Elixir Repertoire User Manual

Release 2.5.0

Elixir **Technology Pte Ltd**

Elixir Repertoire User Manual: Release 2.5.0

Elixir Technology Pte Ltd

Published 2013

Table of Contents

List of Figures

Chapter 1
About Elixir Repertoire

Overview

Elixir Repertoire is an integrated Business Intelligence suite, designed for enabling Intelligent Enterprises to compete effectively in this fast-moving globalized market. Addressing the end-to-end information life cycle, one can activate the entire suite or the individual products as required to aggregate and transform, present and deliver, navigate and visualize, monitor and activate enterprise data.

Launching Repertoire Designer

There are 2 ways of launching Repertoire Designer.

1. Go to Repertoire/bin directory. Double-click on *Elixir Repertoire.exe*. Repertoire Designer will be launched.

2. Open a command prompt window. Go into Repertoire/bin directory. Enter the following:

   ```
   java -jar Repertoire-Launcher.jar
   ```

   After a few moments, Repertoire Designer will be launched.

Launching Repertoire Designer with Specific Report, DataSource or Dashboard

User can launch Repertoire Designer and open a specific file at the same time. This is applicable to Report(.rml), DataSource(.ds) and Dashboard(.pml). In order to do this, user will need to launch Repertoire Designer using a command prompt window, which is the second method mentioned in the section called “Launching Repertoire Designer”. However, instead of the command mentioned previously, enter the following instead: java -jar Repertoire-Launcher.jar -maxWorkspace -initialFile repository:/<*PathOfFile*>.

Here is an example for launching Repertoire Designer with a Report:

```
java -jar Repertoire-Launcher.jar -maxWorkspace -initialFile
  repository:/ElixirSamples/Report/Components/Checkbox.rml
```

Repertoire Designer Features

Elixir Repertoire Designer is a design tool containing the following components:

- **Elixir Data Designer** for extracting, merging and processing data from a variety of datasources, either to generate direct data output (for example, Excel files or database records), or to feed data into Elixir Report and Elixir Dashboard Designers.

- **Elixir Report Designer** for designing report templates and rendering data into a variety of output formats, including PDF, Excel and HTML.

- **Elixir Dashboard Designer** for creation of dashboards, allowing interactive visualization and manipulation of data, combining interactive reporting and data analysis.

One other tool is provided:

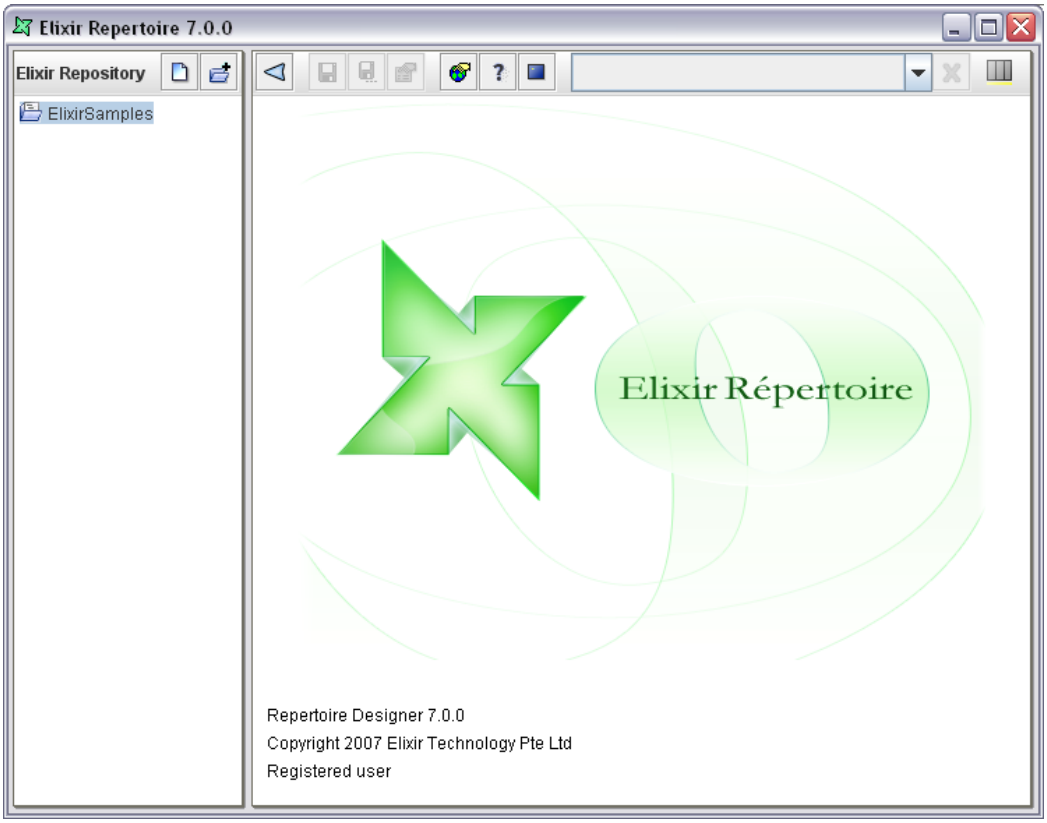
- **Elixir Report Interactive** for rendering reports using an editable snapshot of the data, requiring no datasource connection.

Chapter 2
The Elixir Interface

Overview

On launching an Elixir Repertoire application, you will see a window similar to that shown in Figure 2.1, “Elixir Repertoire Main Frame”. The window consists of three parts, the Elixir Repository, Action Bar and Workspace.

Figure 2.1. Elixir Repertoire Main Frame



The panel on the left is the Elixir Repository. The Action Bar is across the top on the right and the Workspace is the area currently filled by the logo and build information below it.

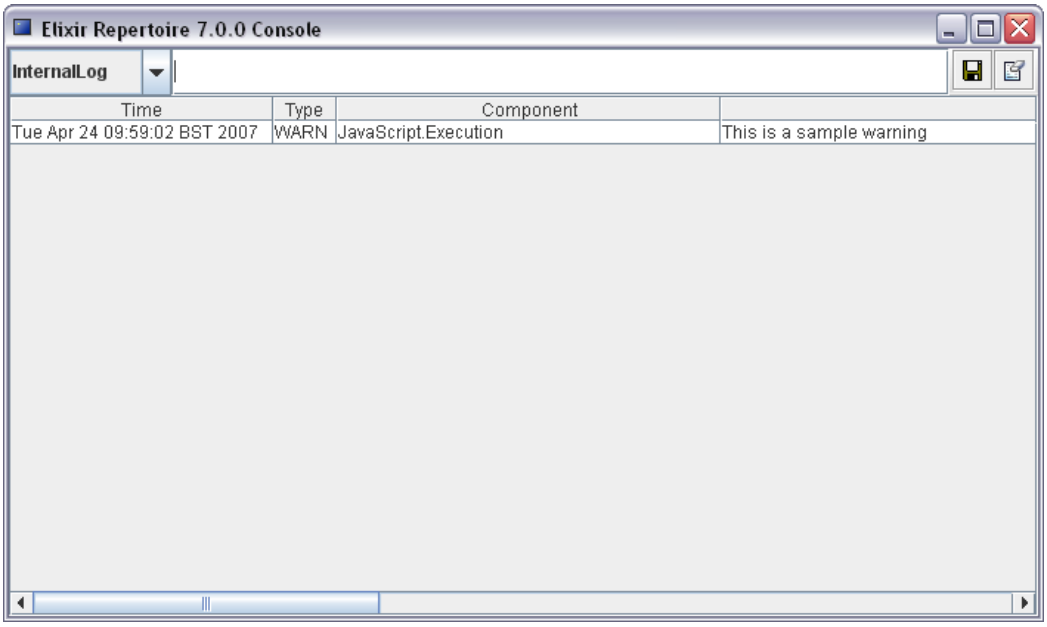
Action Bar

The action bar present at the top of the right panel contains the Expand/Collapse, Save, Save As, Properties, Global Properties, Help, Show Console, Administration Tools and Close View buttons. It also includes a Combo Box and the Progress Indicator.

- *Expand/Collapse:* The Expand/Collapse icon allows you to hide or show the Elixir Repository panel. The triangle points left to Collapse and then changes to point right, indicating that you can Expand again.

- *Save:* The Save icon will be enabled when the current view has been modified and not saved.
- *Save As:* The Save As option allows you to save the current view under a different name in the repository. Subsequent saves will use the new name. The original file remains unchanged, so this option is useful for versioning.
- *Properties:* The Properties icon is only enabled when a view is open in the workspace. On clicking the icon the appropriate wizard appears.
- *Global Properties:* The Global Properties icon allows properties of the toolset to be edited. The precise options available will vary based on the combination of Elixir tools and extensions that are installed.
- *Show Console*: On clicking the Show Console icon, the Console window appears as shown in Figure 2.2, “Console Window”. The Console window lists all the log details. The type of log details can be selected from the combo box. There are three types of log details displayed they are Internal Log, JavaScript and the User Log.

Figure 2.2. Console Window



When the Internal Log is selected, all the logged events from the launching of Elixir Ensemble are displayed. The JavaScript log details are displayed if any errors have occurred during the scripting process. The User Log is displayed if there are any errors in setting up the security parameters of the data sources.

Clicking the Clear icon clears the log. The Save icon is used to save a copy of the log to a file. This is a particularly useful file that can be sent to Elixir's support team so that they can assist with configuration problems.

- *Administration Tools:* On clicking of the Admin Tools... icon, the Administration Tools window appears. It provides a set of tools that help you administer jobs, JDBC driver libraries and the scheduling of calendars and triggers. For detailed information, refer to Elixir Administration Tools User Manual.
- *Combo Box:* The Combo Box present in the toolbar lists the various data sources and other views currently open in the workspace. The user can select the view they wish to work on by selecting from the list.

- *Progress Indicator:*The Progress Indicator indicates the progress of the application while the data is being loaded. This indicator includes a popup menu that allows certain long operations to be aborted.
- *Close View:* A left click will close the current view. Right-clicking will show a popup menu with a list of options:
    1. Close This View: On selecting this option, the currently active view is closed (same as left-click).
    2. Close Others: On selecting this option, the views other than the currently active view are closed.
    3. Close All: On selecting this option all the views are closed.

Elixir Repository

The toolbar icon at the top of the Repository panel is used to add new files and file systems. Each toolbar button launches a wizard to guide you through the process.

Types of File

CSS

CSS is used to create and edit a Cascading Style Sheets file. It can define the styles of a report, including background, fonts, header, footer and more.

Connection Pool

A Connection Pool allows connections to JDBC databases and connection properties to be shared amongst multiple datasources. Connection Pools are described in the Elixir Data Designer User Manual.

DataSource

The DataSource is the principal building block in an Elixir Repertoire solution. External data may be wrapped as a DataSource and multiple DataSources may be merged and processed by a Composite DataSource. Each kind of DataSource is described in the Elixir Data Designer User Manual.

HTML

HTML is used to create and edit a Hypertext Markup Language file. HTML can be imported into Dashboard and embedded in Frame or Page.

JavaScript

Elixir Repertoire is an extensible tool, allowing users familiar with Java or JavaScript to integrate the tool with their code or customize the tool to meet their own requirements. Many parts of the tool allow JavaScript to be embedded, however if the same scripts are required in multiple locations it is preferable to create a common JavaScript file in the repository and then to import it into each DataSource, Report or Dashboard that requires it. This editor is discussed further in Chapter 5, *Elixir JavaScript Editor*.

Job

Job is used to add tasks that are intended to run together.

Map

Map is usually created to represent geographic areas and adds new dimension to data visualisation. More information about Map can be found in Elixir Map Designer documentation.

Perspective

A Perspective holds the definition for a dashboard - a set of interactive views, including tables, charts, reports and data cubes. Dashboards are stored in Perspective Markup Language (an XML syntax) with a file extension .pml. Perspectives are described in the Elixir Dashboard Designer manual.

Report Template

A Report Template holds the specification for a report, including the layout of report components and reference to the datasources needed to provide the data during rendering. Reports are stored in Report Markup Language (an XML syntax) with a file extension .rml. Report Templates are described in the Elixir Report Designer User Manual.

Safe

A Safe is an encrypted text file, used for holding valuable text-based information, such as database passwords etc. The Safe is described in Chapter 4, *Elixir Safe*.

Text

A simple text editor/viewer is included for reviewing log files, CSV data etc.

Universe

A Universe is an interface to data. The user can add more than one Universe to handle higher-volume data processing. Universes hold many tables. There are two types of Universes: Repository and JDBC. In the Repository Universe, each table maps to a DataSource. Universes can exist anywhere in the Repository, but are only accessible when mapped to a Universe name. All modules that use Universe reference the universe name instead of the path. This enables you to easily switch between different Universes by re-mapping. Universe mapping is a feature only available for administrators.

XML

XML is used to create and edit an Extensible Markup Language file. It can be imported into DataSources and pass the data.

Types of FileSystem

Regardless of the type of file system, each has a name. This name must be unique as it forms the base for each repository path. For example, if a file system called FS contains a datasource called mine.ds, then the full name of the datasource is /FS/mine.ds. Elixir Repository always uses '/' for path separators, like URLs, regardless of platform. Anywhere a URL can be specified, you can use the special protocol `repository:` to refer to files. For example, the URL

```
repository:/FS/mine.ds
```

refers to the datasource described above.

Local FileSystem

This file system maps directly to the file system of your operating system. Usually this is used while designing the data. However, for server side deployment other file systems might be preferred as they ease the problem of relative path configuration.

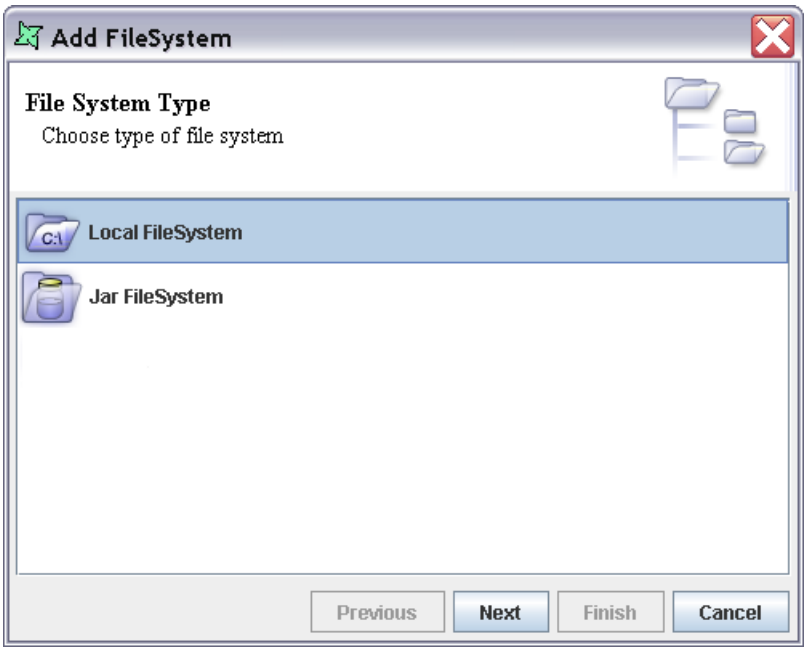
Jar FileSystem

This file system is easier to deploy. The design and data is stored either in local or remote machine in a compressed file i.e. a zip file in Jar format.

Working with FileSystems

Clicking the Add FileSystem icon shows the "Add FileSystem" wizard which lists the types of FileSystem that can be defined.

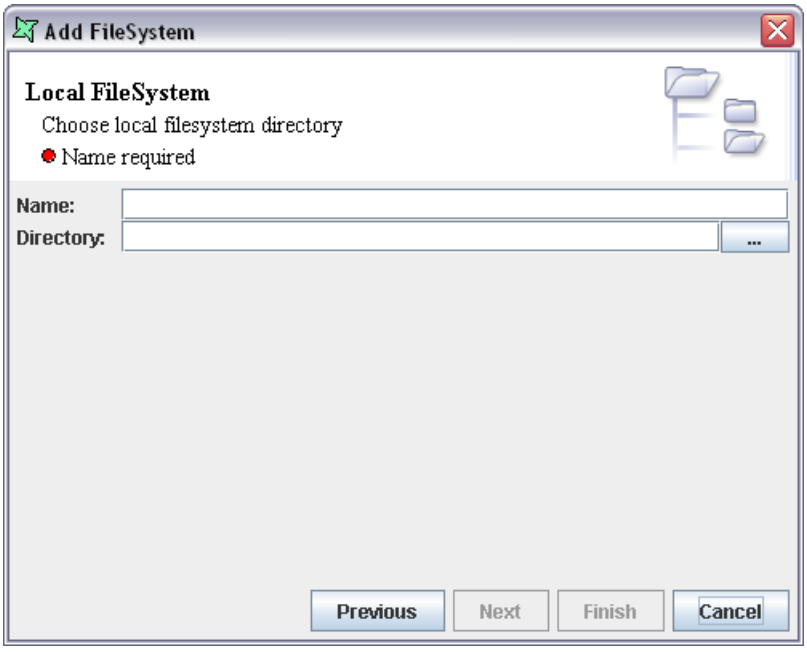
Figure 2.3. Add FileSystem



Local FileSystem

Choose the Local FileSystem option, as shown in Figure 2.3, “Add FileSystem” and click on the Next button to see the screen as shown in Figure 2.4, “Add Local FileSystem”. On this page a local directory name can be entered. Alternatively by clicking the button on the right of the text field, a directory can be chosen from a dialog. Upon clicking the Finish button the Local FileSystem will be created and displayed in the Elixir Repository tree.

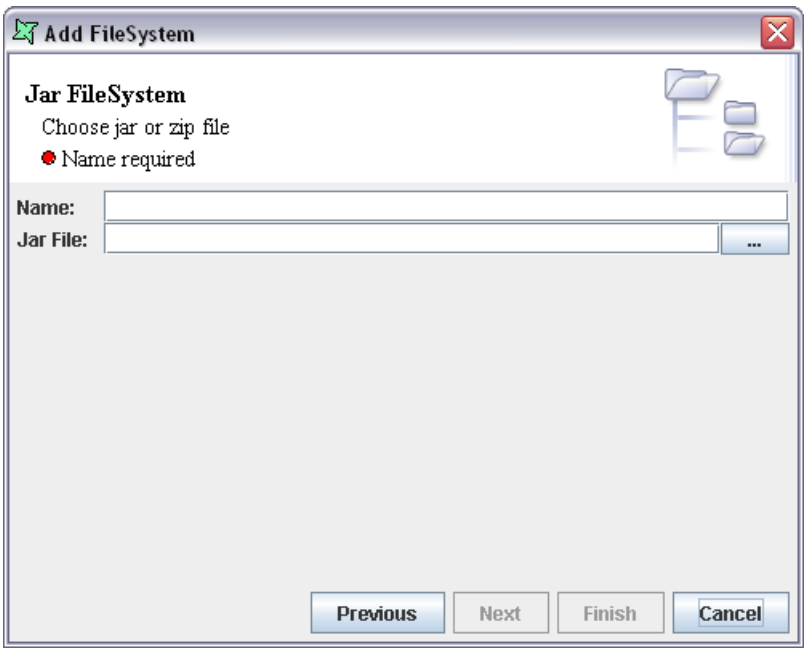
Figure 2.4. Add Local FileSystem



Jar FileSystem

The Jar file system allows read-only access to files in either a jar file or a zip file. Add a FileSystem and choose the Jar FileSystem option and click on the Next button to see the screen as shown in Figure 2.5, “Add Jar FileSystem”. On this page the name can be entered. The directory path of the jar or zip file can be typed in the Jar File text box. Alternatively by clicking the button on the right of the text field, a jar or zip file can be chosen from a dialog. Upon clicking the Finish button the Jar FileSystem will be created and displayed in the Elixir Repository tree.

Figure 2.5. Add Jar FileSystem



Working with Files

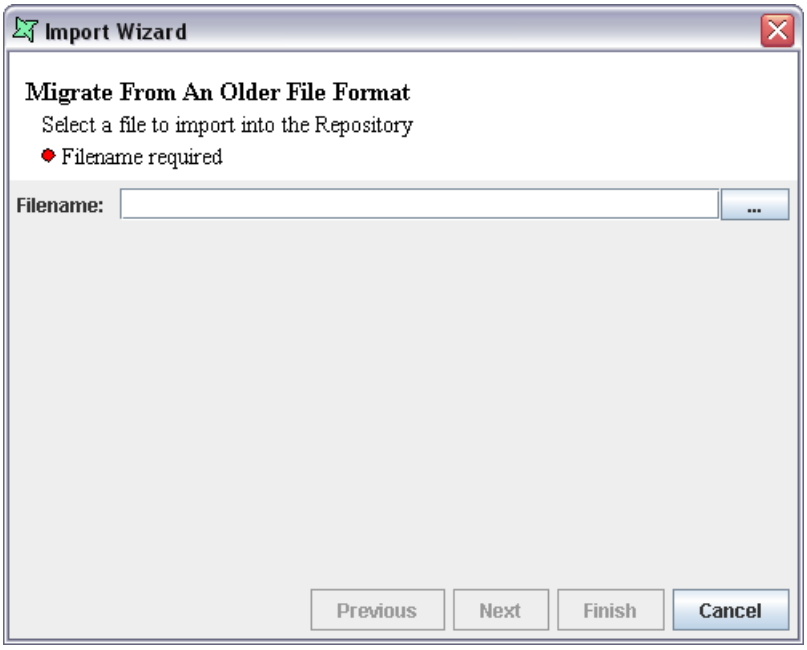
After adding a file system to the Repository, different files, for example DataSources, Dashboards, Reports and Folders can be added to it.

Each file system, folder and file has a popup menu showing the available actions. File systems and folders have similar options:

- *Add*: Using this menu item several kinds of file or a folder can be added. Each will invoke a specialized wizard to guide you through the creation process.
- *Import*: Import allows files to be imported into the repository. This option is for files that need some migration or extraction to be used in the Repertoire tool. Primarily this is for Elixir Report 4 .sav and .template files.

  Choose a file system or folder into which the data source or template should be imported, select Import from the popup menu. The "Import Wizard" appears as shown in Figure 2.6, “Import Wizard”. You can use the browse button to choose the file to be imported. On clicking Finish, the file gets imported into the repository. A .sav file contains many datasources, so each will become a separate .ds file in the chosen location.

  Figure 2.6. Import Wizard



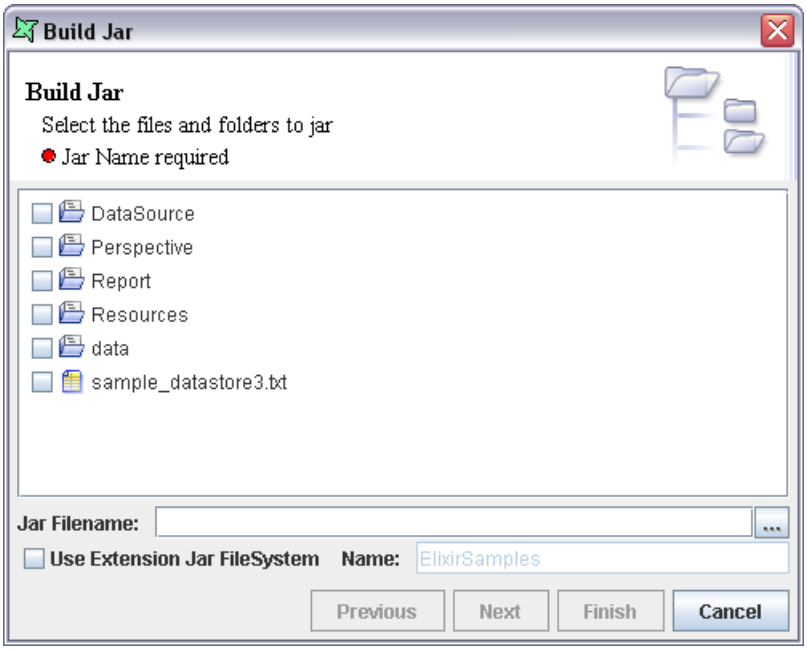
  ### Note

  It is only necessary to import files from Elixir Report prior to version 5.0. Version 5.0 and later files can be read by the tool without any special import/extraction/migration step.

- *Refresh*: (Only applies to file systems, not folders.) On selecting this option the file system will be refreshed and any changes to the files and folders made outside the tool will be visible.
- *Compact*: (Only applies to local file systems, not folders.) On selecting this option the file system will be compacted, meaning that all backup files (.bak) will be deleted to reclaim disk space.
- *Build Jar*: A jar file can be built using the subtree of files or folders in the repository. Select a file system, and select Build Jar from the popup menu. The "Build Jar" dialog window opens displaying

all the folders and files present in the repository as shown in Figure 2.7, “Build Jar”. The files or folders to be jarred are selected. On clicking the browse button, the Save dialog box pops up allowing the output jar filename to be defined.

Figure 2.7. Build Jar



The Extension Jar Filesystem allows a jar to be created with a special loader. If this jar file is placed in the Elixir Repertoire ext directory it will be automatically loaded as a filesystem next time the tool is started. The name field allows the jar filesystem to be given a name (if you leave it blank it will default to the filename). The name should be chosen carefully as all items within the generated jar will be known based on the full path, which includes the filesystem name at the start.

- *Remove*: A file system or folder can be removed from the repository when this menu item is selected.
- *Delete*: A file or folder can be deleted from the filesystem when this menu item is selected. Delete actually only hides the file by giving it a .bak extension. Files and folders with .bak extensions are hidden from Repository views, but can be restored (if you made a mistake) by renaming them using operating system commands. To permanently delete a file or folder, you should Compact after deleting, which will remove all .bak files.

Recent Files

Below Elixir Repository, files that have been opened recently will be listed under the `Recent Files`. Files that are deleted will also be listed, but when user tries to open them, an error will occur informing the user that the file is no longer available.

A maximum number of five files will be listing under `Recent Files`. By right-clicking within `Recent Files`, user will have the option to clear the history and files listed will be removed from the panel.

The Workspace

Each component of Elixir Repertoire consists of one or more views that will display in this area. See the documentation for the individual components for a description of the features available from that view.

Chapter 3
Elixir Repertoire Universe

Overview

There are two categories of Universe: Repository Universe and JDBC Universe. For example, you can create a Repository Universe and map it for use in Ad-hoc Dashboard.

Repository Universe is for DataSources (except Composite), for backward compatibility for Repertoire. JDBC Universe is for JDBC relational datasources.

All the modules that use Universe, use the Universe name not the path. Therefore you can easily switch to a different Universe by re-mapping. A Universe must be given a name before it can be seen in the Ad-hoc tools. For details on Universe mapping and access control, refer to Elixir Administration Tools User Manual .

This chapter describes how to create a Universe, work on Universe tables and more.

Create a Universe

Complete the following steps to create a Universe:

1. Right-click a filesystem or sub-directory. On the pop-up menu, select **Add > Universe**. The Add Universe window opens.
2. Select **JDBC** or **Repository** from the list. Enter a name for the Universe.
3. Click **Finish**. The Universe file is successfully created with a .universe extension. Double-click to open the Universe Designer and make further changes.

Tables

JDBC Universe Tables

A JDBC Universe allows data to be read from a relational database using a JDBC driver. If your data is all defined in a relational database, and a JDBC driver is available, then JDBC Universe is recommended. Before getting started, add the JDBC driver into /opt/elixir/lib/, so that it will be available to all Java Virtual Machines.

Right-click the **Tables** node, and select one of the following options:

Configure connection pool...: JDBC allows access to data via SQL statements. For example, you can install MySQL Workbench and use com.mysql.jdbc.Driver for the JDBC Universe. Modify the host and dbname in the **URL** field, enter user name and password, and click **Test Connection**. If the test of connection succeeds, you have the option to define Connection Pool Parameters on the next page. On the Connection Pool JDBC Properties page, you can set any customized properties for your JDBC driver by setting the keys and values. After you complete the settings, click **Finish**.

Add tables...: Tables from the relational datasource will display in the Add Tables window. You can choose from these tables, or invert your selection. Click **OK**.

Add join table: Join SQL combines records from two or more tables from the relational database. Join SQL combines columns from two tables by using values common to each. There are often two columns with the same name, and the Universe will add the table name as a prefix to ensure the columns are unique. Click **Finish**, and the corresponding SQL will be calculated and executed to determine the schema.

Add custom table: Enter your SQL expression into the Custom SQL field. If you have entered some expression in the SQL panel, it will be the default content when you create a new custom table. Dynamic parameters are supported, and you can use ${substitutions}. Click **Finish** and the schema will be automatically inferred. If parameters are involved, you will be prompted to enter a parameter value.

Add folder...: This option helps you categorize tables, which is extremely useful when you have a large number of tables to work on. Fill in the folder name and click **OK**.

Edit: This option is available for each table under the **Tables** tree. Right-click a table, and select this option. The Edit Table window opens. Select from the Generated SQL list to show the entire table, or hide certain columns. You can create several Universe tables that are backed by the same physical table, but expose different columns. This is much more efficient than fetching all columns and filtering locally.

Show SQL: This option is available for each table under the **Tables** tree. Right-click a table, and select this option. It will display the SQL that will be executed for the table. Click the **Load Data** blue button to load the table contents.

Paste: This option enables you to copy tables from another JDBC Universe and paste here.

Repository Universe Tables

A Repository Universe is similar to a small repository, which may include hundreds of tables. Each table maps to a datasource, which is a full copy of the original datasource. If you change either the table or the mapping datasource, the other one remains unchanged.

A Repository Universe supports all kinds of datasource except Composite, which is replaced by transforms.

Right-click the **Tables** node, and select one of the following options:

Add table...: This option enables you to create a new table. Choose a datasource type and fill in datasource parameters. Click **Finish**.

Add folder...: This option helps you categorize tables. Fill in the folder name and click **OK**.

Import datasource...: This option enables you to import from an existing datasource. Choose from the repository and click **OK**.

Paste: This option enables you to copy tables from another Repository Universe and paste here.

Settings

The **Settings** tab displays table settings, including schema, parameters, table attributes and column attributes. The **Settings** tab also enables you to add transforms for data manipulation.

Schema

The **Schema** tab displays column names and data types. The PrimaryKey column is highlighted with a key icon. If a primary key is not defined, you can add one using the column attributes.

Parameters

Column Attributes

The following types of attributes are supported:

- **Enumeration:** By defining the attributes of individual fields, Repertoire Data Designer can then validate the record values using the cleansing process. For example, we might specify that the field "Gender" is a nominal enumeration of "M" and "F". The cleansing process would then warn us of any records containing "m" or "Female". Ordinal enumeration attributes allow us to specify the order to display data, while retaining the original order. For example, we might specify that the field "Fruit" is an ordinal enumeration of "Apple" and "Orange". Only "Apple" and "Orange" records can pass the validation in cleansing process.
- **ForeignKey:** This type of attributes is created by JDBC DataSources, and therefore cannot be edited.
- **Format:** Define the format.
- **Nullable:** Specify whether the data can be Null values.
- **PrimaryKey:** A PrimaryKey is necessary for all tables in the Repository Universe. Ensure that each table has a unique PrimaryKey column. This is important for Ad Hoc Dashboard. For most cases, a table has only one PrimaryKey, therefore the sequence number is 0. If you are working on a datasource without obvious PrimaryKey such as FruitSales.ds, add a Sequence Transform to create a new ID column, and add a Column Attribute to mark it as a PrimaryKey. Because Fruit-Sales.ds uses a composite PrimaryKey, an alternative approach is to add a Column Attribute to mark “Company” as “PrimaryKey:0”, and add another Column Attribute to mark “Fruit” as “PrimaryKey:1”.
- **Range:** Specify the start value and end value.
- **Comments:** Enter text for comments.
- **RegExp:** Define and test syntax.

Transforms

The **Transforms** tab enables you to create and edit transforms to a schema. Click the plus sign, and the Transform Wizard will be invoked. For detailed information on transform categories, refer to Elixir Transform User Manual.

JDBC Transforms

Compare

For more information on Compare Transform, refer to Elixir Transform User Manual.

Credentials Check

This restricts access to rows where a specified **Credentials Field** matches the user credentials. You have the option to specify users or groups who can access all records.

User credentials refer to the user name and all group names where the user belongs to. In credentials, users and groups are not distinguished. A blank string matches no credential, while "*" matches any credential.

The following table shows an example of the input (**Credentials Field:** Field 3):

Field 1	Field 2	Field 3
87	1348.50	Sales
75	1162.50	Sales
64	3628.80	Marketing
51	2891.70	Marketing
107	6473.50	*

The following table shows the output for users and group with the "Sales" credential:

Field 1	Field 2	Field 3
87	1348.50	Sales
75	1162.50	Sales
107	6473.50	*

The following table shows the output for users and group with the "Marketing" credential:

Field 1	Field 2	Field 3
64	3628.80	Marketing
51	2891.70	Marketing
107	6473.50	*

Date Filter

This enables the filtering of Dates and Timestamps. You can filter by the current day of month, month and year. You also have the option to filter by offsets, either earlier or later, from the current day of month, month and year.

The following table shows an example of the input:

HireDate	EmployeeName
2011-05-17	Chad Mattson
2009-03-25	Richard Davis
2012-01-17	Amy Franks
2009-05-09	Willie Costa
2010-05-30	Rebecca Smith

When you set the **Filter Field** to HireDate, choose to keep records with **Offset Month**, and set the **Offset** value to -1 (assume the current month is June), the following table shows the output:

HireDate	EmployeeName
2011-05-17	Chad Mattson
2009-05-09	Willie Costa
2010-05-30	Rebecca Smith

Formula

This provides a variety of functions which can calculate a new field from your formula. The types of functions include Numeric, String, System and TimeDate. For more information on how to write a formula with functions, refer to the online documentation for your chosen database and version. These documentation are easily searchable online but too numerous and the links are too likely to change to be included here.

The following shows an example of using the CONCAT function in this formula:

```
CONCAT ([Name], " joined the company on ", [HireDate])
```

Name	HireDate	Output
Richard Davis	2009-03-25	Richard Davis joined the company on 2009-03-25
Amy Franks	2012-01-17	Amy Franks joined the company on 2012-01-17
Willie Costa	2009-05-09	Willie Costa joined the company on 2009-05-09

The following shows an example of using the SUBSTRING function in this formula:

```
SUBSTRING ([Name], 1, 3)
```

Name	HireDate	Output
Richard Davis	2009-03-25	Ric
Amy Franks	2012-01-17	Amy
Willie Costa	2009-05-09	Wil

Sort

For more information on Sort Transform, refer to Elixir Transform User Manual.

Rows

The **Rows** tab displays either the original data of a table, or the output of column data if there are transforms created. Click the **Load Data** blue button to refresh.

Inspector

The **Inspector** tab enables you to review the contents of a column. It displays column information, value frequency based on the data type, as well as the chart type and plot.

Summary

The **Summary** tab displays the column names, data types and PrimaryKey information.

SQL

Select a table under the **Tables** tree, and click the **SQL** tab. It will display the SQL that will be executed for the table. The SQL statements are editable. Click the **Load Data** blue button to load the table contents.

Chapter 4
Elixir Safe

Introduction

Elixir Repertoire tools include a Safe file type. A Safe is for holding valuable text-based information, for example passwords. Each Safe is encrypted on disk using a password. If you lose the password, no one can get it back for you - the contents are gone for good.

To create a Safe file, choose a location in the Repository and from the popup menu select Add > Safe... A wizard will appear, as shown in Figure 4.1, “Add Safe Wizard”.

Figure 4.1. Add Safe Wizard

Enter a unique filename and enter and repeat the password. The password will be required each time you open the file. If the password is lost, the file will be unreadable. When you have entered all required details and pressed Finish you will be presented with a text editor. You can enter plain text here, in any format and for any purpose and it will be encrypted automatically each time it is saved.

On subsequent loading, the Safe file prompts for a password to decrypt the file, as shown in Figure 4.2, “Safe Password Prompt”.

Figure 4.2. Safe Password Prompt

Using a Safe

As we've seen, the Safe file contains plain text, so you can use it for any purpose. For example for storing all the passwords you need to remember. If you store text in the form:

```
# This is a comment
Name=Value
Another=Something Else
```

then the file can be read as properties, which can be used to parameterize a report or datasource. To use these external properties within a report, you put a script in your Report OnRenderBegin:

```
var props = elxfn.getSafeProperties("/Workspace/Data.safe","pass");
setParameters(props);
```

where "pass" is the password to unlock the Safe. The call to setParameters adds the name=value pairs into the report parameter list, so they will be passed to datasources etc. as required. Another benefit of this approach is that it allows you to share a common set of properties across multiple reports and datasources.

If you don't want to hardcode the password in the script, you can get it from another parameter:

```
var pass = getParameterValue("Password");
var props = elxfn.getSafeProperties("/Workspace/Data.safe",pass);
setParameters(props);
```

This can either read Pass from the Report parameters, or prompt for a dynamic parameter as you choose.

Chapter 5
Elixir JavaScript Editor

Introduction

Elixir Repertoire tools include a JavaScript editor, connected to a JavaScript engine for immediate testing and evaluation of expressions. This editor allows the creation of a library of common JavaScript functions that can be added to any other JavaScript evaluation, such as a Report Function Definitions script or a Composite DataSource script.

The JavaScript editor supports syntax colouring to make it easy to see at a glance the different aspects of the code. You can evaluate code using the Popup menu and selecting an option.

- Do It: executes the selected text as a JavaScript expression.
- Show It: evaluates the selected text as a JavaScript expression and outputs the result of evaluation back into the editor.
- Reset: resets the JavaScript engine to the default state.

Function Availability

Different parts of the Repertoire toolset expose different objects and functions that you can interact with. For example, a Report script can utilise the Renderer object, to query the mime-type and interact with the RawReport object. However, it is an error to reference these objects when executing an Ensemble script (e.g. a Composite DataSource script), because these objects are not available in the Ensemble context. You need to be aware when creating a reusable JavaScript file what context it will be running in so that you can ensure you are accessing the appropriate objects.

Referencing JavaScript

Once you have created a JavaScript (.js) file holding some functions you want to reuse, you need to import it into your tool scripts. The syntax for importing a javascript file is:

```
importScript("/ElixirWorkspace/MyScripts/Core.js");
```

To avoid repeated import of the same scripts, the import should be done in a script that is only executed once. For example the Function Definitions of a Report template, or the Script tab of a Composite DataSource.

Chapter 6
Function Reference

Overview

Many Elixir designers incorporate some form of data manipulation:

- Taking the Count of the customers in Washington using a Cube in a Composite DataSource
- Showing a Running Sum of your accumulated sales in a Report
- Highlighting the Percent of frozen goods sold to a chosen retailer in a Dashboard

There are a standard set of functions such as Count, Sum and Percent mentioned above that can be used throughout the Repertoire Suite.

General Functions

Average

The Average takes a sequence of values, sums them and divides by the number of values. This function will work on all numeric types. The result will always be a double.

Comma Separated List

The Comma Separated List takes a sequence of values and returns a single string holding these values separated by commas. For example, if the selected fields from three records are Apple, Orange, Strawberry, you will get back a String "Apple, Orange, Strawberry".

Comma Separated Set

The Comma Separated Set takes a sequence of values and returns a single string holding the discrete values separated by commas. This means duplicates are removed. For example, if the selected fields from five records are Apple, Berry, Apple, Orange, Berry, you will get back a String "Apple, Berry, Orange" - each item is only listed once.

Count

The Count result is the number of values that the function has received. This is most useful in cubes where usually different numbers of records are partitioned into each cell.

First

The First function always replies with the first value that it receives.

Last

The Last function always replies with the last value that it receives.

Max

The Max result is the largest value received. This function works for all comparable types, including strings and dates as well as numbers.

Median

The median function takes a sequence of values, which should have been sorted in increasing order and returns the value in the middle. If the number of elements is even then it returns the average of the two values closest to the middle. Hence this function only works on numbers and dates. For instance, if there are numbers 1,2,2,3,3,4,5,6. There is an even number of values, so the middle or median is between the first and the second three. As they are the same the median is three, but if they were different say if the median was between 3 and 4, we would do (3+4)/2=3.5.

Min

The Min result is the smallest value received. This function works for all comparable types, including strings and dates as well as numbers.

Percent

The Percent result is the sum of all values received divided by the sum of all values available. The result will be a number between 0 and 1, which can be formatted as a percentage (for example using the Field Format in Report).

Percent100

The Percent100 result uses the same algorithm as Percent, but the value returned is scaled into the range of 0 to 100.

PercentCount

The PercentCount result is the count of all values received divided by the count of all values available. The result will be a number between 0 and 1, which can be formatted as a percentage (for example using the Field Format in Report).

PercentCount100

The PercentCount100 result uses the same algorithm as PercentCount, but the value returned is scaled into the range of 0 to 100.

Standard Deviation

The Standard Deviation is the square root of the Variance (see below).

Sum

The Sum result is the summation of all values received. The result will always be a double.

Variance

The Variance is the measure of how spread out a distribution is. It is computed as the average squared deviation of each number from its mean(average). For example, for the number 1, 2 and 3 the mean is 2 and the variance is:

```
[(1-2)²+(2-2)²+(3-2)²]/(3-1) = 0.5
```

Year To Date

The *Year To Date* sums value(s) of data that has the year corresponding to the current year of the system.

Month To Date

The *Month To Date* sums value(s) of data that has the month and year corresponding to the current month and year of the system.

Additional Cube Functions

Nested Percent Variants

Cube makes available four special functions, which are Nested versions of the Percent, PercentCount, Percent100 and PercentCount100. The functions behave similarly to the basic percent functions but the result is the derived from all values received divided by all values in that group. Therefore, unlike Percent, which will give a percentage out of all records, Nested Percent will give the percentage out of all siblings in the same level of the cube.

Here's an example, just looking at one Cube axis:

```
                        Percent    Nested Percent
USA                     100        100
      AZ                 30         30
           Apple         15         50
           Orange        10         33
           Pear           5         17
      WA                 70         70
           Apple         40         57
           Orange        20         29
           Pear          10         14
```

Using Percent for USA-AZ-Apple would sum the value of Apples sold in Arizona and sum the value of all fruit in the USA and work out the percent from that. In other words, the value would be the percent of Arizona Apples compared to all fruit (15% in this example). Using Nested Percent on the other hand works out the sum relative to the siblings. In this case USA-AZ-Apple would give the percentage of Apple sales compared to all fruit sales in Arizona (AZ-Apple+AZ-Orange+AZ-Pear). In this nested case that is 50%.

In the Percent column all the items at the same level in the tree add up to 100%. In the Nested Percent column, each element's children add up to 100%.