**Exercise02\_01\_01 – Step 1**

In this Exercise we will learn the fundamentals of HTTP communication, obtain a Twitter account, set up a Twitter application, and set up our development environment.



1. Open a chrome browser and turn the Developer tools on with F12. Go to the Network tab and expand it upwards. Enter the following URL:  
   ***www.packtpub.com***
2. The various HTTP requests should start appearing in the list. Scroll up and click on the first request ***www.packtpub.com***. Find the Request Method, it should show as ***GET***.
3. Now let’s click the Log in hyperlink and give it something. . Scroll up and click on the first request ***www.packtpub.com***. Find the Request Method, it should show as ***POST***. Scroll down to the bottom and find the Form Data section. We see it has the email and password we just entered, and the password is in the clear. Click on ***view source*** next to ***Form Data***. We can see the actual request body that the browser sent to the server.
4. Scroll up to the General area and look for the Status Code. We can see that the value is ***200 OK***. That does not mean our login was accepted, just that the HTTP response was valid.
5. Now let’s enter a bogus URL:  
   ***www.packtpub.com/blabla.php***Click on that request and we can see that we got a Status Code of ***404 Not Found***.

**Exercise02\_01\_01 – Step 2**



1. Log in to your Twitter account or obtain a new one. Follow slightly more than 20 accounts. Please make sure that they are appropriate and that they would be demonstrable to guests or your Principal.

**Exercise02\_01\_01 – Step 3**



1. Go to the website ***https://developer.twitter.com*** and login with your Twitter login and password. Click the ***Create New App*** button. You may name your App whatever you please, I will build one for each class: ***C202-1 REST APP*** and ***C202-2 REST APP.***
2. Fill in the Description with something like ***An application for leaving notes on Twitter friends and followers.*** For Website we will put in a temporary placeholder. Let’s use ***http://www.example.com***. For ***Callback URL***, we need to put in a URL to return to during the authorization process. For now we will put in another temporary URL ***http://www.example.com/callback***. Check the ***Developer*** ***Agreement*** and click ***Create Your Twitter Application***.
3. For now, make sure that the ***Access level*** is set at ***Read and write***. If it is not, use the link to ***modify app permissions*** to set it.
4. Now that the app is set up we can find our API keys under the ***Keys and Access Tokens*** tab. We will come back to this later when we set up our code for authentication. For now, let’s log out of the site.
5. Now let’s set up our database, MongoDB. Go to the terminal and enter the following commands:  
   ***cd /  
   mkdir data  
   cd data  
   mkdir db***
6. Run the following command and show the instructor your results, one of the two of them should work correctly  
   ***mongod --dbpath-data/db  
   mongod --dbpath ~/data***

**Exercise02\_01\_01 – Step 4**



1. Create a folder named ***Exercise02\_01\_01*** and open it with your IDE. First we need to set up a node.js project in the folder. Go to the folder in your terminal and enter the following commands:   
   ***npm config set package-lock false***  
   ***npm init***For the ***package name***, enter: ***twitter-notes***Take the default for **version**For description we will use the same one as our Twitter App:  
   ***An application for leaving notes on Twitter friends and followers.***default all the rest, except put your own name in as author.Go back to your IDE and you should have a ***package.json file***. Open it up and make sure it is all okay. If not, modify it as necessary and save it.
2. Now let’s get ***ExpressJS*** installed in our local node\_modules and as a dependency in package.json. From the terminal or IDE enter the following npm command:  
   ***npm install express --save***  
   You should get a local ***node\_modules***. You may not see it in the IDE if you have turned it off with the plug-in, but it should be there in the terminal. Check ***package.json*** to make sure express was added as a dependency.
3. We can now create a node.js application as a server. Create a file called index.js and enter the following code:  
   ***var express = require('express');  
   var app = express();  
   var port = 8080;  
     
   app.listen(port, function() {  
    console.log("Server listening on localhost:%s", port);   
   });***Run the application and make sure the server is listening.
4. Now let’s add a ***GET*** ***endpoint*** and test to see if our browser can connect to it:  
   var port = 8080;  
     
   ***app.get("/", function(req, res) {  
    res.send("<h3>Hello, world!</h3>");  
   });***Bring up the browser and navigate to the endpoint.