**School of Computing**

**ST0503 Back End Web Development**

**Practical 2**

**Core Node JS**

|  |
| --- |
| **Objectives:**  After completing this lab, you should be able to:   * Setup and run a node js web server * Process and read in the http message body * Handle get and post method * Use Postman to test the web server api for different web methods |

**POST request**

We will be building our web server to support login based on REST principles and we will simulate the creation of a login api functionality in our web server.

In chapter 4, we will be introducing RESTFUL web api and understanding the differences between some of the key http methods. But for now, just take it that post is the most suitable method for login.

**LOGIN** - create a resource

Request:

METHOD POST => http://localhost:3000/login

BODY => {“user”: “admin@abc.com”, “password”: “1234567”}

Response

**If successful:**

http status code 200

{'message': 'Welcome admin!'}

**If fail:**

Redirect user to http://localhost:3000/index.html

**Part 1: Extracting the request message body**

Unlike a simple get request whereby we can retrieve the data from the request headers, when submitting a post request, the data passed to the web server is encapsulated in the request message body. The request req object we have seen in the lecture does not enable an easy way for us to retrieve the message body data.

Study the below web article from nodejs organization to understand the anatomy of an http transaction:

<https://nodejs.org/es/docs/guides/anatomy-of-an-http-transaction/>

|  |
| --- |
| const { headers, method, url } = req;  let body = [];    req.on('error', (err) => {  console.error(err);  }).on('data', (chunk) => {  body.push(chunk);  }).on('end', () => {  body = Buffer.concat(body).toString();  console.log(body.length);  // At this point, we have the headers, method, url and body, and can now  // do whatever we need to in order to respond to this request.  }); |

Incorporate the above code into your original server.js file you did in the lecture.

Note that the request methods and file serving will have to be “re-positioned” in the code.

**Part 2: Handling a new post method:**

Add a new post method to handle the login:

|  |
| --- |
| if (req.method == 'POST') {    ………  } |

For now, we assume username must always be [admin@abc.com](mailto:admin@abc.com) and password is 1234567

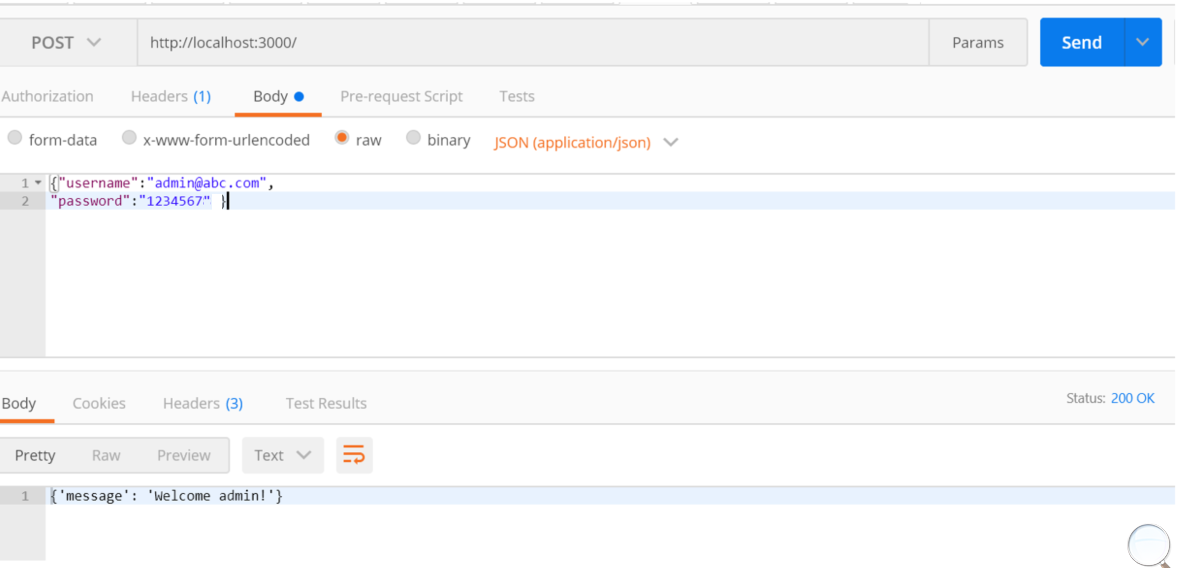
Hint: You can use JSON.parse(…); to parse the body content and read in the username and password data accordingly. To redirect, you can use the writeHead(…) function from response object.

**Part 3: Testing with Postman**

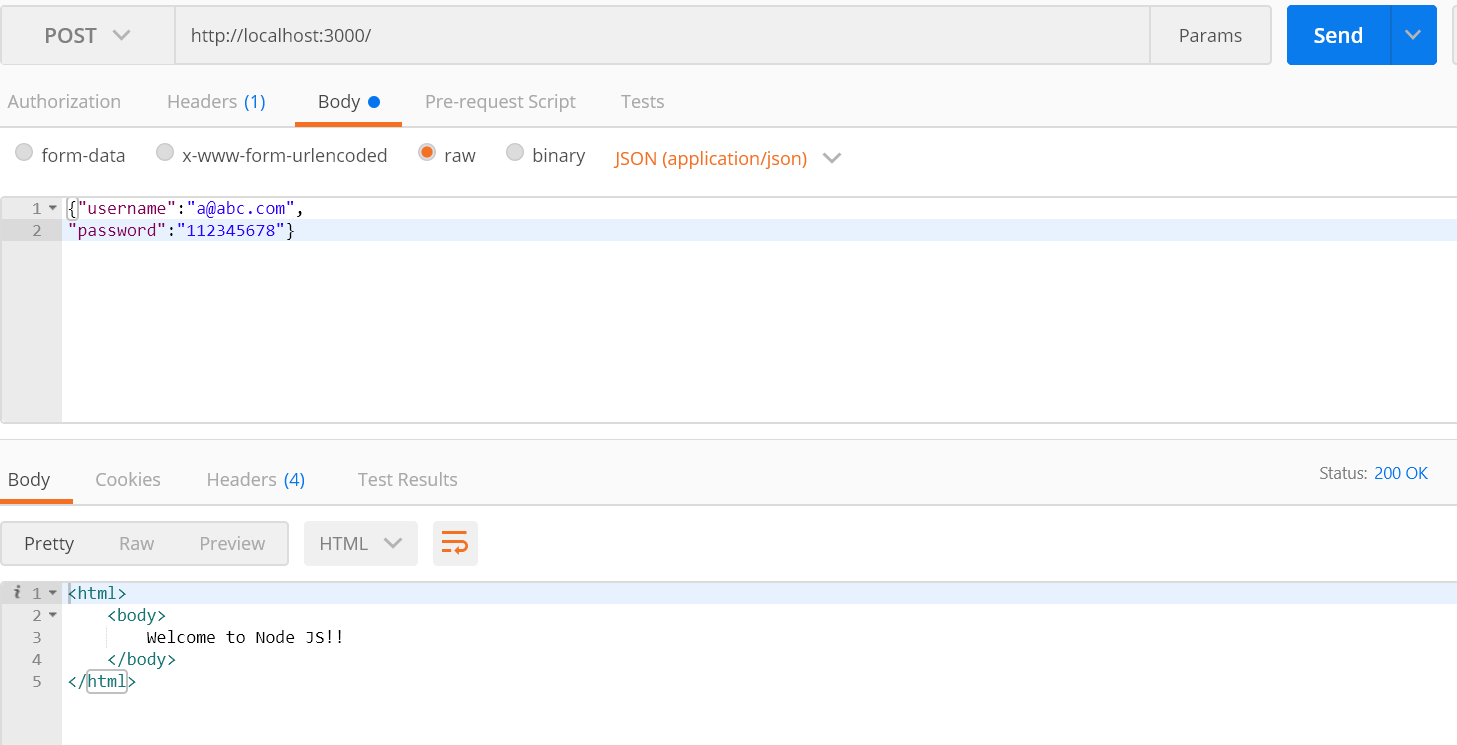
In order to test our POST webservice, we need to use another application POSTMAN which is downloadable from https://www.getpostman.com/.

Start your application and web server by running node server.js.Then test your web application by using postman.

**Correct Credentials:**

****

**Wrong Credentials:**

****