

1. How to pass Parameters and Datasource from Master report to subreport:

1.1. Subreport Expression:

`$P{SUBREPORT_DIR} + "address_report_template.jasper"`

1.2. Data Source Expression:

`new net.sf.jasperreports.engine.data.JRMapCollectionDataSource($V{dataList})`

The screenshot shows the JasperReports Designer interface. The main workspace displays a report template with sections: Title, Detail 1, Column Footer, and Summary. A subreport is being configured in the 'Detail 1' section. The 'Data Source Expression' is set to `new net.sf.jasperreports.engine.data.JRMapCollectionDataSource($V{dataList})`. The 'Subreport Expression' is set to `$P{SUBREPORT_DIR} + "address_report_template.jasper"`. The 'Expression Class' is `java.lang.String`. The 'Using Cache' checkbox is checked. The 'Run to bottom' checkbox is unchecked. The 'Parameters Map Expression' is set to 'Use a datasource expression'. The 'Connection Expression' is set to `new net.sf.jasperreports.engine.data.JRMapCollectionDataSource($V{dataList})`. The 'Parameters' and 'Return Values' are set to 'No parameters defined' and 'No return values defined' respectively.

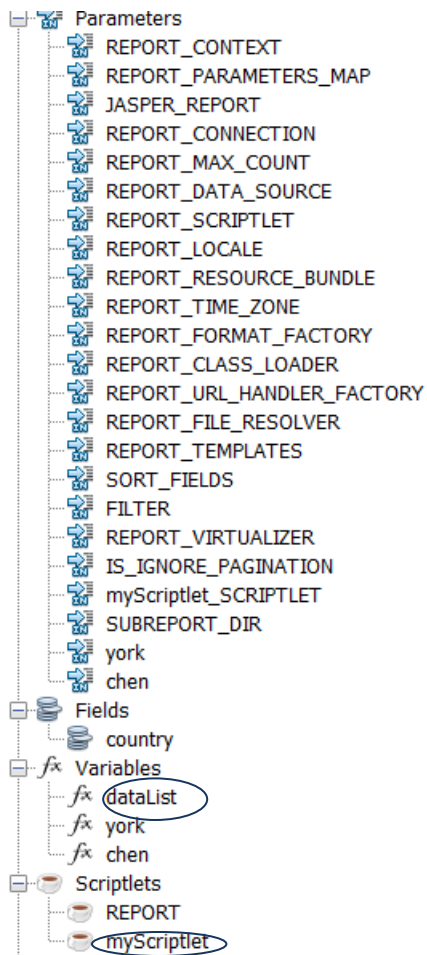
The 'Variables' panel on the left shows the following variables:

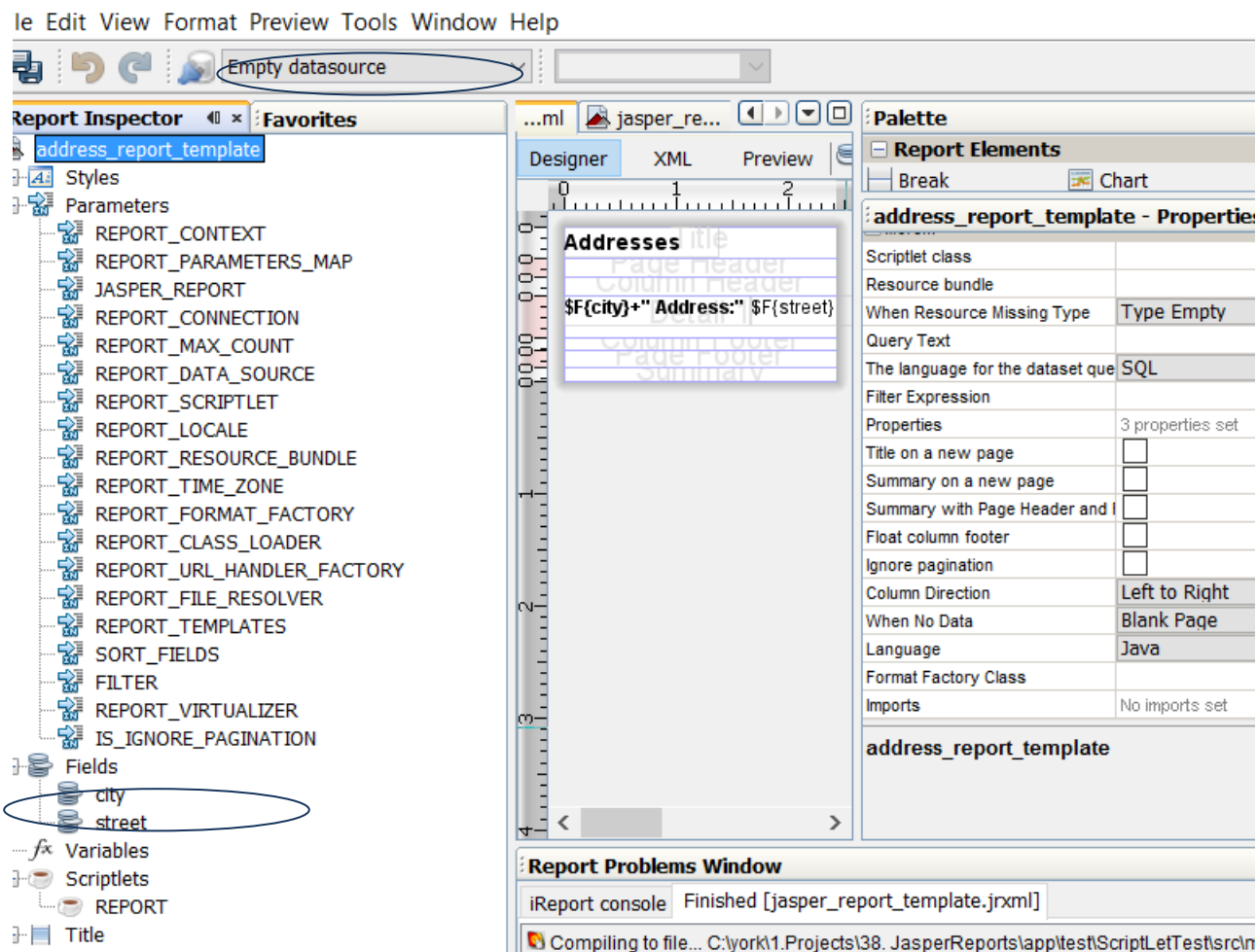
- PAGE_NUMBER Variable Integer
- COLUMN_NUMBER Variable Integer
- REPORT_COUNT Variable Integer
- PAGE_COUNT Variable Integer
- COLUMN_COUNT Variable Integer
- dataList Variable ArrayList
- york Variable String
- chen Variable String

The 'Expression Wizards' panel on the right shows the following methods:

- numberOfLeadingZeros(int) int
- numberOfTrailingZeros(int) int
- bitCount(int) int
- equals(Object) boolean
- toString() String
- toString(int, int) String
- toString(int) String
- hashCode() int

The 'Import...' and 'Export...' buttons are visible at the bottom left. The 'OK', 'Reset to default', and 'Cancel' buttons are visible at the bottom right.





1.3. Follow the steps below to pass parameters into subreports:

1. Create main report parameter, such as DATE_PARAM.
 2. Open sub report and Create parameter with the same and and same type.
 3. Go back to main report
 4. Right-click on sub report, select properties
 5. Choose parameter
 6. Add parameter from main report to sub report with parameter name same parameter name
- The parameter is passed from the main report to the subreport.

2. Use ScriptLets (ScriptLets can change variables.

(1) Set the sub report data source to variable:

```
new net.sf.jasperreports.engine.data.JRMapCollectionDataSource($V{dataList})
```

(2) Make sure the master datasource has at least one record.

For example, use following into SQL, but only use variables in the report

```
select 'name' as name, 'country' as country
```

(3) Put subreport into detail section and re-set all variables in afterDetailEval()

2.1.

```
package org.york.jasper.jasperapp;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import net.sf.jasperreports.engine.JRDefaultScriptlet;
import net.sf.jasperreports.engine.JRScriptletException;

public class MyScriptlet extends JRDefaultScriptlet {
    // List<Map<String, ?>> dataList = new ArrayList<Map<String, ?>>();

    public void getDataList() {
        List<Map<String, ?>> maps = new ArrayList<Map<String, ?>>();
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("city", "Beijing99");
        map.put("street", "11 Wulidian00");
        maps.add(map);

        map = new HashMap<String, Object>();
        map.put("city", "Beijing299");
        map.put("street", "22 Wulidian200");
        maps.add(map);

        try {
            this.setVariableValue("dataList", maps);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void updateVariables() throws JRScriptletException {
        System.out.println("call afterReportInit()");

        String york = (String) this.getParameterValue("york");
        String chen = (String) this.getParameterValue("chen");

        System.out.println("updateVariables, v york=" + york);
        System.out.println("updateVariables, v chen=" + chen);
        this.setVariableValue("york", new String("updateVariables, V york="
            + york));
        this.setVariableValue("chen", new String("updateVariables, V chen="
            + chen));

        getDataList();
    }

    public String hello() throws JRScriptletException {
        return "Hello! I'm the report's scriptlet object.";
    }

    public void beforeReportInit() throws JRScriptletException {
        System.out.println("call beforeReportInit()");
    }
}
```

```

    public void afterReportInit() throws JRScripletException {
        System.out.println("call afterReportInit()");
    }

    public void beforePageInit() throws JRScripletException {
        System.out.println("call beforePageInit()");
    }

    public void afterPageInit() throws JRScripletException {
        System.out.println("call afterPageInit()");
    }

    public void beforeColumnInit() throws JRScripletException {
        System.out.println("call beforeColumnInit()");
    }

    public void afterColumnInit() throws JRScripletException {
        System.out.println("call afterColumnInit()");
    }

    public void beforeDetailEval() throws JRScripletException {
        System.out.println("call beforeDetailEval()");
    }

    public void afterDetailEval() throws JRScripletException {
        System.out.println("call afterDetailEval()");
        updateVariables();
    }
}

```

2.2. Put subreport into detail section

```

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.york.jasper.jasperapp.*;

import net.sf.jasperreports.engine.JRDataSource;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperExportManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;
import net.sf.jasperreports.engine.data.JRMapCollectionDataSource;

public class CreateReport {

    public static void main(String[] args) {
        String path = "C:\\york\\1.Projects\\38.
JasperReports\\app\\test\\ScriptLetTest\\src\\main\\resources";
    }
}

```

```

// String masterReportFileName = path + "/jasper_report_template.jrxml";
// String subReportFileName = path + "/address_report_template.jrxml";
String destFileName = path + "/jasper_report_template.jrprint";
String jasperMasterReport = path + "/jasper_report_template.jasper";
String pdfName = path + "/jasper_report_template.pdf";

try {
    /* Compile the master and sub report */
    // JasperReport jasperMasterReport =
    // JasperCompileManager.compileReport(masterReportFileName);
    // JasperReport jasperSubReport =
    // JasperCompileManager.compileReport(subReportFileName);

    Map<String, Object> parameters = new HashMap<String, Object>();
    // parameters.put("subreportParameter", null);
    parameters.put("chen", "chen1");
    parameters.put("york", "york1");

    List<Map<String, ?>> maps = new ArrayList<Map<String, ?>>();
    for (int i = 0; i < 1; i++) {
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("name", "Shoud not be here");
        map.put("country", "Shoud not be here");
        maps.add(map);
    }
    JRMapCollectionDataSource dataSource = new JRMapCollectionDataSource(
        maps);

    JasperFillManager.fillReportToFile(jasperMasterReport,
        destFileName, parameters, dataSource);

    export(destFileName, pdfName);

    System.out.println("PDF created...");
} catch (JRException e) {

    e.printStackTrace();
}
System.out.println("Done filling!!! ...");
}

public static void export(String sourceFileName, String pdfName) {
    try {
        // 1- export to PDF
        JasperExportManager.exportReportToPdfFile(sourceFileName, pdfName);
        // exporter.exportReport();
    }
}

```

```

    } catch (JRException e) {
        e.printStackTrace();
    }
}

```

```

}

```

2.3.

Master report:

The screenshot displays the JasperReports Designer interface. On the left, a preview of the 'Contact Report' is shown with a table structure. The table has a header section with a title 'Contact Report', a 'Title' field, and a 'Detail 1' section containing two rows: 'V York: \$V{york}' and 'V Chen: \$V{chen}'. Below the detail is a 'Column Footer' and a 'Summary' section. On the right, the 'Tools' palette is visible, showing various report elements like Barcode, Rectangle, Round Rectangle, List, Sort, Spider Chart, Static Text, and Subreport. Below the tools, the 'Web Framework' section is expanded, showing a 'Sort' button. The 'Properties' panel for the selected subreport is also visible, showing various settings like 'Print In First Whole Band', 'Print When Detail Overflows', 'Print When Group Changes', 'Print When Expression', 'Properties expressions', 'Subreport properties', 'Subreport Expression', 'Expression Class', 'Using Cache', 'Run to bottom', 'Parameters Map Expression', 'Connection type', 'Connection Expression', and 'Data Source Expression'.

Contact Report	
Title	
V York: \$V{york}	Detail 1
V Chen: \$V{chen}	
Column Footer	
Summary	

Tools

- Barcode
- Rectangle
- Round Rectangle
- List
- Sort
- Spider Chart
- Static Text
- Subreport

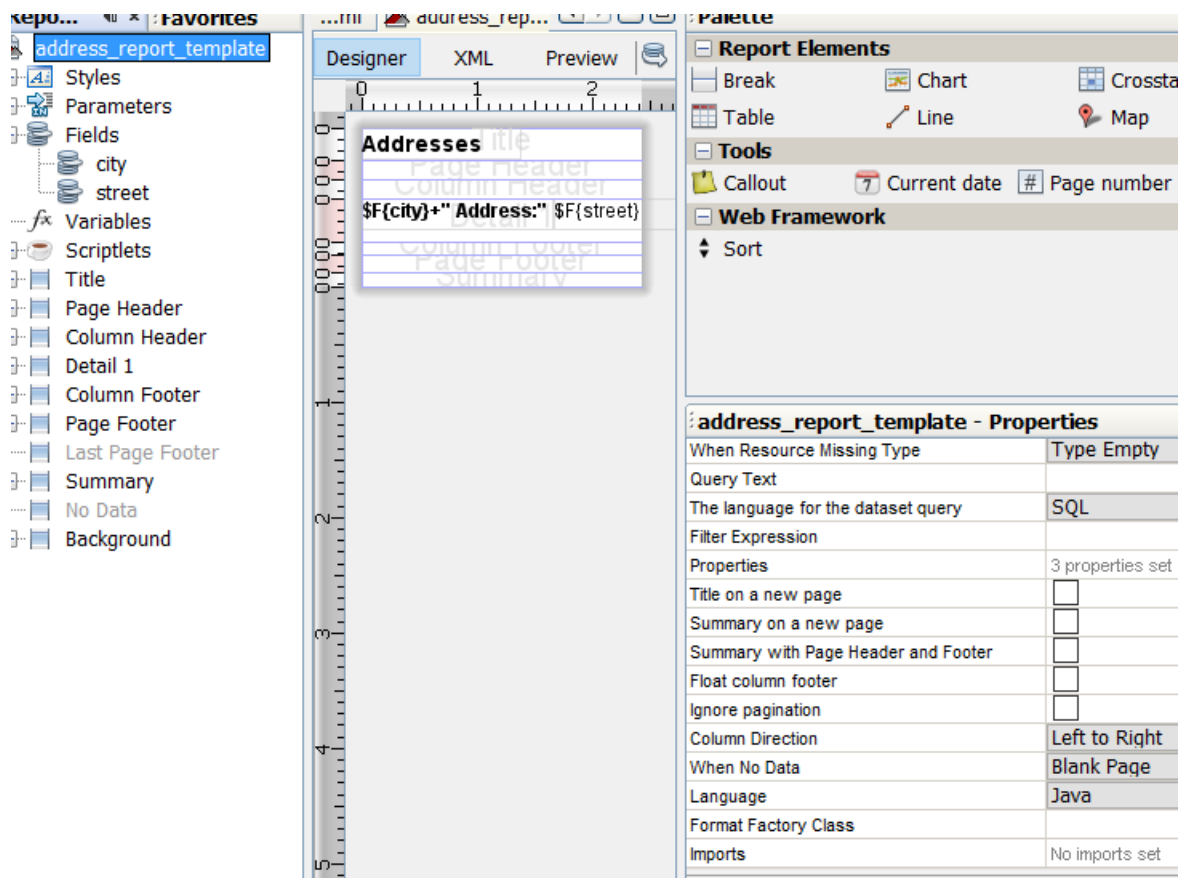
Web Framework

- Sort

Properties

- Print In First Whole Band
- Print When Detail Overflows
- Print When Group Changes
- Print When Expression
- Properties expressions: No properties set
- Subreport properties
- Subreport Expression: \$P{SUBREPORT_DIR} + "address_report_template.jasper"
- Expression Class: java.lang.String
- Using Cache: ☒
- Run to bottom
- Parameters Map Expression
- Connection type: Use a datasource expression
- Connection Expression
- Data Source Expression: new net.sf.jasperreports.engine.data.JRMapCollectionDataSource(\$V{dataList})

2.4. Subreport:



For testing in iReport, need to add the Jar file into class path:

