Intro to Express.js and View Engines

Router, Static Files, Middleware, Handlebars

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

sli.do

#js-back-end

Table of Contents

Introduction to Express.js

Introduction to Express.js

```
npm install express
```

```javascript
const express = require('express')
const app = express();

const port = 3000;

app.get('/', (req, res) => {

  res.status(200);

  res.send('Welcome to Express.js!');

})

app.listen(port, () => console.log(`Express running
on port: ${port}...`));
```

Create a new **instance** of the **application**

Router in Express.js

Router

- Routing has the following syntax

```
app.METHOD(PATH, HANDLER)
```

- Where

  - **app** is an **instance** of express

  - **METHOD** is an HTTP **request** method, in **lowercase**

  - **PATH** is a path on the **server**

  - **HANDLER** is the function **executed** when the route is **matched**

Route Methods

```javascript
// GET method route
app.get('/', (req, res) => {
  res.send('GET request to the homepage')
})
// POST method route
app.post('/', (req, res) => {
  res.send('POST request to the homepage')
})
// PUT method route
app.put('/', (req, res) => {
  res.send('PUT request to the homepage')
})
```

Route Methods

```javascript
// All methods route

app.all('/about', (req, res, next) => {

  console.log('Middleware execution..')

  next()    // The next handler to be called

}, (req, res) => {

  res.send('Show about page.')

})    // Shows the about page after middleware execution
```

Router Paths

- Paths can **contain** special characters:

```
app.get('*',(req, res) => {

    res.send('Matches everything')

})
```
Based on string patterns
```
app.get('/ab*cd', (req, res) => {

    res.send('abcd, abANYTHINGcd')

})
```
Based on regular expressions
```
app.get(/.*fly$/, (req, res) => {

    res.send('butterfly, dragonfly')

})
```

Extracting Parameters

- Paths can have **parameters**

```javascript
app.get('/users/:userId', (req, res) => {
    const paramsObj = req.params
    res.send(paramsObj) })
```

- You can also **validate** parameters with **regular expressions** (it is not recommended)

```javascript
app.get('/users/:userId(\\d+)', (req, res) => {
    const paramsObj = req.params
    res.send(paramsObj) })
```

Chainable Routes

- You can create **chainable** route handlers using '**app.route()**'

```
app.route('/home')     Better for ordering routes

  .get((req, res) => {

    res.send('GET home page') })

  .post((req, res) => {

    res.send('POST home page') })

  .all((req, res) => {

    res.send('Everything else')

  })     Always place 'all' as a final method
```

Router Responses

- Responses

  - res.**download** - **prompt** a file to be **downloaded**

    ```
    app.get('/pdf', (req, res) => {
        res.download('FULL PATH TO PDF') })
    ```

  - res.**end** - end the response **process**

  - res.**json** - send a **JSON** response

Router Responses

- res.**redirect** - redirect a request (to **another** page)

```
app.get('/about/old', (req, res) => {
    res.redirect('/about') })
```

- res.**sendFile** - send a **file** as an **octet**-stream

```
app.get('/file/:fileName', (req, res) => {
    const fileName = req.params.fileName
    res.sendFile("PATH TO FILE" + fileName) })
```

- res.**render** - render a **view template**

Modular Routers

- You can use **express.Router** for modular route handlers

  - **Mounted** on a route (e.g. '/about')

  - Can use middleware, specific **only** to that router

```
const express = require('express')

const router = express.Router()

router.use(/* add middleware */)

router.get(/* define route handlers */)

app.use('/about', router)
```

Middleware

Middleware

- **Function** that has **access** to
  - The **request** and **response** object
  - The **next** middleware in the application's **request-response cycle**
- Different **kinds** of middleware exist
  - Application, route, error

```
const app = express()
app.use((req, res, next) => {
  console.log('Time:', Date.now())
  next() })
```

Next handler to be called

Custom Middleware

- Middleware can be for **specific** path

```
app.use('/user/:userId', (req, res, next) => {
    const userId = req.params.userId
    // TODO: Check if user exists in db/session
    let userExists = true
    if (!userExists) { res.redirect('/login') }
    else { next() }
})
app.get('/user/:userId', (req, res) => {
    res.send('User home page!')
})
```

Custom Middleware

- Can be on Application-level:

```
app.use(function (req, res, next) {
    console.log('Time:', Date.now()) next()
});
```

- Can be used for error-handling

```
app.use(function (err, req, res, next) {
    console.error(err.stack)
    res.status(500).send('Something broke!')
});
```

Third-Party Middleware

```
app.set('view engine', 'pug');

app.set('views', __dirname + '/views');

app.use(cookieParser());

app.use(session({secret: 'magic unicorns'}));

app.use(passport.initialize());

app.use(passport.session());

app.use(express.static(config.rootPath +'/public'));
```

Static Files

Static Files

- Serving static files

```
app.use(express.static('public'))

app.use('/static', express.static('public'))

app.use('/static', express.static(__dirname + '/public'))
```

- And all **files** from the directory will be **public**

```
http://localhost:3000/images/kitten.jpg

http://localhost:3000/css/style.css

http://localhost:3000/js/app.js

http://localhost:3000/images/bg.png

http://localhost:3000/hello.html
```
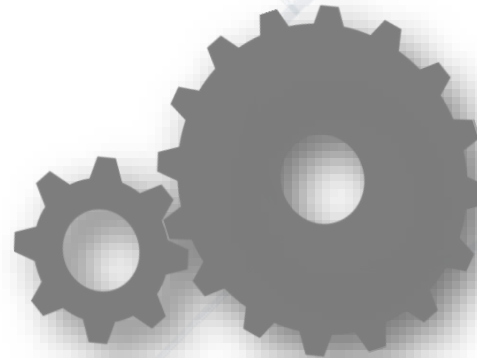
Templating Concepts

Templating

- Allows similar content to be **replicated** in a web page, **without repeating** the corresponding markup everywhere

HTML

<div>
 <span>
<button>

Dynamic

Ivan,
Maria,
Jordan

Ivan Ivanov   i

☎ 0888 123 456

✉ i.ivanov@gmail.com

Maria Petrova   i

☎ 0899 987 654

✉ mar4eto@abv.bg

**Jordan Kirov**   i

☎ 0988 456 789

✉ jordk@gmail.com

Templating Concepts

- The **static parts** of a webpage are stored as **templates**

- The **dynamic content** is kept separately (e.g. in a **database**)

- A **view engine** combines the two

- Benefits

  - **Productivity** - avoid writing the same markup over and over

  - **Easier upkeep** - only change the code in one place

  - **Composability** - a single element can be used on multiple pages

Examples

- Display articles in a blog

- Display a gallery of photos

- Visualize user profiles

- Show items in a catalog

Server View Engines
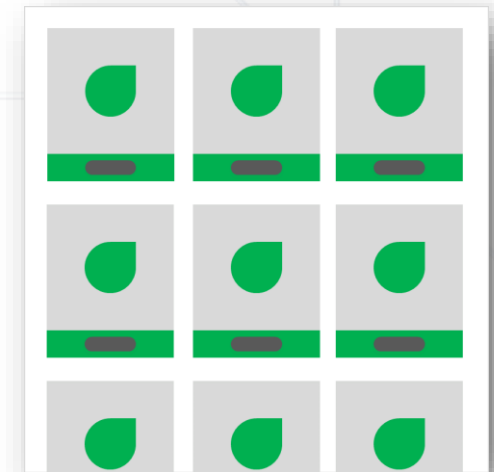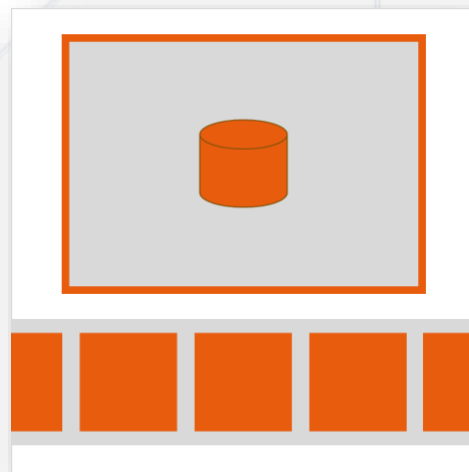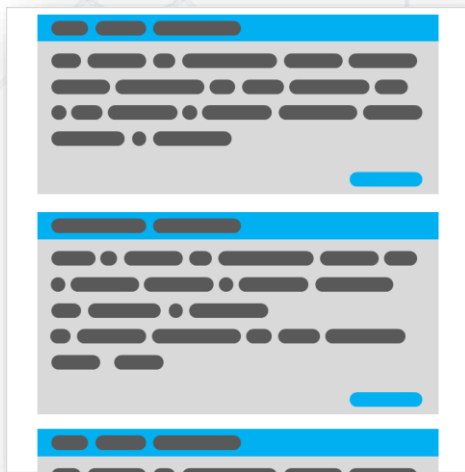
- Server view engines **return** ready-to-use **HTML** to the **client** (the browser)

  - They parse the **data** to **HTML** on the **server**

  - Web applications, created with **server** view engines are **not** real **SPA** apps (In **most** cases)

- Famous View Engines

  - Pug, Mustache, Handlebars, EJS, Vash
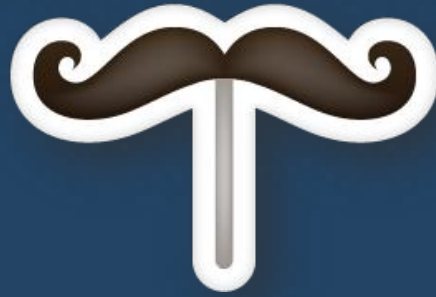
Templating with Handlebars

Handlebars

- Based on **Mustache** specification
- Expressions are **initialized** with '**{{**' and finish with '**}}**'

```
<div class="entry">
  <h1>{{title}}</h1>
  <div class="body">
    {{body}}
  </div>
</div>
```

```
<div class="entry">
  <h1>My New Post</h1>
  <div class="body">
    This is my first post!
  </div>
</div>
```

Integration in Express

```
npm install express-handlebars
```

```
const app = require('express')()

const exphbs= require('express-handlebars')

const handlebars = exphbs.create({ extname: '.hbs'});

app.engine('.hbs', handlebars.engine)

app.set('view engine', '.hbs')
```

```
app.get('/', function (req, res) {
    res.render('home');
})
```

Call the template

Handlebars directory structure

- Must have a **views** folder containing all the Handlebars templates:

```
        views
            layouts
                main.hbs
```

By Default

- The **layouts** folder inside the views folder will contain the layouts or the template wrappers

- The **main.hbs** file is the main layout

- *Note:* The default name of the folders can be changed with an appropriate setting

For-Loops

- A template can be **repeated** for every entry in an **array**

```
const context = {
  contacts: [
    { name: 'Maria Petrova', email: 'mar4eto@abv.bg'},
    { name: 'Jordan Kirov', email: 'jordk@gmail.com'} ]};
```

```
<ul id="contacts">
  {{#each contacts}}
    <li>{{name}}: {{email}}</li>
  {{/each}}
</ul>
```

The expression inside
the loop uses each
entry as context

Conditional Statements

```
{{#if sunny}}
  The sky is clear
{{else}}
  The sky is overcast
{{/if}}
```

Variable to check for truthiness

```
<ul id="contacts">
  {{#each contacts}}
    <li>{{name}}: {{email}}</li>
  {{else}}
    <i>(empty)<i>
  {{/each}}
</ul>
```

Will be shown if the array is empty

33

Partials

- **Templates** that can be **inserted into** other templates

```
<div id="contacts">
    {{#each contacts}}
        {{> contact}}
    {{else}}
        <i>(empty)<i>
    {{/each}}
</div>
```

HTML Escaping

- By default, any strings that are evaluated will be **HTML-escaped**

- To prevent this, use the "**triple-stash**"

```
title: "All about <p> Tags"

body: "<p>This is a post about &lt;p&gt; tags</p>"
```
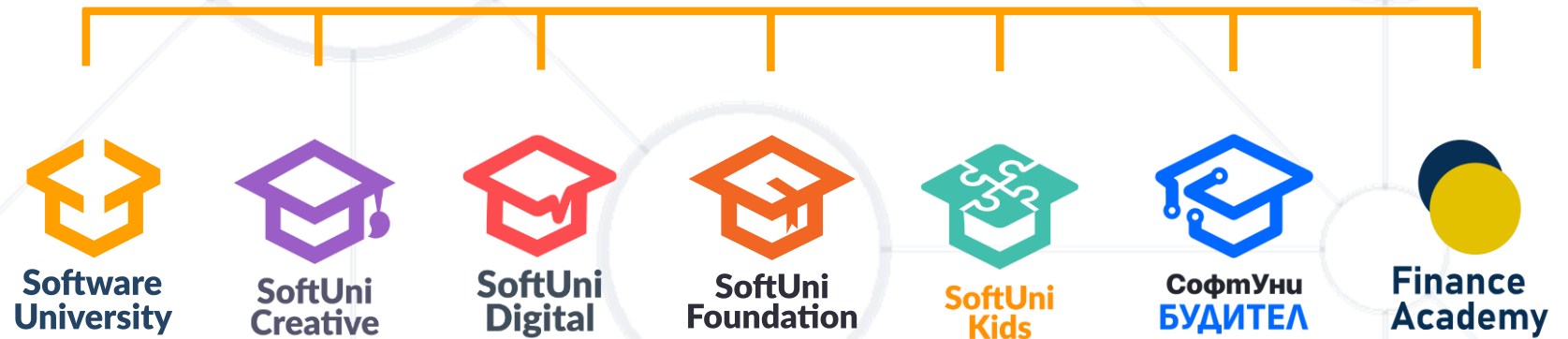
```
<h1>{{title}}</h1>
  <div class="body">
    {{{body}}}
</div>
```

```
<h1>All About &lt;p&gt; Tags</h1>
  <div class="body">
    <p>This is a post about &lt;p&gt; tags</p>
</div>
```

Summary

- **Express.js is a fast web framework for Node.js**

- **Middlewares can manipulate requests and responses**

- **Templates speed up and simplify development**

- **View Engines render templates**

- **Handlebars offers effective templates and simple helper functions**

Questions?

SoftUni Diamond Partners

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers

  - softuni.bg

- Software University Foundation

  - softuni.foundation

- Software University @ Facebook

  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**

- Unauthorized copy, reproduction or use is illegal

- © SoftUni – https://about.softuni.bg/

- © Software University – https://softuni.bg