Lab 05: XML-RPC introduction

Sep 06, 2021

Prof. Shinnazar Seytnazarov

Faculty of Computer Science & Engineering

Implementing stubs revisited

❑ Recall the client/server stubs implementing a list

- We previously abstracted the socket with channel
- Here, how the implementation of stubs may look like using sockets

```
5   class DBClient:
6     def sendrecv(self, message):
7       sock = socket()                          # create a
8       sock.connect((self.host, self.port))     # connect
9       sock.send(pickle.dumps(message))         # send som
10      result = pickle.loads(sock.recv(1024)) # receive
11      sock.close()                             # close th
12      return result
13
14    def create(self):
15      self.listID = self.sendrecv([CREATE])
16      return self.listID
17
18    def getValue(self):
19      return self.sendrecv([GETVALUE, self.listID])
20
21    def appendData(self, data):
22      return self.sendrecv([APPEND, data, self.listID])
```

Implementing stubs revisited

❑ Recall the client/server stubs implementing a list

- Here, how the implementation of server stub may look like

```
 6  class Server:
 7      def __init__(self, port=PORT):
 8          self.host = 'localhost'                 #
 9          self.port = port                        #
10          self.sock = socket()                    #
11          self.sock.bind((self.host,self.port)) #
12          self.sock.listen(5)                     #
13          self.setOfLists = {}                    #
14
15      def run(self):
16          while True:
17              (conn, addr) = self.sock.accept() # ac
18              data = conn.recv(1024)              # fe
19              request = pickle.loads(data)        # un
20              if request[0] == CREATE:
21                  listID = len(self.setOfLists) + 1
22                  self.setOfLists[listID] = []
23                  conn.send(pickle.dumps(listID))
```

```
24
25              elif request[0] == APPEND:
26                  listID = request[2]
27                  data   = request[1]
28                  self.setOfLists[listID].append(data)
29                  conn.send(pickle.dumps(OK))
30
31              elif request[0] == GETVALUE:
32                  listID = request[1]
33                  result = self.setOfLists[listID]
34                  conn.send(pickle.dumps(result))
35              conn.close()                          # cl
```

Introduction XML-RPC

❑ XML-RPC

- is a Remote Procedure Call (RPC) method that uses XML passed via HTTP as a transport.
- With it, a client can call methods with parameters on a remote server (the server is named by a URI) and get back structured data.

❑ xmlrpc

- https://docs.python.org/3/library/xmlrpc.html
- is a package that collects server and client modules implementing XML-RPC. The modules are:
  - xmlrpc.client
  - xmlrpc.server

Example server

❑ Server

```
from xmlrpc.server import SimpleXMLRPCServer

# Create server
with SimpleXMLRPCServer(('localhost', 8000)) as server:
    server.register_introspection_functions()

    # Register pow() function; this will use the value of
    # pow.__name__ as the name, which is just 'pow'.
    server.register_function(pow)

    # Register a function under a different name
    def adder_function(x, y):
        return x + y
    server.register_function(adder_function, 'add')

    # Register an instance; all the methods of the instance are
    # published as XML-RPC methods (in this case, just 'mul').
    class MyFuncs:
        def mul(self, x, y):
            return x * y

    server.register_instance(MyFuncs())

    # Run the server's main loop
    server.serve_forever()
```

Example client

❑ Client

```python
import xmlrpc.client

s = xmlrpc.client.ServerProxy('http://localhost:8000')
print(s.pow(2,3))  # Returns 2**3 = 8
print(s.add(2,3))  # Returns 5
print(s.mul(5,2))  # Returns 5*2 = 10

# Print list of available methods
print(s.system.listMethods())
```

❑ Output

```
Lab05-xmlrpc> python .\rpc_server.py
127.0.0.1 - - [05/Sep/2021 22:34:48] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [05/Sep/2021 22:34:50] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [05/Sep/2021 22:34:52] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [05/Sep/2021 22:34:54] "POST /RPC2 HTTP/1.1" 200 -
```

```
Lab05-xmlrpc> python .\rpc_client.py
8
5
10
['add', 'mul', 'pow', 'system.listMethods', 'system.methodHelp', 'syst
em.methodSignature']
```

Any questions?