



Eclipse & Distributed Testing Framework (DTF)

Rodney Gomes and Luis Alves

<http://wiki.github.com/rlgomes/dtf/>

Overview



- What is DTF
- What DTF isn't
- What DTF solves
- Writing a simple test
- Writing a distributed test
- What Eclipse gives DTF
- Future Eclipse Integration

What is DTF



- Framework written in Java with test cases written in an XML language similar to a few other XML languages out there with its own distinctive tags and features.

What is DTF



- Framework written in Java with test cases written in an XML language similar to a few other XML languages out there with its own distinctive tags and features.
- DTF tries to solve distributed testing in an easy to read and yet flexible manner that doesn't make it hard for the test writer to write complex tests.

What is DTF



- Framework written in Java with test cases written in an XML language similar to a few other XML languages out there with its own distinctive tags and features.
- DTF tries to solve distributed testing in an easy to read and yet flexible manner that doesn't make it hard for the test writer to write complex tests.
- A DTF setup can consist of multiple Agents (DTFA), a Controller (DTFC) and a Test Runner (DTFX). These components make up the testing components that allow you to write your tests and execute remote actions.

What is DTF



- Test framework for all of your testing needs from Functional Testing all the way through System Testing, Load, Performance and End-to-End Testing.
- A testing framework for your products lifetime being able to adapt tests and tags with ease when moving between different product releases.
- DTF tries to make reading tests developed by others easier, of course, there is always a way of making your XML test case very obscure, it just tends to be a bit harder than with most other scripting/testing languages.



What DTF isn't

- Its not a unit testing framework as unit tests should always be written in the same language that the API or product being tested is written in.
- Its not like other frameworks you've seen or like any other use of XML that you may have seen to date. From a very particular way of writing XML to the actual way that remote vs local actions are handled DTF is in a very different and new category of testing frameworks.



What DTF Solves

- Easily migrate tests between different product release with an XSL style sheet since all tests are written in XML and you can easily adapt test in one easy step.
- XML tags are documented from the DTF JavaDoc elements in the code, which allows you to keep your code changes and documentation all in on place.
- Tests are easy to document in appropriate description section of the XML data and using a style sheet you can easily keep track of your test inventory as well.



What DTF Solves

- Test case portability. Aside from the framework being written in Java the tests written in XML allow the developer of a tag to hide environment specific logic in the Java code and outside of the test case logic.



What DTF Solves

- Test case portability. Aside from the framework being written in Java the tests written in XML allow the developer of a tag to hide environment specific logic in the Java code and outside of the test case logic.
- Plugin system allows you to share functionality across different products very easily by building with different set of plug-ins that are common to multiple teams.



What DTF Solves

- Test case portability. Aside from the framework being written in Java the tests written in XML allow the developer of a tag to hide environment specific logic in the Java code and outside of the test case logic.
- Plugin system allows you to share functionality across different products very easily by building with different set of plug-ins that are common to multiple teams.
- Event system allows for any action to easily be monitored for performance measuring. The statistics calculation is already built into the framework easy to use manner.

Writing a simple test



- With Eclipse's support for writing XML makes writing your first test very quick since auto completion fills in the right XML elements.

```
helloworld.xml x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <script xmlns="http://dtf.org/v1" name="mytest">
3   <info>
4     <author>
5       <name>Rodney Gomes</name>
6       <email>rlgomes@yahoo-inc.com</email>
7     </author>
8     <description>A simple Hello World in DTF</description>
9   </info>
10  <log>Hello World!</log>
11 </script>
```

Writing a distributed test



- Now taking the previous test this is how easily you can execute the same log statement on a different host.

```
remote_helloworld.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <script xmlns="http://dtf.org/v1" name="remote_helloworld">
3   <info>
4     <author>
5       <name>Rodney Gomes</name>
6       <email>rlgomes@yahoo-inc.com</email>
7     </author>
8     <description>A simple Hello World in DTF</description>
9   </info>
10
11   <local>
12     <lockcomponent id="DTFA1"/>
13   </local>
14
15   <component id="DTFA1">
16     <log>Hello World!</log>
17   </component>
18 </script>
```

What Eclipse gives DTF



- Editing Java Code is a breeze in Eclipse.
- Editing XML is also very easy with auto completion kicking at just the right moment.



What Eclipse gives DTF

- Editing Java Code is a breeze in Eclipse.
- Editing XML is also very easy with auto completion kicking at just the right moment.
- DTF components can be easily executed within Eclipse.

What Eclipse gives DTF



- Editing Java Code is a breeze in Eclipse.
- Editing XML is also very easy with auto completion kicking at just the right moment.
- DTF components can be easily executed within Eclipse.
- Remote debugging is also very easy thanks to Eclipse and allows for diagnosing issues quick and easy.

Future Eclipse integration plans



- DTF Documentation View to allow the test case writer to easily see what each tag does and various examples displayed that are already written in the DTF JavaDocs and part of the DTF generated documentation.
- Pop-up during XML editing with additional information about each tag currently being used. Very similar to JavaDoc documentation that appears during Java editing in Eclipse.
- Test case template generation from various example scenarios for testing distributed services of different types.
- Tighter integration with DTF to allow spawning of components on the current system or other hosts just like Hudson starts up its various slaves and allows you to monitor them in your browser.

Future Eclipse integration plans



Auto-completion with DTF Documentation:

The screenshot shows the Eclipse IDE interface with a Java project named 'dtf/tests/examples/helloworld.xml'. The main editor displays an XML document with the following content:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <script xmlns="http://dtf.org/v1" name="mytest">
3   <info>
4     <author>
5       <name>Rodney Gomes</name>
6       <email>rlgomes@yahoo-inc.com</email>
7     </author>
8     <description>A simple Hello World in DTF</description>
9   </info>
10  <log>Hello World!</log>
11
12 <
```

An auto-completion popup is visible, showing a list of XML tags: abs, ad, assert, attribute, break, call, cat, choices, compare, component, and cookiegroup. The 'ad' tag is currently selected.

Overlaid on the editor is a documentation window for the 'Add' tag. It contains the following information:

- Add**: Add to values and store the result in the name of the property identified by the result attribute.
- Author**: Rodney Gomes
- Required Attributes**:

op1	first operand of the arithmetic operation
op2	second operand of the arithmetic operation
result	property name of where to store the result of the arithmetic operation

The status bar at the bottom indicates 'script/#text', 'Writable', 'Smart Insert', '12 : 6', and 'Insert mode:'. The bottom right corner of the editor shows 'Design Source' tabs.

Future Eclipse integration plans



- DTF documentation and examples view:

The screenshot shows the Eclipse IDE interface. The top toolbar includes icons for File, Edit, Source, Navigate, Search, Project, Run, Window, and Help. The Package Explorer on the left shows the project structure. The DTF View is active, displaying documentation for the `Http_get` tag. The main editor shows the XML file `helloworld.xml` with the following content:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <script xmlns="http://dtf.org/v1" name="mytest">
3   <info>
4     <author>
5       <name>Rodney Gomes</name>
6       <email>rlgomes@yahoo-inc.com</email>
7     </author>
8     <description>A simple Hello World in DTF</description>
9   </info>
10
11   <log>Hello World!</log>
12
13   <http_get uri="www.google.com"></http_get>
14 </script>
```

The documentation in the DTF View includes the following sections:

- Http_get**: The `http_get` tag is used to generate HTTP GET requests and can test webservers quite well. At the moment you can't handle cookies, but you do have the ability to add headers to the request. There are various attributes to control if we use keepalives and also to control redirects in `http`. One important attribute is the `perform` attribute which when set to false will only log the necessary information to calculate performance statistics for this run but won't log the data that was sent or received by the request. So for those cases in which you want to measure the performance you should set that value to true so that you can get more precise performance numbers.
- Author**: Rodney Gomes
- Children Tags**: [Gmp_config](#) | [headerout](#) | [cookiesgroup](#) | [cookie](#) | [header](#) (0,*)
- Events**:

http_get Event	
body	the HTTP get data that was received along with the HTTP response
headerout	each of the output headers received from the HTTP request response will generate an event with this attribute name as the prefix to the header attribute name. So if you had a header with the name X then your resulting event would be named event headerout x
bodysize	the size in bytes of the data that was received in the HTTP response
status	the status code for the HTTP request
bodyhash	the hash of the http get data that was received along with the HTTP response
headerin	each of the input headers for the HTTP request will have an event with this attribute name as the prefix to the header attribute name. So if you had a header with the name X then your resulting event would be named event headerin x
statusmsg	the status code for the HTTP request
uri	the exact uri passed to the http request
- Optional Attributes**:

perform	the perform attribute controls the amount of data that is recorded about each HTTP request
---------	--

The status bar at the bottom shows the current file is `script/http_get`, it is `Writable`, `Smart Insert` is active, and the cursor is at `13 : 14` in `Command mode`.



Contact Information

DTF - <http://wiki.github.com/rlgomes/DTF/>

Luis Alves <luis.alves@lafaspot.com>

Rodney Gomes <rodneygomes@gmail.com>