

J4L FO Designer 1.6

Contents

J4L FO Designer 1.6	1
1. Introduction	5
Requirements	5
Setup & startup	5
Windows installer	5
Manual installation	6
Start up	6
Delivered files	7
2. The user interface	9
The information panel	9
Properties tab	9
XML Schema tab	11
The test data tab	12
The work panel	13
The code tab	16
The toolbox bar	17
Versioning system	19
The XPath editor	20
Working with the user interface	21
Selecting objects	21

Operation on objects	22
Operations on areas	23
The settings dialog	23
User administrator	24
3. Structure of the template	24
Page masters	24
Areas	27
Background colors and images	31
Background area	32
Columns and rows markers	32
Drawing lines with columns and rows markers	35
Properties of the objects	36
Properties of the template	36
Properties of the areas	36
Properties of a text field	37
Properties of a combobox	38
Properties of a picture	38
Properties of a line	38
Properties of a www links	39
Properties of a free code objects	40
Example 1	40
Example 2	42
4. Executing the template to create PDF files	44
FOP web server (servlet)	44
Java objects to PDF conversion	46
Sending the PDF as an email attachment	47

Adding a digital signature to the PDF file	47
5. Learn by doing: tutorial.....	49
6. The invoice IDOC example explained	57
how to work without a schema file.	60
Use of second level detail areas	60
Use of conditions in the XPath.....	61
Use of combo boxes.....	62
7. "Group by" example	63
Group footers	64
8. The Barcodes and dynamic images example explained.....	65
9. The chart example explained	69
10. The running totals example explained	70
11. PDF interactive forms.....	72
Introduction	72
Form objects.....	72
Text field.....	73
Combo box	73
Checkbox	73
Button.....	73
Filling in the form	74
Form submission	74
Input form example.....	75
12. Flavours	78
13. FAQs	80
How to prevent NaN values in numeric fields.....	80
How to create a user defined xpath function.....	80

How to add new fonts to J4L FO Designer	82
How does the designer deal with namespaces	83
How does support for international character sets work.....	84
How to add my own xslfo attributes to the fields	84
How to add page numbers and page total count.....	84
14. Startup troubleshooting	85
FO Designer does not start up.....	85
Error opening jnlp file from.....	85
Permission error while creating a PDF or saving a report	85
15. Third party licenses	86
16. Contact	86

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 5 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 XRP file (also called document ***template*** 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**. You load and save XRP files from the designer using the load and save buttons.
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 output ***PDF file***.

Requirements

FO Designer requires:

- Java 1.6 or later (included in the windows installer version)
- Adobe reader to view the generated PDF files
- Browser (for example Internet Explorer or Mozilla Firefox)
- Apache FOP (included in the delivery). See also third party [licenses section](#).

Setup & startup

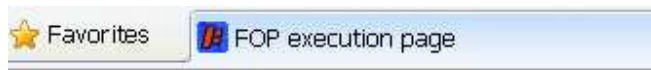
Windows installer

If you have downloaded and run the windows installer, you can start the FO Designer from the windows menu.



If you are using the multiple user version you can access the designer remotely from the URL:

<http://servername:8087/J4LFOPServer/FODesigner.html>



<http://www.java4less.com>

Manual installation

If you have downloaded the ZIP file you have to create an empty directory and unzip the file in the directory.

The designer has to be started with the **run.bat** file in the root directory of the delivery.

If you have problems with **run.bat** maybe you are not using the right java version (1.6). You can check that by running **runDebug.bat** instead. If you get an error like "*Bad version number..*" it means you are using a java version below 1.6. If you do not want to install a new java version on your computer, you can use our windows installer (see download section in our web).

Start up

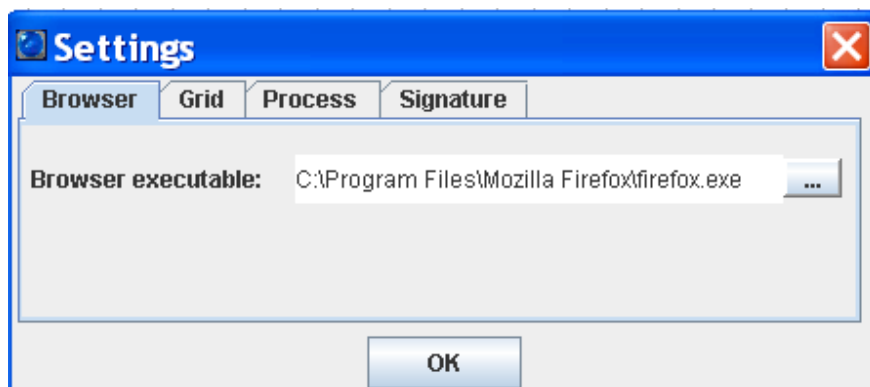
After starting the designer using one of the methods mentioned above, you have to login with user "admin" and password "admin"



In case you have problems see: [Start up troubleshooting](#)

Starting with FO Designer 1.6 there is no need to setup the PDF reader for viewing the generated PDF documents. However if this is not working on your computer, proceed with the setup below.

When you generate a PDF document you have the option to open it manually or let the designer open the PDF document for you, in this case you must first go to the main menu, administration, settings item and enter the path to the internet browser. In the case of windows it is the path to IEXPLORER.EXE (Internet Explorer) or Firefox.exe (Mozilla Firefox), as you see in the following screenshot:



Delivered files

The delivery includes the following files and directories:

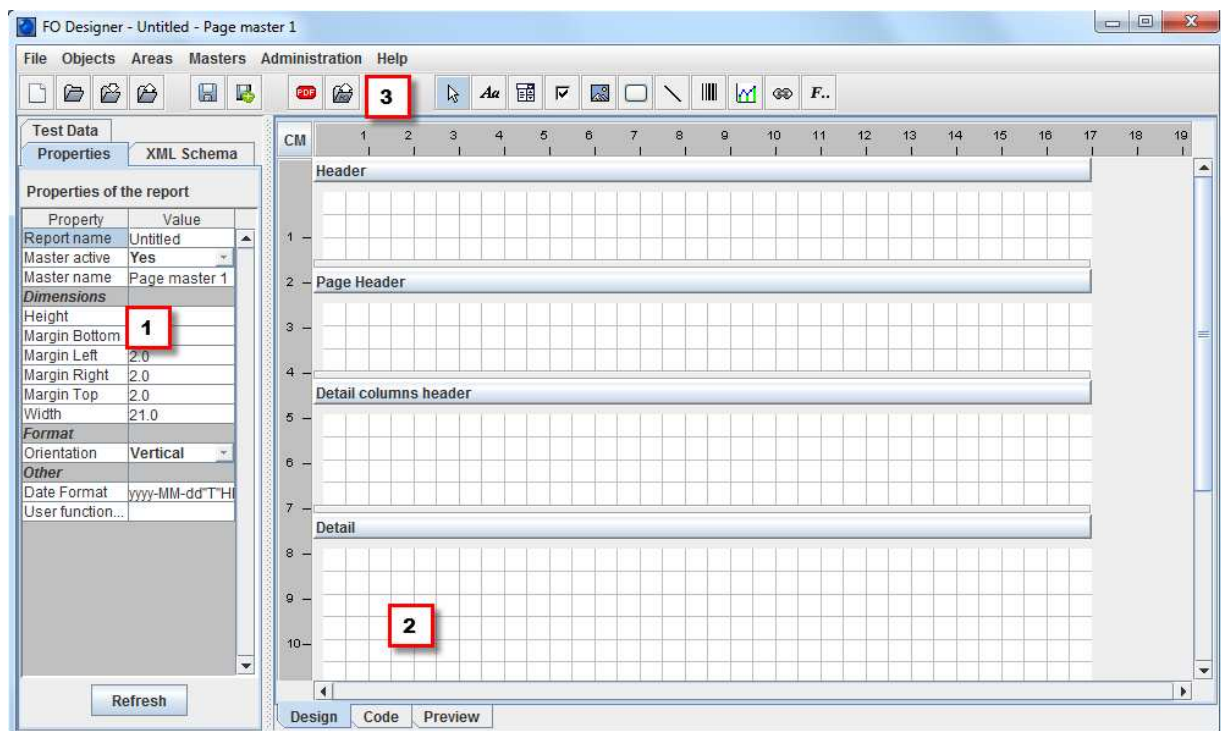
- **examples:** several examples including schema, xml files and template for the designer
 - **employees:** example for the [tutorial section](#) in this guide, including grouping and [cover page example](#).
 - **idoc_invoice:** SAP IDOC invoice example, [explained later](#) in this directory.
 - **xcbl_order:** 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:

- **images:** directory that contains images files for the examples
- **db:** database directory
- **lib:** Apache FOP files
- **certs:** certificate files for testing the digital signature
- **web:** file for deployment to web server as servlet
- **help:** this user guide
- **j4IFODesigner.jar and j4IFORReport.jar:** FO designer main files.

2. The user interface

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

1. On the left side there is an information panel which contains 3 tabs:
 - The first one displays the properties of the selected object (template, area or field).
 - The test data tab contains a XML document used for testing purposes. This XML document can be loaded , edited and saved to a file.
 - The XML schema tab contains the XML document definition, this one is used for selecting fields which can be placed on the document.
2. On the right side you can see the work panel which is divided in areas (header, page header). The work panel has 3 tabs at the bottom
 - The **design** tab where you can design the report.
 - The **code** tab where you can see the o6utput XSL-FO code. If you have selected an object in the design tab, the code tab will automatically scroll and highlight (in yellow) the selected object.
 - The **preview** tab where you can see the generated PDF file.
3. Below the main menu you can find the toolbox bar.



The information panel

The information panel contains 3 tabs:

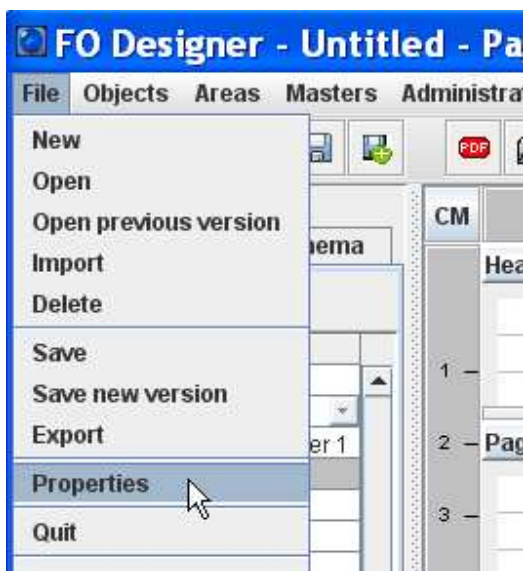
Properties tab

This tab displays the properties of the current selected object. For example this screenshot displays the global properties of the document:

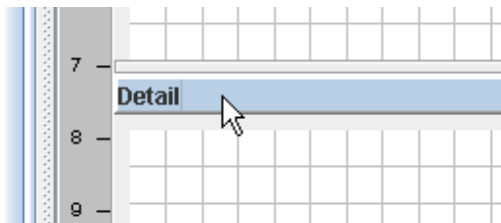
Properties		XML Schema
Properties of the report		
Property	Value	
Name	Untitled	▲
Dimensions		
Height	29.7	
Margin Bottom	2.0	
Margin Left	2.0	
Margin Right	2.0	

There are 3 ways for viewing the properties of an object:

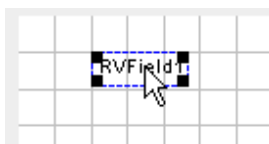
- Select main menu , file , properties to display the global properties of the current template (for example the page size):



- Click on the name of an area in the work panel to display the properties of an area:



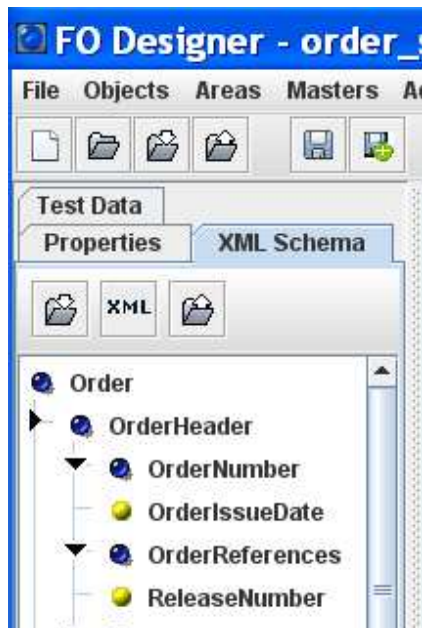
- click on an object (field, combobox, picture ...) to display the properties of the object:



In the [properties section](#) you will find more information about the meaning of the properties.

XML Schema tab

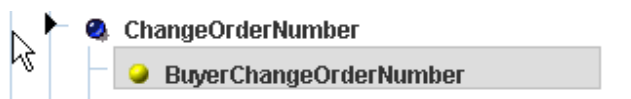
Before you start designing your template you need to load the XML schema using the import button in the XML Schema tab:



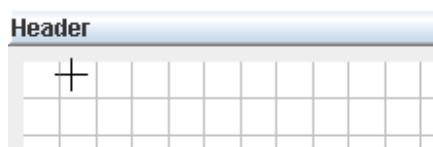
In the XML schema you have a tree structure which contains all possible elements of the input XML document. The blue nodes represent complex element which have children and the yellow ones are elements without children. Yellow icons with a red A on it, represent XML attributes.

You use the elements in the schema to add the XML elements to the template in the following way:

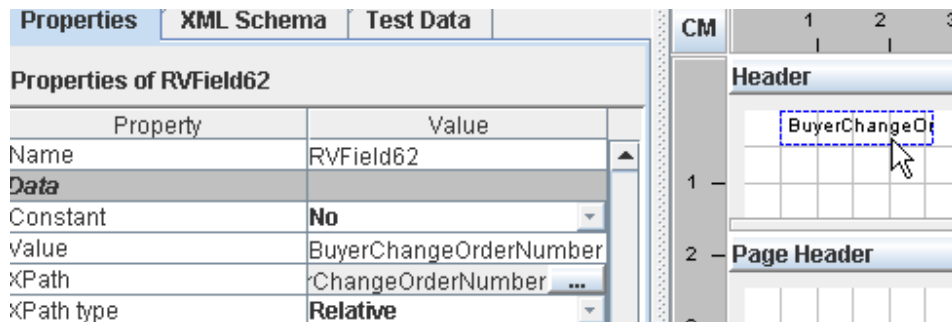
1. Select the XML element.



2. The cursor will change to the crosshair cursor when it rolls over the work panel. Move it to the position where you want to place it and click:



3. The field will be added, as you can see in the screenshot the XPath property of the new field will reference the XML node you selected in the schema:

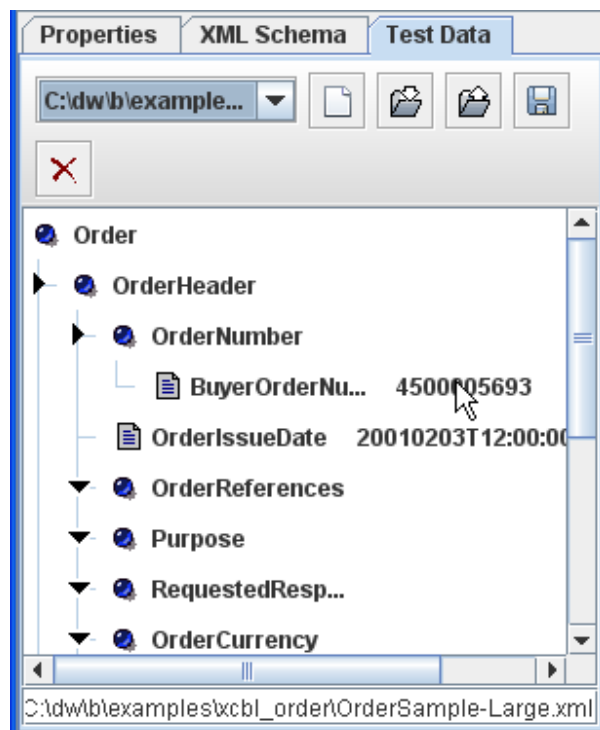


Additionally to the tree, the panel contains the following items:

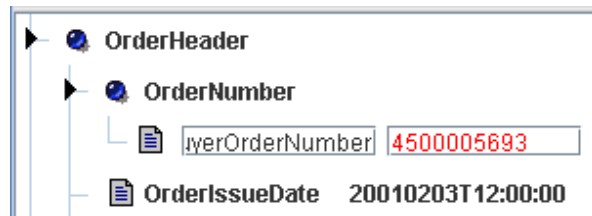
- The load XML schema button to load a new schema file (XSD file).
- The create XML schema button to generate an schema from a test XML file (see test data tab in the next section). This should be used only if you have no real XSD file.
- The export XML schema button to export a previously imported XSD file.
- The status bar below the tree which contains the name of the loaded schema (xsd) file.

The test data tab

This panel displays and allows modification of the input XML document which will be used for testing the XSL-FO execution in the designer. The designer allows you to load several test XML files and save them to the database. In this way you can test your report with different examples or scenarios.



The tree displays the XML content, on the left side you see the name of the elements and attributes and on the right side the value of the elements or attributes. You can edit the values of the elements with **triple click**:



You can also add or remove nodes to the tree using the popup menu:



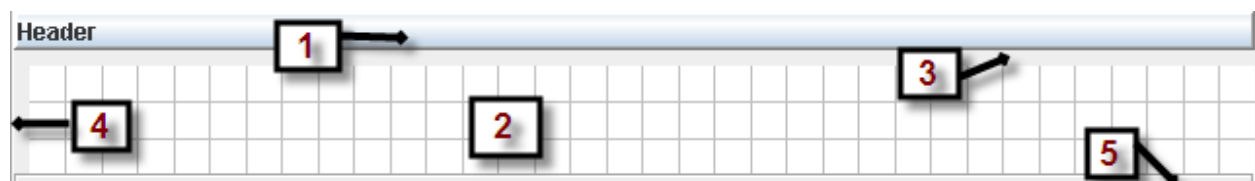
Note however, the designer will not check the validity of the XML nodes you create, that is, you can create nodes that are not part of the schema or you can create them in the wrong sequence or position.

Additionally to the tree, the panel contains the following items:

- The selection combobox which allows you to select the test case you want to work with.
- The “create a new test instance” button which will use the imported schema as template to create a new test case.
- The import test instance button to load an input XML file.
- The export test instance button to export the current XML content to a file.
- The save test instance button to save the XML test data to the database.
- The delete test instance deletes the XML file from the database.
- The status bar below the tree which contains the name of the loaded XML file.

The work panel

The work panel on the right of the window contains a sequence of areas. Each area has 5 elements:



1. The title button. Click on this button to display the properties of the area in the properties window.

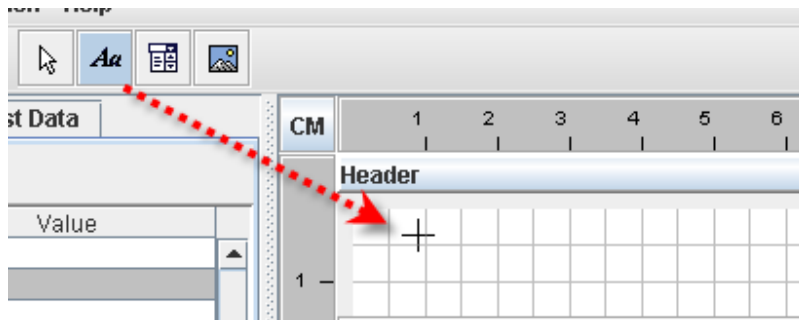
The title button contains the name of the area and the XML node element that has been assigned to the area:



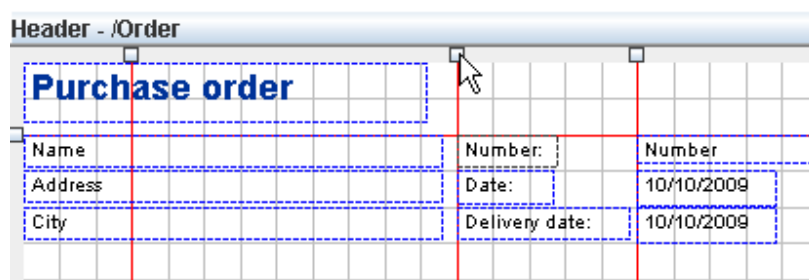
In this example you see the Detail area has been assigned to a XML element whose path is `/Order/OrderDetail/ListOfItemDetail/ItemDetail`.

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 toolbox and click on the position of the work area where you want to add the object. As an alternative, you select an element in the XML schema tree and click on the work panel, a new field for the selected XML element will be created (this option has been explain in the [XML schema tab](#) section).

In the following screenshot you can see the text field tool has been selected and the cursor shows the crosshair icon when it rolls over the work panel of one of the areas:



3. The gray line below the title button is the column markers panel. Click on the panel to add a new column to the area. The following screenshots contains an area with 4 columns. The column markers/separators can be dragged or removed by double clicking on them.



4. The row markers panel is the gray column on the left of the area. Click on the panel to add a new row to the area. Row markers can be dragged or removed in the same way as column markers.

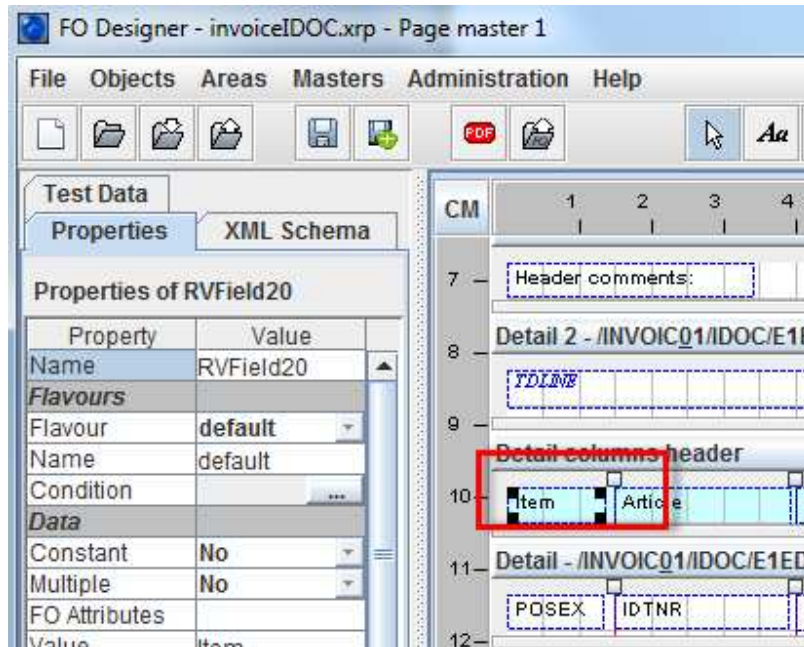
Columns are row markers are explained later in this guide.

5. The drag panel, used to resize the area. As you can see in the next screenshot, the cursor changes to the resize cursor when it rolls over the resize panel:

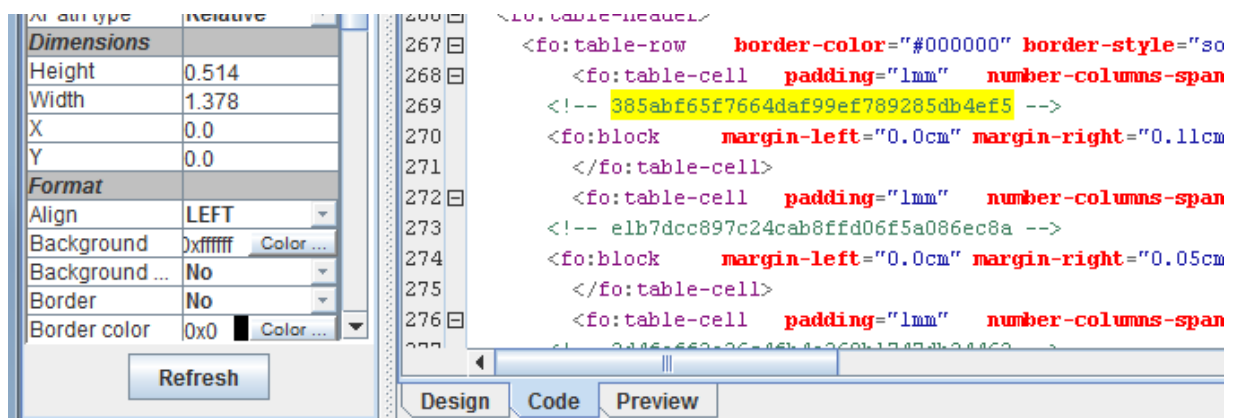


The code tab

The code tab works in the following way. You select an item in the design tab.



After clicking on the code tab, the xsl-fo code will be generated. The tab will show the position of the selected item so that you can see the generated code. It works the same way if you have no selected object, the scroll bars will be moved to the position of the currently selected area.









This feature is useful when working with the free code objects of the tool box to see where your code is placed


The toolbox bar

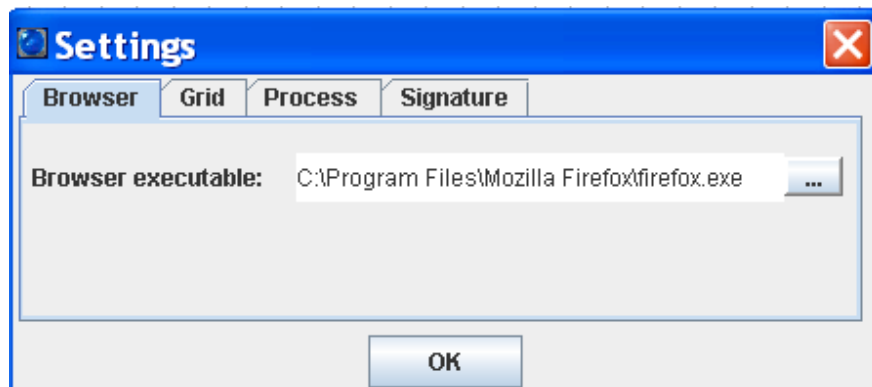
The buttons in the toolbox bar are divided in 3 groups:


1. Administration buttons

-  create new template
-  open existing template from database
-  import template created by the designer and saved as *.xrp file
-  export template to a *.xrp file
-  save template to database (overwrite current version).
-  save template to database and create new version



2. Output buttons








-  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, in this case you must first go to the main menu, administration, settings item and enter the path to the internet browser. In the case of windows it is the path to IEXPLORER.EXE (Internet Explorer) or Firefox.exe (Mozilla Firefox), as you see in the following screenshot:



-  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.



3. Tools buttons:

-  selection tool (default one), used to select objects in the template.
-  text field. Click on this button to add a field to the template, the cursor will change to the crosshair cursor, then click on the position where you want to add the field.

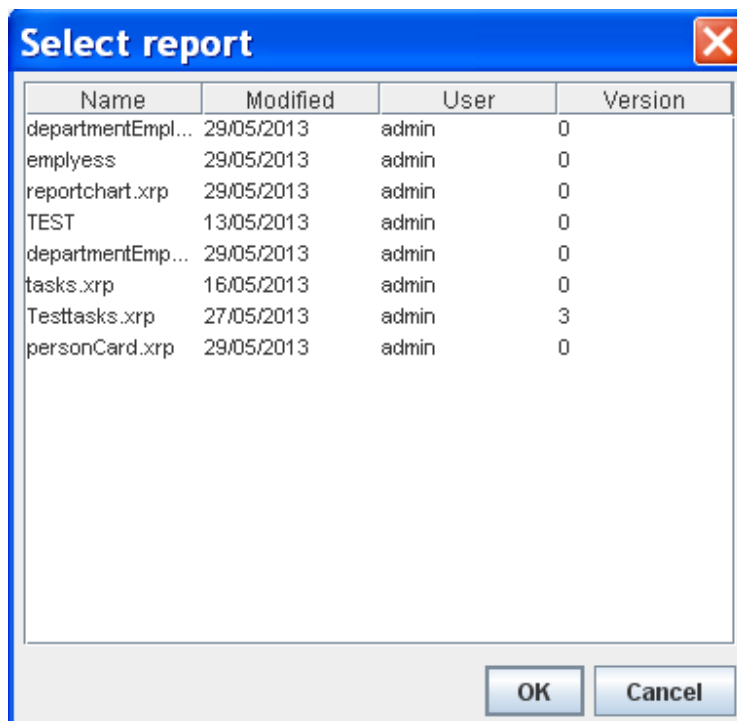
-  combobox object. Select this tool to add a combobox. A combobox is a list of pairs key/value. The value in the input XML document will be used as key to find the value in the list. The value will be output in the PDF document.
-  image object. Select this tool to add an image to the template.
-  line object. Select this tool to add an horizontal line.
-  barcode object. Select this tool to add a 1D or 2D barcode. Note the designer includes the evaluation version of the barcoding components. These require a separate license if you are going to create barcodes.
-  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 a separate license if you are going to create charts.
-  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 see *examples\apex\tasks.xrp*.
-  Free code object. Use this object to place you own XSL-FO in the output of the report.

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 3 for the Testtasks.xrp template.



Once you have opened the template you can also retrieve an older template version with this menu item:



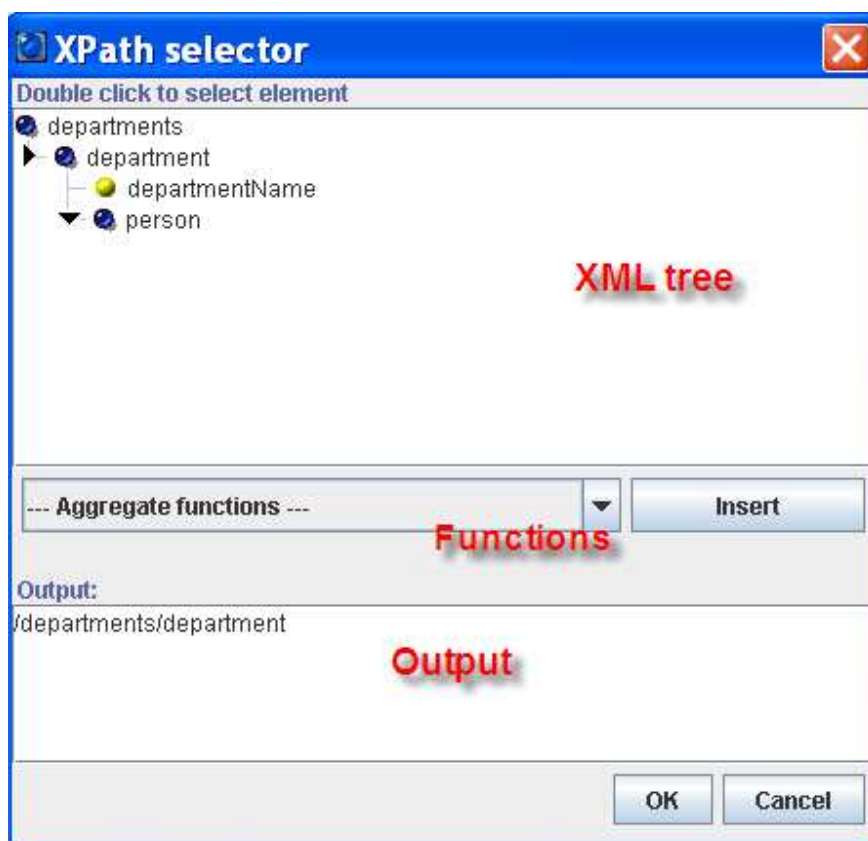
The XPath editor

The XPath editor is used for selecting XML nodes and 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.

The Editor has 3 parts:

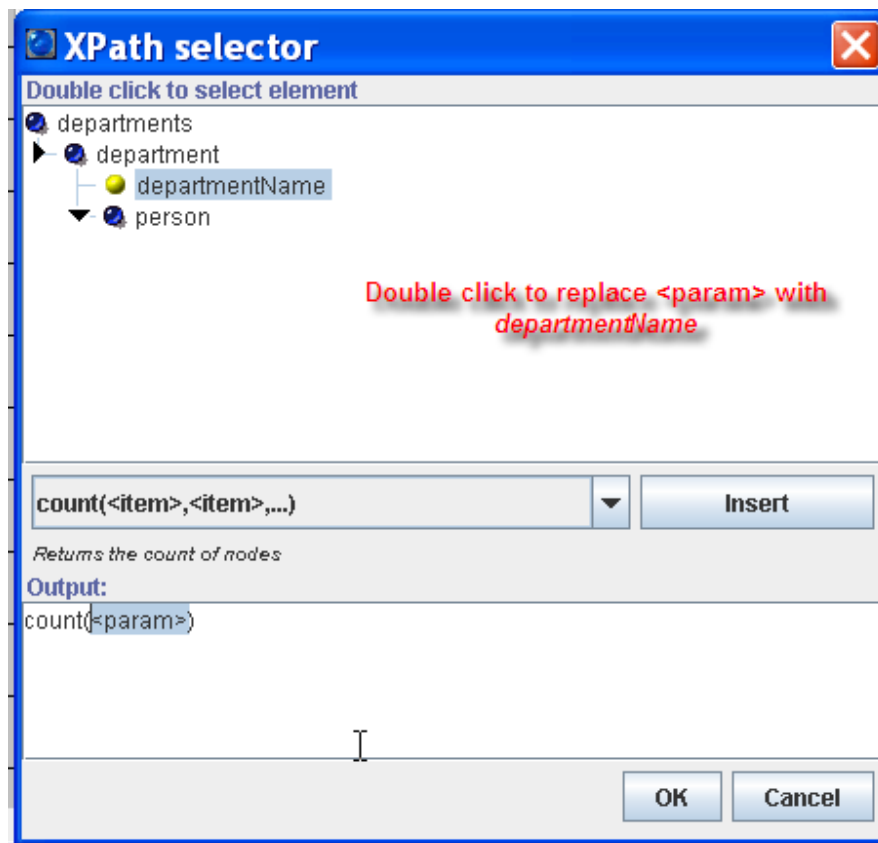
1. on the top you can select the XML node, double click on it to select it and add it to the output.
2. In the middle of the window you can select one of the available functions. The function will be added to the output only if you click the *Insert* button.

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 *j4l:ext*. If you use any of these *j4l:ext* 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](#).



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.



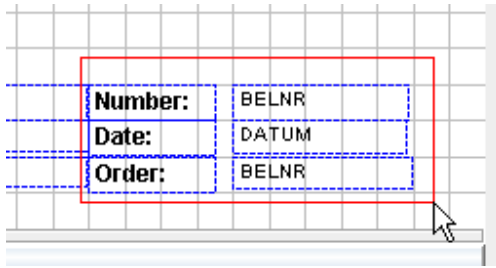
Working with the user interface

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

Selecting objects

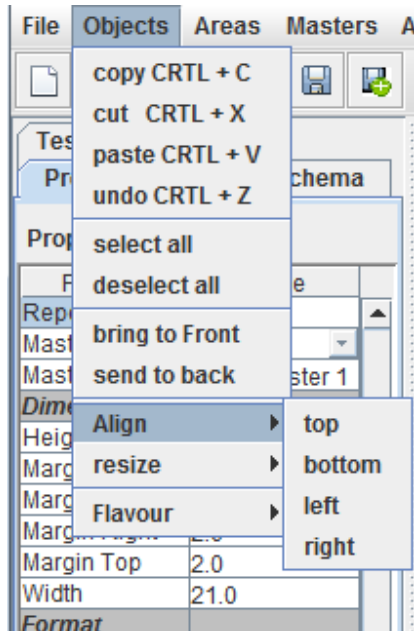
Objects in the areas can be selected in 3 ways:

- 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.
- Multiple object selection: click on the area's background and drag the cursor, a red rectangle will show the selected area. After you release the mouse button, all objects in the area will be selected:



Operation on objects

The objects item in the main menu lists a set of operations you can perform on the object/s you have selected:

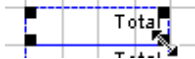


the same operations area available in the popup menu (mouse's right button).

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 in the menu the following operations can be performed:

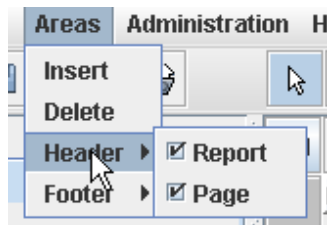
- copy: *control key + C*
- cut: *control key + X*
- paste: *control key + V*
- undo: *control key + Z*. This undoes the last change in objects in the current area. You can undo the last 10 changes. Note this refers to changes to the current area's objects, this function will not work for changes in the report structure, like adding a new area.

- select next object in area: *tab or right key*.
- select previous object in area: *left key*.
- delete: *del* key
- drag/move: select one or several objects and drag them by moving the mouse while keeping the mouse's button pressed.
- resize: place the cursor on the right bottom corner of the object and drag it.



Operations on areas

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



- Insert: select an area in the template (by clicking on the areas name/title button). Select then the *insert* option in the *areas* menu. A new detail area (and the associated header area) will be added.
- Delete: select a detail area and select the *delete* option in the *areas* menu.
- The template header and footer and the page header and footer areas cannot be deleted but they can be hidden in case you do not need them by selecting the corresponding item in the *areas* submenu.

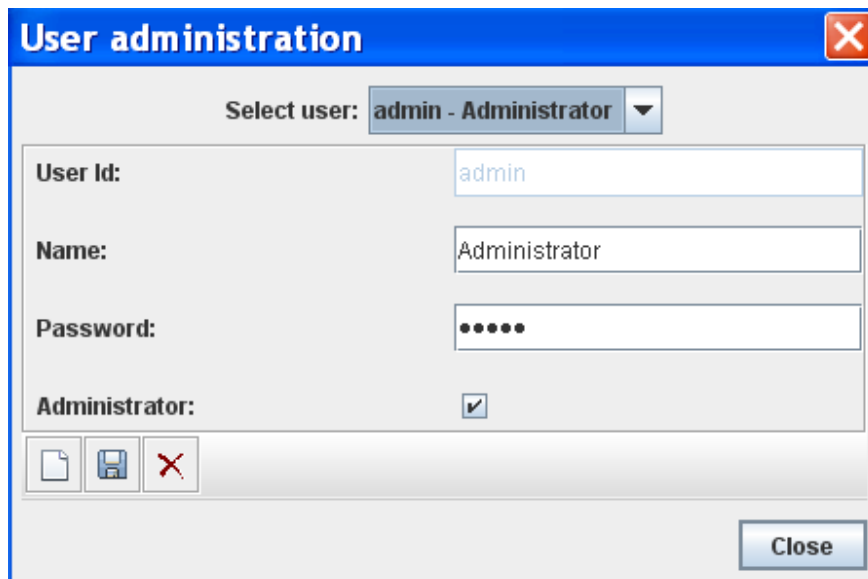
The settings dialog

In the administration menu you can set the global settings of the FO Designer. These are:

- Browser tab
 - the executable file for displaying generated PDF files.
- Grid tab
 - grid setup
- Process tab
 - XML namespace processing setup. See also this [FAQ](#).
 - Oracle APEX encoding setup: select this option if you are working with APEX. See also [this FAQ](#). Do not use however this option for the Oracle APEX Listener as print server.
- Signature tab. See [this section](#).

User administrator

The user administration is useful in a team environment where you need to keep track of who made changes in the templates and on which date. The user administration window is located in the *Administration* --> Users menu item. It can only be opened by users with the administrator permission.



The image shows a 'User administration' dialog box. At the top, there is a blue title bar with the text 'User administration' and a red close button. Below the title bar, there is a 'Select user:' label followed by a dropdown menu showing 'admin - Administrator'. The main area contains four labeled input fields: 'User Id:' with the value 'admin', 'Name:' with the value 'Administrator', 'Password:' with six dots, and 'Administrator:' with a checked checkbox. At the bottom left, there are three icons: a document, a floppy disk, and a red 'X'. At the bottom right, there is a 'Close' button.

3. 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.

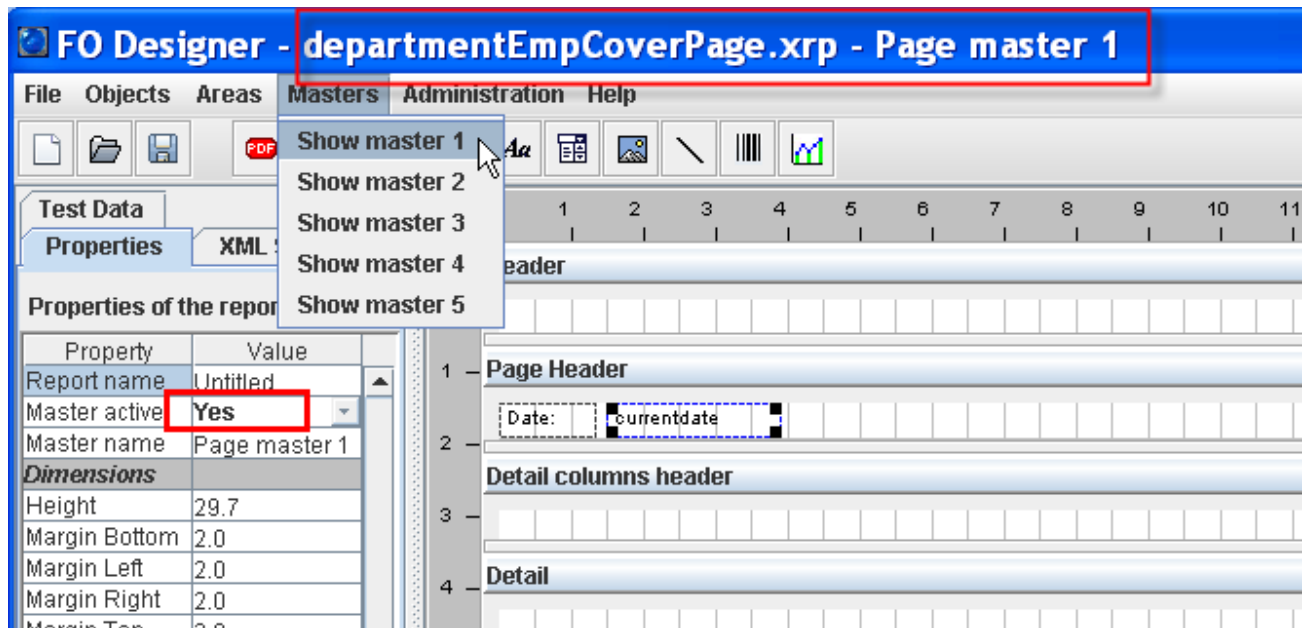
Page masters

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.

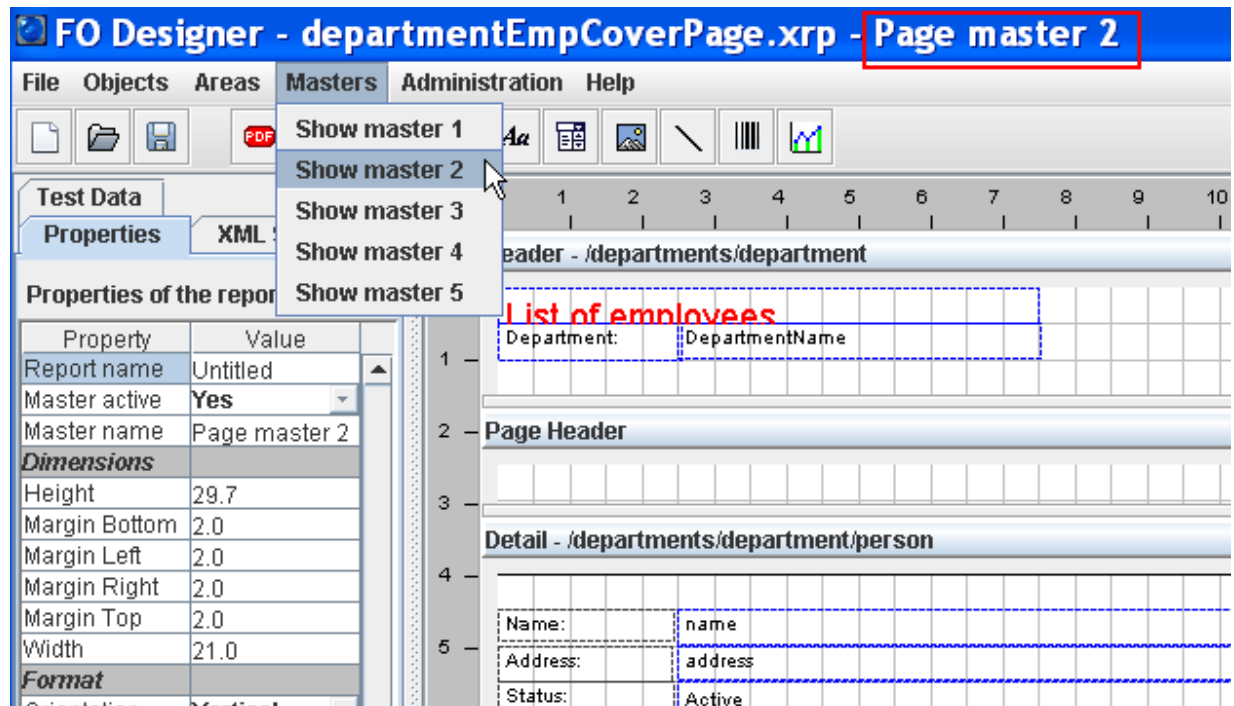
The example located in *examples\employees\departmentEmpCoverPage.xrp* contains such a template.

Each template can have up to 5 page masters that can be selected from the *Masters* menu. **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.

Note also the name of the currently selected page master will be displayed on the window title bar.

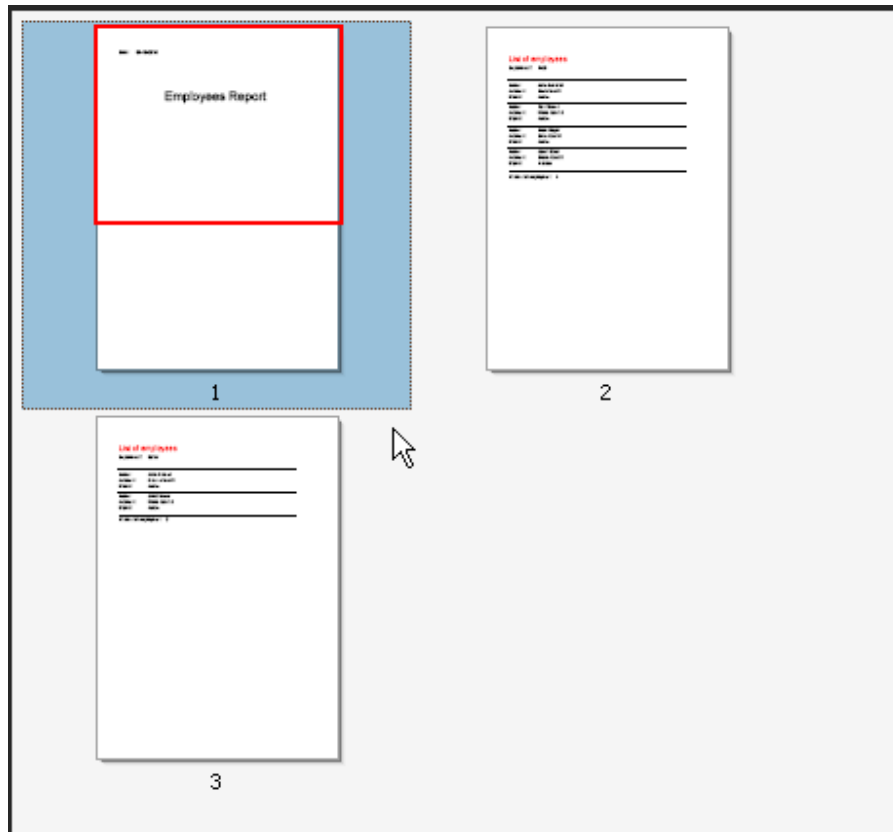


In this example the first master contains just the title of the report as a cover page, while the master 2 contains the layout of the pages that contains the employee data.



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.



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.

The screenshot shows a purchase order form with the following structure:

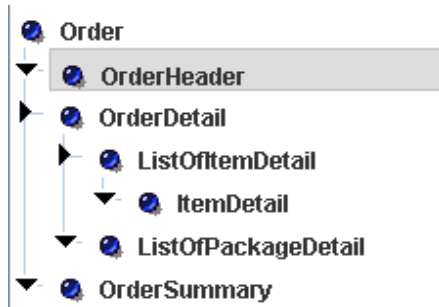
- 1**: A large empty rectangular box at the top of the form.
- 2**: A section containing the company name "ABC Enterprises", address "ABC Road", and "Alpine". To the right, it lists "Number: 4500005693", "Date: 03/02/2001", and "Delivery date: 07/02/2001". Further right is a yellow oval labeled "LOGO".
- 3**: A table header with columns: "Number", "Article", "Description", "Price", "Quantity", "Tax", and "Amount".
- 4**: Two data rows in the table:

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
- 5**: A summary section at the bottom right showing "Tax: 337.6" and "Total: 2447.6".

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:

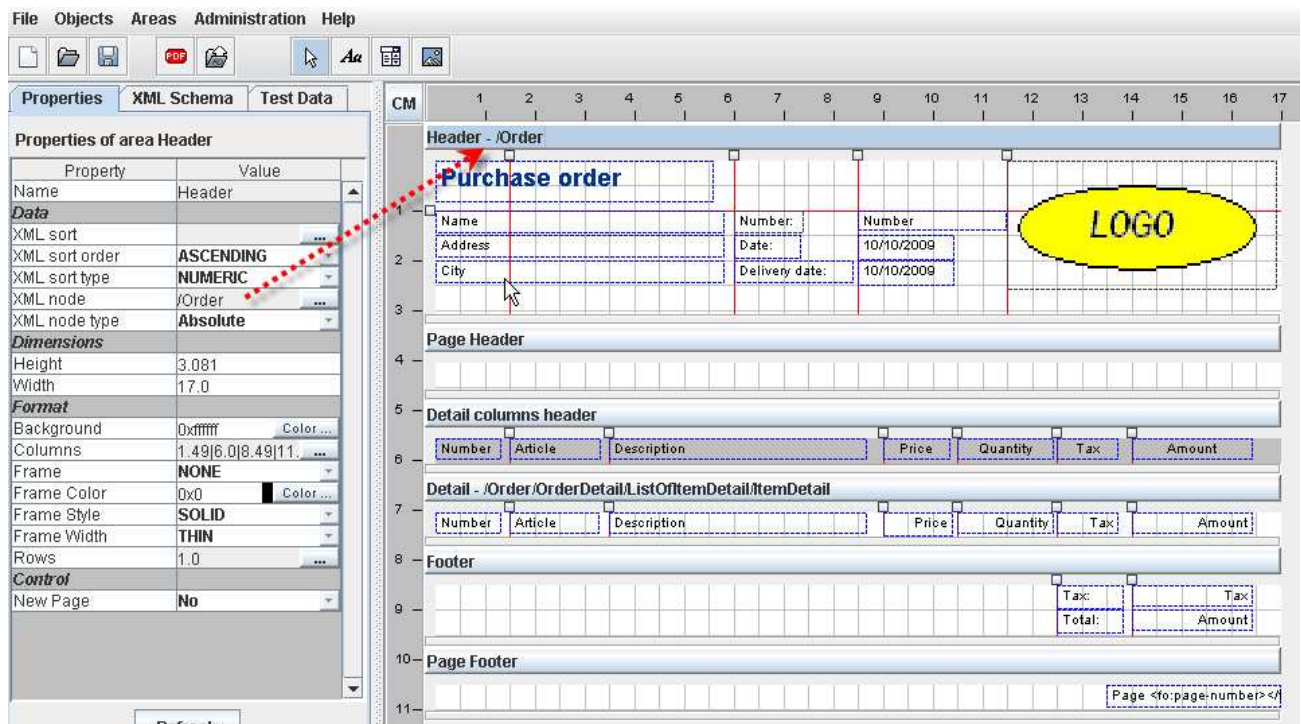
- One template **header and footer**. Each PDF file will normally contain only one document header and one document footer, however it is possible to have several headers in one PDF file, for example if you are printing several invoices in one PDF file.
- One **page header and footer**. These areas are printed in each page.
- **detail** areas (optional). Detail areas have 2 key properties:
 - the **super area** which points to the area it depends on. Each time an area is generated (added to the PDF file), the dependent areas will be generated. All detail areas must have a super area otherwise it will never be generated. The template Header is the first area generated and the one first level detail areas must depend on.
 - 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).

The delivery includes a file called *order.rxp* 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 an 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.



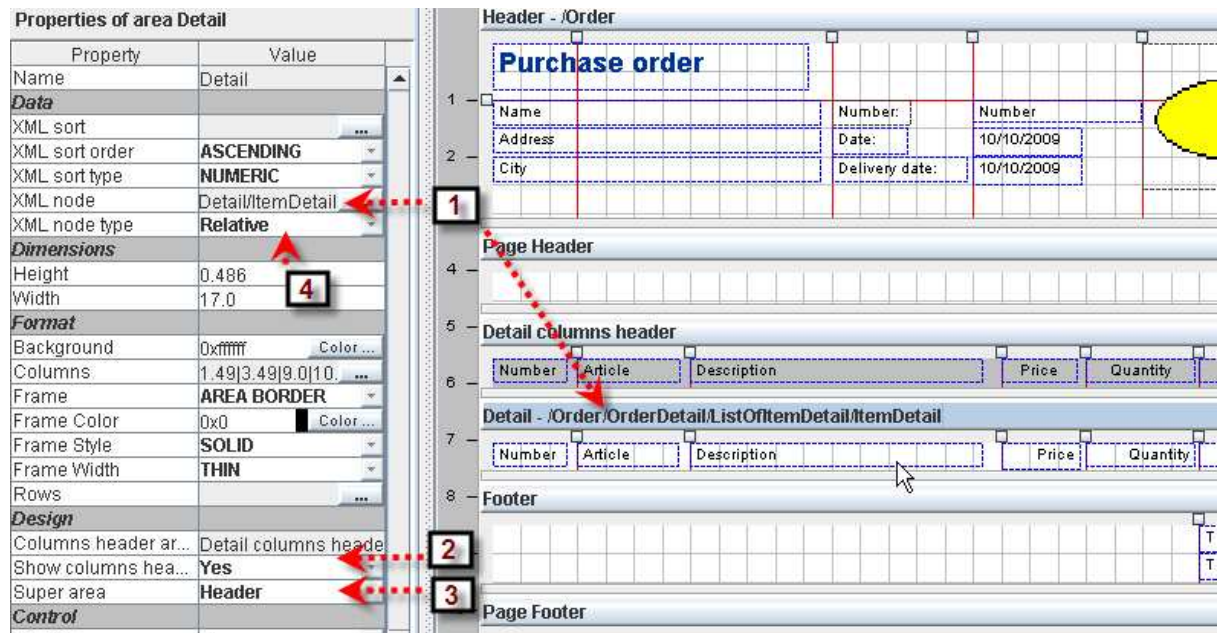
It contains the 6 areas we mentioned above:

- 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 returned 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. You can see in the following screenshot (see arrow number 3) the *super area* of the detail area is the header.

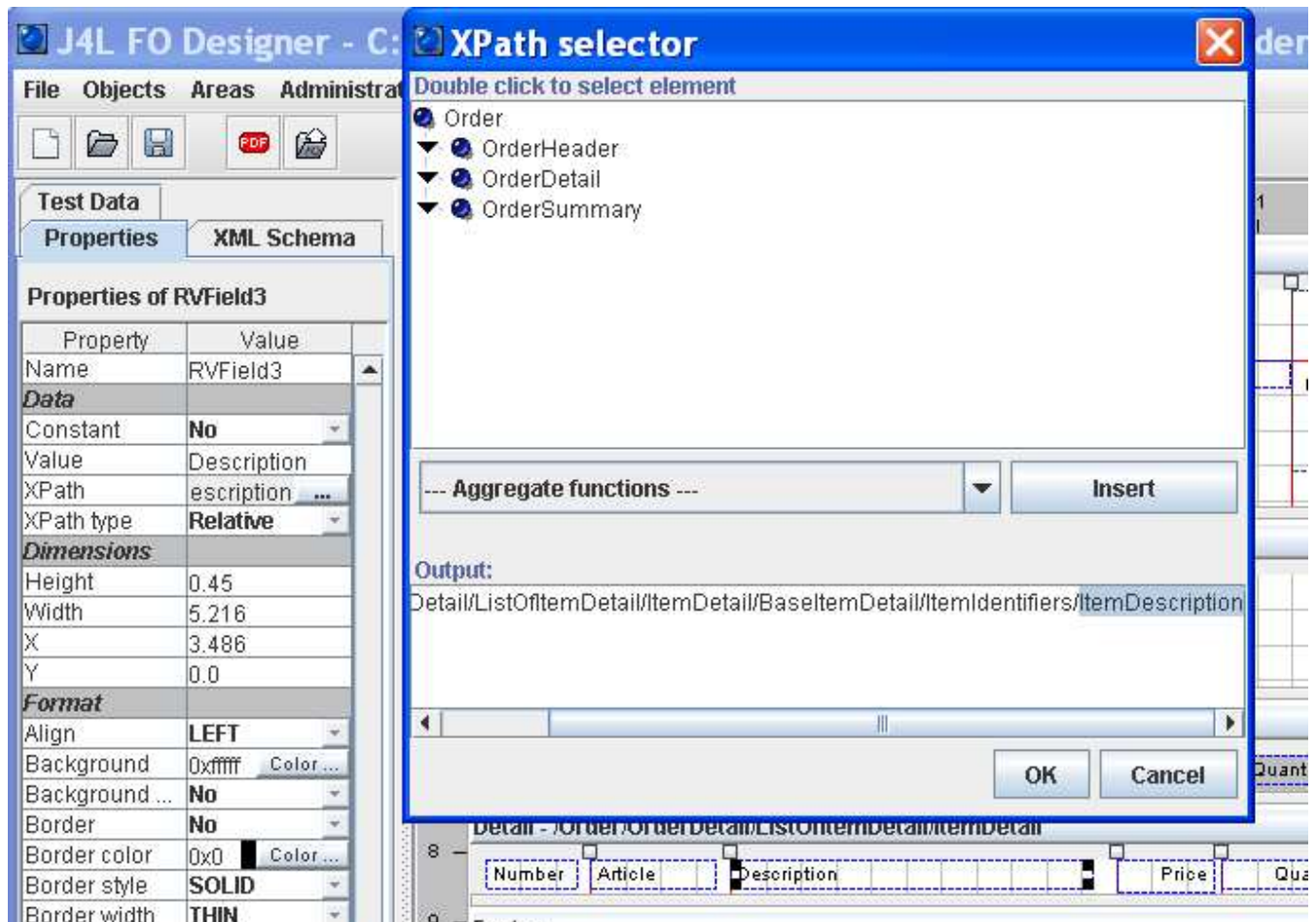


so we know the detail area will be generated after the header area, but how often? this depends on the ***XML node*** property (see arrow 1 in the screenshot). In this example we will create one repetition (line of the detail area) for each `ItemDetail` node in the input XML file. In other words, we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

Last, as you see in the ***Show columns header*** property (arrow number 2), there is a header area 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* (see arrow 4). 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.

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. The path to the image file can be a relative path to the working directory or an absolute path.

Format	
Back. color	0xffff Color ...
Back. opaque	No ▼
Background image	images/logo.gif ...
Background h. align	CENTER ▼
Background v. align	CENTER ▼

Background area

The background area contains objects that will be placed on each page at a fixed location. The background area can be displayed with the following flag:



If you use the background area, make sure regular areas have the opaque property set to *no*, otherwise they will overlap the background area.

You can see an example of a background area in *examples\apex\tasks_background.xrp*.

Columns and rows markers

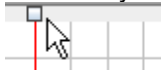
FO designer will always internally create a XSL-FO table (similar to a HTML table) and place each field or image in one cell in the table. In most cases this is done automatically however there are some situations where you explicitly have to define the columns and rows of the area as explained below.

Each area has a column marker panel (grey panel on the top) and a row marker panel (grey panel on the left), where you can place the column and row separators. The separator:

- Inserted by clicking on the marker panel

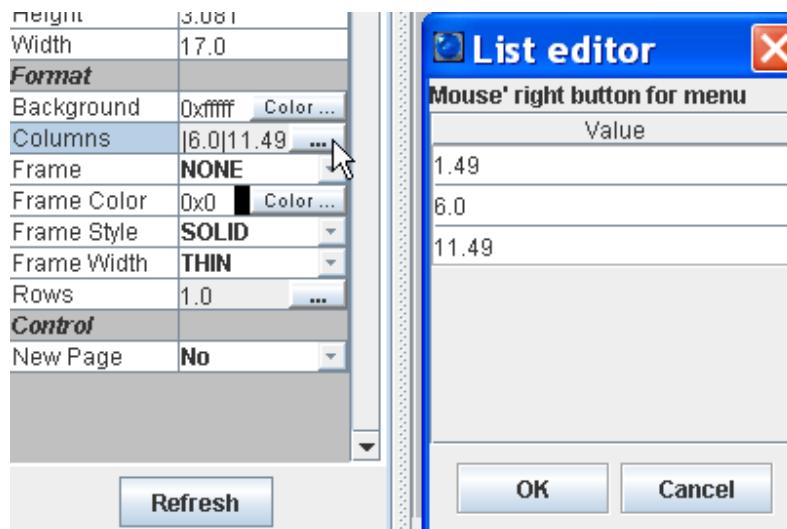


- deleted by double clicking on the marker



- moved by dragging panel

Additionally you can enter the position of the marker manually in the areas's properties. This example shows 3 column separators at positions 1.49, 6.0 and 11.49



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

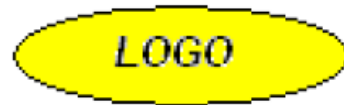
1. if you have an object that expands several rows (for example an image)
2. if you want to paint a frame or line around the column, row or cell (see the *frame* properties in the area's properties)
3. in case the automatic layout of fields does not produce the expected output, you may need to define 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 image shows a purchase order template layout on a grid. On the left, there is a form with the title 'Purchase order' in a blue box. Below the title are three rows of input fields: 'Name', 'Address', and 'City'. To the right of these fields are three more rows of input fields: 'Number:', 'Date:', and 'Delivery date:'. Each of these three rows has a corresponding input field to its right. On the far right, there is a large yellow oval containing the word 'LOGO' in black capital letters. The entire layout is set against a light gray grid background.

the output PDF is

Purchase order



ABC Enterprises	Number:	4500005693
ABC Road	Date:	03/02/2001
Alpine	Delivery date:	07/02/2001

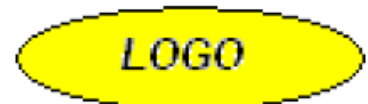
the problem here is the image and the "Purchase order" text are placed on the top which is correct but all other fields and moved further down in the page. If you however define the rows and columns yourself like this:

Purchase order			
Name	Number:	Number	
Address	Date:	10/10/2009	
City	Delivery date:	10/10/2009	

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.

Purchase order

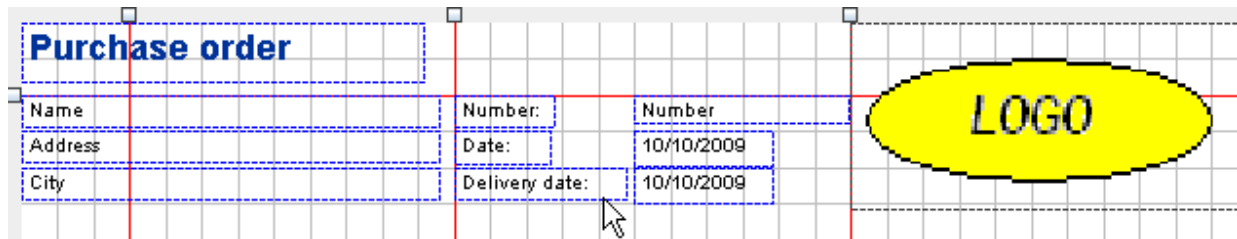
ABC Enterprises	Number:	4500005693
ABC Road	Date:	03/02/2001
Alpine	Delivery date:	07/02/2001



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 bewteen the *Number:* label and the *Number* field like this:

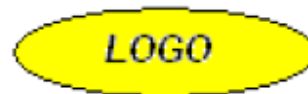


You have now 2 objects at the same height and the output will be incorrect (see below) because the designer is unable to put 2 object at the same Y position unless there is a column separator:

Purchase order

ABC Enterprises
ABC Road
Alpine

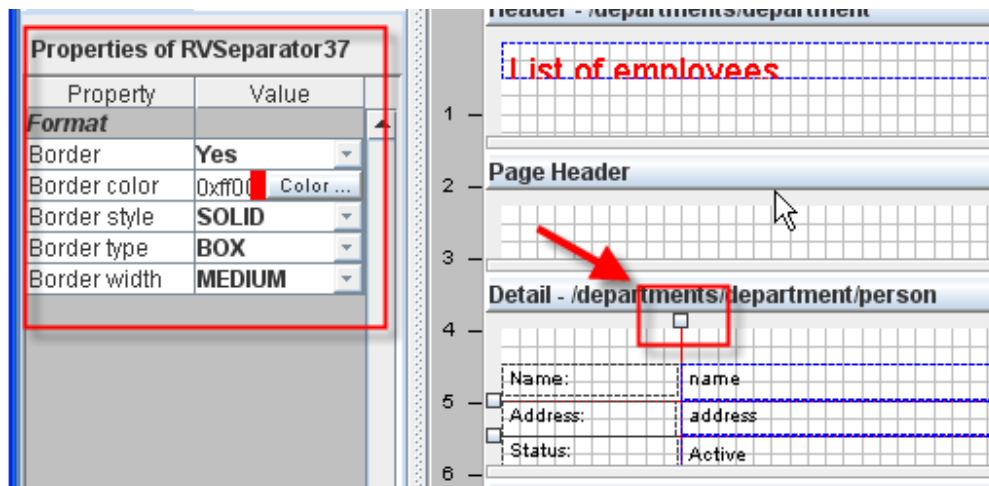
Number: 4500005693
Date: 03/02/2001
Delivery date: 07/02/2001



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 main menu , file, properties to view the template properties

- **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)
- **Orientation:** horizontal for landscape and vertical for Portrait.
- **Margins:** define the margins of the page.
- **Name:** name of the template, used for information purposes only.

Properties of the areas

Click on the area's title button to display the properties of an area:

- **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/BaselItemDetail/ItemIdentifiers/PartNumbers/BuyerPartNumber/PartNum/PartID*.
- **XML Sort type:** whether the sorting elements are numeric or alphanumeric values.
- **XML Sort order:** ascending or descending.
- **XML Group by:** used for grouping values (see [additional section of this topic](#))
- **Background:** background color of the area
- **Columns:** see [Columns and rows](#) section.
- **Frame:** select a value different from NONE to enable the area's frame. Possible values are:
 - AREA BORDER: paint only the border of the area
 - ONLY ROWS: paint only horizontal lines to separate area repetitions and rows within the area.
 - ONLY COLUMNS: paint only vertical lines to separate columns in the area (see column markers).
 - GRID: all above will be painted.
- **Frame color:** select the color with the color picker (*color...* button) or enter the hexadecimal RGB value of the color in the field (the format is 0xRRGGBB, where RR is the red component, GG the green component and BB the blue component).
- **Frame style:** select one of the styles

- **Frame width:** select THIN, MEDIUM or THICK.
- **Rows:** see [Columns and rows](#) section.
- **Columns header area:** Each detail area may have a columns header area, this is used to enter the name of the columns in case your detail area has a table-like layout. This field is a read only field and contains the name of the columns header area.
- **Show columns header area:** Whether the columns header area will be shown and used or not.
- **Super area:** the area this area depends on. For a detail area to be generated in must depend directly or indirectly from the Header area.
- **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.

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 technic 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.
- **Align:** text alignment
- **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.
- **Font:** currently only the built-in fonts are supported (SansSerif, Courier, Times Roman and Symbol). Support for TTF will be added in the next version.
- **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
 - d: 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 combobox

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 will 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"*.
- **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.

- **Table contents:** If set to yes, the line will be used to connect 2 fields (like in a contents table). See example *examples\apex\tasks_background.xrp*.
- **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

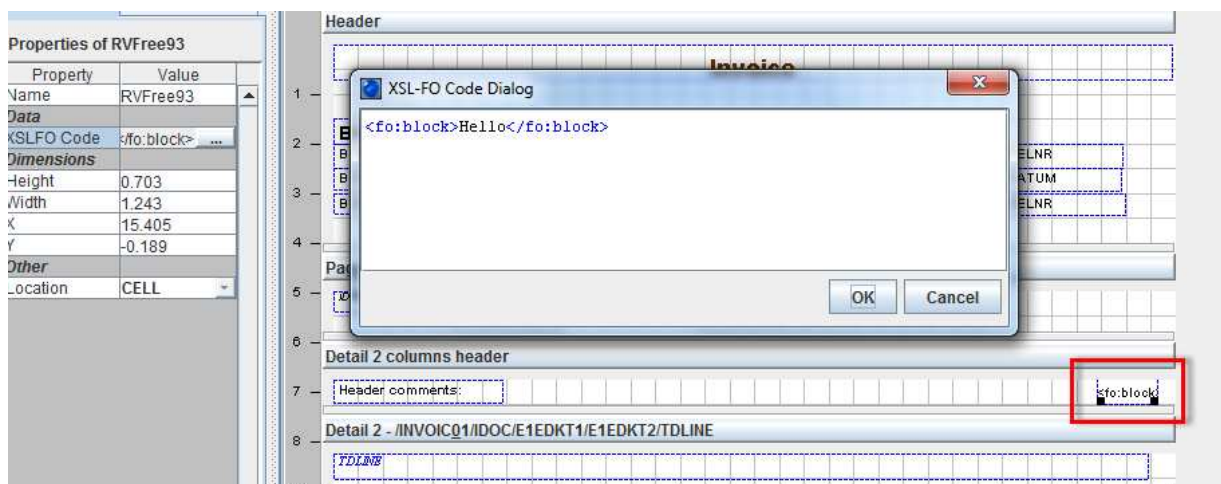
The free code objects are required when you need to add code to the xsl-fo output. This should be done only if you are familiar with the xsl-fo language. If the code is not correct the PDF generation will fail.

The 2 main properties are:

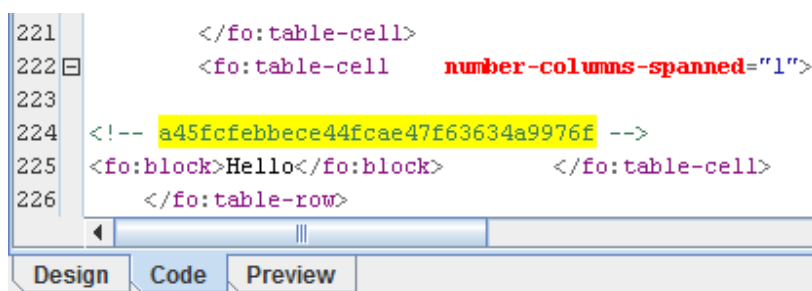
- **XSL-FO Code:** use the editor to add code. Note you have to use the prefix “fo:” for fo elements and the prefix “xsl:” for xsl elements.
- **Location:** of the code. The code can be handled as attributes of the current cell/row/table or as standalone code before the area or inside the current cell.

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 :

ABC industries
London street 8
Manchester

Factory 1
Business park 28
London

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

Header comments:

This order was placed by Mr. Roger

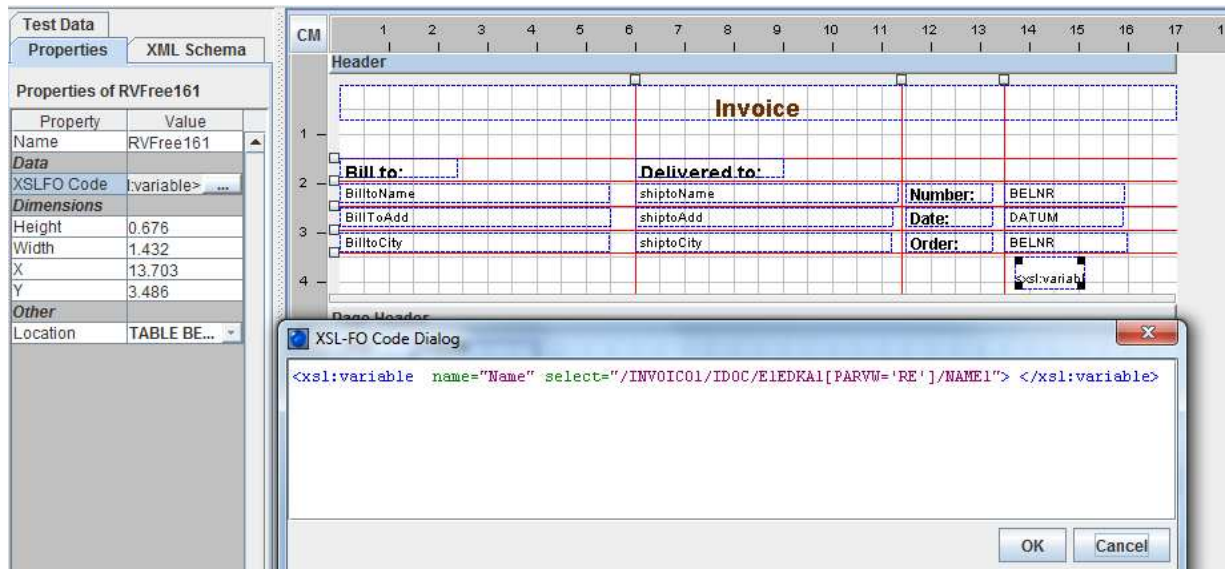
Hello

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

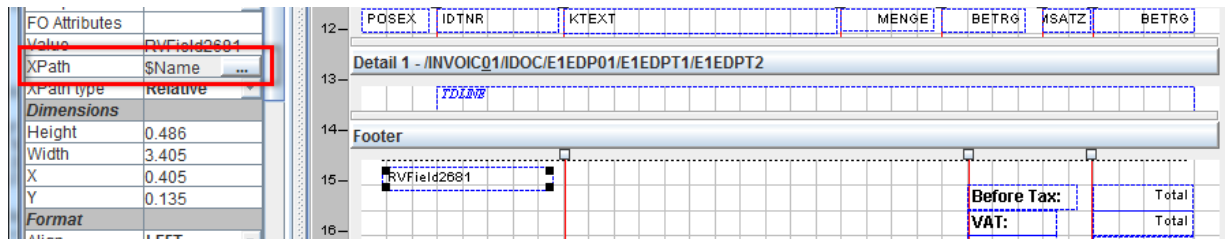
Before
Tax: 438.04

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.



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



The output shows then the variable value:

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
<i>This is a comment to the delivery of item 1</i>						
000020	Mat2	Container 45 inches	17.00	18.45	0	313.65
<i>This is a comment to the delivery of item 2</i>						
<i>This is another comment for item 2</i>						

ABC industries

Before

Tax:

438.04

VAT:

4. 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.
 - o The body of the request contains the XML document to be converted.
 - o 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.

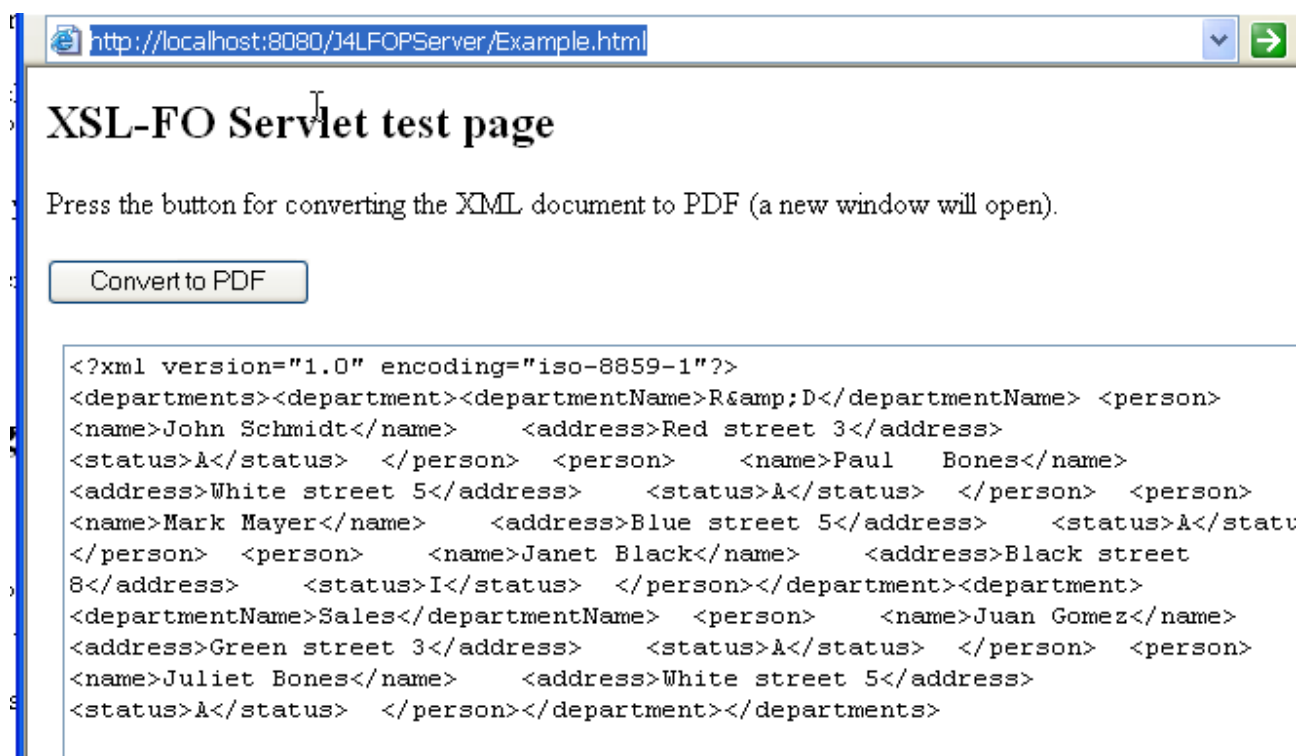
3. the servlet returns the PDF file

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

- departmentEmployees.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 opened:

<http://localhost:8080/J4LFOPServer/servlet?TEMPLATE=departmentEmployees.fo&ENCODING=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 *examples\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 approximately:

```
// 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
FOPProcessor processor=new FOPProcessor();
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](#).

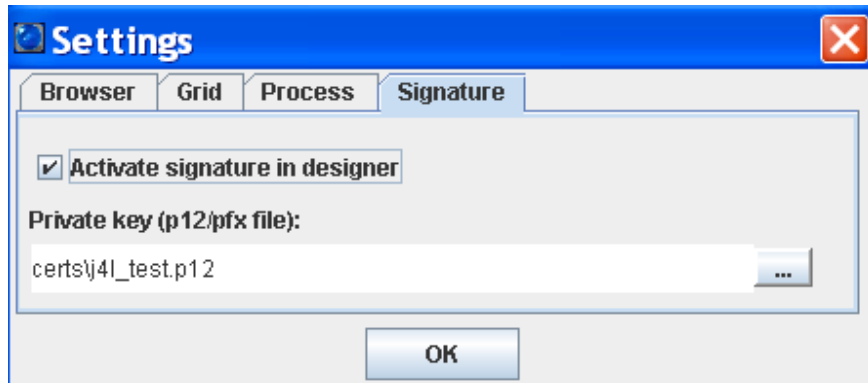
Adding a digital signature to the PDF file

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](#) or [SAP PI](#)) to learn how to activate the signature.

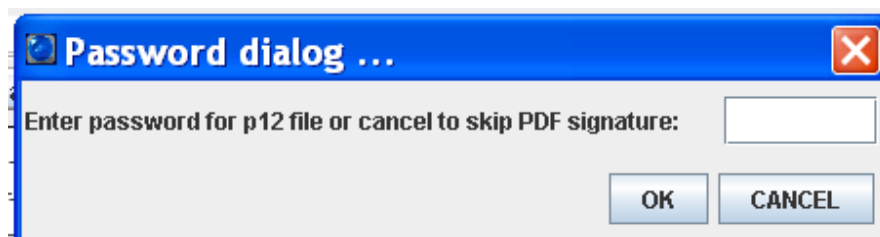
The rest of this section describes how to test the digital signature in the designer and view it in Acrobat reader.

In order to activate the digital signature you use the signature tab in the settings window (in the administration menu of the designer). 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.



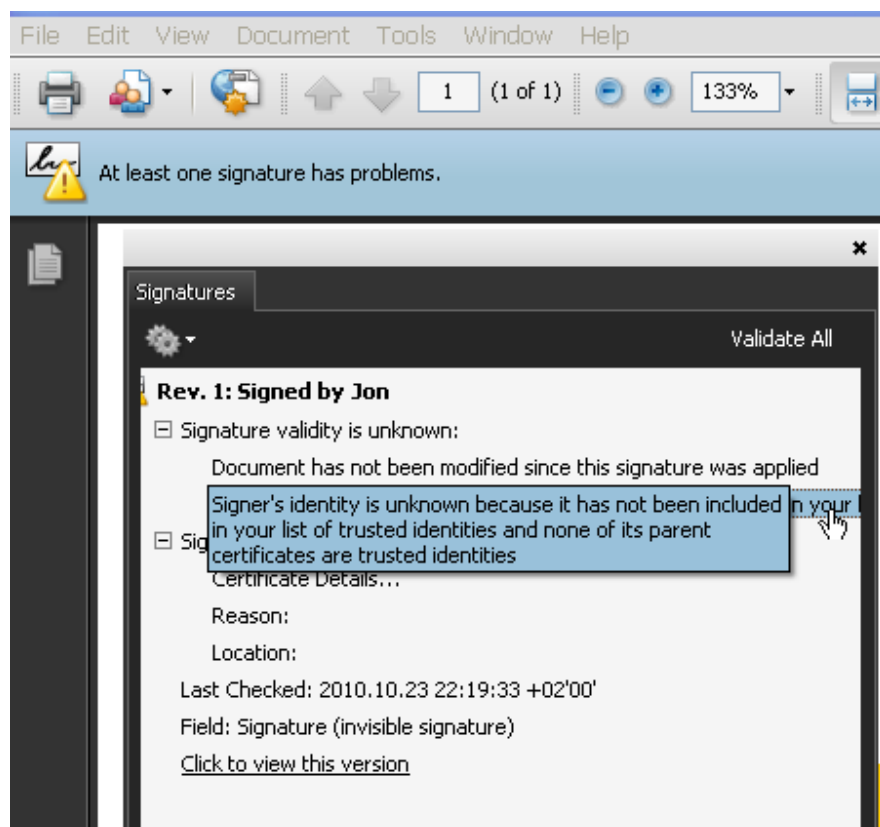
If you have activated the signature in the settings window, each time you generate a PDF file with the PDF button, the designer will ask for the password of the p12 file. The password of the test file we provide is **test**.

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

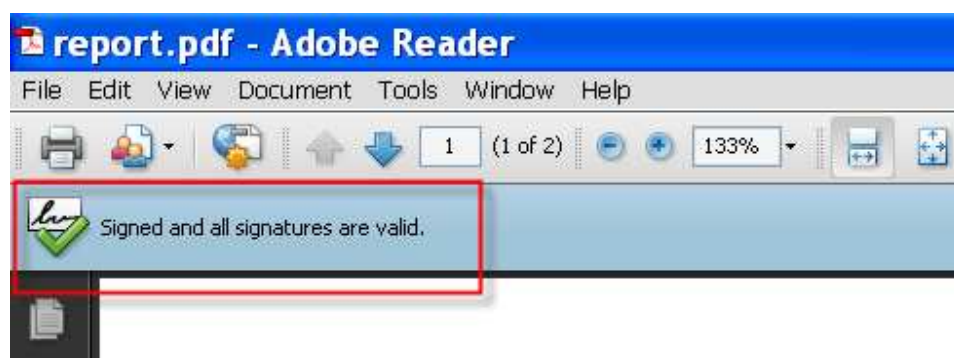


Once you open the created file, you will notice Acrobat reader warns you with the text "the signature has problems". The reason for this is, Acrobat does not know the certificate authority (CA) we have used for creating the test certificate. Looking at the signature closely in the signatures panel you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

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

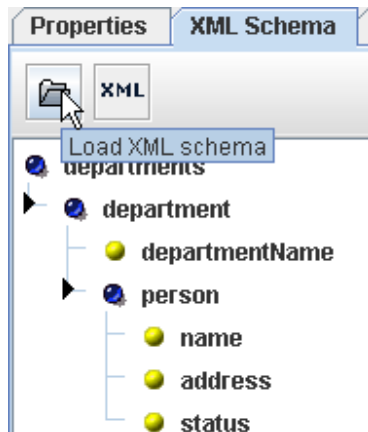


5. Learn by doing: tutorial

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.

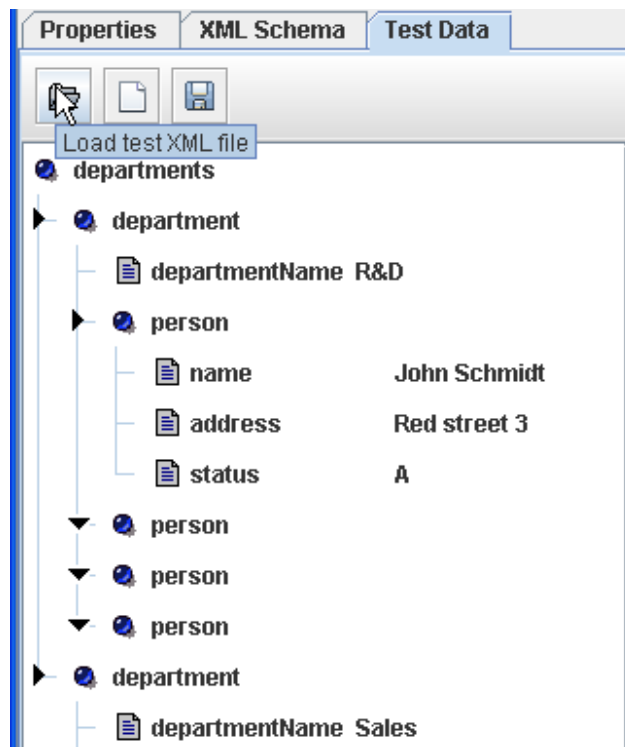
We will start running the FO Designer and we get an empty template.

1. First we load the XSD file examples\employees\departmentEmployees.xsd. The result will be:

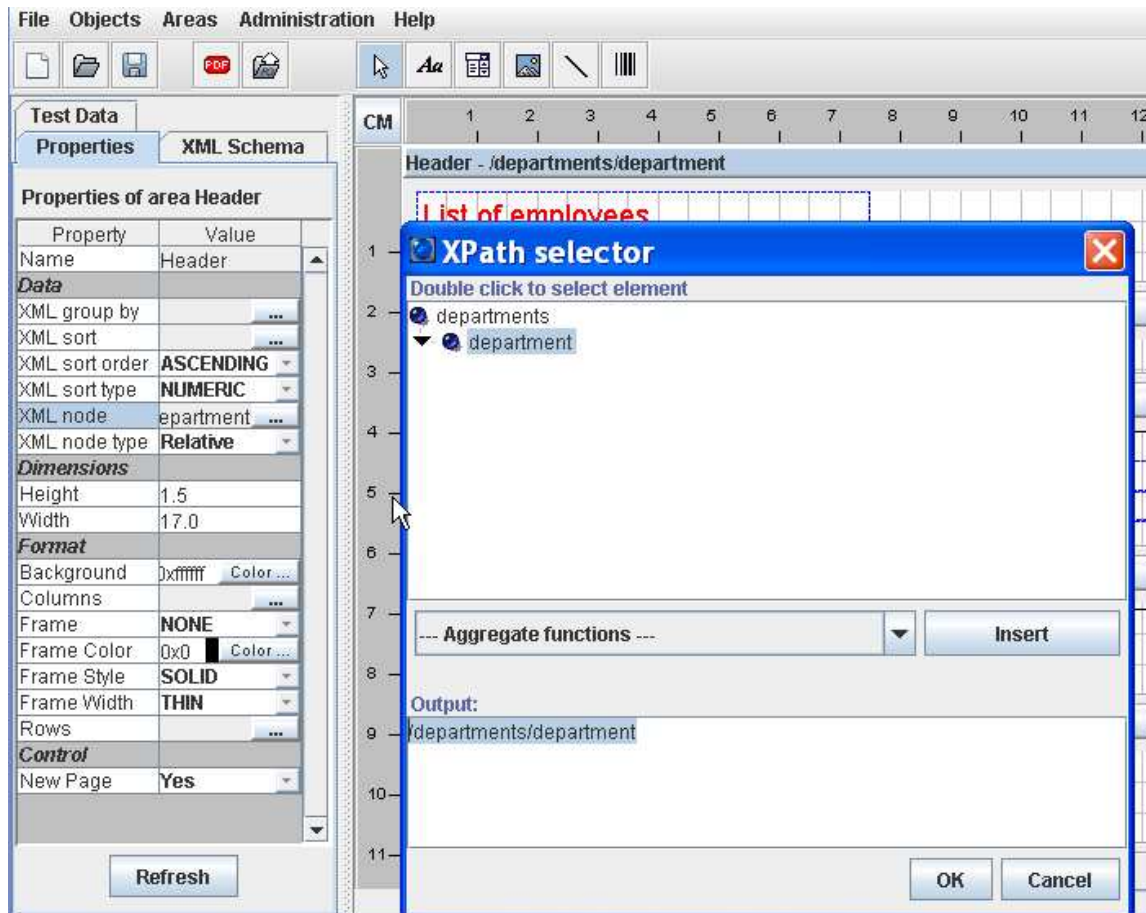


as an alternative (if we do not have a XSD file), we can click on the *XML button* and load the XML document examples\employees\departmentEmployees.xml. This would read the XML document and list of existing XML nodes, as a result it would create a kind of "virtual" schema.

2. As second step we load the XML test file examples\employees\departmentEmployees.xml. As you can see this file contains 2 departments:

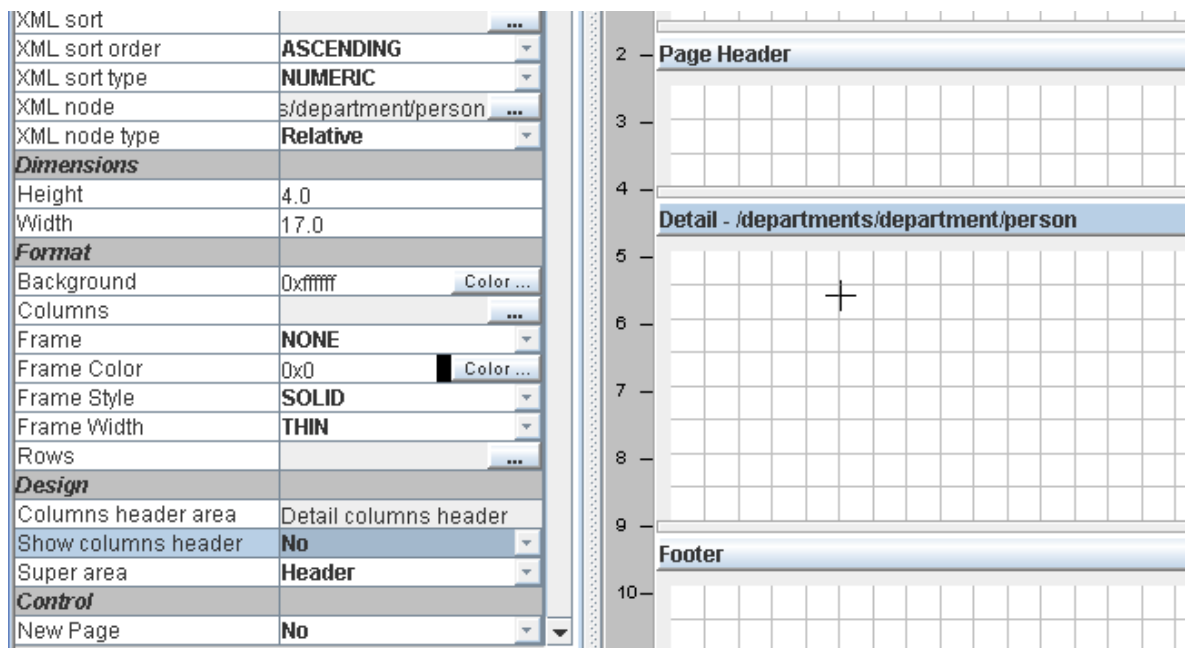


3. Our template will use 2 areas, the header area for the name of the department and the detail area for the employees. So first we will tell FO Designer we want to have a header area for each department in the XML document. So we click on the header area title button to display the properties of the area and we set *XML node* to */departments/department*.




furthermore we set the **New page** property to **true** because we want to have a new page for each department.

- 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 *Show Columns area* property to *No* because we will not use that area.

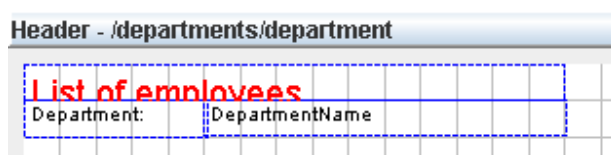
5. We will now add some fields to the header area. Click on the  button and place the following fields in the header area:



the only properties you must change are:


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

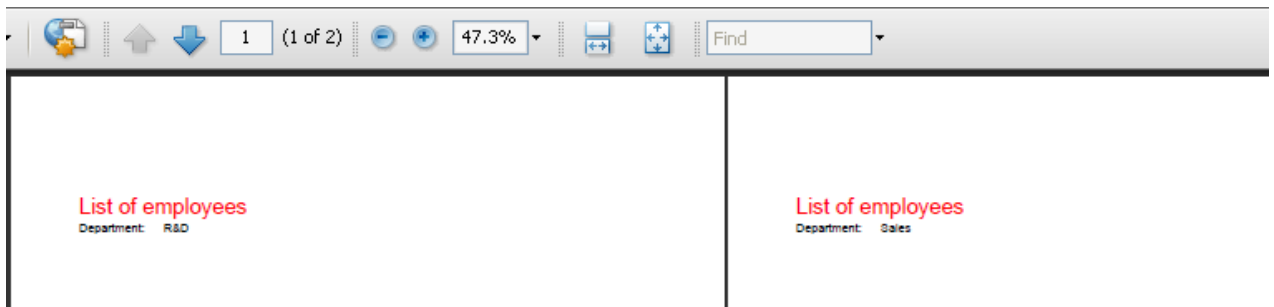
6. next you add 2 more fields



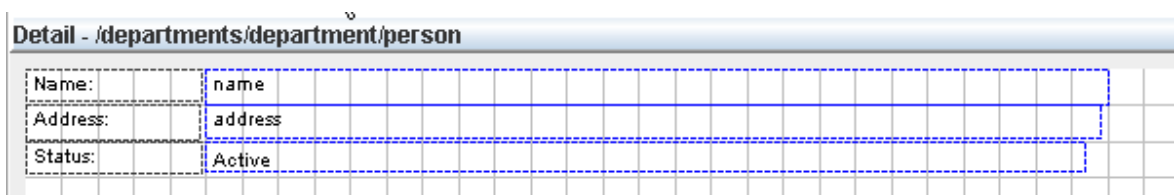
the properties you have to set are:

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

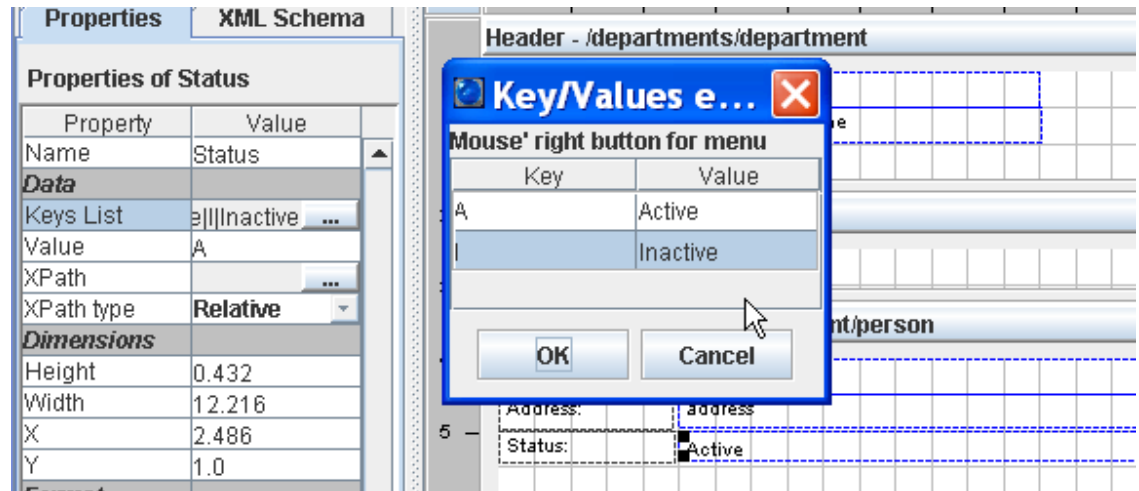
- Click on the field tool  button and click on the area to place the field.
- change the constant property to true and the value property to the value you see in the screenshot (Name:, Address: and Status:)

for the first 2 variable fields (blue border) you proceed like this:


- Locate in the XML schema tree the `/departments/department/person/name` node, click on it and then click on the area to place the field.
- Locate in the XML schema tree the `/departments/department/person/address` node, click on it and then click on the area to place the field.

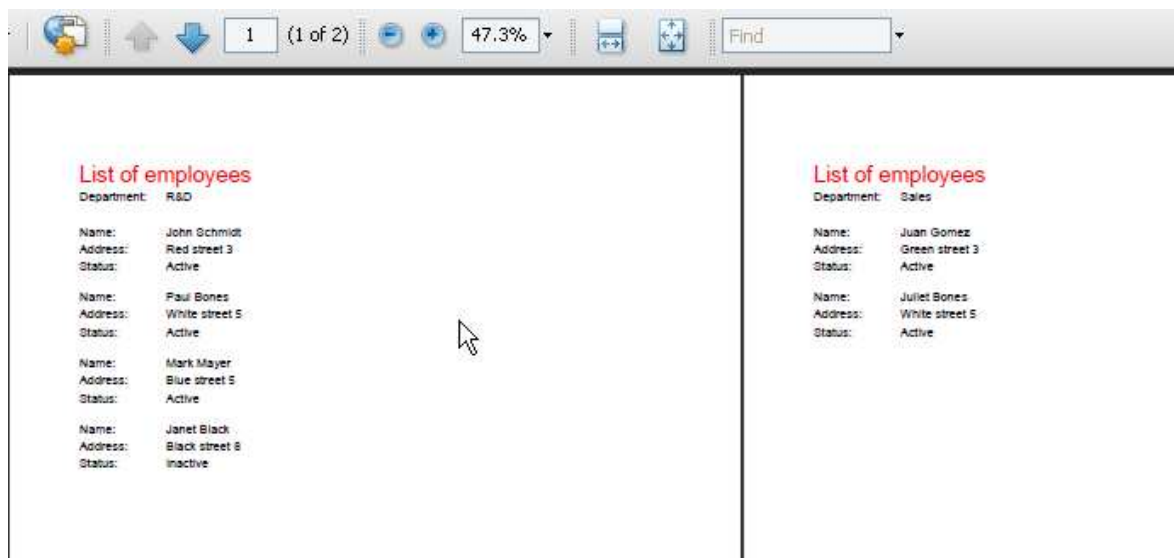
the third variable field is actually a combobox and must be defined like this:

- Click on the combobox tool button  and click on the area to place the field
- Set the XPath to `/departments/department/person/status`
- The Keys list must be defined as in this screenshot:

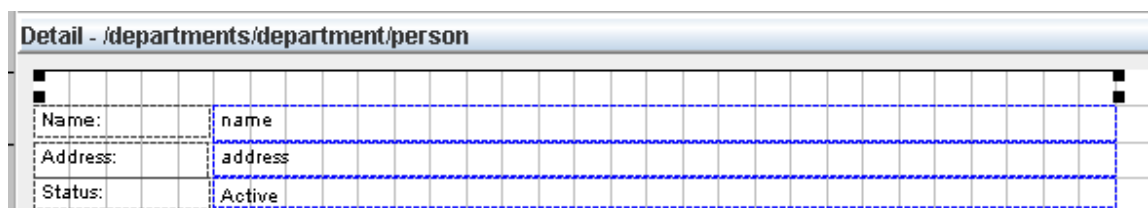


in this way the values A and I which are contained in the XML document will be replaced with the more descriptive values "Active" and "Inactive".

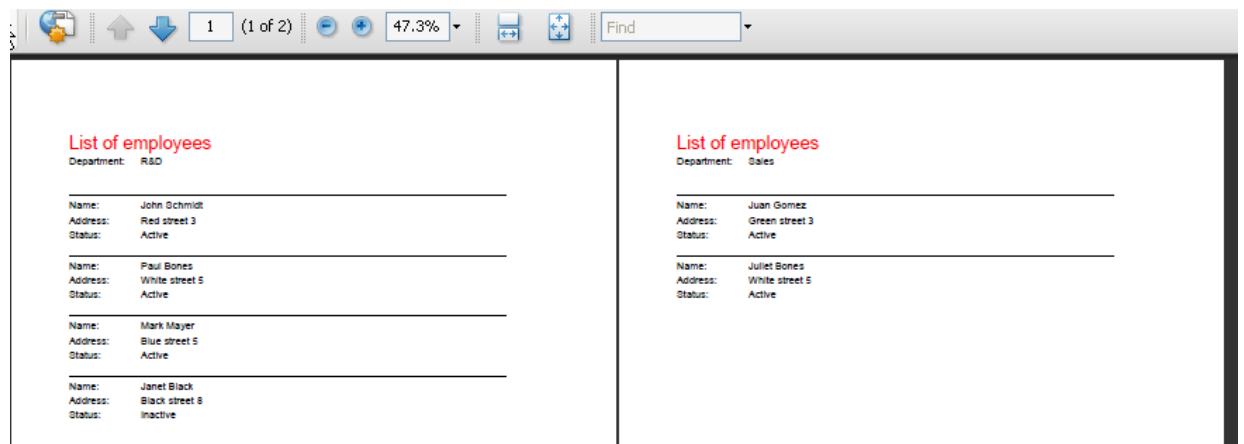
9. We can test the template by clicking on the PDF button . The output will now contain the detail information (persons):



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



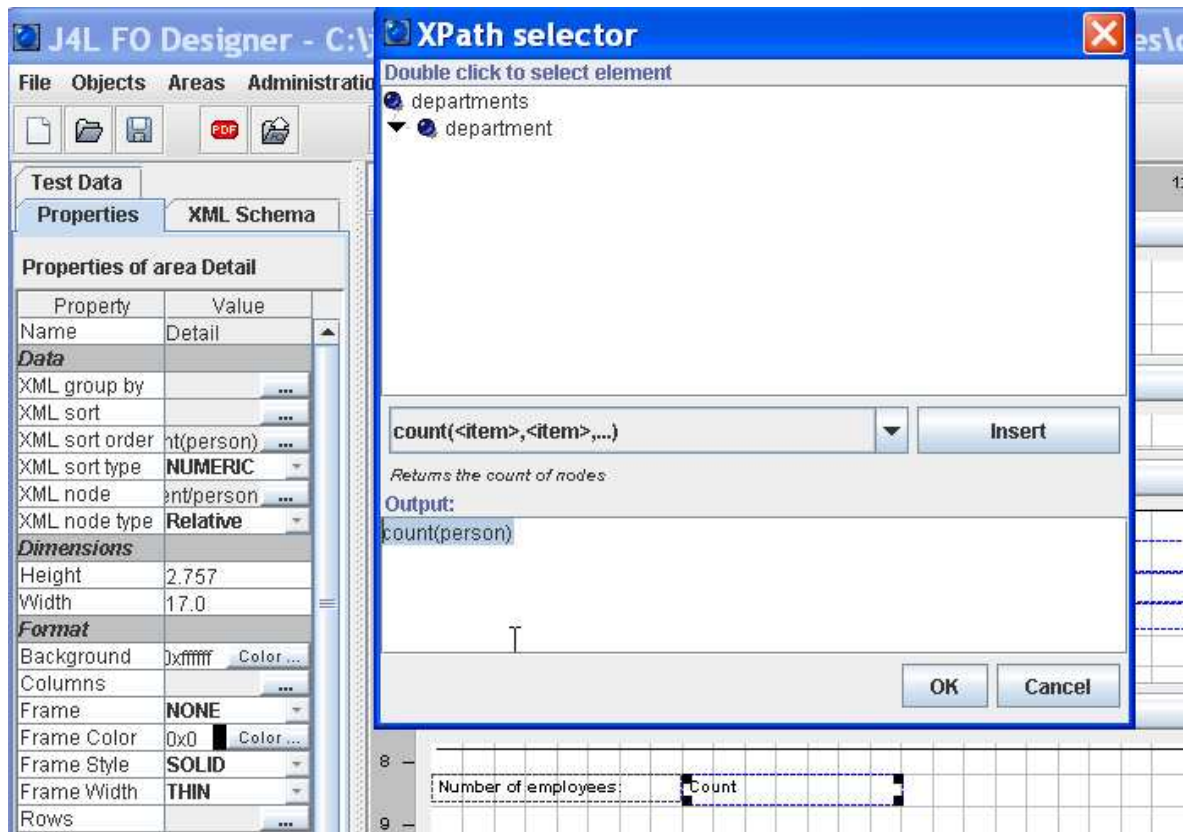
11. The output is now:



12. 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.



6. The invoice IDOC example explained

The example placed in the examples\idoc_invoice subdirectory shows how you can create a user friendly PDF file from a SAP XML IDOC. This examples 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
		<i>This is a comment to the delivery of item 1</i>				
000020	Mat2	Container 45 inches	17.00	18.45	0	313.65
		<i>This is a comment to the delivery of item 2</i>				
		<i>This is another comment for item 2</i>				

Before Tax: 438.04

VAT: 0

Total: 438.04

The and invoiceIDOC.xrp file is:

Header									
Invoice									
Bill to:			Delivered to:			Number:		BELNR	
BilltoName			shiptoName			Date:		DATUM	
BillToAdd			shiptoAdd			Order:		BELNR	
BilltoCity			shiptoCity						
Page Header									
IDOC		DOCNUM							
Detail 2 columns header									
Header comments:									
Detail 2 - /INVOIC01/IDOC/E1EDKT1/E1EDKT2/TDLINE									
TDLINE									
Detail columns header									
Item	Article	Description	Quantity	Price	VAT	Amount			
Detail - /INVOIC01/IDOC/E1EDP01									
POSEX	IDTNR	KTEXT	MENGE	BETR0	MSATZ	BETR0			
Detail 1 - /INVOIC01/IDOC/E1EDP01/E1EDPT1/E1EDPT2									
TDLINE									
Footer									
						Before Tax:	Total		
						VAT:	Total		

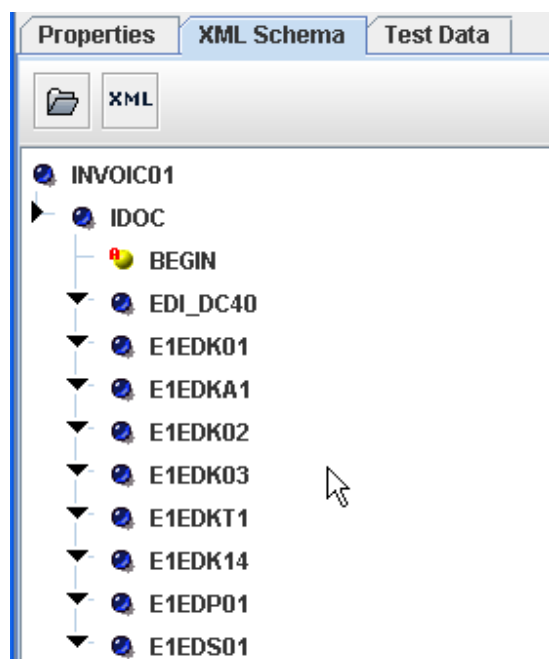
The structure of the template is:

- There is a Page header that contains the IDOC number
- There is a Invoice header that contains the buyer information (bill to and delivered to)
 - 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.
 - after the comments there is a table header for the items in the invoice

- 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.
 - 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.
- 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 exists only (and always) once in each invoice. This therefore assumes the input IXML document (IDOC) can contain only one invoice.
- 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 XML button, select the INVOIC-out.xml file and we get a pseudo-schema 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 properties to achieve this are:

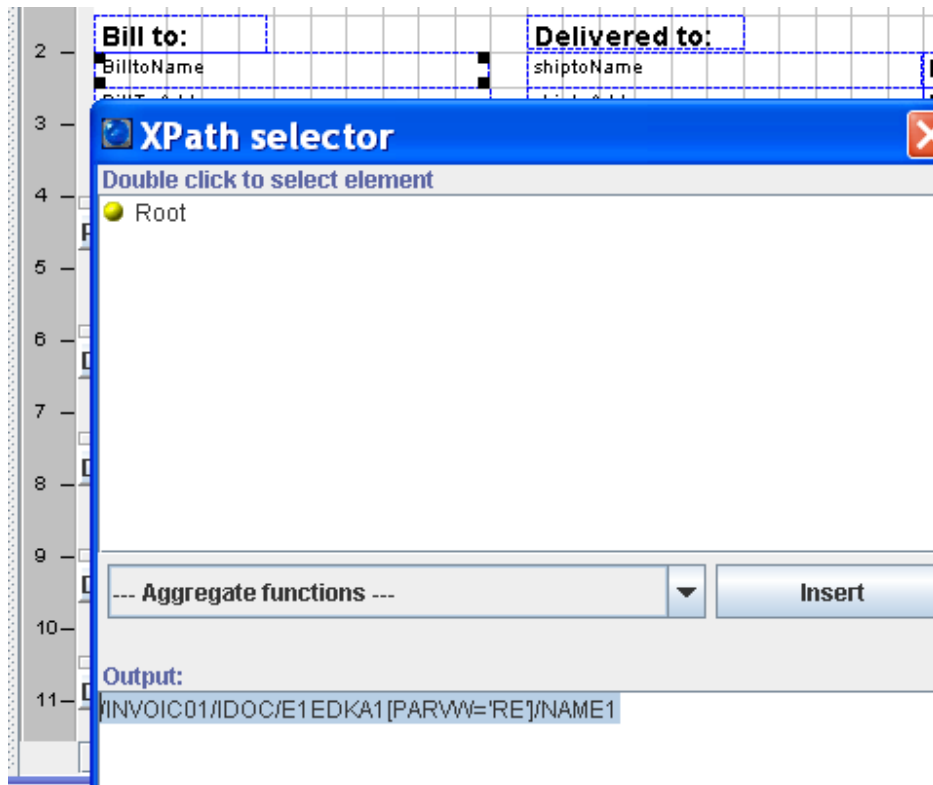
- The *Super area* property is set to the Detail area. First level detail areas point to the Header area.
- The *XML node* property is pointing to the `/INVOIC01/IDOC/E1EDP01/E1EDPT1/E1EDPT2` which contains the comments of the item.
- The *XML node type* is set to relative, since the super area is pointing 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
```

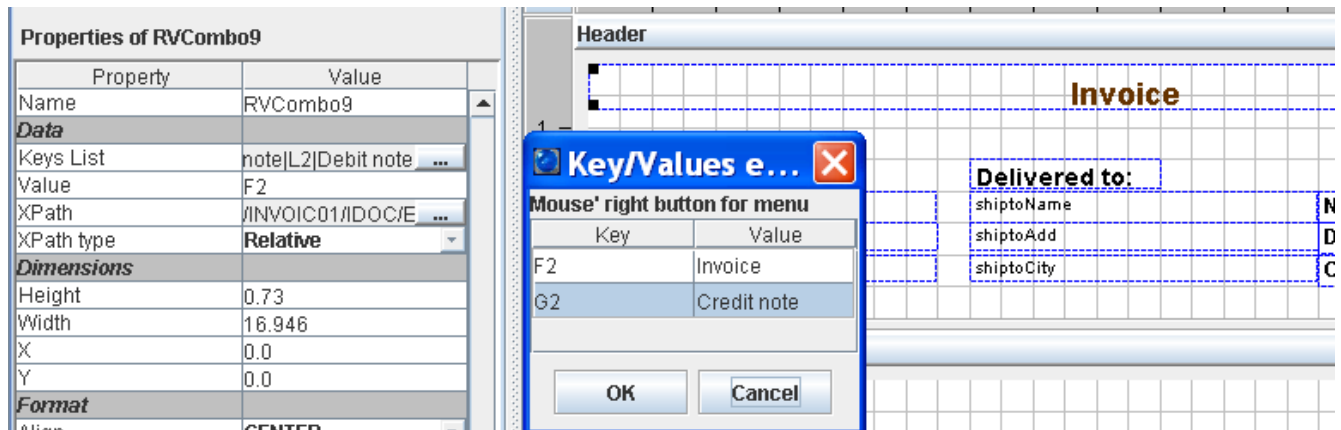
which means, select the `NAME1` node in the `/INVOIC01/IDOC/E1EDKA1` node that meets the condition `PARVW='RE'`.



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.



7."Group by" example

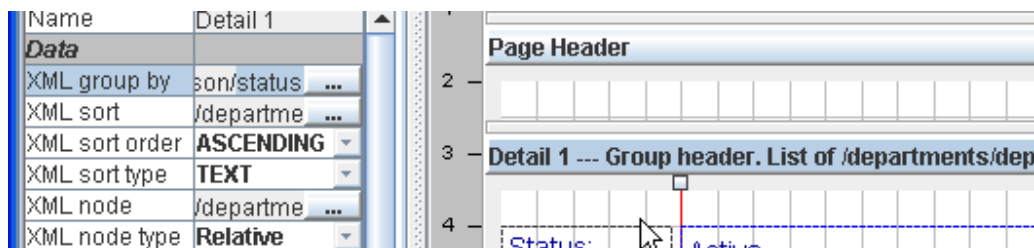
The directory *examples\employees* there is a file called *departmentEmployeesGrouping.xrp*. This example shows how to use the "group by" feature. The example takes as input a list of employees and groups them by status (in the test file there are 2 status, active and inactive).

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.



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

Once you set a value for the "XML group by" property, the area title bar will contain the text "Group header, List of ...".

- The group detail is an area whose super area is the group header. This areas contain the text "Group detail. Groups of ..." in the title bar. The group detail areas are used to output the elements we are grouping , in this case *persons*.

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 file *examples\employees\departmentEmployeesGrouping_footer.xrp* 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)
```

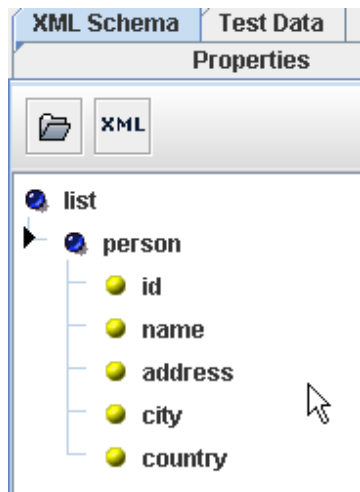

this means select the salary of all employee whose status is the same status as the current group and then sum all selected salaries.

8.The Barcodes and dynamic images example explained

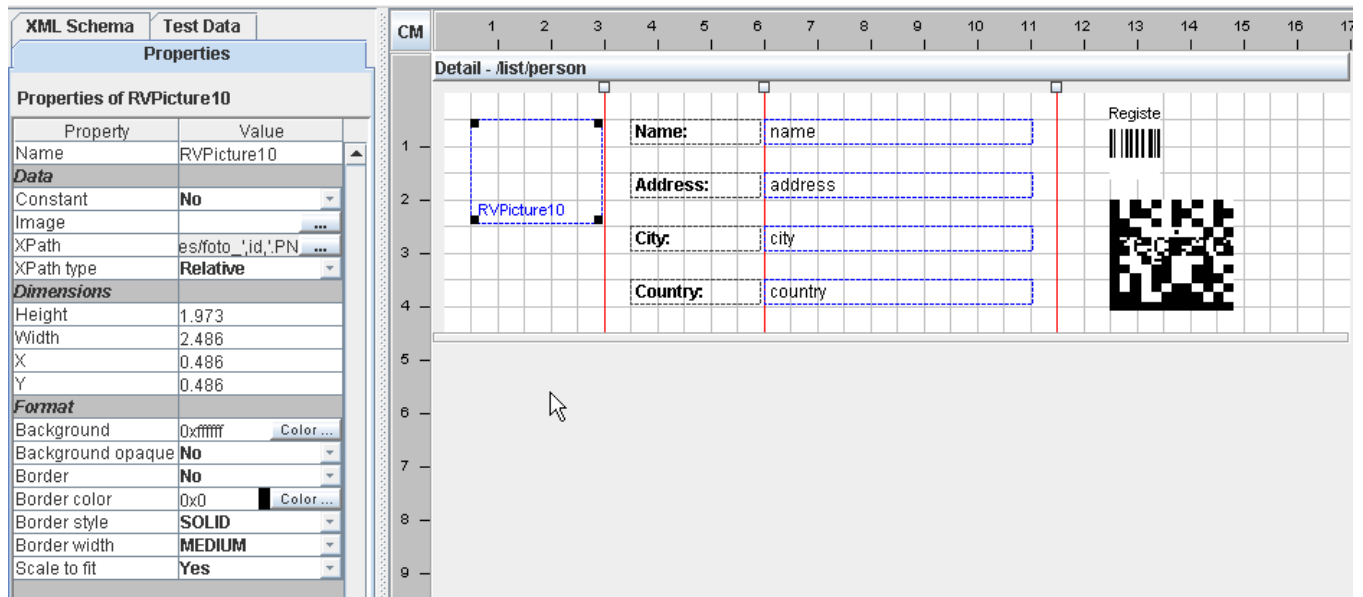
The directory examples\barcodes contains the personal card example (file personCard.xrp) which shows how to:

- use barcodes
- dynamically select an image

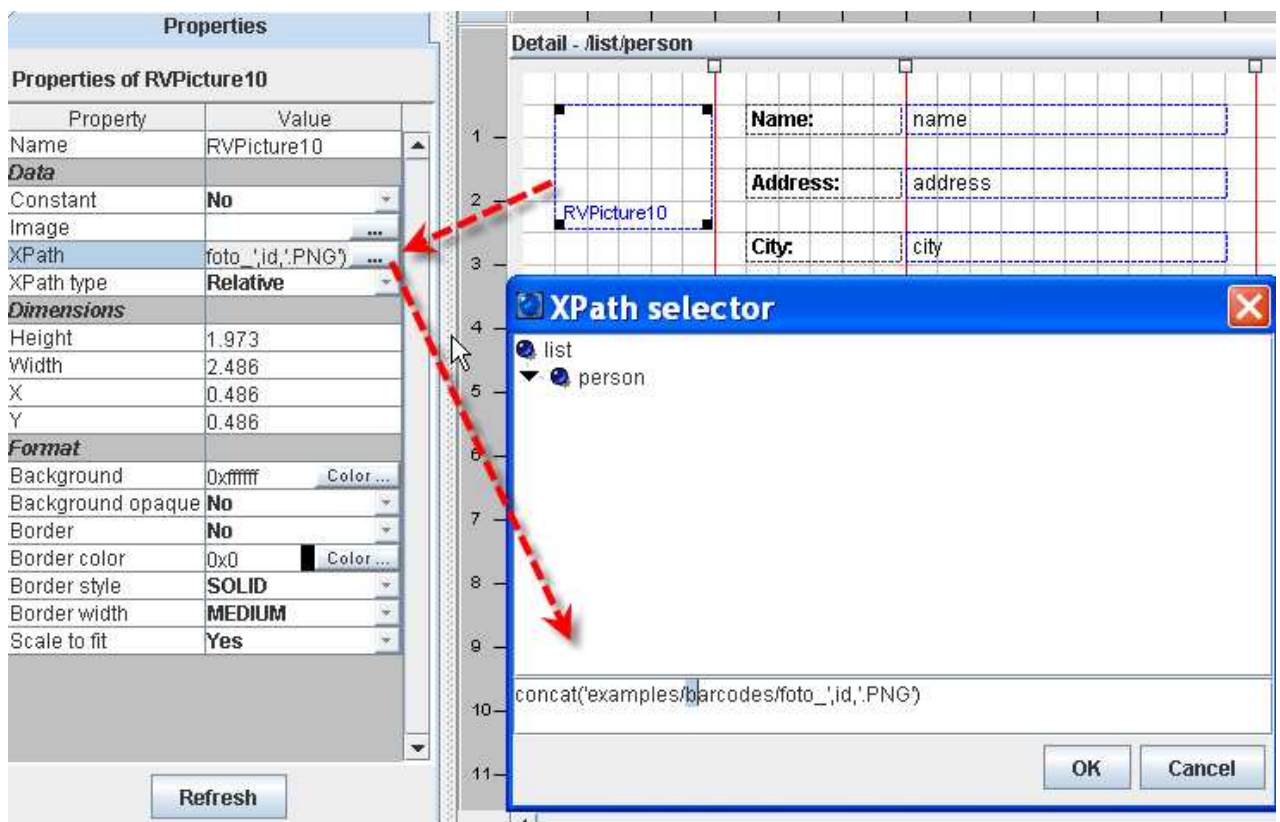
This example creates a personal card with the name and address of a person together with the foto and a barcode which encodes the name of the person. The input file is a XML file with a list of persons and their information, the schema of the XML file is very simple:



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



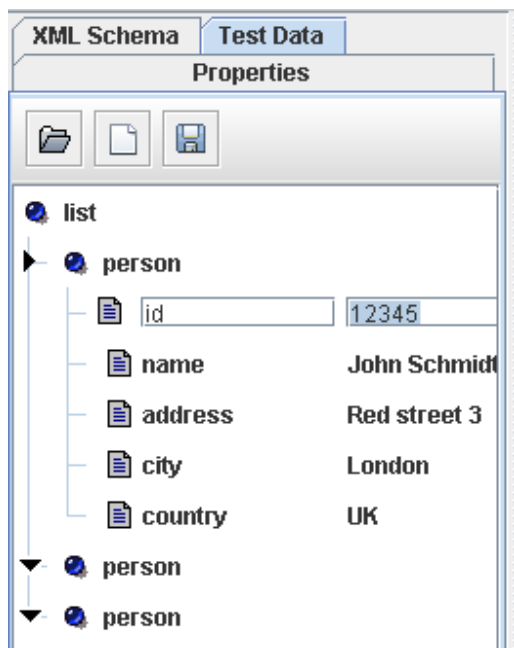
The image object could be a static image file if you enter a file name in the *image* field, however in this example we want to select a different image for each person (the foto), so we use the *xpath* field and enter the value `concat('examples/barcodes/foto_',id,'.PNG')`.



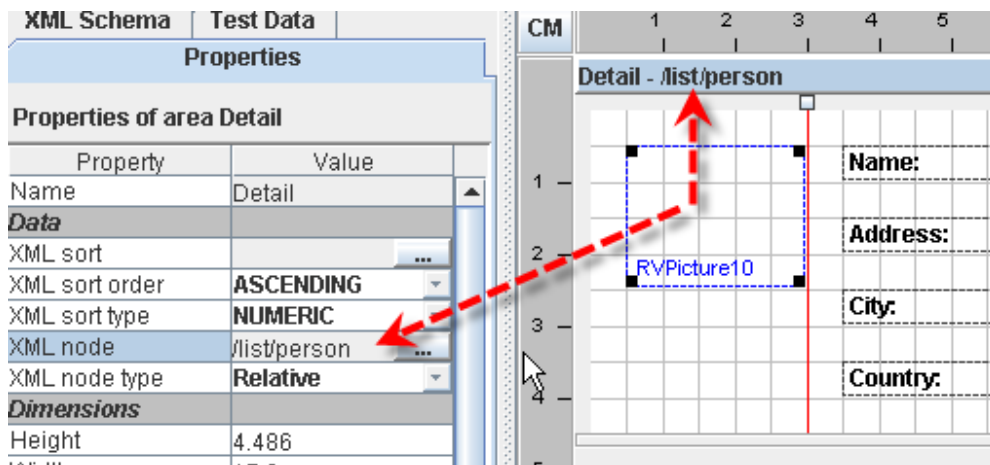
This expression concatenates the 3 values:

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

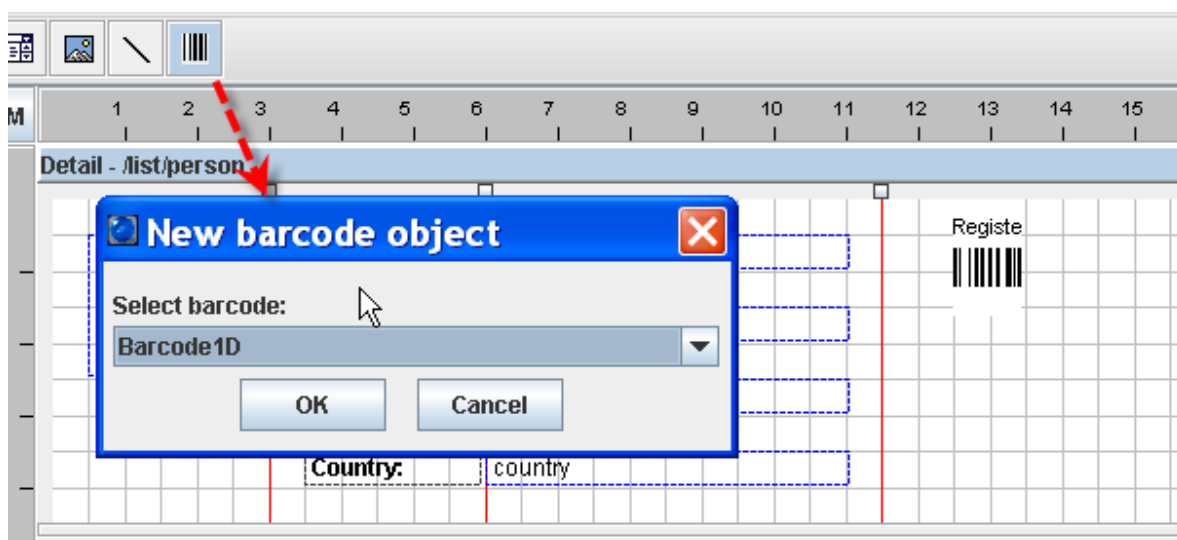
Since one of the persons has the id 12345 the result for that person will be *examples/barcodes/foto_12345.jpg*. So that is the file that will be used as foto for that person.



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.



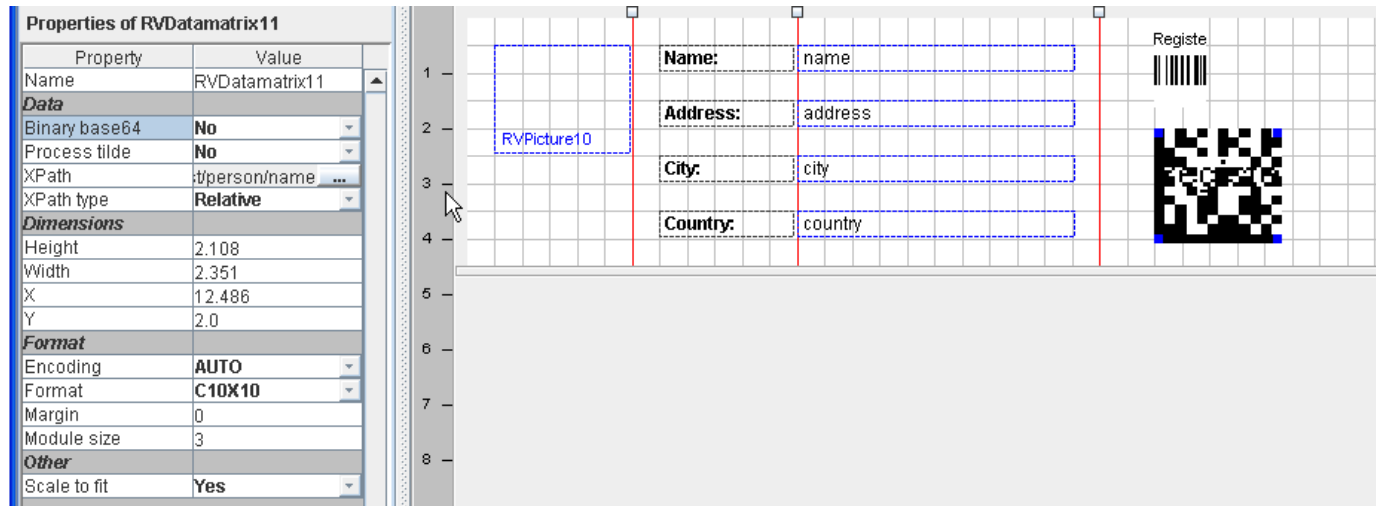
The barcode is the second interesting object in this example. When you add a barcode to your document the barcode type selection window will popup. At this point you have to decide if it will be a 1D barcode (e.g. EAN or Code128) or a 2D barcode (Datamatrix, PDF417 ...).



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>



9. 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.html>

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):
 - o the "Data File" field must point to the chart's parameter file
 - o 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).

Property	Value
Name	RVGraph1
Data	
1 Parameter	SERIE_DATA_1
1 XPath	/sales/month/products ...
2 Parameter	SERIE_DATA_2
2 XPath	/sales/month/services ...
3 Parameter	XAXIS_LABELS
3 XPath	/sales/month/name ...
4 Parameter	
4 XPath	...
5 Parameter	
5 XPath	
6 Parameter	
6 XPath	...
7 Parameter	
7 XPath	...
8 Parameter	
8 XPath	...
9 Parameter	
9 XPath	...
Data File	mples\chart\stackedBar.txt ...
Dimensions	

10. The running totals example explained

A very common 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
--------	----------	-------

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

Page 2

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:

Detail - /Order/OrderDetail/ListOfItemDetail/ItemDetail

Number	Article	Description	Price	Quantity	Tax	Amount

Footer

Page Footer

Page <fo:page-number></fo:page-number>

subtotalLabel subtotal

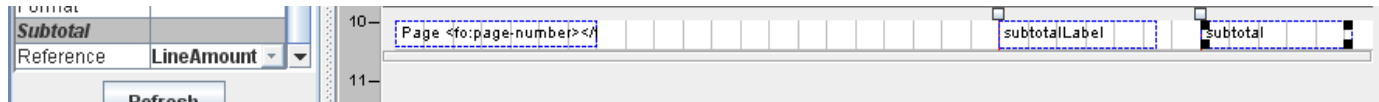
The *LineAmount* field has the property *Use for Subtotal* set to *Yes* and the subtotal type *SUM* will cause the values of this field to be summed up during the PDF generation.

Subtotal	
Constant value	
Constant last...	
Subtotal type	SUM
Use for Subto...	Yes

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.

Subtotal	
Constant value	Subtotal
Constant last...	Total
Subtotal type	CONSTANT
Use for Subto...	Yes

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



11. PDF interactive forms

Introduction

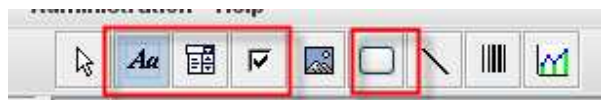
Starting with version 1.5 you can create PDF forms using J4L FO Designer (Suite edition), however this requires that you use our FOP server since the generation of PDF forms is not part of the XSL-FO standard.

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 an additional license key file. This file is included in the "**FO Designer Suite**" version only. The file is called *FormsLicense.txt* and must be copied to the working directory of FO Designer. This is not required in the evaluation version.

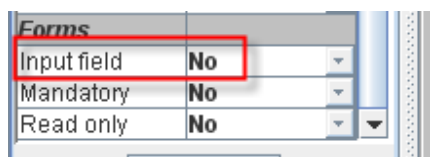
Form objects

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

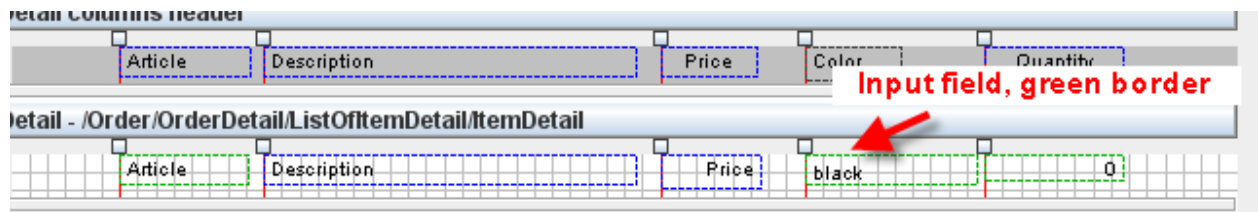


- Text fields. This object is also used in non-form PDF files to display data.
- Combo boxes. This object is also used in non-form PDF files to display data.
- Checkbox. This object can be used **in forms only**.
- Button (to submit or reset a form). This object can be used **in forms only**.

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 in the 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.

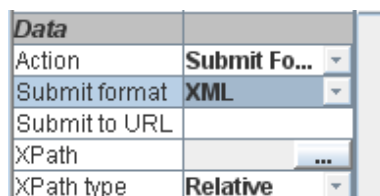
Combo box

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

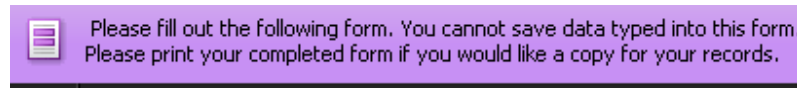
- The submission format, HTML, XML or PDF

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 save 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 the 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.

Order form

ABC Enterprises Customer number: 3000 

ABC Road Express delivery: ☒ checkbox

Alpine Requested delivery date

Article	Description	Price	Color	Quantity
R-5000	ABC Skirt	<input type="text"/>	black <input type="text"/>	0 <input type="text"/>
R-3456	ABC T-Shirt	1000.0	black <input type="text"/>	0 <input type="text"/>
R-5001	ABC Jeans	5500.0	black <input type="text"/>	0 <input type="text"/>
R-5009	ABC Pullover	2000.0	black <input type="text"/>	0 <input type="text"/>

Comments

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

report.pdf - Nitro PDF Reader

File Save (Strg+S) Save the active document.

Hand Zoom Fit Page Rotate View Select Add Markup Type QuickSign Create Extract Extract from File Text Images Reset Do More With Pro Upgrade

This document contains fillable form fields.

report x

You are using a demo version

Order form

ABC Enterprises Customer number: 3000

My new street 1 Express delivery: ☒

Alpine Requested delivery date dd/mm/yyyy

Article	Description	Price	Color	Quantity
R-5000	ABC Skirt	5000.0	red	20
R-3456	ABC T-Shirt	1000.0	black	0
R-5001	ABC Jeans	5500.0	white	1
R-5009	ABC Pullover	2000.0	black	0

Comments

thank you

LOGO

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:



Select the filled PDF file and click extract, this will extract the fields in XML format:

5\src\res\filledorder.pdf Browse... extract

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:

```
<?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>
  </field>
</fields>
</formData>
```

12. Flavours

You can use flavours to change the look of a field or combobox 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:

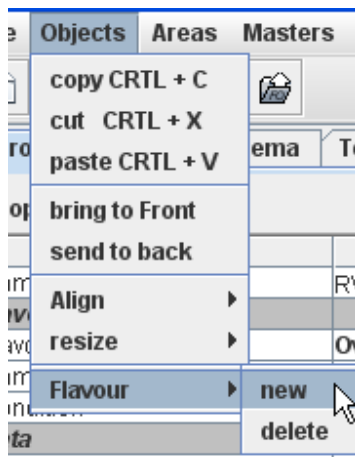
examples\xcbl_order\order_flavour.xrp

This report creates an output like this:

Quantity	Tax	Amount
11.0	16	1110.00
1.0	16	1000.0
Tax:		337.6
Total:		2447.6

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:
 - o The combobox "Flavour" is used to select the current flavour you are working on.
 - o The "Name" entry, is the name of the current flavour. Use the entry to change the name of the flavour.
 - o 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.

Flavours	
Flavour	Over1000
Name	Over1000
Condition	ue/MonetaryAmount > 1000 ...
Data	

In this example we have defined a new flavour called "Over1000" and we have set the condition to be " ..MonetaryAmount >1000". The font color of the flavour has been set to red.

13. FAQs

We have create how-tos guides for the following products:

- [SAP XI / PI](#)
- [Oracle APEX](#)
- [Oracle BPEL](#)
- [Apache Cocoon](#)
- [Mule ESB](#)

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 *j4lex: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:

```
j4lex: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 *j4lex* prefix you must use the J4L FOP server, if you use another XSL-FO processor these functions are not available.

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.mycompany.fo;

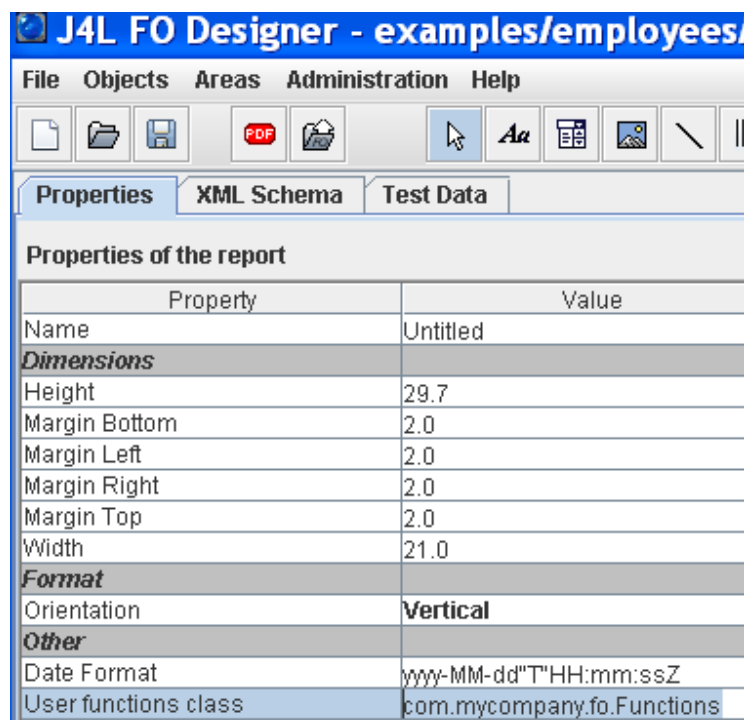
public class Functions {

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

}

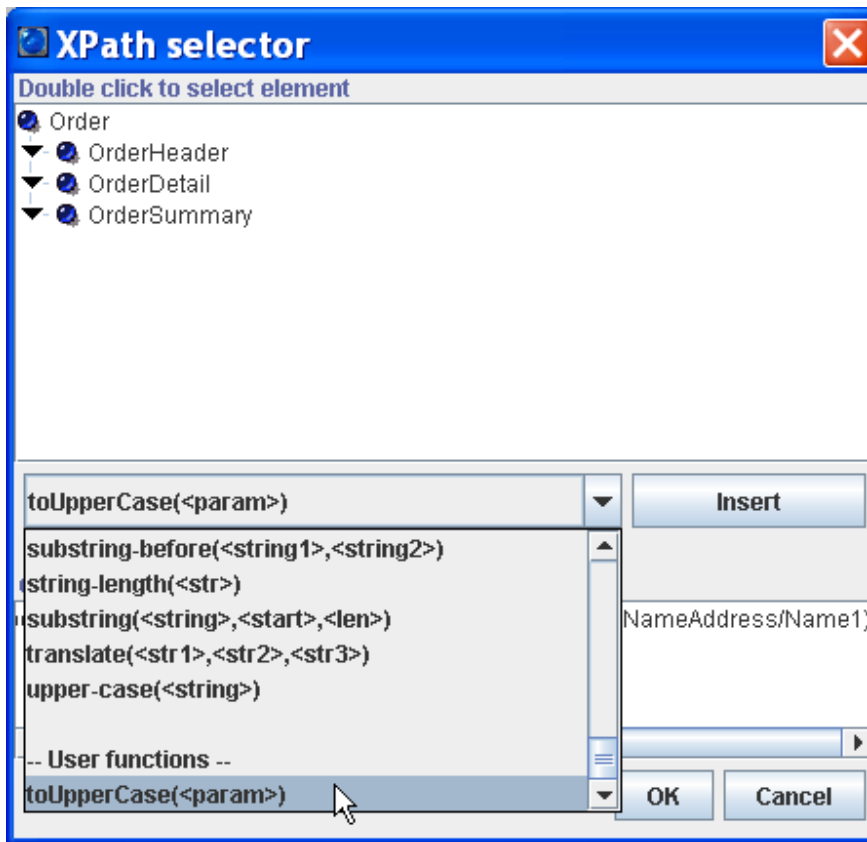
```

2. Compile the java class into a jar file. The file must be called **userfunctions.jar** and you have to copy it to the **lib** subdirectory of the J4L FO Designer installation.
3. Restart the J4L FO Designer
4. In the properties tab of your template you must enter the Java class name in the *User functions class* field, see highlighted field in the screenshot:



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. edit the file *tools/createFontMetrics.bat*. This batch is used to create an XML metrics file for each TTF file, so you have to change these two lines:

```
SET TTFFILE=c:\windows\Fonts\comic.ttf
SET OUTPUT=comic.xml
```

the TTFFILE variable must point to your TTF file and the OUTPUT file to the output XML file (you can select any name you like). We recommend to use an absolute file name for the OUTPUT file.

2. run the modified *tools/createFontMetrics.bat*.
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>
```

- 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"/>
</font>
```

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.

- 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:



However, if the font's family name is not recognized as an installed font in your Java virtual machine you will see an * as a prefix to the font name (see example MyFont in the screenshots). In these cases the designer is not able to use your font at design time, however the font will be used by Apache FOP for creating the PDF file. If you want to make the new font available for design time also, you have to add the TTF file to the Java runtime also. This is achieved by copying the ttf file to the Java *jre/lib/fonts* directory.

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:

- In the designer activate the namespace removal. This setting is in the Administration menu, settings window, process tab.
- At runtime you use:
 - either our servlet which supports the *REMOVENS* parameter
 - or our *com.java4less.xreport.fop.FOPProcessor* class whose *process()* method supports the *removeNS* parameter that must be set to true.

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:

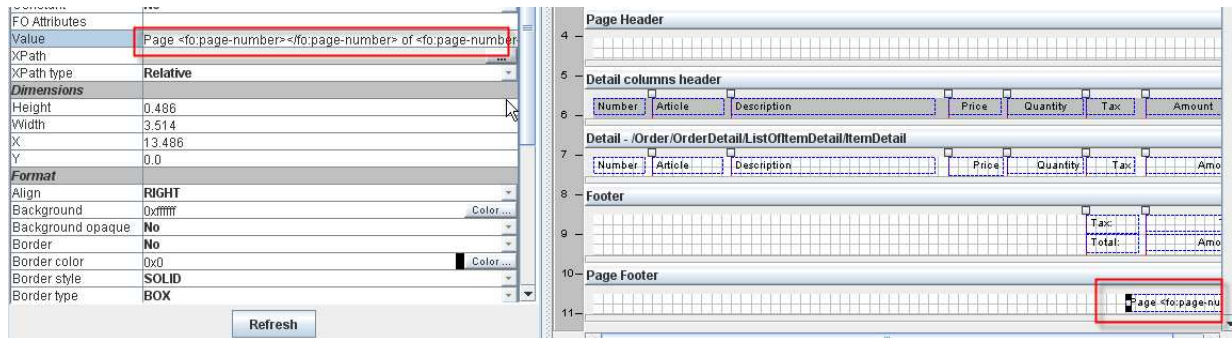
Condition	
Data	
Constant	No
FO Attributes	white-space-collapse='true'
Value	Purchase order
XPath	
XPath type	Relative

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.



14. Startup troubleshooting

FO Designer does not start up

1. Make sure you are using Java 1.6 or later. You can see this error by running *runDebug.bat* instead of *run.bat*. If you use the wrong Java version you will see an error "*Bad version number..*". You can check your current Java version by opening a command window and running the command "*java -version*"
2. Another possible reason for a failed start is if the working directory is not the one where FO Designer is installed.

Error opening jnlp file from

If you use the URL <http://servername:8087/J4LFOPServer/FODesigner.html> to run the FO Designer in the Server you will be downloading the FO Designer using Java Webstart. If the browser reports it cannot open the file "*FODesigner.jnlp*" (extension *jnlp*), you have to associate the file type "*jnlp*" to the Java Web Start application which is in "*c:\your_java_directory\jre\bin\javaws.exe*".

Note the browser will ask you which program should be used to open these type of files (*jnlp*) that are now known. You have to select the option "*Select program from list*" and then select "*Browse..*" to locate the *javaws.exe* file.

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 has read and write permissions for the FO Designer folder. Using right mouse click on the folder you can set the permissions.

15. Third party licenses

This product 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 library icepdf for the pdf viewer is also delivered under the Apache 2.0 license.

The Bounce library has an own license type described in the file lib/bounce_license.txt.

The lib subdirectory of the delivery file 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.

16. Contact

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

