Express JS

Lesson 02 : Templating Engines

Capgemini

Lesson Objectives

Introduction
Jade Templating Engine
Working with Tags in Jade
Working with id and classes in Jade
Attributes and Nesting Tags in Jade
Using if & unless in Jade
Using for & each in Jade
Using case & mixins in Jade
Include and Extend in Jade
EJS Templating engine

Introduction

➢A template engine is a library or a framework that uses some rules/languages to interpret data and render views.

➢Templating solves the problem by allowing you to write in the target language, while at the same time providing the ability to insert dynamic data.

➢For those of you familiar with the model–view–controller concept, templates belong to the view.

➢In the Node world, we have many templating engines to choose. Here are some criteria to consider to choose a template engine :

- **Performance** : Templating engine to be as fast as possible
- **Client, server, or both** : Most of the templating engines are available on both the server and client sides which makes our coding get well acquainted with the template engine
- **Abstraction** : Abstracting the details of HTML away from you

➢Jade/PUG, ejs and handlebars are some of the popular templating engines.

Jade/Pug templating engine

➢Jade/Pug abstracts the details of HTML away from us. It uses whitespace and indentation as part of its language

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Jade Demo</title>
<script>
    if (foo) {
        bar(1 + 5)
    }
</script>
<body>
<h1>Jade</h1>
<div id="container">

<p>You are amazing</p>


<p>
  Jade is a terse and
  simple templating
  language with a
  strong focus on
  performance and
  powerful features.
</p>
</body>
</html>
```

JADE

```
doctype html
html(lang="en")
  head
    title= pageTitle
    script.
      if (foo) {
         bar(1 + 5)
      }
  body

    h1 Jade
    #container
      if youAreUsingJade
        p You are amazing
      else
        p Get on it!
      p.
        Jade is a terse and
        simple templating
        language with a
        strong focus on
        performance and
        powerful features.
```

Working with Tags in Jade

```
doctype html
html
    head
        title IGATE
    body
        div
            h1 IGATE
            p IGATE Corporate University.
        div
            footer &copy; IGATE 2015
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>IGATE</title>
  </head>
  <body>
    <div>
      <h1>IGATE</h1>
      <p>IGATE Corporate University.</p>
    </div>
    <div>
      <footer>© IGATE 2015</footer>
    </div>
  </body>
</html>
```

Working with id and classes in Jade

```
doctype html
html
    head
        title IGATE
    body
        #content
            .block
                input#foo.sty1.sty2
```

```
<!DOCTYPE html>
<html>
  <head>…</head>
  <body>
    <div id="content">
      <div class="block">
        <input id="foo" class="sty1 sty2">
      </div>
    </div>
  </body>
</html>
```

Attributes and Nesting Tags in Jade

```
doctype html
html
    head
        title IGATE
    body
        ul#departments
            li
                a(href="#training") Training
            li
                a(href="#HR") HR
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>IGATE</title>
  </head>
  <body>
    <ul id="departments">
      <li>
        <a href="#training">Training</a>
      </li>
      <li>
        <a href="#HR">HR</a>
      </li>
    </ul>
  </body>
</html>
```

Passing options to Jade template

```
doctype html
html(lang="en")
    head
        title #{title}
    body
        h1 #{name} welcoming you all to Jade!
        br
        br
        input(type="text", value="#{name}")
        // Single line comment
        //- Invisible comment
```

```
<!DOCTYPE html>
<html lang="en">
  <head>…</head>
  <body>
    <h1>Karthik welcoming you all to Jade!</h1>
    <br>
    <br>
    <input type="text" value="Karthik">
    <!-- Single line comment-->
  </body>
</html>
```

Options : `{"title":"IGATE","name":"Karthik"}`

Using if & unless in Jade

```
doctype html
html(lang="en")
    head
        title #{title}
    body
        if name == "Abishek"
            h1 Hello Abishek
        else
            h1 My name is #{name}
        unless condition
            p You have passed false in condition
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>IGATE</title>
  </head>
  <body>
    <h1>My name is Karthik</h1>
    <p>You have passed false in condition</p>
  </body>
</html>
```

Options : `{"title":"IGATE","name":"Karthik","condition":false}`

Using for & each in Jade/Pug

```
doctype html
html(lang="en")
    head
        title #{title}
    body
        ul
            for department in departments
                li= department
            else
                li No departments!
        br
        select
                each department, counter in departments
                    option(value=counter) Department #{department}
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>IGATE</title>
  </head>
  <body>
    <ul>
      <li>HR</li>
      <li>Training</li>
      <li>Admin</li>
    </ul>
    <br>
    <select>
      <option value="0">Department HR</option>
      <option value="1">Department Training</option>
      <option value="2">Department Admin</option>
    </select>
  </body>
</html>
```

Options: `{"title":"IGATE","departments":["HR","Training","Admin"]}`

Using case & mixins in jade/Pug

➢Mixins are functions that take parameters and produce some HTML.

- Syntax : *mixin name(param, param2,...)*
- To use the function *+name(data)*

```
doctype html
html(lang="en")
    head
        title #{title}
    body
        mixin printName(name)
            p Hi #{name}
        case name
            when "Karthik"
                +printName("Karthik")
            when "Ganesh"
                +printName("Ganesh")
            default
                +printName("Guest")
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>IGATE</title>
  </head>
  <body>
    <p>Hi Ganesh</p>
  </body>
</html>
```

Options : `{"title":"IGATE","name":"Ganesh"}`

Include & Extend in Jade/Pug

- include & extend are the ways to split logic into a separate file for the purpose of reusing it across multiple files
- *include* is a top-to-bottom approach. i.e. The file that includes is processed then the included file is processed.
- To include a Jade/pug template use *include /path/filename*
- *extend* is a bottom-to-top approach (opposite to include).
- It works with *extend filename* and *block blockname* statements.

Include & Extend in Jade

Header.pug

```
doctype html
html
 head
  title
 body
  block content
  br
  hr
  footer
   p copyright capgemini
```

Home.pug

.

```
extends layout

block content
 h1 #{mydata}
```

EJS Templating engine

- EJS simply embeds JavaScript into the templates with a few simple tags to define how the JavaScript needs to be interpreted.
- EJS combines data and a template to produce HTML.
- JavaScript code gets executed inside<% %> and any code placed inside the tag <%= %> gets added with HTML.

```
const express=require('express');
const path=require('path');
//home route
app.get('/home',function(req,res){
res.render('home',{
mydata:'This is Capgemini L&D'
});
});
```

Demo

express02