**Mocking services**

* Virtualization or simulation of web services is called **mocking**, that means even though the web service is not implemented completely we will use that once we invoke the WSDL file for testing. So, when the actual web service is ready we will take that.

**Example**: x(request) sends -> y(response) and based on y’s response x will interact with Z (end to end communication). So, here y is not implemented properly, but x and z are 100% implemented. So, without y they x and z cannot communicate each other. So in this case we will mock y (creating dummy webservice) and will take the data from it and conduct end to end testing and once it is implemented completely we will use it again.

* **Mocking** can be useful because of the following things:

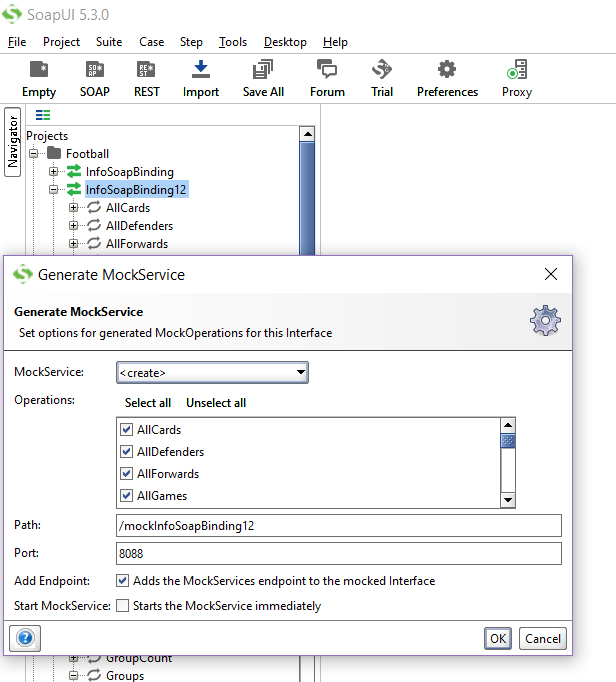
1. Especially in **Regression testing –** it is not always practical to call webservice every time because it may cause performance issues.
2. Web services are the integration of **Heterogeneous (different)** applications of SOA. So, one web service may connect to different web service which is outside of your organization.
3. Services are blocked by **firewalls**

**Regression testing** is a type of software **testing** which verifies that software which was previously developed and tested still performs the same way after it was changed or interfaced with other software. Changes may include software enhancements, patches, configuration changes, etc.

**Implementation**

1. **Create mock service:**

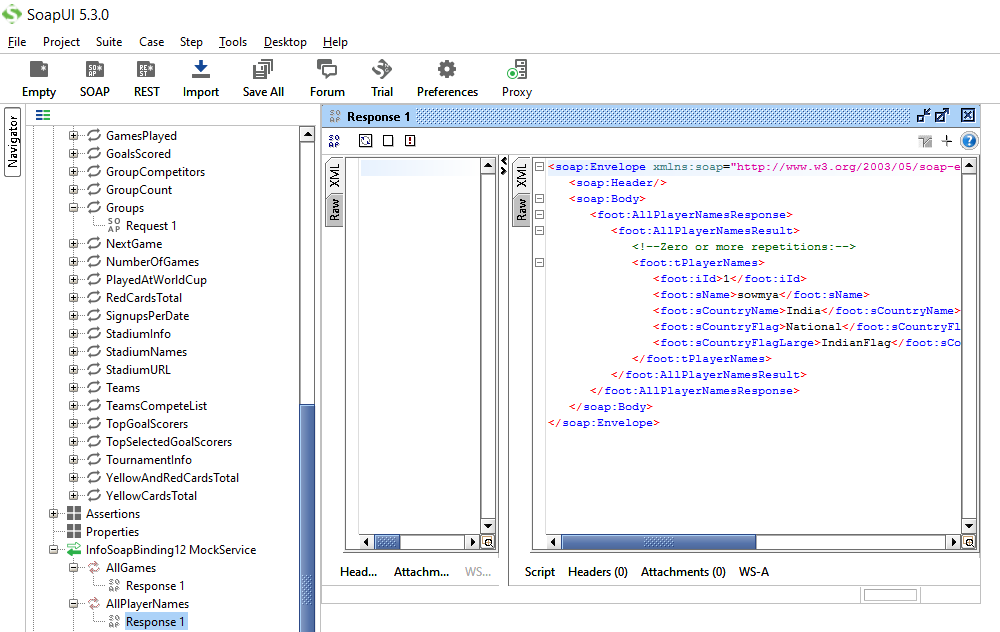
First invoke the wsdl file by taking from local desktop -> right click on any binding -> select generate mock service -> here we can select our operations to mock. it will access the local path (mock followed by interface – CurrencyConverterSoap12 and port number

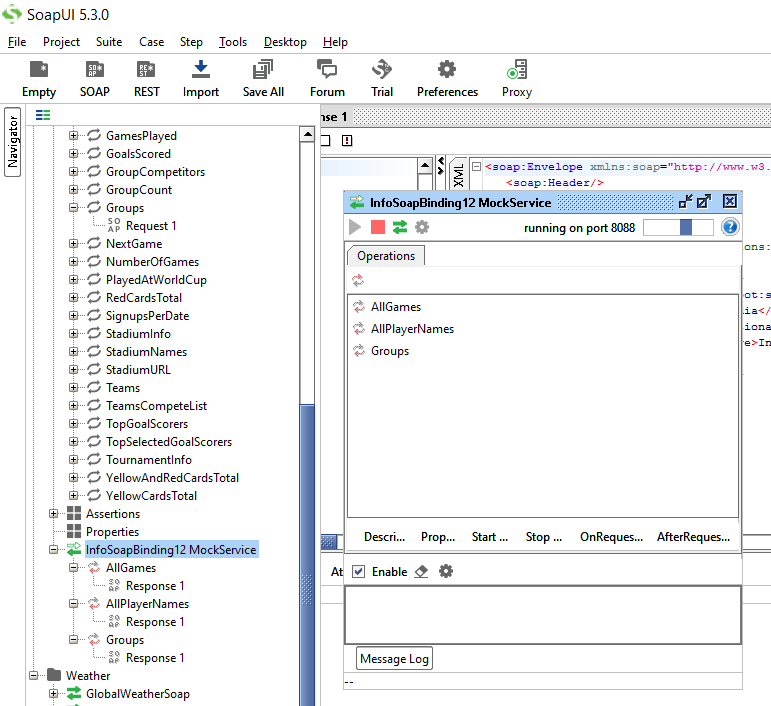


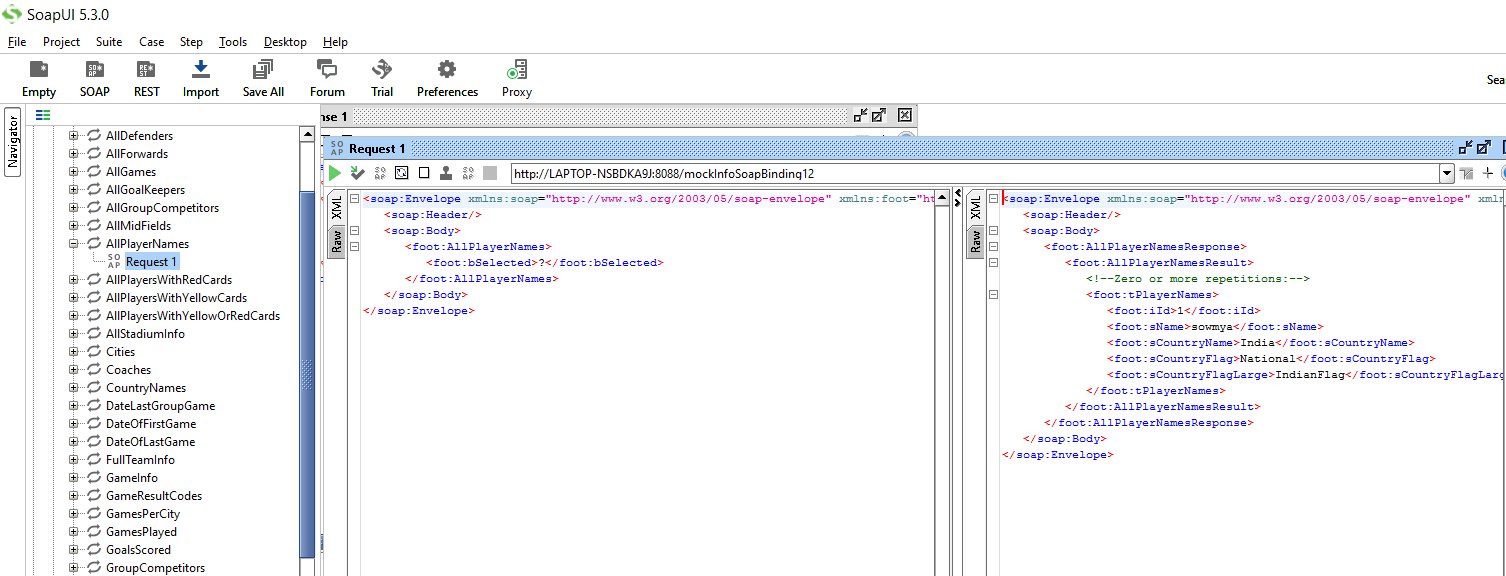
2. If you want to select mock operation afterwards then right click on mock service -> click on mock operation

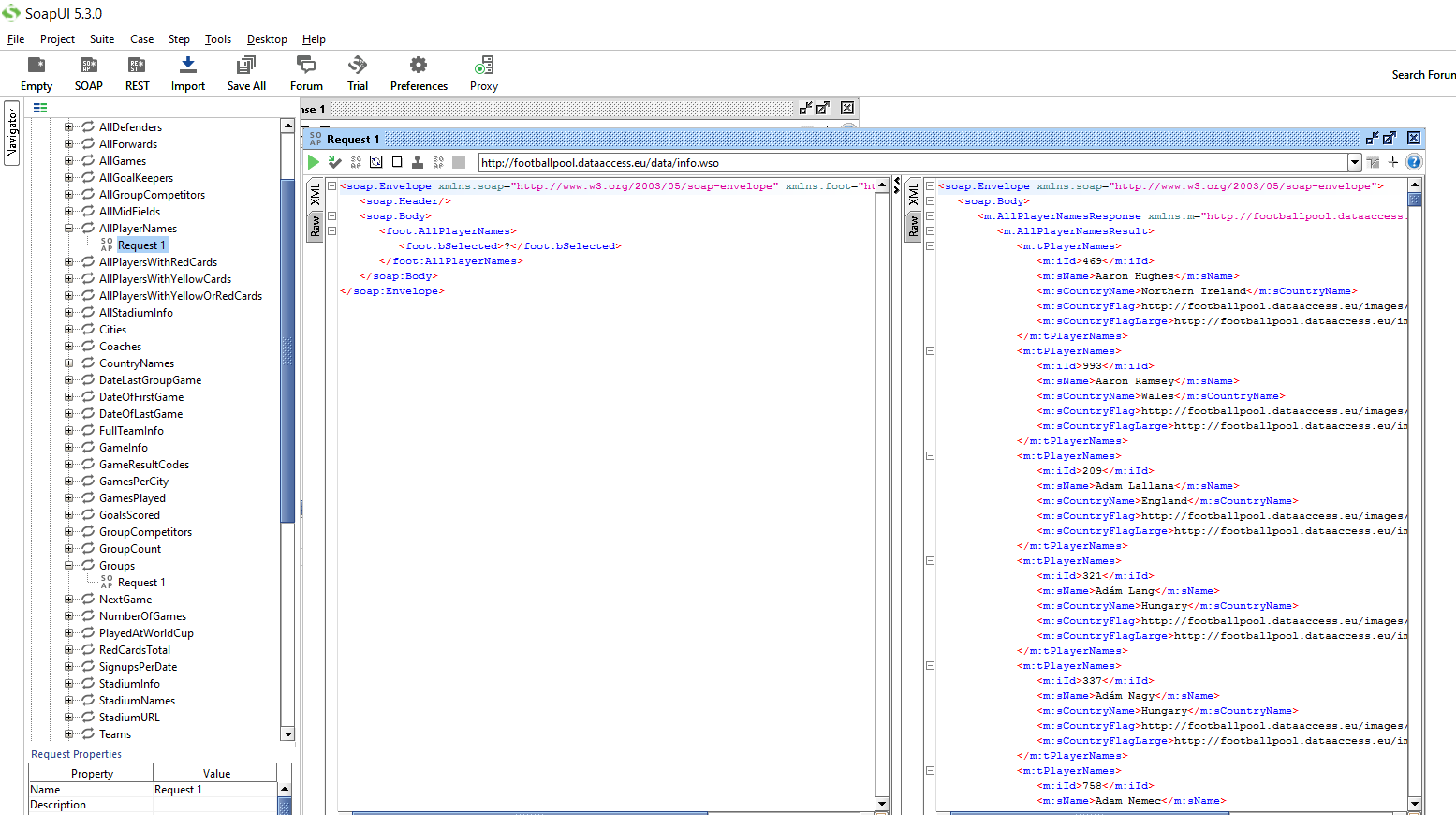
**Static scenario:**

* Will get only static response and once if we assign values and start running the mock service. Then no matter what are the parameters which we give in the request, we will get the static response for the mock operation for that particular request in the interface. It depends on the end point like
* 1) if we give local path will get static response
* 2) if we give online path then will get actual response from webservice.





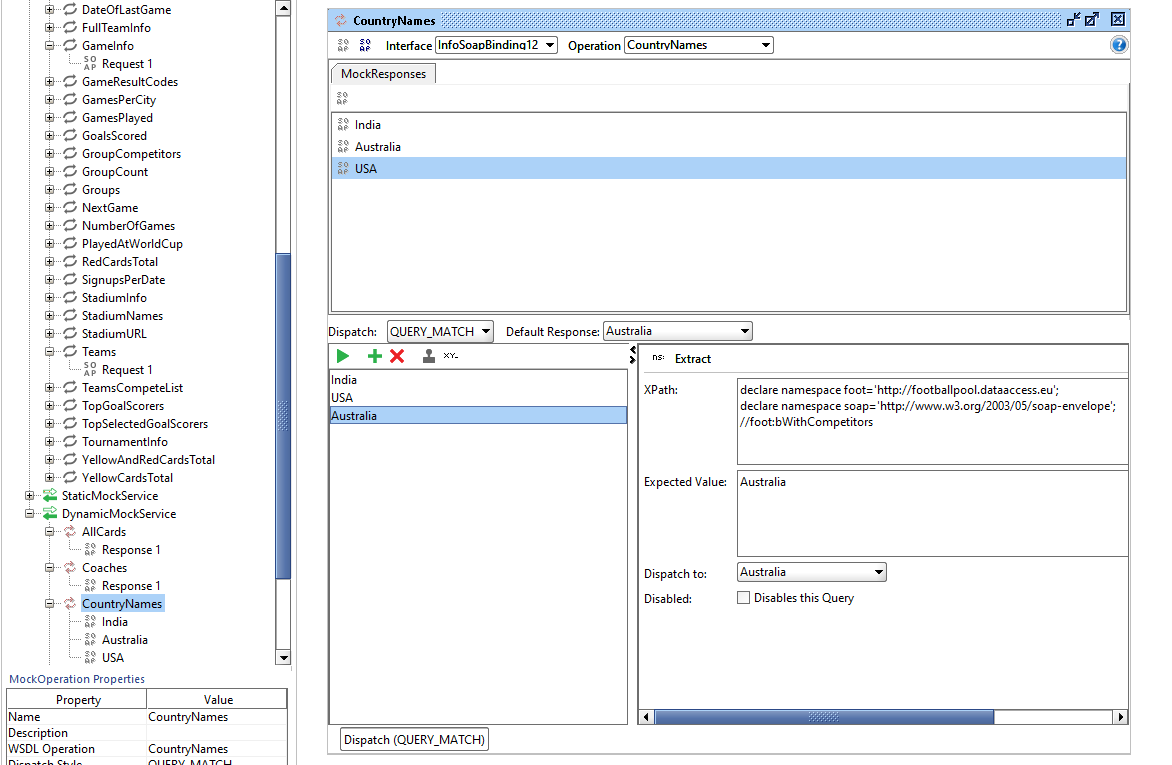




**Dynamic scenario:**

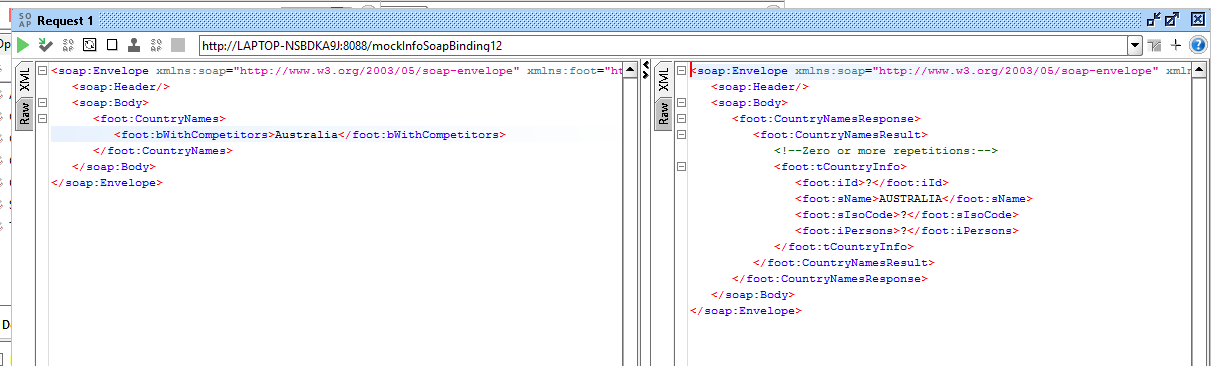
Now, even though it is local but we want our response based on our request. So, in that case we will do with the help of **DISPATCH** mechanism.

1. **Query\_Match:**



Here, we have to create diff responses and based on the request which we give in the interface (bindings) we will get that output. So, here as soon as you enter the value in the request the xpath will be triggered and based on that the response will be dispatched and if there is no correct request then default response will be triggered.

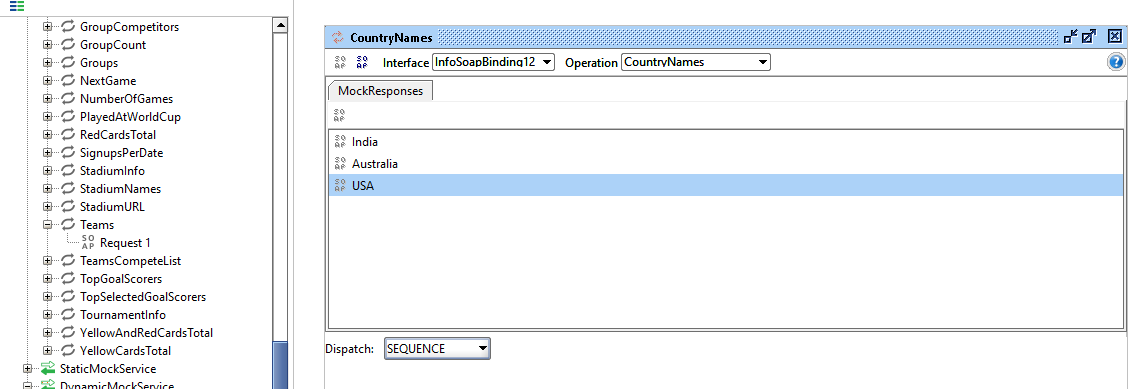
**Note:** u should stop the service of static and run dynamic



So for the above we have given Australia as our request and we got output as given.

1. **SEQUENCE:**

In this, it will invoke the response in the sequence like India, Australia and USA irrespective of any request



1. **RANDOM:**

In this, it will invoke random response like India, USA, Australia (or) any other order.

This is helpful in generating **SESSION-ID** randomly

1. **SCRIPT: / XPATH**

We can write groovy script and get the response based on our request condition.

1. First import the package for XmlHolder

**XmlHolder:** this package will load the xml which we send as an argument and seperates all the nodes. And will store this in one xml variable. So that we can access the desired node that we want.

1. Define the xml by passing the request as an argument into the XmlHolder
2. Now will get the whole request in “xml”, take the path //foot:bWithCompetitors
3. Define the response

