

Elixir Scheduler User Manual

Release 3.0.0



Elixir Technology Pte Ltd

Elixir Scheduler User Manual: Release 3.0.0

Elixir Technology Pte Ltd

Published 2014

Copyright © 2014 Elixir Technology Pte Ltd

All rights reserved.

Java is a trademark or registered trademark of Oracle Corporation in the United States and other countries. Microsoft and Windows are trademarks of Microsoft Corporation.

Table of Contents

1. About Elixir Scheduler	1
Overview	1
2. Jobs and Tasks	2
Overview	2
Job	2
Parameters	3
Log	3
Tasks	4
Action	4
AdminCommand	5
Ant	5
CallJob	5
DataLoop	5
Echo	6
FileLoop	6
GenerateData	6
Loop	6
OnError	7
Parallel	7
RenderReport	7
Script	7
SendMail	7
3. Triggers	9
Overview	9
Trigger	9
Basic Information Page	9
Job Schedule Page	9
Trigger Active Period	10
Time Zone	10
Trigger Calendar	10
Misfire Configuration	10
Hourly, Daily and Monthly Triggers	10
4. Calendars	12
Overview	12
Mark periods for daily events	12
Mark periods for weekly events	12
Mark periods for monthly events	12
Mark days with specific dates	13
Mark days with CRON expressions	13
Time zone selection	13
5. Cookbook	14
Overview	14
Job Recipes	14
Render All Reports In A Folder	14
Trigger Recipes	15
Trigger On The First Monday Of The Month	15
Trigger On The Last Friday Of The Month	17
Trigger On The Last Working Day Of The Month	18
Trigger On The Last Working Day Of The Quarter	19

List of Figures

1.1. Elixir Scheduler	1
2.1. Job Designer	2
2.2. Grouping of Parameters	3
2.3. Log File	4
2.4. DataLoop	6
2.5. Encoding in Attachment of SendMail task	8
4.1. Calendar Wizard	12
5.1. FileLoop Tree	14
5.2. File Criteria	15
5.3. Target Parameter	15
5.4. Trigger Wizard	16
5.5. Job Schedule	17
5.6. Every Last Friday Of The Month	18
5.7. Every Last Working Day Of The Month	19
5.8. Every Last Working Day Of The Quarter	20

Chapter 1

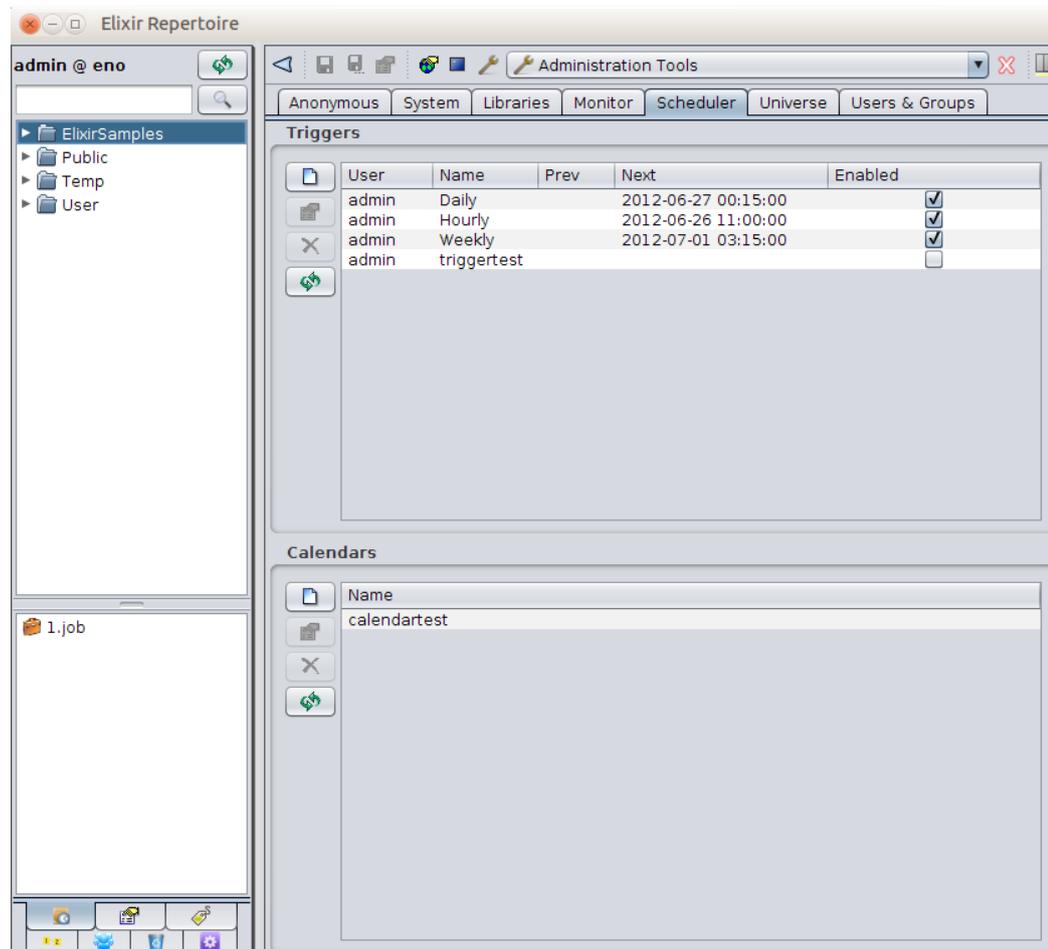
About Elixir Scheduler

Overview

Elixir Scheduler provides control over triggers and calendars on the Elixir Administration Tools panel. Through the Scheduler interface, triggers and calendars may be created, tested and modified. These will help you schedule jobs and fire triggers more easily.

The Scheduler resides in the Designer and uses the same repository tree on the left pane, as shown in [Figure 1.1, “Elixir Scheduler”](#).

Figure 1.1. Elixir Scheduler



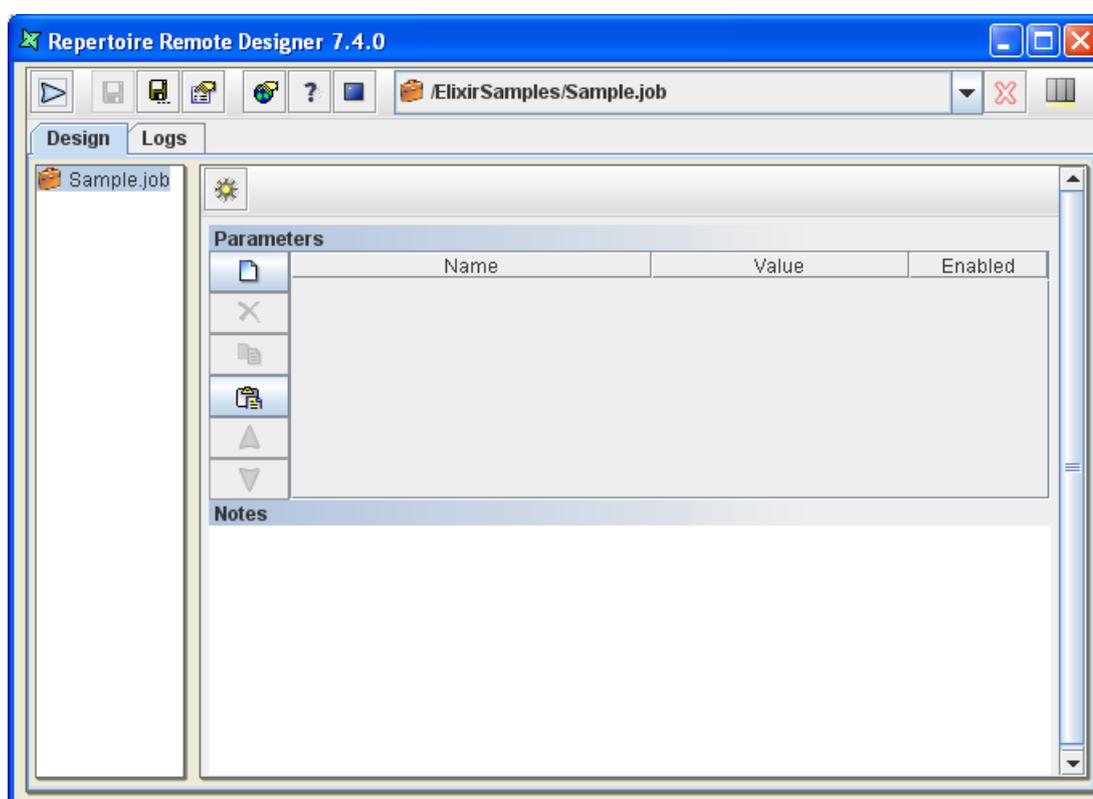
Chapter 2

Jobs and Tasks

Overview

A Job is a set of tasks that are intended to be executed together. The Job creation interface is shown in Figure 2.1, “Job Designer”

Figure 2.1. Job Designer



When a job is open, there is a task tree on the left of the workspace along with a panel, which shows the details of the currently selected task. When a new job is created, only the job is visible as the root of the tree. Use the popup menu on the Job item in the tree to add new tasks to the job. If you have several tasks in the job you can reorganize them by dragging them around.

Job

Selecting the job itself at the root of the task tree allows parameters to be set. These parameters can be used by all the tasks within the job and it is useful to have a single point of maintenance. Job parameters can either be hard-coded, for example User = Bill, or can be dynamic, for example, User = \${User Name##Bill}. This indicates that the triggering code should supply a value for User, which may be by prompting for "User Name", and that the default value is Bill. See the Elixir Repertoire User Manual for details of how dynamic parameters can be used throughout the Elixir Repertoire suite. In the case of Elixir Scheduler, if the job is triggered manually, then a popup form will prompt for

dynamic parameter values to be entered. However, if the job is triggered by a scheduled Trigger, then it is the Trigger that needs to supply any dynamic parameters.

Each job may define a log file in the repository to which it will write the progress of execution. This is configured on the Logs tab in the job workspace. If no log name is specified, the job will still execute, but you won't be able to review the progress of execution. You can choose the level of logging using the combo box on the right of the Log File entry. If you choose Debug you will get details of the start and stop time of every task as it is executed. Info, Warn and Error provide increasing smaller amounts of logging output, which can make it easier to spot problems that might otherwise get lost in the most verbose log output.

Parameters

Some jobs may require the use of parameter(s) and the necessary parameter(s) should be enabled. If the name of a parameter is highlighted in blue or red, a tooltip will be seen to let the user know what can be done when the cursor rolls over the name of the parameter.

Users can change the value of a parameter by clicking on the Value column of a particular parameter, then edit the value directly or click on the "..." button to bring up a dialog box where the user can choose a value from the choices available. Parameter of different type will have a different dialog box.

- If the parameter is of "choice" type, the dialog box will show a list of available choices.
- If the parameter is of "date" type, the dialog box will show a date chooser.
- If the parameter is of "lookup" type, the dialog box will show a list of available choices loaded from the datasource at runtime.
- If the parameter is of "password" or "string" type, the "..." button will not be present.

In the case whereby the Scheduler is unable to identify the type of parameter, the parameter will be treated as a string type.

Users can group certain parameters such that the parameters can be enabled or disabled as a whole. To be able to do so, add a new parameter with name "<group>" as seen in [Figure 2.2, "Grouping of Parameters"](#)

Figure 2.2. Grouping of Parameters

Parameters			
	Name	Value	Enabled
	<group>		<input type="checkbox"/>
	DBUser	user	<input type="checkbox"/>
	OrderID	1004	<input type="checkbox"/>
	<group>		<input checked="" type="checkbox"/>
	DBUser	admin	<input checked="" type="checkbox"/>
	OrderID	1004	<input checked="" type="checkbox"/>

A "<group>" defines a starting of a new group and the ending of the previous group. When the checkbox for a particular "<group>" is checked or unchecked, all the parameters that belong to the group will be enabled or disabled respectively.

Log

A log file is a JSON file that records the events that happen in the background when a job is running. Log files are stored in the /User/{User_Name}/logs/ directory in Repository. If the job runs successfully, a similar screen like the following figure, [Figure 2.3, "Log File"](#), will be shown.

Figure 2.3. Log File

Date: 2012-07-18 Elapsed time: 882ms Status: No error			
Time	Type	Text	Action
16:21:13	Logging	Begin	
16:21:13	Started		
16:21:13	Authentication	eno:admin	
16:21:12	Received	/host/default/first	
16:21:13	Started		
16:21:13	elx.JobEngine	Begin elx.JobEngine	
16:21:13	Job	Job Start	
16:21:13	elx.ActionHelper	/User/admin/1.form set enabled=true	
16:21:14	Job	Job End Successfully	
16:21:14	elx.JobEngine	End elx.JobEngine	
16:21:14	Ended		
16:21:14	Ended		
16:21:14	LogFile	/User/admin/logs/tmp-0000000027.json	
16:21:14	Logging	End	

If the job encounters an error, the log viewer will show one or more actions on the right hand side for follow-up purpose, including **Open**, **View** and **Follow**, allowing you to look at job results, view more details of exceptions or follow handover to other jobs. For example, the **Follow** action helps you navigate through a potential tree of logs produced by multiple engines working together to solve a problem. The parent log will show the status and elapsed time of each child processing log, saving you from the trouble of following irrelevant logs.

The log viewer also allows you to send support mail with this log, or save locally to attach to support mail. The support mail address is configurable in **Administration Tools > System > Targets > support**. Enable the log viewer to send emails by entering the recipient's address into the support target, or disable it by deleting the support target.

Tasks

Each task added to a job has a name and a space for notes to be entered. Both of these are optional, and are to allow documentation of the job so that subsequent administrators and maintainers can interpret it in future. The following sections describe each individual task in detail. Some are composite tasks, which means they can also contain child tasks. Composite tasks are useful for loops, optional routines and parallelism. Where text can be entered in a task - that is, in any text field, variable substitutions such as `${Name}` can be used. The appropriate parameter values will be substituted when the task is executed.

Note

A task can be disabled by right-clicking on it and selecting **Disable Run**. Apply similar steps to enable the task.

Action

The Action task allows one job to perform an action when it is running. The following actions are for your selection:

- **Enable/Disable Form:** This action enables you to control when a form will accept input while jobs are scheduled to run at certain days or times. To enable a form to accept data, select the form from the repository or enter the repository path, and select the **Enabled** checkboxes.

AdminCommand

The AdminCommand task will execute only when the job is run by an administrator. The following actions are for your selection:

- **Clean unused temporary files:** This command clears all those temporary files in /Temp/<user> folder in the Repository, which have not been accessed within the period of time specified on the Administration Tools panel.
- **Clean unused log files:** This command clears all those job log files in /User/<user>/logs folder in the Repository, which have not been accessed within the period of time specified on the Administration Tools panel.
- **Clean inactive sessions:** This command clears all those sessions that have had no activity within the period of time specified on the Administration Tools panel.
- **Clean inactive sessions:** This command cleans away old ZooKeeper logs to ensure the disk space is not wasted. The most recent three snapshots and logs will be retained.
- **Aggregate RequestTracker data:** This command will aggregate RequestTracker data from Zookeeper into a database, freeing Zookeeper resources and allowing dashboards and report access to the request tracking data.
- **Full backup:** This command performs a complete backup of the chosen filesystem. It writes the results into an archive file in the chosen folder from which they can be manually restored. Choose the filesystem from the **FileSystem** dropdown list. Specify the folder in the **To folder (url)** field.
- **Incremental backup:** This command performs an incremental backup of a filesystem. It writes only those files modified within the chosen duration into an archive file in the chosen folder, from which they can be manually restored. Choose the filesystem from the **FileSystem** dropdown list. Specify the folder in the **To folder (url)** field. Specify the duration that changes occurred from the **Changes in the last** dropdown list.

Ant

The Ant task allows Ant build files, as described by the Apache Foundation, to be launched from within a job. Ant is extremely flexible, allowing file operations, ftp, mail etc. to be coordinated. In order to use the Ant task, you need only define the location of the Ant file (typically called build.xml) that you wish to launch. Optionally, you can specify the name of the target within the build file that should be executed. If no target is named, then the default target will be chosen. Please see the Ant manual at <http://ant.apache.org/manual/> for more details on the Ant file format and options.

CallJob

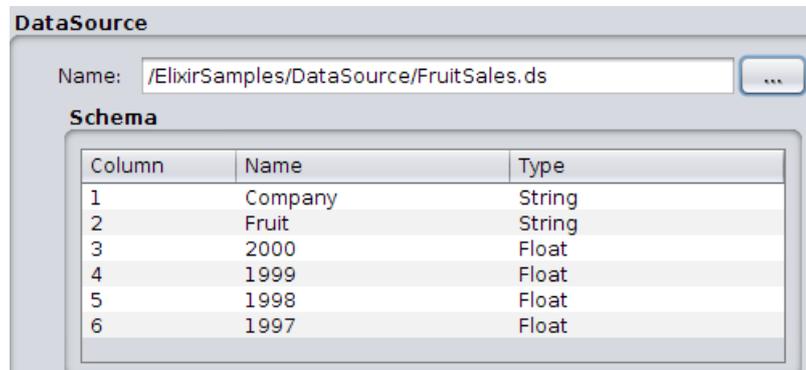
The CallJob task allows one job (master) to invoke another (child). The child job is identified by name, and may have parameters passed to it to control its execution. When the CallJob task is executed within the master job, it locates the child job by name and executes it within the scope of the master job. This means all parameters, scripts etc. of the master job are available to the child. In addition, the child job will write its log to the master job log. This task will only complete when the child job has completed.

DataLoop

The DataLoop task is a composite task that executes each of the child tasks sequentially. The DataLoop has a datasource that supplies records. For each record, the DataLoop will set the execution scope to include the record fields, so they can be accessed by name from the children. The children will then be executed sequentially. The process will repeat until all records in the datasource have been used. In order for users to know that the datasource is valid, the fields of the datasource will be displayed

under Schema, as shown in Figure 2.4, “DataLoop”. If not, the text in the Name field will be red in colour.

Figure 2.4. DataLoop



Echo

The Echo task allows information to be written to the log. As with all tasks, text values may include variable substitution strings of the form `${Name}` which will be replaced by the appropriate parameter value during execution. This is useful for logging progress of a job and also for debugging. For example, you can echo a message such as

```
Job run by ${User} to render ${Report}
```

FileLoop

The FileLoop task is a composite task that executes each of the child tasks sequentially. The FileLoop requires the name of a folder in the repository. You can also define file criteria that allow the files within the folder to be filtered, for example by extension or modification date. The FileLoop will iterate through all files and execute the child tasks sequentially, once for each file that matches the criteria. While executing the children, the loop makes available the current filename so that it can be used in variable substitutions. The name of the variable is taken from the Parameter Name value defined in the task panel, the default name is `FileName`. If no criteria are entered, all files in the folder will be used. Selecting the Recursive option repeats the process for all subdirectories under the chosen folder.

GenerateData

The GenerateData task invokes an Elixir Data Designer's Composite DataSource to output to one of its DataStores. The name of the datasource is required, along with the name of the datastore within it that is to receive the records. It is also possible to pass parameters with the invocation which are then accessible within the datasource. Note that only parameters explicitly identified in the GenerateData task will be passed to the datasource - those defined within the scope of the job are not implicitly available to the Ensemble engine.

Note

When the job is running, information regarding the progress of the job can be seen in the Log tab. The starting, processing and the ending of the job will be recorded. The log file can be sent to support mail, or saved locally to attach to support mail. This applies to all jobs.

Loop

The Loop task is a composite task that executes each of the child tasks sequentially. The loop requires a simple repeat count that indicates the number of times the children should be executed. If there are

three children, A,B and C, and the repeat count is two, then the sequence of execution will be A,B,C,A,B,C. Loop has two common uses, first, to skip a set of tasks, and second to perform the same tasks repeatedly, for example for performance testing or benchmarking.

OnError

The OnError task can only be the child of a job and cannot be added under other composite tasks such as Loop. It should always be the last top-level task listed under the job. There can only be one OnError task for each job. When the user try to add another one, he will be prompted to overwrite the existing one. Similarly, when the user copy an OnError task from one job to another that has an existing OnError task, he will be prompted to overwrite the existing one.

OnError task is triggered to run when error occurs during the job execution. The user can configure the number of times to re-try the job when it hits an error. The default value is zero, which means there is no re-trying.

A user can also add tasks under the OnError task, which are executed when the OnError task runs. Typically, it will be those tasks that clean up the leftover from previous job failure such that Repertoire can re-try the job again in a consistent way.

If there are no errors during job execution, the OnError task is skipped.

Parallel

The Parallel task is a composite task that executes all of the child tasks at the same time. Each child task runs as a separate thread and the parallel task will not complete until all of the child tasks have completed. This task allows a thread count to be specified. If no count is given then each child task runs in a separate thread. Otherwise a pool of threads is created and each child task uses a thread from the pool when it is available. For example, by using a thread count of two, only two of the child tasks would be running simultaneously.

RenderReport

The RenderReport task invokes the Elixir Report engine to render a report on the server. The required information includes the report name, the chosen target on the server and the mime-type that is required. Additional parameters may be passed to the report and to the target. Report parameters would include information needed to control the datasource or rendering process. Target parameters would include information needed by the chosen target. For example, if the target is userhome, you should specify the parameters for filename and folder. Alternatively, if the target is mail, you need to identify the parameter for the recipient (this depends on how the target is configured on the server - i.e. whether the recipient name has been pre-defined). Note that only parameters explicitly identified in the RenderReport task will be passed to the rendering engine - those defined within the scope of the job are not implicitly available.

Script

The Script task allows Elixir Scheduler to invoke JavaScript and Java codes within the job. Variable substitutions of `${Name}` variables is performed before executing the code.

SendMail

After the SendMail task is created, the email panel is empty. You will need to fill up From and To field in order to send out an email. The rest of the fields are optional.

The user can right-click in the Message field to find the list of parameters that can be embedded in the email. The parameters are evaluated and substituted with real values on the server side before the

email is being sent out. If there are no parameters available, the popup menu will show *No available parameters*. If the the user define some parameters in the task that encloses the SendMail task (such as the job), he will also be able to find the parameters in the popup menu. However, disabled parameters defined in the job will not be shown in the popup menu. Only enabled parameters are shown.

The user can click on the Add button in the email panel to add attachments. Add more than one attachment by clicking on Add again. To remove the attachment, simple click on the Remove button.

Note

SendMail task allows encodings to be included in attachments so as to allow the receiver to see the original text correctly. An example is shown in [Figure 2.5, “Encoding in Attachment of SendMail task”](#).

Figure 2.5. Encoding in Attachment of SendMail task

The screenshot shows a 'Send Mail' configuration window. At the top, there is a 'Name:' field. Below it is a header 'Send Mail'. The main area contains several fields: 'From:' with the value 'sending@abc.com', 'To:' with 'receiver@abc.com', 'Cc:' (empty), 'Subject:' with 'This is the subject', and 'Message:' with 'Message is added here'. Below the message field is a hint: '(Hint: Right click in message field to insert dynamic parameters)'. At the bottom, there is an 'Attachments:' field containing '/attachment/여보세요.txt|euc_kr', a 'Browse...' button, a 'Remove' button, and an 'Add' button.

Chapter 3

Triggers

Overview

A Trigger is a mechanism for deciding when to invoke a job. Most commonly, triggers are time-based - for example invoke the job every Tuesday at 6am. All triggers have a name, an enabled/disabled flag and some specialized fields for identifying when to run.

All triggers have a Start and Stop time and can only fire between those times. It is possible to set the Stop to be `Never`, so that the trigger is always operational.

Trigger

There are six pages in the Trigger Wizard.

- **Basic Information Page:** Defines the name of trigger, location of the job and parameters required.
- **Job Schedule Page:** Configures the schedule of the trigger.
- **Trigger Active Period:** The start and stop time of the trigger.
- **Time Zone:** The time zone where the user wants the job to be fired.
- **Trigger Calendar:** The Calendar to refer to. (Calendars are discussed in [Chapter 4, Calendars](#))
- **Misfire Configuration:** set start/end timeout intervals, and choose what to do when the trigger does not fire as it is supposed to.

Basic Information Page

On the first page of the Trigger Wizard you need to enter the name of the trigger and the location of the job. If the job requires parameters, you will be prompted to enter them here. In the `Parameters` table, the rows might appear in red or blue. Red parameters indicate a value has been supplied that is no longer required by the job, and it can be safely deleted (it will be ignored anyway). Blue parameters are required for the job and so appropriate values should be entered.

Job Schedule Page

On the Job Schedule page of the Trigger Wizard, you can schedule a job to run once only, daily, weekly, monthly, at regular intervals or controlled by a CRON expression.

For weekly events, you need to choose the day(s) of the week when the job should run. For monthly events, you need to choose the day(s) and month(s) to run the job. You can also choose specific day(s) within each month. For jobs that run daily, weekly or monthly, the job will fire at the same time on the scheduled days.

Trigger Active Period

This page of the wizard lets the user configure the absolute time range in which a trigger can fire. The actual firing time itself is governed by the values entered on the previous page - this page only indicates the earliest and latest times that it could possibly fire.

Note

By default, a trigger becomes active upon completion of the wizard and never stops. Also, when the trigger has completed the job run, it will be set to disabled automatically. If user would like to run the job again, the trigger will need to be enabled and make adjustment in *Trigger Active Period*.

Time Zone

This page enables you to select the time zone, which can be your current location or a different geographical location, according to where you want the job to be fired. The time zone selection is not limited by your locale. With the correct time zone selected, job firing will be accurate, which follows the rules below:

- The time should always follow the changes of daylight saving time. Therefore 01:00 is the same whether Summer or Winter.
- When a time doesn't exist (a gap due to the clocks moving forwards), job firing will choose the first valid time after the gap.
- When the time exists twice, job firing will choose the first time and ignore the second.

Trigger Calendar

If you need to exclude certain periods from the trigger schedule, such as public holidays, you can mark them on a calendar and point the trigger to the calendar. The trigger will not fire during the period(s) marked in the calendar. The calendar options are discussed in [Chapter 4, Calendars](#).

Misfire Configuration

This page allows the user to override the default job timeouts by setting start/end timeout intervals, and choose the action to be taken when a trigger fails to work the way it was supposed to.

- **Timeout for Start:** Set the start timeout value for the job. If the job does not get started within the start timeout value, the trigger will be considered misfired.
- **Timeout after Start:** Set the end timeout value for the job. If the job does not complete within the end timeout value, the trigger will be considered misfired.
- **Fire the trigger immediately upon detection:** This option enables the trigger to fire soon after the misfire is discovered.
- **Do not fire. Wait until the next scheduled time:** This option enables the trigger to fire in the next round.

Hourly, Daily and Monthly Triggers

The three triggers are used to invoke the Hourly, Daily and Weekly job files in /User/admin/cron folder in the Repository. The Hourly trigger starts at every hour sharp, and invokes a command task to clean inactive sessions. The Daily trigger starts at 00:15 every day, and invokes a command task to clean

the unused temporary files and logs files. The Weekly trigger starts at 03:15 am on every Sunday. These triggers start at different time to avoid conflict.

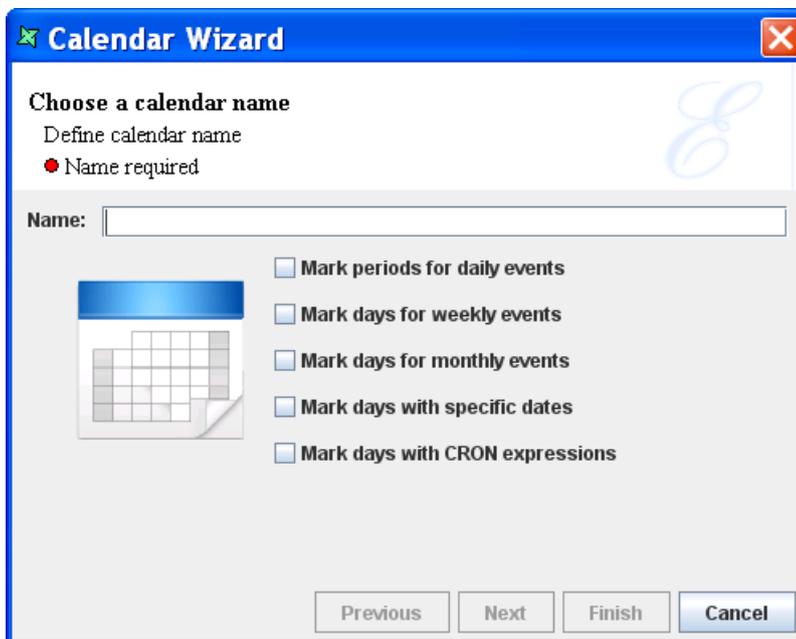
Chapter 4

Calendars

Overview

Triggers may use calendars to determine special date or times that should be excluded from the firing schedule. Upon adding a Calendar you will see [Figure 4.1, “Calendar Wizard”](#).

Figure 4.1. Calendar Wizard



Mark periods for daily events

You can use this option to mark ranges of the day when the trigger should not fire. For example, if you want to avoid performing an hourly data loading operation between midnight and 6 am, you would set Start Time 12 am End Time 6 am to exclude that range.

Mark periods for weekly events

This option allows you to choose which days of the week to exclude. For example, you could select Saturday and Sunday to prevent associated triggers from firing at the weekend.

Mark periods for monthly events

This option allows you to choose which days of the month to exclude. For example, you could select all but 1 and 15 in order to have the trigger restricted to fire on only the first and fifteenth of each month.

Mark days with specific dates

To prevent triggers being fired on public holidays, you can identify them using this calendar option. You can also choose whether the dates apply only for one year (some holidays change date each year) or whether the same date should be excluded every year. For example, to exclude triggers from firing on Christmas Day you would set the Date to Dec 25th and tick "Repeat every year".

Mark days with CRON expressions

The CRON expression option is the most powerful, but requires the most configuration. Remember that the expressions chosen are used to exclude not include times from the firing of a trigger. In most cases a combination of the previous options will be easier to maintain. If a trigger is scheduled to fire every hour, e.g. 00:00:00, 01:00:00, 02:00:00 etc. then you can use a CRON expression

```
0 0 /2 ? 0,6 *
```

to restrict the trigger to fire only once every two hours on weekends. The CRON sample does this by excluding 00:00:00, 02:00:00, 04:00:00 etc. for days 0 and 6 (Sunday and Saturday).

Note

You can attach more than one trigger to a job, so an alternative strategy for regular restrictions such as this would be to define a one hourly trigger for weekdays and a second two-hourly trigger for weekends. This would avoid any exclusion ranges and would perhaps be more flexible to maintain.

Time zone selection

After marking periods or days, you will be guided to the Time Zone page. Select the correct time zone where you want the job to be fired, which will be used when interpreting this calendar.

Chapter 5

Cookbook

Overview

This cookbook describes a number of typical uses for Elixir Scheduler.

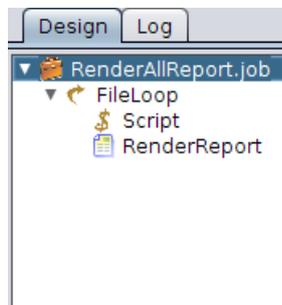
Job Recipes

Render All Reports In A Folder

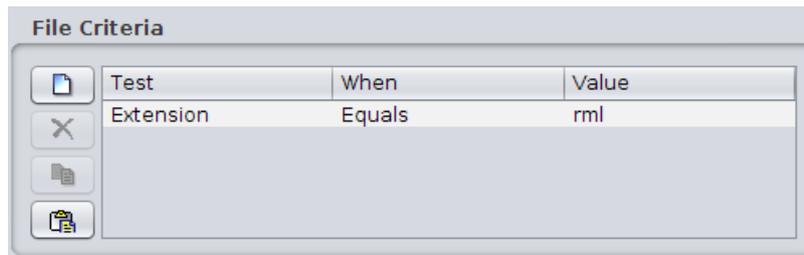
This section will show the user the steps to rendering all the reports in a specified folder into the desired output format. PDF will be the chosen output format in this case.

1. In a repository or folder, right-click and select **Add** and **Job**. Enter a unique name for the job. For instance, *RenderAllReport*, then click **Finish**. The job will be created successfully.
2. In **Design** tab, right-click on the job, select **Add**, then **FileLoop**.
3. Right-click on **FileLoop** and select **Script**. Then, add **RenderReport**. The structure will be similar to [Figure 5.1, "FileLoop Tree"](#).

Figure 5.1. FileLoop Tree



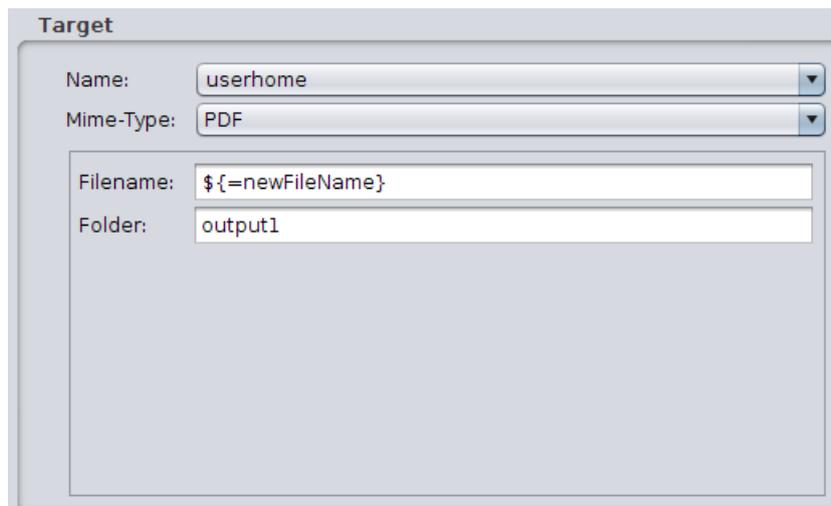
4. Click on *FileLoop*. In the **FileLoop** panel, enter the folder directory in the **Folder:** field. User can also click on **...** to select the folder. Check **Recursive** if the files are contained within the folders of folders.
5. In the **File Criteria** panel, add a new criteria. In **Test**, choose *Extension* in the dropdown list. For **When** condition, select *Equals*. As for **Value**, enter *rml* like in [Figure 5.2, "File Criteria"](#).

Figure 5.2. File Criteria

- In Script, enter the following. This is to retain the directory structure:

```
newFileName = FileName.replace(/.rml/g, ".pdf");
```

- In the Render Report job, enter `${FileName}` for the Name: field in the Report panel. This is so that the name of the PDF generated will take on the name of the rml file.
- For the Target panel, select `userhome` and `PDF` for the output target name and output mime-type from the dropdown list respectively. For the Filename: field, enter `#{=newFileName}`. For the Folder: field, enter the desired folder name, for example `output1`, as seen in Figure 5.3, “Target Parameter”.

Figure 5.3. Target Parameter

- If user wants to see the progress of file processed, user can click the Log tab to view the log.
- Now, run the job. After the job has finished running, the generated PDF files can be found in `/User/{User_Name}/output1/.../`, and the sub-folders will keep the same structure from the original folders which contain the reports.

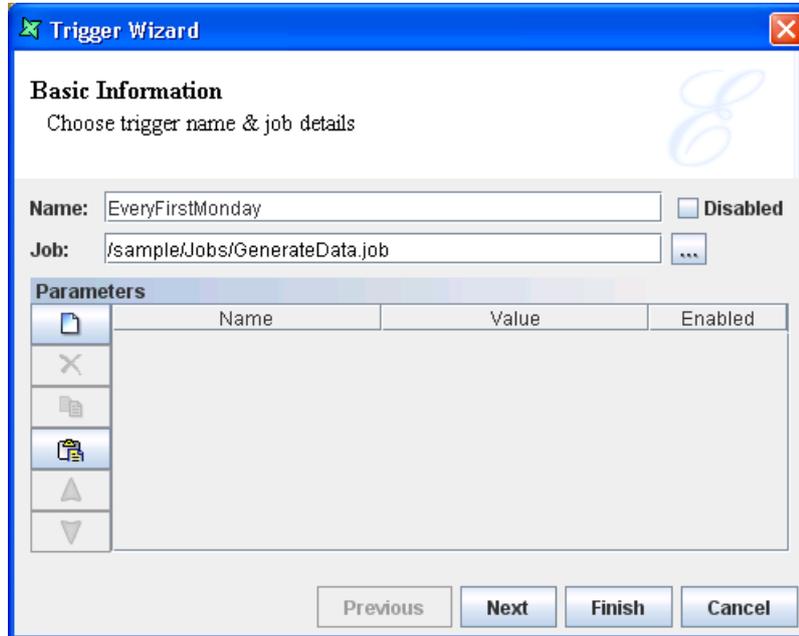
Trigger Recipes

Trigger On The First Monday Of The Month

- Before creating a trigger, user will need to have an existing job that is working perfectly.

- When a working job is available, create a trigger in `Administration Tools > Scheduler` by clicking `Add`. The Trigger Wizard window opens.
- In the Trigger Wizard, enter the user name and a unique name for the trigger. Select the job to be run as seen in [Figure 5.4, “Trigger Wizard”](#). If the job requires parameters to be entered, it will be done in this page of the wizard.

Figure 5.4. Trigger Wizard



- The next page of the Trigger Wizard is used to configure the schedule of the trigger, setting the time and frequency the trigger will run, like in [Figure 5.5, “Job Schedule”](#). To run the trigger on every Monday of the month, select `monthly` from the drop down list. For `Job fire time:`, enter the time the trigger should run on every Monday of the month. For `Months`, enter `*` to imply all months. Else, click on `Select` and select the months respectively. Then, select the second radio button and select `First` for the first drop down list and `Mon` for the second drop down list.

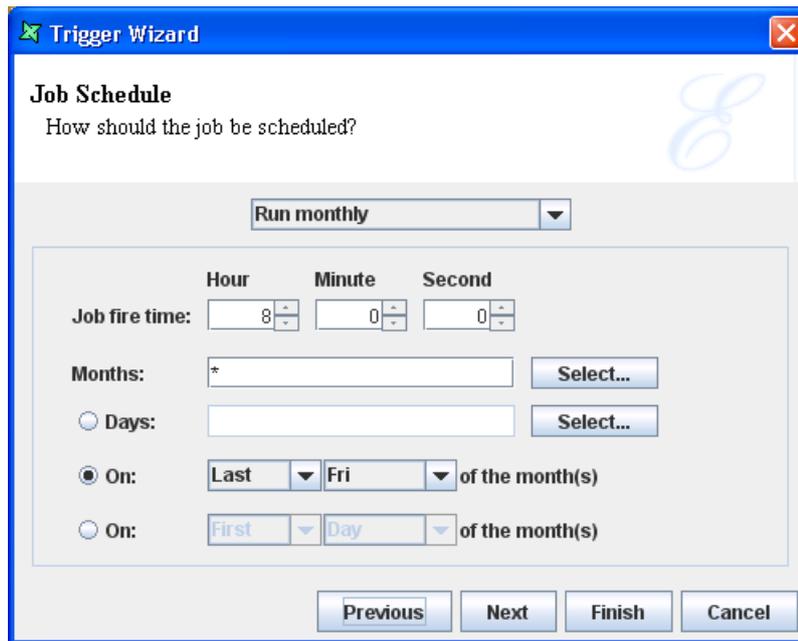
Figure 5.5. Job Schedule

5. The following page of the wizard allows the user to configure the active period of the trigger. If the trigger needs to run forever, select the checkboxes at *Now* and *Never* at . Else, select configure the starting and ending time respectively.
6. The Time Zone page is for users to explicitly specify the time zone where they want the job to be fired, which will be used when interpreting this trigger.
7. The Trigger Calendar page is for users to load the calendar file, explicitly excluding particular dates. However, this is not compulsory. Users can continue without defining a calendar file.
8. Misfire Configuration page is on the next page. This enables the user to override the default job timeouts and makes the trigger know what to do when a misfire occurs.
9. Last but not least, the Security page which is exactly the same for all elements in the Designer.
10. Clicking on *Finish* will complete the creation process.
11. This trigger will now run at 8a.m. on every first Monday of each month, never ending.

Trigger On The Last Friday Of The Month

This trigger is very similar to the one in the section called “Trigger On The First Monday Of The Month”.

1. Follow the instructions from step 1 to step 3 in the section called “Trigger On The First Monday Of The Month”.
2. In step 4 of the section called “Trigger On The First Monday Of The Month”, the configurations are done for every first Monday of the day. For now, it needs to be changed to every last Friday of the month. Simply, change *First* to *Last* and *Mon* to *Fri* in the dropdown list respectively, like in Figure 5.6, “Every Last Friday Of The Month”.

Figure 5.6. Every Last Friday Of The Month

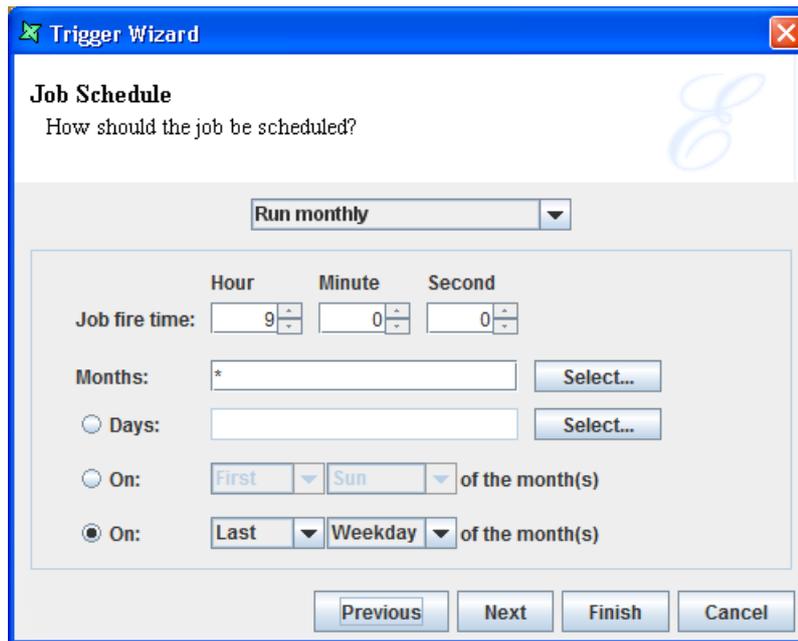
The screenshot shows the 'Trigger Wizard' dialog box with the 'Job Schedule' section. The 'Run monthly' option is selected in the dropdown menu. The 'Job fire time' is set to 8:00. The 'Months' field contains an asterisk (*). The 'Days' field is empty. The 'On:' radio button is selected, and the dropdowns are set to 'Last' and 'Fri' of the month(s). The 'Previous', 'Next', 'Finish', and 'Cancel' buttons are visible at the bottom.

3. The rest of the steps will be similar to the ones mentioned in the section called “Trigger On The First Monday Of The Month”, from step 5 onwards.
4. This trigger will now run at 8a.m. on every last Friday of the month, forever.

Trigger On The Last Working Day Of The Month

In this example, it is assumed that there are 5 working days in a week, which implies Monday to Friday.

1. Follow the instructions from step 1 to step 3 in the section called “Trigger On The First Monday Of The Month”.
2. In order to configure the trigger to run on every last working day of the month, change the settings to according to Figure 5.7, “Every Last Working Day Of The Month”.

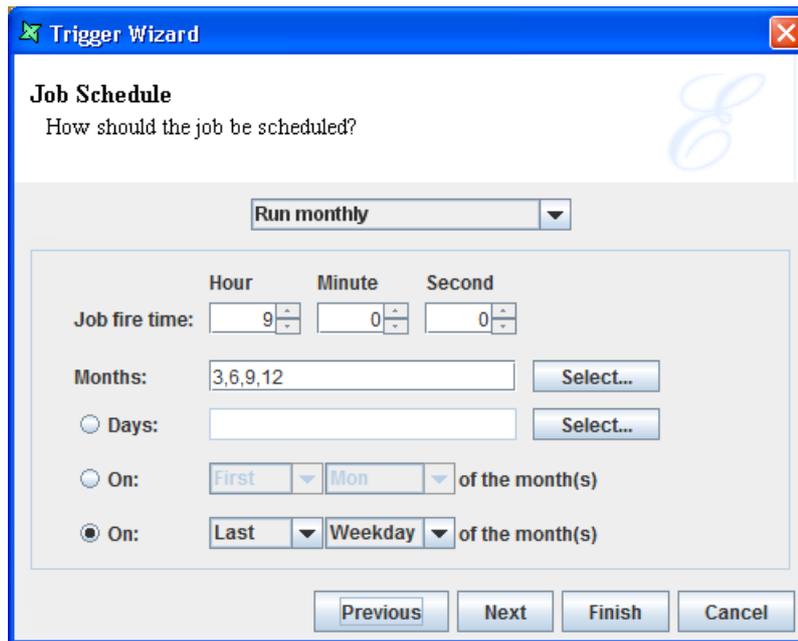
Figure 5.7. Every Last Working Day Of The Month

The screenshot shows the 'Trigger Wizard' dialog box with the 'Job Schedule' tab selected. The question 'How should the job be scheduled?' is displayed. A dropdown menu is set to 'Run monthly'. Below this, there are three spinners for 'Job fire time' set to 9:00. The 'Months' field contains an asterisk (*). The 'Days' field is empty. The 'On:' options are 'First' (selected) and 'Sun' of the month(s). The 'On:' options are 'Last' (selected) and 'Weekday' of the month(s). At the bottom, there are buttons for 'Previous', 'Next', 'Finish', and 'Cancel'.

3. Follow step 5 to 9 mentioned in the section called “Trigger On The First Monday Of The Month”.
4. When created successfully, the trigger will run on every last working day of the month forever. If user decides not to run this trigger forever, it can be configured in the Trigger wizard.

Trigger On The Last Working Day Of The Quarter

1. Follow the instructions from step 1 to step 3 in the section called “Trigger On The First Monday Of The Month”.
2. In order to configure the trigger to run on the last working day of each quarter, change the settings as shown in Figure 5.8, “Every Last Working Day Of The Quarter”.

Figure 5.8. Every Last Working Day Of The Quarter

The screenshot shows a 'Trigger Wizard' dialog box with a 'Job Schedule' section. The question 'How should the job be scheduled?' is followed by a dropdown menu set to 'Run monthly'. Below this, there are three spinners for 'Hour', 'Minute', and 'Second', with values 9, 0, and 0 respectively. The 'Months' field contains '3,6,9,12' and has a 'Select...' button. The 'Days' field is empty and also has a 'Select...' button. There are two 'On:' options: 'On: First Mon of the month(s)' (unselected) and 'On: Last Weekday of the month(s)' (selected). At the bottom are 'Previous', 'Next', 'Finish', and 'Cancel' buttons.

3. Follow step 5 to 9 mentioned in the section called “Trigger On The First Monday Of The Month”.
4. When the configurations are done correctly and saved, this trigger will run on every last working day of each quarter.