# **Express.js**

"Express is a fast, unopinionated minimalist web framework for Node.js"

npm install -g express

## Advantages of Express.js

1. Makes Node.js web application development fast and easy.
2. Easy to configure and customize.
3. Allows you to define routes of your application based on HTTP methods and URLs.
4. Includes various middleware modules which you can use to perform additional tasks on request and response.
5. Easy to integrate with different template engines like Jade, Vash, EJS etc.
6. Allows you to define an error handling middleware.
7. Easy to serve static files and resources of your application.
8. Allows you to create REST API server.
9. Easy to connect with databases such as MongoDB, Redis, MySQL

# **Express.js Web Application**

## Web Server –

var express = require('express');

var app = express();

// define routes here..

var server = app.listen(5000, function () {

console.log('Node server is running..');

});

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

## Example: Configure Routes in Express.js

var express = require('express');

var app = express();

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

res.send('<html><body><h1>Hello World</h1></body></html>');

});

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

res.send('POST Request');

});

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

res.send('PUT Request');

});

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

res.send('DELETE Request');

});

var server = app.listen(5000, function () {

console.log('Node server is running..');

});

## Handle POST Request –

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

## Example –

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title></title>

</head>

<body>

<form action="/submit-student-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**

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

**Note -** To handle HTTP POST request in Express.js version 4 and above, you need to install middleware module called [body-parser](https://github.com/expressjs/body-parser).

**npm install body-parser –save**

Now, import body-parser and get the POST request data as shown below.

**Example**: Handle POST Route in Express.js

var express = require('express');

var app = express();

var bodyParser = require("body-parser");

app.use(bodyParser.urlencoded({ extended: false }));

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

res.sendFile('index.html');

});

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

var name = req.body.firstName + ' ' + req.body.lastName;

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

});

var server = app.listen(5000, function () {

console.log('Node server is running..');

});

## Serve Static Resources using Express.js - Using express.static() method, you can server static resources directly by specifying the folder name where you have stored your static resources.

var express = require('express');

var app = express();

//setting middleware

app.use(express.static(\_\_dirname + 'public')); //Serves resources from public folder

var server = app.listen(5000);

**Note -** If you have different folders for different types of resources then you can set express.static middleware as shown below.

**Example: Serve resources from different folders**

var express = require('express');

var app = express();

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

//Serves all the request which includes /images in the url from Images folder

app.use('/images', express.static(\_\_dirname + '/Images'));

var server = app.listen(5000);

## Note - You can also create a virtual path in case you don't want to show actual folder name in the url.

**Example: Setting virtual path**

app.use('/resources',express.static(\_\_dirname + '/images'));

## Serve Static Resources using Node-static Module - The node-static module is an HTTP static-file server module with built-in caching.

## npm install node-static

## Example: Serving static resources using node-static

var http = require('http');

var nStatic = require('node-static');

var fileServer = new nStatic.Server('./public');

http.createServer(function (req, res) {

fileServer.serve(req, res);

}).listen(5000);