

# Express.js

- Developed by TJ Holowaychuk in Nov 2010
- Is a web application framework for Node.js
- Free and open source software
- License under MIT
- Built on the top of the Node.js that helps manage a server and routes.

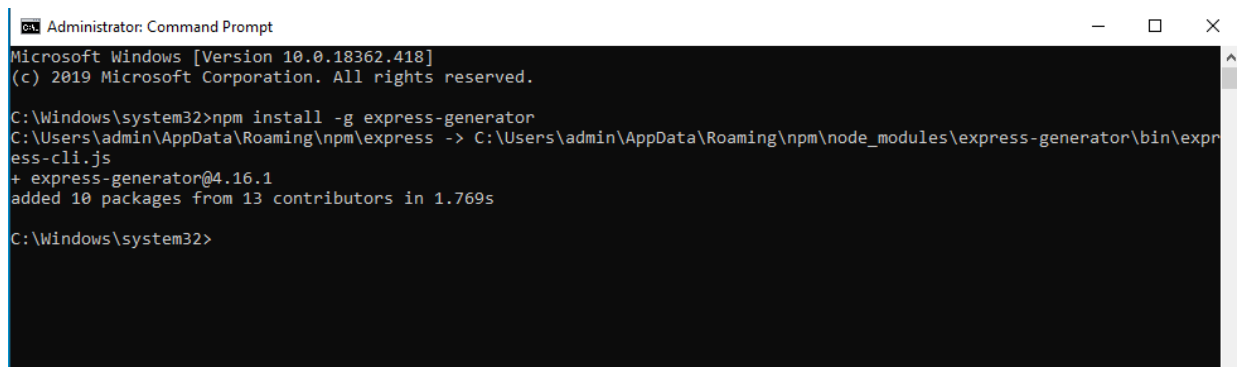
## Features:

- Facilitates the rapid development
- Allows to set up middleware to respond to HTTP Requests
- Defines a routing table
- Allows to dynamically render HTML Pages
- Uses template engines

## Installation:

Node.js should be installed on your machine.

### 1. Install express-generator



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.18362.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>npm install -g express-generator
C:\Users\admin\AppData\Roaming\npm\express -> C:\Users\admin\AppData\Roaming\npm\node_modules\express-generator\bin\express-cli.js
+ express-generator@4.16.1
added 10 packages from 13 contributors in 1.769s

C:\Windows\system32>
```

### 2. Create project in the particular folder using express command.

```
C:\Users\harshita\Documents\express_generator_tryout>express myfirstapp

warning: the default view engine will not be jade in future releases
warning: use '--view=jade' or '--help' for additional options

create : myfirstapp\
create : myfirstapp\public\
create : myfirstapp\public\javascripts\
create : myfirstapp\public\images\
create : myfirstapp\public\stylesheets\
create : myfirstapp\public\stylesheets\style.css
create : myfirstapp\routes\
create : myfirstapp\routes\index.js
create : myfirstapp\routes\users.js
create : myfirstapp\views\
create : myfirstapp\views\error.jade
create : myfirstapp\views\index.jade
create : myfirstapp\views\layout.jade
create : myfirstapp\app.js
create : myfirstapp\package.json
create : myfirstapp\bin\
create : myfirstapp\bin\www

change directory:
> cd myfirstapp

install dependencies:
> npm install

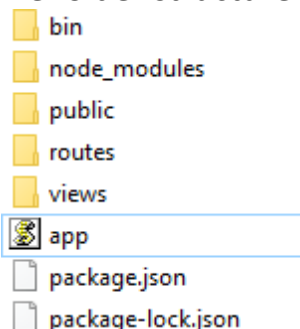
run the app:
> SET DEBUG=myfirstapp:* & npm start
```

### 3. Install node libraries:

```
C:\Users\harshita\Documents\express_generator_tryout>cd myfirstapp

C:\Users\harshita\Documents\express_generator_tryout\myfirstapp>npm install
npm WARN deprecated jade@1.11.0: Jade has been renamed to pug, please install the latest version of pug instead of jade
npm WARN deprecated transformers@2.1.0: Deprecated, use jstransformer
npm WARN deprecated constantinople@3.0.2: Please update to at least constantinople 3.1.1
npm notice created a lockfile as package-lock.json. You should commit this file.
added 99 packages from 139 contributors and audited 194 packages in 16.972s
found 4 vulnerabilities (3 low, 1 critical)
run `npm audit fix` to fix them, or `npm audit` for details
```

### 4. The folder structure will look like below given:



### 5. Start the node server using npm start command:

```
C:\Users\harshita\Documents\express_generator_tryout\myfirstapp>npm start

> myfirstapp@0.0.0 start C:\Users\harshita\Documents\express_generator_tryout\myfirstapp
> node ./bin/www

GET / 200 468.100 ms - 170
GET /stylesheets/style.css 200 12.170 ms - 111
GET /favicon.ico 404 30.109 ms - 1452
```

- Go to browser and write localhost:3000 in the address bar to execute the express code.

## Express.js Template Engine:

A template engine facilitates you to use static template files in your applications.

At runtime, it replaces variables in a template file with actual values, and transforms the template into an HTML file sent to the client.

Some popular template engines are:

Pug (formerly known as jade)	handlebars	haml-coffee
Mustache	hogan	ect
Dust	jazz	ejs
Atpl	jqtpl	haml
Eco	hbs	JUST

## How to use hbs :

Express handlebars provides us HTML kind of environment so it is easy to work with handlebars than jade engine.

- Uninstall jade

```
C:\Users\harshita\Documents\express_generator_tryout\myfirstapp>npm uninstall jade --save
removed 47 packages and audited 140 packages in 1.482s
found 0 vulnerabilities
```

- Install hbs

```
npm install hbs --save
```

- Make following changes in app.js file:

```
const hbs=require('hbs')
```

```
app.set('view engine', 'hbs');
```

4. Go to view folder change all jade files by hbs.
5. Remove all jade code and write hbs code.
6. To print the value which is coming from routes we use {{ }}.

### Views/layout.hbs

```
<html>
<head>
<title>Express App</title>
<link rel="stylesheet" href="/stylesheets/style.css">
</head>
<body>
| {{{body}}}
</body>
</html>
```

:

### Views/Index.hbs

```
<h1>{{title}}</h1>
Welcome {{title}}
```

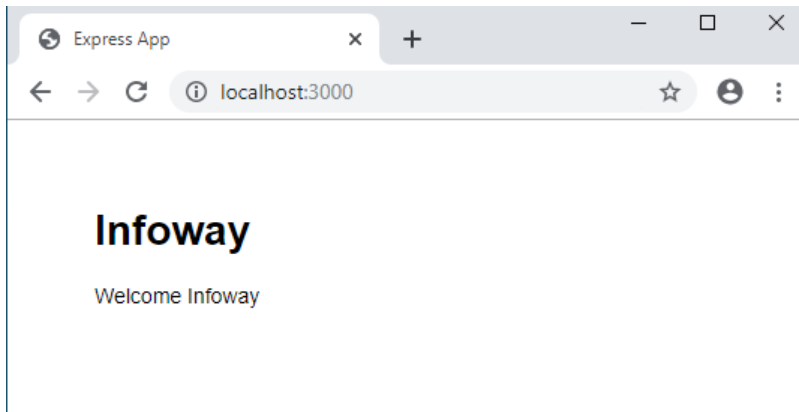
### Routes/index.js

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
|   res.render('index', { title: 'Infoway' });
});

module.exports = router;
```

7. Start the server using npm start and go to browser



## Express.js Request Object:

The express.js request object represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers, and so on.

### Syntax:

```
router.get('/', function (req, res) {
```

```
  // --
```

```
  })
```

## Express.js Request Object Properties:

**req.body:** It contains key-value pairs of data submitted in the request body. By default, it is undefined, and is populated when you use body-parsing middleware such as body-parser.

**req.ip:** It specifies the remote IP address of the request.

**req.params:** An object containing properties mapped to the named route ?parameters?. For example, if you have the route /user/:name, then the "name" property is available as req.params.name. This object defaults to {}.

**req.query:** An object containing a property for each query string parameter in the route.

## Express.js Response Object:

The Response object (res) specifies the HTTP response which is sent by an Express app when it gets an HTTP request.

### **Response End method:**

This method is used to end the response process.

Syntax: `res.end([data] [, encoding])`

### **Response Get method:**

This method provides HTTP response header specified by field.

Syntax: `res.get(field)`

### **Response JSON method:**

This method returns the response in JSON format.

Syntax: `res.json([body])`

### **Response Redirect method:**

This method redirects to the URL derived from the specified path, with specified HTTP status

Syntax: `res.redirect([status,] path)`

### **Response Render method:**

This method renders a view and sends the rendered HTML string to the client.

Syntax: `res.render(view [, locals] [, callback])`

### **Response Send method:**

This method is used to send HTTP response.

Syntax: `res.send([body])`

### **Response Status method:**

This method sets an HTTP status for the response.

Syntax: `res.status(code)`

## **Simple Express.js program with helpers:**

Let's demonstrate the example:

- print the value of variable
- check the condition

- print values of an array
  - Use hbs (handlebars) template engine.
- Write the following code in index.js file:

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Infoway', address: 'Pune', arr: [1,2,3], c:true });
});

module.exports = router;
```

Write the following code in view (index.hbs):

```
<h1>{{title}}</h1>
<h2>{{address}}</h2>
{{arr}}<br><br>

{{#if c}}
condition is true
{{address}}
{{else}}
condition is false
{{/if}}
<br><br>

{{#each arr}}
  {{@key}}: {{this}}
{{/each}}
<br><br>

{{#each arr as |key val|}}
  {{key}}: {{val}}
{{/each}}
```

{{# }} {{/}} – is the syntax of helper classes.

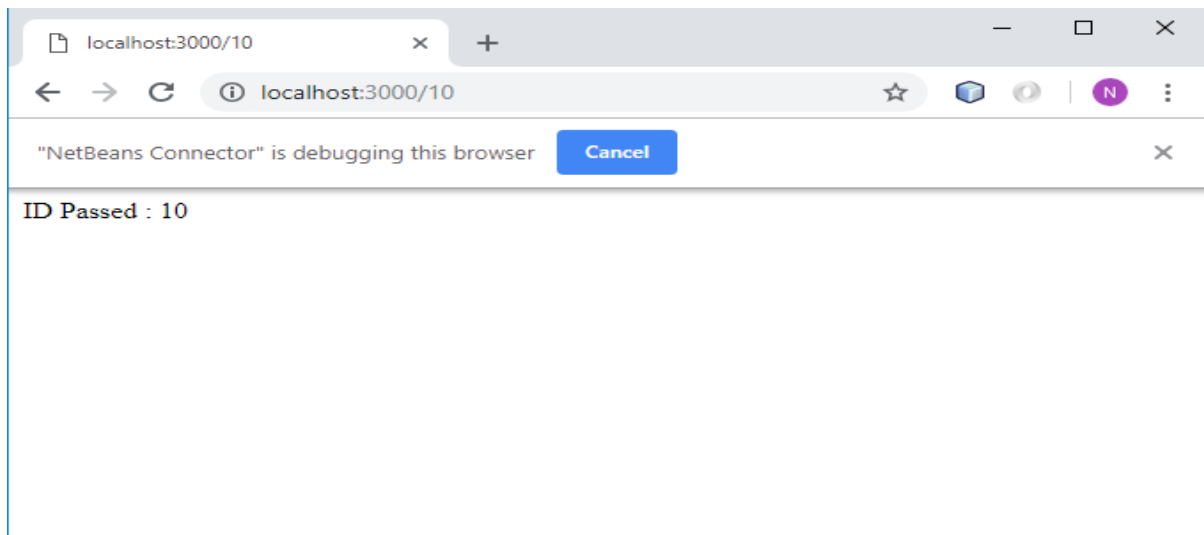
## Express.js GET Request:

Get method facilitates you to send only limited amount of data because data is sent in the header. It is not secure because data is visible in URL bar.

Let's take an example to demonstrate GET method:

```
var express = require('express');  
var router = express.Router();  
  
router.get('/:id', function(req, res, next) {  
  res.send("ID Passed : " + req.params.id);  
});  
module.exports = router;
```

To test this go to <http://localhost:3000/10>. The following response will be displayed.



One more example with 2 parameters:-

```
var express = require('express');  
var router = express.Router();  
  
router.get('/test/:id/:name', function(req, res) {  
  res.send('id: ' + req.params.id + ' and name: ' + req.params.name);  
});  
module.exports = router;
```

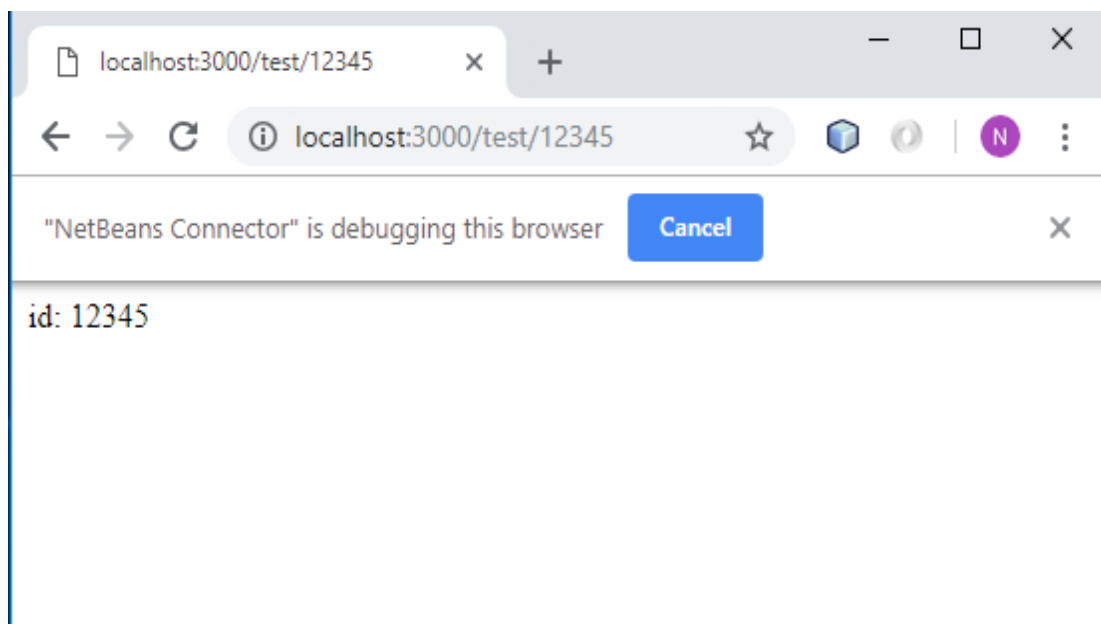




### Pattern Matched Routes:-

```
var express = require('express');  
var router = express.Router();  
  
router.get('/test/:id([0-9]{5})', function(req, res) {  
  res.send('id: ' + req.params.id );  
});  
module.exports = router;
```

This will only match the requests that have a 5-digit long id.



---

### Express.js POST Request:

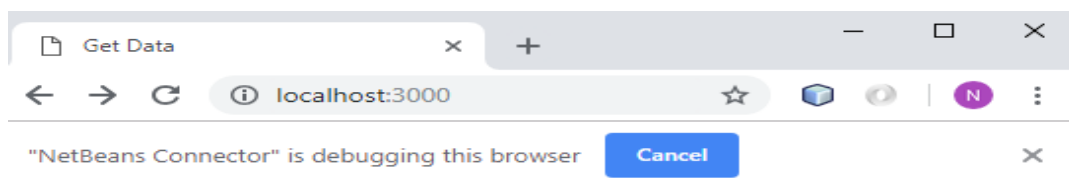
Post method facilitates you to send large amount of data because data is send in the body. Post method is secure because data is not visible in URL bar but it is not used as popularly as GET method. On the other hand GET method is more efficient and used more than POST.

Let's take an example to demonstrate POST method.

```
<h1>{{title}}</h1>
<form method="POST" action="/test/submit">
  ID:<input type="text" name="id"><br>
  Name:<input type="text" name="name"><br>
  <button>Submit</button>
</form>
<br>
<br>
{{id}} {{name}}
```

```
var express = require('express');
var router = express.Router();

router.get("/", function(req, res, next) {
  res.render('index', {title: 'Get Data'});
});
router.post('/test/submit', function(req, res, next) {
  res.send(req.body.id+" "+req.body.name);
  //res.render('index', {title: 'data', id: req.body.id, name: req.body.name });
});
module.exports = router;
```



## Get Data

ID:

Name:

