Externalisation – A Simple Client and Server Application in Java

Externalisation is a process which allows a collection of objects to be converted into a format suitable for transmission between running processes.

In Part A of this lab, we will see how simple Java client and server applications can exchange data using externalisation.

In Part B of this lab, you will use the EclipseLink library to marshal Java objects representing orders within a server application, send those objects to a client application via a socket, and then unmarshal the order objects on the client side.

Part A instructions:

1. Open your favourite Java IDE and add the files Item.java, Order.java, OrderClient.java and OrderManager.java. Order.java to a new project (dsWeek4A). Inspect each of the files in turn. Order.java provides a simple class template for orders, while Item.java provides a template for items in an order. OrdrManager.java is the server application, which accepts requests from OrderClient.java. Note that the client makes requests to the server using a simple text-based message protocol (e.g. “bytecodes>>orders>>null”).
2. Build and run OrderManager.java and then OrderClient.java. We should see that the client receives the requested order data from the server
3. Notice that on the server side, Strings containing XML versions of the objects are generated in the getOrderAsXML() method. Could we do better using libraries from last week’s lab?

Part B Instructions:

1. Create a new Java project (dsWeek4B), and import the .java files which you generated using xjc and JAXB in last week’s lab. (If you do not have these files, download the ds_wk3_DataBindingLab.zip from Moodle and follow the instructions to generate the .java files using the XML Schema Definition).
2. Create a server class and a client class (use OrderClient.java and OrderManager.java in Part A above as templates).
3. When the client passes the message “xml” to the server, the server should return a PurchaseOrder object marshalled into XML back to the client. The client will then unmarshal the XML back into a PurchaseOrder object, and print its details to the console.
4. The XML passed to the client by the server should be generated using JAXB (examples of how to use the marshal and unmarshal methods provided by JAXB are provided in JAXBPOExample.java from last week’s lab). HINT: marshal the PurchaseOrder object into a string rather than a file, then send the string over the socket as per the example in Part A
5. Add extra functionality to the client and server which allows the client to request the PurchaseOrder details in JSON format.