



Pete Helgren
pete@valadd.com

Open Source Report Writing Tools for IBM i

Value Added Software, Inc
801.581.1154

18027 Cougar Bluff
San Antonio, TX 78258

Disclaimer

The original presentation looked at both Jasper Reports and BIRT. This presentation will focus on Jasper Reports. Anyone interested in BIRT?

Agenda

- ▶ Review the current “report writing” options
- ▶ Take a look at “traditional” RPG approaches
- ▶ Take a look at designing reports using report writing tools
- ▶ Take a look at integrating Open Source and RPG
- ▶ Take a look at deployment strategies

Report writing options

- ✓ Query/400 (or whatever it is called now)
- ✓ DB2 Web Query
- ✓ Sequel
- ✓ NGS - IQ
- ✓ Crystal Reports
- ✓ Business Objects (SAP)
- ✓ Jasper Reports
- ✓ BIRT
- ✓ *Roll your own – Open Source



Why use Open Source?

Affordable (like, um, free)

IBM i is the perfect environment for Open Source

ILE RPG easily accommodates FOSS

Low investment (time, human CPU cycles and money)



Why NOT use Open Source?



Learning curve (+ for some, - for others)

Licensing Issues – Need to clearly understand what you can and can't do with FOSS.

Support Issues – Most support is by user forum but many commercial offerings are available.

Possible security issues like “Heartbleed”

Performance and “stack” stability – not really an issue.
(maybe upgrade from that model 270/800!)



Traditional RPG Approaches



- ✓ RPG writing to an External Print File
- ✓ With CL Wrapper --> PDF (V6R1)
- ✓ Add the iText wrapper



“Traditional” RPG

(code example – Sample1.rpgle)



```
FOEMPRPT  O    E                PRINTER  OFLIND(Overflow)
  D Overflow                S                n
  D SQLStmnt                S                2048A  Varying
  D PrepSQLGetRS            PR                5A
  D FetchNextRow            PR                5A
  D CloseSQLCursor          PR                5A
  D WriteLine                PR
  D WriteHeader              PR
  D SQLSuccess              S                5A  Inz('00000')
C/free
```

```
// Prep the SQL statement
If PrepSQLGetRS() = SQLSuccess;
  // Read the employee file
  WriteHeader();
  Dow FetchNextRow() = SQLSuccess;
    WriteLine();
  enddo;
  // Close the cursor opened in prep
  CloseSQLCursor();
endif;

*inlr = *on;
/end-free
```




Sample1.rpgle (cont)

```
p*****
p* Prepare and open the SQL Statement
p*****
P PrepSQLGetRS      B
D PrepSQLGetRS      PI              5A
D/free
  SQLStmnt = 'select ' +
    'emfnam || ' ' ' || substr(emmnam,1,1) || ' ' ' || emlnam as ename, ' +
    'emadd1,emcity, emst, emzip1,emzip2 from employee ' +
    'order by emlnam,emfnam';

EXEC SQL
  PREPARE S1 FROM :SQLStmnt;

EXEC SQL
  DECLARE C1 CURSOR FOR S1;

EXEC SQL
  OPEN C1;

  RETURN SQLSTT;
/end-free
P PrepSQLGetRS      E
```



Sample1.rpgle (cont)

```
p*****  
p* Fetch a row at a time  
p*****  
P FetchNextRow      B  
D FetchNextRow      PI              5A  
/free  
EXEC SQL  
    FETCH FROM C1 INTO :ename, :emadd1, :emcity, :emst, :emzip1, :emzip2 ;  
  
    RETURN SQLSTT;  
/end-free  
P FetchNextRow      E
```



Sample1.rpgle (cont)

```
C*****  
C* Write the header  
C*****  
P WriteHeader      B  
D WriteHeader      PI  
/free  
    write header;  
/end-free  
P WriteHeader      E
```



Sample1.rpgle (cont)

```
C*****  
C* Write the detail  
C*****  
P WriteLine          B  
D WriteLine          PI  
  /free  
    write detail;  
    if Overflow ;  
      write header;  
      reset Overflow;  
    endif;  
  /end-free  
P WriteLine          E
```



Sample1.rpgle (cont)

```
C*****  
C* Close the Open Cursor  
C*****  
P CloseSQLCursor  B  
D CloseSQLCursor  PI          5A  
/free  
EXEC SQL  
CLOSE C1;  
  
RETURN SQLSTT;  
/end-free  
P CloseSQLCursor  E
```



Sample1 spooled file (cont)



PROG: EMP.RPT
DATE: 5/01/10
TIME: 17:17:09

PAGE

Sample Employee Report

Name	Address	City	State	Zip	+4
OCHAOMPUNGMOO	ADWEN	085 DEKALB AVENUE#2	SLCSEY CITY	UT	84117 0000
RUBARS	H ASSRENA	068 KENSINGTON AVE.	SLCSEY CITY	UT	84117 0000
VEVEON	Y BALL	045 SO. HARRISON STREET.	SLCT ORANGE	UT	84117 0000
RAENOLDU	BANAJON	097 FRANKLIN STREET	SLCSEY CITY	UT	84117 0000
JANNEFAR	BESSEG	058 SMITH ROAD	SLCSIPPANY	UT	84117 0000
PHELEP	D BOERD	069 VERSIDE DR.	SLCNFORD	UT	84117 0000
OKAAM	BOLUGON	087 KYLINE DRIVE	SLCSEY CITY	UT	84117 0000
FRONCAS	BORNS	081 39TH ST	SLCON CITY	UT	84117 0000
MORY LOU	BOSELA	022 OAK AVENUE	SLCWOOD	UT	84117 0000
VECSUREO	BRESS	075 WASHINGTON AVENUE	SLCT ORANGE	UT	84117 0000
LESO	BRUWN	028 HOLMDEL ROAD	SLC	UT	84117 0000
JUSAPH	CERELLU	037 SHIPPEN STREET	SLCHAWKEN	UT	84117 0000
RUBARS	COSSELLU	053 ALES AVE	SLCSEY CITY	UT	84117 0000



Sample1C (override to PDF)



```
PGM
OVRPRTF      FILE(OEMPRPT) DEVTYP(*AFPDS) SPLFNAME(SAMPLE) +
              TOSTMF('\Reports\output') WSCST(*PDF)

call        sample1

endpgm
```



Sample1C output



PROG: EMP.RPT
DATE: 5/02/10
TIME: 13:00:30

Sample Employee Report

PAGE 1

Name	Address	City	State	Zip	+4
OCHAOMPUNGMOO	ADWEN	085 DEKALB AVENUE#2	SLCSEY CITY	UT	84117 0000
RUBARS	H ASSRENA	068 KENSINGTON AVE.	SLCSEY CITY	UT	84117 0000
VEVEON	Y BALL	045 SO. HARRISON STREET.	SLCT ORANGE	UT	84117 0000
RAENOLDU	BANAJON	097 FRANKLIN STREET	SLCSEY CITY	UT	84117 0000
JANNEFAR	BESSEG	058 SMITH ROAD	SLCSIPPANY	UT	84117 0000
PHELEP	D BOERD	069 VERSIDE DR.	SLCNFORD	UT	84117 0000
OKAAM	BOLUGON	087 KYLINE DRIVE	SLCSEY CITY	UT	84117 0000
FRONCAS	BORNS	081 39TH ST	SLCON CITY	UT	84117 0000
MORY LOU	BOSELA	022 OAK AVENUE	SLCWOOD	UT	84117 0000



Sample2.rpgle (using iText)



```
D SQLStmnt          S           2048A  Varying
D PrepSQLGetRS      PR           5A
D FetchNextRow      PR           5A
D CloseSQLCursor    PR           5A
D WriteLine         PR
D WriteHeader       PR
D SQLSuccess        S           5A   Inz ('00000')
D lHeading          s           like(jString)
D lColumns          s           like(jString)
D lReportName       s           like(jString)
D lFileName         s           like(jString)
D iTextReport       s           like(jReportView)
D lColView          s           like(jReportColumnView)
D counter          S           10I 0

D ename             s           40A
D emadd1            s           30A
D emcity            s           25A
D emst              s           2A
D emzip1            S           5S 0
D emzip2            S           5S 0

D colEmp            S           like(jString)
D colAddr           S           like(jString)
D colCity           S           like(jString)
D colState          S           like(jString)
D colZip1           S           like(jString)
D colZip2           S           like(jString)
D results           S           N
D lRows             s           10I 0
```

Sample2.rpgle (cont)

```
C/free
  rre_begin_object_group(16);
  // Initial Object creation
  lHeading = new_String('Sample Report using iText and RPG');
  lColumns = new_jArrayList();
  lReportName = new_String('Sample iText Report (Sample 2) ');
  lFileName = new_String('/reports/output/Sample_2_iText_Report.pdf');

  WriteHeader();

  iTextReport = new_RREReportView(lHeading:lColumns:lReportName:lFileName);

  rreRV_initialize(iTextReport); // Creates the report shell
  counter = 1;
  // Prep the SQL statement
  If PrepSQLGetRS() = SQLSuccess;
    // Read the employee file
    Dow FetchNextRow() = SQLSuccess;

      counter = counter + 1;

      WriteLine();

    enddo;
    // Close the cursor opened in prep
    CloseSQLCursor();
    lrows = rreRV_finalize(iTextReport:counter);
  endif;

  *inlr = *on;
/end-free
```



Sample2.rpgle (cont)



```
C*****
C* Write the header
C*****
P WriteHeader      B
D WriteHeader      PI
/free
  // Not so much a write as it is a CREATE
  // Before creating the ReportView, add heading columns to the array
  // The pattern here is to construct and then add to the ArrayList
  //Column Headings MUST Total to 100%
  colEmp = new_String('Employee Name');
  colAddr= new_String('Address');
  colCity = new_String('City');
  colState= new_String('State');
  colZip1 = new_String('Zip');
  colZip2= new_String('Zip + 4');

  lColView = new_RREReportColumnView(colEmp:30);
    rre_jArrayList_add(lColumns:lColView);

  lColView = new_RREReportColumnView(colAddr:20);
    rre_jArrayList_add(lColumns:lColView);

  lColView = new_RREReportColumnView(colCity:20);
    rre_jArrayList_add(lColumns:lColView);

  lColView = new_RREReportColumnView(colState:10);
    rre_jArrayList_add(lColumns:lColView);

  lColView = new_RREReportColumnView(colZip1:10);
    rre_jArrayList_add(lColumns:lColView);

  lColView = new_RREReportColumnView(colZip2:10);
    rre_jArrayList_add(lColumns:lColView);

/end-free
P WriteHeader      E
```



Sample2.rpgle (cont)

```
C*****  
C* Write the detail  
C*****  
P WriteLine          B  
D WriteLine          PI  
/free  
  if %REM(counter: 2) = 1;  
    RRERV_SetGrayFill(iTextReport:.9);  
  endif;  
  
  rreRV_addCell(iTextReport:ename);  
  rreRV_addCell(iTextReport:emadd1);  
  rreRV_addCell(iTextReport:emcity);  
  rreRV_addCell(iTextReport:emst);  
  rreRV_addCell(iTextReport:%char(emzip1));  
  rreRV_addCell(iTextReport:%char(emzip2));  
  
  if %REM(counter: 2) = 1;  
    RRERV_SetGrayFill(iTextReport:1);  
  endif;  
  
/end-free  
P WriteLine          E
```



Sample2 output - pdf

Sample Report using iText and RPG

Employee Name	Address	City	State	Zip	Zip + 4
OCHAOMPUNGMOO ADWEN	085 DEKALB AVENUE#2	SLCSEY CITY	UT	84117	0
RUBARS H ASSRENA	068 KENSINGTON AVE.	SLCSEY CITY	UT	84117	0
VEVEON Y BALL	045 SO. HARRISON STREET.	SLCT ORANGE	UT	84117	0
RAENOLDU BANAJON	097 FRANKLIN STREET	SLCSEY CITY	UT	84117	0
JANNEFAR BESSEG	058 SMITH ROAD	SLCSIPPANY	UT	84117	0
PHELEP D BOERD	069 VERSIDE DR.	SLCNFORD	UT	84117	0
OKAAM BOLUGON	087 KYLINE DRIVE	SLCSEY CITY	UT	84117	0
FRONCAS BORNES	081 39TH ST	SLCON CITY	UT	84117	0
MORY LOU BOSELA	022 OAK AVENUE	SLCWOOD	UT	84117	0
VECSUREO BRESS	075 WASHINGTON AVENUE	SLCT ORANGE	UT	84117	0
LESO BRUWN	028 HOLMDEL ROAD	SLC	UT	84117	0
JUSAPH CERELLU	037 SHIPPEN STREET	SLCHAWKEN	UT	84117	0
RUBARS COSSELLU	053 ALES AVE	SLCSEY CITY	UT	84117	0
WOYNA CRAAD	101 COUSE ROAD	SLCTUNE	UT	84117	0

Questions?

Three “native i” approaches:

“Traditional” spoolfile

“Traditional” spoolfile with override to PDF
Use iText (Java) through wrapping with RPG

Designing Reports

The purpose of this session was not to introduce you to report design, just how to run those designed reports from RPG. However, a quick tutorial will get you oriented.

There are MANY tutorials and design guides on the web for Jasper and iReport.

Remember: RRE can handle overriding your connection to DB2 for i. So you can develop with one DB and deploy on another. You could even use a MySQL or MSSQL DB to develop and then deploy to I as long as your table/column references don't change.

Designing Reports (cont)

Jasper uses iReport for design (the Eclipse plugin is under re-development as Jasper Studio).

By the way: RRE is *currently* using JasperReports version 4.5.0 (iReport 4.5.0). Make sure you have designers that are compatible with the correct version otherwise running the reports can get ugly.

Basic Setup

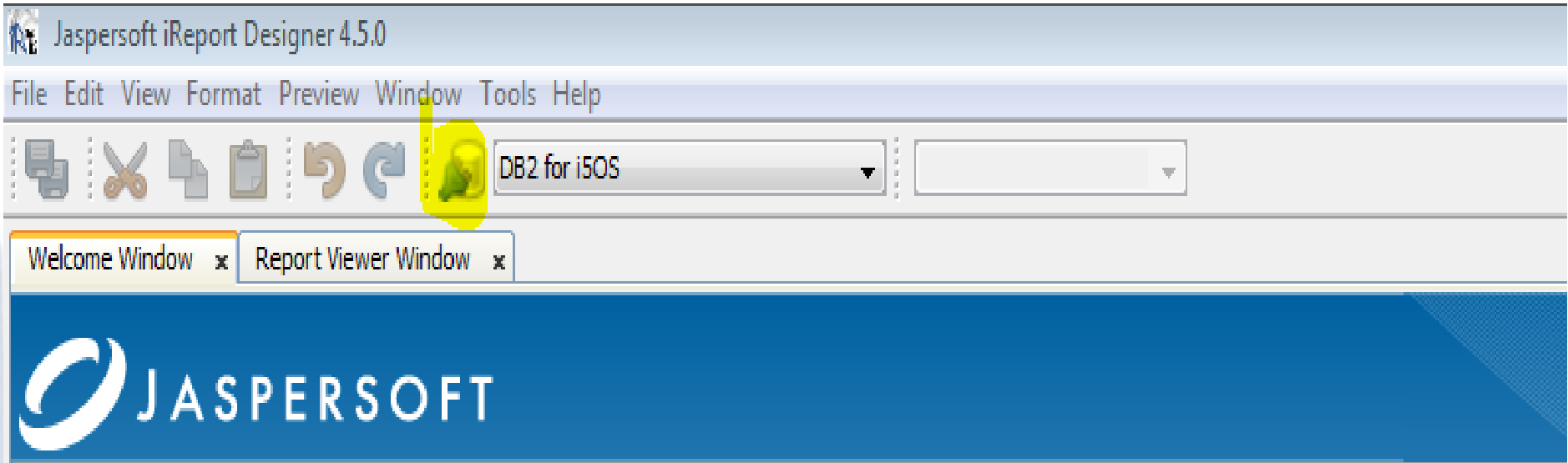
Download and install iReport

Install the jt400.jar (JDBC driver) into any location.
(get it from <http://jt400.sourceforge.net/>)

Then from within iReport choose the Tools-->options and then patiently wait for the all the tabs to appear. Select the “classpath” tab and then “Add” the jar location to the classpath.

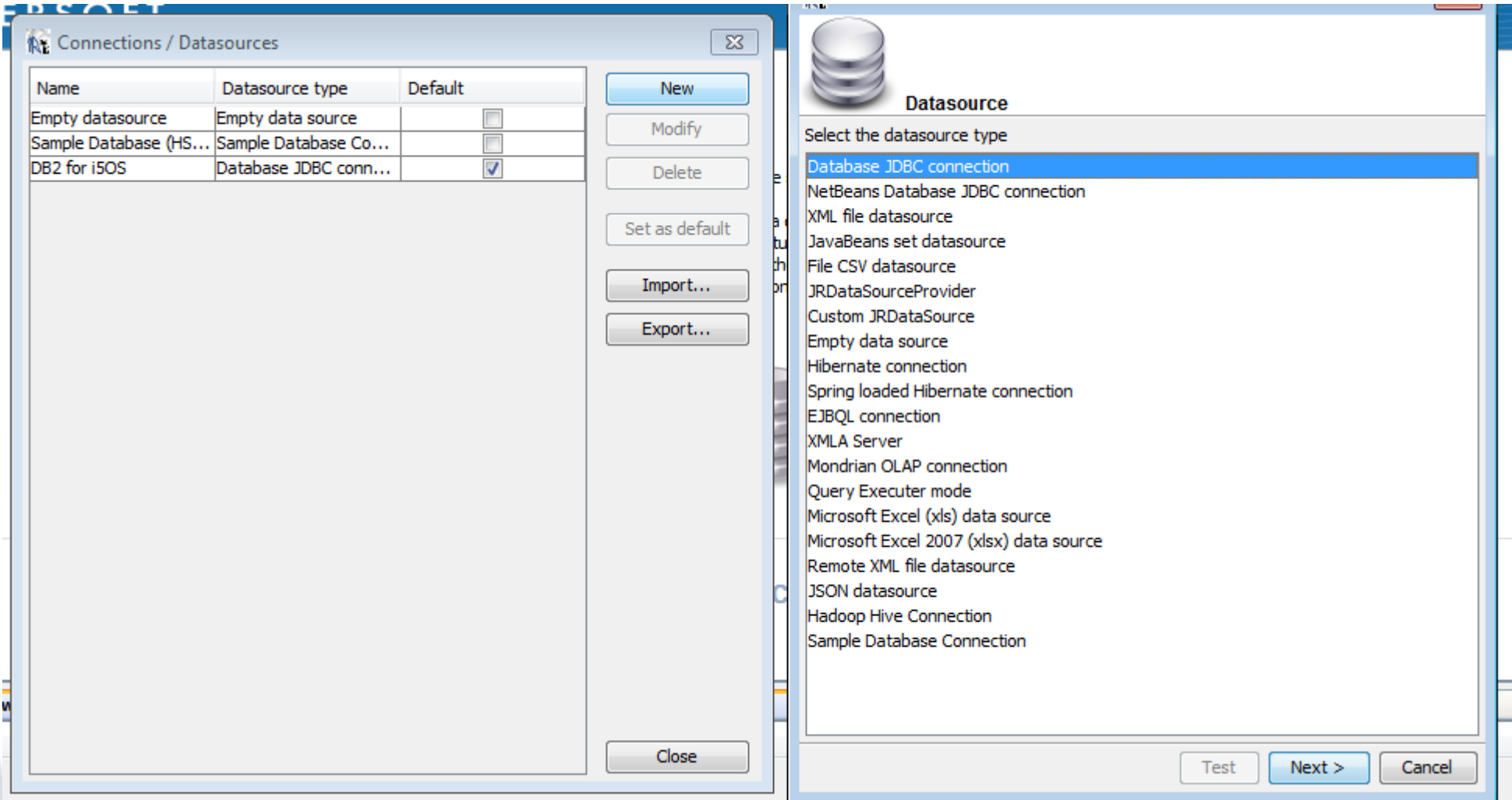
You'll need the JDBC drivers so you can access the DB2 for I databases

Start iReport and create a data source



Click on the database connection icon (highlighted above)

Start iReport and create a data source



The screenshot shows the 'Connections / Datasources' window in iReport. The window is divided into two panes. The left pane contains a table with the following data:

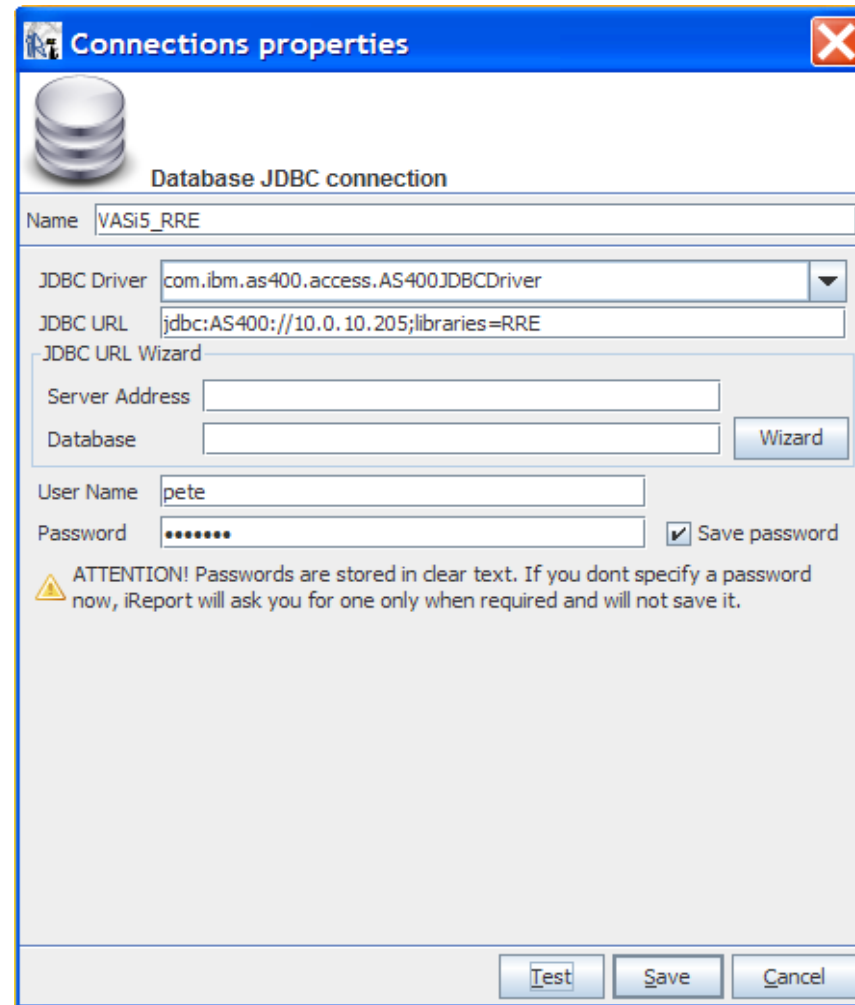
Name	Datasource type	Default
Empty datasource	Empty data source	<input type="checkbox"/>
Sample Database (HS...)	Sample Database Co...	<input type="checkbox"/>
DB2 for i5OS	Database JDBC conn...	<input checked="" type="checkbox"/>

Below the table are buttons for 'New', 'Modify', 'Delete', 'Set as default', 'Import...', and 'Export...'. A 'Close' button is at the bottom right of the left pane.

The right pane is titled 'Datasource' and shows a list of available datasource types. The 'Database JDBC connection' option is selected and highlighted in blue. The list includes: Database JDBC connection, NetBeans Database JDBC connection, XML file datasource, JavaBeans set datasource, File CSV datasource, JRDataSourceProvider, Custom JRDataSource, Empty data source, Hibernate connection, Spring loaded Hibernate connection, EJBQL connection, XMLA Server, Mondrian OLAP connection, Query Executer mode, Microsoft Excel (xls) data source, Microsoft Excel 2007 (xlsx) data source, Remote XML file datasource, JSON datasource, Hadoop Hive Connection, and Sample Database Connection. At the bottom of the right pane are buttons for 'Test', 'Next >', and 'Cancel'.

In the Connections/Datasources window click on "New" and then select "Database JDBC Connection"

Start iReport and create a datasource



Connections properties

Database JDBC connection

Name: VASI5_RRE

JDBC Driver: com.ibm.as400.access.AS400JDBCDriver

JDBC URL: jdbc:AS400://10.0.10.205;libraries=RRE

JDBC URL Wizard

Server Address: []

Database: [] Wizard

User Name: pete

Password: ***** Save password

ATTENTION! Passwords are stored in clear text. If you dont specify a password now, iReport will ask you for one only when required and will not save it.

Test Save Cancel

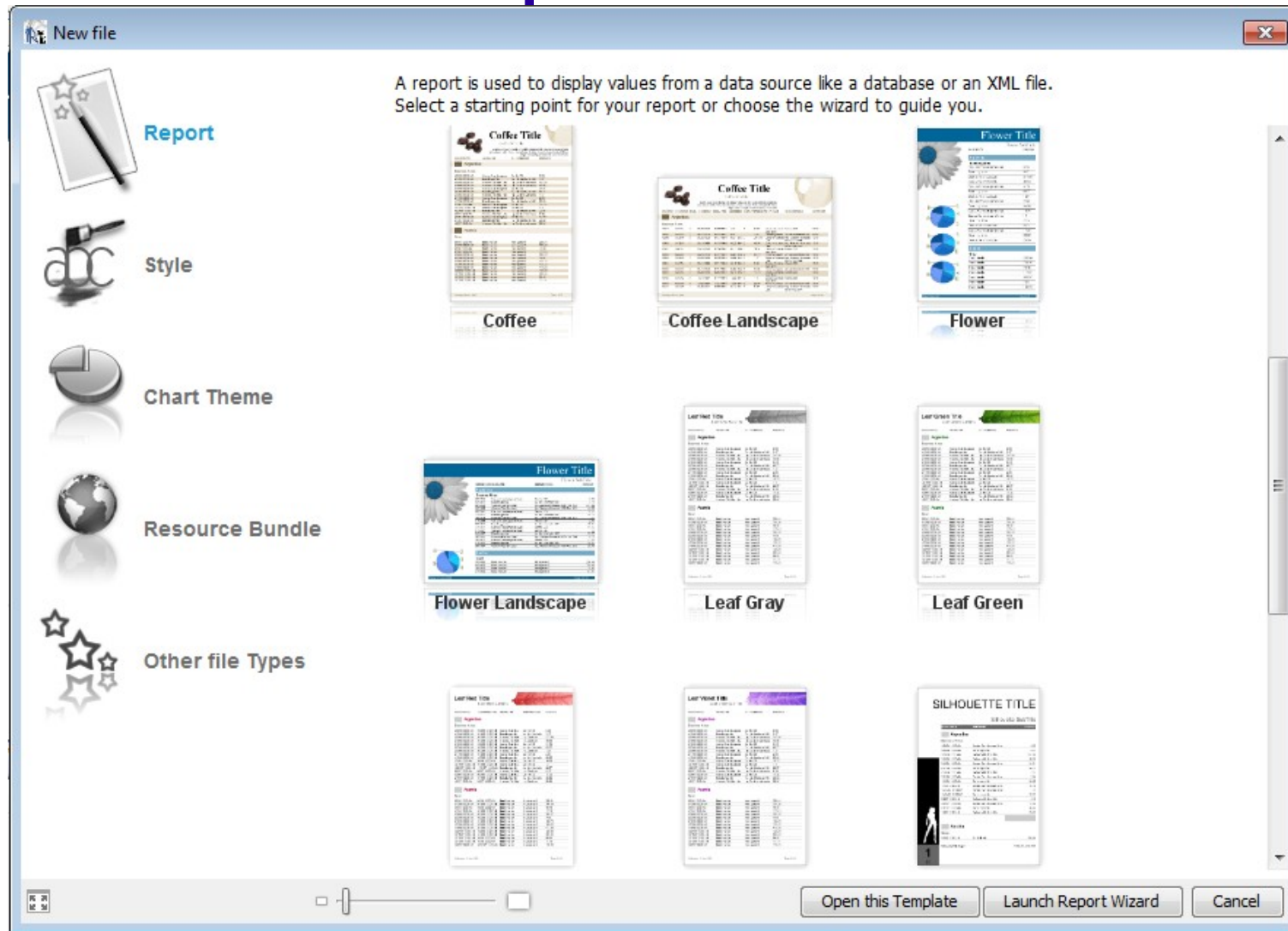
Complete the dialog as indicated – try the test button to make sure all is well

Jasper – Report Wizard

Generally the best approach to get started is to use the wizard in Jasper Reports. It'll step you through the options. SQL comes first.

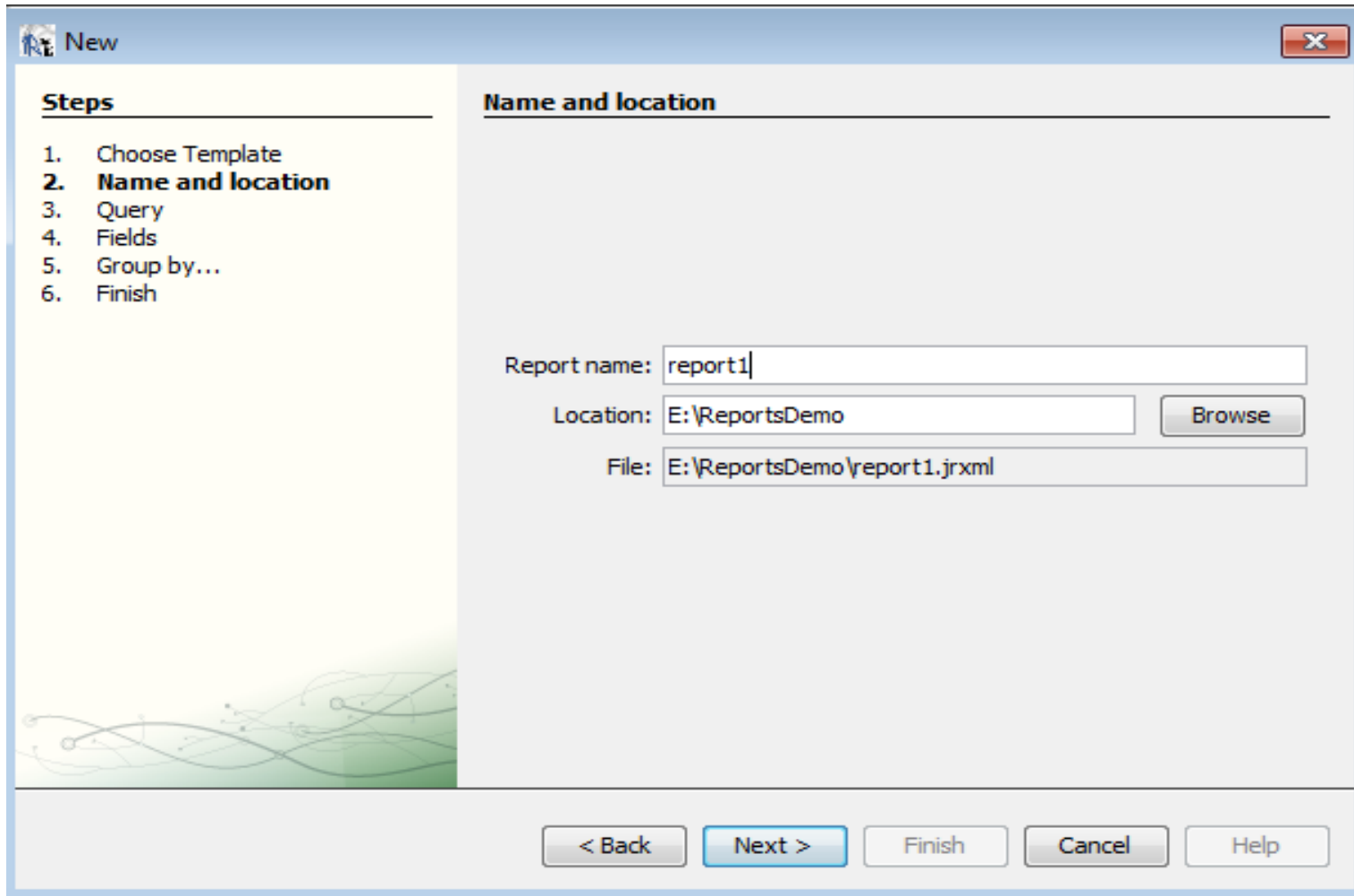
If you are proficient at SQL or already have the SQL in some form, you can cut and paste the SQL Statement into the SQL window. Or you can use the “Design Query” button which will step you through creating the SQL:

Report Wizard



File--> new will start the wizard. The frame may take a few seconds to load But it will then display the available templates. Choose one and then click on Launch Report Wizard

Report Wizard



The screenshot shows a 'New' dialog box for the Report Wizard. The 'Steps' pane on the left lists: 1. Choose Template, 2. **Name and location**, 3. Query, 4. Fields, 5. Group by..., 6. Finish. The main area is titled 'Name and location' and contains three input fields: 'Report name:' with the text 'report1', 'Location:' with 'E:\ReportsDemo' and a 'Browse' button, and 'File:' with 'E:\ReportsDemo\report1.jrxml'. At the bottom are buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

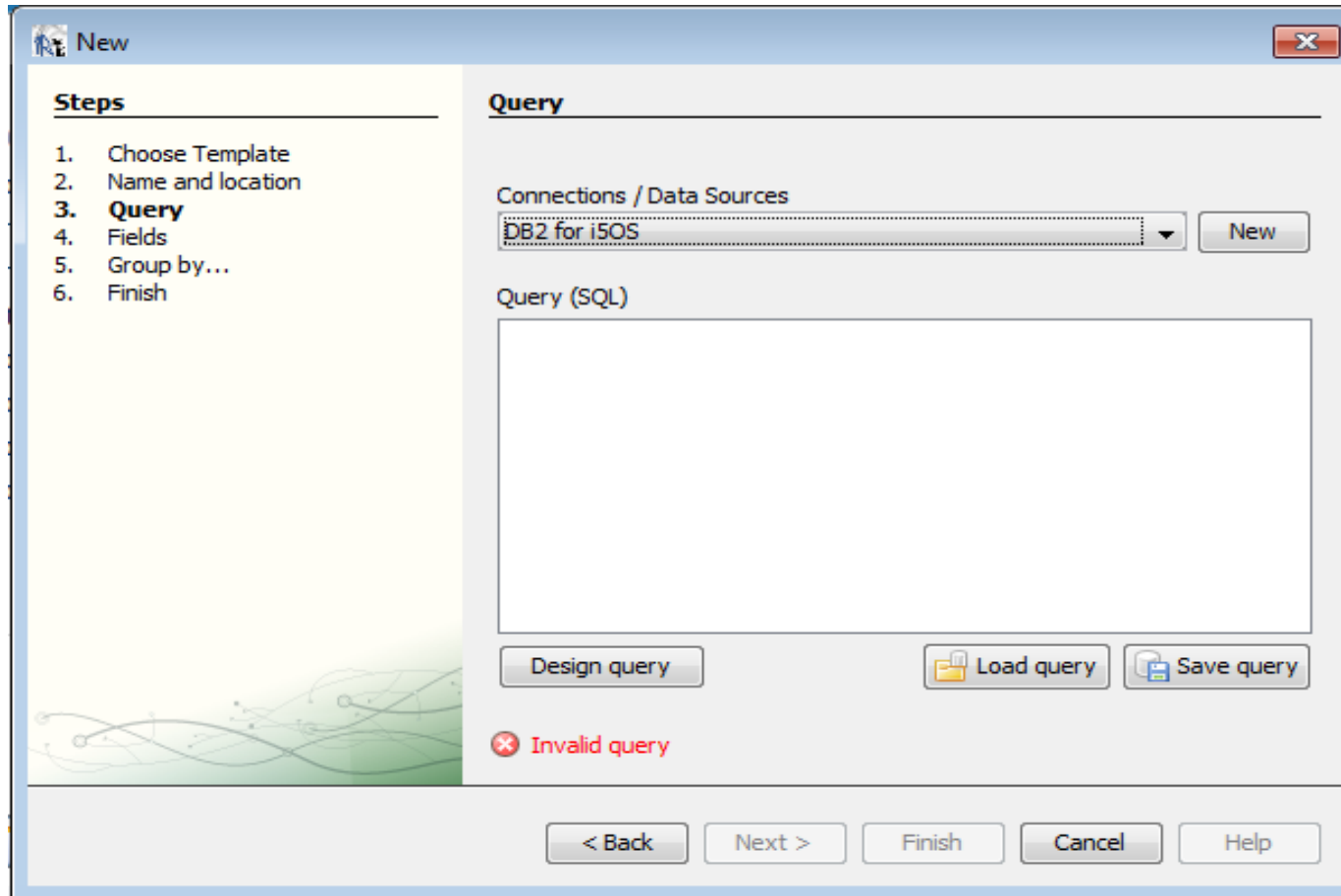
Give the report a file name and a folder to save it in. Then click next.

Build a query

The basic challenge of any report writer/designer is that the end user **usually** needs to know something about the database schema and relationships in order to create valid queries.

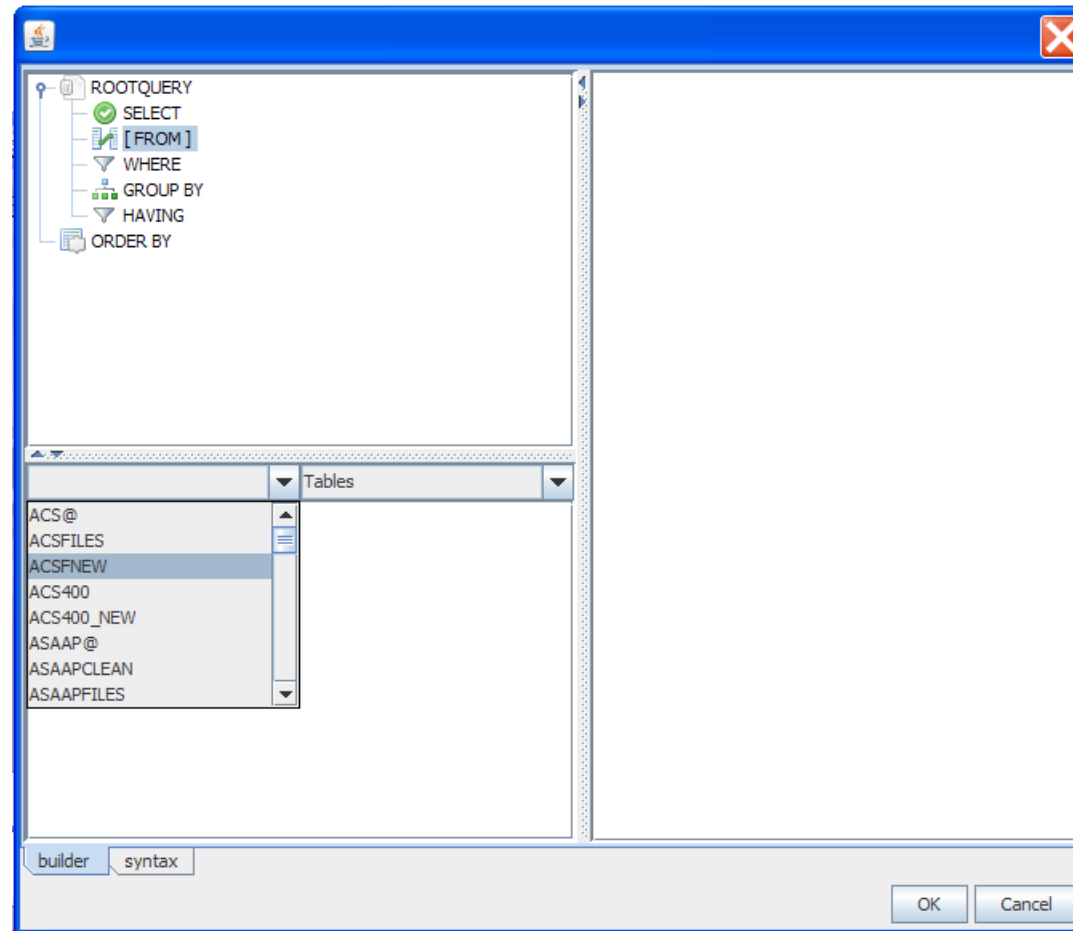
JasperReports is no exception. You still need to know your SQL!

Design a Query



If you need to design a query click the “Design Query” button. If you have already designed a query then select it from the “Load query” button.

Design Query Wizard



Report Wizard - SQL

iReport Wizard [Close]

Steps

- 1. Query
- 2. Fields selection
- 3. Group by...
- 4. Layout
- 5. Finish

Step 1: Specify the query to retrieve report fields

Use the following template...

None

Connections/Data Sources

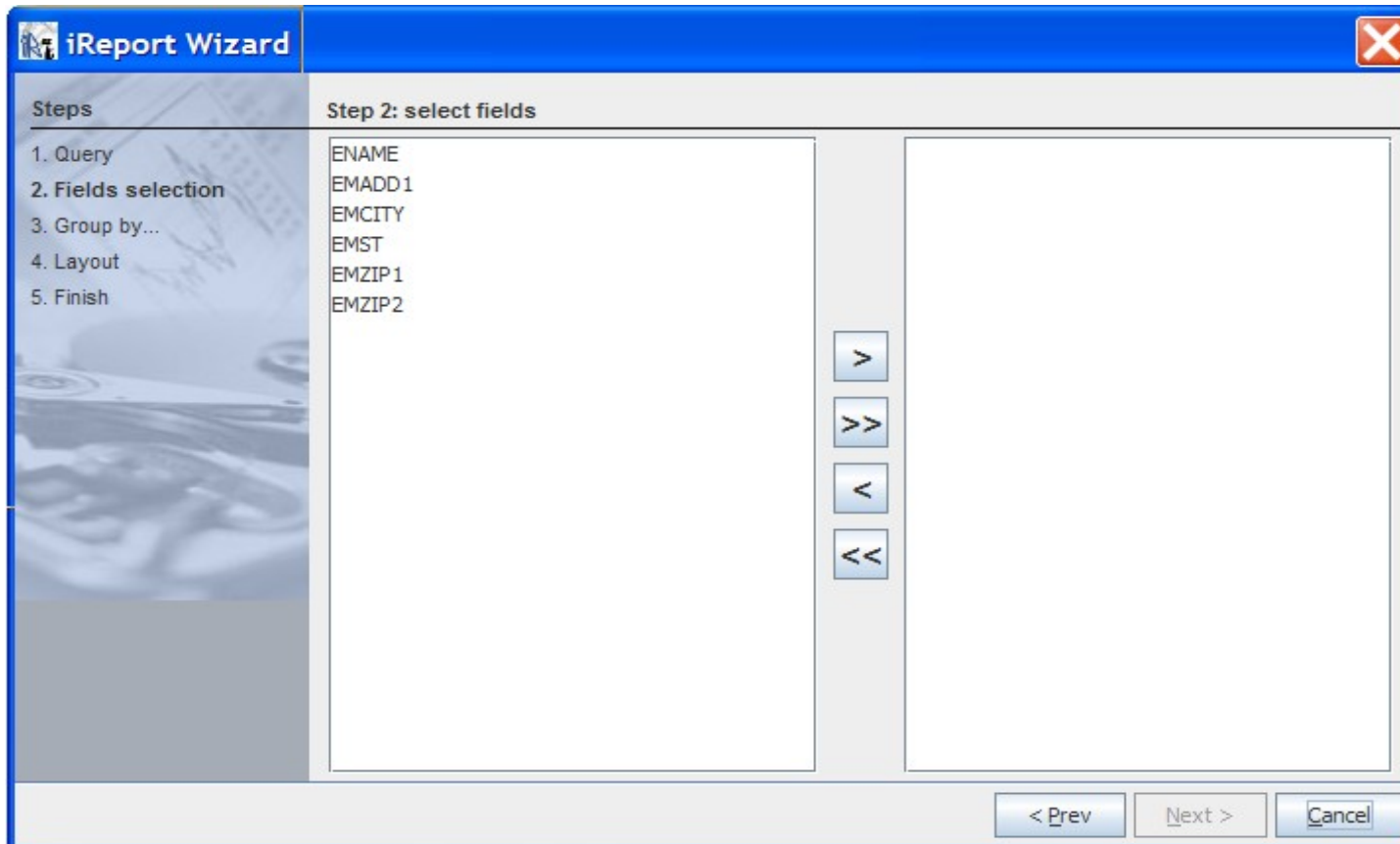
RRE Demo [New]

SQL query

```
select trim(emfnam) || ' ' || trim(emlnam) as ename, emadd1, emcity,
emst, emzip1, emzip2
FROM employee
order by emlnam, emfnam
```

Design qu... [Load Query] [Save Query] [Prev] [Next >] [Cancel]

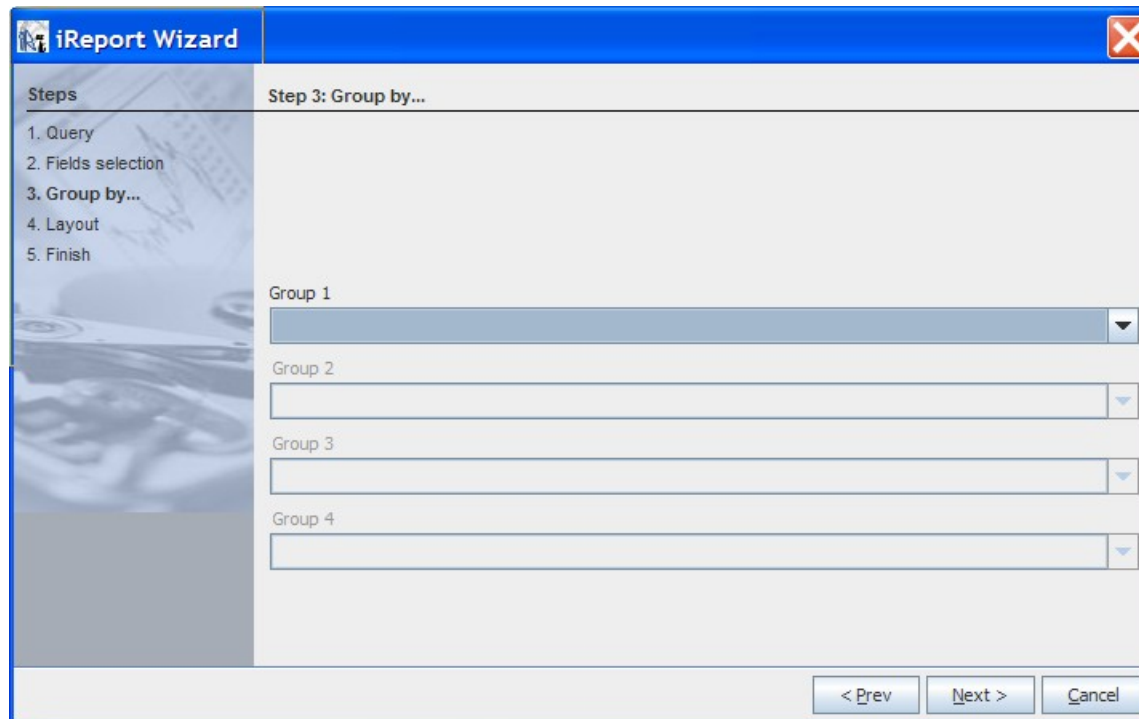
Report Wizard – Select Fields



The screenshot shows the 'iReport Wizard' window at Step 2: select fields. The window has a blue title bar with the text 'iReport Wizard' and a close button. On the left, a 'Steps' sidebar lists five steps: 1. Query, 2. Fields selection (highlighted), 3. Group by..., 4. Layout, and 5. Finish. The main area is split into two panes. The left pane contains a list of fields: ENAME, EMADD1, EMCITY, EMST, EMZIP1, and EMZIP2. The right pane is empty. Between the panes are four navigation buttons: a single right arrow (>), a double right arrow (>>), a single left arrow (<), and a double left arrow (<<). At the bottom right, there are three buttons: '< Prev', 'Next >', and 'Cancel'.

Report Wizard – Grouping

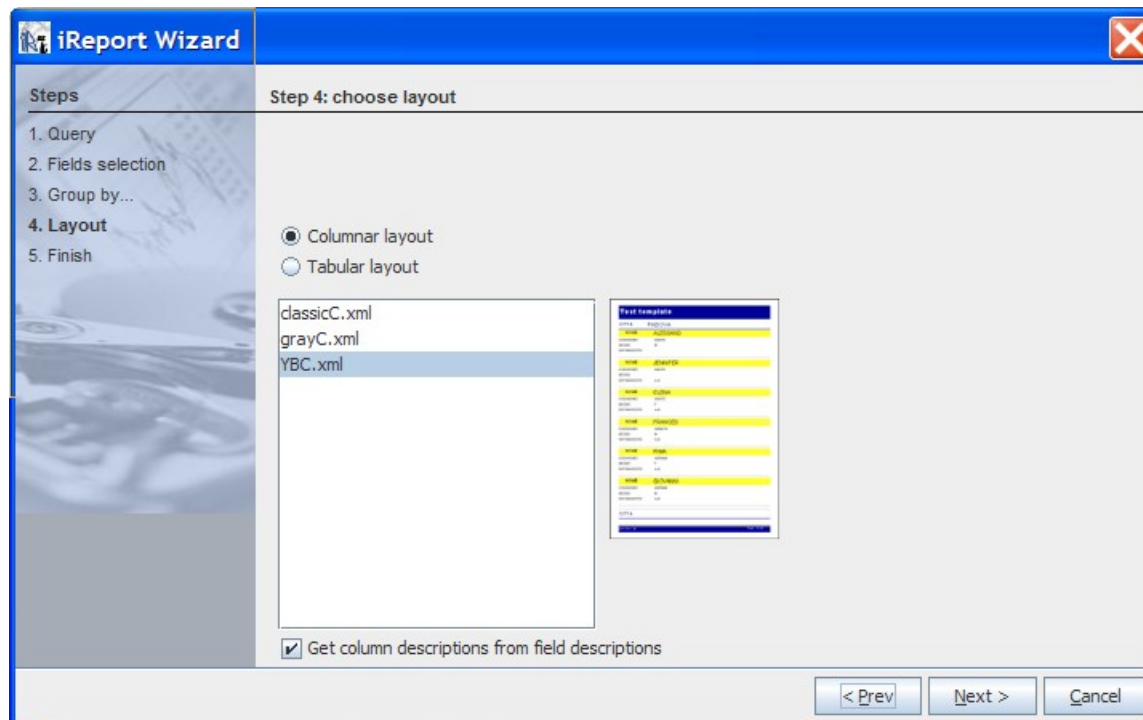
If you have aggregation (count, min, max, etc) functions or want to break at certain groups, then add a group by clause to the SQL



The screenshot shows the 'iReport Wizard' window at 'Step 3: Group by...'. On the left, a 'Steps' sidebar lists: 1. Query, 2. Fields selection, 3. Group by... (highlighted), 4. Layout, and 5. Finish. The main area contains four 'Group' sections, each with a dropdown menu. At the bottom, there are '< Prev', 'Next >', and 'Cancel' buttons.

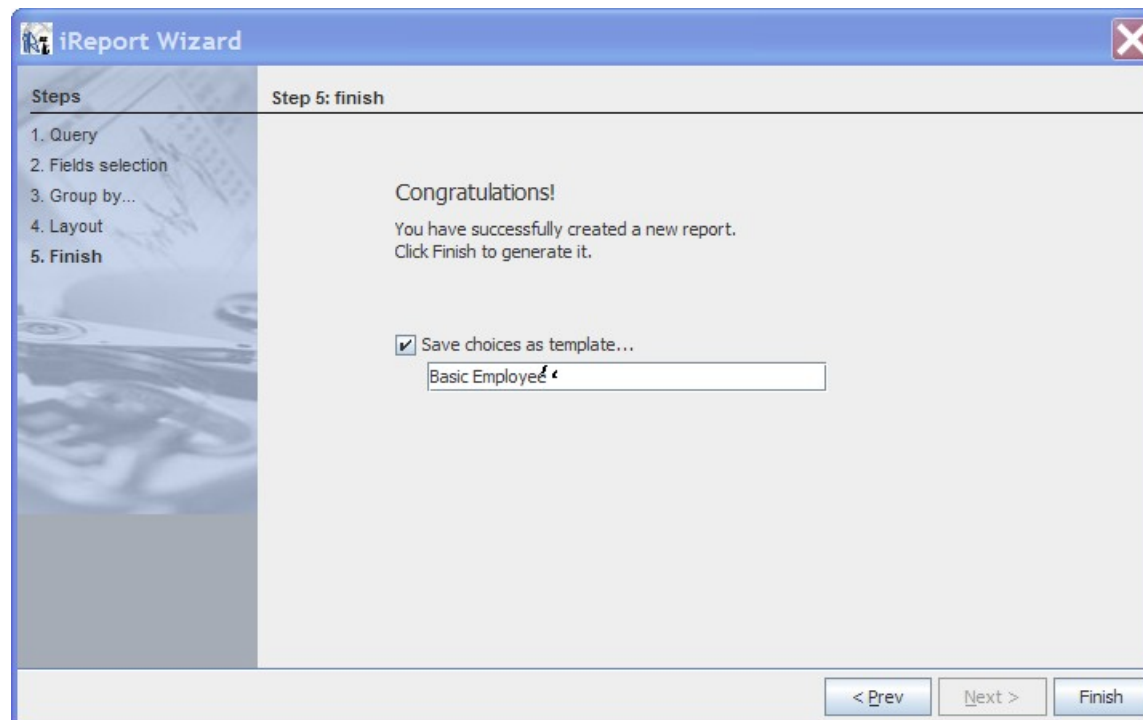
Report Wizard – Choose layout

There are several to choose from. The “classic” columnar report (more like a “form”) or a tabular layout (multiple rows)



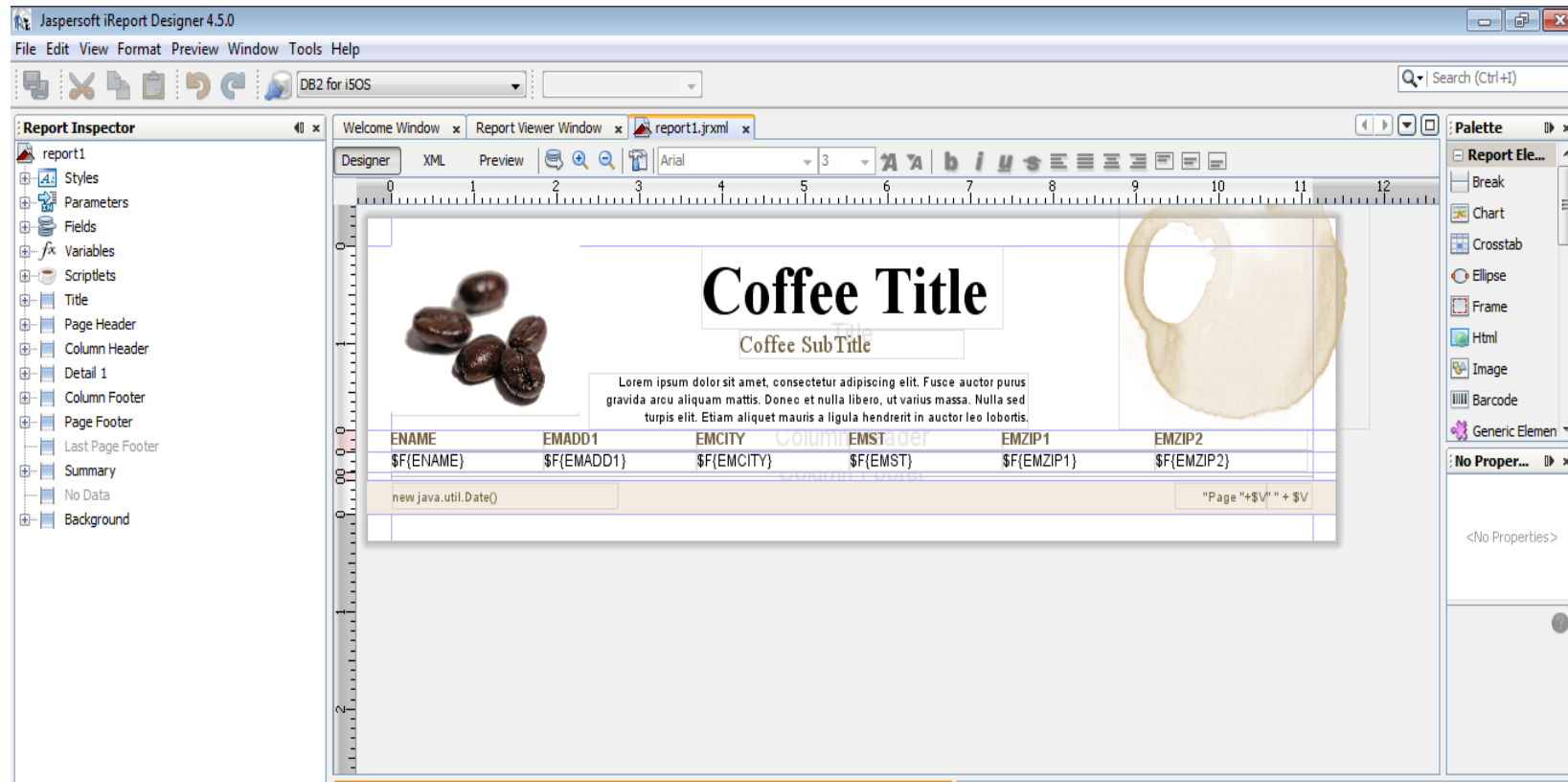
Report Wizard - Finish

One useful option is to save the report as a “template” which you can use to retrieve settings in the first step of using the wizard in the future. This is handy for saving complex queries that may be reused.



Jasper Report - modify

The wizard will give you a basic form so you will probably want to modify the report title and column headings if nothing else.



Jaspersoft iReport Designer 4.5.0

File Edit View Format Preview Window Tools Help

DB2 for iSOS

Search (Ctrl+F)

Report Inspector

- report1
 - Styles
 - Parameters
 - Fields
 - Variables
 - Scriptlets
 - Title
 - Page Header
 - Column Header
 - Detail 1
 - Column Footer
 - Page Footer
 - Last Page Footer
 - Summary
 - No Data
 - Background

Designer XML Preview

Arial 3

Coffee Title

Coffee Sub Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce auctor purus gravida arcu aliquam mattis. Donec et nulla libero, ut varius massa. Nulla sed turpis elit. Etiam aliquet mauris a ligula hendrerit in auctor leo lobortis.

ENAME	EMADD1	EMCITY	EMST	EMZIP1	EMZIP2
\$(ENAME)	\$(EMADD1)	\$(EMCITY)	\$(EMST)	\$(EMZIP1)	\$(EMZIP2)

new java.util.Date() "Page "+\$V" + \$V

Report Elements

- Break
- Chart
- Crosstab
- Ellipse
- Frame
- Html
- Image
- Barcode
- Generic Element

No Properties



Jasper Reports - Execute

You will need to save the jrxml file before you run the report primarily because Jasper will compile the report to a .jasper file. BOTH compiled and uncompiled reports can be used by RRE.

Jasper finally compiles down to a java object which is executed and produces the required output.

Jasper Output Options

Jasper currently supports:

PDF

XLS (ugly)

Text

CSV

RTF

ODF (Open Office et al)

HTML

Employee Listing Jasper - PDF

Employee Listing

Employee Name	Address	City	State	Zip	Zip + 4
OCHAOMPUNGMOO ADWEN	085 DEKALB AVENUE#2	SLCSEY CITY	UT	84117	0
RUBARS ASSRENA	068 KENSINGTON AVE.	SLCSEY CITY	UT	84117	0
VEVEON BALL	045 SO. HARRISON STREET.	SLCT ORANGE	UT	84117	0
RAENOLDU BANAJON	097 FRANKLIN STREET	SLCSEY CITY	UT	84117	0
JANNEFAR BESSEG	058 SMITH ROAD	SLCSIPPANY	UT	84117	0
PHELEP BOERD	069 VERSIDE DR.	SLCNFORD	UT	84117	0
OKAAM BOLUGON	087 KYLINE DRIVE	SLCSEY CITY	UT	84117	0
FRONCAS BORNIS	081 39TH ST	SLCON CITY	UT	84117	0
MORY LOU BOSELA	022 OAK AVENUE	SLCWOOD	UT	84117	0
VECSUREO BRESS	075 WASHINGTON AVENUE	SLCT ORANGE	UT	84117	0
LESO BRUWN	028 HOLMDEL ROAD	SLC	UT	84117	0
JUSAPH CERELLU	037 SHIPPEN STREET	SLCHAWKEN	UT	84117	0
RUBARS COSSELLU	053 ALES AVE	SLCSEY CITY	UT	84117	0
WOYNA CRAAD	101 COUSE ROAD	SLCTUNE	UT	84117	0

Isn't that lovely....

Deployment Approaches

iText wrapper (as previously seen)
(limited to currently wrapped methods)

iReport directly

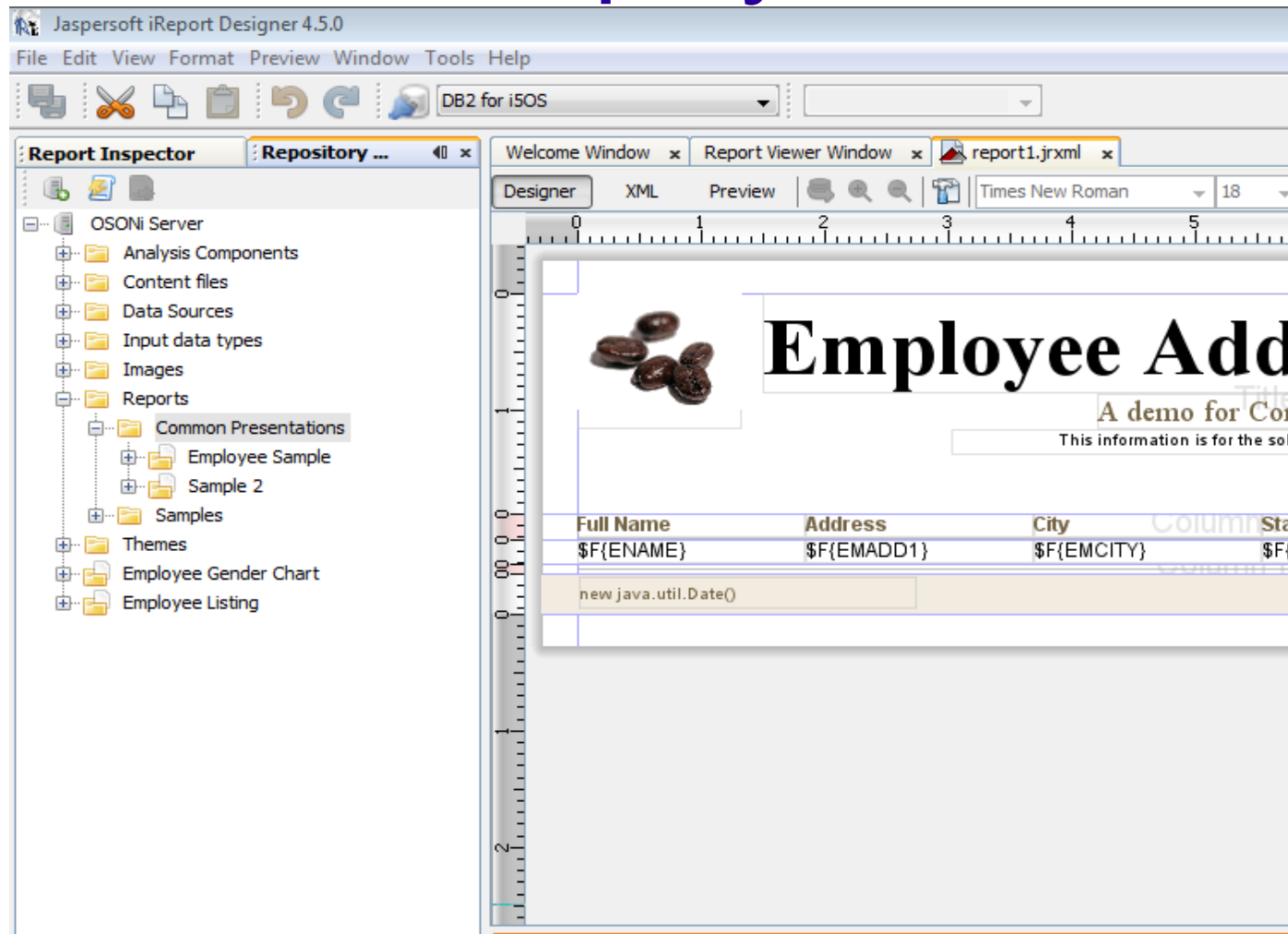
Jasper Reports Server

Call from RPG:
Jasper wrapper
(RPG Report Engine – RRE)

JasperReports Server

- Web Based
- Centralized
- Individual and group security
- Easy Deployment
- User selected output options
- Runs under Tomcat (so it runs on i!)
- Download the free community edition at the www.jasperforge.org site

Jasper Reports Server Deployment



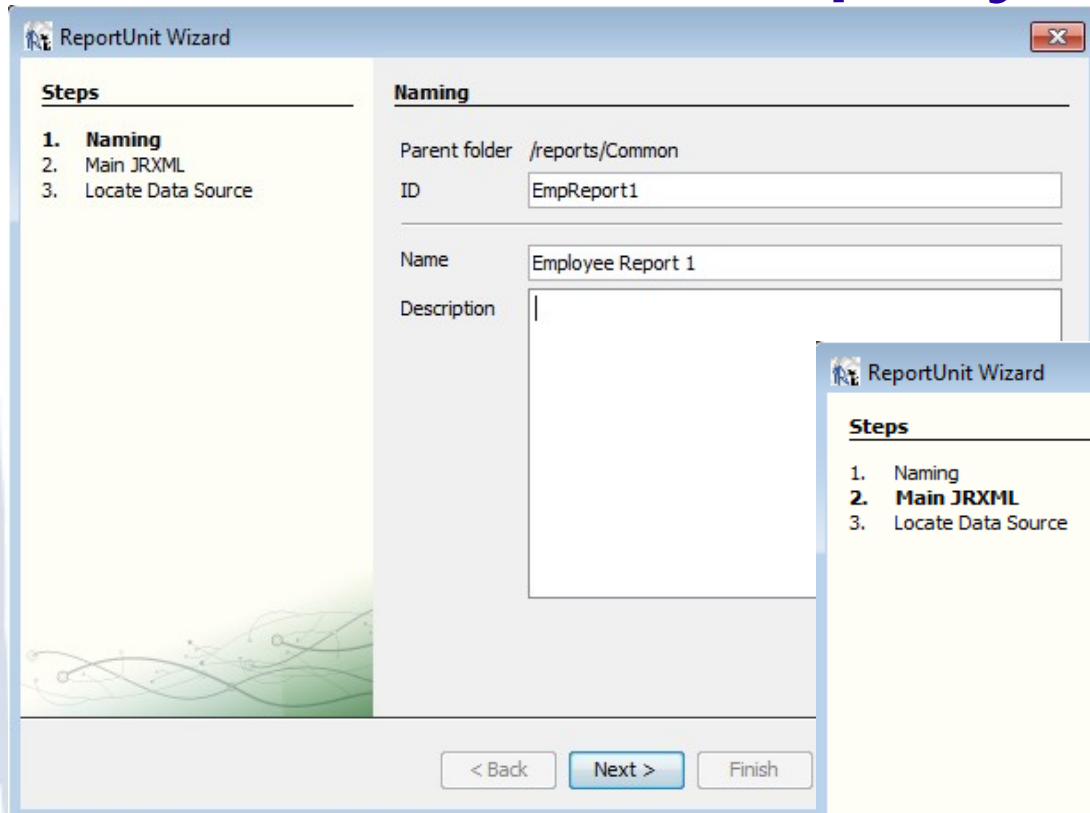
The screenshot shows the JasperSoft iReport Designer 4.5.0 interface. The 'Repository' view is active, displaying a tree structure under 'OSONi Server'. The 'Report Inspector' shows a preview of a report titled 'Employee Add'. The report content includes a title, a subtitle 'A demo for Cor', and a table with columns 'Full Name', 'Address', 'City', and 'State'. The table data is as follows:

Full Name	Address	City	State
<code>#{ENAME}</code>	<code>#{EMADD1}</code>	<code>#{EMCITY}</code>	<code>#{EMSTATE}</code>

Below the table, there is a code snippet: `new java.util.Date()`.

Select the JasperServer Repository view from the “Window” menu item. Navigate the repository tree to the location you want to add the report to. Right click and choose “Add” and select “JasperServer report

Jasper Reports Server Deployment



ReportUnit Wizard

Steps

1. **Naming**
2. Main JRXML
3. Locate Data Source

Naming

Parent folder: /reports/Common

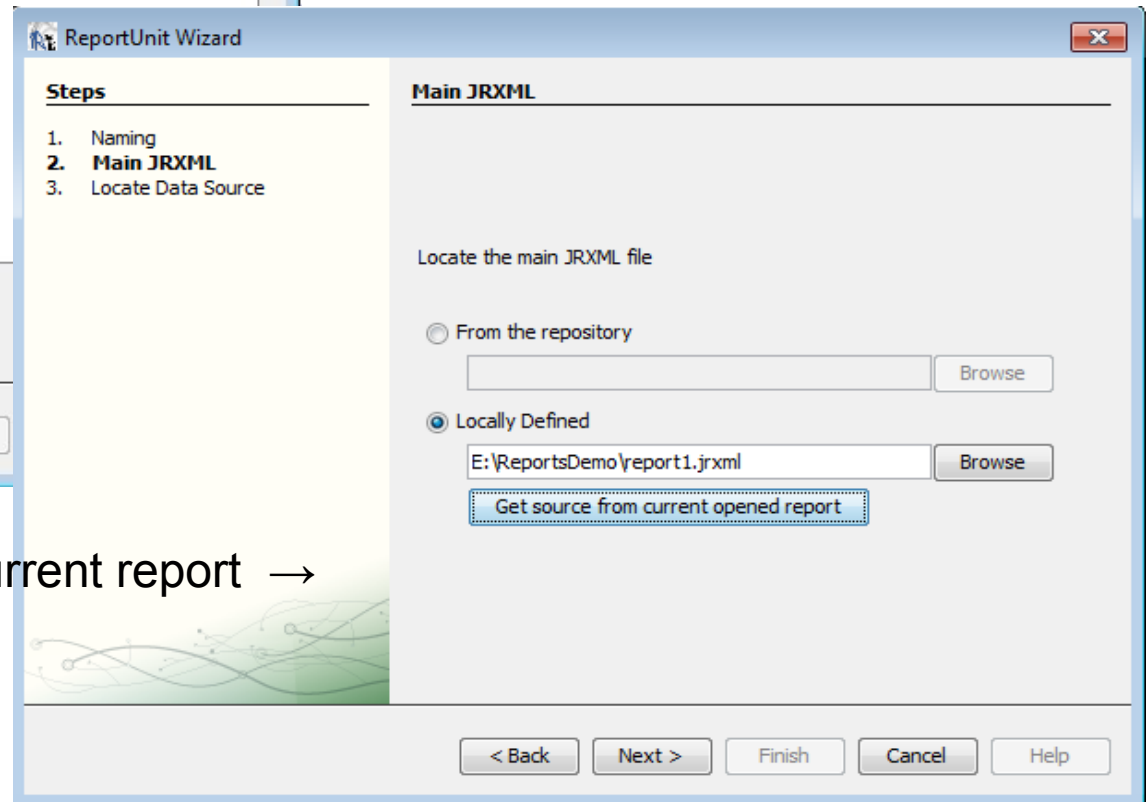
ID: EmpReport1

Name: Employee Report 1

Description:

< Back Next > Finish

← Name the report



ReportUnit Wizard

Steps

1. Naming
2. **Main JRXML**
3. Locate Data Source

Main JRXML

Locate the main JRXML file

From the repository

Browse

Locally Defined

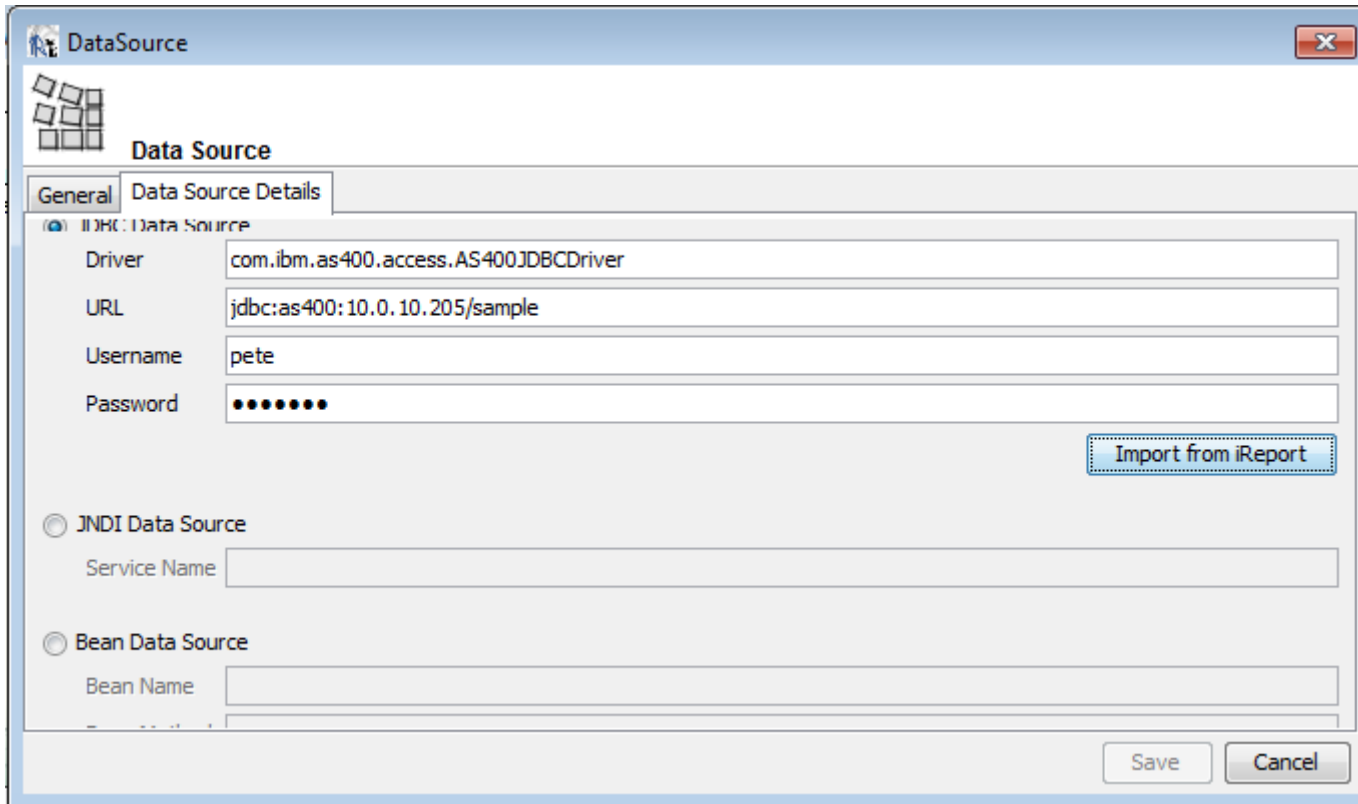
E:\ReportsDemo\report1.jrxml Browse

Get source from current opened report

< Back Next > Finish Cancel Help

Select the source from the current report →

Jasper Reports Server Deployment



The screenshot shows a 'DataSource' dialog box with the following fields and options:

- General** / **Data Source Details** tabs
- JDBC Data Source** (selected):
 - Driver:
 - URL:
 - Username:
 - Password:
 -
- JNDI Data Source** (unselected):
 - Service Name:
- Bean Data Source** (unselected):
 - Bean Name:
-

Choose a datasource by importing the datasource from current environment

JNDI would be a better choice....



Jasper Reports Server Deployment



JASPERSOFT

jasperadmin user | Log Out

View Manage

Report Viewer

Back Export

Employee Address

A demo for Common
This information is for the sole use of Common members

Full Name	Address	City	State	Zip	Zip+4
Toni Toni	123 East Main Street	Anytown	UT	84105	0
Sally Substitute	123 East Oak Ave	Anytown	UT	84000	0
Dave Dispatcher		Anytown	UT	84105	0
Alice Administrator				0	0
Diella Rhoads	1939 View Street	Du Quoin	UT	99999	0
Miley Stover	501 Lake Street	West Linn	UT	84066	0
Ryne Craig	507 Main Street	Normal	UT	84078	0
Chanel Mize	2952 Washington Road	Mamakating	UT	84035	0
Yvette Russo	1550 First Blvd.	Round Lake Beach	UT	84035	0
Hang Watts	1520 Washington Blvd.	Canyon Lake	UT	84078	0
Oralia Renner	2086 Cedar Road	West Haven	UT	84078	0
Rylan Toliver	624 Park Blvd.	East Palo Alto	UT	84078	0
Katlin Helm	4303 Cedar Street	Rockford	UT	84078	0

Page 1 of 78

About JasperReports Server Copyright © 2000-2011 Jaspersoft Corporation



Demo

- Let's deploy and run our demo report on the JasperReports Server.

Deployment via RPG

- Since JasperReports is a Java library, you'll need to wrap the java with RPG.
-
- Fortunately this has been done for you with the RPG Report Generator project.



Open Source = Java

(for me at least)



Modernizing RPG applications

Learn ILE RPG techniques

Don't **have** to learn JNI techniques
(unless you need a method that hasn't been wrapped)

Do you have to learn Java? No! But, being familiar with it can
make your life easier.

It's a wrap!

“Wrapping” is a method for interfacing between methods in API's. Either to simplify them or to make them more accessible in a specific language.

I use both approaches. I built an API with several wrappers in Java and then wrapped the wrappers in RPG.



So is the RPG report generator a
Java program or RPG?

Both. I am wrapping Java with RPG.

Scott Klement does this with HSSFR
(POI)

Aaron Bartell does this with the RPG Chart
Engine

(which inspired me to do the Report Engine)

Kudos to both of them for sharing their code!

Java Basics

(a short side trip)

Dealing with objects.

Objects have both a way to store information and they have ways of acting on that information, both internally and externally.

Information is stored in “fields” (variables) in the object.

Fields are acted upon by “methods”

Java Basics

(a short side trip - cont)

For example:

An “bank account” object might have fields to store account type, account number and a balance.

You might want to be able to add money (deposit), take money (withdraw) or just get a balance (these would be your methods)

Java Basics

(a short side trip - cont)

// We have a list of the fields and one method (a constructor) shown here

```
public class Account {  
  
    /**  
     *  
     */  
    String type;  
    String account;  
    private BigDecimal balance;  
  
    public Account(String type, String account) {  
        // TODO Auto-generated constructor stub  
  
        this.type = type;  
        this.account = account;  
        this.balance = new BigDecimal("0");  
    }  
}
```

Java Basics

(a short side trip - cont)

```
public boolean addMoney(BigDecimal deposit){
    boolean success = false;

    this.balance.add(deposit);
    // Maybe some DB I/O to update a table

    return success;
}

public boolean takeMoney(BigDecimal withdrawal){
    boolean success = false;

    this.balance.subtract(withdrawal);
    // Maybe some DB I/O to update a table

    return success;
}

public BigDecimal getBalance(String type, String number){

    return this.balance;
}
```

Java Basics

(a short side trip - cont)

Generally, the class represents a “blueprint” for how the object works in the world. Rarely do we act on the classes themselves. In most cases we create and “instance” of the class which will be unique during it's lifetime.

Java Basics

(a short side trip - cont)

So, rather than acting on the account class itself, we create an instance of it with the “new” operator which returns us a fresh object, built on our blueprint. e.g.

```
Account pete = new Account(“checking”, “123345”)
```

This constructs an instance of a checking account with account # 12345 and a balance of zero (my usual balance...)

Basically the thing is built in memory and referenced in our programs with the variable name “pete”.

Java Basics

(a short side trip - cont)

Now that we have an object, we can act on it.

```
Account pete = new Account("checking", "123345");
```

```
BigDecimal mydeposit = new BigDecimal("20.00");
```

```
boolean OK = pete.addMoney(mydeposit);
```

```
BigDecimal mybalance = pete.getBalance();
```



Example (Java – Java)



(sometimes called a “convenience method”)

```
// Main print routine for both Jasper within class Generator
```

```
public boolean printReport(Connection pConnection, String reportName, String
reportOutput, HashMap reportParams, String outputFormat, boolean compileFirst,
String engine ) {

boolean success = true;

if(engine.toUpperCase().equals("JASPER")) {

        JasperPrint jasperPrint = returnReportPrint(pConnection, reportName,
reportParams, compileFirst);

        File out = new File(reportOutput);
        .....

        return success;
    }
}
```

Example (Java to Java)

```
public boolean iEmailReport(String sender, String recipient, String reportName, String
reportOutput, HashMap reportParameters, String outputFormat, boolean compileFirst, String engine)
{
    Connection aConn = null;
    boolean success = true;

    aConn = setConnect();

    success = printReport(aConn, reportName, reportOutput, reportParameters,
outputFormat, compileFirst, engine);

    if(success){

        try {
            _smtpHost = _props.getProperty(SMTP_HOST);
            EzMailer.sendMessageAttach(_smtpHost,
                sender, recipient,
                "Your report completed normally", "Your report " + reportName
+ " ran and completed normally and is located here: " + reportOutput + "." + outputFormat,
                reportOutput + "." + outputFormat);
        } catch (MessagingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    disconnect(aConn);

    return success;
}
```

Example (Java to Java)

```
public boolean iEmailCompiledReport(String sender, String recipient, String  
reportName, String reportOutput, HashMap reportParameters, String outputFormat,  
String engine){  
  
    boolean success = true;  
  
    success = iEmailReport(sender, recipient, reportName, reportOutput,  
reportParameters, outputFormat, false, engine );  
  
    return success;  
}
```




OK. So how do we call this from RPG?



All Java programs run within a Java Virtual Machine. IBM has built an interface between the Java world and the RPG world so when an RPG program references a Java program the JVM is invoked (if the current job doesn't have one running). The RPG program invokes Java constructors to build the objects within the JVM and then provide pointers to the objects in the JVM so that they can be found and used by the RPG programs.

Drawbacks?

The construction of the JVM is I/O and memory intensive so there is a “warm up” period.

The two worlds are basically oblivious to each other so normal garbage collection of objects that are no longer in use in the Java world does NOT happen automatically.

Fixes to JVM drawbacks

JVM warm up can be minimized by running the process in a server job. But then a mechanism for communicating with the server job needs to be implemented (like with a data queue).

There are some callable routines that can “clean up” after calls to Java objects are made. These are already part of RRE

(but you need to code additional ones if you decide to wrap more Java methods)



Example RPG to Java

```
*+++++
D RREGenerator_iPrintReport...
D          PR          N   EXTPROC(*JAVA
D                               : 'com.valadd.report.engine-
D                               .Generator'
D                               : 'iPrintReport')
D host                    like(jString)
D libraryList             like(jString)
D userID                  like(jString)
D password                 like(jString)
D reportName              like(jString)
D reportOutput            like(jString)
D reportParam             like(jMap)
D outputFormat            like(jString)
D compFirst               N   value
D engine                   like(jString)
```



Plus an RPG convenience wrapper



```
*-----*
*
*   RPG Wrapper Prototype for RRE_iPrintReport
*
*-----*
D rre_iPrintReport...
D          PR          N
D   prHost          1024A   const varying
D   prLibList       1024A   const varying
D   prUserID        1024A   const varying
D   prPassword      1024A   const varying
D   prReportName    1024A   const varying
D   prReportOut     1024A   const varying
D   prRepParam      1024A   like(jMap)
D   prOutFormat     1024A   const varying
D   prCompile       N      value
D   prEngine        1024A   const varying
```

Basically you wouldn't know this was calling a java program except for this. You might be able to find an alternative...



RPG Report Engine Basics



Both Jasper and BIRT are supported.

Same API for both reports to keep it simple (for now)

The only difference? How each report is designed.

RRE API

Variations on a theme:

Two basic api's:

1. Print the report with output going to the IFS
2. Print the report and email it.

Two different report types for Jasper:

(ignored for BIRT)

Compiled (.jasper)

Uncompiled(.jrxml)

The result is 4 “flavors” of api's:

Email or not. Compiled or not



Parameters passed to RRE API



For the **most** basic API, printReport we need the following:

An SQL Connection object (handled for you automatically if you want)

The full path and name of the report file (with extension)

Full path and file name for the output (extension will be added automatically)

Report parameters (as a Java HashMap)

Compile indicator (true/false boolean)

Report type: Jasper or BIRT



Convenience methods

In RPG the easiest approach is to let RRE handle your connection to DB2 using the RRE properties file. Then you can use the convenience methods to run each report.



iPrintCompiledReport

```
*-----*
*
*   RPG Wrapper Prototype for RRE_iPrintCompiledReport
*
*-----*
D rre_iPrintCompiledReport...
D           PR           N
D   prReportName      1024A   const varying
D   prReportOut       1024A   const varying
D   prRepParam        like(jMap)
D   prOutFormat       1024A   const varying
D   prEngine          1024A   const varying
*+++++
```

RRE components

Prototypes are in RRE_H.rpgle

Actual procedure interfaces are coded in
RRE.rpgle



Example of RPG wrapper for Java procedure



```
P rre_iPrintCompiledReport...
P          B                                EXPORT
D rre_iPrintCompiledReport...
D          PI                                N
D   peReportName          1024A    const varying
D   peReportOut           1024A    const varying
D   peRepParam            like(jMap)
D   peOutFormat           1024A    const varying
D   peEngine              1024A    const varying
D success                 s          N
D gen                     s          like(RREGenerator)
D
/free

    gen = new_RREGenerator();

// For convenience convert RPG strings to string objects to pass
lReportName = new_String(peReportName);
lReportOut = new_String(peReportOut);
lOutFormat = new_String(%trim(peOutFormat));
lEngine = new_String(peEngine);

    success = RREGenerator_iPrintCompiledReport(gen: lReportName:
                                                lReportOut: peRepParam:
                                                lOutFormat: lEngine);

    rre_freeLocalRef(gen);

    return success;
/end-free
P          E
```

Java equivalent

```
// This wrapper assumes that connection from properties settings will be used  
and that file is  
// is a compiled .jasper file (needs no compiling)
```

```
public boolean iPrintCompiledReport(String reportName,String reportOutput,HashMap  
reportParameters, String outputFormat, String engine){
```

```
    Connection aConn = null;  
    boolean success = true;
```

```
    aConn = setConnect();
```

```
    success = printReport(aConn, reportName, reportOutput, reportParameters,  
outputFormat, false, engine );
```

```
    disconnect(aConn);
```

```
    return success;
```

```
}
```



RPG Example

```
/free
```

```
rre_begin_object_group(100);

ReportName = '/rre/reports/templates/employee_listing_with_Parms.jasper';

  ReportOut = '/rre/reports/output/employee_listing_with_parms_2_test8';

  lReParam = new_jMap();

  lTempMap = new_jMap();

  lkey = new_String('selectZip');

  // Maybe this should be a BigDecimal

  lvalue = new_jInteger(84078);

  lTempMap = rre_jmap_put(lReParam:lkey:lvalue);

  OutFormat = %trim(OutputType);

  Engine = 'jasper';

success = rre_iPrintCompiledReport(

    ReportName :ReportOut

    :lReParam

    :OutFormat :Engine);

rre_end_object_group();

*inlr = *on;
```

```
/end-free
```

What we didn't cover

Most reports have parameters that are passed to it. RRE does accommodate the passing of parameters to the reports. You could have a web front end or a green screen front end that captures the values and passes them RRE.

The challenge in using parameters in reports isn't in the running of them, it is in the design.

What we didn't cover

Sub reports

Perhaps a future session is needed dealing only with the iReport report designer.



Thanks! Questions?

Pete Helgren
Value Added Software, Inc.
pete@valadd.com

Code samples and the complete RRE package is
available here:

<http://www.opensource4i.com>