JasperServer User's Guide

Version 2.1

© 2007 JasperSoft Corporation. All rights reserved. Printed in the U.S.A. JasperSoft, the JasperSoft logo, JasperAnalysis, JasperServer, JasperETL, JasperReports, JasperStudio, iReport, and Jasper4 products are trademarks and/or registered trademarks of JapserSoft Corporation in the United States and in jurisdictions throughout the world. All other company and product names are or may be trade names or trademarks of their respective owners.

Table of Contents

1 What is JasperServer?

JasperServer is a framework that lets you build a Web application around JasperReports and use JasperAnalysis, an implementation of Online Analytical Processing or OLAP, to perform on-line analytical tasks. Using JasperReports, it lets you create, schedule and run reports, and manage resources, roles, and permissions. It also provides a sample Web-based user interface that lets you create and manage reports and other resources. Using JasperAnalysis, it lets you interact with analysis views to perform operations such as slice/dice, drill down/dirll through, etc. It also lets you create new analysis views with OLAP schemas and various data sources.

2 Who is this guide for?

This guide is for users of the sample user interface provided with JasperServer.

With respect to JasperReports, in the current release, you must use iReport, or write JRXML code, to create the JRXML file that is the basis of the report. The sample JasperReports UI only assembles the JRXML file and other resources into a report that you can run. This UI cannot create a JRXML file.

***Note**: To use this user interface, you must be very familiar with the JasperReport (.jrxml) files that are the basis of the JasperServer report. It is assumed that you understand JRXML and you are very familiar with all the resources that a JRXML file refers to.*

With respect to JasperAnalysis, in the current release, you must supply OLAP schemas in XML format, corrresponding data sources as JDBC or JNDI, and MDX queries. The sample JasperAnalysis UI only assembles the schemas, data source and MDX queries into analysis views that you can interact. This UI cannot create schemas or MDX queries.

3 Related documents

- JasperServer Installation Guide
- JasperServer Developer’s Guide
- JasperServer Web Services Client Install Guide
- JasperServer Database Scripts install Guide
- The Definitive Guide to JasperReports
- iReport manual

4 Terms and concepts

4.1 JasperServer Report

A *JasperServer report* is what you actually run to generate output. A report consists of a set of resources. (See the *Resource* section.)

A JasperServer report may contain resources in either the file system or the repository.

4.2 JRXML

JRXML is JasperReport's XML-based report definition language. Every JasperServer report must contain one main JRXML file and one for each subreport it might contain.

4.3 Resource

A *resource* is one of the following elements of a report:

- a JRXML file (see the *JRXML* section)
- a data source (JNDI or JDBC)
- a query
- fonts
- images
- data types (see the *Data type* section)
- JAR files for JasperReports scriptlets
- resource bundles for localization and internationalization
- subreports

Every JasperServer report must contain a JRXML file and a data source. The other resources may or may not be required, depending on the contents of the JRXML file. Therefore, to create a report, you must know the resources required by the JRXML file.

You can retrieve resources from the file system or the repository.

4.4 Data source

The three supported data source types are JDBC, JNDI and Bean.

For reports with MDX queries in their main JRXMLs, OLAP client connections are used as data sources. Such reports use the Mondrian and XML/A query executers included in JasperReports when launched.

4.5 Query

Instead of the query stored inside the JRXML file, a report can make use of a reusable query from the repository.

4.6 Image

You can point to an image in one of the following three ways:

- using a URL
- using the classpath
- using the "repository" syntax (i.e., repo:/image/logo.jpg)

4.7 Input control

Input controls are values that can be mapped to JasperReports parameters.

4.8 Data type

In JasperServer, a *data type* is a set of constraints on the value of an input control. It contains more information (for example, maximum and minumum values) than a data type in most programming languages.

4.9 Analysis view

An *analysis view* consists of an OLAP client connection and a MDX query for providing an entry point to OLAP operations, such as slice/dice, drill down and drill through.

4.10 OLAP client connection

An *olap client connection* contains the connection definitions for retrieving an analysis view. The two types of OLAP client connection are Mondrian connection and XML/A connection.

4.11 Mondrian connection

A *Mondrian connection* is a type of OLAP client connection, which consists of an OLAP schema and a data source used to access an analysis view.

4.12 XML/A connection

An *XML/A connection* is a type of OLAP client connection, which consists of SOAP definitions used to access an analysis view on a remote XML/A Provider.

4.13 Mondrian XML/A source

A *Mondrian XML/A source* is a server-side XML/A source definition for defining how XML/A connections will be processed from remote clients to access an analysis view via a Mondrian connection on this server.

4.14 Analysis schema

An analysis *schema* is a file resource for defining and documenting the structure of OLAP cubes, dimensions and measures in XML format.

4.15 MDX query

MDX query is a language for querying multidimensional objects, such as cubes, and return cube data for analytical processing.

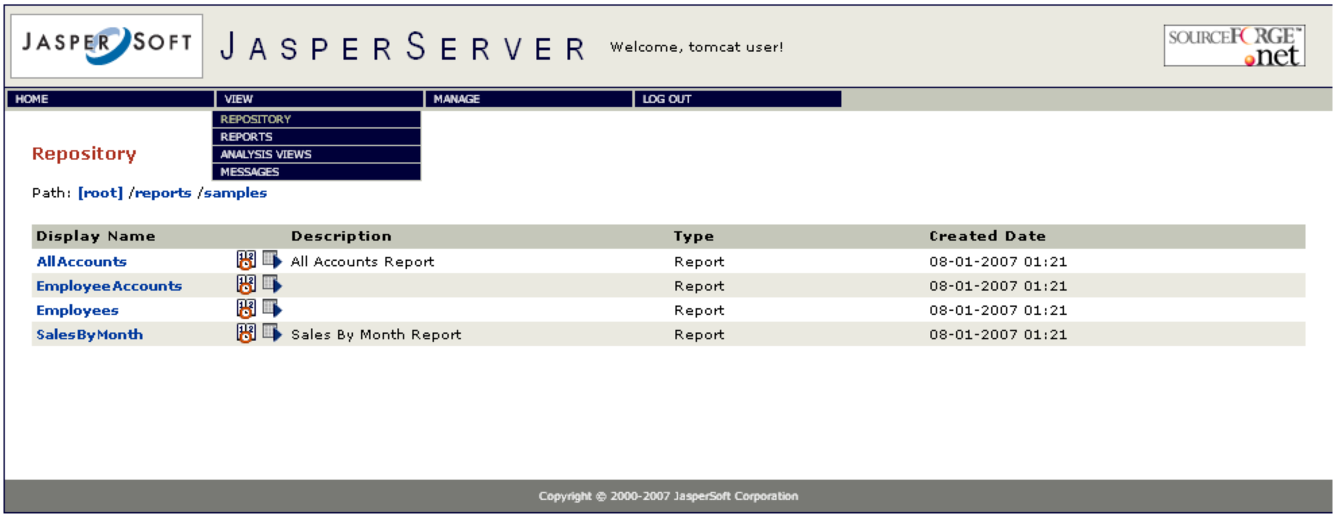
5 Accessing Reports and Analytics

The user interface features four menu buttons:

Menu name	*Function*
Home	Describes JasperServer.
View	Shows metadata for, and lets you run, reports in the repository. All reports for which you have read permission appear regardless of which folder they're in.
Manage	Contains the Repository, Roles, and Users submenus. It appears only if you have administration privilege.
Logout	Logs you out of JasperServer.

5.1 Running a report

First, you'll run the sample SalesByMonth report. This report contains four input controls: a text input control, a Boolean checkbox, a list input control, and a date input control.



Welcome, tomcat user!

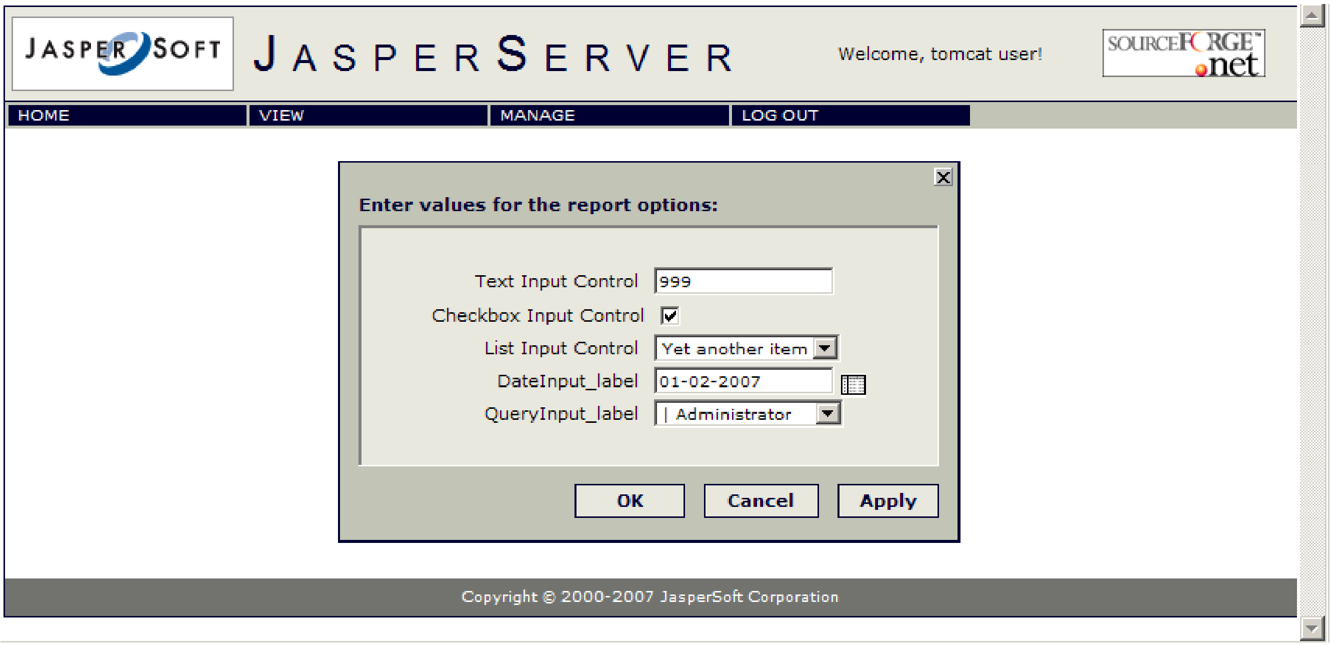
Repository

Path: [root] /reports /samples

Display Name	Description	Type	Created Date
AllAccounts	All Accounts Report	Report	08-01-2007 01:21
EmployeeAccounts		Report	08-01-2007 01:21
Employees		Report	08-01-2007 01:21
SalesByMonth	Sales By Month Report	Report	08-01-2007 01:21

Copyright © 2000-2007 JasperSoft Corporation

- Mouse on the VIEW menu.
- Click the REPORTS menuitem.
- Click the samples folder
- Click the SalesByMonth sample. A list of controls appears.
- For the Text Input Control, enter *999*.
- Check the Checkbox Input Control.
- In the *List Input Control* dropdown list, select *Yet Another Item*.
- In the *Date Input Control*, click the calendar icon (on the right).
- Choose a date.



Enter values for the report options:

Text Input Control: 999

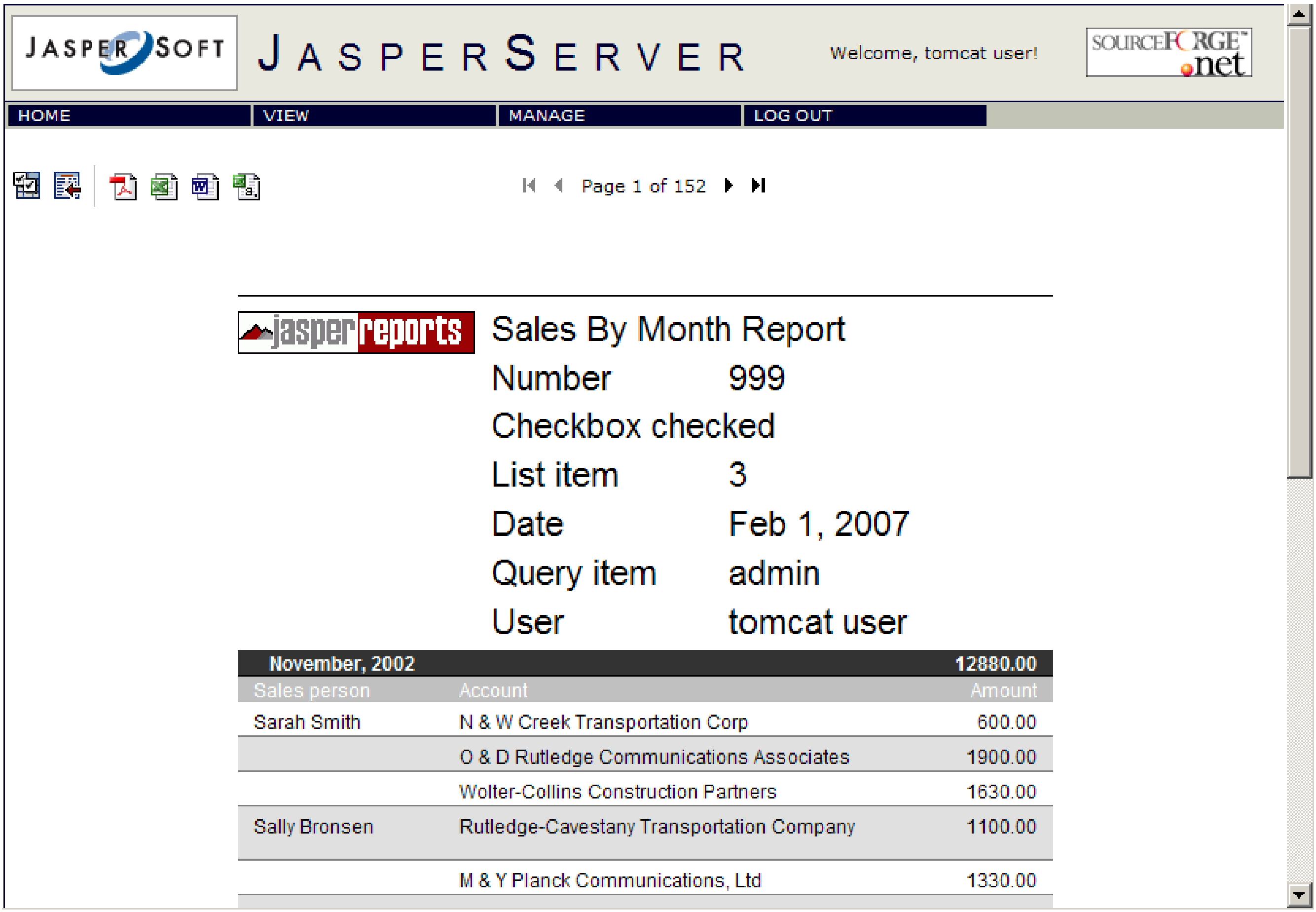
Checkbox Input Control: ☑

List Input Control: Yet another item

DateInput_label: 01-02-2007

QueryInput_label: | Administrator

- Click OK and wait for the report to run. Notice the values in the report title. These are derived from the values you gave for the input controls. Navigate through the report using the left- and right-arrow controls at the top of the report.

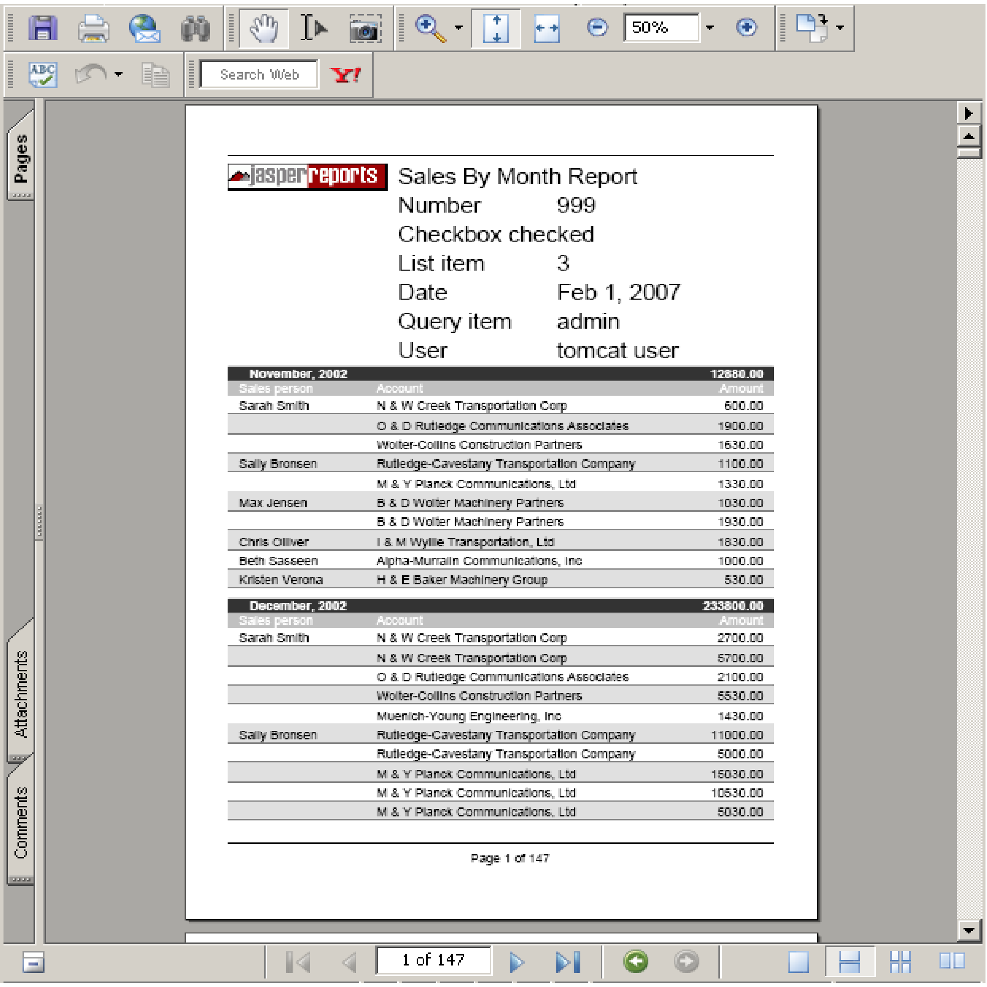


Sales By Month Report

Number	999
Checkbox checked	
List item	3
Date	Feb 1, 2007
Query item	admin
User	tomcat user

November, 2002		12880.00
Sales person	Account	Amount
Sarah Smith	N & W Creek Transportation Corp	600.00
	O & D Rutledge Communications Associates	1900.00
	Wolter-Collins Construction Partners	1630.00
Sally Bronsen	Rutledge-Cavestany Transportation Company	1100.00
	M & Y Planck Communications, Ltd	1330.00

- Click the PDF icon. The report appears in a separate browser in PDF format.
- Click the Close icon (□) to close the report.



Sales By Month Report

Number	999
Checkbox	checked
List item	3
Date	Feb 1, 2007
Query item	admin
User	tomcat user

November, 2002		12880.00
Sales person	Account	Amount
Sarah Smith	N & W Creek Transportation Corp	600.00
	O & D Rutledge Communications Associates	1900.00
	Wolter-Collins Construction Partners	1630.00
Sally Bronsen	Rutledge-Cavestany Transportation Company	1100.00
	M & Y Planck Communications, Ltd	1330.00
Max Jensen	B & D Wolter Machinery Partners	1030.00
	B & D Wolter Machinery Partners	1930.00
Chris Olliver	I & M Wylie Transportation, Ltd	1830.00
Beth Sasseen	Alpha-Murrain Communications, Inc	1000.00
Kristen Verona	H & E Baker Machinery Group	530.00

December, 2002		233800.00
Sales person	Account	Amount
Sarah Smith	N & W Creek Transportation Corp	2700.00
	N & W Creek Transportation Corp	5700.00
	O & D Rutledge Communications Associates	2100.00
	Wolter-Collins Construction Partners	5530.00
	Muenich-Young Engineering, Inc	1430.00
Sally Bronsen	Rutledge-Cavestany Transportation Company	11000.00
	Rutledge-Cavestany Transportation Company	5000.00
	M & Y Planck Communications, Ltd	15030.00
	M & Y Planck Communications, Ltd	10530.00
	M & Y Planck Communications, Ltd	5030.00

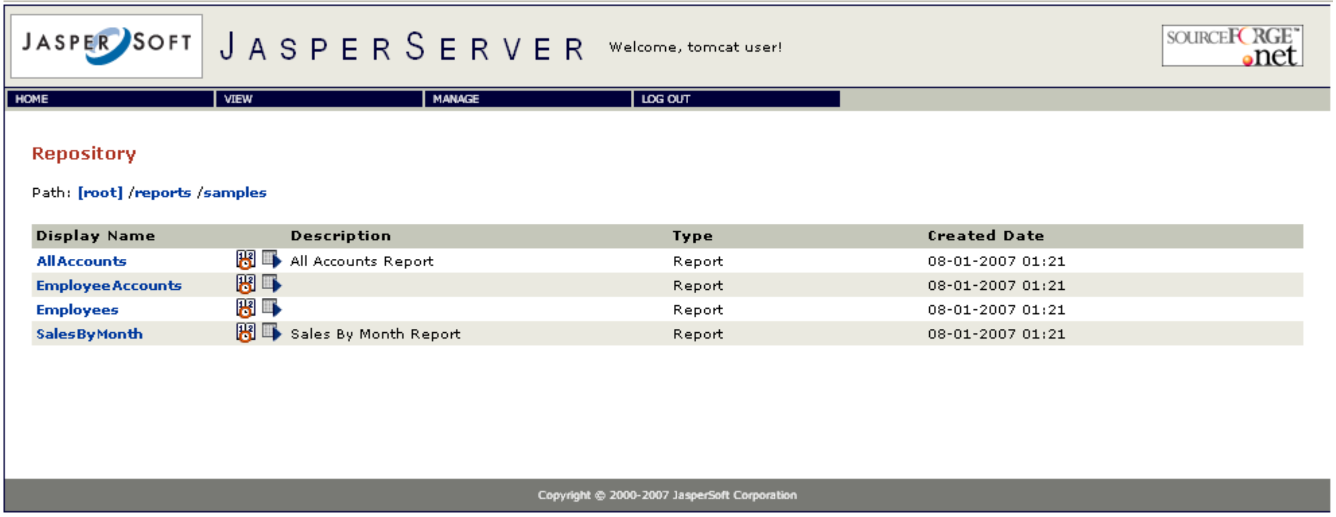
Page 1 of 147

5.2 Using Report Scheduler

The report scheduler runs report execution jobs in the background and saves the execution result in the repository. Users can define such jobs and schedule them to run at specific moments in the future and find the resulting reports in the repository or receive the results by email.

5.2.1 Report browser

Each report is accompanied in the report browser by two icons that provides access to the report shceduling functionality.



Repository

Path: [root] /reports /samples

Display Name		Description	Type	Created Date
AllAccounts	📅 ▶	All Accounts Report	Report	08-01-2007 01:21
EmployeeAccounts	📅 ▶		Report	08-01-2007 01:21
Employees	📅 ▶		Report	08-01-2007 01:21
SalesByMonth	📅 ▶	Sales By Month Report	Report	08-01-2007 01:21

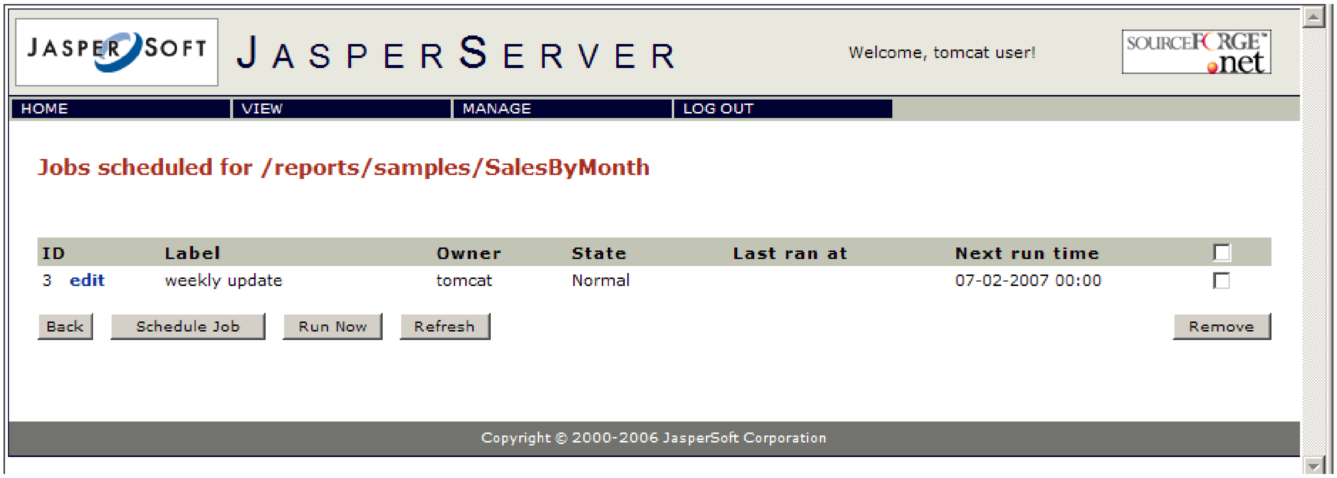
Copyright © 2000-2007 JasperSoft Corporation

The two icons have the following purposes:

- The "Schedule report" (📅) icon open the list of jobs defined for the report.
- The "Run in background" (▶) provides a quick way to run a report in background (asynchronously), i.e. to schedule a report execution job that would get executed immediately. More about this here.

5.2.2 Report jobs list

When the "Schedule report" icon is clicked, the list of report jobs opens.



Jobs scheduled for /reports/samples/SalesByMonth

ID		Label	Owner	State	Last ran at	Next run time	□
3	edit	weekly update	tomcat	Normal		07-02-2007 00:00	□

Copyright © 2000-2006 JasperSoft Corporation

Each regular user only sees his own jobs, while adminstrative user will see all the report jobs.

The following information is displayed in the jobs list:

1. the job ID, accompanied by a link that opens the job for editing.
2. the job label, which is a short description provided while creating the job
3. the job owner, i.e. the user that created the job

4. the job state, which is one of:
    1. Normal: the job is scheduled
    2. Running: the job is currently running
    3. Complete: the job has completed all its executions.
    4. Error: the scheduler encountered an error while scheduling or triggering the job. This does not include the cases when the job is successfuly triggered, but some error occurs during execution.
5. the last time the job was fired
6. the next time the job will get fired

Several controls are placed on the report jobs list:

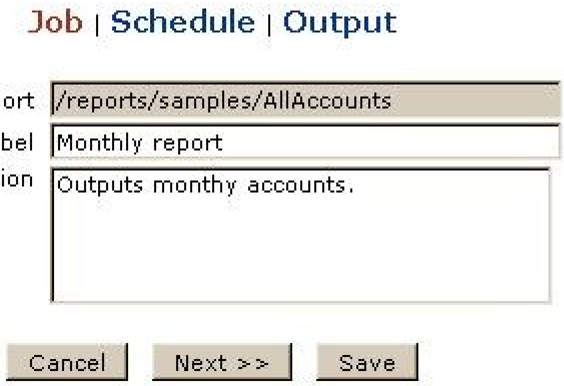
1. The "Back" button is used to return to the report browser screen.
2. The "Schedule Job" button is used to create a new job for the report. Clicking it opens the job details screen.
3. The "Run Now" has the same purpose as the "Run in background" icon on the report browser screen (more details here).
4. The "Refresh" button is used to refetch from the server an updated jobs list.
5. The "Remove" button is used to delete the selected jobs. Jobs are selected via the checkboxes placed on all rows; the checkbox placed on the list header can be used to select/unselect all the jobs.

When a jobs gets deleted, it will be unscheduled but if the job is currenlty running the execution will not be stopped. Completed report jobs, i.e. jobs that will not be fired again, will be automatically deleted from the scheduler.

5.2.3 Report job details

The report job details consist of information required to schedule a report excution:

- schedule information: when should the report be executed
- report parameters
- output: where should the resulting report be sent



Report: /reports/samples/AllAccounts

* Job label: Monthly report

Job description: Outputs monthy accounts.

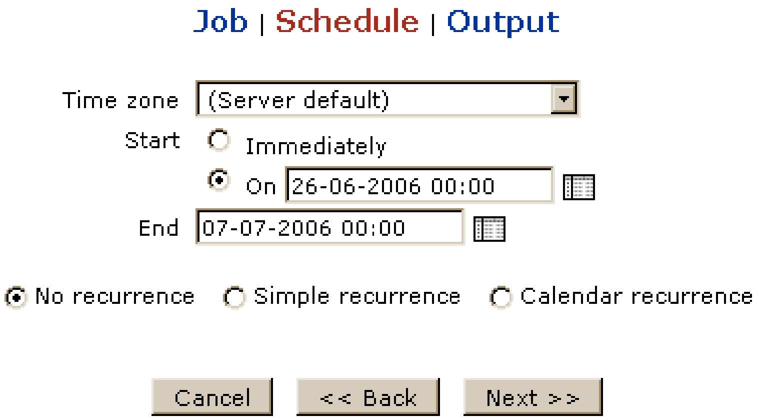
The job details screens are used both for scheduling a new job and for viewing/updating an existing one.

The first screen contains general job description attributes:

- The job label is a short description that will get displayed in the jobs list.
- The job description (optional) is a long job description.

The second screen is used to define job scheduling information. This information consists of job start and end times and from optional recurrence information. The user can set the following scheduling attributes:

- A time zone to use for the scheduling dates. By default, the scheduling dates and times are interpreted in the server time zone, but the user has the option to specify the dates in another time zone and the scheduler will take this into account when firing the job.
- A start date to indicate when the job should become effective. The user has the option of instructing the job to start immediately or at a specific date and time in the future. The start date has slightly different meaning depending on the recurrence type. If no recurrence is defined, the start date will be the date the job is goind to be executed at.
- An end date to indicate when the job should stop firing.



Time zone (Server default)

Start ○ Immediately

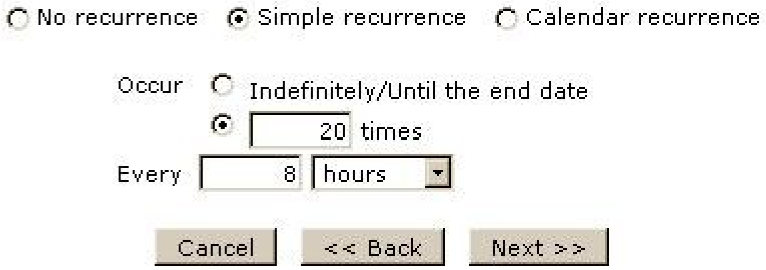
● On 26-06-2006 00:00

End 07-07-2006 00:00

● No recurrence ○ Simple recurrence ○ Calendar recurrence

Apart from this attributes, the user can optionally specify a recurrence scheme for the job. There are two types of recurrence mechanisms that can be used: simple recurrence and calendar recurrence.

The simple recurrence mechanism allows the user to schedule the job to recur at fixed time intervals.



○ No recurrence ● Simple recurrence ○ Calendar recurrence

Occur ○ Indefinitely/Until the end date

● 20 times

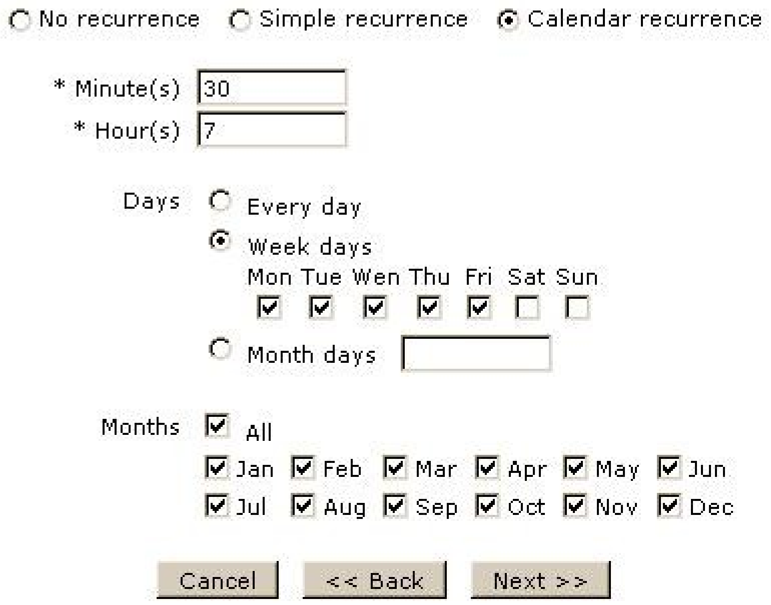
Every 8 hours

The user can set the following attributes:

- Maximum number of occurrences is optional and specifies the maximum number a job can occur. The job would occur until either the occurrence count reaches this value or the end date (if defined) is reached. The user can also choose not to specify a maximum number of occurrences and let the job recur indefinitely or until the end date.
- The fixed recurrence interval, composed of a numerical value and a unit of measure. The user can specify the interval in minutes, hours, days or weeks.

In case of simple recurrence, the job start date will be the date the job is going to be first fire at and the next occurrences will be computed based on the start date and the recurrence interval.

The calendar recurrence mechanism allows users to define jobs that recur at specific calendar moments.



○ No recurrence ○ Simple recurrence ● Calendar recurrence

* Minute(s) 30

* Hour(s) 7

Days ○ Every day

● Week days

Mon	Tue	Wen	Thu	Fri	Sat	Sun
☑	☑	☑	☑	☑	□	□

○ Month days

Months ☑ All

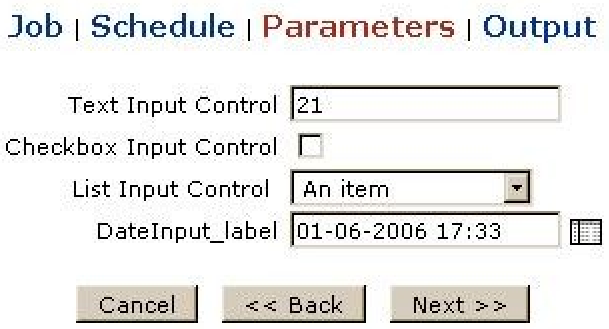
☑ Jan ☑ Feb ☑ Mar ☑ Apr ☑ May ☑ Jun

☑ Jul ☑ Aug ☑ Sep ☑ Oct ☑ Nov ☑ Dec

The user needs to specify the minutes, hours, days and months the job should occur at:

- One can specify one or more minutes the job should be executed at. The accepted values are in the 0 to 59 range. In the most common scenarios, only one value would be set for the minutes. If the report needs to be executed multiple times per hour, the minutes can be specified using the following syntax:
  - X,Y,Z,... – the job will be executed at minutes X, Y, Z, ..
  - X-Y – the job will be executed every minute from X to Y
  - * - the job will be executed every minute
- The hour(s) field is similar to the minutes, with the difference that the accepted values are in the 0 to 23 range.
- To select the days the job should run on, the user has three options:
  - Set the job to run every day.
  - Set the job to run on specific week days.
  - Set the job to run on specific month days. The month days field is similar to the minutes field, with accepted values in the 1 to 31 range.
- The months the job should be executed on are selected individually. A checkbox to select/deselect all months is available for convenience.

If the user chooses calendar recurrence, the job will first be executed at the first moment after the start date that



Text Input Control 21

Checkbox Input Control □

List Input Control An item

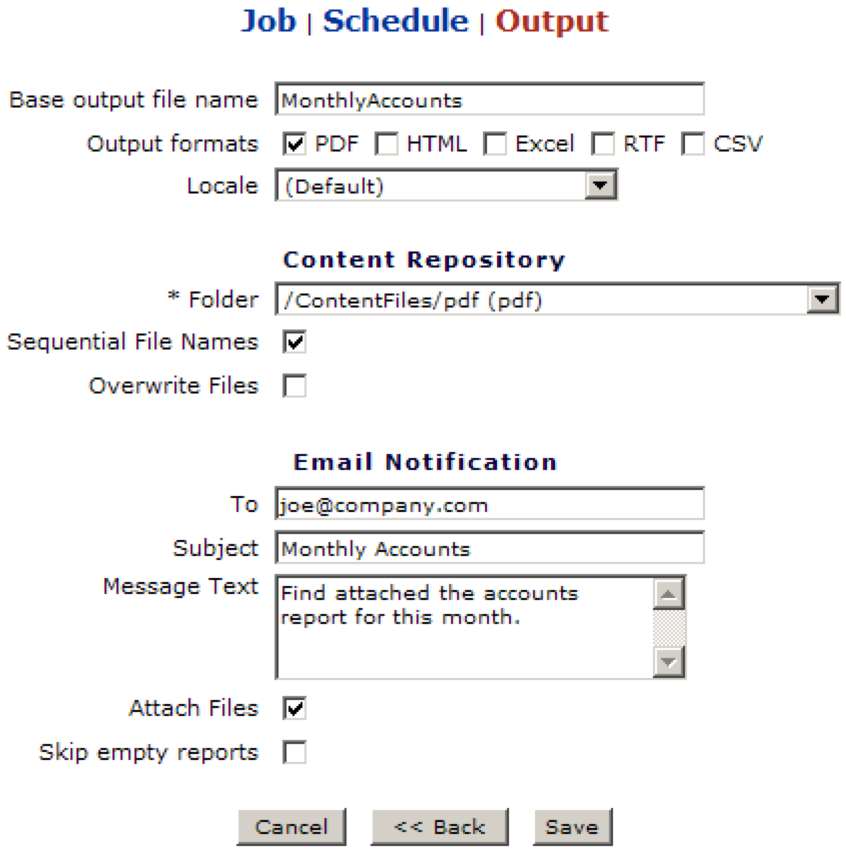
DateInput_label 01-06-2006 17:33

matches all the defined date and time values. The job would then recur at each matching date and time until the end date is reached or indefinitely if an end date is not set.

If the report has input controls, the job details will contain a set of input values which will be used when executing the report. A separate screen, identical to the one used when directly executing the report, will allow the user to input value for the report parameters.

Finally, the user has to define the job output attributes. Each time the job runs, the results are saved into the content repository. Optionally, the result can also be distributed by email to one or more recipients.

The following attributes can be set in the job output screen:



Base output file name: MonthlyAccounts

Output formats: ☑ PDF □ HTML □ Excel □ RTF □ CSV

Locale: (Default)

Content Repository

* Folder: /ContentFiles/pdf (pdf)

Sequential File Names: ☑

Overwrite Files: □

Email Notification

To: joe@company.com

Subject: Monthly Accounts

Message Text: Find attached the accounts report for this month.

Attach Files: ☑

Skip empty reports: □

- The base output filename is used to derive all output filenames by appending an optional timestamp and a format extension.
- The user needs to choose at least one output format. The available list includes PDF, HTML, Microsoft Excel, RTF and CSV.
- The user can choose a locale that should be used when filling the report. If no specific locale is selected, the server will use the default locale.
- The content repository folder where all the resulting files should be saved has to be selected.
- The user can choose to include a timestamp in the file names used to save the job results. This is useful for recurring jobs when multiple results need to be kept.
- The user can instruct the scheduler whether overwriting existing content files is allowed. If the scheduler wants to save a report and another saved report having the same name already exists, it will either overwrite it or generate an error, depending on this flag.
- In addition to saving the result in the content repository, the job creator can choose to enable an email notification for the job. Each time the job would get executed, an email would be sent to one or more

recipients that would optionally include the job results as attachments.

A user needs to specify the following attributes for the email notification:

- One or more recipient addresses, separated by comma.
- The message subject.
- The message body.
- Whether the job results should be attached to the message.
- Whether the mail should not be sent when the generated is empty.

As we mentioned before, the report job details screens are used for both creating a new job and for updating an existing job. The users should note that if the scheduling attributes of a job are updated, the scheduler will recreate the trigger that fires the job execution. This might have unexpected results if the job start date is in the past and should generally be avoided. If other job attributes are updated, they will take effect from the next job execution.

5.2.4 Quick background execution

The "Run in background" report execution allows users to define a report job that would get fired immediately without manually entering all the job attributes. This functionality can be used to launch asynchronously long report executions and save the results in the repository (and optionally receive the results via email).

If the user wants to quickly launch a report execution in background, he will only be required to enter the report parameters if the report has input controls and to define the job output. The screens used for these tasks are the same as the ones used when scheduling a new report job.

5.2.5 Built-in report parameters

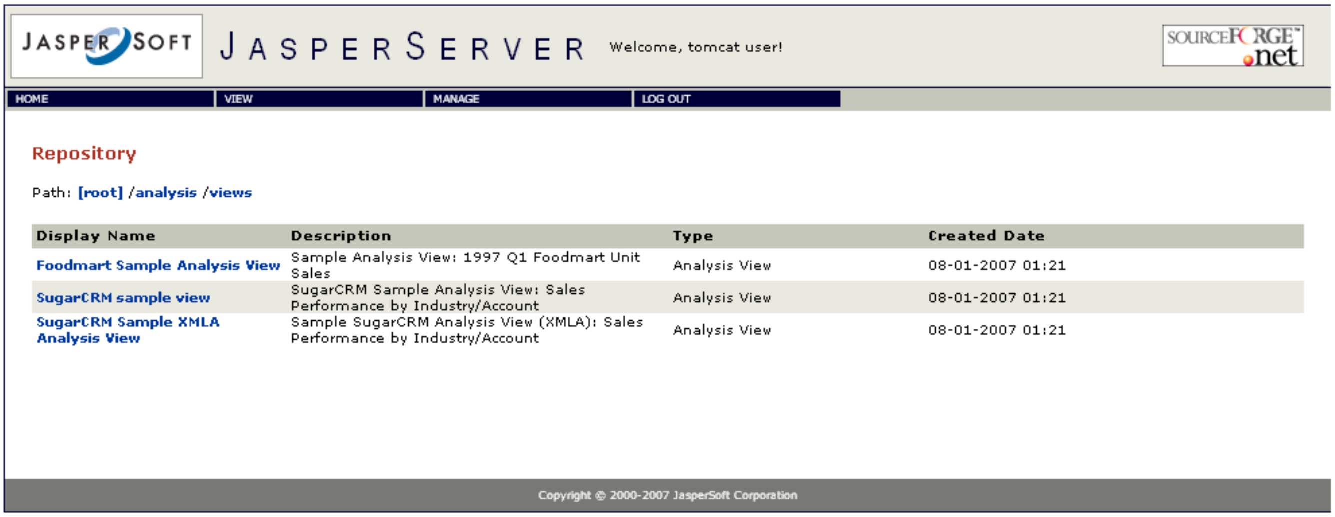
When a report is executed by the report scheduler, a built-in parameter named "_ScheduledTime" is used to pass to the report the date and time the report was scheduled at. The parameter needs to be declared in the JRXML in order to be used:

```
<parameter name="_ScheduledTime" class="java.util.Date"/>
```

The value can be used to filter the reported date or for displaying purposes.

5.3 Interacting with Analysis Views

An analysis view consists of connection resources and an MDX query and. The current version contains a number of sample analysis views. To browse an analysis view, mouse on the VIEW menu and click on ANALYSIS VIEWS, the following view list displays.



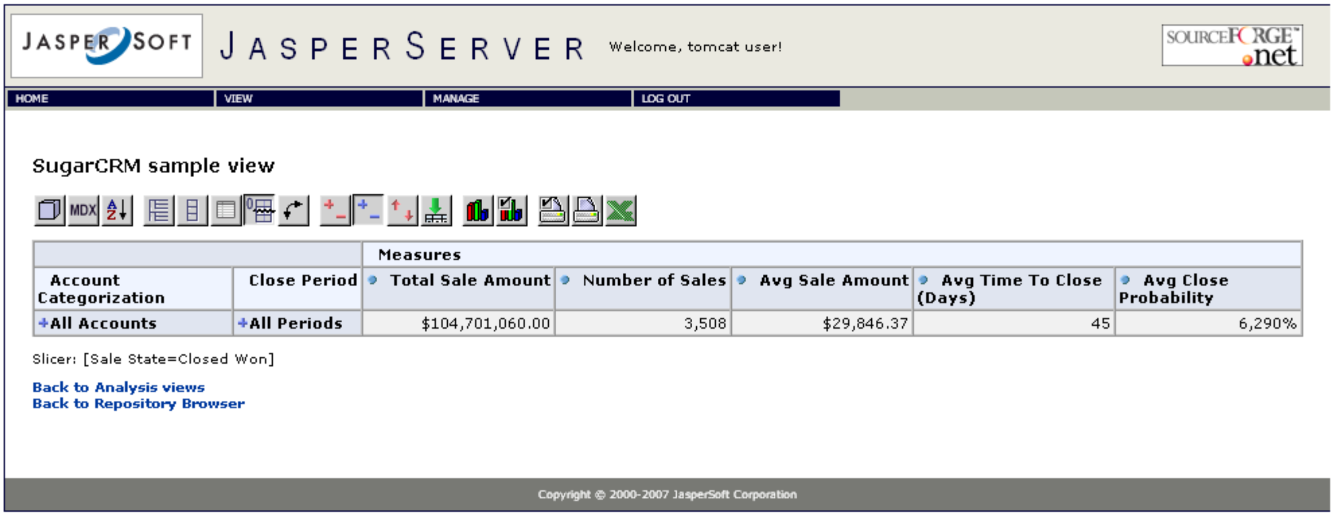
Repository

Path: [root] /analysis /views

Display Name	Description	Type	Created Date
Foodmart Sample Analysis View	Sample Analysis View: 1997 Q1 Foodmart Unit Sales	Analysis View	08-01-2007 01:21
SugarCRM sample view	SugarCRM Sample Analysis View: Sales Performance by Industry/Account	Analysis View	08-01-2007 01:21
SugarCRM Sample XMLA Analysis View	Sample SugarCRM Analysis View (XMLA): Sales Performance by Industry/Account	Analysis View	08-01-2007 01:21

Copyright © 2000-2007 JasperSoft Corporation

To browse an analysis view, click the view name under the 'Analysis View Name column', e.g., 'SugarCRM_sample_unit_2', and the following page displays.



Welcome, tomcat user!

SugarCRM sample view

		Measures				
Account Categorization	Close Period	Total Sale Amount	Number of Sales	Avg Sale Amount	Avg Time To Close (Days)	Avg Close Probability
+All Accounts	+All Periods	$104,701,060.00	3,508	$29,846.37	45	6,290%

Slicer: [Sale State=Closed Won]

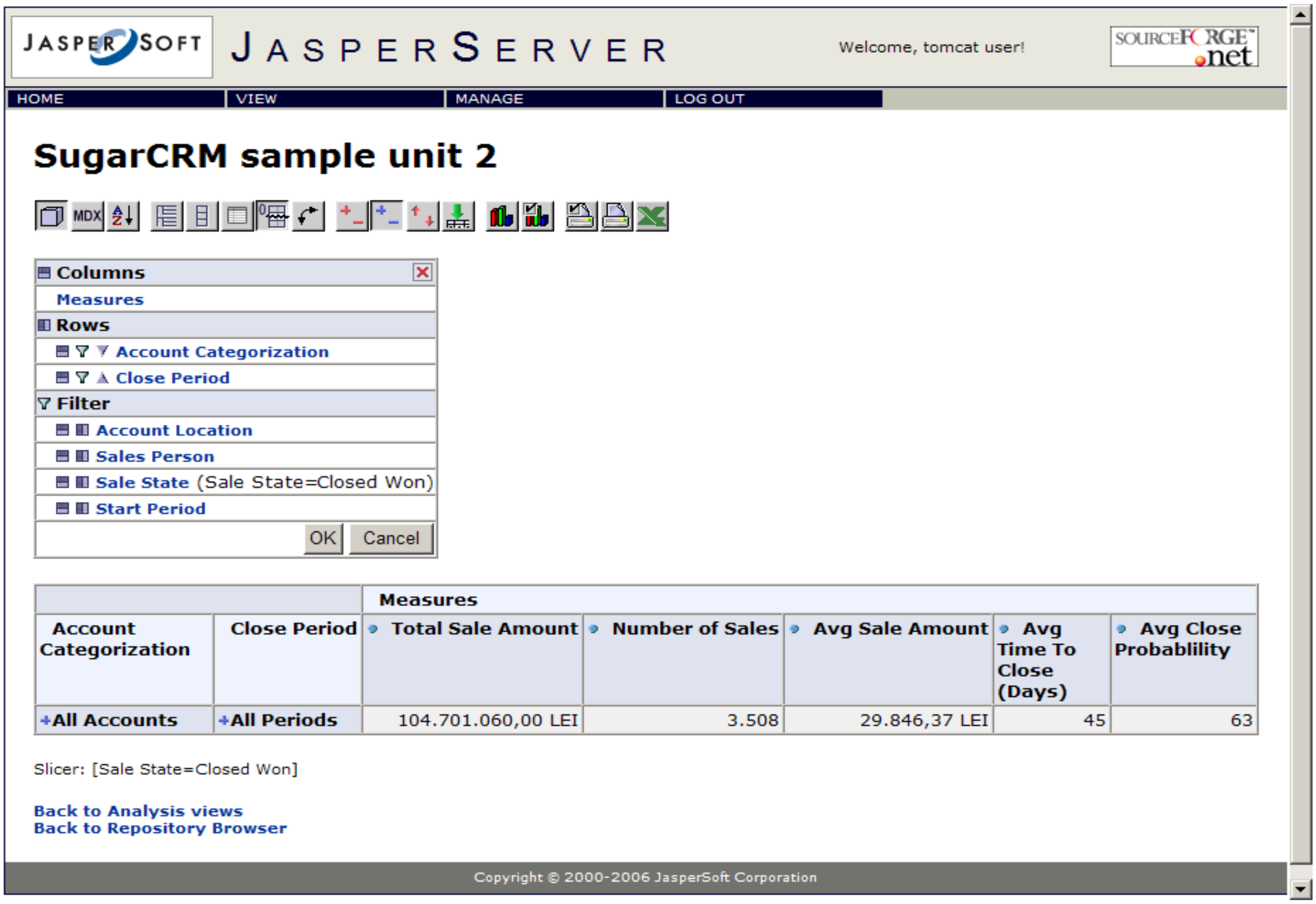
Back to Analysis views
Back to Repository Browser

Copyright © 2000-2007 JasperSoft Corporation

Currently, JasperAnalysis utilizes JPivot for analytical processing. The following is a description of the analytical operations.

5.3.1 Open OLAP navigator

This operation () allows a user to change an analysis view and define dimension filters. Click the Open OLAP Navigator to display the following properties.



Welcome, tomcat user!

SugarCRM sample unit 2

Columns
Measures
Rows
Account Categorization
Close Period
Filter
Account Location
Sales Person
Sale State (Sale State=Closed Won)
Start Period

		Measures				
Account Categorization	Close Period	Total Sale Amount	Number of Sales	Avg Sale Amount	Avg Time To Close (Days)	Avg Close Probablility
+All Accounts	+All Periods	104.701.060,00 LEI	3.508	29.846,37 LEI	45	63

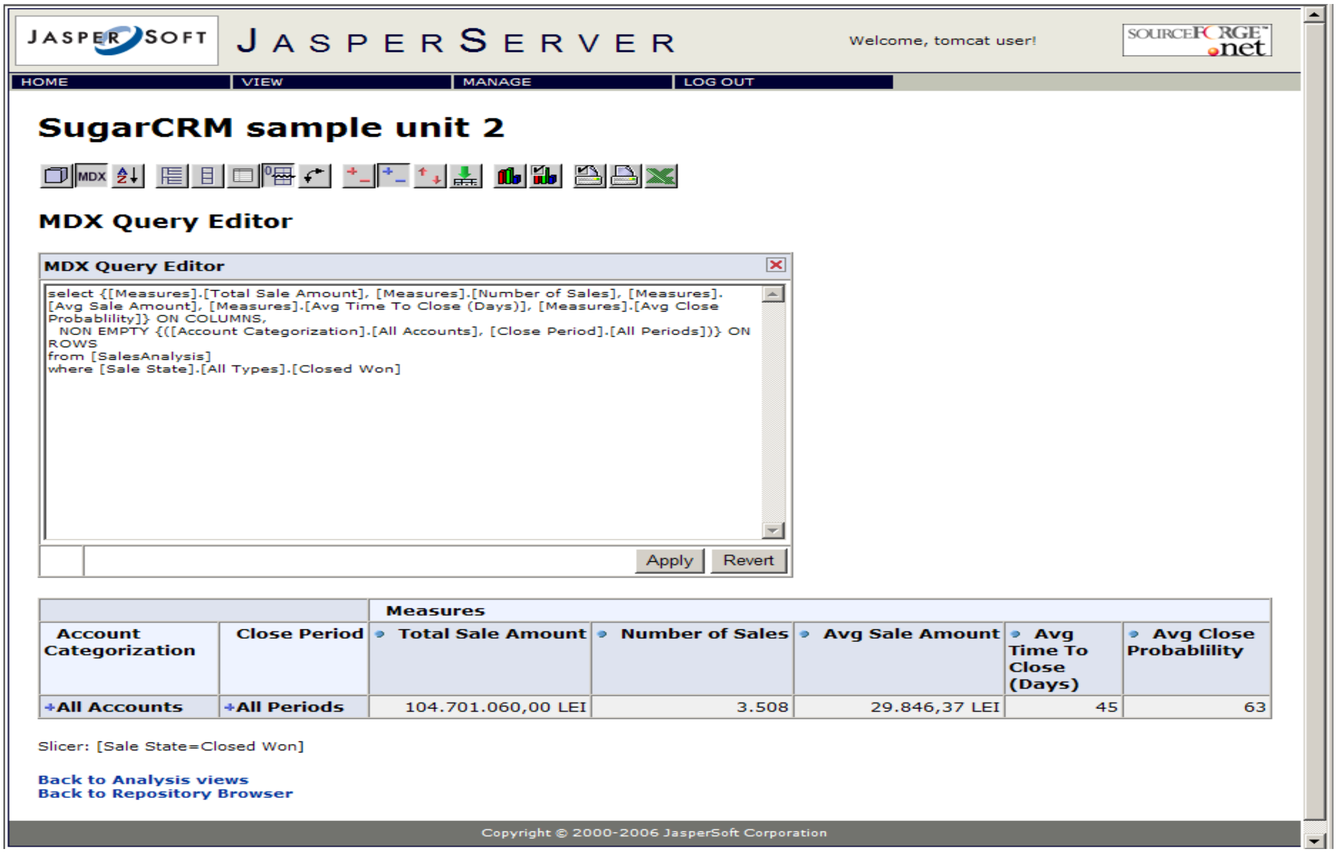
Slicer: [Sale State=Closed Won]

Back to Analysis views
Back to Repository Browser

Copyright © 2000-2006 JasperSoft Corporation

5.3.2 Show MDX editor

This operation (MDX) allows a user to change MDX query.



SugarCRM sample unit 2

MDX Query Editor

MDX Query Editor

```
select {[Measures].[Total Sale Amount], [Measures].[Number of Sales], [Measures].
[Avg Sale Amount], [Measures].[Avg Time To Close (Days)], [Measures].[Avg Close
Probablility]} ON COLUMNS,
  NON EMPTY {([Account Categorization].[All Accounts], [Close Period].[All Periods])} ON
ROWS
from [SalesAnalysis]
where [Sale State].[All Types].[Closed Won]
```

		Measures				
Account Categorization	Close Period	Total Sale Amount	Number of Sales	Avg Sale Amount	Avg Time To Close (Days)	Avg Close Probablility
+All Accounts	+All Periods	104.701.060,00 LEI	3.508	29.846,37 LEI	45	63

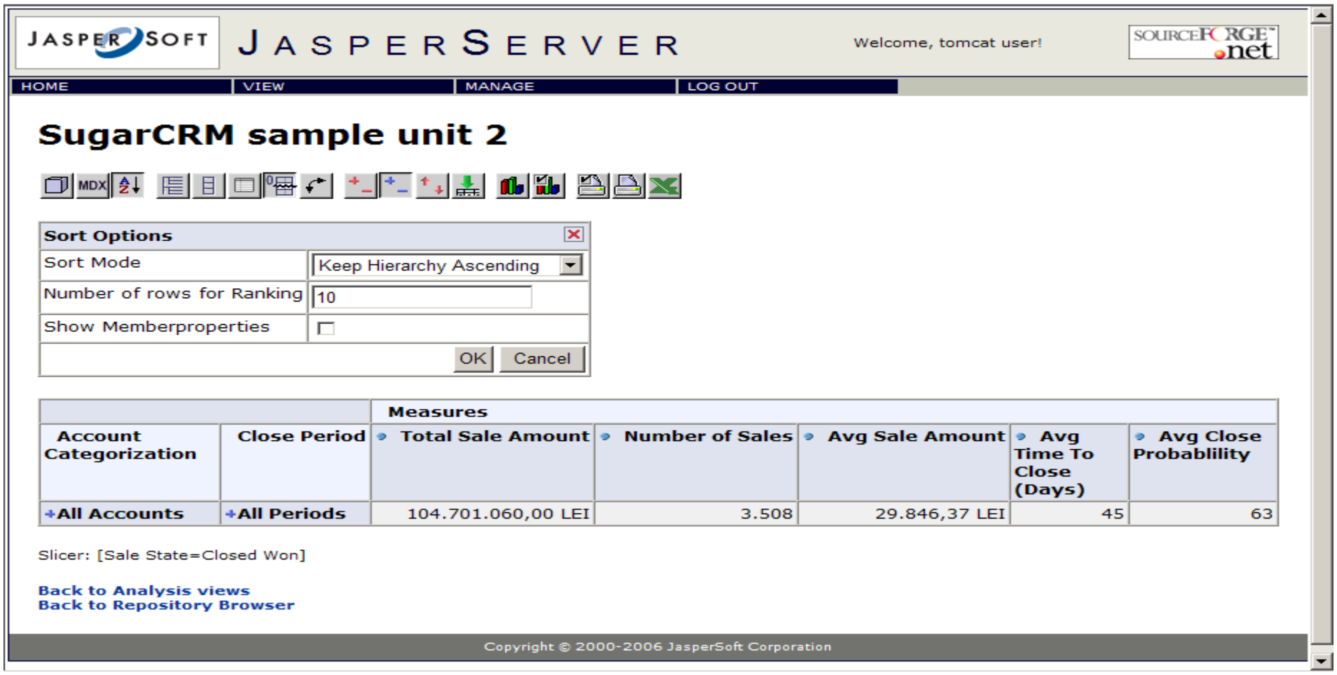
Slicer: [Sale State=Closed Won]

Back to Analysis views
Back to Repository Browser

Copyright © 2000-2006 JasperSoft Corporation

5.3.3 Config OLAP table

This operation (AZ↓) allows a user to change the layout of the information displayed in a table.



SugarCRM sample unit 2

Sort Options

Sort Mode	Keep Hierarchy Ascending
Number of rows for Ranking	10
Show Memberproperties	□

		Measures				
Account Categorization	Close Period	Total Sale Amount	Number of Sales	Avg Sale Amount	Avg Time To Close (Days)	Avg Close Probablility
+All Accounts	+All Periods	104.701.060,00 LEI	3.508	29.846,37 LEI	45	63

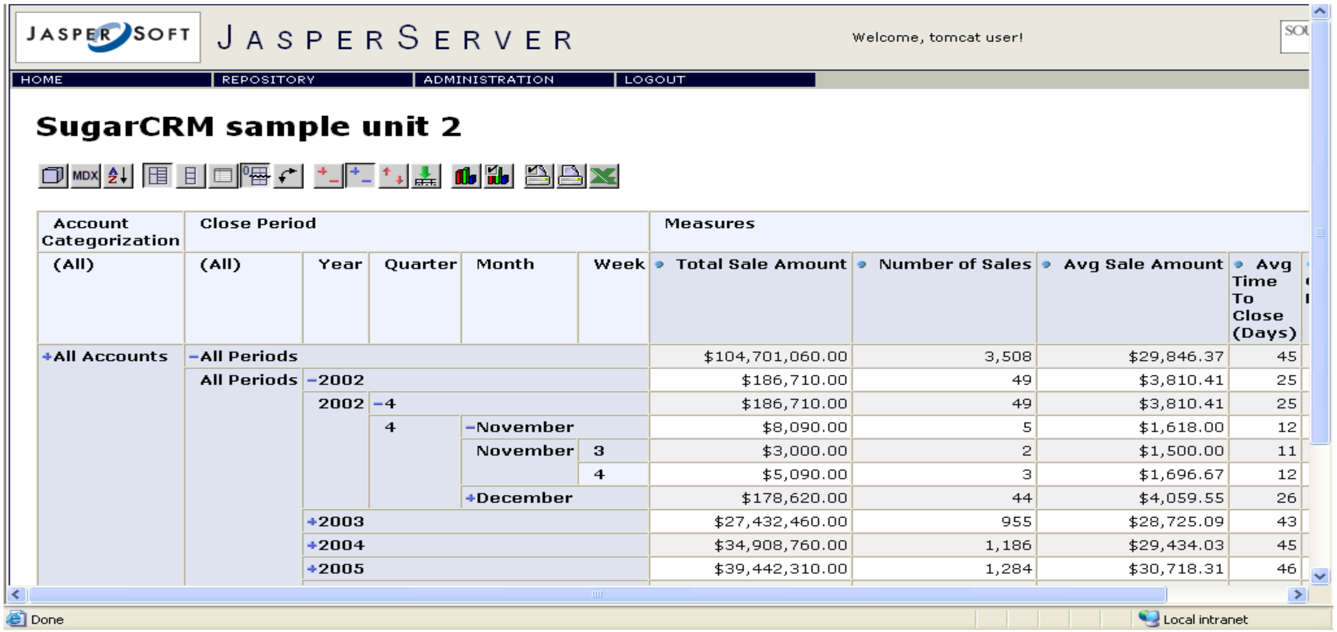
Slicer: [Sale State=Closed Won]

Back to Analysis views
Back to Repository Browser

Copyright © 2000-2006 JasperSoft Corporation

5.3.4 Show/Hide parent members

This operation ( ) displays or removes column labes of the parent dimensions. Below is an example of displaying parent dimension as separate columns for the *Close Period*.

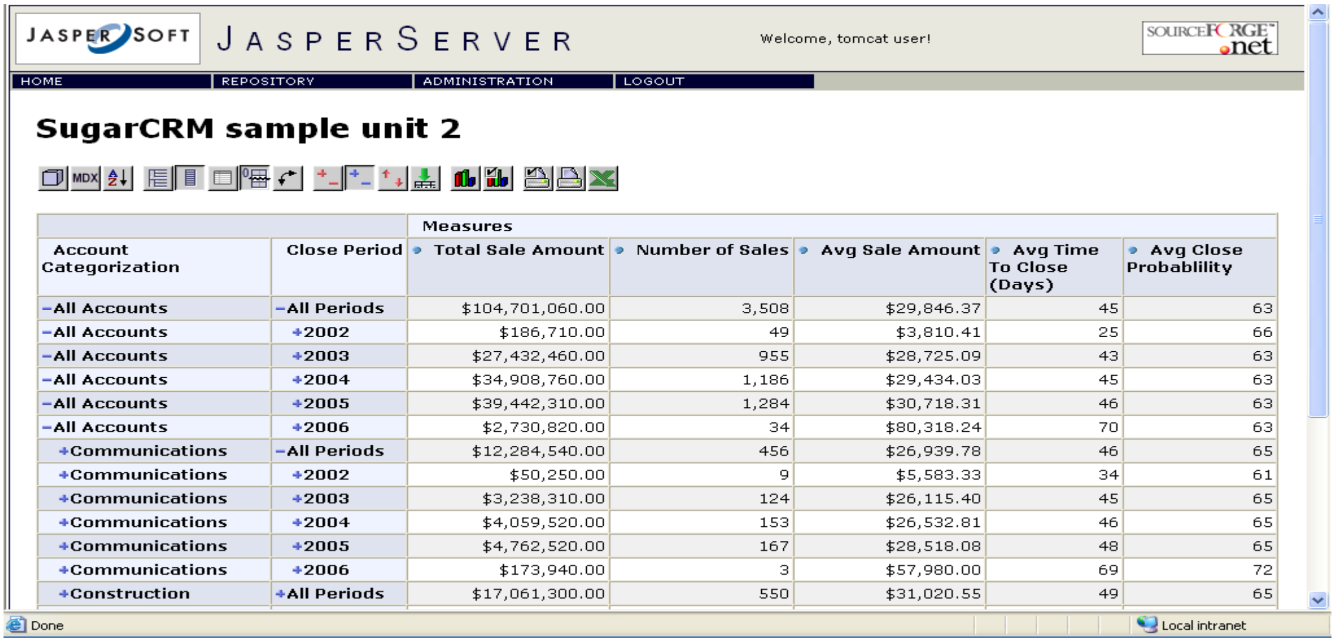


SugarCRM sample unit 2

Account Categorization	Close Period					Measures			
(All)	(All)	Year	Quarter	Month	Week	• Total Sale Amount	• Number of Sales	• Avg Sale Amount	• Avg Time To Close (Days)
+All Accounts	−All Periods					$104,701,060.00	3,508	$29,846.37	45
	All Periods	−2002				$186,710.00	49	$3,810.41	25
		2002	−4			$186,710.00	49	$3,810.41	25
			4	−November		$8,090.00	5	$1,618.00	12
				November	3	$3,000.00	2	$1,500.00	11
					4	$5,090.00	3	$1,696.67	12
				+December		$178,620.00	44	$4,059.55	26
		+2003				$27,432,460.00	955	$28,725.09	43
		+2004				$34,908,760.00	1,186	$29,434.03	45
		+2005				$39,442,310.00	1,284	$30,718.31	46

5.3.5 Show/Hide spans

This operation ( ) exposes or conceals redundant information in the parent dimension columns. Below is an example of showing duplicated attribute values for *Account Categorization*.

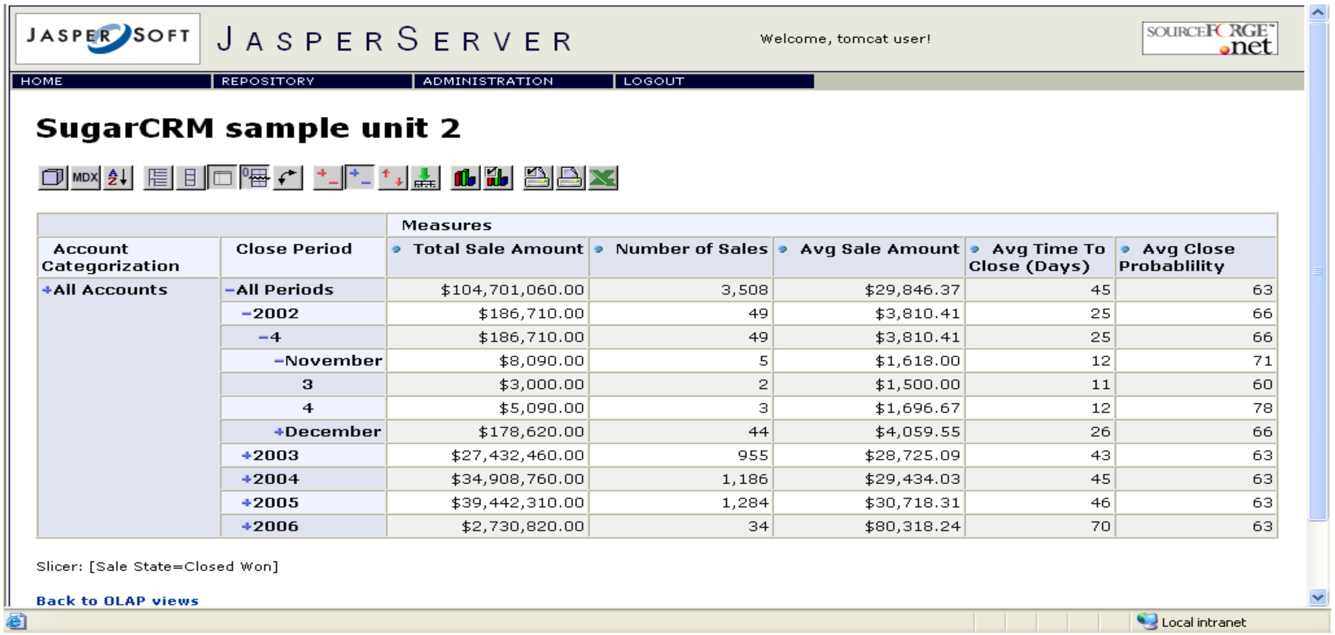


SugarCRM sample unit 2

		Measures				
Account Categorization	Close Period	• Total Sale Amount	• Number of Sales	• Avg Sale Amount	• Avg Time To Close (Days)	• Avg Close Probablility
−All Accounts	−All Periods	$104,701,060.00	3,508	$29,846.37	45	63
−All Accounts	+2002	$186,710.00	49	$3,810.41	25	66
−All Accounts	+2003	$27,432,460.00	955	$28,725.09	43	63
−All Accounts	+2004	$34,908,760.00	1,186	$29,434.03	45	63
−All Accounts	+2005	$39,442,310.00	1,284	$30,718.31	46	63
−All Accounts	+2006	$2,730,820.00	34	$80,318.24	70	63
+Communications	−All Periods	$12,284,540.00	456	$26,939.78	46	65
+Communications	+2002	$50,250.00	9	$5,583.33	34	61
+Communications	+2003	$3,238,310.00	124	$26,115.40	45	65
+Communications	+2004	$4,059,520.00	153	$26,532.81	46	65
+Communications	+2005	$4,762,520.00	167	$28,518.08	48	65
+Communications	+2006	$173,940.00	3	$57,980.00	69	72
+Construction	+All Periods	$17,061,300.00	550	$31,020.55	49	65

5.3.6 Show/Hide properties

This operation ( ) displays or removes additional information (if any) in the fact data defined by the same dimension. TBD



SugarCRM sample unit 2

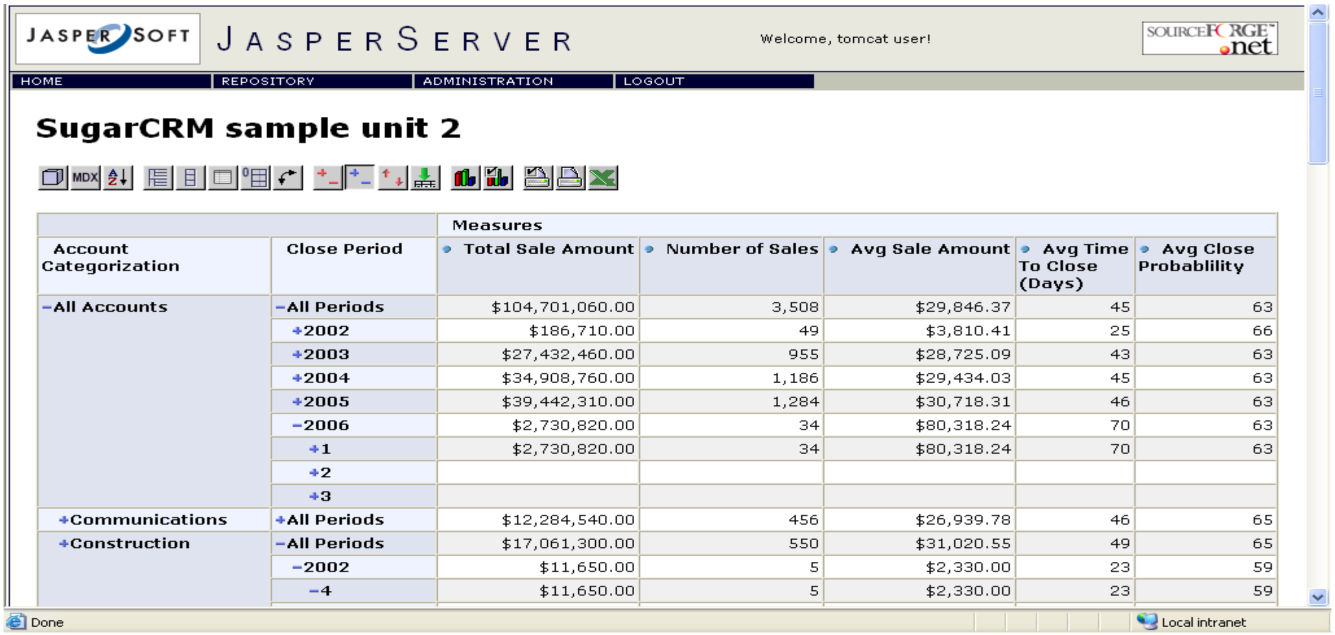
		Measures				
Account Categorization	Close Period	Total Sale Amount	Number of Sales	Avg Sale Amount	Avg Time To Close (Days)	Avg Close Probablility
+All Accounts	-All Periods	$104,701,060.00	3,508	$29,846.37	45	63
	-2002	$186,710.00	49	$3,810.41	25	66
	-4	$186,710.00	49	$3,810.41	25	66
	-November	$8,090.00	5	$1,618.00	12	71
	3	$3,000.00	2	$1,500.00	11	60
	4	$5,090.00	3	$1,696.67	12	78
	+December	$178,620.00	44	$4,059.55	26	66
	+2003	$27,432,460.00	955	$28,725.09	43	63
	+2004	$34,908,760.00	1,186	$29,434.03	45	63
	+2005	$39,442,310.00	1,284	$30,718.31	46	63
	+2006	$2,730,820.00	34	$80,318.24	70	63

Slicer: [Sale State=Closed Won]

Back to OLAP views

5.3.7 Surpress/Reveal empty rows/columns

This operation () suppresses or reveals rows and/or columns that does not have relevant fact data with respect to the dimensions. Below is example that reveals the empty rows for year 2006.

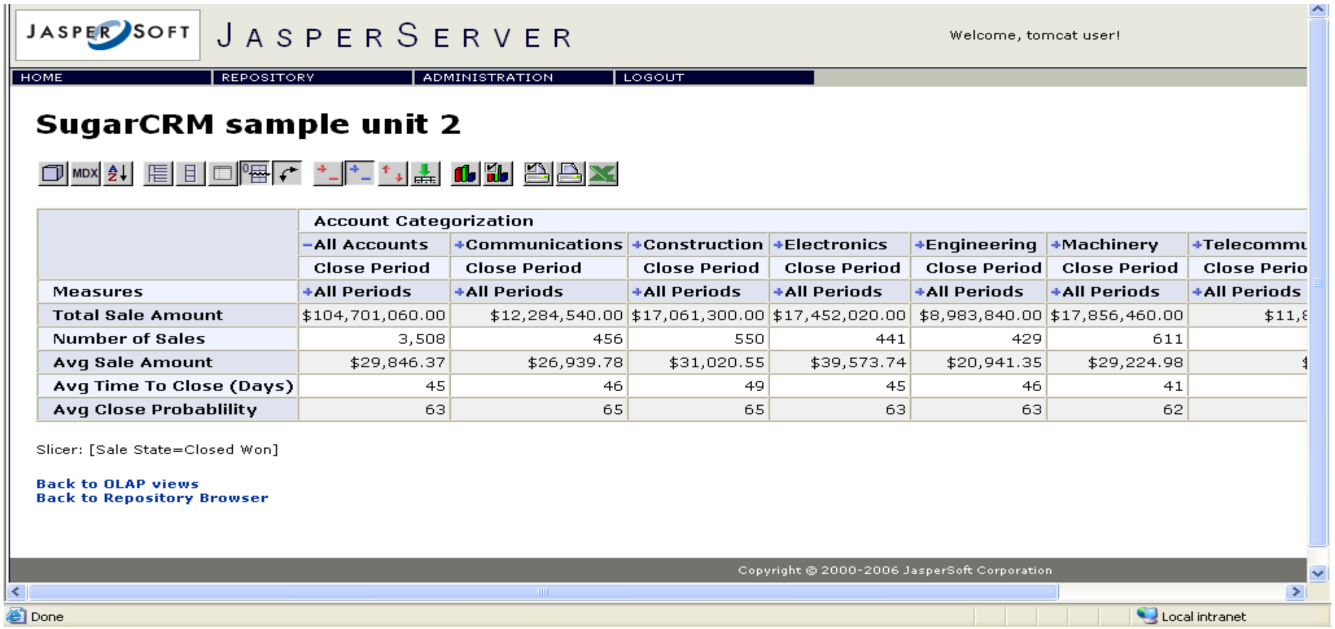


SugarCRM sample unit 2

		Measures				
Account Categorization	Close Period	Total Sale Amount	Number of Sales	Avg Sale Amount	Avg Time To Close (Days)	Avg Close Probablility
-All Accounts	-All Periods	$104,701,060.00	3,508	$29,846.37	45	63
	+2002	$186,710.00	49	$3,810.41	25	66
	+2003	$27,432,460.00	955	$28,725.09	43	63
	+2004	$34,908,760.00	1,186	$29,434.03	45	63
	+2005	$39,442,310.00	1,284	$30,718.31	46	63
	-2006	$2,730,820.00	34	$80,318.24	70	63
	+1	$2,730,820.00	34	$80,318.24	70	63
	+2					
	+3					
+Communications	+All Periods	$12,284,540.00	456	$26,939.78	46	65
+Construction	-All Periods	$17,061,300.00	550	$31,020.55	49	65
	-2002	$11,650.00	5	$2,330.00	23	59
	-4	$11,650.00	5	$2,330.00	23	59

5.3.8 Swap axes

This operation () changes the orientation of a table by interchanging the columns with the rows. Below is an example of swap axes that interchanges measure columns with dimension rows.



SugarCRM sample unit 2

	Account Categorization						
	−All Accounts	+Communications	+Construction	+Electronics	+Engineering	+Machinery	+Telecommu
	Close Period	Close Period	Close Period	Close Period	Close Period	Close Period	Close Perio
Measures	+All Periods	+All Periods	+All Periods	+All Periods	+All Periods	+All Periods	+All Periods
Total Sale Amount	$104,701,060.00	$12,284,540.00	$17,061,300.00	$17,452,020.00	$8,983,840.00	$17,856,460.00	$11,
Number of Sales	3,508	456	550	441	429	611	
Avg Sale Amount	$29,846.37	$26,939.78	$31,020.55	$39,573.74	$20,941.35	$29,224.98	$
Avg Time To Close (Days)	45	46	49	45	46	41	
Avg Close Probablility	63	65	65	63	63	62	

Slicer: [Sale State=Closed Won]

Back to OLAP views
Back to Repository Browser

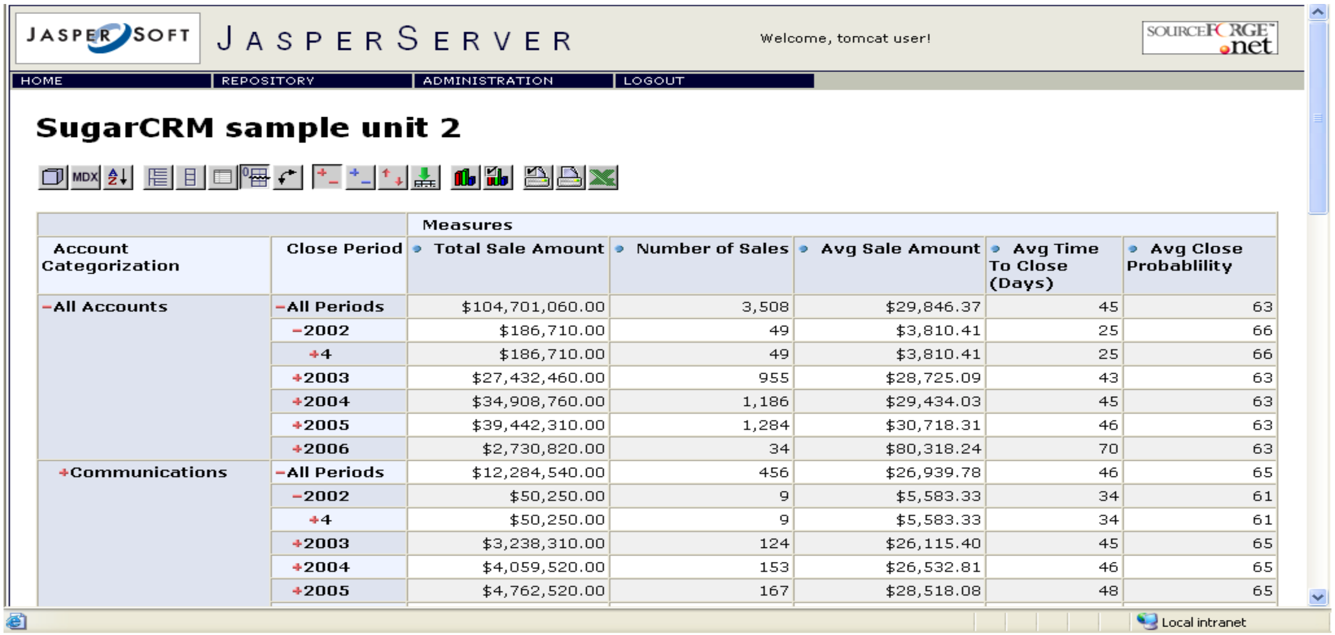
Copyright © 2000-2006 JasperSoft Corporation

Done

Local intranet

5.3.9 Drill member

This operation (+ −) synchronizes the contraction or expansion of rows across all dimension members. Below is an example of drill member at *Close Period* for dimension member *Account Categorization*.



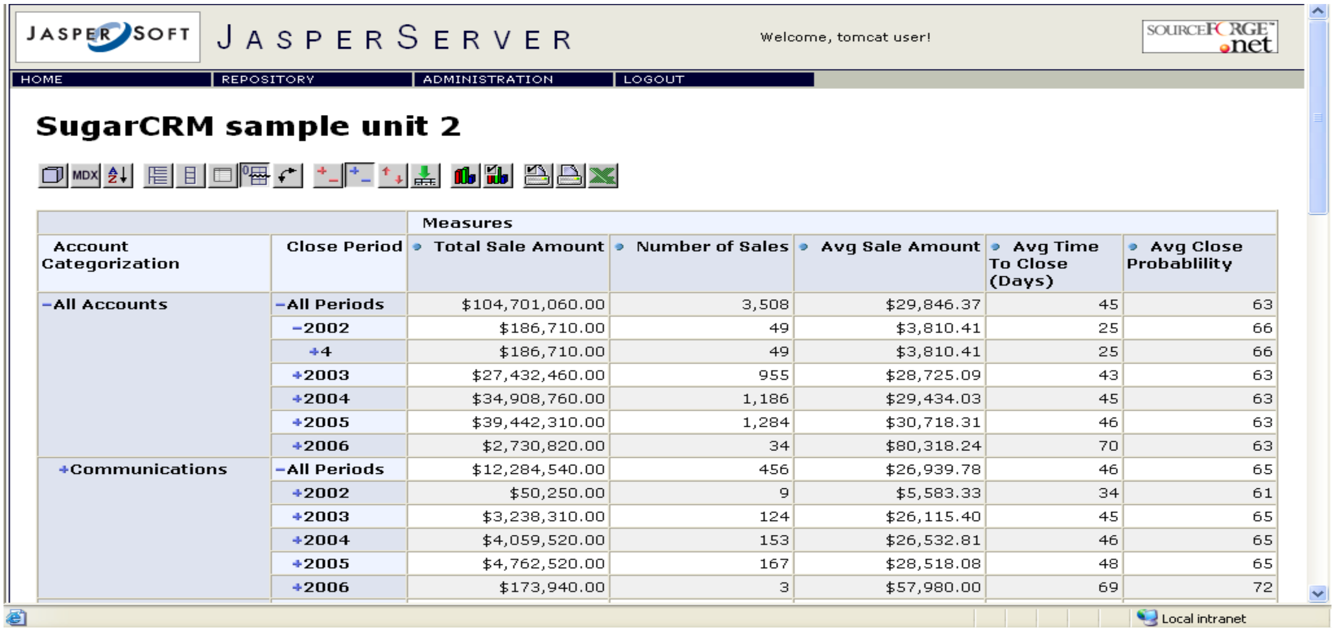
SugarCRM sample unit 2

		Measures				
Account Categorization	Close Period	● Total Sale Amount	● Number of Sales	● Avg Sale Amount	● Avg Time To Close (Days)	● Avg Close Probablility
−All Accounts	−All Periods	$104,701,060.00	3,508	$29,846.37	45	63
	−2002	$186,710.00	49	$3,810.41	25	66
	+4	$186,710.00	49	$3,810.41	25	66
	+2003	$27,432,460.00	955	$28,725.09	43	63
	+2004	$34,908,760.00	1,186	$29,434.03	45	63
	+2005	$39,442,310.00	1,284	$30,718.31	46	63
	+2006	$2,730,820.00	34	$80,318.24	70	63
+Communications	−All Periods	$12,284,540.00	456	$26,939.78	46	65
	−2002	$50,250.00	9	$5,583.33	34	61
	+4	$50,250.00	9	$5,583.33	34	61
	+2003	$3,238,310.00	124	$26,115.40	45	65
	+2004	$4,059,520.00	153	$26,532.81	46	65
	+2005	$4,762,520.00	167	$28,518.08	48	65

Local intranet

5.3.10 Drill position

This operation (+ −) collapses or expands rows at a specific dimension member.

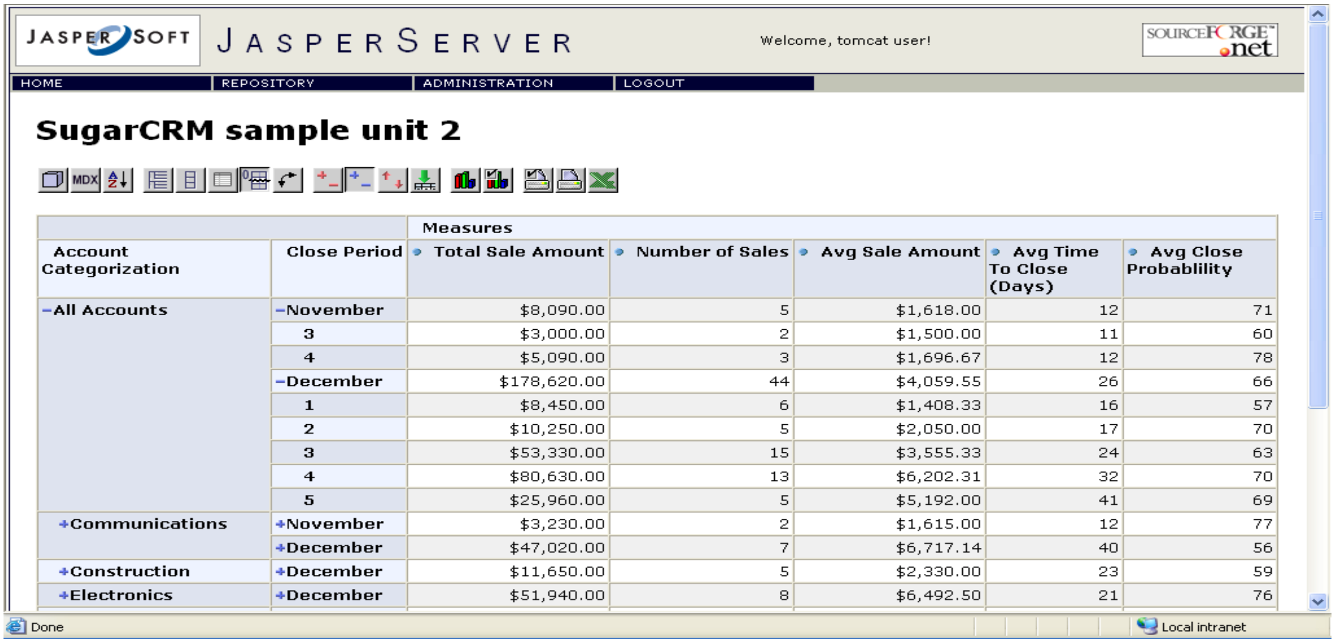


SugarCRM sample unit 2

		Measures				
Account Categorization	Close Period	Total Sale Amount	Number of Sales	Avg Sale Amount	Avg Time To Close (Days)	Avg Close Probablility
-All Accounts	-All Periods	$104,701,060.00	3,508	$29,846.37	45	63
	-2002	$186,710.00	49	$3,810.41	25	66
	+4	$186,710.00	49	$3,810.41	25	66
	+2003	$27,432,460.00	955	$28,725.09	43	63
	+2004	$34,908,760.00	1,186	$29,434.03	45	63
	+2005	$39,442,310.00	1,284	$30,718.31	46	63
	+2006	$2,730,820.00	34	$80,318.24	70	63
+Communications	-All Periods	$12,284,540.00	456	$26,939.78	46	65
	+2002	$50,250.00	9	$5,583.33	34	61
	+2003	$3,238,310.00	124	$26,115.40	45	65
	+2004	$4,059,520.00	153	$26,532.81	46	65
	+2005	$4,762,520.00	167	$28,518.08	48	65
	+2006	$173,940.00	3	$57,980.00	69	72

5.3.11 Drill replace

This operation () replaces the current table with a sub-table determined by the location of the drill replace operation. Below is an example of drill replace at *Close Period* 4 of *Year 2002*.

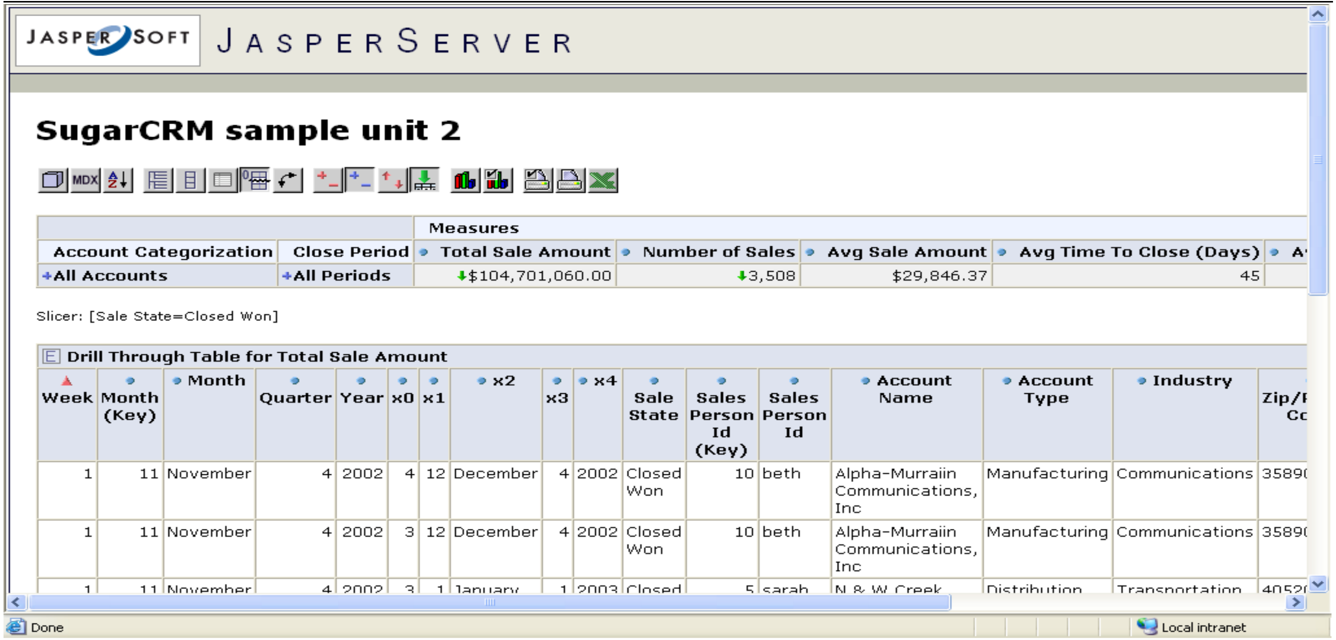


SugarCRM sample unit 2

		Measures				
Account Categorization	Close Period	Total Sale Amount	Number of Sales	Avg Sale Amount	Avg Time To Close (Days)	Avg Close Probablility
-All Accounts	-November	$8,090.00	5	$1,618.00	12	71
	3	$3,000.00	2	$1,500.00	11	60
	4	$5,090.00	3	$1,696.67	12	78
	-December	$178,620.00	44	$4,059.55	26	66
	1	$8,450.00	6	$1,408.33	16	57
	2	$10,250.00	5	$2,050.00	17	70
	3	$53,330.00	15	$3,555.33	24	63
	4	$80,630.00	13	$6,202.31	32	70
	5	$25,960.00	5	$5,192.00	41	69
+Communications	+November	$3,230.00	2	$1,615.00	12	77
	+December	$47,020.00	7	$6,717.14	40	56
+Construction	+December	$11,650.00	5	$2,330.00	23	59
+Electronics	+December	$51,940.00	8	$6,492.50	21	76

5.3.12 Drill through

This operation () extends the display by showing additional columns related to the set of facts shown in the dimension hierarchy. Below is an example of drill through the *Total Sale Amount* measure.



SugarCRM sample unit 2

		Measures				
Account Categorization	Close Period	Total Sale Amount	Number of Sales	Avg Sale Amount	Avg Time To Close (Days)	A
All Accounts	All Periods	$104,701,060.00	3,508	$29,846.37	45	

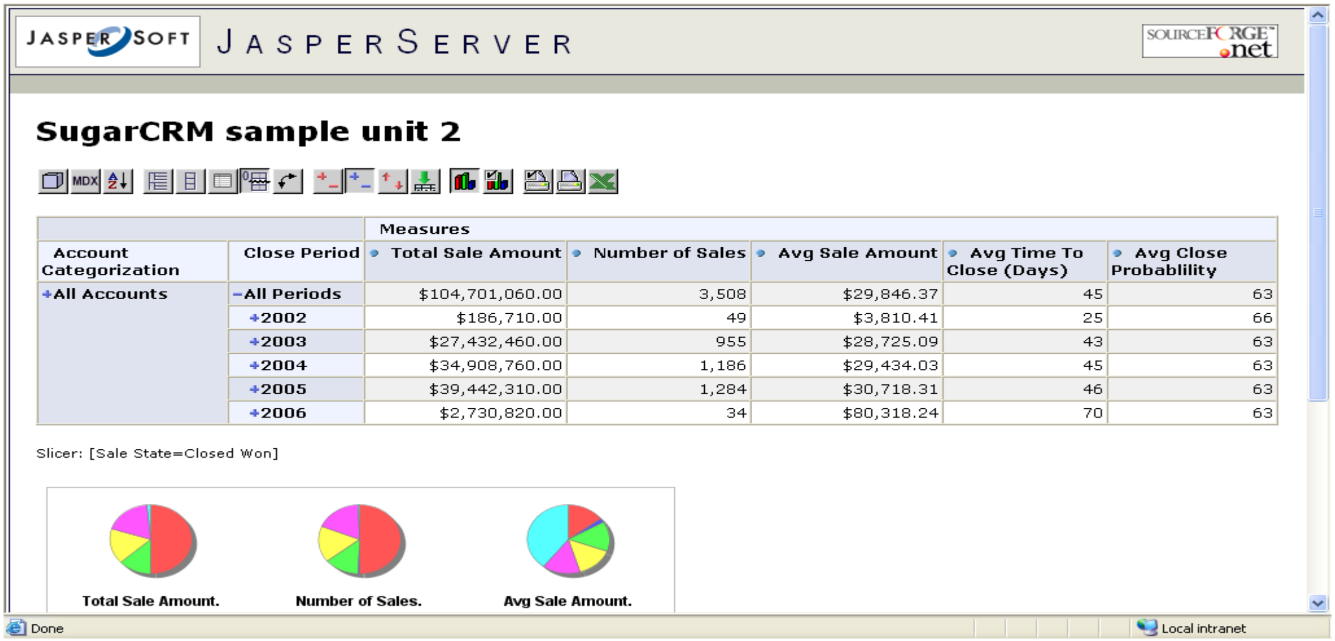
Slicer: [Sale State=Closed Won]

Drill Through Table for Total Sale Amount

Week	Month (Key)	Month	Quarter	Year	x0	x1	x2	x3	x4	Sale State	Sales Person Id (Key)	Sales Person Id	Account Name	Account Type	Industry	Zip/F Cc
1	11	November	4	2002	4	12	December	4	2002	Closed Won	10	beth	Alpha-Murraiin Communications, Inc	Manufacturing	Communications	3589
1	11	November	4	2002	3	12	December	4	2002	Closed Won	10	beth	Alpha-Murraiin Communications, Inc	Manufacturing	Communications	3589
1	11	November	4	2002	3	1	January	1	2003	Closed	5	sarah	N & W Creek	Distribution	Transportation	4052

5.3.13 Show chart

This operation (▥) expands the display by showing table data as charts.



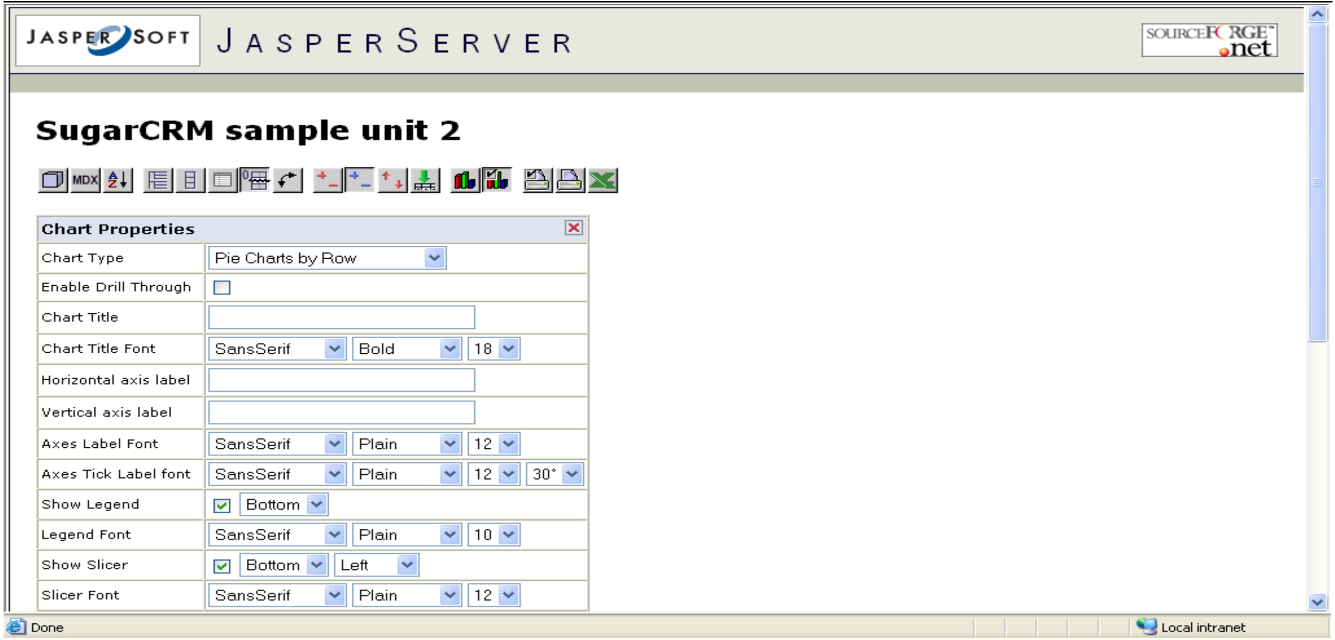
SugarCRM sample unit 2

		Measures				
Account Categorization	Close Period	Total Sale Amount	Number of Sales	Avg Sale Amount	Avg Time To Close (Days)	Avg Close Probablility
All Accounts	All Periods	$104,701,060.00	3,508	$29,846.37	45	63
	2002	$186,710.00	49	$3,810.41	25	66
	2003	$27,432,460.00	955	$28,725.09	43	63
	2004	$34,908,760.00	1,186	$29,434.03	45	63
	2005	$39,442,310.00	1,284	$30,718.31	46	63
	2006	$2,730,820.00	34	$80,318.24	70	63

Slicer: [Sale State=Closed Won]

5.3.14 Chart configuration

This operation (▥) allows a user to define various charting options.



SugarCRM sample unit 2

Chart Properties ☒

Chart Type	Pie Charts by Row
Enable Drill Through	□
Chart Title	
Chart Title Font	SansSerif, Bold, 18
Horizontal axis label	
Vertical axis label	
Axes Label Font	SansSerif, Plain, 12
Axes Tick Label font	SansSerif, Plain, 12, 30°
Show Legend	☑ Bottom
Legend Font	SansSerif, Plain, 10
Show Slicer	☑ Bottom, Left
Slicer Font	SansSerif, Plain, 12

5.3.15 Print configuration

This operation (🖨) allows a user to define various print options.

5.3.16 Print as PDF

This operation (🖨) outputs the current display as a PDF file.

5.3.17 Start Excel

This operation (X) outputs the current display to a Microsoft Excel spreadsheet.

6 JasperServer Administration

The Manage menu appears only if you have administrator privileges. It consists of the following submenus:

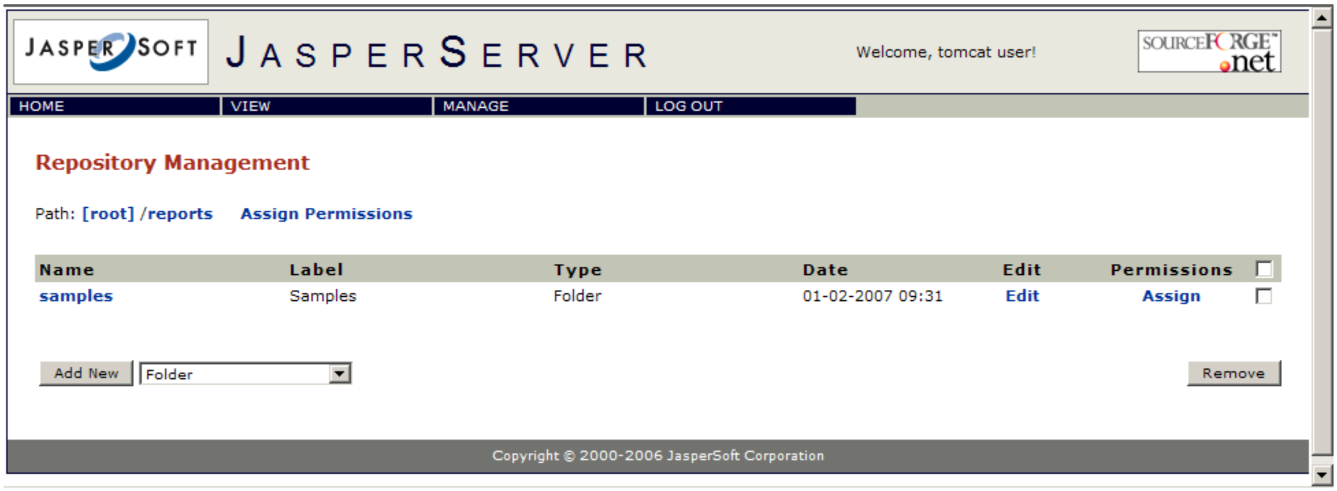
Repository	Displays the repository browser, which contains data sources, images, reports, and other user-created folders.
Roles	Lets you manage roles that you assign to users.
Users	Lets you see and administer users.
Analysis Properties	Displays current Mondrian properties
Flush OLAP Cache	Clears cache used by OLAP operations

6.1 Creating a folder

If you have administrator privileges, JasperServer lets you create a tree of folders in which to put your reports and resources. To do this:

1. Click the REPOSITORY submenu of the MANAGE menu.
2. Choose or create the repository folder in which you want to create your new report. For this example, click reports. The path should display as follows:

   [root]/reports



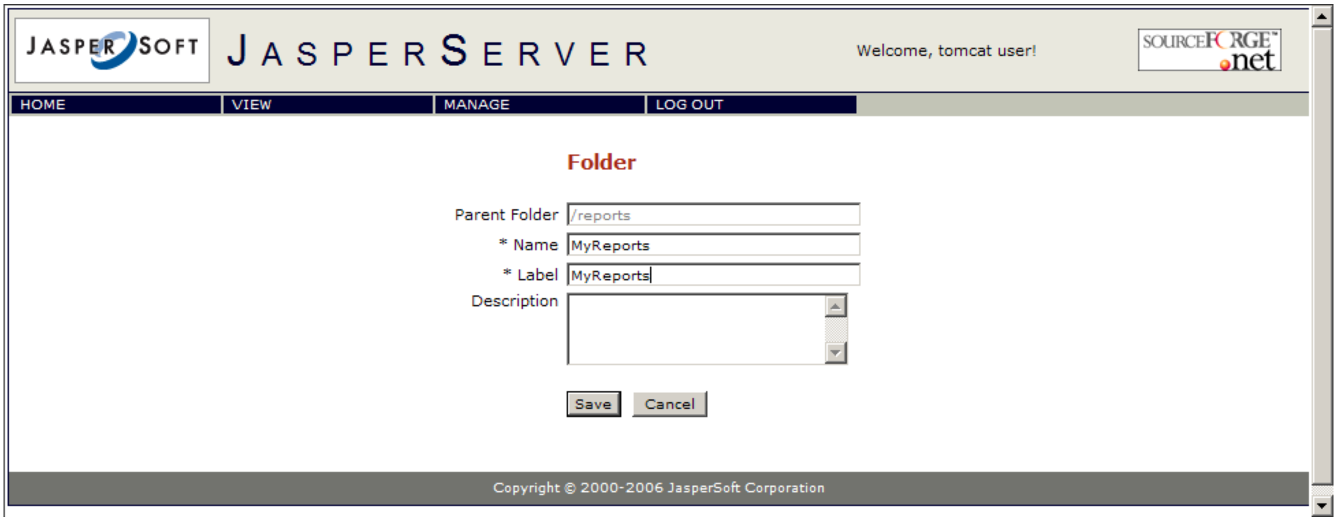
Repository Management

Path: [root] /reports   Assign Permissions

Name	Label	Type	Date	Edit	Permissions	□
samples	Samples	Folder	01-02-2007 09:31	Edit	Assign	□

3. Select *Folder* from the drop-down list besides the *Add New* and click *Add New*. The Edit Folder screen appears.
4. In the *Name* and *Label* fields, enter the name of the new folder you want to create. In this case, enter *MyReports*.

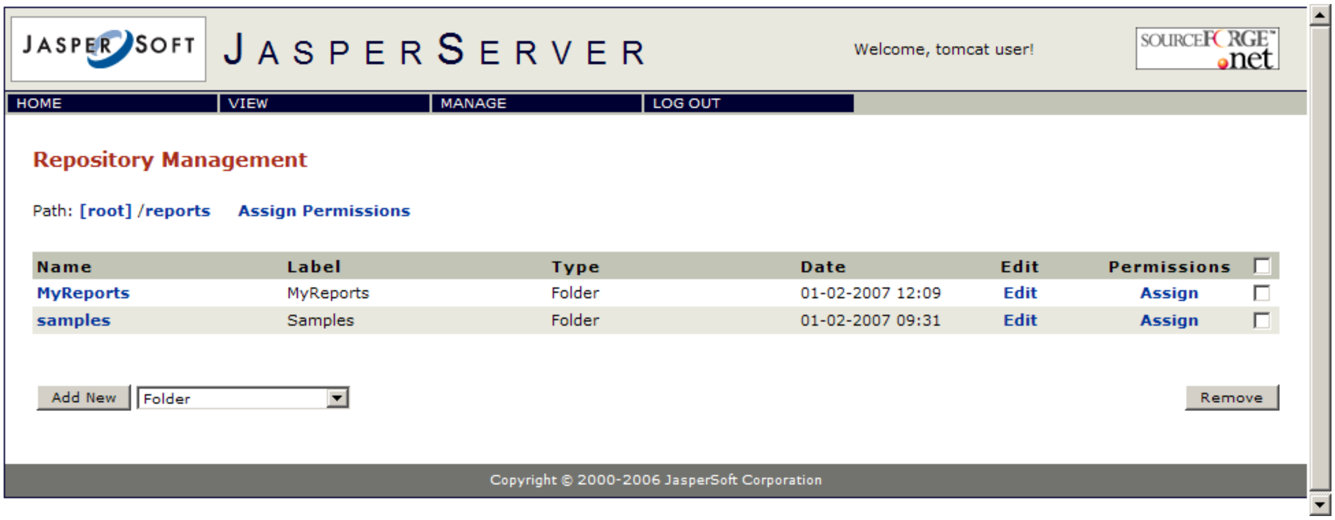
   Note: *Spaces are not permitted in the Name field*.



Folder

Parent Folder: /reports
* Name: MyReports
* Label: MyReports
Description:

5. Click Save. The folder appears in the repository.



Repository Management

Path: [root] /reports   Assign Permissions

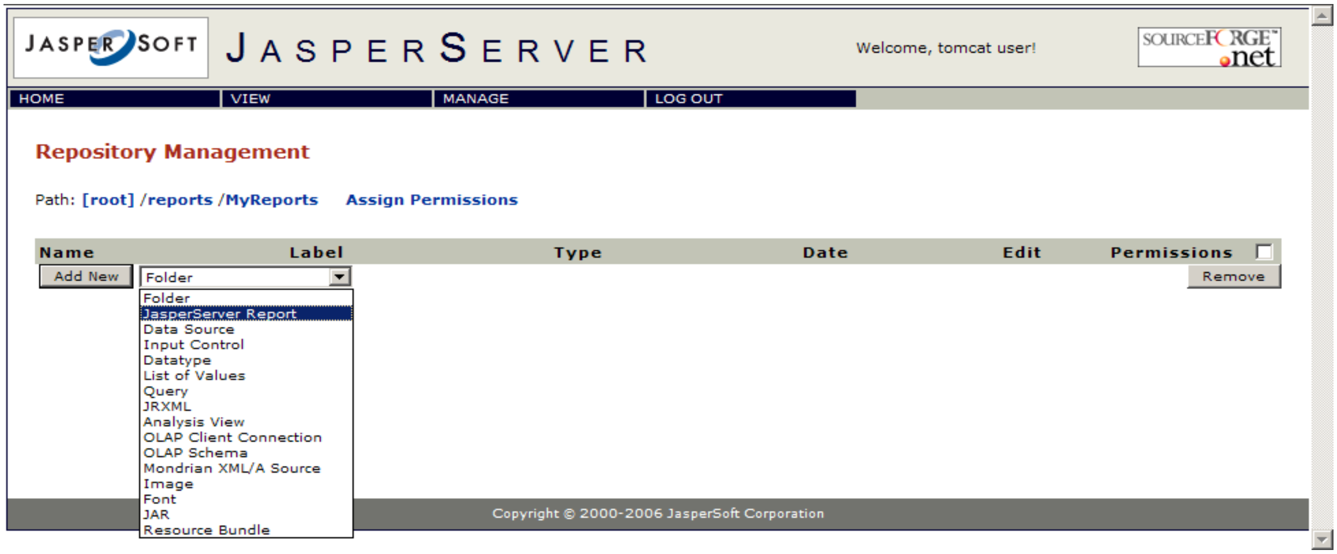
Name	Label	Type	Date	Edit	Permissions	□
MyReports	MyReports	Folder	01-02-2007 12:09	Edit	Assign	□
samples	Samples	Folder	01-02-2007 09:31	Edit	Assign	□

6.2 Creating a simple report

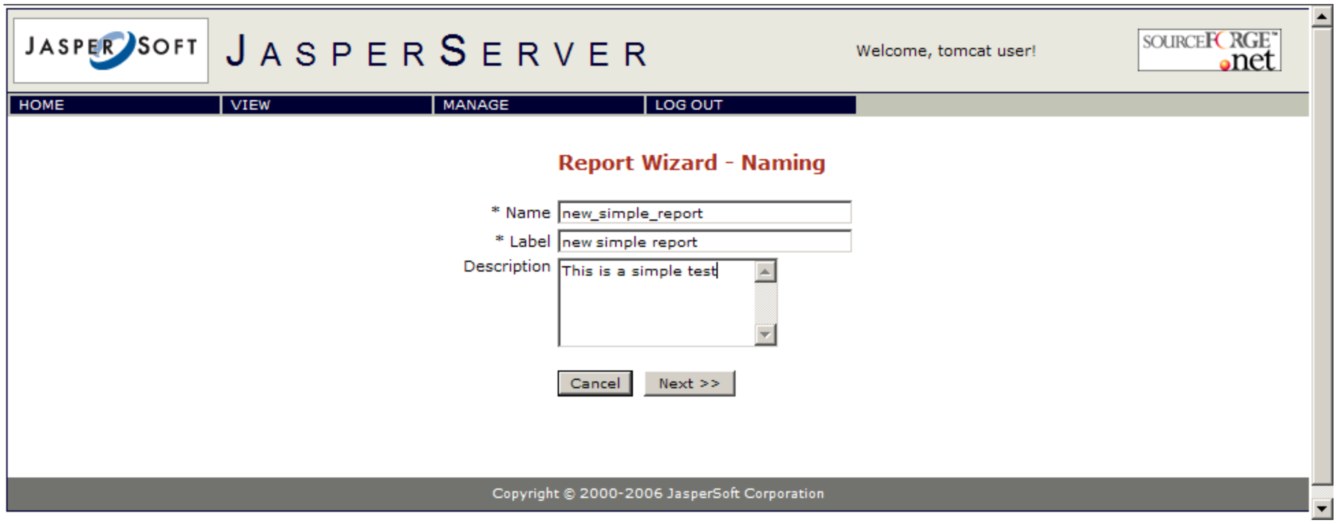
In this example, you'll create a report to run the *AllAccounts.jrxml* file. This is a simple report that takes no input parameters. To create the report, you must have access to the *samples/reports/AllAccounts.jrxml* and the *samples/image/logo.jpg*. Both of these resources can be found in the *samples* directory at the root of the JasperServer installation directory.

6.2.1 Naming the new report

1. In the REPOSITORY browser, select the *MyReports* folder that you have created.
2. In the drop-down list besides the *Add New* button, select *JasperServer Report*.



3. Click *Add New*. The Create Report Wizard appears.
4. For *Name*, enter *new_simple_report*.
5. For *Label*, enter *New Simple Report*.
6. For *Description*, enter *This is a simple test*.

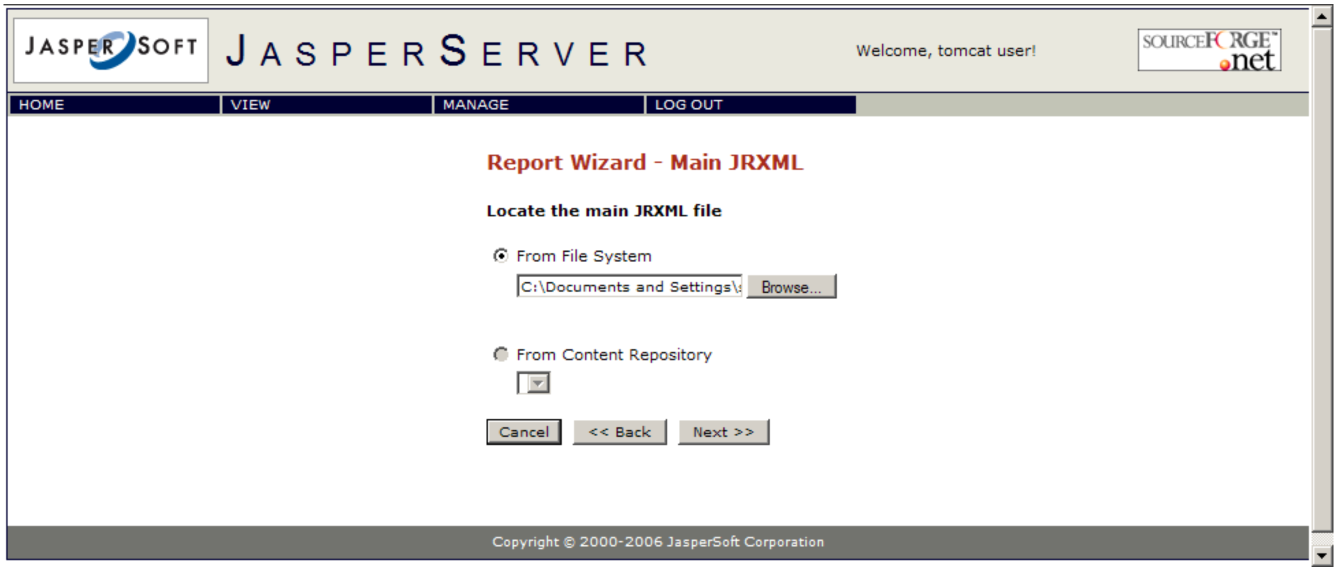


* Name new_simple_report
* Label new simple report
Description This is a simple test

7. Click Next.

6.2.2 Add the JRXML file, logo, and data source

8. Now that you've named the report, you'll add the resources you need to run it. Under *Locate main JRXML File:,* click the *From file System* radio button.
9. Click the *Browse* button. Locate *samples/reports/AllAccounts.jrxml*. (You should find it in the *<JasperServer-install-dir>/samples/reports* directory). Select the *AllAccounts.jrxml* file.



Report Wizard - Main JRXML

Locate the main JRXML file

● From File System

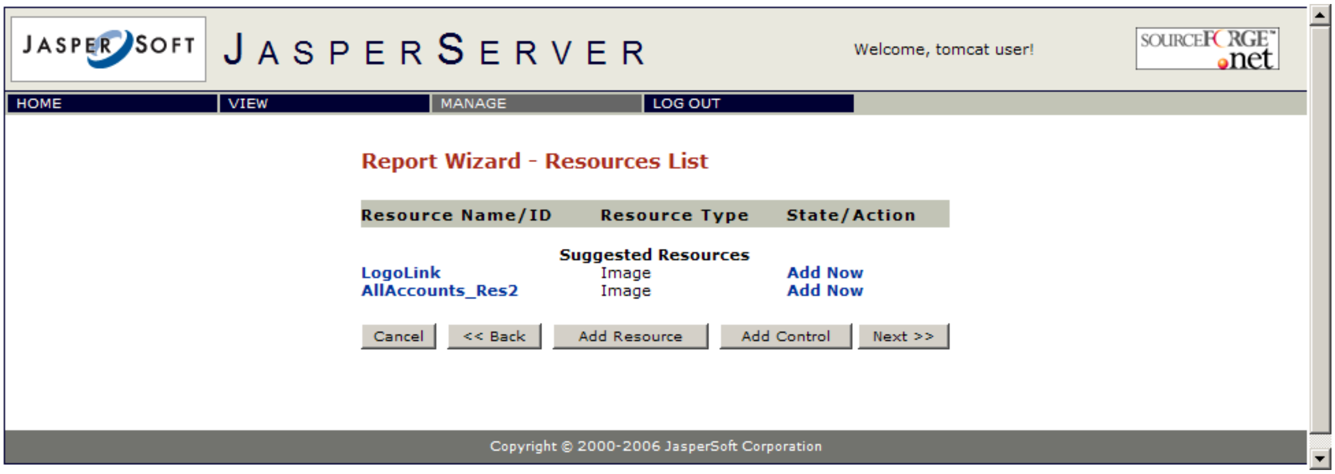
C:\Documents and Settings\

○ From Content Repository

Copyright © 2000-2006 JasperSoft Corporation

10. Click *Next*.

11. The Resource List screen shows you need the *LogoLink* resource.



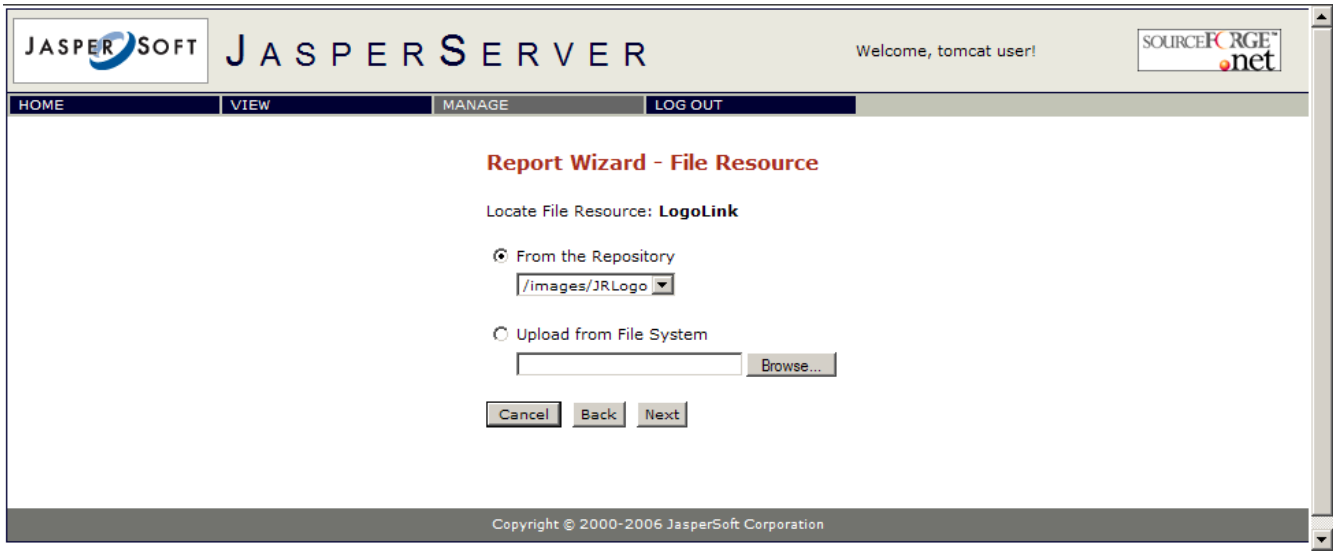
Report Wizard - Resources List

Resource Name/ID	Resource Type	State/Action
	Suggested Resources	
LogoLink	Image	Add Now
AllAccounts_Res2	Image	Add Now

Copyright © 2000-2006 JasperSoft Corporation

12. On the right side of the LogoLink line click *Add Now*.

13. Click the *From the Repository* radio button and select */images/JRLogo*.



Report Wizard - File Resource

Locate File Resource: **LogoLink**

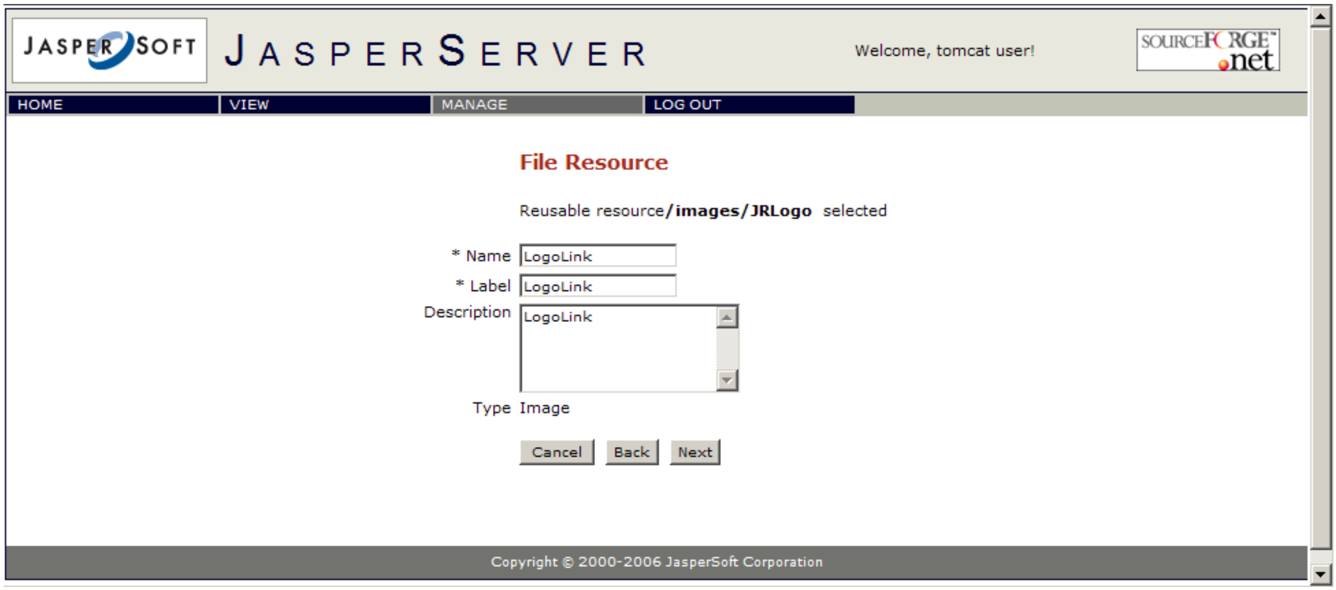
● From the Repository

/images/JRLogo

○ Upload from File System

14. Click *Next*.

15. Accept the default naming of the image in the Repository. Click *Next*.



File Resource

Reusable resource **/images/JRLogo** selected

* Name LogoLink

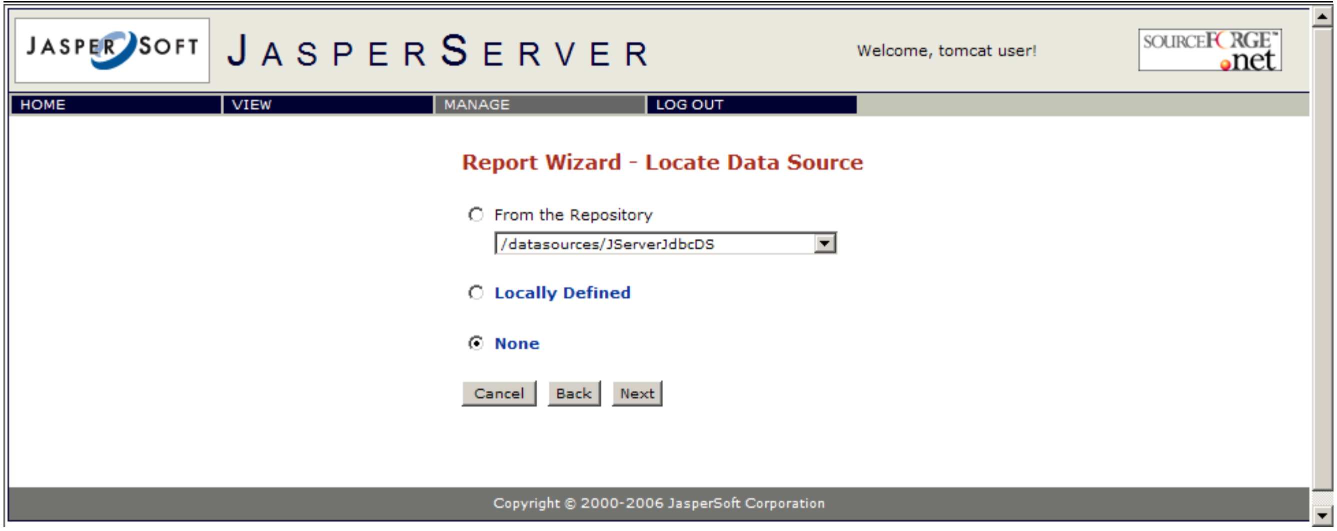
* Label LogoLink

Description LogoLink

Type Image

16. You should now back on the Resource List screen. Click *Next*.

17. On the Locate Data Source screen, click the *From the Repository* radio button.



Report Wizard - Locate Data Source

○ From the Repository

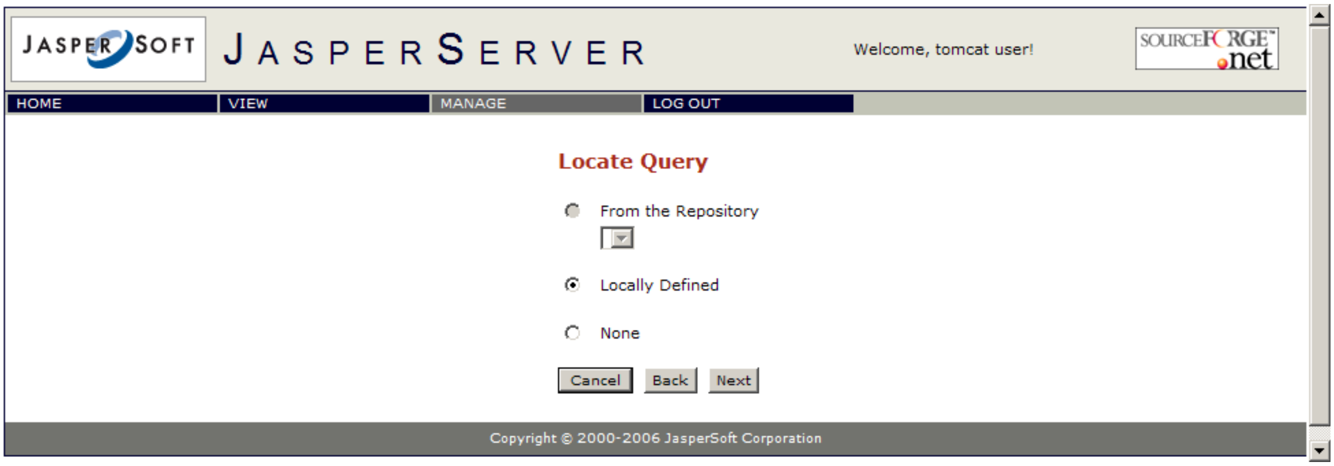
/datasources/JServerJdbcDS

○ Locally Defined

● None

18. Accept the default data source. Click *Next*.

19. Our JRXML file already has a query inside, but are going to overide it by creating local query resource; pick *Locally Defined* in the *Locate Query* page. *None* is for using the JRXML query, if present.



Locate Query

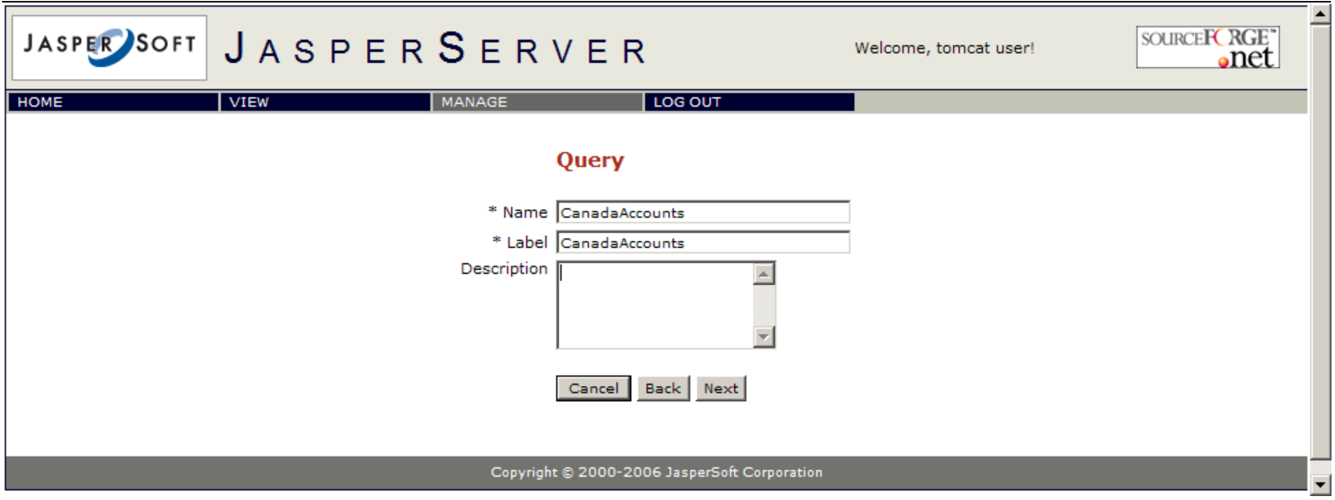
○ From the Repository

● Locally Defined

○ None

20. Name they local query *CanadaAccounts* and use the same for label.

21. Click *Next*.



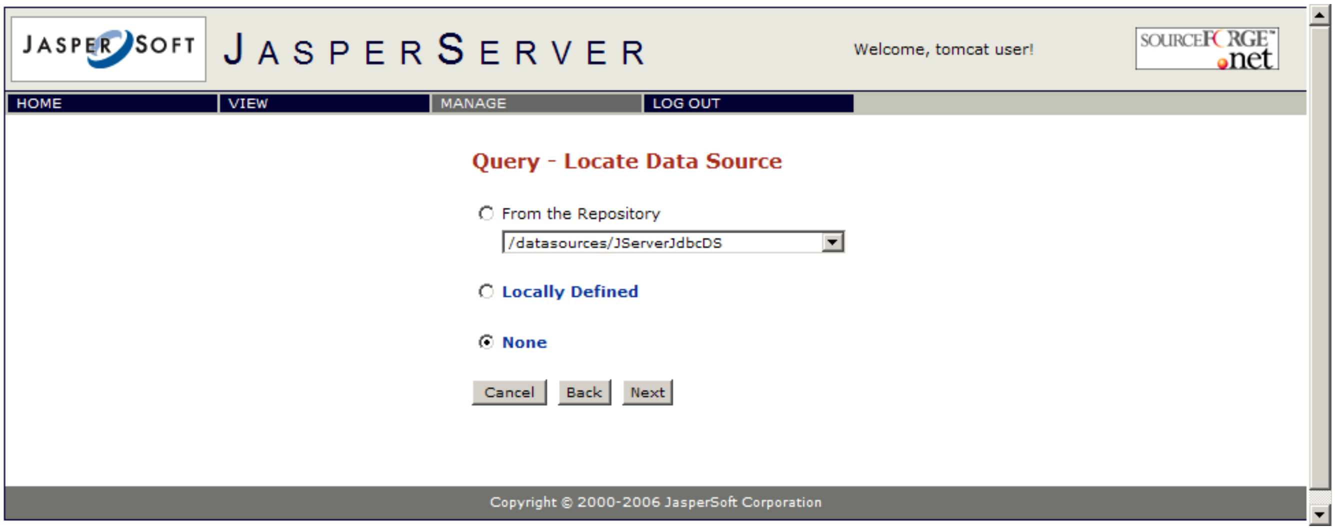
Query

* Name: CanadaAccounts

* Label: CanadaAccounts

Description:

22. The query itself can point to a different data source. But we want to use the data source that was already picked for the report and thus pick *None*.



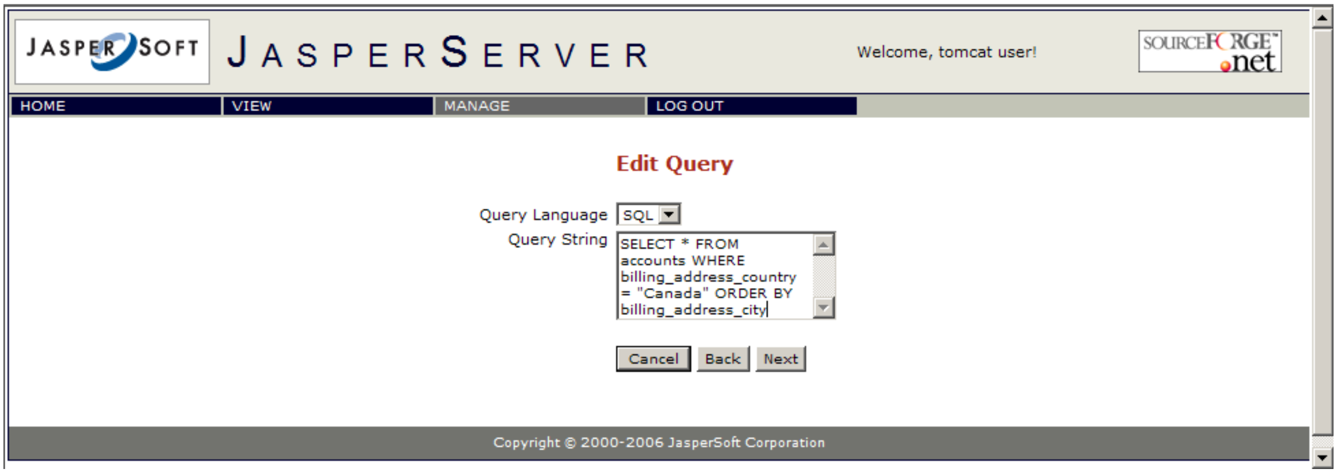
Query - Locate Data Source

○ From the Repository: /datasources/JServerJdbcDS

○ Locally Defined

● None

23. Use *SQL* as query language and the following query text: *SELECT * FROM accounts WHERE billing_address_country = "Canada" ORDER BY billing_address_city*



Edit Query

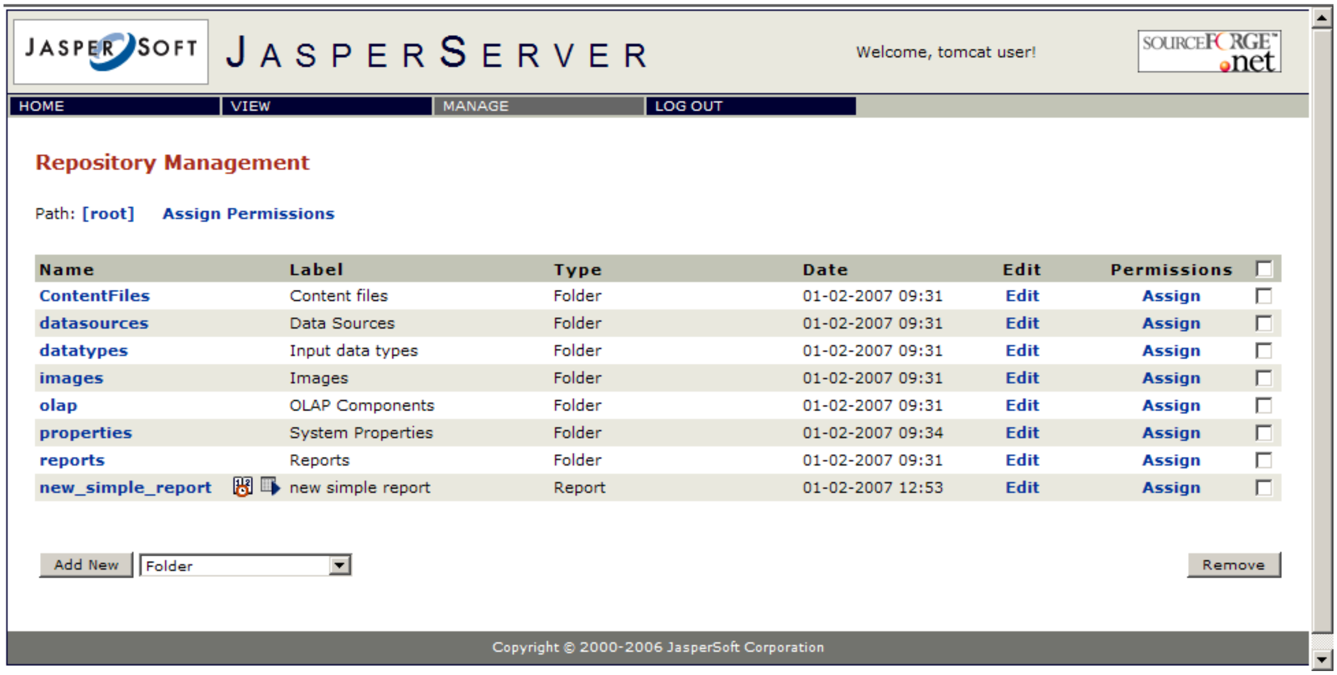
Query Language: SQL

Query String:

```
SELECT * FROM
accounts WHERE
billing_address_country
= "Canada" ORDER BY
billing_address_city
```

24. Skip the *Custom Report View* page by clicking *Next*.

25. The report should be successfully validated. Click *Save*. You have now created your first report!



Repository Management

Path: [root] Assign Permissions

Name	Label	Type	Date	Edit	Permissions	□
ContentFiles	Content files	Folder	01-02-2007 09:31	Edit	Assign	□
datasources	Data Sources	Folder	01-02-2007 09:31	Edit	Assign	□
datatypes	Input data types	Folder	01-02-2007 09:31	Edit	Assign	□
images	Images	Folder	01-02-2007 09:31	Edit	Assign	□
olap	OLAP Components	Folder	01-02-2007 09:31	Edit	Assign	□
properties	System Properties	Folder	01-02-2007 09:34	Edit	Assign	□
reports	Reports	Folder	01-02-2007 09:31	Edit	Assign	□
new_simple_report	new simple report	Report	01-02-2007 12:53	Edit	Assign	□

26. Now, you should see your new JasperServer Report within the Repository Browser screen. Click on the *new_simple_report* to view the report.

The next section shows how to create a more complex report that has a full set of input contols as well as scriptlet class JAR file dependencies.

6.3 Creating a complex report

You will now create a more complex report that contains several input parameters. To create the report, you must have access to all resources, including the *SalesByMonth.jrxml* file, the *SalesByMonthDetail.jrxml* file (the subreport), a data type, and an image for the logo.

Note*: It is assumed you are very familiar with the SalesByMonth.jrxml file and the resources it requires.*

SalesByMonth.jrxml and the resources it needs can be found in the *samples* directory included in the JasperServer Installers. After installation, the *samples* directory is located in the root of the installation directory. Additionally, a *samples.zip* is available for download from the *jasperintel.sourceforge.net* site.

Now, you'll use the available resources to create a report that behaves just like SalesByMonth.

6.3.1 Naming the new report

1. In the repository browser, select the *MyReports* folder that you have created.

2. In the drop-down list besides the Add New button, select JasperServer Report.

3. Click Add New. The Create Report Wizard appears.

4. For Name, enter *new_report*.

5. For Label, enter *New Report*.

6. For Description, enter *This is a test report*.

7. Click Next.

6.3.2 Add the JRXML file, logo, and data source

8. Now that you've named the report, you'll add the resources you need to run it. Under *Locate JRXML File:,* click *From file System*.

9. Search on your file system for the *SalesByMonth.jrxml* file. (You should find it in the <install-dir>/samples/reports directory if you used the JasperServer installer. Otherwise, you can go to the jasperintel.sourceforge.net site and download the samples.zip file.)

10. Click *Next*.

11. The next screen shows you that you need the following resources: a subreport (the *SalesByMonthDetail .jrxml* file) and an image (a logo).

12. Click *Add Now* to the right of the SalesByMonthDetail resource name.

13. Click *Upload From File System* and browse for the *SalesByMonthDetail.jrxml* file.

14. Click Next.

15. Leave the name as *SalesByMonthDetail*. (Note that spaces are not allowed in the name field.)

16. For the Label, enter *Sales By Month* Detail.

17. Click Next. You'll see that you still have to locate the image for the logo.

18. Click *Add Now* to the right of the Logo resource name.

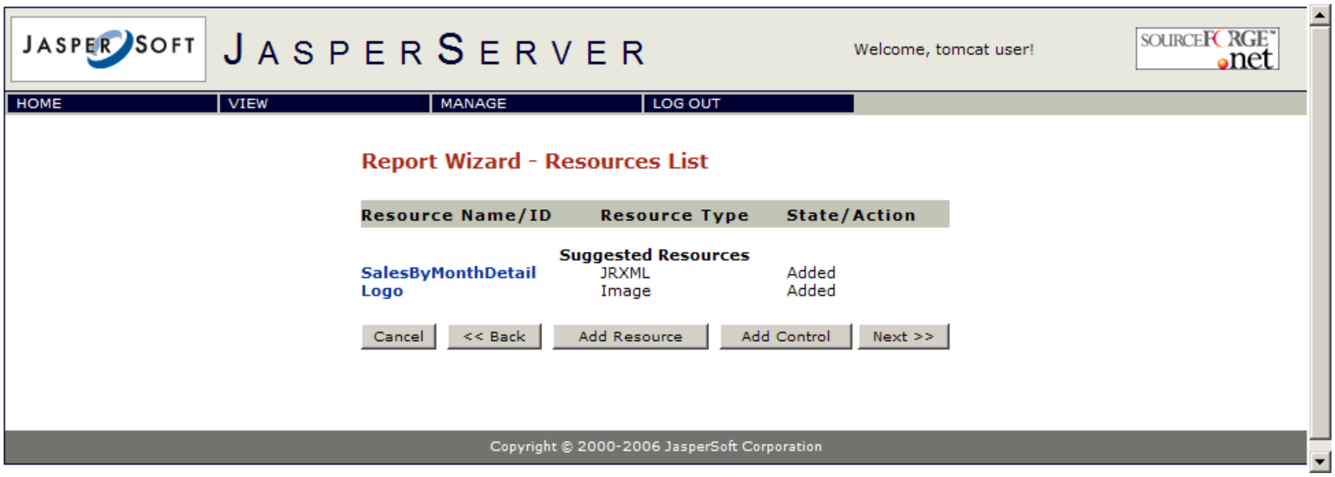
19. Click the *From Content Repository* radio button.

20. Use repository file URI */images/JRLogo*.

21. Click Next.

22. Leave the name *Logo* for name, label, and description.

23. Click Next. You'll see that you've successfully located all necessary resources.



Report Wizard - Resources List

Resource Name/ID	Resource Type	State/Action
	Suggested Resources	
SalesByMonthDetail	JRXML	Added
Logo	Image	Added

6.3.3 Adding input controls

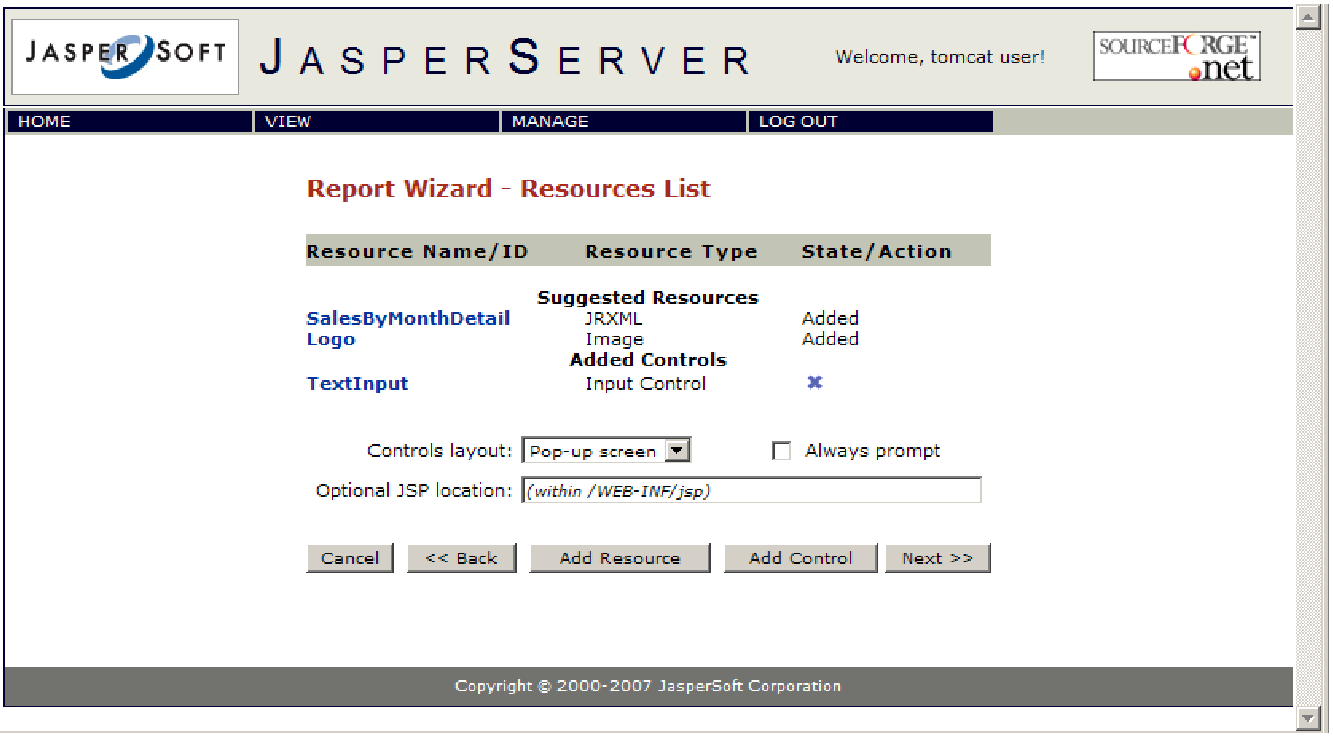
Now, continuing with the same report, you'll add the input controls.

The *SalesByMonth.jrxml* file has four input controls: *TextInput* (an integer), *CheckboxInput* (a Boolean), *ListInput* (a single-select list), and DateInput (a date). You must use these exact names when creating the input controls.

25. Click *Add Control*.
26. Click the *Locally Defined*.
27. Click Next.
28. For Name, enter *TextInput*.
29. For Label, enter *Text Input Control*.
30. For Type, select *Single Value*.
31. Click Next.
32. Now you must define the data type (validation criteria) for the input control. Click *Locally Defined*.
33. Click Next.
34. For Name, enter *integer_type*.
35. For Label, enter *Integer Control*.
36. For Type, select *Number*.

***Note**: In this example, you will not fill in most of the values on this screen. See More on Datatypes below for more information on this screen.*

37. Click Save. You are sent back to the Resources List Screen in the report wizard. You should see *TextInput* in the list of resources.



Report Wizard - Resources List

Resource Name/ID	Resource Type	State/Action
	Suggested Resources	
SalesByMonthDetail	JRXML	Added
Logo	Image	Added
	Added Controls	
TextInput	Input Control	✖

Controls layout: Pop-up screen ▾ □ Always prompt

Optional JSP location: (within /WEB-INF/jsp)

Copyright © 2000-2007 JasperSoft Corporation

38. Click Add Control.
39. Click the *Locally Defined*.

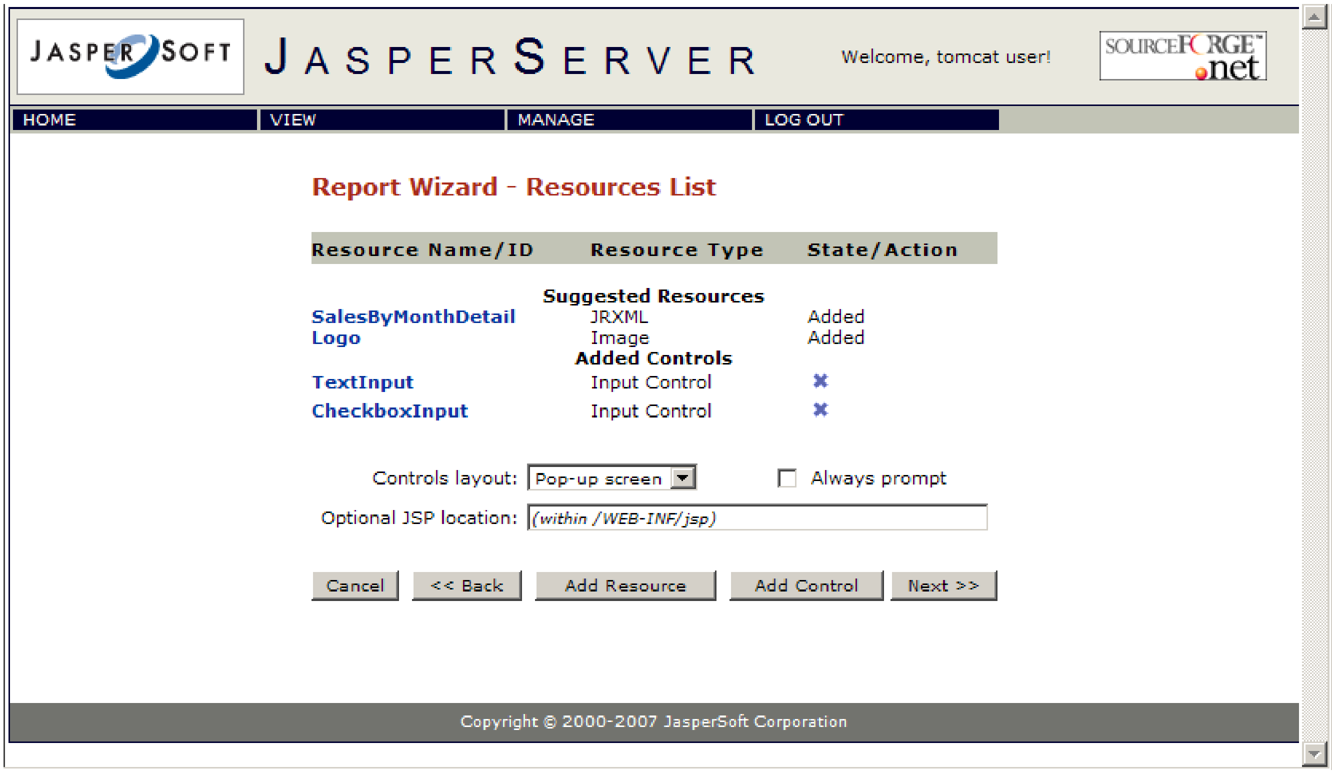
40. Click Next.

41. For Name, enter *CheckboxInput*.

42. For Label, enter *Checkbox Input Control*.

43. For Type, select Boolean.

44. Click Next. You should see the *CheckboxInput* in the list of resources.



Report Wizard - Resources List

Resource Name/ID	Resource Type	State/Action
	Suggested Resources	
SalesByMonthDetail	JRXML	Added
Logo	Image	Added
	Added Controls	
TextInput	Input Control	✖
CheckboxInput	Input Control	✖

Controls layout: Pop-up screen ▾ □ Always prompt

Optional JSP location: *(within /WEB-INF/jsp)*

45. Click Add Control.

46. Click the *Locally Defined*.

47. Click Next.

48. For Name, enter *ListInput*.

49. For Label, enter *List Input Control*.

50. For Type, select *Single Select List of Values*.

51. Click Next.

52. Click *Locally Defined*.

53. Click Next.

54. For Name, enter *list_type*.

55. For Value, enter *List Type*.

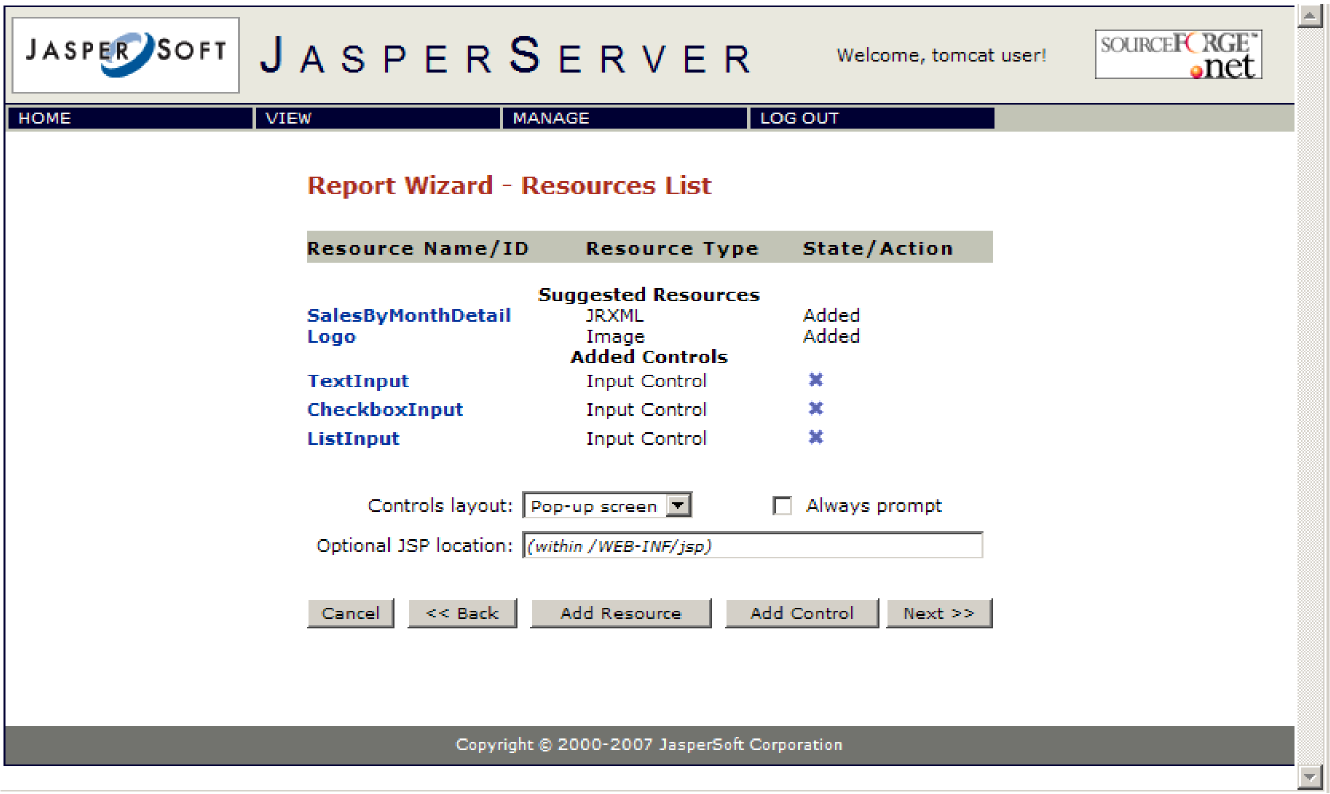
56. Click Next.

57. Under Label, enter *First Item*, under Value, enter *1.* Click the plus button (+).

58. You are presented with a new row for Label and Values. Enter *Second Item* and *2* and click the plus button (+).

59. Lastly, add *Third Item* and *3* and click the plus button (+).

60. Click Save. ListInput appears in your list of resources.



Report Wizard - Resources List

Resource Name/ID	Resource Type	State/Action
	Suggested Resources	
SalesByMonthDetail	JRXML	Added
Logo	Image	Added
	Added Controls	
TextInput	Input Control	✖
CheckboxInput	Input Control	✖
ListInput	Input Control	✖

Controls layout: Pop-up screen ▼ □ Always prompt

Optional JSP location: (within /WEB-INF/jsp)

61. Click Add Control.

62. Click the *Locally Defined*.

63. Click Next.

64. For Name enter *DateInput*.

65. For Label, enter *Date Input Control.*

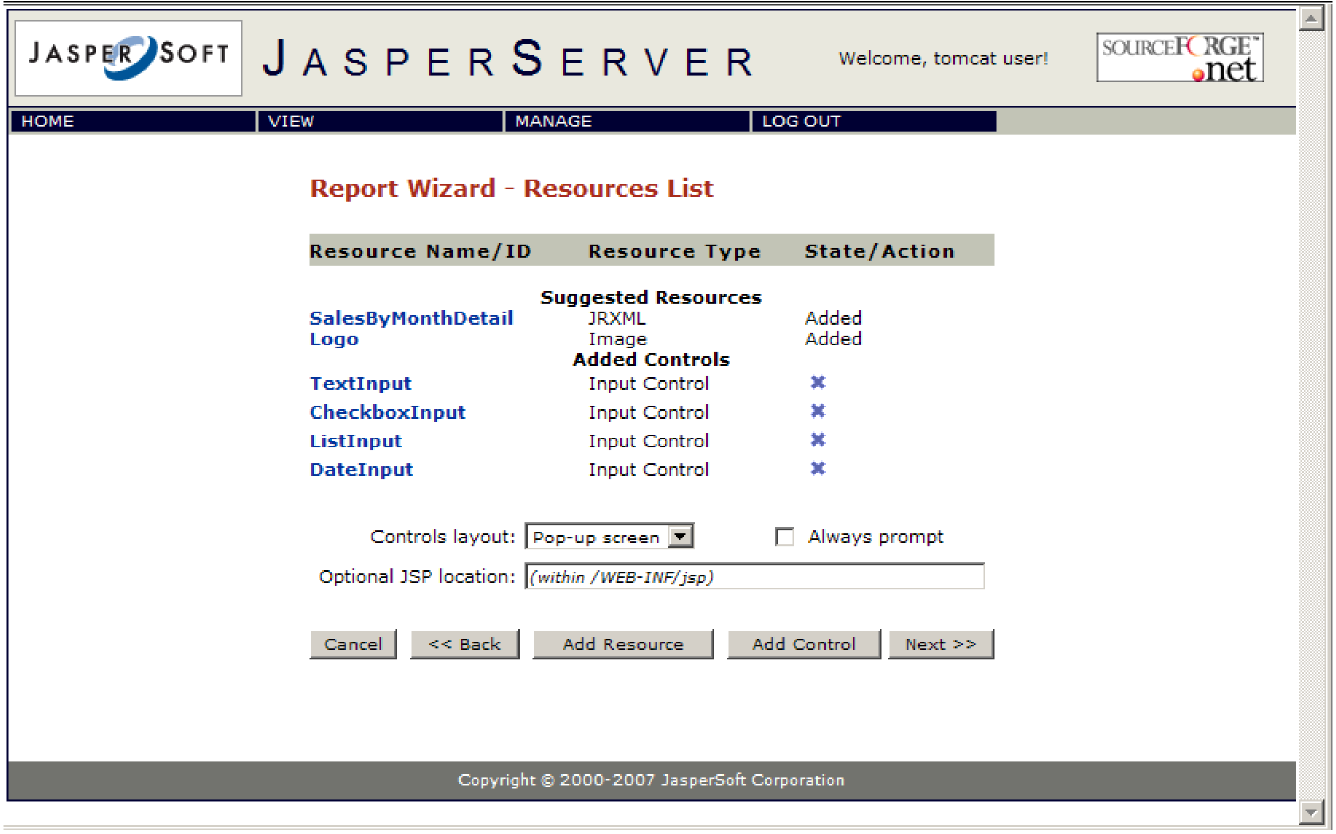
66. For Type, select *Single Value*.

67. Click Next.

68. Click the *From the Repository* radio button.

69. From the drop down list, select */datatypes/date.*

70. Click Next.



Report Wizard - Resources List

Resource Name/ID	Resource Type	State/Action
	Suggested Resources	
SalesByMonthDetail	JRXML	Added
Logo	Image	Added
	Added Controls	
TextInput	Input Control	✖
CheckboxInput	Input Control	✖
ListInput	Input Control	✖
DateInput	Input Control	✖

Controls layout: Pop-up screen ▼ □ Always prompt

Optional JSP location: (within /WEB-INF/jsp)

71. Click Add Control.

72. Click the *Locally Defined*.

73. Click Next.

74. For Name enter Query*Input*.

75. For Label, enter *Query Input Control.*

76. For Type, select *Single Select Query*.

77. Click Next.

78. Click *Locally Defined* for the location of the query.

79. For query name and for label enter *testQuery*

80. Click Next.

81. Click *None* for the query data source and then Next.

82. For Query languate leave the default SQL and for the query text enter the following: SELECT user_name, first_name, last_name FROM users

83. Click Next.

84. For Value column enter *user_name*

85. Add *first_name* and *last_name* for Visible columns using the + button.

86. Click Save.

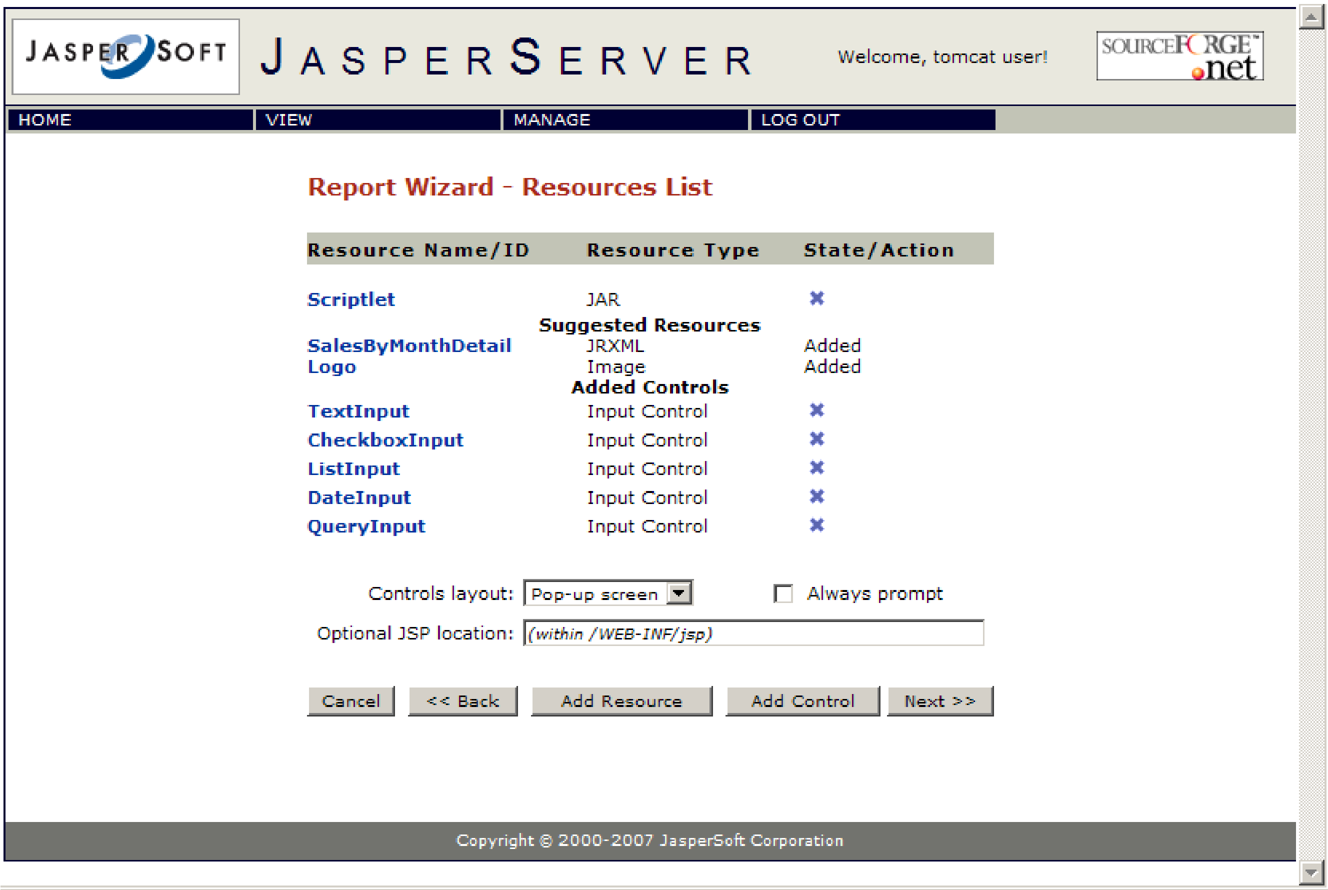
87. You are now done with input controls! Begin adding external resources below.

In the same screen, one can choose whether the report input controls are to be displayed in a pop-up screen on the "View Report" page or in a separate page. A custom JSP page to be used for displaying the input controls can also be specified here; the JSP file needs to already be present on the server.

6.3.4 Adding external resources

The JRXML file contains two external resources: a JAR file scriptlet and a resource bundle for localization. This section shows how to include them in the report.

- Click Add Resource.
- Choose *From File System* and browse for the scriptlet.jar file (samples/jars). Click Next. A screen appears saying the scriptlet was successfully loaded.
- For Name, enter *Scriptlet*. (This name is specified in the .jrxml file, so you must use this name.)
- For Label, enter *Scriptlet JAR*.
- Click Next.



Report Wizard - Resources List

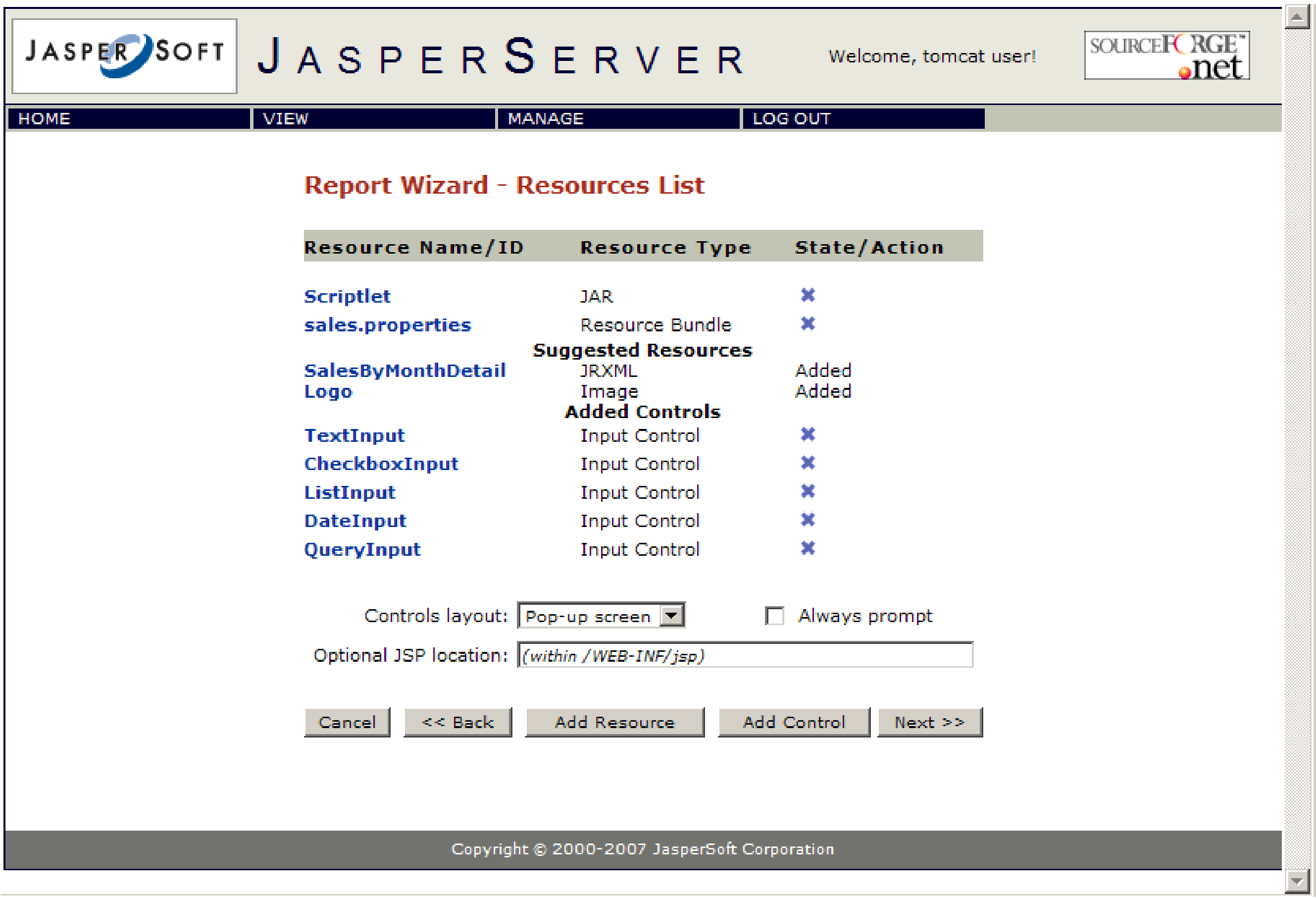
Resource Name/ID	Resource Type	State/Action
Scriptlet	JAR	✖
	Suggested Resources	
SalesByMonthDetail	JRXML	Added
Logo	Image	Added
	Added Controls	
TextInput	Input Control	✖
CheckboxInput	Input Control	✖
ListInput	Input Control	✖
DateInput	Input Control	✖
QueryInput	Input Control	✖

Controls layout: Pop-up screen ☐ Always prompt

Optional JSP location: (within /WEB-INF/jsp)

- Now you'll upload the resource bundle. Click Add Resource.
- From the file system, browse for *sales.properties* (samples/resource_bundles) and click Next. A screen appears saying *sales.properties* was successfully loaded.
- For Name, enter *sales.properties*. (You must use the same name as the file specified in the *SalesByMonth.jrxml* file.)
- For Label, enter *Sales Properties*.

- Click *Next*.
- You have returned to the Resource List screen. Continue to the next section to complete your report.



Report Wizard - Resources List

Resource Name/ID	Resource Type	State/Action
Scriptlet	JAR	✖
sales.properties	Resource Bundle	✖
	Suggested Resources	
SalesByMonthDetail	JRXML	Added
Logo	Image	Added
	Added Controls	
TextInput	Input Control	✖
CheckboxInput	Input Control	✖
ListInput	Input Control	✖
DateInput	Input Control	✖
QueryInput	Input Control	✖

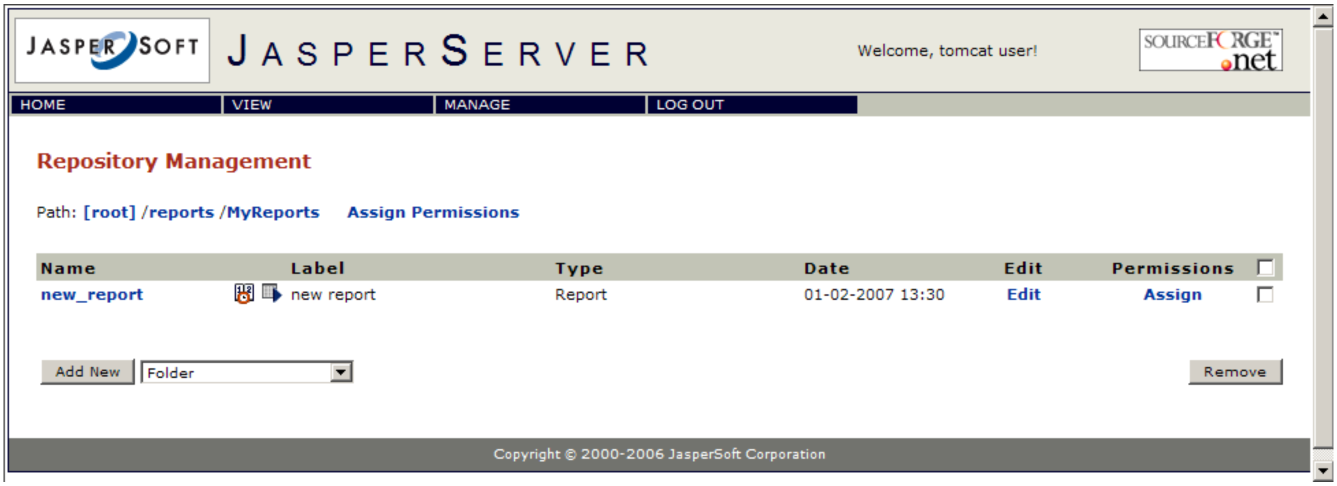
Controls layout: Pop-up screen ☐ Always prompt

Optional JSP location: (within /WEB-INF/jsp)

6.3.5 Adding a datasource

You have now finished adding all the report's resources. Now you'll add a datasource and validate the report.

75. On the Resources List screen, click Next.
76. Click the *From the Repository* radio button.
77. From the drop-down list, choose */datasources/JServerJdbcDS* (default selection).
78. Click Next. A screen appears to say the report was successfully validated.
79. Click Save. The report appears as published in the repository.



Repository Management

Path: [root] /reports /MyReports Assign Permissions

Name	Label	Type	Date	Edit	Permissions	□
new_report	new report	Report	01-02-2007 13:30	Edit	Assign	□

You can now click on your new report to view it. When you enter values in the the input control fields, these values appear toward the top of the report. Input Controls are typically used to modify runtime selections such as WHERE clauses in SQL queries.

6.3.6 More on data types

A data type in JasperServer is more than just a classification of the data. It also includes validation criteria. The data type description page contains the following fields:

Field name	***Description***
Name	A name of your choosing for the data type.
Label	A label of your choosing for the data type.
Description	Any additional information you want to provide about the data type.
Type	The data type, in a stricter sense. Possible values are *Text*, *Number*, *Date*, or *Date-Time*.
Max length	The maximum number of characters in the field.
Decimals	The number of decimal places in the field.
Pattern	A regular expression that restricts the possible values of the field.
Minimum value	The lowest permitted value of the field.
Is strict minumum	If checked, the minumum value itself is not permitted; only values greater than the minumum value are permitted.
Maximum value	The highest permitted value of the field.
Is strict maximum	If checked, the maximum value itself is not permitted; only values less than the maximum value are permitted.

6.4 Creating reports using the iReport plugin

The JasperServer plugin for iReport provides an easy and quick way to manage JasperServer Reports inside the JasperServer repository. The plugin uses a set of web services to interact with the server, and it is able no handle more than one server at the same time.

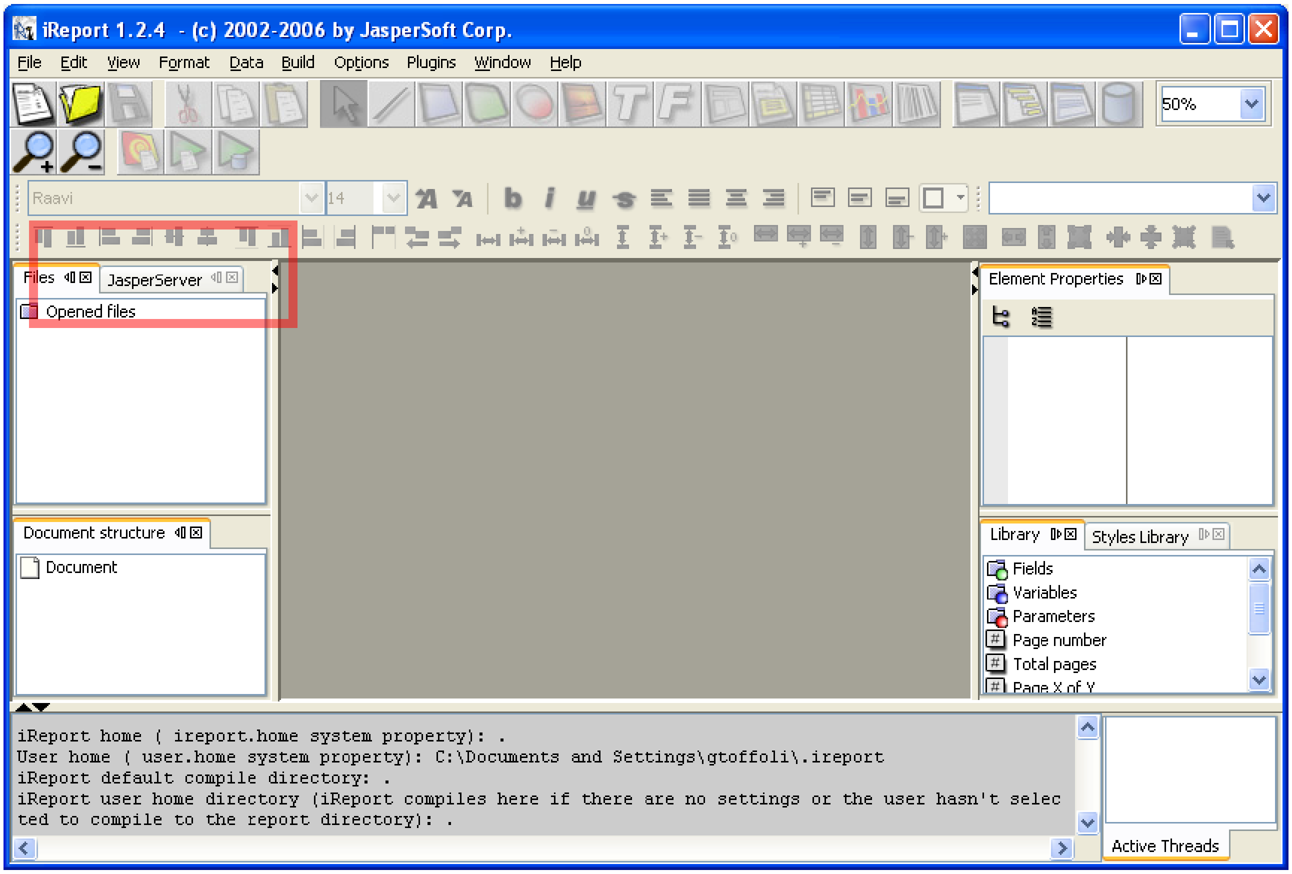
6.4.1 Plugin features

- Repository browsing
- Create / modify / delete folders

- Create / modify / delete generic resources (imges, fonts, jrxml files, jars, property files, ...)
- Create/Modify/import datasources (JDBC, JNDI, JavaBean)
- Create / modify / delete JasperServer Reports
- Create/modify/delete datatypes
- Create/modify/delete List of values
- Create/modify/delete controls
- Add/remove/link controls to/from JasperServer Reports
- Drag and drop of images and subreports into the design window
- Repository resource chooser dialog

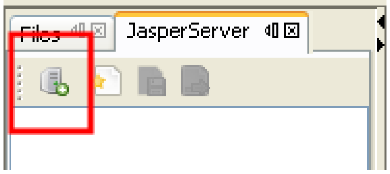
6.4.2 Making connection to JasperServer

For Windows platform, select the Start/All Programs/JasperSoft/iReport-1.3.x/iReport-1.3.x menu item to start iReport. (See 8.3 for a detailed description of installing and configuring the iReport plugin.)

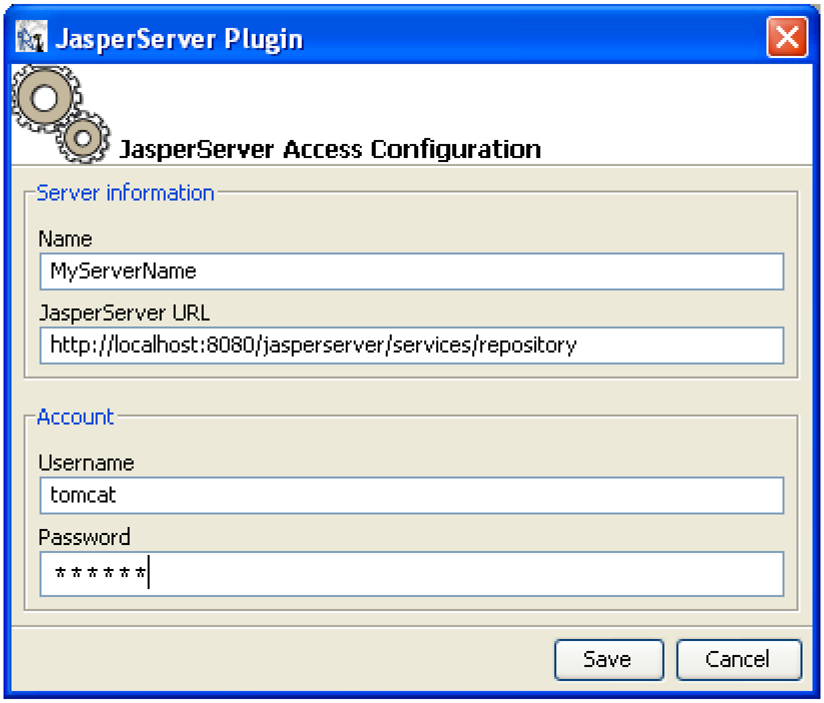


```
iReport home ( ireport.home system property): .
User home ( user.home system property): C:\Documents and Settings\gtoffoli\.ireport
iReport default compile directory: .
iReport user home directory (iReport compiles here if there are no settings or the user hasn't selec
ted to compile to the report directory): .
```

When the plugin is started for the first time, no servers are yet configured. To add a server, press the first button in the plugin toolbar.



You get a dialog box titled"JasperServer Plugin" that allows for JasperServer access configuration. In this dialog box, you will enter the information that will enable you to connect to your JasperServer Web Services application.



Name
MyServerName

JasperServer URL
http://localhost:8080/jasperserver/services/repository

Username
tomcat

Password
******

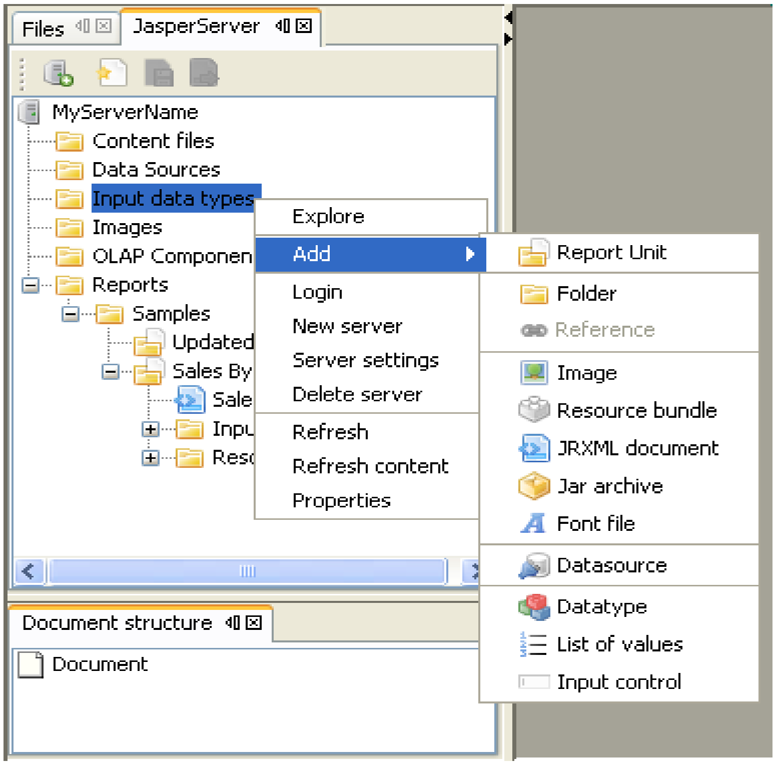
For instance:

Name:	MyServerName
JasperServer URL:	<you should only need to change the hostname>
Username:	<jasperadmin>
Password:	<password>

A server entry should now appear in this pane.

6.4.3 Using the plugin

Double-click on the server entry to see a list of Resources at the root of the JasperServer repository. You may view the property information on the Resources in the repository. You may download, edit and upload jrxml files. You may download images and other files (using right-click "export file").



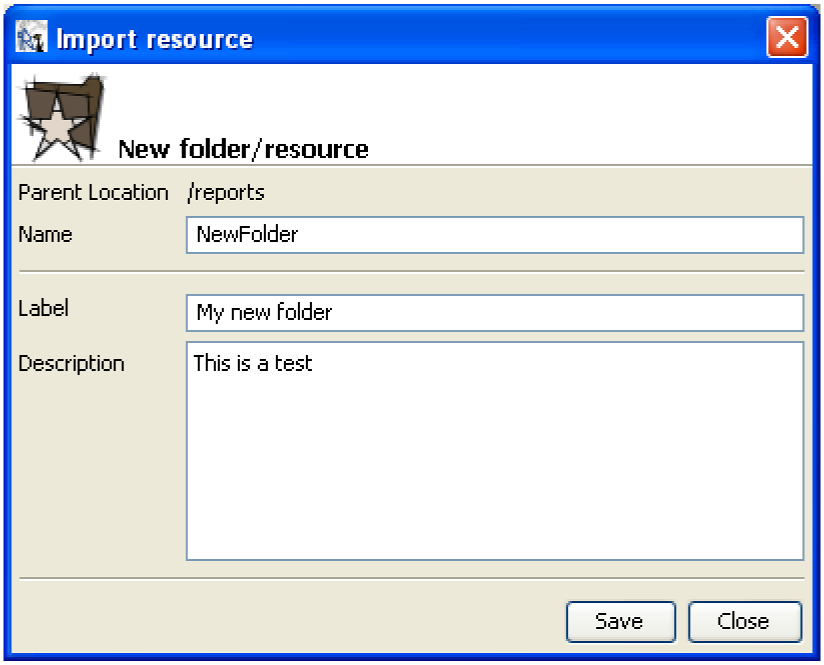
You can browse the repository double-clicking the server and folder icons.

6.4.4 Folders

When a folder is selected, you can add one of the following objects into a folder:

- another folder
- a brand new JasperServer Report
- a generic resource (like an image or a JRXML file)
- a datasource definition
- a datatype
- a list of values (used inside a control)
- an input control

To create a folder, select the parent folder (or the server entry to create a folder in the root), and click Add / Folder.



Import resource

New folder/resource

Parent Location /reports

Name: NewFolder

Label: My new folder

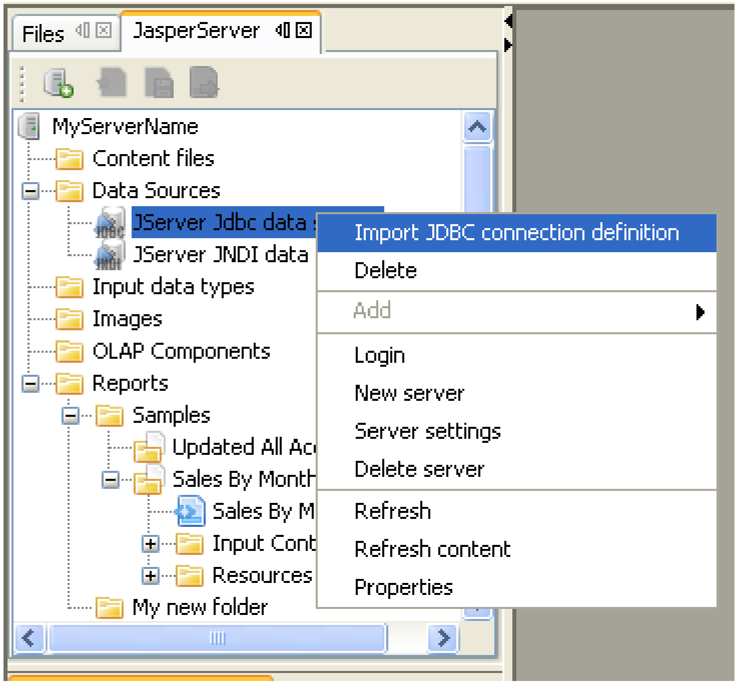
Description: This is a test

Only the name and the label are mandatory fields. To delete a folder select the "Delete" from the popup menu.

6.4.5 Importing JDBC connections into iReport

You can import a JDBC datasource from the server into iReport selecting the JDBC datasource from the repository tree. Right-click on the datasource, and select the item Import JDBC connection definition.

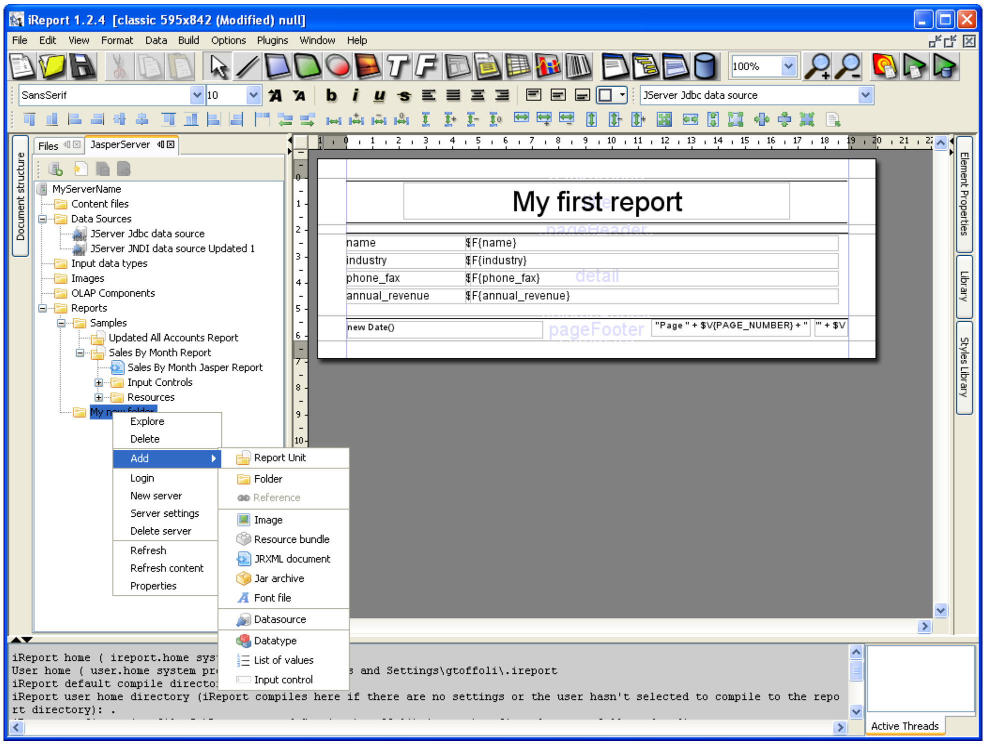
The new connection will stored in the iReport datasources/connections list. Now you can use this new connection with the iReport wizard to create your reports.



6.4.6 Creating a JasperServer Report from iReport

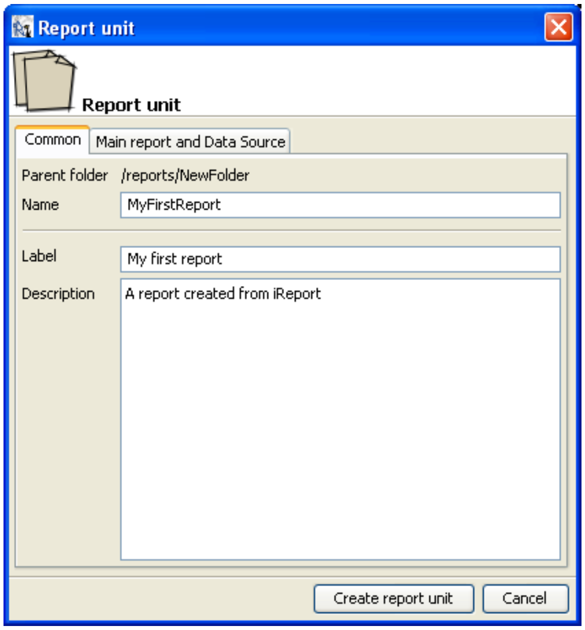
JasperServer Reports are displayed as special folders with the following icon:

To create a brand new JasperServer Report, you need at least a JRXML file (it will become the main JRXML for the JasperServer Report.



```
iReport home ( ireport.home sys
User home ( user.home system pr                    s and Settings\gtoffoli\.ireport
iReport default compile directo
iReport user home directory (iReport compiles here if there are no settings or the user hasn't selected to compile to the repo
rt directory): .
```

Choose the parent folder for the new JasperServer Report, select Add / JasperServer Report. You'll get the JasperServer Report dialog.

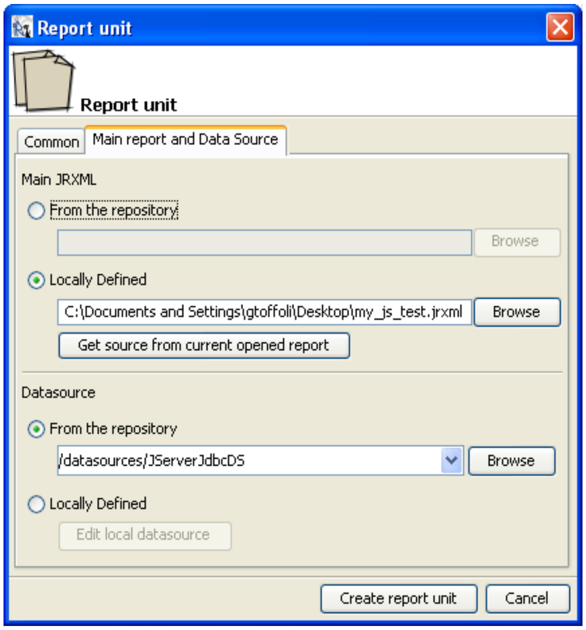


Parent folder /reports/NewFolder

Name: MyFirstReport

Label: My first report

Description: A report created from iReport



Main JRXML

○ From the repository

● Locally Defined

C:\Documents and Settings\gtoffoli\Desktop\my_js_test.jrxml

Datasource

● From the repository

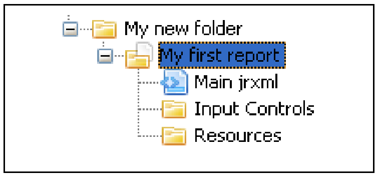
/datasources/JServerJdbcDS

○ Locally Defined

The dialog is composed by two tabs: “Common” and “Main Report and Datasource”. All fileds, except Description, are mandatory.

If you want import the current opened file, save it before creating the report, and press the button “Get source from current opened report”.

If you want, you can pick an existing JRXML file from the repository, and/or define a local datasource (that is belonging to the JasperServer Report).



In any given JasperServer Report, there is always a main jrxml file, a set of input controls (zero or more) and a set of resources (zero or more).

6.4.7 Edit a JRXML

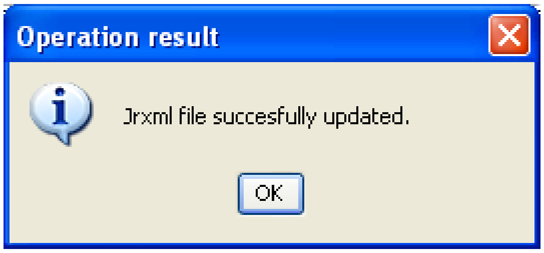
To edit a JRXML, double click on the JRXML file entry in the repository tree. If you have created a JasperServer Report from the current opened report, close this report, because this IS NOT the report present now on the server.

When you double click on the JRXML file, the JRXML is extracted from the repository and stored in a local copy into the directory <USER_HOME>/.ireport/jstmp. This directory is never clean.

The JRXML source is automatically opened in iReport.

To restore the modified file on the server, save it, then select the file entry in the repository tree, and click the button on the plugin toolbar.

A message will confirm that the operation was completed successful.

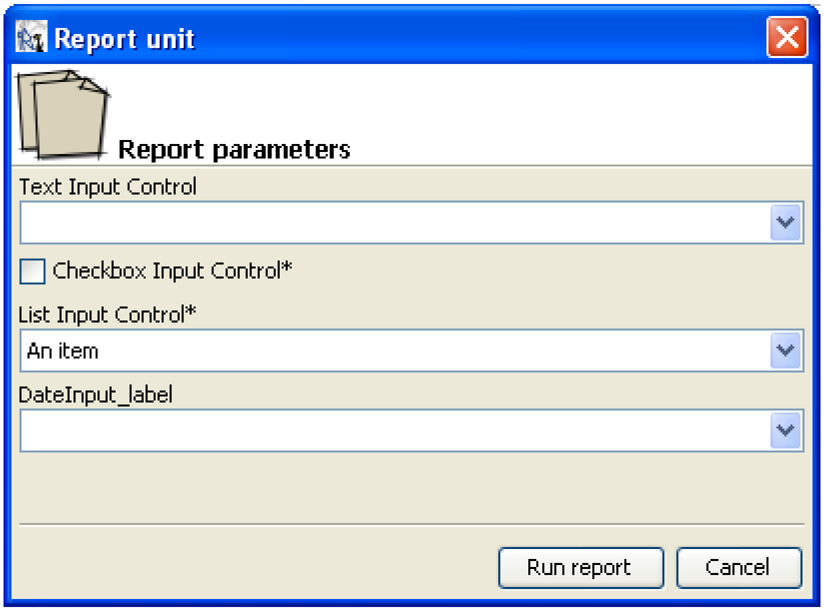


Jrxml file succesfully updated.

6.4.8 Running a JasperServer Report from iReport

To run a JasperServer Report on the server from iReport using all export options set in iReport, select a JasperServer Report from the repository tree, and click the button on the plugin toolbar.

If the JasperServer Report has input controls, the report parameters window will ask for values.



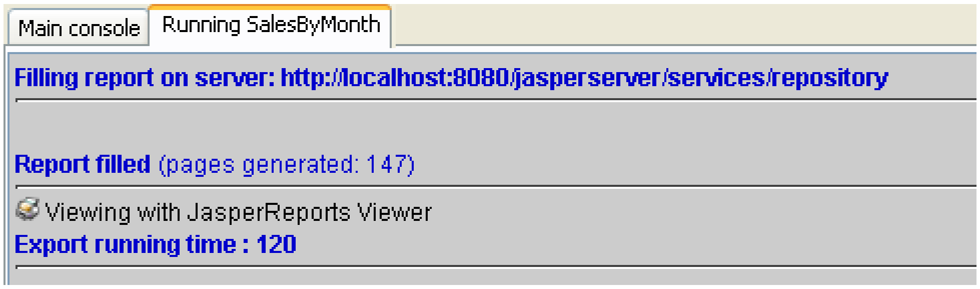
Text Input Control

□ Checkbox Input Control*

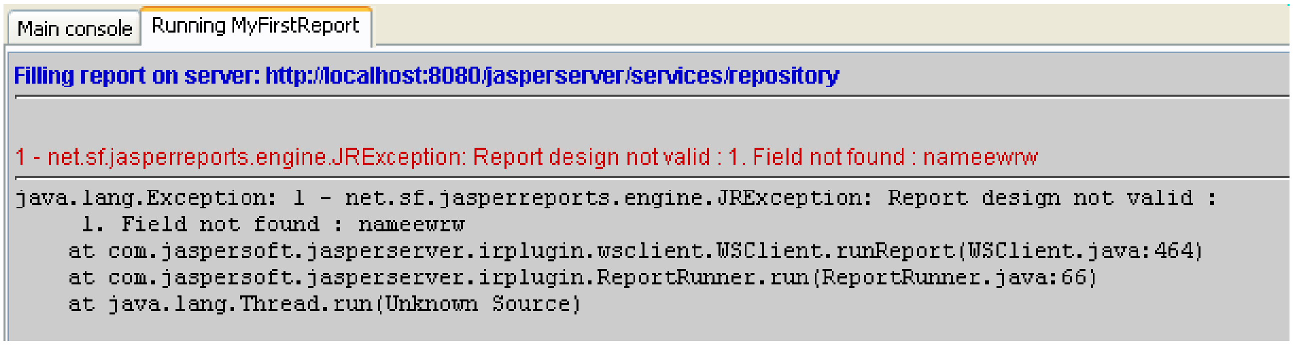
List Input Control*
An item

DateInput_label

All messages coming from the server will be displayed on the iReport log pane.



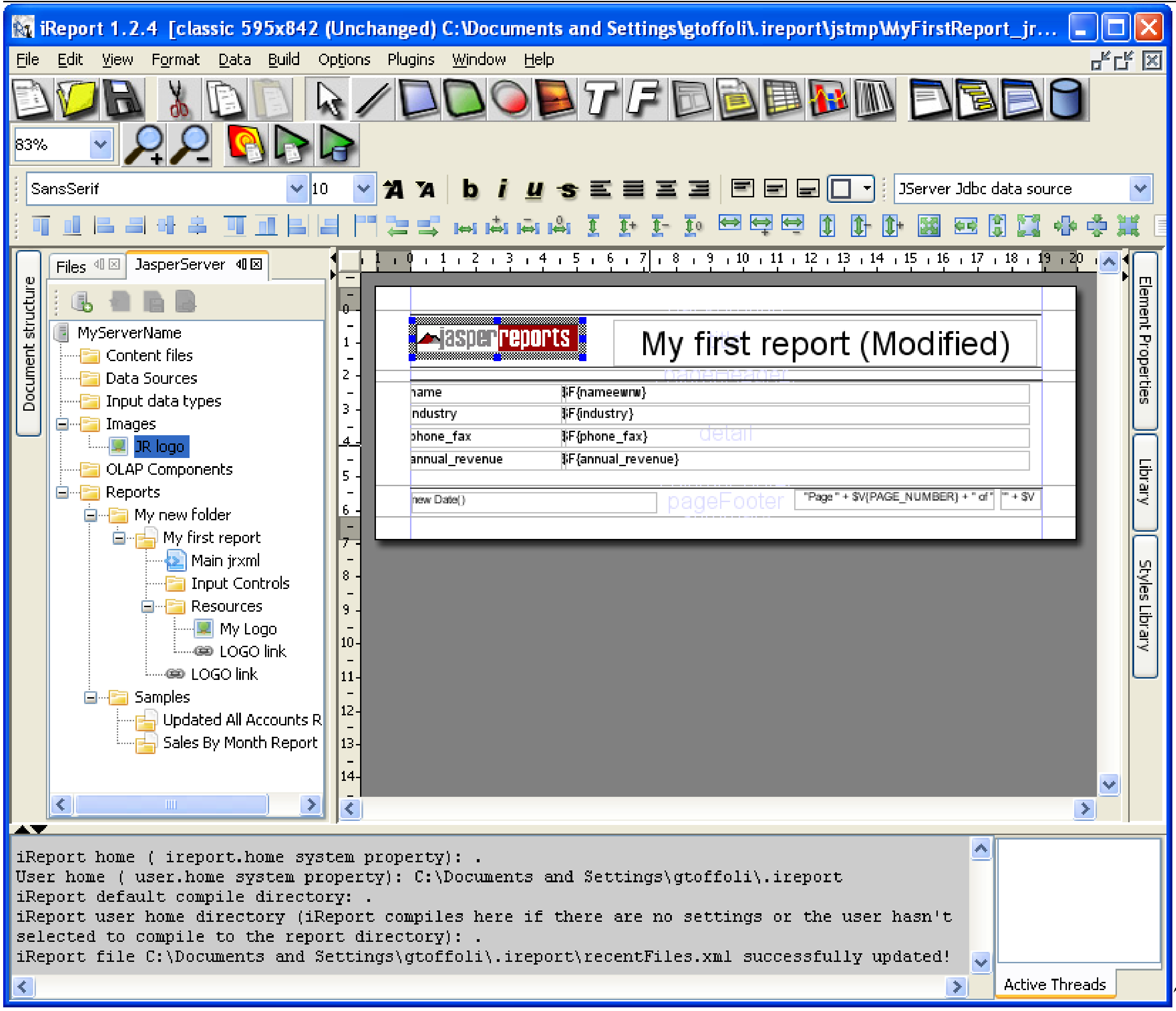
```
Filling report on server: http://localhost:8080/jasperserver/services/repository

Report filled (pages generated: 147)
Viewing with JasperReports Viewer
Export running time : 120
```



```
Filling report on server: http://localhost:8080/jasperserver/services/repository

1 - net.sf.jasperreports.engine.JRException: Report design not valid : 1. Field not found : nameewrw
java.lang.Exception: 1 - net.sf.jasperreports.engine.JRException: Report design not valid :
     1. Field not found : nameewrw
    at com.jaspersoft.jasperserver.irplugin.wsclient.WSClient.runReport(WSClient.java:464)
    at com.jaspersoft.jasperserver.irplugin.ReportRunner.run(ReportRunner.java:66)
    at java.lang.Thread.run(Unknown Source)
```

6.4.9 Adding an image to a report

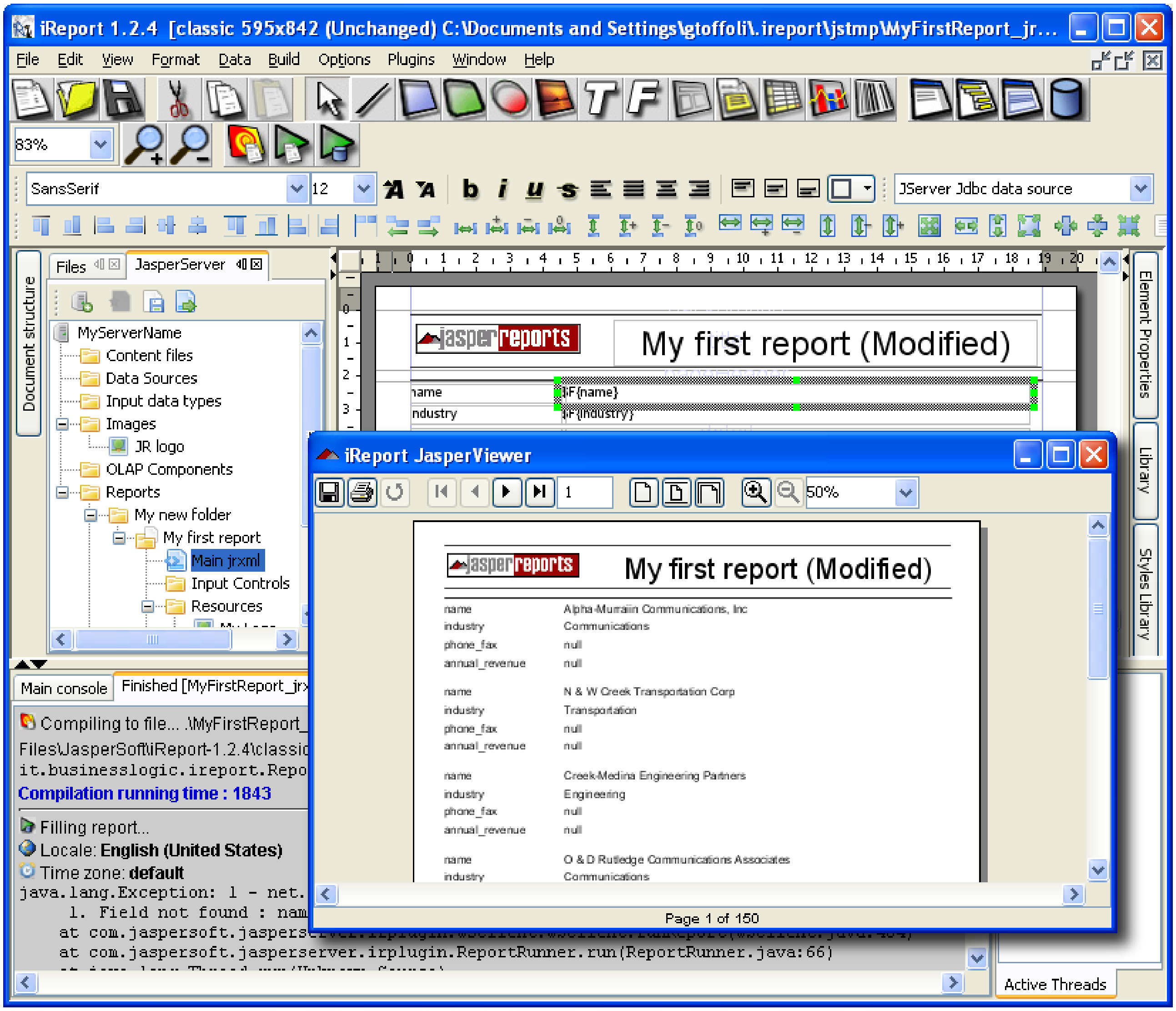
If you want add an image stored in your repository into a report, you can drag the image into the design window. The plugin will add the image element for you, setting the right expression to refer the image into the server repository.



```
iReport home ( ireport.home system property): .
User home ( user.home system property): C:\Documents and Settings\gtoffoli\.ireport
iReport default compile directory: .
iReport user home directory (iReport compiles here if there are no settings or the user hasn't
selected to compile to the report directory): .
iReport file C:\Documents and Settings\gtoffoli\.ireport\recentFiles.xml successfully updated!
```

74

Save and run the report as described in the previous paragraph to see the result.



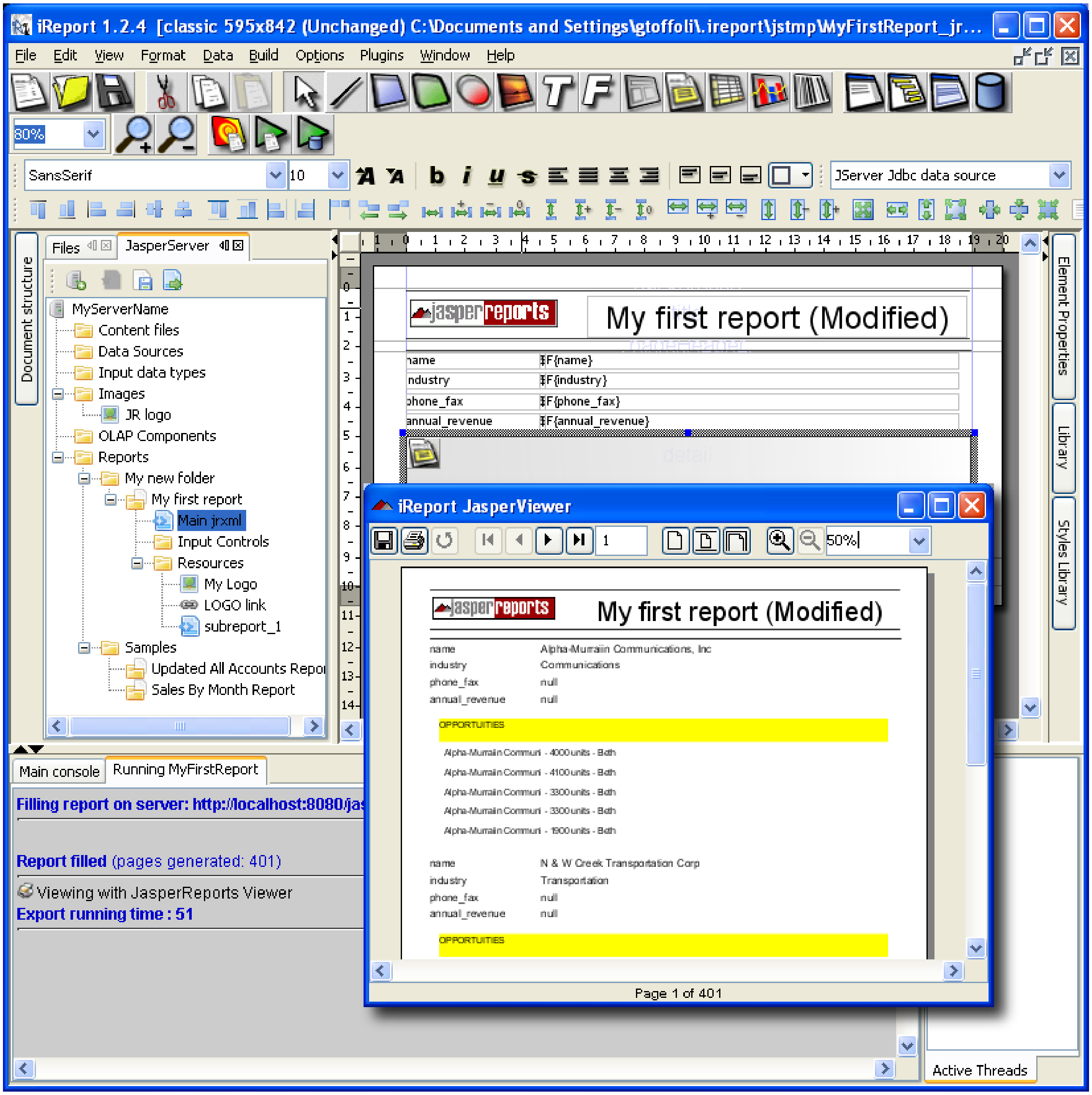
```
name            Alpha-Murraiin Communications, Inc
industry        Communications
phone_fax       null
annual_revenue  null

name            N & W Creek Transportation Corp
industry        Transportation
phone_fax       null
annual_revenue  null

name            Creek-Medina Engineering Partners
industry        Engineering
phone_fax       null
annual_revenue  null

name            O & D Rutledge Communications Associates
industry        Communications
```

Page 1 of 150

```
Compiling to file... .\MyFirstReport_
Files\JasperSoft\iReport-1.2.4\classic
it.businesslogic.ireport.Repo
Compilation running time : 1843
Filling report...
Locale: English (United States)
Time zone: default
java.lang.Exception: 1 - net.
    1. Field not found : nam
        at com.jaspersoft.jaspers
        at com.jaspersoft.jasperserver.irplugin.ReportRunner.run(ReportRunner.java:66)
```

6.4.10 Adding an subreport to a report

Similarly to what we have done to add an image, we can add a subreport to a report. First of all we have to create the JRXML that will be used as subreport, then we will add the new JRXML as resource to the repository, or to the JasperServer Report.



iReport 1.2.4 [classic 595x842 (Unchanged) C:\Documents and Settings\gtoffoli\.ireport\jstmp\MyFirstReport_jr...

My first report (Modified)

name	$F{name}
industry	$F{industry}
phone_fax	$F{phone_fax}
annual_revenue	$F{annual_revenue}

Filling report on server: http://localhost:8080/jas

Report filled (pages generated: 401)

Viewing with JasperReports Viewer

Export running time : 51

My first report (Modified)

name	Alpha-Murrain Communications, Inc
industry	Communications
phone_fax	null
annual_revenue	null

OPPORTUNITIES

Alpha-Murrain Communi - 4000 units - Both
Alpha-Murrain Communi - 4100 units - Both
Alpha-Murrain Communi - 3300 units - Both
Alpha-Murrain Communi - 3300 units - Both
Alpha-Murrain Communi - 1900 units - Both

name	N & W Creek Transportation Corp
industry	Transportation
phone_fax	null
annual_revenue	null

OPPORTUNITIES

There is no way to try a subreport on the server, you can only try the whole report.

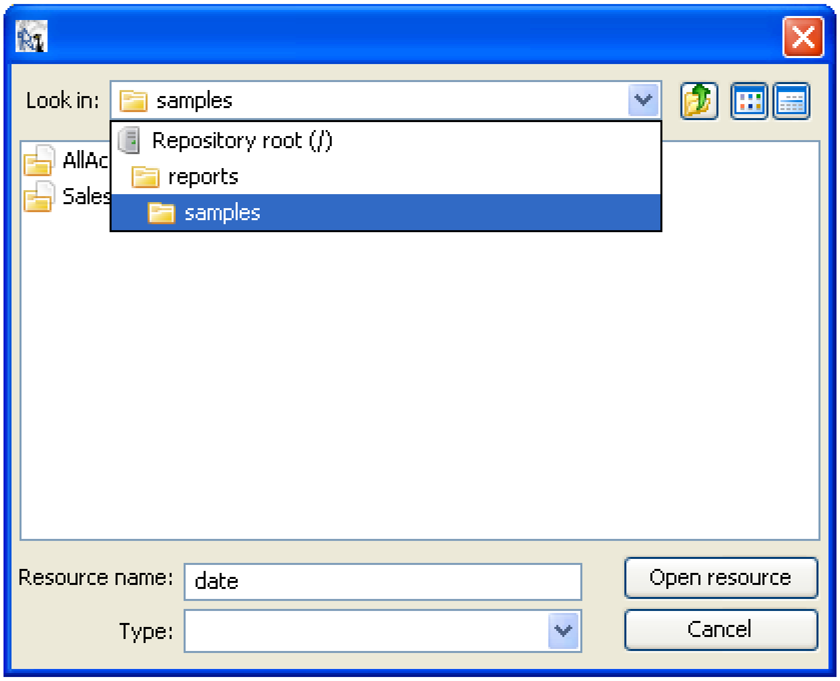
Edit the main jrxml and drag into the detail band the subreport jrxml entry. A subreport element will be created. The main report uses a JDBC connection so you have to set the connection expression of the subreport to $P{REPORT_CONNECTION} in the element properties dialog, subreport tab.

6.4.11 Resources, input controls, data types, lists of values and data sources

The plugin provides a way to manage resources like images, jars, jrxml files, properties file for localizable reports, etc..., input controls, data types, lists of values and data sources. The user interface provided by iReport to deal with these objects is similar to the one provided by the web UI. Refer to the sections below for the meaning of each field requested creating or managing a specific resource.

6.4.12 Repository resource chooser

To locate a resource in the repository (i.e. defining a JasperServer Report that uses a JRXML already stored on the repository), the plugin provides a special component similar to a file chooser dialog.



Use it to easily navigate the repository and pick the resource your want.

6.5 Adding resources directly to the repository

The example in this document shows how to create a complete report and place it into the repository. You can also use the Add New button in the repository browser to configure the following individual resources and add them to the repository for future use:

- Data Source
- Query
- Input Control
- Data Type
- JRXML
- Image
- Font
- JAR
- Resource Bundle

To add any of these resources to the repository:

1. Navigate to the folder where you want to put the resource.
2. Choose the resource type from the drop down list next to the Add button.
3. Click Add
4. Follow the steps in the resulting screen(s).
5. When you complete the screen(s), click Save. The resource is added to the repository.

6.6 Security Maintenance

6.6.1 Overview

JasperServer has an integrated security framework that operates on several levels, namely:

- Authentication: identifying users
- Authorization
    - Access to screens
    - Menu options
    - Objects in the metadata repository

6.6.1.1 Authentication

In order to customize the user experience and restrict access to resources in the system, users need to be authenticated to identify them and their user profiles, including their roles within the system. JasperServer tracks users with its metadata repository. The metadata repository can be the authentication source, i.e., Definition and management of user passwords, or we can also authenticate against a variety of authentication services, such as LDAP (i.e., Microsoft Active Directory, Novell eDirectory). The open source Acegi security framework that is used has many configurable options for:

- Single sign-on: JA-SIG's Central Authentication Service (CAS)
- Java Authentication and Authorization Service (JAAS)
- Container security – Tomcat, Jetty
- SiteMinder

Encryption and authentication standards are also supported.

- HTTPS: including requiring HTTPS
- HTTP Basic
- HTTP Digest
- X509

The Acegi framework is readily extensible to integrate with custom and commercial authentication services and transports.

Authentication occurs by default through the Web user interface, forcing login, and also through HTTP Basic authentication for Web Services, such as iReport and for XML/A traffic. JasperServer will synchronize automatically with an external authentication service without the external users needing to be created in JasperServer first. Roles passed in with the externally authenticated user are included in the JasperServer metadata.

6.6.1.2 Authorization

With the user's identity and roles established, JasperServer controls what the user can do and see in a variety of ways.

- Menu options

  Menu options appear depending on the user's roles. Normal users do not have access to administrative functions. This can be modified through reconfiguration. See the Developer's Guide for details.

- Screen access

Individual screens can be blocked for access by people with given roles. For example, the default security framework only allows administrators access to repository maintenance screens. This configuration of what screens are available for what roles is done in configuration files. Assigning appropriate roles to individual users through the administrative user interface configures the precise screen access for users.

- Object level security

  Objects in the repository and folders can have permissions set on them so that users without sufficient rights cannot see and access those resources. For example, you may have management reports that are not for all users. Permissions can be set to block access for only users with a management role. This filtering capability is used regardless of whether the user is going through the Web user interface or Web Services. See Setting Permissions below.

6.6.2 Creating a role

Roles are sets of permissions that you can assign to users.

Note: *Permissions can be assigned only on reports and other resources. You cannot control access to folders*.

To create a role:

1. Click the Role menu.
2. Type the name of the role in the Role Name field.
3. Click *Save*.

6.6.3 Creating a user

To create a user:

1. Click the Users menu.
2. Fill in the User name, Full Name, Email Address, Password, and Confirm Password fields.
3. Click Enabled to allow this user to log in.
4. Select the roles that you want to assign to this user (ROLE_USER will be assigned by default).
5. Click Save.

6.6.3.1 Changing role assignments for existing users

1. Click the Roles menu.
2. Select the role that you want to apply to an existing user.
3. For users that you want to add to this this role, click the arrow and move them from the *Available Users* list into the *Users in this role* list.
4. For users that you want to delete from this role, click the arrow and move them from the *Users in this role* list into the *Available Users* list.
6. Click Save.

6.6.4 Setting Permissions

In the repository browser for administrators, the "assign" permissions link is available for resources and users. Selecting this link takes you to:

Role Permissions Screen

Here you can set the level of permissions for the resource or folder for individual roles.

- None set: permissions have not been set for this resource or folder for the role. The permissions in effect for the role will be for the parent folder.
- No access: users with this role will not be able to access this resource or folder
- Read: Users with the role can read but not update the resource
- Administration: Users with the role can read, update and delete the resource

Select the level of permission and Update.

From the Role Permissions screen, the administrator can opt to set permissions at the user level for the resource.

6.7 JasperAnalysis Management

6.7.1 Overview

ROLAPJasperAnalysis lets you analyze data organized into a hierarchical dimensional model, which has cubes and dimensions as its primary entities. In a relational database implementation of online analytical processing (OLAP), the entities reside in relational tables. This is sometimes referred to as Relational OLAP (ROLAP).

JPivotJasperAnalysis is based on an open source ROLAP server called Mondrian. JPivot, an open source web-based OLAP user interface, enables users to display and navigate Mondrian's results. JasperAnalysis Professional includes an improved JPivot user interface (as compared to JPivot or JasperServer Open Source).

MDX queriesThe de facto standard for OLAP query languages is MDX. Many analysis applications use MDX. In a distributed computing environment, XML for Analysis (XML/A) is the standard for accessing OLAP data sources remotely. XML/A uses a web services architecture. It transmits MDX queries using the SOAP protocol.

To implement and maintain JasperAnalysis, perform the following steps:

- Store cube data in a relational database and use an OLAP schema file to define the cubes. Note that using existing transactional databases may be inefficient with large amounts of data. To improve performance, use an extract, transform, and load (ETL) process: extract information from one or more data sources, integrate and transform the data, and load the result into a separate cube database.
- Identify facts or measures (the values to summarize) and dimensions (divisions of the measures – for example, dividing time into weeks, months, and years) in the cube database.
- Define a JasperAnalysis schema, mapping logical facts and dimensions onto the physical database. The JasperAnalysis OLAP engine uses the schema to interpret the database and perform OLAP queries. The JasperAnalysis Schema Workbench enables you to develop, validate, and test schemas against the database. The *JasperAnalysis Technical Guide* gives details of schemas and their options.
- Create an analysis client connection in JasperAnalysis through the web based user interface, with your schema and a defined database connection.The JasperServer/JasperAnalysis repository stores the information you provide to define the analysis client connection.
- Define entry points for analysis of the cube data, known as analysis views in JasperAnalysis. Analysis views enable end users to interact with cubes without having to know query languages, database connections, and other technical details. For administrators, an analysis view is an MDX query that is run against the cube as a starting point for interactive analysis. Analysis views appear as options that users can select through the web-based user interface. You can use the Schema Workbench to develop and test MDX queries.
- Plan for updates to data in the cube, usually on a regular basis (for example, nightly or weekly). After an update, flush (empty) the JasperAnalysis data cache, which the server maintains to improve performance, so that subsequent queries get the results of new data.

- Design your environment for scalability and availability. JasperAnalysis can run on a single machine. However, for large numbers of users, large data volumes, and high availability, you can use a multi-machine environment with load balancers and some machines dedicated to either OLAP user interface or OLAP server duties. You can use XML/A to distribute processing.

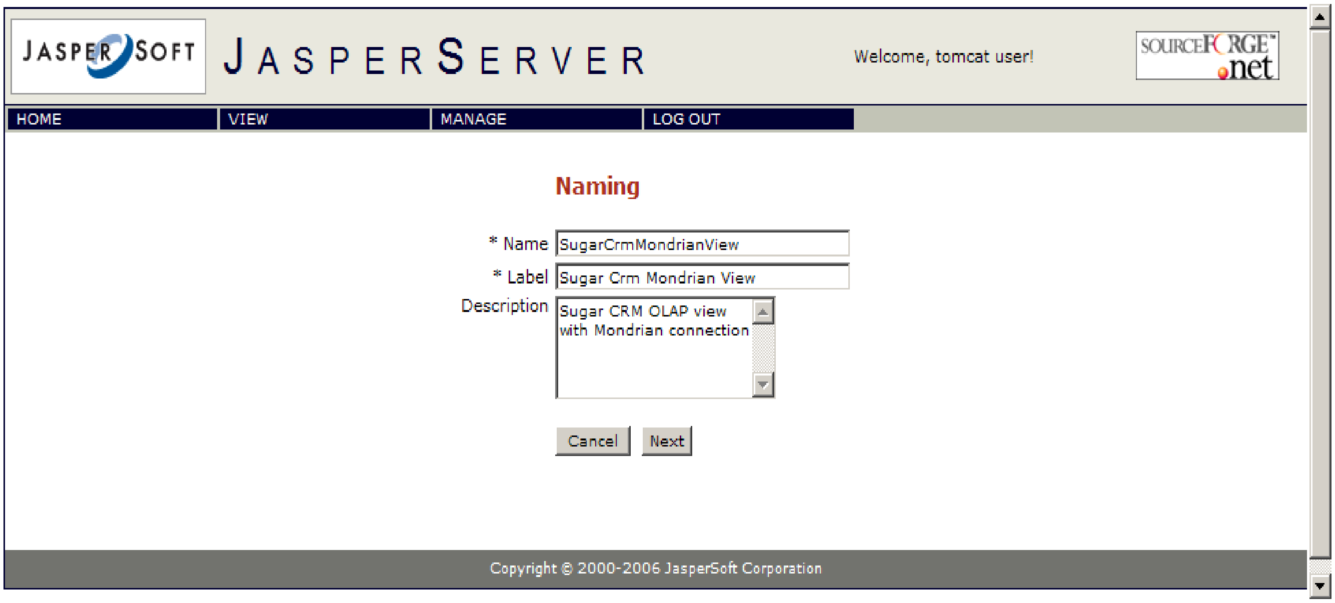
The detailed procedures beginning in the following section step you through the process of setting up all components of an analysis view.

6.7.2 Creating analysis view with a Mondrian connection

An analysis view can be created with a Mondrian connection, one of two types of OLAP connection, plus a MDX query. Follow the example below to create an analysis view using a Mondrian connection:

1. Select the REPOSITORY menu item from the MANAGE menu.
2. Click the *olap*, and then the *views* folders under the Name column
3. Select *Analysis View* from dropdown list next to the Add New button.
4. Click the Add New button.

6.7.2.1 Naming

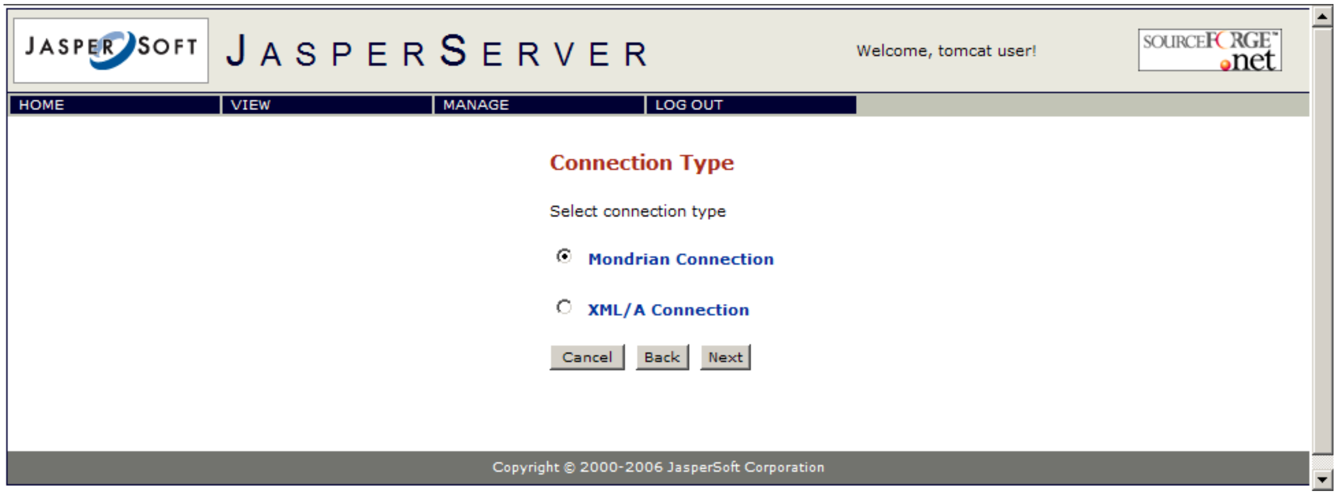


Naming

* Name: SugarCrmMondrianView
* Label: Sugar Crm Mondrian View
Description: Sugar CRM OLAP view with Mondrian connection

5. Enter the naming for the view, e.g., Name=*SugarCrmMondrianView*, Label=*Sugar Crm Mondrian View*, Description=*Sugar CRM OLAP view with Mondrian connection.*
6. Click the Next button.

6.7.2.2 Connection type



Connection Type

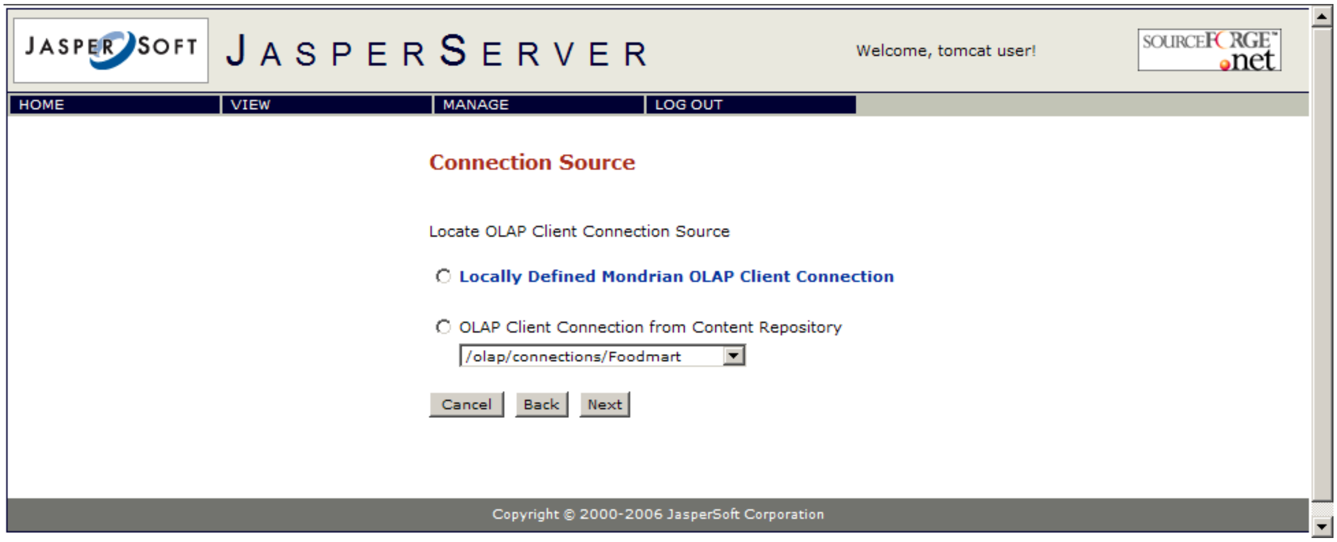
Select connection type

● Mondrian Connection

○ XML/A Connection

7. Select the *Mondrian Connection*.

6.7.2.3 Connection source



Connection Source

Locate OLAP Client Connection Source

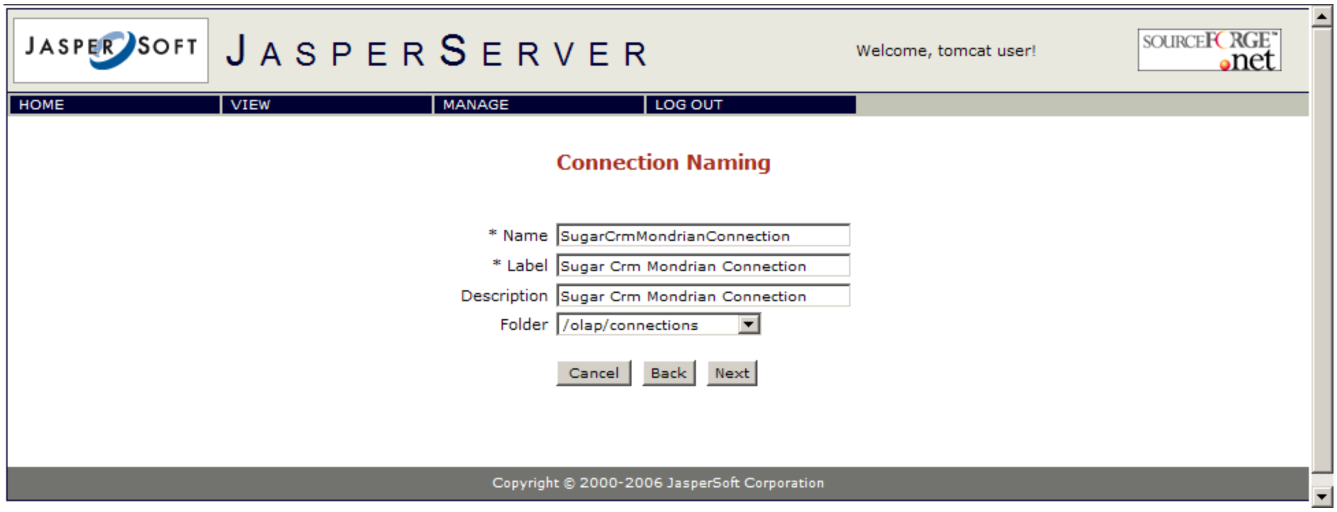
○ Locally Defined Mondrian OLAP Client Connection

○ OLAP Client Connection from Content Repository

/olap/connections/Foodmart

8. Select *Locally Defined Mondrian OLAP Client Connection*.

6.7.2.4 Connection naming



Connection Naming

* Name SugarCrmMondrianConnection

* Label Sugar Crm Mondrian Connection

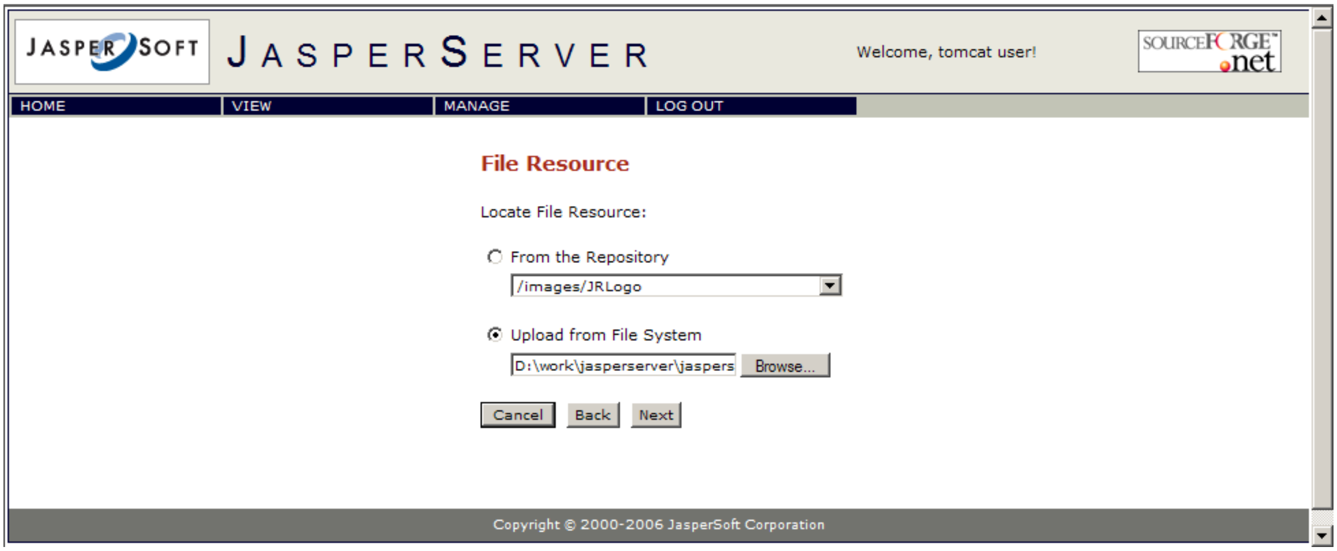
Description Sugar Crm Mondrian Connection

Folder /olap/connections

9. Enter the naming for the new connection, e.g., Name=*SugarCrmMondrianConnection*, Label=*Sugar Crm Mondrian Connection*, Description=*Sugar CRM Mondrian Connection*. and Folder=*/olap/connections*.

10. Click the Next button.

6.7.2.5 Schema



File Resource

Locate File Resource:

○ From the Repository

/images/JRLogo

● Upload from File System

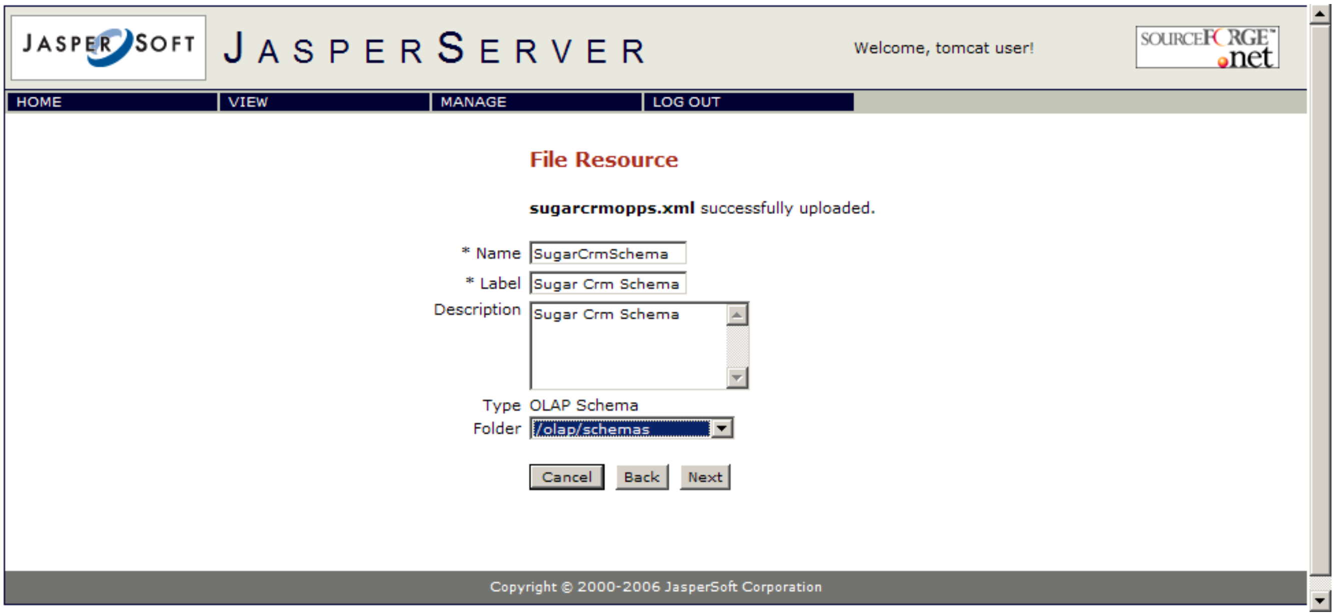
D:\work\jasperserver\jaspers

11. Click the Browse… button under Upload from File System to load the schema file from the file system.

12. Navigate to the *SugarCRMOpps.xml* schema file, e.g.,*c:/svn/jasperi/jasperserver-unit-test/src/test/resources/olap/schemas*, and click the Open button.

13. click the Next button.

6.7.2.6 Schema naming



File Resource

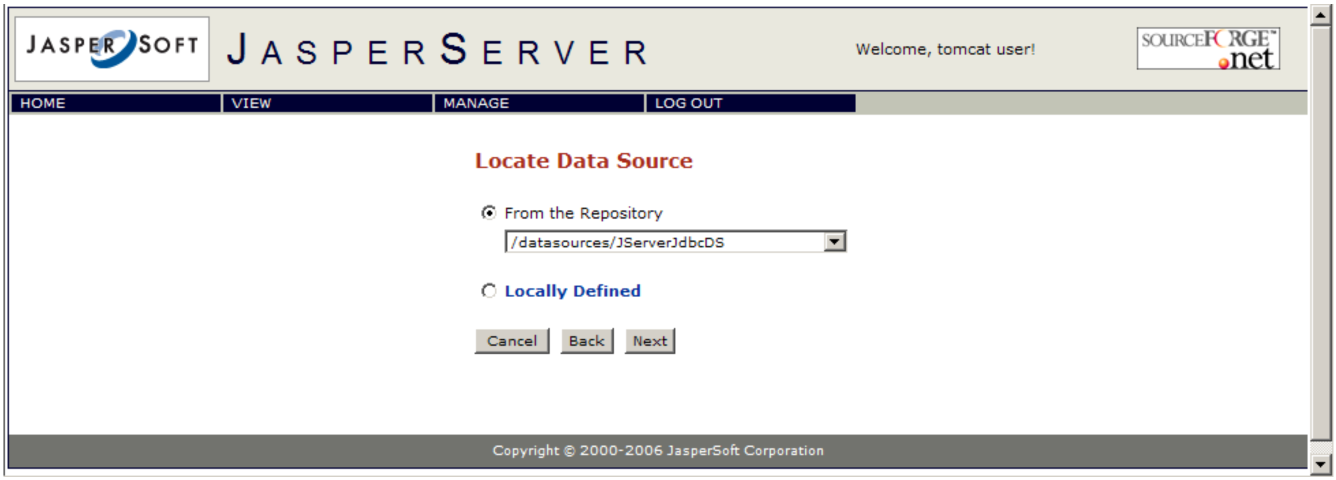
sugarcrmopps.xml successfully uploaded.

* Name: SugarCrmSchema
* Label: Sugar Crm Schema
Description: Sugar Crm Schema
Type: OLAP Schema
Folder: /olap/schemas

14. Enter the naming for the new schema, e.g., Name=*SugarCrmSchema*, Label=*Sugar Crm Schema*, Description=*Sugar CRM Schema.*, and Folder=*/olap/schemas*.

15. Click the Next button.

6.7.2.7 Locate data source



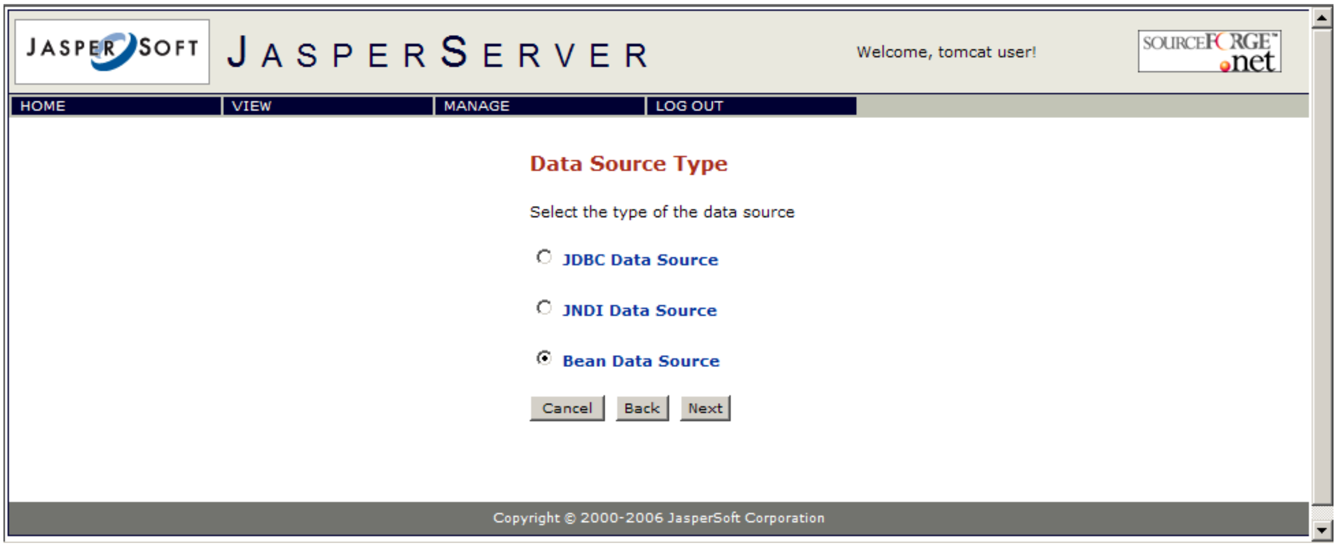
Locate Data Source

● From the Repository
/datasources/JServerJdbcDS

○ Locally Defined

16. Select *Locally Defined*.

6.7.2.8 Data source type



Data Source Type

Select the type of the data source

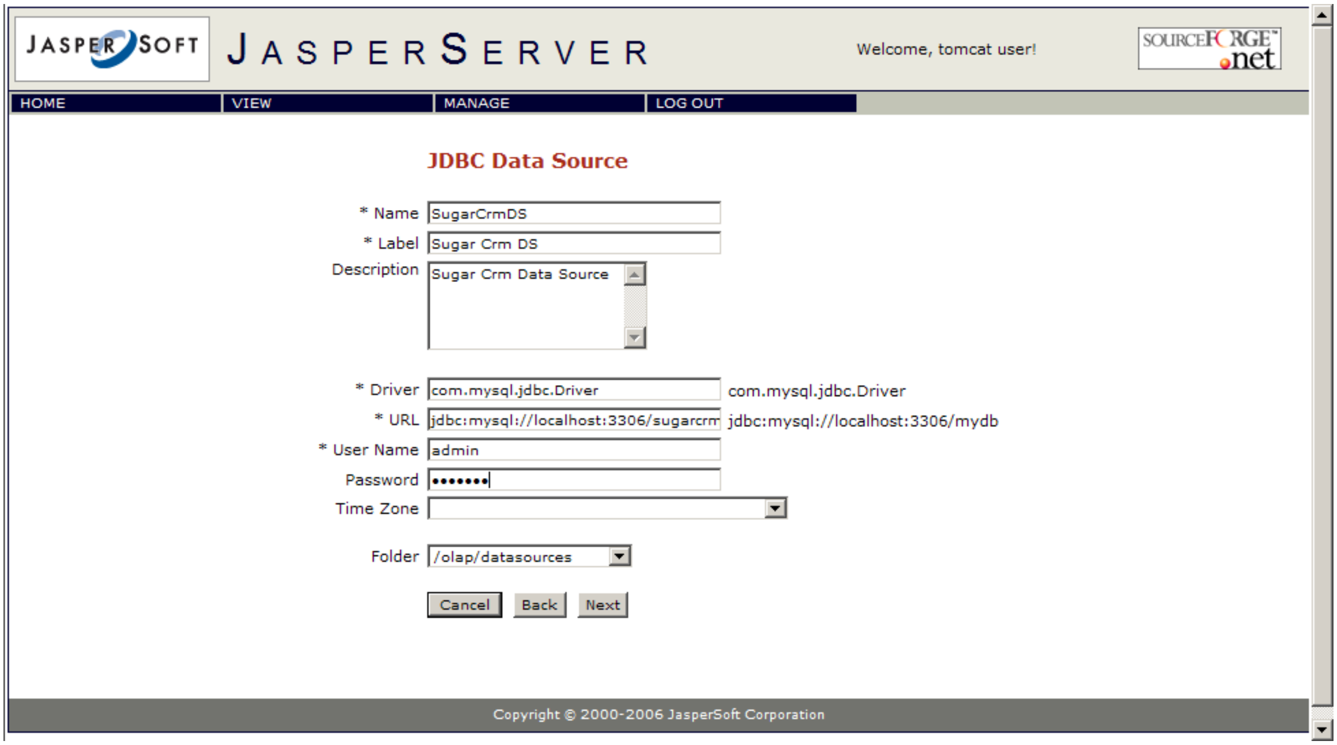
○ JDBC Data Source

○ JNDI Data Source

● Bean Data Source

17. Select *JDBC Data Source*.

6.7.2.9 JDBC data source



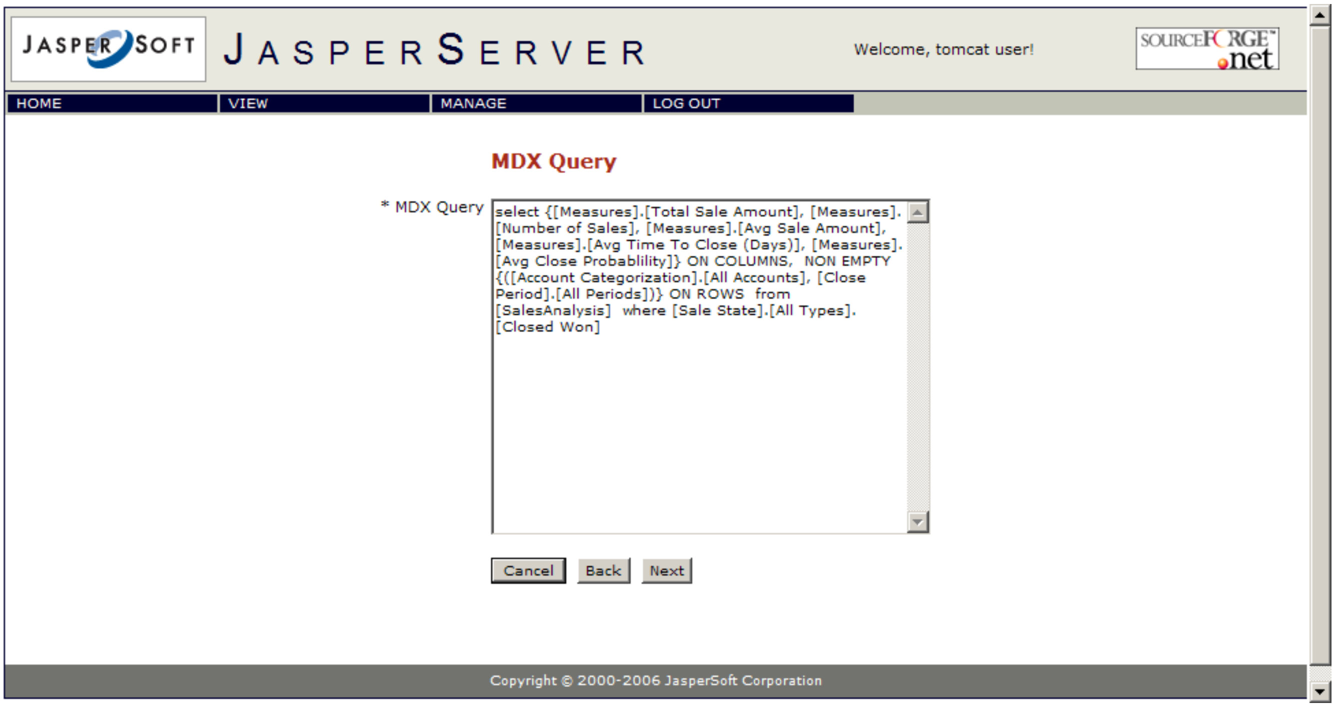
JDBC Data Source

* Name	SugarCrmDS	
* Label	Sugar Crm DS	
Description	Sugar Crm Data Source	
* Driver	com.mysql.jdbc.Driver	com.mysql.jdbc.Driver
* URL	jdbc:mysql://localhost:3306/sugarcrm	jdbc:mysql://localhost:3306/mydb
* User Name	admin	
Password	•••••••	
Time Zone		
Folder	/olap/datasources	

18. Enter the naming and JDBC connection information for the data source, e.g., Name=*SugarCrmDS*, Label=*Sugar Crm DS*, Description=*Sugar CRM Data Source*, Driver=*com.mysql.jdbc.Driver*, URL=*jdbc:mysql://localhost:3306/sugarcrm*, Username=*root*, Password=<*password for root*>, and Folder=*/olap/datasources* (This example assumes the MySQL database is used and the user *root* exists.)

19. Click the Next button.

6.7.2.10 MDX query



MDX Query

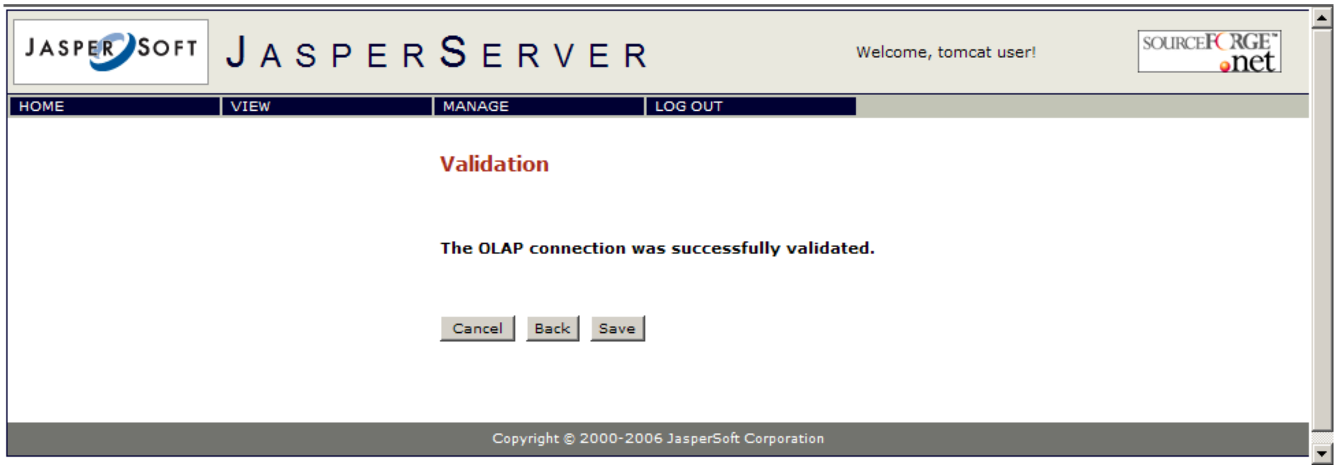
* MDX Query

```
select {[Measures].[Total Sale Amount], [Measures].
[Number of Sales], [Measures].[Avg Sale Amount],
[Measures].[Avg Time To Close (Days)], [Measures].
[Avg Close Probablility]} ON COLUMNS,  NON EMPTY
{([Account Categorization].[All Accounts], [Close
Period].[All Periods])} ON ROWS  from
[SalesAnalysis]  where [Sale State].[All Types].
[Closed Won]
```

20. Enter an MDX Query in the text area, e.g., *select {[Measures].[Total Sale Amount], [Measures].[Number of Sales], [Measures].[Avg Sale Amount], [Measures].[Avg Time To Close (Days)], [Measures].[Avg Close Probablility]} ON COLUMNS, NON EMPTY {([Account Categorization].[All Accounts], [Close Period].[All Periods])} ON ROWS from [SalesAnalysis] where [Sale State].[All Types].[Closed Won].*

21. Click the Next button.

6.7.2.11 Validation

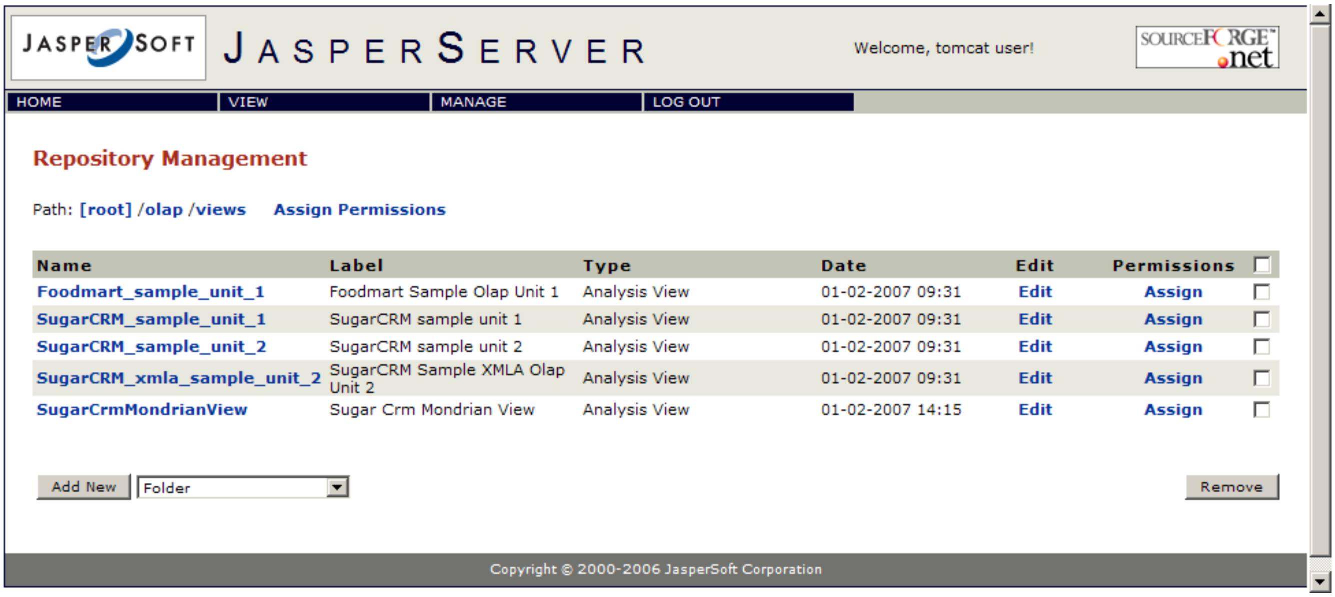


Validation

The OLAP connection was successfully validated.

22. Click the Save button to save the new view.

The newly created analysis view is listed in the view list under *[root] /olap /views*. To interact with the analysis view, click *SugarCrmMondrianView*.



Repository Management

Path: [root] /olap /views Assign Permissions

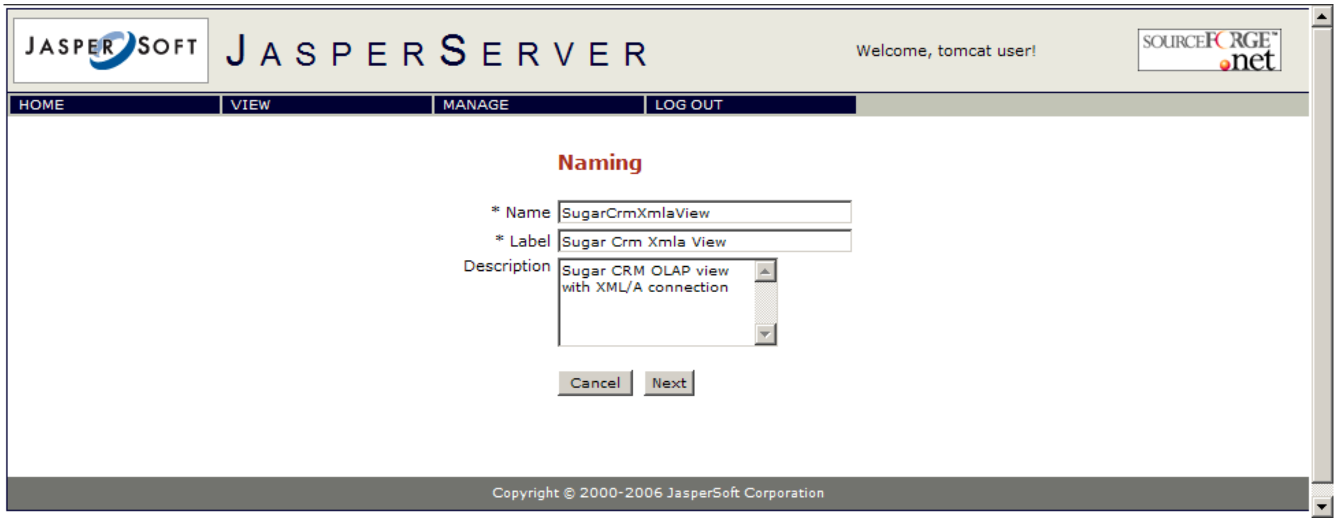
Name	Label	Type	Date	Edit	Permissions	□
Foodmart_sample_unit_1	Foodmart Sample Olap Unit 1	Analysis View	01-02-2007 09:31	Edit	Assign	□
SugarCRM_sample_unit_1	SugarCRM sample unit 1	Analysis View	01-02-2007 09:31	Edit	Assign	□
SugarCRM_sample_unit_2	SugarCRM sample unit 2	Analysis View	01-02-2007 09:31	Edit	Assign	□
SugarCRM_xmla_sample_unit_2	SugarCRM Sample XMLA Olap Unit 2	Analysis View	01-02-2007 09:31	Edit	Assign	□
SugarCrmMondrianView	Sugar Crm Mondrian View	Analysis View	01-02-2007 14:15	Edit	Assign	□

6.7.3 Creating analysis view with an XML/A connection

An analysis view can be created with an XML/A connection, one of two types of OLAP connection, plus a MDX query. Follow the example below to create an analysis view using an XML/A connection:

1. Select the REPOSITORY menu item from the MANAGE menu.
2. Click the *olap*, and then the *views* folders under the Name column
3. Select *Analysis View* from dropdown list next to the Add New button.
4. Click the Add New button.

6.7.3.1 Naming



Naming

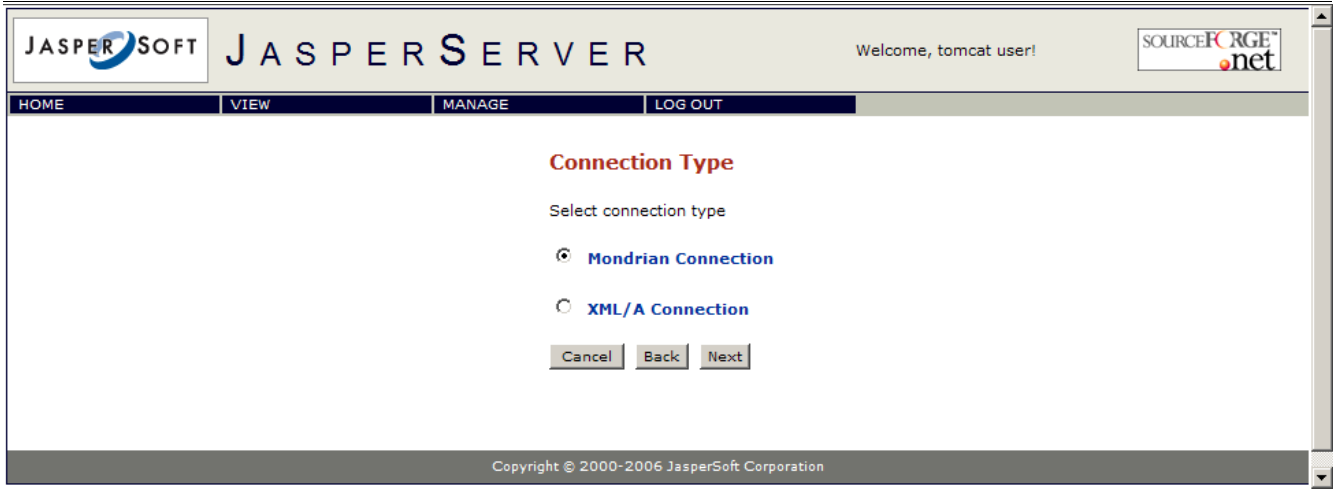
* Name: SugarCrmXmlaView

* Label: Sugar Crm Xmla View

Description: Sugar CRM OLAP view with XML/A connection

5. Enter the naming for the view, e.g., Name=*SugarCrmXmlaView*, Label=*Sugar Crm Xmla View*, Description=*Sugar CRM OLAP view with XML/A connection.*
6. Click the Next button.

6.7.3.2 Connection type



Connection Type

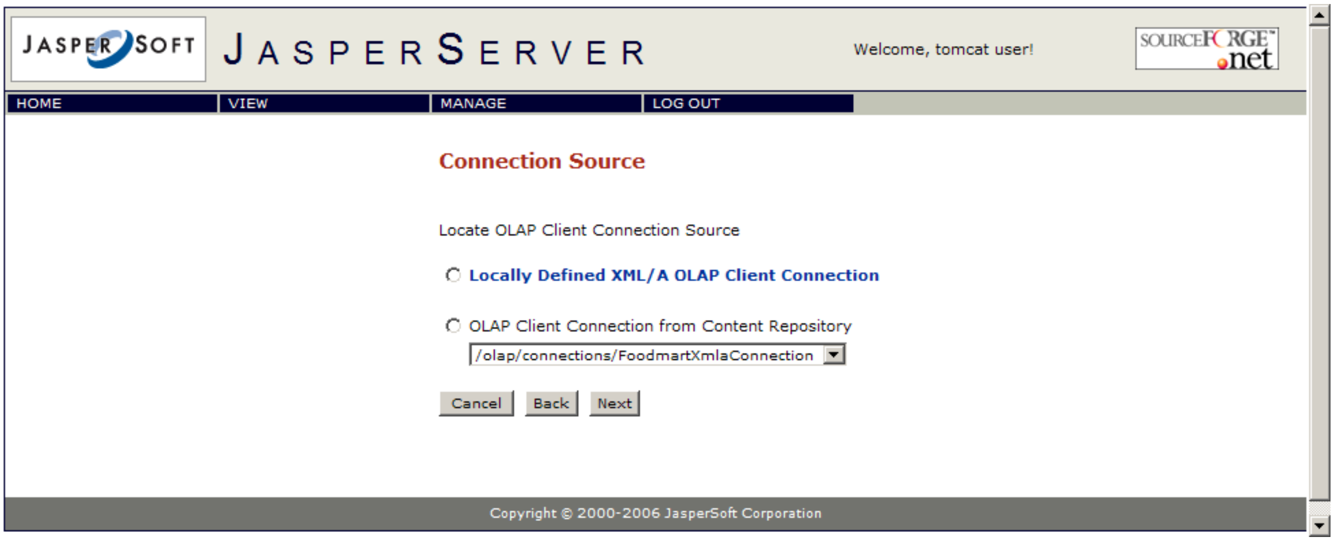
Select connection type

● Mondrian Connection

○ XML/A Connection

7. Click the *XML/A Connection*.

6.7.3.3 Connection source



Connection Source

Locate OLAP Client Connection Source

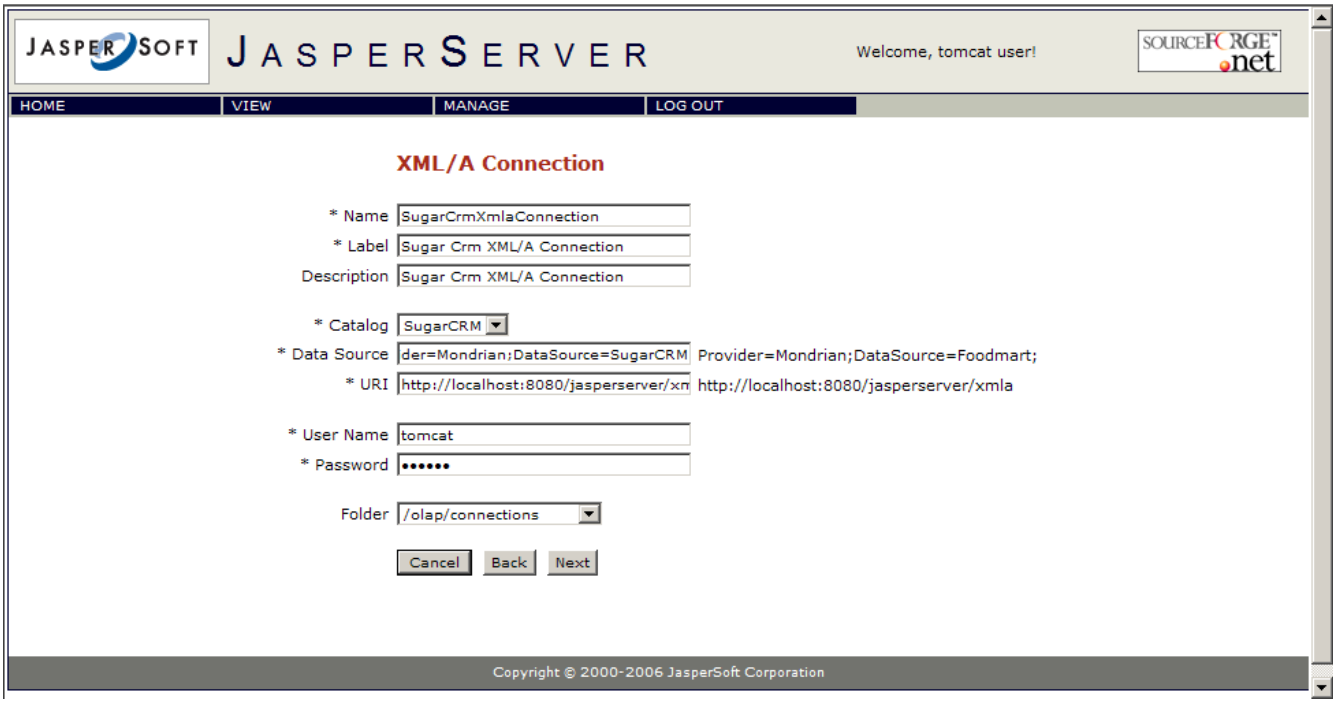
○ Locally Defined XML/A OLAP Client Connection

○ OLAP Client Connection from Content Repository

/olap/connections/FoodmartXmlaConnection

8. Click *Locally Defined XML/A OLAP Client Connection*.

6.7.3.4 XML/A connection



XML/A Connection

* Name: SugarCrmXmlaConnection
* Label: Sugar Crm XML/A Connection
Description: Sugar Crm XML/A Connection

* Catalog: SugarCRM
* Data Source: der=Mondrian;DataSource=SugarCRM Provider=Mondrian;DataSource=Foodmart;
* URI: http://localhost:8080/jasperserver/xm http://localhost:8080/jasperserver/xmla

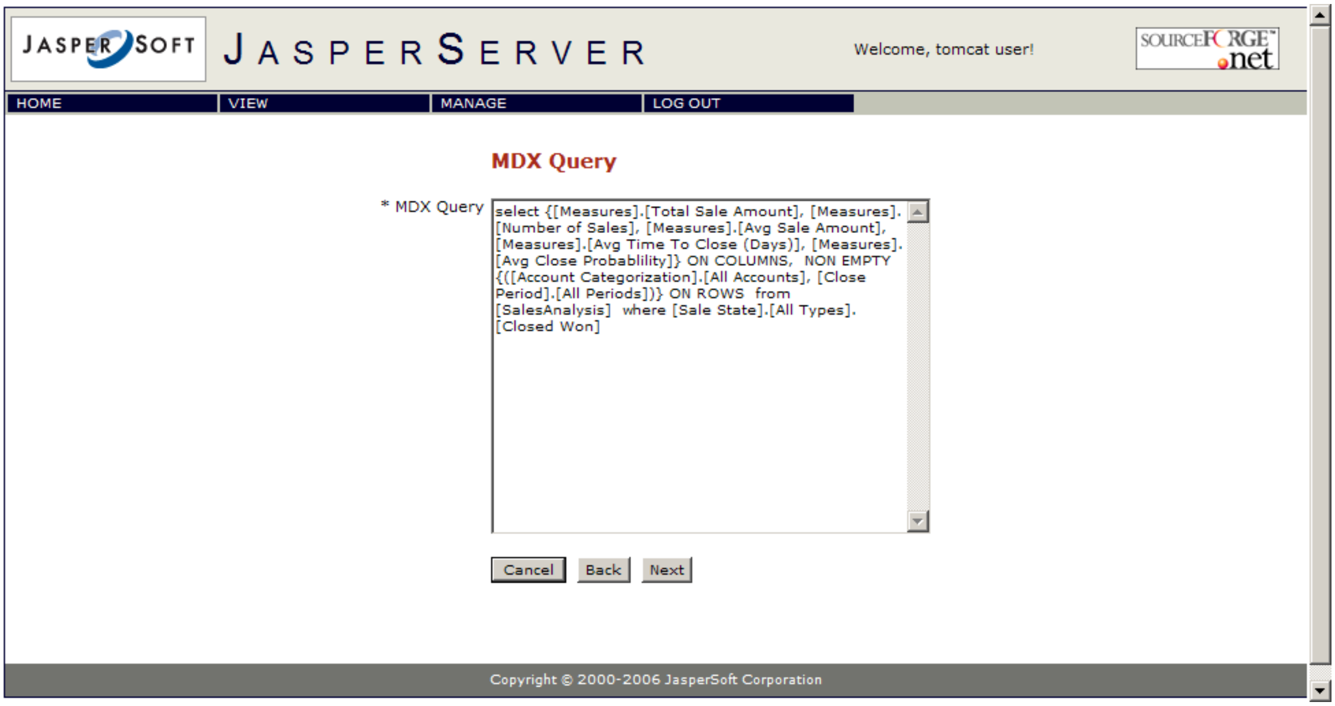
* User Name: tomcat
* Password: ••••••

Folder: /olap/connections

9. Enter the naming for the new connection, e.g., Name=*SugarCrmXmlaConnection*, Label=*Sugar Crm Xmla Connection*, Description=*Sugar CRM XML/A Connection.*, Catalog=*SugarCRM*, Data Source=*Provider=Mondrian;DataSource=SugarCRM;*, URI=*http://localhost:8080/jasperserver/xmla*, Username=*tomcat*, Password=<password for *tomcat*>, and Folder=*/olap/connections*. Note: beginning with JasperAnalysis 2.0.1, Username and Password are no longer required when defining an XML/A Connection. If either of these fields is left blank, the connection will take the current user's information to supply in the XML/A Request, which will be authenticated by the XML/A Server.

10. Click the Next button.

6.7.3.5 MDX query



MDX Query

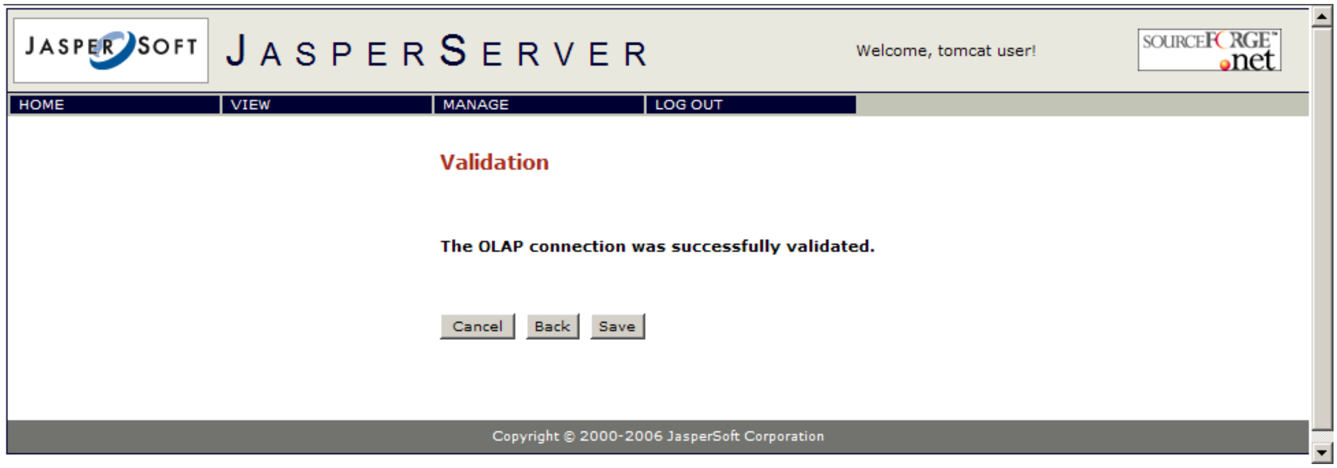
* MDX Query

```
select {[Measures].[Total Sale Amount], [Measures].
[Number of Sales], [Measures].[Avg Sale Amount],
[Measures].[Avg Time To Close (Days)], [Measures].
[Avg Close Probablility]} ON COLUMNS,  NON EMPTY
{([Account Categorization].[All Accounts], [Close
Period].[All Periods])} ON ROWS  from
[SalesAnalysis]  where [Sale State].[All Types].
[Closed Won]
```

11. Enter an MDX Query in the text area, e.g., *select {[Measures].[Total Sale Amount], [Measures].[Number of Sales], [Measures].[Avg Sale Amount], [Measures].[Avg Time To Close (Days)], [Measures].[Avg Close Probablility]} ON COLUMNS, NON EMPTY {([Account Categorization].[All Accounts], [Close Period].[All Periods])} ON ROWS from [SalesAnalysis] where [Sale State].[All Types].[Closed Won].*

12. Click the Next button.

6.7.3.6 Validation



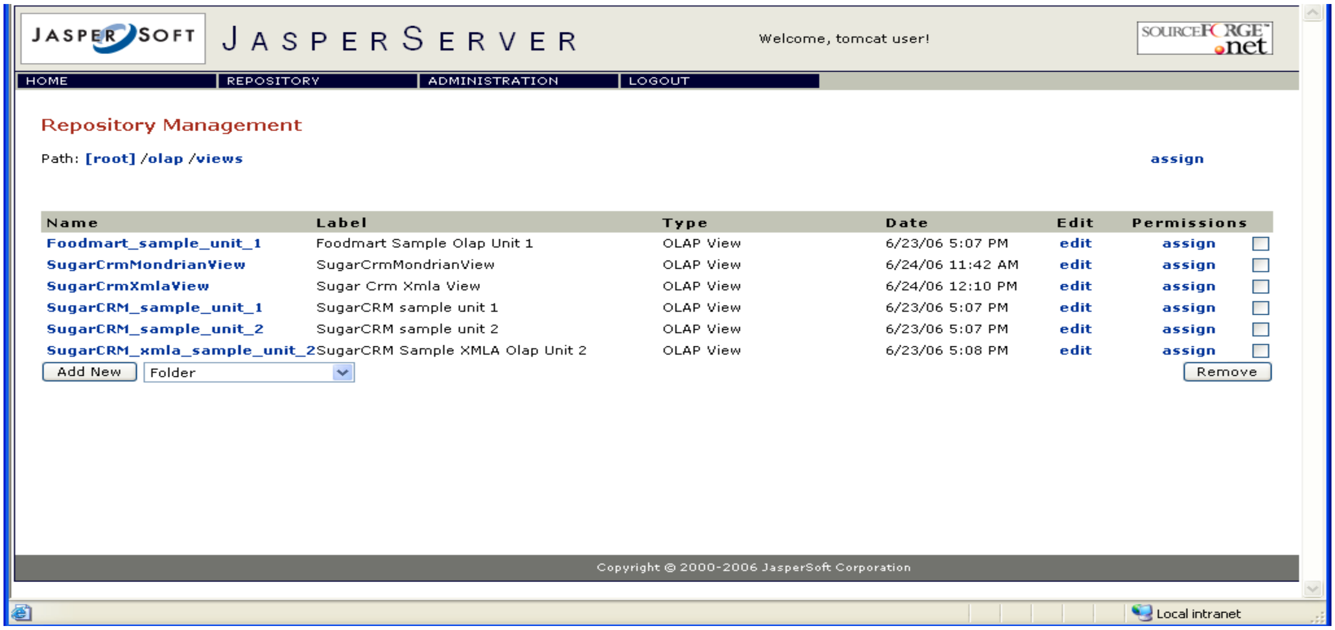
Validation

The OLAP connection was successfully validated.

Copyright © 2000-2006 JasperSoft Corporation

13. Click the Save button to save the new view.

The newly created OLPA view is listed in the view list under *[root] /olap /views*. To interact with the analysis view, click *SugarCrmXmlaView*.



Repository Management

Path: [root] /olap /views

assign

Name	Label	Type	Date	Edit	Permissions	
Foodmart_sample_unit_1	Foodmart Sample Olap Unit 1	OLAP View	6/23/06 5:07 PM	edit	assign	□
SugarCrmMondrianView	SugarCrmMondrianView	OLAP View	6/24/06 11:42 AM	edit	assign	□
SugarCrmXmlaView	Sugar Crm Xmla View	OLAP View	6/24/06 12:10 PM	edit	assign	□
SugarCRM_sample_unit_1	SugarCRM sample unit 1	OLAP View	6/23/06 5:07 PM	edit	assign	□
SugarCRM_sample_unit_2	SugarCRM sample unit 2	OLAP View	6/23/06 5:07 PM	edit	assign	□
SugarCRM_xmla_sample_unit_2	SugarCRM Sample XMLA Olap Unit 2	OLAP View	6/23/06 5:08 PM	edit	assign	□

Copyright © 2000-2006 JasperSoft Corporation

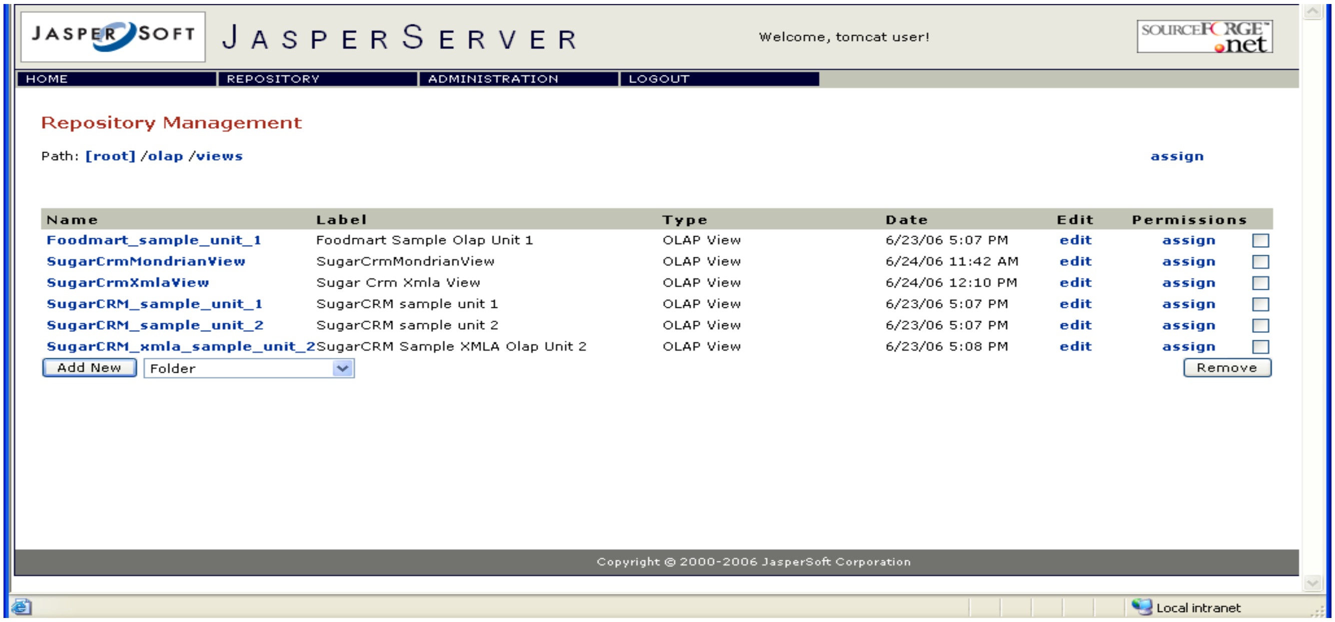
6.7.4 Editing analysis view

You may change an analysis view's naming, connection and associated MDX query. Following the following steps to change an analysis view:

1. Select the REPOSITORY menu item from the MANAGE menu.

2. Click *olap* and *views* under the Name column to display the sample views.

6.7.4.1 Analysis views



Repository Management

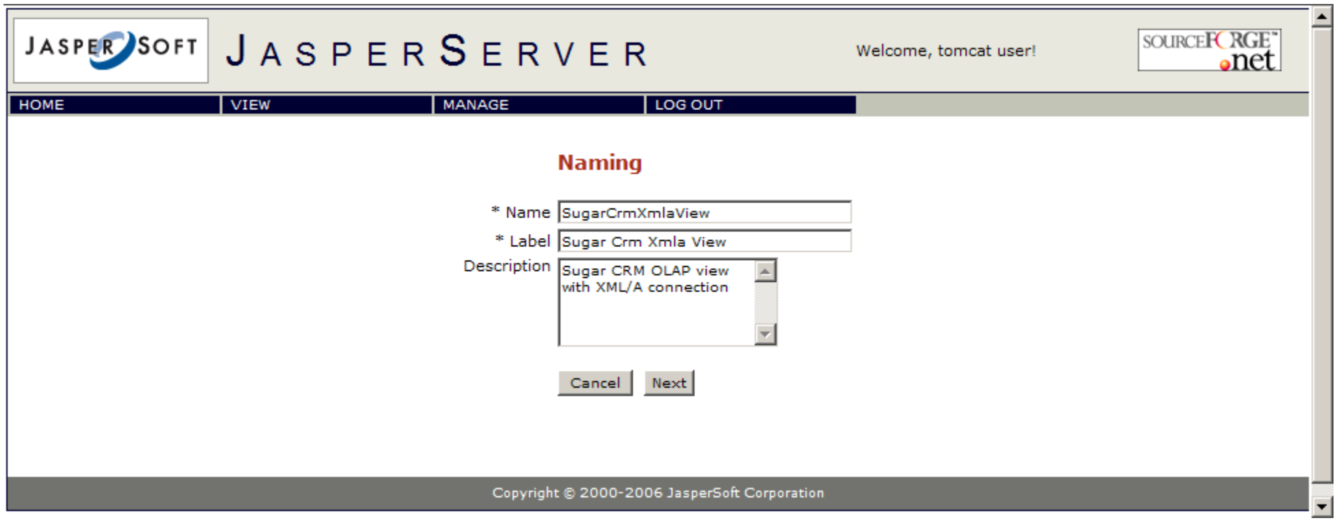
Path: [root] /olap /views

assign

Name	Label	Type	Date	Edit	Permissions	
Foodmart_sample_unit_1	Foodmart Sample Olap Unit 1	OLAP View	6/23/06 5:07 PM	edit	assign	□
SugarCrmMondrianView	SugarCrmMondrianView	OLAP View	6/24/06 11:42 AM	edit	assign	□
SugarCrmXmlaView	Sugar Crm Xmla View	OLAP View	6/24/06 12:10 PM	edit	assign	□
SugarCRM_sample_unit_1	SugarCRM sample unit 1	OLAP View	6/23/06 5:07 PM	edit	assign	□
SugarCRM_sample_unit_2	SugarCRM sample unit 2	OLAP View	6/23/06 5:07 PM	edit	assign	□
SugarCRM_xmla_sample_unit_2	SugarCRM Sample XMLA Olap Unit 2	OLAP View	6/23/06 5:08 PM	edit	assign	□

3. Select an analysis view, e.g., *SugarCrmMondrianView* created in 6.7, and Click the Edit button to the right.

6.7.4.2 Naming



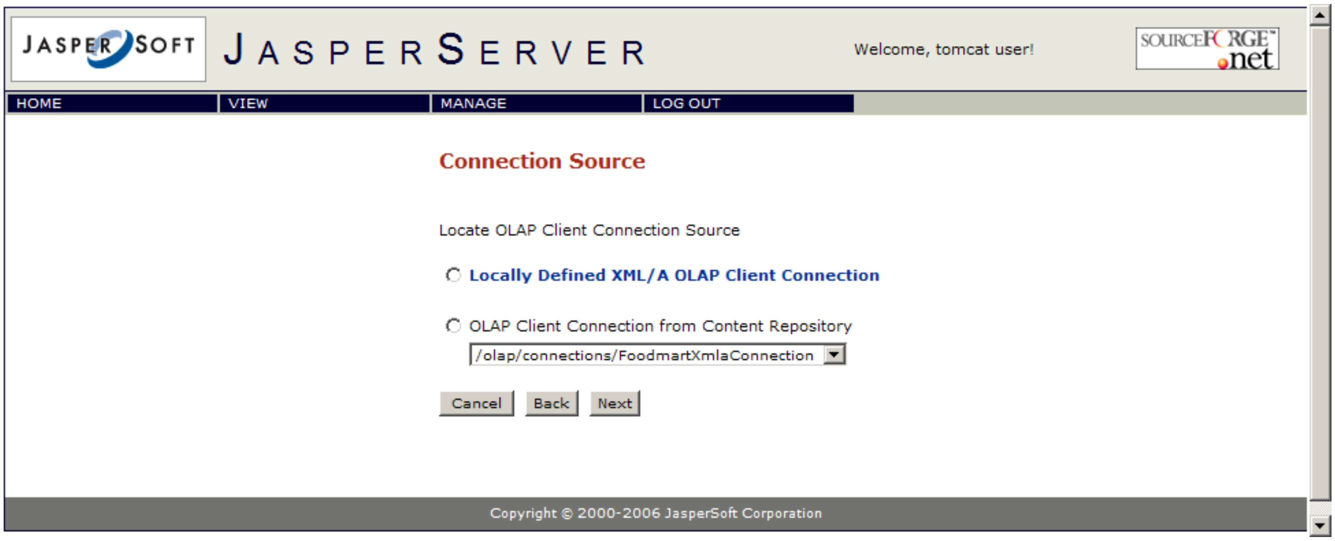
Naming

* Name	SugarCrmXmlaView
* Label	Sugar Crm Xmla View
Description	Sugar CRM OLAP view with XML/A connection

4. Change the label and/or description to reflect the new analysis view. Note: Edit does not allow the Name to be changed.

5. Click the Next button.

6.7.4.3 Connection source



Connection Source

Locate OLAP Client Connection Source

○ Locally Defined XML/A OLAP Client Connection

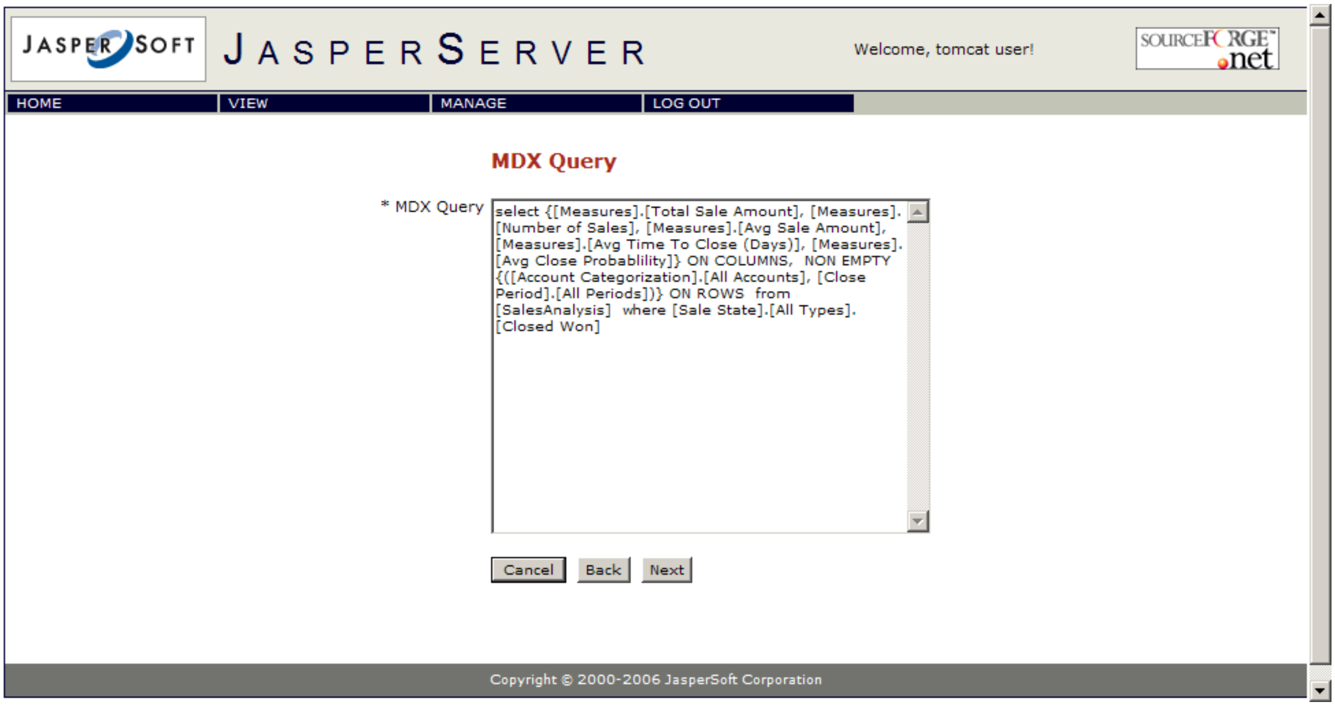
○ OLAP Client Connection from Content Repository

/olap/connections/FoodmartXmlaConnection

Copyright © 2000-2006 JasperSoft Corporation

6. Select a new connection from the dropdown list, e.g., */olap/connections/SugarCrmMondrianConnection*.
7. Click the Next button.

6.7.4.4 MDX query



MDX Query

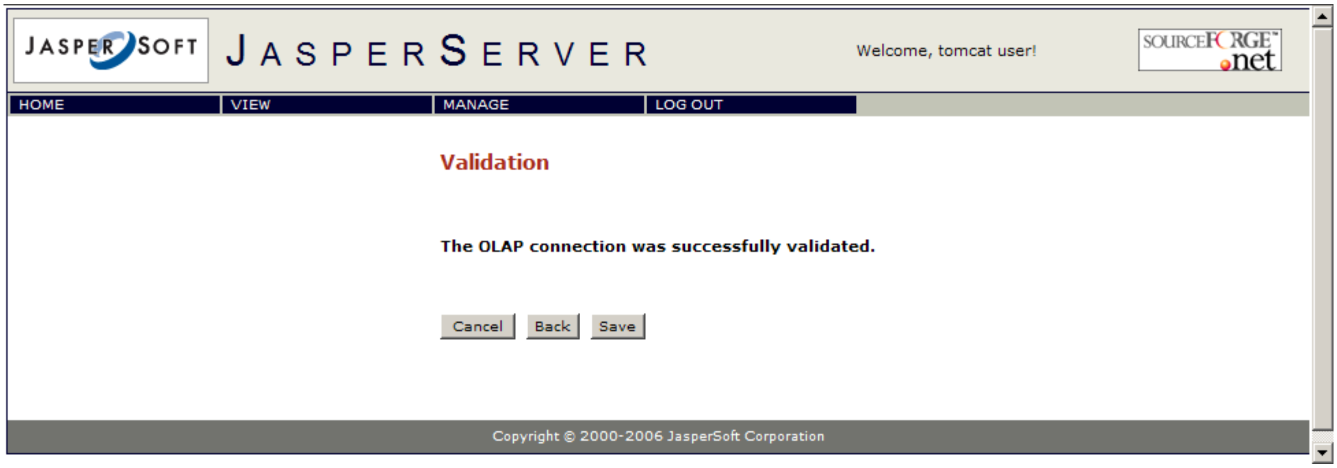
* MDX Query

```
select {[Measures].[Total Sale Amount], [Measures].
[Number of Sales], [Measures].[Avg Sale Amount],
[Measures].[Avg Time To Close (Days)], [Measures].
[Avg Close Probablility]} ON COLUMNS,  NON EMPTY
{([Account Categorization].[All Accounts], [Close
Period].[All Periods])} ON ROWS  from
[SalesAnalysis]  where [Sale State].[All Types].
[Closed Won]
```

Copyright © 2000-2006 JasperSoft Corporation

8. Change the MDX Query accordingly.
9. Click the Next button.

6.7.4.5 Validation



Validation

The OLAP connection was successfully validated.

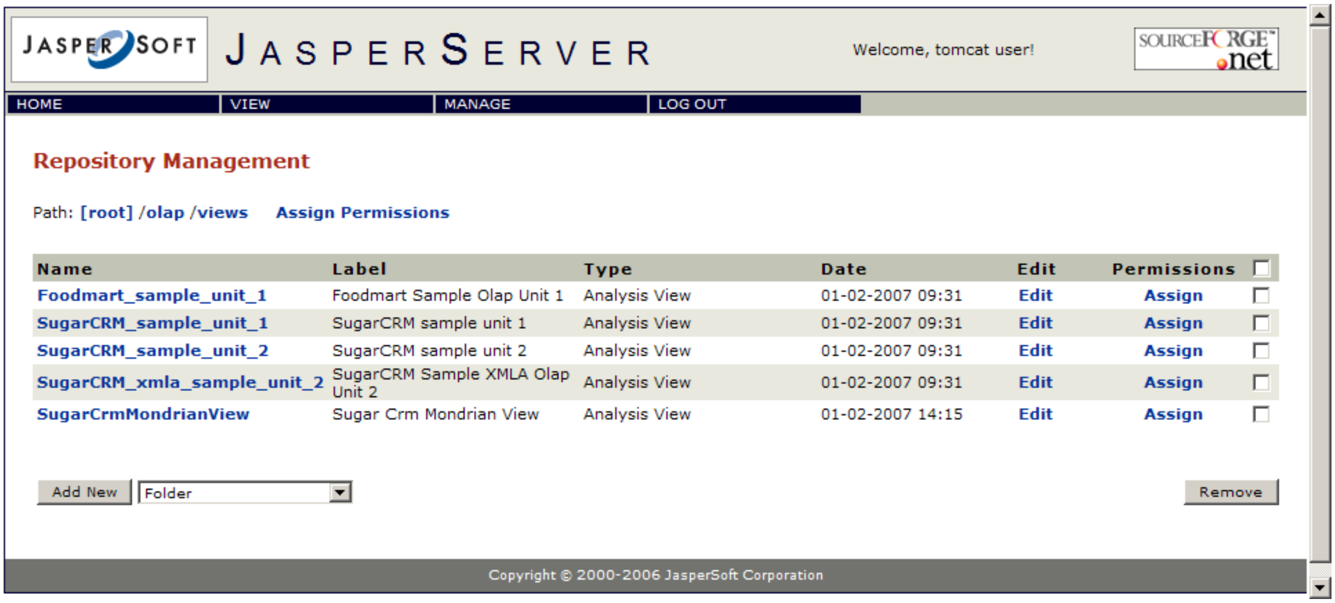
10. Click the 'Save' button to save the changes.

6.7.5 Editing Mondrian connection

You may change a Mondrian connection's naming, schema, and data source. Following the following steps to change a Mondrian connection:

- Select the REPOSITORY menu item from the MANAGE menu.
- Click *olap* and *connections* under the Name column to display the connections in the Repository.

6.7.5.1 Connections



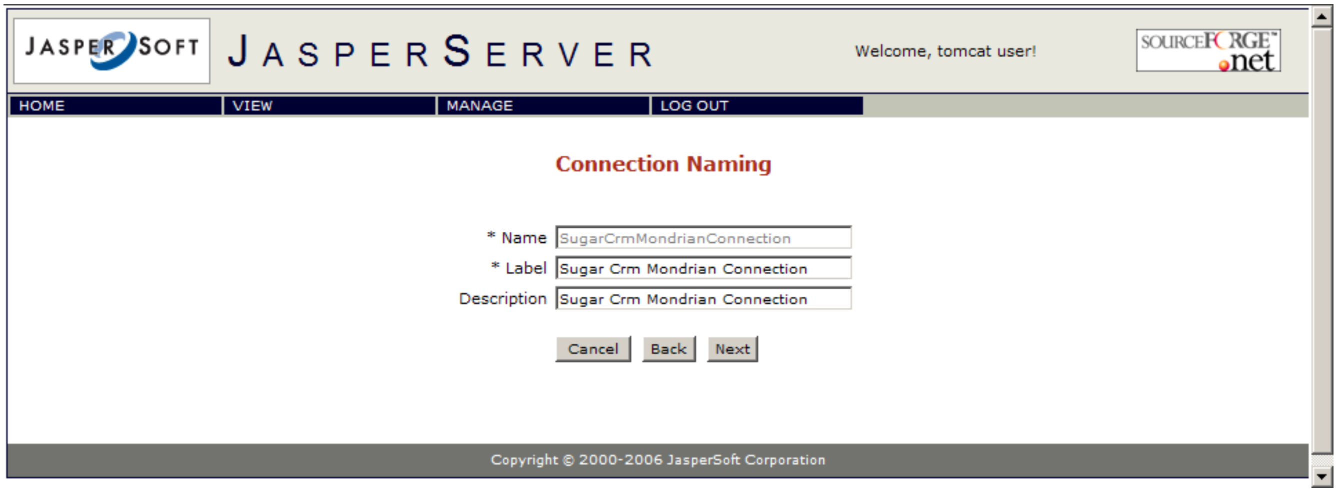
Repository Management

Path: [root] /olap /views   Assign Permissions

Name	Label	Type	Date	Edit	Permissions	□
Foodmart_sample_unit_1	Foodmart Sample Olap Unit 1	Analysis View	01-02-2007 09:31	Edit	Assign	□
SugarCRM_sample_unit_1	SugarCRM sample unit 1	Analysis View	01-02-2007 09:31	Edit	Assign	□
SugarCRM_sample_unit_2	SugarCRM sample unit 2	Analysis View	01-02-2007 09:31	Edit	Assign	□
SugarCRM_xmla_sample_unit_2	SugarCRM Sample XMLA Olap Unit 2	Analysis View	01-02-2007 09:31	Edit	Assign	□
SugarCrmMondrianView	Sugar Crm Mondrian View	Analysis View	01-02-2007 14:15	Edit	Assign	□

- Select a *Mondrian Connection*, e.g., *SugarCrmMondrianConnection* created in 6.7.2.4., and Click the Edit button to the right.

6.7.5.2 *Connection naming*



Connection Naming

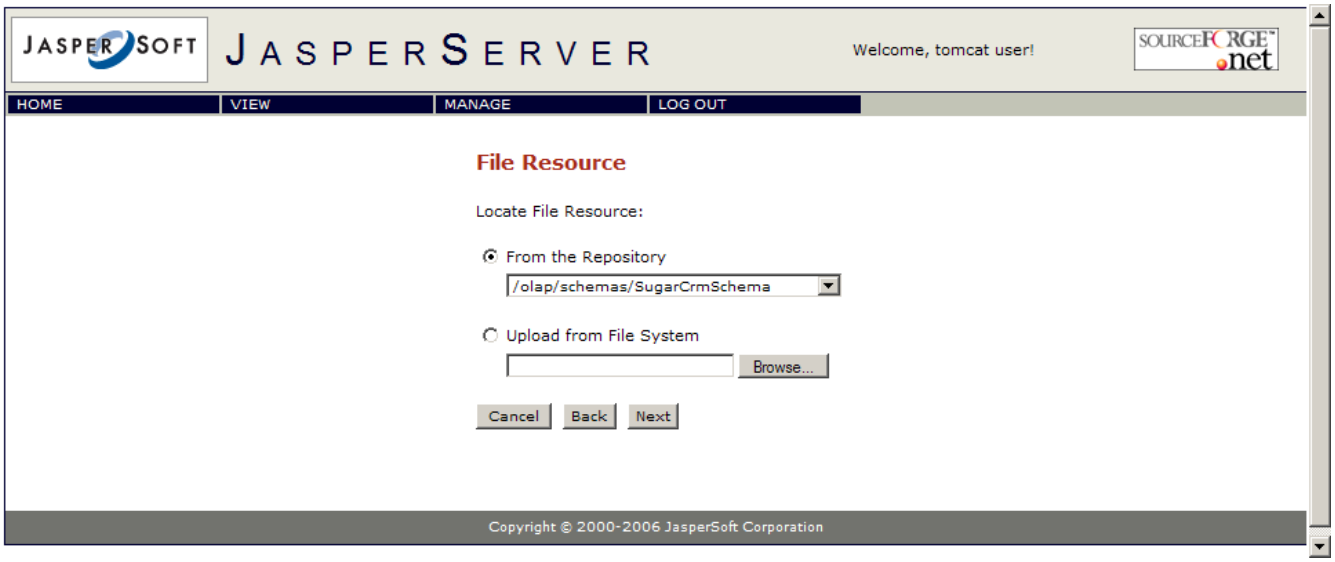
* Name SugarCrmMondrianConnection

* Label Sugar Crm Mondrian Connection

Description Sugar Crm Mondrian Connection

- Change the label and/or description to reflect the new Mondrian connection. Note: Edit does not allow the Name to be changed.
- Click the Next button.

6.7.5.3 *Schema*



File Resource

Locate File Resource:

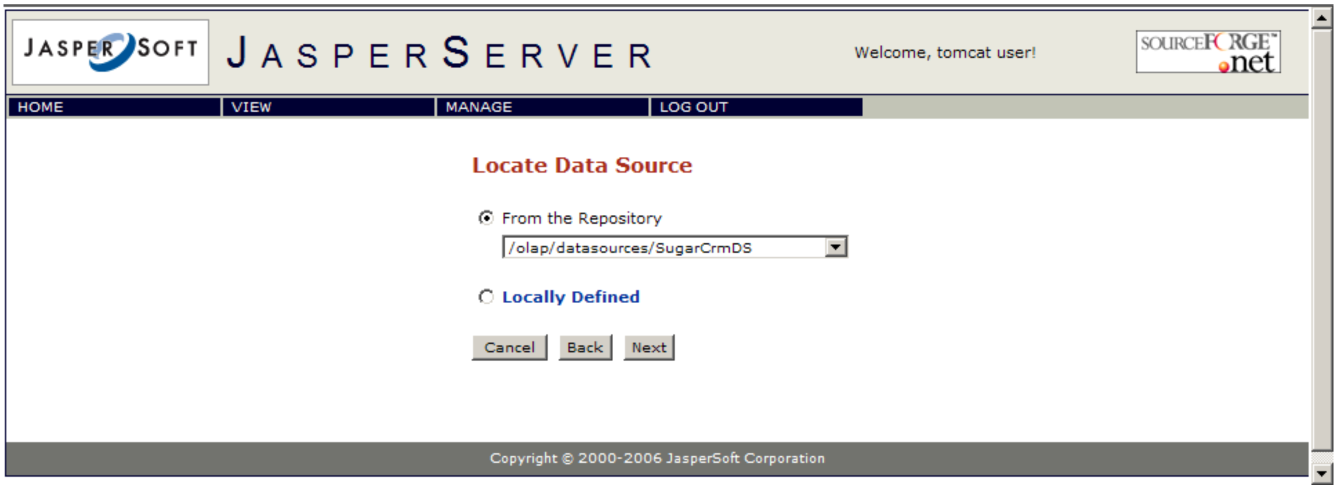
● From the Repository

/olap/schemas/SugarCrmSchema

○ Upload from File System

- Select a schema from the repository.
- Click the Next button to view schema naming.
- Click the Next button to continue.

6.7.5.4 Data source



Locate Data Source

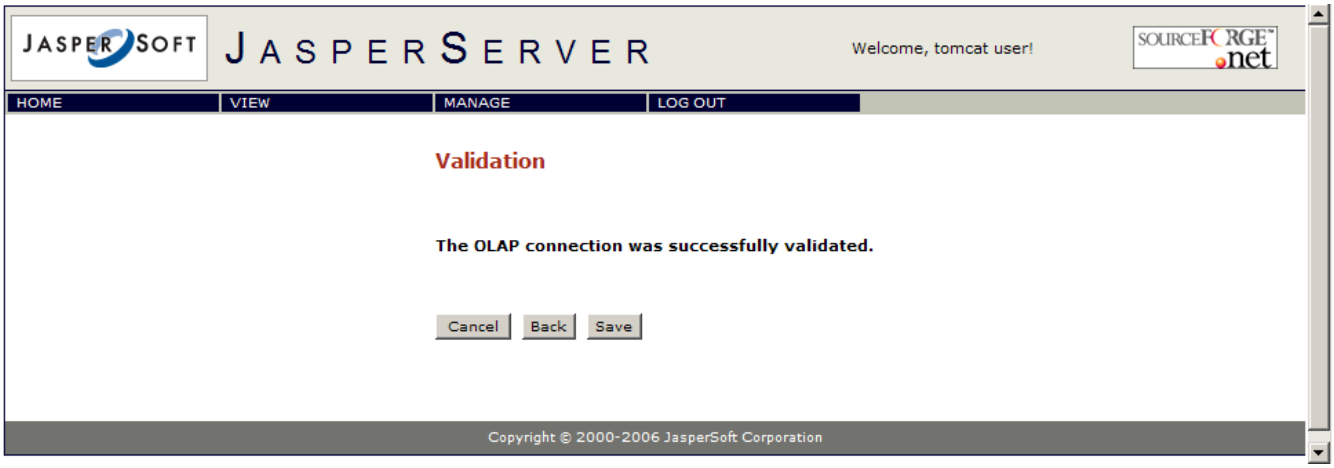
● From the Repository

/olap/datasources/SugarCrmDS

○ Locally Defined

- Select a data source from the repository.
- Click the Next button.

6.7.5.5 Validation



Validation

The OLAP connection was successfully validated.

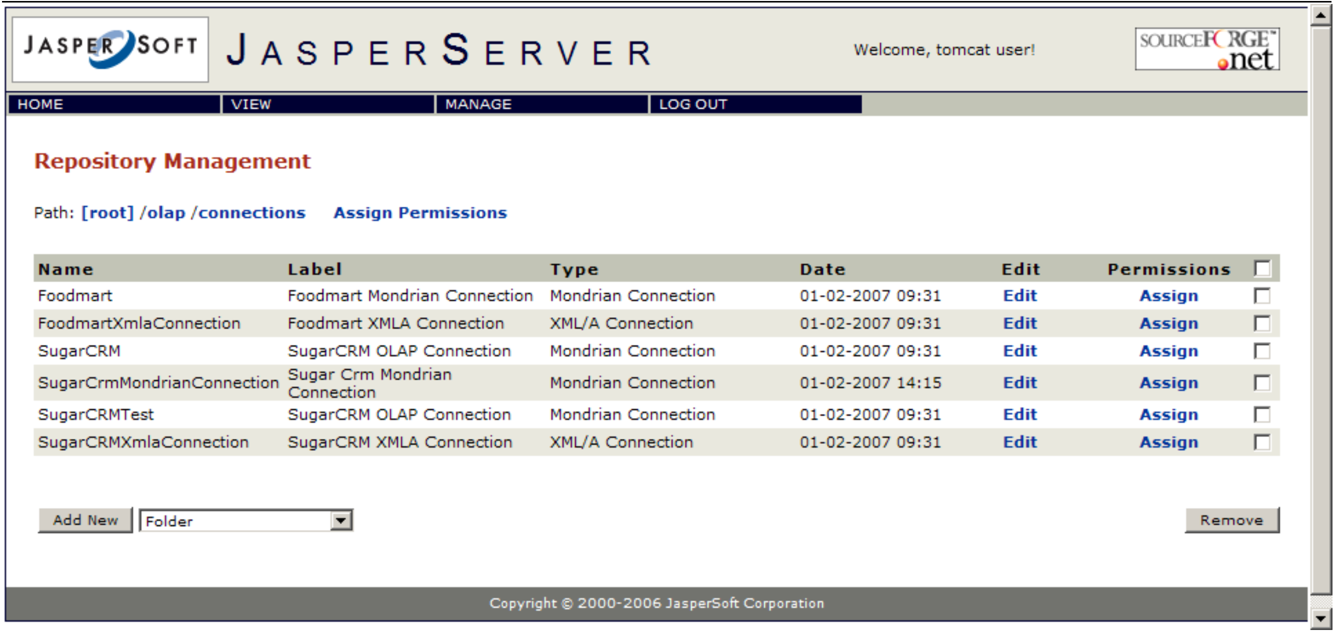
- Click the Save button to save the changes.

6.7.6 Editing XML/A connection

You may change an XML/A connection's naming and connection properties. Following the following steps to change a Mondrian connection:

1. Select the REPOSITORY menu item from the MANAGE menu.
2. Click *olap* and *connections* under the Name column to display the connections in the Repository.

6.7.6.1 Connections



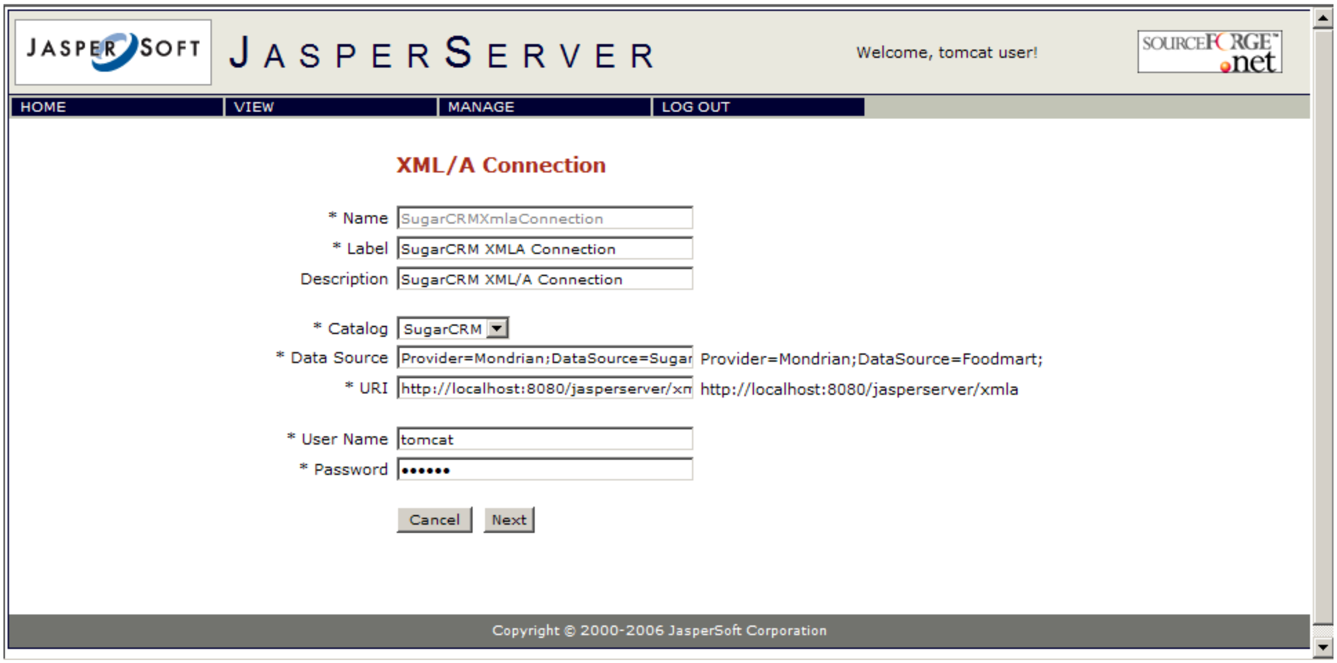
Repository Management

Path: [root] /olap /connections Assign Permissions

Name	Label	Type	Date	Edit	Permissions	□
Foodmart	Foodmart Mondrian Connection	Mondrian Connection	01-02-2007 09:31	Edit	Assign	□
FoodmartXmlaConnection	Foodmart XMLA Connection	XML/A Connection	01-02-2007 09:31	Edit	Assign	□
SugarCRM	SugarCRM OLAP Connection	Mondrian Connection	01-02-2007 09:31	Edit	Assign	□
SugarCrmMondrianConnection	Sugar Crm Mondrian Connection	Mondrian Connection	01-02-2007 14:15	Edit	Assign	□
SugarCRMTest	SugarCRM OLAP Connection	Mondrian Connection	01-02-2007 09:31	Edit	Assign	□
SugarCRMXmlaConnection	SugarCRM XMLA Connection	XML/A Connection	01-02-2007 09:31	Edit	Assign	□

3. Select an XML/A connection, e.g., *SugarCRMXmlaConnection*, and Click the Edit button to the right

6.7.6.2 XML/A connection



XML/A Connection

* Name: SugarCRMXmlaConnection
* Label: SugarCRM XMLA Connection
Description: SugarCRM XML/A Connection

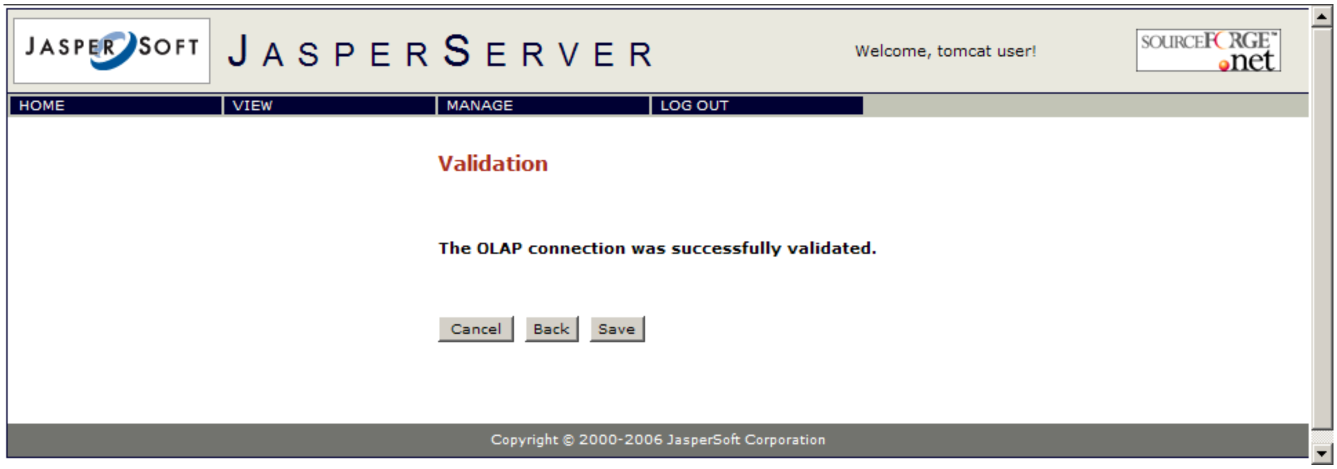
* Catalog: SugarCRM
* Data Source: Provider=Mondrian;DataSource=Sugar Provider=Mondrian;DataSource=Foodmart;
* URI: http://localhost:8080/jasperserver/xm http://localhost:8080/jasperserver/xmla

* User Name: tomcat
* Password: ••••••

4. Change the label, description and/or connection properties of connection.

5. Click the Next button.

6.7.6.3 *Validation*



Validation

The OLAP connection was successfully validated.

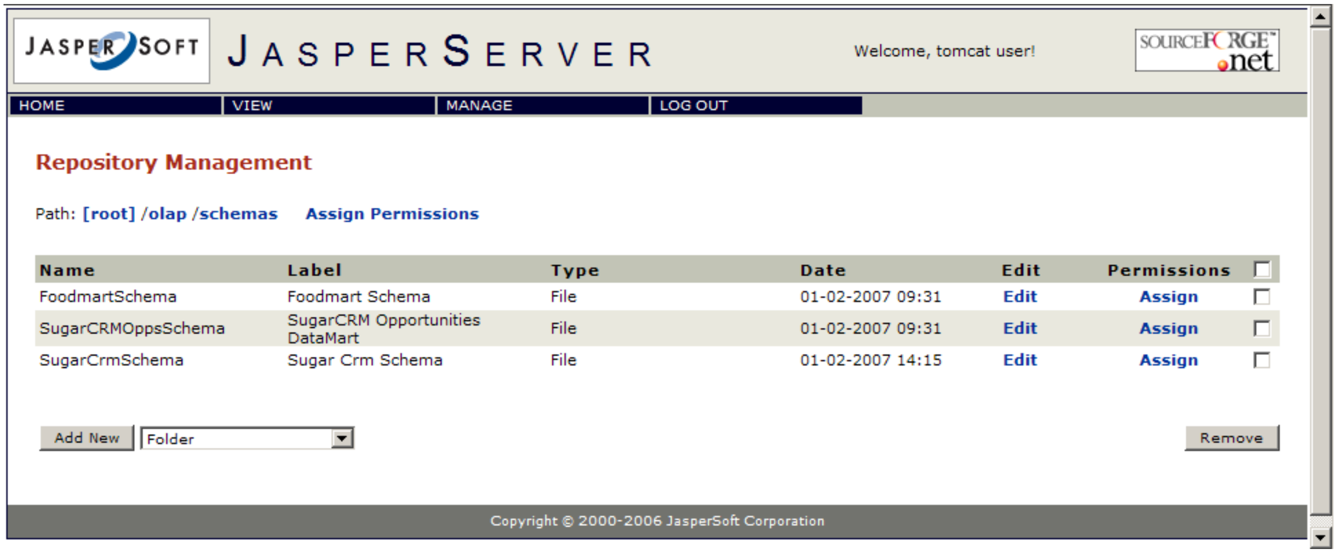
6. Click the Save button to save the changes.

6.7.7 Editing an analysis schema

You may change a analyis schema's naming and file source. Following the following steps to change a Mondrian schema:

1. Select the REPOSITORY menu item from the MANAGE menu.
2. Click *olap* and *schemas* under the Name column to display the schemas in the Repository.

6.7.7.1 *OLAP schemas*



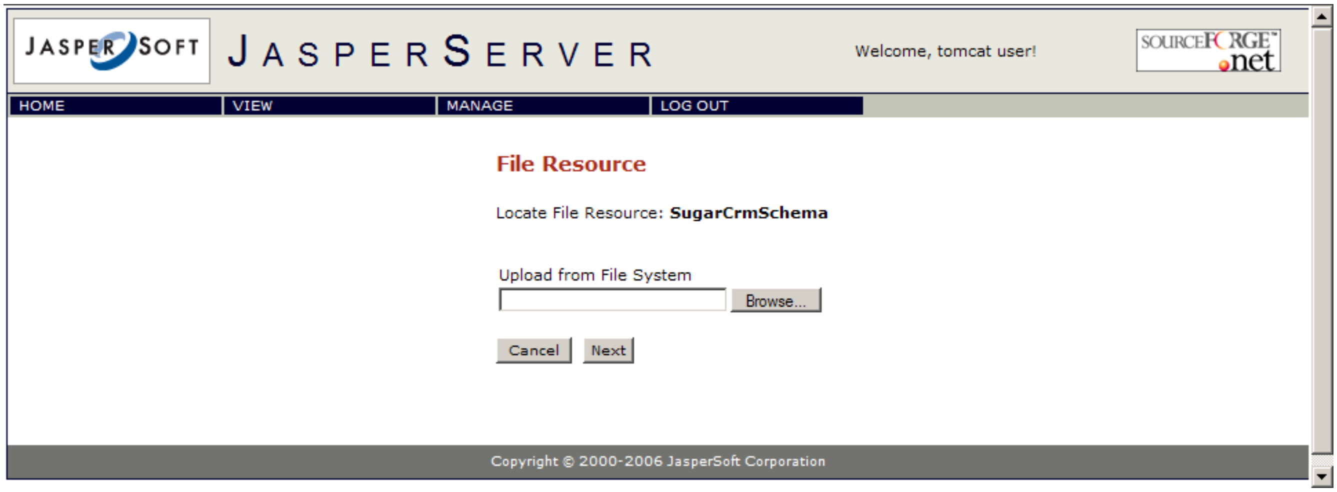
Repository Management

Path: [root] /olap /schemas   Assign Permissions

Name	Label	Type	Date	Edit	Permissions	□
FoodmartSchema	Foodmart Schema	File	01-02-2007 09:31	Edit	Assign	□
SugarCRMOppsSchema	SugarCRM Opportunities DataMart	File	01-02-2007 09:31	Edit	Assign	□
SugarCrmSchema	Sugar Crm Schema	File	01-02-2007 14:15	Edit	Assign	□

3. Select a Mondrian schema, e.g., *SugarCrmSchema*, and Click the Edit button to the right.

6.7.7.2 Upload file resource



File Resource

Locate File Resource: **SugarCrmSchema**

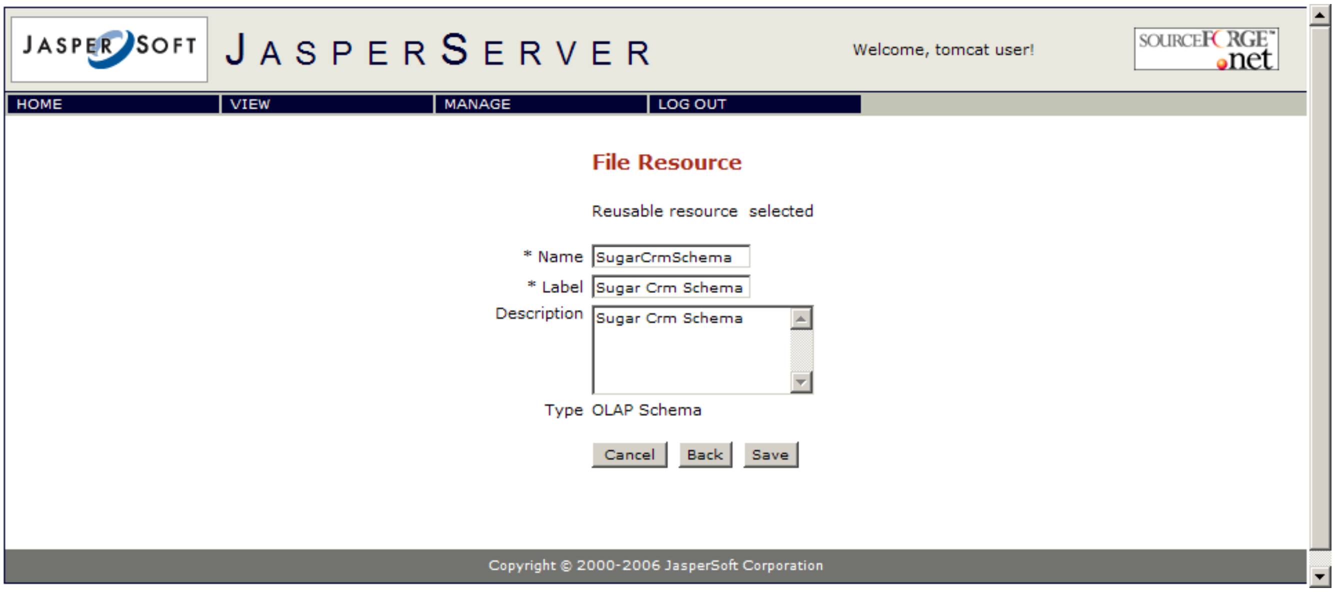
Upload from File System

4. Click the 'Browse…' button to upload a new schema file.

5. Select a schema file in the file system and click the Open button

6. Click the Next button.

6.7.7.3 File resource naming



File Resource

Reusable resource selected

* Name SugarCrmSchema

* Label Sugar Crm Schema

Description Sugar Crm Schema

Type OLAP Schema

7. Change the naming of the schema.

8. Click the Save button.

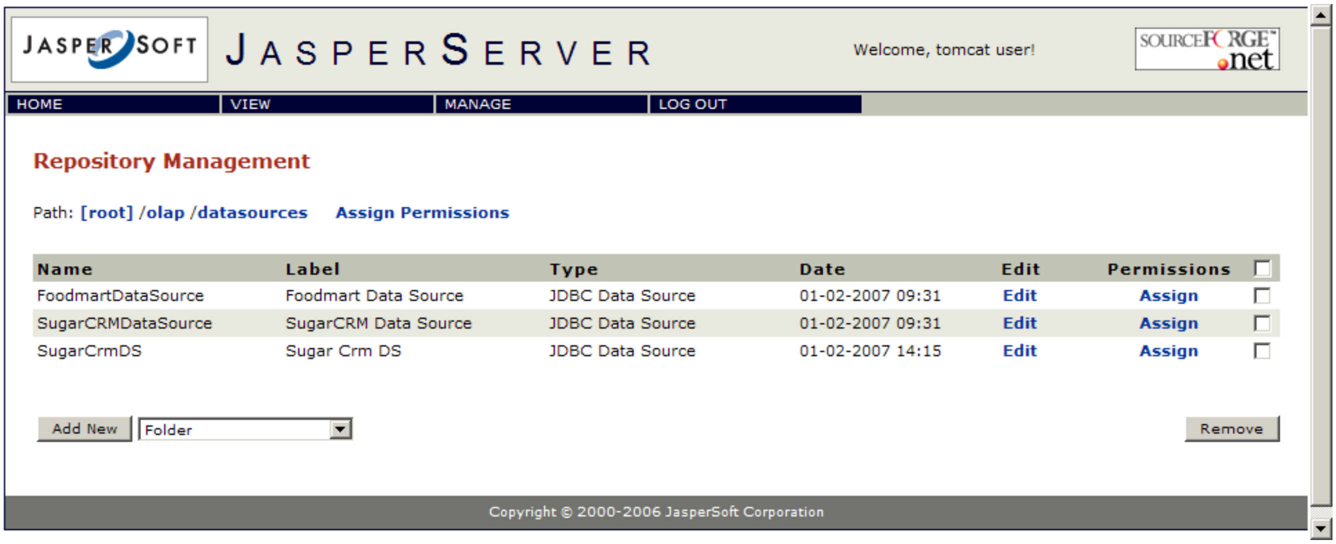
6.7.8 Editing data source

You may change a Data source's naming and connection properties. Following the following steps to change a data source:

1. Select the REPOSITORY menu item from the MANAGE menu.

2. Click *olap* and *datasources* under the Name column to display the data sources in the Repository.

6.7.8.1 Data sources



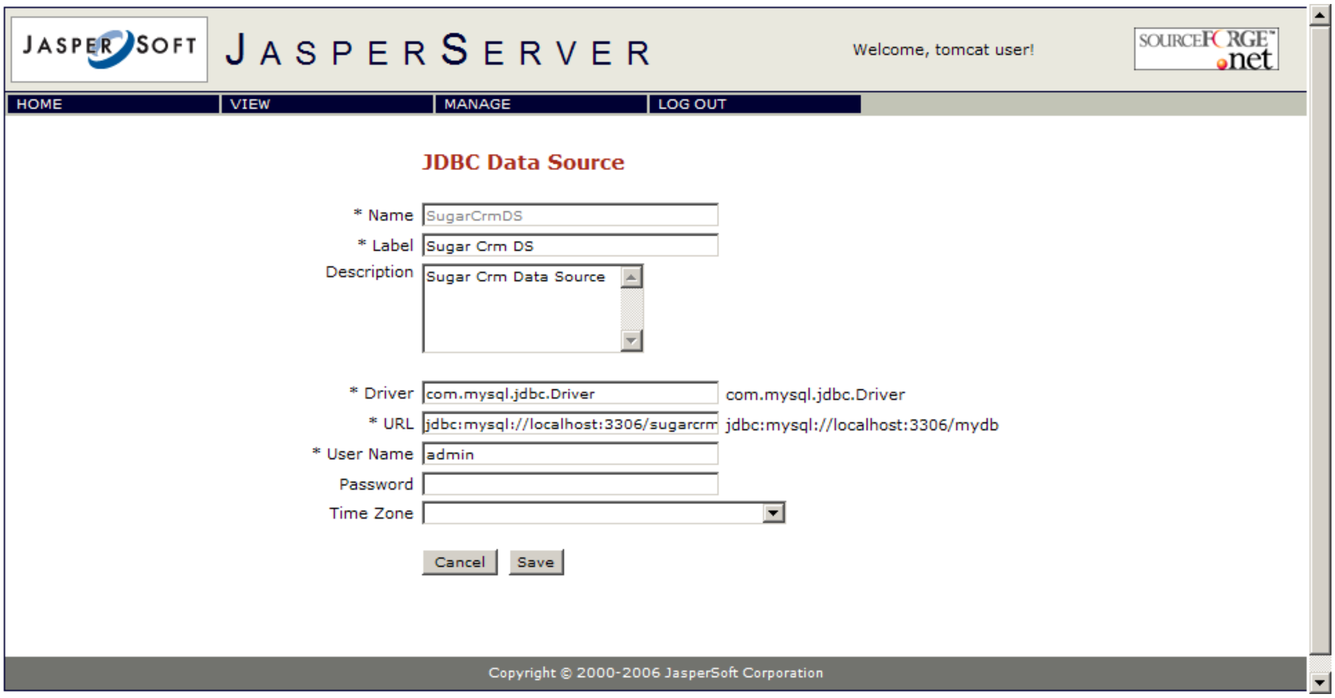
Repository Management

Path: [root] /olap /datasources Assign Permissions

Name	Label	Type	Date	Edit	Permissions	□
FoodmartDataSource	Foodmart Data Source	JDBC Data Source	01-02-2007 09:31	Edit	Assign	□
SugarCRMDataSource	SugarCRM Data Source	JDBC Data Source	01-02-2007 09:31	Edit	Assign	□
SugarCrmDS	Sugar Crm DS	JDBC Data Source	01-02-2007 14:15	Edit	Assign	□

3. Select a data source, e.g., *SugarCrmDS*, and Click the Edit button to the right.

6.7.8.2 JDBC data source



JDBC Data Source

* Name	SugarCrmDS	
* Label	Sugar Crm DS	
Description	Sugar Crm Data Source	
* Driver	com.mysql.jdbc.Driver	com.mysql.jdbc.Driver
* URL	jdbc:mysql://localhost:3306/sugarcrm	jdbc:mysql://localhost:3306/mydb
* User Name	admin	
Password		
Time Zone		

9. Change the naming and JDBC connection properties of the data source.

10. Click the Save button.

6.7.9 Viewing Mondrian properties

Various configurable properties control aspects of Mondrian's behavior, and therefore affect JasperAnalysis. For most applications, the properties can be left with the default values, but in some cases (such as performance tuning) it may be desirable to change them. They can be altered by editing the file *jasperserver/WEB-INF/classes/mondrian.properties*. More information about these properties can be found at http://mondrian.sourceforge.net/configuration.html.

The following properties can be viewed by clicking on the "Analysis Properties" item of the Manage menu:

Property	Value
JdbcDrivers	controls which drivers will be loaded. Only modify this if you are using a database which is not in the current list.
DrillThroughMaxRows	limits the number of rows returned from a drill-through operation.
DisableCaching	turns off caching completely. May make performance very slow.
TraceLevel	setting to a value greater than 0 will cause all SQL statements to be logged. Useful for optimizing database performance.
DebugOutFile	tells the system where to log SQL statements when TraceLevel is set. The default will log to standard out.
QueryLimit	maximum number of concurrent queries allowed.
ResultLimit	when set to a number greater than 0, limits the result to this many rows.
LargeDimensionThreshold	when a dimension has a greater number of members than this value, a special reader will be used by Mondrian.
SparseSegmentDensityThreshold	performance tuning variable.
SparseSegmentCountThreshold	performance tuning variable.
UseAggregates	instructs Mondrian to enable use of aggregate tables. Building aggregate tables will greatly help performance of high-level queries on large tables.
ReadAggregates	instructs Mondrian to scan the database for tables that may be aggregates. For a detailed explaination of aggregate tables and how they work, see http://mondrian.sourceforge.net/aggregate_tables.html.
ChooseAggregateByVolume	performance tuning variable for aggregates.

There are many other properties described in the Mondrian guide, but many of these are only interesting to developers working on Mondrian. Some of these other properties do not matter when using JasperAnalysis, because the JasperServer content repository provides a means of controlling them from the UI. For example, it is not necessary to provide a connectString property, because the proper information is automatically generated by JasperAnalysis when the user selects a DataSource for an OlapClientConnection.

6.7.10 Flushing OLAP Cache

This page provides a simple way of clearing the various in-memory caches that Mondrian builds to improve query performance. Flushing the cache is not usually necessary except in the case when the data has changed. For example, after running an ETL process the cache will be out of date until the application server is restarted, or until the cache is flushed by hitting this page. Future versions of JasperAnalysis will provide finer grain control for clearing subsets of the cache appropriate to the data that has changed, and will provide other options for integrating with ETL jobs.

7 Server components

7.1 JasperAnalysis components

7.1.1 Getting started with JasperAnalysis

JasperAnalysis, implementing On Line Analytical Processing (OLAP), is a means of storing and retrieving information based on a hierarchical dimensional model, which is made up of cubes and dimensions as the primary entities. In a relational database implementation of OLAP, facts which make up cubes and members which comprise dimensions are stored in relational tables, and is sometimes referred to as ROLAP. In a ROLAP-based architecture, most of the computation is done by the RDBMS, and therefore most of the optimization to improve performance is in the database, in the form of adding indexes, aggregate tables, and other tuning.

JasperServer provides OLAP capabilities via an open source product called Mondrian, with is a fully functional ROLAP implementation. JPivot is another open source product used for displaying and navigating results.

The de facto standard for query languages among OLAP products is Multidimensional Expressions (MDX). In a distributed computing environment, XML for Analysis (XML/A) is the standard for relaying a SOAP message consisting of an MDX statement and the information about which data source to use.

7.1.2 Designing a multidimensional schema

In order to retrieve and display multidimensional reports, it is necessary to specify how the tables in a datasource are arranged and how they map to cubes and dimensions. In JasperServer, this is done by creating a FileResource which describes the OLAP schema to be used. The syntax of this schema file is specific to Mondrian, and a guide to creating one can be found at [http://mondrian.sourceforge.net/schema.html].

7.1.3 Single server vs. multiple servers

For a simple application with a relatively small dataset where performance is not critical, (for example, a small intranet application, or developer environment) the easiest server configuration is to run a single application server with the JasperServer web application which connects to a local database.

For larger OLAP applications, the database is often a performance bottleneck, so when performance matters, it is strongly recommended to run the database on its own dedicated hardware, optimized for disk reads and writes, and run the application server on a different machine.

Inside the application, the result visualization and navigation software (JPivot) competes for resources with the OLAP engine (Mondrian). So, the next step in increasing the scalability of a deployment is to separate the processing resources for these two functions. The supported method for doing this is to run the OLAP engine as a standalone application server in its own process, and preferably on its own machine. It will receive XML/A requests from remote clients, connect to the database for the appropriate datasource to retrieve the SQL results, process the results to create the final OLAP result objects, and return this response via XML/A. For purposes of this document, the application that sends the XML/A request is refered to as an XML/A client, while the application that sends the result back is refered to as an XML/A server or XML/A provider.

The JasperServer web application is already configured to run as either an XML/A client or server. This means it is possible to run one JasperServer instance as an XML/A provider and run one or more other instances of an application server with JasperServer deployed as XML/A clients.

7.1.4 XML/A security

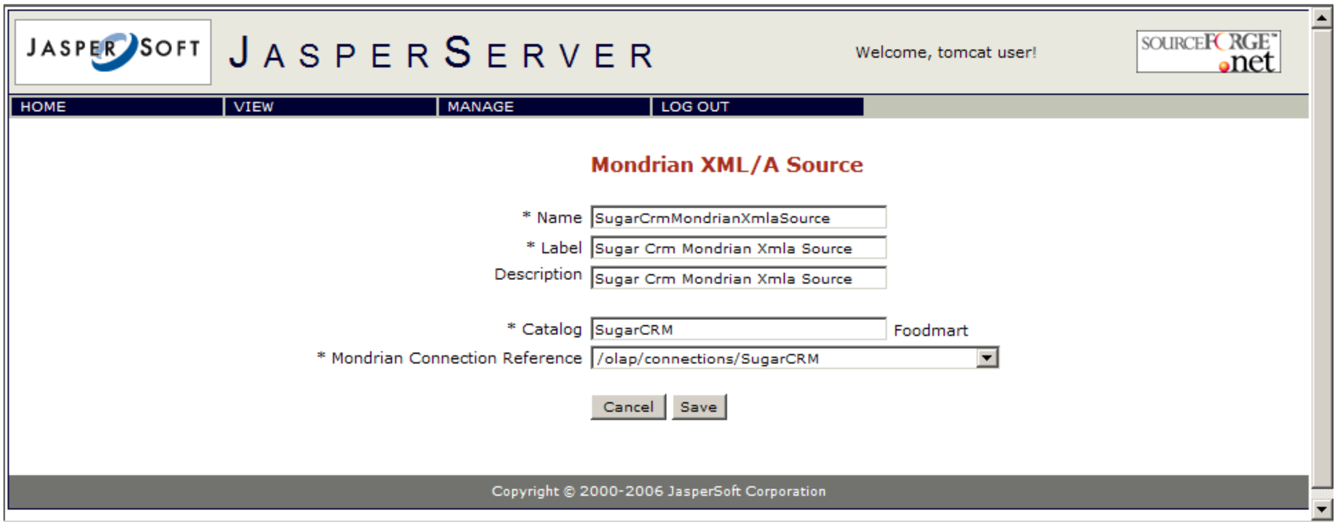
The default configuration uses HTTP Basic authentication to challenge requests for the path */xmla*. The client will need to have defined a valid username and password in its XMLAConnection source. Note that with HTTP Basic authentication, clear-text passwords are transmitted in the header of an HTTP request, so if encryption is desired, one must run inside a network secured with SSL to protect those passwords.

7.1.5 XML/A configuration

To accomplish the multi-server configuration described above, you need to create a Mondrian XML/A source followed by defining an OLAP client connection using the XML/A conneciton source.

7.1.5.1 Creating Mondrian XML/A source

Mondrian XML/A source is a XML/A connection definition for accessing Mondrian connections.



Mondrian XML/A Source

* Name	SugarCrmMondrianXmlaSource
* Label	Sugar Crm Mondrian Xmla Source
Description	Sugar Crm Mondrian Xmla Source
* Catalog	SugarCRM Foodmart
* Mondrian Connection Reference	/olap/connections/SugarCRM

1. Select the REPOSITORY menu item from the MANAGE menu.

2. Click *olap*, *xmla* and *definitions*.

3. Select *Mondrian XML/A Source* from the dropdown list next to the Add New button.

4. Click the Add New button.

5. Enter Mondrian XML/A Source information, e.g., Name=*SugarCrmMondrianXmlaSource*, Label=*Sugar Crm Mondrian Xmla Source*, Description=*Sugar CRM Mondrian Xmla Source.*,

6. Click the Save button to save the XML/A source.

Once the Mondrian XML/A connection source is created, you can create an OLAP client connection using this XML/A conneciton source. Once the XML/A connection is created, you can create an analysis view using this XML/A connection. (See 6.7.3 for a description to create an OLAP client connection using the Mondrian XML/A connection source, and to create an analysis view using the XML/A connection.)

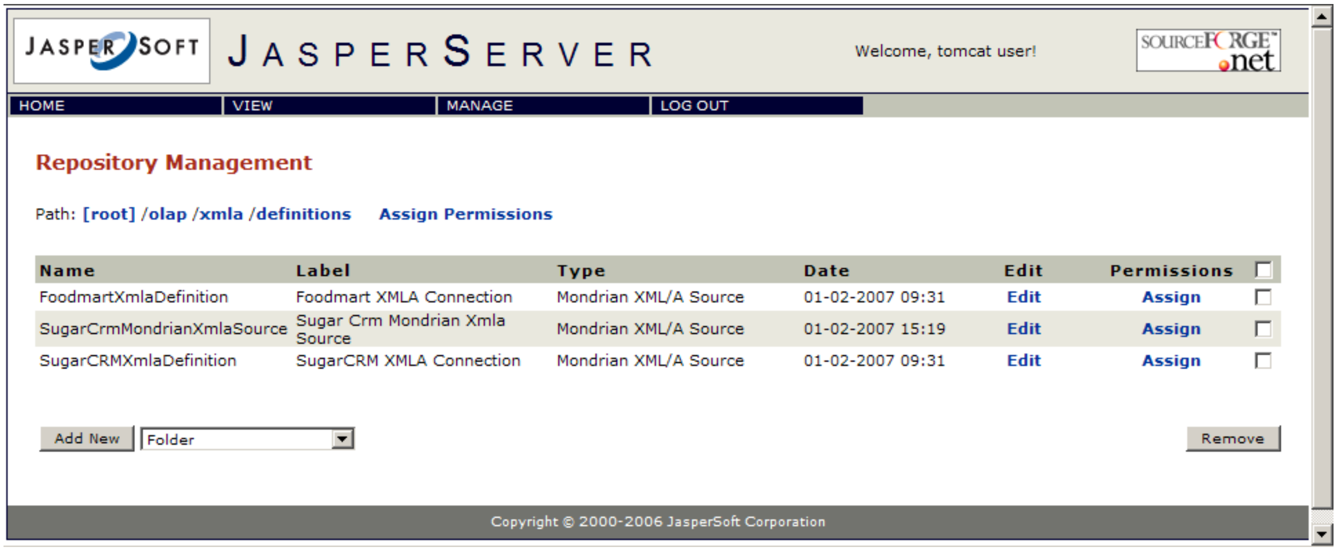
7.1.6 Edit Mondrian XML/A source

You may change a Mondrian XML/A source's naming and connection properties. Following the following steps to change a Mondrian XML/A source:

1. Select the REPOSITORY menu item from the MANAGE menu.

2. Click *olap*, *xmla* and *definitions* under the Name column to display the Mondrian XML/A sources in the Repository.

7.1.6.1 Mondrian XML/A source



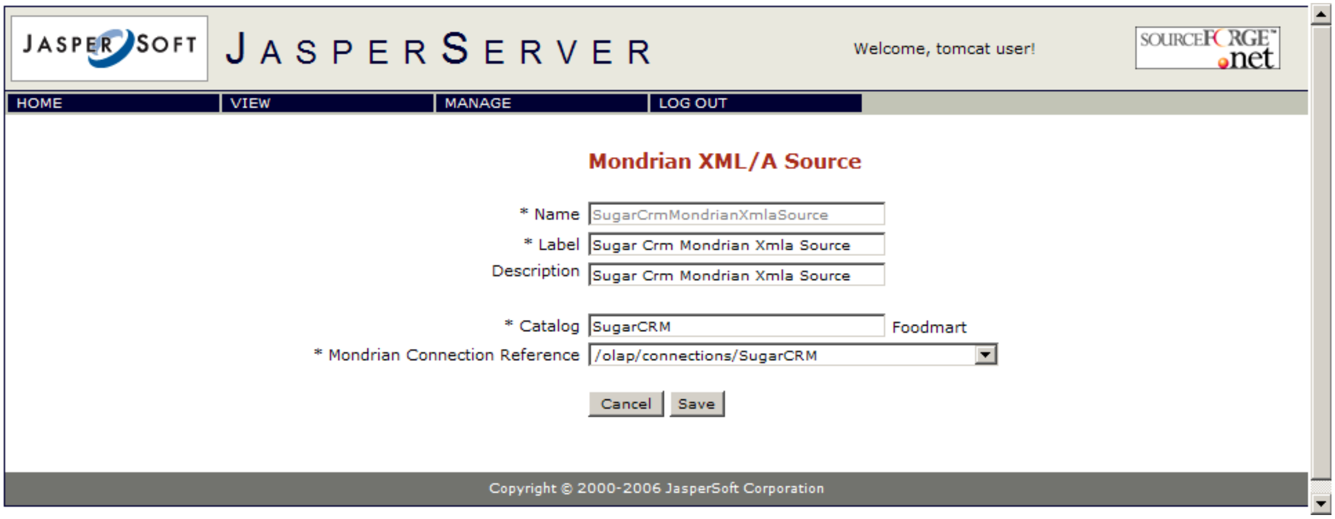
Repository Management

Path: [root] /olap /xmla /definitions   Assign Permissions

Name	Label	Type	Date	Edit	Permissions	□
FoodmartXmlaDefinition	Foodmart XMLA Connection	Mondrian XML/A Source	01-02-2007 09:31	Edit	Assign	□
SugarCrmMondrianXmlaSource	Sugar Crm Mondrian Xmla Source	Mondrian XML/A Source	01-02-2007 15:19	Edit	Assign	□
SugarCRMXmlaDefinition	SugarCRM XMLA Connection	Mondrian XML/A Source	01-02-2007 09:31	Edit	Assign	□

3. Pick an XML/A source, e.g., *SugarCrmMondrianXmlaSource* created in 7.1.5.1.
4. Click the Edit button to the right.

7.1.6.2 Mondrian XML/A source naming and properties



Mondrian XML/A Source

* Name: SugarCrmMondrianXmlaSource
* Label: Sugar Crm Mondrian Xmla Source
Description: Sugar Crm Mondrian Xmla Source
* Catalog: SugarCRM   Foodmart
* Mondrian Connection Reference: /olap/connections/SugarCRM

5. Change the naming and/or the connection properties of the connection source.
6. Click the Save button to save the changes.

7.1.7 External resources

Since MDX was originally developed by Microsoft, many guides focus on SQL Server, but the language concepts and syntax are the same.

1. Historical overview of OLAP [http://en.wikipedia.org/wiki/OLAP]
2. Mondrian SourceForge project [http://mondrian.sourceforge.net]
3. JPivot SourceForge project [http://jpivot.sourceforge.net/]
4. Mondrian MDX reference [http://mondrian.sourceforge.net/mdx.html]
5. Mondrian Performance Optimization Guide [http://mondrian.sourceforge.net/optimizing_performance.html]

6. William Pearson's “Introduction to SQL Server MDX Essentials” [http://www.databasejournal.com/features/mssql/article.php/1495511]
7. Microsoft's MDX Reference [http://msdn2.microsoft.com/en-us/library/ms145506.aspx]
8. Wikipedia MDX overview and links[http://en.wikipedia.org/wiki/Multidimensional_Expressions]
9. A collection of other MDX article links [http://www.mosha.com/msolap/mdx.htm]
10. Official XMLA home [http://www.xmla.org/]

8 Utilities

8.1 Import/Export

The import and export utilities let you extract resources from or add resources to a JasperServer repository. Import and export can be helpful when migrating between versions of JasperServer, or when moving between test and production environments.

8.1.1 Import Resources

usage: ji-import [OPTIONS]

Reads a repository catalog from the disk (created using the ji-export command) and creates the named resources in the current JasperServer application repository.

Options:

--prepend-path	string to prepend to a URI path for all imported resources
--input-dir	path for import catalog directory
--input-zip	path for import catalog zip file
--help	print usage message

Examples:

- Import the `myDir` catalog folder, prepending `/importDir` to all repository URIs:
```
ji-import --input-dir myDir --prepend-path /importDir
```
- Import the `myExport.zip` catalog archive file:
```
ji-import --input-zip myExport.zip
```

Notes:

The prepend-path option is handy for avoiding URI path conflicts during import. For example, if the resource in the catalog file is /images/JRLogo and you set a prepend-path of myNewDir, then the resource is imported and created under the URI path /myNewDir/images/JRLogo. So, if you are importing a set of resources into a repository, and the URI naming conflicts with objects already in the repository, adding a prepend-path allows you to import the resources to a different location. However, this does not affect repository URIs in JRXML reports (for example, if a report image has <imageExpression>repo:/images/JRLogo</imageExpression>, that URI is not prepended during import).

During import, if a resource is found in the target repository that has the same URI as the resource that you are attempting to create, the create operation is skipped, and the existing resource is left unchanged (no overwrite occurs).

8.1.2 Export Resources

usage: ji-export [OPTIONS]

Specifies repository resources such as reports, images, folders, users, roles, and scheduled jobs to export to a folder structure on disk.

The export output is known as a repository catalog.

Options:

--uris	Comma separated list of repository folder/resource URI paths
--repository-permissions	When this option is present, repository permissions are exported along with each exported folder and resource. This option should only be used in conjunction with --uris
--report-jobs	Comma separated list of repository report unit and folder URIs for which report unit jobs should be exported. If a folder URI is specified, the folder is recursively traversed and each report unit found has its jobs exported.
--users	Comma separated list of users to export; if no users are specified, all users are exported
--roles	Comma separated list of roles to export; if no roles are specified, all roles are exported
--role-users	When this option is present, each role export triggers the export of all users belonging to the role. This option should only be used in conjunction with --roles
--everything	Export everything: all repository resources, permissions, report jobs, users and roles. Equivalent to --uris / --repository-permissions --report-jobs / --users --roles
--output-dir	Output catalog folder
--output-zip	Output catalog zip file
--help	Print usage message

Examples:

- Export the /reports/samples/AllAccounts resource to a catalog folder:
  ```
  ji-export --uris /reports/samples/AllAccounts --output-dir myExport
  ```
- Export the /images and /fonts folders:
  ```
  ji-export --uris /images,/fonts --output-dir myExport
  ```
- Export all the resources with permissions to a zip catalog:
  ```
  ji-export --uris / --repository-permissions --output-zip myExport.zip
  ```
- Export all the resource and all the report jobs:
  ```
  ji-export --uris / --report-jobs / --output-dir myExport
  ```
- Export the report jobs of the /reports/samples/AllAccounts report unit
  ```
  ji-export --report-jobs /reports/samples/AllAccounts --output-dir myExport
  ```
- Export all the roles and users:
  ```
  ji-export --roles --users --output-dir myExport
  ```
- Export the ROLE_USER and ROLE_ADMINISTRATOR roles along with all users belonging to one of these roles:
  ```
  ji-export --roles ROLE_USER, ROLE_ADMINISTRATOR --role-users --output-dir myExport
  ```
- Export two users:
  ```
  ji-export --users jasperadmin,joeuser--output-dir myExport
  ```

Notes:

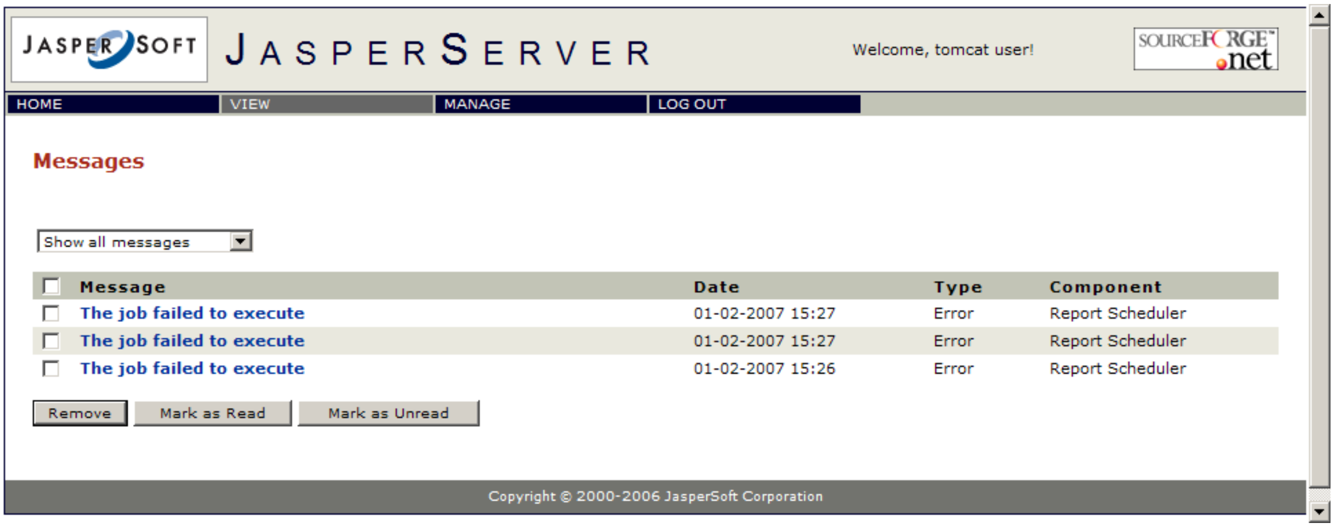
- The --uris option allows you to specify one or more resources or folder URIs. A URI can specify a resource such as a report. In this case, all associated resources such as images, subreports, data sources, resource bundles, and classfiles are exported. A URI can also specify a folder. If a folder is specified, the export operation exports all files and folders contained in the folder. In addition, it recurses through all its subfolders.
- When you export a user, in addition to the user information, the user’s roles are also exported. When you import a user, if its roles exist in the repository, the user is given these roles.

8.2 Message viewer

The message viewer screens allows users to browse the server event log. This log is meant to be used by server component to log messages for events that require user notification. Currently the event log is exclusively used by the report scheduling module to log report execution errors.

8.2.1 Messages list

The *messages list* screen displays the list of events logged for the current user.



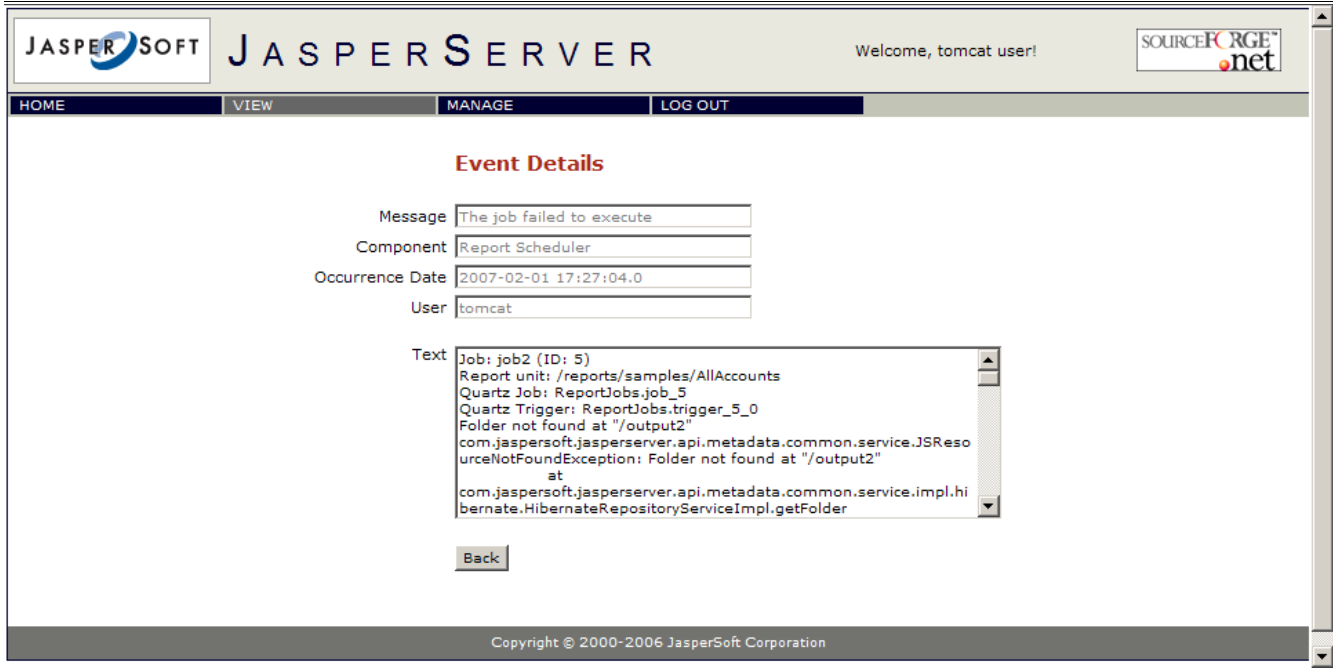
Messages

☐	Message	Date	Type	Component
☐	The job failed to execute	01-02-2007 15:27	Error	Report Scheduler
☐	The job failed to execute	01-02-2007 15:27	Error	Report Scheduler
☐	The job failed to execute	01-02-2007 15:26	Error	Report Scheduler

8.2.2 Message details

The full *message details* can be opened by clicking the event message.

The user can choose to browse all the messages or to see only the unread messages.



Event Details

Message: The job failed to execute

Component: Report Scheduler

Occurrence Date: 2007-02-01 17:27:04.0

User: tomcat

Text:

```
Job: job2 (ID: 5)
Report unit: /reports/samples/AllAccounts
Quartz Job: ReportJobs.job_5
Quartz Trigger: ReportJobs.trigger_5_0
Folder not found at "/output2"
com.jaspersoft.jasperserver.api.metadata.common.service.JSReso
urceNotFoundException: Folder not found at "/output2"
            at
com.jaspersoft.jasperserver.api.metadata.common.service.impl.hi
bernate.HibernateRepositoryServiceImpl.getFolder
```

8.3 iReport Plugin installation and configuration

8.3.1 Requirements

- IReport 1.3.1 or greater
- Java Sun JDK 1.5 or greater (not mandatory but strongly suggested)
- JasperServer with enabled Web Services

8.3.2 Plugin installation

To install the plugin you have to drop into the *lib* directory of iReport the following jars:

jasperserver-ireport-plugin-1.1.0.jar

XmlSchema-1.0.2.jar

activation-1.1.jar

axiom-api-1.0.jar

axiom-impl-1.0.jar

axis2-1.0.jar

castor-1.0-xml.jar

commons-codec-1.3.jar

commons-httpclient-3.0.jar

mail-1.4.jar

neethi-1.0.1.jar

stax-api-1.0.jar

wsdl4j-1.5.2.jar

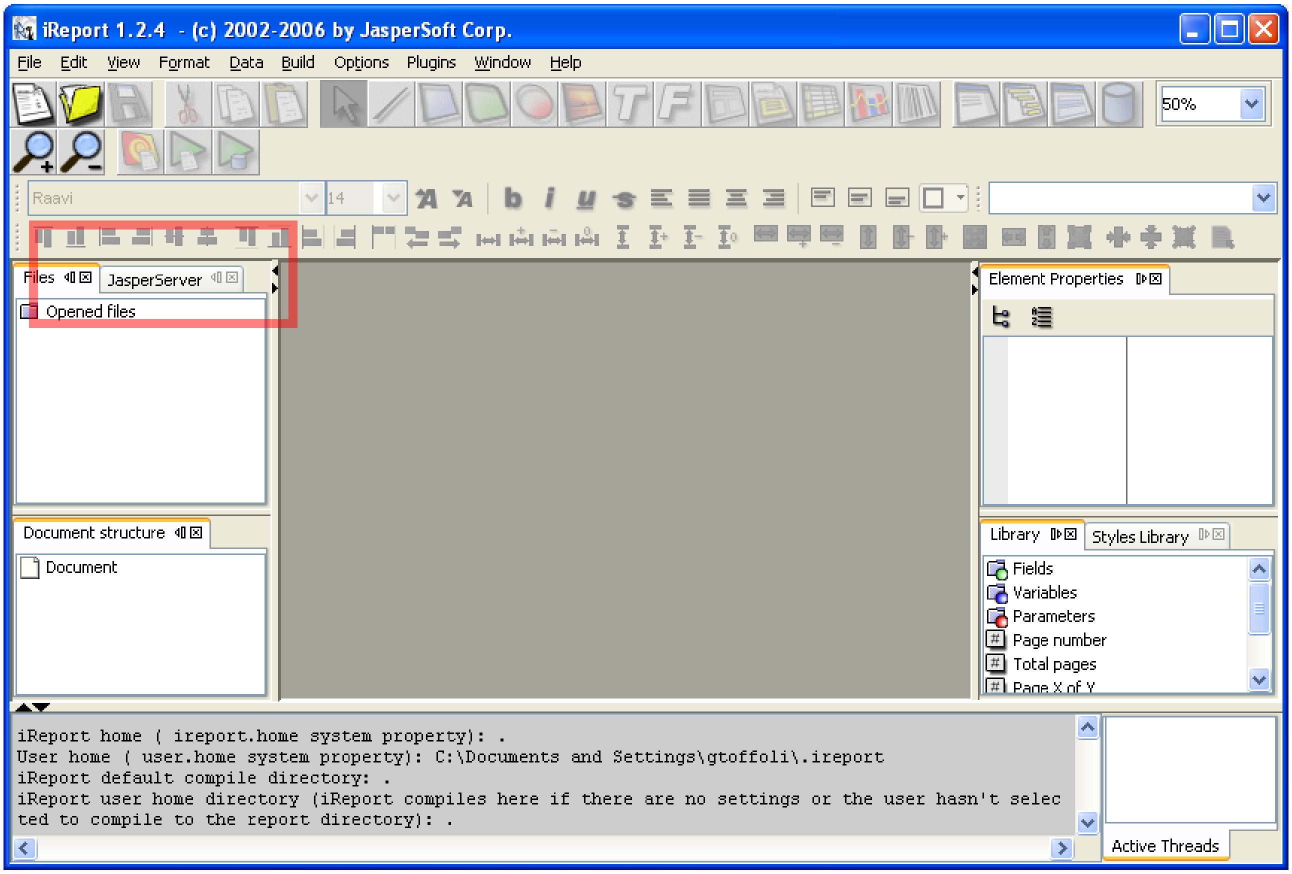
wstx-asl-2.8.2.jar

xbean-2.1.0.jar

All this jars are already in place if iReport is shipped in boundle with the JasperServer. iReport will load the plugin automatically when started. To lunch iReport on the Windows platform, you may use the Start/All Programs/JasperServer/iReport menu item. On Linux simply type:

./ iReport.sh

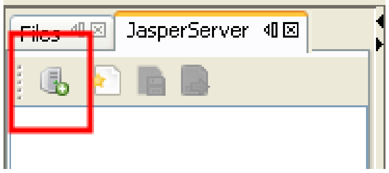
from the iReport home directory. You will see a JasperServer pane in the upper left hand side of your iReport workspace.



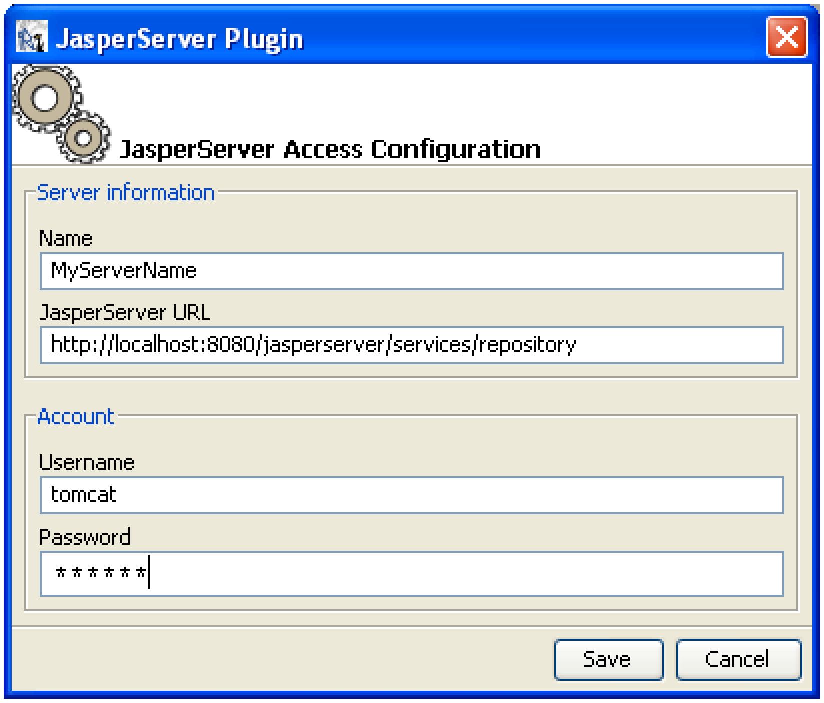
```
iReport home ( ireport.home system property): .
User home ( user.home system property): C:\Documents and Settings\gtoffoli\.ireport
iReport default compile directory: .
iReport user home directory (iReport compiles here if there are no settings or the user hasn't selec
ted to compile to the report directory): .
```

8.3.3 Configuring the plugin

When the plugin is started for the first time, no servers are yet configured. To add a server, press the first button in the plugin toolbar.



You get a dialog box titled"JasperServer Plugin" that allows for JasperServer access configuration. In this dialog box, you will enter the information that will enable you to connect to your JasperServer Web Services application.



JasperServer Plugin

JasperServer Access Configuration

Server information

Name: MyServerName

JasperServer URL: http://localhost:8080/jasperserver/services/repository

Account

Username: tomcat

Password: ******

For instance:

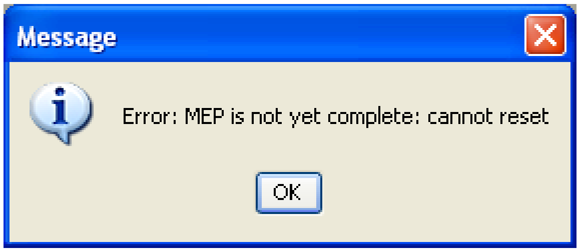
Name:	MyServerName
JasperServer URL:	<you should only need to change the hostname>
Username:	<jasperadmin>
Password:	<password>

A server entry should now appear in this pane.

Double-click on the server entry to see a list of Resources at the root of the JasperServer repository. You may view the property information on the Resources in the repository. You may download, edit and upload jrxml files. You may download images and other files (using right-click "export file").

8.3.4 Troubleshooting

If the connection fails, you can get a message similar to the following:



Error: MEP is not yet complete: cannot reset

Check the login and password and be sure that the URL set for the server is exact. If needed, you can modify the server settings selecting it in the JasperServer pane and choosing the properties menu item from the popup menu.

9 Advanced configuration

This chapter presents some of the most common extension points that can be used to enrich or customize the JasperServer functionality and achieve the desired level of integration with the enviroment where it is deployed.

9.1 Custom Data Sources

JasperServer provides built-in support for many commonly used data sources, such as JDBC, Java beans, Mondrian, and XML/A. However, not all JasperReports data sources have corresponding JasperServer data sources, and you may have written your own custom JasperReports data source that you want to use within JasperServer. In either case, you will need to extend JasperServer to support additional data sources. Fortunately, the current version of JasperServer has a framework which makes it possible to add a new data source by adding just a few files to your JasperServer configuration.

This section will cover the following subjects:

- Instructions for installing examples of custom data sources in your JasperServer web application
- A description of JasperServer data sources and how they interact with JasperReports data sources
- Descriptions of all the components required to implement a new JasperServer data source
- Instructions on how to deploy your new JasperServer data source

9.1.1 Background on data sources in JasperServer and JasperReports

A JasperServer data source is not the same thing as a JasperReports data source, but they work together closely. A JasperReports data source is an implementation of the `JRDataSource` interface which provides data organized in rows and columns to the JasperReports filler, which is responsible for producing a `JasperPrint` object. Each field declared in the JRXML corresponds to a column in the `JRDataSource` output.

A JasperServer data source is a persistent object in the JasperServer repository which stores properties which tell JasperServer how to create a `JRDataSource` (possibly in collaboration with a `JRQueryExecuter`, see below). These properties vary by the type of data source; for example, a JDBC data source will have a JDBC driver, URL, user, and password. A data source, like other repository objects such as data types and input controls, can be defined as a “public” repository object which can be used by any report unit (for example, `/datasources/JserverJdbcDS`), or as a “local” object used by a specific report unit. Public and local data sources can be created and edited with the JasperServer web interface.

When JasperServer receives a request to run a report unit, it looks up the the report unit's data source, and maps that to an implementation of ReportDataSourceService, which is in turn responsible for handing back a JRDataSource based on the data source's persistent properties. The JRDataSource is used to fill the report and produce a JasperPrint object, which can be turned into HTML or any other supported output format.

Each JasperServer data source implementation has to do the following:

- Read and write persistent properties in the JasperServer repository
- Provide a user interface for creating and editing instances which is integrated in the JasperServer web interface
- Create a JRDataSource using the property values for a specific data source instance, or pass parameters to a JRQueryExecuter which produces the JRDataSource.

The existing data sources each require about a half dozen Java classes, along with many changes to Spring bean files, WebFlow configurations, message files, and JSP files. The custom data source framework described here provides the same functionality using a Spring bean file, a message catalog, and a minimum of one Java file (more for optional features).

9.1.1.1 Query executers

Query executers are implementations of the JasperReports interface called JRQueryExecuter. They are responsible for interpreting the queryString inside the JRXML and producing a JRDataSource. Some JasperServer data sources don't need queryStrings and will produce a JRDataSource which is passed directly to the filler with no query executer involved. However, the ability to pass a queryString gives the data source a lot more flexibility. In this case, the data source will put implementation-specific objects in the report parameter map that get passed down to the JRQueryExecuter, which uses them to produce the JRDataSource. The prime example of this is the JDBC data source, which puts a JDBC Connection object in the parameter map. The JRJdbcQueryExecuter then uses the connection to create a JDBC Statement using the queryString, run it on the connection, and generate a JRDataSource with the result set.

The webscraper example below shows examples of both approaches—it can be used either with or without a queryString in the JRXML.

9.1.2 Custom Data Source Examples

The examples are found in <js-install>/samples/customDataSource. Once you have deployed a JasperServer web application to your application server, you can use Ant to build and deploy the examples.

If you used an installer to install JasperServer version 2.1 or later, you will have Ant installed already. Ant may be run with the following command:

<js-install>/ant/bin/ant <ant-arguments>(for Linux/Unix)

<js-install>\ant\bin\ant <ant-arguments> (for Windows)

If you installed JasperServer manually with a WAR file, you will need to download Ant from http://ant.apache.org. Ant 1.6.2 was used for testing, but earlier versions should also work.

The JVM used for installing the examples needs to be a full Java Development Kit, because it needs to compile Java source files. Ensure that the JAVA_HOME environment variable points to a JDK installation.

The sample directory includes the following files and directories:

build.xml: The Ant build file

src: Java source directory

webapp: A directory containing other files required by the examples, such as JSPs and Spring configuration files, which are copied directly to the JasperServer web application directory

reports: A directory containing example JRXML files that use the sample custom data sources

Take the following steps to install the samples in your JasperServer web application (this can be built from the source code or the delivered version of JasperServer):

At the command line, change directories to the custom data source sample directory (<js-install>/samples/customDataSource)

Edit `build.xml` and set the `webAppDir` property to the root of the JasperServer web application.

Run the Ant command (see above) with no arguments, which will execute the default target, which is named `deploy`. The `deploy` target will run the following tasks:

Compile the Java source under the `src` directory

Deploy the compiled Java class files to the JasperServer web application

Deploy files under the `webapp` directory to the the JasperServer web application

Restart the application server

9.1.2.1 Custom Bean Data Source

The custom bean data source implementation creates a data source from a collection of Java beans declared in the source code. Its Spring bean definition file is located in <js-install>/samples/customDataSource/webapp/WEB-INF/applicationContext-sampleCDS.xml. It An example of a report using this data source is located in <js-install>/samples/customDataSource/reports/simpleCDS.jrxml.

9.1.2.2 Webscraper Custom Data Source

The webscraper custom data source implementation can fetch a web page, decode the HTML, and extract selected data which is turned into field values in the data source. Its Spring bean definition file is located in <js-install>/samples/customDataSource/webapp/WEB-INF/applicationContext-webscraperDS.xml.

These are the configuration items for the datasource:

URL: An HTTP URL which refers to the HTML page containing the desired content

DOM path: An XPath expression which locates HTML elements to be turned into rows in the data source

Field paths: XPath expressions for each field defined in the JRXML which are used to locate the field value within each row selected by the DOM path

The implementation creates a data source by taking the following steps:

Uses the URL to issue a GET request for an HTML page.

Converts the HTML response into XML using JTidy (http://jtidy.sourceforge.net).

Uses the DOM path to select XML elements from the converted response.

Create a new data source row for each selected element

For each field, use the field path to determine the content for each field

The data source has two parameters: the URL of the web page, and the XPath which determines that elements in the HTML page become rows in the data source. The parameters can be specified either by a data source definition in the repository or by a query string in the JRXML.

The example reports for this data source read a web page from http://www.craigslist.org and extract a list of items for sale. The report `reports`/webscrapertest.jrxml has no query defined. Instead, it relies on an instance of the custom data source that has been created in the repository. Typical parameters to use with this data source are:

URL: http://sfbay.craigslist.org/search/car/eby?query=&neighborhood=62

DOM Path: /html/body/blockquote[2]/p

The reports/webscraperQEtest.jrxml example contains a queryString element which specifies the URL and the DOM path. It should be used without defining a data source instance, because JasperServer will not run the query executer for this particular implementation if a data source is defined for the report unit.

9.1.3 Creating a Custom Data Source

A custom data source consists of Java code, a message catalog, and a Spring bean definition file that configures all the parts of the implementation with JasperServer. This section defines the steps for implementing a custom data source.

9.1.3.1 Files used by a custom data source implementation

Type of files	Path (relative to web application directory)	Description
Spring bean definition	WEB-INF/applicationContext-name.xml	Defines Spring beans needed to configure the data source. Choose a unique name starting with "applicationContext-" and ending with ".xml"
Message catalog	WEB-INF/bundles/xyz.properties	Defines messages used by the data source implementation (this path is referenced in the Spring bean definition file).
Implementation classes	WEB-INF/lib or WEB-INF/classes	Any Java code required by the implementation.

9.1.3.2 Implementing the ReportDataSourceService Interface

A custom data source requires an implementation of the ReportDataSourceService interface, which is responsible for setting up and tearing down data source connections in JasperServer. It relies on:

void setReportParameterValues(Map parameterValues): called before running a report - creates resources needed by JasperReports to obtain a JRDataSource and adds them to the parameter map

void closeConnection(): clean up any resources allocated in setReportParameterValues()

9.1.3.3 Defining Custom Data Source Properties

A custom data source can define properties that can be used to configure each data source instance differently, in the same way that a JDBC datasource has properties for JDBC driver class, URL, user name, and password. The definition of properties is covered later (refer to section "" on page), but you will need to consider what properties you want to use while implementing your ReportDataSourceService.

There are two kinds of properties:

- Editable properties are string values. When you use the JasperServer data source wizard to create an instance of your custom data source, you can enter values for the editable properties using a text field. These values are persisted when you save the data source.
- Hidden properties can be of any type, but their values are determined by the Spring configuration file, so they are not persisted, and are not visible in the data source wizard. They can be used to give your ReportDataSourceService implementation access to a Spring bean instance. For an example of a hidden property, see the repository property in the custom bean data source definition below.

These property values are set by the custom data source framework after it instantiates your ReportDataSourceService implementation. You need property setters and getters corresponding to the property name; for example, if you defined a property with the name foo you will need getFoo() and setFoo() methods.

9.1.3.4 Implementing Optional Interfaces

If you wish to use the value of the queryString in the JRXML to obtain your data source, you need to create implementations of the JRQueryExecuter and JRQueryExecuterFactory interfaces.

JRQueryExecuterFactory has this method:

- JRQueryExecuter createQueryExecuter(JRDataset dataset, Map parameters): return a JRQueryExecuter for the given dataset and parameter map.
- JRQueryExecuter has these methods:
- JRDataSource createDatasource(): returns the actual data source based on the parameter map that was passed to the JRQueryExecuterFactory above; most likely, you will create a JRDataSource implementation suitable for your data source.
- close(): called when report filling process is done with the data source.
- cancelQuery(): called to clean up resources if the report filling process is interrupted.
- If you wish to provide validation in the user interface for creating and editing custom data source instances, you need to create an implementation of CustomDataSourceValidator. It has the following method:
- validatePropertyValues(CustomReportDataSource ds, Errors errors): check parameters and call errors.rejectValue() with the appropriate property name and error code (defined in a message catalog, refer to section 9.1.3.5 "Creating the Message Catalog," next).

9.1.3.5 Creating the Message Catalog

The message catalog contains messages displayed by the data source wizard when creating and editing custom data source instances. The various types of messages are shown in the table below, along with conventions for choosing message names:

Message Type	**Naming Convention**
Name of custom data source type	Cdsname.name (where cdsname is the value of the name property of the custom data source).
Name of custom data source property	Cdsname.properties.propname (where propname is the value of the property name).
Validation messages	Cdsname.any.message.code (there is no convention enforced, but JasperSoft recommends starting with cdsname for consistency).

For example, the webscraper message catalog contains the following:

```
webScraperDataSource.name=Web Scraper Data Source
webScraperDataSource.properties.url=URL
webScraperDataSource.properties.path=DOM Path
webScraperDataSource.url.required=A value is required for the URL
webScraperDataSource.path.required=A value is required for the DOM path
```

9.1.3.6 Defining the Custom Data Source in Spring

To configure the data source, you must add an instance of CustomDataSourceDefinition to the Spring bean definition file. This class has the following properties:

Name	**Required**	**Value**
factory	▲	(Fixed value) ref="customDataSourceFactory" (this is the bean that manages all the custom data sources).
name	▲	Unique name used to refer to custom data source in messages, etc.
serviceClassName	▲	A class name for your ReportDataSourceService implementation.
validator		An instance of your CustomDataSourceValidator implementation.
propertyDefinitions		A list containing a map of information about each property used by the custom data source (details on map entries are listed in the table below).
queryExecuterMap		Map with query languages (uses language attribute of JRXML queryString element) as keys, and JRQueryExecuterFactory class names as values.

The propertyDefinitions property is a list of maps, each one describing a property of the custom data source implementation. These are the entry keys used currently:

Name	Required	Value
name	▲	Name of property, which matches a JavaBean property in the ReportDataSourceService implementation, and is also used in message catalog keys
default		A default value for the property
hidden		Properties with hidden set to true are set to fixed values using the "default" entry. They are not be editable in the UI or persisted. This is handy for making Spring beans accessible to ReportDataSourceService implementations

The following XML creates a CustomDataSourceDefinition bean for the custom bean data source example:

```
<bean id="myCustomDataSource"
class="com.jaspersoft.jasperserver.api.engine.jasperreports.util.CustomDataSourceDefinition">
    <property name="factory" ref="customDataSourceServiceFactory"/>
    <property name="name" value="myCustomDataSource"/>
    <property name="serviceClassName" value="example.cds.CustomSimplifiedDataSourceService"/>
    <property name="validator">
        <bean class="example.cds.CustomTestValidator"/>
    </property>
    <property name="propertyDefinitions">
        <list>
            <map>
                <entry key="name" value="foo"/>
            </map>
            <map>
                <entry key="name" value="bar"/>
                <entry key="default" value="b"/>
            </map>
            <map>
                <entry key="name" value="repository"/>
                <entry key="hidden" value="true"/>
                <entry key="default" value-ref="repositoryService"/>
            </map>
        </list>
    </property>
</bean>
```

9.1.3.7 Configuring the Message Catalog

To configure your message catalog with JasperServer, add a bean definition similar to the following to the Spring definition file that you created above. For the value of the value property, substitute the location of your message catalog file, omitting the .properties extension. It is not necessary to edit the messageSource bean definition in applicationContext.xml.

```
<bean class="com.jaspersoft.jasperserver.api.common.util.spring.GenericBeanUpdater">
    <property name="definition" ref="addMessageCatalog"/>
    <property name="value" value="WEB-INF/bundles/cdstest"/>
</bean>
```

9.1.4 Installing a Custom Data Source

Add all the files required by the custom data source implementation to the JasperServer web application and restart the application server.

The new custom data source type appears in the list of choices for data source type when you create a new data source in JasperServer. If the new type is selected, JasperServer displays a form containing the list of properties you configured.

When the form is submitted, the parameter values are validated with the CustomDataSourceValidator implementation and appropriate validation messages are displayed.

9.1.5 Using a Custom Data Source

When defining `<queryString>` in JRXML, use a `language` setting that your custom data source supports.

When you create a report and are prompted for the data source to use, any custom data sources created by the administrator are displayed.

If you select a locally defined data source, you can choose the new custom data source type and edit it, just as if you were creating a data source in a folder (see section 9.1.4 "Installing a Custom Data Source" on page 86).

9.2 Report output channels

When running a report in JasperServer, the result can be delivered in several supported formats such as HTML, PDF, RTF, XLS and CSV. The report viewer page, where reports are viewed initially in HTML format, has a toolbar on top with buttons for exporting the current report to the above mentioned formats. A similar document export process occurs when reports are scheduled and the output is delivered in one of these formats.

The document export process is controlled using the `WEB-INF/applicationContext.xml` file where for each of the supported export formats there is a bean with properties that help fine-tune the export.

9.2.1 Excel output

The Excel output channel is controlled by the `xlsExportParameters` bean inside the `WEB-INF/applicationContext.xml` file. JasperServer is shipped with a default configuration that produces a data-centric Excel output where graphic elements are ignored, spacer cells are removed and the data type of cells is preserved.

The following properties can be set for the Excel exporter:

Property Name	Default	Description
`detectCellType`	`true`	Preserve the type of the original text field expressions and use it for the cell data type
`onePagePerSheet`	`false`	Each report page should be written in a different XLS sheet
`removeEmptySpaceBetweenRows`	`true`	Empty spaces that could appear between rows should be removed or not.
`whitePageBackground`	`false`	Force cell white background
`ignoreGraphics`	`true`	Export text elements only
`collapseRowSpan`	`true`	Collapse row span and avoid merging cells across rows
`ignoreCellBorder`	`true`	Do not draw the cell border
`fontSizeFixEnabled`	`true`	Decrease font size so that texts fit into the specified cell height
`maximumRowsPerSheet`	`0`	Maximum number of rows allowed before continuing on a new sheet

In report templates, Java specific format patterns are used to format dates and numbers (see JasperReports documentation for details). But these patterns might not work inside the Excel document viewer program, as their syntax might not be fully supported there.

In such cases, the Java format patterns have to be converted to equivalent proprietary format patterns using the `formatPatternsMap` bean inside the `WEB-IN/applicationContext.xml` file, where pairs of equivalent patterns can be added.

9.2.2 CSV output

The CVS export options can be changed by modifying the csvExportParameters bean configuration inside the WEB-INF/applicationContext.xml file.

For the moment, only the field delimiter can be configured for the CSV output channel.

9.3 Configuring Login Page

Starting with version 2.1, JasperServer comes with a new login page which displays welcome information about the product. If you want to switch to the simpler login page that was shipped prior to version 2.1, you can alter the paramResolver bean inside the WEB-INF/applicationContext.xml file by changing the line:

<prop key="/login.html">login_welcome</prop>

into

<prop key="/login.html">login</prop>