**Exercise 3**

*Using SOAPUI to interact with a sample SOAP Service*

**Prior Knowledge**

XML

**Objectives**

Deploy a ready built service into Tomcat

Call the service using SOAP message

See sample SOAP messages

**Software Requirements**

These are already installed on the VM.

* OpenJDK Java Development Kit 8
* Tomcat 8.035 or later
* SOAPUI 5.0.0 or later

**Step 1. Deploy the service into Tomcat**

Firstly, you can make sure all previous servers (node, mitmdump, etc) are closed down as we don’t need them for this exercise.

The sample service is already coded and available as a WAR file. It was written with Apache CXF and is available in the Downloads directory

~/Downloads/sample-service-1.0.war

Simply copy the WAR file into <tomcat>/webapps/ directory

From the terminal window:

cp ~/Downloads/sample-service-1.0.war ~/servers/tomcat/webapps

Now start Tomcat as follows

*Hint: Since tomcat also runs on port 8080 make sure your other servers that use that port are shut down!*

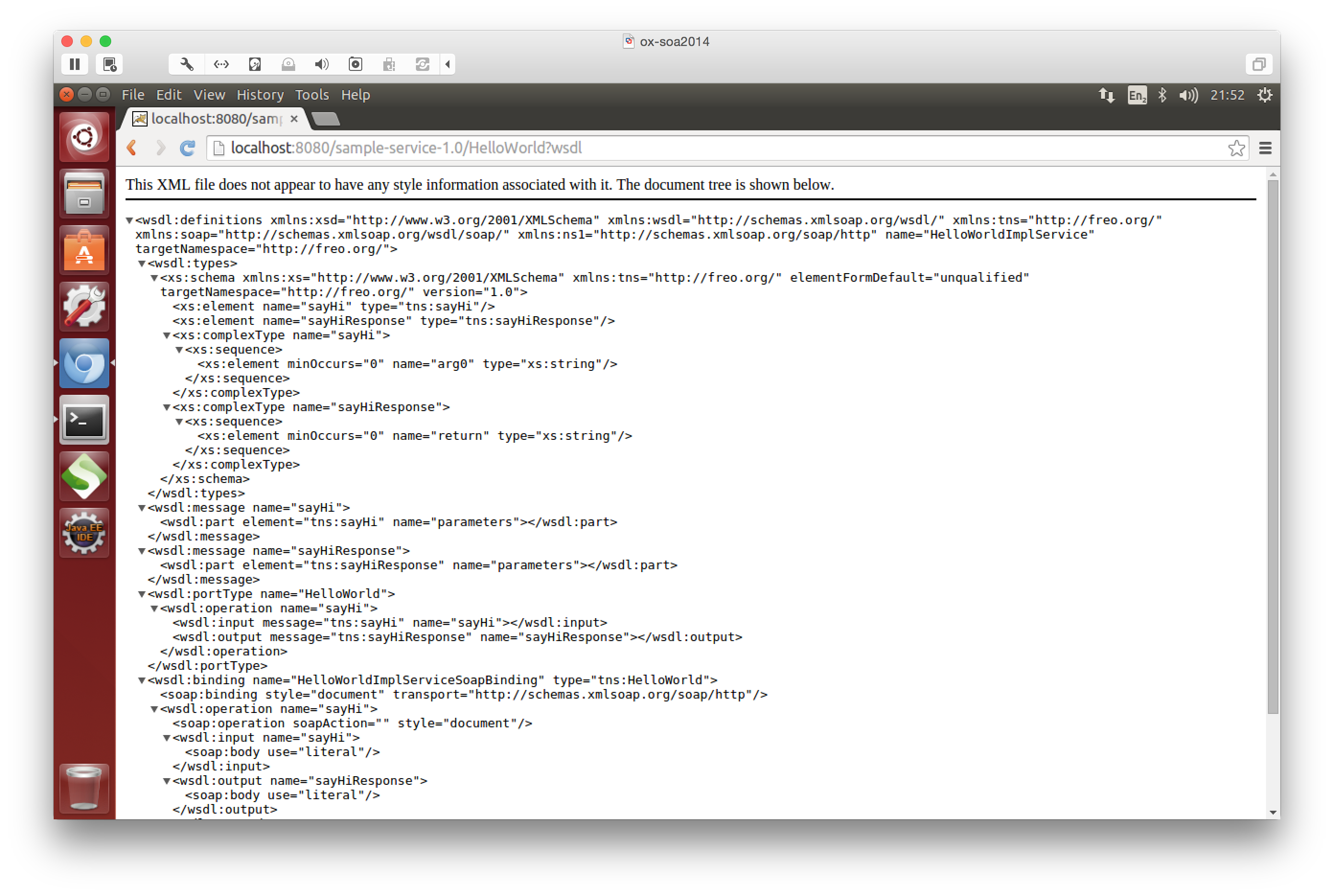
Rather than use the “daemon” server startup we will use the model that keeps the server running in the open window.

cd ~/servers/tomcat

bin/catalina.sh run

Check to see if its running:

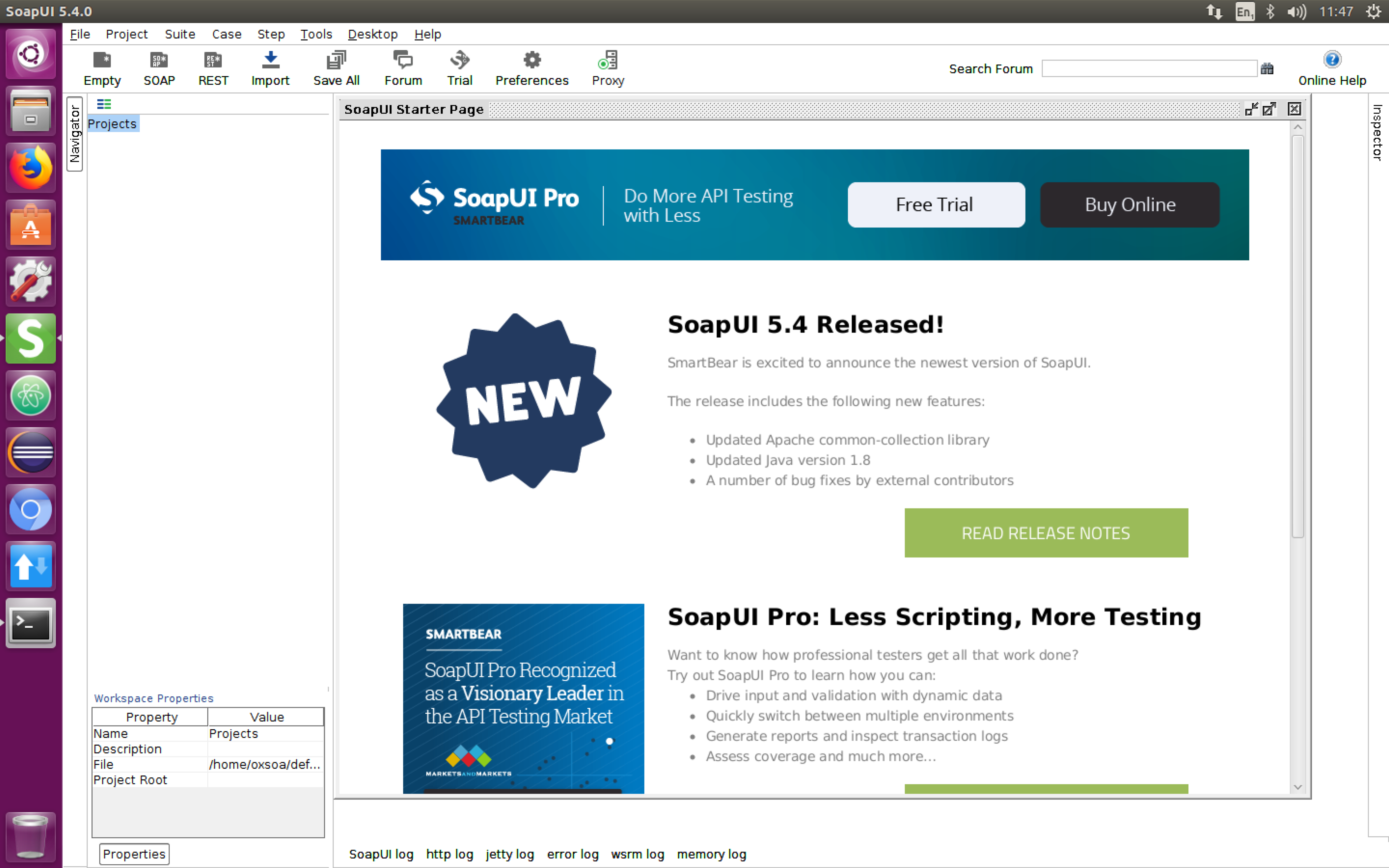
Browse <http://localhost:8080/sample-service-1.0/>

You should see a SOAP Web Service listed with a link to the WSDL. Click on this link.  
  
**Copy the WSDL Link into the Clipboard.**

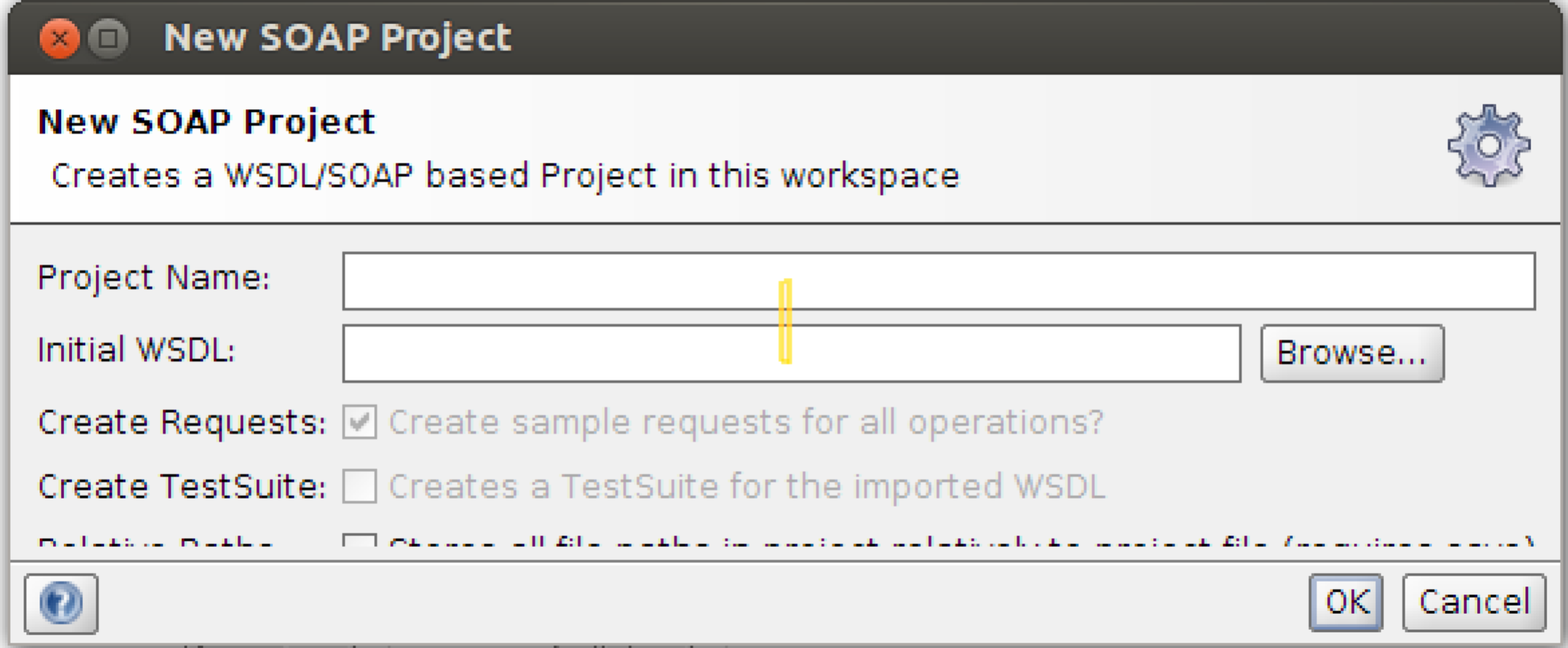
Now start up SOAPUI from the launcher:



You should see a screen like this:



File->New SOAP Project



Type in a name for the project (e.g. Sample)

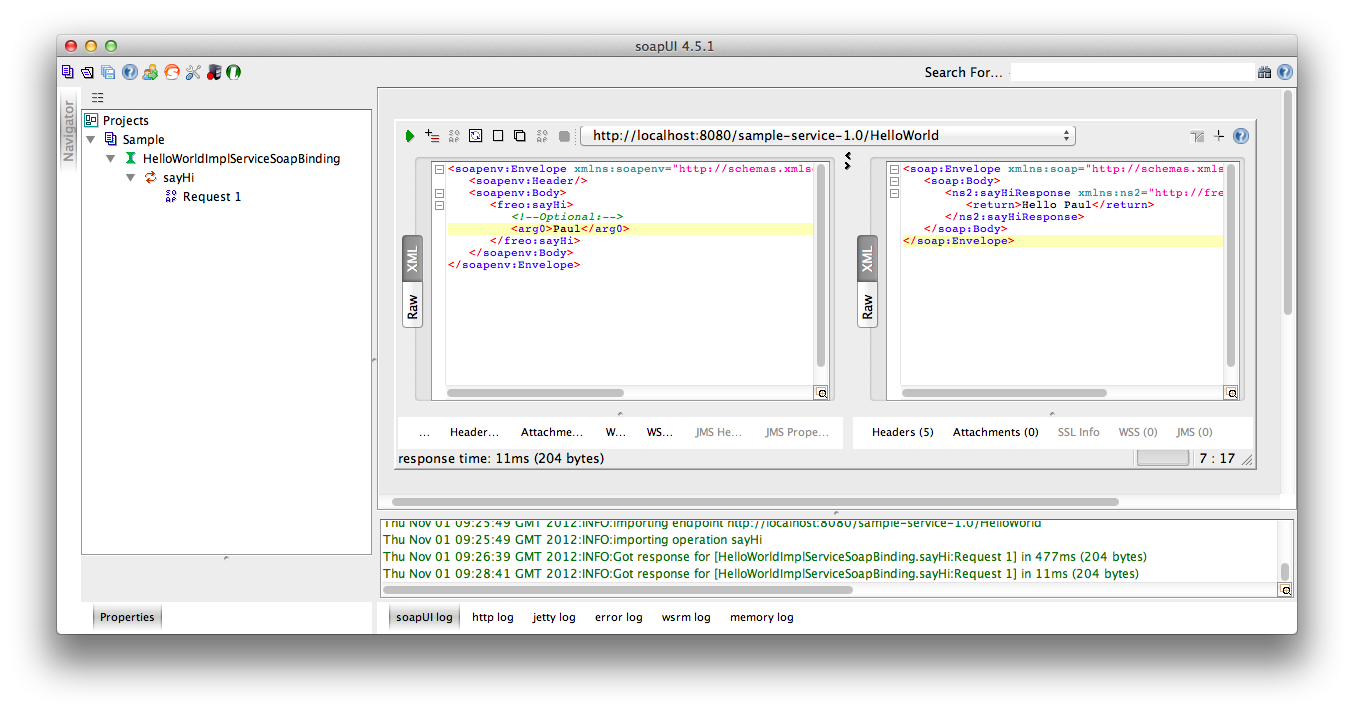
Paste the WSDL URI into the **Initial WSDL field**

Hit **OK**

Now open up the Request editor for one of the operations. You can do this by navigating the service tree in the left window until you see a Request object and click on that.

In the XML Payload, change any ‘?’ fields into something useful. Now hit the little green arrow (Run) button.

You should see a response from the service.



Use some of the SOAP UI capabilities to inspect the HTTP Headers, etc.

Try some other options and see what happens.

Congratulations! That’s all.