**Mocking in web services testing**

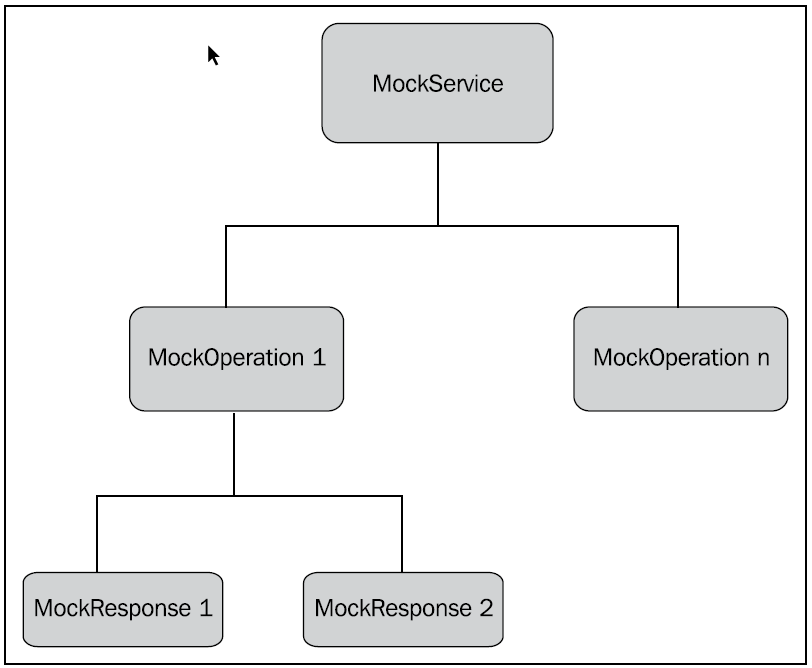
Mock services come in handy for web services testing in many ways. In simple terms, when you do not have access to the real web service, you have no option other than to simulate that service.

**Mock services with soapUI**

soapUI allows you to create a simulation of the web service from its WSDL. This simulation is known in a soapUI project as a **MockService**. A consumer application can connect to the MockService as if it is the real web service.

**The structure of soapUI MockService**

We can identify three basic elements in the soapUI MockService model—MockService, MockOperations, and MockResponses.



* A MockService can include any number of MockOperations that in turn contain multiple MockResponses as shown in the preceding diagram.
* MockOperations represent operations of the WSDL that is imported in the soapUI project.
* MockResponses are the response messages that correspond to those operations.

**MockService details**

* The MockService **Properties** tab at the left-hand side pane can be used to edit the default properties of the MockService.
* In the right pane of the above screen, soapUI allows to do some pre- and post processing for the MockService.
* **Start Script** can be used to call a Groovy script at the time of the mock service start. Usually, if we want to initialize some global resources such as database connections, we can call **Start Script**. **Stop Script** is called when the mock service is stopped.
* When the MockService receives a request, **OnRequest Script** can be called.
* Similarly, **AfterRequest Script** is called after a request is processed by the MockService.

**MockOperation details**

**Dispatch** style property as it is very important in determining the mock response.

**Dispatching styles of MockOperation**

* A MockOperation can include multiple MockResponses. Therefore, when we send a request to the MockService and subsequently dispatch to the MockOperation, there should be a way to determine to which MockResponse the request must be forwarded to.
* The dispatch style of the MockOperation is used to select the preferred dispatching mechanism for mock responses.

**SEQUENCE**: This is the default dispatching style. With this style, when sending requests to the MockOperation, MockResponses are selected iteratively one after the other as they appear in the MockResponses list.

**SCRIPT**: This dispatching style gives the ability to control responses based on Groovy scripting. The MockResponse is determined by the execution of the specified script.

**RANDOM**: This style chooses the MockResponses randomly without any order.

**QUERY\_MATCH**: This style returns MockResponse by evaluating multiple XPath expressions.

If the value returned by the XPath expression matches the expected value, the request is forwarded to the response which is defined under the **ispatch to** drop-down list. Using this style, you can use different XPath queries to evaluate different types of requests and dispatch the requests to MockResponses appropriately

**XPATH**: This dispatching method is used to dispatch the request to a response based on the XPATH expression result of the request.