COMP2068 JavaScript Frameworks

Lesson 4
Express.js

Lesson Objectives

In this Lesson you will learn about:

1. Express.js middleware
2. The HBS view template engine
3. Cloud implementation for a Node.js app

Intro to Express.js

express

Building an Express Web Application

- The **Express framework** is a small set of common web application features, kept to a minimum in order to maintain the **Node.js** style.
- It is built on top of **Connect** and makes use of its middleware architecture.
- Its features extend **Connect** to allow a variety of common web applications' use cases, such as the inclusion of **modular HTML template engines**, extending the response object to support various data format outputs, a routing system, and much more.

Creating your first Express application

- After creating your **`package.json`** file and installing express as one of your dependencies, you can now create your first **Express** application by adding your already familiar **`server.js`** file with the following lines of code:

```
const express = require('express');
const app = express();
app.use('/', (req, res) => {
   res.send('Hello World');
});
app.listen(3000);
console.log('Server running at
http://localhost:3000/');
```

The application, request, and response objects

- **Express** presents three major objects that you'll frequently use.
  - The **application object** is the instance of an Express application you created in the first example and is usually used to configure your application.
  - The **request object** is a wrapper of Node’s HTTP request object and is used to extract information about the currently handled HTTP request.
  - The **response object** is a wrapper of Node's HTTP response object and is used to set the response data and headers.

The application, request, and response objects (cont’d)

The application object

- The application object contains the following methods to help you configure your application:
  - **`app.set(name, value)`**: This is used to set environment variables that Express will use in its configuration.
  - **`app.get(name):`** This is used to get environment variables that Express is using in its configuration.
  - **`app.engine(ext, callback):`** This is used to define a given template engine to render certain file types, for example, you can tell the **EJS template engine** to use HTML files as templates like this: **`app.engine('html’, require('ejs').renderFile).`**
  - **`app.locals:`** This is used to send application-level variables to all rendered templates.

The Express Generator

- Express offers a scaffolding tool (similar to an ASP.NET project template) called the **Express Generator**
- We'll use this tool moving forward to build an application shell
- To install the Express Generator globally:

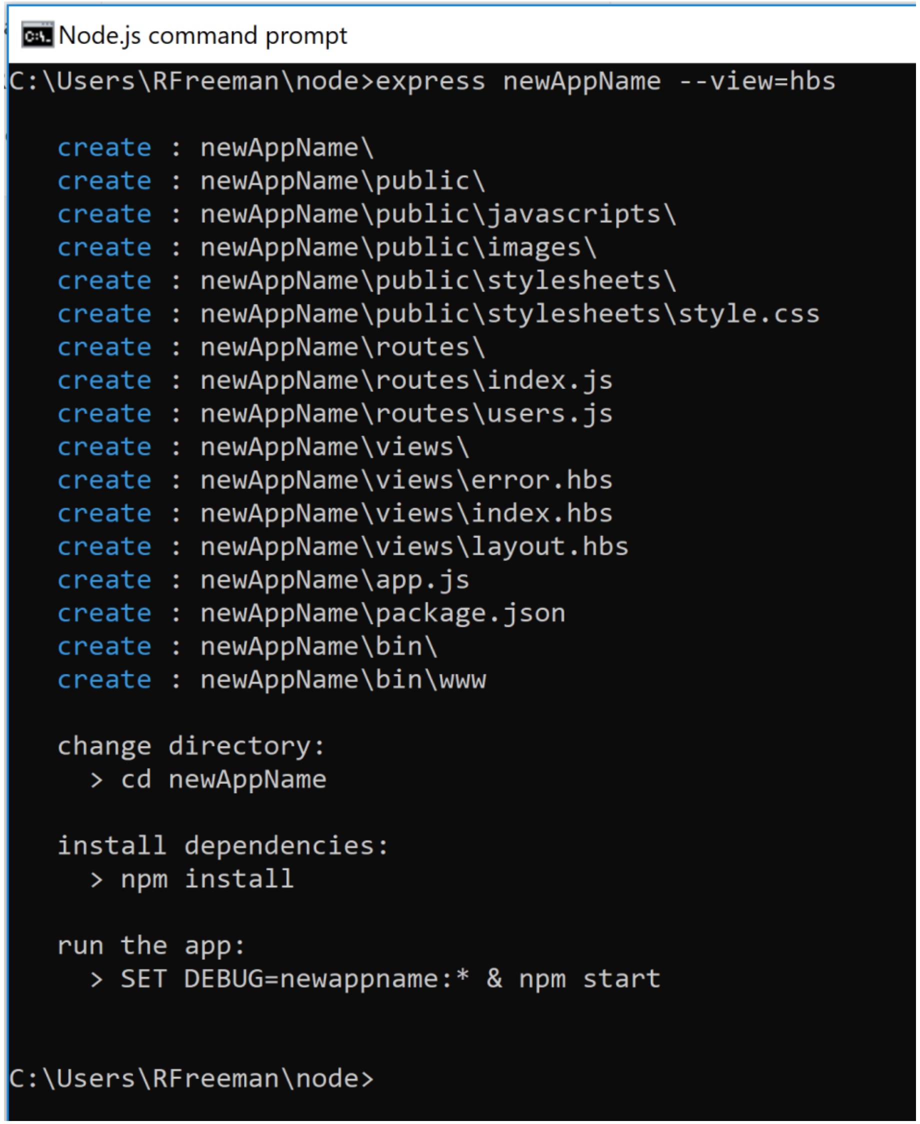
```
npm i –g express-generator
```

Scaffolding a new app using Express Generator

- By default, the Express Generator uses a View Template Engine called **Pug**, which is cool but very hard to debug
- When creating a new app with the Generator, we will change the View Engine to **handlebars** or **hbs** instead
- **hbs** makes debugging easier and also more closely replicates **Angular** view syntax, which we'll use later in the term
- To create a new Express app using hbs:

```
express newAppNameHere –view=hbs
cd newAppNameHere
npm i
```

Express Generator Results



Node.js command prompt

```
C:\Users\RFreeman\node>express newAppName --view=hbs

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

   change directory:
     > cd newAppName

   install dependencies:
     > npm install

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


C:\Users\RFreeman\node>
```