Practical 7

Theme

- connecting to an existing SOAP RPC Web service using
  - a generic client
  - one's own Java client written in Eclipse using Apache Axis
- developing a SOAP RPC Web service that makes use of and adds value to another such Web service

Key concepts: Web services, WSDL, SOAP binding, port, port type, XML schema, WS proxy, WS skeleton

7.1. Start up and essential configuration

a) *Log-in to Ubuntu and start a terminal*

b) *(On 10th December) Prepare to open BlackBoard assessment called* `Practical 7` *when required.*

Important: The quiz should be started within the **first 20 minutes** of the practical on 10th December. (**Date to be confirmed!**)

Notice that this is an assessed quiz that contributes to your module score.

7.2. Deploying the canvas Web service.

a) *Download* `lab7-canvas.zip` *from BlackBoard and extract it to* `CS3250-DS-1011`*.*

b) *Open the extracted folder* `lab7-canvas` *that contains a* `src2` *folder as a new* Dynamic Web Project *called* `lab7-canvas` *and move all folders from* `src2` *to* `src`*.*

c) *Refresh the project* `lab7-canvas` *in Eclipse.*

The project should now appear in the Project Explorer with an error indicator. The errors will disappear after the following two steps.

d) *Explore the structure of the file* `LineCanvas.wsdl`*.*

You can switch between source and design views as appropriate.

Notice that it imports the schema `Geom.xsd`.

Check the target name space of the WSDL document and the target name space of the schema.

Find the import command in the source view and ensure you understand how the elements of the schema are used inside the WSDL.

e) *Generate server skeleton from file* `LineCanvas.wsdl`.

(i) Right-click on the WSDL file in Project Explorer and select Web Services > Generate Java bean skeleton.

(ii) Ensure the server generation level is Start service.

(iii) Press Finish and wait until the skeleton has been generated.

f) *Explore the structure of the generated Java classes.*

Notice that apart from the proxy and its supporting classes Axis generated also classes `Point` and `Line` that represent the datatypes defined in the imported XML schema.

The class `PointHolder` enables the method `getCanvasSize` to return two points at once — the method takes two empty point holders as parameters and puts the points that it is meant to return into these holders.

> **Important.** The Eclipse Web service tools generate the implementation class `LineCanvasSOAPImpl` intelligently, merging the previous version (one was provided in the `lab7-canvas.zip` archive) with any newly generated skeleton code. Nevertheless, these tools sometimes go wrong, so it is a good idea to backup the implementation class before generating the skeleton.

7.3. Testing the server using a generic client.

a) *Locate the* ***generated*** *WSDL document* `LineCanvasSOAP.wsdl` *in the* `WebContent/wsdl` *folder.*

This WSDL document is generated by Axis and is almost the same as the original WSDL.

b) *Find one substantial difference between the generated WSDL document and the original.*

To see the differences, select *both* WSDL documents in the Project Explorer, right-click and choose Compare With > Each Other.

✎ **Quiz 10th Dec.** Answer and save **Question 1**, which is related to this exercise.

c) *Load the generated document* `LineCanvasSOAP.wsdl` *in the Eclipse Web Service Explorer.*

Right-click on the generated WSDL document and select Web Services > Test with Web Services Explorer. It will appear either in a new Firefox tab or inside Eclipse.

d) *Using the explorer, invoke the* `getCanvasSize` *operation and investigate its response.*

You should see two pairs of coordinates shown in the status view at the bottom of the explorer window and a new canvas window should pop up on the screen.

> **Hint.** If the WS explorer opened inside Eclipse, you may want to maximise the explorer's window by pressing `Ctlr-M`. To restore the window to its original size press `Ctlr-M` again.

> **Insight.** Why did the canvas not appear when the service was deployed, only after the first operation invocation? When the service is deployed, Tomcat does not load the classes until the service is first used. The canvas is created as soon as the class `LineCanvasSOAPImpl` is loaded into the server's JVM.

✎ **Quiz 10th Dec.** Answer and save **Questions 2 and 3**, which are related to this exercise.

e) *Using the explorer, observe the SOAP envelopes used in the invocation.*

f) *Draw at least one line on the canvas.*

To see the message representations in full, click on the source link in the status view.

✎ **Quiz 10th Dec.** Answer and save **Question 4**, which is related to this exercise.

7.4. Creating a simple client program.

a) *Create a new Eclipse dynamic web project called* `lab7-ptcanvas`.

You can create it at the default location (ie the workspace) or in any freshly created folder.

b) *Generate client proxy classes from the Axis-generated WSDL document* `LineCanvasSOAP.wsdl`.

(i) Right-click on `LineCanvasSOAP.wsdl` and select Web Services > Generate Client.

(ii) Click on the Client project: lab7-canvas link and select `lab7-ptcanvas` instead of `lab7-canvas`.

(iii) Set the generation level to Develop Client and click Finish.

c) *Create an executable class* `points.DrawPoint`.

You need to create the package first.

When creating the class, indicate that you want a main method created in the class. If you forget, you can create the main method conveniently by typing `main` and using the `Ctrl-Space` menu.

d) *Complete method* `DrawPoint.main` *so that it draws a 10x10 square near the centre of the canvas.*

Suggested sequence of actions in the `main` method:

(i) create an instance of the proxy class generated earlier

(ii) create two `PointHolder` objects

(iii) call `getCanvasSize` on the server to get the corner points

(iv) work out the coordinates of the canvas centre (**Hint**: The x coordinate of the centre can be computed as the average of the x coordinates of the two opposite corner points. Analogously compute the y coordinate.)

(v) create four `Point` objects, setting their coordinates approximately 5 points away from the centre in different directions to form a neat little square

(vi) call `drawLine` four times, creating a new `Line` object with each call

e) *Test your program.*

✎ **Quiz 10th Dec.** Answer and save **Question 5**, which is related to this exercise.

7.5. Creating a simple point-drawing canvas service.

a) *In project* `lab7-ptcanvas`*, create a new WSDL document called* `PointCanvas.wsdl`*.*

(i) Right-click on the project and use the New > other... dialogue to enter the WSDL wizard.

(ii) Enter `PointCanvas.wsdl` as the name of the new file and click Next.

(iii) Modify the default target namespace to `ptcanvas.wsrpc` and click Finish.

This will create the document with one default binding to one default port type with one default operation.

b) *Change the name of the operation to* `drawPoint`*.*

c) *Copy and paste the schema file* `Geom.xsd` *from the* `lab7-canvas` *project to the* `lab7-ptcanvas` *project.*

d) *Make the schema embedded in* `PointCanvas.wsdl` ***import*** *the schema* `Geom.xsd`*.*

For this task you need to switch to the *source* view.

Exactly the same import is used in `LineCanvas.wsdl`. You can copy and paste the appropriate portion from this document.

> **Important.** Continue *only* when the WSDL source editor shows no errors.

e) *Assign the prefix* `geom` *for the imported schema namespace using the attribute* `xmlns:geom` *in the root element of the embedded schema.*

Again, the same feature is used in `LineCanvas.wsdl`. You can copy and paste the appropriate portion from this document.

f) *Change the request message parameters type for operation* `drawPoint` *so that the method gets one point as a parameter.*

(i) Switch to the *design* view in the WSDL editor.

(ii) Click on the blue arrow next to the `drawPoint` *element* in the table.
This opens the schema editor, focusing on the definition of the `drawPoint` element. Its default definition is a complex type with one component named `in` of type `string`.

(iii) Switch to the design view in the schema editor.

(iv) Rename the component element from `in` to `point`.

(v) Change the type of the `point` element to `point` (`geom:point` in the source view).

g) *Change the response message parameters type for operation* `drawPoint` *so that it is empty.*

Delete the element `out` that was put into the message by default.

Finally delete the sequence constructor so that the type is an empty complex type.

h) *Generate binding content for the WSDL document.*

(i) Switch to the design view in the WSDL editor.

(ii) Right-click on the little binding box that connects the two larger blocks (ie the service and the port type) and select Generate binding content in the pop-up menu.

(iii) Survey and accept the default values and click Finish.

(iv) Check that the WSDL file has no errors nor warnings.

i) *Generate the service skeleton following the same pattern as you did with the* `lab7-canvas` *project.*

j) *Open the generated dummy implementation class* `PointCanvasSOAPImpl`.

k) *Add a static field called* `lineServer` *of type* `LineCanvasProxy` *and initialise it using the default constructor.*

l) *Write an appropriate body of the method* `drawPoint` *that draws a little rectangle around the given point, making calls to the proxy stored in the variable* `lineServer`.

m) *Test the new service in the Web Service Explorer (using the generated* `PointCanvasSOAP.wsdl`*).*

✎ **Quiz 10th Dec.** Answer and save **Question 6**, which is related to exercises in section 7.5.