# Lab 8: HTTP

March 24, 2016

In this lab you will implement a simple *key-value* database server using HTTP as the communication protocol. A client may store pairs of key-value strings and later retrieve values by their corresponding keys via PUT and GET HTTP requests. Instead of coding a client, use the `curl` command to communicate with the server.

- To store a key-value pair on the on the server, issue a POST request using curl.

  ```
  curl -v -X POST -d key='hello' -d value='world'
      localhost:4321
  ```

  This command issues an HTTP POST to the server as shown below.

  ```
  POST / HTTP/1.1
  content-length: 20
  host: localhost
  content-type: application/x-www-form-urlencoded
  user-agent: curl/7.43.0
  accept: */*

  key=hello&value=world
  ```

  The server must store the data and respond with `200 OK` HTTP code with an empty body.

- To retrieve a value from the server using the corresponding key, issue a GET request using curl

```
curl -v localhost:4321/hello
```

This command issues an HTTP GET to the server as shown below.

```
GET /hello HTTP/1.1
Host: localhost
User-Agent: curl/7.43.0
Accept: */*
```

The server responds with `200 OK` HTTP code and the corresponding value in the body. If the key does not exist send a `404 NOT FOUND` response.

Complete the `KVServer.run` method to implement the logic described above. For simplicity assume that URL decoding of the POST body is not required. The `Message` class provides serialisation and deserialisation HTTP requests and responses.