COMP4711 Lab 8 (Fall 2014)
Dates: Nov 6, 2014

Application Glue – RPC & REST

Version 2

After the first lab using this writeup, the students asked if I could write it up instead as two webapps, not one. Here you go!

Background

This week and last, we looked at two types of "application glue", technologies used to connect distributed components. XML-RPC is appropriate to use a general purpose remote service, while REST is appropriate to use remote CRUD.

We shall apply XML-RPC to the ferry schedule model from lab 7, and REST to the menu maintenance from lab 5. This lab is to be done as your project pairs, and adds to your ongoing lab/assignment webapp project(s).

Our example will be somewhat contrived, in that we will set up both the client and the server on the same machine, albeit in separate webapps. The lessons will apply in the broader sense, though, and you could actually connect your client to a classmate's server to prove that the remote connection works.

Preparation

Make a project for this lab. I suggest starting with your lab 5, and incorporating the travel planner from lab 7. You could also do it the other way around :-/ Simplify the menu maintenance by keeping only one of the editing techniques – it doesn't matter which, as this lab is about the glue not the view.

Whichever approach you take, make sure that both the menu maintenance and the travel planner are clearly accessible from your homepage, as either prominent links/buttons, or by having them in your menu navbar.

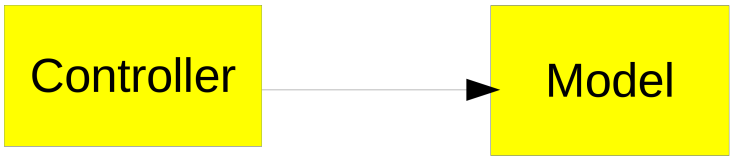
Now clone this webapp, calling the new one MyRemote (or something similar). I shall refer to the two webapps as MyLocal and MyRemote, for convenience.

Add a new virtual host entry for the new webapp, called **services.local**.

Usecase Description

You have working logic to handle travel planning and menu maintenance. Our goal is to decouple the models for these. Instead of using a local model inside your webapp, you will use appropriate glue to use a remote service, which in turn will use a local model at its end.

Original:



Decoupled:

Our approach will be to replace the model calls in MyLocal controllers with logic to make those same requests as distributed calls to services inside MyRemote.

Each “service” will be implemented through a controller in MyRemote. MyRemote would normally literally be remote, but we are going to fake it by running that webapp on our local XAMPP setup. The same code is used, but it is bound together differently :)

Without decoupling, a model reference is to a local object. After decoupling, we use a network reference to connect to a service which has that model as a local object.

Goodies

The lab folder on D2L contains 3 goodies for you:
- remote, a project with an XML-RPC server (remote)
- 4711labs-fall2012, a project with a REST client (lab11) & server (product)
- CI_rest, the third party pieces you need to add to your project

Lab Tasks – XML-RPC Client

1. Add two constants, RPC_SERVER and RPC_PORT, for the endpoint and port of the remote service. These will be “services.local/schedules” and “80” for our lab.

2. Where your logic originally referenced your local model, have it reference a new method which implements an XML-RPC client to retrieve the same data remotely. Use the names appropriate to your code!

   In your Planner controller:

   ```
   $ports = $this->ferryschedule->get_ports();
   ```
   becomes
   ```
   $ports = $this->get_ports_remotely();
   ```

and

```
$ports = $this->ferryschedule->get_trips($dest);
```
becomes
```
$ports = $this->get_trips_remotely($dest);
```

3. Implement the two new methods to retrieve the data from your XML-RPC server, using the constants defined above. These two methods (get_ports_remotely and get_trips_remotely, are new methods inside your Planner controller. Each implements an XML-RPC client.

4. Your MyLocal project no longer needs the ferry schedule model or XML data!

Lab Tasks – XML-RPC Server

5. Your MyRemote project needs, at this point, only the one controller, **Schedules** (referenced in the RPC_SERVER constant inside your MyLocal webapp. The MyRemote webapp will never be called upon to devliver a web page – it is only providing services to remote clients!

6. Using the “remote” project (goodies) as an example, map the two incoming requests used by your XML-RPC client to appropriate local methods. These methods will reference your local model... for instance

```
// get the local data
 $ports = $this->ferryschedule->get_ports();
// structure it to send back to the remote client
```

7. Remove any other controllers fro this webapp. We don't need them.

Lab Tasks – REST Client

8. Add two constants, REST_SERVER and REST_PORT, for the endpoint and port of the remote service. These will be “services.local/menuitem” and “80” for our lab.

9. Where your logic originally referenced your local model, have it reference a new method which implements an XML-RPC client to retrieve the same data remotely. Use the names appropriate to your code!

   In your Admin controller:

   ```
   $item = $this->menuitems->get($num);
   ```
   becomes
   ```
   $item = $this->get_item_remotely($num);
   ```

and

```
$this->menuitems->update($item);
```
becomes
```
$this->update_item_remotely($item;
```

10. You should end up with up to five new methods in your Admin controller, each of which maps to a CRUD request handeld by your REST server. Use the comp4711-labs-2012 projedt as an example.

11. Implement these new methods to retrieve the data from your REST server, using the constants defined above. Each implements a RESTful client.

12. Your MyLocal project no longer needs the menu model or XML data!

Lab Tasks – REST Server

13. Your MyRemote project needs, at this point, one more controller, **Menuitem** (referenced in the REST_SERVER constant inside your MyLocal webapp.

14. Using the “4711labs-fall2012” project (goodies) as an example, implement each of the incoming request types to perform CRUD on your local menu model.

Marking Guideline

This lab will be marked out of 10 ...

- 2 marks for your XML-RPC client
- 3 marks for your XML-RPC server
- 2 marks for your REST client
- 3 marks for your REST server

Submission

- Your zipped NetBeans lab project, submitted to the provided dropbox
- Due by 21:30 Sunday Nov 9