# Building the Node.js Web Application Framework with Express Framework

## Web Color Picker: <https://rgbcolorcode.com/color/E6FFE6>

## Let’s start by building a simple HTML page:

## <https://www.tutorialspoint.com/html5/html5_tags.htm>

## Instructions:

## Make a website on some topic of personal interest:

## Learning Accountabilities are:

## Demonstrate the ability to create a web application using Node.js, Express.js, and showing some images on the web page!.

## At the end include in this Word Document a screen shot of a web browser running on your computer showing the HTML page served up by your Express.js application!

## You must use Node.js and Express.js

## You must use some middleware such as the middle ware to server files. You MUST serve up through your Express Web Application 3 or more images. (Your images.google.com)

## Express Overview

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework −

* Allows to set up middlewares to respond to HTTP Requests.
* Defines a routing table which is used to perform different actions based on HTTP Method and URL.
* Allows to dynamically render HTML Pages based on passing arguments to templates.

## Installing Express

Firstly, install the Express framework globally using NPM so that it can be used to create a web application using node terminal.

$ npm install express --save

The above command saves the installation locally in the **node\_modules** directory and creates a directory express inside node\_modules. You should install the following important modules along with express −

* **body-parser** − This is a node.js middleware for handling JSON, Raw, Text and URL encoded form data.
* **cookie-parser** − Parse Cookie header and populate req.cookies with an object keyed by the cookie names.
* **multer** − This is a node.js middleware for handling multipart/form-data.

$ npm install body-parser --save

$ npm install cookie-parser --save

$ npm install multer --save

## Hello world Example

Following is a very basic Express app which starts a server and listens on port 8081 for connection. This app responds with **Hello World!** for requests to the homepage. For every other path, it will respond with a **404 Not Found.**

var express = require('express');

var app = express();

app.get('/', function (req, res) {

res.send('Hello World');

})

var server = app.listen(8081, function () {

var host = server.address().address

var port = server.address().port

console.log("Example app listening at http://%s:%s", host, port)

})

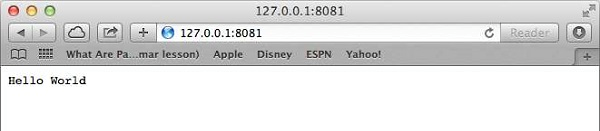
Save the above code in a file named server.js and run it with the following command.

$ **node server.js**

You will see the following output −

Example app listening at http://0.0.0.0:8081

Open http://127.0.0.1:8081/ in any browser to see the following result.



## Request & Response

Express application uses a callback function whose parameters are **request** and **response** objects.

app.get('/', function (req, res) {

// --

})

* [Request Object](https://www.tutorialspoint.com/nodejs/nodejs_request_object.htm) − The request object represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers, and so on.
* [Response Object](https://www.tutorialspoint.com/nodejs/nodejs_response_object.htm) − The response object represents the HTTP response that an Express app sends when it gets an HTTP request.

You can print **req** and **res** objects which provide a lot of information related to HTTP request and response including cookies, sessions, URL, etc.

## Basic Routing

We have seen a basic application which serves HTTP request for the homepage. Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).

We will extend our Hello World program to handle more types of HTTP requests.

var express = require('express');

var app = express();

// This responds with "Hello World" on the homepage

app.get('/', function (req, res) {

console.log("Got a GET request for the homepage");

res.send('Hello GET');

})

// This responds a POST request for the homepage

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

console.log("Got a POST request for the homepage");

res.send('Hello POST');

})

// This responds a DELETE request for the /del\_user page.

app.delete('/del\_user', function (req, res) {

console.log("Got a DELETE request for /del\_user");

res.send('Hello DELETE');

})

// This responds a GET request for the /list\_user page.

app.get('/list\_user', function (req, res) {

console.log("Got a GET request for /list\_user");

res.send('Page Listing');

})

// This responds a GET request for abcd, abxcd, ab123cd, and so on

app.get('/ab\*cd', function(req, res) {

console.log("Got a GET request for /ab\*cd");

res.send('Page Pattern Match');

})

var server = app.listen(8081, function () {

var host = server.address().address

var port = server.address().port

console.log("Example app listening at http://%s:%s", host, port)

})

Save the above code in a file named server.js and run it with the following command.

$ node server.js

You will see the following output −

Example app listening at http://0.0.0.0:8081

Now you can try different requests at http://127.0.0.1:8081 to see the output generated by server.js. Following are a few screens shots showing different responses for different URLs.

Screen showing again http://127.0.0.1:8081/list\_user



Screen showing again http://127.0.0.1:8081/abcd

Screen showing again http://127.0.0.1:8081/abcdefg



## Serving Static Files

Express provides a built-in middleware **express.static** to serve static files, such as images, CSS, JavaScript, etc.

You simply need to pass the name of the directory where you keep your static assets, to the **express.static** middleware to start serving the files directly. For example, if you keep your images, CSS, and JavaScript files in a directory named public, you can do this −

app.use(express.static('public'));

We will keep a few images in **public/img** sub-directory as follows −

node\_modules

server.js

public/

public/img

public/img/studying.png

Let's modify "Hello Word" app to add the functionality to handle static files.

var express = require('express');

var app = express();

app.use(express.static('public'));

app.get('/', function (req, res) {

res.send('Hello World');

})

var server = app.listen(8081, function () {

var host = server.address().address

var port = server.address().port

console.log("Example app listening at http://%s:%s", host, port)

})

Save the above code in a file named server.js and run it with the following command.

$ node server.js

Now open http://127.0.0.1:8081/img/studying.png in any browser and see observe following result.

