

Steps to build / test / client server.

1. Find out where stuff is routed. This is where a path from a URL is converted into a function call. The URL, '<http://Name.com:Port/path?queryParams=Values&more=val>' is split into
 1. http or https - this can be other protocols but these are the 2 common ones.
 2. name.com or an IP address like 127.0.0.1 - this is the address that is used to talk to the server.
 3. Port is a number, 80 for http, 443 for https or some other port number like 9000.
 4. /path is some sort of an end point path. It has become popular to put data into the path but I usually don't do that.
 5. ?queryParams=value this is the set of parameters if this is a GET request. POST / PUT / DELETE use a different format. If this is a multi-part-mime format for a file upload then it is a different format. POST may have its data in JSON or XML also. The "route" or "endpoint" is the /path part. This can be a more complex path like /path/bob/xyzy and is often the location of a file on the system for a static file. With a default NgNix install it will be something like /var/nginx/name.com/html/path . This will depend on the NgNix configuration.

For homework 5 this is xyzy In some projects you may need to add routes. Add the routes if necessary.

Test that the routes work with some dummy calls and just a static return data.

1. Split the functions that get called from the route into 2 chunks.
 1. The processing of parameters into values in memory.
 2. The implementation of whatever the function is to do (this is the business logic section)
2. Test the parameter processing section. This is usually very easy. Make client call - see code print out value on server.
3. Implement the business logic so you can call it from the command line. This will become your automated test stuff for the business logic. You will need to be able to call this and get data out so that you can have accurate data to test your client with.

xyzy - expaound - examples

xyzy - in this case some of this has been done for you look at ./sig-test - this is a command line program that will allow you to generate signatures and sign messages. (Hint: you are probably going to want to cut/paste code from this into your implementation of the client/server code.)

4. Test the business logic.

5. Put the business login into the server - replace the dummy functions.
6. Test again. This time with some existing clients like `curl` , `wget` or with a browser like `chrome` or `firefox` . `curl` and `wget` are easier to test with because you can put the tests into a script file and run them. Brwosers will work but you will have to cut/paste signatures and they are large.
7. You should have a working server at this point. Now on to the client.
8. Build a test case for the client that just pulls back some static data. This will test that your client is connecting to the server. For this it is handy to have a route like `/api/status` that just sends back a response that you can print out and verify connectivity (In the real world this is a huge step. Suppose that I am working on my computer that has to connect through outgoing security, across the internet, through a firewall, through some routing to a system at IBM, through IBM security, to an end point at IBM Food Trust. Lots of stuff can go wrong. The first thing that I want to know is can I talk to from my end to the correct system on that end.)
9. Take code from the CLI in step 4 and use that to build the client. You will need to use some code to get/post data across the web. In the case of Homework 5 this is the function `DoGet` that performs a GET request.