Express Templating Engines and Scaffolding

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. So this approach is preferred to design HTML pages easily.

Following is a list of some popular template engines that work with Express.js:

- Pug (formerly known as jade)
- mustache
- dust
- atpl
- eco
- ect
- ejs
- haml
- haml-coffee
- handlebars
- hogan
- jazz
- jqtpl
- JUST
- liquor
- QEJS
- swig
- templayed
- toffee
- underscore
- walrus
- whiskers

In the above template engines, pug (formerly known as jade) and mustache seems to be most popular choice. Pug is similar to Haml which uses whitespace. According to the template-benchmark, pug is 2x slower than Handlebars, EJS, Underscore. ECT seems to be the fastest. Many programmers like mustache template engine mostly because it is one of the simplest and versatile template engines.

Using template engines with Express

Template engine makes you able to use static template files in your application. To render template files you have to set the following application setting properties:

- **Views:** It specifies a directory where the template files are located.

  For example: app.set('views', './views').

- **view engine:** It specifies the template engine that you use. For example, to use the Pug template engine: app.set('view engine', 'pug').

PUG:

Pug is a template engine for Node.js. Pug uses whitespaces and indentation as the part of the syntax. Its syntax is easy to learn.

Create a Pug template file named index.pug in the views directory, with the following content:

Index.pug:

```
doctype html
html
  head
    title A simple pug example
  body
    include ./header.pug
    h1 PES university information
    p
      |People's Education Society University Ring Road Campus
      | (PESU RR or just PESU), formerly PES Institute of Technology
      |(PESIT), is one of three private universities under the name
      |PES University in Bengaluru, India. Established in 1972,
      |it is focused on five main educational areas: Engineering,
      |Medicine, Management, Law and Life Sciences.
      |The institutions offer both foundation courses in these areas,
      |as well as specializations, with a Bachelors, Master or PhD
      |degree. It was upgraded to a university status in 2013.
    div.container#divid(width = "100",height="100")
      | hi this is a new div
    a(href = url) URL
```

```
    p my name is #{name}
    include ./footer.pug
```

header.pug:

```
div.header.
  this is the header for my app
```

footer.pug:

```
div.footer.
  this is the footer for my app
```

Then create a route to render the index.pug file. If the view engine property is not set, you must specify the extension of the view file. Otherwise, you can omit it.

Server.js:

```
var express = require('express');
var app = express();
app.set("view engine","pug")
app.set('views', './views')
//set view engine

app.get('/', function(req, res){
  res.render('index');
});
var server = app.listen(5000, function () {
   console.log('Node server is running..');
});
```

When you make a request to the home page, the index.pug file will be rendered as HTML.

Note: The view engine cache does not cache the contents of the template's output, only the underlying template itself. The view is still re-rendered with every request even when the cache is on.

Developing template engines for Express

Use the app.engine(ext, callback) method to create your own template engine. ext refers to the file extension, and callback is the template

enginefunction, which accepts the following items as parameters: the location of the file, the options object, and the callback function.

The following code is an example of implementing a very simple template engine for rendering .ntl files.

```
var express = require('express');
var app = express();
var fs = require('fs') // this engine requires the fs module
app.engine('ntl', function (filePath, options, callback) { // define the template en
gine
  fs.readFile(filePath, function (err, content) {
    if (err) return callback(err)
    // this is an extremely simple template engine
    var rendered = content.toString()
      .replace('#title#', '<title>' + options.title + '</title>')
      .replace('#message#', '<h1>' + options.message + '</h1>')
    return callback(null, rendered)
  })
})
app.set('views', './views') // specify the views directory
app.set('view engine', 'ntl') // register the template engine



app.get('/', function (req, res) {
    res.render('index', { title: 'Hey', message: 'Hello there!' })
  });

  var server = app.listen(5000, function () {
    console.log('Node server is running..');
});
```

When you make a request to the home page, index.ntl will be rendered as HTML.

Scaffolding:

Scaffolding allows us to easily create a skeleton for a web application. We manually create our public directory, add middleware, create separate route files, etc. A scaffolding tool sets up all these things for us so that we can directly get started with building our application.

Scaffolding is creating the skeleton structure of application. It allows users to create own public directories, routes, views etc. Once the structure for app is built, user can start building it.

Using **express-generator** package to install **'express'** command line tool. **express-generator** is used to create the structure of application.

Installing express-generator:
Steps:
1. Navigate to the folder where app is to be built out using the terminal.
2. Now in the terminal, install **express-generator** using the following command.

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

Basic structure of the application like Public directories, paths, routes, views, etc. are created which would form the structure of application.

It will create a new application, install all the dependencies, add few pages to the application(home page, 404 not found page, etc.) and gives a directory structure to work on.