Current Skill Environment

Installation

Assuming you've already installed Node.js, you should create a directory to hold your application and make that your working directory.

```
$ mkdir myapp
$ cd myapp
```

Once we are inside the project folder, we can create a package.json manually in a text editor or with the npm init terminal command.

```
$ npm init
```

This command prompts you a number of things, such as the name and version of your application. For now, you can simply hit RETURN to accept the defaults for most of them, with the following exception:

```
entry point: (index.js)
```

Enter app.js, or whatever you want the name of the main file to be. If you want it to be index.js, hit RETURN to accept the suggested default file name.

Finally, we can install the module utilizing NPM

```
$ npm install express --save
```

Express application generator

Use the application generator tool, express-generator, to quickly create an application skeleton.

Install the application generator as a global npm package and then launch it.

```
$ npm install -g express-generator
$ express
```

For example, the following creates an Express app named myapp. The app will be created in a folder named myapp in the current working directory and the view engine will be set to Pug:

```
$ express --view=pug myapp
```

The generated app has the following directory structure:

```
.
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.pug
    ├── index.pug
    └── layout.pug

7 directories, 9 files
```

Express application generator

Then install dependencies:

```
$ cd myapp
$ npm install
```

run the app with this command:

```
$ npm start
```

Then load http://localhost:3000/ in your browser to access the app, you should see this common response

Command Line Options

This generator can also be configured using command flags:

```
$ express -h

  Usage: express [options] [dir]

  Options:

    -h,  --help          output usage information
         --version       output the version number
    -e,  --ejs           add ejs engine support
         --hbs           add handlebars engine support
         --pug           and pug engine support
    -H, --hogan          and hogan.js engine support
         --no-view       generate without view engine
    -v,  --view <engine> add view <engine> support (ejs|hbs|hjs|jade|pug|twig|vash) (
    -c, --css <engine>   add stylesheet <engine> support (less|stylus|compass|sass) (d
        --git            add .gitignore
    -f, --force          force on non-empty directory
```

Watching for File Changes

Node.js applications are stored into memory and if we make changes to the source code, we need to restart the process, i.e. node.

So to make our development process a lot easier, we will install a tool from npm, nodemon. This tool restarts our server as soon as we make a change in any of our files.

To install nodemon, use the following command:

```
$ npm install -g nodemon
```

You can now start working on Express.

‹ Previous next ›