Express.js Web Application Tutorial

In this tutorial, you will learn how to create Web applications using Express.js.

Express.js is a Node.js framework that provides an easy way to create Web server and render HTML pages for different HTTP requests by configuring routes for your application.

1. Create a folder (directory) for your project.

Ex: if you have a D: drive it is better to create a workspace for your web applications such as www.

```
D:\> mkdir www

D:\> cd www

D:\www>
```

Now for each project you create a directory. For example app1 will be in a separate directory.

```
D:\www> mkdir app1

D:\www> cd app1

D:\www\app1>
```

2. Install Express.js using the NPM tool

(Note: you must have internet connection and install Express locally in your application folder)

```
D:\www\app1>npm install express
```

3. Create the Web Server

In your application folder create app.js file using VS Code

```
D:\www\app1>code . <enter>
```

(note: notice the . after the command code. It is important.)

App.js

```
//1. Load the module and create the app object
const express = require('express');
const app = express();

//2. define routes here..
app.get('/', function (req, res) {
    res.send('<h1>Hello World</h1>');
});

//3. Create the server to listen to web requests
const server = app.listen(5000, function () {
    console.log('Node server is running..');
});
```

In the code above we loaded the express module using require(), then the express() to create an object "app" which can be used to configure Express application. The 'app' object has methods for routing HTTP requests, configuring middleware, rendering HTML views and registering a template engine. [we will see all this by examples] . The app.listen() function creates the Node.js web server at the specified host and port.

Save the file and run the code as follows:

```
D:\www\app1>node app
```

```
Node server is running on port 5000..
```

Now in the browser write: http://localhost:5000/

4. Defining Routes

Use app object to define different routes for your application. The app object includes get(), post(), put() and delete() methods to define routes for HTTP GET, POST, PUT and DELETE requests respectively.

The following example shows you how to configure routes for HTTP requests.

App.js

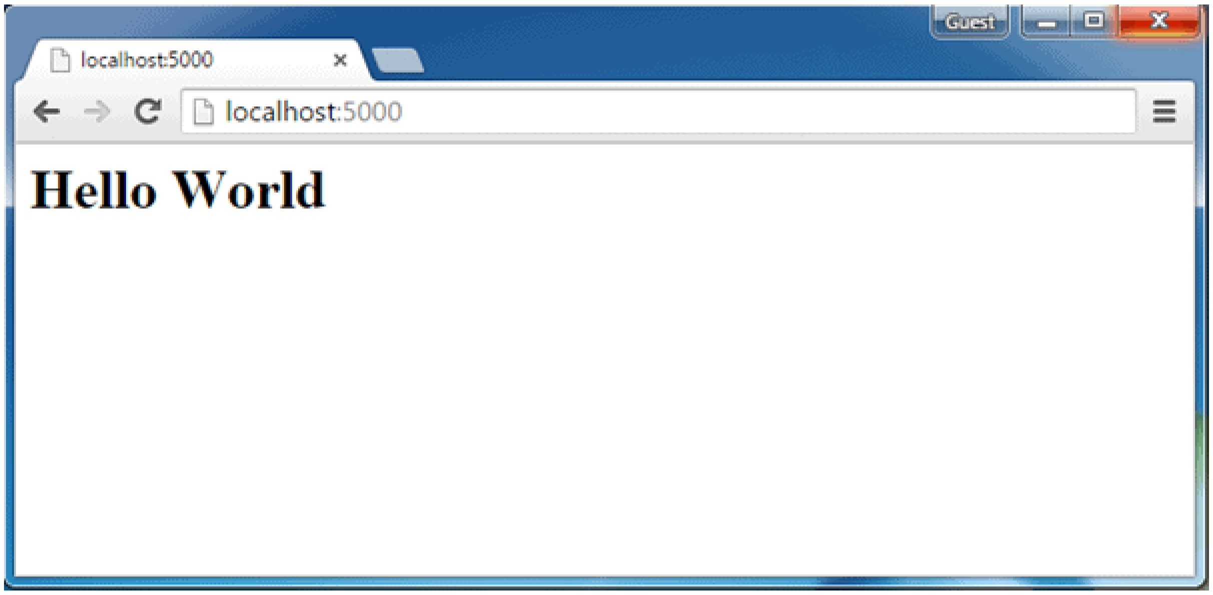
```
//1. load the express module and create the app object

const express = require('express');
const app = express();
// 2. Define routes
app.get('/', function (req, res) {
    res.send('<h1>Hello World</h1>');
});
app.get('/submit-data', function (req, res) {
    res.send('POST Request');
});

app.post('/submit-data', function (req, res) {

    //some code to handle submit-data

});

app.put('/update-data', function (req, res) {

    //some code to handle update-data

});

app.delete('/delete-data', function (req, res) {

    //some code to handle delete-data

});
```

In the above example, app.get(), app.post(), app.put() and app.delete() methods define routes for HTTP GET, POST, PUT, DELETE respectively. The first parameter is a path of a route which will start after base URL [/submit-data]. The second parameter is a callback function which includes request and response objects. This function will be executed on each request.

Run the above example using

```
d:\www\app1>node app.js
```

command, and point your browser to http://localhost:5000 and you will see the following result.



Hello World

5. Handle POST Request

Here, you will learn how to handle HTTP POST request and get data from the submitted form.

First, create form1.html file in the root folder of your application and write the following HTML code in it.

form1.html

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <title></title>
</head>
<body>
    <form action="/submit-data" method="post">
        First Name: <input name="firstName" type="text" /> <br />
        Last Name: <input name="lastName" type="text" /> <br />
        <input type="submit" />
    </form>
</body>
</html>
```

Body Parser

To handle HTTP POST request in Express.js version 4 and above, you need to install middleware module called body-parser. This module was a part of Express.js earlier but now you have to install it separately.

This body-parser module parses the JSON, buffer, string and url encoded data submitted using HTTP POST request. Install body-parser using NPM as shown below.

```
D:\www\app1> npm install body-parser
```

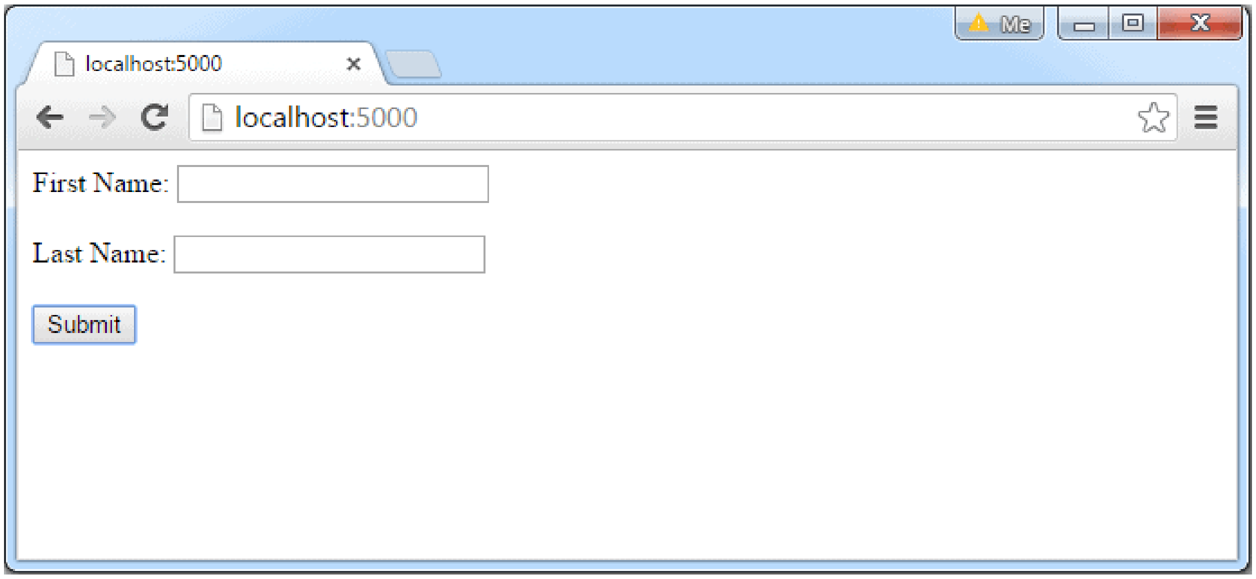
Now, import body-parser in your app.js code and use the POST request data as shown below.

App.js

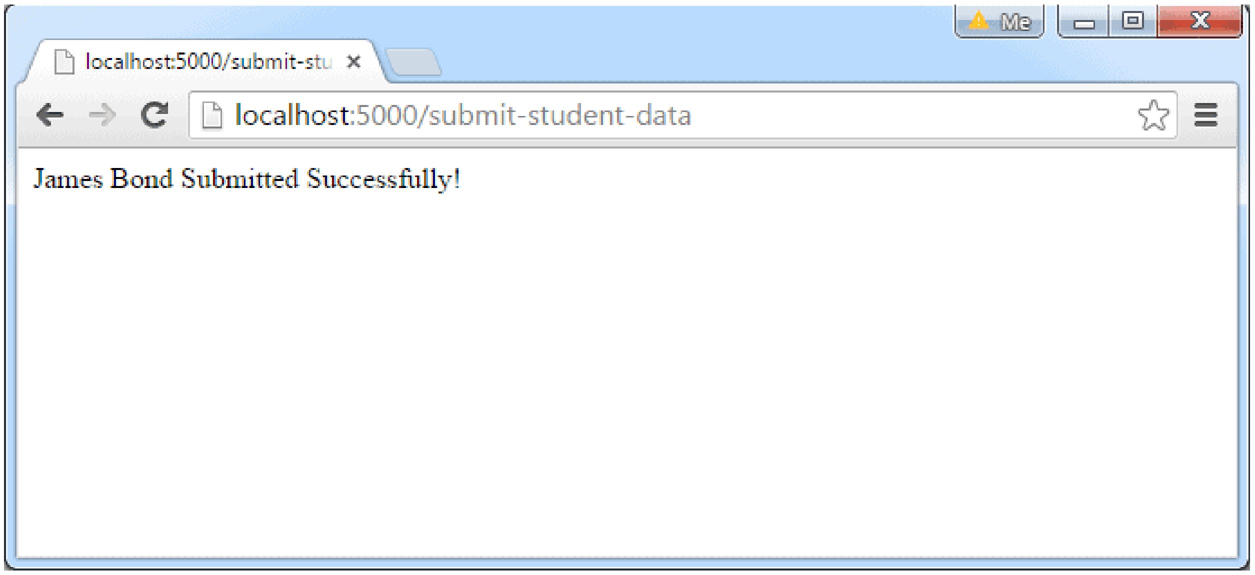
```
const express = require('express');
const app = express();

const bodyParser = require("body-parser");
app.use(bodyParser.urlencoded({ extended: false }));

app.get('/', function (req, res) {
    res.sendFile(__dirname + '/form1.html');
});

app.post('/submit-data', function (req, res) {
    var name = req.body.firstName + ' ' + req.body.lastName;

    res.send(name + ' Submitted Successfully!');
});

const server = app.listen(5000, function () {
    console.log('Node server is running..');
});
```

In the above example, we used req.body to access POST data. The req.body is an object that includes properties for each submitted form. Our form1.html contains firstName and lastName input fields, so you can access them using req.body.firstName and req.body.lastName.

Now, run the above example using `node  app.js` command, point your browser to *http://localhost:5000* and see the following result.



Fill the First Name and Last Name in the above example and click on **submit**. For example, enter "James" in First Name textbox and "Bond" in Last Name textbox and click the submit button. The following result is displayed.



James Bond Submitted Successfully!

6. Handle Static Files

Here, you will learn how to handle static files such as images, css, javascript, html, and text files.

To serve static files you can use the built-in middleware module in Express.js called express.static. The express.static() method, you can server static resources directly by specifying the folder name where you have stored your static resources as follows:

App.js

```
var express = require('express');
var app = express();

//setting middleware
app.use('/static', express.static(__dirname + '/public')); //Serves
resources from public folder using the /static route


var server = app.listen(5000, ()=>{
console.log('Server is running on port 5000'
});
```

Now, run the above code using node app.js command and point your browser to http://localhost:5000/static/myImage.jpg and it will display myImage.jpg from the /public folder (public folder should have myImage.jpg)