Scribble Java User Guide

Chapter 1. Eclipse

1.1. Installation

- Download Eclipse

The Scribble Java tools are installed as plugins within the Eclipse Integrated Development Environment (IDE).

Therefore the first step is to download a copy of Eclipse from *http://www.eclipse.org/downloads*. This download page includes various configurations of the IDE customised for specific needs. However if you are simply wanting to experiment with the Scribble tooling then it is best to initially just use the standard version.

- Start Eclipse

Once you have downloaded Eclipse, and unpacked it into an appropriate location, start it up by running the *eclipse* command in the top level folder.

- Install Scribble Java Tooling

Go to the *Help* → *Install New Software…* menu item. This will show a dialog window.

Enter the URL *http://download.jboss.org/scribble/tools/latest* into the *Work With* field and press the `return` key. This will show the available Eclipse features and plugins (from this update site) in the main window. Select the checkbox against the top level *Scribble* node and then press the `Next` button and following the remaining instructions.

When the plugins have been installed you will need to restart the Eclipse IDE when requested.

1.2. How To …

This chapter will describe how the Scribble tooling can be used within the Eclipse IDE.

1.2.1. Create a Scribble Protocol

Before being able to create a protocol, we need to first create an Eclipse project, if one does not already exist. This is achieved by selecting the *New* → *Project …* menu item from the context menu associated with the *Project Explorer*, which appears on the left hand side of the Eclipse tool.

This will result in a dialog box being displayed showing the different types of project that can be created. If the project is not required for any other purpose, then expand the *General* top level node and choose the *Project* child node. Press the `Next` button and then enter the project name before pressing the `Finish` button.

Once a project has been created (or selected if already exists), then we need to create a hierarchy of folders representing the module path within which the protocol will be defined.

A folder is created by selecting *New* → *Folder* menu item from the context menu associated with either the project, or a parent folder. This will show a dialog window in which the new folder's name can be specified.

Once the appropriate folder hierarchy has been created, then the next step is to create the protocol file. This can be achieved by selecting the *New* → *Other ...* menu item from the context menu associated with the containing folder. This will present a dialog window with the list of items that can be created. Expand the top level *Scribble* node and select the *Protocol* child node. When the `Next` button is pressed it will offer the ability to enter the name of the protocol. Finally press the `Finish` button to create the file.

When the protocol file is created, it will also launch the editor for the file. For further information, on how to edit the protocol, see the next section.

1.2.2. Edit a Scibble Protocol

To edit a Scribble protocol file (with extension *.scr*) simply locate the file within the *Project Explorer*, by expanding the relevant project and folders, and then double click on the file to launch the Scribble editor.

The editor will appear in the main area. It is essentially a standard text editor, although will provide some guidance (e.g. keyword highlighting).

When a change has been made to the file, an astrisk will appear against the filename to indicate that it is in a *dirty* state. To save the file, use Ctrl-S, Alt-F followed by S, or the disc icon in the top left part of the toolbar.

1.2.3. Validate a Scribble Protocol

When a protocol is edited, and saved, it will automatically trigger the validation of the protocol.

An issues that are found with the protocol will be displayed in the *Markers* view (window) in the bottom region of the Eclipse window. Double clicking on any of the errors in this area will navigate to the protocol file (and specific text in that file).

1.2.4. Project a Global Protocol

Projecting a Global Protocol will create a set of Local Protocols, one per role within the Global Protocol. To perform this task, select the *Scribble* → *Project* menu item from the context menu associated with the Scribble protocol file in the *Project Explorer*.

This will cause the local protocol files to be created in the same folder as the global protocol. This is because they are associated with the same module. However the filename will be appended with the role, indicating that it is a local protocol associated with that role.

1.2.5. Simulate a Message Trace

A message trace represents a sequence of messages that are exchanged between communicating parties. These can be defined using a JSON format, in a file with extension .trace.

Note

There is now an early preview version of the Eclipse trace editor. We would welcome feedback on its usability. Alternatively the trace files can be edited using a standard text editor.

An example of a trace file is:

```
{
 "name":"RequestResponse@Buyer-1",
 "steps":[{
  "type":"MessageTransfer",
  "message":{
   "operator":"buy",
   "types":["{http://scribble.org/example}OrderRequest"],
   "values":[""]
  },
  "fromRole":"Buyer",
  "toRoles":["Seller"]
 },{
  "type":"MessageTransfer",
  "message":{
   "operator":"buy",
   "types":["{http://scribble.org/example}OrderResponse"],
   "values":[""]
  },
  "fromRole":"Seller",
  "toRoles":["Buyer"]
 }],
 "roles":[{
  "name":"Buyer",
  "simulator":{
   "type":"MonitorRoleSimulator",
   "module":"scribble.examples.RequestResponse",
   "role":"Buyer",
   "protocol":"First"
  }
 },{
  "name":"Seller",
  "simulator":{
   "type":"MonitorRoleSimulator",
   "module":"scribble.examples.RequestResponse",
```

```
    "role":"Seller",
    "protocol":"First"
   }
  }]
 }
```

The trace file has the following top level elements:

- name

The name of the trace.

- steps

A list of steps documenting the message trace.

- roles

The definition of the roles used within the trace. The roles can optionally have a *simulator* that defines how the role, based on the steps in the trace, can be simulated against a scribble protocol definition.

Currently only one type of *step* is supported, the *MessageTransfer*. This defines the message definition, and the *from* to *to* roles.

A role definition contains a name property, and an optional simulator definition. Role simulator definitions only need to be provided for the roles that you are interested in being simulated.

The only type of role simulator currently supported is the *MonitorRoleSimulator* which uses the Scribble protocol monitor to evaluate the message trace against the specified local protocol. The *module* property is used to locate the scribble module, and the *protocol* property identifies the protocol within the module. The *role* property identifies the role name within the protocol, which may be different to the one used in the trace.

When the trace has been defined, select the *Run As* $\rightarrow$ *Simulation* context menu item associated with either the file, or a folder in which it is contains (if you wish to simulate multiple trace files at the same time). The results from the simulation will be displayed in the JUnit result view. If any simulation steps failed, they will be shown against a red cross.

Chapter 2. Command Line

2.1. Installation

Download the latest binary distribution of the command lines tools from the scribble website and unpack the zip file in an appropriate location.

The directory structure within the distribution is:

- bin

The folder containing the scripts for running the scribble tools.

- docs

The folder containing the user guide, developer guide and Java API documentation.

- lib

The jar files used by the scribble command line tools.

2.2. How To …

Before running any of the commands, you will need to setup the MODULE_PATH environment variable, to identify which folder (or folders) contain the module definitions. The MODULE_PATH will define a list of directories, separated by the *:* character.

If you don't wish to define the environment variable, then the list of folders can be supplied on the command line, by specfying: -path <directories>

2.2.1. Parse and Validate a Scribble Protocol

To parse and validate that a Scribble module is correct, run the following command from the bin folder:

On windows

```
scribble -validate <module>
```

On linux

```
./scribble.sh -validate <module>
```

If any parse or validation errors are detected, then they will be displayed on the console.

2.2.2. Project a Global Protocol

To project a Scribble global module to a local module per role, run the following command from the bin folder:

On windows

```
scribble -project <module>
```

On linux

```
./scribble.sh -project <module>
```

If any parse or validation errors are detected, then they will be displayed on the console. Otherwise a local module will be created, at the same location as the global module, for each role defined in the global module.

2.2.3. Simulate a Message Trace

To simulate a Scribble protocol, run the following command from the bin folder:

On windows

```
scribble -simulate <traceFile or folder>
```

On linux

```
./scribble.sh -simulate <traceFile or folder>
```

If a trace file is specified, then only that file will be simulated. If a folder is specified, then all trace files within that folder (or sub-folders) will be simulated. If a simulation fails, then it will immediately terminate the command.