COMP4711 Lab 10 (Winter 2009)
Dates: Mar 23-27, 2009

XML-RPC

Background

XML-RPC is a powerful tool for client-server or distributed processing.

We have the example client and server projects on share-out, and can use them as a starting point for building our own distributed application – a realtime survey service.

Your "survey server" will present a survey question and possible responses, and tally results. Your "survey client" will connect to the server, get the survey question & responses, and let you choose one and submit it to the server.

Lessons From Last Year

We discovered a few hiccups when we did a similar lab last year:

- a method to be called remotely needs to return something
- a String[] is returned as Object[], and we don't appear able to cast the result
- this is easiest to do using the demo client/server as a starting point, rather than copying from the PP slides

Lab Tasks

The "server" should implement the following interface

```
  public interface SurveyServer {
    /** Return a survey questionaire, with the question
        as the first array element and the possible
        responses as subsequent elements (numbered 1-?) */
    public String[] survey();
    /** Process a response, returning the most commonly
        chosen one so far. */
    public int vote(int item);
    /** Produce a "report", showing the survey results
        as a text message, with embedded newlines for
        formatting. */
    public String report();
}
```

The XML-RPC server should listen on port 2000.

Bind your handler class to “survey” inside the XML-RPC server program.
That way we can all work together ... clients would name the remote service they want “survey.x”, where “x” is the method they want to invoke.

The client program only needs to be command line driven, with two arguments: the host name/address and the port of the XML-RPC server. There should be default settings for these inside the program, in case the parameters are not supplied (localhost and 2000).

The client should ask the server for a survey question and possible responses, then present these as a “menu” to the user, with responses numbered from 1. It should then prompt for your vote, which it forwards to the survey server. Entering an # of 0 should cause it to redisplay the survey menu. Entering a # of 99 should exit the client.

The server should have an interesting question, and at least three possible answers. It will have to keep track of the description of each item and the tally. It should display a suitable message to the Java console with each vote.

We want to be able to have any of your client programs connect with any of your server programs. Hint: this could be an excellent fun conclusion to the lab ... having each of you run your client (multiple times) to connect to your classmates' servers.

If you are ambitious or want to show off, you can build GUI programs for the client and/or the server.

If you are not keen on doing the lab using Java (some of the data comm students expressed an interest in completing their lab in C, as that is variously the only language that real programmers know, or the only language that they know well), you are welcome to do so. Some sample C implementations are on shareout. I got them from the xmlrpc site, but have no idea if they work and can offer no support for a C implementation. I trust that any such submissions will be crystal clear on how to build and run them, as it has been a number of years since I have used C.

Submission

1. Build your project(s), per the specs above. It could be a single project, with two packages (client and server, naturally), and then two application entry points.
2. Zip up your project(s), and submit the zip file to share-in, using a suitable name, like ParryJimComp4711Lab10.zip

- Due by the end of the weekend (sunday 11:59pm)

Marking Guideline

This lab will be marked out of 10 ...

λ *3 marks for the client*
λ *4 marks for the server*
λ *3 marks for interopability*