The Open University

T320 E-business technologies: foundations and practice

Block 3 Part 6 Activity 3: Specifying WSDL style and encoding using Eclipse

Prepared for the course team by Neil Simpkins

Introduction

Earlier in this block you developed, deployed and tested the 'Hello' web service. In this practical activity you will quickly see that the same service can be created with different WSDL style and encoding.

You saw in Part 2 Activity 5 that the options for style are 'RPC' or 'document' and the options for encoding are 'encoded' or 'literal'. There is also one other combination called 'document/literal wrapped', giving five possibilities in total:

1 RPC/encoded
2 RPC/literal
3 document/encoded
4 document/literal
5 document/literal wrapped

When you generated the 'Hello' web service earlier, you skipped the step that allows the style and encoding to be selected. Here you are simply going to repeat the generation of the 'Hello' web service but this time including the choice of a specific WSDL style and encoding.

Initial steps towards a web service

Return to the practical activity in which you created the 'Hello' web service, *Implementing a Simple Web Service*. You can take either of the following routes:

- Repeat all the steps in *Implementing a Simple Web Service* up to the stage where you are presented with the dialogue box shown in Figure 18 (reproduced below as Figure 1), where you have made the same selections as before (shown in the figure). Use another project if you don't want to lose the previous version of the service (in the instructions that follow, I have called my project 'HelloNewWSDL').
- Alternatively, if you have the old 'Hello' project in your workspace, simply start from the section 'Generating a web service and client' in *Implementing a Simple Web Service* and repeat the steps from that position. In this instance some files from the old project will have to be overwritten; Eclipse will ask you for permission to do this at the appropriate stage and you should allow the overwriting in all cases. Again, you should continue through the activity until you reach the position shown in Figure 18 (Figure 1 here).

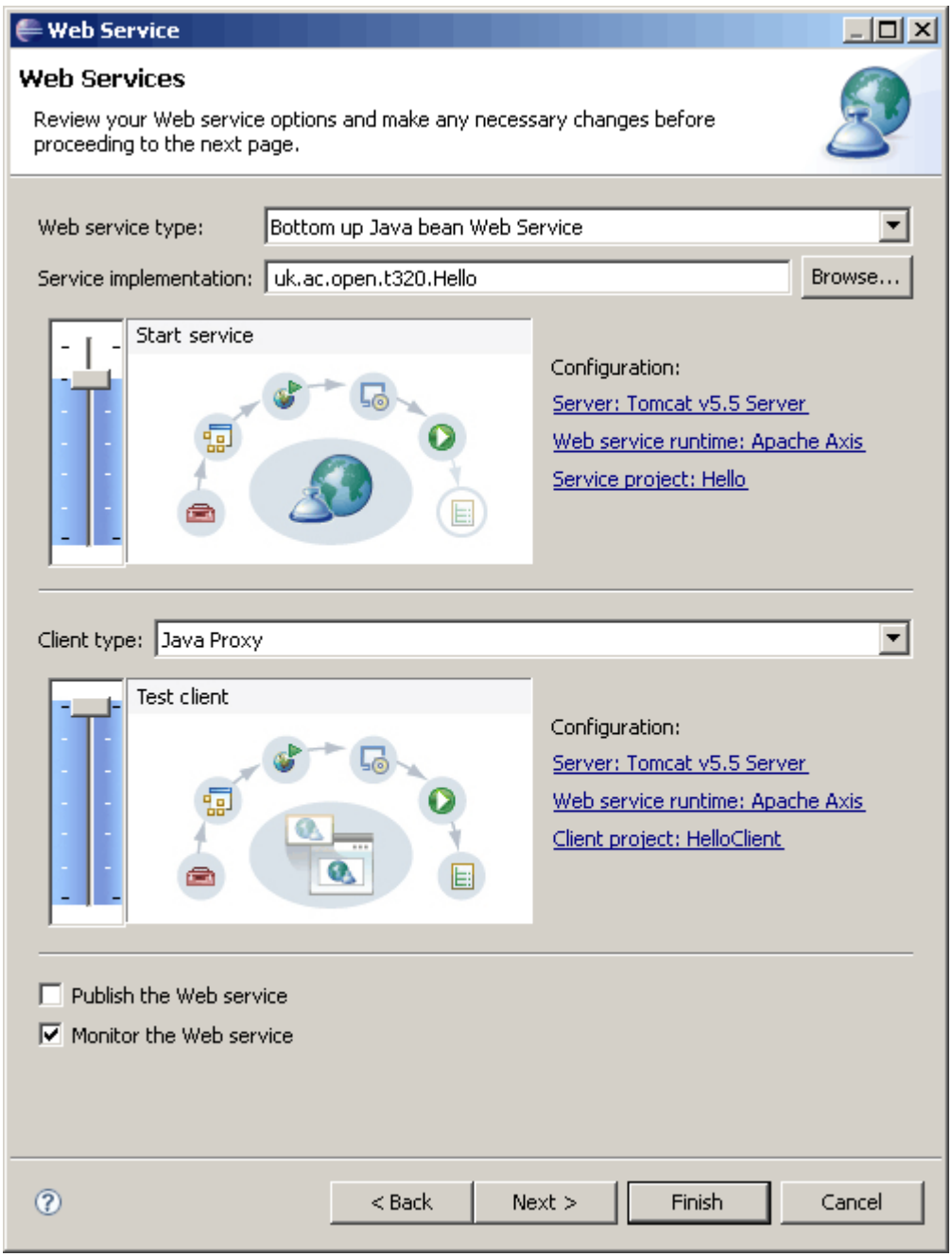

Figure 1 Web service configured (Figure 18 from *Implementing a Simple Web Service*)

When you arrive at the situation depicted in Figure 1, you should follow the steps listed in the rest of this document rather than continue with the original activity.

Choosing style and encoding with Eclipse

When you last created the 'Hello' web service and reached the state shown in Figure 1, you clicked on the 'Finish' button. However, it's also possible to click 'Next'; do this, and you will be presented with the dialogue box shown in Figure 2.

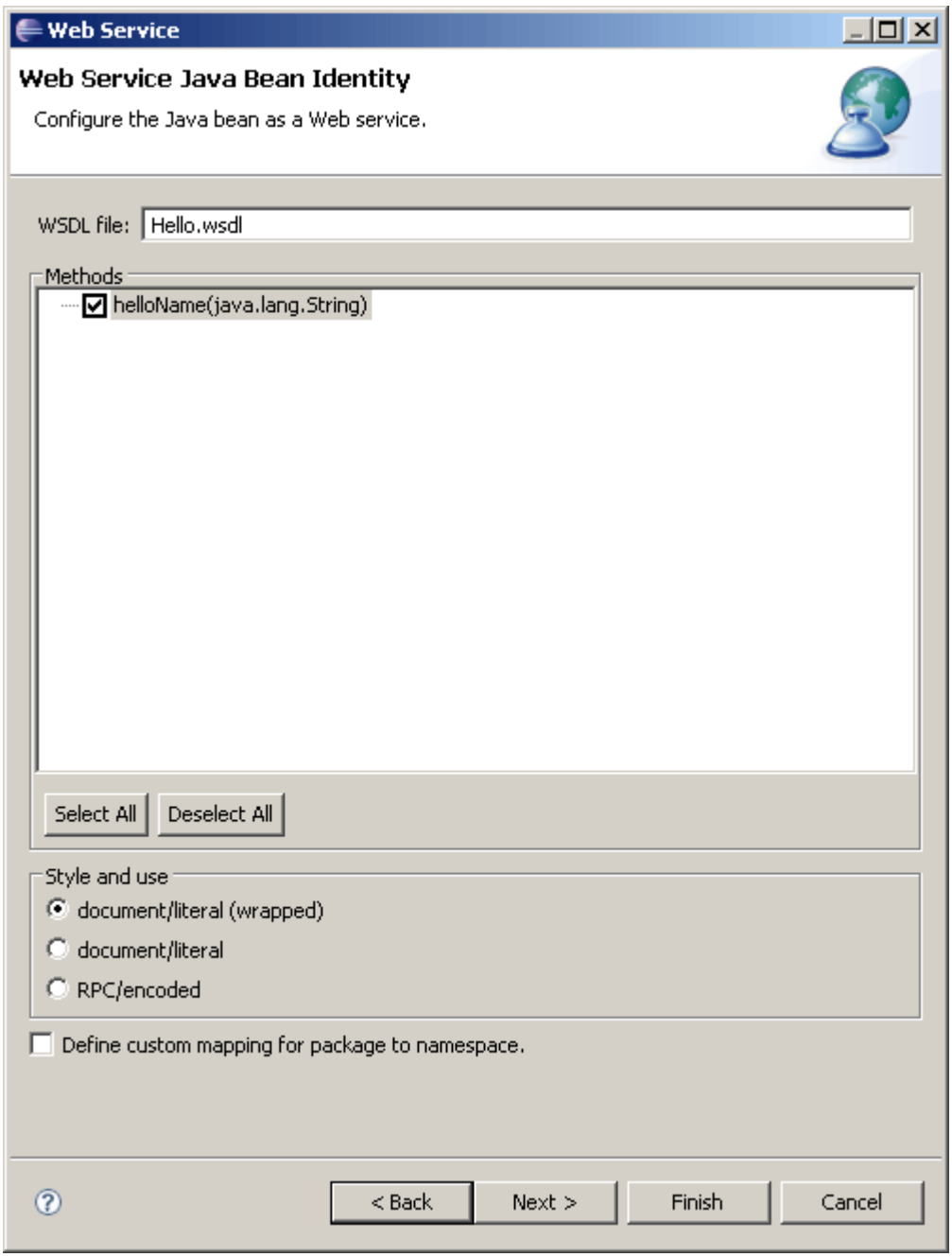

Figure 2 Configuring the Java class as a web service

Towards the bottom of this dialogue box you will see that 'Style and use' can be selected. There are three options available, of which the default option is 'document/literal (wrapped)'. The other two options, RPC/literal and document/encoded, are not offered by Eclipse.

Document/encoded is a style that is not used, as noted in Part 2 (refer to the article *Which style of WSDL should I use?* if you need to refresh your memory). RPC/literal is WS-I compliant and might be useful, but is not supported by Eclipse when creating a web service in this fashion.

In most cases it is appropriate to use the document/literal wrapped approach. If, however, you select RPC/encoded then you can experiment with this approach and we can also continue on and look at some other facilities in Eclipse. So select 'RPC/encoded' in the dialogue box and click the 'Next' button.

You will then be presented with a warning message (Figure 3). This, perhaps not unexpectedly, warns you that the web service you are creating may not conform to the

WS-I basic profile. (Unfortunately, even if you click 'Details' to learn more you will not be told exactly why the warning has been issued.) Click the 'Ignore' button to continue.

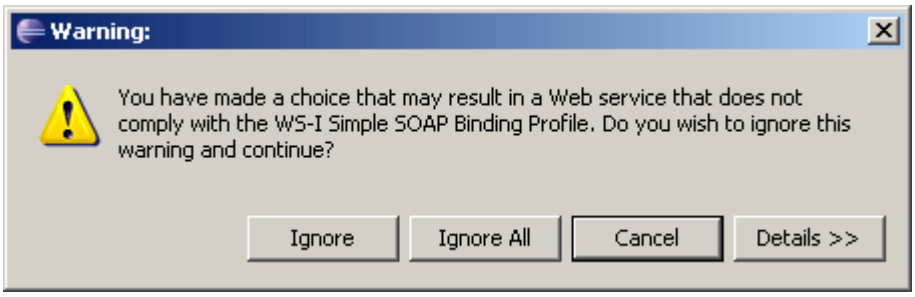

Figure 3 Warning that web service may not comply with WS-I profile

At this stage, if Tomcat is not already running then you will be asked to start it (Figure 4). Click on the 'Start server' button; when Eclipse has started Tomcat, the 'Next' button will become available and you should click on it.

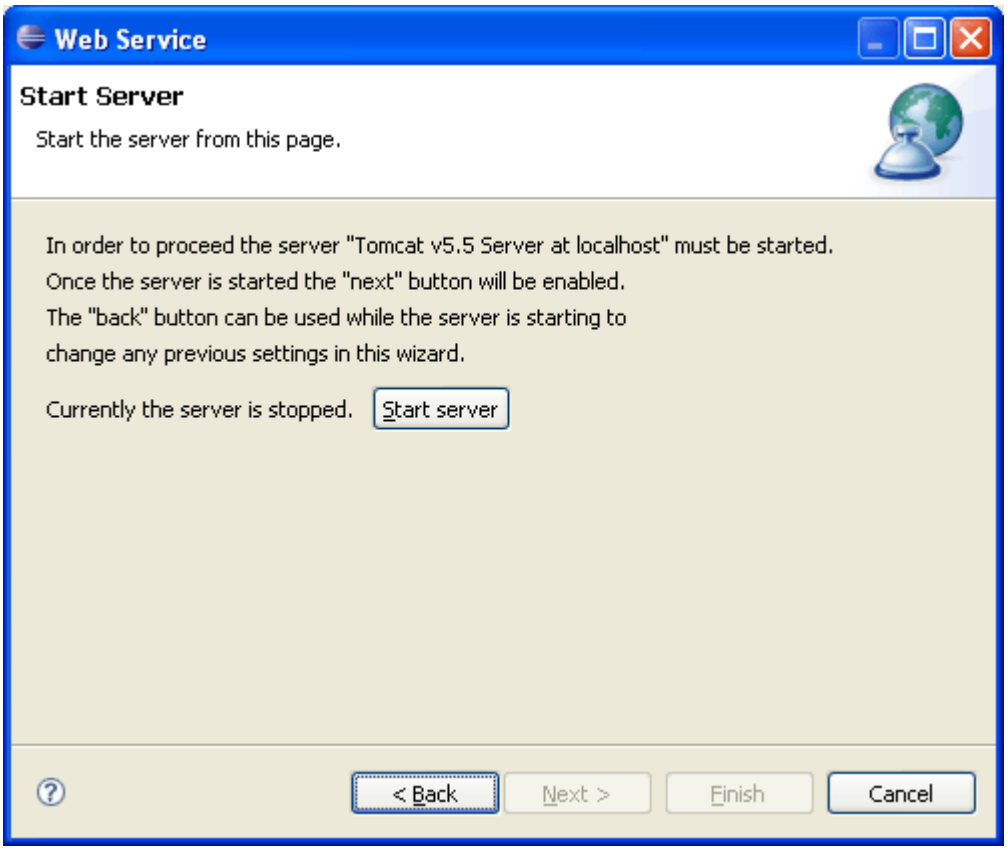

Figure 4 Start Server dialogue box

The next dialogue box is used to select a 'test facility'. Other facilities for testing can be added to Eclipse, but the default course version will offer to use the 'Web Services Explorer' (Figure 5). This is intended, as you have seen, to access a service via UDDI; however, we will not be using UDDI here, so this option is irrelevant.

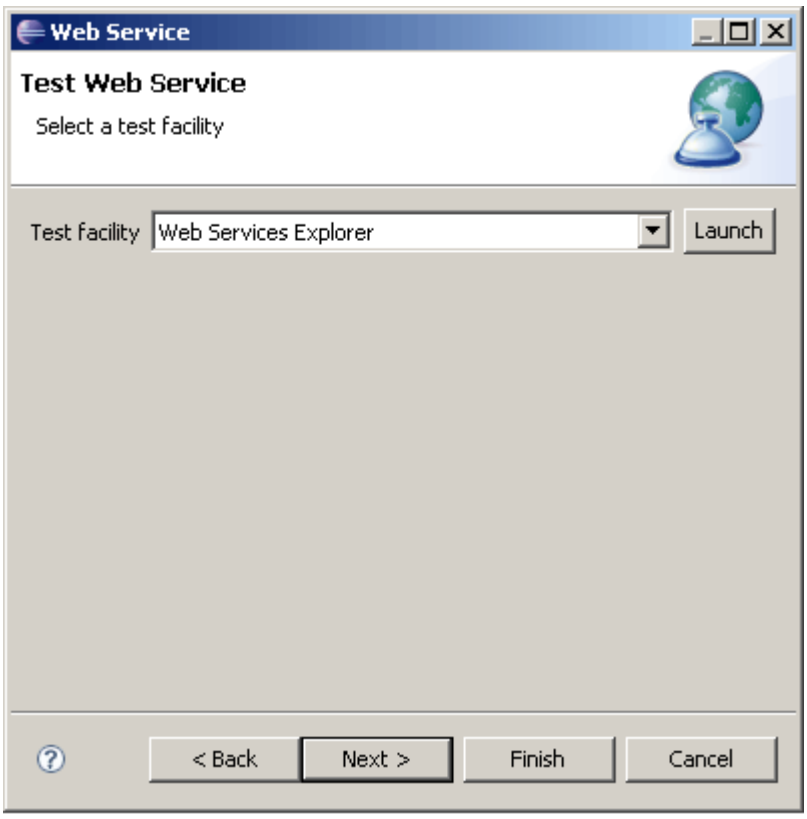

Figure 5 Dialogue box to select a test facility

Clicking 'Next' will move you to the next dialogue box (Figure 6). Here you can specify where the proxy code will be placed, or simply accept the default location.

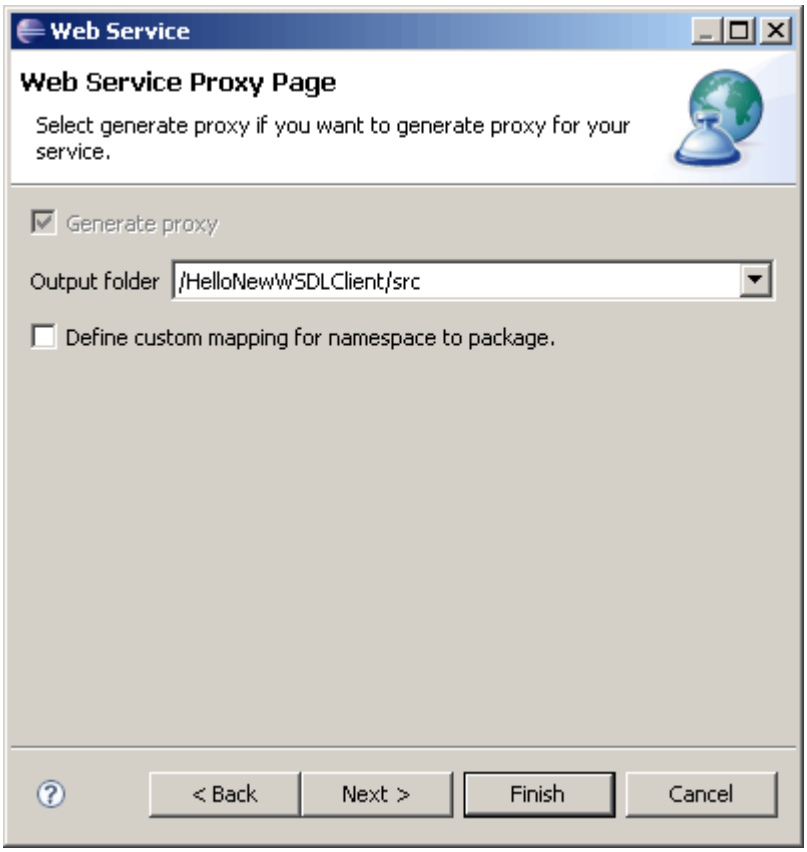

Figure 6 Dialogue box to select a web proxy output location

When you are ready, click 'Next' to continue to the next dialogue box, which is shown in Figure 7.

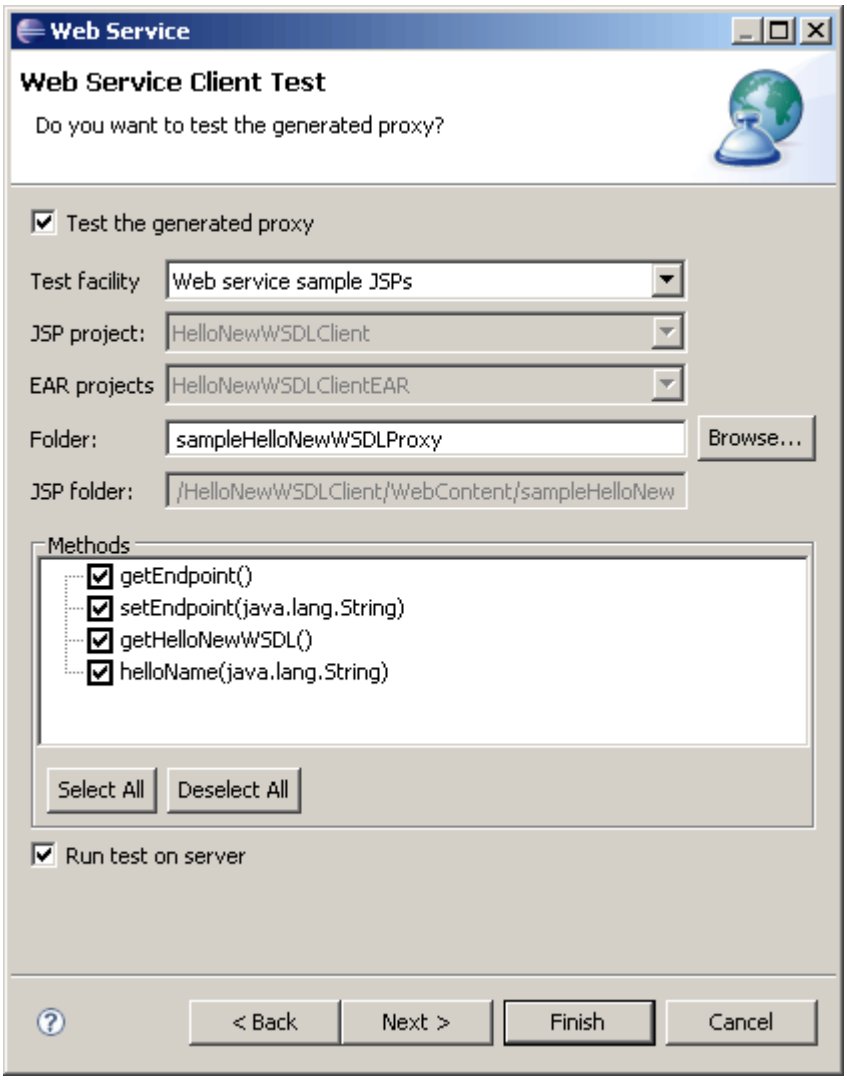

Figure 7 Configuring the web service client

Here, amongst other things, you can see that the client will by default be a set of Java Server Pages (the client web pages that you used earlier in the block to call the 'Hello' web service), and that four methods are listed and will be made accessible. You can decide which methods to expose in the test client's pages by deselecting those you are not interested in. In Figure 8, all but the method that was explicitly written as the service 'helloName' are deselected.

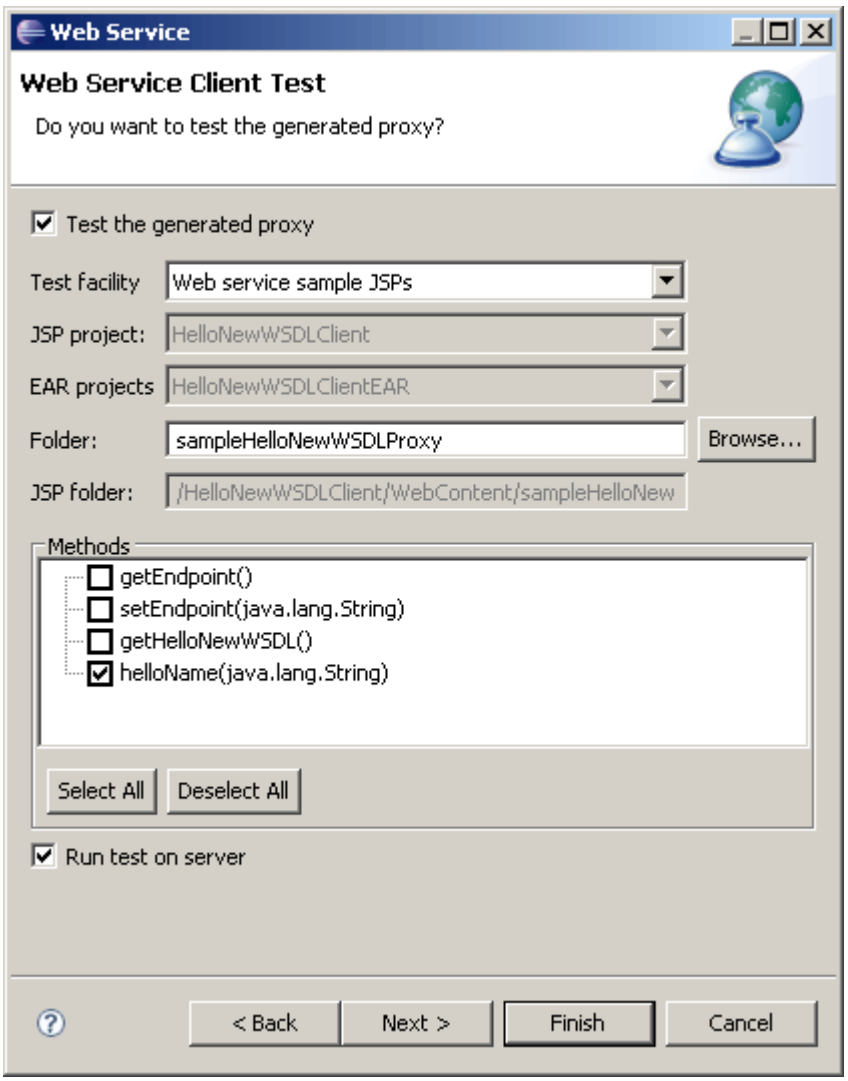

Figure 8 Configuring the web service client: methods deselected

The only remaining dialogue box after Figure 8 allows you to specify a UDDI directory for publication. Unless you want to do that, simply click 'Finish' at this point; after a short time you will be presented with the test client in Eclipse (Figure 9). You can see that the 'HelloNewWSDL' project that I created to hold this new service's components is listed in the Project Explorer on the left.

You could test the service out now, of course, but what about the WSDL that the process has generated as a by-product in the project's 'WebContent' folder? This should be different to the previous WSDL.

If you expand your project folders out, you will see that there is a small warning sign on the icon for the 'WebContent' folder, which is repeated on the WSDL file's icon (Figure 10).

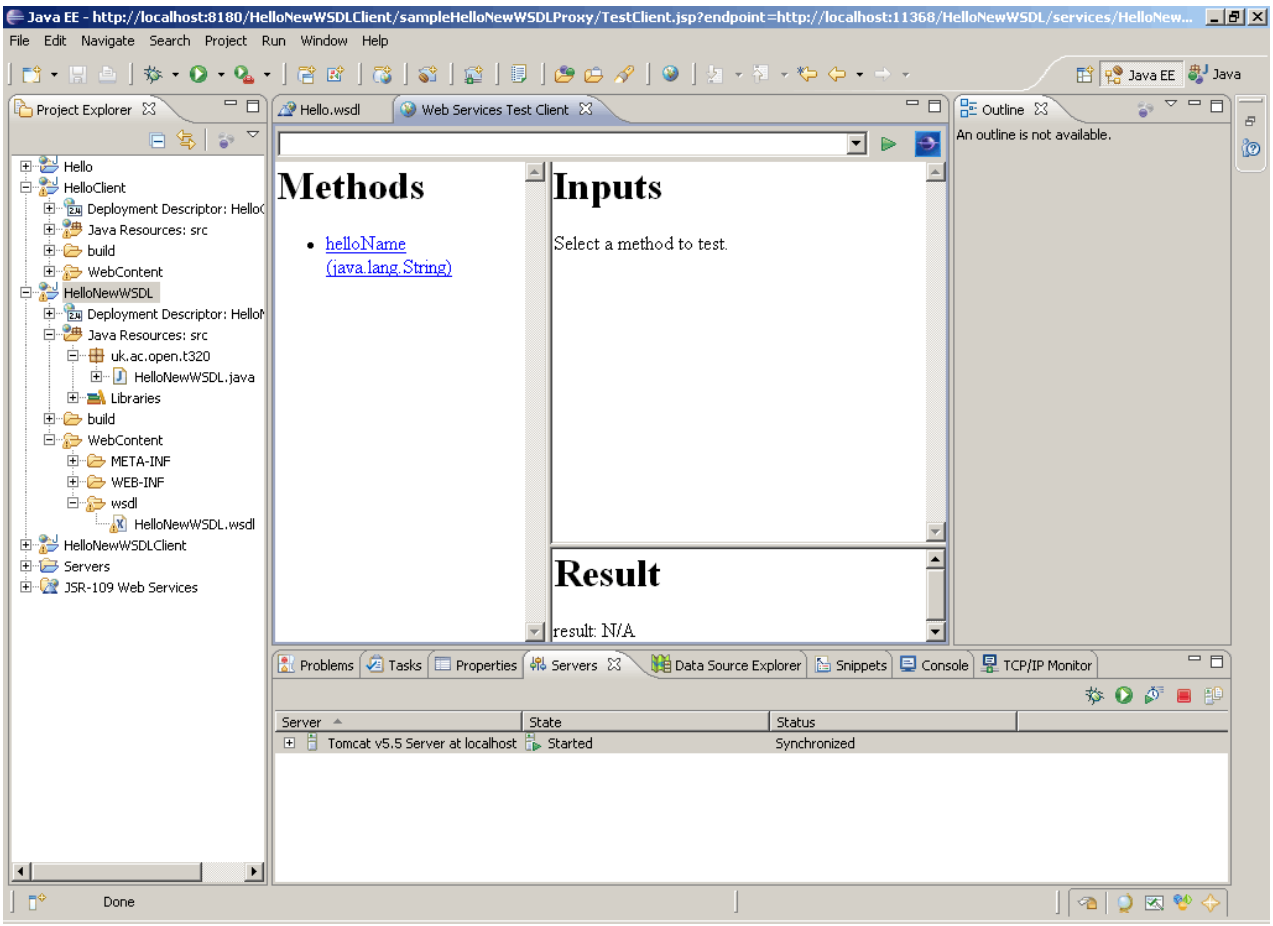

Figure 9 Client web pages in Eclipse

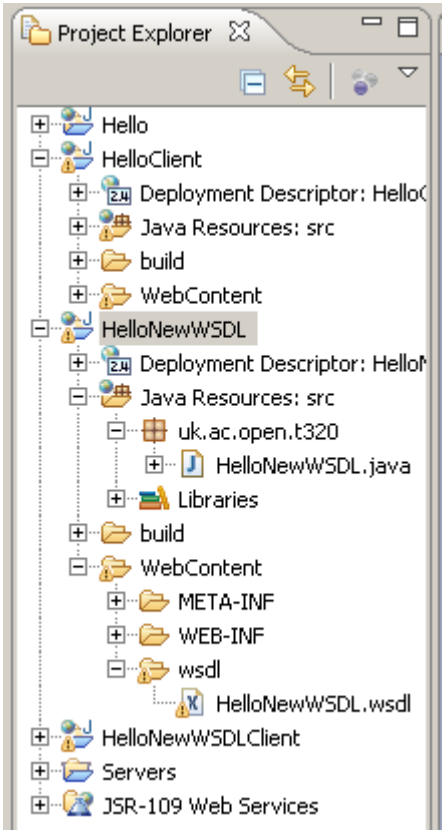

Figure 10 Project folders expanded to show warning icons

The warning concerns the WSDL file. If you start the WSDL editor (by double-clicking on the WSDL file and then switching to the Source view), you will see that there is a yellow marker on the right-hand edge of the file's Editor window in Eclipse (Figure 11). If you hover with the mouse over the yellow marker then you will see a pop-up message that describes the nature of the warning (Figure 12).

The pop-up message lists a WS-I basic profile requirement, BP2406. You should be able to find this requirement in the basic profile specification on the WS-I web site. The description given there can hardly surprise you. What might surprise you is how tightly the profile specifies such areas as encoding.

Compare the WSDL that you have just generated with the WSDL that you generated in the earlier activity and identify the significant differences. The Source view will provide some details of the WSDL that may not be so apparent in the graphical Design view.

You should find that the key differences centre around the `use` attribute and local schemas. If you return to the paper *Which style of WSDL should I use?* you should be able to correlate the WSDLs you have produced for the 'Hello' web service with Listing 2, which illustrates the RPC encoded style of WSDL, and Listing 8, which illustrates the document literal wrapped style. You can also examine the entire WSDL for the 'Hello' service, including the bindings that are not given in the paper.

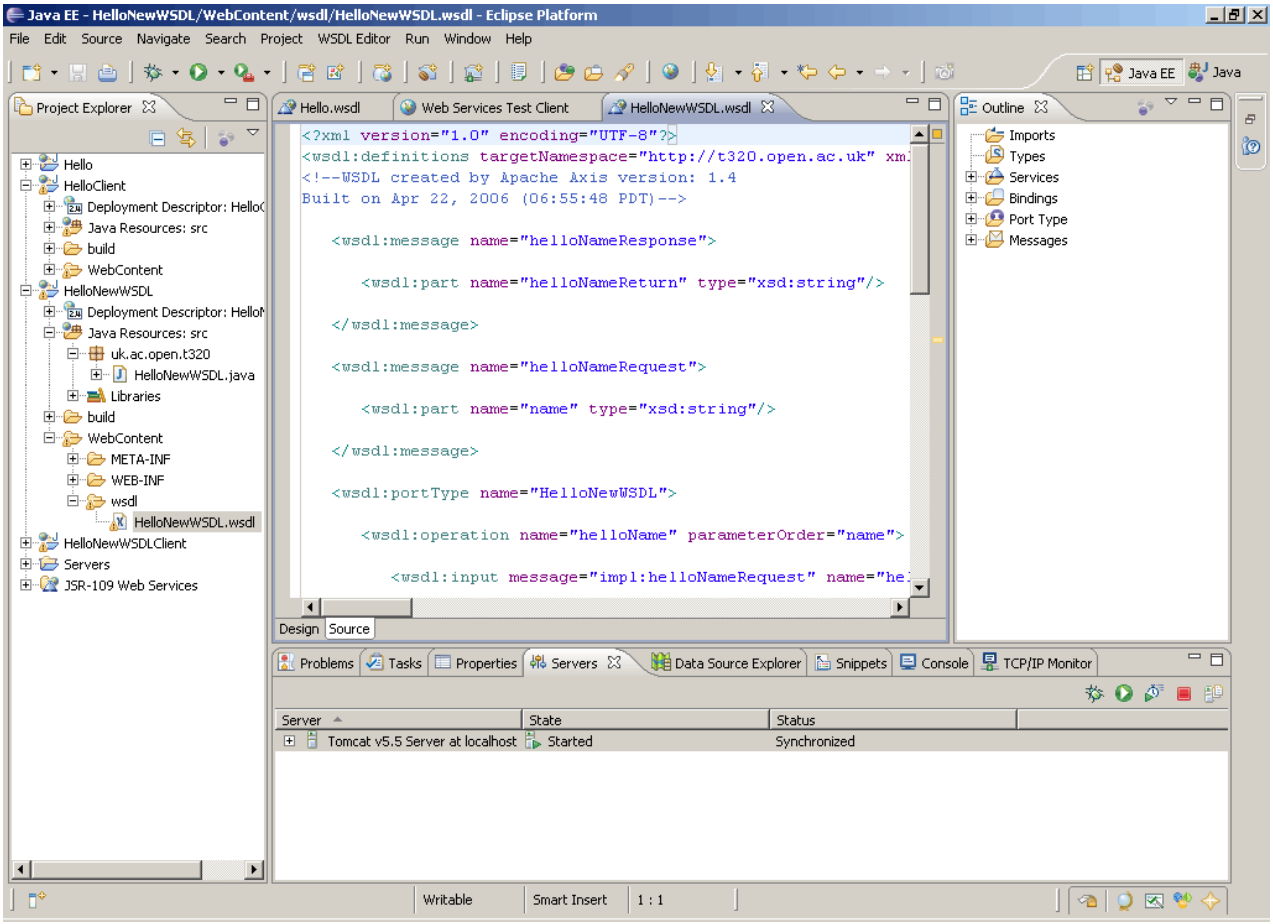

Figure 11 WSDL editor operating on WSDL with warnings

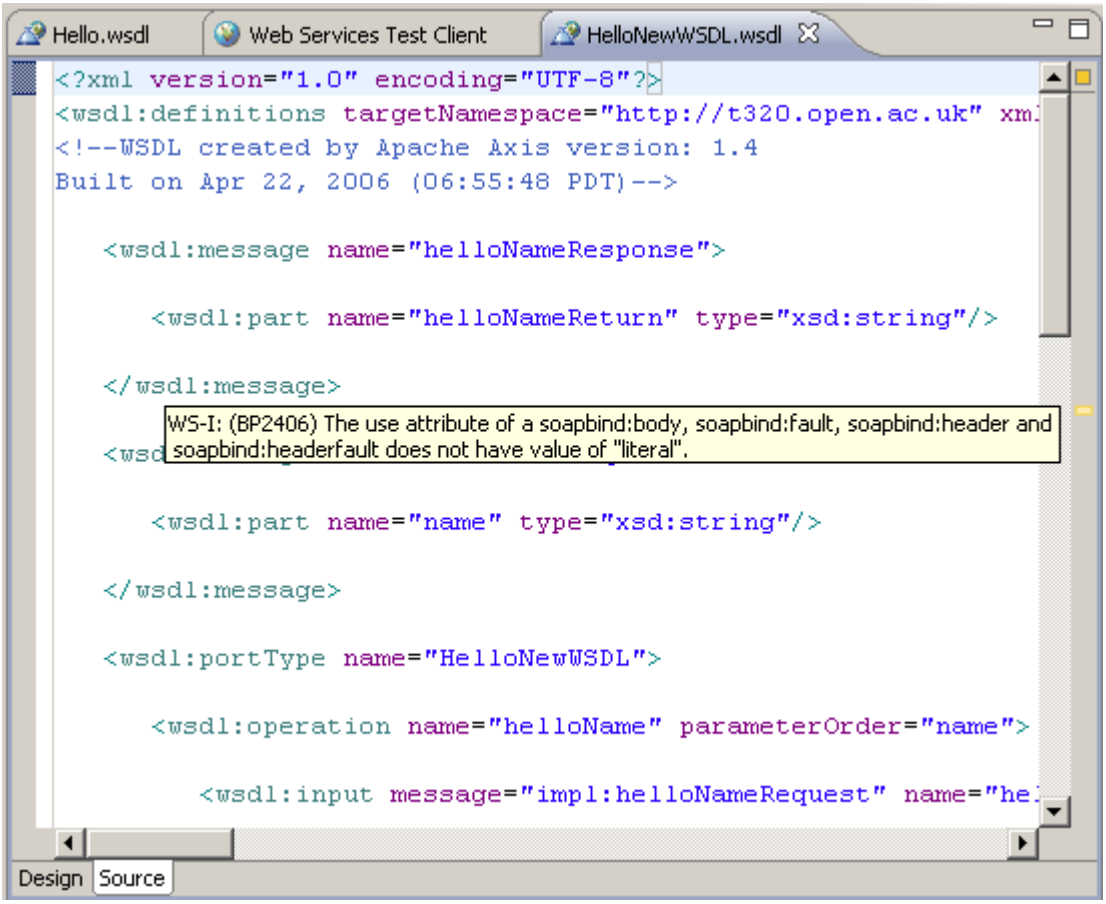

Figure 12 Pop-up warning message in WSDL editor