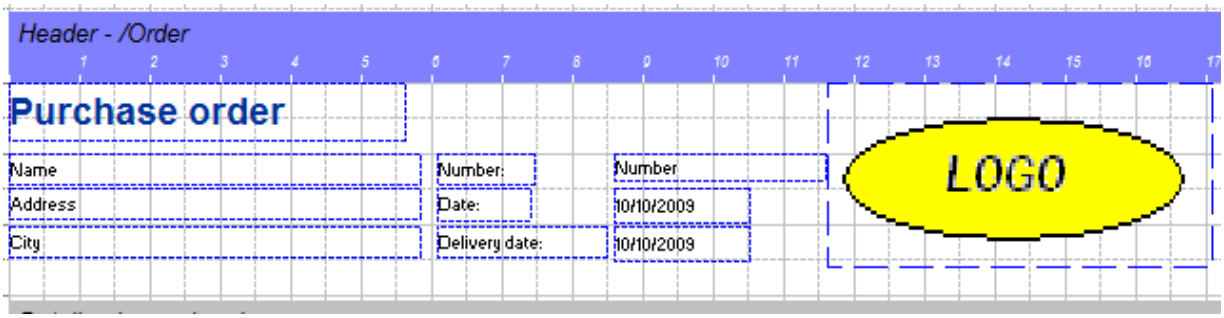
You use the Row and Column marker object of the palette and place them in the position where you want to have the row and column separation of the generated table



The cases where you need to define the columns and rows separator yourself are:

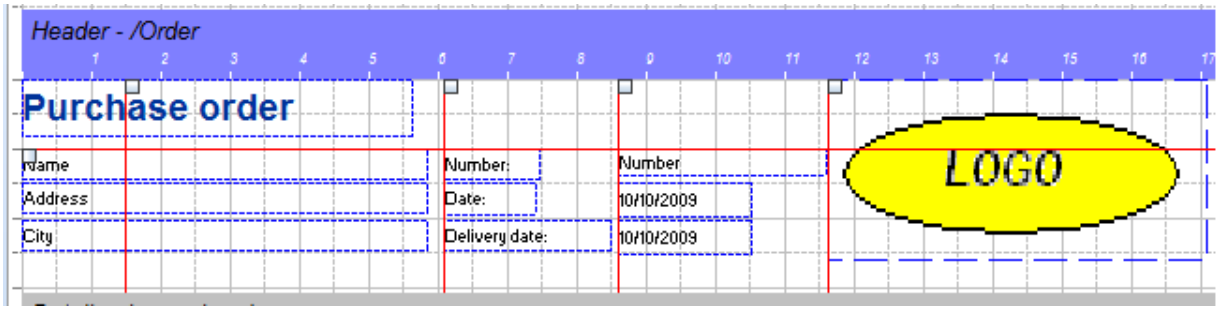
1. if you have an object that expands several rows (for example an image)
2. if you want to paint a frame or line around the column, row or cell (see the *frame* properties in the area's properties)
3. in case the automatic layout of fields does not produce the expected output, you may need to define the columns and rows yourself to have a better control how the objects will be aligned.

The following example illustrates case 1. The following screenshot shows a purchase order template with an image on the right side



the output PDF is:

the problem here is the image and the "Purchase order" text are placed on the top which is correct but all other fields and moved further down in the page. If you however define the rows and columns yourself like this:



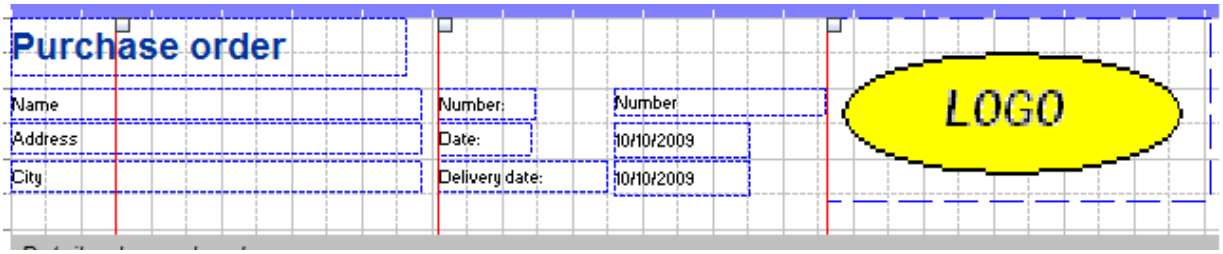
the output will look correct because you tell the designer the area has 2 rows, in the first one you have the "Purchase order" text and the image, furthermore the image occupies both rows.

Note the rules for placing the column and row separators are:

1. Objects can span one or more rows, if they span more than 1 row, no other objects must be placed in the cells below it (as in the logo *image* object above)
2. Objects can span one or more columns. See for example, *Name, address* and *City* fields above, they occupy columns 1 and 2.
3. You can have several objects in one cell but they have to be place at different heights (top to bottom placement). Placing several objects in the same cell, from left to right will not produce the desired output.

The following example illustrates case 3. If you remove the columns separator between the *Number:*

label and the *Number* field like this:

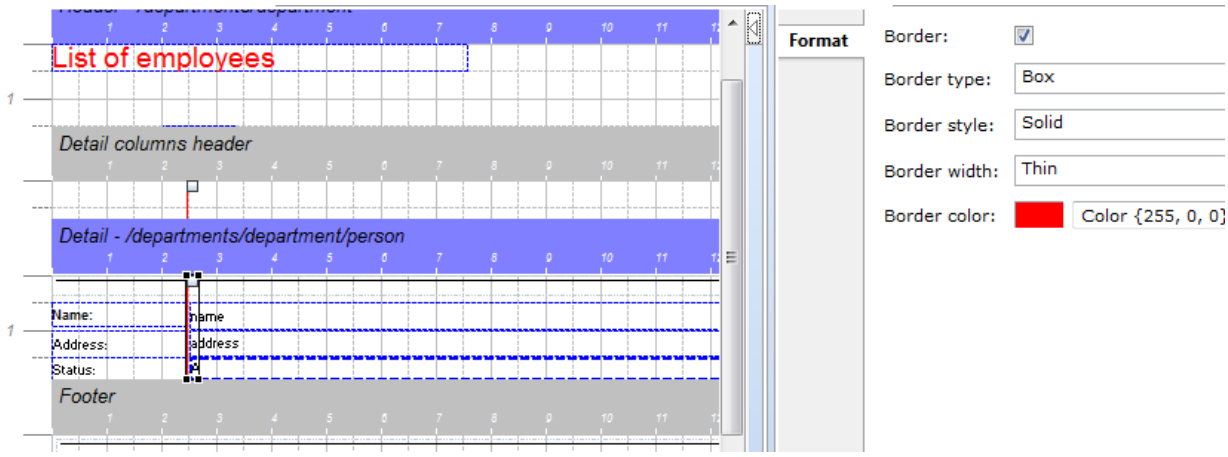


You have now 2 objects at the same height and the output will be incorrect (see below) because the designer is unable to put 2 object at the same Y position unless there is a column separator:

Note: if you do not create any column, the designer will automatically (internally) do it for you, however if you define at least one column separator, the automatic mode will be disabled and in this case you have to place all columns separators (as shown in this last example).

Drawing lines with columns and rows markers

Columns and row marker can be used for drawing vertical and horizontal lines. By clicking on the marker icon, the properties will be displayed. If the property *Border* is set to *yes*, the line will be painted.



Properties of the objects

Some properties of the object, like the position (X and Y) and size are self-explanatory, therefore they will not be listed here.

Properties of the template

Select the report object in the outline view to display the properties of the template.

- **Date format:** format of the dates in the input XML file. The default one is yyyyMMdd"T"hh:mm:ss. This property is important if the template has to reformat the date fields (see format property in the field object). This is the input date format in tthe XML file.
- **Orientation**: horizontal for landscape and vertical for Portrait.
- **Size:** page size. It can be customor predefined.
- **Margins:** define the margins of the page.
- **Name**: name of the template, used for information purposes only.
- **Base template file:** this is a template report file which contains 3 areas (page header, page footer and background area). It can be used to have a common page header for all reports in a company (see the Base Template example further down in this document)
- **Documentation**: free text for documentation purposes.
- **Sign flag**: select this if you want the J4L FOP server to digitally sign the PDF file.
- **Mail flag:** select this if you want the J4L FOP server to send the report PDF file per email (see Premium and Cloud J4L FOP Server section).
- **Default font:** font to be use in all fields by default. It can be changed at any time and the chnage will be cascaded to all fields which have the „*Use default font*“ flag set.
- **User functions class:** See FAQ How to create a user defined xpath function.
- **Flat form**: see section Form's internal structure.
- **XML attachment:** see Attachment to the PDF file.
- **Attach. name**: see Attachment to the PDF file.
- **Init code**: see Adding your own code section.

- **Excel report only**: check this flag for Excel only report, this will check the validity of your targe server (according to the preferences window).
- **Number tab, Use this settings:** activate report's number format default settings (Decimal separator,Group separator,Inifinity text,Not a number text,Minus sign).
- **Protect tab**:see PDF protection section.
- **Metadata Tab**: see Reports metadata section.

Properties of the areas

Click on the area ruler or on the area in the outline view to display the area‘s properties:

- **XML Node**: if this element is empty, the area will be generated only once. If this area however contains a XPath expression, the area will be generated as many times as elements returned by the XPath.
- **XML Node Type**: If *absolute* the XPath will be evaluated as you see it in the designer. If *relative*, the XPath will be relative to the current element (*XML Node* property) in the super area.
- **XML Sort**: enter an XPath if you want to sort the elements returned by the *XML Node* property. The *XML Sort* property is a XPath that returns a list of elements used as key for sorting. For example, you could set the *XML Node* to be the items in the purchase order as in our example */Order/OrderDetail/ListOfItemDetail/ItemDetail.* The you could sort using the article number */Order/OrderDetail/ListOfItemDetail/ItemDetail/BaseItemDetail/ItemIdentifiers/PartNumbers/B uyerPartNumber/PartNum/PartID*.

- **XML Sort type**: whether the sorting elements are numeric of alphanumeric values.
- **XML Sort order**: ascending or descending.
- **XML Group by (3 fields):** used for grouping values (see **additional section of this topic**)
- **Local group**: select this if the grouping shall not be propagated to the subareas.
- **Set row height:** this setting makes sure the area will have the same height as you define in the designer. If you remove this setting the area will shrink when there is no data in it. In some cases if you have conditional fields (using flavours) this can be useful to avoid empty spaces.
- **Background**: background color of the area
- **Alternate colors:** set this flag to have another background color every second repetition.
- **Alt. Back. Color:** if the previous flag has been set, this will be the background color of every second repetition.
- **Background image:** select background image for the area.
- **Back H align:** horizontal alignment of the background image.
- **Back V align:** vertical alignment of the background image.
- **Image dir server:** directory where the file will be found at runtime.
- **Show b image:** show or hide image in the designer.
- **Scale b image:** Scale the image to fill the area , the aspect ratio will not change.
- **Embedd b image:** embed the image in the XSLFO as Base64 encoded resource.
- **Frame**: select a value different from NONE to enable the area's frame. Possible values are: o AREA BORDER: paint only the border of the area
  o ONLY ROWS: paint only horizontal lines to separate area repetitions and rows within the area.
  o ONLY COLUMNS: paint only vertical lines to separate columns in the area (see column markers).
  o GRID: all above will be painted.

- **Frame color**: select the color with the color .
- **Frame style**: select one of the styles
- **Frame width**: select THIN, MEDIUM or THICK.
- **Rows**: see **Columns and rows** section.
- **New page** (page break): if true the area will be generated in a new page.
- **Keep together** (in detail areas only): if true the area will be printed in the next page when not all fields fit in the current page.
- **Fixed fields**: This flag allows placing the fields at a fixed position in the area. This means fields are not allowed (shall not) to expand since they have fixed positions and would overlap the fields below or to the right. Also this setting disables the creation automatic of

columns and rows, furthermore columns and row markers you place in the area will be ignored. The main purpose of this flag is to align the fields to a fixed background image (for predefined form images).

- **Rows per page**: if you set this value to a number higher than 0, a new page after X repetitions of the area (being X the number you enter).

Properties of a text field

Click on the object to display its properties.

- **constant**: set it to true if this is a constant field (ie. a label). Note if a field has been declared as constant (as a label), the border will be **black** instead of **blue**, in this way you can easily recognize which fields contain variable data and which field are labels.
- **value**: constant value (instead of using a XPath). It will be used if the *constant* property is set to true or the XPath property is empty.
- **XPath**: path to the source XML element of this field. It can be any valid XPath expression, that is, it can contain conditions and functions. The employee tutorial contains an example how to use functions and the invoice IDOC example uses the condition technik extensively.
- **XPath type**: If *absolute* the XPath will be evaluated as you see it in the designer. If *relative*, the XPath will be relative to the current element (*XML Node* property) of the area.
- **Convert from Html:** the field contains HTML data that will be converted to PDF. Note this feature will only work if you use the J4L Print Server. See explained HTML examples.
- **Unescape:** used together with „Convert from html“, see explained HTML examples.
- **Split string at (chars):** if the value is greater that 0 it will split the input value in fixed length chunks.
- **Skip missing elements:** it set, there will be no xsl-fo output when the input XML node is missing or empty.
- **Wrapping:** self explaining, default value is the same as wrap
- **Overflow:** if overflow is hidden and wrapping is set to no-wrap, values that do not fit in the fields length will be cut. Default and Visible will allow overflow, potentially overlapping the field on the right. Scroll will about the user to take advangate of scroll bars, if supported by the PDF viewer.
- **Xpath Var1 to 5**: these are the Xpath to variable data in long text fields. See *Letter* example to learn how to work with variables.
- **Align**: text alginment
- **Background:** select background color of the field.
- **Background opaque**: set it to true to activate the background color.
- **Border**: set it to true to activate the border of the field
- **Border Color, Style and width**: properties of the border.
- **FO attributes:** any xsl-fo attributes to add to the field. Note this might lead to errors if you use attributes not compatible with the one generated by the designer.
- **Use default font:** set this flag to use the default font as defined in the report‘s properties.Uncheck the flag to use a specific font for the field.
- **Font**: currently only the built-in fonts are supported (SansSerif, Courier, Times Roman and Symbol).
- **Font color**: self explanatory.
- **Format**: output format for dates and numbers. For numbers the format is:
    - # denotes an optional digit.
    - 0 denotes a digit.
    - . decimal point.
    - , is the group separator for thousands.
    - ; Pattern separator. The first pattern will be used for positive numbers and the
    - second for negative numbers.

- % percentage sign

For dates the format is:

- y: year
- M: month
- day of month
- H: hour (0-23)
- m: minute
- s: second
- further information can be found in the **Java documentation of the SimpleDateFormat** class.

· **Rotation**: use this field to rotate the text
· **Preserve LF**: set it to true to keep all spaces in the value..

Properties of a memo field

- **Memo text:** contains the long styled text.
- **Variables:** Xpath for the placeholders in the memo text. The variable placeholder texts are $1 to $5 . See *HelloWorldMemo* example in this document.

Properties of a combo box

Click on the object to display its properties (see also properties of a text field).

· **Key List**: a list or key/value pair. The element returned by the XPath will be used as key to find the value in the list. This value will be the output in the PDF File.

Properties of a picture

Click on the object to display its properties (see also properties of a text field).

· **Image**: path to the image. There are 3 types of images (**Constant property**):
  - CONSTANT FILE. These are fixed image files which can be loaded into the report.
  - They do not have to be available at runtime. See example *"order.xrp"*
  - DYNAMIC FILE: these are images which are located on the file system but the name
  - is calculated at runtime using a XPath expression. See example *personCard.xrp*. We
  - recommend placing images below the working directory. That will be either in the FODesigner directory or in the server, below the J4LFOPServer directory.
  - FROM XML. These are images located inside the source XML file as a base64 image, they are also referenced with an XPath expression. See for example *"order_logo.xml"* and test file *"OrderSample-ABC-Image.xml".*
· **Load image in FO file** (for constant images): select to load a base64 representation of the image, otherwise the image will be read from the file system at runtime.

- **Local image URL** (for constant images): if the the above flag is not selected this will be the location of the image in your computer at design time.
- **Server image URL** (for constant images): if the the above flag is not selected this will be the location of the image in the server at runtime.
- **Scale to Fit**: scale the image to fit the size of the picture object in the template.
- **Scaling**: the default behaviour is to scale the image using "uniform" scaling which makes sure the aspect ratio of the image does not change. Use "non-uniform" if you want to scale the image without aspect ratio considerations.
- **Constant**: see Image property.
- **Type:** set automatically when the image is loaded.

Properties of a line

- **Line Width**: width of the line.
- **Style:** type of line (dots, dashed ...).
- **Line length: (default 100%).** You can define the length in % or in cm.
- **Orientation:** vertical or horizontal.

Properties of a www links

- **Link label**: label displayed in the PDF file. If empty, the www address will be shown.
- **Value:** www value, either as a constant value or as a XPath expression..

Properties of a free code objects

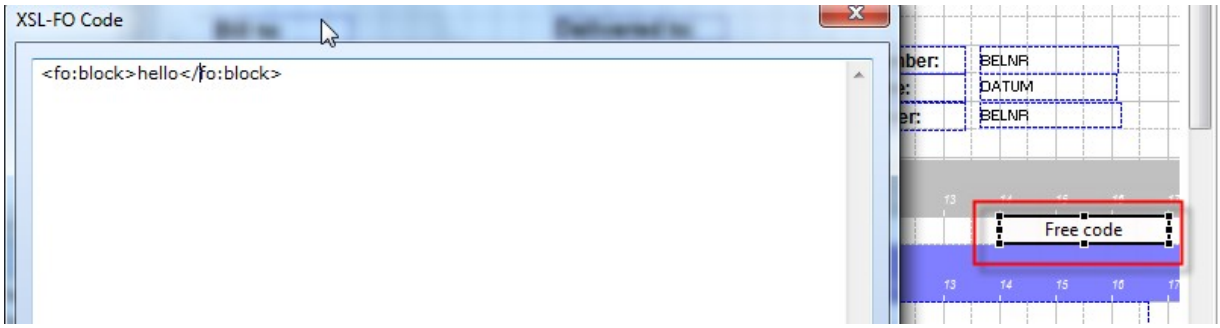
The free code objects are required when you need to add code to the xsl-fo output. This should be done only if you are familiar with the xsl-fo language. If the code is not correct the PDF generation will fail.

The 2 main properties are:

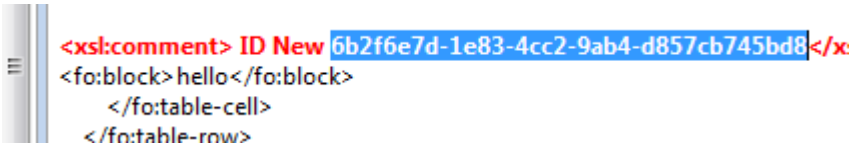
- **XSL-FO Code**: use the editor to add code. Note you have to use the prefix “fo:” for fo elements and the prefix “xsl:” for xls elements.
- **Location:** of the code. The code can be handled as attributes of the current cell/row/table or as standalone code before the area or inside the current cell.

Example 1

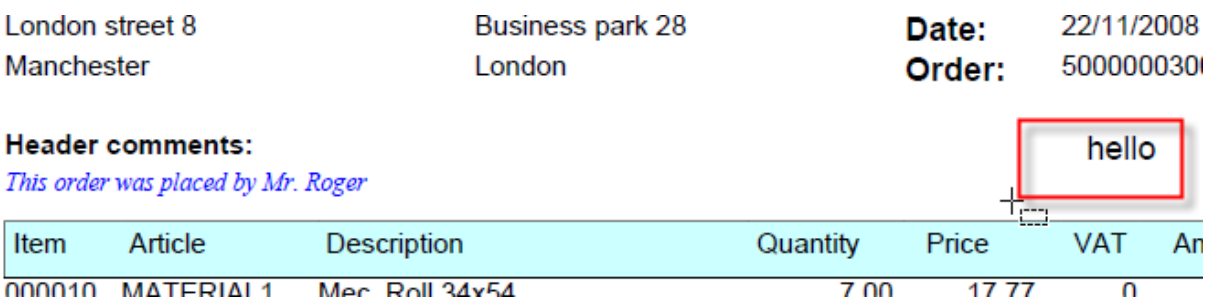
This simple code will add the work “Hello” to the output



In the code tab you will see the code has been added to the output

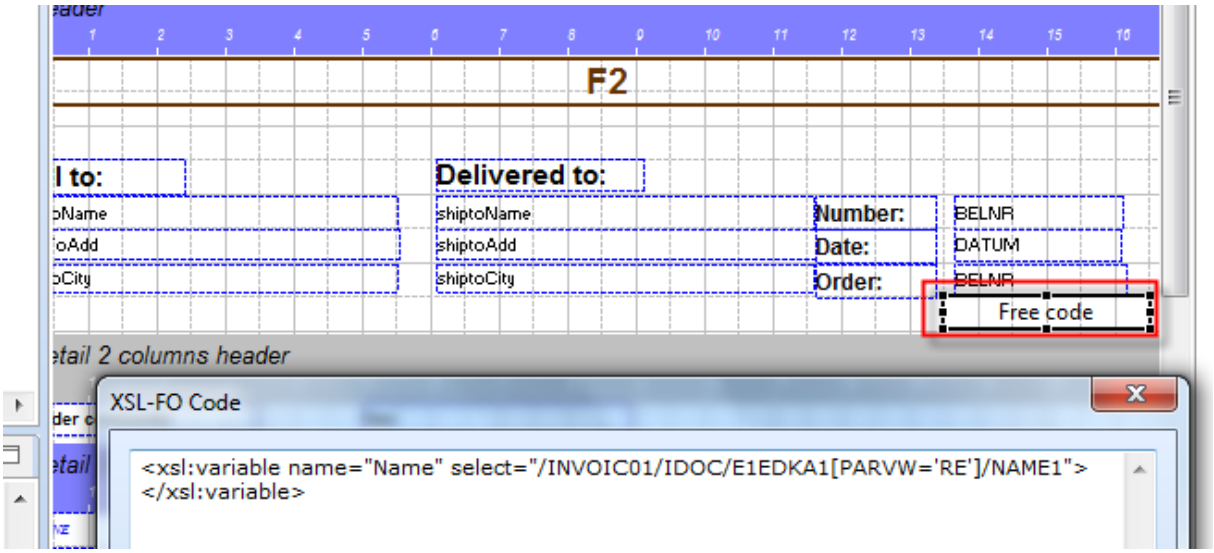


And in the preview tab you can see the generated PDF :

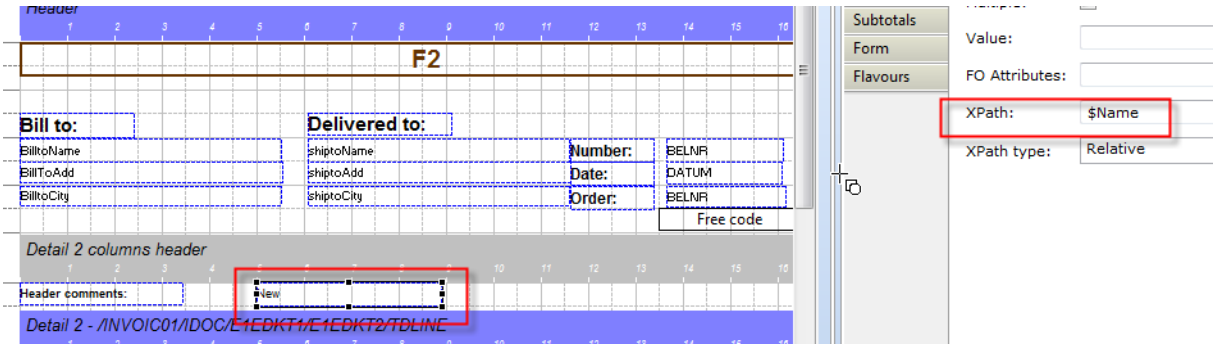


Example 2

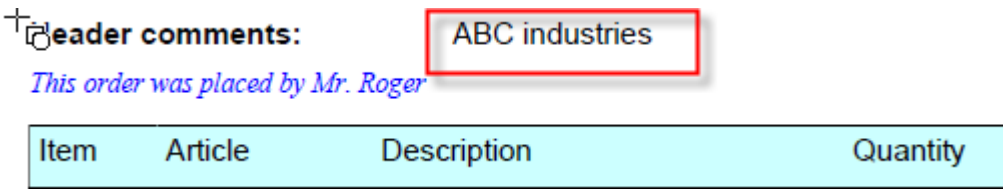
This code shows how to store the value of an input field into a variable. The variable will be set before the area is created in this way it can then be used by all sub areas. Select the property location „table before“



We can then place a text field in another area to read and show the variable (note you use the sign $ following by the variable name):



The output shows then the variable value:



Properties of a table frame

The table frame object creates a complex frame (table like) around existing fields.Note this object is only a decoration object, it does not contain any fields or logic.

the properties are:

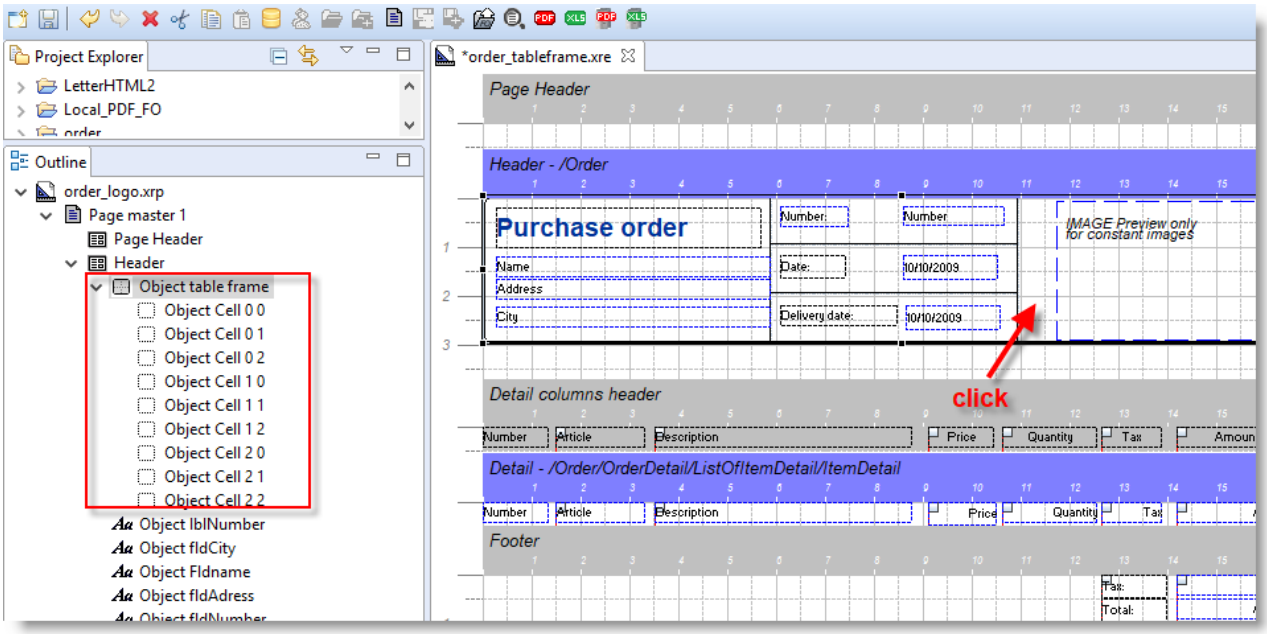
- **rows:** number of rows of the table.
- **columns:** number of columns of the table.
- **round corners:** select this flag to have round corners in the tabke
- **Line Width:** width of border the line.
- **Style:** type of the border line (dots, dashed ...).
- **Color:** color of the border line.
- **Background:** select background color of the table.
- **Background opaque:** set it to true to activate the background color.

Properties of the cell

- **expand down:** set it to 2 to join the cell with the one below. The value 3 will join 3 cells, and to on.
- **expand right:** set it to 2 to join the cell with the one to the right. The value 3 will join 3 cells, and to on.
- **Line Width:** width of the border line.
- **Style:** type of the border line (dots, dashed ...).
- **Color:** color of the border line.
- **Background:** select background color of the cell.
- **Background opaque:** set it to true to activate the background color.

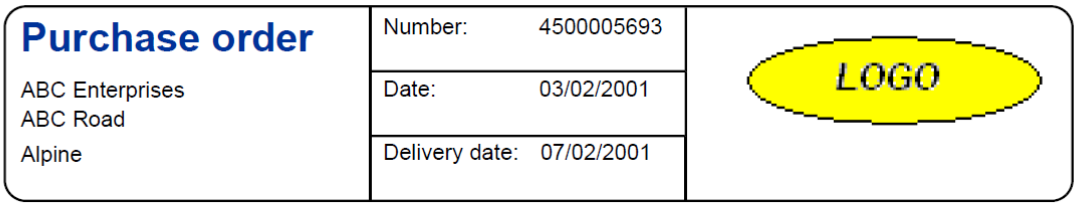
Table frame object are always added to the background of existing objects. You work with table frames in this way:

- In order **to select the table y**ou can **click** on the empty spaces between the fields, or you select the table frame in the Outlibe window
- In order to **select a table's cell** you can **double click** on the empty spaces between the fields, or you select the cell object in the Outlibe window. Note the naming of the cells is "cell <row> <column>".
- In order **to resize a cell** (and therefore the row or column), you need to select the cell and drag the borders of the cell, as you do with other objects.



the example *order_tableframe.xre* produces the output below, for this the properties of the table and cells area:

- the table has 3 rows and 3 columns.
- cell 0 0 (the top left) has property "expand rows" set to 3, so it will span the 2 cells below.
- cell 0 2 (the top right) has property "expand rows" set to 3, so it will span the 2 cells below.
- cell 0 1 (the top middle) and the 2 below have no change in the properties.



Number	Article	Description	Price	Quantity	Tax	Amount
00010	R-5000	ABC red 250 gr	10.0	111.0	16	1110.0
00011	R-3456	ABC magic 500 gr	1000.0	1.0	16	1000.0
					Tax:	337.6
					Total:	2447.6

5. Executing the template to create PDF files

Once you have generated the XSL-FO file you can use Apache FOP for converting your XML files to PDF.

However we provide the following class:

com.java4less.xreport.fop.FOProcessor

which has a very simple interface:

```
/**
* generate PDF file for XML document
* @param xmlStream input XML document
* @param xsltStream XSL-FO file as created by the designer
* @param os output PDF file
* @throws Exception
*/
public void process(InputStream xmlStream, InputStream xsltStream,OutputStream os)
throws Exception {
```

for example:

```
FOProcessor processor=new FOProcessor();
processor.process(new FileInputStream("Order.xml"), new FileInputStream("order.fo") ,
new FileOutputStream("report.pdf"));
```

FOP web server (servlet)

The delivery includes a WAR file which can be deployed on a web server. This includes a servlet that operates as follows:

1. It receives a HTTP request using the POST method.
    - The body of the request contains the XML document to be converted.
    - or the body contains a FORM and in the form there is a text field that contains the XML data. In this case the parameter DATAFIELD must provide the name of the field (see file Example.html)
2. In the URL of the servlet you provide the name of the XSL-FO file to be used for the conversion, using the TEMPLATE parameter. The template must be the name of the file and it must be placed in the default working directory of the server (it is also possible to provide

the file as a relative or absolute file name). As an alternative the file can be located within the war file in the WEB-INF/classes subdirectory or below it.

3. the servlet returns the PDF file

The delivery includes the following files that you can use for testing:

- *departmentEmployees.fo.*included inside the war file (directory WEB-INF/classes).
- the war file: web/J4LFOPServer.war

You can test the servlet by deploying the war file and then executing the URL:

http://servername/J4LFOPServer/Example.html

When you click on the button the following URL will be openned:

http://localhost:8080/J4LFOPServer/servlet?
TEMPLATE=departmentEmployees.fo&ENCODIN G=iso-8859-1&DATAFIELD=S1

Where

- the file departmentEmployees.fo is inside the war file
- encoding ISO-8859-1 is the encoding of the input data
- S1 is the name of the text field that contains the XML data. Note, if the DATAFIELD is missing, the servlet assumes the body of the HTTP request contains the XML payload.

Additionally the REMOVENS=YES parameter must be used if your input XML file contains namespace information.

Java objects to PDF conversion

This section describes how to convert Java objects to PDF. This description is based on the example contained in the *JavaClass_to_PDF* subdirectory of the delivery.

This example simulates a sales Java application which works with purchase order objects. These purchase order objects are composed of 3 classes:

- com/java4less/examples/po/PurchaseOrderHeader.java
- com/java4less/examples/po/BuyerInformation.java
- com/java4less/examples/po/PurchaseOrderItem.java

This application needs to provide a mean to print a purchase order document out of a Java purchase order object. As a solution this example proposes using Apache FOP for creating a PDF file which can be printed and using J4L FO Designer for designing the layout of the document. Since FOP requires an XML document as input, the Java objects will be first converted to XML using the JAXB (Java XML Binding) API. Note this means you require Java 1.5 with JAXB or Java 1.6.

The steps to be performed are:

1. run JAXB schema tool to generate a XML schema for your Java classes. In our example the file *generateSchema.bat* was used for that and the *schema1.xsd* file was generated
2. Use FO Designer to generate a document template using the schema. The created file is *JavaPurchaseOrder.xrp.* This file can be imported in FO Designer.
3. Use FO Designer to generate a FO file which will be used at runtime. The generated file is *JavaPOExample.fo*
4. At runtime you have to
    1. create your Java objects
    2. convert them to XML using JAXB
    3. convert the XML to PDF using *JavaPOExample.fo* and Apache FOP

    You can see how this is done in the POTest.java file. The source code is aproximately:

```
/   1. create order object
PurchaseOrderHeader po=new PurchaseOrderHeader("1");
po.setBuyer(new BuyerInformation("John Solo","Street
ABC 1","Manchester","AB 673","UK"));
PurchaseOrderItem[] items={ new PurchaseOrderItem("X1","Printer
Injet",1), new PurchaseOrderItem("R4","Optic mouse",1),
new PurchaseOrderItem("M3","Ergo
keyboard",1), new PurchaseOrderItem("X4","CD-
RW",10) };
po.setItems(items);


// 2. create now XML representation of the order
```

```
JAXBContext jc =
JAXBContext.newInstance(PurchaseOrderHeader.class); Marshaller
marshaller=jc.createMarshaller(); ByteArrayOutputStream ba=new
ByteArrayOutputStream(); marshaller.marshal(po,ba);


/  3. create now the PDF output for the XML data FOProcessor
processor=new FOProcessor(); processor.process(new
ByteArrayInputStream(ba.toByteArray()), new
FileInputStream("JavaPOExample.fo") , new
FileOutputStream("JavaPOExample.pdf"));
```

You can use the file POTest.bat for testing the delivered files.

Sending the PDF as an email attachment

This option is currently available only for the APEX server, please check the **APEX specific documentation.**

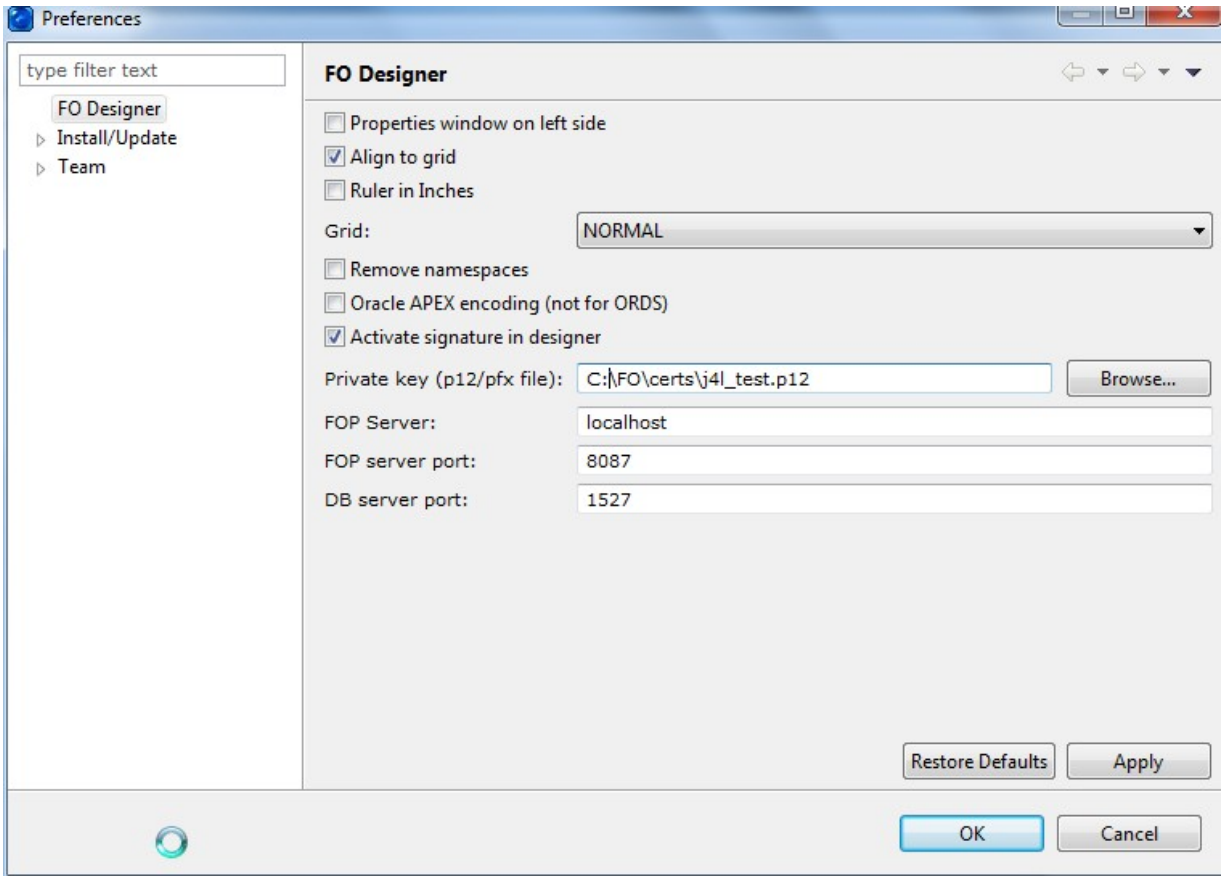
Adding a digital signature to the PDF file

You can add a digital signature to the PDF files you generate. This signature ensures the integrity of your document and its authentication. In other words, the receiver of the document knows you have created that document (authentication) and no one has modified it (integrity). The signature created will use the algorithms RSA signature, SHA1 hash and PKCS7 encoding.

The signature can be created in the design environment for testing purposes and of course also in the runtime environment. Please read the environment specific document (for example **Apex**) to learn how to activate the signature.

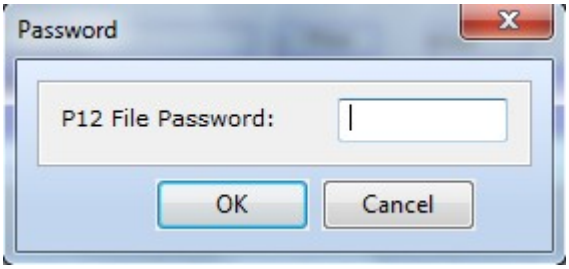
The rest of this section describes how to test the digital signature in the designer and view it in Acrobat reader.

In order to activate the digital signature you use the preferences window. In this window you activate the signature and select the p12 or pfx file which contains the private key used for the signature. Note our product contains a test certification authority (CA) and a test certificate in the ***certs*** subdirectory. You can use these for testing purposes.



If you have enabled the signature in the settings window, each time you generate a PDF file with the PDF button, the designer will ask for the password of the p12 file. The password of the test file we provide is ***test***.

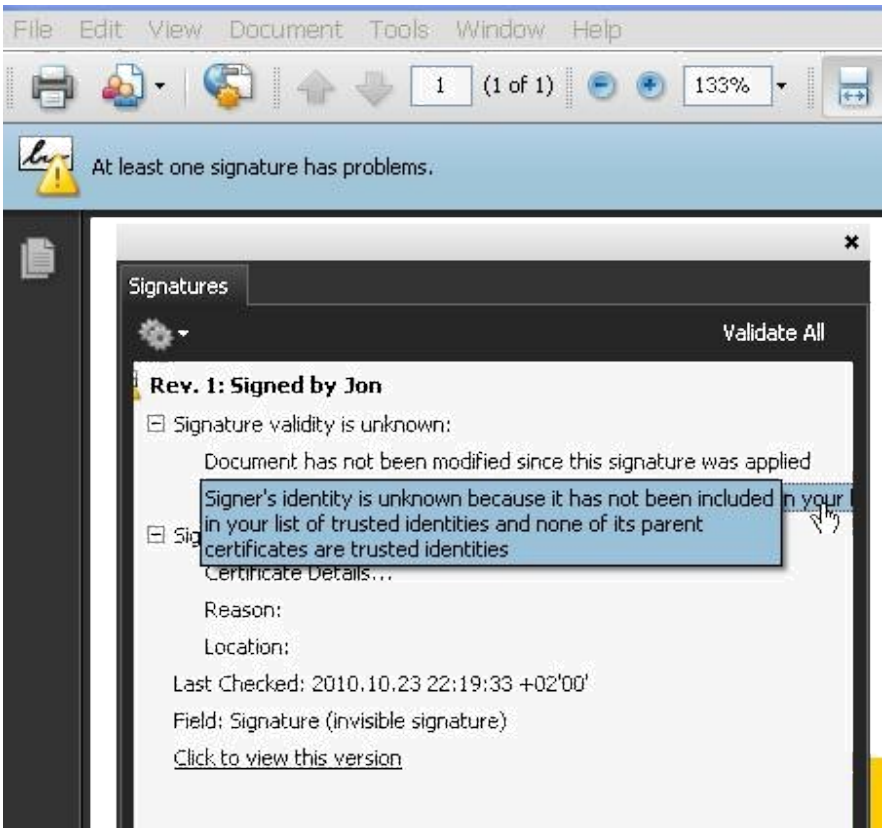
Note, if you click on the cancel button, the PDF file will be created without the signature.



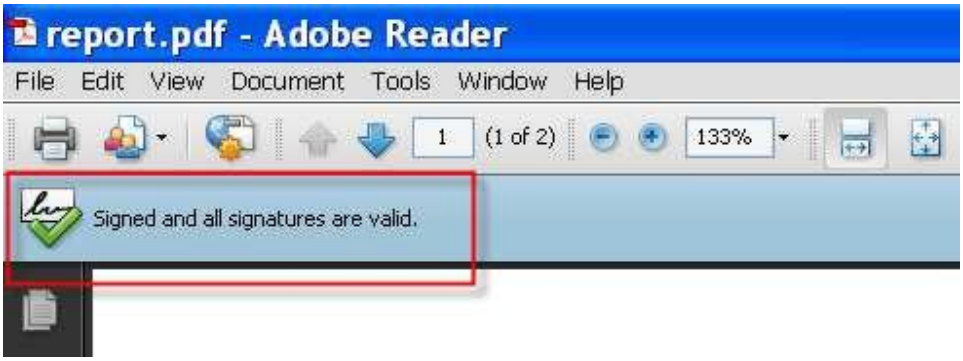
Once you open the created file, you will notice Acrobat reader warns you with the text "the signature has problems". The reason for this is, Acrobat does not know the certificate authority (CA) we have used for creating the test certificate. Looking at the signature closely in the signatures panel you will see:

- The PDF reader says the *document has not been modified since the signature has been applied* (integrity of the document).

- The s*igner's identity is unknown* since the reader does not know our test CA.



If you however click on the certificate details link and add the CA certificate as a *trusted certificate*, you will see the reader now accepts the signature.

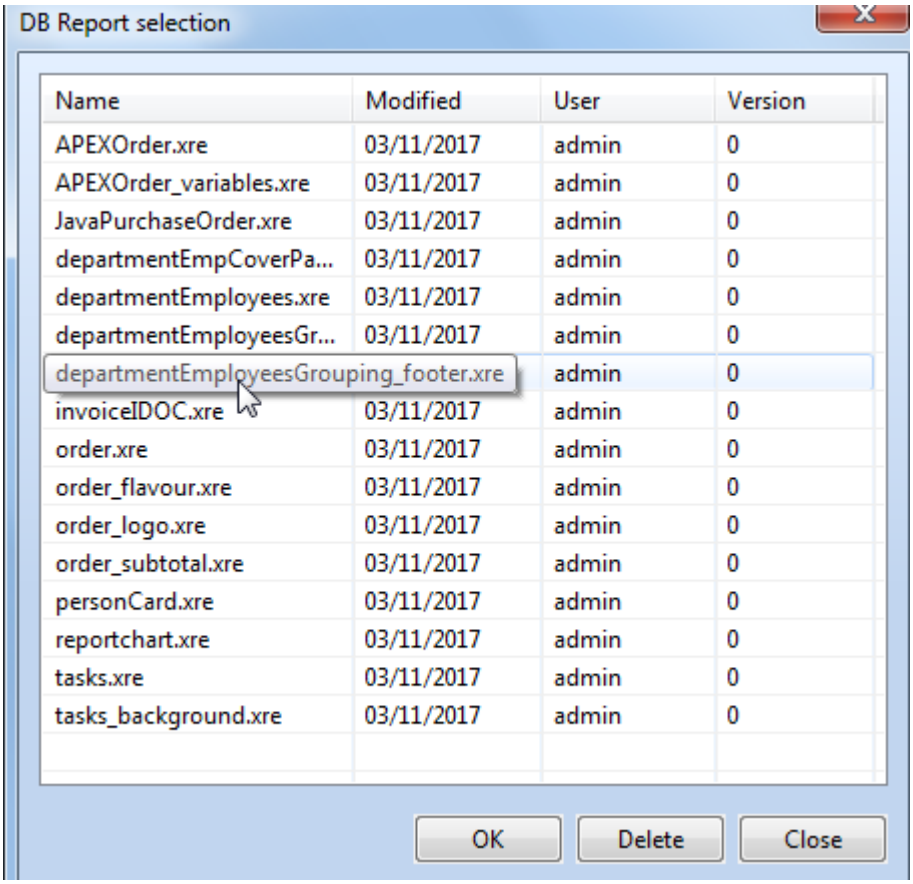


6. Learn by doing: tutorial

The examples in the database

Use the open database icon to open the database. Use the default user *admin* and password *admin.*

Use the open report from database icon to access the examples. Please note you will find more examples to be imported in the installation *examples/projects* directory.

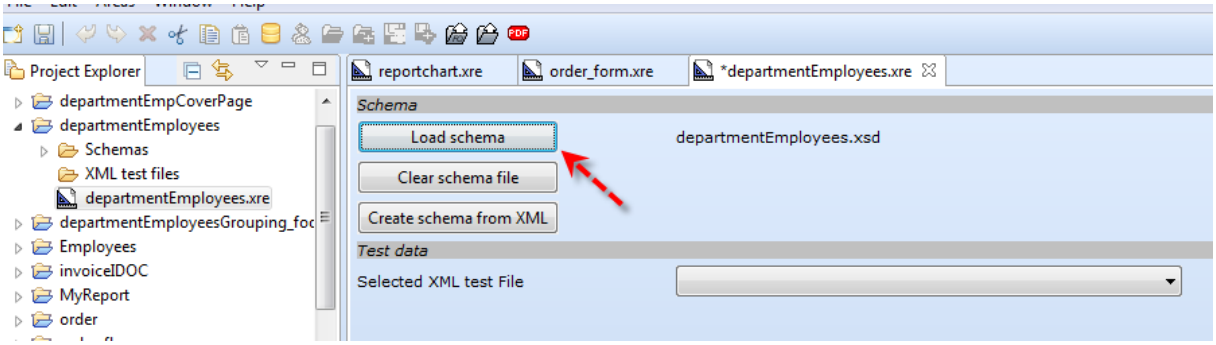


Hands on example

In this tutorial we will take as input an XML document which contains the list of employees for each department. We want to create a PDF file with the list of departments and for each department the list of employees.

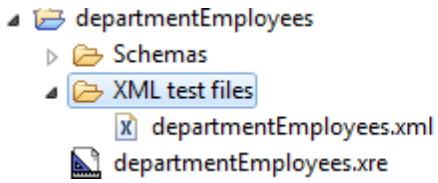
We will start running the FO Designer and we get an empty template.

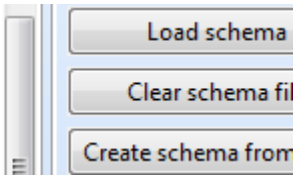
1. First we load the XSD departmentEmployees.xsd. The result will be:



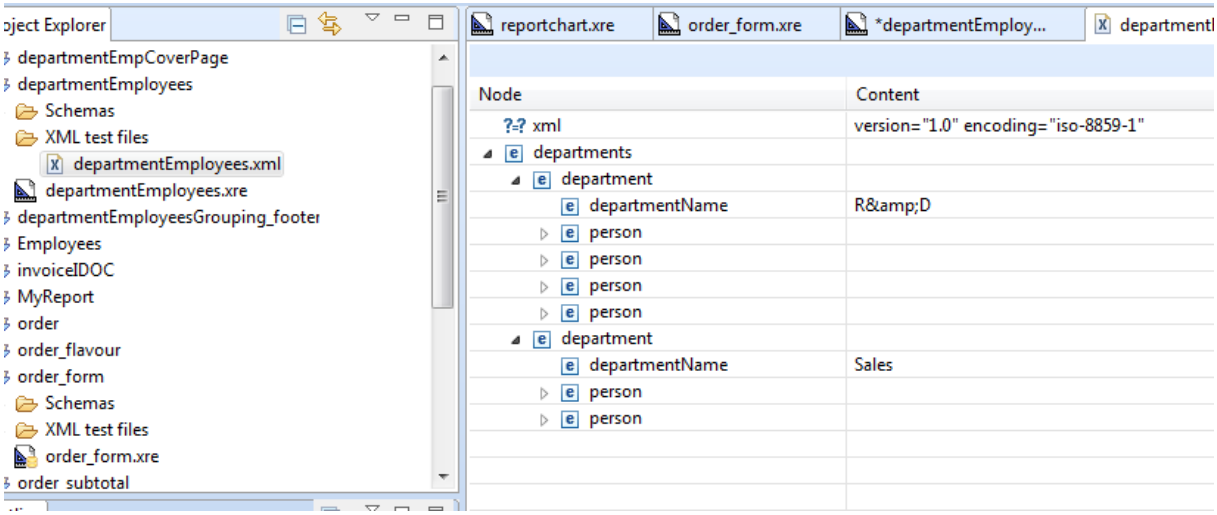
   as an alternative (if we do not have a XSD file), we can click on the *create schema from XML button* and load the XML document departmentEmployees.xml. This would read the XML document and list of existing XML nodes, as a result it would create a kind of "virtual" schema.

2. As second step we copy and paste the XML test file in the *XML test files* folder departmentEmployees.xml. As you can see this file contains 2 departments:

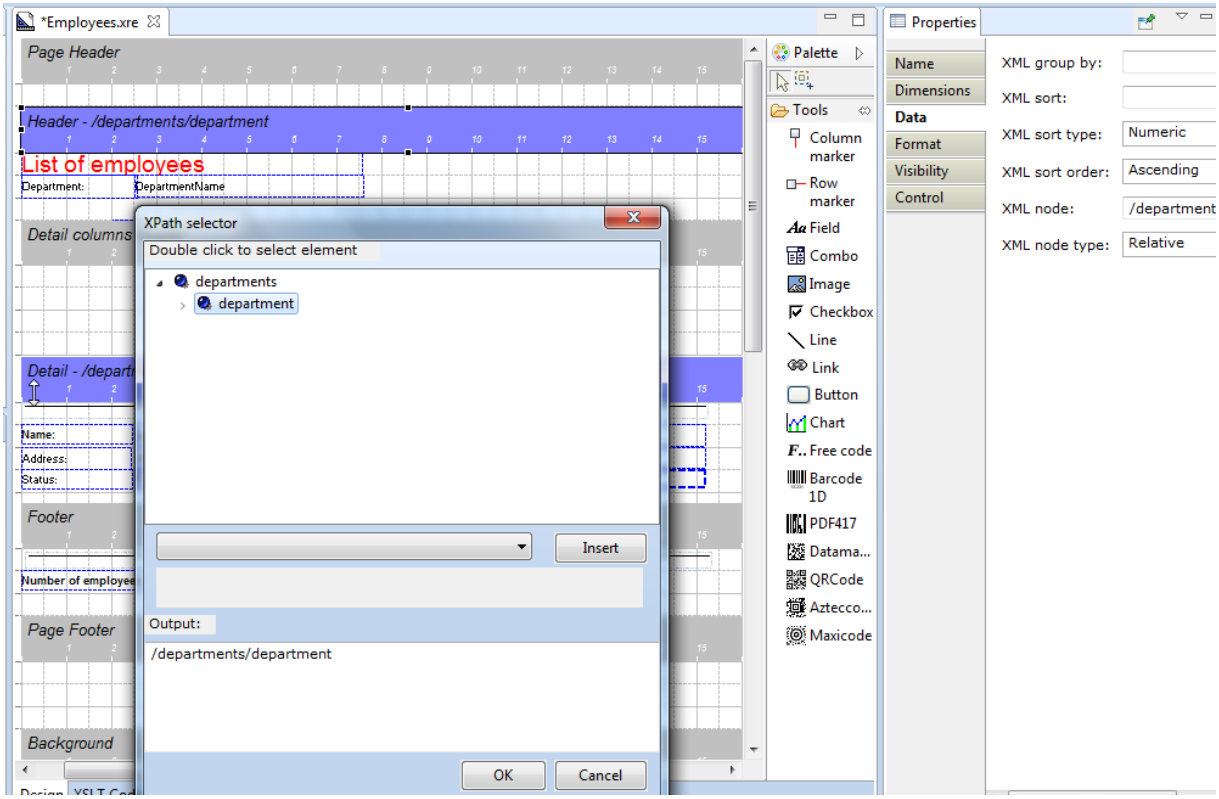




3. Our template will use 2 areas, the header area for the name of the department and the

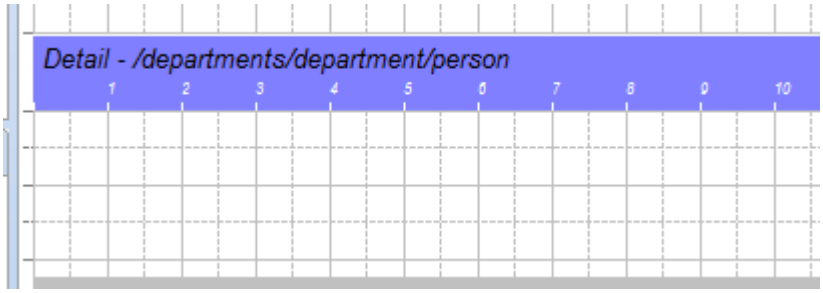


detail area for the employees. So first we will tell FO Designer we want to have a header area for each department in the XML document. So we click on the header area title button to display the properties of the area and we set *XML node* to */departments/department*.

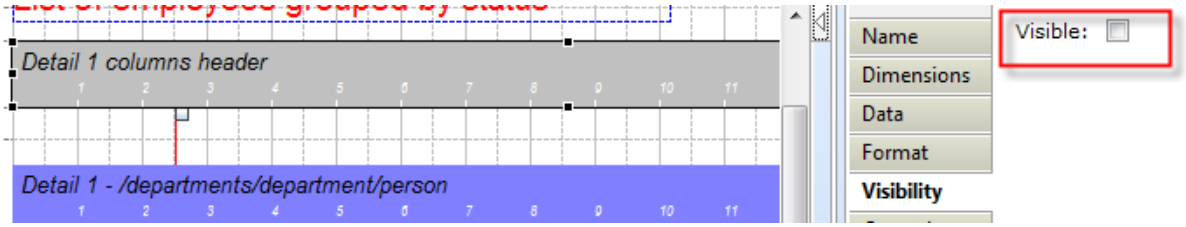


furthermore we set the ***New page*** property (in the control tab) to ***true*** because we want to have a new page for each department.

4. Now we do something similar with the detail area, we click on the detail are title button and set the *XML Node* to */departments/department/person,* since we want to have a detail line for each employee, furthermore, you have to make sure the *XML Node type* is relative since we want to list only the employees of the current department.

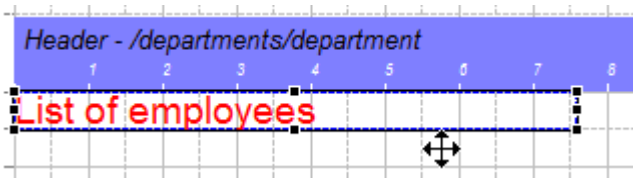


Note also we have set visibility flag in the detail heaer area to *false* because we will not use that area.



5. We will now add some fields to the header area. Click on the button

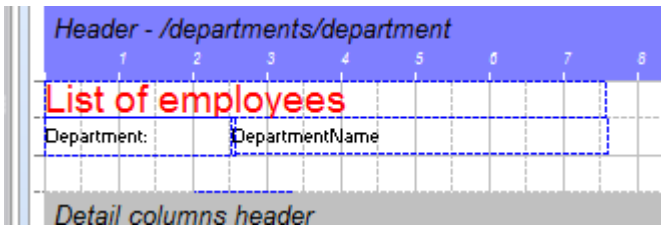
and place the following fields in the header area:



the only properties you must change are:

- value = List of employees
- font = size 18
- font color=red

6. next you add 2 more fields



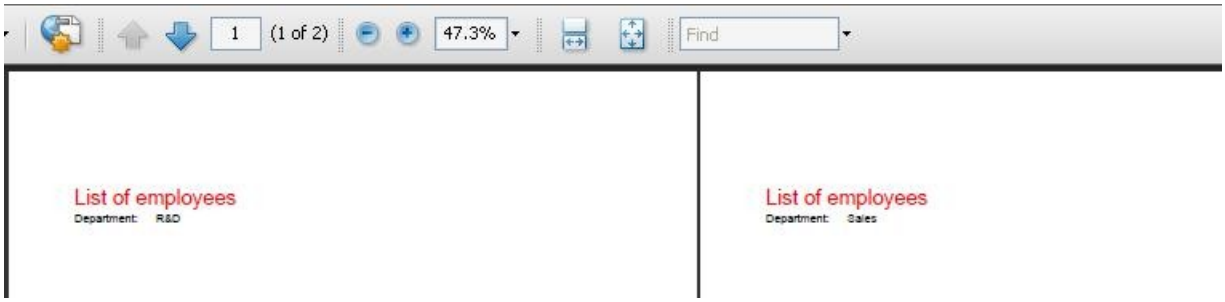
the properties you have to set are:

- **value** property of the **left** field = Department:
- **value** property of the **right** field= empty

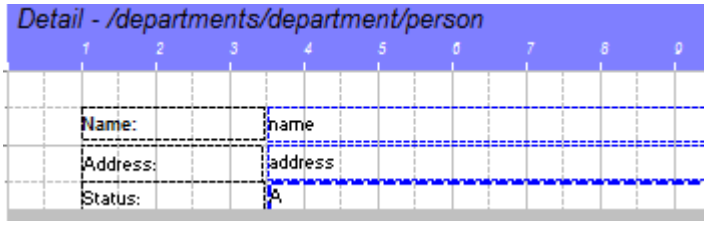
- **name** property of the **right** field= DepartmentName
- **XPath** property of the **right** field= /departments/department/departmentName

7. Now we can test the template by clicking on the PDF button

   The output will be a PDF file with two pages:



8. As next step you can proceed adding the fields in the detail area:



the three fields on the left have a **black** border because we have defined the *constant* property to true. For all these 3 fields you have to:

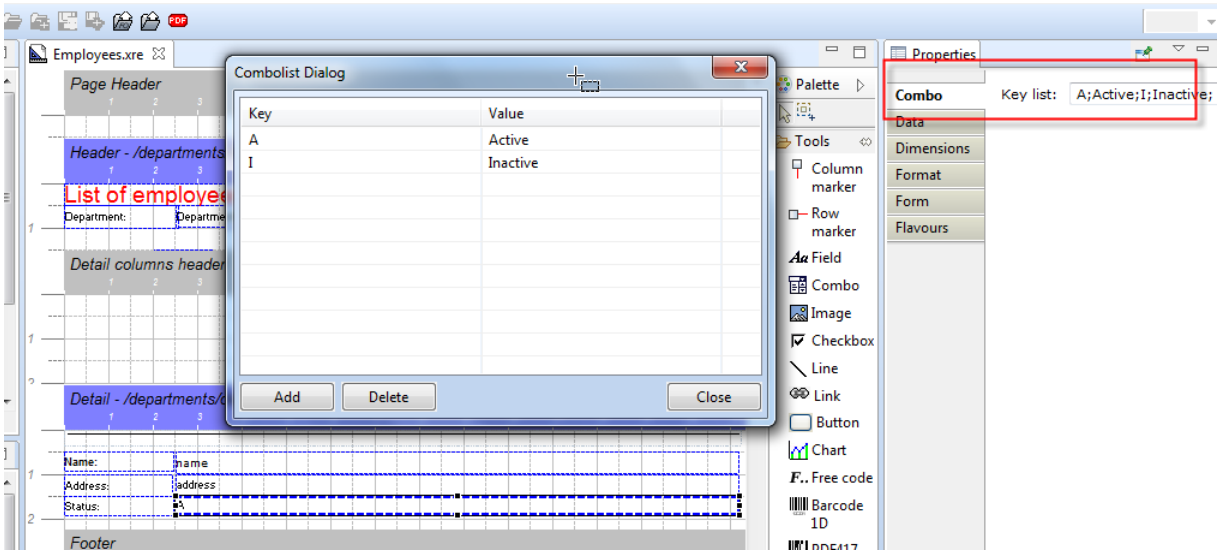
- drag the field tool

  Field

  and to the place in the area
- change the constant property to true and the value property to the value you see in the screenshot (Name:, Address: and Status:)
- Add 2 variable fields (the ones with blue borders) and select as Xpath

  /departments/department/person/name and */departments/department/person/address*
- the third variable field is actually a combobox and must be defined like this:
- Drag the combo box tool to the area

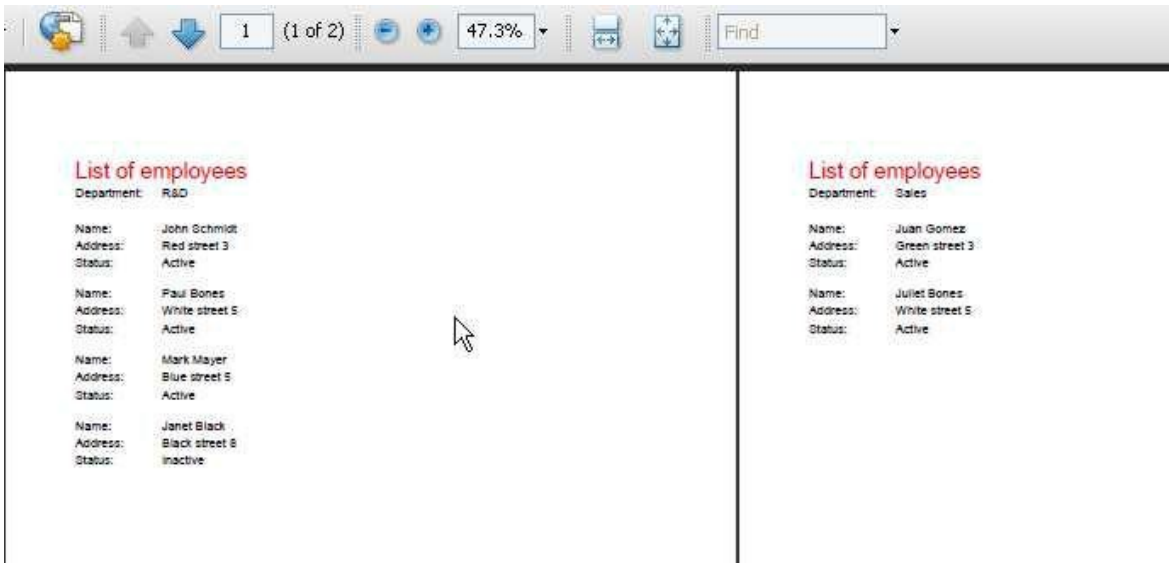
  Combo
- Set the XPath to */departments/department/person/status*
- The Keys list must be defined as in this screenshot:



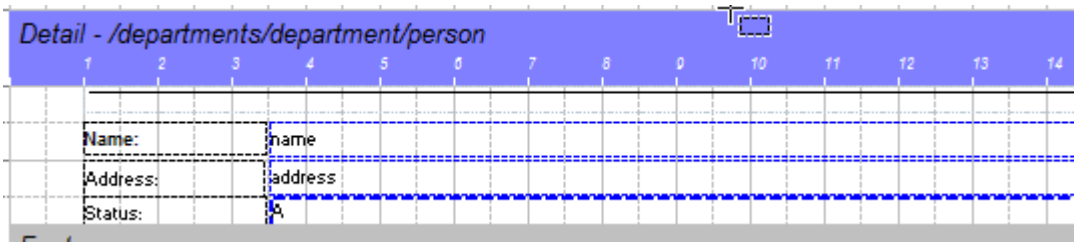
in this way the values *A* and *I* which are contained in the XML document will be replaced with the more descriptive values "Active" and "Inactive".

9. We can test the template by clicking on the PDF button

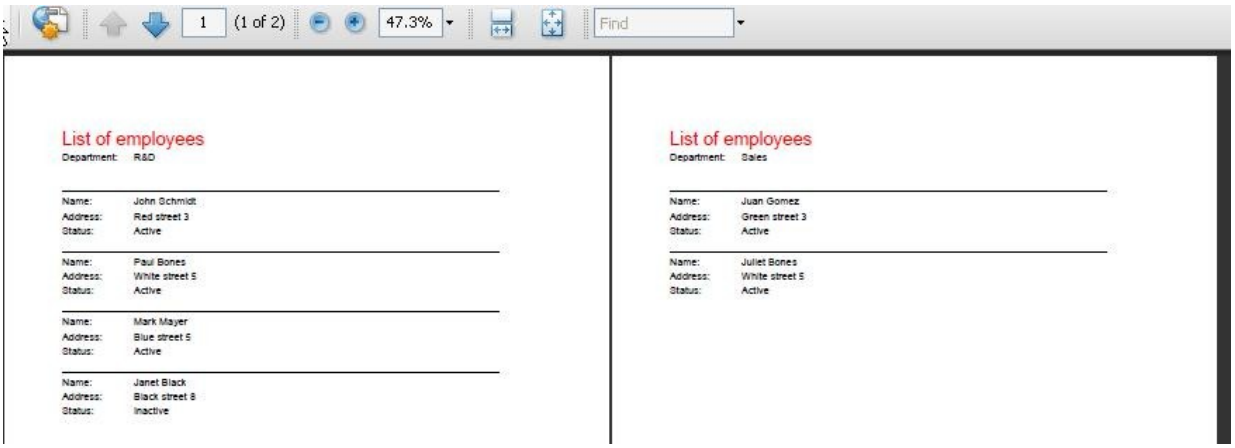
10. The ouput will now contain the detail information (persons):



11. we can also add a line to separate the persons, note we added in this screenshot a line object:



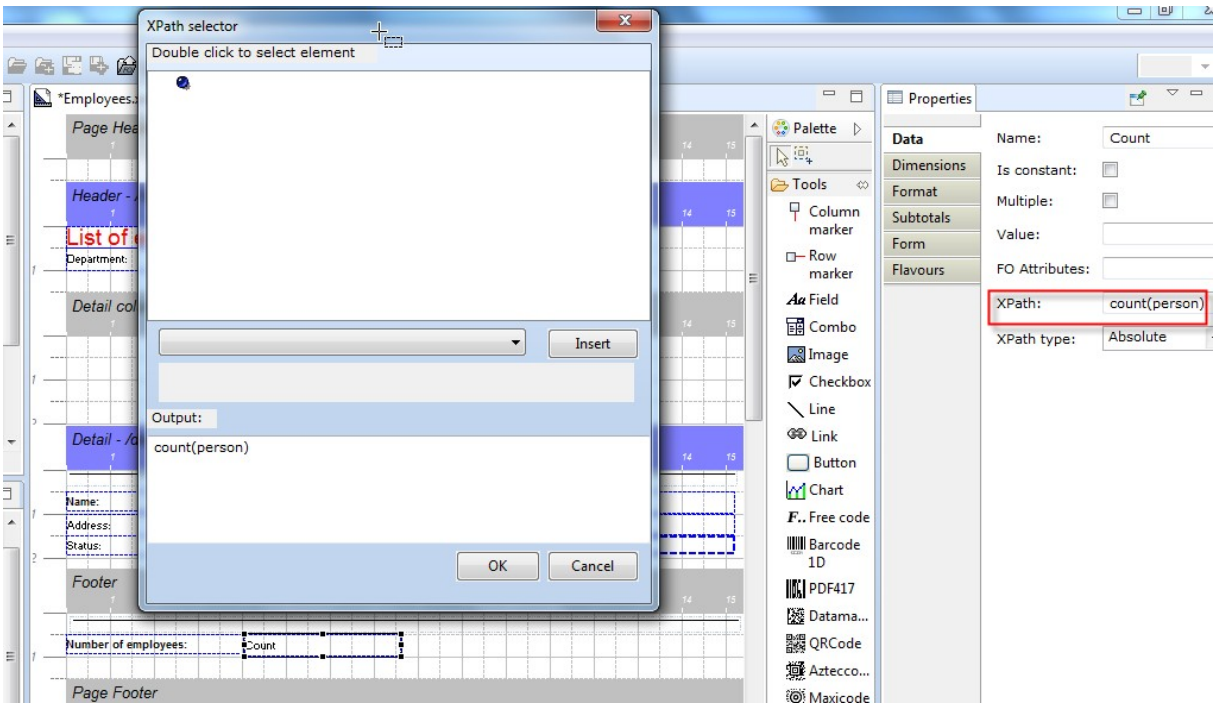
12. The output is now:



13. And last we will add a counter field in the template footer. This field has the following special handling:

- We have used the *count()* function in the Xpath to counter the number of persons in the department.
- We use the *XPath type* absolute which means the Xpath will be issued exactly
- as we enter it.
- The XPath itself references only the *person* node. The reason for this is, the footer is implicitly dependent on the header and the header is associated to the *department* node.

It would also be correct to set the *Xpath type* to *relative* and the XPath to *count(/departments/department/person).* However the use of functions and the xpath type relative is in this version of the designer not supported.



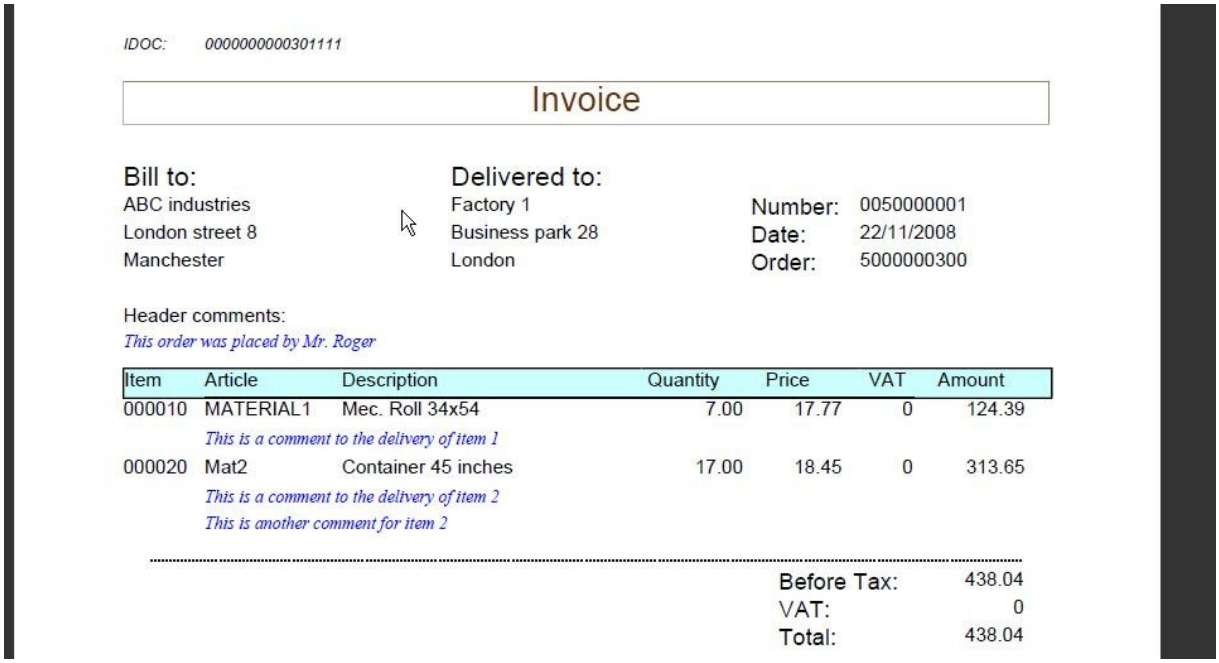
7. The invoice IDOC example explained

The example placed in the idoc_invoice subdirectory shows how you can create a user friendly PDF file from a SAP XML IDOC. This example has been based on the INVOIC01 format but it can be easily changed to be used with other IDOC types or versions.

In this section we will highlight some common task like:

- How to work without a schema file.
- Use of conditions in the XPath
- Use of second level detail areas.
- Use of comboboxes

The PDF ouput of the invoice will be:



IDOC: 0000000000301111

Invoice

Bill to:
ABC industries
London street 8
Manchester

Delivered to:
Factory 1
Business park 28
London

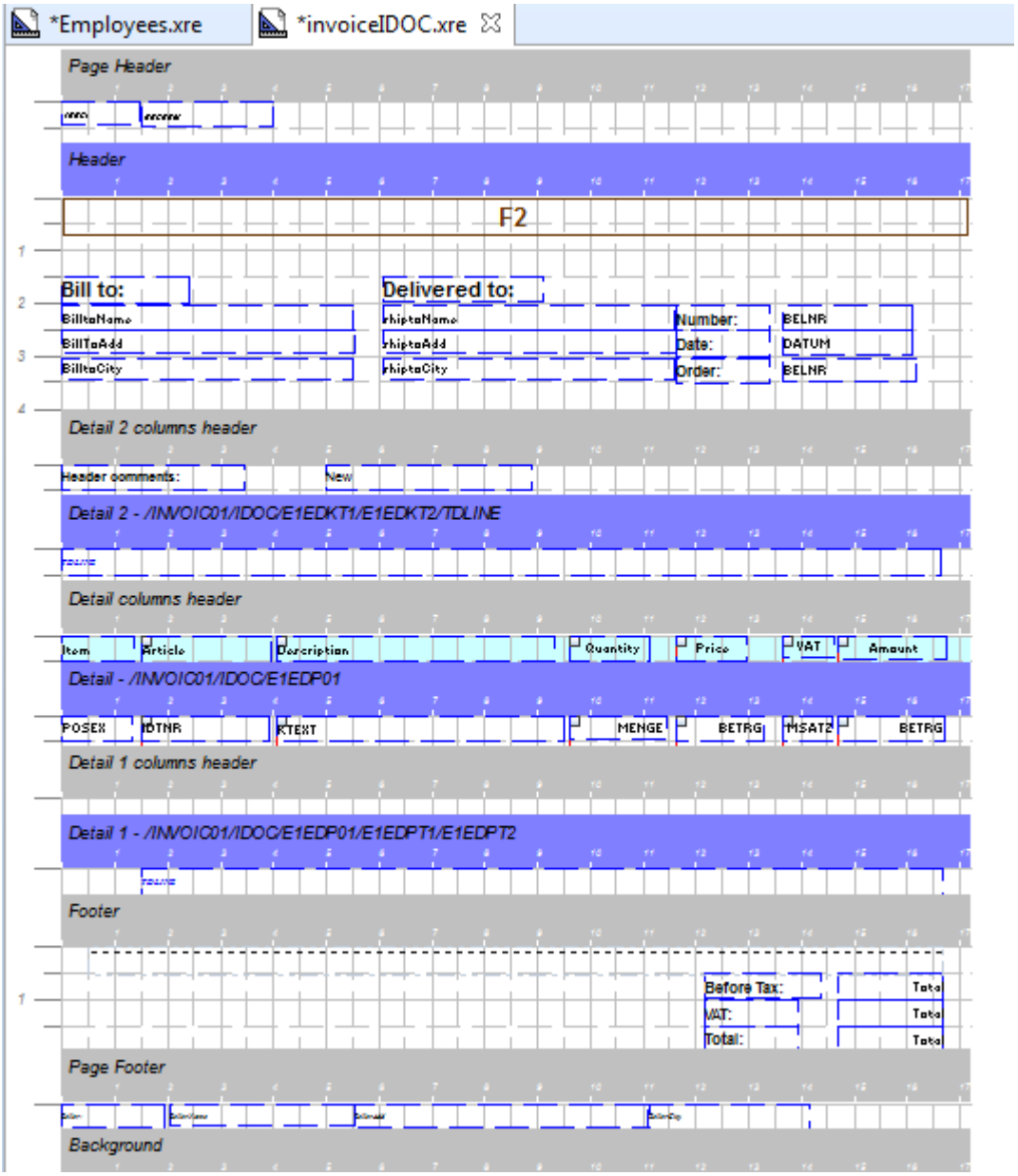
Number: 0050000001
Date: 22/11/2008
Order: 5000000300

Header comments:
This order was placed by Mr. Roger

Item	Article	Description	Quantity	Price	VAT	Amount
000010	MATERIAL1	Mec. Roll 34x54	7.00	17.77	0	124.39
	This is a comment to the delivery of item 1					
000020	Mat2	Container 45 inches	17.00	18.45	0	313.65
	This is a comment to the delivery of item 2					
	This is another comment for item 2					

Before Tax: 438.04
VAT: 0
Total: 438.04

The report's *invoiceIDOC.xre* layout is:



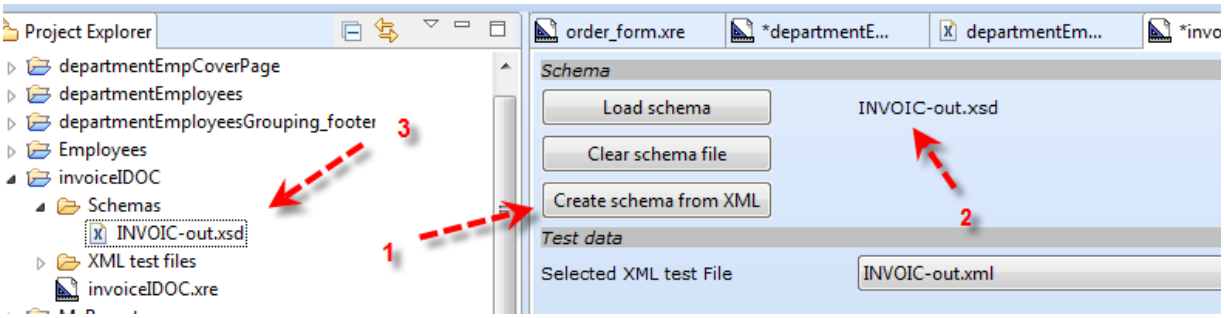
The structure of the template is:

1. There is a Page header that contains the IDOC number
2. There is a Invoice header that contains the buyer information (bill to and delivered to)
3. after the invoice header there may be (optional) some comments to the invoice: Note the Detail 2 area has been associated to the /INVOIC01/IDOC/E1EDKT1/E1EDKT2/TDLINE node. There is one such node in the XML document for each comment line, so the area will be repeated for each comment line. If no comment lines exist, the area will not be created at all. The detail 2 area also has a detail 2 columns header area, this area will be created also only if at least one comment line exists. Furthermore the *super area* property of the Detail 2 area has been set to the header area.
4. after the comments there is a table header for the items in the invoice
5. The detail area will be repeated for each item in the invoice. Note the Detail area has been associated to the node /INVOIC01/IDOC/E1EDP01. This element exist for each item in the invoice. Furthermore the *super area* property of the Detail area has been set to the Header area.

6. for each item, there may be (optional) one or several comment lines. Note the Detail 1 area has been associated to the node /INVOIC01/IDOC/E1EDP01/E1EDPT1/E1EDPT2 which contains the item level comments. Furthermore the *super area* property of the Detail 1 has been set to the Detail area.
7. The invoice footer contains some totals information. In the same way as the invoice header, the footer does not have to be associated to any XML node in the XML document since both areas exits only (and always) once in each invoice. This therefore assumes the input XML document (IDOC) can contain only one invoice.
8. The page footer contains the legal information of the company issuing the invoice.

how to work without a schema file

In this example we do not have a XSD file that describes the IDOC invoice, therefore we click on the *create schema from XML* button in the settings tab of the report, select the INVOIC-out.xml file and we get a pseudo-schema (INVOIC-out.xsd) created from the test file:

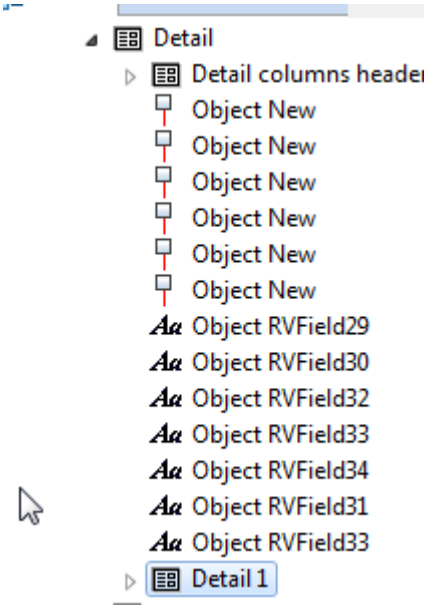


Use of second level detail areas

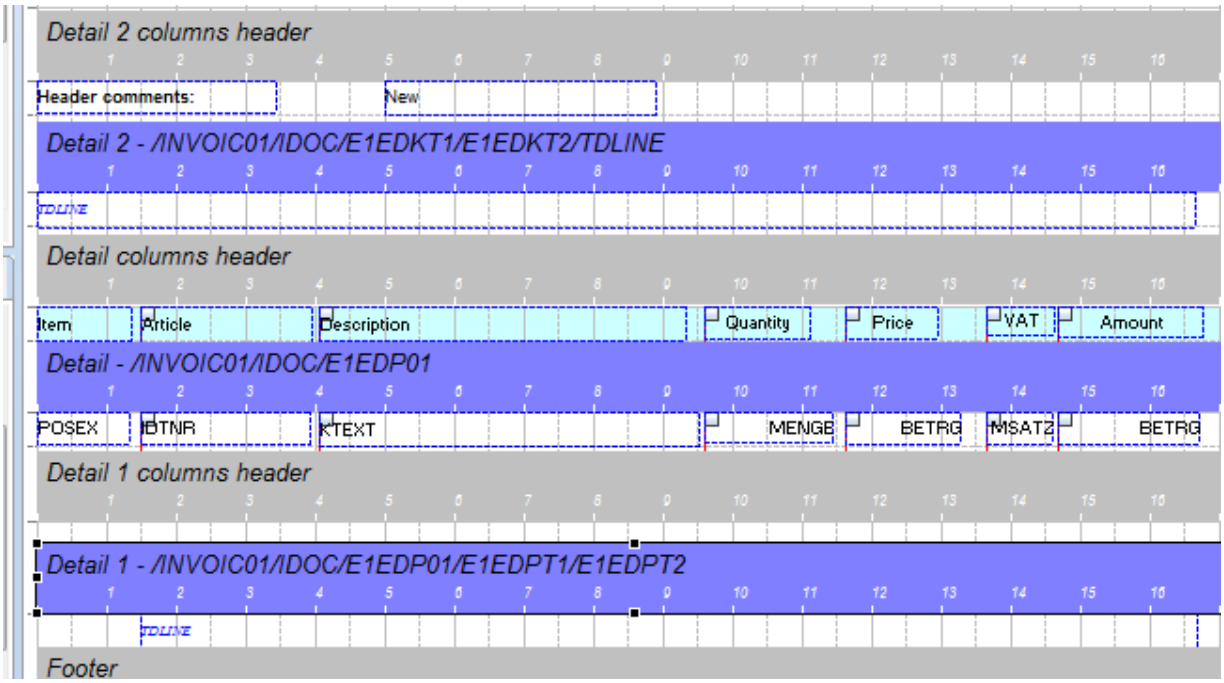
As already described in the structure of the template, this example contains a second level detail area called "detail 1". This area contains the comments associated to the items of the invoice.

The key steps to achieve this are:

- An area (Detail 1) has been inserted below the Detail area



- The *XML node* property of *Detail1* is pointing to the /INVOIC01/IDOC/E1EDP01/E1EDPT1/E1EDPT2 which contains the comments of the item.
- The *XML node type* is set to relative, since the super area is pointing to /INVOIC01/IDOC/E1EDP01, the XML node property will refer to the comments of the current line only.

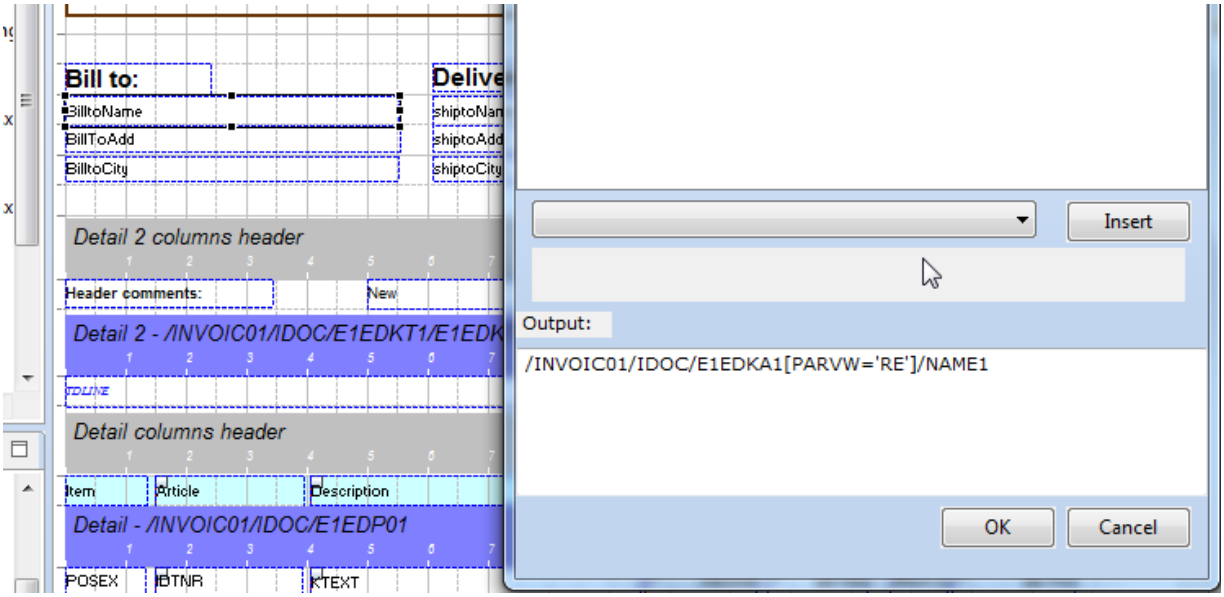


Use of conditions in the XPath

The bill to and delivered to information show a very common pattern in XML structures. The XML node */INVOIC01/IDOC/E1EDKA1/NAME1* contains the name of a company, but the meaning of the company is described by another node, namely the */INVOIC01/IDOC/E1EDKA1/PARVW.* If the content of that node is RE, it means the data in the parent node refer to the bill to party. That is why the used Xpath is:

/INVOIC01/IDOC/E1EDKA1[PARVW='RE']/NAME1

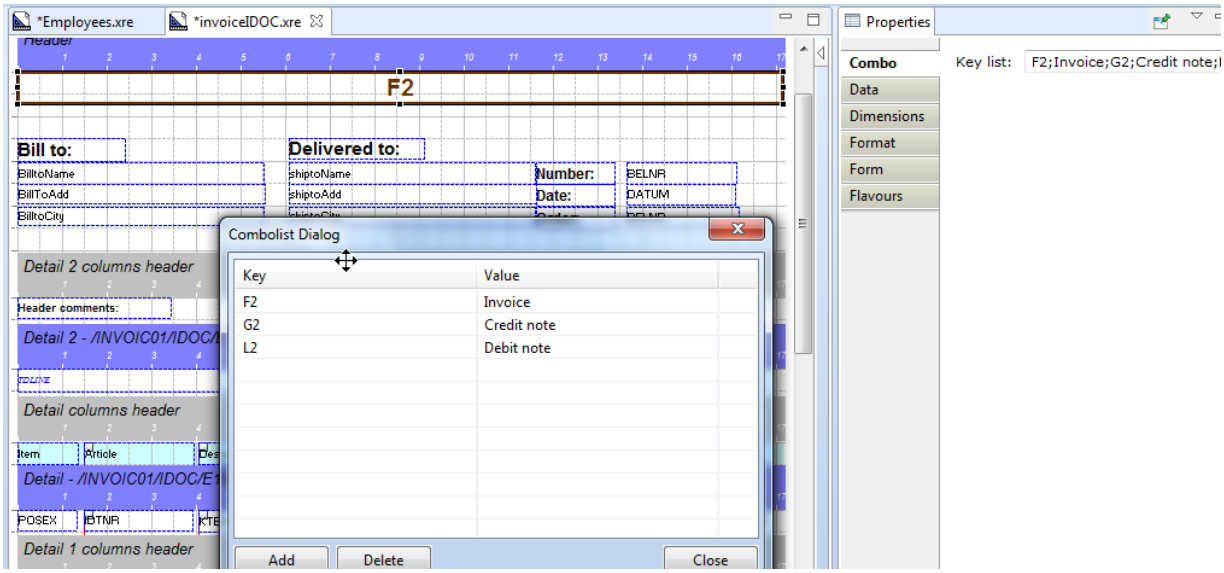
which means , select the *NAME1* node in the */INVOIC01/IDOC/E1EDKA1* node that meets the condition *PARVW='RE'.*



Note you have to **type in** the condition part of the Xpath in the bottom field of the *XPath selector* window. You can add condition in square brackets [ ] at any position of the XPath.

Use of combo boxes

The first field in the invoice header is a good example for a combobox. The XML node */INVOIC01/IDOC/E1EDK14[QUALF='015']/ORGID* can contain the values F2 or G2 which do not mean anything to a normal user. The combobox allows you to replace this codes with the descriptive values "invoice" and "credit note" which are the business meaning of those code.



8."Group by" example

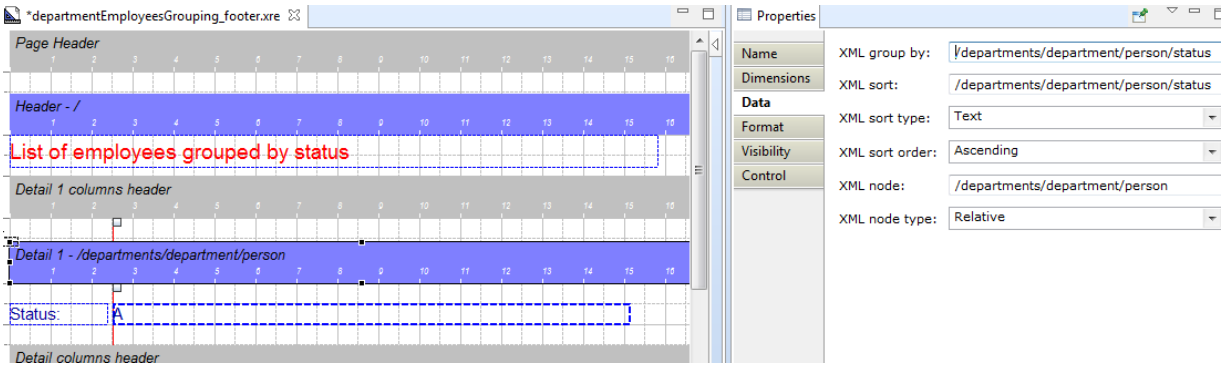
In the database there is an example called departmentEmployeesGrouping.xre. This example shows how to use the "group by" feature. The example takes as input a list of employees and groups them by status (in the test file there are 2 status, active and inactive).

For grouping 2 areas are required,

one group header and one group detail:

- The group header must contain the value use for grouping in the "XML group by" property. In this example the status element. You must also set the value for "XML Node", these are the nodes you will be grouping, in this case persons.

  Note, starting with version 2.1 it is possible to group by up to 3 fields.



In the group header you can output the value of the element you are grouping by (in this case *status*).

Group detail areas have the following restrictions:

- the value for the XML Node property will be inherited from the header.
- the XML Group by property may not be used.

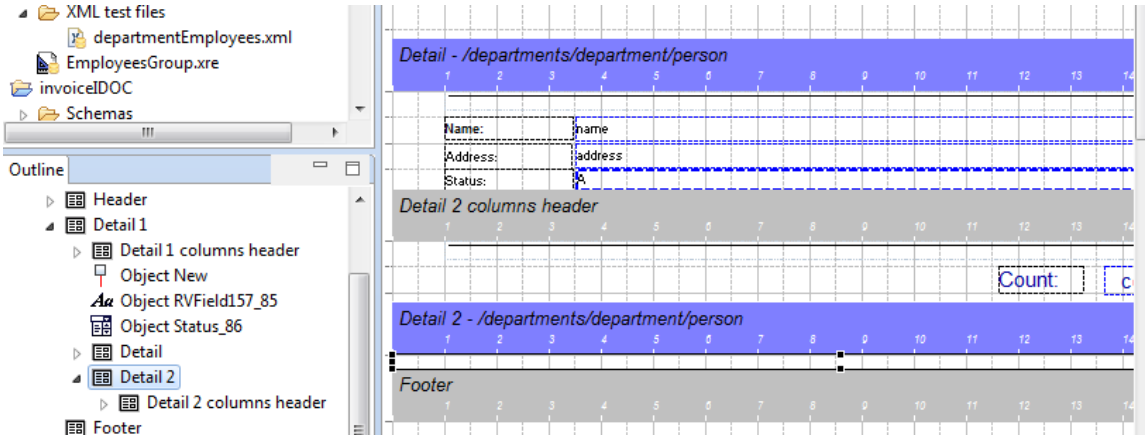
Note nested groups are not supported.

Group footers

The example *departmentEmployeesGrouping_footer.xre* shows how to create group footers. Group footers are used to show total fields of the values in the group for each value being grouped. In the example the group footer is used to show the total number of employees for each status.

Group footers have to be implemented as follow:

· create a new area placed after the group detail area. In the example the footer area is the "detail 2" area.



· The footer area must have as "super area" the "group header". In our example the super area of "detail 2" is "detail 1" which is the group header.

· If you want to count or sum the fields of the current group you use an xpath expression like this one:

count(/departments/department/person[status = current()/status])

this means, count all persons (/departments/department/person) whose status has the same value as the current's group status. Note we are grouping by status so when this is executed for the "active" employees status, it will count all employees whose status is "active".

Let's assume each person has a child element called "salary", we could sum all salaries of the employees in the current status like this:

sum(/departments/department/person[status = current()/status]/salary)

this means select the salary of all employee whose status is the same status as the current group and then sum all selected salaries.

9. The two columns example

The example *tasksTwoColumns* shows how to use Xpath to create a two columns report. The output will look like this:

List of Tasks

Copyright 2017
http://www.java4less.com

HR software needs upgrade *Pam King*	Apply Billing System updates *Russ Sanders*
Investigate new Virus Protection software *Pam King*	Arrange for holiday coverage *Al Bines*
Complete plan *Mark Nile*	Check software licenses *Mark Nile*
Get RFPs for new server	Purchase backup server

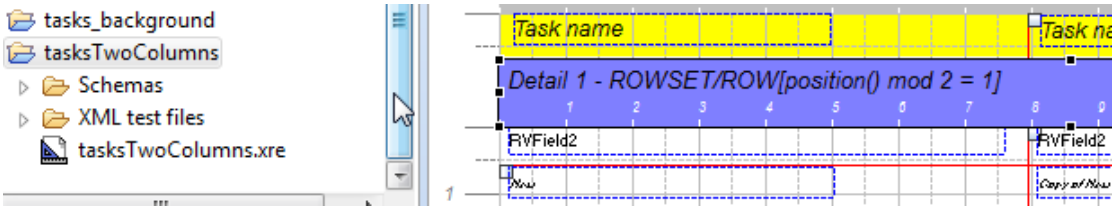
It is a list of tasks organized in 2 columns, with the task name and the employee assign to the task.

The layout of the report is:

- there is a detail area which will contain 2 task bame fields (2 columns). Therefore the area will need to be repeated for every second task. The Xpath to achieve this is:

  ROWSET/ROW[position() mod 2 = 1]

  this will select every second „ROW“ element in the XML



- In the area there are 2 TaskName field. The Xpath of the fields are:
  - *TASK_NAME* : to select the TASK_NAME field in the current ROW node
  - *TASK_NAME/../following-sibling::ROW/TASK_NAME:* to select the TASK_NAME field in the next ROW after the currently selected ROW (this will be the second column)

    The XPATH expression means:

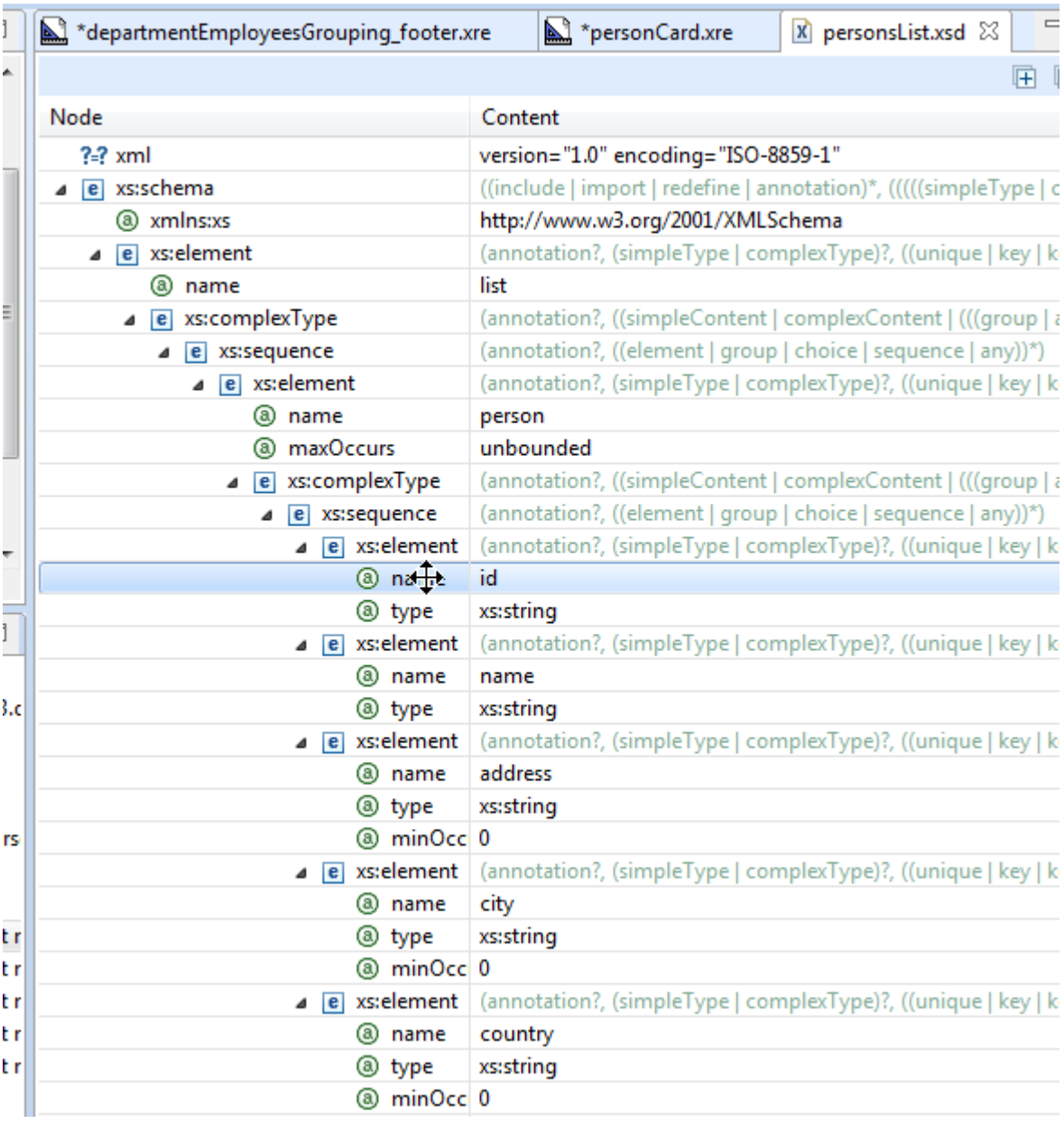
    - select current TASK_NAME
    - use the /../ to select the parent node (the ROW node)
    - *following-sibling::ROW:* will select the next ROW
    - /TASK_NAME selects the field in the ROW

10. The Barcodes and dynamic images example explained

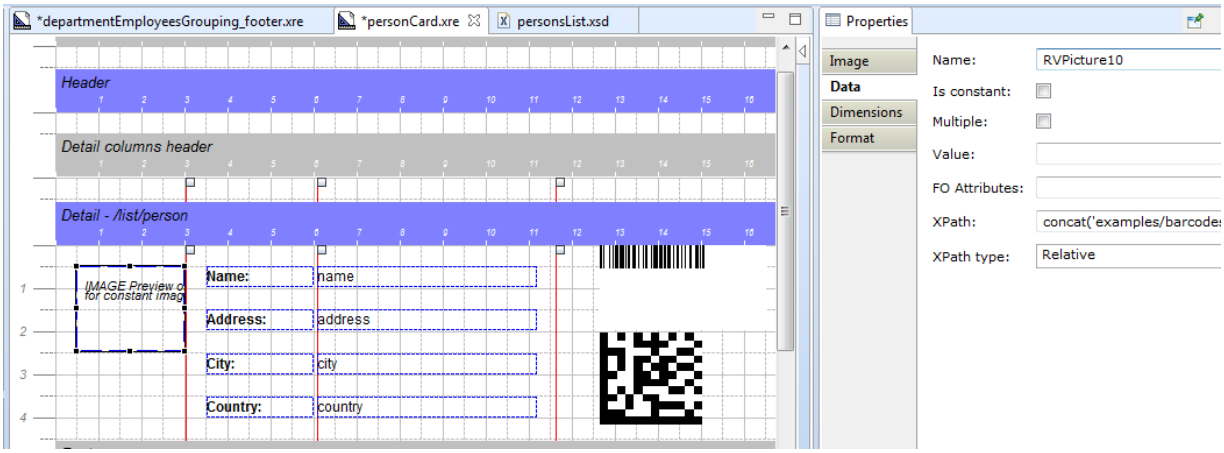
The example *personCard.xre* in the database which shows how to:

- use barcodes
- dynamically select an image

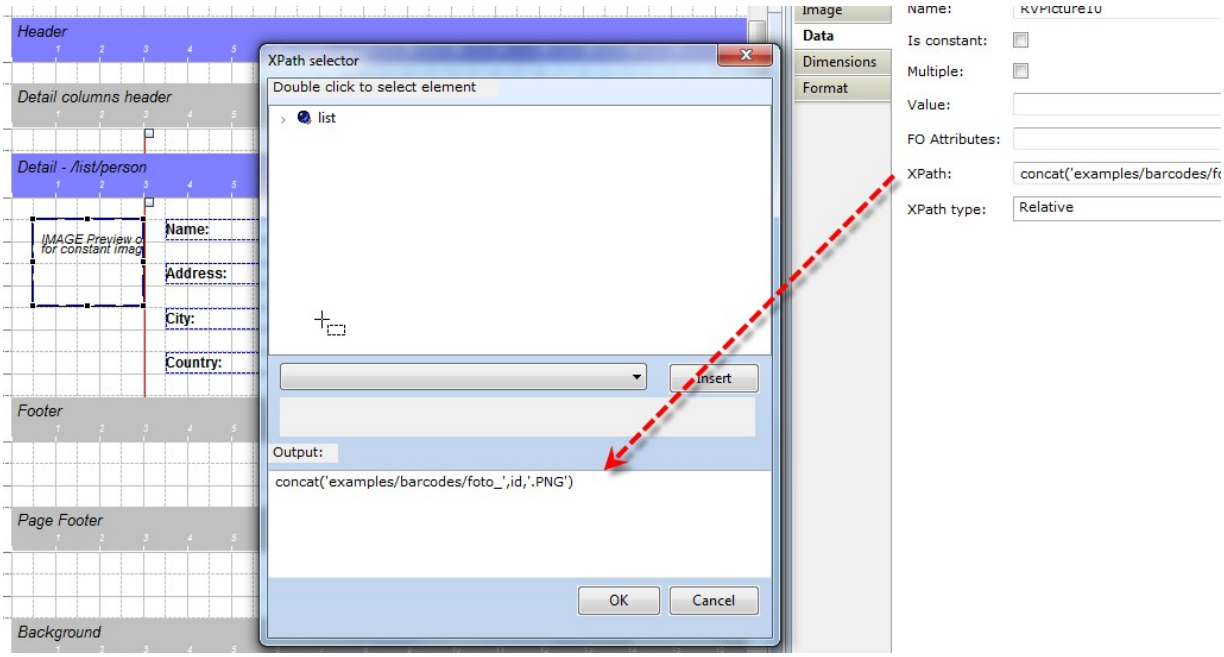
This example creates a personal card with the name and address of a person together with the photo and a barcode which encodes the name of the person. The input file is a XML file with a list of persons and their information, the schema of the XML file is very simple:



The card has been split in 4 columns, the first one contains the photo and the last one contains the barcodes. These are the 2 objects we are interested in.



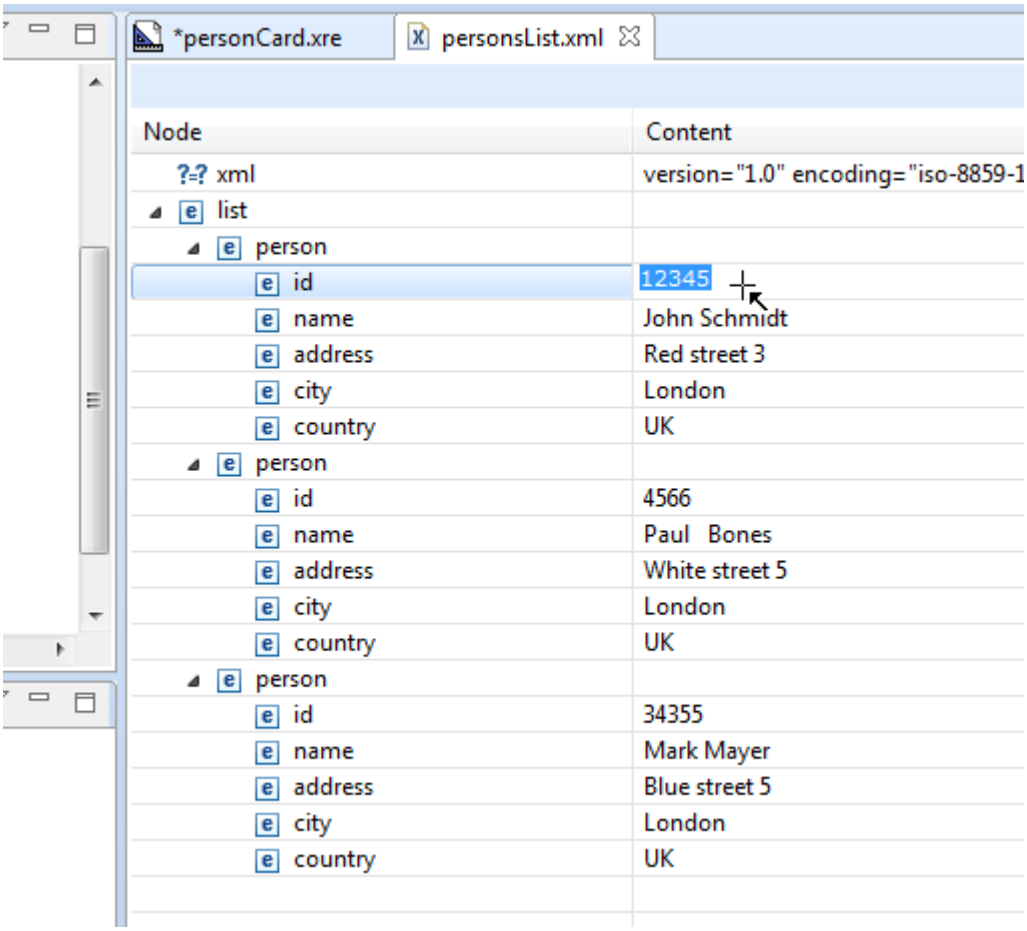
The image object could be a static image file if you enter a file name in the *image* field, however in this example we want to select a different image for each person (the photo), so we use the *xpath* field and enter the value *concat('examples/barcodes/foto_',id,'.PNG')*.



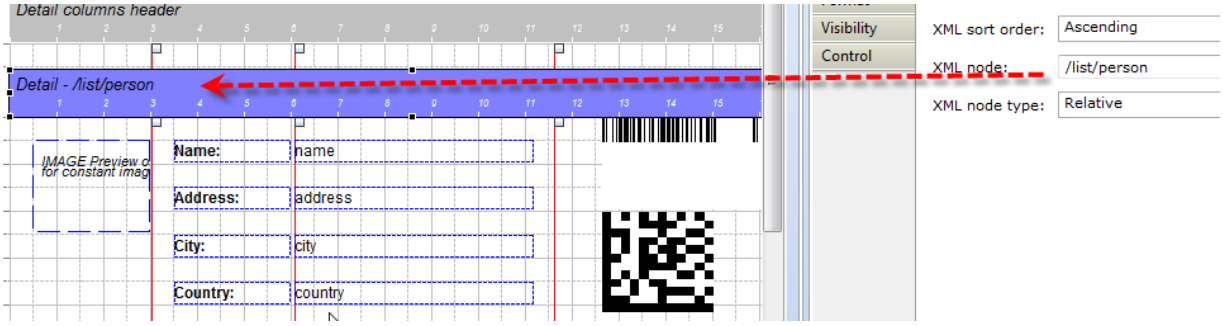
This expression concatenates the 3 values:

- examples/barcodes/foto_
- id of a person
- .PNG

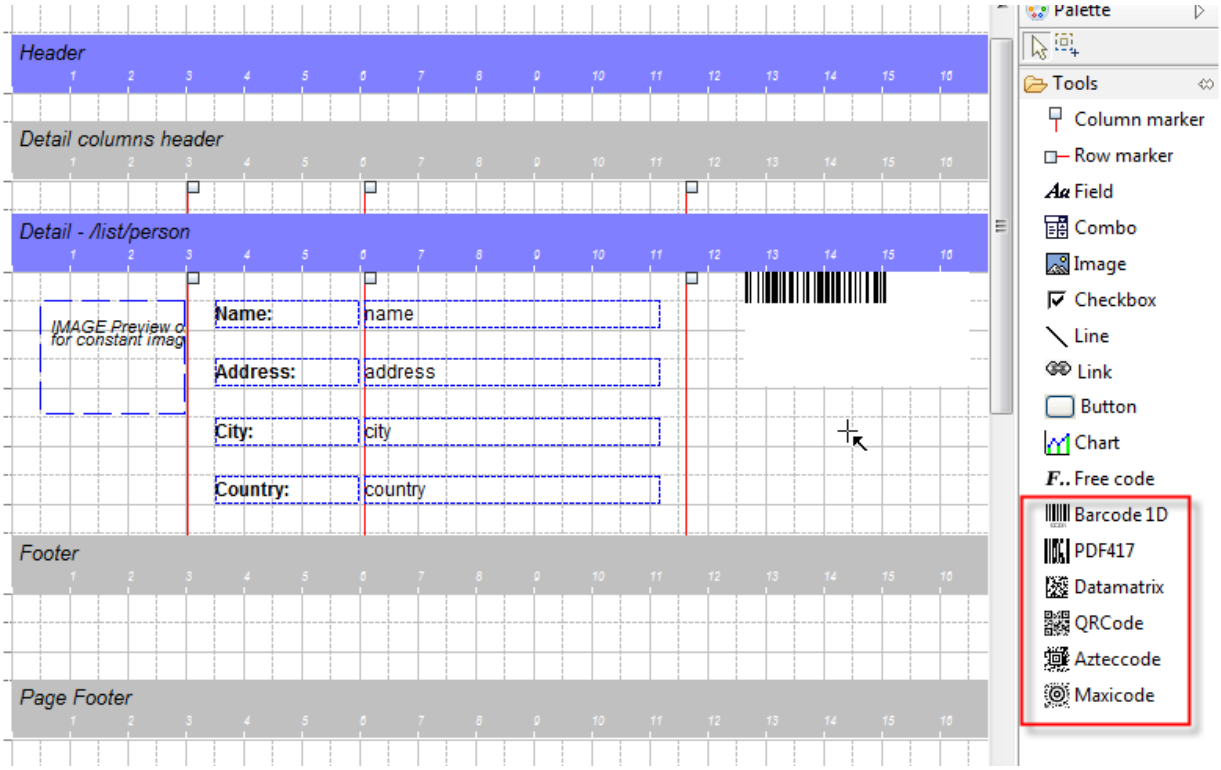
Since one of the persons has the id 12345 the result for that person will be *examples/barcodes/foto_12345.jpg*. So that is the file that will be used as photo for that person.



Note we used the value *id* instead of */list/person/id* in the *concat* function. Since the *XMLNode* of the detail area (the card) is */list/person*, we have to use the value *id* to find the correct value for the current person.

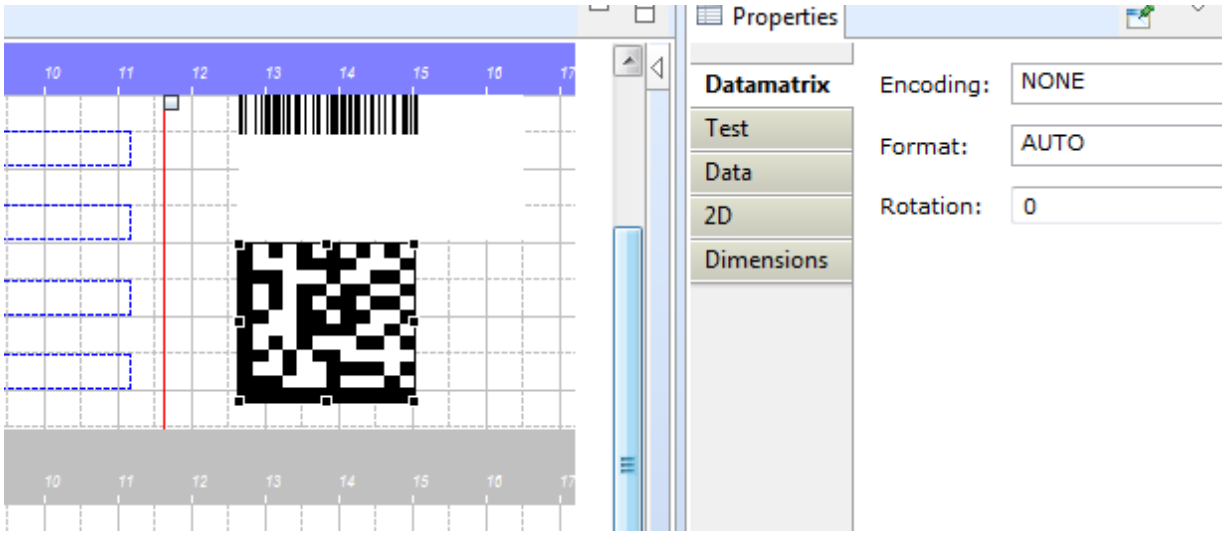


The barcode is the second interesting object in this example. When you add a barcode to your document the barcode type selection window will popup. At this point you have to decide if it will be a 1D barcode (e.g. EAN or Code128) or a 2D barcode (Datamatrix, PDF417 ...).



Once the barcode is in your document you can distinguish 2 kind of properties:

- In the Data section you can find properties which are common to all barcodes types. These are:
  - XPath: as with other objects this is the location in the XML document where the information for the barcode can be found. This is the information to be encoded in the
  - barcode, in our example the name of the person.
  - Binary base64: this is required if you are going to encode binary data. In this case the XML document must contain base64 encoded binary data and the xpath field must
  - point to a XML node which contains Base64 encoded data.
  - Process tilde: if true the tilde character (~) will be processed accordingly. Please refer to the specific barcode documentation (see link below).
- In the Format section the barcode type specific properties are located. In this Datamatrix example, you can see the datamatrix encoding and format properties. You can find more information about the barcode specific properties here **http://www.java4less.com/barcodes/barcodes.php?info=guide**



11. The chart example explained

The directory examples\chart contains an example that shows how to use J4L RChart within the designer. The usage of RChart involves 2 steps:

1. First you have to design the chart independently of the FO Designer. The documentation for RChart is here:

   http://www.java4less.com/charts/userguide/tutorialcontents.hml

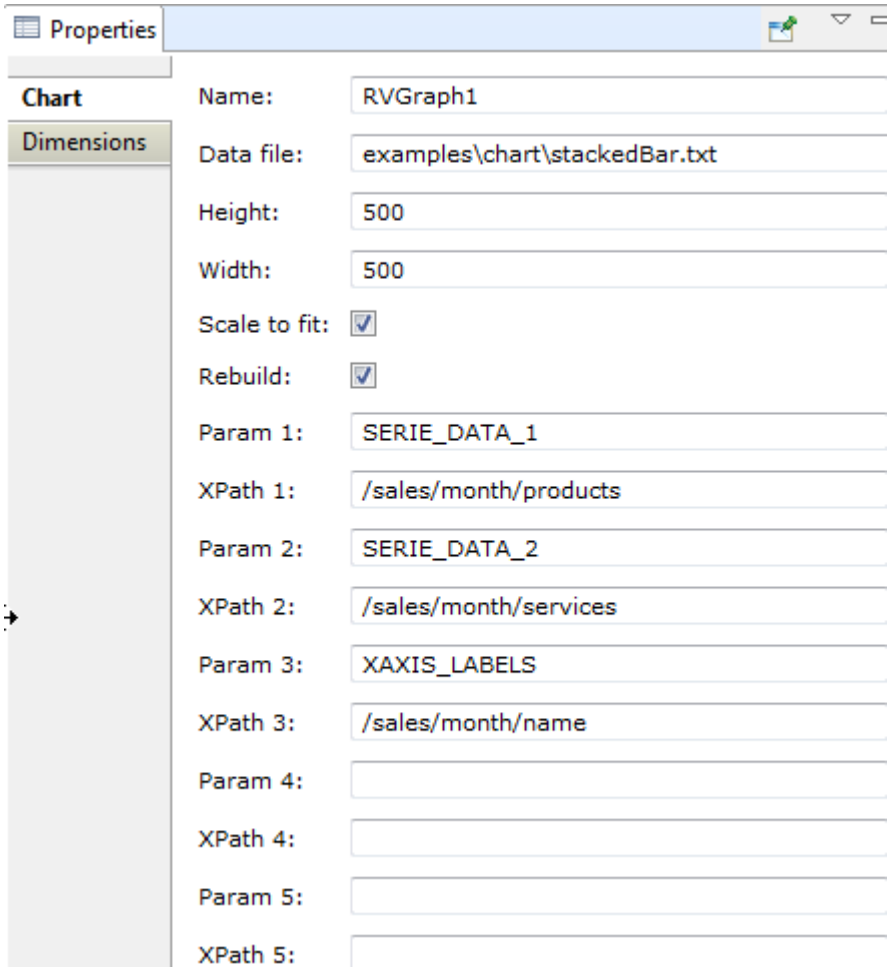
   Optionally you can use the **Visual Builder** to design you chart.

   In any case the output of the chart designer process is a parameters file (see example\chart\stackedBar.txt) which contains the chart and some test data you have used during the development.

   We do not recommend to develop a new chart from scratch but take one of the existing example and modify it according to your needs. You can download the **Rchart evaluation package** and you will find many examples in the data\examples directory.
2. in the second step you create your report, add a chart object and specify the following properties (see screenshot below):

   - the "Data File" field must point to the chart's parameter file
   - the parameters of the data file whose values must be overwritten by the values from the XML file. You can enter up to 9 of these parameters.

     In the example the values for the parameters SERIE_DATA_1, SERIE_DATA_2 and XAXIS_LABELS will be read from the XML elements defined in the "XPath" fields (thus the test values for these parameters in the stackedBar.txt file, will be overridden).


Properties
Chart
Dimensions
Name:
RVGraph1
Data file:
examples\chart\stackedBar.txt
Height:
500
Width:
500
Scale to fit:
Rebuild:
Param 1:
SERIE_DATA_1
XPath 1:
/sales/month/products
Param 2:
SERIE_DATA_2
XPath 2:
/sales/month/services
Param 3:
XAXIS_LABELS
XPath 3:
/sales/month/name
Param 4:
XPath 4:
Param 5:
XPath 5:

12. The running totals example explained

A very commoun requirement is to have a page footer with the subtotal of the items in the current and previous pages and a total amount at the end of the document. The running totals example located in

examples\xcbl_order\order_subtotal.xrp

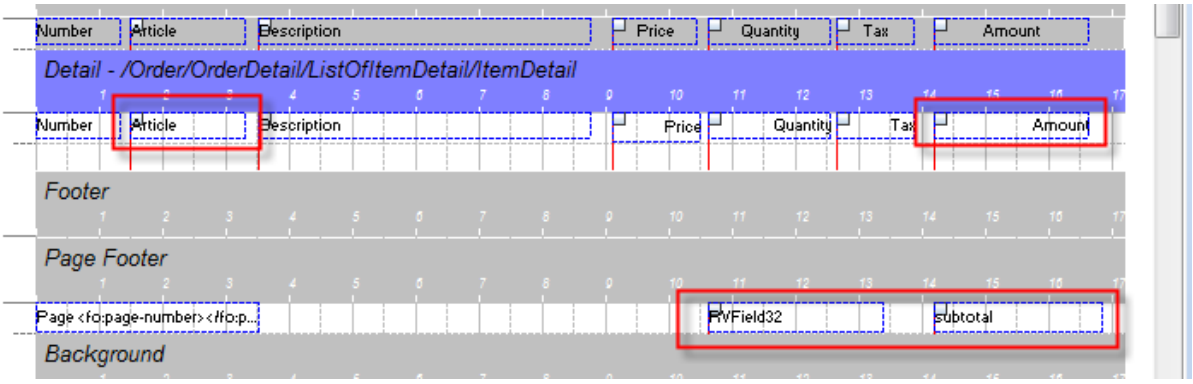
shows how create the subtotal item at the end of each page. The output of this example is a 2 page document, in the first page the label "Subtotal" will be generated, together with the sum of all items in the page

Page 1	Subtotal	24200.0

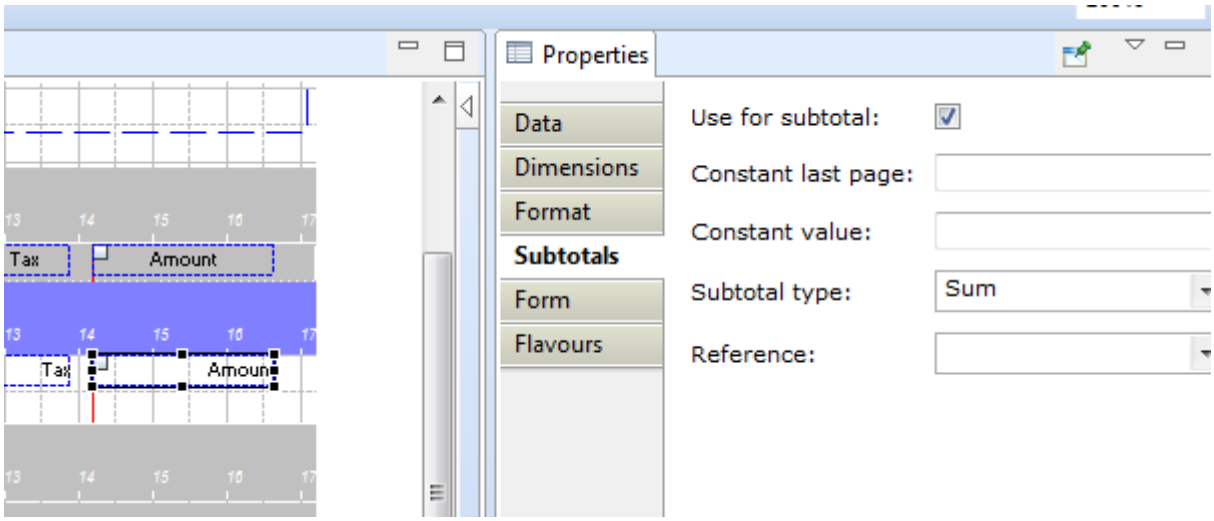
in the second page the text is "Total" and the value is the sum of all items in the document.

Page 2	Total	29200.0

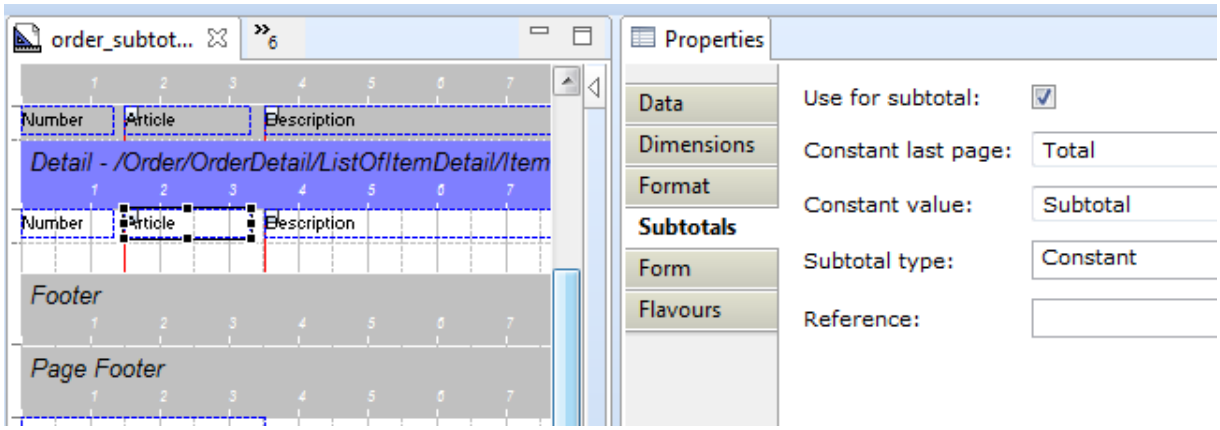
to achieve this result, two items in the detail area must be selected, one will be the *LineAmount* field which needs to be summed up, and the other can be any field, in this case the *article* field has been taken:



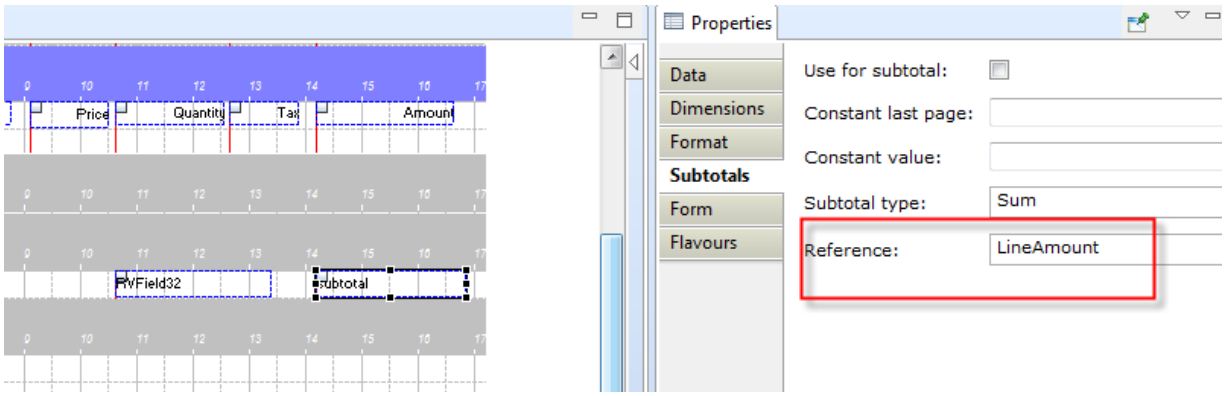
The *LineAmount* field has the property *Use for Subtotal* set to *Yes* and the subtotal type *SUM* will cause the values of this field to be summed up during the PDF generation.



The *Article* field will be used to create the "subtotal" and "total" labels. Note the subtotal type is now *CONSTANT* and the two constant value fields have been filled with the correct value.



To finish up the set up, the 2 fields in the page footer must reference the subtotal field in the detail section.



13. PDF interactive forms

Introduction

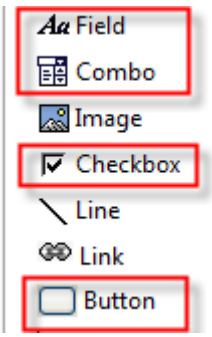
Starting with version 1.5 you can create PDF forms using J4L FO Designer (Suite edition), however this requires that you use our FOP server since the generation of PDF forms is not part of the XSL-FO standard.

You will find an introduction to the usage of and motivations for PDF forms at **this page in our site**.

Note the generation of forms requires the suite or enterprise license and the use of J4L FOP Server.

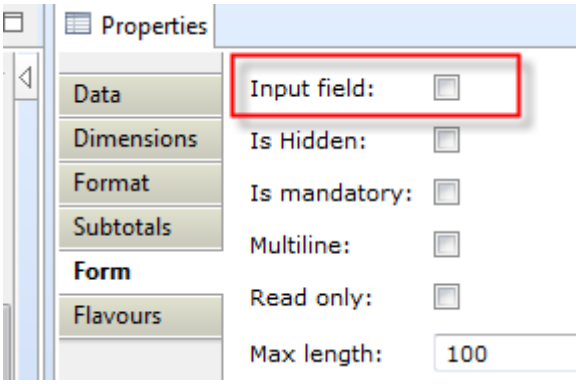
Form objects

The highlighted objects in the toolbar can be used for data entry:

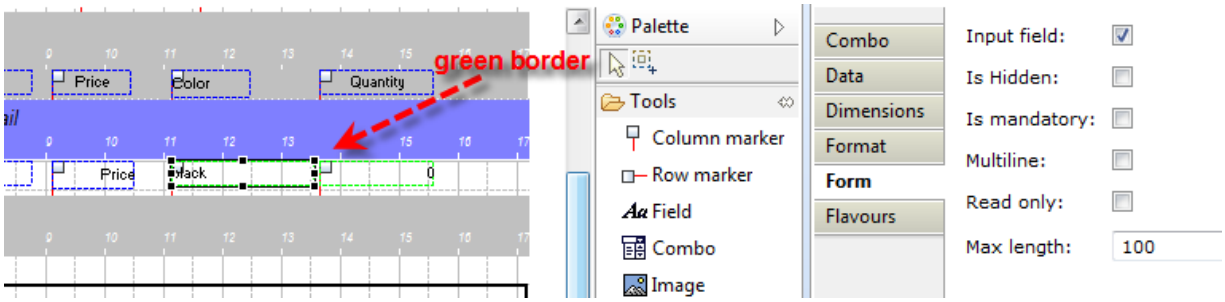


- Text fields.This object is also used in non-form PDF files to display data.
- Combo boxes. This object is also used in non-form PDF files to display data.
- Checkbox. This object can be used **in forms only**.
- Button (to submit or reset a form).This object can be used **in forms only**.

Text fields and combo boxes are normally used to display data. In order to activate them for data entry the "*Input field*" property must be set to yes.



Input fields have a **green** border in the designer.



Text field

Text fields can be used to allow free entry of data. The following properties are available for these kind of input fields:

- Read only: It is possible to set the field to read only, this is useful to prefill fields which you later can read when the user has submitted the PDF form. The read only field can contain a user id which you would use to identity the form.
- Is hidden field. If true the field will not be displayed in the PDF reader but the field is still contained inthe PDF file.
- Mandatory: the field must contain a value
- Multiline: several lines of text can be entered in the field.
- Max length: maximum number of characters that can be entered

The default value of the text field will be the constant one (see value property) or the one returned by the Xpath.

Combo box

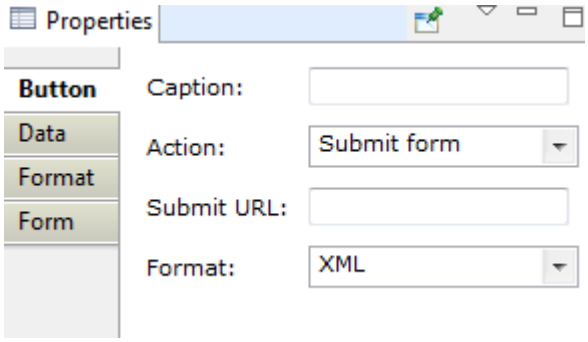
This kind of fields offer a list of values the user can select. They behave in a similar way as text fields.

Checkbox

Checkboxes are input fields which can only contain the value true or false (selected or not selected).

Button

Button objects are used to execute actions. The actions available at this time are:



- reset form to the default values.
- submit form. In this case you have to enter the url in the "Submit to URL" field. This can be:

  Email address URL, for example:

  1. *mailto:receiver@company.com?Subject=My%20Order&body=bla*

  2. A web server HTTP URL, for example *http://myserver/submit*.

     The submit URL value can also be filled dynamically from the XML data using the Xpath.

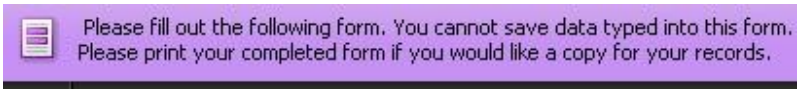
- The submission format, HTML, XML or PDF

Filling in the form

There are several ways to submit the forms:

1. submitting to a web server (as PDF, HTML or XML)
2. email (as PDF or XML)
3. printing and sending paper
4. sending the filled PDF file by any other means

The option (4 or partly 2) that requires you to save the filled form to a PDF file has a potential issue. If you use Adobe Reader you will get an error like this one "You cannot saved data typed into this form".



The reason for this is, Adobe Reader allows saving filled forms only if the forms has been created and digitally signed by Adobe tools. At this point you have some options:

1. Buy *Adobe LiveCycle reader extensions server* which will add the signature to the form.
2. Use *Adobe standard* to fill in the form (instead of the free adobe reader)
3. or use another reader.

   there are several free PDF readers that will allow you to fill and save forms. One of them is **Nitro PDF Reader**

Form submission

The PDF form can be submitted in several ways:

- to a web server using HTTP
- by email
- or saving the filled form to a PDF file and submitting by any other means.

There are several formats which can be used to receive the form data:

- HTML, the data will be submitted to your web server (the URL you have specified) using the HTTP POST method. The body on the HTTP request will be something like this:

  FieldName1=value1&FieldName2=value2&...
- XML. The format of the data when submitted using XML is:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<formData>
<fields>
<field  name="fieldname1"><value>value1</value></field>
<field name="fieldname2"><value>value2</value></field>
...
<fields>
</formData>
```

- PDF. Submitting in PDF format requires you to have a reader that allows you to do that (see previous section), it has the advantage that the user can keep a copy of the same file she has submitted. The receiver of the submitted PDF can also display the received data in a user friendly way.

  In order for your application to extract the content of the submitted PDF you have 2 options:
  - Use our servlet which converts PDF to XML. The file *J4LFOPServer.war* file contains a servlet, once this file has been deployed to a web server you can reach our servlet in this URL:

    http://servername/J4LFOPServer/FieldExtractorServer

    This servlet accepts HTTP POST requests which contain the PDF as body of the request, and returns the XML file with the field values.

    It also accepts files to be uploaded using an HTML form using the parameter FORMMODE=ON, see example FormRead.html inside *J4LFOPServer.war*.
  - Or use our Java API to extract tha values of the fields. This example shows how to do it:

```
import com.java4less.xreport.fop.FormValuesExtractor;
....
FormValuesExtractor extractor=new FormValuesExtractor();
FileInputStream is=new FileInputStream("myform.pdf");
Hashtable fieldsTable=extractToTable(is);

Enumeration enum=fieldsTable.keys();
while (enum.hasMoreElements()) {
    String key=(String) enum.nextElement();
    System.out.println("Field "+key+"="+fieldsTable.get(key));
```

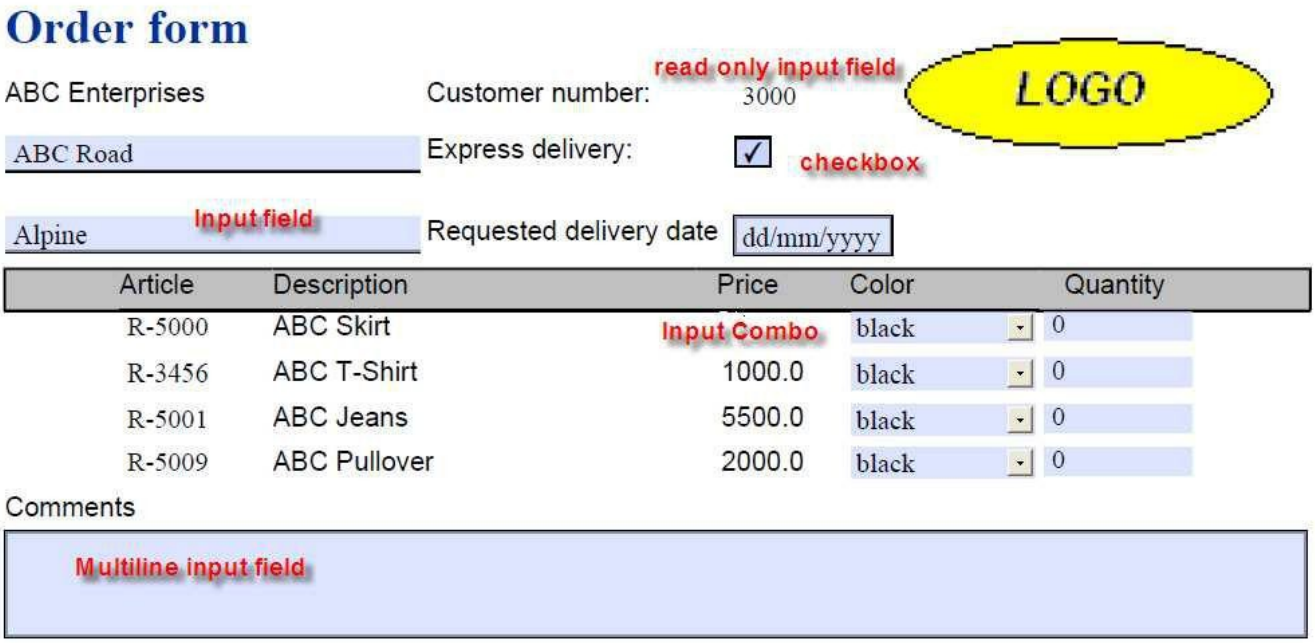
}

Input form example

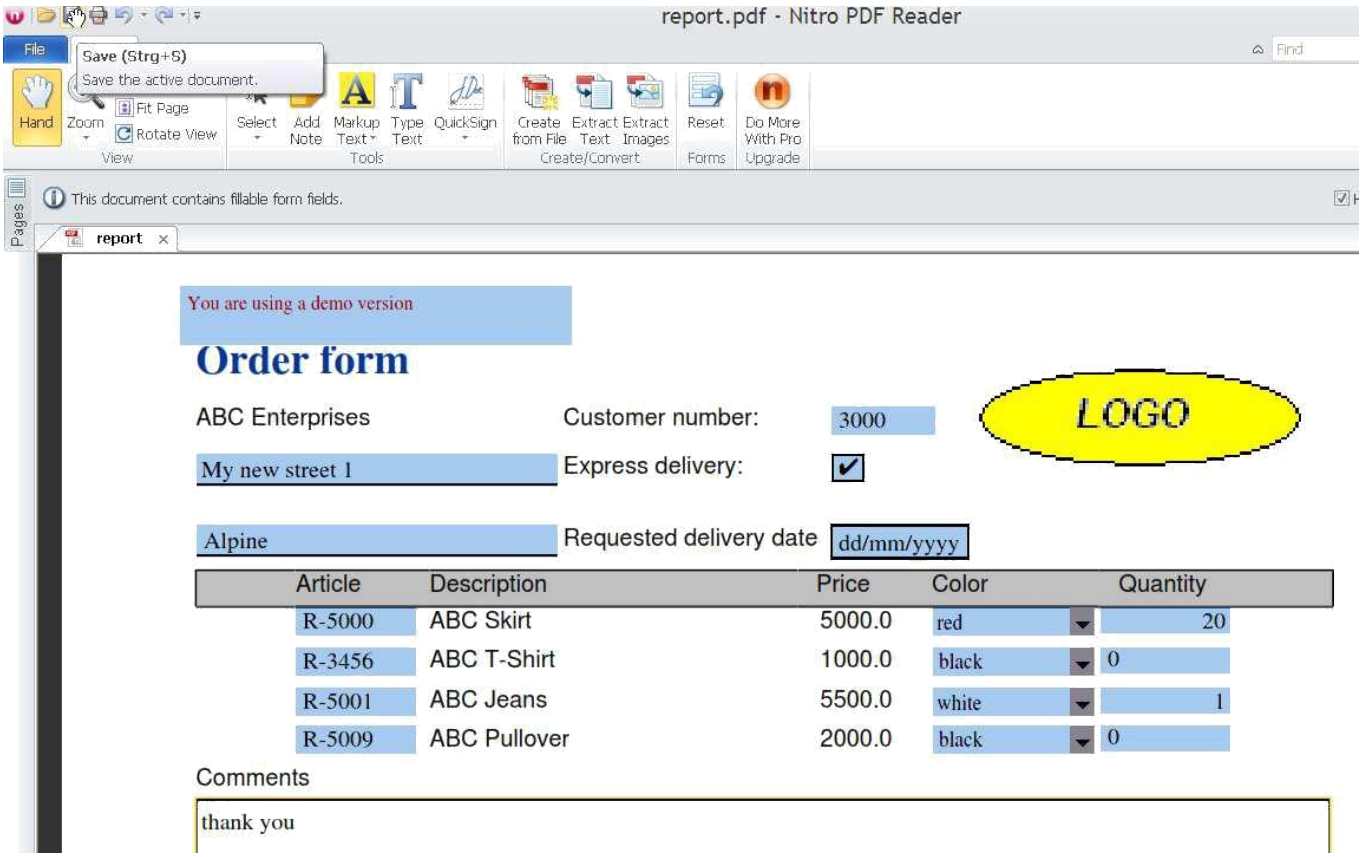
The file *examples\form\order_form.xrp* example shows how the input fields can be used. The example contains one input field of each type, furthermore it shows how fields can be repeated in the detail area.

The example contains no submit button since it assumes the user will fill the form with a PDF reader that can save forms (see the Filling forms section) and submit the saved PDF by email. This example is an order form which:

- Contains a read only field (customer number) which can be used as key for identifying the customer.
- The address and city fields are prefilled with the customer data but the customer can change the delivery address by modifying the fields.
- Contains a list of items the customer can order by just entering the quantity at selecting the color.
- There is a multiline comments field at the bottom.

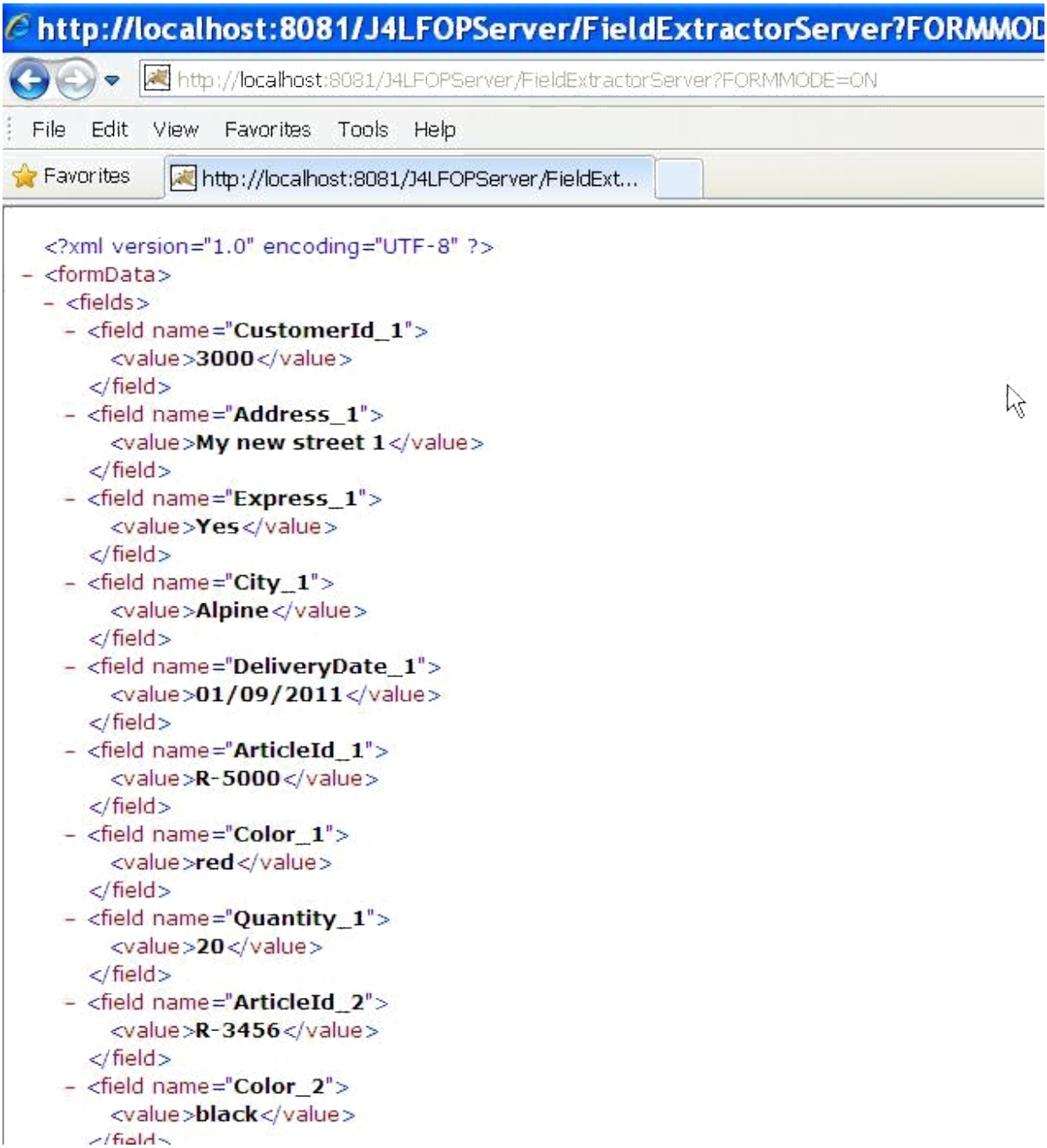


the screenshot shows how the PDF form has been filled and can be saved using Nitro PDF Reader:



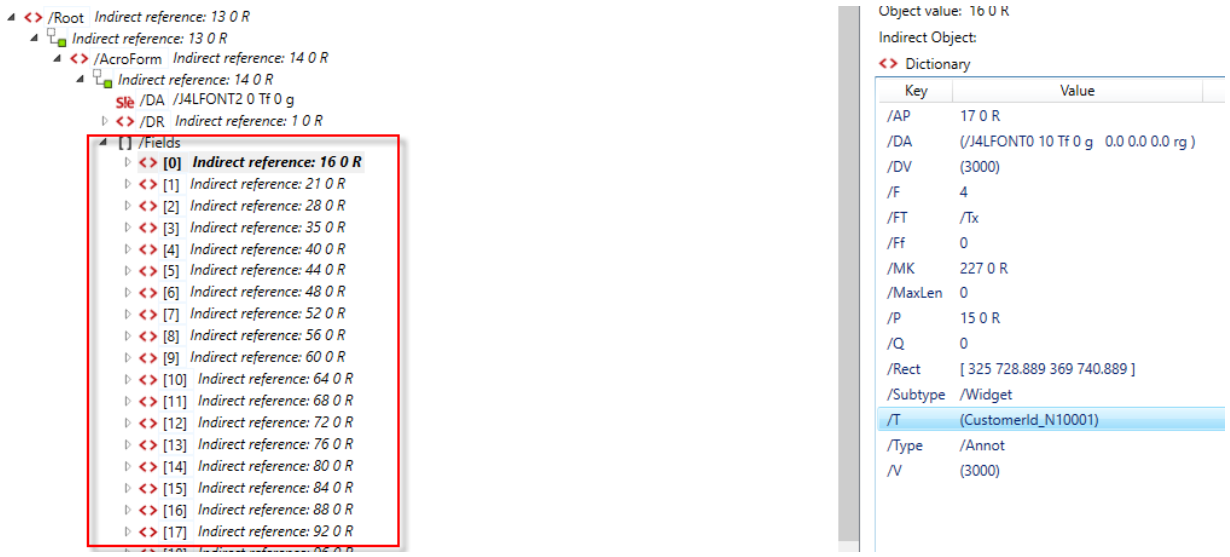
As last step the filled forms has been received by the selling company and the values of the PDF can be extracted (see section Form submission). We include a simple HTML page inside *J4LFOPServer.war* which calls our form extractor server:

once submitted the server returns the form content as XML, note field names have a suffix (_1 ,_2 ..) because fields in a detail section (like the items in a purchase order) can be repeated:

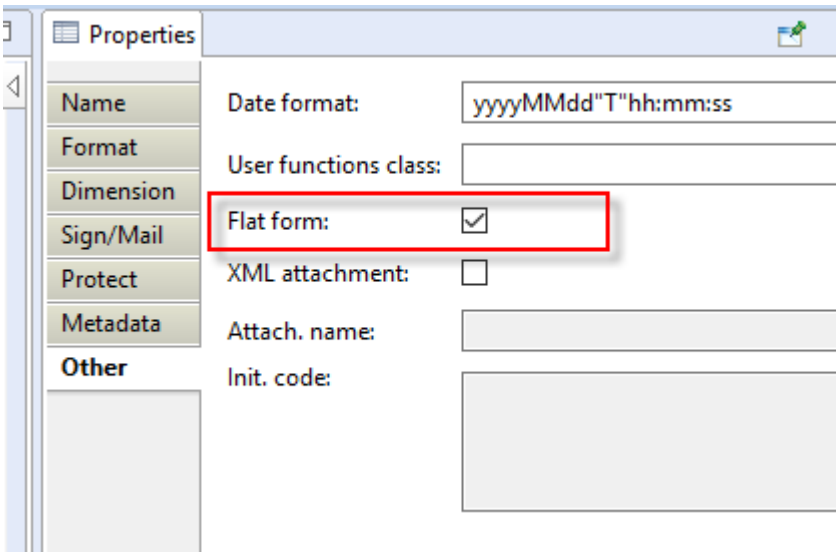

http://localhost:8081/J4LFOPServer/FieldExtractorServer?FORMMOD
http://localhost:8081/J4LFOPServer/FieldExtractorServer?FORMMODE=ON
File Edit View Favorites Tools Help
Favorites
http://localhost:8081/J4LFOPServer/FieldExt...
<?xml version="1.0" encoding="UTF-8" ?>
<formData>
<fields>
<field name="CustomerId_1">
<value>3000</value>
</field>
<field name="Address_1">
<value>My new street 1</value>
</field>
<field name="Express_1">
<value>Yes</value>
</field>
<field name="City_1">
<value>Alpine</value>
</field>
<field name="DeliveryDate_1">
<value>01/09/2011</value>
</field>
<field name="ArticleId_1">
<value>R-5000</value>
</field>
<field name="Color_1">
<value>red</value>
</field>
<field name="Quantity_1">
<value>20</value>
</field>
<field name="ArticleId_2">
<value>R-3456</value>
</field>
<field name="Color_2">
<value>black</value>

Form‘s internal structure

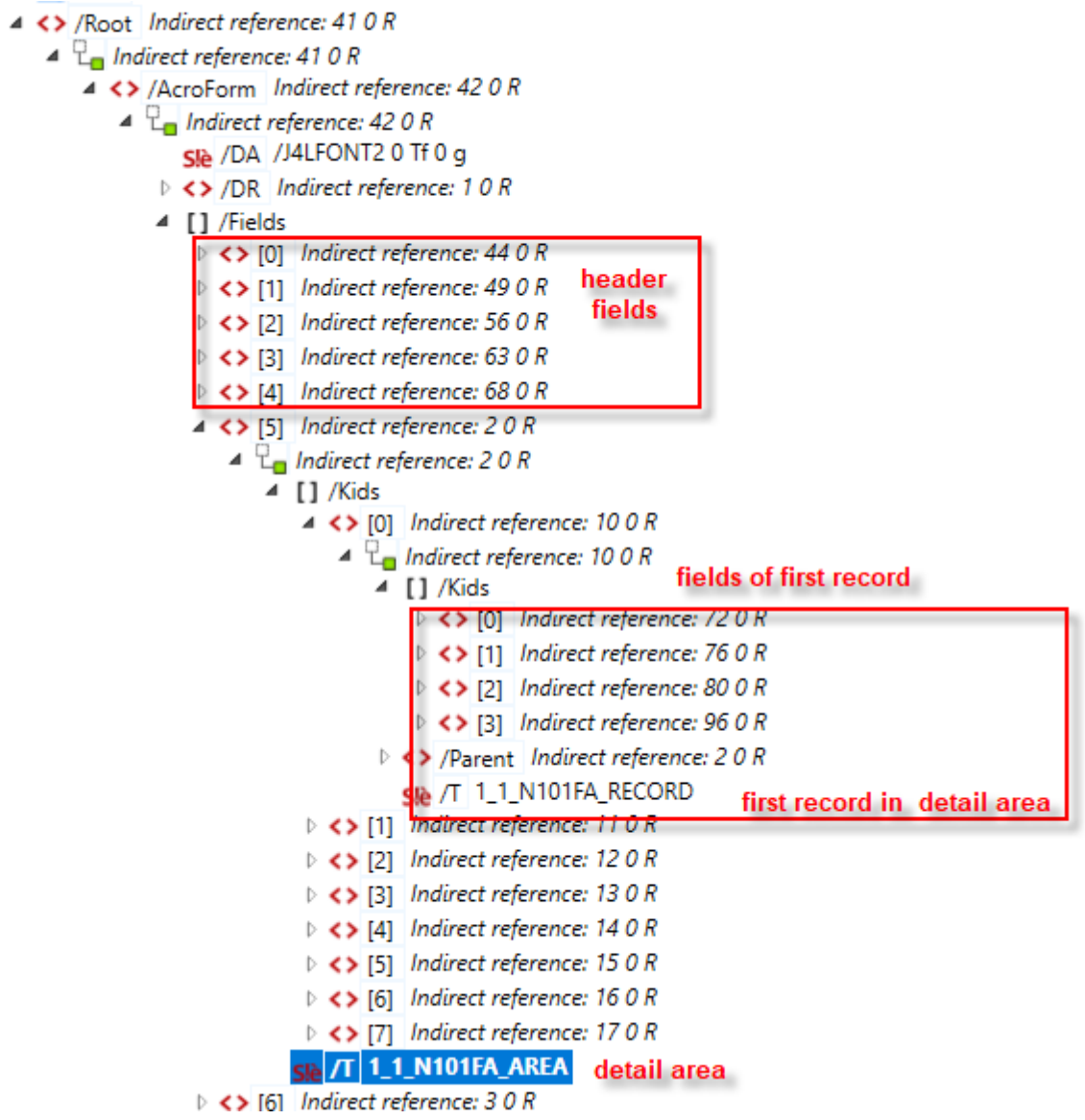
By default all input fields in a form will be stored as an list in the PDF form (Flat form). If you would use a PDF analizer tool, you would see the internal list of fields:



It is however possible to create a hierarchical structure for the field in the form. This can be achieved by removing the *Flat form* flag in the report‘s properties.



After removing the flag the created form fields in the PDF will have the same internal strtucture as the report in the designer:



that is it will have a tree structure:

- the form header fields will be in the first level
- the areas will be a node in the tree
- for each repetition of the area there will be a record node
- finally for each record there will be list of form fields

The naming of the form input fields (also called terminal nodes) and areas/record (non-terminal nodes, because the have children) is as follows:

- for form fields: <fieldname>_<GeneratedUniqueId>. For example: **CustomerID_N10001**
- for areas: <level>_<areacounter>_<GeneratedUniqueId>_AREA. For example the first detail area after the report header will be named **1_1_N10001_AREA.**
- for records (area repetitions) : <level>_<areacounter>_<GeneratedUniqueRecordId>_RECORD. For example the first detail area repetition after the report header will be called **1_1_N10001_RECORD ,** the second repetition of the same area could be **1_1_N102B5_RECORD**

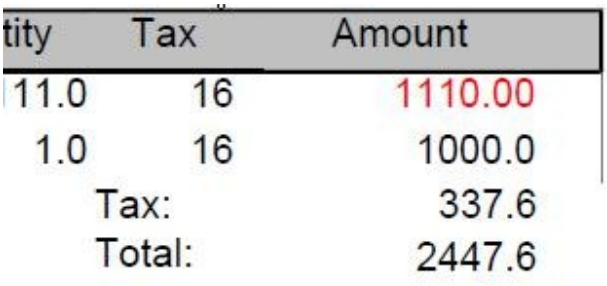
14. Flavours

You can use flavours to change the look of a field or combo box based on a condition. Each field has a default flavour , but you can add new ones and assign a condition to them. For each flavour you can define a font, border, color and all the set of properties of the field.

We deliver an example in the file which has a field with 2 flavours:

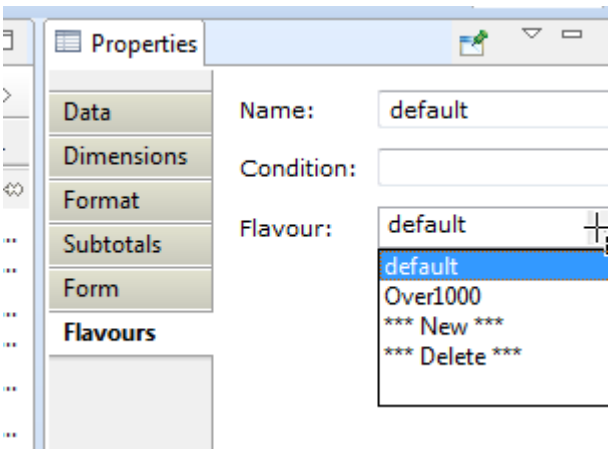
examples\xcbl_order\order_flavour.xre

This report creates an output like this:



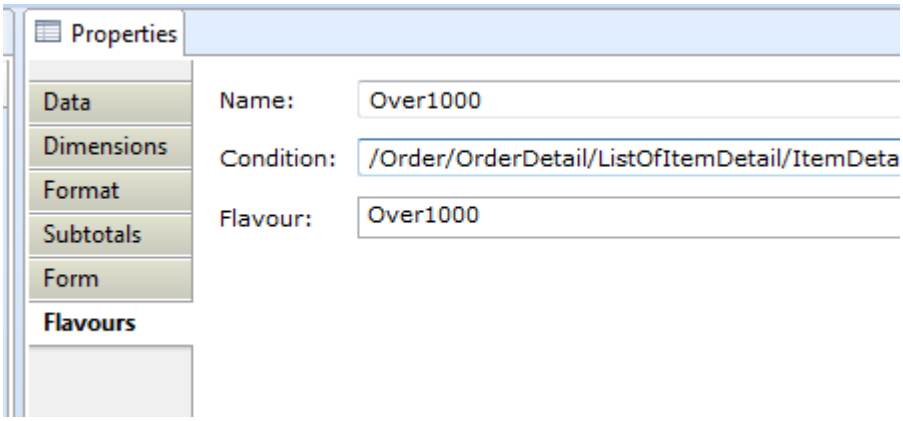
the amount field will get a red color if the value is over 1000. In order to achieve this, the process has been:

1. Select the amount field and from the main menu execute create a new flavour:



   Once the flavour has been added, you can select it from the "Flavour" combobox (see below). The initial name of the new flavour will be *New_<timestamp>*.
2. The flavours section of the field properties has 3 entries:
   - The combobox "Flavour" is used to select the current flavour you are working on.
   - The "Name" entry, is the name of the current flavour. Use the entry to change the name of the flavour.
   - The "Condition" field, is a XPath condition which defines when the flavour should be used. You should have one flavour without a condition, which will be

the default one. If you have more than 1 flavour without condition, they will be ignored, since only the default flavour may have an empty condition.



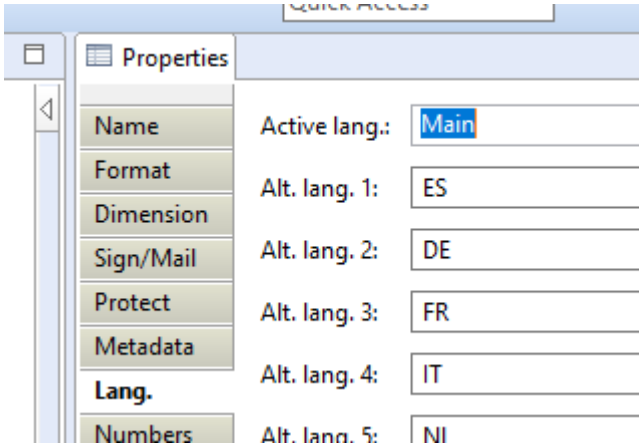
In this example we have defined a new flavour called "Over1000" and we have set the condition to be "/Order/OrderDetail/ListOfItemDetail/ItemDetail/PricingDetail/TotalValue/MonetaryValue > 1000 ". The font color of the flavour has been set to red.

15. Multilanguage support

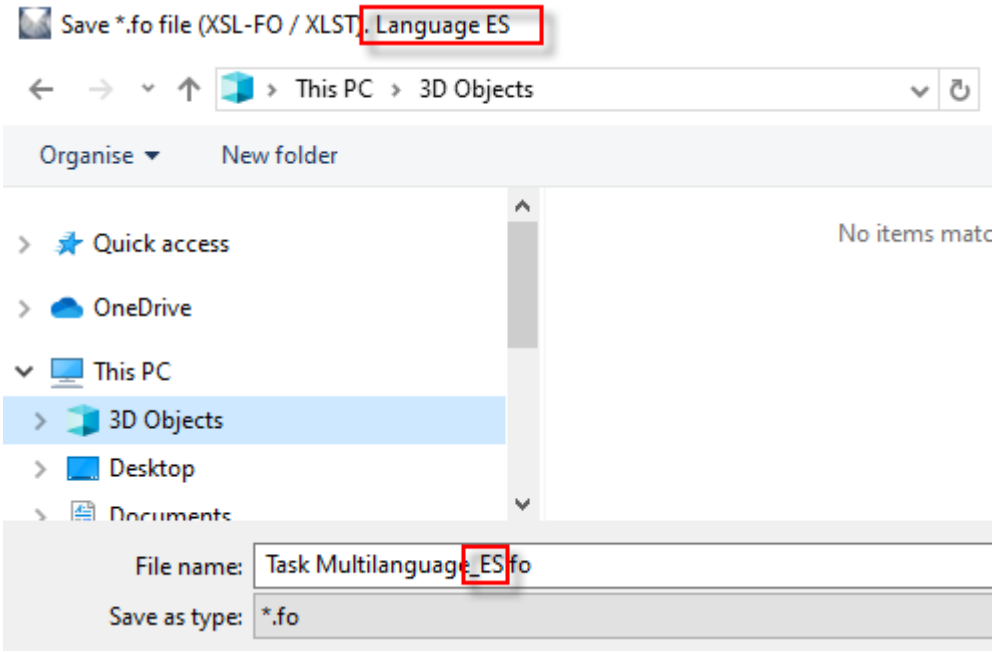
The same report template can be reused for up to 6 different languages (the default one and 5 alternative). Multilanguage support means the label fields can be entered in different languages. The report *tasks_lang.xre* is an example of this.

In the report‘s properties, tab „*Lang.“* up to 5 additional languages can be defined. Recomended is the use of the ISO-3166 country codes to identify the languages but this is not mandatory.

In the *Active lang.* Field you can select the current active language which wil be used when the PDF generation button is pressed or for the *Export FO* function. Set it to *Main* is you have only one language.

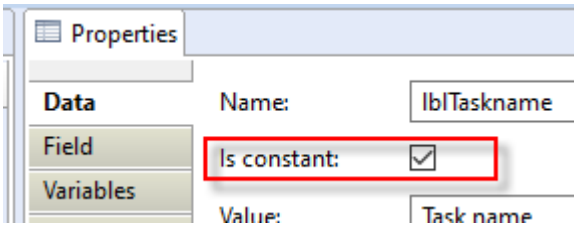


Note that when exporting to FO, the file selection dialog will display the current selected language and the proposed output file will contain the country code suffix.

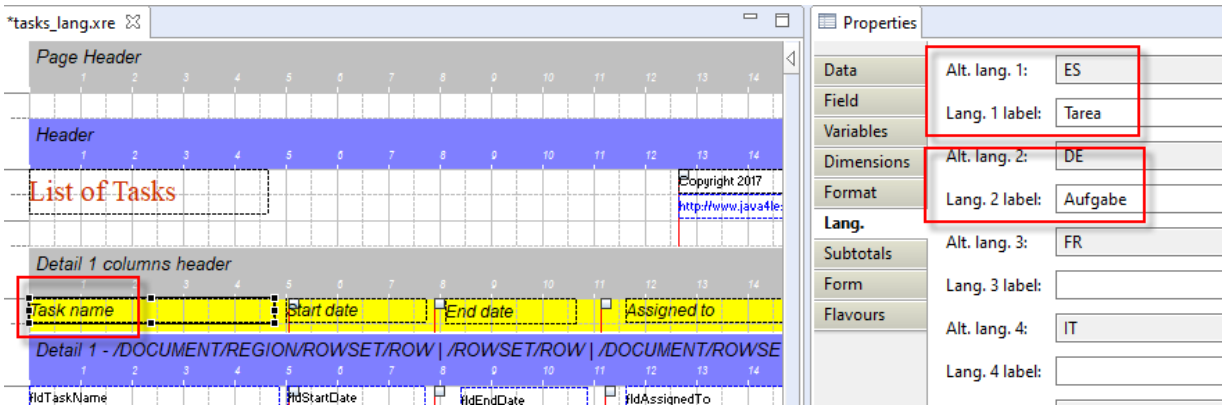


But before the alternative languages can be used, you need to assign a value to each label field. Label fields are those without an Xpath expression and therefore contain constant values.

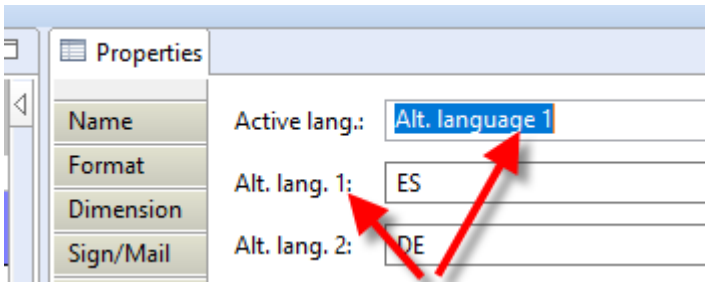
Labels fields normally have this flag set:



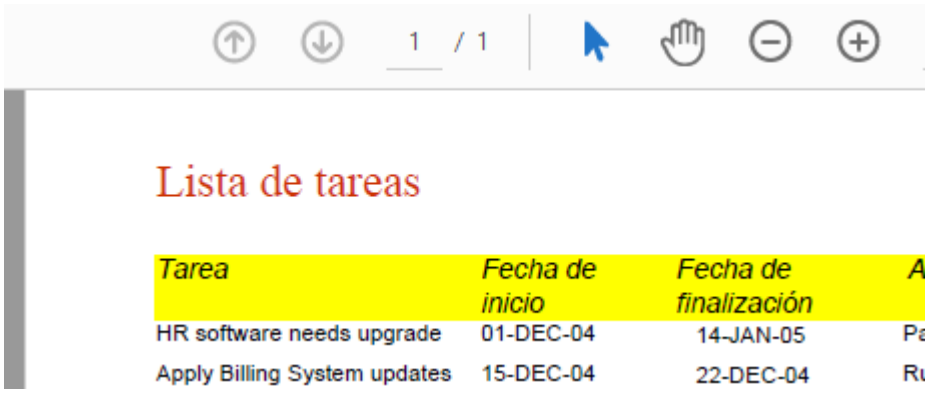
The example below depicts how the „Task name“ field, has been translated to spanish and german, being the default language english.



At this point if you set the active language property of the report to be „Alt. language 1“ which maps to the code ES for spanish, the PDF output will be in spanish

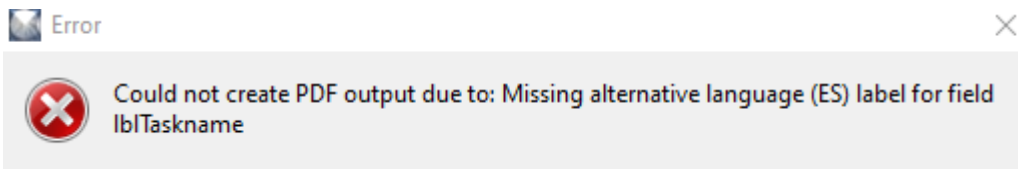


Output after generating the PDF



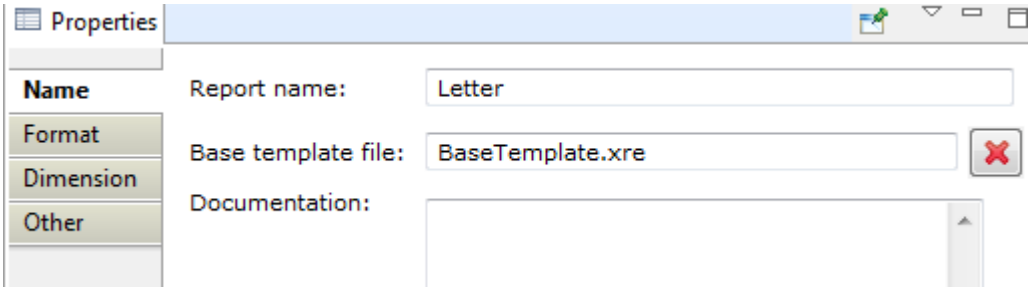
Note however:

1. The value „HR software needs upgrade“ is in english because this is an element coming from the input XML file, this is therefore not a label.
2. Each individual exported FO file can be used for one language only. Therefore you will need to generate one FO file for each language,
3. The following objects support multilanguage:
    - fields
    - buttons
    - links
4. If you use an alternative language you must define the language label for each individual label. If you miss one you will get an error as below:



16. The base template

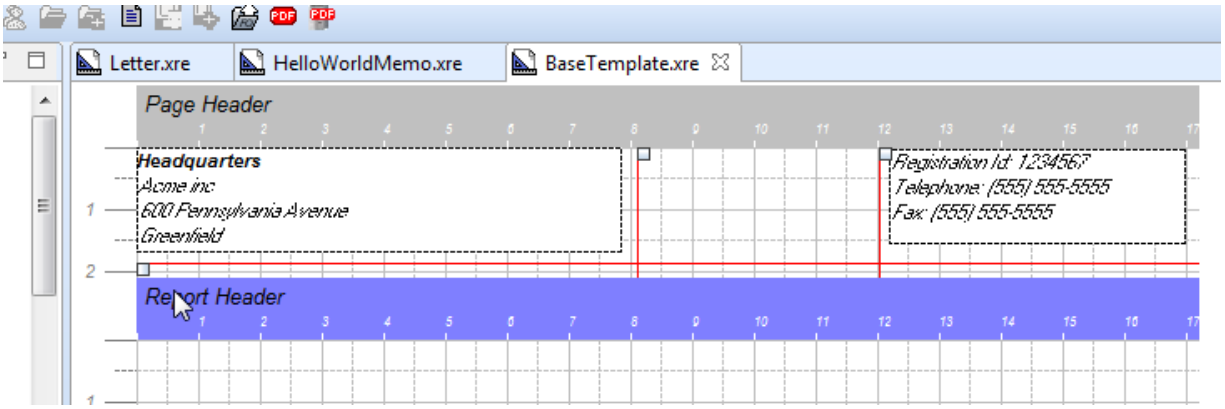
The *base template file* can be set in in the report properties. The base template is a report like any other and it has to be located in the current workspace since it will be used while generating the output xsl-fo.



The purpose of base template is to reuse report areas, this allows for example having a template file which defined the company data in the page header. This template can be shared by all other reports.
Only the following base template 3 areas will be used while generating the XSL-FO output:

- The **page header** of the base template will be used **only if the** current report does not contain any field in the page header.
- The same apply to the **page footer**.
- The **background area** of the base template will always be used and **merged** with the background area of the current report.

The example *BaseTemplate.xre* shows how to use it to create a common header for all company reports:



The example contains the company data in the page header and the page number in the page footer. This template has been used in the *Letter.xre* report. If you run the report you will see the page header being added to the PDF from the base template:

Usually base templates will contain only constant data. Even then, you can still test the report by using any XML file. The designer always requires a test XML file to be select in order to run the report, in this case you can select any XML file, then you can view the output of the base template.

17. Long texts and HTML fields

You can create mail merge letters with the FO Designer as you do in Microsoft Word. For this you need:

- An XML file with, for example, addresses and names.
- An a XML field which contains the letter text.

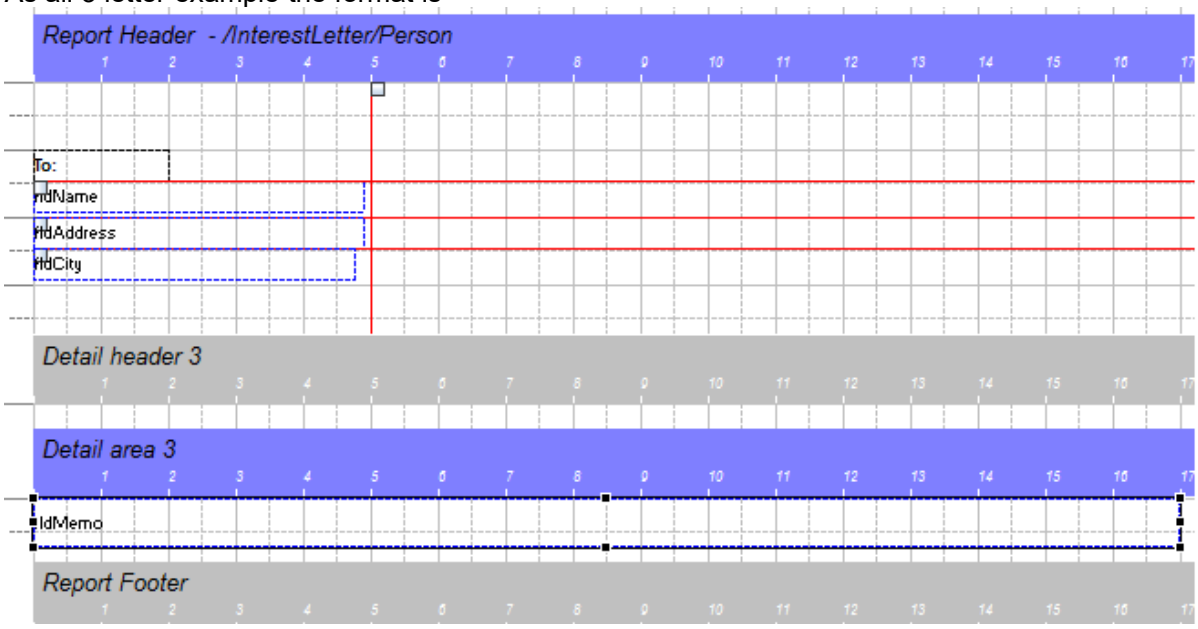
At runtime the designer will add the names and addresses by replacing placeholders in the letter text. There are 5 placeholders which are identified by using the text $1,$2,$3,$4 and $5.

We deliver 3 examples:

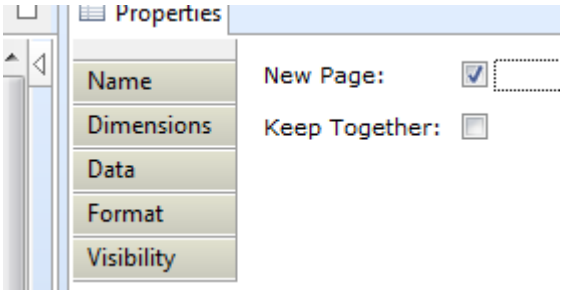
- *Letter.xre:* uses plain text for the letter contents
- *LetterHTML.xre and LetterHTML2.xre:* use HTML code within the XML file which provides rich text formatting features. Note however this will work at runtime only if you use the J4L FOP Printer server (or the WAR file in your own server).

The Letter example

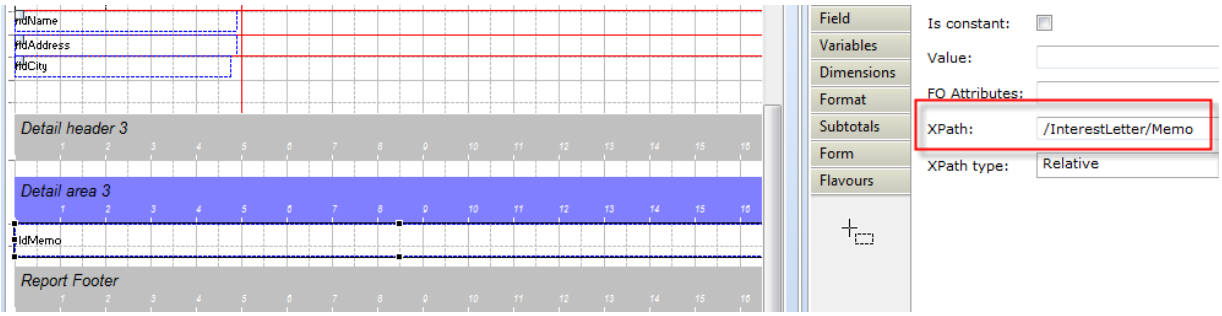
As all 3 letter example the format is



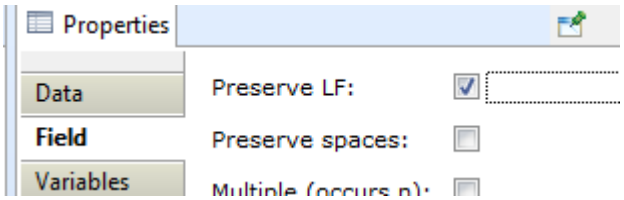
A report header with the addressee data, note the new page flag to make sure each letter will be printer in as separate page:



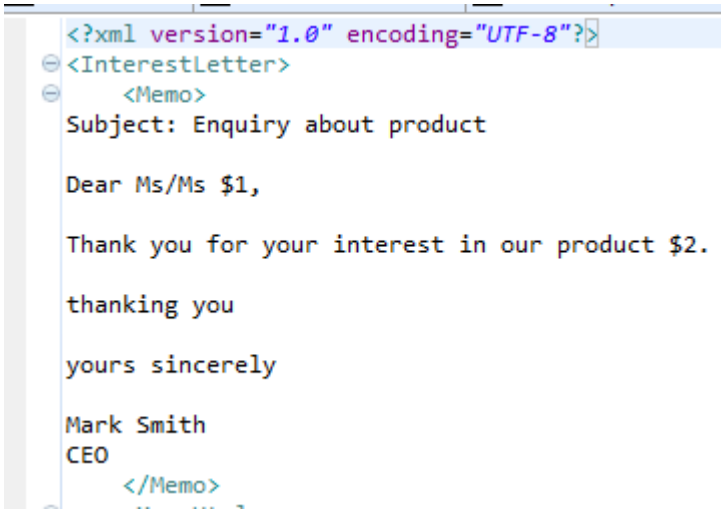
And a detail area which contain one very large field which has been associated with the XML element */InteresetLetter/Memo*



Note here the Preserve LF setting:

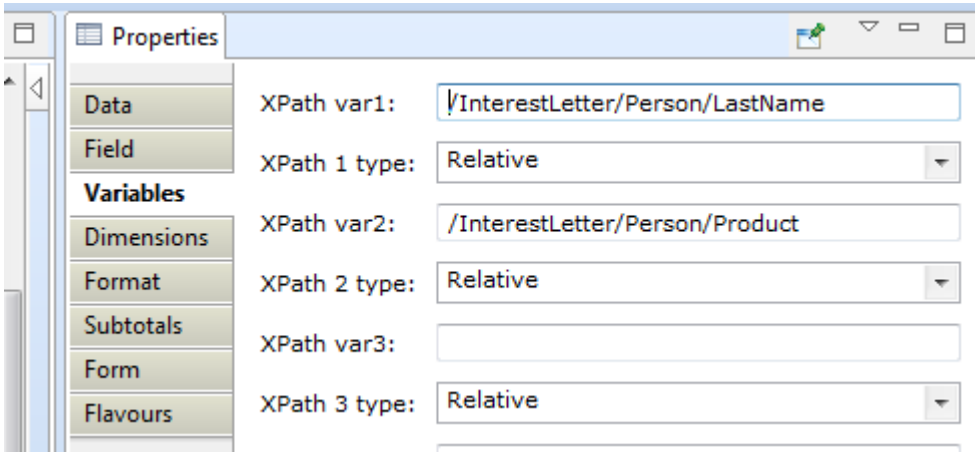


This setting is required, since the line feeds within the XML element need to be preserved. See XML element below:



```
<?xml version="1.0" encoding="UTF-8"?>
<InterestLetter>
    <Memo>
Subject: Enquiry about product

Dear Ms/Ms $1,

Thank you for your interest in our product $2.

thanking you

yours sincerely

Mark Smith
CEO
    </Memo>
```

Note the placeholders $1 and $2 in the XML file. At runtime these will be replaced by the content of the variables defined below:



This will be the output in the first page (note the placeholder replacements)

Subject: Enquiry about product

Dear Ms/Ms Smith,

Thank you for your interest in our product Hair Dryer Novo 300. I have
contains more detailed information about our products and services. If
product or its features, please contact our customer service team at 01

thanking you

The HTML fields

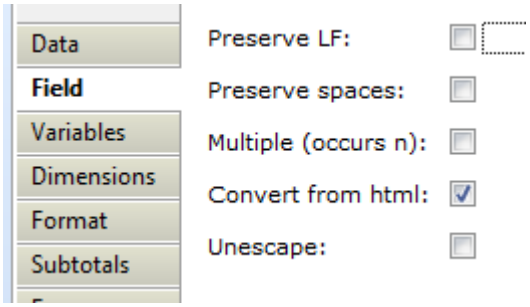
Warning: *the HTML conversion is only available if you use the J4L FOP printer server.*

The two Letter HTML example work in the same was as the plain text example with 2 minor differences, the content of the XML file and the flags in the designer. A use case for this is the Oracle APEX Rich Text editor. A business user can utilize the editor to enter text, this will save the HTML data in a Oracle table. Later on a XSL-FO can be created based on the stored HTML field.

The *LetterHTML* example takes an input field an CDATA element which contain HTML code.

```
<MemoHtml>
<![CDATA[<html>
    <body>
   <h2><span style="font-size:14px
   <p>Dear Ms/Mr<strong> $1</stron
   <br/>
   <br/>
   <p>Thank you for your interest
   <br/>
   <p>thanking you</p>
   <br/>
   <p>yours sincerely</p>
```

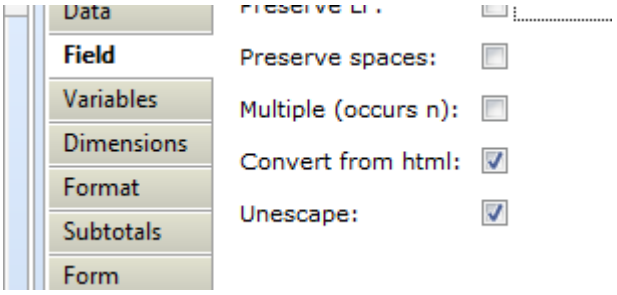
In this case you only need to set this flag:



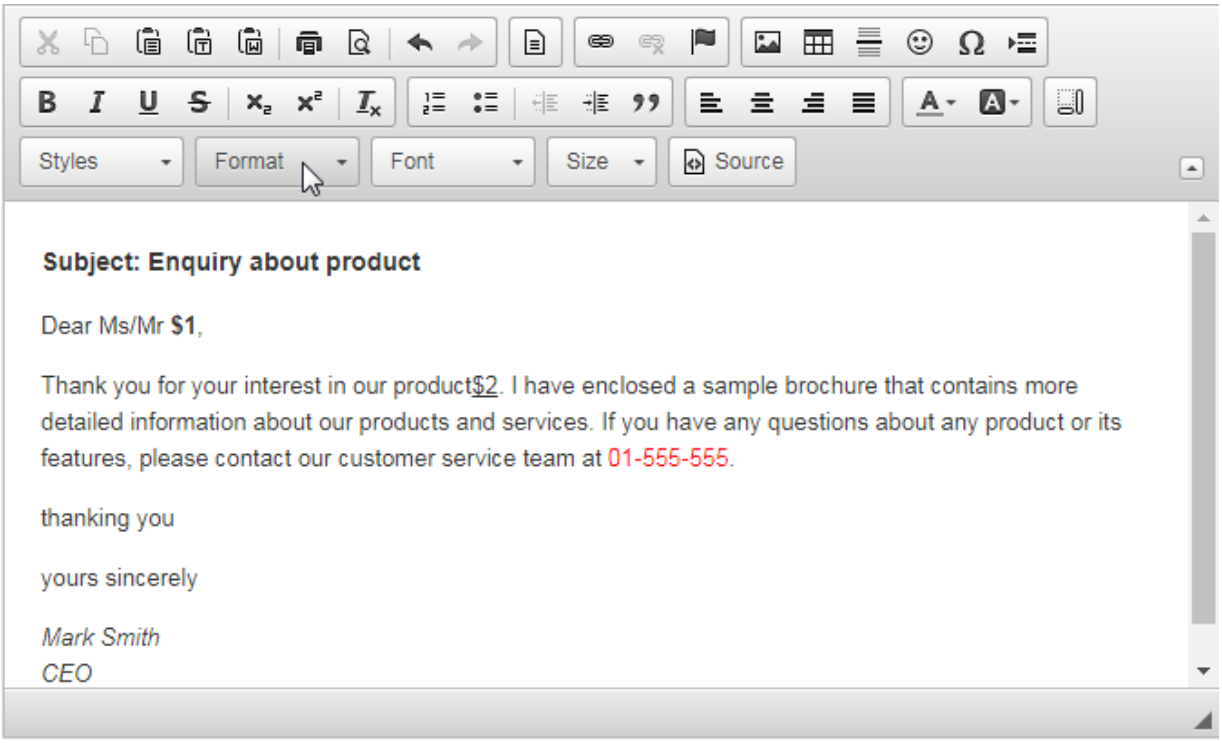
The *LetterHTML2.xre* example has a different input element, there is no CDATA type therefore the HTML data has been *escaped* otherwise the XML file would be invalid

```
    <MemoHtml2>
    &lt;h2&gt;&lt;span style="font-size:14

&lt;p&gt;Dear Ms/Mr&lt;strong&gt;&amp;nbsp

&lt;p&gt;Thank you&amp;nbsp;for your inter

&lt;p&gt;thanking you&lt;/p&gt;

&lt;p&gt;yours sincerely&lt;/p&gt;

&lt;p&gt;&lt;em&gt;Mark Smith&lt;/em&gt;&l
&lt;em&gt;CEO&lt;/em&gt;&lt;/p&gt;
    </MemoHtml2>
```

if this is the case you need to select these 2 flags



This is for example what happens if you use Oracle APEX Rich Text Editor and generate a the XML file for the field.



Please note only the following HTML tags are supported in the HTML conversion:

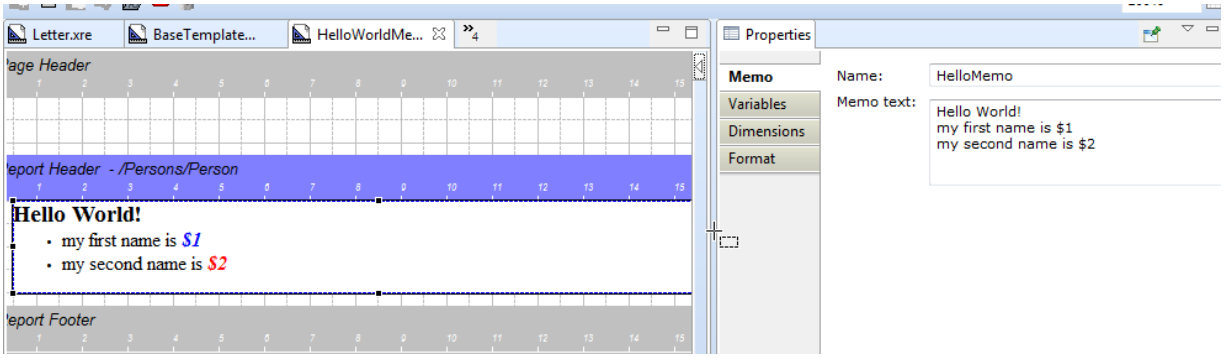
- <h1>, <h2>, <h3>, <h4> , <h5> headers
- <p> paragraph
-
- or <b> bold text
- or <i> italic text
- <u> underlined text
- <big> or <small> text size
-
 line feed
- /<il> bullets
- color , font-size, font-style, font-family ,text-align attributes (for <p> and)

18. The Memo field explained

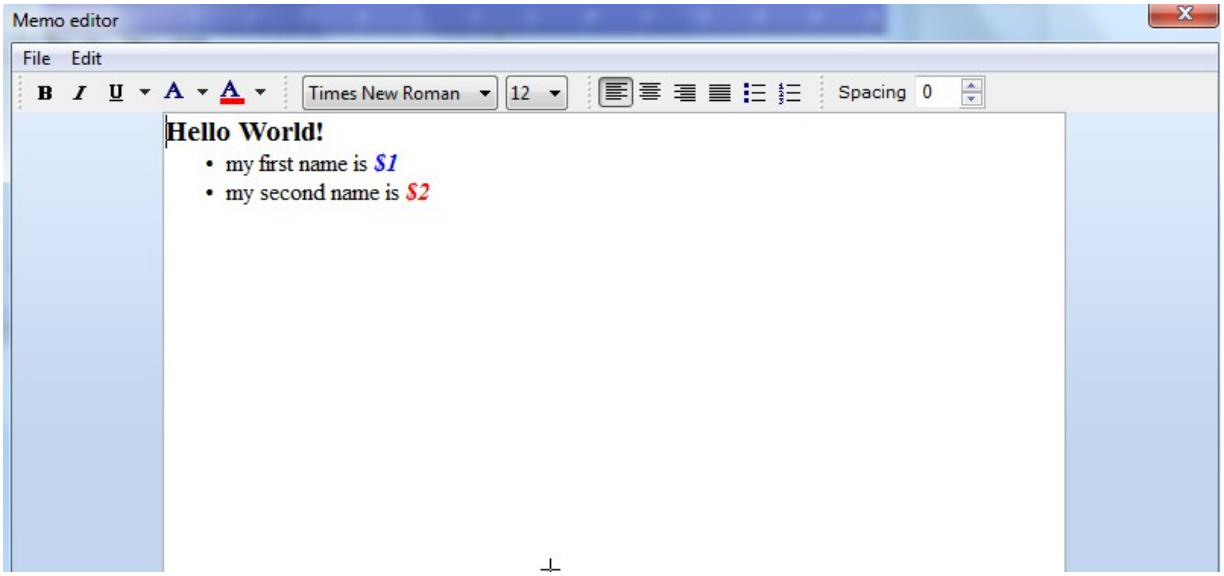
The memo field can be used to create constant text in the report and at the same time adding text style in a very easy way using the memo text editor.

Even if most of the text will be constant, it is possible to add variable data using the placeholders ($1 to $5) as explained in the previous section. Note the Variables tab where you define the Xpath for the place holders.

The *HelloWorldMemo.xre* report is a very simple example that shows how to use the memo field. Open the memo field editor by clicking on *Memo text* property



The editor will open up and the input area will be **as wide as the field** in the report.

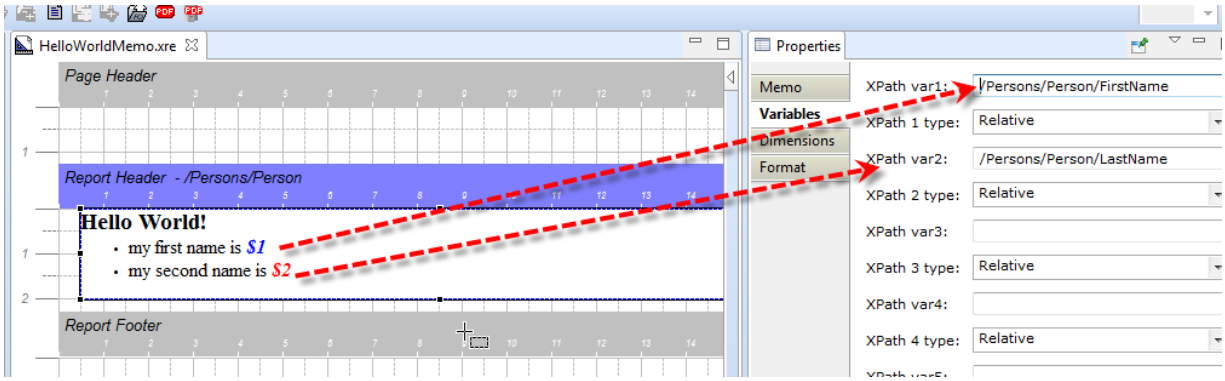


The editor is very intuitive however there are a couple of attention points:

1. Use the *Edit - Font* menu item to define the default working font. The default is Times New Roman 12.
2. If you need to change the font or style of the text, you select a range of text and then use the tool bar to set color, style and font size
3. Use *File – Save and exit* to save your changes

Note while being very powerful, memo fields generate a large amount of output code, therefore it is not advisable to used them as a replacement for constant text labels. Use them only if you have a long text and you need to change the style of the words within the text.

Finally, in the variables tab you can see the assignment of the placeholders to the XML elements



The output is then

Hello World!

- my first name is *Peter*
- my second name is *Smith*

Hello World!

- my first name is *David*
- my second name is *Brown*

Hello World!

- my first name is *James*
- my second name is *Davis*

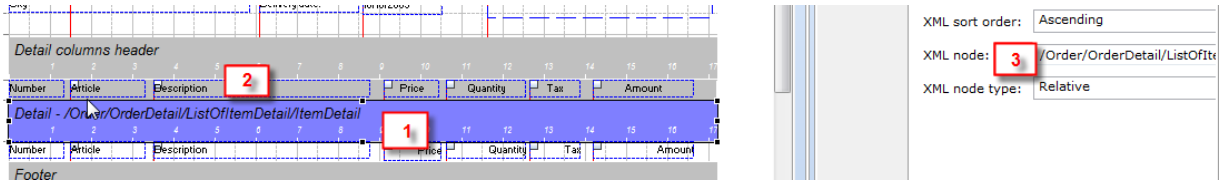
19. Advanced functionalities

This section will describe how the designer generates the XSL-FO code and how you can add your own code. Note this section requires some basic knowledge of XSLT and XPATH.

Understanding the generated code for areas

Each detail area (see 1 in the screenshot) in the designer will be printed as a table:

- The table has a header (see 2) which will contain the area's header content.
- And a body which contains the detail area content.
- The body can contain many rows which are the repetitions of the detail area. For these many rows to happen the designer creates a XSLT loop using the *for-each* expression based on the *XML Node* that has been selected for the area (see 3)



In the *order.xre* example we can see the described structure for the order lines:

- There is a table being created if any node */Order/OrderDetail/ListOfItemDetail/ItemDetail* exists. That is what you see right below the blue comment line (xsl:if expression).
- There is a table header which contains the „*Detail Columns Header*" fields.

```
<xsl:comment> START Area Detail </xsl:comment>
<xsl:if test="OrderDetail/ListOfItemDetail/ItemDetail" >
<xsl:comment> ID Detail 554b3b90-5b58-445d-9fd9-c68deb4fdba4 <
<fo:table table-layout="fixed" width="16.99cm" border-color="#00
 <fo:table-column column-width="1.49cm"/>
 <fo:table-column column-width="2.0cm"/>
 <fo:table-column column-width="5.51cm"/>
 <fo:table-column column-width="1.49cm"/>
 <fo:table-column column-width="2.0cm"/>
 <fo:table-column column-width="1.51cm"/>
 <fo:table-column column-width="2.99cm"/>
 <xsl:comment> ID Detail columns header 097eda75-abd4-4d9d-a6a3
 <fo:table-header>
  <fo:table-row border-color="#000000" border-style="solid" border-wi
   <fo:table-cell padding="1mm" number-columns-spanned="1">
   <xsl:comment> ID RVField1 ea2a25fd-99b2-4b5e-bdfe-9fc0c5524
```

- There is a table body
- And there is a loop. This is the expression

<xsl:for-each select="***OrderDetail/ListOfItemDetail/ItemDetail***">

```
</fo:table-header>
<fo:table-body>
<xsl:for-each select="OrderDetail/ListOfItemDetail/ItemDetail">
  <fo:table-row   height="0.49cm"   >
    <fo:table-cell   number-columns-spanned="1">
   <xsl:comment> ID RVField1 e0b125e9-6376-498d-9145-06a3d9f552
   <fo:block  font-size="10pt"  font-family="SansSerif"  color="#000000"  t
   </fo:block>

    </fo:table-cell>
    <fo:table-cell   number-columns-spanned="1">
   <xsl:comment> ID RVField2 76487d49-18ef-4c62-8d5b-443418c0ce
   <fo:block  font-size="10pt"  font-family="SansSerif"  color="#000000"  t
```

After the loop there is the definition of the tables rows *<fo:table-row* , therefore a new table row will be created of each repetition of the loop, that is for each *ItemDetail* in the input XML. In each row the fields of the area will be printed.

- Further down in the code there is the end of the loop and the table to finish the area printing

```
  </fo:table-row>
 </xsl:for-each>
 </fo:table-body>
</fo:table>

</xsl:if>
```

The result is:

- One table header (the grey area below)
- One table body with 2 rows (for 2 items in the order, since we used an input XML file with 2 items only)

Number	Article	Description
00010	R-5000	ABC red 250 gr
00011	R-3456	ABC magic 500 gr

Note the code generated for the report header is similar but slightly different in the locations.

Introduction to Variables and Keys

This section will introduce the use of XSLT Variables and Key in the designer.

Variables can be defined like this:

<xsl:variable name="Variablename" select="value"/>

Some examples are:

- <xsl:variable name="color" select="'blue'"/> : this creates a variable named *color* and sets the value to *'blue'*
- <xsl:variable name="color" select="/MYDOCUMENT/COLOR"/> : this creates a variable named *color* and sets the content of the XML node /MYDOCUMENT/COLOR.
- <xsl:variable name="color" /> : this creates the variable and the value will be an empty string.
-

The variable can later on be read using the syntax $variable name. For example:

- You can use $color as value in the Xpath of a field name
- Or you can type in this expression in a free code field:

  <xsl:value-of select="$color"/>

Keys are similar to database lookups. First you need to defined a key (this has to be done in the initialization part of the report). The syntax is:

<xsl:key name="keyname" match="node" use="indexexpression"/>

- Name: is the name of the key.
- Match: is the node that the key should return. It is like a table name in a database lookup. For example, if we have a structure like */Persons/Person/FirstName* and */Persons/Person/LastName* and we want to create a lookup to find all persons named „Peter", we use */Persons/Person* as Node or Table.
- Use: this is the index field (like an index field in a database table). Since we are going to search by name we will use the element *firstname*.

  The key will therefore be:

  <xsl:key name="mySearch" match="*/Persons/Person*" use="*FirstName*"/>

The Lookup can be the executed using the key() function:

key(keyname,expression)

The expression can be a constant like *'Peter'* , a variable or a node (XPath)

For this input data:

```
<Persons>
	<Person>
		<FirstName>Peter</FirstName>
		<LastName>Smith</LastName>
	</Person>
	<Person>
		<FirstName>David</FirstName>
		<LastName>Brown</LastName>
	</Person>
	<Person>
		<FirstName>James</FirstName>
		<LastName>Davis</LastName>
	</Person>
	<Person>
		<FirstName>Peter</FirstName>
		<LastName>Jones</LastName>
	</Person>
</Persons>
```

The key function can be used in the report like this:

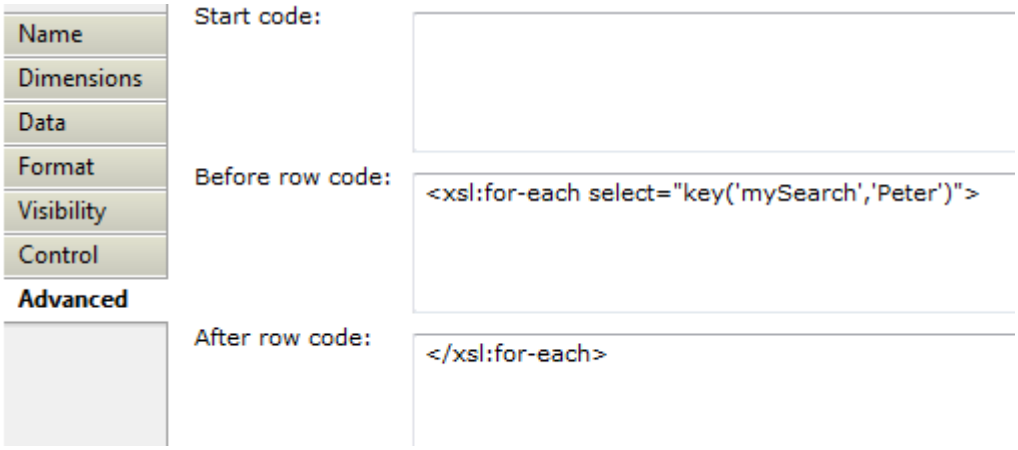
- Execute the lookup and store the output in a variable named *result.* You can do this in the „*Start code“* in the *Advanced* tab of the area (see next section)

  <xsl:variable name="result" select="key('mySearch','Peter')[1]"/>

  pay attention to the [1] which will store only the first returned node (in case there are several persons with name Peter)
- You can use the variable of the previous step in a field of your area, for example setting the Xpath value to *$result/LastName.*

- *<xsl:for-each select="key('mySearch','Peter')">* : This would return nodes 1 and 4. This creates a loop which need to be closed with *</xsl:for-each>.*

  Tip! You can leave the *XML Node* of the area empty and instead you can use the *Advanced* fields (see next section) to create a custom loop for the area as below:



We deliver an example called *PersonKey.xre* where you can see the implementation of this section. **Please note there are other ways of achieving the same result without using variables or keys. The only purpose of this example is to show these features in action.**

The first 'Peter' in the list is: Smith

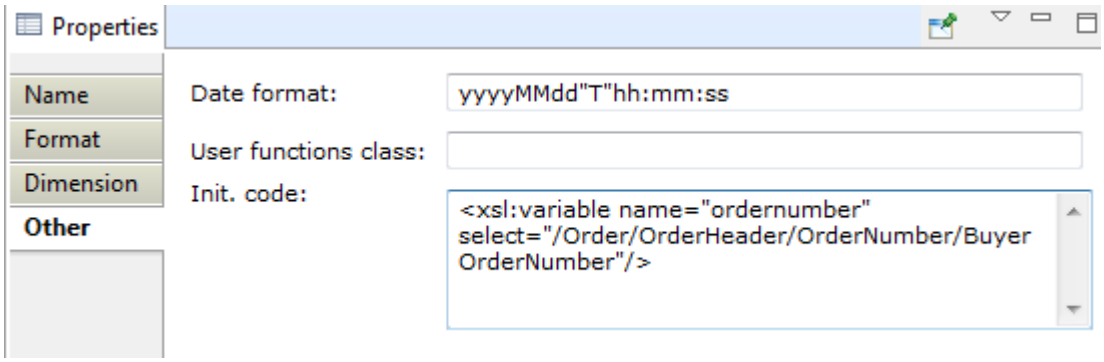
All the 'Peter' in the list are:

Smith	*Peter*
Jones	*Peter*
Johnson	*Peter*

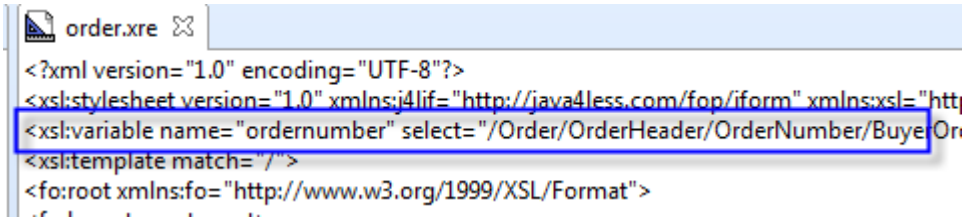
Adding your own code

There are several way of adding XSL-FO code to your report:

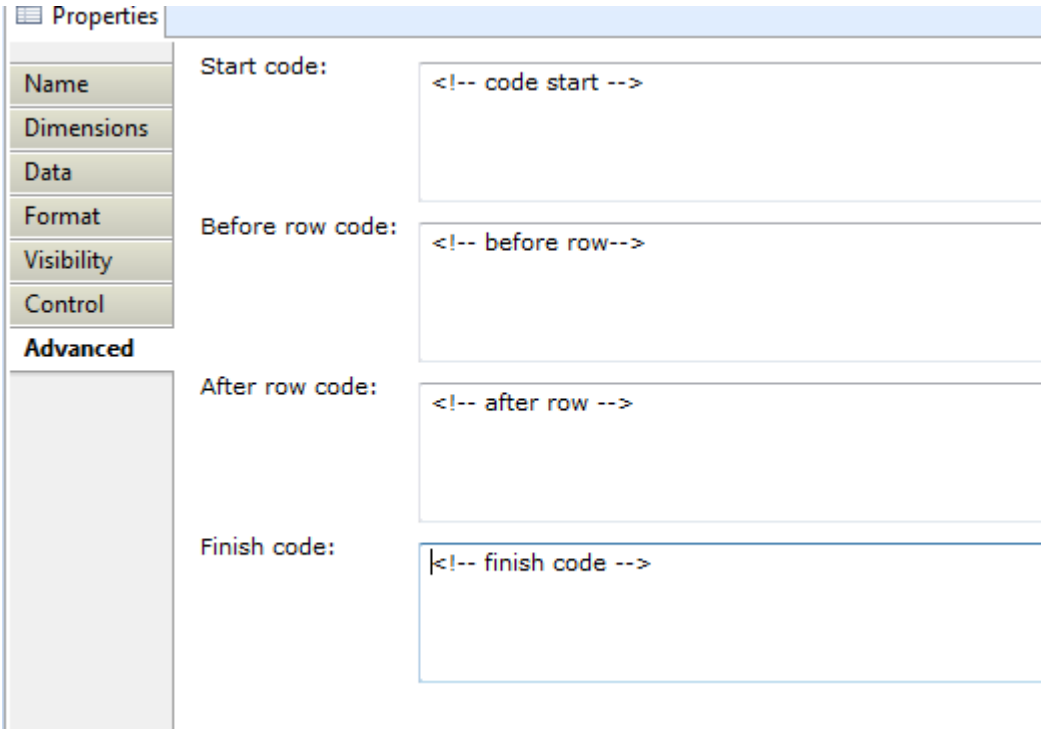
- The first one in the Free Code field. This allows the insertion of code in cells, rows and tables. We will not explain it in this section since it has already been explained in this guide.
- The report inititialization code can be set in the report properties, *Other* tab. This field can be used to define keys or report variables. For example we create here a variable which will contain the order number. The variable can be used later in the report using $ordernumber.



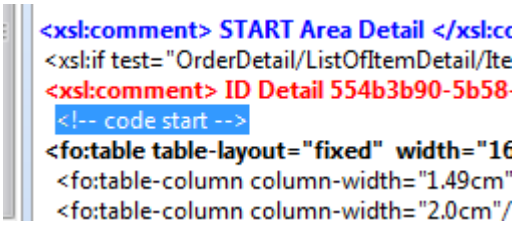
the code will then be placed at the very top of the generated code:



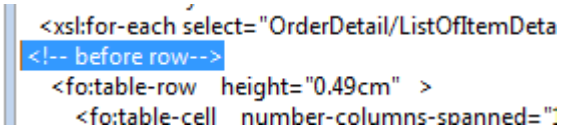
- Each detail area and the report header contain an A*dvanced* tab where you can place code in different locations. In the screenshot below we have added XML comments to highlight the output code location.



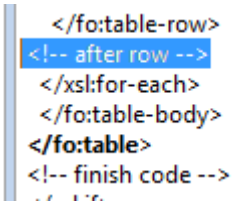
- Start Code: will be place before the table creation for the area.



- Before row code: the code will be placed after the table loop and before the row generation.



- After row code: after each row repetitions



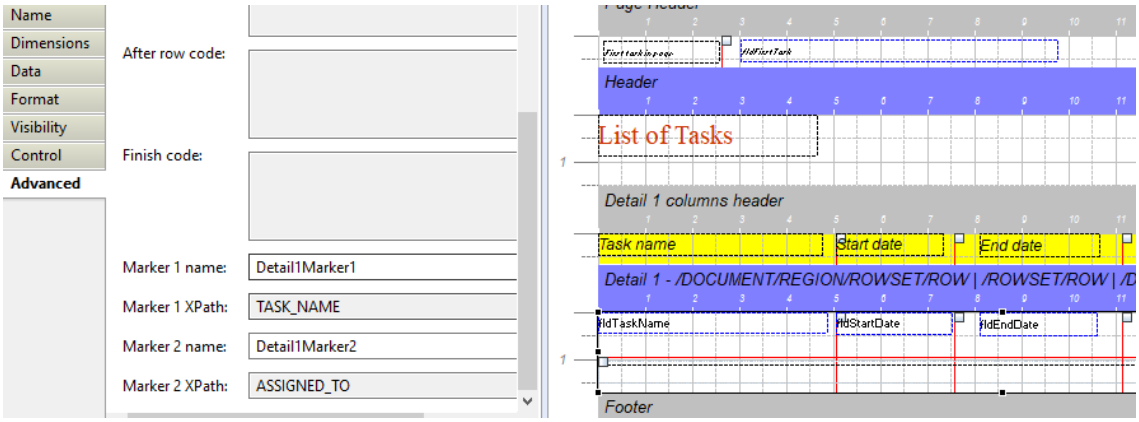
- Finish code: after the loop.

Markers (variable data in page header and footer)

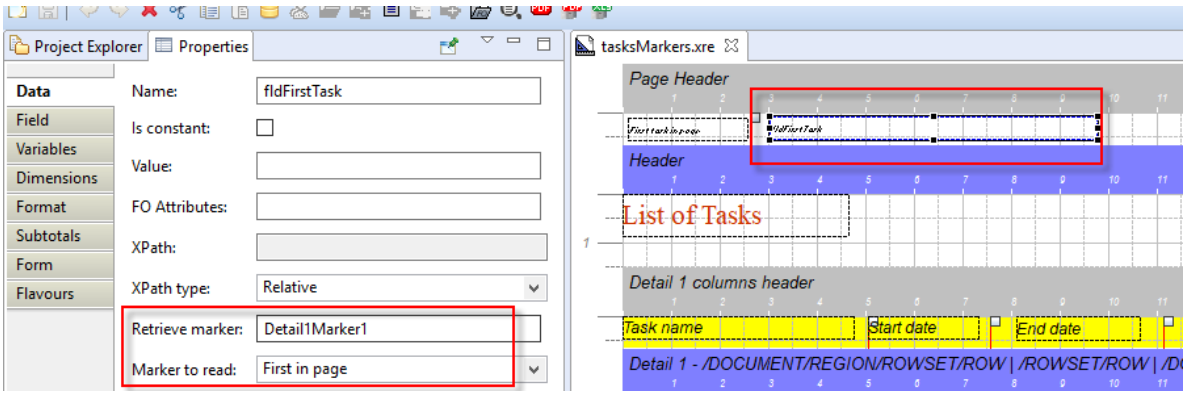
Markers are used in XSL:FO to keep track of the values in the pages as they are being rendered. The main purpose of the markers is to be used in Page Headers and Page Footer. By using markers you can have variable data in the page header and footers which otherwise can only contain constant data.

The delivered example *tasksMarkers.xre* shows how to use the markers.

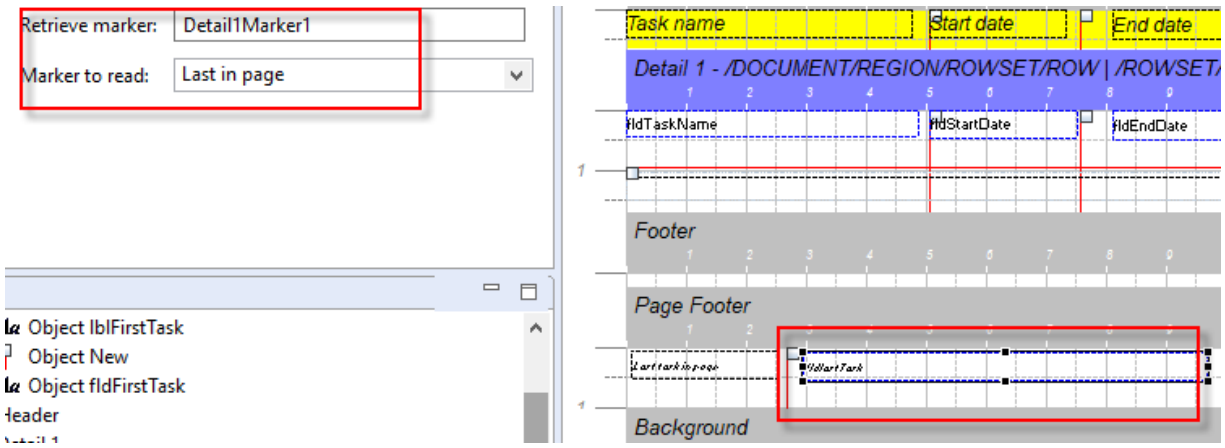
- First the markers much be defined for an area. Two markers can be defined by default. In this case the engine will collect information about the TASK_NAME and ASSIGNED_TO values in the page.



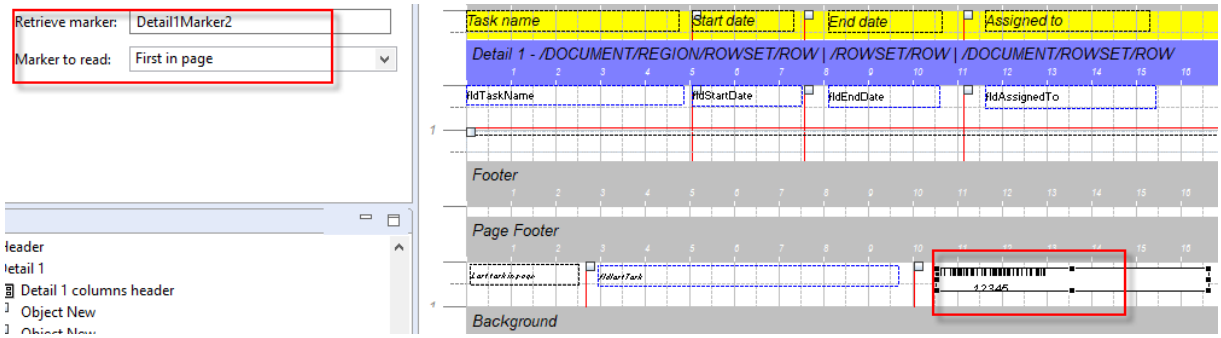
- As second step the marker can be used in the pages header and footers
  - The page header field below will display the **first** marker value in the page. In this case the first TASK_NAME.



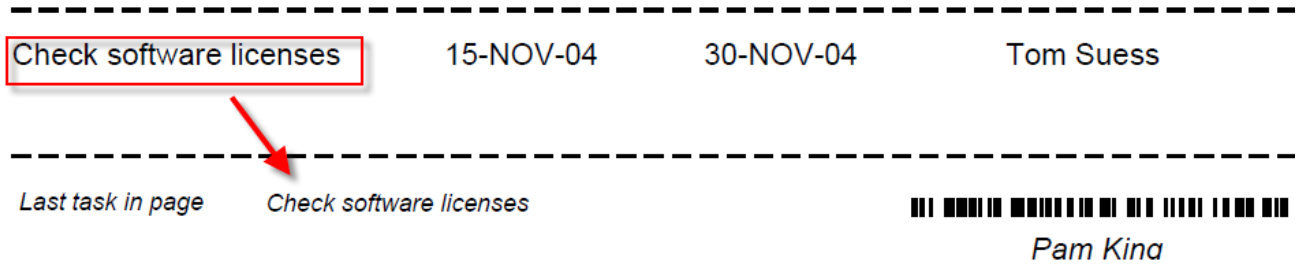
- The page footer field below will display the **last** marker value in the page. In this case the last TASK_NAME



- The page footer barcode below will display the **first** marker value in the page. In this case the first ASSIGNED_TO value (note it uses Marker2).



The following screenshot shows the output of one of the page footer fields, the last value in the page has been added to the page footer, while the barcode shows the first name in the page



Note: if you need to define more than 2 markers the code below can be used in the "After row code" field

for report headers use:

```
<fo:block>
<fo:marker marker-class-name="markerName">
<xsl:value-of select="yourXPATH"/>
</fo:marker>
</fo:block>
```

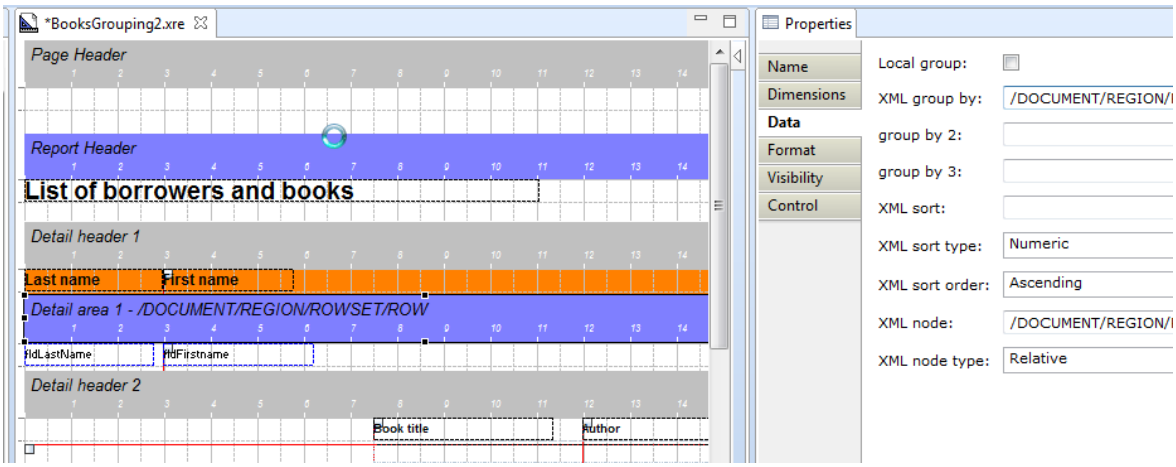
for details areas use:

```
<fo:table-row max-height="0mm" height="0mm" border-collapse="separate" border-style="none">
<fo:table-cell border-style="none">
<fo:block>
<fo:marker marker-class-name="markerName">
<xsl:value-of select="yourXPATH"/>
</fo:marker>
</fo:block>
</fo:table-cell>
</fo:table-row>
```

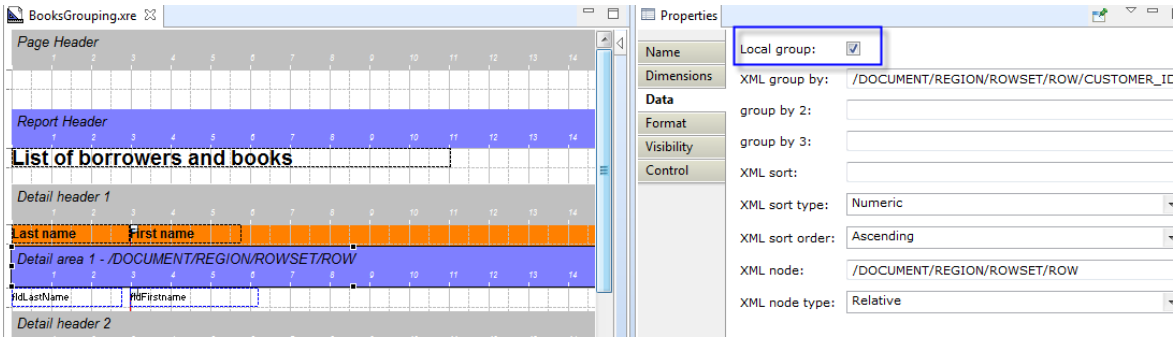
Advanced examples

The following examples show how to use some of the advanced features:

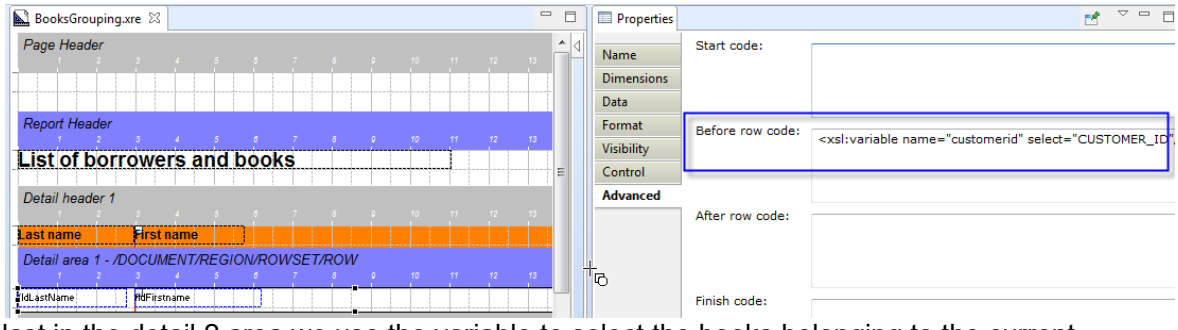
- **BooksGrouping2.xre**: this is a „groupby“ report which creates a master detail structure using a flat input structure. The report groupy by the field CUSTOMER_ID to create a list of persons and a grouped list of booked borrowed by each person



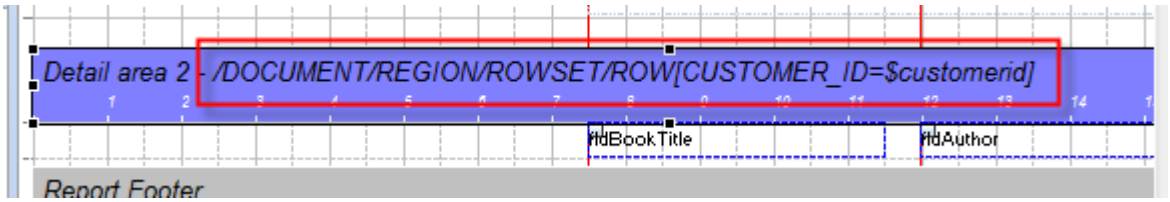
- **BooksGrouping.xre:** This is the same example as above but instead of using the built-in grouping functionality we group the books using a variable. This is done by selecting „Local group“



and setting a variable to store the customer name:

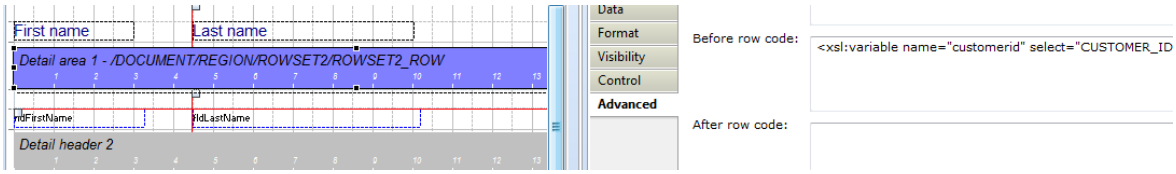


last in the detail 2 area we use the variable to select the books belonging to the current customer:

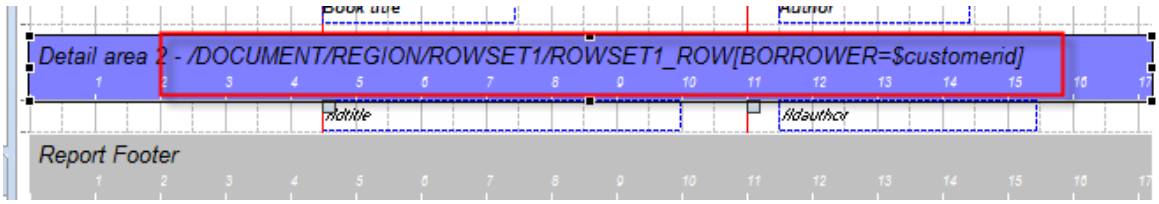


- **BooksAndCode.xre**: This example is similar to the previous one. The difference here is that the input XML is not one flat structure but has 2 parallel structures that will be linked using a variable.

  In the ROWSET2_ROW nodes we have the list of customers which we will print out, this is the main detail area. We save the current customer in a variable for each row.



  In the subarea „Detail 2“ we use the variable to select only the corresponding records from the other structure ROWSET1_ROW.



Using third party XLT templates

You can use third party XSLT template files to extend the functionality of the designer. One example is this library:

https://sourceforge.net/projects/xsltsl/

In the library you will find functions to, for example, format date/time date or string values.

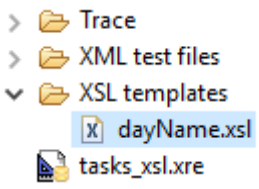
The example *tasks_xsl.xre* shows how to use this feature. The steps are:

1. extract one of the templates from the library. This is required to keep the report size small (do not use the original large files).

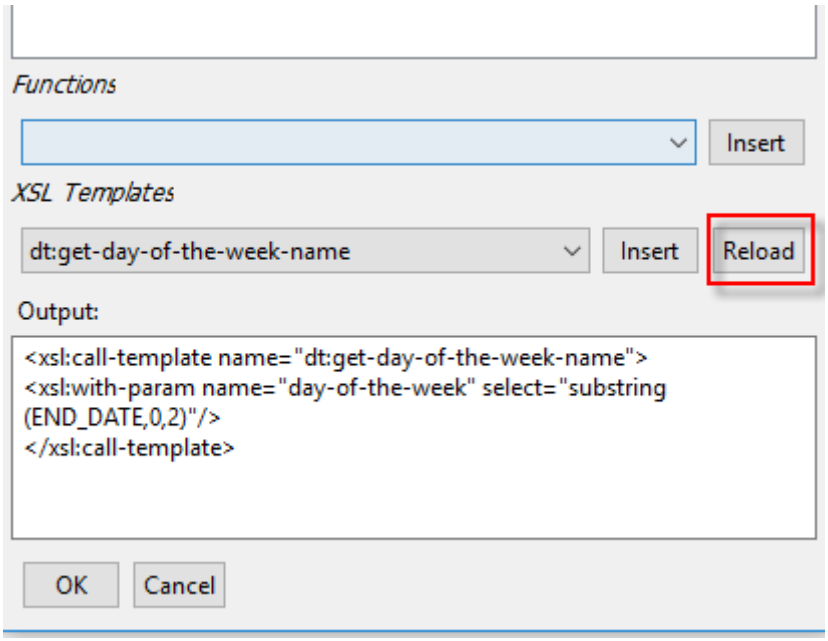
   For example this snippet would convert a day number to the name of the day in english (make sure you have namespace and prefix, see red texts):

```
<xsl:template name="dt:get-day-of-the-week-name" xmlns:dt="http://xsltsl.org/date-time">
<xsl:param name="day-of-the-week"/>
 <xsl:choose>
 <xsl:when test="$day-of-the-week = 0">Sunday</xsl:when>
 <xsl:when test="$day-of-the-week = 1">Monday</xsl:when>
 <xsl:when test="$day-of-the-week = 2">Tuesday</xsl:when>
 <xsl:when test="$day-of-the-week = 3">Wednesday</xsl:when>
 <xsl:when test="$day-of-the-week = 4">Thursday</xsl:when>
 <xsl:when test="$day-of-the-week = 5">Friday</xsl:when>
 <xsl:when test="$day-of-the-week = 6">Saturday</xsl:when>
 <xsl:otherwise>error: <xsl:value-of select="$day-of-the-week"/></xsl:otherwise>
</xsl:choose>
</xsl:template>
```

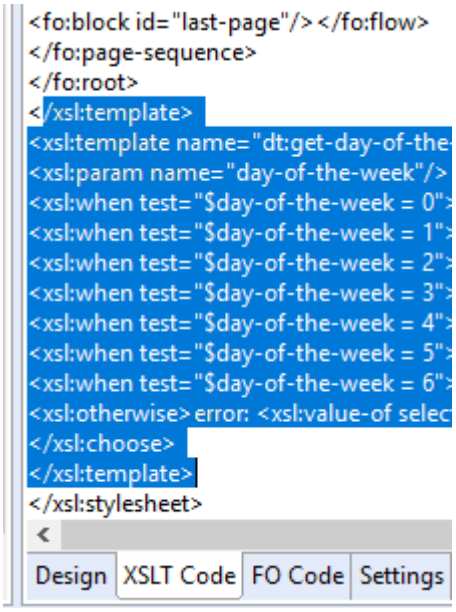
2. copy this code to an XSL file (dayName.xsl) and place it in the designer in the directory as below (create the directory if not existing):



3. Use the reload templates button in the Xpath designer, you can use the insert button to add the new function:



4. The code generated by the designer will now include the external code:



5. The report will now show the result of the external function *get-day-of-the-week-name*

Pam King

List of Tasks

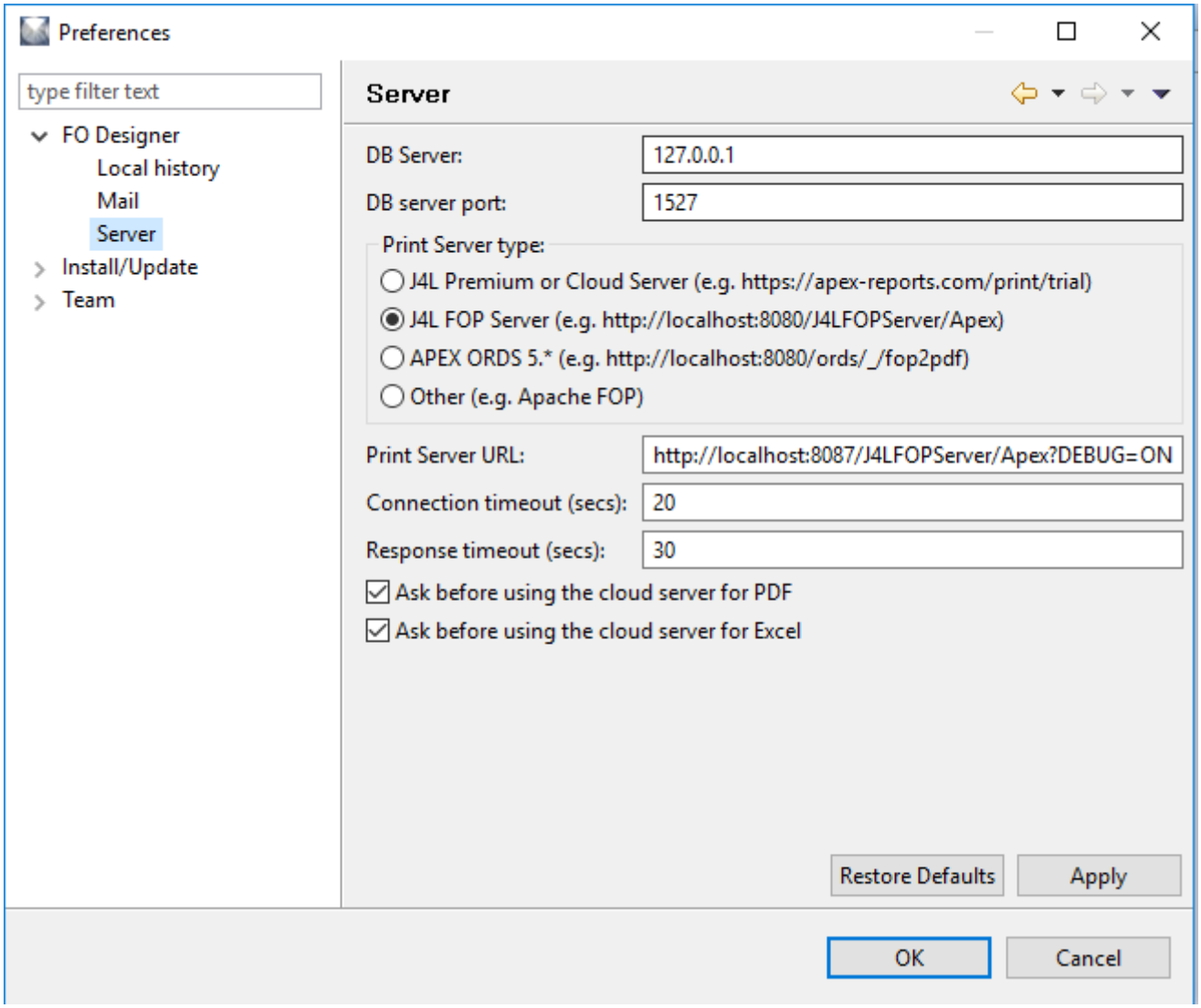
Task name	*Start day*	*End day*
HR software needs upgrade	Sunday	Monday
Apply Billing System updates	Monday	Tuesday
Investigate new Virus	Monday	Tuesday

20. The inhouse Premium and Cloud J4L FOP Servers

The features described in this section are available only on the premium inhouse Server or the cloud server.

In the preferences windows you can select the „J4L Cloud Server“ if you do not have or do not want to install a local FOP Server in your network. It can be used both at design time from the designer and at runtime from Oracle Apex using the URL below for test purposes.

For production purposes however you will receive a separate URL if you have contracted the service.

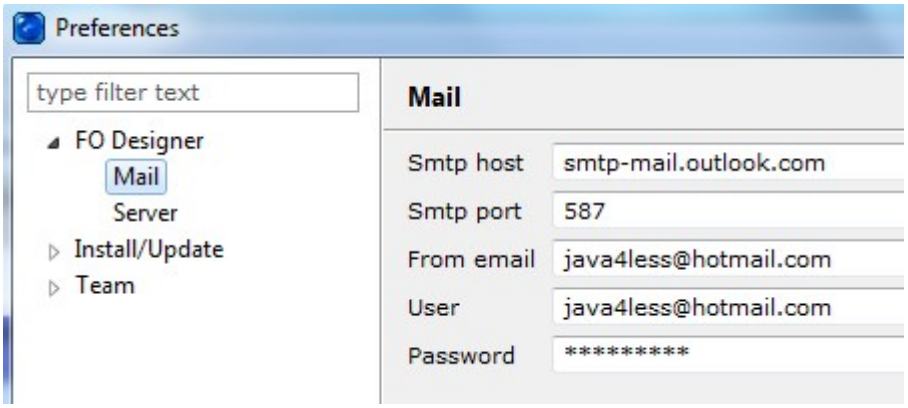


Note you need to **select the correct server type before exporting the XSL-FO file** since for the use of the Cloud server the XSL-FO file will contain a reference to your FO Designer license. This will be used to authenticate your requests.

Sending the PDF per email

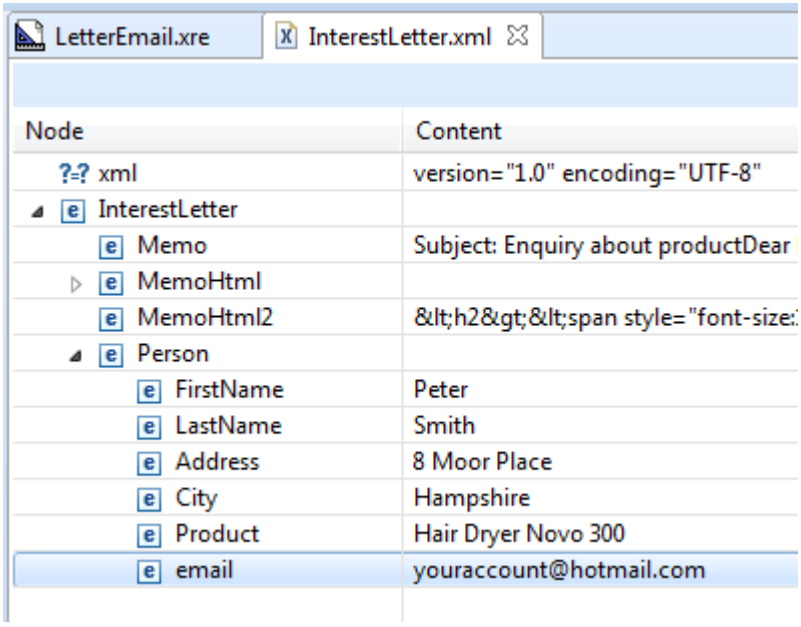
One of the capabilities of the premium and cloud servers is that they can send the report files per email. This feature can also be tested without the server.

First you need to set up the mail server in the windows preferences. This is for example the setup for a hotmail account (of course use your account instead of java4less@hotmail.com)



Second you can test it with our example *LetterMail.xre.* In the properties of the report, the „Send mail“ flag has been selected. Make sure you add also the fields:

- **Mail to:** this is the recipient, it will commonly be an XPATH to a node in the XML file. In this example */InterestLetter/Person/email*

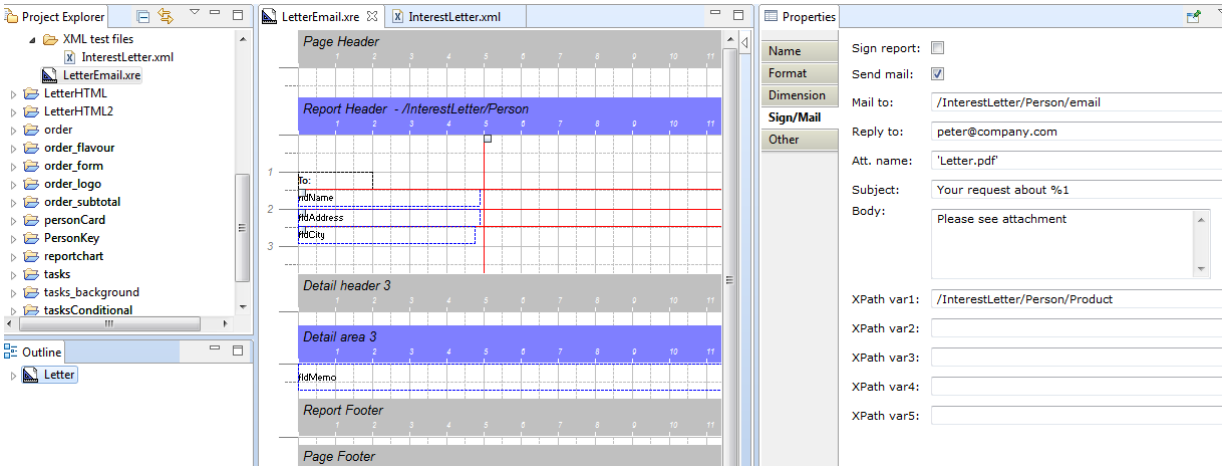


- **Att. Name:** this is the attachment name (PDF). It can be a constant name or you can combine it with data from the XML file. For example you could have:

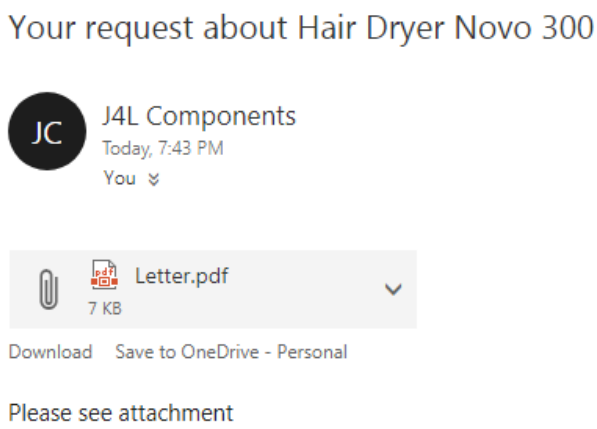
  concat('Letter' , /InterestLetter/Person/LastName , '.pdf')

this would create *LetterSmith.pdf* as attachment name.

- **Subject:** The subject of the email
- **Body:** The body of the email. In these fields you can use %1 to %5. These place holders will be replaced by the Xpath variales 1 to 5. In this example we set the product name in the subject of the email.



The result of executing the report using the PDF button should an email as below:



This will work in the same in the server side since the generated XSL-FO will contain the required service metadata (see below) which will be used by the server to send the email.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:j4lif="http://java

<service xmlns="com.java4less.xreport.fop.server" >
 <mail enable="1" >
  <to>/InterestLetter/Person/email</to>
  <replyto>peter@company.com</replyto>
  <filename>&amp;apos;Letter.pdf&amp;apos;</filename>
  <subject>Your request about %1</subject>
  <body>Please see attachment</body>
  <var1>/InterestLetter/Person/Product</var1>
  <var2></var2>
  <var3></var3>
  <var4></var4>
  <var5></var5>
 </mail>
 <sign enable="0" />
</service>
```

PDF Protection

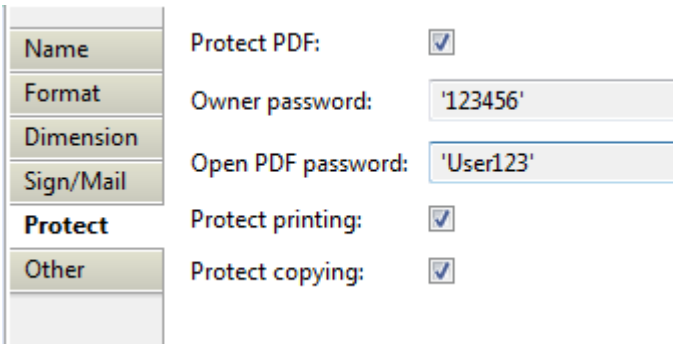
You can protect (encrypt) your PDF files to prevent unauthorized use. This is useful for protecting sensitive data as for example price lists.

You can:

- Ask the end user to enter a password to open a file
- or avoid printing the PDF
- or avoid copy/paste of data

This is how you setup PDF protection. In the *Report properties* > *Protect* tab you can define protection level. The properties are:

- **Protect PDF**: Yes/No.
- **Owner password**: this is is the master password which would allow you to change the permissions if your PDF reader has such functionality. It can be a constant value of a Xpath to the XML document.
- **Open PDF password**: this is the password required to open the document. You would give this password to the intended document recipients.
- **Protect printing**: disables PDF printing.
- **Protect copying/editing**: disables copy/paste or editing.



Note this functionality is available in the Cloud FOP server only.

Creation of Excel files

The inhouse Premium and J4L Cloud FOP servers can create Excel (**xlsx)** files for your reports. The Excel files will include:

- font size, style and color
- cell background color
- overall design of the page
- images (each image must fit in one cell)

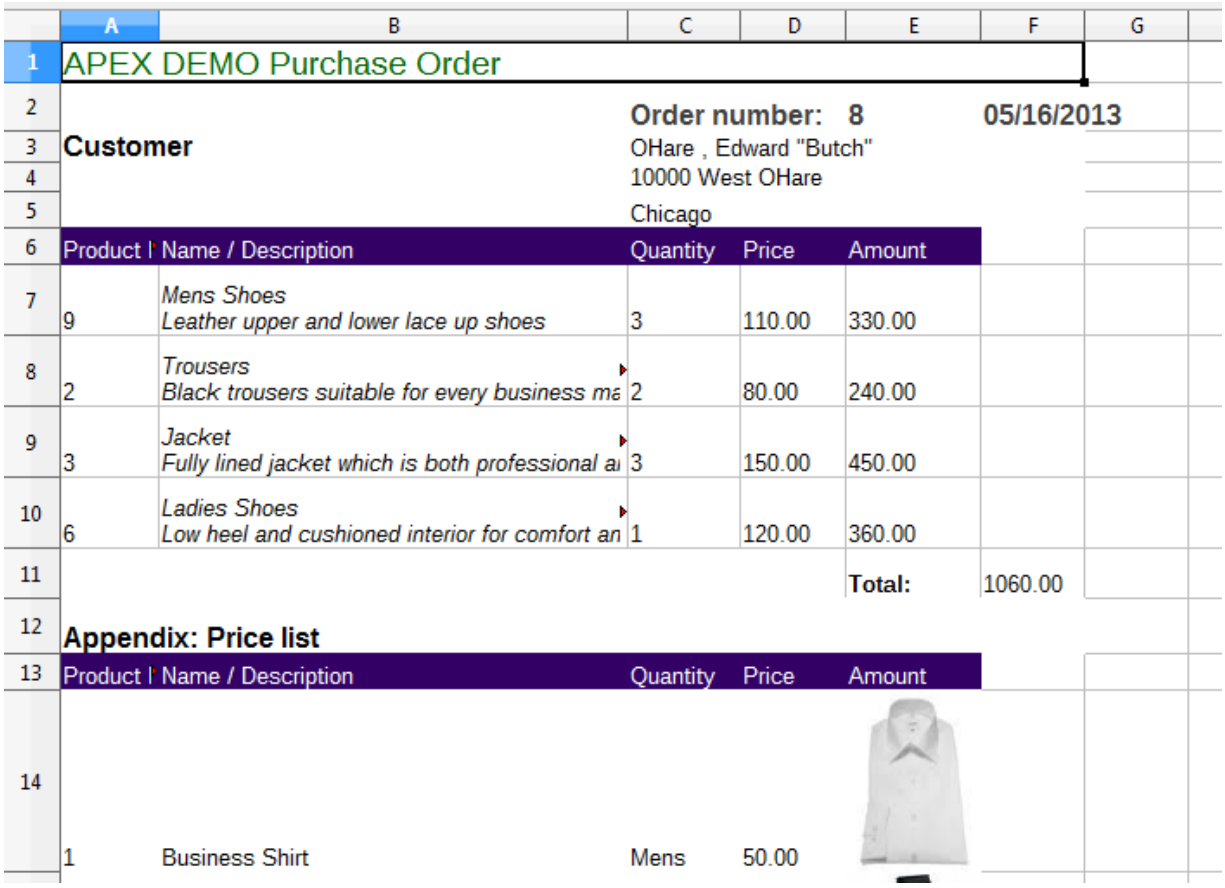
For Excel file you will however not get the same *what you see is what you get* result as for PDF.

Note in order to get the best results in Excel you have to use row and column markers since these will be used to create the rows and cells in the excel file.

Objects as Memo fields, boxes, charts or barcodes are not supported in the Excel output.

When using the server from outside the designer you have to add *?XLS=1* to the URL to get the excel output. For example:

http://eval.apex-reports.com//SaasFOP/ApexTrial ?XLS=1



	A	B	C	D	E	F	G
1	APEX DEMO Purchase Order						
2			**Order number: 8**			**05/16/2013**	
3	**Customer**		OHare , Edward "Butch"				
4			10000 West OHare				
5			Chicago				
6	Product I	Name / Description	Quantity	Price	Amount		
7	9	*Mens Shoes* *Leather upper and lower lace up shoes*	3	110.00	330.00		
8	2	*Trousers* *Black trousers suitable for every business ma*	2	80.00	240.00		
9	3	*Jacket* *Fully lined jacket which is both professional a*	3	150.00	450.00		
10	6	*Ladies Shoes* *Low heel and cushioned interior for comfort an*	1	120.00	360.00		
11					**Total:**	1060.00	
12	**Appendix: Price list**						
13	Product I	Name / Description	Quantity	Price	Amount		
14	1	Business Shirt	Mens	50.00			

21. Migration of existing XSL-FO files.

This section describes how to migrate existing XSL-FO files to the FO Designer. Note we explicitelly use the word **migrate** to remark that the FO Designer is not an XSLT editor, therefore it is not posible to import XSL-FO files 1 to 1. Instead we provide a tool that will preserve most of the layout of your existing report and most of the style (fonts, colors and alignments). However you will still need to do some fine adjustments.

We will improve the tool over the next releases, contact us if you have some files you cannot migrate.

Limitations

You will get the best result will the miigration tool if:

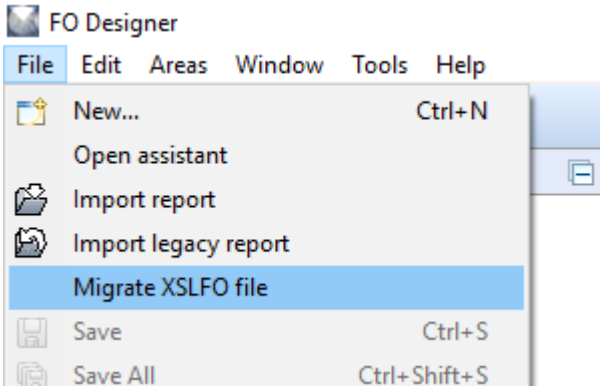
- you only have a limited amount of XSLT code in the file
- and the FOP formating code has been structured in form of FO tables, rows and cells.

Some current limitation in the tool today are:

- The supported XSLT elements are*: variable, attribute, attribute-set, use-attribute-sets, apply-templates, template, stylesheet, with-param, param , choose, when, otherwise, if, key, call-template, output, value-of, comment, decimal-format* and *for-each.*
- The supported FOP elements are: *block, block-container, external-graphic, flow, inline, inline-container, layout-master-set, marker, retrieve-marker, page-sequence, page-sequence-master, region-body, region-after, region-before, root, simple-page-master, static-content, table, table-header, table-body, table-row, table-column* and *table-cell.*
- The supported style attributes are: text-align, font-size, font-weight (bold), font-family, font-style (italic), color, background-color, text-decoration (underline). These are supported for *block, block-container, inline* and *inline-container.*
- The supported position attributes are: *margin-left, margin-top, margin-right, width, height, position, top, left* and *right.*
- Only one simple page master is supported.

Migration step by step

The migration tool is located in the *File* menu:

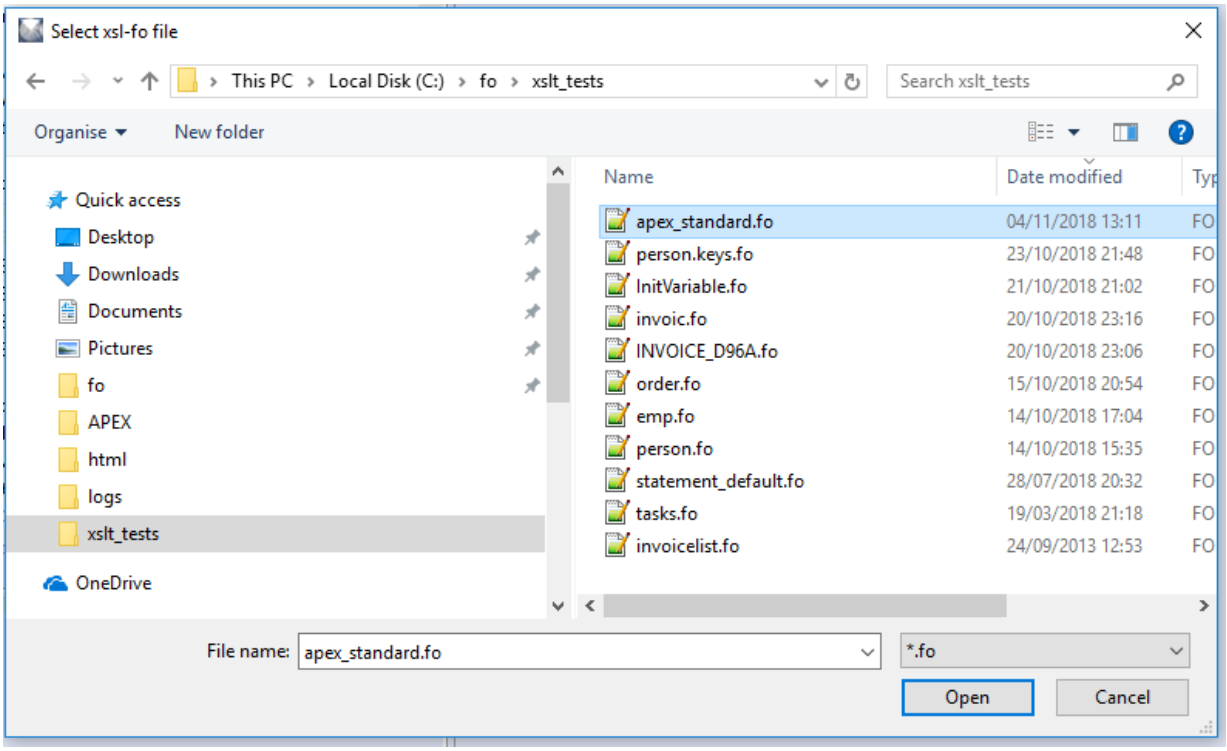


after selecting it, the open file dialog will shown. Make sure you select a valid xsl-fo file:

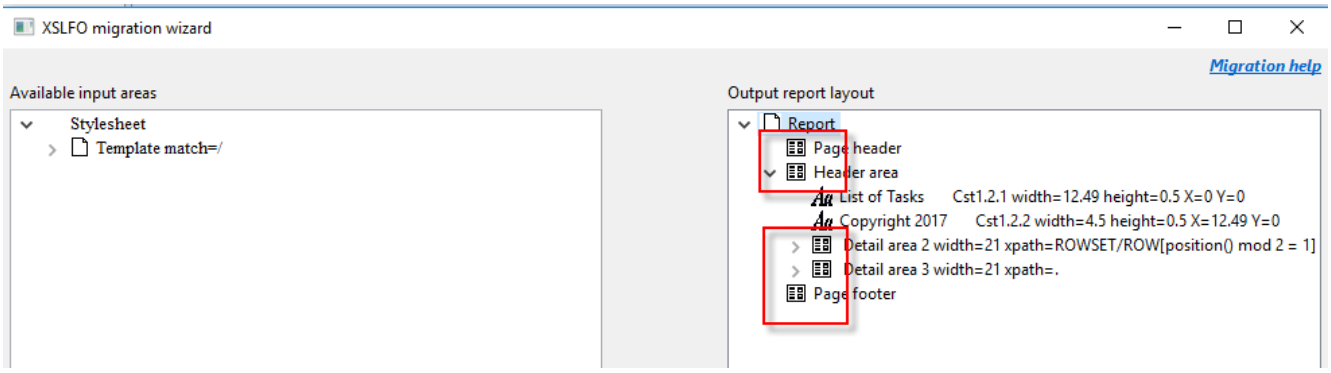
- the file must be UTF-8 encoded
- a valid XSL-FO file starts with

  <?xml version="1.0" encoding="UTF-8"?>
  *<xsl:stylesheet **xmlns:xsl**="http://www.w3.org/1999/XSL/Transform" **xmlns:fo**="http://www.w3.org/1999/XSL/Format"*
  ....
  .....
  *<**fo**:root>*
  .....

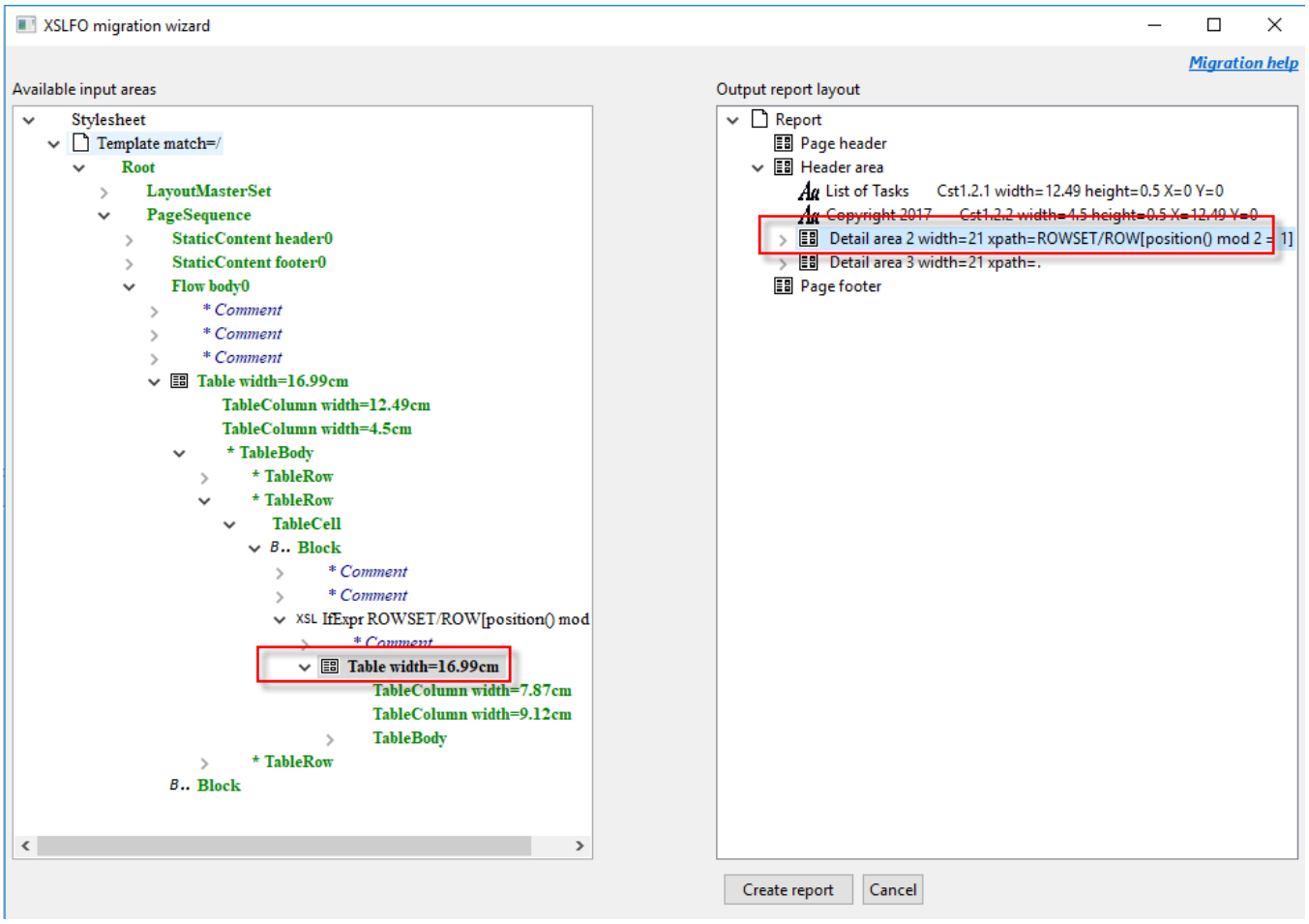
- and contains both XSLT (xsl) and FOP (fo) elements, as in the lines above.



If the file is valid the system will propose the creation of a FO Designer report. On the right hand side the report areas will be displayed (marked below with red squares) and the fields within the areas:



If you double click on the output area on the right (*"detail area 2"*), the corresponding source element in the input file will be selected on the left handside window "*Table width...*" in this case.



As rule of thumb (there are some exceptions), the tool will create a new output area for each one of these input elements:

- Table element. For example:

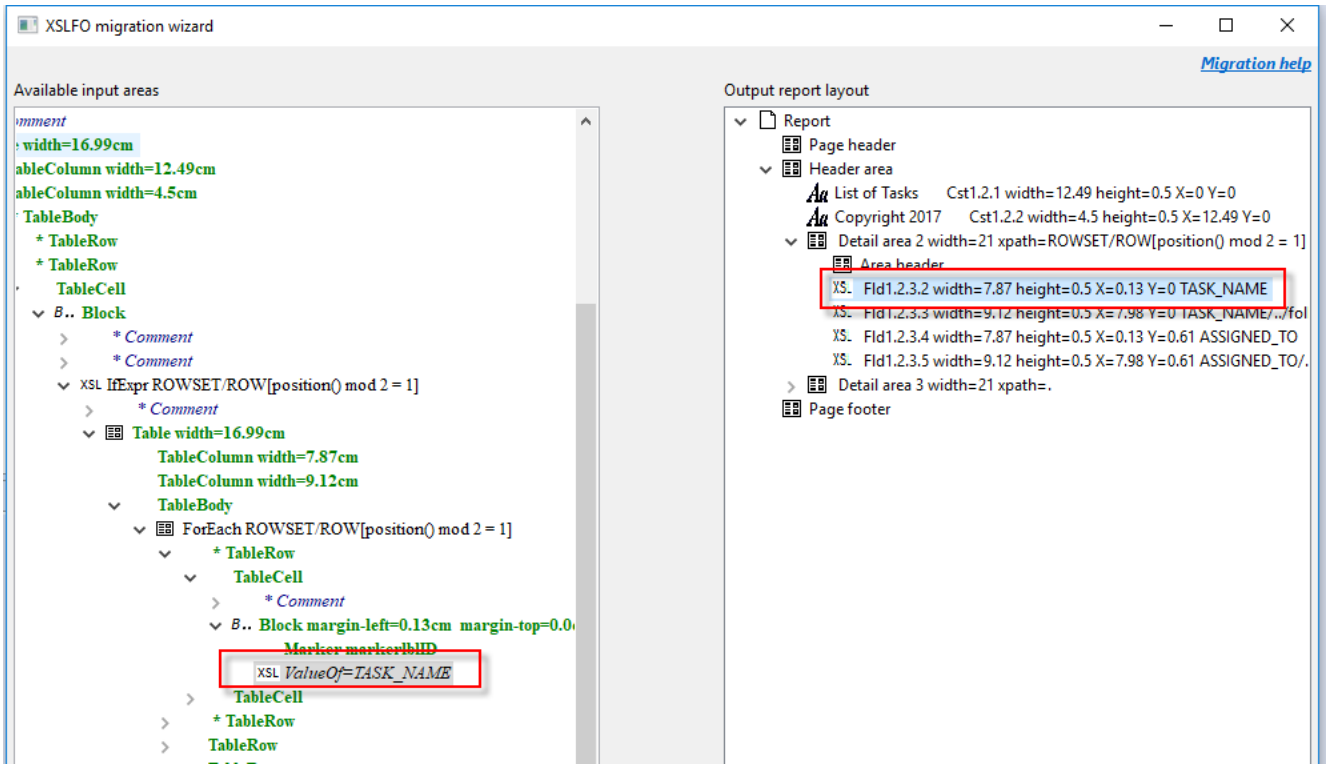
  <fo:table ....
- for-each element. For example:

  <xsl:for-each select="OrderDetail/ListOfItemDetail/ItemDetail">

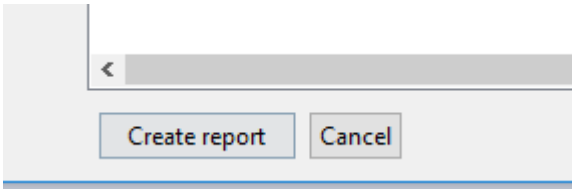
- apply-templates element. for example:

  <xsl:apply-templates select="/LIST/S_UNB/S_UNH" />

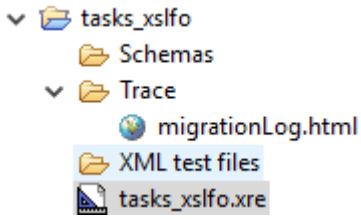
The same concept applies for output fields which will be mapped to input *"value-of"*:



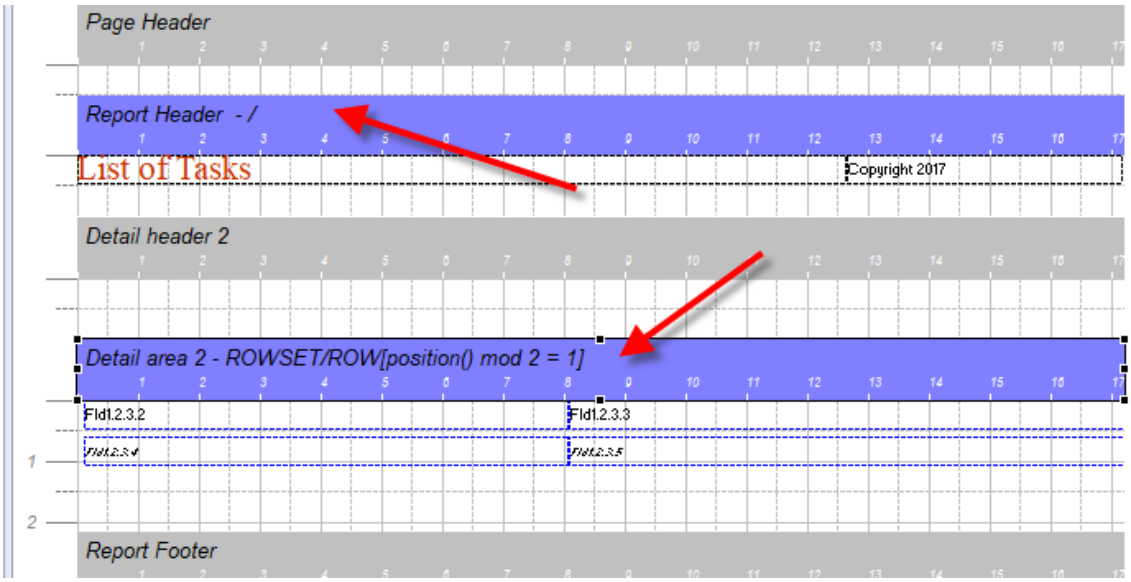
Use now the "Create report" button



this will create a new project and the report template.



the next step is checking whether the Xpath's of the areas are correct. Note for this you need to have an understanding of how the designer works. If you are completely new to the FO Designer please have a look at our "Learn by doing" chapter in the main help page.
Afterwards check the individual XPath of th fields.



In the project there will be also a file called *migrationLog.html.* In this file you can see what the designer was able to migrate and how. Look for the OUTPUT lines with yellow or red background color and whose text is "IGNORING" or "ERROR". These will give you a hint on what you need to manually correct in the report.

```
IfExpr ROWSET/ROW[position() mod 2 = 1]
 OUTPUT  IGNORING IfExpr.
  * Comment
Table width=16.99cm
 OUTPUT  Create Area Detail area 2.
 TableColumn width=7.87cm
 TableColumn width=9.12cm
 TableBody
  ForEach ROWSET/ROW[position() mod 2 = 1]
    * TableRow
    TableCell
      * Comment
     Block margin-left=0.13cm  margin-top=0.0cm  margin-right=0.35cm
      Marker markerlblID
      ValueOf=TASK_NAME
      OUTPUT  Create Field Fld1.2.3.2 = TASK_NAME
      OUTPUT   x=0.13 y=0.0 width=7.87
    TableCell
```

The final step would be running the report.

For this:

- copy a test XML file to the "*XML test fles*" folder of the project
- select the file in the "*settings*" tab
- and click the "*PDF*" button (1 below) in the tool bar.

In case the output is not the expected one, use the trace button (2 below). Please refer to the chapter "*Structure of the template -> How to analyze the report executio*n".

22. FAQS

How to prevent NaN values in numeric fields

If you are using the *format-number* function or another numeric function on a non numeric field (or a numeric field that is empty) you will get the value **NaN** in your report. If you want to remove the NaN values you can use the *j4lext:replaceStr* function to replace them with an empty string.

For example if your current area is based on a XML node that has a child node called QUANTITY, which you want to format using the format-number function you can use the following xpath expression:

j4lext:replaceStr(format-number(QUANTITY, '##.0'),'NaN',' ')

this would prevent the NaN values in case the QUANTITY node is empty (empty string).
Note however if you use functions with the *j4lext* prefix you must use the J4L FOP server, if you use another XSL-FO processor these functions are not available.

How to enable the designer console

Edit the file fodesigner.ini in the installation directory, add the line

-consoleLog

after

-startup
plugins/org.eclipse.equinox.launcher_1.3.0.v20130327-1440.jar

Restart the designer, now the designer console will be shown, where you can eventually see error messages.

How to debug Oracle APEX requests

If you are using Oracle APEX and:

- the report is working in the designer
- but not in Oracle APEX

this is what you need to check:

- make sure the XSL-FO file has encoding UTF-8, depending on your APEX installation you might need to change it to ANSI.
- If you using Oracle Data Rest Services as print server, make sure the „APEX encoding“ has been disabled in the designer

- If none of the above helps, try removing fields from the report one by one until the one causing the problem can be found. Look first at field having potential issues like have international characters in the text.
- the last alternative is installing the J4L FOP Server (http://www.java4less.com/apache/fop.php), for debugging purposes the server will allow you to log the APEX request (http://www.java4less.com/apache/fop.php?info=faq#debug) and find out where the problem is.

How to enable Oracle APEX logging on Glassfish

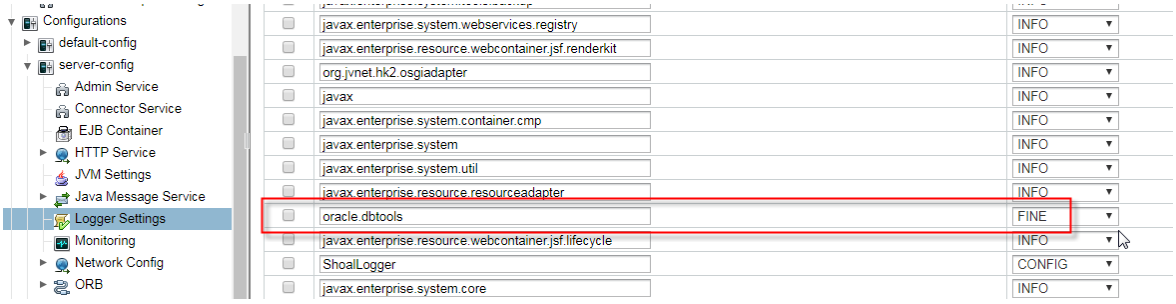
You can enable 2 types of loggings:

1. In the ORDS configuration file *default.xml* set this value:

   <entry key="debug.debugger">**true**</entry>

2. On the Glassfish administration console → Configurations → server-config → Logger settings, add the entry „oracle.dbtools“ and set the desired log level.

   The log file will located in the directory *glassfish\domains\yourdomain\logs*



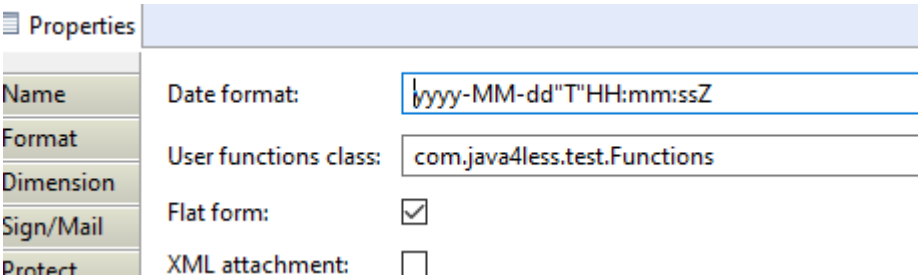
How to create a user defined xpath function

If you want to create your own function the following example shows how to do it:

1. Create a Java class with your favorite development tool. The class should contain your function as an **static** method, for example, we will create a function called *toUpperCase*

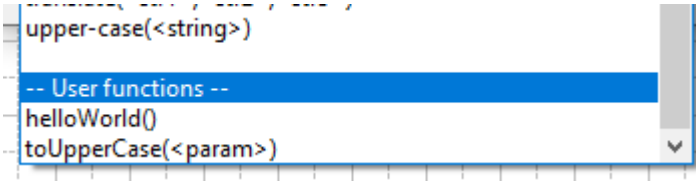
```
package com.java4less.test;

public class Functions {

/**
* this function shows a hello world message
*/
public static String helloWorld() {

        return  “hello World“;
}

/**
* this function converts a string to upper case
*/
public static String toUpperCase(String input)
        { return input.toUpperCase();
}

}
```

2. Compile the java class into a jar file. The file must be called **userfunctions.jar** .
3. Place **userfunctions.jar** on the *userfunctions* subdirectory.
4. Stop the J4l FO Designer.
5. Run *InstallUserFuncitoins.cmd* to install the file.
6. Start the J4L FO Designer
7. In the properties of your template you must enter the Java class name in the *User functions class* field, see highlighted field in the screenshot, **then close and reopen the report** in the editor.



5. Now you can use your function in the XPath editor since the function will appear in the list of available functions:

Note: When you insert the function in the output, the function name will have the prefix **j4luserext:**

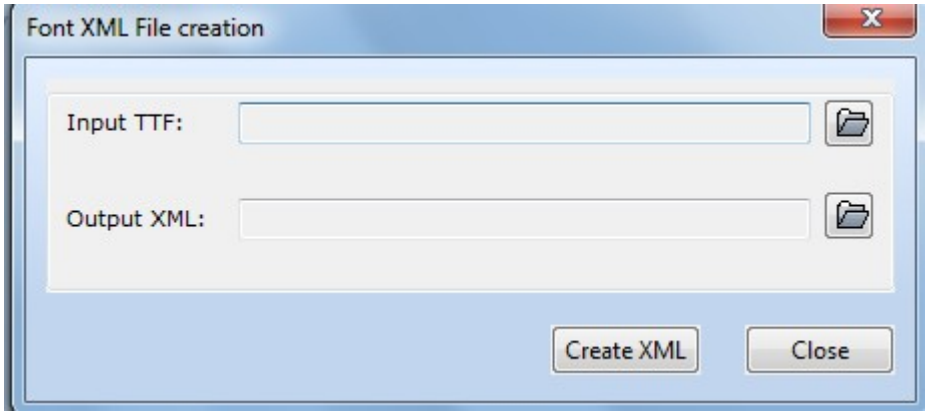


How to add new fonts to J4L FO Designer

J4L FO Designer supports the five built-in PDF base fonts (Helvetica, Times, Courier, Symbol and Zapfdingbats) which must be supported by any PDF reader.

However you can add new fonts using a TTF file in the following way:

1. Open the FO Designer and select *Tools -> TTF Font tool*



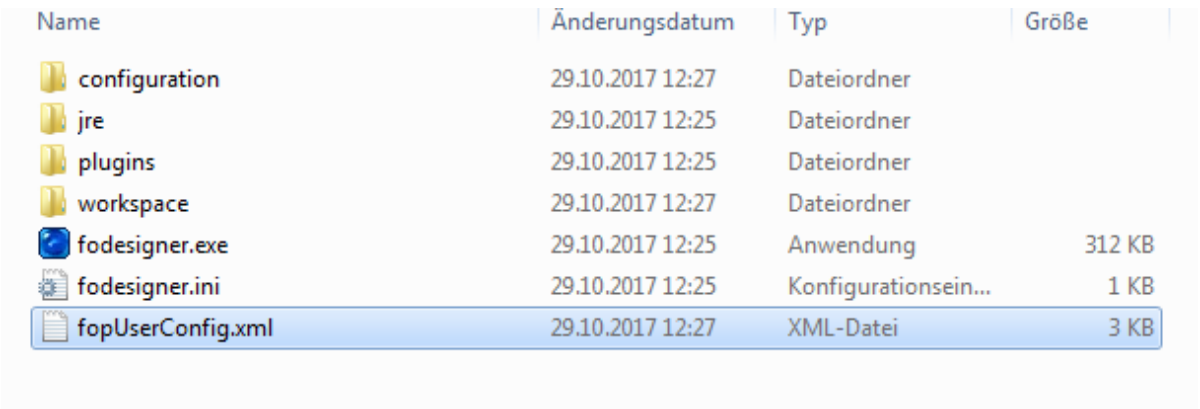
   the *Input File* must point to your TTF file and the *Output XML* file to the output metrics file (you can select any name you like).

2. Click on „*Create XML*“ and close the dialog.
3. Edit the created XML metrics file and make sure the *font-name* element has the same value as the *family-name* node. For example:

   <?xml version="1.0" encoding="UTF-8"?> <font-metrics metrics-version="2" type="TYPE0">

   <font-name>Comic Sans MS</font-name>
   <full-name>Comic Sans MS</full-name>
   <family-name>Comic Sans MS</family-name>

4. Now you have to tell J4L FO Designer to use the XML metrics file you created. You do this by editing the file *fopUserConfig.xml* located in the root J4L FO Designer directory.

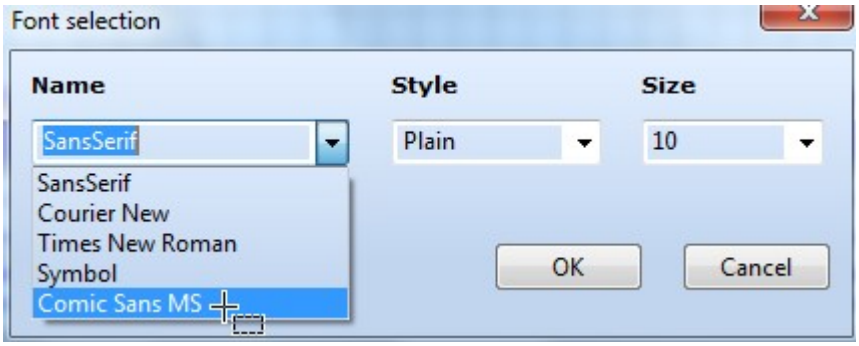


You have to add these lines:

<font metrics-url="**file:///c:/yourdirectory/comic.xml**"
kerning="yes" embed-url="**file:///C:/windows/fonts/comic.ttf**">
<font-triplet name="**Comic Sans MS**" style="normal" weight="normal"/
>

note you have to change the *metrics-url , the embed-url* and the *name* values. The name must be the same *family-name* value from step 3.

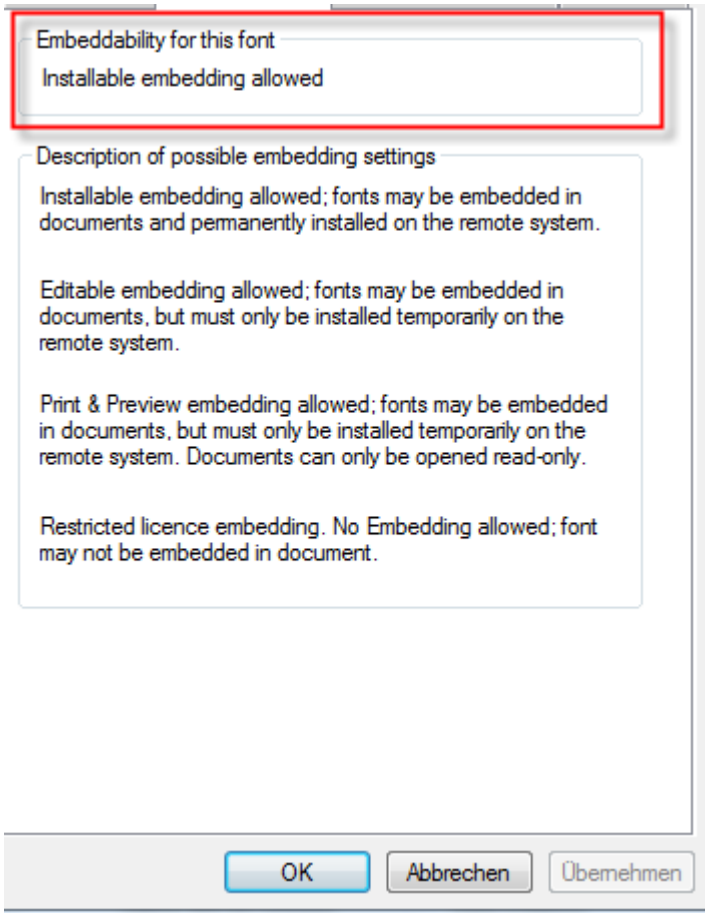
5. Now you can start the J4L FO Designer and open the font dialog of any field, the new font will be available in the font selection dialog:



Note about microsoft TTF files

If you want to know if you may use microsoft TTF files in your PDF Document you need to:

1. Download the Microsoft ***Font properties extension*** (https://www.microsoft.com/en-us/Typography/TrueTypeProperty21.aspx)
2. Install the software
3. Located the TTF File you want to use and select right mouse click, properties. If the properties extension has been installed you will see a new tab called „*Embedding*“ where you can see the license type for embedding the fonts in documents.



How to add new fonts to Oracle ORDS

Once you added the new fonts to the designer as explained in the previous FAQ, the next step is adding it to the runtime server. This section will show how to add new fonts to Oracle ORDS.

- Edit the ORDS configuration file *default.xml ,* add the following line:

  <entry key="fop.configfile">**C:\ords\ords**fopUserConfig.xml</entry>

  Replace the red text with your own directory (place the fopUserConfig.xml where you like)

- In the *fopUserConfig.xml* file (which you can copy from the FO Designer installation directory) you might have an entry like this:

  <font metrics-url="file:///C://fonts//verdana.xml" kerning="yes"
  embed-url="file:///C://fonts//verdana.ttf">
  <font-triplet name="**Verdana"** style="normal" weight="normal"/>

  This will add Verdana font support to your reports.

How does the designer deal with namespaces

The xsl-fo file generated by the designer is not namespace aware. This means the XML to PDF conversion will fail in Apache FOP if your input file contains namespace information. In order to avoid this error you have to:

- At runtime you use:
  - either our servlet which supports the REMOVENS parameter
  - or our *com.java4less.xreport.fop.FOProcessor* class whose *process()* method supports the *removeNS* parameter that must be set to true.
- In the designer activate the namescape removal. In the designer activate the namescape removal. This has to be don ein the windows-> preferences dialog .

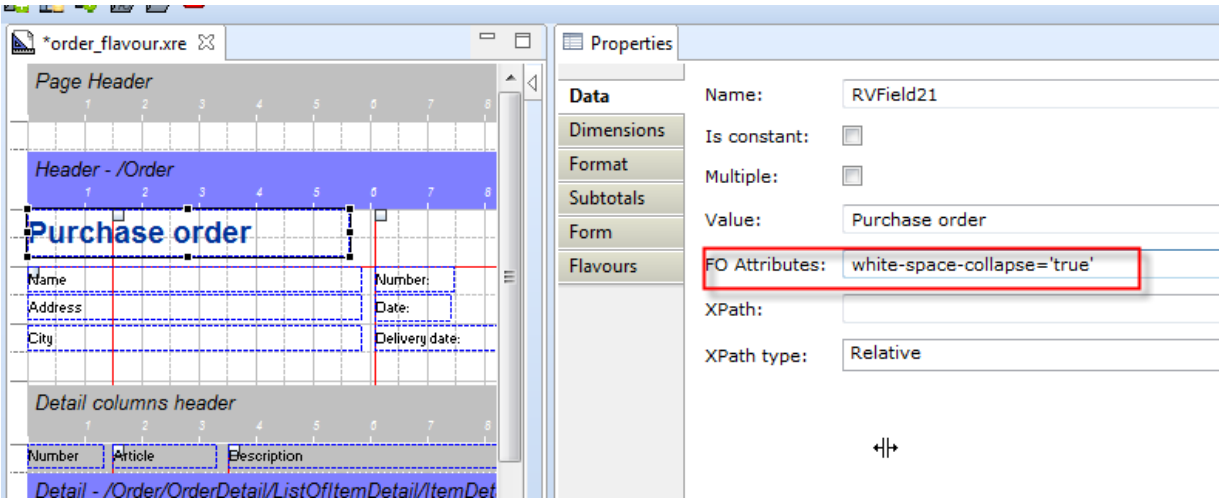
How does support for international character sets work

You can use J4L FO Designer with any language, the following features support internationalization:

- You can **add new fonts** that support your language.
- The XSL-FO files generated by the designer use UTF-8 encoding, so any national character can be used.
- At runtime UTF-8 encoding is recommended however the runtime module will look into the XML preamble to determine the correct encoding.

How to add my own xslfo attributes to the fields

In most cases FO Designer will generate the xsl-fo output so you do not have to worry about the details of the language. However if you are familiar with xsl-fo and you want to add some additional attributes to the ***fo:blocks*** created by FO Designer you can use the property shown below:

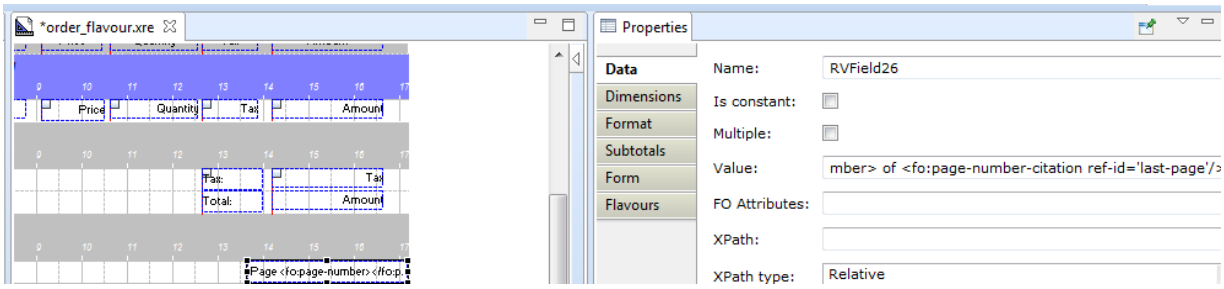


How to add page numbers and page total count

You can add page numbers and page total counts in the page header and page footer areas using the following value for the field:

Page <fo:page-number></fo:page-number> of <fo:page-number-citation ref-id='last-page'/>

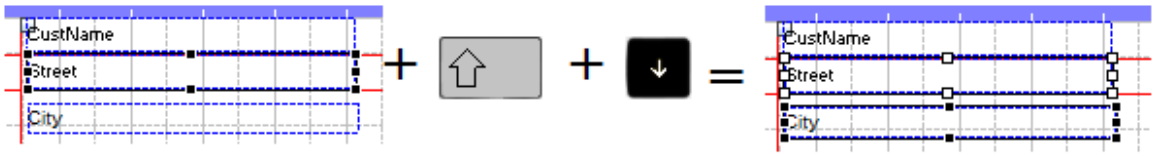
the text *<fo:page-number></fo:page-number>* stands for the current page number and the text *<fo:page-number-citation ref-id='last-page'/>* will be replaced with the total page count.



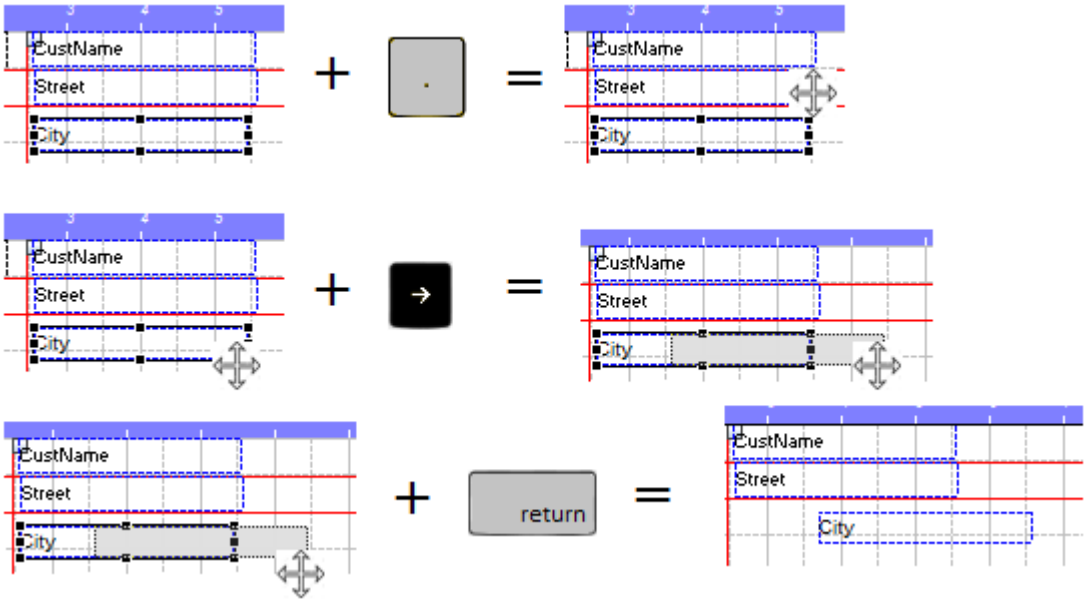
23. Shortcuts

- **CTRL+C**: copy fields
- **CTRL+X**: cut fields
- **CTRL+V**: paste fields
- **CTRL+Z**: undo last field operation
- **Left , right, up and down key**: navigate to neighbour fields. Example

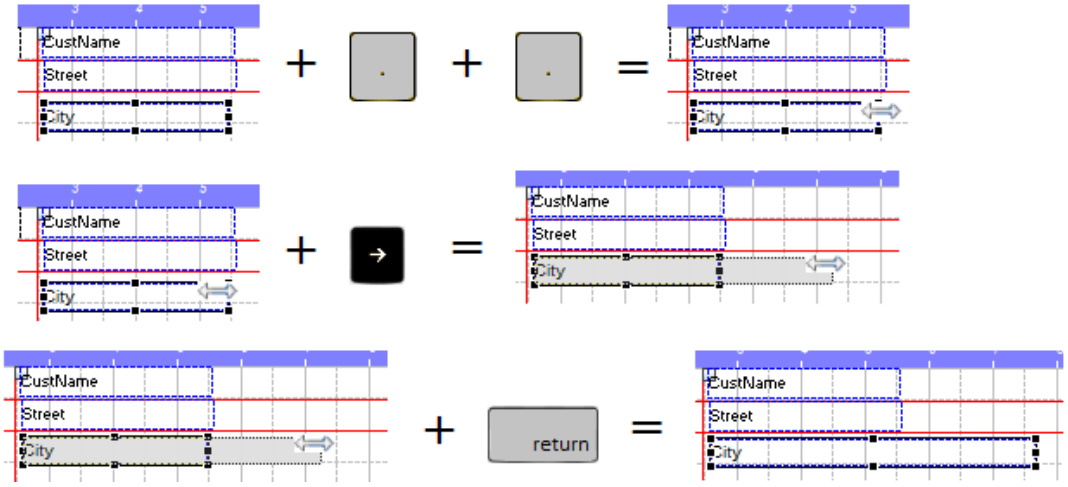
- **Shift + click**: multiple field selection
- **Shift + navigation** (Left , right, up and down) keys: multiple field selection. Example



- **Period (once) + navigation** (Left , right, up and down) keys: move field, press enter to set location



- **Period (multiple) + navigation keys**: resize field.



- Alt+Up key and Alt+down key: template navigation (including, areas, masters, report and fields)

24. Troubleshooting

Permission error while creating a PDF or saving a report

If you get a permission or database error while saving or creating a PDF file (common error in Windows 7), make sure your user have read and write permissions for the FO Designer folder. Using right mouse click on the folder you can set the permissions.

25. Third party licenses

This product is based on the Ecilpse platform (all associated products are based on the Eclipse Public License, https://www.eclipse.org/legal/epl-v10.html) and uses as runtime module the Apache FOP artifacts (**http://xmlgraphics.apache.org/fop/license.html**), the Bouncy Castle library, the Apache PDBox library, the Apache Jempbox library , the Apache Fontbox library , the Apache Derby database and the iBatis persistence artifacts.

The Bounce library has an own license type described in the file lib/bounce_license.txt (inside the file plugins/com.java4less.fo2*.jar) .

The lib subdirectory of the delivery file (inside the file plugins/com.java4less.fo2*.jar) contains all jar files together with the respective license and notice files.

The version with the windows installer also includes the Java Runtime environment in the JRE subdirectory with the respective license information.

26. Contact

You can contact us at **java4less@confluencia.net** if you have any question.