

## 10. Reports

JRapid provides integration with reporting tools. Initially, it allows to easily including reports designed for Jasper Reports. These reports can be fully parameterized through JRapid generated screens.

This feature is much stronger when developing SaaS applications, because it allows managing a model where reports are deployed for each tenant.

### 10.1 About Jasper Reports

JasperReports is an open source Java reporting tool that can write to screen, to a printer or into PDF, HTML, Microsoft Excel, RTF, ODT, Comma-separated values and XML files. It can handle scriptlets in the report definition, subreports, and multiple datasources merging.



**Home page:** <http://jasperforge.org/>

### 10.2 Integration modes

Reports can be integrated in two main different ways: As application reports, and dynamically deploying them in the repository.

When developing multi-tenant SaaS applications (a single instance of software serving multiple client organizations –tenants-) it becomes necessary to identify the features that are common to all tenants from those who are specific for each one, and different integration models appear for each one of these cases.

#### 10.2.1 Application

When working in single-tenant applications, with a limited scope of reports, application reports are the best option in seek of quick inclusion and easiness of administration.

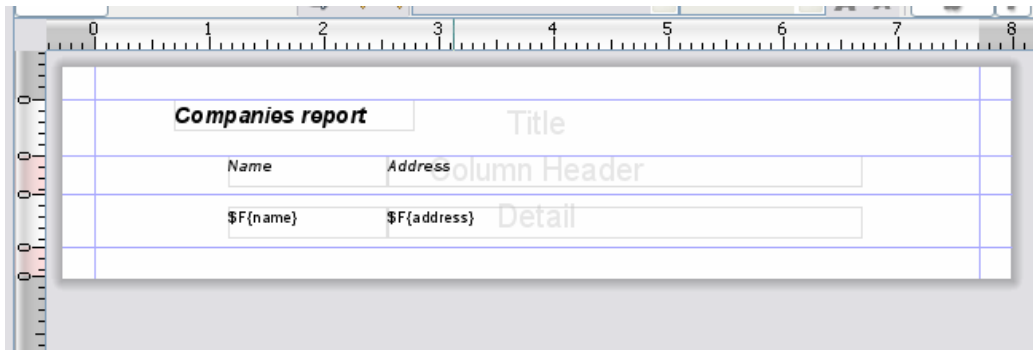
The way of deploying these reports is copying the compiled `.jasper4` file to the web application. The most simple method is just adding a new folder `reports/` in `WEB-INF` directory and saving the file there. Finally, committing changes in that folder through SVN will put the file in the server.

#### Example

After creating a report called *companies* with the report design tool, showing two fields (*name* and *address*) from a record set that is the result of bringing all the companies of the database.

---

<sup>4</sup> Compiled `.jasper` files can be obtained using open source tools like iReport (<http://ireport.sf.net>)



The resulting source code (companies.jrxml) is:

```
<?xml version="1.0" encoding="UTF-8"?>
<jasperReport
xmlns="http://jasperreports.sourceforge.net/jasperreports"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports
http://jasperreports.sourceforge.net/xsd/jasperreport.xsd" name="report
name" pageWidth="595" pageHeight="842" columnWidth="535"
leftMargin="20" rightMargin="20" topMargin="20" bottomMargin="20">
  <queryString>
    <![CDATA[select c.name, c.address from company c]]>
  </queryString>
  <field name="name" class="java.lang.String">
    <fieldDescription><![CDATA[]]></fieldDescription>
  </field>
  <field name="address" class="java.lang.String">
    <fieldDescription><![CDATA[]]></fieldDescription>
  </field>
  <background>
    <band/>
  </background>
  <title>
    <band height="35">
      <staticText>
        <reportElement x="50" y="0"
width="151" height="20"/>
        <textElement>
          <font size="14" isBold="true"
isItalic="true" isUnderline="false"/>
        </textElement>
        <text><![CDATA[Companies report]]></text>
      </staticText>
    </band>
  </title>
  <pageHeader>
    <band/>
  </pageHeader>
  <columnHeader>
    <band height="24">
      <staticText>
```

```

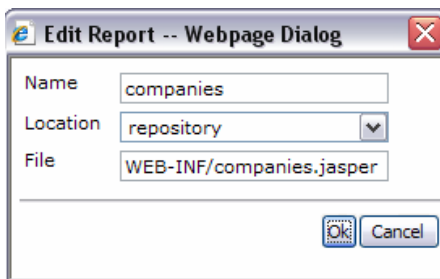
        <reportElement x="84" y="0"
            width="100" height="20"/>
        <textElement>
            <font isBold="true" isItalic="true"/>
        </textElement>
        <text><![CDATA[Name]]></text>
    </staticText>
    <staticText>
        <reportElement x="184" y="0"
            width="298" height="20"/>
        <textElement>
            <font isBold="true" isItalic="true"/>
        </textElement>
        <text><![CDATA[Address]]></text>
    </staticText>
</band>
</columnHeader>
<detail>
    <band height="33">
        <textField hyperlinkType="None">
            <reportElement x="84" y="8"
                width="100" height="20"/>
            <textElement/>
            <textFieldExpression class="java.lang.String">
                <![CDATA[ ${F{name}} ]]>
            </textFieldExpression>
        </textField>
        <textField hyperlinkType="None">
            <reportElement x="184" y="8"
                width="298" height="20"/>
            <textElement/>
            <textFieldExpression class="java.lang.String">
                <![CDATA[ ${F{address}} ]]>
            </textFieldExpression>
        </textField>
    </band>
</detail>
<columnFooter>
    <band/>
</columnFooter>
<pageFooter>
    <band/>
</pageFooter>
<summary>
    <band/>
</summary>
</jasperReport>

```

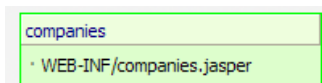
The report can be previewed from the development tool with the datasource of the tutorial of chapters 4,6 and 8, resulting in:

<b>Companies report</b>	
<b>Name</b>	<b>Address</b>
Mars	6885 Elm Street
Chrysler	1000 Chrysler Ave. Auburn Hills MI
Cargill	null

In order to include the report inside the tutorial application, the compiled *companies.jasper* file must be uploaded. If WEB-INF is chosen as the containing folder, when adding the report through *New > Report* this folder should be specified. The following pop-up will appear.



The entity diagram will show:



Since this moment, the report can be accessed through the URL

```
http://SERVER_NAME.jrapid.com/APP_NAME/report/REPORT_NAME
```

Although this mode allows including reports really quickly, it is quite limited as the reports must be deployed by the developers of the system (new versions of the jasper file must be put in the same folders as the code) and also there is only one version of the report for all tenants (for all schemas of the application).

### 10.2.2 Repository

When there is a need for a custom version of reports for each schema, or it is just necessary letting the administrator to upload new versions, the **repository integration mode** turns useful.

The way of activating this integration mode is choosing *location = repository* in the edit report window.

The screenshot shows a dialog box titled "Edit Report -- Webpage Dialog". It contains three input fields: "Name" with the value "CompaniesRepo", "Location" with a dropdown menu showing "repository", and "File" with the value "companies.jasper". At the bottom right, there are "Ok" and "Cancel" buttons.

After defining the jasper name, it can be uploaded for each repository. This means that it can have a design that varies for each tenant.

If the application is sold as a service (SaaS) for 2 clients, and each one wants a custom design for the company list, these custom designs can be kept inside each one's own repository.

So, in a multi-tenant application, there can be two different designs coexisting inside the application, as they are uploaded for each schema:

**Client 1 Schema.** Client 1 has 3 companies in the database and a black & white design that just shows name & address (*companies1.jasper* in client 1 repository).

The screenshot shows a report titled "Companies report" in a black and white style. It contains a table with two columns: "Name" and "Address".

Name	Address
Mars	6885 Elm Street
Chrysler	1000 Chrysler Ave. Auburn Hills MI
Cargill	null

**Client 2 Schema.** Client 2 has 2 companies in the database, and uses a colorful design (*companies2.jasper* in client 2 repository).

The screenshot shows a report titled "COMPANIES REPORT" in a colorful style. It contains a table with three columns: "Name", "Address", and "Total Employees".

Name	Address	Total Employees
HSBC Holdings	8 Canada Square	330000
General Electric	3135 Easton	327000

### 10.3 Parameterized reports

Reports need to be parameterized to make them really useful. The trick is using a transient entity together with the Next element. In the previous example, the report could receive a minimum amount of employees filter named *fromEmployees*

```
<entity name="GetCompanyReportParams"
  label="Companies report parameters" transient="transient">
  <property name="fromEmployees" label="From Employees"
    type="integer"/>
  <next report="AllCompanies"/>
</entity>
```

So the resulting parameters window should look like any other transient entity, with the difference that when it is submitted, the values are converted to report parameters and they are sent automatically.

