SoapUI
Supported by SMARTBEAR

Deploying Mock Services as WAR Files

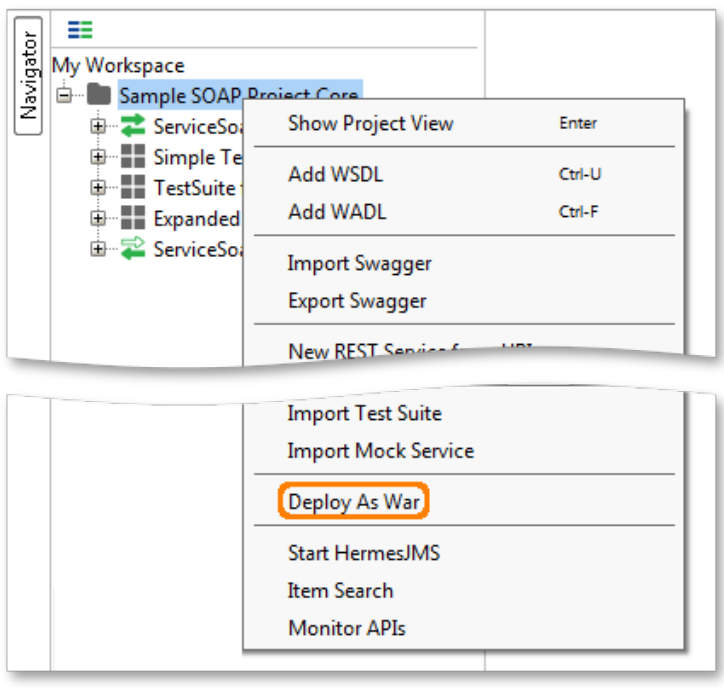
Once your MockService is all set up and configured, you have several options for how to run it:

- Manually from inside soapUI
- With the command-line MockService runner (as described at ...)
- As a standard war file deployed in a servlet container

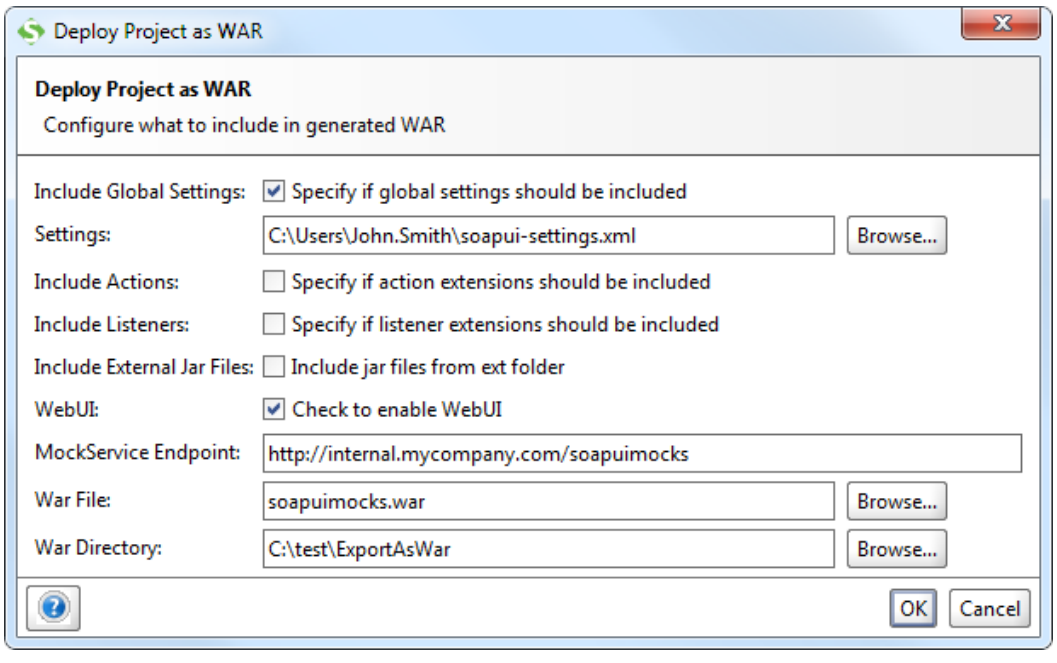
Let's have a look at the last option in more detail.

The *Deploy As War* functionality allows you to package your soapUI project and deploy it to a standard servlet container (Tomcat, etc). All MockServices in the project will be exposed by the war and optionally a simple web interface can be enabled to view request and script logs.

Get started by right-clicking on your project and select the *Deploy as War* option at the bottom of the menu:



This brings up the following dialog:

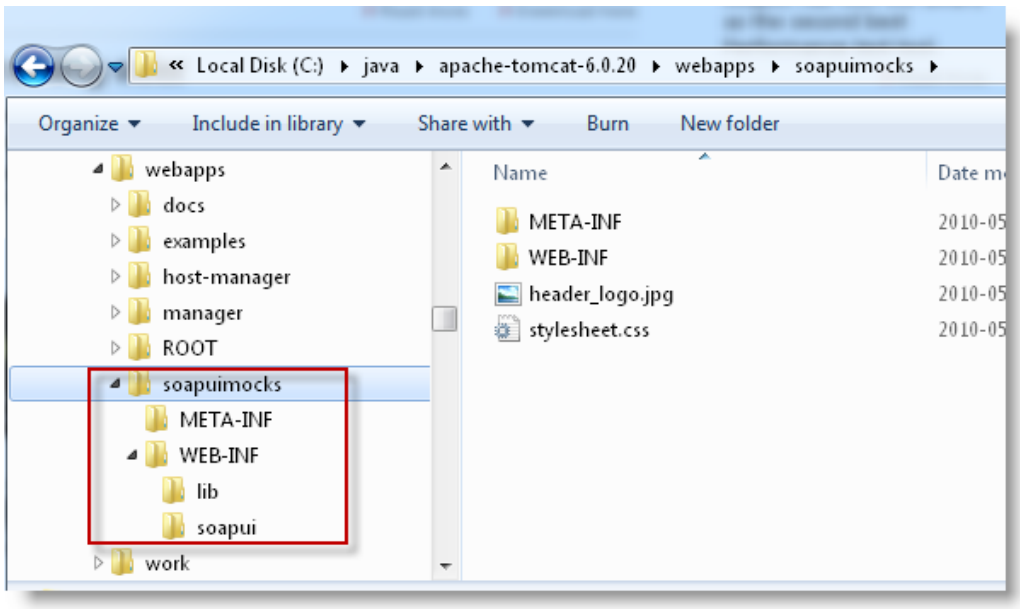


The settings here control the content and functionality of the generated war;

- Include global settings if you have made any configurations in the global soapUI preferences that apply to the functionality of the MockService
- Include XXX options allow you to bundle extensions and jars in the generated war
- The WebUI (see below) is disabled by default, enable it if you want the WebUI

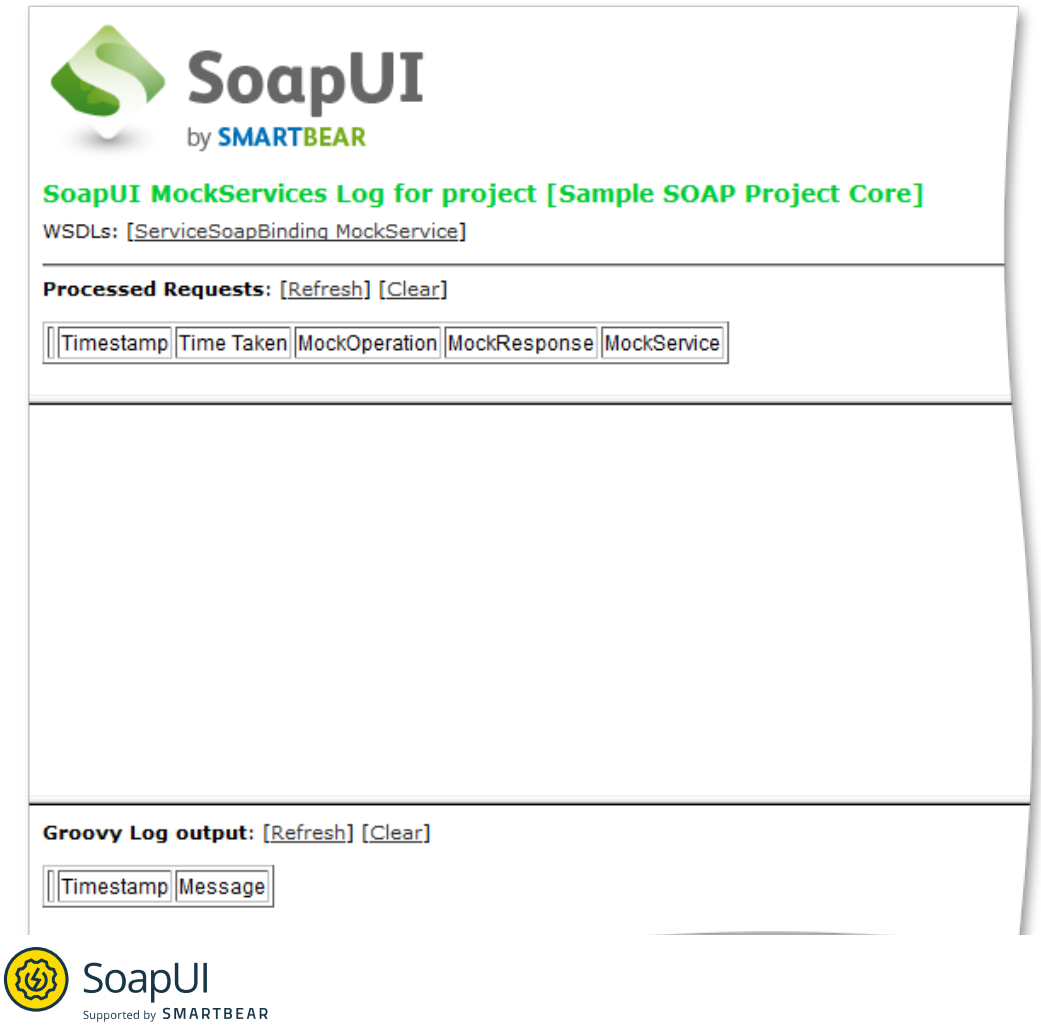
If you do not specify a war file soapUI will just generate the web application directory for you, use this if you want to generate it directly into the webapps folder of your application server.

After running the example above you will get the specified war file and directory, in our example we'll copy the war to the webapps folder of a local tomcat instance, Tomcat detects the file and unzips and deploys it, the resulting folders can be seen below:



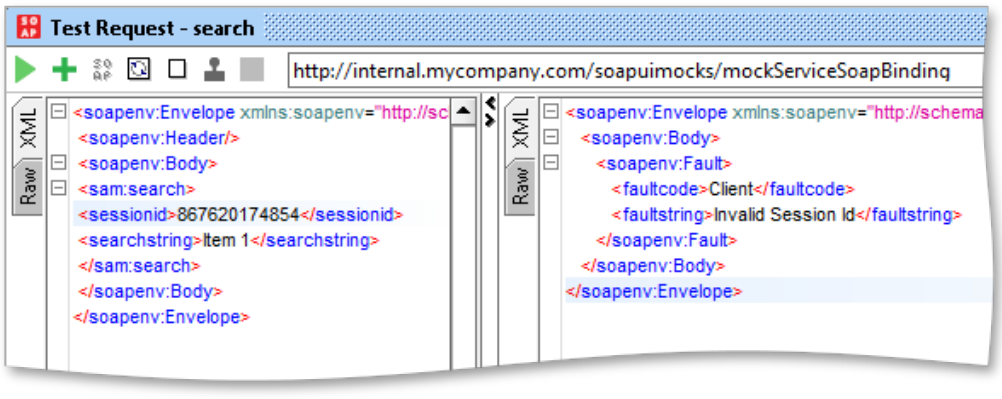
(This is the same folder structure as the War Directory created by soapUI initially)

Now if we open a web browser and point it to the root of the deployed war we get the following interface:



The WSDLs exposed are marked in the screenshot, click any of these to get the corresponding WSDL presented to you in your browser.

Now if we use soapUI to fire off a request to the MockService:



We can see the response in the panel to the left, and in the web interface of the MockService we can see the dispatched request:

SoapUI by SMARTBEAR

SoapUI MockServices Log for project [Sample SOAP Project Core]

WSDLs: [ServiceSoapBinding MockService]

Processed Requests: [Refresh] [Clear]

	Timestamp	Time Taken	MockOperation	MockResponse	MockService
1	Fri Jun 26 12:25:52 MSK 2015	836	login	Ok Response	ServiceSoapBinding MockService
2	Fri Jun 26 12:26:16 MSK 2015	45	search	Item 1 Response	ServiceSoapBinding MockService
3	Fri Jun 26 12:28:26 MSK 2015	3	search	Unknown Response	ServiceSoapBinding MockService
4	Fri Jun 26 12:28:46 MSK 2015	2	search	Invalid Session Id Fault	ServiceSoapBinding MockService

Details for MockResult at Fri Jun 26 12:28:46 MSK 2015 (2ms)

Request Headers:

Header	Value
SOAPAction	"http://www.soapui.org/sample/search"
host	localhost:8181
connection	Keep-Alive
Content-Length	281
accept-encoding	gzip,deflate
Content-Type	text/xml;charset=UTF-8
user-agent	Apache-HttpClient/4.1.1 (java 1.5)

Incoming Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sam="http://www.soapui.org/sample/">
 <soapenv:Header/>
 <soapenv:Body>
 <sam:search>
 <sessionid>?</sessionid>
 <searchstring>Item1</searchstring>
 </sam:search>
 </soapenv:Body>
</soapenv:Envelope>
```

Response Headers:

Header	Value
Content-Length	242

Returned Response:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Body>
 <soapenv:Fault>
 <faultcode>Client</faultcode>
 <faultstring>Invalid Session Id</faultstring>
 </soapenv:Fault>
 </soapenv:Body>
</soapenv:Envelope>
```

Groovy Log output: [Refresh] [Clear]

Timestamp	Message

(Obviously, this web-interface won't win any prestigious awards)