LEARN. DO. EARN

ACADGILD

NODE JS

Website : http://www.acadgild.com
LinkedIn : https://www.linkedin.com/company/acadgild
Facebook : https://www.facebook.com/acadgild

Session 10 – HTTP, MongoDB

Agenda – HTTP, MongoDB

Sl No	Agenda Title
1	Creating a Simple HTTP Server
2	Understanding Express JS
3	Express Core Component
4	Installing Express JS
5	Express Middleware
6	Kinds of Middleware
7	Application & Route Level Middleware
8	Error-Handling & Third-Party Middleware
9	Built-In Middleware
10	Jade
11	Creating a Basic Express.js App
12	Starting and Stopping the App

Creating a Simple HTTP Server

- **Node.js** provides extremely easy to use HTTP API.
- The HTTP module makes it simple to create an HTTP server via its simple but powerful API.
- **Example of Simple HTTP Server:**

```
var http = require('http');
var requestListener = function (req, res) {
                    res.writeHead(200);
                    res.end('Hello, World!\n');  }
var server = http.createServer(requestListener);
server.listen(8080);
```

- Save this in a file called **server.js** and open the terminal/node cmd and run the following command to execute this application.

```
node server.js
```

- Now open the **browser** and type http://localhost:8080 and see the output in browser.

Understanding Express JS

- **Express** is a minimal yet flexible and powerful web development framework for the **Node.js** (Node) platform that provides a robust set of features for web and mobile applications.
- The flexibility in **Express** comes from the use of middleware and Node modules**.**
- **Express** is a powerful framework because it gives you complete access to the core Node APIs.
- **Express** was inspired by **Ruby's Sinatra** and is built on top of **Node's web server API.**

Express Core Component

- **There are only three core components of Express which are given below:**
- **Application object –**
  - The application object is an instance of Express which is represented by the variable named **app**.
    - This is the main object of Express app and the bulk of the functionality is built on it.
- **Request object –**
  - The HTTP request object is created when a client makes a request to the Express app.
  - The object is represented by a variable named as **req**, which contains a number of properties and methods related to the current request.
- **Response object –**
  - The **response object** is created along with the **request object** and is represented by a variable named **res**.

Installing Express.js

- Once Node.js is installed on your system, we can install Express through **NPM (Node Package Manager)**.
- Express can be installed globally and locally on your system:
  - **If you want to installed Express globally** then run the following command:

```
npm install express -g
```

  - **If you want to install locally** then run the following command:

```
npm install express
```

Express Middleware

- **An Express application** is a series of middleware calls.
- **Middleware** is a function which has access to the request object (*req*), the response object (*res*) and the next middleware in line with the request-response cycle of an Express application. The Express application is commonly denoted by a variable named ***next***.
- **Following are the things that a Middleware can do:**
  1. It can execute any code.
  2. It can make changes to the request and the response objects.
  3. It can end the request-response cycle.
  4. It can call the next middleware in the stack.

Kinds of Middleware

An **Express** application can use the following kinds of middleware:

1. Application-level middleware
2. Router-level middleware
3. Error-handling middleware
4. Built-in middleware
5. Third-party middleware

Application & Route Level Middleware

- **Application-level middleware -** Application level middleware are bound to an instance of express, using *app.use()* and *app.VERB()*.
- **Router-level middleware -** Router level middleware work just like application level middleware except they are bound to an instance of *express.Router()*.

  var router = express.Router();
- Router level middleware are loaded using *router.use()* and *router.VERB()*.

Error-Handling & Third-Party Middleware

- **Error-handling middleware** are defined like other middleware, except with four arguments instead of three, specifically with **the signature (err, req, res, next)).**
- **Third-party middleware -** Express is a routing and middleware web framework with minimal functionality of its own.
- Functionality to Express apps are added via third-party middleware.
- Install the **node** module for the required functionality and load it in your app at the application level or at the router level.

Built-In Middleware

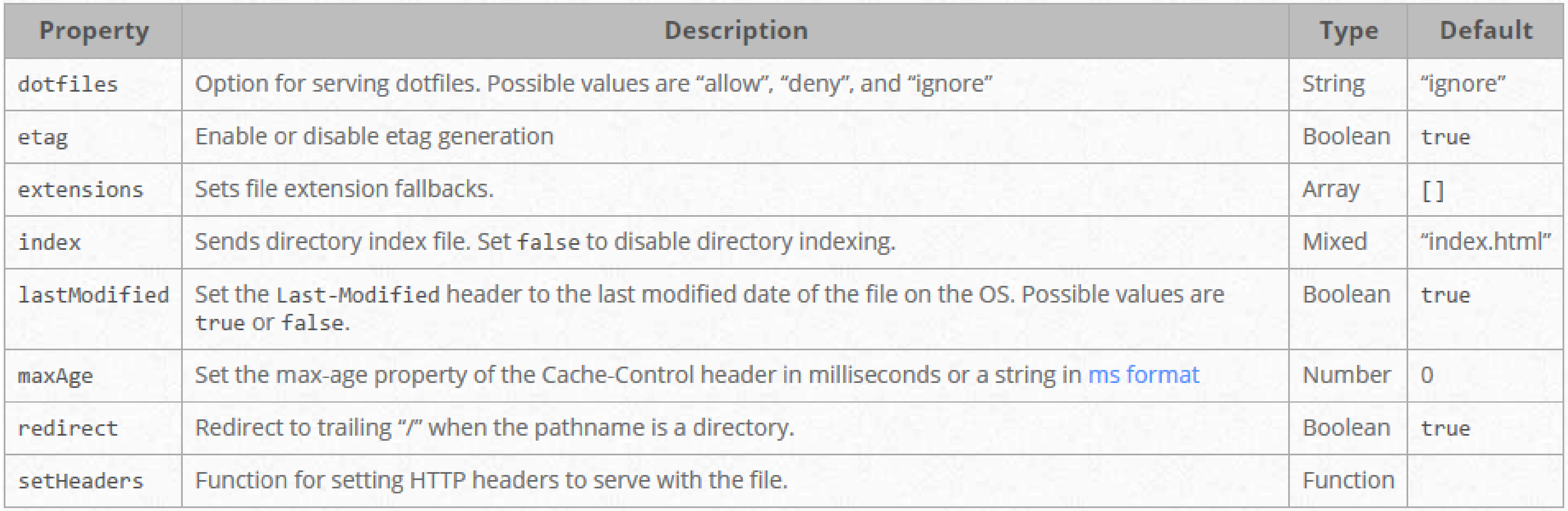
- **express.static** is the only built-in middleware in Express.
- It is based on serve-static and is responsible for serving the static assets of an Express application.
- The root argument refers to the root directory from which the static assets are to be served.

  express.static(root, [options])

Built-In Middleware (contd.)

- The optional options object can have the following properties.

  express.static(root, [options])



Property	Description	Type	Default
`dotfiles`	Option for serving dotfiles. Possible values are "allow", "deny", and "ignore"	String	"ignore"
`etag`	Enable or disable etag generation	Boolean	`true`
`extensions`	Sets file extension fallbacks.	Array	`[]`
`index`	Sends directory index file. Set `false` to disable directory indexing.	Mixed	"index.html"
`lastModified`	Set the `Last-Modified` header to the last modified date of the file on the OS. Possible values are `true` or `false`.	Boolean	`true`
`maxAge`	Set the max-age property of the Cache-Control header in milliseconds or a string in ms format	Number	0
`redirect`	Redirect to trailing "/" when the pathname is a directory.	Boolean	`true`
`setHeaders`	Function for setting HTTP headers to serve with the file.	Function	

Jade

- **Jade is a high performance template engine** heavily influenced by **Haml (**HTML Abstraction Markup Language**)**.
- It is implemented with JavaScript for **node** and **browsers.**
- Jade can be used as a shorthand for HTML.
- Jade is a language that complies to HTML.
- Jade is whitespace sensitive, so there's no need to close your tags. Jade does that for you.
- You can also nest tags within other tags just by indenting them.
  - **Example :- HTML :-**

```
<div>
  <address></address><i></i><strong></strong>
</div>
```

JADE :-

```
div
  address
  i
  strong
```

Creating a Basic Express.js App

- **Step 1 :** Create a directory named **express-app** in your home directory and build our app there :-

```
mkdir express-app
cd express-app
```

- **Step 2 : The Express manifest file :-** Create a file named **package.json** in the app directory. The package.json file can have more than a dozen fields.

```
{    "name": "test-app",
"version": "0.0.1",
"private": true,
"scripts": {  "start": "node app“ },
"dependencies": { "express": "3.2.6",  “jade": "*“  }
}
```

Executing the npm install command in the directory will install all the dependencies in the directory: **npm install**

Creating a Basic Express.js App (contd.)

- **Step 3** : **app.js :-** Create a file called app.js and put the following code in it :-

```
var http = require('http'); // Include the Node HTTP library
var express = require('express'); // Include the Express module
var app = express(); // Create an instance of Express
// Start the app
http.createServer(app).listen(3000, function() {
console.log('Express app started');
});
// A route for the home page
app.get('/', function(req, res) {
res.send('Welcome!');
});
```

Starting and Stopping the App

- Since Express apps are Node programs, starting an Express app is similar to executing a Node program.
- In our case, the program resides in a file named *app.js*, so this is how you can start the server :-

  node app

- => express app started here

  To stop the server, press Ctrl + C.

Lets Discuss Assignments

Assignment

Contact Info:

- Website : http://www.acadgild.com
- LinkedIn : https://www.linkedin.com/company/acadgild
- Facebook : https://www.facebook.com/acadgild
- Support: support@acadgild.com