`

503106 – Advanced Web Programming

Prepared by Dang Minh Thang

dangminhthang@tdtu.edu.vn

January 2020

Table of Content

I – Objectives

The objective of this lesson is to get yourselves familiar with NodeJS and ExpressJS. By the end of this lesson, you'll be able to:

- How to initialize a NodeJS and ExpressJS web project
- How to use views and layouts in ExpressJS
- How to use static middleware to fetch static files
- How to pass dynamic content into views

II – Initial Steps

- Start by creating a new directory.
- Open command line interface and run **npm init**

```
Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (temp) web
version: (1.0.0)
description:
entry point: (index.js) app.js
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\thang\Documents\Teaching\2020-2021-HK2\LTW-NangCao\Demo\Temp\package.json:

{
  "name": "web",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes) yes
```

- The next step will be installing Express. Run the following npm command: **npm install express**

```
C:\Users\thang\Documents\Teaching\2020-2021-HK2\LTW-NangCao\Demo\Temp>npm install express
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN web@1.0.0 No description
npm WARN web@1.0.0 No repository field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 50 packages in 2.203s
found 0 vulnerabilities
```

- Now create the **app.js** file:

```
const express = require('express')
const app = express()
const port = 3000
// home page
app.get('/', (req, res) => {
        res.type('text/plain')
        res.send('Web Travel');
})
// about page
app.get('/about', (req, res) => {
        res.type('text/plain')
        res.send('About Web Travel')
})
// custom 404 page
app.use((req, res) => {
        res.type('text/plain')
        res.status(404)
        res.send('404 - Not Found')
})
// custom 500 page
app.use((err, req, res, next) => {
        console.error(err.message)
        res.type('text/plain')
        res.status(500)
        res.send('500 - Server Error')
})
app.listen(port, () => console.log(`Express started on http://localhost:${port}; ` +
`press Ctrl-C to terminate. `))
```

- To start the web server, run **node app.js**
- Now visit **localhost:3000**, **localhost:3000/about**, etc…

III – Views and Layouts

- In this tutorial we will use handlebars as the view engine
- To provide Handlebars support, we'll use Eric Ferraiuolo's express-handlebars package: **npm install express-handlebars**
- Then in our **app.js** file, modify the first few lines:

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

```
const expressHandlebars = require('express-handlebars')
const app = express()
// configure Handlebars view engine
app.engine('handlebars', expressHandlebars({
        defaultLayout: 'main',
}))
app.set('view engine', 'handlebars')
```

- Now create a directory called **views** that has a subdirectory called **layouts**
- Let's create a layout for our site. Create a file called **views/layouts/main.handlebars**:

```
<!doctype html>
<html>
        <head>
                <title>Web Travel</title>
        </head>
        <body>
                {{{body}}}
        </body>
</html>
```

- Now let's create view pages for our Home page, **views/home.handlebars**:

```
<h1>Welcome to Home Page</h1>
```

- Then our About page, **views/about.handlebars:**

```
<h1>About Page</h1>
```

- Then our Not Found page, **views/404.handlebars**:

```
<h1>404 - Not Found</h1>
```

- And finally, our Server Error page, **views/500.handlebars**:

```
<h1>500 - Server Error</h1>
```

- Now that we have some views set up, we have to replace our old routes with new ones that use these views:

```
app.get('/', (req, res) => res.render('home'))
app.get('/about', (req, res) => res.render('about'))
// custom 404 page
app.use((req, res) => {
        res.status(404)
        res.render('404')
})
// custom 500 page
app.use((err, req, res, next) => {
        console.error(err.message)
        res.status(500)
```

```
    res.render('500')
})
```

IV – Static Files

- The **static** middleware allows you to designate one or more directories as containing static resources that are simply to be delivered to the client without any special handling
- This is where you would put things such as images, CSS files, and client-side JavaScript files.
- In your project directory, create a subdirectory called **public**
- Then, in **app.js** before you declare any routes, you'll add the static middleware:
  app.use(express.static(__dirname + '/public'))
- Let's create an **img** subdirectory inside **public** and put our **logo.png** file in there
- Now we can simply reference **/img/logo.png** (note, we **do not** specify **public**; that directory is invisible to the client), and the static middleware will serve that file.
  <img src="**/img/logo.png**" alt="Web Logo">

V – Dynamic Content in Views

- The real power of views is that they can contain dynamic information
- Let's say that on the About page, we want to deliver a message
- Modify the route /about to deliver the random fortune cookie:

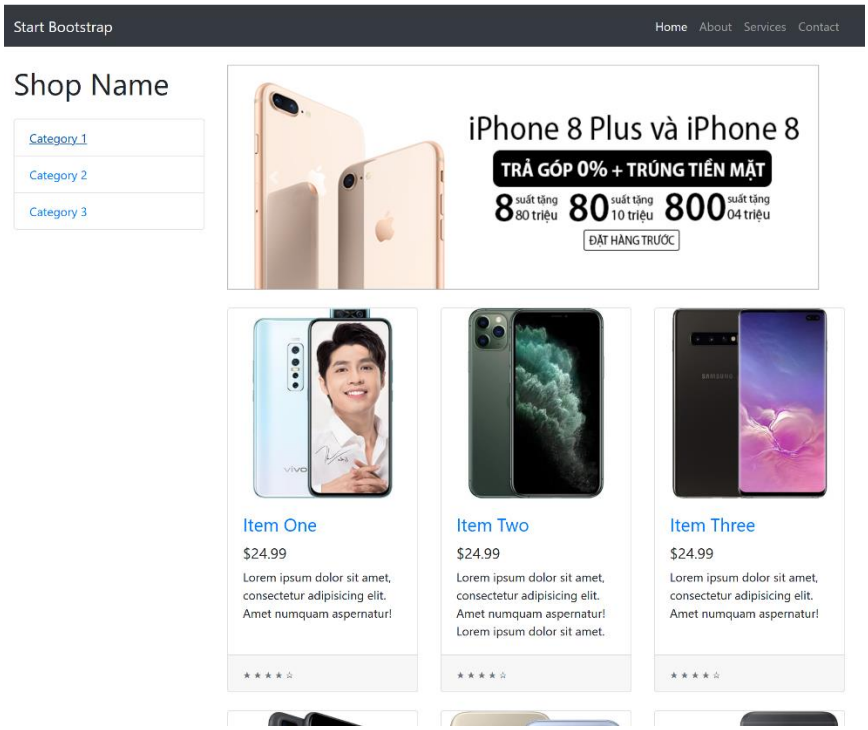
```
app.get('/about', (req, res) => {
    res.render('about', { message: 'Hello World' })
})
```

- Modify the view (views/about.handlebars) to display the message:

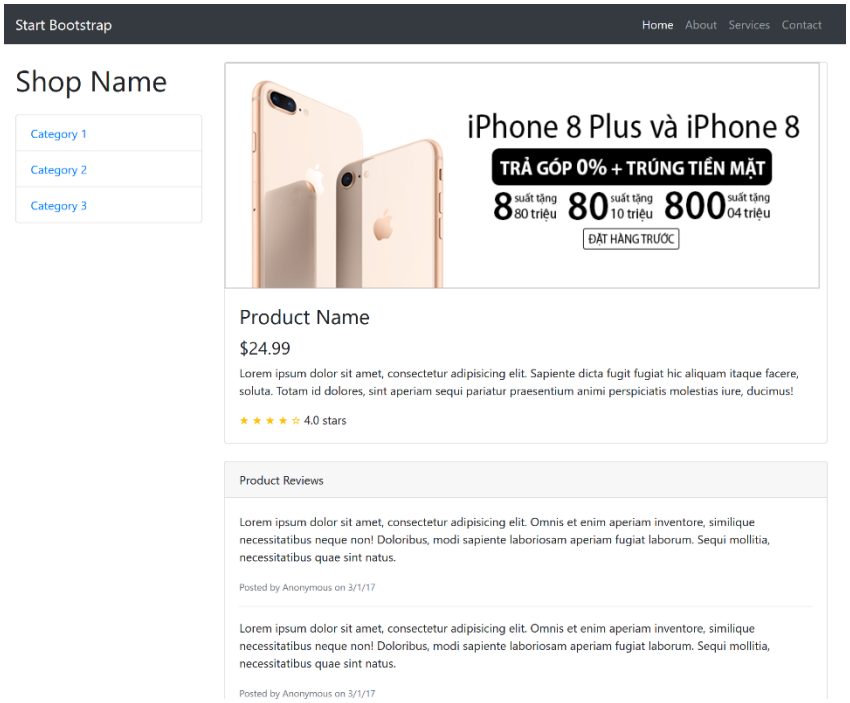
```
{{#if message}}
    <p>{{message}}</p>
{{/if}}
```

VI – Excercises

- In the attached **web** folder, you have two html pages:



Page 1.



Page 2.

- Create two NodeJS pages: **localhost:3000** which displays **Page 1** and **localhost:3000/detail** which displays **Page 2**.

- These two pages share the same layout, so create a default layout for them.
- Set static middleware, directories and paths correctly so that all images, css and javascript are loaded correctly.
- When user clicks on an item in Page 1, take him to Page 2.
- Replace the **Product Name** in Page 2 by a param passed from the app.js.