Test

Test Component

Testing of distributed and asynchronous processing is notoriously difficult. The Mock, Test and DataSet endpoints work great with the Camel Testing Framework to simplify your unit and integration testing using Enterprise Integration Patterns and Camel's large range of Components together with the powerful Bean Integration.

The `test` component extends the Mock component to support pulling messages from another endpoint on startup to set the expected message bodies on the underlying Mock endpoint. That is, you use the test endpoint in a route and messages arriving on it will be implicitly compared to some expected messages extracted from some other location.

So you can use, for example, an expected set of message bodies as files. This will then set up a properly configured Mock endpoint, which is only valid if the received messages match the number of expected messages and their message payloads are equal.

Maven users will need to add the following dependency to their `pom.xml` for this component when using **Camel 2.8** or older:

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-spring</artifactId>
    <version>x.x.x</version>
    <!-- use the same version as your Camel core version -->
</dependency>
```

From **Camel 2.9**: the Test component is provided directly in `camel-core`.

URI format

```
test:expectedMessagesEndpointUri
```

Where `expectedMessagesEndpointUri` refers to some other Component URI that the expected message bodies are pulled from before starting the test.

URI Options

Name	Default Value	Description
`anyOrder`	`false`	**Camel 2.17:** Whether the expected messages should arrive in the same order, or in any order.
`delimiter`	`\n|\r`	**Camel 2.17:** The delimiter to use when `split=true`. The delimiter can be a regular expression.
`split`	`false`	**Camel 2.17:** If `true` messages loaded from the test endpoint will be split using the defined `delimiter`.For example to use a `file` endpoint to load a file where each line is an expected message.
`timeout`	`2000`	**Camel 2.12:** The timeout to use when polling for message bodies from the URI.

Example

For example, you could write a test case as follows:

```
from("seda:someEndpoint")
  .to("test:file://data/expectedOutput?noop=true");
```

If your test then invokes the MockEndpoint.assertIsSatisfied(camelContext) method, your test case will perform the necessary assertions.

To see how you can set other expectations on the test endpoint, see the Mock component.

See Also

- Configuring Camel
- Component
- Endpoint
- Getting Started

- Spring Testing