genero®
report writer

CREATING THE REPORT APPLICATION

Using Genero Report Writer
GRS 3.00

Objectives

After this instruction, you will be able to:

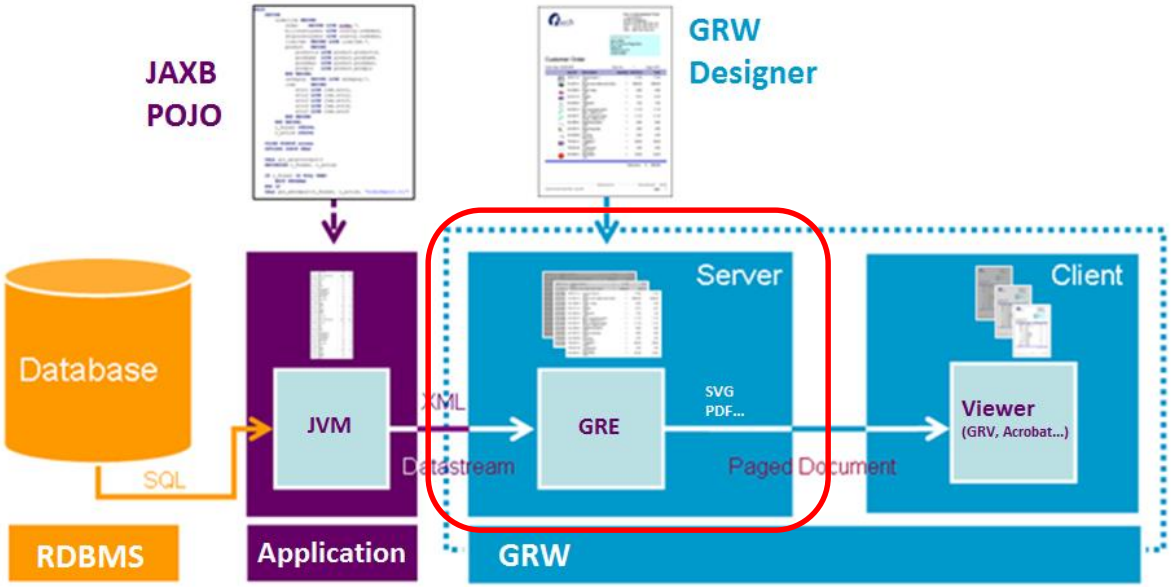
- Create a simple report program in Java
- Generate XSD schema file (required by Report Designer)
- Change some output options
  - Genero Web Viewer, PDF, Image…
  - Preview, Save on Disk, Print
- Know about tips for writing the Java report program

The simple Java report - Overview

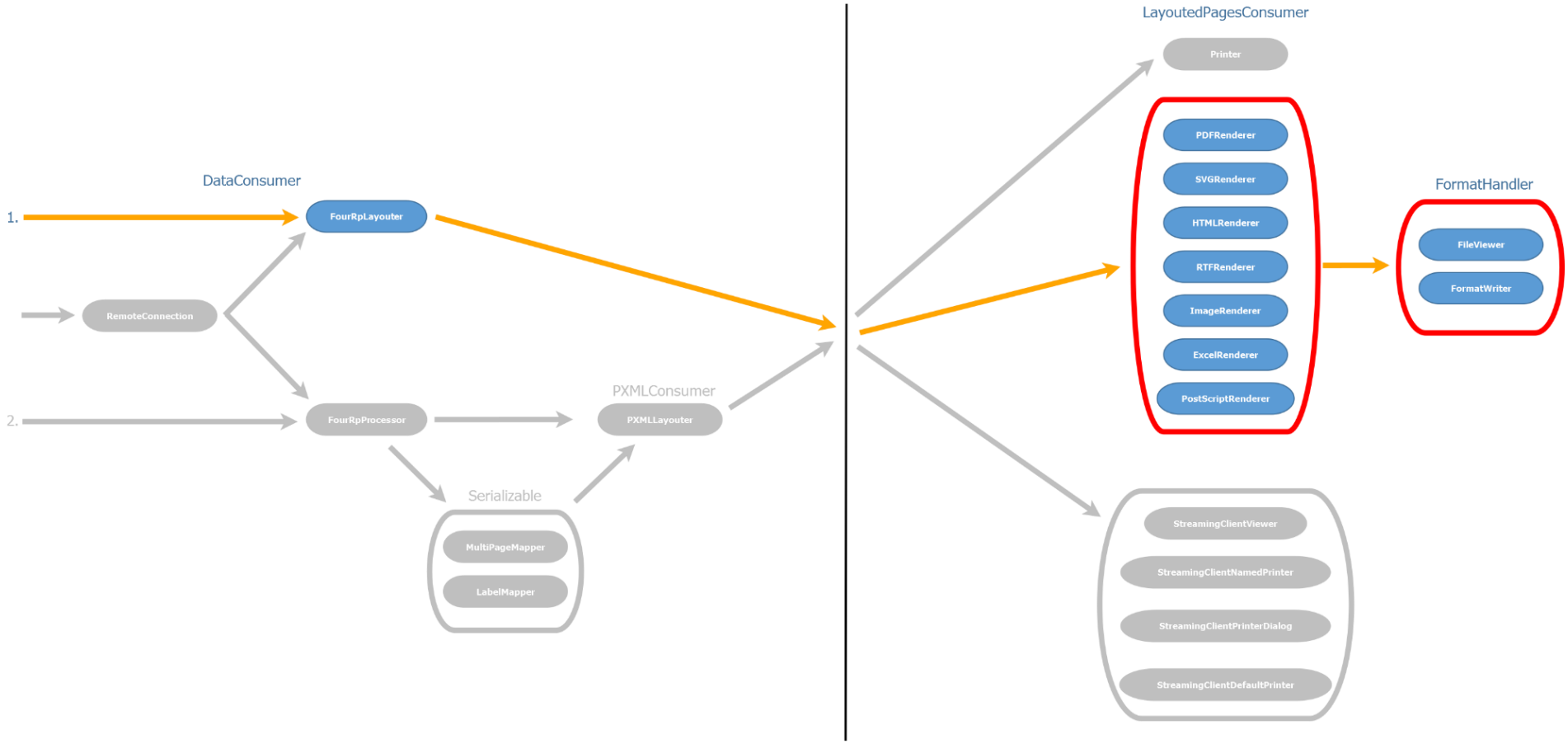
- 2 Plain Old Java Objects (POJO) act as model to the list report
  - Do not implement any specific interface
  - Not an extension of a specific class
  - Annotated to hint to JAXB how to be serialized to XML
- The Java program contains
  - The Report Model Objects (data to include)
  - Main method to run the report

Reminder: Architecture

- Genero Report Engine (GRE)
  - Uses the data and the report design to process the report
  - Outputs the report in accordance with the runtime API functions



Processing pipe (simplified)



The simple Java report – Data Definition

- 2 methods to serialize data
  - Ship data to report's content handler
    - Implementation of org.xml.sax.ContentHandler
    - Ship arbitrary sized XML documents
    - Low memory consumption
    - Example:

```
1 <?xml   version="1.0" encoding="UTF-8"?>
2 <List>
3   <Item>Item 1</Item>
4   <Item>Item 2</Item>
5 </List>
```

```
1     ContentHandler handler=processor.createContentHandler();
2
3     handler.startElement("List");
4     handler.startElement("Item");
5     handler.characters("Item 1");
6     handler.endElement("Item");
7     handler.startElement("Item");
8     handler.characters("Item 2");
9     handler.endElement("Item");
10    handler.endElement("List");
```

  - Use JAXB (Java Architecture for XML Binding)
    - Provides means to serialize plain Java objects in a streaming manner
    - Provides a schema generator to create an XML schema from annotated Java classes
    - Since Java 6

The simple Java report – Code description

- Import required Java classes

```
14 import com.fourjs.pxml.standardpipe.runtimeapi.*;
15 import java.awt.Desktop;
16 import java.io.File;
17 import javax.xml.bind.annotation.XmlElement;
18 import javax.xml.bind.annotation.XmlRootElement;
19 import java.io.IOException;
20 import javax.xml.bind.JAXBException;
21 import org.xml.sax.SAXException;
22 import java.util.Vector;
23 import java.util.Date;
24
```

  - com.fourjs.pxml.standardpipe.runtimeapi.*
    - Mandatory classes, defined in ‘gre.jar’
  - javax.xml.bind.annotation.*
    - Mandatory classes for XML element annotation

The simple Java report – Code description

- XML annotations
  - Provide information in the XSD schema
  - @XmlRootElement
    - causes a global element declaration to be produced in the schema

```
26 @XmlRootElement
27 public class Sales {
```

```
4   <xs:element name="sales" type="sales"/>
5
```

  - @XmlElement
    - maps a property to an XML element
    - required=true: GRE doesn't support optional variables
    - nillable=true: Allow null values

```
33  @XmlElement(required = true, nillable = true)
34  public String shopName;
35  @XmlElement(required = true)
36  public int zipCode;
37  @XmlElement(required = true, nillable = true)
38  public Date day;
```

```
6   <xs:complexType name="sales">
7     <xs:sequence>
8       <xs:element name="shopName" type="xs:string" nillable="true"/>
9       <xs:element name="zipCode" type="xs:int"/>
10      <xs:element name="day" type="xs:dateTime" nillable="true"/>
11      <xs:element name="items" type="salesItem" nillable="true" minOcc
12    </xs:sequence>
13  </xs:complexType>
```

  - @XmlAttribute
    - Maps a property to an XML attribute

The simple Java report – Code description

- Default Constructor required for JAXB deserialization

```
63      public Sales() { assert false; }
```

```
126     public SalesItem() { }
```

- Specify the report definition file (.4rp)
  - Simple String variable
  - Value hardcoded or passed as parameter

```
73          String designFile;
```

```
76          if (args.length == 0) {
77              designFile = "../Sales/SalesList.4rp";
78          } else {
79              designFile = args[0];
80          }
```

The simple Java report – Code description

- Specify the report output format (PDF)
  - Output file name
    - Simple String variable specifying the name of the file on disk
  - Renderer
    - PDFRenderer in this case
    - Check other renderers for other output formats in DOC
- Specify the handler
- Specify the report
  - FourRpLayouter

```
74          String outputFilename = "SalesList.pdf";
75
```

```
82          FormatHandler handler = new FormatWriter(outputFilename);
83          PDFRenderer renderer = new PDFRenderer(handler);
84          FourRpLayouter report = new FourRpLayouter(designFile, renderer);
85
```

The simple Java report – Code description

- Specifying the data

```
42      public Vector<SalesItem> items=new Vector<SalesItem>();
43
44      public Sales(String shopName, int zipCode, Date day)
45      {
46          this.shopName=shopName;
47          this.zipCode=zipCode;
48          this.day=day;
49          items.add(new SalesItem("Tablelamp",SalesItem.Category.Furniture,23.00,null));
50          items.add(new SalesItem("Tablelamp",SalesItem.Category.Furniture,267.00, items.lastElement()));
51          items.add(new SalesItem("Officechair",SalesItem.Category.Furniture,155.00, items.lastElement()));
52          items.add(new SalesItem("Grandfather clock",SalesItem.Category.Furniture,329.00, items.lastElement()));
53          items.add(new SalesItem("Scissors",SalesItem.Category.Supplies,19.00, items.lastElement()));
54          items.add(new SalesItem("Measuring tape",SalesItem.Category.Supplies,23.00, items.lastElement()));
55          items.add(new SalesItem("Sunglasses",SalesItem.Category.Travelling,15.95, items.lastElement()));
56          items.add(new SalesItem("Penknife",SalesItem.Category.Travelling,6.25, items.lastElement()));
57          items.add(new SalesItem("Ornateangel",SalesItem.Category.Art,1.95, items.lastElement()));
58      }
```

```
87          Sales data = new Sales("Columbus Arts", 75038, new Date());
```

- Run the report
  - Uses JAXB to stream the object data thru the rendering pipe

```
89          report.runFromJAXBObject(data);
```

- Generate and open the report file
  - Use Desktop.open()

```
92          File result = new File(outputFilename);
93          Desktop desktop = Desktop.getDesktop();
94          desktop.open(result);
```

Creating the XSD file

- Report template must match the XML data stream
- Edit report template against XSD schema
- Generate XSD schema with JAXB schema generator: schemagen
  - From the command line
    schemagen <file_name>.java
  - From GRS
    Add post compile command

Post Compile commands :

```
schemagen "$(InputDir)/$(InputBaseName).java"
$(move) "$(InputDir)/schema1.xsd" "$(InputDir)/$(InputBaseName).xsd"
```

Summary of minimum Java report code

- Use a handler object
  - FormatHandler handler = new FormatWriter(outputFilename);
- Define a renderer according to the expected output format
  - PDFRenderer renderer = new PDFRenderer(handler);
- Create the report object specifying the report design file and the renderer object
  - FourRpLayouter report = new FourRpLayouter(designFile, renderer);
- Specify the data
  - Sales data = new Sales("Columbus Arts", 75038, new Date());
- Run the report
  - report.runFromJAXBObject(data);
- Generate the report file on disk
  - File result = new File(outputFilename);

Changing output format

- Different output formats currently available
  - PDF (already seen), SVG, HTML, XLS/XLSX, RTF, Image, PostScript
- Specify the format by defining the appropriate renderer object
  - PDFRenderer, SVGRenderer, HTMLRenderer, ExcelRenderer, RTFRenderer, ImageRenderer, PostscriptRenderer
  - No FormatHandler object required for ImageRenderer

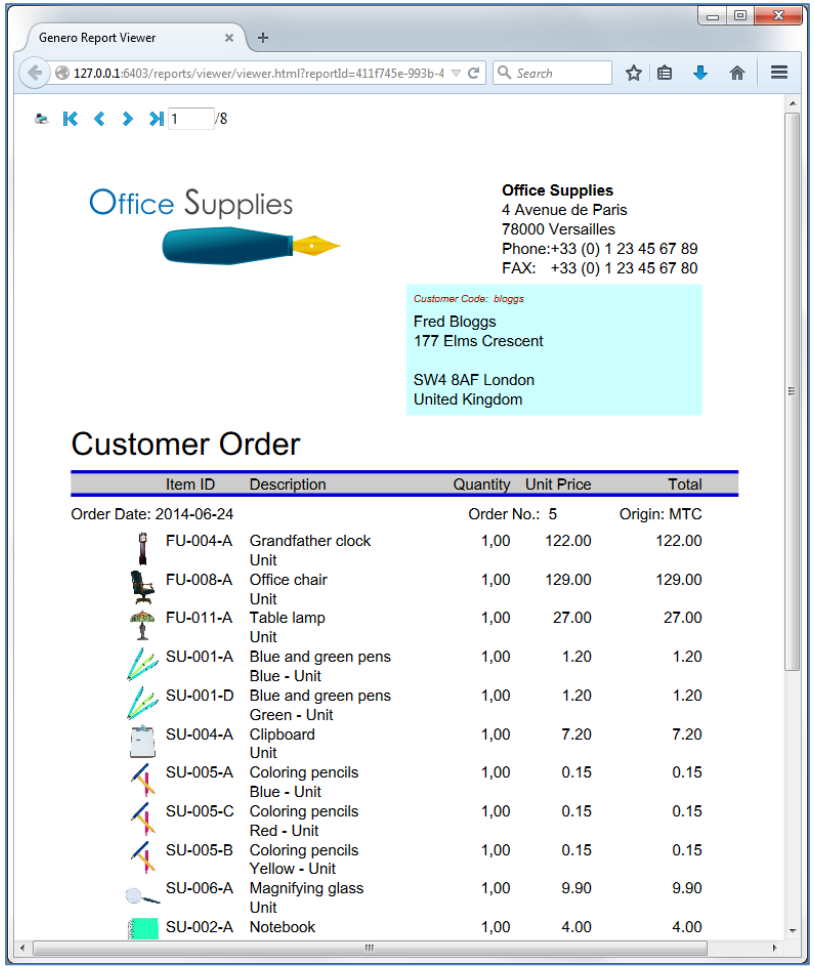
Changing output format

- Each output format can be configured by specific methods
  - Examples
    - excelRenderer.setMergePages(true);
    - imageRenderer.setFileType("png");
    - htmlRenderer.setEmbedImages(true);
    - pdfRenderer.setJPEGQuality(0.5);
- Report file opens with default associated desktop application

```
91        // open the file
92        File result = new File(outputFilename);
93        Desktop desktop = Desktop.getDesktop();
94        desktop.open(result);
```

Preview SVG report in browser

- Purpose
  - Render a report in SVG output format
  - Benefit from streaming for big reports
  - Preview the report over the Internet in Web browser
  - Get navigation and print options



Preview SVG report in browser

- Prerequisites
  - Web Server
    - Or use ‘NanoHTTPD.java’ as in the demo
  - Web browser
    - activated Java Script
    - capability to render SVG 1.2 Tiny
    - support for web fonts in the formats "ttf", "eot" or "woff"
  - Web Viewer application: viewer.html
    - hosted on the same server in the "../../viewer" directory

Preview SVG report in browser

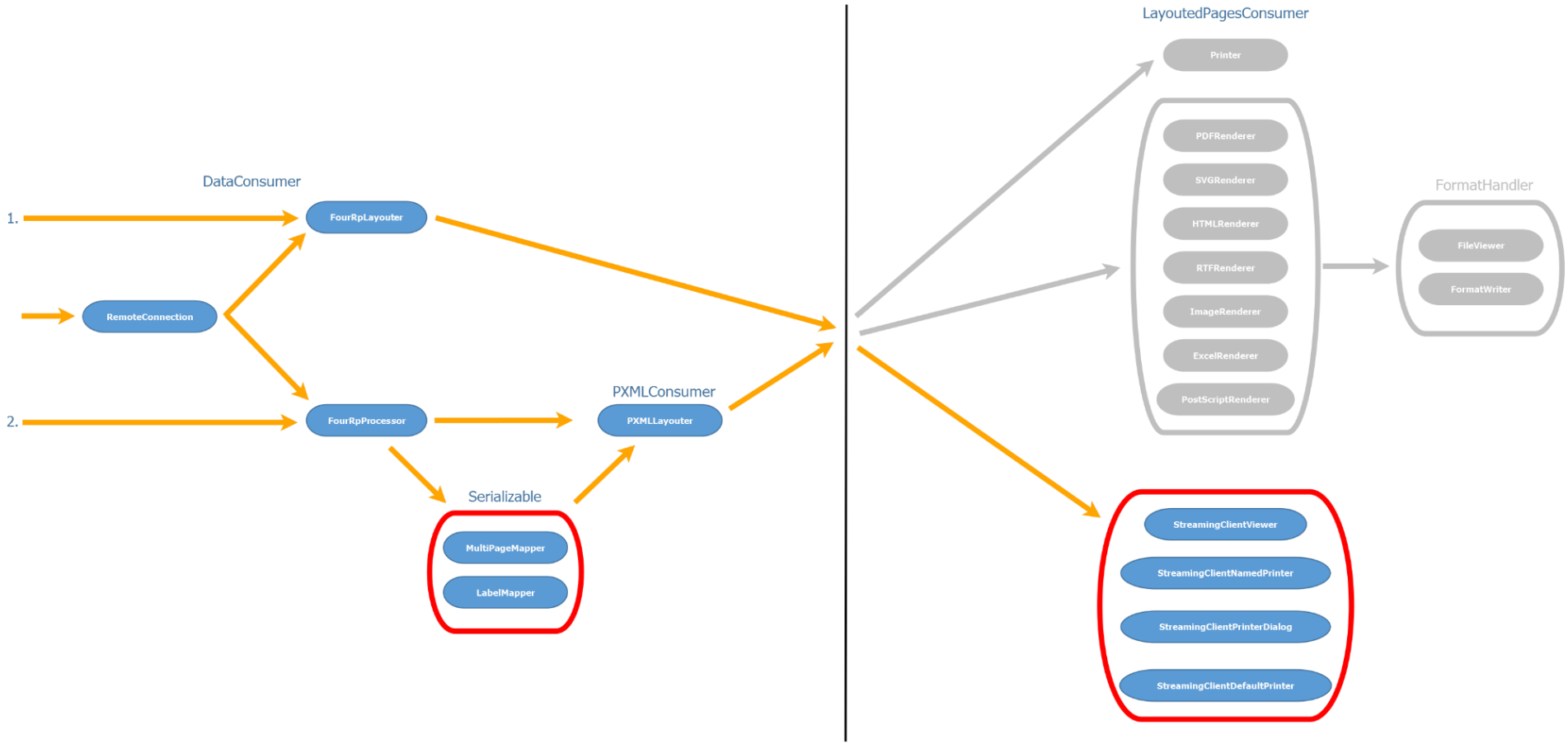
- Steps
  - Define a BrowserViewer object
  - Get Web Server's Root Directory
  - Set document directory
    - Ensure to be unique for each report
  - Set font directory
  - Browse for the generated files
    - URL should point to viewer.html & last item of document directory

```
390         {
391             BrowserViewer viewer = new BrowserViewer();
392             String uuid=java.util.UUID.randomUUID().toString();
393
394             viewer.setDocumentDirectory(webRootDirectory+"/reports/documents/"+uuid);
395             viewer.setFontDirectory(webRootDirectory+"/reports/documents/fonts");
396
397             if (Desktop.isDesktopSupported())
398             {
```

```
405                 Desktop desktop = Desktop.getDesktop();
406                 try
407                 {
408                     desktop.browse(new java.net.URI("http://127.0.0.1:6403/reports/viewer/viewer.html?reportId="+uuid));
409                 }
```

```
480         }
481         report.runFromJAXBObject(data);
```

Processing pipe (streaming client)



Using an intermediate XML datastream

- Does your existing application save reports for reprinting?
  - What if you want to reprint in a different output format?
- How long does it take to create the data?
  - Hit the database only once
- Each time you run the report do you have to reset configuration flags?
  - Printed flags don't have to be continually reset
- Each time you run the report do you have to enter new data?
  - Get all your test configurations into one XML datastream

XML data file best practices

- Create two applications
  - The first application generates the XML data file
  - The second application uses the XML data file to run reports
- Why?
  - Hit the database ONCE
  - Repeatedly run the report
    - Against the same XML data file
    - With different output settings

Output report to XML data file

- Specific method to produce the XML file
  - public void setRecordingFileName(String recordingFileName)
- Available in two classes
  - FourRpProcessor
  - FourRpLayouter
- Can be generated while running a report

```
500         //Generate XML data file
501         report.setRecordingFileName("my_data_file.xml");
502         report.runFromJAXBObject(data);
```

```
1  <?xml version="1.0" encoding="windows-1252"?>
2  <?PrettyPrinterStyleSheet  tee="true" writeToFile="jdebug0.xml"?>
3  <?AnyXMLToRTLInputStyleSheet?>
4  <?PrettyPrinterStyleSheet  tee="true" writeToFile="jdebug1.xml"?>
5  <?ScopeExtenderStyleSheet  systemId="file:/C:/Users/rw/Documents/My%20Genero%20Report%20Studio%20Files/samples/Reports/OrderReportJava/../OrderReport/OrderReport.
6  <?PrettyPrinterStyleSheet  tee="true" writeToFile="jdebug2.xml"?>
7  <?RTLStyleSheet  ignoreUnmatchedInput="true" styleURL="file:/C:/Users/rw/Documents/My%20Genero%20Report%20Studio%20Files/samples/Reports/OrderReportJava/../OrderR
8  <?PrettyPrinterStyleSheet  tee="true" writeToFile="jdebug3.xml"?>
9  <?BoxDecoratorStyleSheet  systemId="file:/C:/Users/rw/Documents/My%20Genero%20Report%20Studio%20Files/samples/Reports/OrderReportJava/../OrderReport/OrderReport.4
10 <?H1PxmlToPxmlStyleSheet  notUsed="true"?>
11 <?PrettyPrinterStyleSheet  tee="true" writeToFile="jdebug4.xml"?>
12 <?PxmlLayouter  pageWidth="a4width" pageHeight="a4length" topMargin="1.3cm" leftMargin="1.3cm" rightMargin="1.3cm" bottomMargin="1.3cm" markOverfullBoxesInDocumen
13 <?SVGWriterStyleSheet  qt422workarounds="true" fontFamilyWorkarounds="false"?>
14 <?SVGSplitterStyleSheet  writeToDirectory="C:\Users\rw\Documents\My Genero Report Studio Files\samples\Reports\OrderReportJava/../HTTPDaemon/reports/documents/6dd
15 <Report>
16     <Group>
17         <Group>
18             <Group>
19                 <OnEveryRow>
20                     <orderline.orders.orderid>5</orderline.orders.orderid>
21                     <orderline.orders.userid>bloggs</orderline.orders.userid>
22                     <orderline.orders.orderdate>2014-06-24</orderline.orders.orderdate>
23                     <orderline.orders.shipfirstname>Fred</orderline.orders.shipfirstname>
24                     <orderline.orders.shiplastname>Bloggs</orderline.orders.shiplastname>
25                     <orderline.orders.shipaddr1>177 Elms Crescent</orderline.orders.shipaddr1>
26                     <orderline.orders.shipaddr2 xxx0:nil="true" xmlns:xxx0="http://www.w3.org/2001/XMLSchema-instance"/>
27                     <orderline.orders.shipcity>London</orderline.orders.shipcity>
28                     <orderline.orders.shipstate xxx1:nil="true" xmlns:xxx1="http://www.w3.org/2001/XMLSchema-instance"/>
29                     <orderline.orders.shipzip>SW4 8AF</orderline.orders.shipzip>
30                     <orderline.orders.shipcountry>GBR</orderline.orders.shipcountry>
31                     <orderline.orders.billfirstname>Fred</orderline.orders.billfirstname>
32                     <orderline.orders.billlastname>Bloggs</orderline.orders.billlastname>
33                     <orderline.orders.billaddr1>177 Elms Crescent</orderline.orders.billaddr1>
34                     <orderline.orders.billaddr2 xxx2:nil="true" xmlns:xxx2="http://www.w3.org/2001/XMLSchema-instance"/>
35                     <orderline.orders.billcity>London</orderline.orders.billcity>
36                     <orderline.orders.billstate xxx3:nil="true" xmlns:xxx3="http://www.w3.org/2001/XMLSchema-instance"/>
37                     <orderline.orders.billzip>SW4 8AF</orderline.orders.billzip>
38                     <orderline.orders.billcountry>GBR</orderline.orders.billcountry>
39                     <orderline.orders.totalprice>1061.5</orderline.orders.totalprice>
40                     <orderline.orders.creditcard xxx4:nil="true" xmlns:xxx4="http://www.w3.org/2001/XMLSchema-instance"/>
41                     <orderline.orders.exprdate xxx5:nil="true" xmlns:xxx5="http://www.w3.org/2001/XMLSchema-instance"/>
42                     <orderline.orders.cardtype xxx6:nil="true" xmlns:xxx6="http://www.w3.org/2001/XMLSchema-instance"/>
43                     <orderline.orders.sourceapp>MTC</orderline.orders.sourceapp>
```

Produce report from XML data file

- The previously saved XML file is processed by the XML filter (the Genero Reporting Engine)
- Other configuration APIs can be called

```java
/**
 * RunXML.java
 */

import javax.xml.bind.JAXBException;
import java.io.IOException;
import org.xml.sax.SAXException;
import com.fourjs.pxml.standardpipe.runtimeapi.*;
import java.io.File;
import java.awt.Desktop;
import javax.xml.parsers.ParserConfigurationException;

public class RunXML {

  /**
   * @param args the command line arguments
   */
  public static void main(String[] args) throws JAXBException,
            IOException, SAXException, ParserConfigurationException
  {
    System.out.println("-- Run report from XML file --");
    String designFile;
    String outputFilename = "PDFoutput.pdf";
    if (args.length == 0) {
      designFile = "OrderReport.4rp";
    } else {
      designFile = args[0];
    }
    FormatHandler handler = new FormatWriter(outputFilename);
    PDFRenderer renderer = new PDFRenderer(handler);
    FourRpLayouter report = new FourRpLayouter(designFile, renderer);

    //Run report from XML file:
    report.runFromXML("my_data_file.xml");

    // open the generated file:
    File result = new File(outputFilename);
    Desktop desktop = Desktop.getDesktop();
    desktop.open(result);
  }

}
```


QUESTIONS ?

Exercises

- Exercise 1
  - Open the **'OrderReportJava' demo project**
    - Get familiar with Project Manager
- Exercise 2
  - Create a new project with the files provided ('Sales.java' source and 'SalesList.4rp' report design)
    - Create a new Java project
    - Rename (and reorganize) the project nodes
    - Add the files
    - Add an option to generate the XSD file on each compilation
    - Run the application
  - Modify the 'Sales.java' source code to generate another output format (XLS or RTF)

Exercise

- Exercise 3
  - Open the **'OrderReportJava' demo project** again
  - Modify the Java source code of 'OrderReportJava.java' to generate an XML data file
  - Run a report and check if the XML data file was generated
- Exercise 4 (optional)
  - Create a new Java project running a demo report from the XML data file
  - Write a new Java source using the XML data file as input (use 'Sales.java' as basis)
  - Use one of the report designs of the demo ('OrderReport.4rp')