Web Reporter for Rich Internet Applications

Version 1.1

User Guide

Revision: November 18, 2007

Table of Contents

1.0 CLEARBI OVERVIEW

ClearBI is a Web reporter and business intelligence engine that allows software developers automate report generation process. End users view and customize the reports using Flash Player. ClearBI is available in two editions: ClearBI Developer and ClearBI Server:

- ClearBI Developer edition allows a software developer to create and customize a new report in Eclipse IDE. This report can be integrated into any Flex application by including an extra MXML file and recompiling the main application. The end users will be able to work with the report (sorting, filtering, grouping, export to Microsoft Excel, and more), but won't be able to save newly created customized reports.
- ClearBI Server edition has all the functionality of the developer version, and also allows the end users create reports from the universe of the data fields without need to install any software other than Flash Player. The end users create, customize and save reports in the centralized database server without any help from an IT department.

Reports can be *saved*, *published* and *compiled*. A ClearBI report can be *saved* as an MXML file in the project's directory, in a RDBMS table, *published* in the database and *compiled* into .swf file that can be downloaded into the user's PC and run by Flash Player.

ClearBI can work with the data supplied by Plain Old Java Objects (POJO), Web Services or a JavaScript array. While working with POJO, you can use either Adobe LiveCycle Data Services or open source implementation of the AMF protocol called OpenAMF.

ClearBI relies on metadata – the data about the report fields such as their names and data types. ClearBI reports can be built based on an SQL Select statement, a database stored procedure, a SOAP WebService (requires included WebService.swf), which allows automatically collect the metadata from the result set. AJAX developers can configure ClearBI to use data from a JavaScript array, but in this case they'd need to fill out the form describing metadata manually.

Using SQL or stored procedure data sources is the most automated way of creating clearBI reports. In this case all required client and server code is automatically generated by Clear Data Builder plugin, which comes with ClearBI.

You can also specify a Web service and ClearBI will extract the metadata from the supplied WSDL as described in section. To get a quick feeling of how to use of the Web service or a JavaScript array as a data provider for a report, please watch a ten-minute pre-recorded screencast at http://www.myflex.org/screencast/clearbi_ajax/clearbi_ajax.html .

1.1 ClearBI Feature Matrix

Feature	Server Edition	Developer's Edition
Supported data sources		
LiveCycle Data Services ES	X	X
OpenAMF (open source)	X	X
JavaScript Array	X	X
SOAP Web Service	X	X
Free-form formulas	X	X
Free-form computed styles	X	X
Grouping, Sorting, Filtering	X	X
Export to Microsoft Excel (Internet Explorer)	X	X
Charting (requires Flex Data visualization license)	X	X
Report printing	X	X
PDF Creation (requires LCDS ES license)	X	X
Clear Data Builder Eclipse Plugin	X	X
Report publishing in RDBMS	X	
Report saving in the file system	X	X
End-user report customization	X	
Report User administration	X	
Report Player. Reference implementation	X	X
Report Player by DB ID. Ref. implementation	X	
Report Player with Charts. Ref. implementation	X	
Report Administration. Ref. implementation	X	

2.0 SOFTWARE REQUIREMENTS

ClearBI Developer 1.1 requires:

- Clear Data Builder (included)
- Java Development Kit (JDK) 5.0 or later.
- Flex Builder 2 installed as a plugin in Eclipse 3.2.
- ClearBI can create reports using the data provided by one of the following:

1. LiveCycle Data Services ES 2.5 (LCDS)
2. OpenAMF – an open source implementation of AMF protocol, see http://sourceforge.net/projects/openamf/ for more details
3. A SOAP Web Service
4. A JavaScript array

3.0 AUTOMATING REPORTING WITH CLEAR DATA BUILDER

Both Developer and Server editions of ClearBI come with Clear Data Builder (CDB) plugin, which automatically generates the metadata required for creating reports in the form of MXML files. It also deploys all supporting jars, flex deployment descriptors, and the libraries you need to run ClearBI.

CDB plugin creates one Java and one Flex Eclipse proects. We will call the Java piece *Clear Data Builder project* to distinguish it from other (regular) Java projects

CDB project is a Java project upgraded with an extra *nature* (an Eclipse term), which allows automatic code generation and deployment based on the annotated Java files, a.k.a. *CDB source files*.

You can turn a regular Java project into a CDB project by a right mouse click on the project name followed by selection *Configure build* wizard from the menu. Why would you even want to convert your Java project into a CDB one? Since this is a manual about using ClearBI, one of the answers is that you want CDB help in auto-generating your reports. But even without having such a goal, just making any Web-related Java project CDB-enabled would give you such valuable feature as automatic deployment of your Java code to your J2EE application server. Any change in the Java source code would initiate an Ant build that would compile your code and deploy the jars in the configured servlet container.

CDB project can be associated with a *target Flex project*. In that case CDB automatically builds (and keeps in sync) ActionScript classes that facilitate marshalling of remote instances of Java classes to/from Flash Player. During configuring a CDB project, you should specify the name of the target Flex project where generated ActionScript classes should be placed.

CDB automatically creates and/or deploys all artifacts required for a Web application – Java jars and configurations files that enable a standard Web application to send and receive data to/from a Flex application that Flash Player runs. It also deploys required LCDS / SDK files under your Java servlet container.

You have to specify the physical location and the URL of your Web application while configuring CDB project via *Configure build…* wizard. You can also let the wizard borrow these settings from the associated target Flex project if exists.

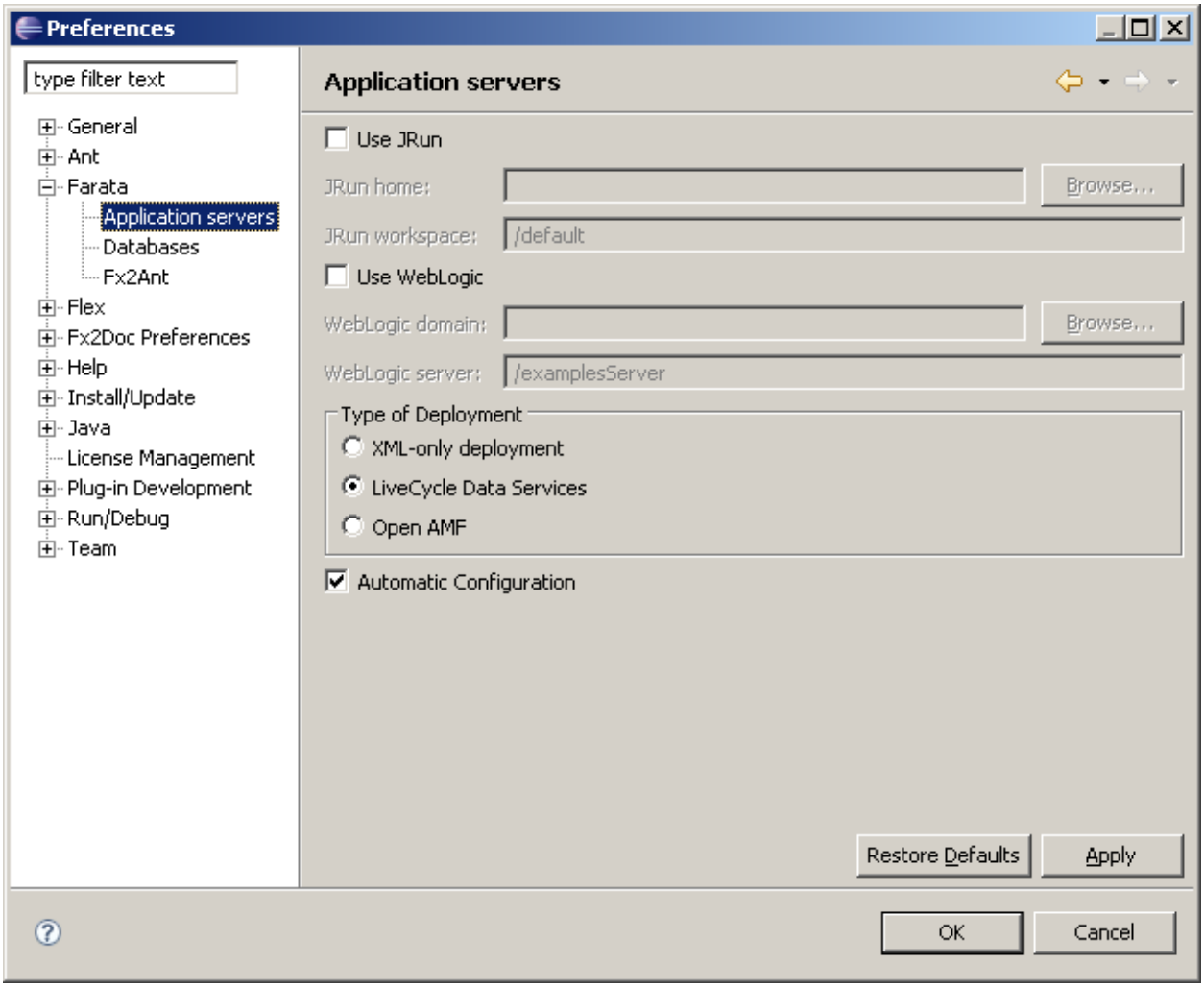
You can create your CDB application in Eclipse with Flex Builder using one of the following methods:

- Create CDB project first, and then create Flex Project. In this case, during creation of the new CDB project, it'll automatically create a Web application for your Java EE Application server. Then, during creation of your Flex (Flex Data Services, *compile locally*) project you will point at already created Web application and your server.
- If you already have a Flex project, create a CDB project specifying location of an existing Web application. The files required by CDB will be added to this Web application automatically.

We'll go over these methods in the next section, but if you are eager to learn how to quickly create an example ClearBI project using CDB, go right to section 4.1 of this manual.

3.1 Creating and configuring CDB project

Before configuring CDB project you need to select the type of server deployment for the code generated by CDB. This is done on the configuration screen located under the menu Window > Preferences> Farata > Application Servers as shown below:



ClearBI 1.1 supports various types of server side deployment, which you need to specify on the screen above before configuring your CDB project.

XML-only deployment – CDB will not deploy anything on your server other then a small number of XML files required for code generation. You may use this option in cases when the server application already exists.

LiveCycle Data Services – in this mode CDB will deploy required files from your LCDS directory to your Web application's directory.

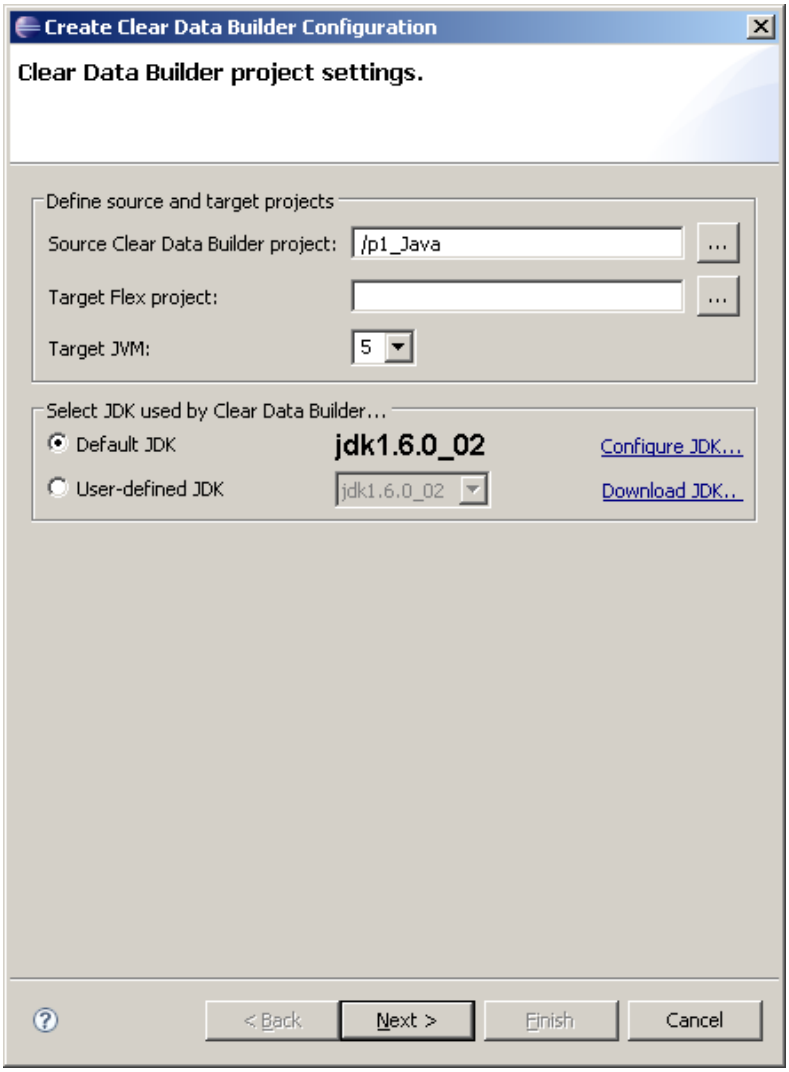
OpenAMF – in this mode CDB will deploy required OpenAMF files in your Web application's directory.

Note. Depending on the selected type of deployment (LCDS or OpenAMF), your Flex project's file `htmltemplate\index.template.html` will have a variable flex-rpc-mode with a value of 0 for LCDS or 2 for openAMF. Currently, the assignment of this value can not be automated and is applied to all new Flex projects created in Eclipse workspace. But if you'd like to change the type of deployment of the existing Flex project created by CDB, please edit the file index.template.html and manually change the value of flex-rpc-mode to 0 or 2 (there are two places to be changed).

To convert an Eclipse Java project into a CDB project perform the following steps:

1. Open Eclipse Java perspective, and create a new Java project. While creating this project, it is recommended to keep the source code and output files in separate directories.

2. Right-click on your Java Project and select **Clear Data Builder > Configure build…** in the menu and select your options for CDB project in the window shown below:

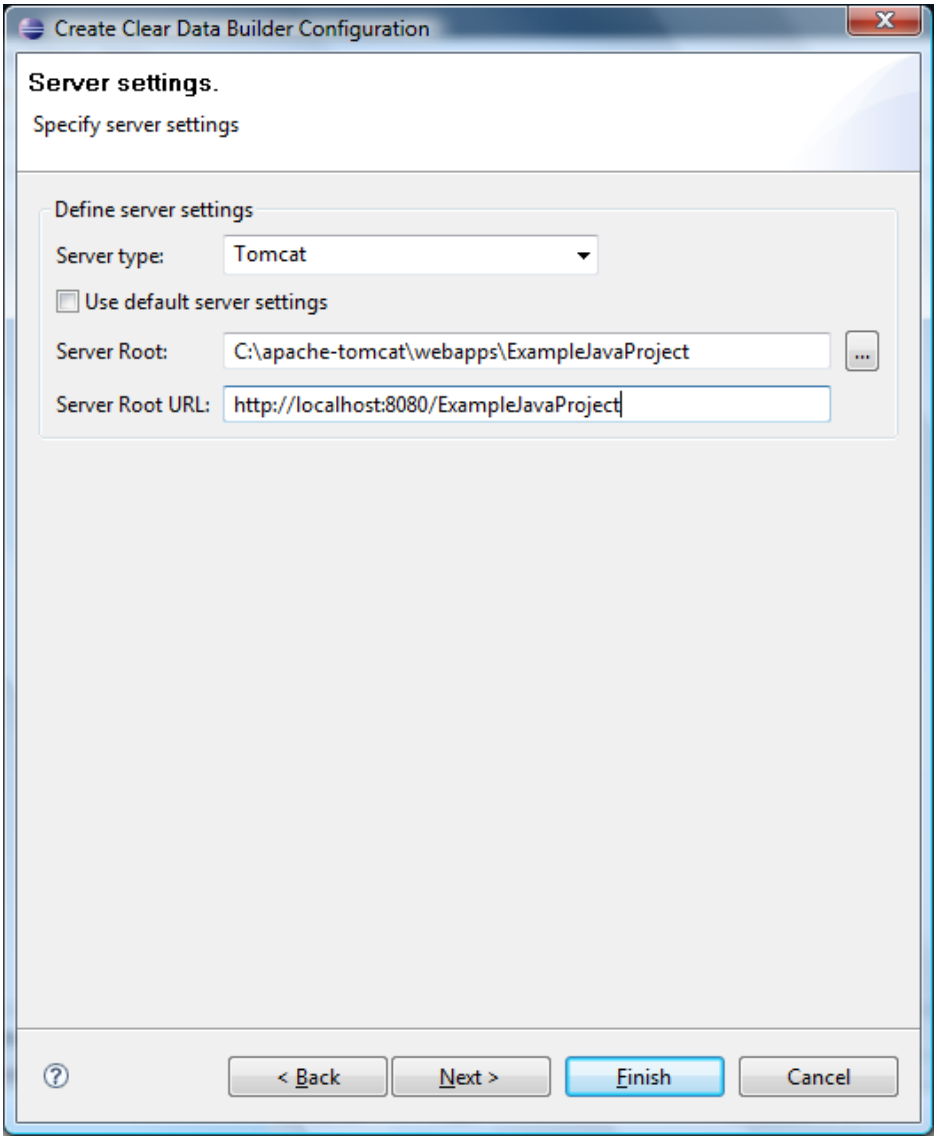


The meaning of these settings are:

- **Source Clear Data Builder project -** the name of your Java project to be converted into a CDB project. The name of the current Java project will be suggested automatically.
- **Target Flex project -** If you already have Flex project configured, enter its name in this field. Otherwise, you may leave it blank and fill it out later.
- **Target JVM** – This is the version of the Java Run-Time in the target run-time environment of your application server.
- **Select JDK to be used by Clear Data Builder -** You can select the version of JDK to be internally used by CDB during code generation. Please note that while Eclipse uses JRE and not JDK for the projects, you need to make Eclipse use **JDK and not JRE** by clicking the link Configure JDK. Please note that JDK used by CDB is not related to the version of the target JVM.

Press the **Next** button.

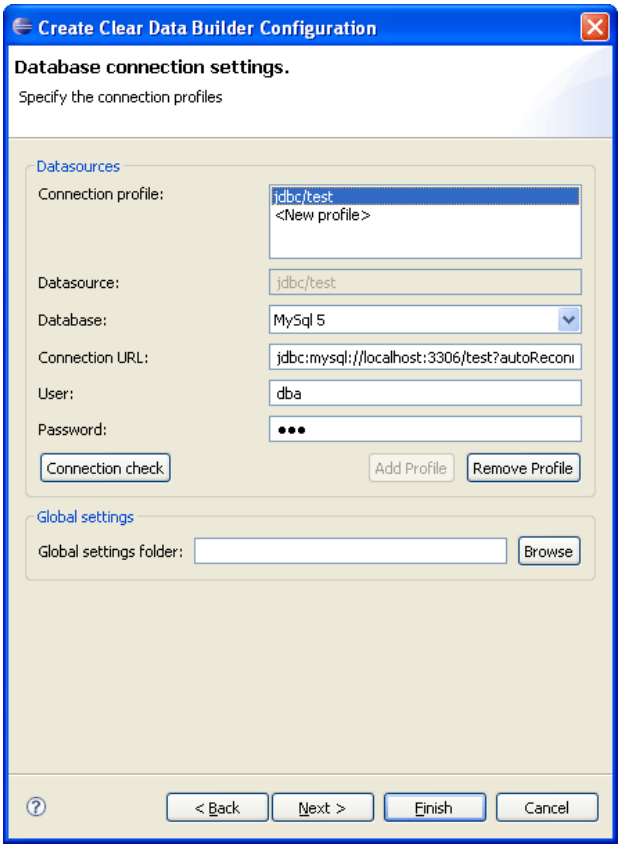
3. In this step we'll configure the Java servlet container where the server side of your application is or will be deployed.



Java application servers from different vendors may have different rules for deployment of Web applications. The Server type dropdown allows you to select from Tomcat, WebLogic and JRun, but if you have J2EE server that is not in the list, select Tomcat as the server type, and when the web application is generated and deployed, i.e. under the folder webapps\myapp, just copy the content of this folder to the location suggested in the documentation of your Java EE server.

If you specified the **Target Flex Project** in the previous step, just check off **Use default server settings** and the rest of the values will be imported automatically. You can also key in the settings manually – just specify the Server Root directory and URL for your application as shown below. Press the **Next** button.
4. Configure your DBMS connection.

Enter the parameters required by the JDBC driver for your server and press the button **Add Profile**. You can also validate the database connection by pressing the button **Connection check**.

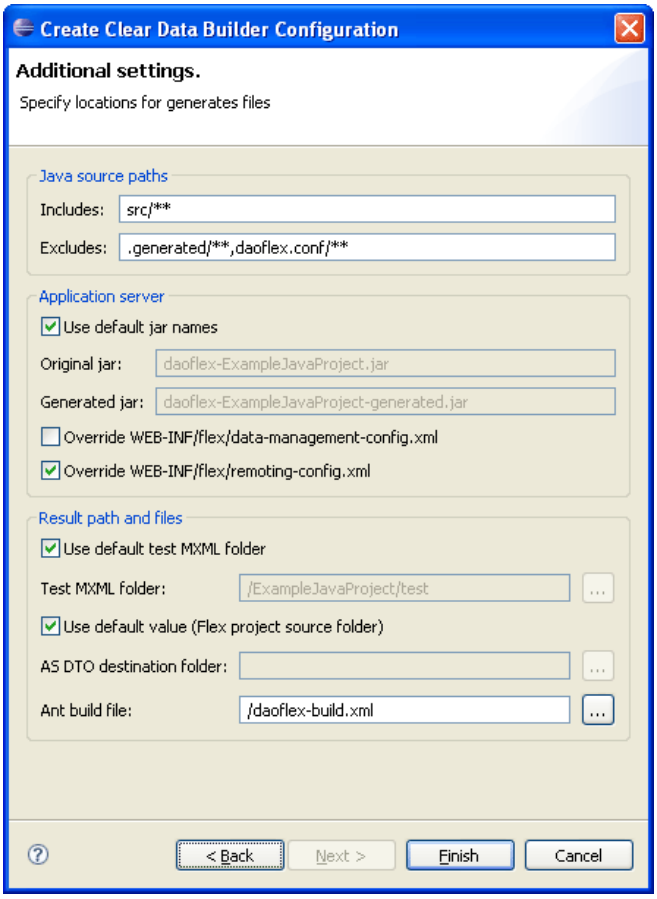


You'll need one more database connection named clearbi (see section 11.1) to persist layouts of your reports and perform user administration if needed.

If you need the same set of database profiles for several projects use the parameter **Global settings folder**. In this case you need to create required database profiles in one of your projects, and just specify this directory in other projects. All these database profiles will become available in other projects. Press the **Next** button.

5. Finalize CDB configuration by setting the following values:

- **Test MXML folder -** CDB generates various MXML files for testing of your application. Enter the name of the directory where these files should be created or accept the default location.
- **Ant build file** - specify location of the Ant build file.
- **Java source folder -** this is the source directory of your Java project. CDB will process only the Java files located there.
- **AS DTO destination folder -** you can specify the directory for CDB to store generated ActionScript Data Transfer Object (DTO) files or agree with the default location.
- **Original and generated jars -** specify the jar name for the compiled classes (original) written by you as well as generated by CDB.
- **Override WEB-INF/flex/data-management-config.xml and WEB-INF/flex/remoting-config.xml -** you can allow overriding pre-existing configuration XML files (both Flex Data Management Services and Flex Remoting) to alow CDB to create required server-side destinations generate supporting code .



Press the button **Finish** and CDB will immediately start building your new project with Ant.

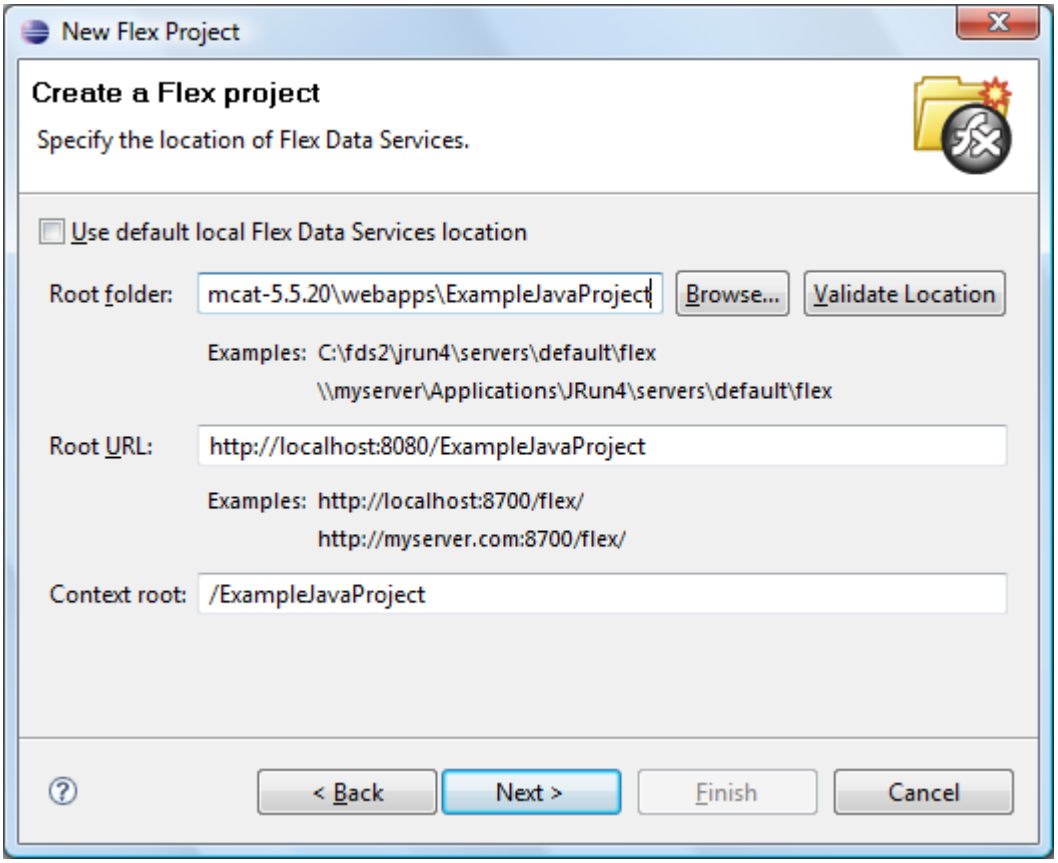
Note. If your Java project does not have any source files at this point, you'll see a message on your console stating that build failed, because "*no source files and no packages have been specified*". Create your first Java class and this message will go away.

If you check the content of your WEB-INF directory in your project (in our case it's webapps\ExampleJavaProject), you'll find that LiveCycle Data Services has been deployed there in the subdirectory named Flex.

If you already have Flex Project, CDB configuration is over; otherwise you'll need to perform the additional steps described below.

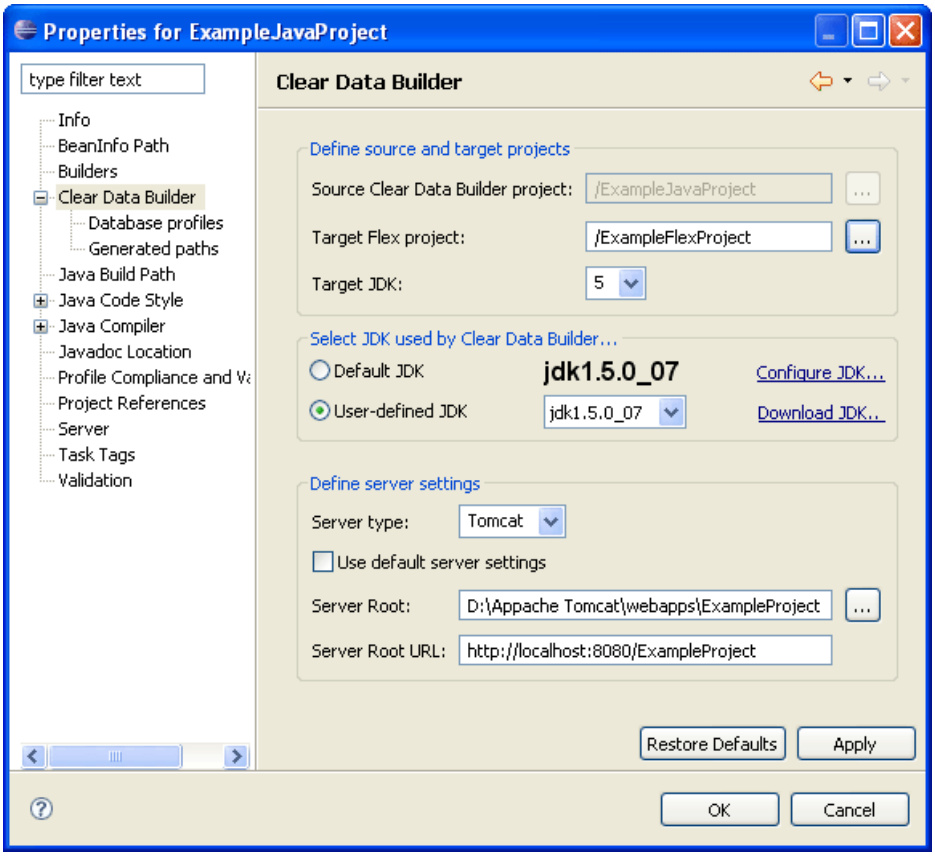
6. Switch to Flex perspective and create your Flex Project.

Note. If you are planning to use LiveCycle Data Services, local compilation) providing location for the server's root folder, the root URL and the context root to match the settings of your CDB project:



Go back to your Java project and select **Clear Data Builder > Reconfigure build** to apply these settings to your CDB project.

Change the **Target Flex Project** to your newly created Flex project.

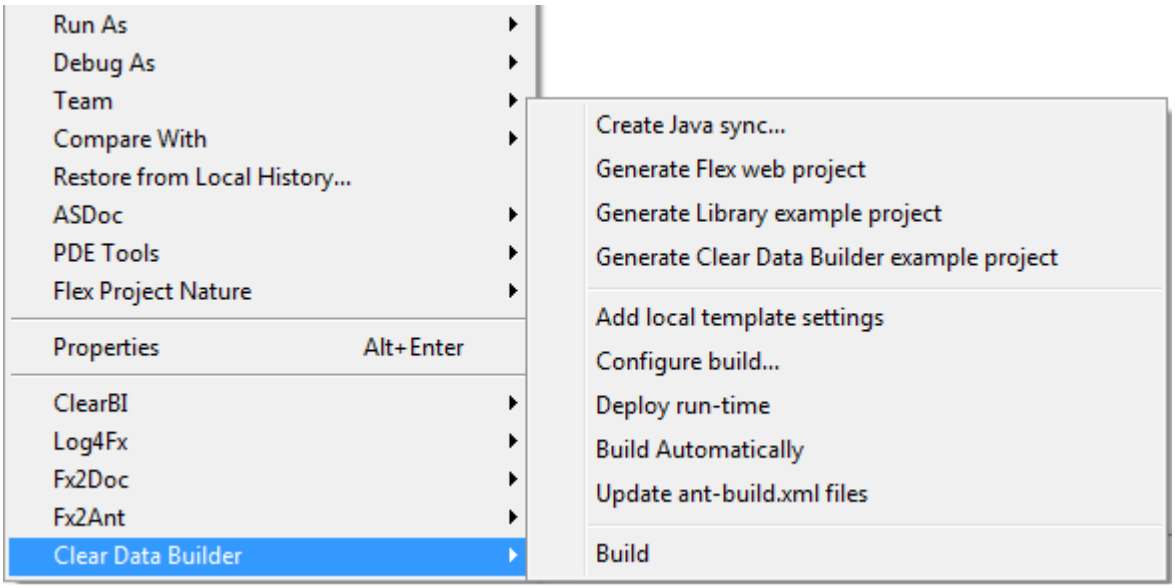


Click **OK**, and your Clear Data Builder project is configured.

Note. After the first run of ClearBI you may need to restart your application server because ClearBI will deploy a couple new jars files over there.

3.2 Other Clear Data Builder Settings

Open Java Perspective, right-click on your Java project and select **Clear Data Builder** from the popup menu.



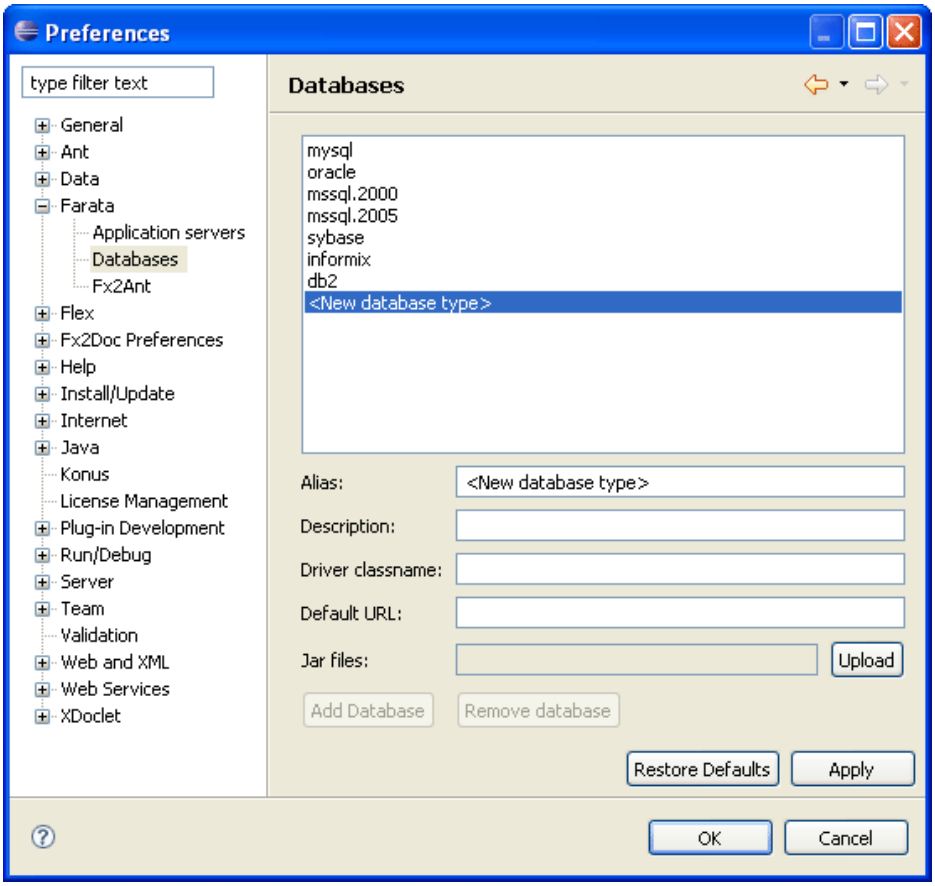
- **Configure (or Reconfigure) build… -** This button will open the CDB configuration page. You can also get there from the Properties page of your Java project**.**
- **Deploy run-time -** This menu item deploys all files required for your Web application server.
- **Build Automatically -** If this menu option is checked, CDB will start regenerating files every time you make a change in the source code and save it. This option is turned on by default.
- **Update ant-build.xml files -** This option refreshes the existing Ant build file. It comes handy if you modified some of the configuration parameters.
- **Add local template settings -** creates or updates the file daoflex.properties in the root directory of your project. This file allows you to change the settings of your CDB project. This will be described later in this document.
- **Build -** This menu option generates all required CDB files.
- **Create Java sync… -** This menu allows you to generate a template of Java class that facilitates retrieval and transactional update of the arbitrary data source. Template indicates places where you would need to manually add the project specific code. You would also have to manually configure destinations to allow using this class via Remoting or Data Management Services.
- **Generate Clear Data Builder Example project** – generates and deploys two sample projects (Flex and Java) that serve as demo of a reference implementation of CRUD Flex-Java-DB application and ClearBI reporter (see section 4.1 for details).

- **Create Java sync… -** This menu allows you to generate a template of Java class that facilitates retrieval and transactional update of the arbitrary data source. Template indicates places where you would need to manually add the project specific code. You'd also have to manually configure destinations to allow use of this class via Remoting or DataManagement Services.

3.3 Adding more database profiles

1. Select **Windows > Preferences...** from the main menu.

2. Select **Farata > Databases** on the left. You'll see the list of already configured databases and can add another one as long as you have its JDBC drivers.



The preparation part is done, and now you'll apply this knowledge while creating an example ClearBI project.

4.0 CREATING A SAMPLE REPORT IN TEN MINUTES

CDB is preconfigured to generate its MXML files into the folder **fxbi**. These MXML files are plain reports which are used by ClearBI.

Even though ClearBI 1.1 can work with different types of data sources, we'll start with creating a report based on a database query.

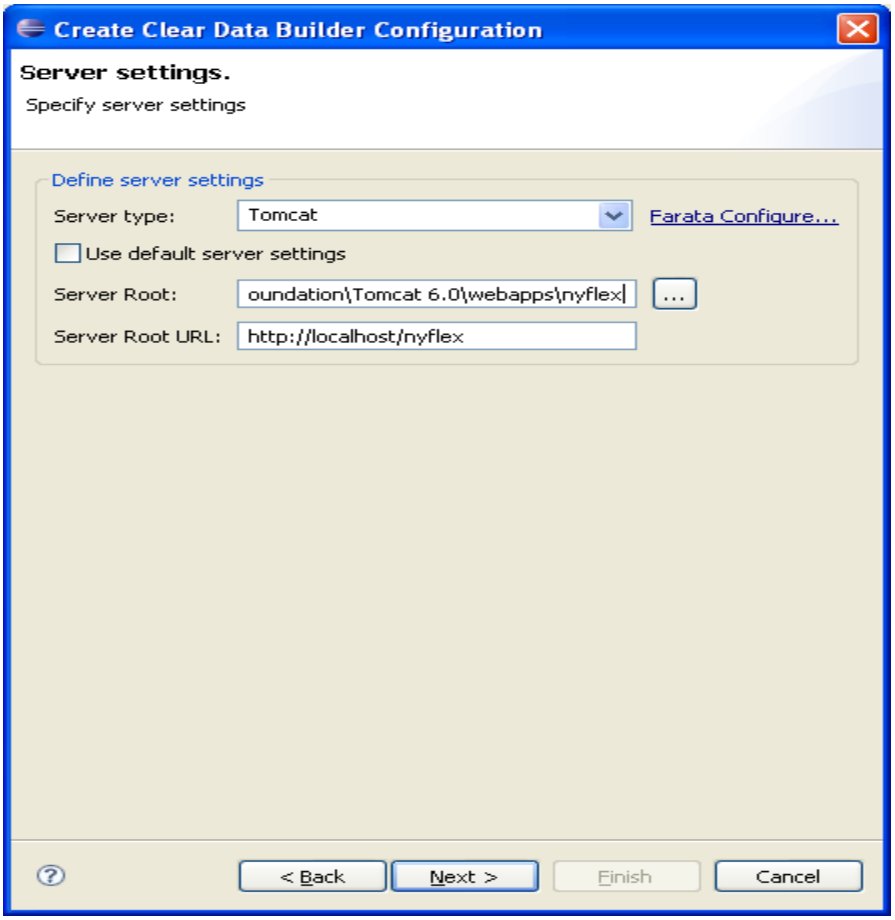
4.1 Quick Start – Creating a CRUD Application and ClearBI Report

The easiest way to start working with ClearBI is by generating an example CDB project that also includes several test ClearBI reports. This process of creating an example CDB project is described below, and you can watch a pre-recorded screencast at
http://www.myflex.org/screencast/cdb_example/cdb_example.html

Note. Please configure the type of deployment first as described in the beginning of the section 3.1.

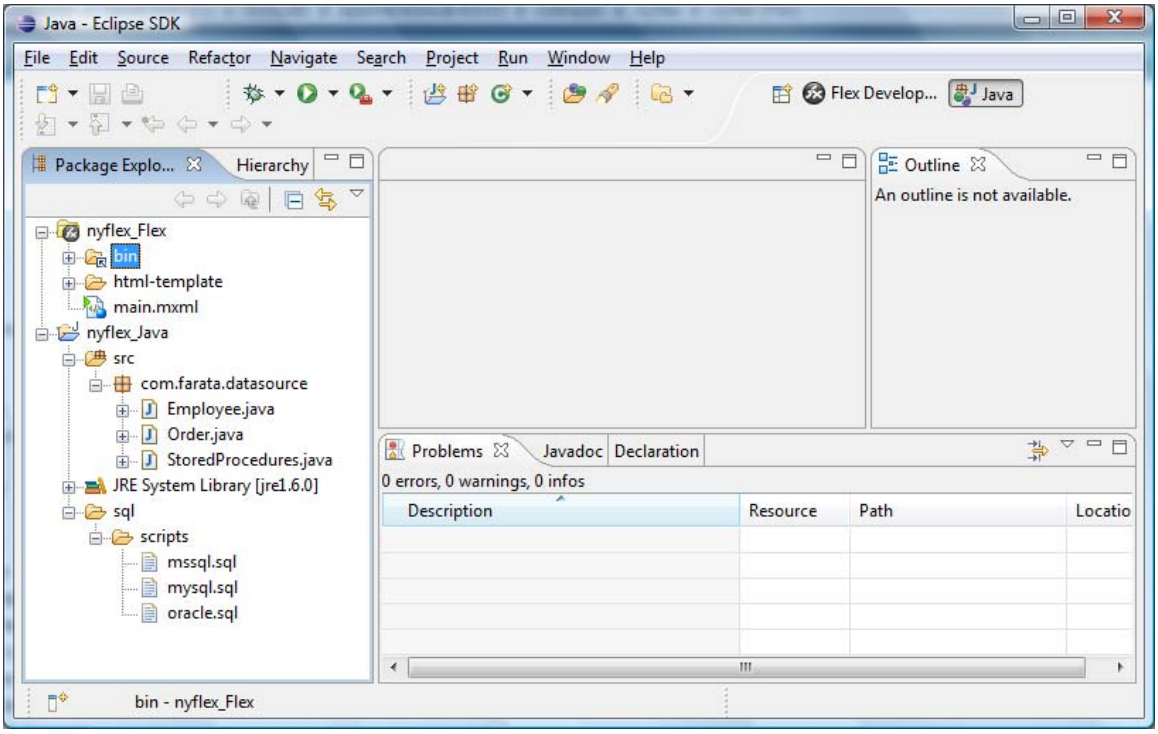
We'll start in Eclipse IDE with installed CDB and ClearBI plugins. Let's create a new example CDB project by selecting the menus File > New > Other > Clear Data Builder > Generate Example Clear Data Builder Project.

Select the server root: browse to your deployment directory (i.e. **webapps** in Apache Tomcat) and create a new folder there (i.e. **nyflex**). Specify the server root URL that users of our example report will be entering in their Web browser.



Note. If you are planning to use LCDS, visit the Farata Configure link and enter the directory where your lcds is installed, i.e. c:\lcds.

In a couple of seconds you'll see two new projects (Java and Flex) in your Eclipse workspace. Since we've selected **nyflex** as a name of the deployment folder, the projects names will be **nyflex_Flex** and **nyflex_Java** accordingly.



Wait for several seconds and check your Java project for the newly generated content. Look for the generated Java abstract class Employee.java. A fragment of this class is shown below. If you decide to create not a sample but a real project with the help of CDB, you can use the sample project as a reference, but modify the abstract class Employee to represent the sql and the data transfer Java class that meets your needs.

```
/**
 * @daoflex:webservice
 *   pool=jdbc/test
 */
public abstract class Employee
{
            /**
            * @daoflex:sql
            *  pool=jdbc/test
            *  sql=::  select * from employee
            *  ::
            *  transferType=EmployeeDTO[]
```

```
*  keyColumns=emp_id
*  updateTable=employee
*/
public abstract List getEmployees();

/**
* @daoflex:sql
*  sql=::  select * from employee
*  ::
*  transferType=EmployeeDTO[]
*  updateTable=employee
*  keyColumns=emp_id
*  tag=sqlCallback:modifyGetEmployeeSQL
*/
public abstract List searchEmployees(Map context);
```

…

4.1.1 Create a sample database

This example project relies on the existence of the sample database with employees information and the three sample DDL scripts (MySQL Server, Microsoft SQLServer and Oracle) for this database is conveniently provided in the folder **sql/scripts** of your Java project.

Select one of the provided scripts and run it in your DBMS[1].

4.1.2 Configure Java Development Kit.

Right-click on the name of your Java project and select **ClearData Builder > Configure Build** from the menu. Select the JDK installed on your computer (please note, you need to select JDK, not JRE). Click on the button Configure JDK, then the button Add and Browse to find the install of your JDK. For example, if you've installed Java 6.0 in the default location under Windows, the proper folder is C:\Program Files\Java\jdk1.6.0.

Select your newly created Flex project in the Target Flex Project field.

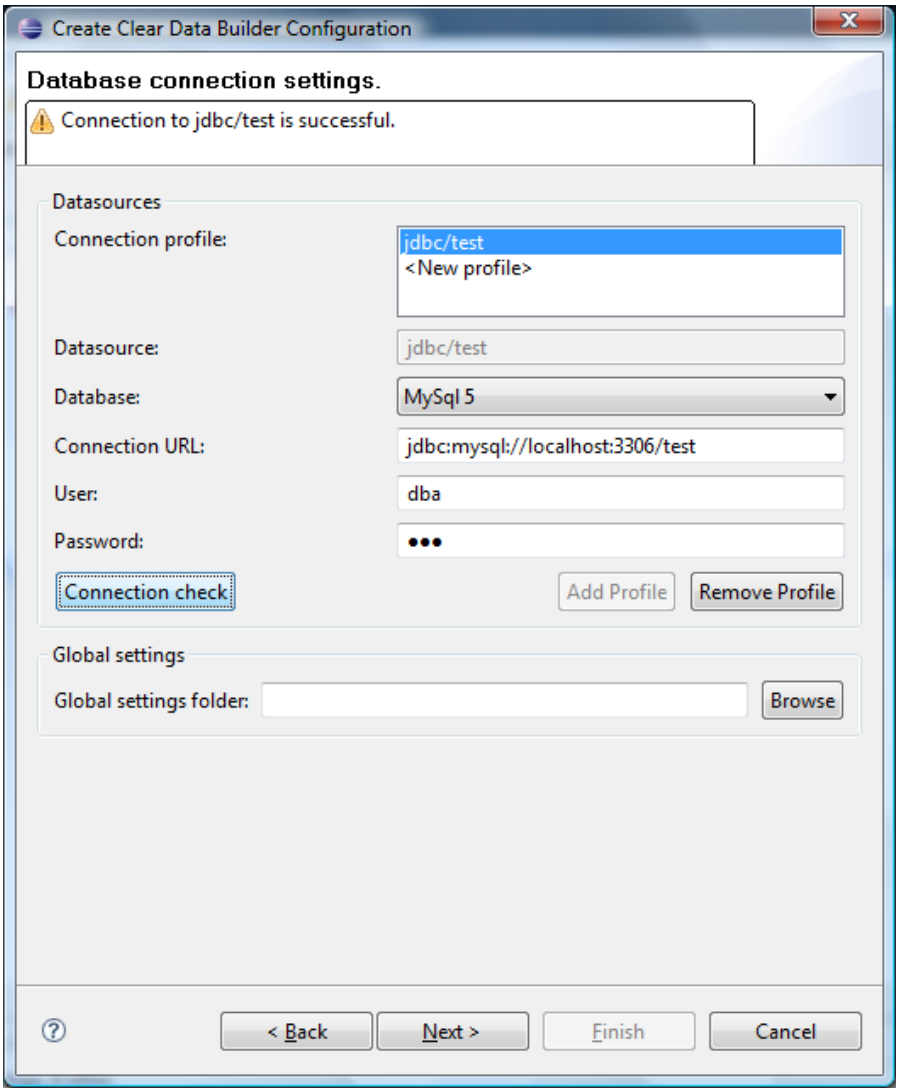
The next screen is required to select an application server type (i.e. Tomcat) to be used for deployment of the example application that we are creating. Just double-check that the Server Root and Server Root URL have the right values.

Finally, you'll need to specify parameters of the database connection pool.

Note. Do not forget to press the button **Add Profile**.

[1] If you use MySQL Server, just run Restore Database option in MySQL Administrator

The window below shows a sample configuration of the MySQL Server database called test – this is our database with the information about employees.



In the screen above, dba/sql are used as the credentials for getting the database connection, and the button **Connection check** can be used to test the connection.

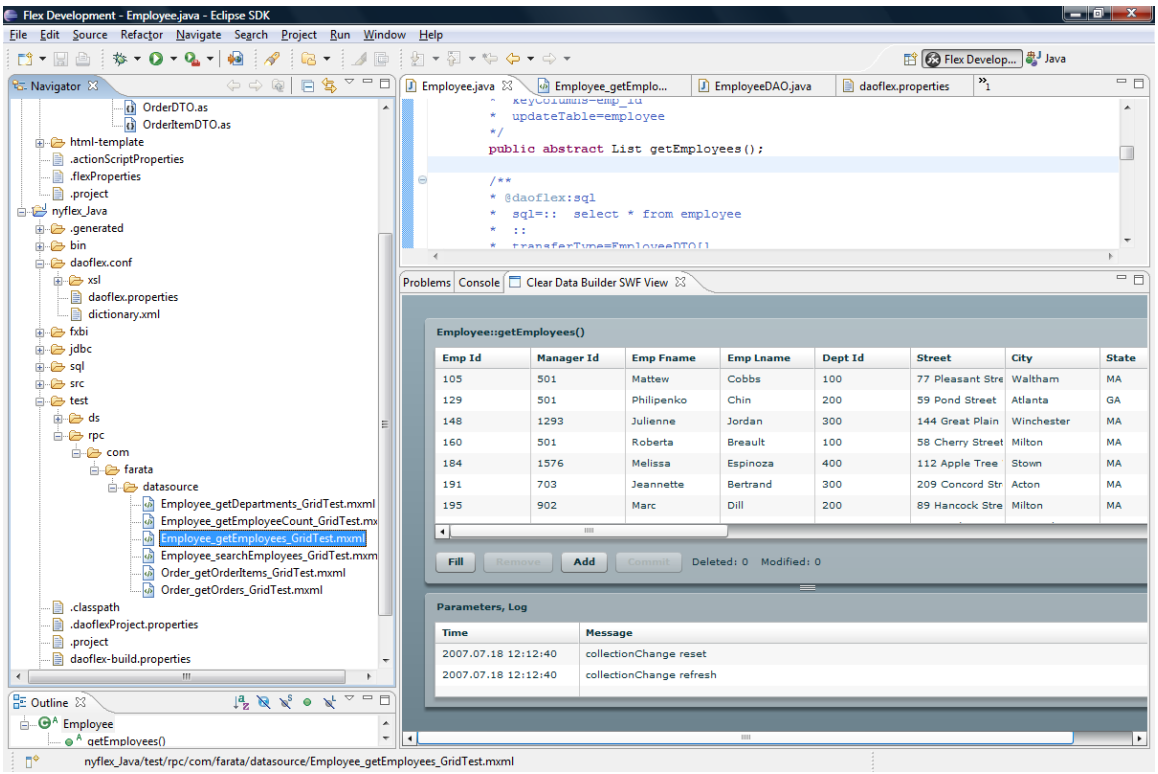
Press the buttons Next and Finish, and wait for another 20 seconds until the Ant build successfully completes.

This build will generate and deploy all required files (including LiveCycle Data Services Express) under your application server.

You'll also find a number of Java classes (EmployeeDTO.java, EmployeeDAO.java et al) under the .generated folder in your Eclipse Java project.

4.1.3 Test your generated CRUD application

Restart your application server. Return to Eclipse Java project; expand the folder **test**, and all folders under **rpc.** You'll see there a file Employee_getEmployees_GridTest.mxml among several others. Right click on this file, select the menu **ClearData Builder > Preview**. You'll see our CRUD application displayed in Eclipse CDB view pane. Maximize the view and press the button Fill to populate with the data on employees:



Now you can add, update and delete the data from the database using this demo application.

Note. If you'd like to run this demo as a regular Flex application (not in the CDB Preview mode), copy Employee_getEmployees_GridTest.mxml in your Flex project, and run it as any other Flex application – Flex Builder will generate the HTML wrapper and start it in your Web browser.

4.2 Test the sample ClearBI report

We can reuse the CRUD application buit by CDB to test a sample ClearBI report. To be able to preview or edit sample reports, you need to enable ClearBI Developer or Server Edition according to your license. Just right click on the Flex project, select Clear BI and enable it.

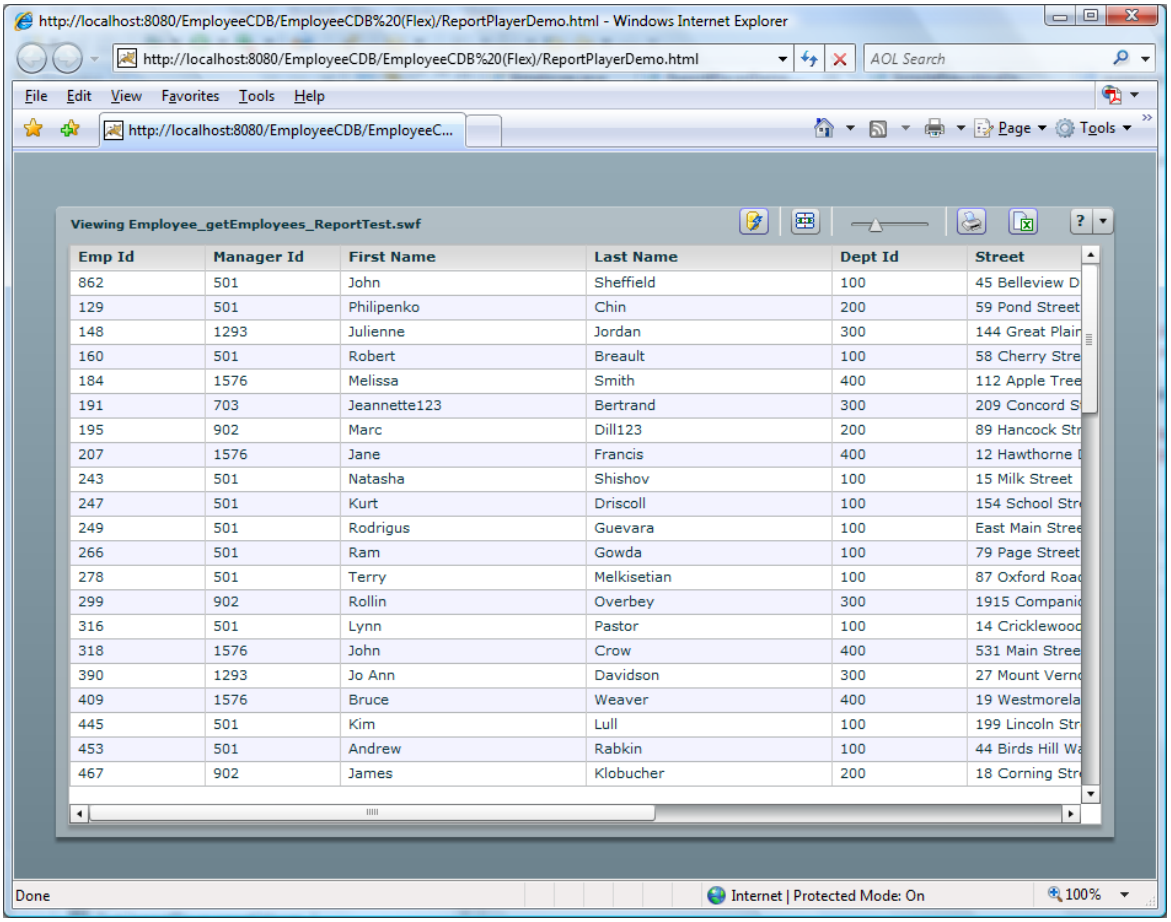
4.2.1 Enabling ClearBI Developer edition

Generation of ClearBI reports includes a number of additional resources that will be added to your Flex project automatically if you right-click on its name and select **ClearBI > Enable ClearBI Developer**. You'll find the following new files in the root directory of your Flex project:

ReportPlayerDemo.mxml – a reference implementation of the ClearBI Player that expects one parameter: the report's SWF, which can be produced interactively by running the menu option ClearBI > Compile. Notice the corresponding SWF that will appear in the fxbi folder of your Java application. Copy this SWF file to your Flex project and set reportUrl property in the source code of the file ReportPlayerDemo.mxml to the name of the report SWF. For example,

```
<control:ReportPlayer id="reportPlayer"    reportUrl="Employee_getEmployees_ReportTest.swf"
                                           reportModuleReady="reportPlayer.report.fill()" />
```

Run the ReportPlayerDemo application and it'll generate a report that may look as follows:



You'll see a new folder **control** in your project with a custom component ReportPlayer.mxml, which is a reference implementation of a custom control to support the demo application ReportPlayerDemo.

The **assets** folder contains the images used in the ClearBI toolbars created in your Web browser.

The file supergrid.css is a style sheet for the *supergrid* component, which is the base component for all ClearBI reports.

4.2.2 Enabling ClearBI Server edition

Upgrade to ClearBI Server edition adds the files described in section 4.2.1, and also creates the following files:

DBReportPlayerDemo.mxml - a player for published reports. To use it, note the report id, displayed in the title of the Editor dialog when you publish the report. Then replace the reportID in this mxml (see the tag <control:DBReportPlayer reportID="8"/>) with the id of the published report. Replace "com.farata.datasource.dto.EmployeeDTO" with the DTO object from your real-world project.

ReportingAdminDemo.mxml - a reference implementation of the report administration application that supports user's authorization, maintaining users groups and roles, and organizing reports in folders. See more details in section 9 of this manual.

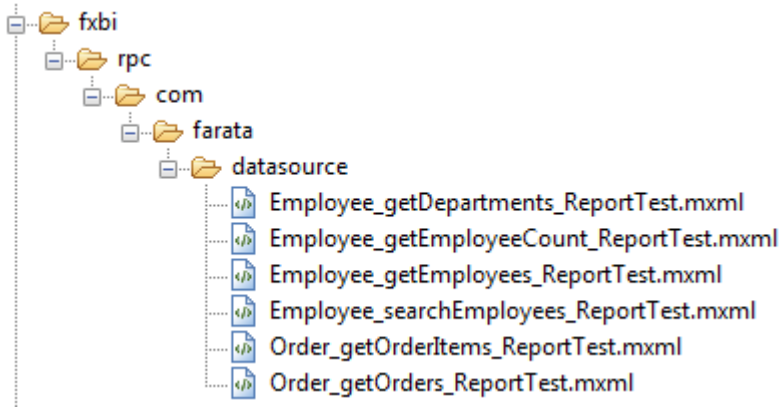
SimpleReportingDemo – a reference implementation of ClearBI, that can work with the following components from the **control** folder: DBReportPlayer, ReportEditor, ReportPlayer, and SimpleReportEditor. These components illustrate how you can integrate ClearBI in your applications, and you can test these reference implementations by changing the source code of SimpleReportingDemo to point to one of these components, for example:
`<control:ReportEditor xmlns="*" />`

SimpleReportingWithCharts – a reference implementation of creating charts in ClearBI.

The upgrade to the Server edition will also create in your Flex project the folder **lib** that contains a CompositionLib.swc library. You'll also find a new folder **sql** containing the scripts (i.e. clearbi_mysql.sql) for creating database tables required for organizing and storing report layouts and user administration.

To work with ClearBI, you'll also need to set up database/connection pool described in section 11.1. When this is done, your ClearBI is ready to work.

Browse the directory structure in the example Java project that you've generated. Note the directory **fxbi** containing several MXML files.



Each of these MXML files can be compiled into a SWF file so you can run them in Flash Player using the menu **ClearBI > Preview**.

ClearBI Developer Edition also allows editing and formatting of each column, setting sorting and filtering, creating groups, computed columns, exporting reports to Microsoft Excel and much more.

On the server side, upgrade to ClearBI Server edition deploys a number of Java libraries (jar files) in your Web application. For example, in case of Apache Tomcat, it adds the following jar files into the WEB-INF\lib directory of your Web application:

- antlr-2.7.5.jar –a library that supports ANTLR, an open source language tool that provides a framework for constructing recognizers, interpreters, compilers, and translators from grammatical descriptions containing actions, see www.antlr.org
- daoflex-FlexBI-Admin and daoflex-FlexBI-Admin-generated – administrative module for ClearBI
- flexBI.jar – a report compiler
- flexBI-composition.jar and flexBI-composition-generated.jar – code that implements report persistence
- flexBI-web-env.jar – code that implements an API that allows another application call FlexBI as a compiler from Eclipse or another Web application. It reads the current setting from a Servlet container and converts them into compiler's directives.

Upgrade to ClearBI Server edition also includes the following filter in the web.xml of your Web application:

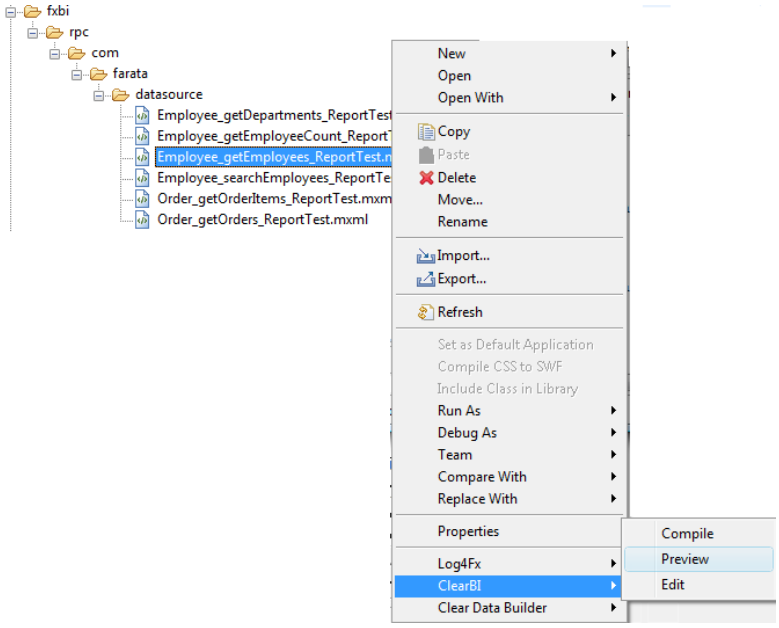
```
<filter>
     <filter-name>clearbi-compiler-invocation-interceptor</filter-name>
  <filter-class>
     com.cti.compiler.env.web.CompilerInvocationInterceptor
  </filter-class>
 </filter>
<filter-mapping>
       <filter-name>clearbi-compiler-invocation-interceptor</filter-name>
        <servlet-name>MessageBrokerServlet</servlet-name>
</filter-mapping>
```

This filter optimizes the process of SWF content caching. It automatically recompiles report's SWF if need be.

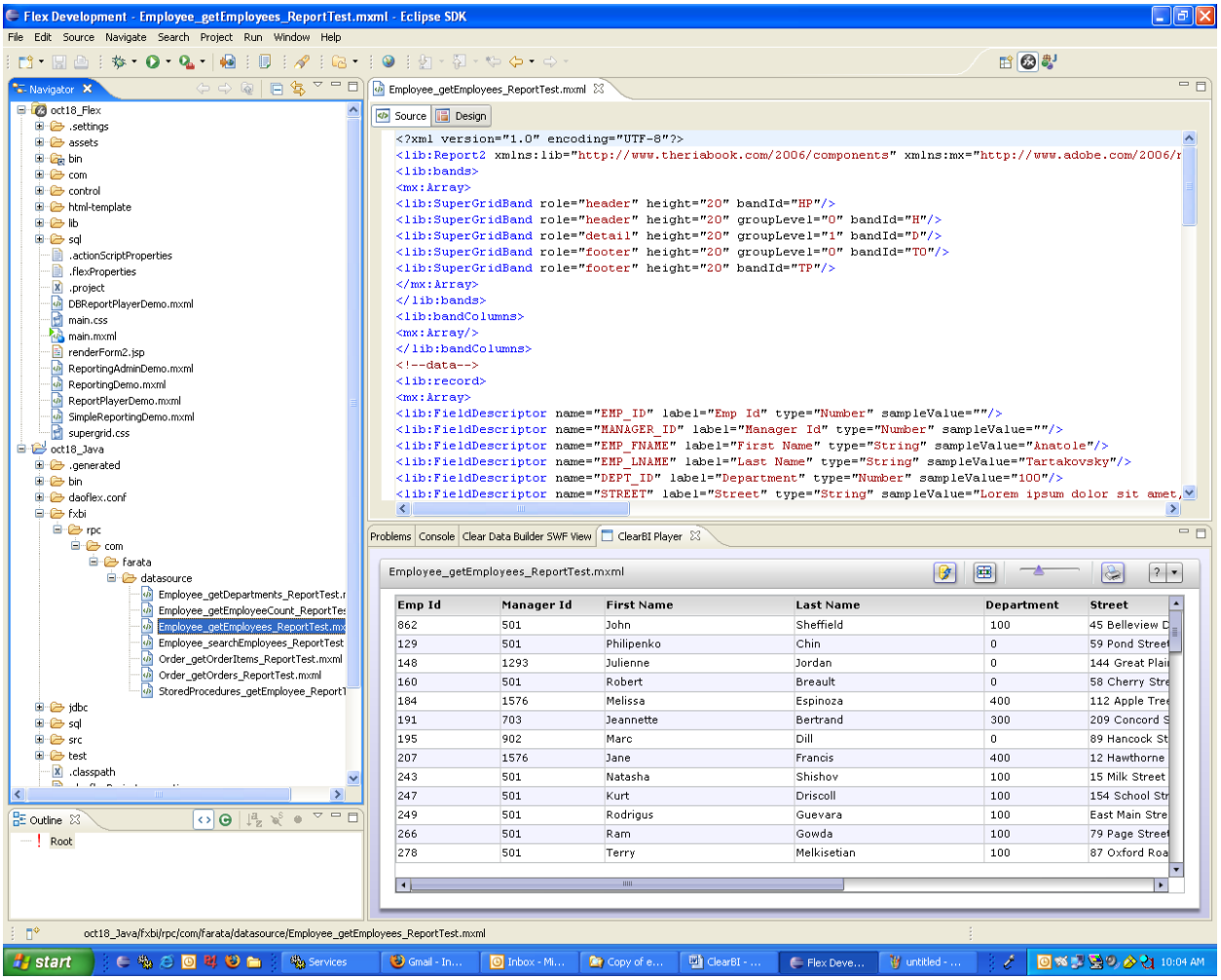
4.3 Previewing ClearBI Reports

Your report is based on generated MXML files, and you can preview it by performing the actions listed below.

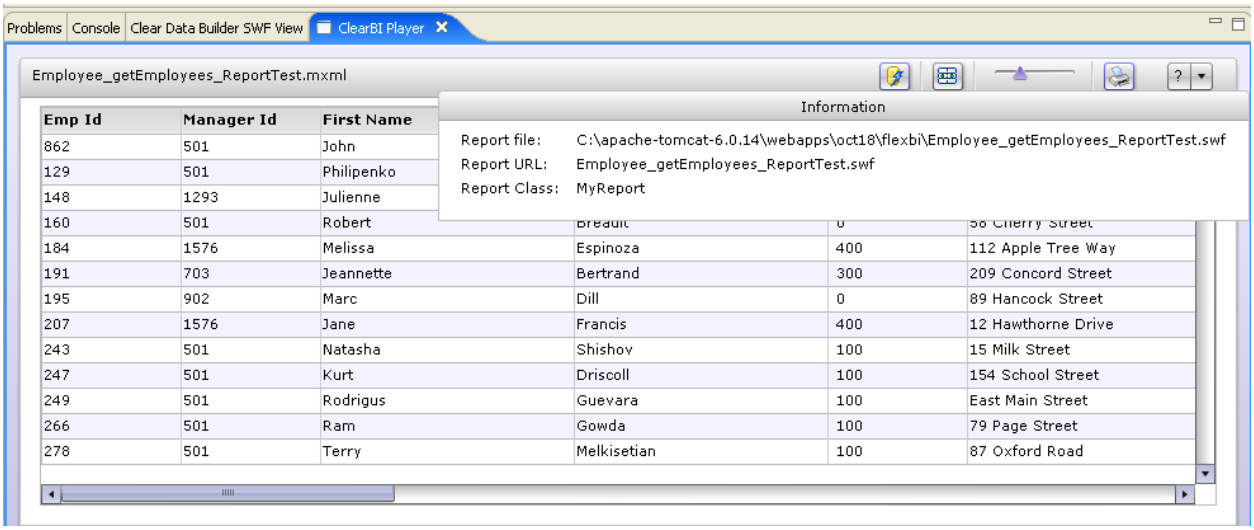
1. Right-click on one of the MXML files and select the menu item **ClearBI > Preview**.



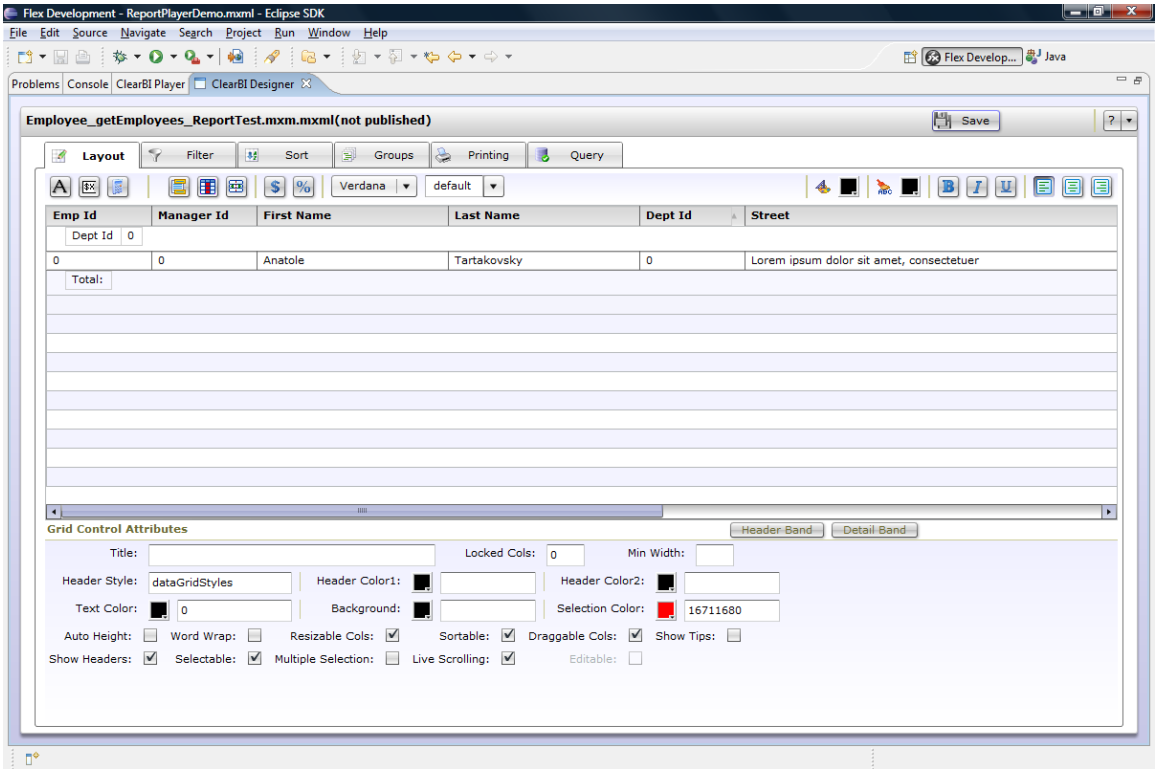
The sample report may look as follows:



Click on the question mark on the right, and you'll see the name an location of the compiled report as shown below:



If you'd like to edit the report, select the menu **ClearBI > Edit**, which will bring you to the ClearBI editor that allows you to customize reports by adding grouping, changing style, applying formulas, filters et al.



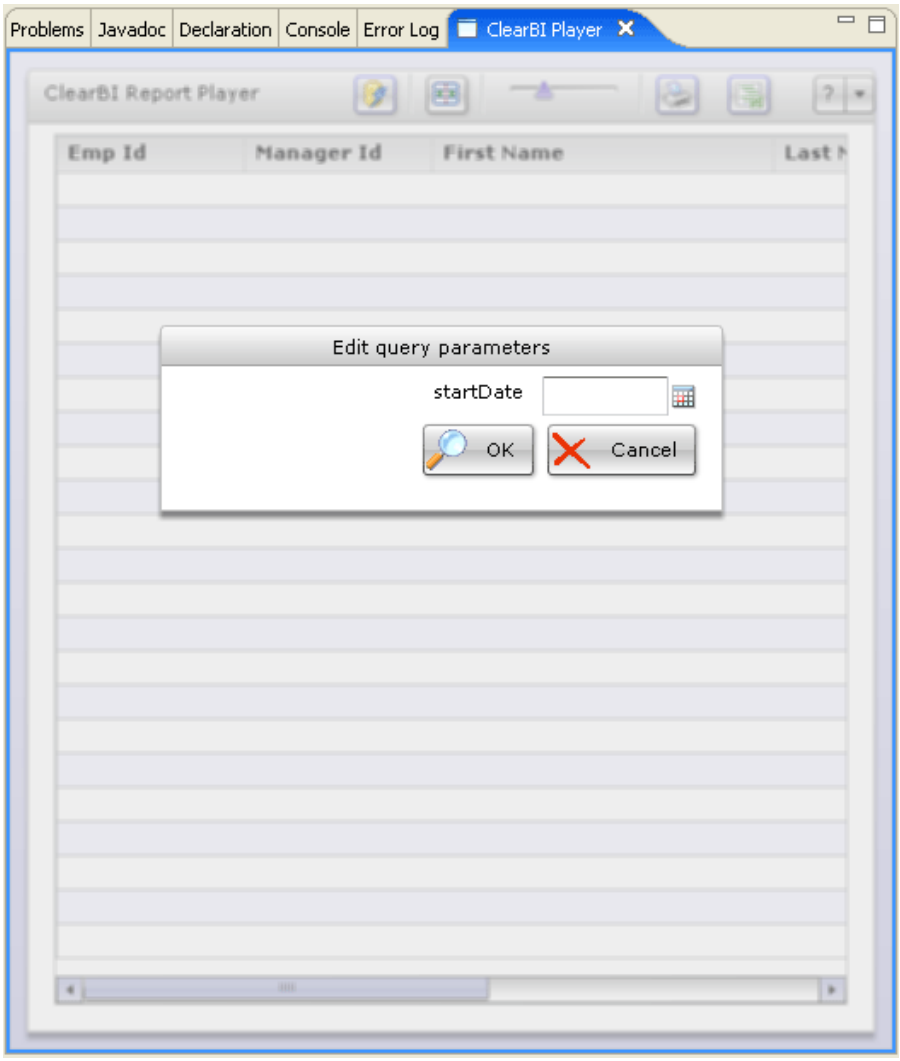
We'll customize this sample report later in this document.

2. If the selected report requires input parameters, you'll see the prompt screen as shown on below.

   If you use an SQL SELECT statement that requires parameters, your report will prompt for their values. Let's say you've generated a report from the following annotated Java method:

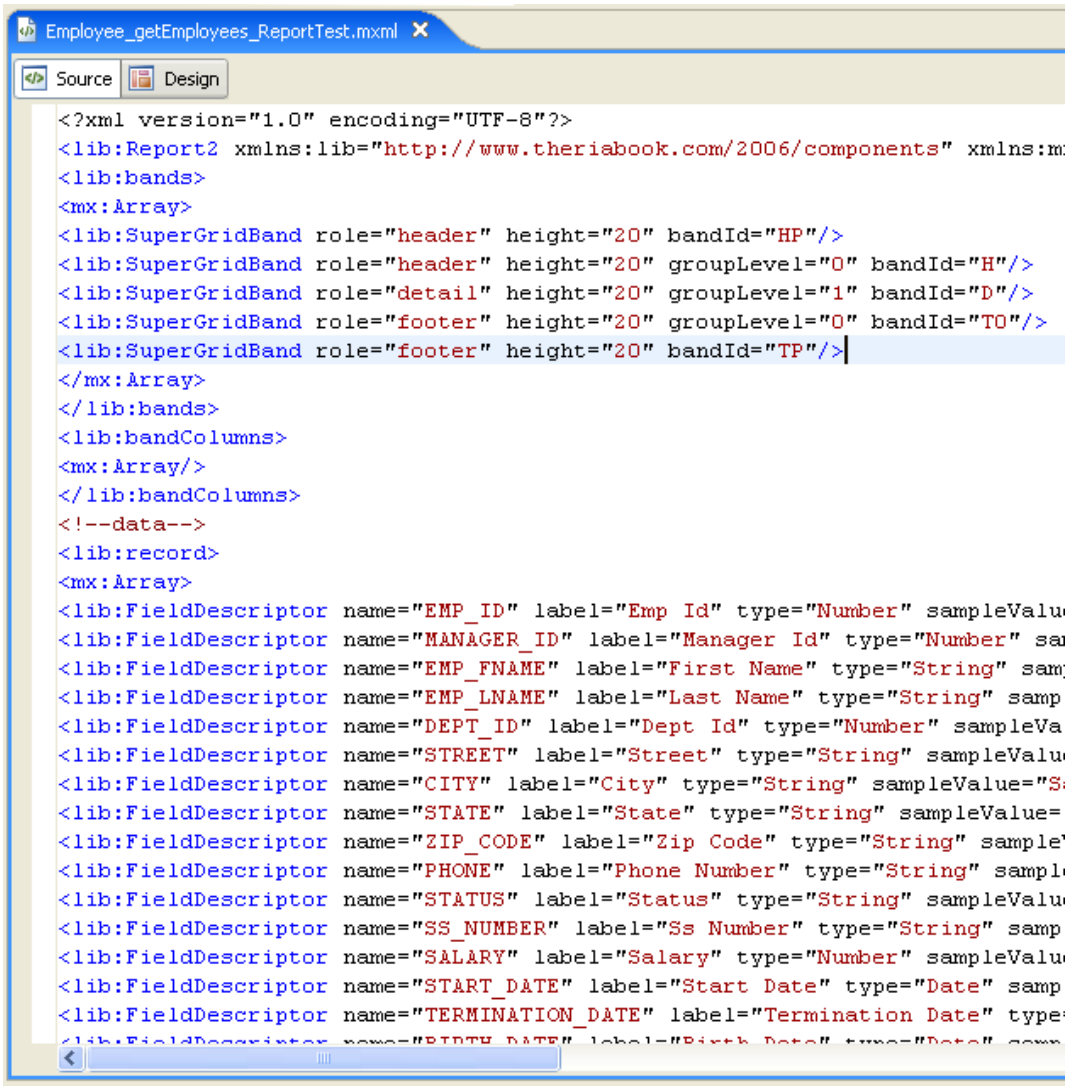
```
/**
* @daoflex:sql
*  sql=::  select * from employee where start_date < :startDate
          or start_date=:startDate
*  ::
*/
public abstract List getEmployees(Date startDate);
```

Start this report in ClearBI Player, and you'll be prompted to enter the value of the start date.



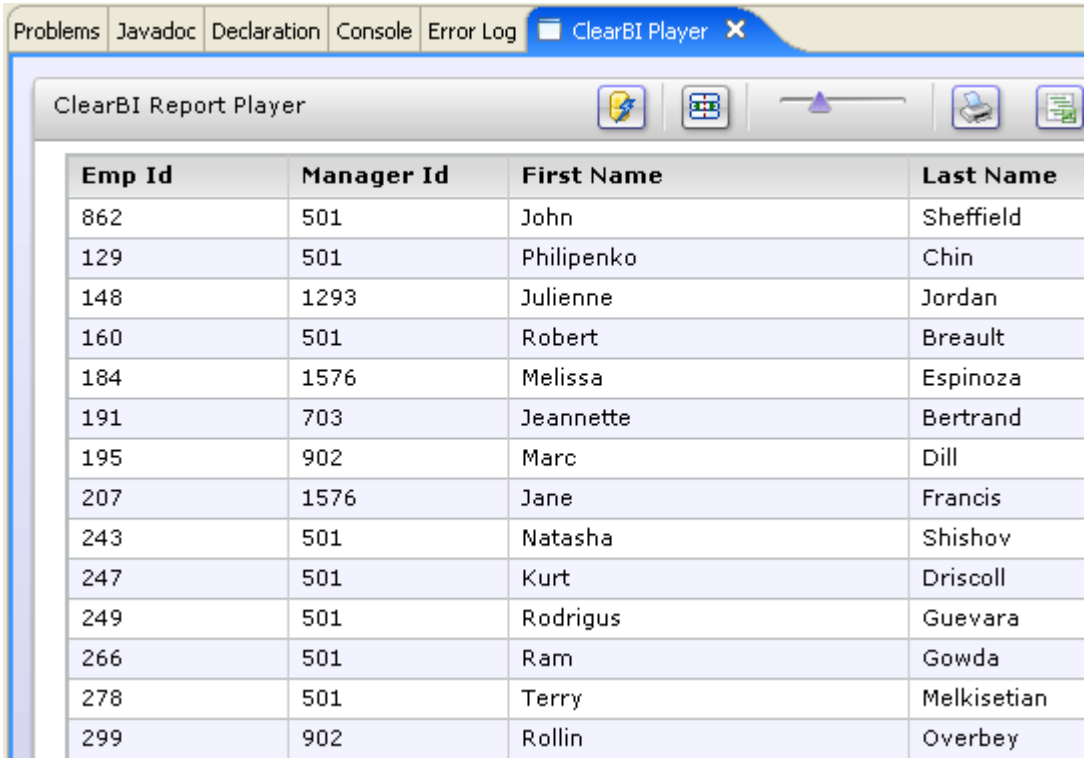
3. After entering parameters, if any, you'll see the report that was generated based on the metadata contained in the selected MXML application.

ClearBI generates report using our custom component called supergrid, which hides the coding complexity. As a result, the code of generated report applications looks pretty straightforward as a grid of rows and columns as shown below:



You can manually edit report's MXML files to customize the report.

Selecting ClearBI Preview on Employee_GetEmployees_TestReport.mxml would display a sample report that may look as follows:



Emp Id	Manager Id	First Name	Last Name
862	501	John	Sheffield
129	501	Philipenko	Chin
148	1293	Julienne	Jordan
160	501	Robert	Breault
184	1576	Melissa	Espinoza
191	703	Jeannette	Bertrand
195	902	Marc	Dill
207	1576	Jane	Francis
243	501	Natasha	Shishov
247	501	Kurt	Driscoll
249	501	Rodrigus	Guevara
266	501	Ram	Gowda
278	501	Terry	Melkisetian
299	902	Rollin	Overbey

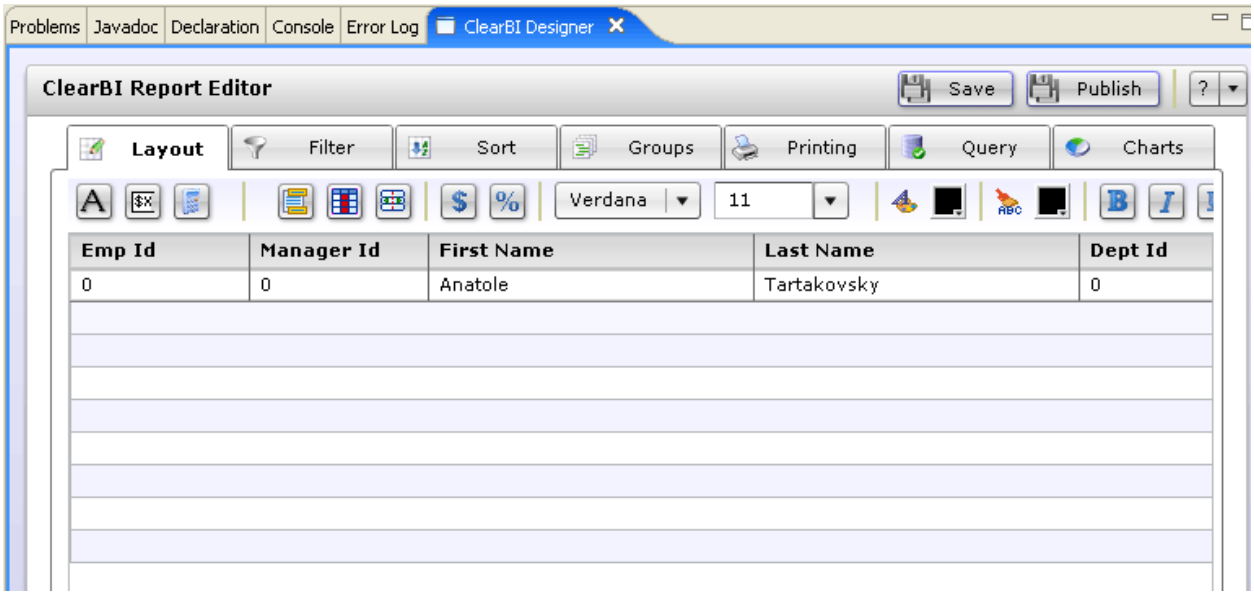
4.4 Editing ClearBI reports

The ClearBI Edit option allows you to customize the style of your report, add computed columns and perform other actions that will change the report's look and feel.

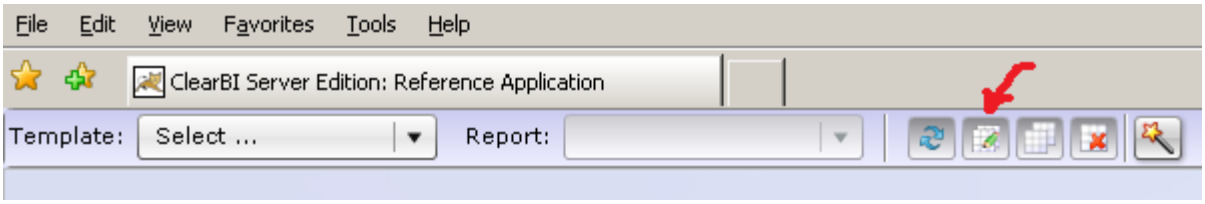
ClearBI Server edition allows end users to edit reports by pressing an Edit icon on the ClearBI toolbar in the Web browser. Developer's edition allows editing report inside Eclipse.

To start report editing from Eclipse IDE, perform the following steps:

1. Right-click on the MXML file that you want to edit and select the menu option **ClearBI > Edit**. You'll see your report in the ClearBI Editor window:



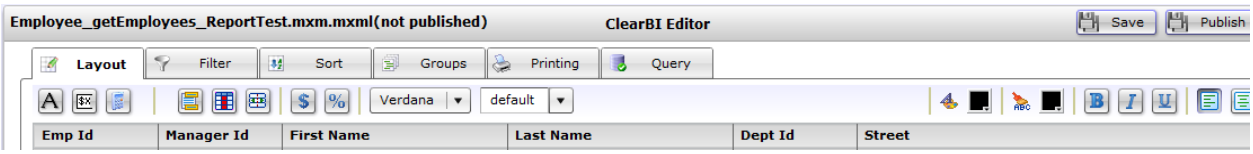
If you are using ClearBI Sevrer edition from the Web browser, you can start the editor by pressing the button on a toolbar as shown below.



Now you can customize your report, by adding computed columns, specify grouping, filtering, applying formulas, et al. After editing is complete, press the button Save and run your report in the Preview mode. The process of report customization is described in greater details in section 5.0.

5.0 CUSTOMIZING CLEARBI REPORTS

In this section we'll discuss the ClearBI Report Editor in details. The main area of the ClearBI editor is a tab folder with the following tabs: Layout, Filter, Sort, Groups, Printing, and Query.



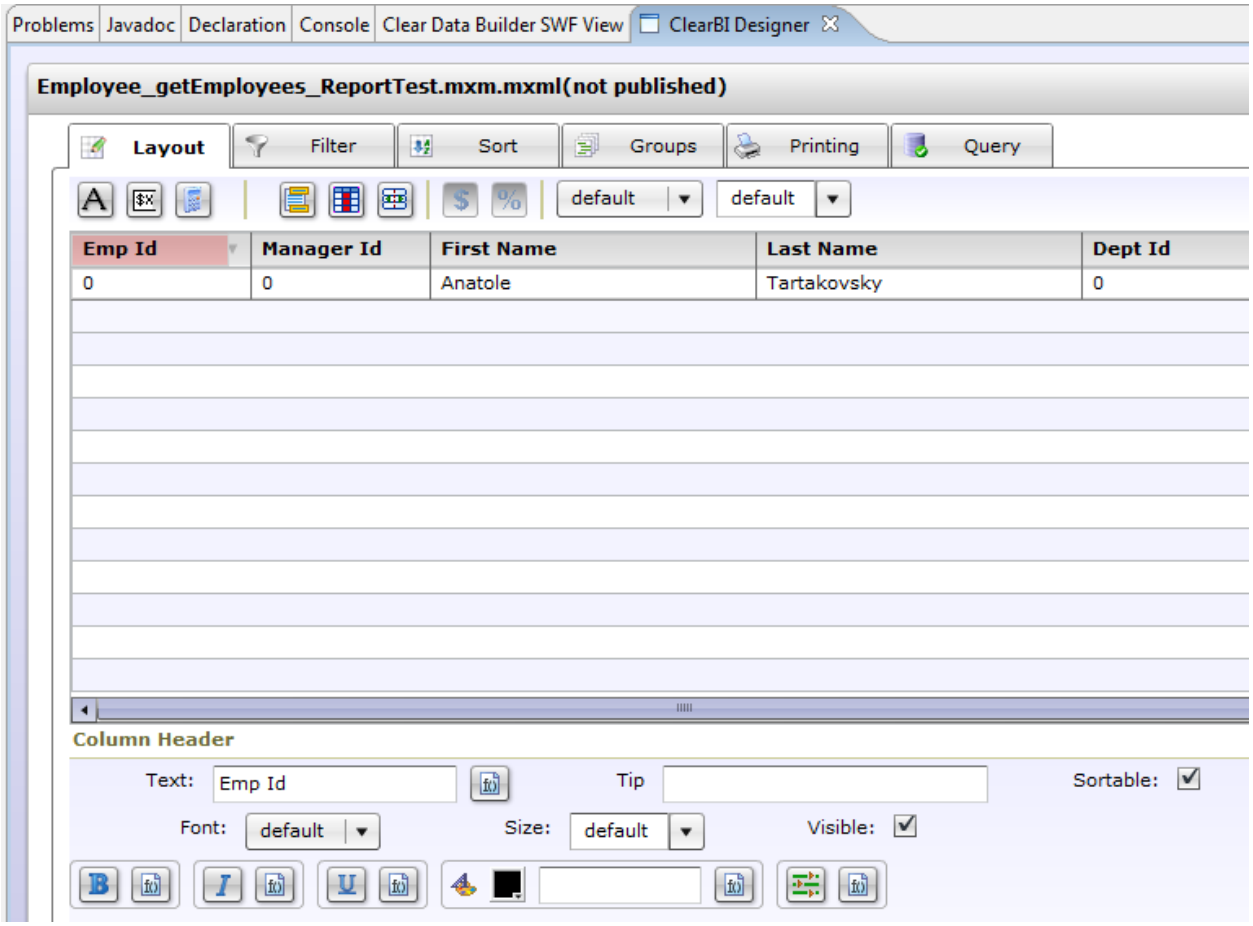
Note two buttons – **Save** and **Publish** at the top right corner of the editor's window. The former allows you to save your report customizations in the database. Pressing the button Save also re-compiles the report. The button **Publish** not only saves the report in the database but also allows the end users to work with this report in a Web browser with installed Flash Player.

5.1 Layouts and styles

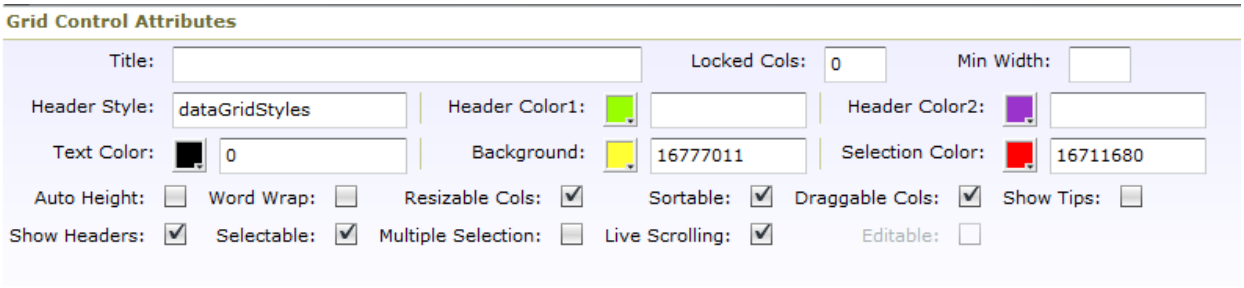
You can change the style of each column in the tab Layout. You can change column names, font (including italic, underscore and bold), size, add prompts, set alignment and colors.

You can specify which columns will be shown and if they should be sort-enabled.

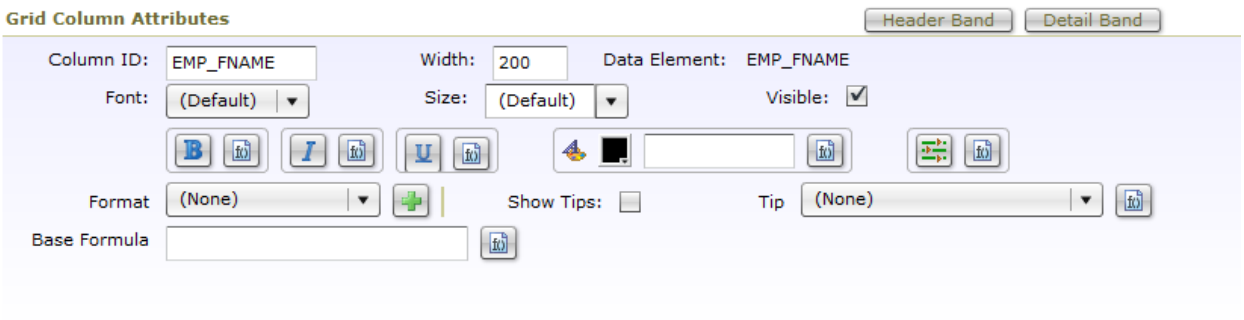
You can also set the auto-size for the columns and text word wrap. The content of the Layout tab changes depending on where the focus is. If you click on the header of the column, the Layout panel looks like this:



To customize the grid area of the reports, click in the banded area of the screen and select the colors and properties of individual columns.



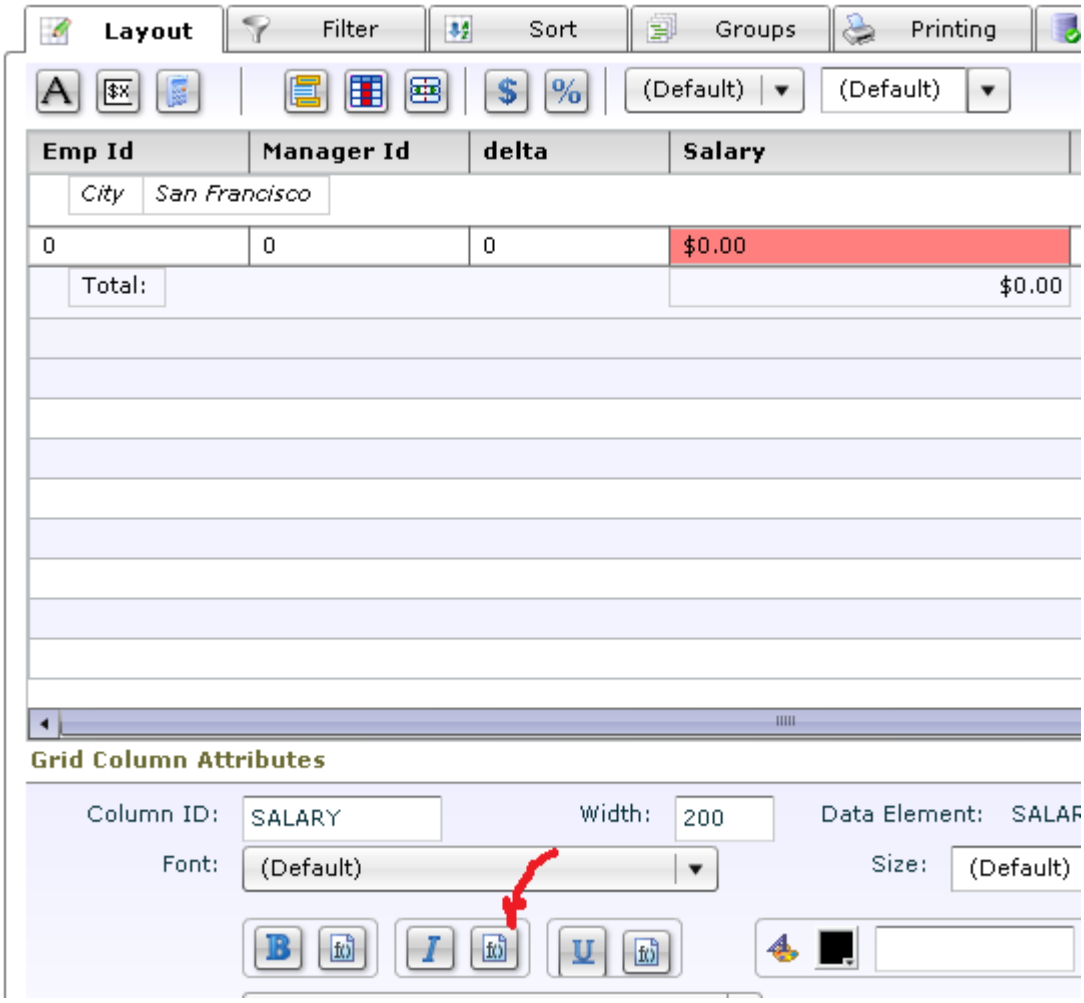
Selection of the data cell right under the title of the report changes the lower portion of the ClearBI editor:



5.1.1 Formulas in styles

Please note several Formula  buttons – you can conditionally customize the look and feel of various column attributes by applying style formulas.

Let's say you want to conditionally apply italic font to all salaries lower that $40000. Select the salary cell and click on the formula icon next to the  button.



This will open the formula painter that will look as follows:

Using drag and drop (or double-click) copy the formula template from left to right and modify it to look as follow:

If(salary<40000, "italic", "normal")

Note. You can verify the correctness of your formula by pressing the button Verify in the screen above. You can use Boolean expression in your formulas by using the keywords AND, OR and NOT.

Save the changes and preview the report. The salary values that are lover than 40000 are shown in *italic*.

Salary	First Name	Last Name
$87,900.00	John	Sheffield
$38,500.00	Philipenko	Chin
$51,432.00	Julienne	Jordan
$57,490.00	Robert	Breault
$36,490.00	Melissa	Johnson
$32,780.00	Jeannette	Bertrand
$54,800.00	Marc	Dill

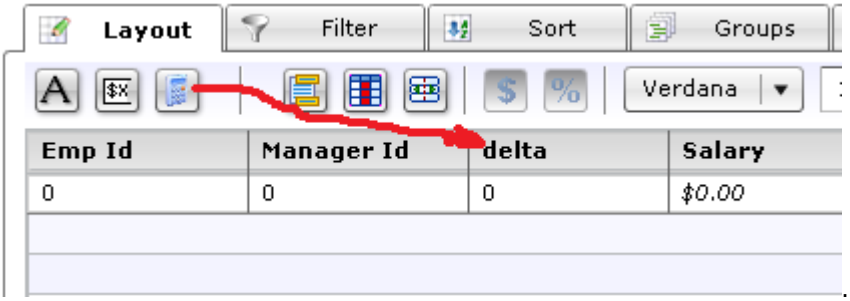
You can apply formulas to change the style of both data and the headers of your report.

5.1.2 Creating computed columns with formulas

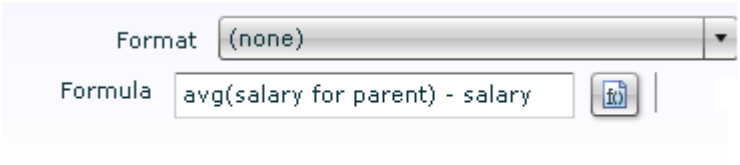
While in the previous section we've been changing styles of single values, you may also create new computed columns, apply formulas to an entire column, for example calculate the total salary for the entire report or for a group. In this section we'll go through a simple example that will illustrate the concept.

Let's create a new column that will contain the difference (deltas) between a averagen salary and each individual salary.

This time we'll drag the button  from the editor's toolbar to the detail band of the report



This will create a new computed column with colum title SUM. Change it to delta, and enter the formula `avg(salary for parent)` in the field for the grid column :



This column will calculate the average value for the parent (the entire report in this case) and subtracts the value if the salary in the current row.

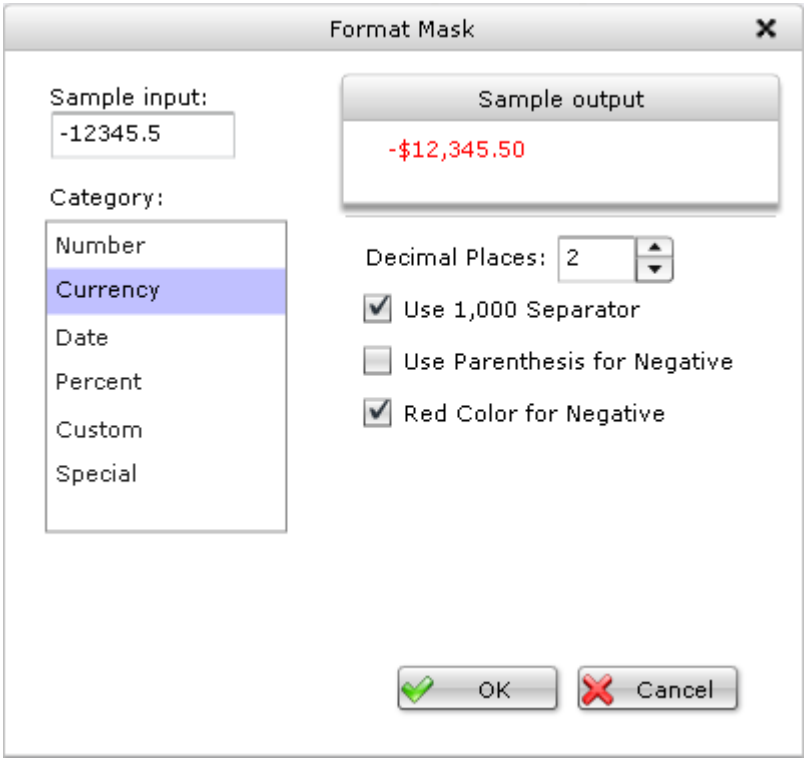
Emp Id	Manager Id	delta	Salary	First Name
862	501	-36838.194246[illegible]	$87,900.00	John
129	501	12561.805753[illegible]	*$38,500.00*	Philipenko
148	1293	-370.1942465[illegible]	$51,432.00	Julienne
160	501	-6428.19424657	$57,490.00	Robert
184	1576	14571.805753[illegible]	*$36,490.00*	Melissa
191	703	18281.805753[illegible]	*$32,780.00*	Jeannette
195	902	-3738.19424657	$54,800.00	Marc
207	1576	-2808.19424657	$53,870.00	Jane
243	501	-21933.194246[illegible]	$72,995.00	Natasha

If this report would define some groups, you'd be allowed to drop this computed column on the group band and calculate an average salary for each group. In section 6.1.2 you can see an example of adding a formula for calculating of subtotal within a group. Section 6.1.3 has more details on creating formulas.

The output of the delta column can be formatted using one of the format masks described in the next section.

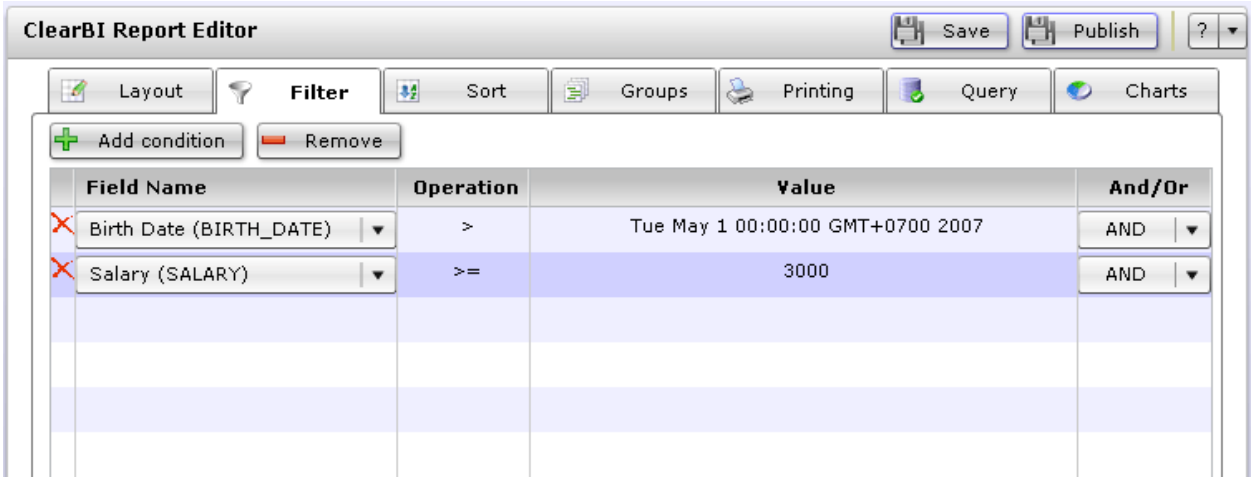
5.1.3 Using format masks

For special columns you can use such styles as numeric data (decimal paces, 1,000 separator, red color and/or parenthesis for negative), currency and percent symbols, some special styles like phone number, social security number and zip code.



5.2 Filter

A Filter tab allows you to create and apply data filters to your report. Any filter consists of one or more conditions, which can be added by the click on the button **Add condition** or removed with the button **Remove**. Each condition is an expression that consists of a field name, operation (<, >, ==, …) and the value to check against. If you have more than one condition, you can link them together by applying boolean operands AND or OR.

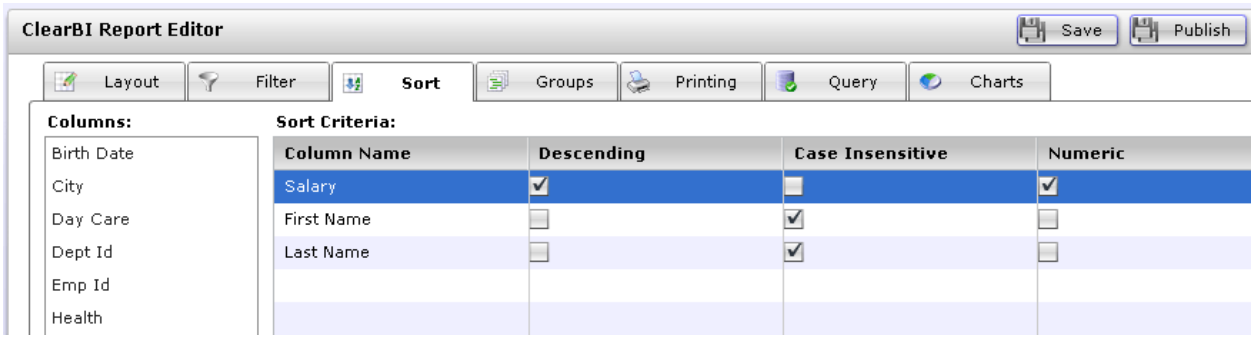


The check box **Manual edit** will allow you to edit the selected filter expression manually (see the bottom part of the Filter tab).

The button **Verify** checks the correctness of the filter expression.

5.3 Sort

The tab Sort is used for sorting your report's data. Drag the required columns from the left to the Sort Criteria area. You can specify the sorting criteria, and for each criterion specify descending/ascending order, case sensitivity, and if the field selected for sorting is numeric or not.



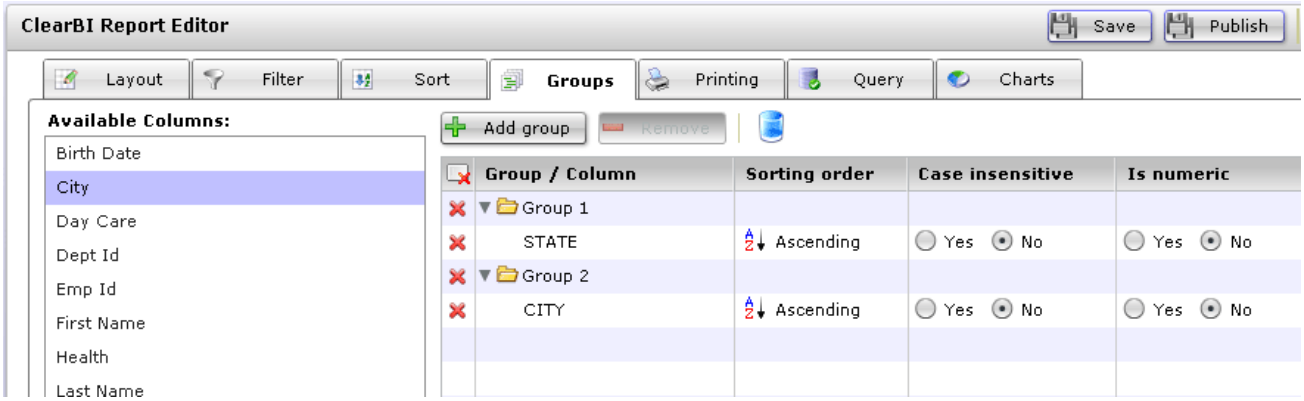
5.4 Groups

In this tab you can specify grouping for your report, for example group the data by the state and/or by city.

The button **Add group** creates a new group that is represented by a band; you can also create a group by simple dragging of the column name to the Group/Column area on the right.

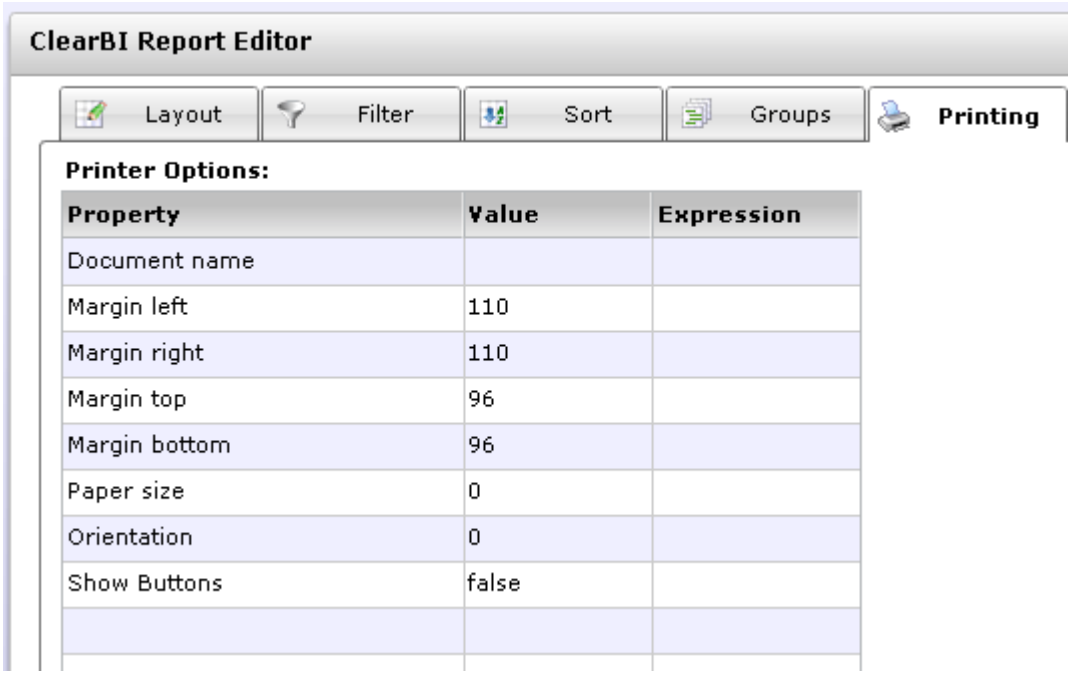
You can assign a filter and specify the sorting order for each column in the group.

Add subtotals for each group for numeric columns if needed.



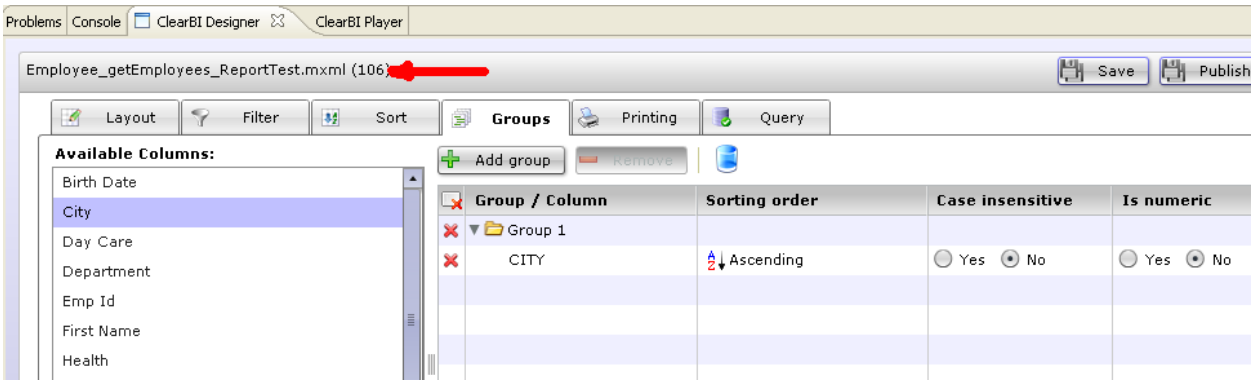
5.5 Printing

This tab is used for setting the print page parameters, the report name and more.



5.6 Publishing reports for end-users

To publish a report you press the button Publish in the editor window. When the report is published, its layout is saved in the database and it gets a unique id. In the screenshot below you can see a sample of an editor window right after the report has been published and saved in the clearbi database under id under id 106:

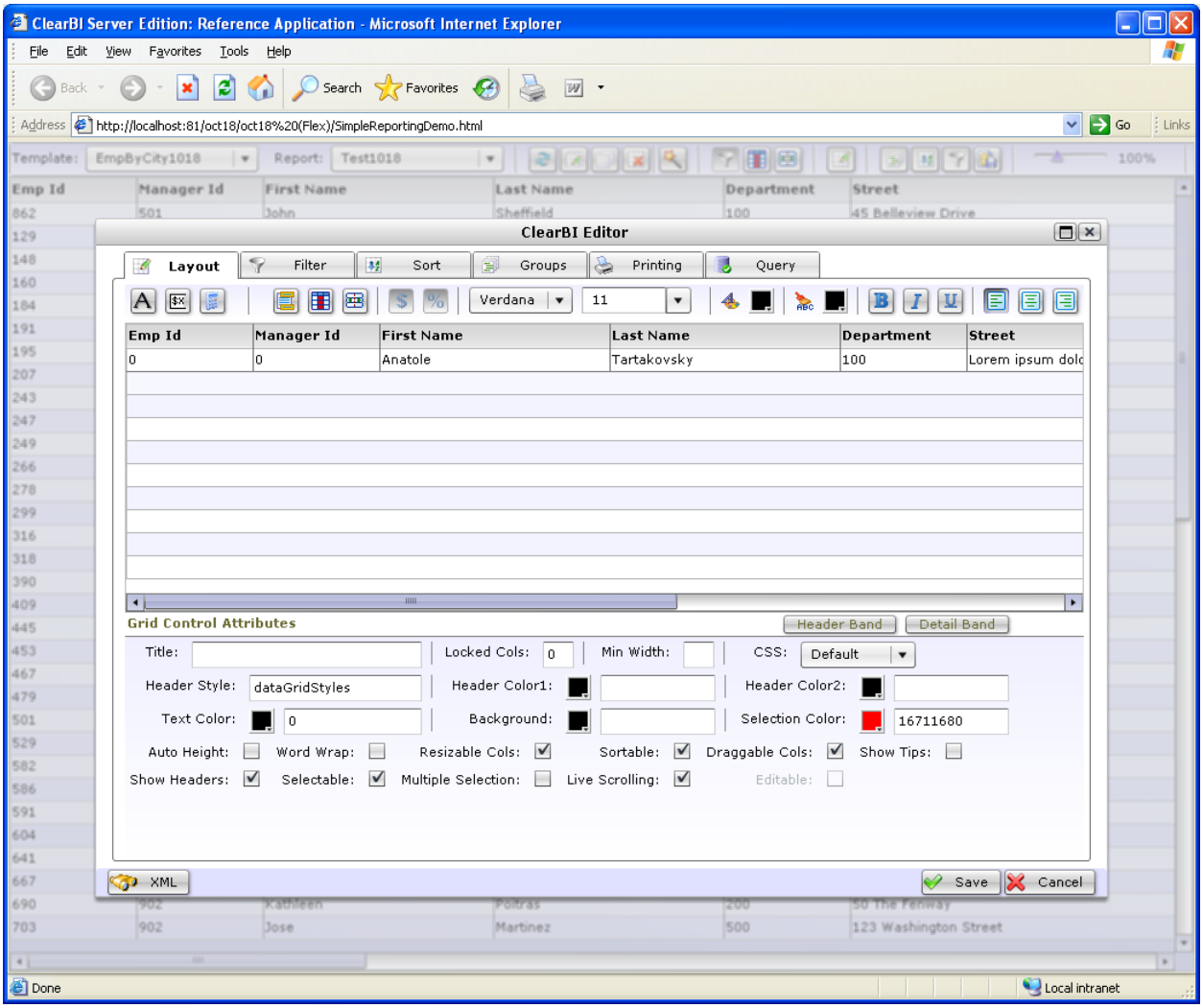


If you want to publish this report for your super users who need to be able to edit reports from the Web browser, you can use the code of a SimpleReportingDemo.mxml.

```
<?xml version="1.0" encoding="UTF-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
      pageTitle="ClearBI Server Edition: Reference Application"
      xmlns:control="control.*">
 <mx:Style source="supergrid.css"/>
 <mx:Style source="main.css"/>

  <control:SimpleReportEditor xmlns="*" viewId="106"/>
</mx:Application>
```

Just add the property viewId with the id of the published report and run it as a Flex application.



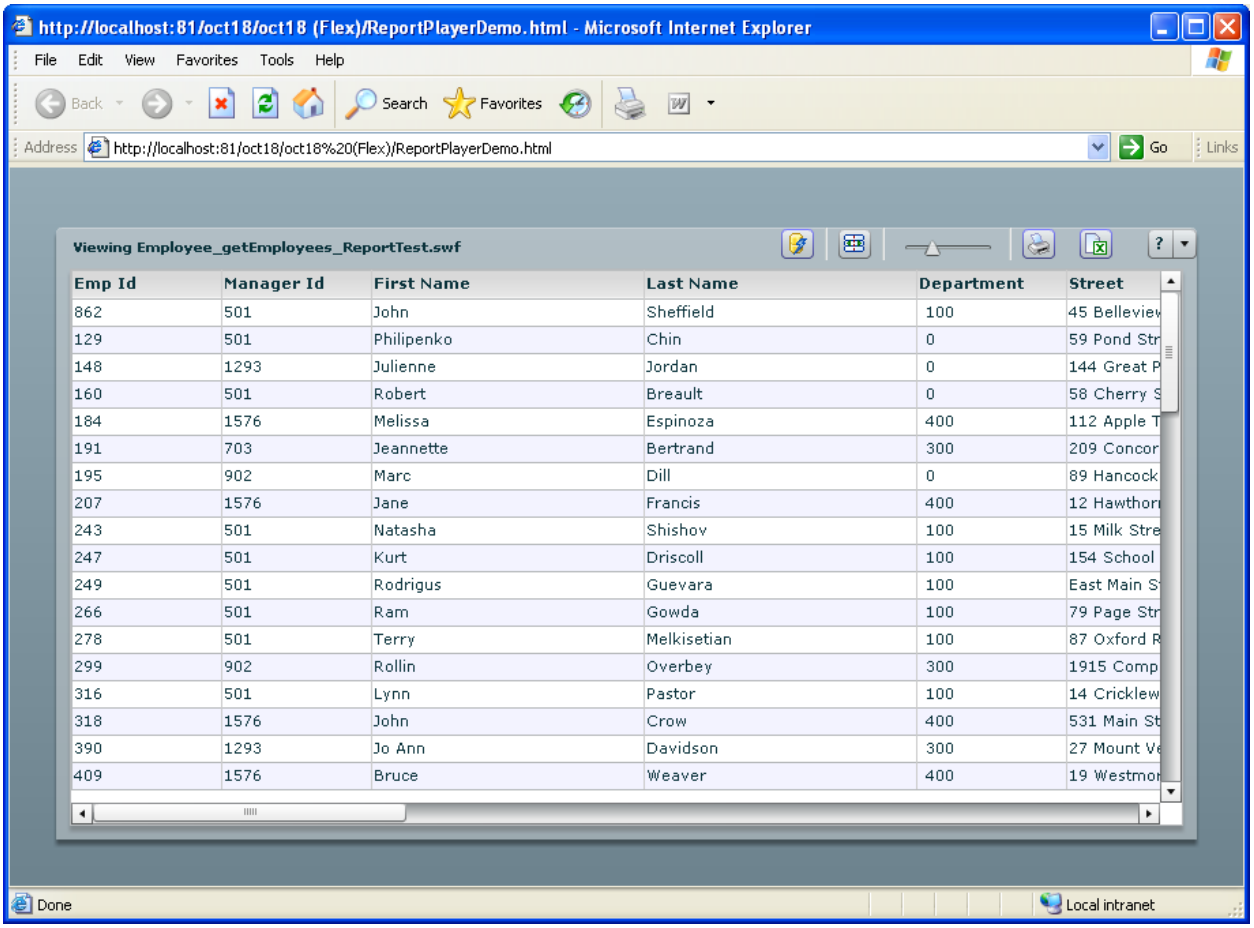
Note. Publishing reports that can be customized by the end-user requires Server Edition license.

If you'd like to publish a report for regular users that can only view the report, use provided ReportPlayerDemo.mxml. You'll need to compile your report into swf (i.e. by using see the menu ClearBI > Compile) and then modify the file ReportPlayerDemo.mxml to include the location of the report in the reportURL attribute. For example,

```
<?xml version="1.0" encoding="UTF-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical"
      xmlns:control="control.*" xmlns="*">
<mx:Style source="supergrid.css"/>
<mx:Style>
      Text {fontWeight:bold; color:green}
</mx:Style>
…
<control:ReportPlayer id="reportPlayer"
reportModuleReady="reportPlayer.report.fill()"
      reportUrl="Employee_getEmployees_ReportTest.swf"/>
```

```
</mx:Application>
```

Run the ReportPlayerDemo.mxml as Flex application, and you'll see a report with a simplified toolbar that does not have edit and several other buttons.



5.7 Charting

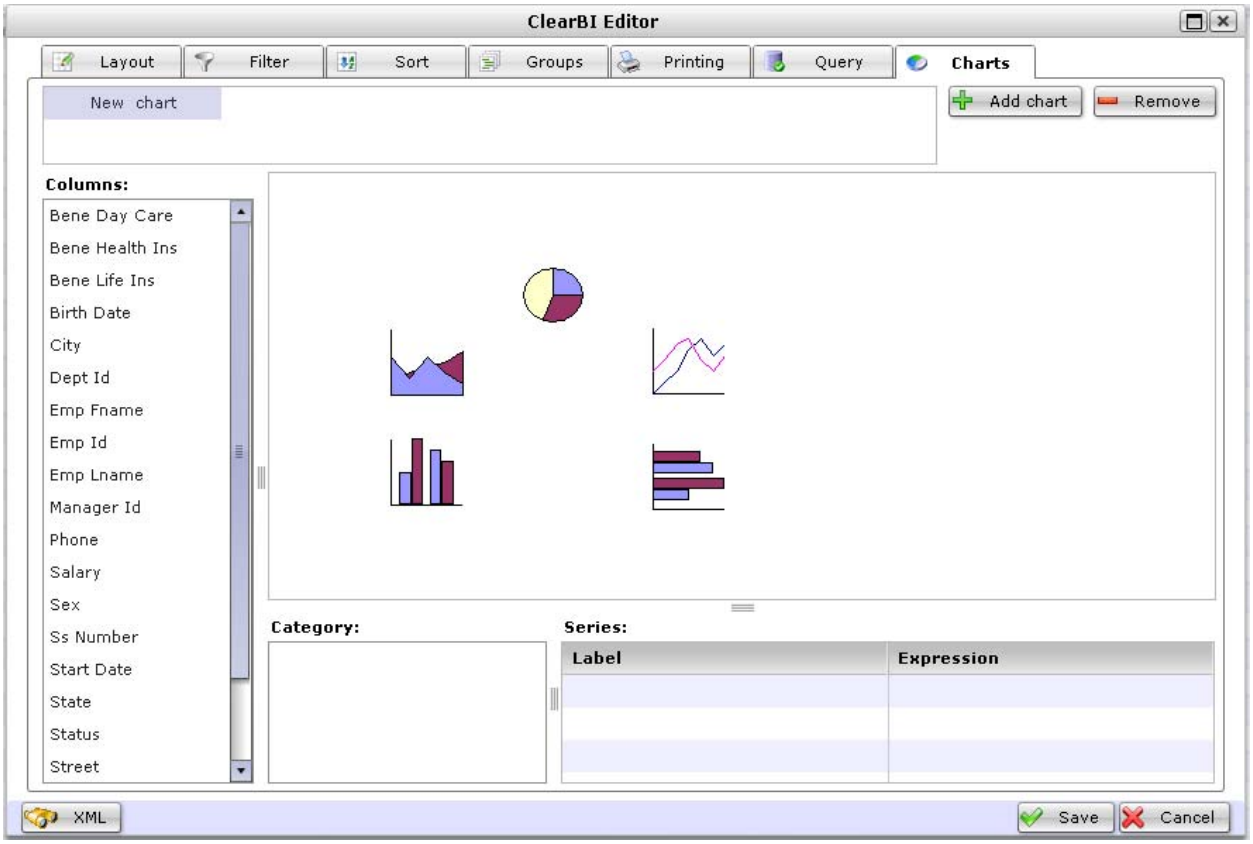
ClearBI comes with several reference implementations of sample reports located in the *control* folder of your example ClearBI project. One such implementation includes a report with charts.

Note. ClearBI requires Adobe charting package (a part of Flex Builder professional). If you only have a standard edition, ClearBI charting is not available.

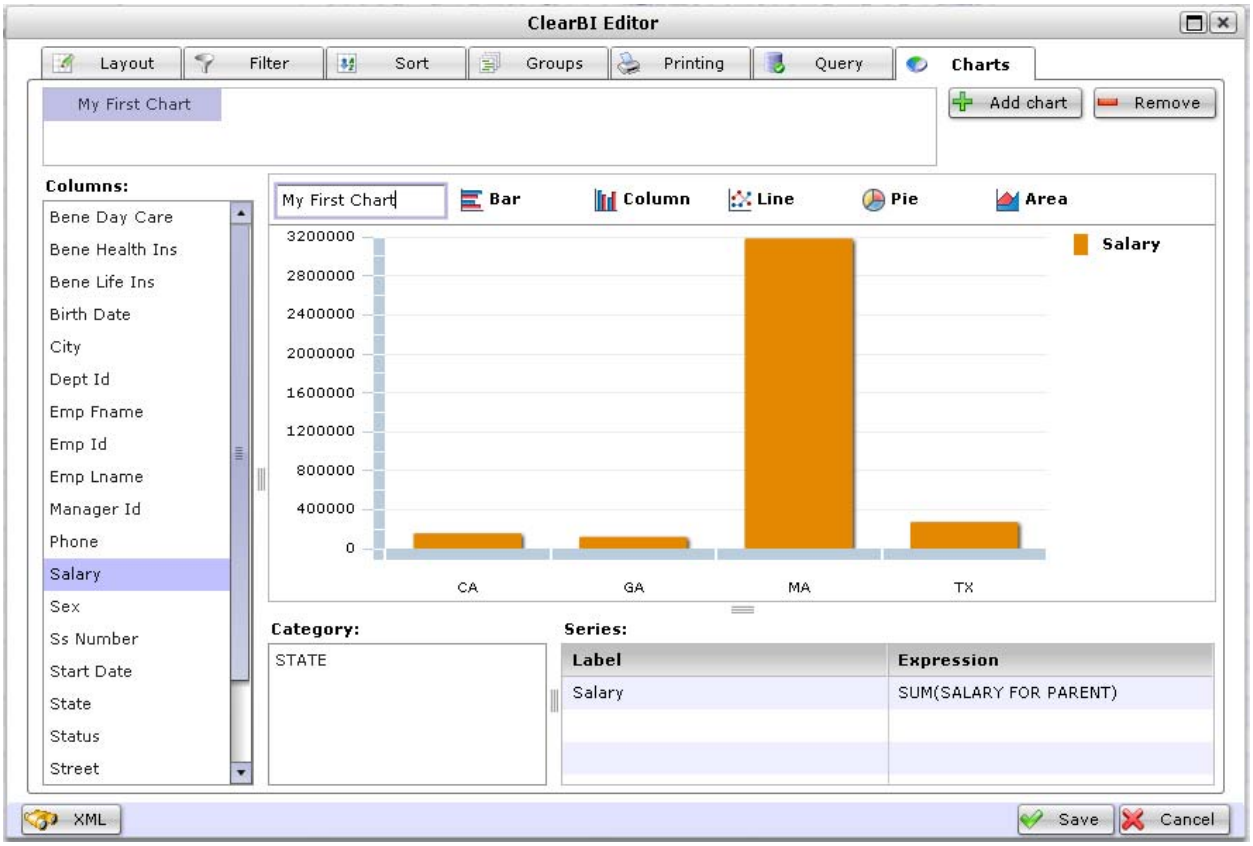
To run this example perform the following operations:

1. Upgrading your Flex project to ClearBI Server Edition creates there an application called SimpleReportingWithCharts.mxml. Run it and you'll see the Web browser with ClearBI toolbar. Select the required report template in the Category dropdown, and then select (or create new) report in the dropdown reports.

When the data is loaded, press the button Edit on the ClearBI toolbar. Open the tab Charts and click on the button **Add chart**. The editor window may look similar to this one:

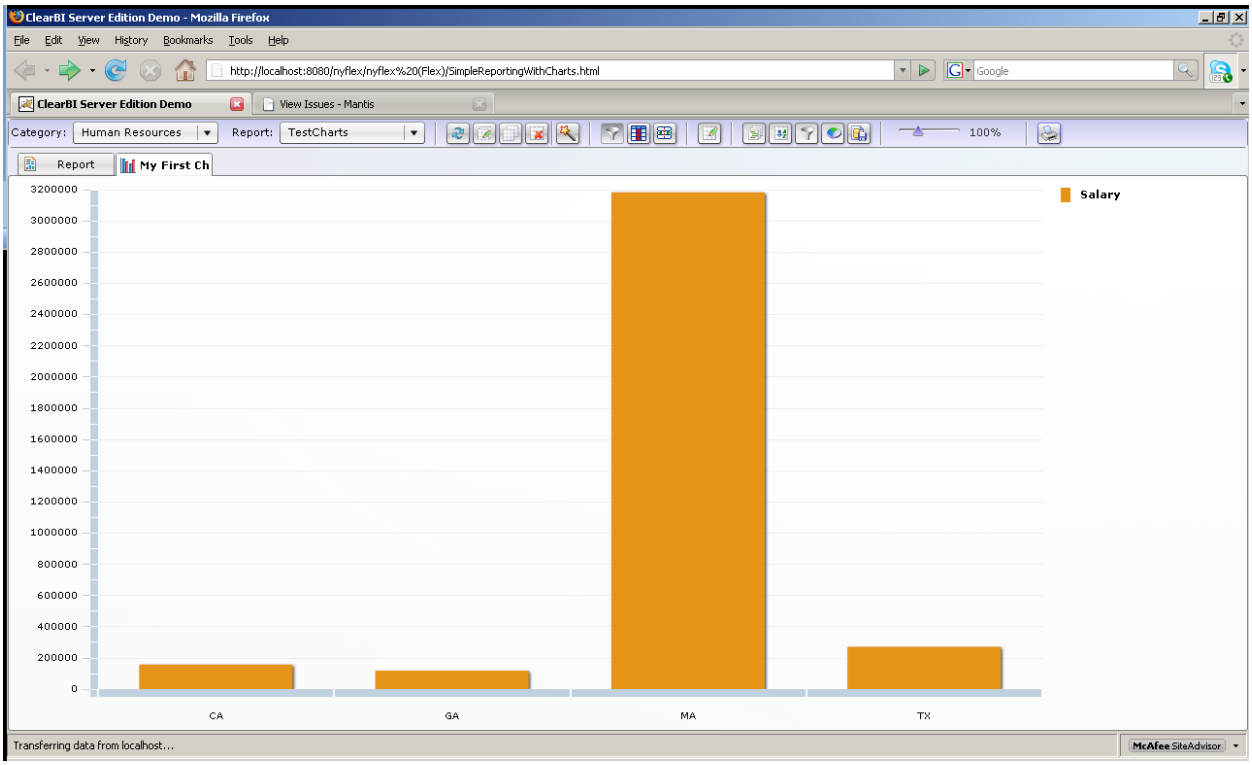


Select the type of the chart, i.e. Column, drag and drop the column State from the left into the Category box and then, drag and drop the column Salary onto the Series grid. Replace the name New Chart of the chart with another title, for example My First Chart:



A simple To change the label the expression just click on the word Salary, and to edit the formula of the expression, double click on it. You can also apply formulas to the Category name – just roll the mouse over the word STATE and you'll see an icon of calculator there – click on it to apply formulas to the category title, if needed.

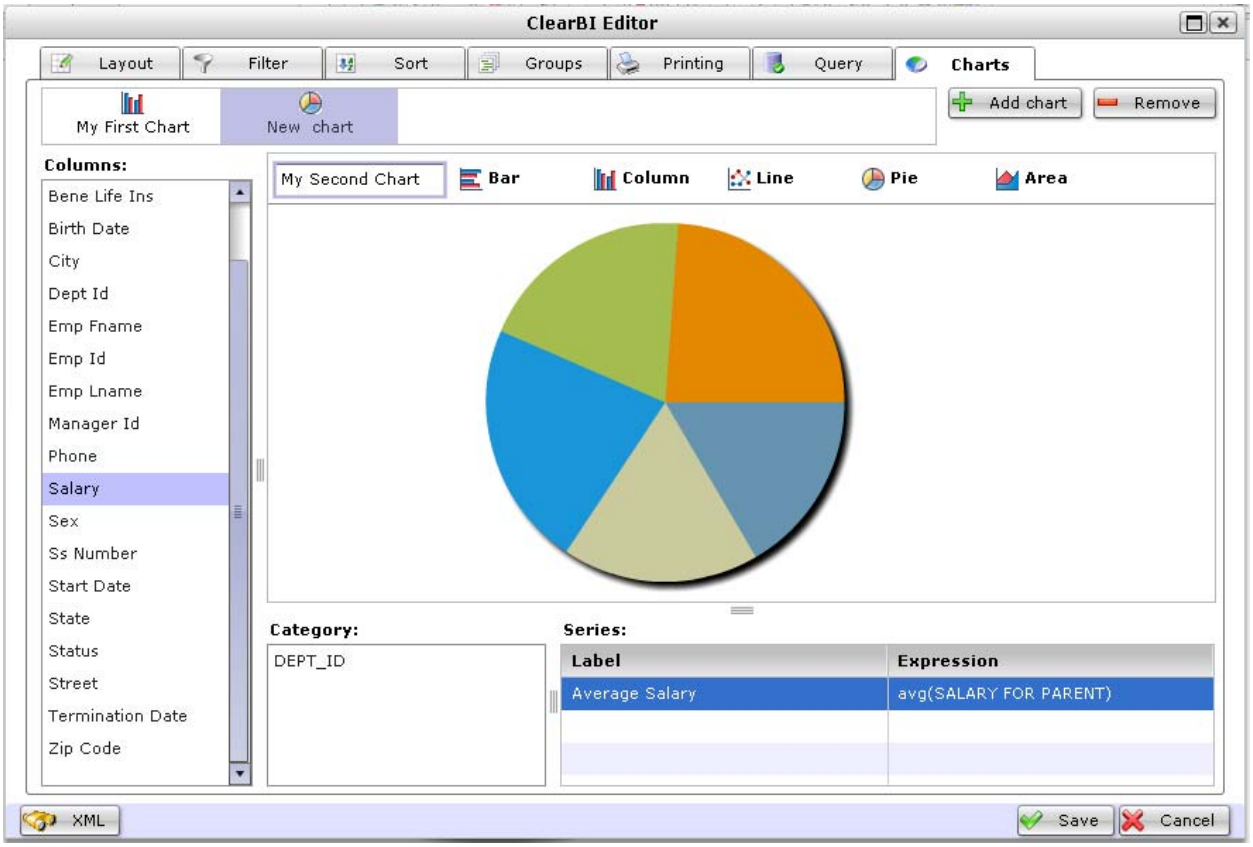
Press the button Save, the report's data will be reloaded and you'll see two tabs under the ClearBI toolbar – one with the tabular data, and the other one with your first chart:



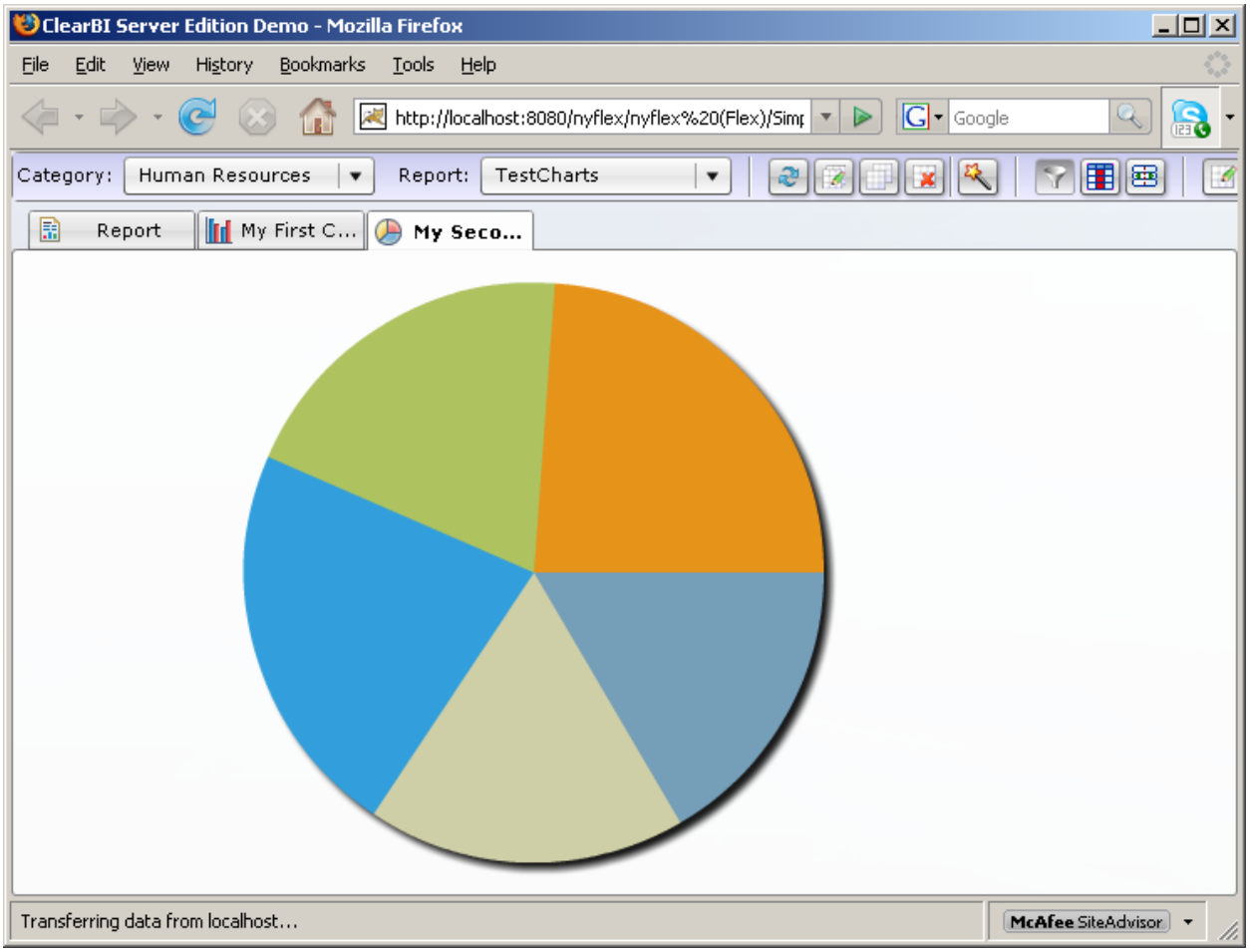
Let's add another chart to this report. Return to ClearBI Editor, open the Charts tab and Add another chart. This time we'll create a Pie chart with DEPY_ID as category and average salary as a series. You'll need to apply formula to change the default SUM(SALARY FOR PARENT) to avg(SALARY FOR PARENT), change the label to Average Salary and using Category's formula editor.

Note. Do not forget to press the button Validate while editing formulas to avoid unexpected results.

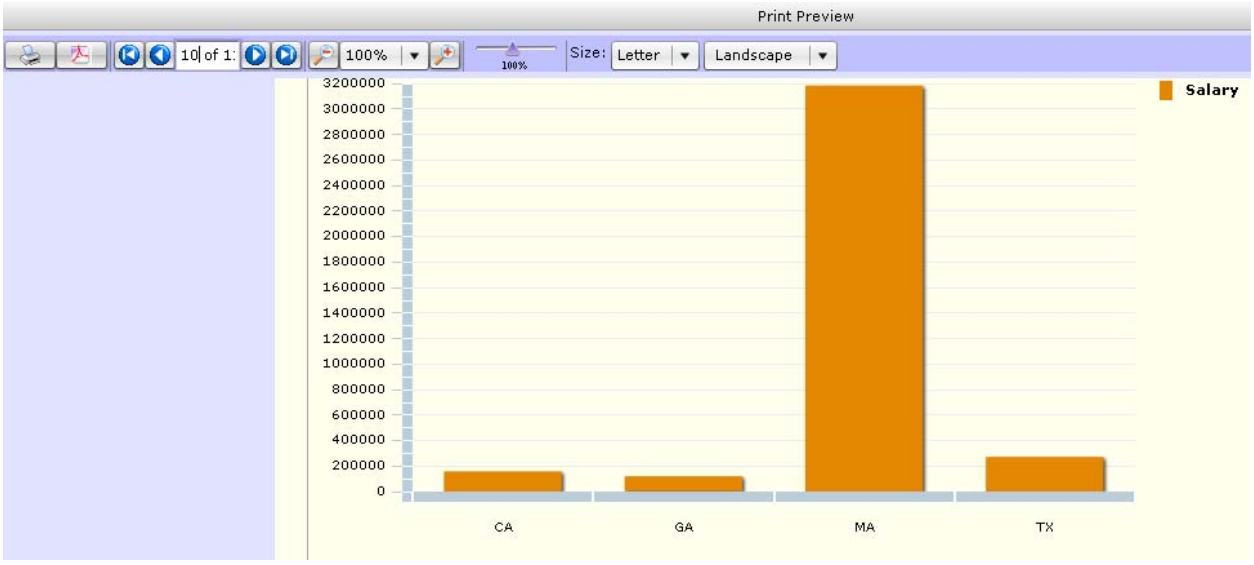
Change the title of this chart to be My Second Chart, save the chart and your report will be reloaded:



Now it will have a tab with tabular data and two charting tabs:

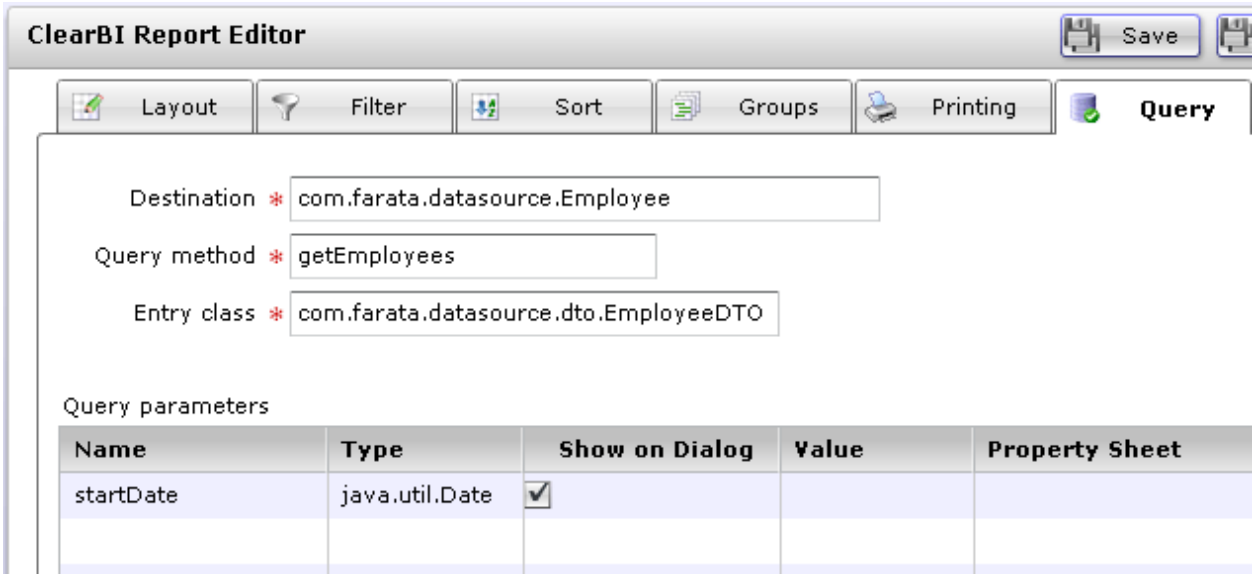


Created charts can be added to the report and printed. The reference implementation adds the charts at the end of the report – open the Pint Preview screen and scroll to the last pages of the report to see your charts there.



5.8 Query

This tab displays the server side source of data that generated report. It can contain names of the Java classes or describe JavaScript data providers. In this tab you can also specify report parameters, if any. You can change the displayed name of the data field, the parameter prompt message, check its type and specify if this parameter should be visible while running the report. You can also specify hard-coded values of parameters.

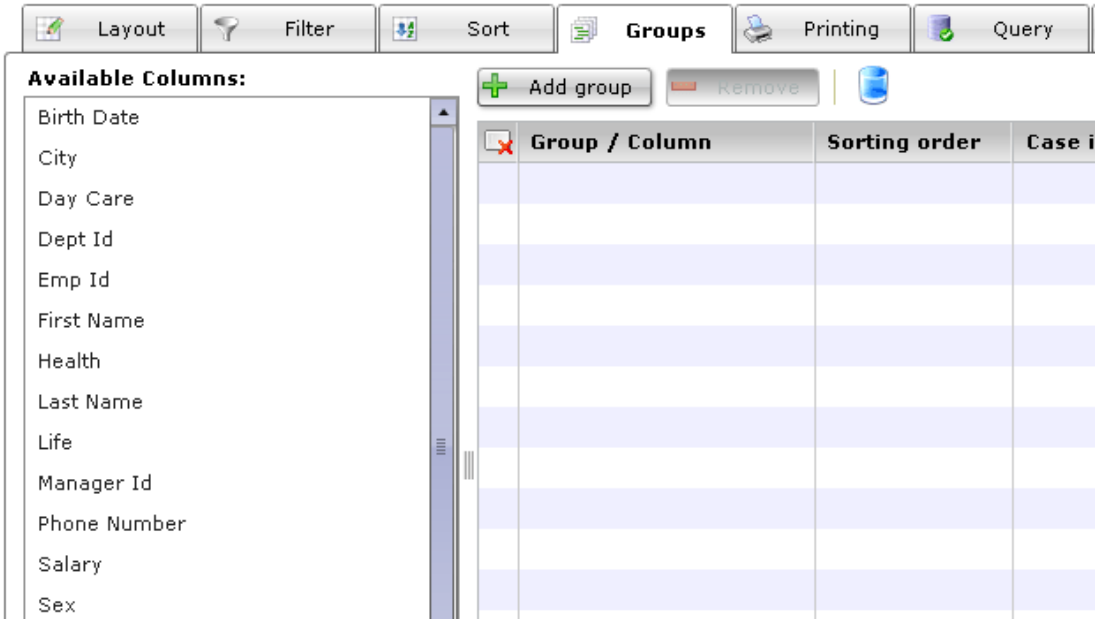


6.0 CUSTOMIZING THE EXAMPLE REPORT

Now we'll continue working with the application created in section 4.1 to demo such features of ClearBI as grouping, using formulas, creation of computed columns and export to Microsoft Excel. Let's continue working with example project (employees) described in CDB User Guide by performing the following actions:

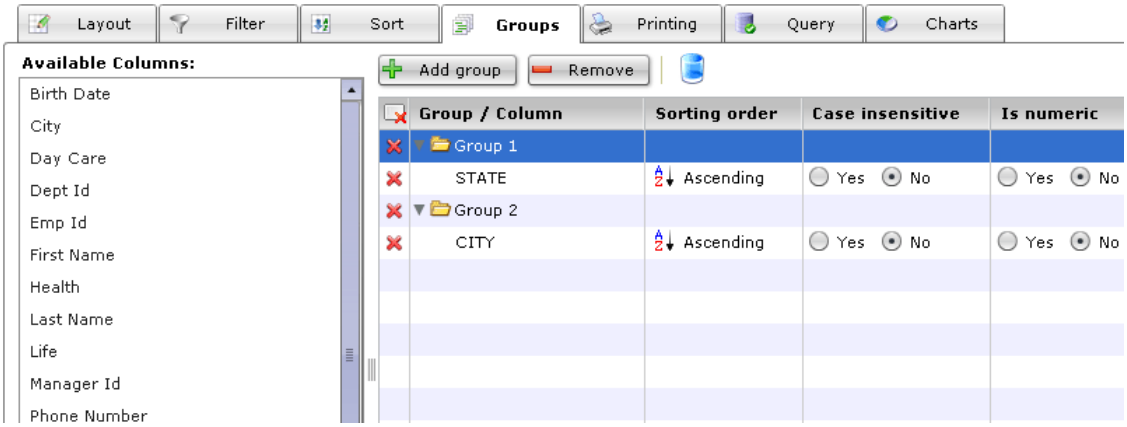
6.1.1 Grouping the data

We'll illustrate grouping capabilities of CLearBI using the generated report called Employee_getEmployees_ReportTest.mxml. Right-click on this file and select the menu option **ClearBI > Edit** and select the tab Groups.

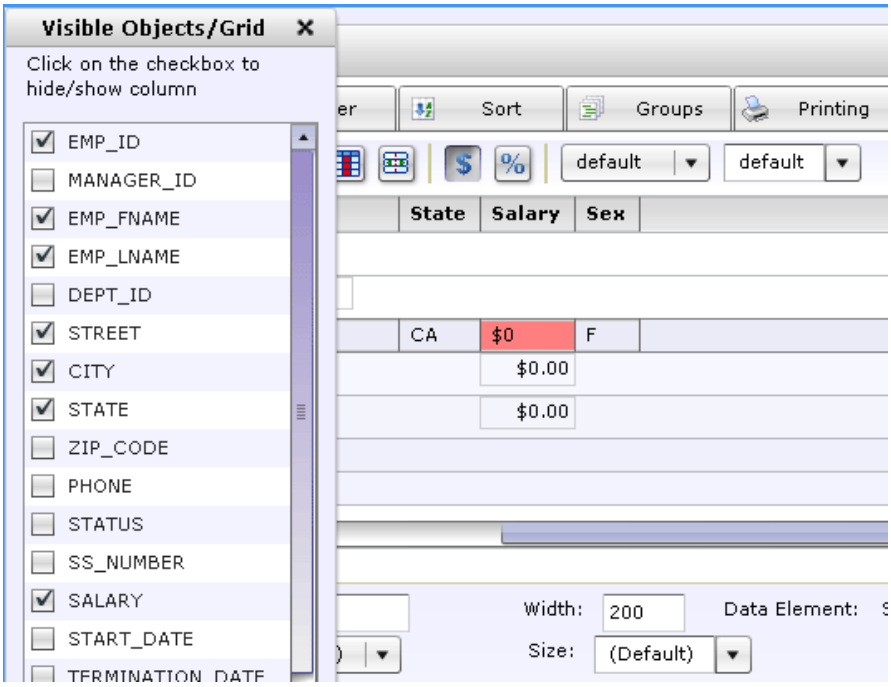


On the left you can see our *data universe*, which in this case based on the database columns contained in table **employee**. Create two groups by dragging the columns **STATE** and then **CITY** to the right. This way we are grouping the data from our report first by state, and within the sate by cities.

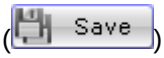
Please note that you can specify the sorting order for each column during group creation, case sensitivity and if the column is numeric. We'll use default settings for our example.



Return to the tab **Layout** and select the columns that you want to include in the report. To make the column visible or invisible, use the toggle button **Hide/Show Columns** that is represented by the following icon  for opening **Visible Object/Grid** panel and then select visible columns.

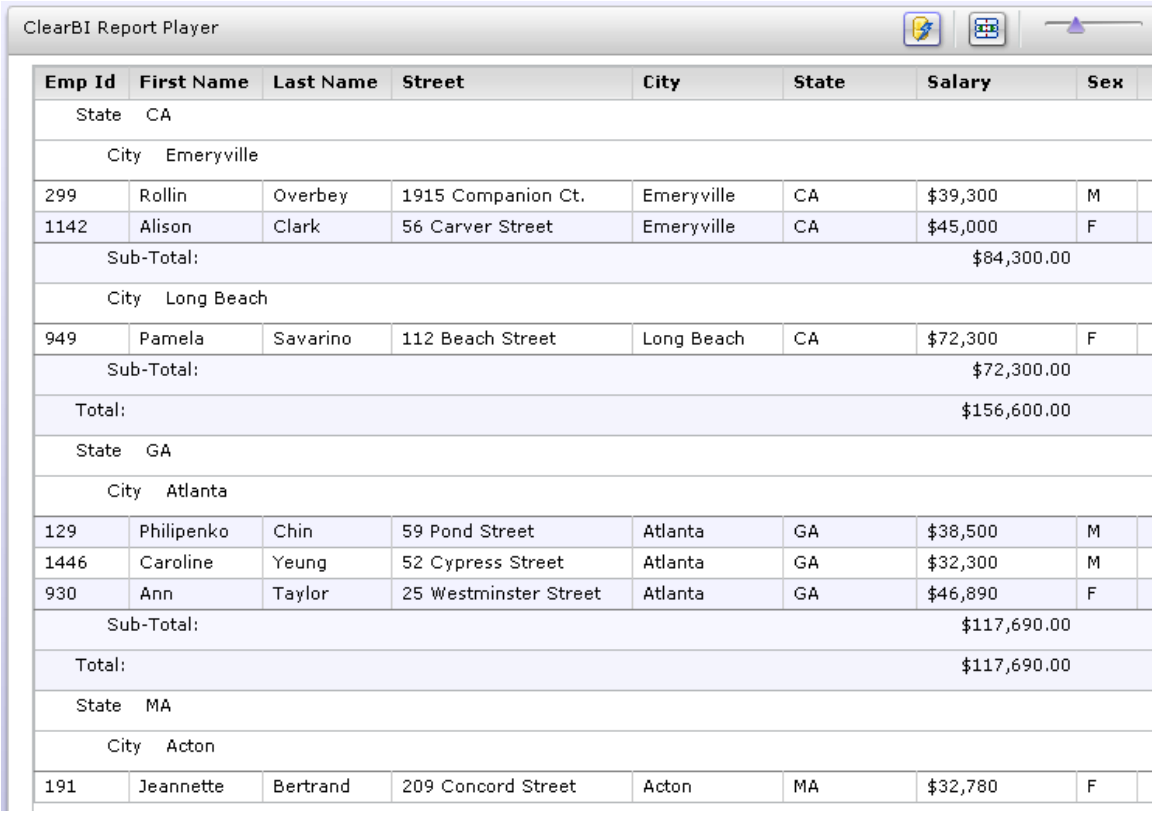


Keep the following columns visible **EMP_ID**, **EMP_FNAME**, **EMP_LNAME**, **STREET**, **CITY**, **STATE**, **SALARY**, **SEX** and press **Done** button.

Save the report with the button Save (

) and run ClearBI Player by right-clicking on the report Employee_getEmployees_ReportTest.mxml and selecting **ClearBI > Preview**.

The button **AutoSize Columns** () is used for automatic sizing of the report columns.



Emp Id	First Name	Last Name	Street	City	State	Salary	Sex
State CA							
City Emeryville							
299	Rollin	Overbey	1915 Companion Ct.	Emeryville	CA	$39,300	M
1142	Alison	Clark	56 Carver Street	Emeryville	CA	$45,000	F
Sub-Total:						$84,300.00	
City Long Beach							
949	Pamela	Savarino	112 Beach Street	Long Beach	CA	$72,300	F
Sub-Total:						$72,300.00	
Total:						$156,600.00	
State GA							
City Atlanta							
129	Philipenko	Chin	59 Pond Street	Atlanta	GA	$38,500	M
1446	Caroline	Yeung	52 Cypress Street	Atlanta	GA	$32,300	M
930	Ann	Taylor	25 Westminster Street	Atlanta	GA	$46,890	F
Sub-Total:						$117,690.00	
Total:						$117,690.00	
State MA							
City Acton							
191	Jeannette	Bertrand	209 Concord Street	Acton	MA	$32,780	F

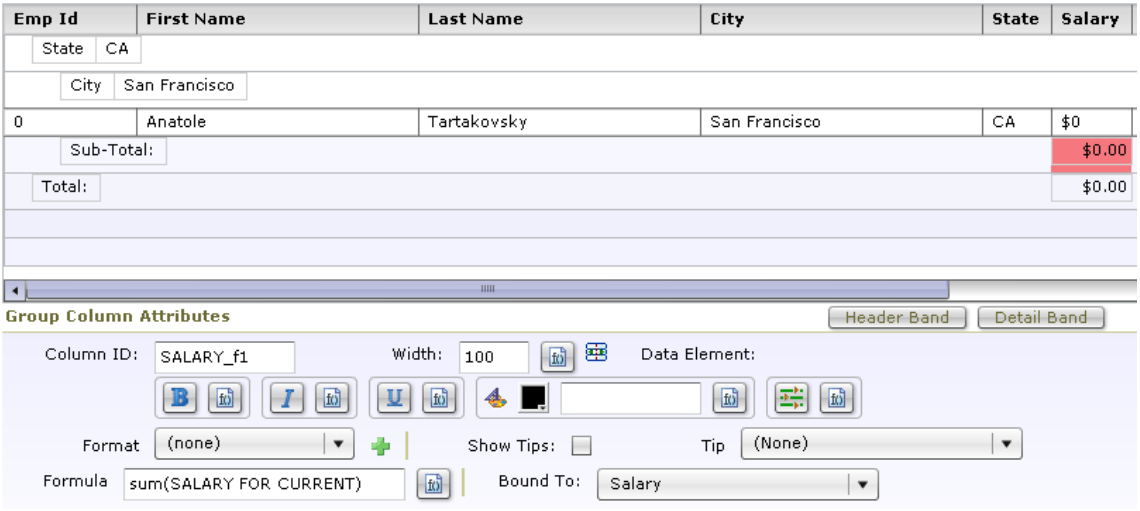
As you can see in the screenshot above, all the data are first groups by the state (CA, GA, et al.). Within the states, employees are grouped by cities (in CA, it's Emeryville and Long Beach).

The data in each group is sorted in the alphabetical order, which is default sorting order.

6.1.2 Creating Totals

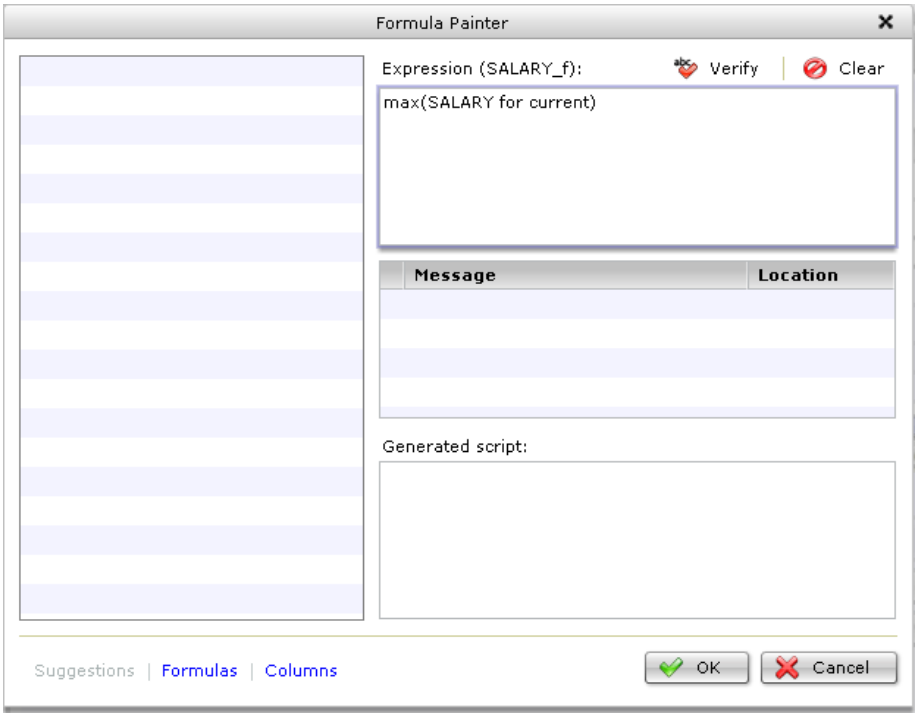
You may notice the calculated **Sub-Total** and **Total** values in the report above allowing you to see total salary for each state and city.

Let's return to ClearBI Designer and see how this total was created. Right-click on cell Sub-Total for the Salary column and not the panel shown below. Note the field **Formula** at the very bottom of the screen.

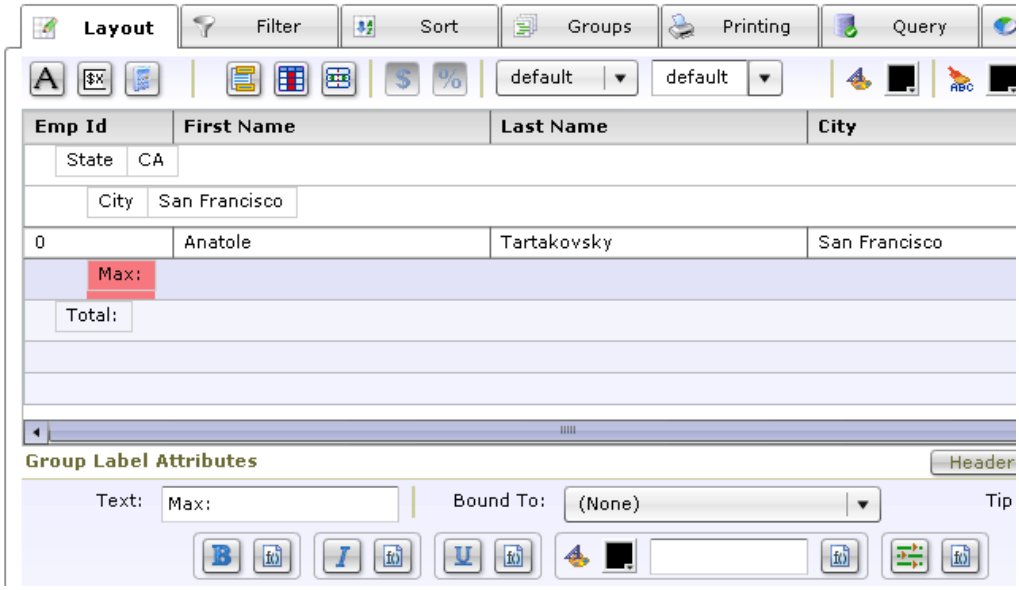


In our example, the text in the field formula shows that we are interested in seeing the sum of all values in the column salary for the group based on cities ("for current" in the formula means for the current group). Let's edit this field by pressing the button Formula  .

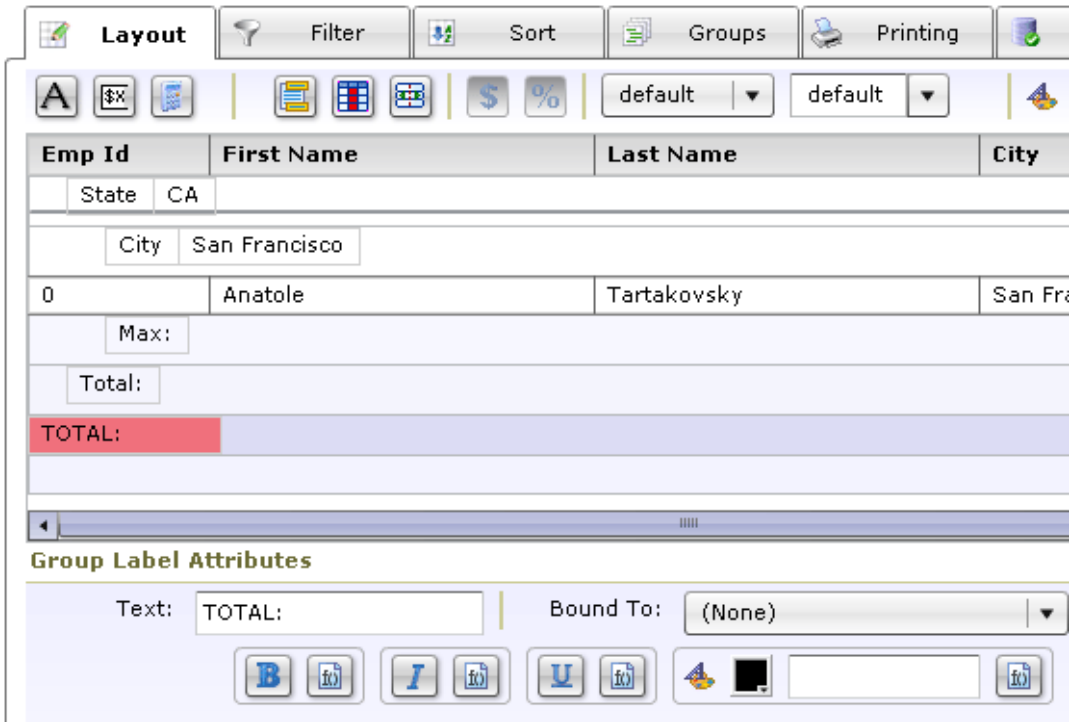
A popup window **Formula Painter** allows you to create and edit formulas. For example, replace the word **sum** with **max**. Now instead of calculating the sum of all salaries, we'll get the max salary for each city.



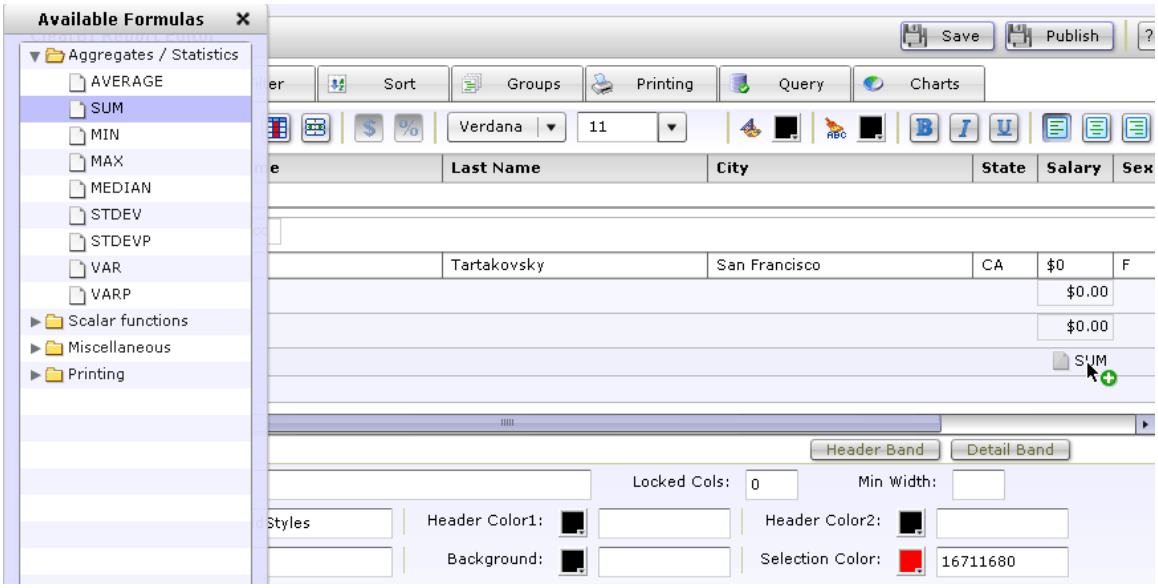
Click on Sub-Total again, and select the style for this field in the panel that appears at the bottom. Change the title of this field to **Max**.



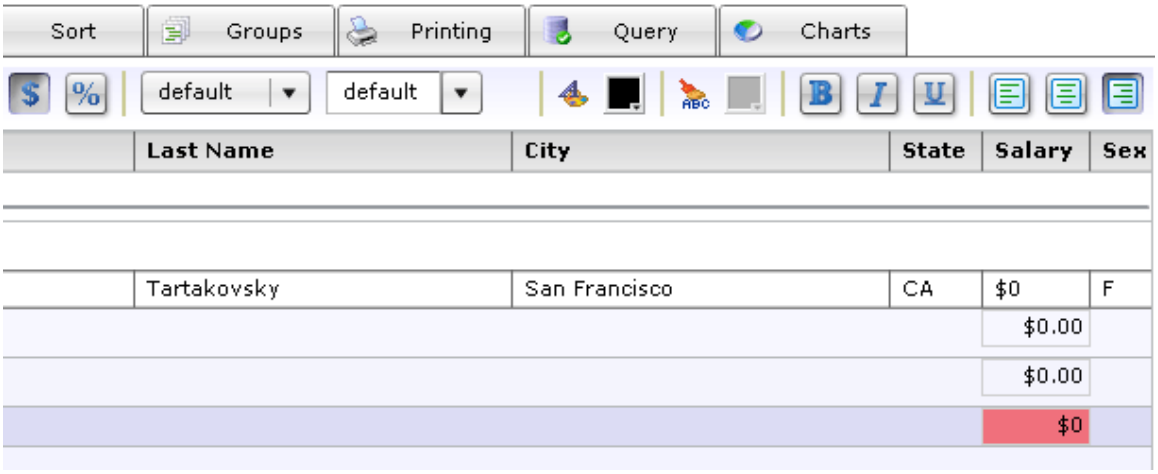
To finish editing, let's add some text by clicking the Text button  and move it to the left right under the word **Total** and change it to **TOTAL**.

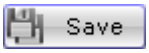


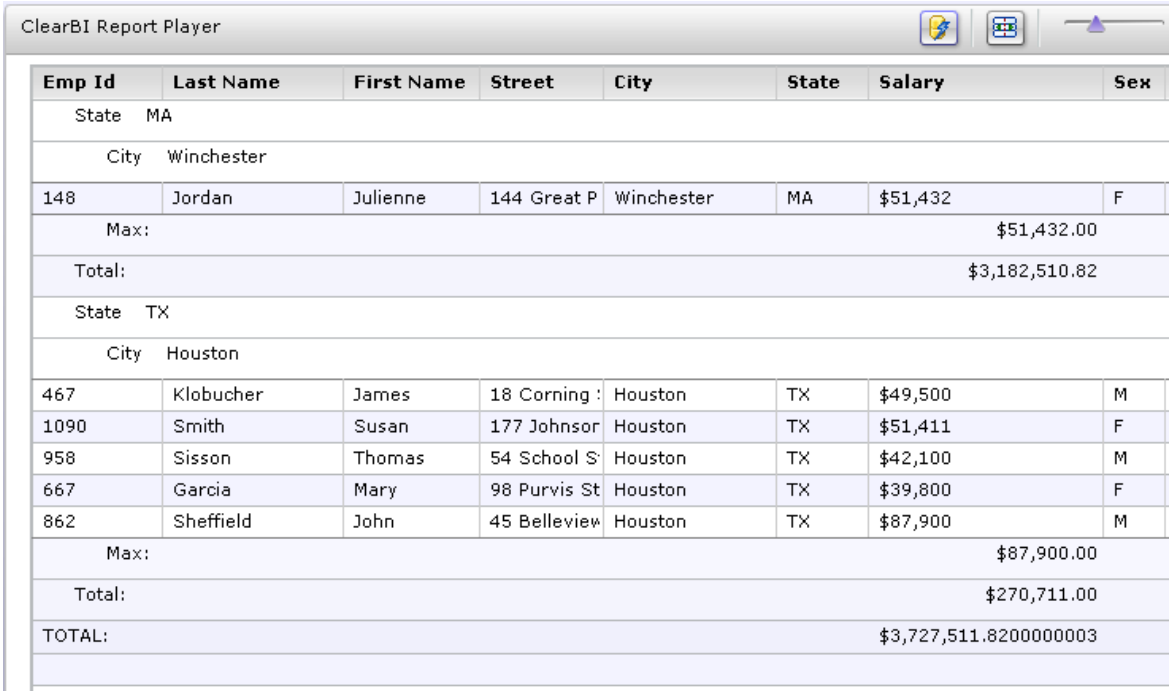
We've added the title for the sum of all salaries, and now let's add the formula. Click on the button Formula and select **Aggregates / Statistics > SUM** in the popup window. Move the SUM into the **TOTAL** row under the column **Salary** to display the total salary here.



To align the style of this cell with the style of the Salary column, select the right alignment and press the button **Currency style**.



Save the report with the button **Save**  and return to ClearBI Player. Scroll the report all the way down to see results of your work.



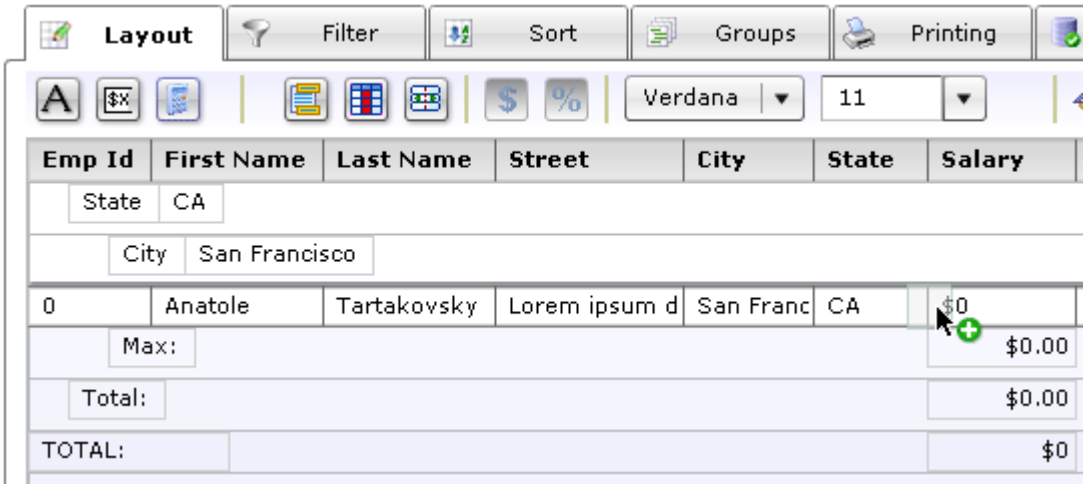
Emp Id	Last Name	First Name	Street	City	State	Salary	Sex
State MA							
City Winchester							
148	Jordan	Julienne	144 Great P	Winchester	MA	$51,432	F
Max:						$51,432.00	
Total:						$3,182,510.82	
State TX							
City Houston							
467	Klobucher	James	18 Corning :	Houston	TX	$49,500	M
1090	Smith	Susan	177 Johnsor	Houston	TX	$51,411	F
958	Sisson	Thomas	54 School S	Houston	TX	$42,100	M
667	Garcia	Mary	98 Purvis St	Houston	TX	$39,800	F
862	Sheffield	John	45 Belleview	Houston	TX	$87,900	M
Max:						$87,900.00	
Total:						$270,711.00	
TOTAL:						$3,727,511.8200000003	

Note that the title **Sub-Total** has been replaced with **Max**, and you do see the maximum salary and not the total. The very last line of the report contains the total salary of all employees.

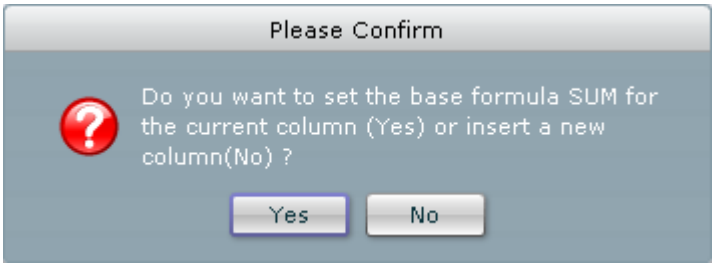
Note. While defining grouping in the ClearBI editor, you may use the field called Base Formula, which will automatically apply the specified formula to every group and display the result at the bottom of each group.

6.1.3 Creating Formulas

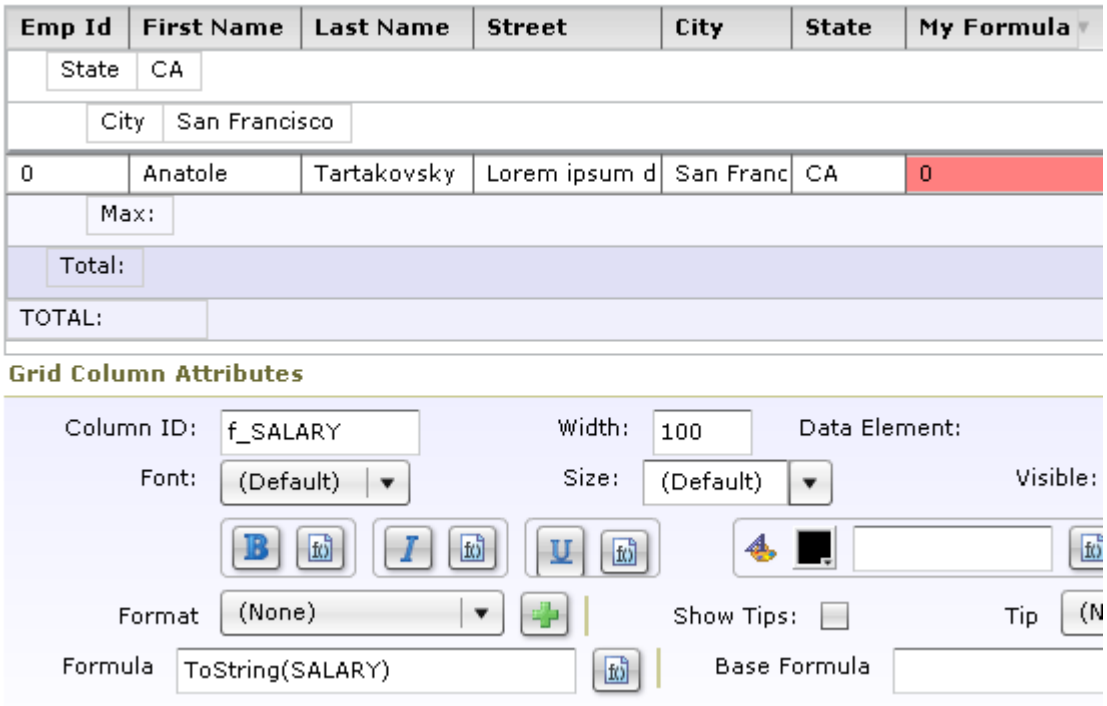
As you know by now, ClearBI Editor allows you to apply formulas to various styles of data and to the entire columns. To start creating formulas, just drag one of the Formula button  to the required position in the report, for example to the left of the column **Salary**.



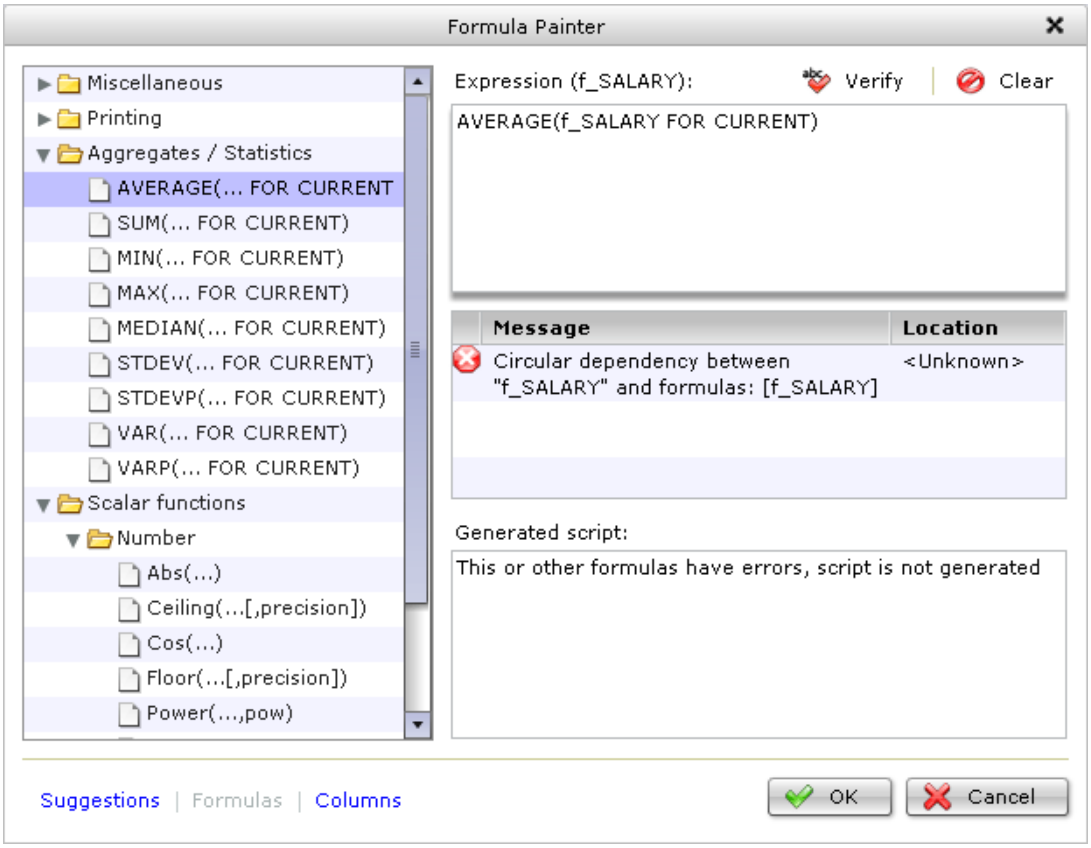
Note. If you see the confirmation dialog shown below, just specify if you'd like to change the formula of the current column (**Yes**), or create a new column with a formula (**No**). We'll select **No** for our example.



The newly created column will be called **ToString** and contain the formula **ToString(SALARY)**. We'll change the title of this column to **My Formula**.



To edit this formula, press the Formula button and you'll see the formula painter window shown below. Highlight and remove the text in the field **Expression**, because we'll be creating the next formula from scratch.

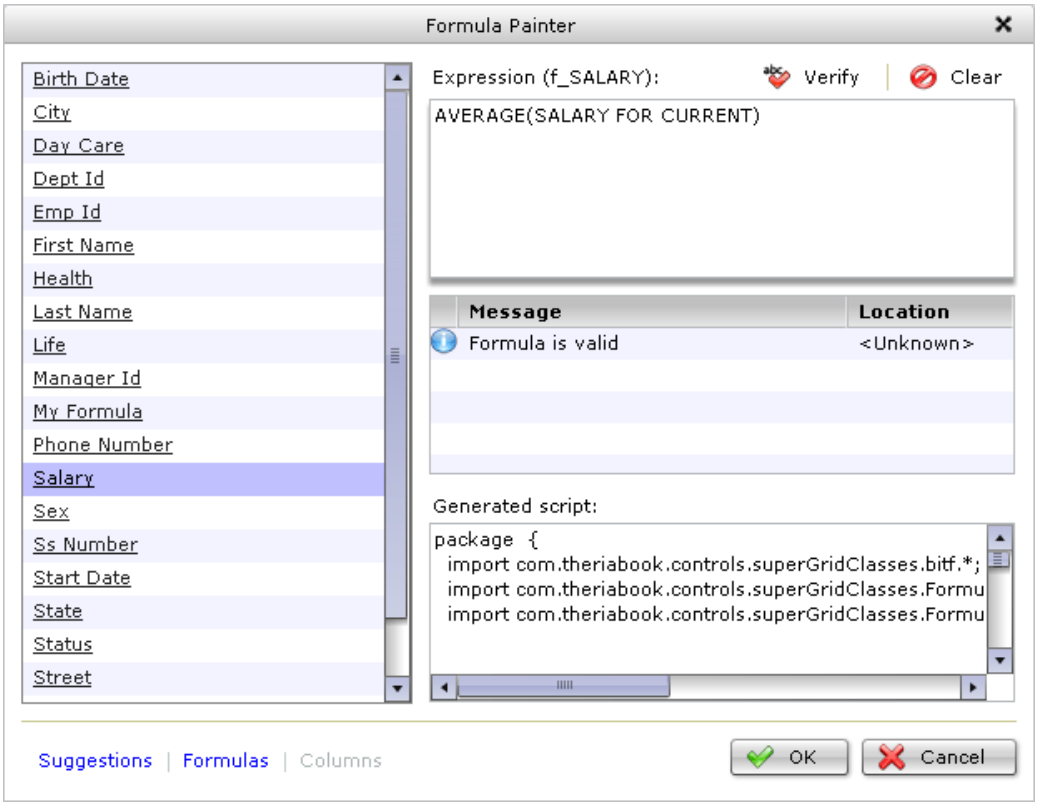


Note three tabs in the left bottom corner:

- **Suggestions** – this tab is empty now
- **Formulas** – this tab contains all available formulas
- **Columns** – this tab lists all column names

Select the tab Formulas to see the list of categorized formulas. Open the category **Aggregates / Statistics** and double-click on **AVERAGE(… FOR CURRENT)** to add this formula to the Expression field.

The button **Verify** allows to validate correctness of formulas. If you press it now, you'll see an error message because you are trying to populate the current field f_SALARY by averaging the value of this field itself. Replace f_SALARY with the field salary by doing the following: erase f_SALARY leaving the cursor inside the parentheses, open the tab Columns and double-click on the column name Salary. The name of selected column will be inserted where your cursor was. Click on the button **Verify** again to make sure that the formula is correct now.

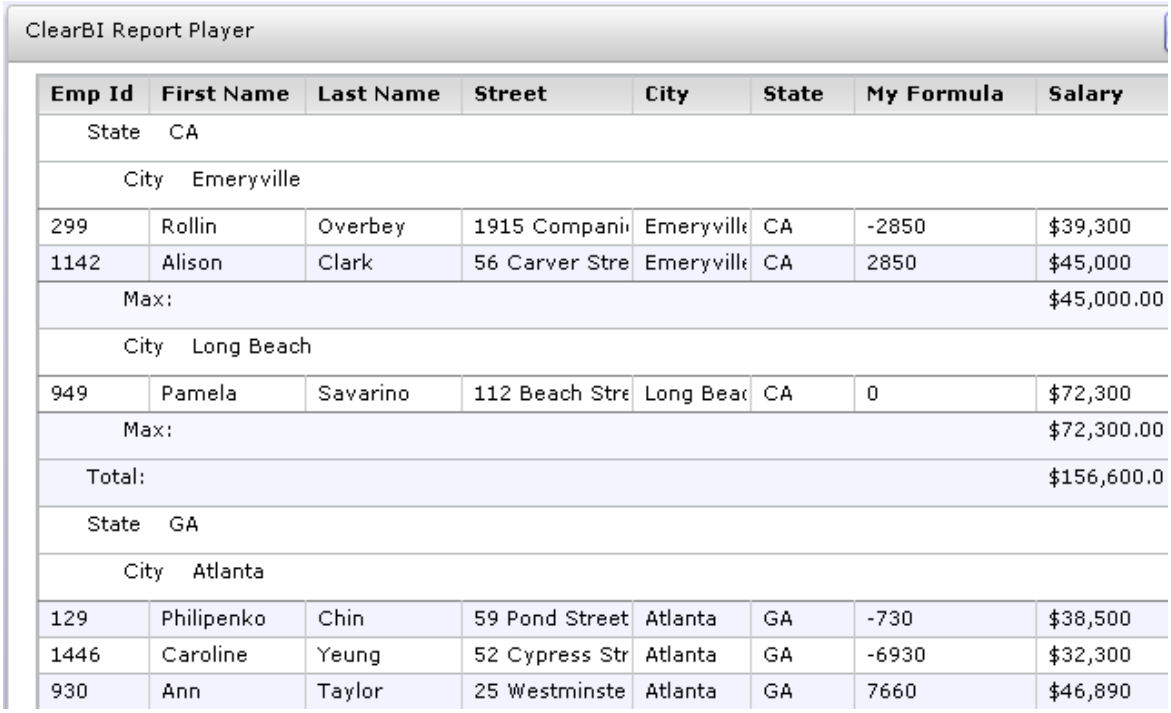


Let's modify the formula a little bit to display the salary deviation from the average:

SALARY - AVERAGE(SALARY FOR PARENT).

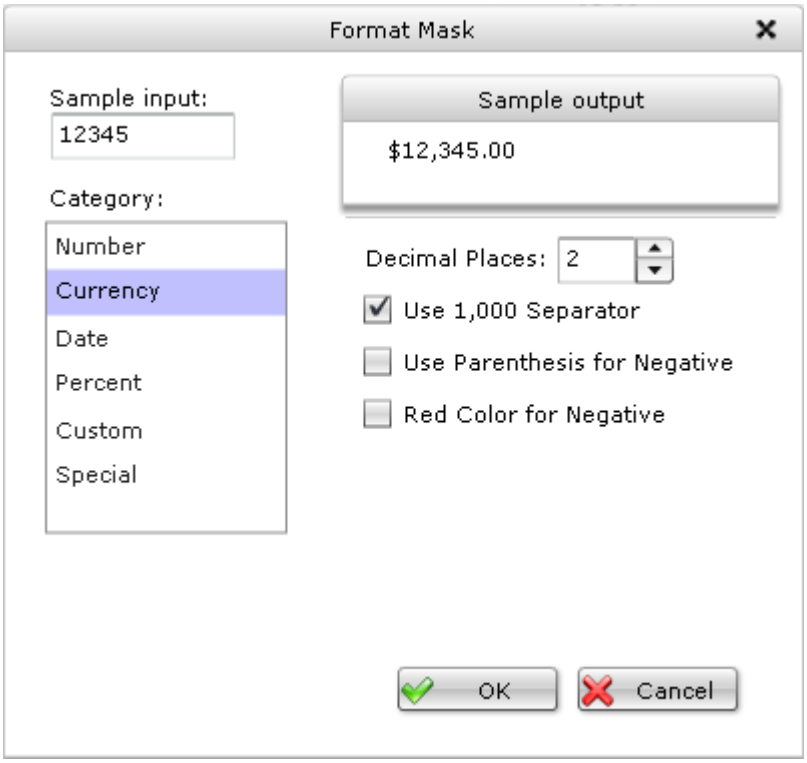
We want to display the deviation by city (note that we've replaced CURRENT with PARENT otherwise the average would be calculated for a single value, and the result of this formula would be always zero).

Press the button **OK**, save the report and open the ClearBI Player.



The column **My Formula** shows the deviation from the average salary by city.

Let's customize the style of this column in the ClearBI Editor. Click in the cell in the column My Formula and press the button **Add new format**  . In the popup window **Format Mask** select the category **Currency** and the option **Use 1,000 Separator.** Enter 2 for the number of **Decimal Places**.

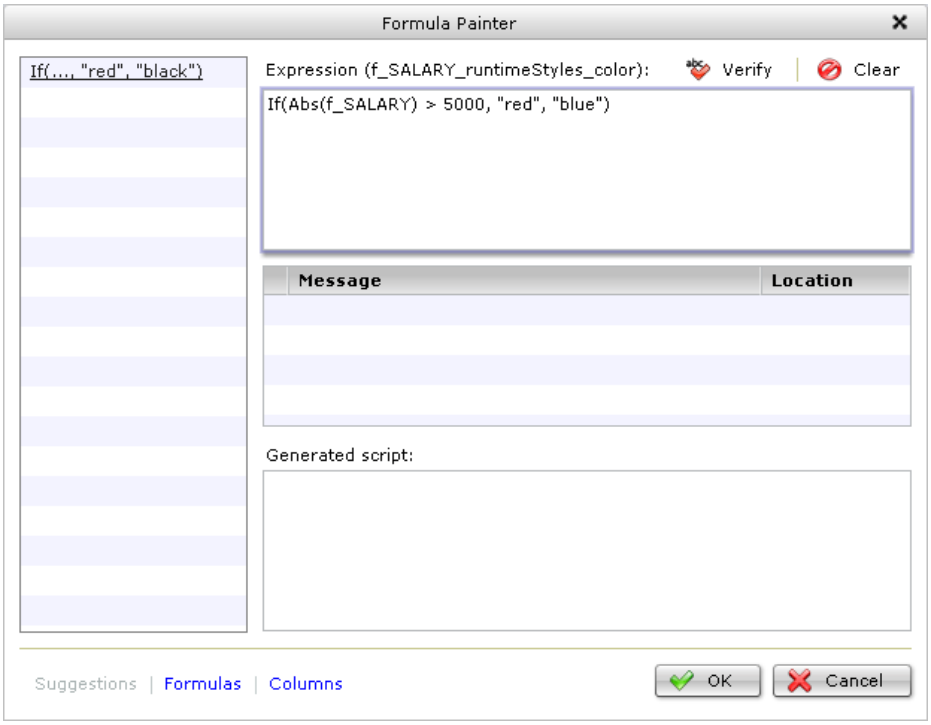


You can use formulas in styles too. Press the button Formula  .

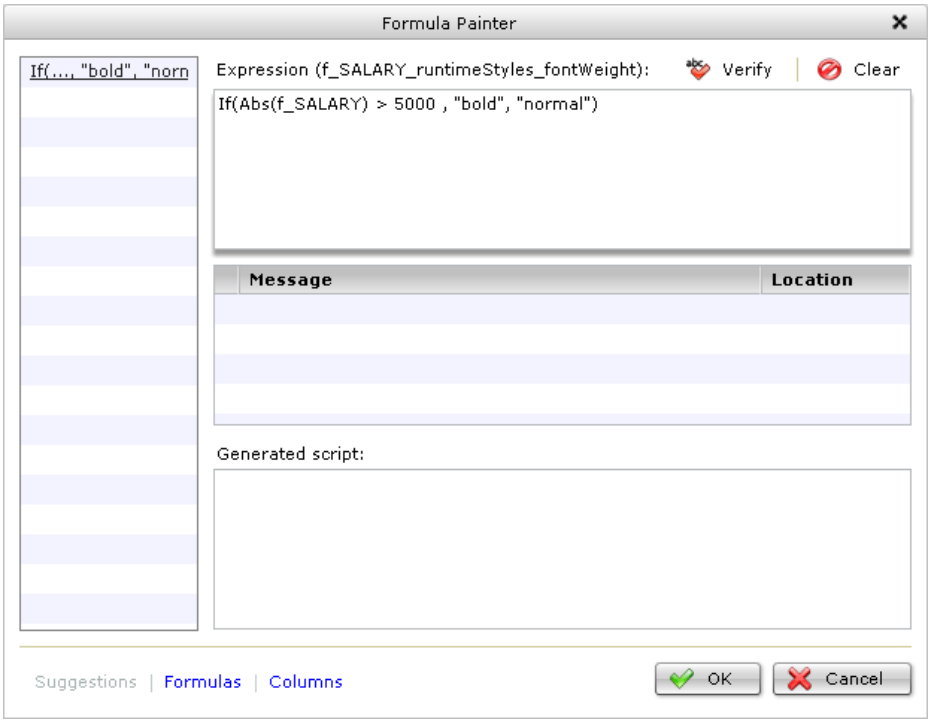
The Formula Painter window will open, and you'll see a new record in the tab Suggestions. Double-click on this record and using the formula **Abs** change the content of the field **Expression** to read

If(Abs(f_SALARY) > 5000, "red", "blue").

Now, if the deviation value in the column My Formula is greater than 5000, the value will be shown in red, otherwise in blue.

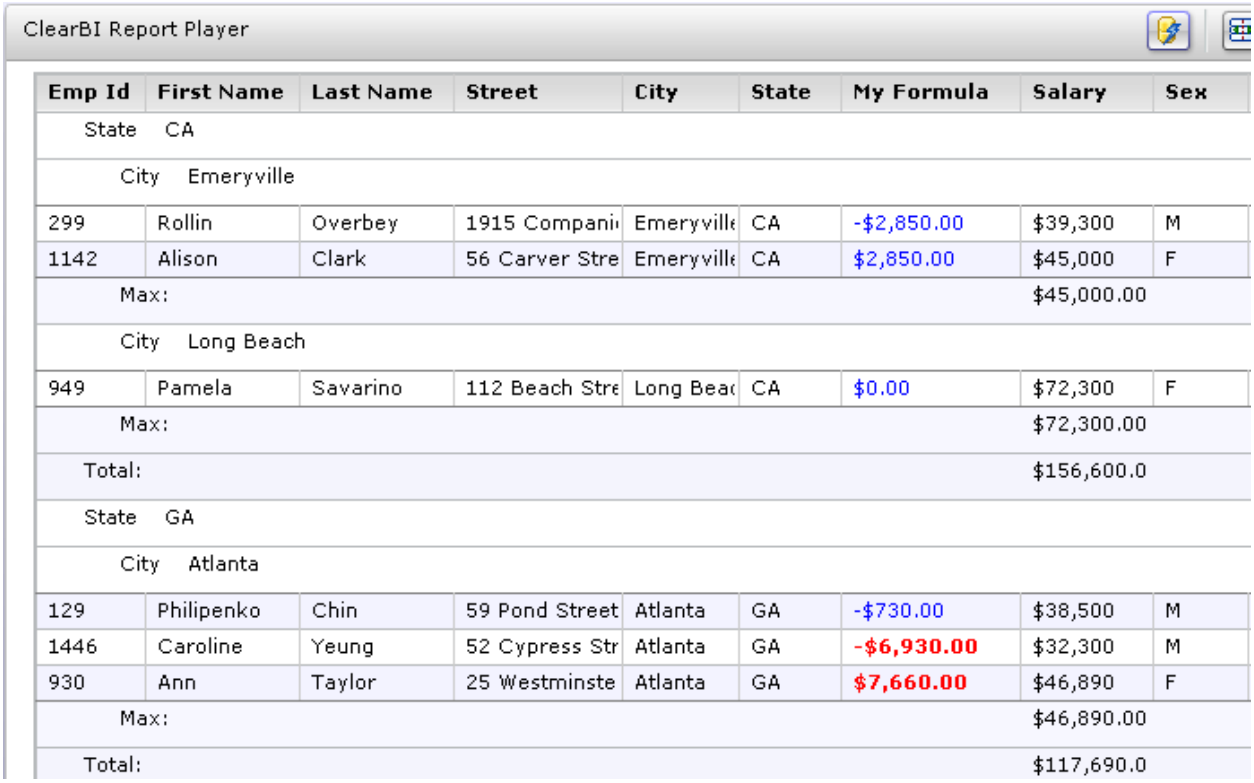


Press **OK**, and then  by the button **Bold**. You'll find a new record in the tab **Suggestions**. Double-click on it and make the formula results conditionally bold similarly to what we did above with colors:



If the value in My Formula is greater than 5000 ClearBI will show it in bold, otherwise it'll be displayed in a normal font style.

You can also underline or italicize text using the buttons located by Italic and Underline icons. For now, just press **OK,** save the report and run ClearBI Player.


ClearBI Report Player

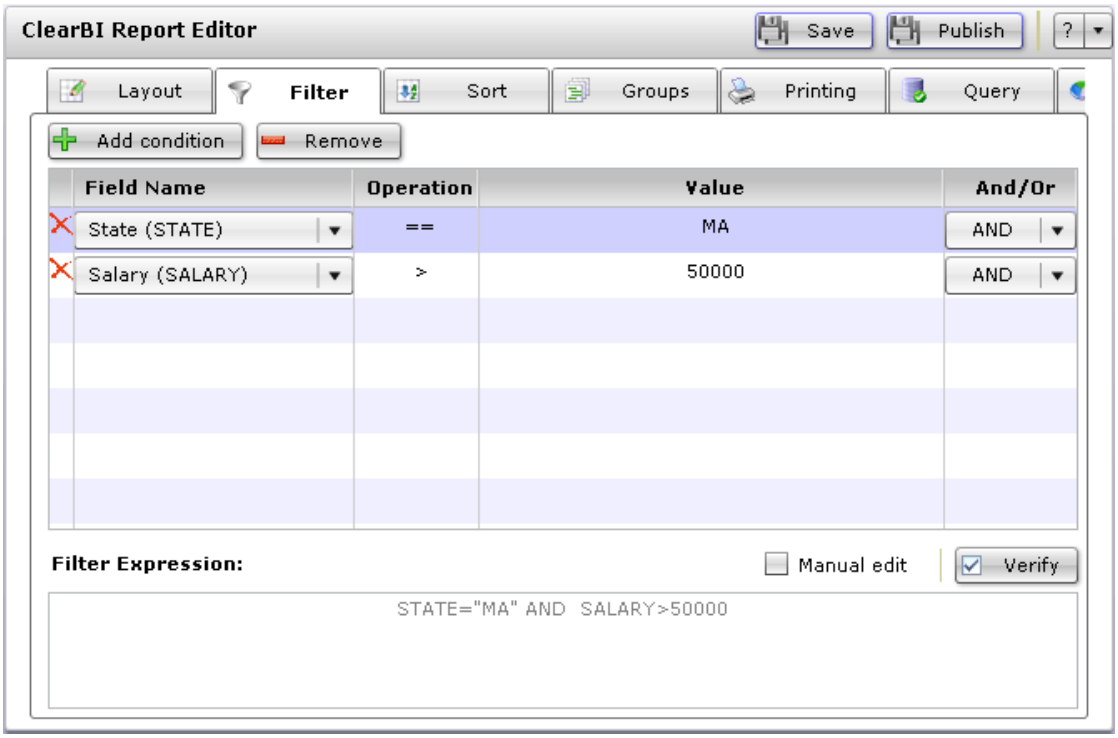
Emp Id	First Name	Last Name	Street	City	State	My Formula	Salary	Sex
State CA								
City Emeryville								
299	Rollin	Overbey	1915 Compani	Emeryville	CA	-$2,850.00	$39,300	M
1142	Alison	Clark	56 Carver Stre	Emeryville	CA	$2,850.00	$45,000	F
Max:							$45,000.00	
City Long Beach								
949	Pamela	Savarino	112 Beach Stre	Long Bea	CA	$0.00	$72,300	F
Max:							$72,300.00	
Total:							$156,600.0	
State GA								
City Atlanta								
129	Philipenko	Chin	59 Pond Street	Atlanta	GA	-$730.00	$38,500	M
1446	Caroline	Yeung	52 Cypress Str	Atlanta	GA	**-$6,930.00**	$32,300	M
930	Ann	Taylor	25 Westminste	Atlanta	GA	**$7,660.00**	$46,890	F
Max:							$46,890.00	
Total:							$117,690.0	

As you can see, the style of the data in the column **My Formula** has been changed.

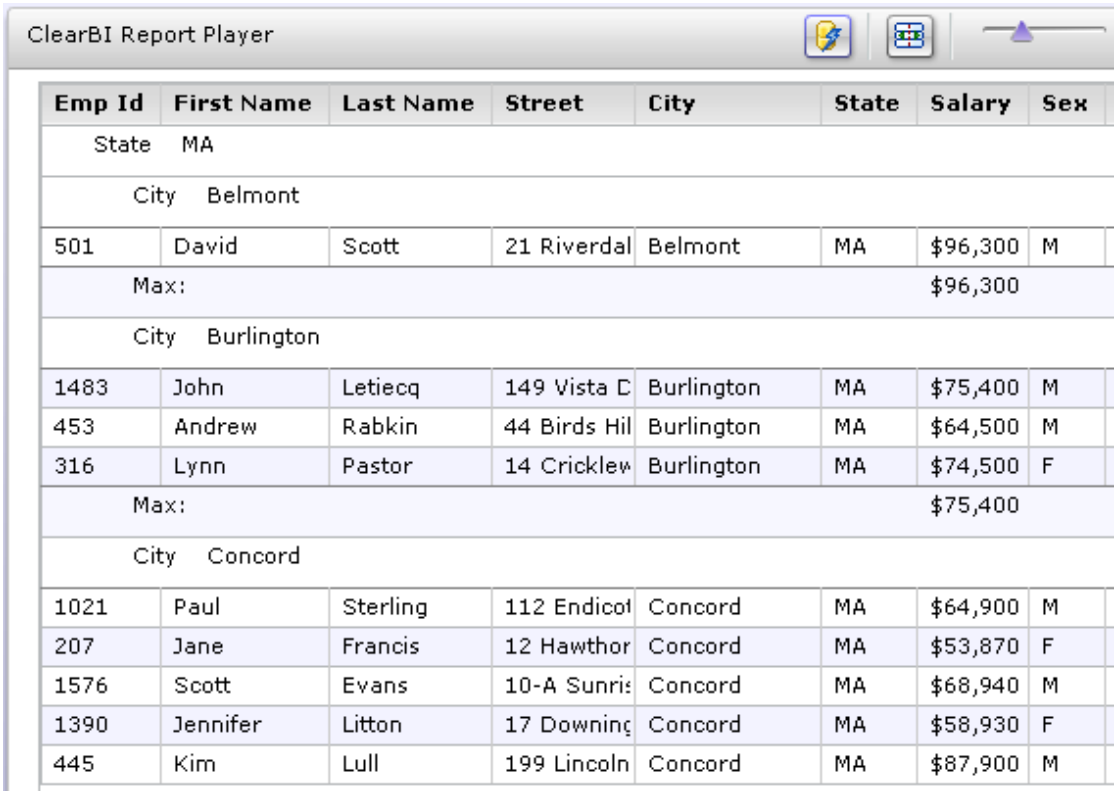
6.1.4 Filtering Reports

To filter report data, open the **Filter** tab in ClearBI Editor.

Using the button **Add condition** add two conditions: **State == MA** and **Salary > 50000**. The filter expression will be displayed at the bottom of the tab folder: **STATE="MA" AND SALARY>50000**.



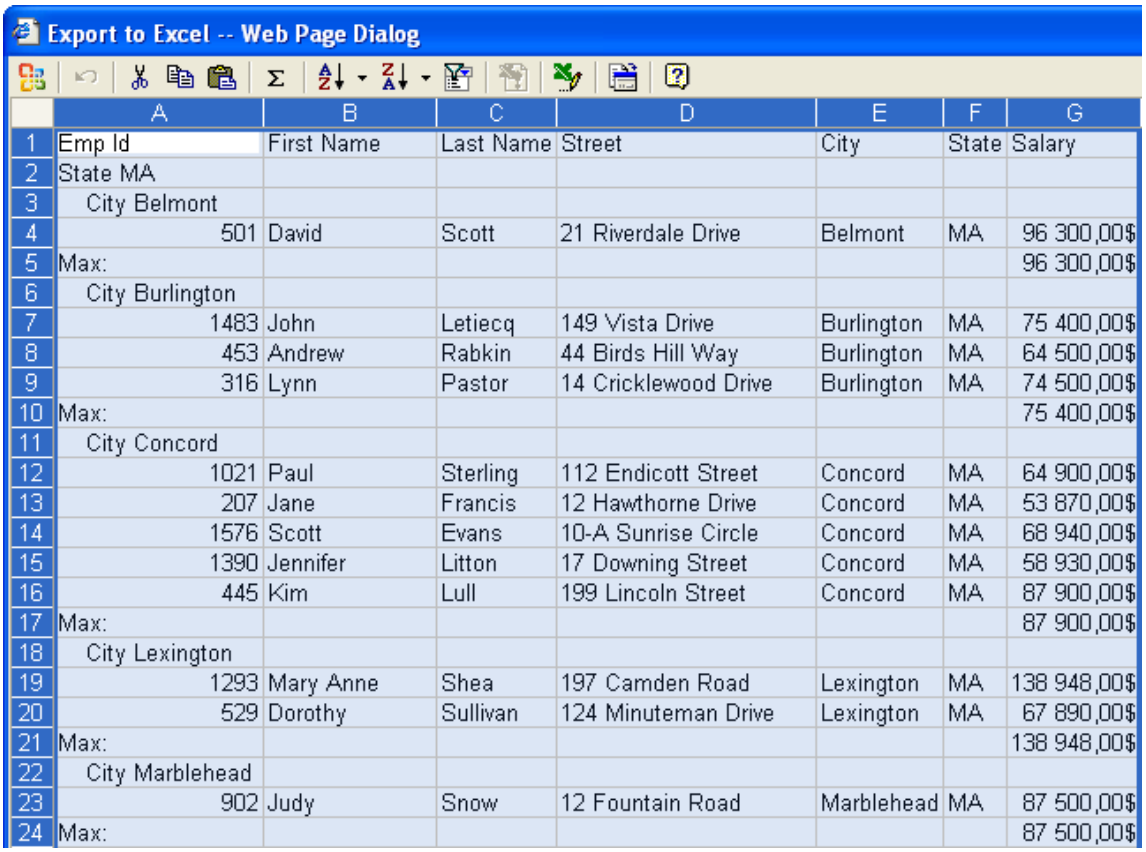
Re-run your report in ClearBI Player, and you'll see only the Massachusetts employee having salary greater than **$50,000**.



Emp Id	First Name	Last Name	Street	City	State	Salary	Sex
State MA							
City Belmont							
501	David	Scott	21 Riverdal	Belmont	MA	$96,300	M
Max:						$96,300	
City Burlington							
1483	John	Letiecq	149 Vista D	Burlington	MA	$75,400	M
453	Andrew	Rabkin	44 Birds Hil	Burlington	MA	$64,500	M
316	Lynn	Pastor	14 Cricklew	Burlington	MA	$74,500	F
Max:						$75,400	
City Concord							
1021	Paul	Sterling	112 Endicot	Concord	MA	$64,900	M
207	Jane	Francis	12 Hawthor	Concord	MA	$53,870	F
1576	Scott	Evans	10-A Sunris	Concord	MA	$68,940	M
1390	Jennifer	Litton	17 Downing	Concord	MA	$58,930	F
445	Kim	Lull	199 Lincoln	Concord	MA	$87,900	M

6.1.5 Exporting to Microsoft Excel

ClearBI can easily convert report data into Microsoft Excel format. Just press the button Excel  while viewing the report in ClearBI Player. You'll see a popup window Export to Excel with all your data.



	A	B	C	D	E	F	G
1	Emp Id	First Name	Last Name	Street	City	State	Salary
2	State MA						
3	City Belmont						
4	501	David	Scott	21 Riverdale Drive	Belmont	MA	96 300,00$
5	Max:						96 300,00$
6	City Burlington						
7	1483	John	Letiecq	149 Vista Drive	Burlington	MA	75 400,00$
8	453	Andrew	Rabkin	44 Birds Hill Way	Burlington	MA	64 500,00$
9	316	Lynn	Pastor	14 Cricklewood Drive	Burlington	MA	74 500,00$
10	Max:						75 400,00$
11	City Concord						
12	1021	Paul	Sterling	112 Endicott Street	Concord	MA	64 900,00$
13	207	Jane	Francis	12 Hawthorne Drive	Concord	MA	53 870,00$
14	1576	Scott	Evans	10-A Sunrise Circle	Concord	MA	68 940,00$
15	1390	Jennifer	Litton	17 Downing Street	Concord	MA	58 930,00$
16	445	Kim	Lull	199 Lincoln Street	Concord	MA	87 900,00$
17	Max:						87 900,00$
18	City Lexington						
19	1293	Mary Anne	Shea	197 Camden Road	Lexington	MA	138 948,00$
20	529	Dorothy	Sullivan	124 Minuteman Drive	Lexington	MA	67 890,00$
21	Max:						138 948,00$
22	City Marblehead						
23	902	Judy	Snow	12 Fountain Road	Marblehead	MA	87 500,00$
24	Max:						87 500,00$

If you press the icon in the top left corner, you'll see that this feature is implemented using Microsoft Office Web Components. Microsoft Office Components come with Microsoft Office, or they can be downloaded separately from the following Web site:

http://www.microsoft.com/downloads/details.aspx?FamilyID=982B0359-0A86-4FB2-A7EE-5F3A499515DD&displaylang=EN

Note. Export to Microsoft Excel is supported only in Internet Explorer.

7.0 USING CLEARBI WITH AJAX AND SOAP WEB SERVICES

Creating reports in AJAX is often a challenging process as report modules usually need to process lots of data, preferably on the client side to minimize the amount of information that goes through the wire. Reporters need to know how to apply formulas, group the data and calculate totals and subtotals. Add to the mix the requirement to give the end users an ability to customize the look and feel of the report, and you are facing a serious project. Let's go over some challenges and solutions of creating Web reporting

software. In this section we'll use an example of a rich Internet application that works as a consumer of a Web Service.

Challenge #1: Browsers' incompatibilities.
Reporting requires complex UI programming, but maintaining browser specific code is a huge effort. Optimization of code delivery is a work of art. Optimization of data delivery is not easy, because it needs to support ever changing protocols (SOAP, REST…)

Challenge #2: Code packaging
In the Internet Explorer, WebServices are packaged as HTC, which is not supported by Mozilla browsers.

Challenge #3: Performance

Large JavaScript projects such as reporting applications need to be loaded, pre-processed and executed on every page load.

Challenge #4: Robustness

HTTP protocol is very forgiving because it was optimized for non-reliable networks. The Web browsers were designed to display whatever has arrived with a Web page. If a page includes an image, which was not available at the moment, you'll see an icon of the broken image, and, maybe an alternative text. But what if a piece of JavaScript code will not arrive for whatever reason? There is no built-in way to ensure the delivery of JavaScript. This means that AJAX programmers need to write an additional code just to check if the application code has arrived. On top of this, Web browsers offer mediocre debugging support.

Challenge #5: Web browsers have unpredictable future

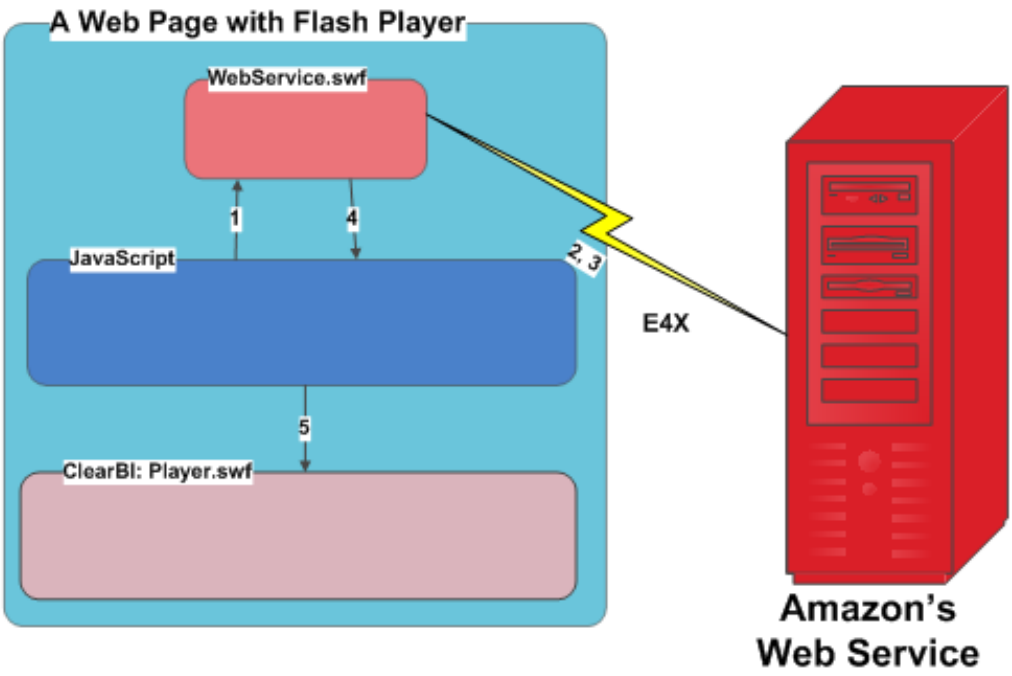
Software vendors do not seem to be eager to invest into browsers. Microsoft is investing in .Net (WPF) and Silverlight that runs on CLR. Adobe is developing AIR. Sun Microsystems is about to release a small Consumer VM for rich Internet applications written in Java and JavaFX.

All these challenges can be addressed by providing components that live in a virtual machine called Flash Player. This VM offers reliable code delivery and good development and debugging tools. It works the same way in every browser. You can also add a browser-agnostic invisible Flash Player-based agent that supports communications with Web Services regardless of what Web browser you are using. Such agents are pre-compiled, compressed and optimized for streaming and caching.

7.1 Adding a ClearBI swf to an AJAX Web page

A compiled ClearBI swf can be added as a component to AJAX applications and get the data for the report from a JavaScript array. ClearBI 1.1 also includes another component called WebService.swf, which can easily turn a SOAP WebService into an object with methods that correspond to WSDL operations. This XML processing is done using ECMAScript for XML (E4X) that allows JavaScript developers to forget about XML parsing headaches.

Below is a diagram that shows you an HTML/JavaScript application that uses an invisible Flash Player component (WebService.swf) and one visible (ClearBI.swf). In this example we'll use the data from a publicly available book search Web Service from Amazon .com.



The entire process works as follows:

1. The JavaScript code gives a URL of Amazon's WSDL to a hidden WebService.swf

2. The WebServices.swf loads WSDL from Amazon, and automatically converts XML into an object with properties (it uses E4X that will be explained below).

3. The JavaScript asks WebService.swf to call the selected operation from Amazon's WSDL.

4. WebService.swf gets the data and passes them to JavaScript (XML or JavaScript objects). If you are just interested in simplified processing of SOAP in your AJAX application, you don't need to perform the step 5 – WebService.swf is all you need.

5. To produce reports that look as shown below and can be customized by the end users, pass the data from JavaScript to ClearBI swf.

7.1.1 What's under the hood

The entire workflow consists of two major steps described below:

1. JavaScript initializes the WebService.swf object, registers event listeners, and loads the WSDL:

```
var ws = com.farata.jsfx.WebServices("MyWebService");
```

Add the WSDL load listener

```
ws.addEventListener("serviceload", onWsdlLoaded);
```

Add an error listener

```
ws.addEventListener("servicefault", onError);
```

Initialize the WebService.swf with wsdl. Load wsdl and notify ServiceLoadListeners on success.

```
ws.useService("http://soap.amazon.com/schemas2/AmazonWebServices.wsdl",
                                                        "Amazon"  );
```

This call will load Amazon's WSDL, which among others will contain the operation KeywordSearchRequest:

```
<wsdl:message name="KeywordSearchRequest">
 <wsdl:part name="KeywordSearchRequest" type="typens:KeywordRequest"/>
</wsdl:message>
```

As you can see, this operation expects an argument of type KeywordRequest, which is described in the same WSDL:

```
<xsd:complexType name="KeywordRequest">
<xsd:all>
<xsd:element name="keyword" type="xsd:string"/>
<xsd:element name="page" type="xsd:string"/>
<xsd:element name="mode" type="xsd:string"/>
<xsd:element name="tag" type="xsd:string"/>
<xsd:element name="type" type="xsd:string"/>
<xsd:element name="devtag" type="xsd:string"/>
<xsd:element name="sort" type="xsd:string" minOccurs="0"/>
<xsd:element name="variations" type="xsd:string" minOccurs="0"/>
<xsd:element name="locale" type="xsd:string" minOccurs="0"/>
</xsd:all>
</xsd:complexType>
```

Now we need to call the operation KeywordSearchRequest providing the data according to the KewordRequest format.

2. Call the Web service

```
var ws = com.farata.jsfx.WebServices("MyWebService");
```

Add the web service operation result listener.

```
ws.addEventListener("serviceresult", onXmlResult);
```

Prepare the arguments for the call (i.e.as an XML object) to find books that have the work AJAX in their titles

```
var args =  '<m:KeywordSearchRequest xmlns:m="http://soap.amazon.com">'
+ '  <KeywordSearchRequest>'    + '    <keyword>AJAX</keyword>'
+ '    <page>1</page>'    + '    <mode>books</mode>'
+ '    <tag>D3HW0PG66IPLAM</tag>'    + '    <type>lite</type>'
+ '    <devtag>D3HW0PG66IPLAM</devtag>'    + '    <sort></sort>'
+ '    <variations></variations>'    + '    <locale></locale>'
+ '  </KeywordSearchRequest>'    + '</m:KeywordSearchRequest>';
```

Call "KeywordSearchRequest" web service operation.

```
 ws.callService("Amazon", "KeywordSearchRequest", args);
```

Later you'll see how to pass the operation's arguments as a JavaScript object instead of XML. Now the data from Amazon come back and WebService.swf will receive something like this :

```
<SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:amazon="http://soap.amazon.com"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
```

```
<SOAP-ENV:Body>-<namesp1099:KeywordSearchRequestResponse
xmlns:namesp1099="http://soap.amazon.com">

<return xsi:type="amazon:ProductInfo">
<TotalResults xsi:type="xsd:string">355</TotalResults>
<TotalPages xsi:type="xsd:string">36</TotalPages>
 <Details xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="amazon:Details[10]">
<Details xsi:type="amazon:Details">
<Url
xsi:type="xsd:string">http://www.amazon.com/gp/product/0596101996%3ftag=D3HW0
PG66IPLAM%26link_code=sp1%26camp=2025%26dev-t=D3HW0PG66IPLAM</Url>
<Asin xsi:type="xsd:string">0596101996</Asin>
<ProductName xsi:type="xsd:string">JavaScript: The Definitive
Guide</ProductName>
<Catalog xsi:type="xsd:string">Book</Catalog>
<Authors xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[1]">
<Author xsi:type="xsd:string">David Flanagan</Author>
</Authors>
<ReleaseDate xsi:type="xsd:string">17 August, 2006</ReleaseDate>
<Manufacturer xsi:type="xsd:string">O'Reilly Media, Inc.</Manufacturer>
<ImageUrlSmall xsi:type="xsd:string">http://ec1.images-
amazon.com/images/I/11G8BIrxn7L.jpg</ImageUrlSmall>
<ImageUrlMedium xsi:type="xsd:string">http://ec1.images-
amazon.com/images/I/21yLdMet2BL.jpg</ImageUrlMedium>

<ImageUrlLarge xsi:type="xsd:string">http://ec1.images-
amazon.com/images/I/51OY5KP5ydL.jpg</ImageUrlLarge>
<ListPrice xsi:type="xsd:string">$49.99</ListPrice>
<Availability xsi:type="xsd:string">This item is currently not
available.</Availability>
<UsedPrice xsi:type="xsd:string">$18.49</UsedPrice>
</Details>
…
```

From the data returned by the Web service you can see that there is 355 AJAX books that can be returned as 36 pages, and Amazon has returned the page number one as was requested in the argument object. Parsing of this XML will be done for you automatically by the WebService.swf component that will turn it into an object for easy access via dot notation.

7.1.2 From SOAP to JavaScript and then to Flash Player

ClearBI was originally created for Flex/Java applications and it was working only with the server side applications that had Adobe LiveCycle Data Services, which implemented a fast communication protocol called AMF. But ClearBI 1.1 can work with the data that come in a form of JavaScript objects, a Web service or use OpenAMF, an open source implementation of communication protocol AMF0.

In this section we'll go over the process of getting the data from Amazon's SOAP Web Service to JavaScript and then to ClearBI. This is what has to be done after the Web Service is initialized by WebService.swf. This time we'll use JavaScript object as an argument to the operation KeywordSearchRequest.

Pass a JavaScript object with retrieval args to a hidden swf object, which will return the data back to JavaScript, which in turn passes the data to ClearBI swf:

```
var ws = com.farata.jsfx.WebServices("MyWebService");    // a hidden swf
```

Add a Web service operation result listener. The next line maps the WebService.swf method `serviceresult` to JavaScript's function `onJavaScriptResult` that will receive the data from WebService.swf when available.

```
ws.addEventListener("serviceresult", onJavaScriptResult);

var args = {
  KeywordSearchRequest: {
    keyword:    “AJAX",
    page:       "1",
    mode:       "books",
    tag:        "D3HW0PG66IPLAM",
    type:       "lite",
    devtag:     "D3HW0PG66IPLAM",
    sort:       null,
    variations: null,
    locale:     null
  }

};
```

Call Amazon’s "KeywordSearchRequest" web service operation.

```
ws.Amazon.KeywordSearchRequest( args );
```

…

The JavaScript function receives the data arriving as JavaScript into a variable lastData:

```
function onJavaScriptResult(ev) {
    lastData = ev.result.Details;
    this.removeEventListener("serviceresult", onJavaScriptResult);
  }
```

Now we can pass the data from lastData to clearBI.swf for displaying and customizing the report.

7.1.3 Passing the data to ClearBI from a JavaScript array

In JavaScript, get a reference to Flash Player object and call the function `loadReport` giving the report name (a template customized by the user and stored in a database) and the data itself:

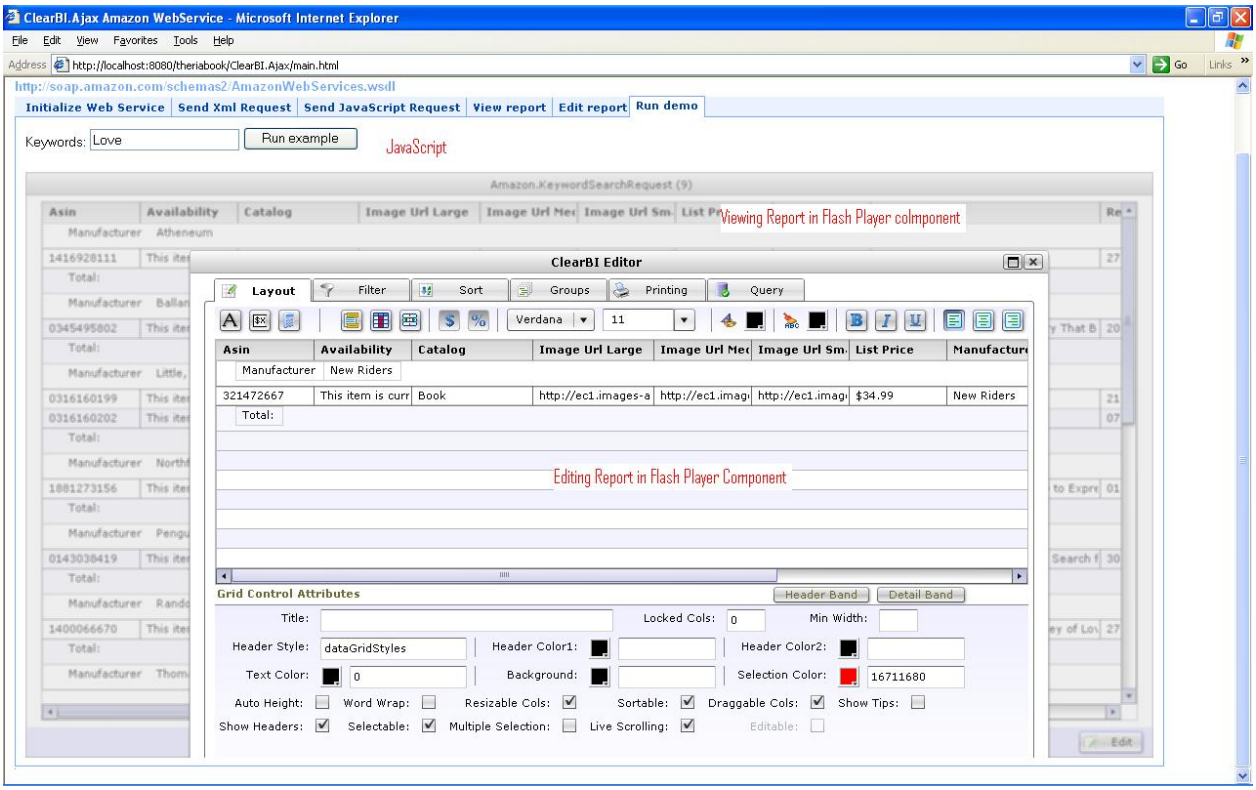
```
var report
var = $plugin(“ClearBIPlayer”);  // a helper function to return reporter’s id

report.loadReport(reportName, lastData);
```

The data exchange between JavaScript and ClearBI.swf is supported internally by the ActionScript class `ExternalInterface`. The variable `report` represents ClearBI.swf, which internally (in ActionScript) allowed JavaScript calling its function `loadReport` that’s mapped to ActionScript’s function `loadReport`:

```
ExternalInterface.addCallback(“loadReport", loadReport);
```

That’s all there is to it. The end users can run and customize reports themselves, and their Web browser window may look as shown below. The top portion of the window is rendered by JavaScript while the report and its editor are rendered by Flash Player 9.



A proper integration of JavaScript and Flash Player's components can enrich your AJAX application with powerful reporting capabilities offered by ClearBI (written in Adobe Flex).

You can also use the WebService.swf to communicate with SOAP Web Services regardless if you are planning to use ClearBI or not. You can download the description of the API for JavaScript wrapper to WebService.swf at http://www.myflex.org/demo/webservice_swf/doc/ . The WebService.swf is located at http://www.myflex.org/demo/webservice_swf/WebService.swf .

This demo application that uses JavaScript, WebService.swf and ClearBI.swf is deployed at http://www.myflex.org/theriabook/ClearBI.Ajax/main.html . It looks like this:

Using Amazon Web Service

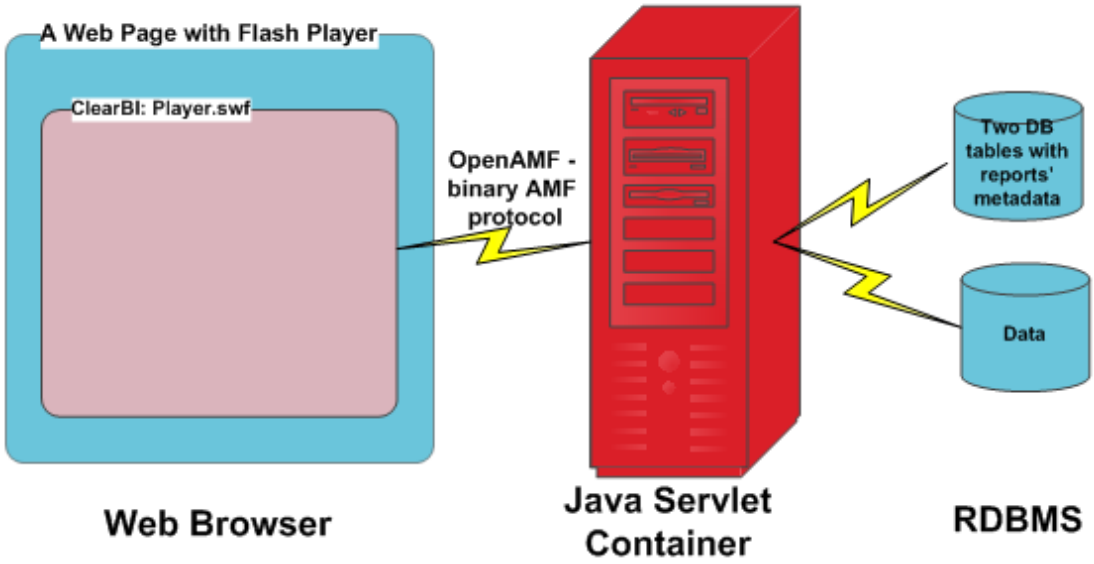
In the first three tabs you'll be able to play with sending XML and JavaScript requests to Amazon, and the last three tabs are supported by ClearBI.

Watch a short pre-recorded demo that shows working with a sample Employees database and using ClearBI editor for report customization. This demo is located at http://myflex.org/screencast/clearbi_ajax/clearbi_ajax.html . It shows how end-users can add grouping, modify styling, and apply formulas.

8.0 USING CLEARBI WITH OPENAMF

Creating of reports using OpenAMF for data retrieval work the same way as with LCDS.
First you select OpenAMF as deployment option on the window shown in the beginning of section 3.1.

Then you create and configure your CDB project as explained in section 3.1. A sample diagram below illustrates the data flow in case of OpenAMF deployment.



The above diagram shows the following players:

Client: a Web browser with Flash Player 9 plugin or an ActiveX

Mid-Tier: A servlet container with an open source component called OpenAMF (5 jars), open source Flex SDK, ClearBI's jars to support report compilation and hot deployment, JOTM, JTS, JDBC drivers

Report persistence: ClearBI uses two database tables to store reports' metadata (can be co-located with other data).

Report Publishing: Reports can be compiled into swf files and published on the Web.

You can generate a sample CDB project using OpenAMF as a means of communication between Flex and Java by following the steps from section 4 above. Just set the deployment type to OpenAMF as explained in section 3.1. Configure CDB build and the ant script will compile and deploy your project.

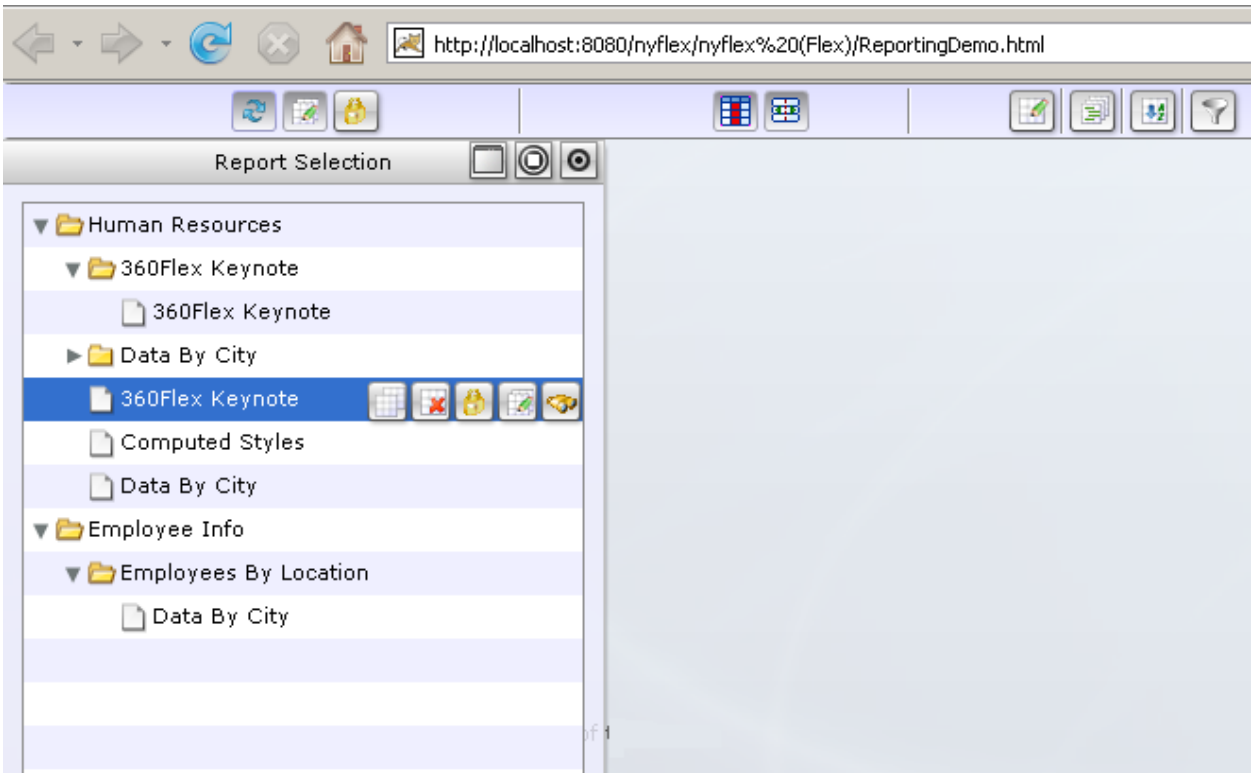
Please note, that the content of the deployment directory under you application server is not the same as with LCDS deployment. The auto-generated file Web.xml located under WEB-INF directory uses OpenAmf classes. The folder WEB-INF\openamf\com\farata\datasource contains the auto-generated destinations Employee.xml, Orer.xml and StoredProcedures.xml.

9.0 REPORT ADMINISTRATION WITH CLEARBI

ClearBI 1.1 Server Edition supports user and report administration. It provides a number of database tables in the clearbi database that allow you to created users, assign reports and arrange reports by "folders". While user authentication and authorization does not belong to the domain of any reporter, we've provided a reference implementation of the report administration tool so you can customize it based on your organization's security requirements.

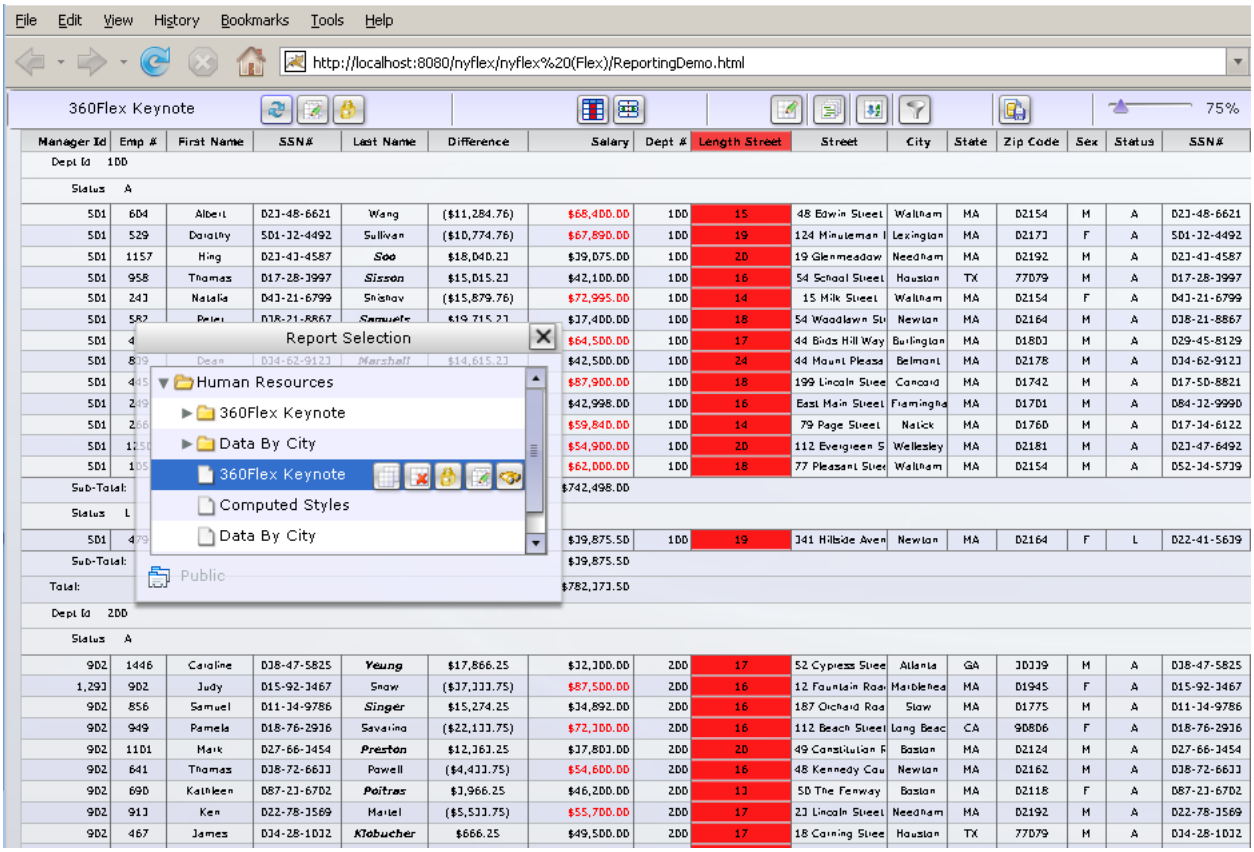
This reference implementation becomes available after your enable the ClearBI Server Edition in your Flex project (right-click on Flex project and select the menu ClearBI>Enable ClearBI Server Edition). Right-click on the ReportingAdminDemo.mxml and run it as Flex application. You'll see the logon screen that will allow you to enter report administrator's id and password. Remember, it's just a reference implementation, and you'll be able to modify the source code of this application to attach another authorization API - look for the call `administration.userIsAdmin(userName.text` in ReportingAdminDemo.mxml and replace it to address your needs.

The screen below shows the Report Selection section on the left. It's a sliding panel that lets the report administrator arrange reports in folders and allow (restrict) users to work with reports. Three icons on the title bar of the Report Selection window allow you to pin this panel to the browser's window so it won't move, go back to sliding mode or turn it into a freely floating panel.

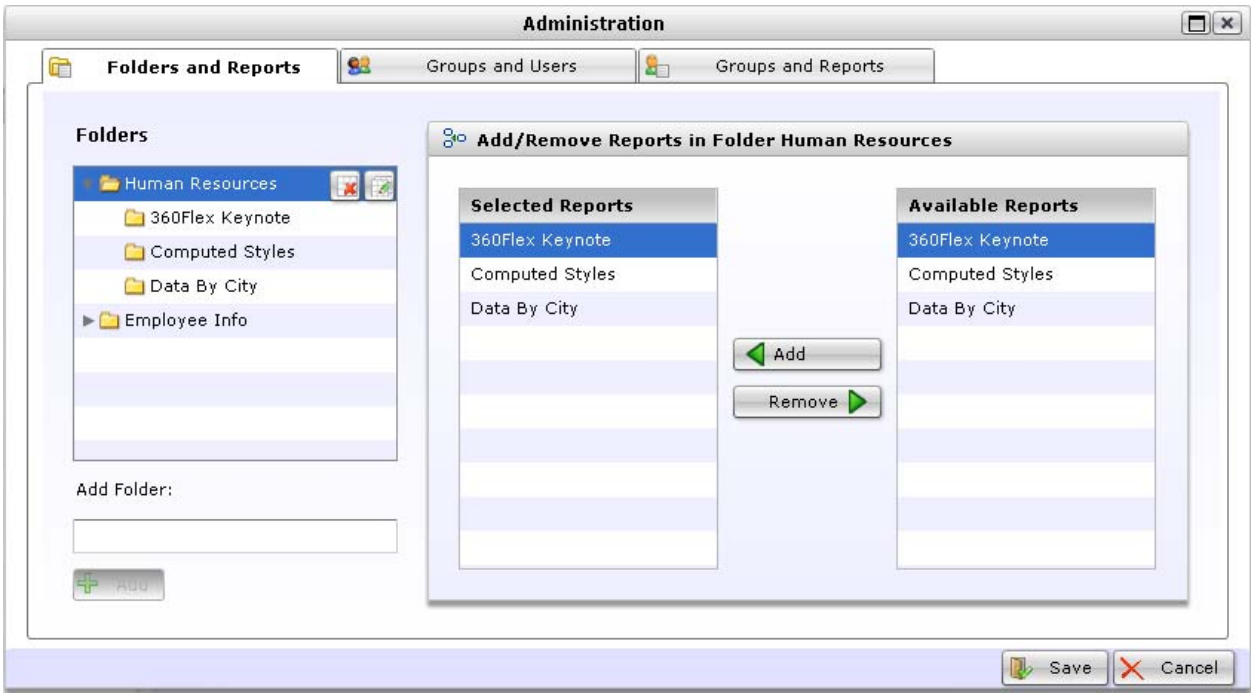


When you select a report, it shows a number of icons right in the selected row as shown above. These icons allow administrator save, remove, edit and preview the selected report without the need to leave the

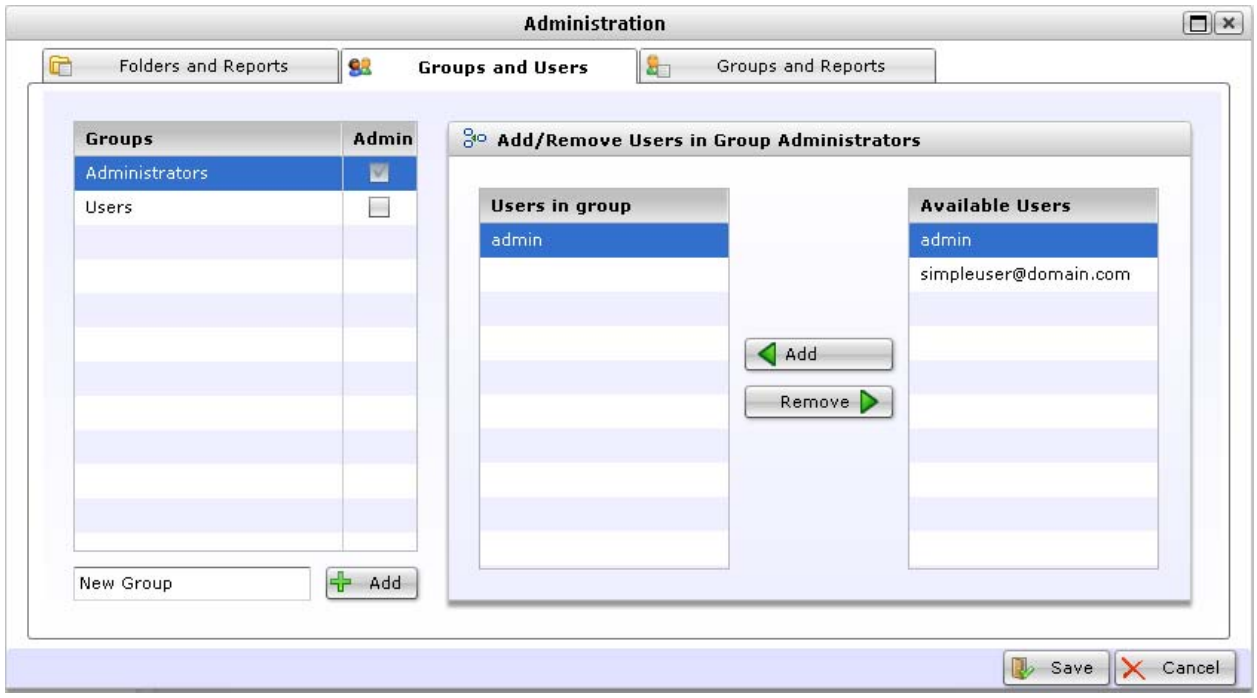
administration module. The next snapshot depicts a view when the administrator previews the report while turning the Report Selection window into a floating panel.



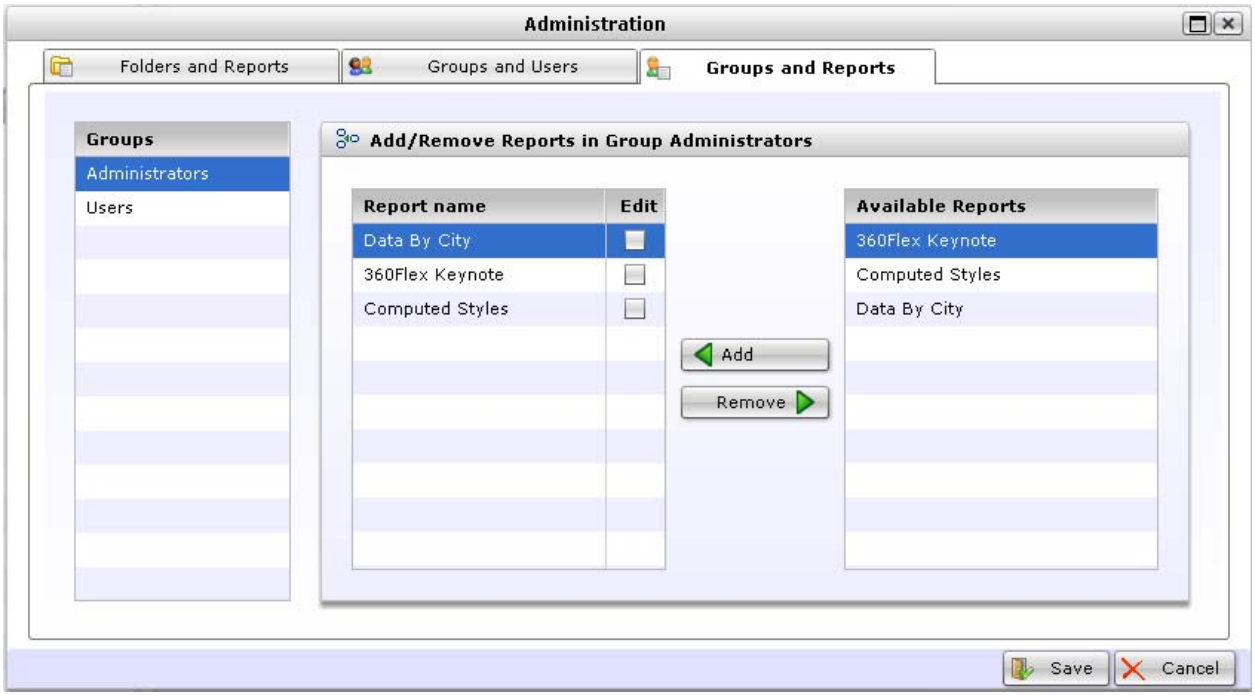
Press the little icon with an image of a lock, and it'll open the report administration panel, which is self explanatory. The next screenshot shows how you can create folders and add/remove reports. The folders' names and report assignments are stored in the clearbi database.



The Groups and Users tab looks as follows:



And this is the Groups and Reports tab:



10.0 APPENDIX A. CLEARBI PLUGIN – INSTALLATION AND CONFIGURATION

ClearBI 1.0 plugin requires the following software:

1. Java Development Kit version 5 or later
2. Eclipse IDE 3.2
3. Flex Builder 2.0.1 or later
4. Flex Builder Hot Fix 2
5. LiveCycle Data Services ES 2.5 or later
6. Clear Data Builder plugin.

You'll also need to have a servlet container for deployment of LCDS and a relational DBMS where you can store the ClearBI metadata (typically, it's the same DBMS where you store application's data). For your convenience, we've included installation instructions for Apache Tomcat and MySQL Server in section 9 of this document.

10.1 Installing Java SE Development Kit

Install Java Development Kit v. 5.0 or later. Get the latest JDK from http://java.sun.com/javase/downloads/index.jsp (no special configuration is required, just run the install with default options).

10.2 Installing Eclipse IDE

Download and install Eclipse 3.2 IDE or later at http://download.eclipse.org/eclipse/downloads/. Installation is a one step process, for example in Windows you just need to unzip the downloaded file into the root directory of your c: drive. It'll create the folder named *eclipse*.

10.3 Installing Flex Builder

Download and install a trial version of Flex Builder 2.0.1 or later from http://www.adobe.com/products/flex/. During the installation select a plugin version of Flex Builder and then specified location of Eclipse IDE installed in the previous step (i.e. **c:\eclipse**).

10.4 Installing LiveCycle Data Services (optional)

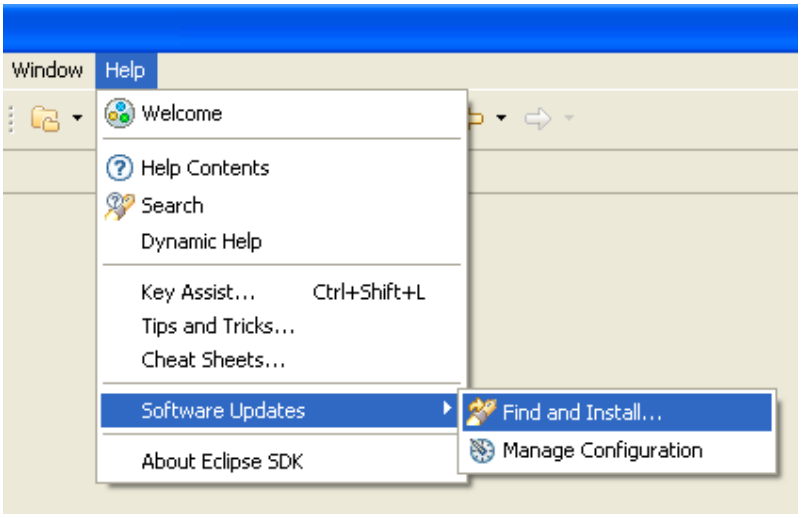
Download and install LiveCycle Data Services Express from http://www.adobe.com/products/flex/. During the installation select the option "J2EE Web Application". If you are planning to work with integrated JRun, reconfigure it to work with Java 5 or later.

10.5 Installing Clear Data Builder Plugin

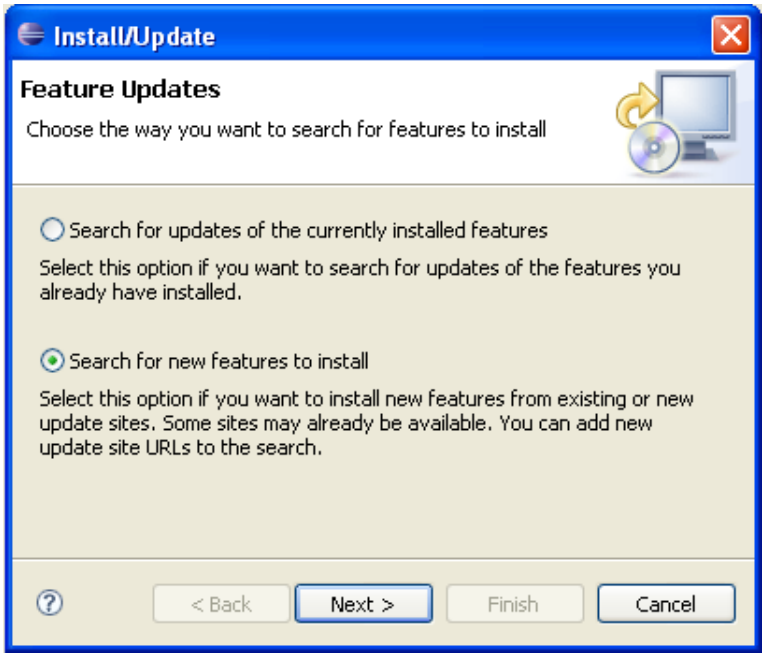
This plugin will be installed automatically as a part of ClearBI.

10.6 Installing ClearBI Plugin

1. Start Eclipse IDE and select the menu **Help > Software Updates > Find and Install**.



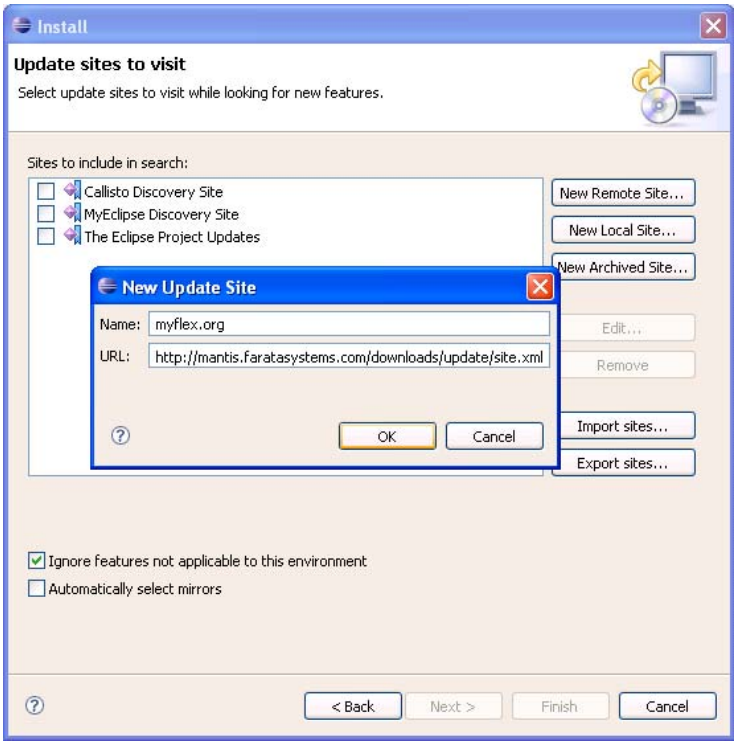
2. Select **Search for new features to install** and press the button **Next**.



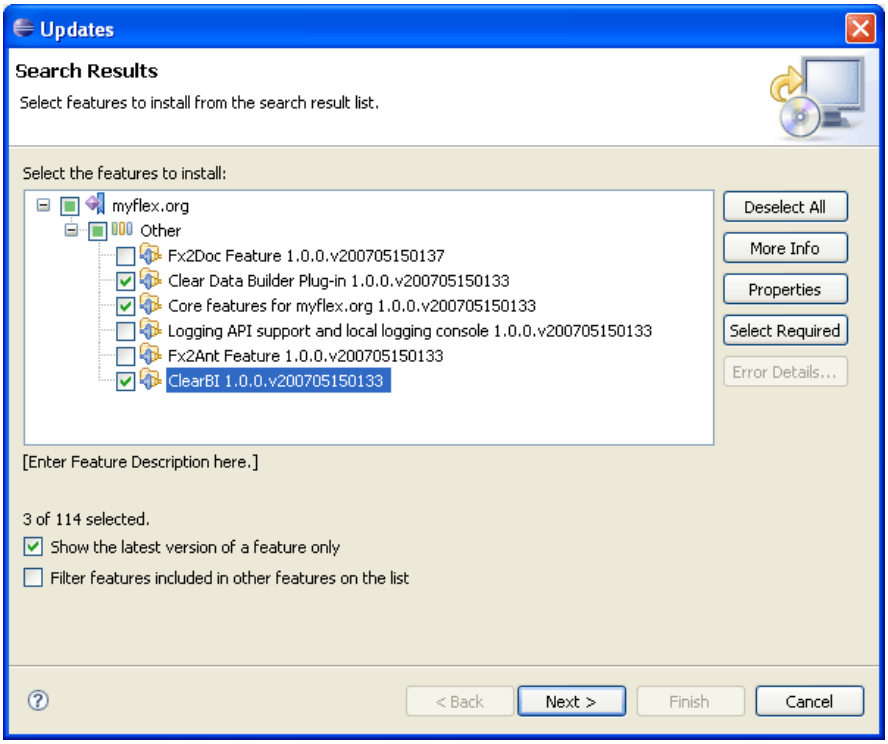
3. You'll see a popup window with a list of available Web sites with Eclipse-related software. Press the button **New Remote Site...** Enter the name of the site (for example, "myflex.org") and the following URL: http://mantis.faratasystems.com/downloads/update/site.xml.

Note. If you'd like to get the up to date version of ClearBI, get its nightly build version located at the following URL: http://mantis.faratasystems.com/downloads/nightly/update-nightly . This URL is also recommended for all users that would like to download the trial version of ClearBI.

Press **OK**, and then **Finish**.

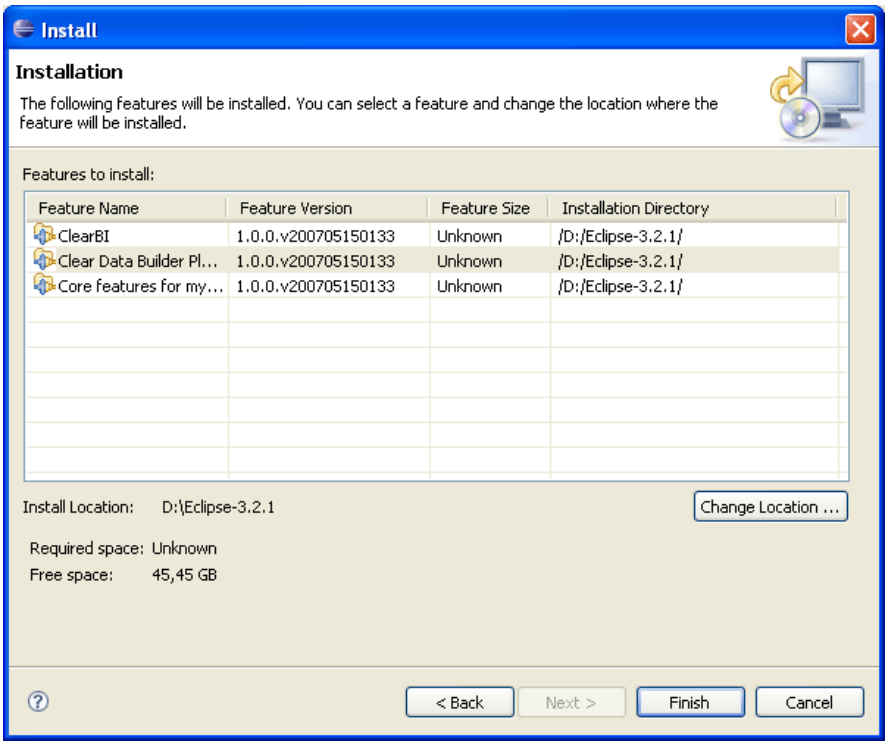


4. In the popup window shown below, select the checkbox **ClearBI Plugin** and press the buttons **Select Required** and then **Next**.

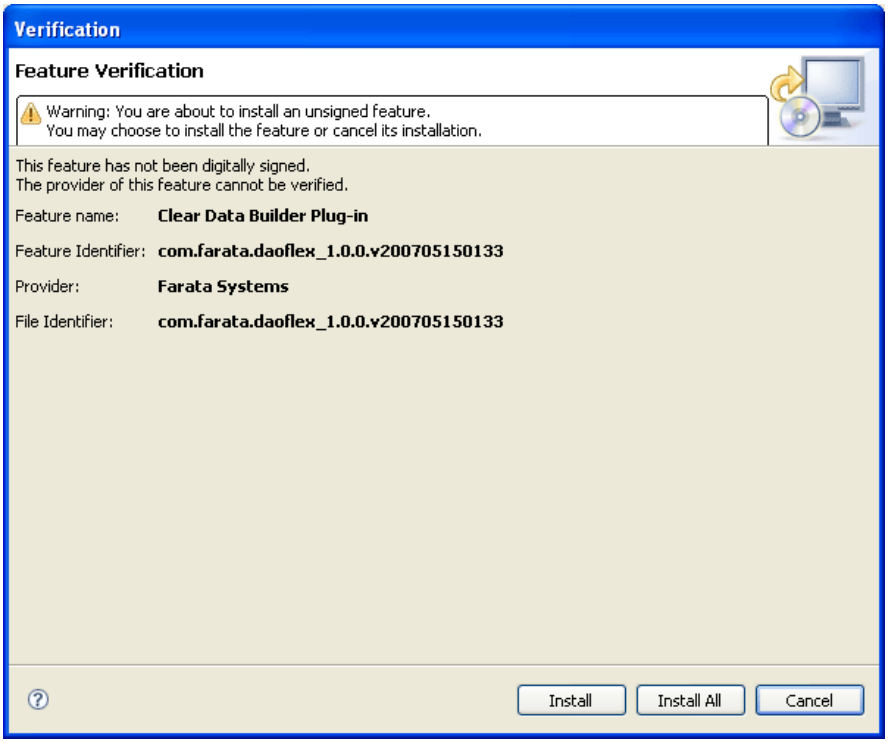


5. Read the license agreement, and select **"I accept the terms in the license agreements"** followed by **Next.**

6. Press the button **Finish** on the confirmation page shown below.



7. Press the button **Install All** on the verification window.



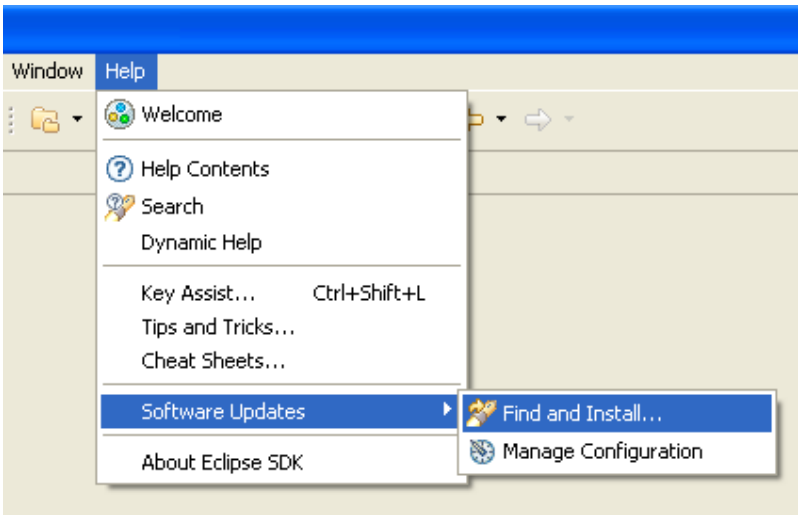
After installation is complete, restart Eclipse and install the license file as explained in section 7.8.

10.7 Installing ClearBI Plugin from a local site

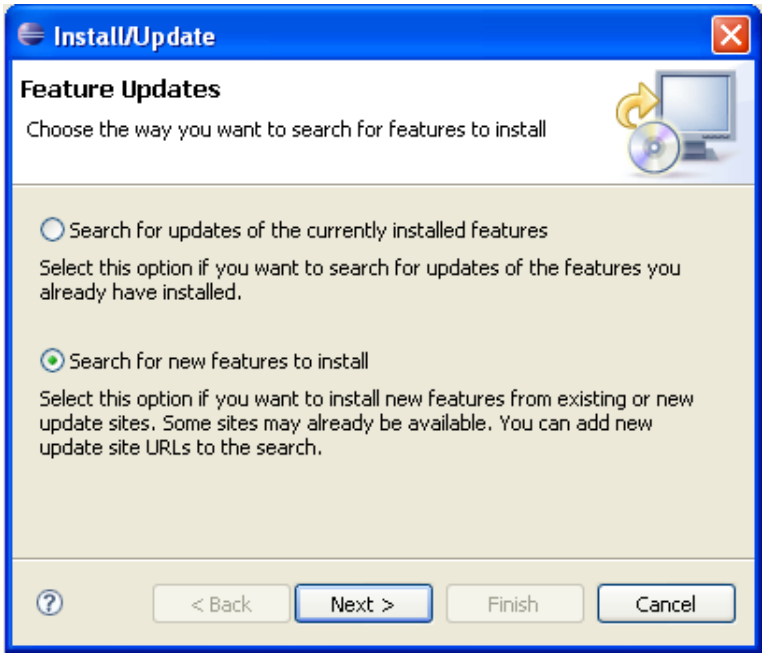
If your computer is protected by a corporate firewall, you might experience difficulties working with remote update site. As a workaround to this issue, create the local update site as described below and load all required plugins from there.

1. Download the zip file at http://mantis.faratasystems.com/downloads/local-update/update.zip, which will become your local update site.

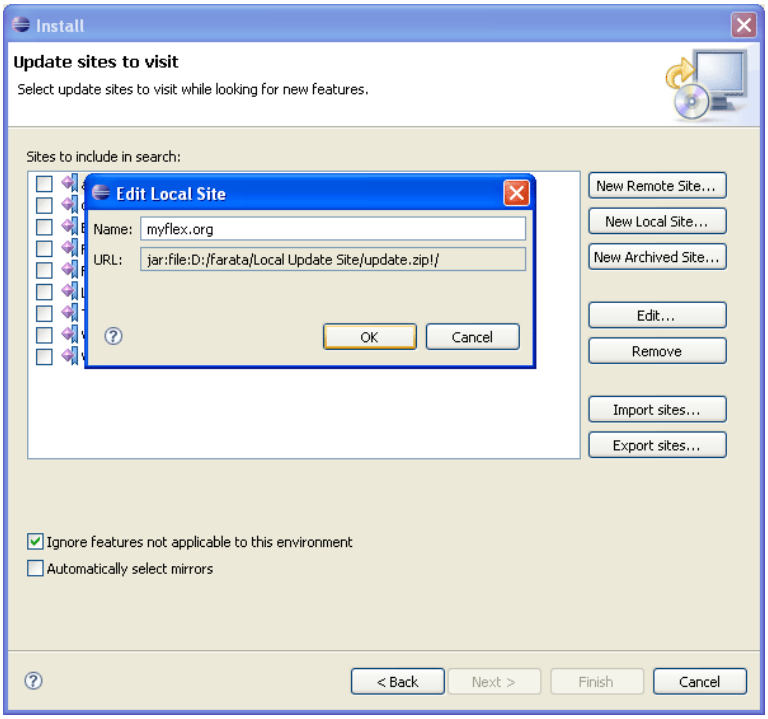
2. Start Eclipse IDE and select the menu **Help > Software Updates > Find and Install**.



3. Select **Search for new features to install** and press **Next**.



4. You'll see a popup window with a list of available Web sites with Eclipse-related software. Press the button **New Archived Site...** select the directory where the file update.zip is located. Enter the name of the site (for example, "myflex.org"). Press **OK**, and then **Finish**.



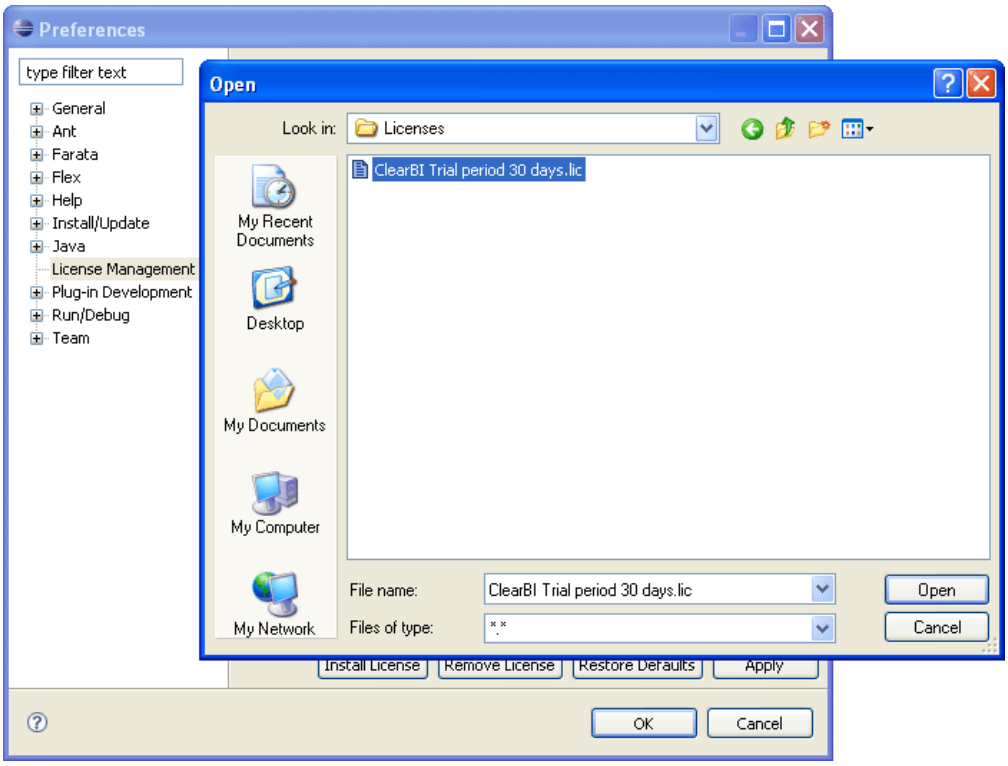
5. Continue installation starting from step 4 in section **7.6**.

10.8 Installing the ClearBI License File

1. Download a free trial or purchase the ClearBI license at www.myflex.org.

2. Select Eclipse menu **Windows > Preferences**.

3. Select **License Management** from the list on the left. On the popup window press **Install License** and select the downloaded license file.



4. The selected license file becomes your default license. Please read the license agreement by selecting **com.farata.flexbi** in the **Products under license** section.

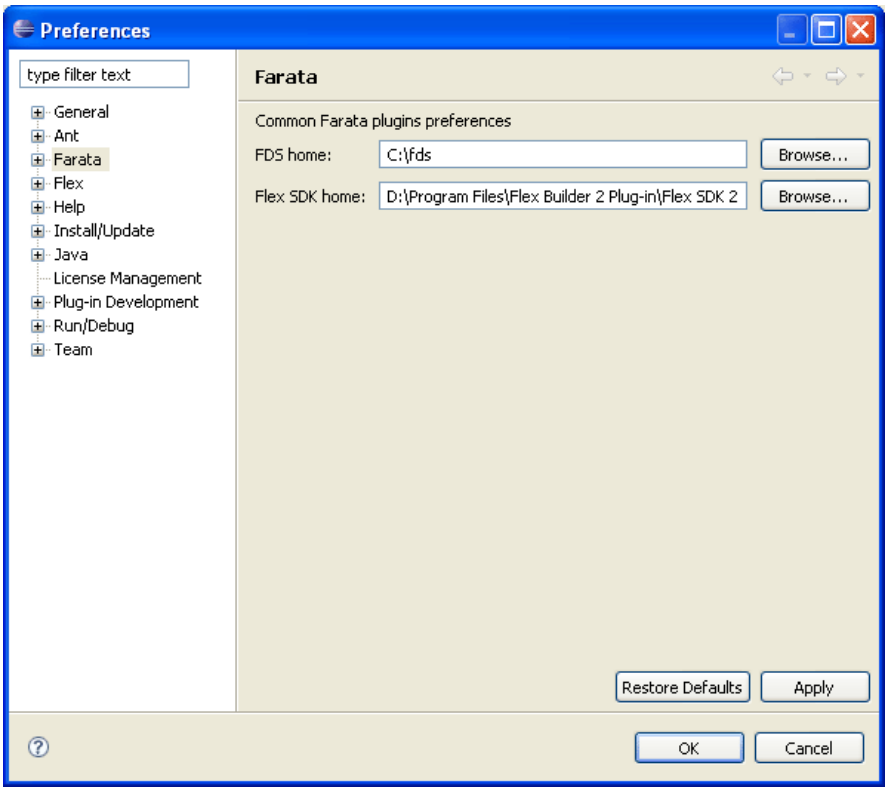
5. Install the CDB license.

6. After adding both licenses press the **OK** button and restart Eclipse.

10.9 Configuring LiveCycle Data Services for ClearBI.

1. Select Eclipse menu **Windows > Preferences**.
2. Select **Farata** from the list on the left.

3. Specify the location of LCDS[2] on your computer and press OK.



11.0 APPENDIX B. CONFIGURING CLEARBI SERVER EDITION

End users can work with ClearBI reports delivered over the Web without the need to have any additional software other than Flash Player. But to do this, several Java code libraries should be deployed in your Java servlet container as described in section 4.2.2 above.

To enable report persistence and CLearBI user administration, the application developer has to create a number of additional database tables (see 4.1.1). These tables can be created either in the database where application data reside, or in a separate database.

The next section describes creation of a database connection pool that points at this database using Apache Tomcat server as an example.

11.1 Creating DB connection pool for persisting reports

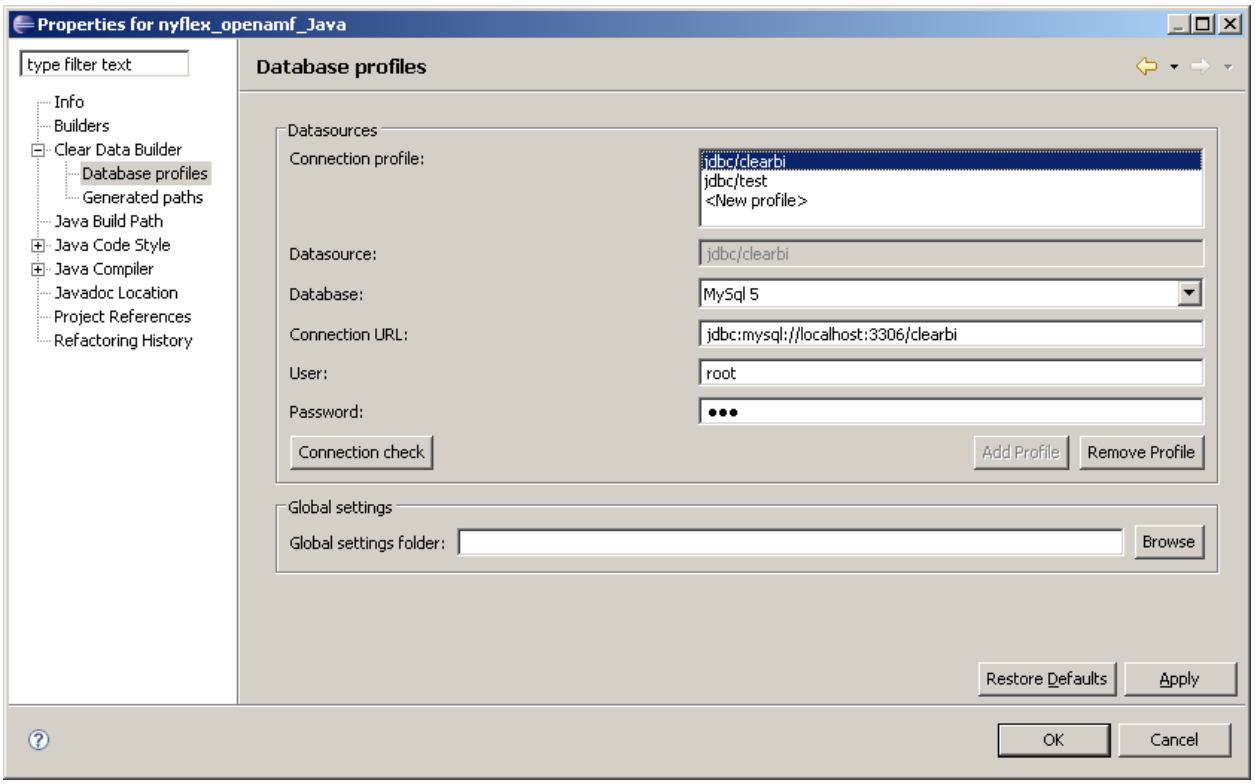
When ClearBI tables are created, you need to configure an additional database connection pool named **clearbi**. This pool should point at the database for storage of the metadata required by ClearBI. The clearbi database profile can be created in Eclipse[3] as follows:

1. Right-click on your Java project and select **Properties**.

[2] LCDS used to be called Flex Data Services, hence you may still see FDS here and there.

[3] Clear Data Builder plugin has to be installed in Eclipse.

2. Select **Clear Data Builder > Database profiles** from the list on the left.
3. Create a new database profile. The name of this database connection profile must be **clearbi**.



4. Run one of the DDL scripts for your DBMS provided in the directory sql of your example project after you upgrade it to ClearBI. The ddl script names start with clearbi_. You can also find DDL in the folder eclipse\plugins\com.farata.flexbi…\resources\export\sql). If the DDL script for your database is not included, please get one of the provided scripts and modify it to meet requirements of your DBMS.

After this database is created, you'll be able to publish your reports from your ClearBI Editor.

12.0 APPENDIX C. INSTALLING APACHE TOMCAT AND MYSQL SERVER

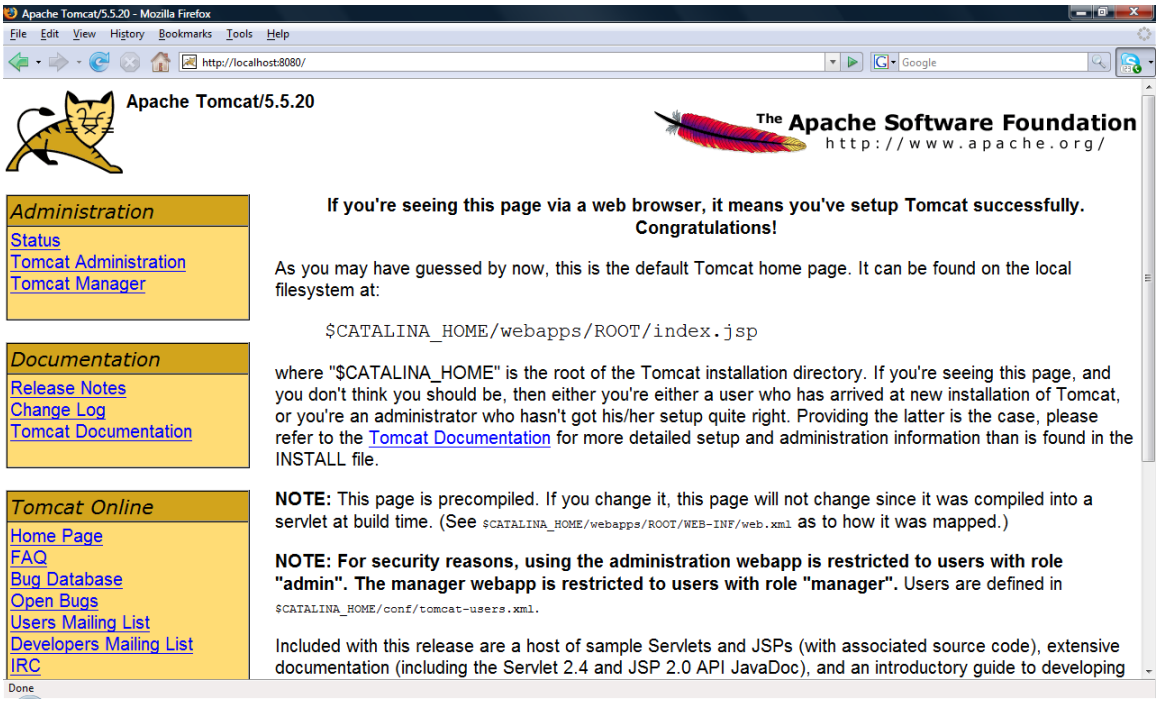
The easiest way to test the entire process of creating ClearBI reports is by installing open source Apache Tomcat as a host of LCDS.

MySQL Server DBMS can serve as a host of your data and ClearBI metadata.

12.1 Installing Apache Tomcat

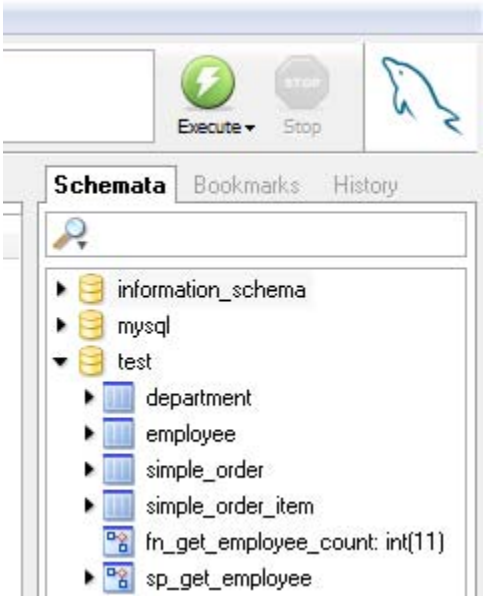
1. Download Tomcat 5.5.20 or later - http://tomcat.apache.org/download-55.cgi
   Select a zip file from Core Downloads section.

2. Unzip the entire archive in your c: drive. A directory called apache-tomcat-5.5.20 (or similar) will be created.
3. You can find Tomcat documentation by opening the file index.hml located in C:\apache-tomcat-5.5.20\webapps\tomcat-docs.
4. Start Tomcat from a command line: go to it's bin directory and type startup.
5. By default, Tomcat starts on port 8080. To make sure that it's up and running, open you browser and point it at http://localhost:8080. You should see the window as shown below:

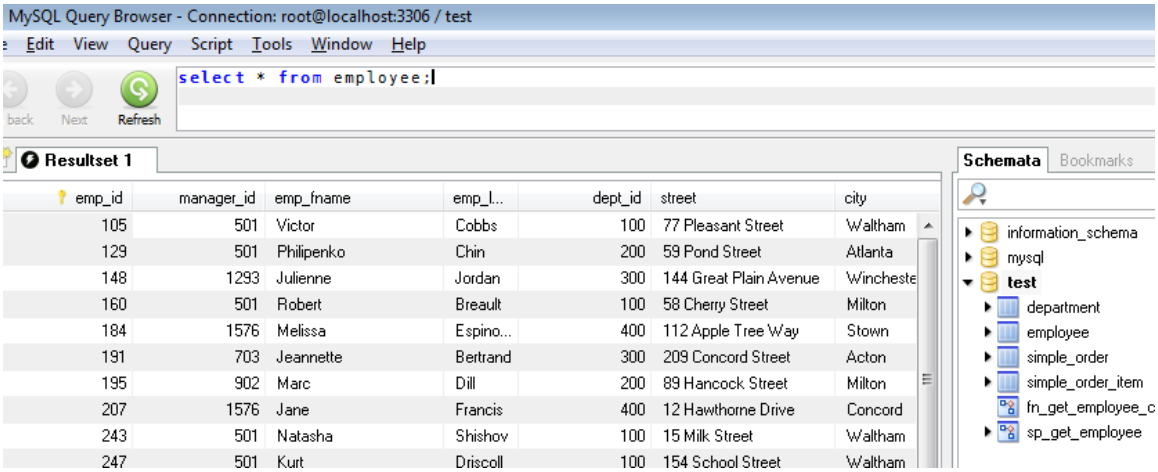


12.2 Installing MySQL 5.0 Server and Creating the Database

1. Download the popular open source database community server MySQL. For example pick the Windows installer at http://dev.mysql.com/downloads/mysql/5.0.html#downloads. Get the file Windows (x86) ZIP/Setup.EXE (if you use Windows) and run the setup.exe, which will install MySql as a service and create MySql menu items in the Windows Start menu. You'll find MySQL documentation in the directory C:\mysql-5.0.37-win32\Docs.
2. Please pay attention, at the end of install, it'll ask you about creation of the password for the root user. Do not skip this part to avoid troubles with connecting to MySql Server.
3. Even though MySQL server allows you to work with the data from a command line, it's a lot easier to use a GUI client. Download and install MySQL GUI Tools Bundle located at http://mysql.org/downloads/gui-tools/5.0.html . You'll find several new items, and we'll use MySQL Administrator for testing our a sample Employees database and MySQL Query Browser for running sample queries.
4. Start MySql Query browser and you should see new tables under the test schema on the right hand side:



5. In MySQL Query Browser open the menu File > Change Default Schema and select test as your default schema.
6. Enter and execute a test query select * from employee; and you should see a result set as shown below:



7. Start MySQL Administrator, select the option User Administration and create a new user: in the User Information tab enter lowercase id *dba* and the password *sql*. Press the tab Schema Privileges, select the test schema and assign to the user all privileges by pressing the button **<<**. Press the button Apply Changes.

13.0 APPENDIX D. CONTACT INFORMATION

For CleaBI support, please send an email at support@faratasystems.com.

You can also leave a comment under the Contacts menu at Farata Systems Web site: http://www.faratasystems.com.

For sales please call: 1-888-520-2220 (from the USA) or 1-917-304-4553 (from outside the USA).