informit.com

Introducing Express, The Web Framework for Node.js

by George Ornbo • original (http://www.informit.com/articles/article.aspx?p=1858263&seqNum=4)

Creating a basic Express site

Now you have installed Express you are ready to get a basic site up and running.

1. Open your terminal and generate a skeleton Express site by running the following command

```
express_express_example
```


()

Figure 6.1 () *Installing a basic site with Express*

2. Express politely reminds you to install the dependencies needed to run Express. So make sure you install the dependencies:

```
cd express_example && npm install
```

3. Start the application by running

```
node app.js
```

4. Open up your web browser of choice and browse to http://127.0.0.1:3000 (http://127.0.0.1:3000). You will see a basic

website served from Express.

Figure 6.2 () *Installing a basic site with Express*

If you have downloaded the code examples for this book this code is chapter06/example01.

Exploring Express

If you look in the folder for the example Express site that you just created you will see the following stucture.

- app.js
- node_modules
- package.json
- public
- routes
- views

app.js

This is the application file that is used to start the application. It contains configuration information for the application.

node_modules

This holds any node modules that are defined in package.json and have been installed.

package.json

This gives information on the application including the dependencies that should be installed in order for it to run.

public

This is the folder that serves the application to the web. You will find stylesheets, javascripts and images in this folder. You will not find any application logic in this folder; this is a common pattern to ensure web application security.

routes

In simple terms a route defines the pages that an application should respond to. For example if you want to have an 'about' page in your application you will need to set up an 'about' route. The routes folder holds these declarations.

views

The views folder defines the layouts for the application.

Introducing jade

Looking inside the views folder for the example project you will see a number of files with the '.jade' extension. Express makes use of template engines to compile views to HTML. By default Express uses Jade as the template engine.

Jade is an indentation based template engine. To understand this compare HTML and how it is represented in Jade:

Listing 6.1

HTML and Jade comparison

```
<div class="wrapper">
  <h1>My holiday in Prague</h1>
  <p>I had a great holiday in Prague where I met some gre
at people.</p>
  <img src="images/photo.jpg" alt="Me on holiday!" />
</div>
.wrapper
  h1 My holiday in Prague
  p I had a great holiday in Prague where I met some grea
t people
  img(src='images/photo.jpg' alt='Me on holiday')
```

You will notice several things here

- Jade is much more terse than HTML
- Jade uses indentation to declare the hierarchy of an HTML document
- You do not have to use tags in Jade. The <> characters are automatically added when the template is compiled.
- You do not have to close HTML tags in Jade. When Jade generates the HTML

When the page is generated Jade will compile the template to HTML. In listing 6.1 the output is exactly the same as vanilla HTML. So why bother with a template engine? The answer is that it allows applications to output data dynamically to HTML. Examples of where you may use a template engine include

- To display a list of blog posts stored in a database.

- To create a single template for displaying many different blog posts.
- To change the <title></title> element of a page based on a variable.
- To create header and footer includes that can be reused across templates.

Defining page structure with Jade

Jade defines page structure by indentation. If you are new to template languages this can take a little getting used to. This should become clear with examples though.

A tag (or HTML element) is defined by the leading word of an indentation.

```
html
```

This will compile to

```
<html></html>
```

You can use any HTML tag here (body, section, p etc).

To add a id to a tag append a pound sign and then the name of your id. Note spaces are not allowed.

```
section#wrapper
```

This will compile to

```
<section id=”wrapper”></section>
```

You can add a class to a tag by appending a dot followed by the name of your class.

```
p.highlight
```

Will compile to

```
<p class="highlight"></p>
```

If you need a class and an id Jade supports chaining

```
section#wrapper.class-name
```

This will compile to

```
<section id="wrapper" class="class-name"></section>
```

Jade also supports more than one class on a tag

```
p.first.section.third.fourth
```

This will compile to

```
<p class="first second third fourth">
```

To create HTML structure indentation is used.

```
p
  span
```

This will compile to

```
<p><span></span></p>
```

To add text within a tag simply add it after the tag declaration

```
h1 Very important heading
```

This will compile to

```
<h1>Very important heading<h1>
```

Jade also supports large bodies of text by using the pipe delineator

```
p
  | Text can be over
  | many lines
  | after a pipe symbol
```

This will compile to

```
<p>Text can be over many lines after a pipe symbol<p>
```

1. Open your terminal and generate a skeleton Express site by running the following command
   ```
   express_jade_structure
   ```
2. Start the application, change into the jade_structure folder then run:
   ```
   node app.js
   ```
3. Open up your web browser of choice and browse to http://127.0.0.1:3000. You will see a basic website served from Express.
4. Edit the file index.jade file found at views/index.jade. You will

see a bare bones structure for the page:

```
h1= title
p Welcome to #{title}
```

5. Add the following code to the page

```
section#wrapper h2 Basic Structure section p span This
is a span within a p within a div!
```

6. Stop and then restart the server and reload the page. Check the HTML on the page by viewing the source. You should see the following HTML.

```
<section id="wrapper">
  <section>
    <p>
      <span>This is a span within a p within a div!</
span>
    </p>
</section>
```

If you have downloaded the code examples for this book this code is chapter06/example02.

Outputting data with Jade

Whilst building page structure with Jade is good the real power of a template language comes from manipulating data and outputting it to HTML.

Jade uses two special characters to decide how it should interpret code. The first, the minus sign (-) is used to tell the Jade that the code that follows should be executed and the result of the execution

(if any) should be outputted. This is most commonly used with loops where you want to output an array of data to the page.

The second is the equals sign (=). This tells the interpreter that the code should be evaluated and then outputted.

This can be a little confusing at first but examples will make it clear.

Variables

In this example the code is executed and there is no output to be returned. Jade just sets the variable of foo to be bar.

```
- var foo = bar;
```

With the variable set it can be used later on

```
p I want to learn to use variables. Foo is #{foo}!
```

The special syntax of #{variable} tells Jade to replace the variable with the string value. This compiles to

```
<p>I want to learn how to use variables. Foo is bar!<p>
```

1. Open your terminal and generate a skeleton Express site by running the following command
   ```
   express_jade_variables
   ```
2. Start the application, change into the jade_output folder then run:
   ```
   node app.js
   ```
3. Open up your web browser of choice and browse to

http://127.0.0.1:3000 (http://127.0.0.1:3000). You will see a basic website served from Express.

4. Edit the file index.jade file found at views/index.jade and add the following code

```
- var name = “Your Name”
h1= Hello #{name}!
```

5. Stop and then restart the server and reload the page. Check the HTML on the page by viewing the source. You should see the following HTML.

```
<h1>Hello Your Name</h1>
```

If you have downloaded the code examples for this book this code is chapter06/example03.

Loops

Loops allow you to iterate over arrays and objects. If you are creating anything other than a basic brochure site you will find yourself using loops a lot. This is commonly known as iteration, meaning that you iterate over an array or object and do the same thing over and over again.

Perhaps because this is such a common pattern Jade dispenses with the minus sign. Admittedly it is confusing that you use a minus sign to specify a variable but not a loop.

```
- users = ['Sally', 'Joseph ', 'Michael', 'Sanjay']
each user in users
  p= user
```

This will compile to

```
<p>Sally</p>
<p>Joseph</p>
<p>Michael</p>
<p>Sanjay</p>
```

If you prefer to use the for keyword this can also be written as

```
-for user in users
  p= user
```

It is also possible to iterate over objects.

```
- obj = { first_name: 'George', surname 'Ornbo'}
each val, key in obj
  li #{key}: #{val}
```

This compiles to

```
<li>first_name: George</li>
<li>surname: Ornbo</li>
```

1. Open your terminal and generate a skeleton Express site by running the following command
   ```
   express_jade_loops
   ```
2. Start the application, change into the jade_loops folder then run:
   ```
   node app.js
   ```
3. Open up your web browser of choice and browse to http://127.0.0.1:3000. You will see a basic website served from Express.
4. Edit the file index.jade file found at views/index.jade and add

the following code

```
- beatles = ['John', 'Paul', 'Ringo', 'George']
ul
  each beatle in beatles
    li
      #{beatle}
```

5. Stop and then restart the server and reload the page. Check the HTML on the page by viewing the source. You should see the following HTML.

```
<ul>
  <li>John</li>
  <li>Paul</li>
  <li>Ringo</li>
  <li>George</li>
</ul>
```

If you have downloaded the code examples for this book this code is chapter06/example04.

Conditions

If you have had exposure to any programming language you will be familiar with conditions. In plain English we can describe conditional flow as 'If something is true do something otherwise do something else'. In Jade conditions look like this:

```
if (awake)
 p You are awake! Make coffee!
else
 p You are sleeping
```

Note that again you do not need the minus sign (-) in front of the if and else keywords.

1. Open your terminal and generate a skeleton Express site by running the following command
```
express_jade_conditions
```
2. Start the application, change into the jade_conditions folder then run:
```
node app.js
```
3. Open up your web browser of choice and browse to http://127.0.0.1:3000 (http://127.0.0.1:3000). You will see a basic website served from Express.
4. Edit the file index.jade file found at views/index.jade and add the following code
```
- raining = 1
if (raining)
 p It is raining. Take an umbrella!
else
 p No rain. Take the bike!
```
5. Stop and then restart the server and reload the page. Check the HTML on the page by viewing the source. You should see the following HTML.
```
<p>It is raining. Take an umbrella!</p>
```
6. Change the value of raining in views/index.jade to false
```
- raining = 0
```
7. Stop and then restart the server and reload the page. Check the HTML on the page by viewing the source. You should see the following HTML.
```
<p>No rain. Take the bike!</p>
```

If you have downloaded the code examples for this book this code is chapter06/example05.

Inline JavaScript

It is possible to execute inline JavaScript in Jade templates. To do this declare a script block and then add your JavaScript within it.

```
script
  alert('You can execute inline JavaScript through Jade')
```

1. Open your terminal and generate a skeleton Express site by running the following command
   ```
   express_jade_inline_javascript
   ```
2. Start the application, change into the jade_inline_javascript folder then run:
   ```
   node app.js
   ```
3. Open up your web browser of choice and browse to http://127.0.0.1:3000 (http://127.0.0.1:3000). You will see a basic website served from Express.
4. Edit the file index.jade file found at views/index.jade and add the following code
   ```
   script
     alert('Inline JavaScript in Jade')
   ```
5. Stop and then restart the server and reload the page. You should see a JavaScript alert.

If you have downloaded the code examples for this book this code is chapter06/example06.

Includes

Most websites have parts of the page that appear on every page of the site. Example of these include

Jade supports includes with the include keyword and then name of the template that you want to include.

```
html
  body
    include includes/header
```

This will include code from the views/includes/header.jade file.

1. Open your terminal and generate a skeleton Express site by running the following command
   ```
   express_jade_includes
   ```
2. Start the application, change into the jade_includes folder then run:
   ```
   node app.js
   ```
3. Open up your web browser of choice and browse to http://127.0.0.1:3000 (http://127.0.0.1:3000). You will see a basic website served from Express.
4. Create a new folder at views/includes
5. Within views/includes add a file called footer.jade
6. Add some content into the footer.jade file
   ```
   p This is my footer. Get me. I have a footer on my website.
   ```
7. Include your footer file in views/index.jade
   ```
   h1 Jade Includes Example
   ```

```
include includes/footer
```

8. Stop and then restart the server and reload the page. You should see your footer included on the page.

If you have downloaded the code examples for this book this code is chapter06/example07.

Mixins

Mixins are a feature of Jade that not many other template engines have. If you find yourself repeating the same blocks of code over and over again using mixins is a good way to keep code maintainable and clean. Think of mixins as includes for your code. An example of where you may wish to use a mixin is outputting data in a loop. To define a mixin use the mixin keyword:

```
mixin users(users)
  ul
    each user in users
      li= user
```

Once the mixin is defined you can use it and reuse it in your templates

```
- users = ['Krist', 'Kurt', 'Dave']
mixin users(users)
```

1. Open your terminal and generate a skeleton Express site by running the following command

```
express_jade_mixins
```

2. Start the application, change into the jade_mixins folder then

run:

```
node app.js
```

3. Open up your web browser of choice and browse to http://127.0.0.1:3000. You will see a basic website served from Express.
4. Edit the file index.jade file found at views/index.jade and add the following code

```
mixin users(users)
  ul
    each user in users
      li= user
- users = ['Tanya', 'Jose', 'Kim']’
mixin users(users)
- more_users = ['Mark', 'Elena', 'Dave', 'Pete', 'Kei
ron']
mixin users(more_users)
```

5. Stop and then restart the server and reload the page. You should see two lists of users, generated from a single mixin.

If you have downloaded the code examples for this book this code is chapter06/example07.

Summary

In this hour you have learned about Express, a web framework for Node.js. You learned about when you might choose to use a web framework and what it can offer you.

You learned about Jade, a template engine for Node.js and how to use it to output variables, loops, conditional statements, inline JavaScript, includes and mixins.

After this hour you are able to create basic sites with Node.js and Express and begin to display data within your applications.

Q&A

17. Should I always use Express with Node.js or just sometimes?

    1. Express is a web framework for Node.js so there are times when it is not appropriate. If your requirements are to work with views and showing data within a web browser Express is a good fit. If you are building a command line script with Node.js or a tiny web service Express is probably not the right fit. In all cases pick the best tool for ths job.

17. I'm not keen on indentation based template engines. Are there any others?

    1. Although Express uses Jade as the default template engine it is agnostic to the template engine that is used. Another popular choice for Express is the EJS (Embedded JavaScript Templates) Template Engine. This is not indentation based and is close to how ERB works in Ruby. New template engines are appearing all the time, many with support for Express. Check the Node.js wiki for a comprehensive list: https://github.com/joyent/node/wiki/modules#wiki-templating (https://github.com/joyent/node/wiki/modules#wiki-templating).

    1. Jeremy Ashkenas create a test to show the performance of different JavaScript template languages. If you are interested in the performance of template languages the post will give a

benchmark and performance comparisons.
http://jsperf.com/dom-vs-innerhtml-based-templating/
(http://jsperf.com/dom-vs-innerhtml-based-templating/)

17. What are the pros and cons of using a template engine like Jade?

1. Using a template engine allows you to develop more quickly and take advantage of in-built checks for your code. With indentation Jade is easy to read and hence maintain. In terms of cons Jades adds a layer of complexity by abstracting plain HTML that for very simple projects may be overkill.

Exercises

1. Create an Express site using the command line generator. Explore each of the folders and understand what they do. Refer to the notes in this chapter on folder structure.
2. From the terminal run 'express --help'. Note how it is possible to change how the generator works by passing in arguments when generating new Express sites. Try switching the template engine that is used when you generate an Express site.
3. Generate a new Express site and practice your understanding of using Jade by assigning a variable and then outputting it. Create an array of data and then use a loop to show the data. Finally create a header include file and use it in your template.

In this chapter, you'll learn what Express is and how you can use it, how to create a basic site with Express, and how to use Jade as a template engine within Express.

What You'll Learn in This Hour:

- What Express is and how you can use it
- How to create a basic site with Express
- How to use Jade as a template engine within Express

What is Express?

Express is a web framework for Node.js. Web applications share common patterns so using a framework is often a good idea. You will find that you can develop faster and write applications on top of stable, tested code.

Some other web frameworks that you may be familiar with are

- Ruby on Rails (Ruby)
- Sinatra (Ruby)
- Django (Python)
- Zend (PHP)
- CodeIgniter (PHP)

Why use Express?

Express is a lightweight framework, meaning it does not make too many assumptions but gives you enough to avoid re-inventing the wheel.

Some of the things you can do with Express include

- Simple brochure websites
- JSON based APIs

- Realtime web applications

Some reasons for using a framework like Express include

- It takes less time to create applications using a framework.
- Common patterns like routing and view layers are accounted for in a framework like Express, meaning you do not have to write code for this.
- A framework like Express is actively used, maintained and tested. The stability of the code can be assumed.

Frameworks like Express are not appropriate for everything though. If you are creating a command line script you would certainly not want to use something like Express.

Installing Express

You can install express via npm:

```
npm install –g express
```

Original URL:
http://www.informit.com/articles/article.aspx?p=1858263&seqNum=4