Leverage the Oracle BI Publisher templating power and produce high fidelity reports

Contenu Afficher

Abstract

Oracle APEX is a tool which is well suited for rapid applications developpement (RAD). It's possible to produce a tabular report based one a query in a very fast way. This is a standard feature. But when the goal is to build high precision reports « pixel perfect », like an invoice, for instance, we must switch to an other approach. The good news is that Oracle APEX gives ability to use XSLT-FO templates instead the standard ones. Perfect ! But how authoring these XSLT-FO templates, while keeping the APEX low code philosophy ? The following in this post explains a possible solution.

The approach is to use Oracle BI PUblisher Desktop product (BIP). It's a plugin for Microsft Word which « transforms » MS Word in a template authoring tool without burden of with XSLT and XPATH syntax. (One other alternative is to use Apache POI project then adding XPATH expressions in the XSL-FO template)
Important notice: It will be possible to use a functionnal subset of the BIP product but not all features. Therefore, the following informations must be considered as an experimental work only.

Materials & Documentation

Prerequisites

License Prerequisites

- A commercial license for using BI Publisher (BIP)

- A open source license for the SAXON xsl parser

We’ll not rely on BIP FOP engine and we’ll just use the BIP Desktop plugin. The license for BIP can be purchased on a *Named User* metric. That means that we can manage to buy at a *low price* but with a minimum of 10 licenses per processor.

SAXON is sold through several editions. The home edition, SAXON-HE is an open source product under Mozilla public license V2. It can be enough for testing and regular use as well.

Download & Documentation

- Download BIP desktop
- Download SAXON-HE
- Conversion tool BIP_TO_FOP

Documentation

- BIP Desktop 12.2.1.3
- RTF templating tutorial

About XPATH and XSLT Syntax, BIP Desktop samples are a good source of inspiration: « C:\Program Files\Oracle\BI Publisher\BI Publisher Desktop\Template Builder for Word\samples\RTF templates »

Technical prerequisites

This post assumes a basic knowledge of RTF template design with BIP.
Once the RTF template is setup, we use the embeded function for exporting the template as an XSL-FO stylesheet.

Export option inside BIP Desktop plugin

From BIP plugin, an export menu option gives ability to export the template as a XSL-FO stylesheet. This exported stylesheet contains additional xsl function which are specific to BIP and can't be used directly in Oracle APEX. In order to use a regular FO Processor (the one imbedded in APEX), we have to either remove or convert it, when possible.
For that purpose, we use a simple plsql procedure which will do a kind of cleanup in the template. It'is a fork of a Pavel Glebov's project https://github.com/patrickmonaco/BIP_TO_FOP.
Some additional details have been added.
The XSL-FO beeing a regular XML document,it would have been ideally more judicious to use a XSLT parsing method, but this discussion is out of the scope of this post.

Implementation

Reminders about the FOP engine

An Apache FOP engine 2.3 is embedded in the ORDS java application. This engine relies on a XALAN (XSLTC) transformerFactory. This parser is compliant whit XSLT V1 but not XSLT v2. And this is a little bit ennoying for our purpose because some group functions generated by BIP should have to be rewritten for working in XSLT v1.
Because we don't want to developp a too complex convert tool, I prefered susbitute for XALAN (XSLTC) an other XSLT parser (SAXON) in order to minimize conversion effort.

This parser, SAXON, is XSLT v2 compliant . Keep in mind that license fees can be required in some cases.

So, once we setup the new transformerFactory (new way to transform XML input) in the CATALINA environmement file (if Apache Tomcat is used as a java container) , and after adding SAXON library in Tomcat *lib* directory, the way becomes much easier.

```
set CATALINA_OPTS=%CATALINA_OPTS% -
Djavax.xml.transform.TransformerFactory=net.sf.saxon.TransformerFactoryImpl
```

Once the query has been created and the layout has been associated to, the report can be launched with the following URL:
`f?p=&APP_ID.:0:&SESSION.:PRINT_REPORT=empdept`

Install database objects

A DDL script (FOP_DDL) and a package, BOP_PKG, have to be installed in the worskpace owner schema.
cf https://github.com/patrickmonaco/BIP_TO_FOP

Adding a template repository

A simple APEX application is provided for managing templates. When a new template BIP template is uploaded, a new converted template is created.
At this stage, the lifecycle is:

- Update/Create RTF template with MS Word & BIP Plugin
- Export RTF template in XSL-FO (with BIP Word Plugin)
- Upload the new version template in BOP repository
- Download the converted template
- Remove the previous layout
- Re-create a new layout with the same name (or not) and upload the converted

template
- Update the report query to setup the new layout

The process is quite cumbersome because on each update, we have to remove then re-create a new layout, then upload the converted wslt-fo template. In order to minimize the tasks, the idea is to synchronise the repository with the apex layouts internal table.
Note: The following step is optionnal. Remove the related code inside the application if this step is to be ignored.

The name of this table is: `WWV_FLOW_REPORT_LAYOUTS`
The pl/sql procedure then updates the layout by uploading the template converted previously. The APEX table is: `WWV_FLOW_REPORT_LAYOUTS` .
Full access rights on this table must be granted to the workspace owner.

As SYS user, let’s grant DML on this table

```
grant ALL on APEX_<XXX>.WWV_FLOW_REPORT_LAYOUTS TO DEMO;
```

Now the lifecycle becomes:

- Update RTF template with MS Word
- Export RTF template in XSL-FO (with BIP Word Plugin)
- Upload the new version template in BOP repository

Because each update in BOP Table will update the conterpart in `WWV_FLOW_REPORT_LAYOUTS` table, no more effort is needed.

Limitations:

- Numerous BIP (xdo) functions are not supported
- Wordarts, shapes are not supported
- Static graphics (ie: an image included in the rtf template) encoded in base64 are not supported
- Formatting in XPATH expression is supported for number datatype only

Discussion about XML dataset

Above, this is a XML sample generated by a Report Query.

```
select emp.*,
dept.dname,
dept.loc
from emp, dept
where emp.deptno = dept.deptno
```

We can observe that the dataset is rendered as a single table (a single rowset), in other words, a flat tree. There is no way to build, with a regular report query, a complex tree document. The consequence is that we have to use Group Function inside the xst-fo template if break page is needed in the final report.

Therefore, because of the previous point, we'll have to rely on XSLT group function in the plugin, for achieving some behaviours like page break. This point explains why we subsitute to XALAN (XSLTC), the SAXON parser which is more suitable.

Use cases

Tabular reports

There is no difficulty to design a template which produce a tabular report. It’s possible to leverage most of the features of BIP Desktop combined with the powerful of MS Word. The page break is managed by the Apache XSL-FO engine.

Tabular output with conditionnal formating

Invoices

It’s a very common type of document where there is :

- a header: (invoice header, customer address, ..)
- one or more set of data multi-lines related to the invoice (product items)
- a footer: total and legal mentions, terms and conditions …

In the context of Oracle APEX, we must keep in mind that the XML output is flat. We just can leverage multiple rowsets in the same report. That leads to two different ways for building the XML dataset. The first one is to get a single rowset by joing all needed data upon a common key (order_id). The second one is to get a first rowset for the header information and other ones for multiple lines data.

One occurence of a result set

Goal: print a single document with a single occurence. ie: An invoice for a given order id.

In this case, the query can contains a single sql statement whichs is a join with all participants tables (order, ords items, products, customer).
Or we can combine two queries: one for the header ans on another for the order lines.

All occurences of a result set

In this case, the goal is to print a single document which contains all occurences returned by a query with an optional page break between occurences.

Query:

```
select emp.*,
dept.dname,
dept.loc
from emp, dept
where emp.deptno = dept.deptno
```

Oracle APEX generates a XML document with the following schema:

XML schema generated by APEX Report Query

Output pdf with page break on Department

BIP RTF template

Fonts

Using specific fonts requires some additionnal setup. We need to adapt ORDS starting parameters.
A parameter in ORDS configuration file must be modified for pointing to the FOP configuration file:

```
<entry key="fop.configfile">
X:\distrib\fop-2.2\fop\conf\fopapex.xconf
</entry>
```

Once the server has been restarted, all the requierd fonts will be honored by Apache FOP. Otherwise, there is a fallback mechanism which maps the original fonts on a small subset (Times Roman).

Fonts embedded by the pdf document with fonts

Fonts embedded in the output pdf without fonts

Conclusion

The method presented previously shows up a good balance between complexity/High quality and simplicity/basic rendition in pdf document design. The BIP desktop plugin gives a very good flexibility provided you get a correct knowledge of MS Word.

APPENDIX

The following sample gives an overview of the XPATH query wich is generated by the

BIP desktop plugin. Of course, it’s always possible to enrich the behavior by adding some additional XPATH expressions.

XPATH generated by BIP Desktop plugin

PDF output with the FOP engine / report with XSLT break

PDF output with the APEX FOP engine – Master detail

Simple Invoice RTF template in BIP desktop

Queries used for invoice demo

Multi-queries – two rowsets

```
select
o.order_id,
o.customer_id,
o.order_total,
to_char(o.order_timestamp,'DD MONTH YYYY') order_timestamp,
c.cust_first_name,
c.cust_last_name,
c.cust_street_address1,
c.cust_street_address2,
c.cust_city,
c.cust_state,
c.cust_postal_code
from demo_orders o,
demo_customers c
where c.customer_id = o.customer_id
and o.order_id = 1
```

```
select
i.unit_price,
i.quantity,
round((i.quantity * i.unit_price),2) amount,
p.product_name,
p.product_description
from
demo_order_items i,
demo_product_info p
where p.product_id = i.product_id and
i.order_id=1
```

Sample multiple queries (rowsets)

Single Query – single rowset

```

select
 o.order_id,
 o.customer_id,
 o.order_total,
 o.order_timestamp,
 i.unit_price,
 i.quantity,
 p.product_name,
 p.product_description,
 c.cust_first_name,
 c.cust_last_name,
 c.cust_street_address1,
 c.cust_street_address2,
 c.cust_city,
 c.cust_state,
 c.cust_postal_code
 from demo_orders o,
 demo_order_items i,
 demo_customers c,
 demo_product_info p
 where o.order_id = i.order_id and
 p.product_id = i.product_id and
 c.customer_id = o.customer_id
```

Author

- Patrick

  GPM Factory