Test tasks

by Ovidiu Predescu, Jeff Turner

1. Test tasks

The object an action task generates is either an HTTP response in the case of httpRequest or soapRequest, or an HTTP request, in the case of listener. To test characteristics of such objects, test tasks are used inside an match task.

A match task contains a set of tests to be performed on an action's object, be it an HTTP response or request object. All the test tasks specified associated with a match task must succeed in order for the match task to succeed.

The test tasks are tested in the order they appear inside the `match` task. If some of the test tasks produce side-effects, like setting a global Ant property, then you should consider carefully the order in which they are executed.

1.1. Extracting values from the action result object

The value a test task checks for can be assigned to an Ant property using the `assign` attribute. Such properties can be later referred to in Anteater and normal Ant tasks.

```
<soapRequest href="${url}"
             content="test/requests/get-quote">
  <match>
    <header name="Content-Length" assign="cl"/>
    <responseCode value="200" assign="rc"/>
    <xpath select="soap:Envelope/soap:Body/n:getQuoteResponse/Result"
           assign="result"/>
    <echo>XPath-selected the value '${result}'</echo>
  </match>
</soapRequest>
```

A major difference between Anteater and Ant is that properties can be assigned values multiple times, so you can reuse the same property across the test script. Properties assigned through the `assign` attribute can also be used immediately after their definition.

1.2. header

This task is used for multiple purposes:

- to set an HTTP header when sending an HTTP or SOAP request.
- to test the value of an HTTP header in the response obtained by httpRequest or soapRequest.
- to test the value of an HTTP request header in a request received by listener.

When used to set an additional HTTP header in an HTTP request, the `header` task should be a child of the action task, either an httpRequest or a soapRequest task. In such a usage, the `header` task is not really used as a test task; we nevertheless chose to use the same name for the task to keep things simple.

If the task is used to test the value of a header in either an HTTP response or an HTTP request, it should appear as any other test task directly as a child of the match element. See the samples below for an example of how the `header` task is used.

The header element can take nested body text, which will be stripped of preceding and trailing whitespace, and used as the header value, overriding the 'value' attribute. A nested jelly element can also be specified to dynamically generate the header value.

Attribute name	Type	Default value	Description
name	String		The name of the header to be added in the HTTP request, or the name of the header to be tested.
value	String		If the task is used for setting headers in an outgoing request, this attribute contains the value of the header. If the `header` task is used to test the value of an HTTP response or request, the presence of this attribute indicates

			that an exact match is expected. If the HTTP header specified by `name` doesn't have this exact value, the `header` test task will fail.
pattern	String		Sets a regular expression with which to match the specified HTTP header in a response document. This should only be used when the element is being used as a matcher. If the pattern contains a group, `(...)`, then the matched value, if any, is placed in the `assign` property.
assign	String		Meaningful only when the header task is used for testing an HTTP header. When this attribute is specified, the property named by it will contain the value of the header, upon the successful test of the header. A successful test can happen only if the HTTP header exists and, if the `value` attribute was specified, its value equals the one specified for the HTTP header. Otherwise the property will not be assigned to.

Table 1: Attributes

Elements allowed inside `header`: none

Examples

The following HTTP request sets the value of the `Content-type` header to `text/html`:

```
<httpRequest href="${url}">
  <header name="Content-type" value="text/html"/>
  <match>
    ...
  </match>
</httpRequest>
```

Test whether the `Content-type` header value returned by an HTTP request is set to `text/html`:

```
<httpRequest href="${url}">
  <match>
    <header name="Content-type" value="text/html"/>
  </match>
</httpRequest>
```

Assign the value of the `Content-type` header to an Anteater property:

```
<httpRequest href="${url}">
  <match>
    <header name="Content-type" assign="type"/>
    <echo>Content-type of the response is ${type}</echo>
  </match>
</httpRequest>
```

Test whether the `Content-type` header value received in an incoming HTTP request is any sort of text response, and store that type in a variable:

```
<httpRequest path="/good.html">
  <match assign="type" value="text">
    <header name="Content-Type" assign="texttype" pattern="text/(.*)"/>
  </match>
  <match assign="type" value="image">
    <header name="Content-Type" pattern="image/.*"/>
  </match>
</httpRequest>
```

1.3. method

This task is used to:

- Set the HTTP method (typically GET or POST) of a HTTP request to send to a server.
- test the HTTP method of an incoming HTTP request accepted by the listener.

Value can be specified as inline text, possibly dynamically generated.

Attribute name	Type	Default value	Description
value	String		The value of the method to check for. If this attribute is present, the specified method will be compared against the

			actual method in the request: if they don't match, this test fails, otherwise it succeeds. If this attribute is not present, any HTTP method is accepted for the incoming request. This can also be specified as nested text, or generated dynamically by a nested jelly element.
assign	String		Assigns the actual value of the request's method to an Anteater property.

Table 1: Attributes

Element name	Description
jelly	Specify a Jelly script, which will dynamically generate a string selecting the HTTP method (GET, POST, HEAD etc).

Table 2: Elements allowed inside method

Examples

Generates a POST request.

```
        <httpRequest path="/text.txt">
          <method>
            POST
          </method>
          <contentEquals>Posted contents</contentEquals>
          <match>
            ...
          </match>
      </httpRequest>
```

This example tests if the incoming request is a `GET` request:

```
<listener path="/abc">
  <match>
    <method value="GET"/>
    ...
  </match>
</listener>
```

Accept an incoming request, no matter what is the method, and assign the method type to the `mth` Anteater property:

```
<listener path="/abc">
  <match>
    <method assign="mth"/>
    <echo>Received a ${mth} request</echo>
  </match>
</listener>
```

1.4. parameter

As with the header task, the `parameter` task is used for multiple purposes:

- to define an additional parameter that should be passed in an outgoing HTTP request when using the httpRequest or soapRequest tasks.
- to test the value of a parameter in an incoming HTTP request accepted by the listener task.

The parameter value may be specified as (dynamically generated) nested text.

Attribute name	Type	Default value	Description
name	String		The name of the parameter to be passed in the HTTP request when using httpRequest or soapRequest, or to be tested for in the request received by listener.
value	String		The value of the parameter to be sent in the HTTP request using httpRequest or soapRequest, or the expected value of the parameter when using the listener task.
type	String		Meaningful only when sending out parameters. Specifies how the parameter should be passed in the HTTP request, possible values being `GET` or `POST`. `GET` parameters are encoded

			in the HTTP URL, while `POST` parameters are passed in the body of the request as headers. Usually this attribute is not used, as Anteater will do the proper thing and pass the parameter according to the request type. However you may want to test a particular behavior of the HTTP or SOAP server, and pass parameters in a different way than they are expected. E.g. you may pass a `GET` parameter as a `POST` parameter and vice-versa.
assign	String		Set the named Anteater property to the value of the actual HTTP request parameter.

Table 1: Attributes

Element name	Description
jelly	Specify a Jelly script, which will dynamically generate the parameter value.

Table 2: Elements allowed inside parameter

Examples

Set a parameter in an outgoing HTTP request:

```
<httpRequest href="${url}">
  <parameter name="a" value="123"/>
  <match>
    ...
  </match>
</httpRequest>
```

Test if the `a` parameter is present in the request, and assign its value to the `paramA` Anteater property:

```
<listener path="/abc">
```

```
  <match>
    <parameter name="a" assign="paramA"/>
    ...
  </match>
</listener>
```

Return a different response based on a parameter value.

```
<listener path="/abc" description="wait for an incoming request">
      <match>
        <parameter name="a" value="123"/>
        <sendResponse href="resources/responses/good.html"
          contenttype="text/html"/>
      </match>

      <match>
        <parameter name="a" value="456"/>
        <sendResponse href="resources/responses/bad.html"
          contenttype="text/html"/>
      </match>
    </listener>
```

1.5. image

This task tests if the HTTP body contains binary image data.

Only a few representative bytes are tested, so some corrupt images may slip through. The type of image my be specified either as a MIME type or as a file extension.

Attribute name	Type	Default value	Description
mimetype	String		MIME type indicating what form we expect the data to follow. The following types are supported: • application/x-shockwave-flash • image/bmp • image/gif • image/iff • image/jpeg • image/pcx • image/png • image/psd • image/ras • image/x-portable-bitmap • image/x-portable-graymap • image/x-portable-pixmap
extension	String		Indicates the type of content we're expecting

			by the common file extension. Useful for types like 'application/x-shockwave-flash' which is much harder to remember than 'swf'. Supported extensions are: • jpg or jpeg • gif • png • bmp • pcx • iff • ras • psd • swf

Table 1: Attributes

Elements allowed inside `image`: none

Examples

Here's a simple example of validating a gif image

```
        <httpRequest href="http://jakarta.apache.org/images/jakarta-logo.gif">
          <match>
            <image mimetype="image/gif"/>
          </match>
        </httpRequest>
```

1.6. contentEquals

This task tests if one of the following exactly matchers a specified character sequence:

- the HTTP or SOAP *response* received from an httpRequest or soapRequest
- the HTTP or SOAP *request* received using listener

The value to be matched against could be specified either inline, as part of the contentEquals element, or in an external resource, like a file. Such an external resource is indicated by using the `href` attribute. You can refer to file relative to the current directory by specifying a relative URL which doesn't start with a `/` character.

You can also choose to ignore any white spaces differences between the two values to be checked. You can do this by setting the `ignoreSpaces` attribute.

Attribute name	Type	Default value	Description

href	String		URL to be used to obtain the resource to check the HTTP response against. This URL can be relative, in which case the resource is a file relative to the directory in which Anteater was started from. You should use either this attribute or you should specify the value of the text inline, in the contentEquals element.
ignoreSpaces	boolean	`false`	Specifies whether spaces should be ignored when checking for equality. Whitespaces in the two values compared are ignored if this attribute is set to `true`.

Table 1: Attributes

Element name	Description
jelly	Specify a Jelly script, which will generate the contents required from the server.

Table 2: Elements allowed inside contentEquals

Examples

The following example shows how to specify the value to be test against inline. It also ignores any whitespace differences between the two values.

```
 <httpRequest path="/text.txt"
  useTidy="false">
  <match>
   <contentEquals ignoreSpaces="true">
    Here is some freeform text saved with DOS linefeeds.
   </contentEquals>
  </match>
</httpRequest>
```

Here is how to generate a HTTP POST body dynamically with a Jelly script, and then require that the HTTP response body contain the text 'hello world':

```
               <httpRequest path="/text.txt">
                 <method>POST</method>
                 <contentEquals>
                   <jelly script="genBody.jelly"/>
                 </contentEquals>
                 <match>
                   <contentEquals ignoreSpaces="true">
                     hello world
                   </contentEquals>
                </match>
            </httpRequest>
```

Note:

This illustrates a potentially confusing aspect of Anteater: the reuse of a single "matcher" to fulfil multiple roles. Here, the contentEquals outside the match is specifying the HTTP request body, while the one inside the match specifies what the HTTP response body should contain.

1.7. regexp

This task checks whether:

- the HTTP or SOAP response received from an httpRequest or soapRequest
- or the HTTP or SOAP request received using listener

match a regular expression pattern.

The language used for specifying the regular expression is that of Perl5. Here is a brief reminder of this language:

- `\` - quote the next metacharacter
- `^` - match at the beginning of the line
- `.` - match any character except newline (but see below on how to alter this behavior)
- `|` - specifies an alternative
- `()` - specifies a group
- `[]` - a character class. Use `-` to specify ranges, or simply enumerate the characters in the set.
- `*` - match 0 or more times
- `+` - match 1 or more times
- `?` - match 1 or 0 times
- `{n}` - match exactly `n` times
- `{n,}` - match at least `n` times
- `{n,m}` - match at least `n` times, but no more than `m` times
- any other character outside the above constructs matches that character

See the Jakarta ORO package documentation for a fuller description.

Attribute name	Type	Default value	Description

pattern	String		Specifies the regular expression pattern to be used. You must specify the pattern using either this attribute or by placing it inline, inside the regexp element.
ignoreCase	boolean	`false`	Whether the match should be performed case insensitive or not.
singleLine	boolean	`true`	This attribute indicates whether `.` in a regular expression should match newlines (`\n`). The default is true, so for example `<html>.*</html>` will match even if there are multiple newlines between the `html` tags.
ignoreSpaces	boolean	`false`	If true, whitespace in the regexp is normalized and converted to '\s+', thus making irrelevant any whitespace differences in the matched text. This is useful for matching automatically generated HTML responses where whitespace (including linefeeds) doesn't matter. If the 'usetidy' property is true, ignoreSpaces is automatically set to true unless explicitly overridden. The JTidy algorithm generally reindents the text, making this a sensible default behaviour.
assign	String		Name of the attribute

			ch will contain, in the case of a successful match, the value matched by the paranthesised group specified using the `group` attribute. If the match is not successful, the property is not modified.
group	int	`0`	In case of a successful match, the value matched by this group number (specified using paranthesis `()`) is assigned to the `assign`property. If the match is not successful the property is not modified.

Table 1: Attributes

Element name	Description
jelly	Specify a Jelly script, which will dynamically generate the required regexp pattern.

Table 2: Elements allowed inside regexp

Examples

This example checks if the page returned by the server is an HTML page:

```
<httpRequest useTidy="false" href="some URL">
  <match>
    <regexp><![CDATA[<html>.*</html>]]></regexp>
  </match>
</httpRequest>
```

Given a text file of the form:

```

...
struts-dev 795
struts-user 1740
taglibs-dev 342
taglibs-user 644
tomcat-dev 934
tomcat-user 2456
turbine-dev 263
turbine-jcs-dev 17
```

```
turbine-jcs-user 21
...
```

The following example extracts a single line from the file, and assigns part of it to a variable, `${tomcat-user}`

```
<httpRequest
 href="http://jakarta.apache.org/~rubys/stats/subscribers/jakarta.apache.org">
  <match>
    <regexp assign="tomcat-user" singleLine="false" group="1">tomcat-user (.*)</regexp>
  </match>
</httpRequest>
```

1.8. responseCode

Tests the response code of an HTTP response received by httpRequest or soapRequest.

See http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html for a link of valid response codes. Common ones include `200` (OK), `301` (Moved Permanently), `404` (Not Found).

Attribute name	Type	Default value	Description
value	int		The expected response code, eg `200`
pattern	int		A regexp pattern which the response code (or part thereof) should match. Eg `3..` for all 300-class response codes.
assign	string		Name of a property to set to the returned response code. If a regexp group was specified in `pattern`, then the group value will be set instead.

Table 1: Attributes

Element name	Description
jelly	Specify a Jelly script, which will dynamically generate the required response code (equivalent to the 'value' attribute).

Table 2: Elements allowed inside responseCode

Examples

The following example shows how to test the response code of an httpRequest task. The expected response code here is 200.

```
<httpRequest href="some URL">
  <match>
    <responseCode value="200"/>
  </match>
</httpRequest>
```

Here we use a regexp pattern to test if the response code was 300-class "Redirection" response.

```
<httpRequest href="some URL">
  <match>
    <responseCode pattern="3.."/>
  </match>
</httpRequest>
```

This is useful in conjunction with a property setter:

```
<httpRequest href="some URL">
  <match assign="ok">
    <responseCode pattern="2.."/>
  </match>
  <match assign="redirected">
    <responseCode pattern="3.."/>
  </match>
  <if>
    <isset property="redirected"/>
    <then>
      <echo>We were redirected... </echo>
    </then>
  </if>
</httpRequest>
```

1.9. xpath

This task is used to test the existence of a particular element or its value. This task assumes that the body of the response or request is an XML document. Action tasks like soapRequest, and httpRequest and listener with the `useTidy` attribute set to `true`, assume the response or request body is an XML document.

Care should be taken when matching against the result returned by the httpRequest task. In most cases, you cannot use xpath to match for elements, attributes or values using XPath expressions. You can do this only when you know the HTML response you obtained is a valid XML document, like in the case of XHTML. If you do want to test for XPath expressions, then you need to run JTidy on the response; you do this using the `useTidy` attribute of the httpRequest element. JTidy can be enabled for a whole script by setting the 'default.usetidy' property to true (see the Configuration section).

Note:

Be aware that JTidy seems to have an annoying habit of stripping the bodies of header <title>, and <h1> elements

To test the existence of a particular element or element, the xpath task uses XPath as the language to address parts of the XML document.

To test the existence of an element or attribute, you should set the `select` attribute to the XPath value you're interested in. The xpath task will verify the existence of that element in the XML document in the body. If in addition to the `select` attribute, you also specify the `assign` attribute, the text value of the selected element or attribute is assigned to the property named by `assign`.

If you're interested in a particular value of an element or attribute, in addition to the `select` attribute, you should specify the `value` attribute. The actual value of the element or attribute is literally compared against this value. If the `assign` attribute is also present, and the element or attribute described by `select` exists, and its value is the same with the one specified by the `value` attribute, the property named by `assign` will contain the matched value.

Attribute name	Type	Default value	Description
select	String		The XPath string to identify an element or attribute in the body of the HTTP response or request. Mandatory.
value	String		The value expected for element or attribute identified by the `select` attribute.
pattern	String		Sets a regular expression which the string value of the 'select' XPath expression must match. If the pattern contains a group, `(...)`, then the matched value, if any, is placed in the `assign` property.
ignoreSpaces	boolean	`false`	If true, whitespace in the

			'pattern' regexp is normalized and converted to '\s+', thus making irrelevant any whitespace differences in the matched text. ignoreSpaces is automatically set to true if the 'usetidy' flag is set, unless explicitly set.
group	int	`0`	Used in conjunction with the 'pattern' attribute. In case of a successful pattern match, the value matched by this group number (specified using paranthesis `()`) is assigned to the `assign`property.
assign	String		In case of a successful match, it will contain the value of the attribute or element specified by `select`.

Table 1: Attributes

Element name	Description
jelly	Specify a Jelly script, which will dynamically generate the string value of the matched xpath node (equivalent to the 'value' attribute).

Table 2: Elements allowed inside xpath

Examples

Check for the existence of a particular element, and assign its value to a property. In this example the message is printed out only if there is an element with with the indicated path. If no suche element exists, the matcher fails and the `echo` task is not executed.

```
<soapRequest href="http://services.xmethods.net:80/soap"
  content="test/requests/get-quote.xml">
  <namespace prefix="soap" uri="http://schemas.xmlsoap.org/soap/envelope/"/>
  <namespace prefix="n" uri="urn:xmethods-delayed-quotes"/>
  <match>
```

```
    <xpath select="soap:Envelope/soap:Body/n:getQuoteResponse/Result"
      assign="result"/>
  </match>
</soapRequest>
<echo>XPath-selected the value '${result}'</echo>
```

This example is the same as above, but it will print the message only if the value matches exactly the one specified in the `value` attribute of `xpath`.

```
<soapRequest href="http://services.xmethods.net:80/soap"
                content="test/requests/get-quote.xml">
  <namespace prefix="soap" uri="http://schemas.xmlsoap.org/soap/envelope/"/>
  <namespace prefix="n" uri="urn:xmethods-delayed-quotes"/>
  <match>
    <xpath select="soap:Envelope/soap:Body/n:getQuoteResponse/Result"
      value="20"
      assign="result"/>
  </match>
</soapRequest>
<echo>XPath-selected the value '${result}'</echo>
```

1.10. relaxng

This task tests XML for conformance to a specified Relax NG schema. It is based on James Clark's Jing task for Ant.

Attribute name	Type	Default value	Description
rngFile	String		Basedir-relative path to a file containing a Relax NG schema
compactSyntax	boolean	`false`	Whether or not the specified file is in Relax NG's compact syntax.
checkId	boolean	`true`	Whether to check for ID/IDREF/IDREFS compatibility

Table 1: Attributes

Elements allowed inside `relaxng`: none

Examples

Checks that an XSLT file is valid

FIXME (JT):

This is a dumb example; rather, use the XHTML stylesheet to validate proper HTML

```
<httpRequest group="std" path="/identity.xsl">
  <match>
    <!--
    <xpath select="xsl:stylesheet/xsl:output"/>
    -->
    <relaxng rngFile="test/xslt.rng"/>
  </match>
</httpRequest>
```

1.11. sendResponse

Sends a response to an HTTP request received by the listener task.

With this task you can specify:

- the body content of the response to be sent back, by a nested 'contentEquals' element, 'href' attribute, nested text or nested jelly task.
- the response code to be used, by a nested 'responseCode' element or 'responseCode' attribute.
- any HTTP headers to send with the response, through nested 'header' elements.
- the MIME type of the response via the 'contentType' attribute.

Attribute name	Type	Default value	Description
href	String		Indicates the URL of a resource that contains the response to be sent back. A relative URL is interpreted as relative to the directory Anteater was started from. You can use any URL supported in Java, including HTTP and FTP, to read an external resource of file. Mandatory.
contentType	String	`text/html`	Specifies the MIME type to be associated with the response.
responseCode	int	`200`	The response code to be sent be as part of the HTTP response.

Table 1: Attributes

Element name	Description

responseCode	Specify the HTTP response code.
header	Adds a header to the HTTP response.
contentEquals	Specifies the HTTP body.
jelly	Specify a Jelly script, which will dynamically generate the HTTP response body.

Table 2: Elements allowed inside sendResponse

Examples

The following listener responds to POSTed message by returning a HTTP 202 response, with header 'X-date' set to whatever ${date} is, and a text body with normalized spaces.

```
              <listener path="/text.txt">
                <match>
                  <method>POST</method>

                  <sendResponse>
                    <responseCode>202</responseCode>
                    <header name="X-date" value="${date}"/>
                    <contentEquals ignoreSpaces="true">
                      Here   is some   freeform text.
                    </contentEquals>
                  </sendResponse>
                </match>
              </listener>
```

This example receives a request on a URL and sends back an HTTP response with a response code of `201`:

```
<listener path="/good.html"
          description="Process a simple request">
  <match>
     <method value="GET"/>
       <sendResponse href="test/responses/good.html"
                     contentType="text/html"
                     responseCode="301"/>
   </match>
</listener>
```