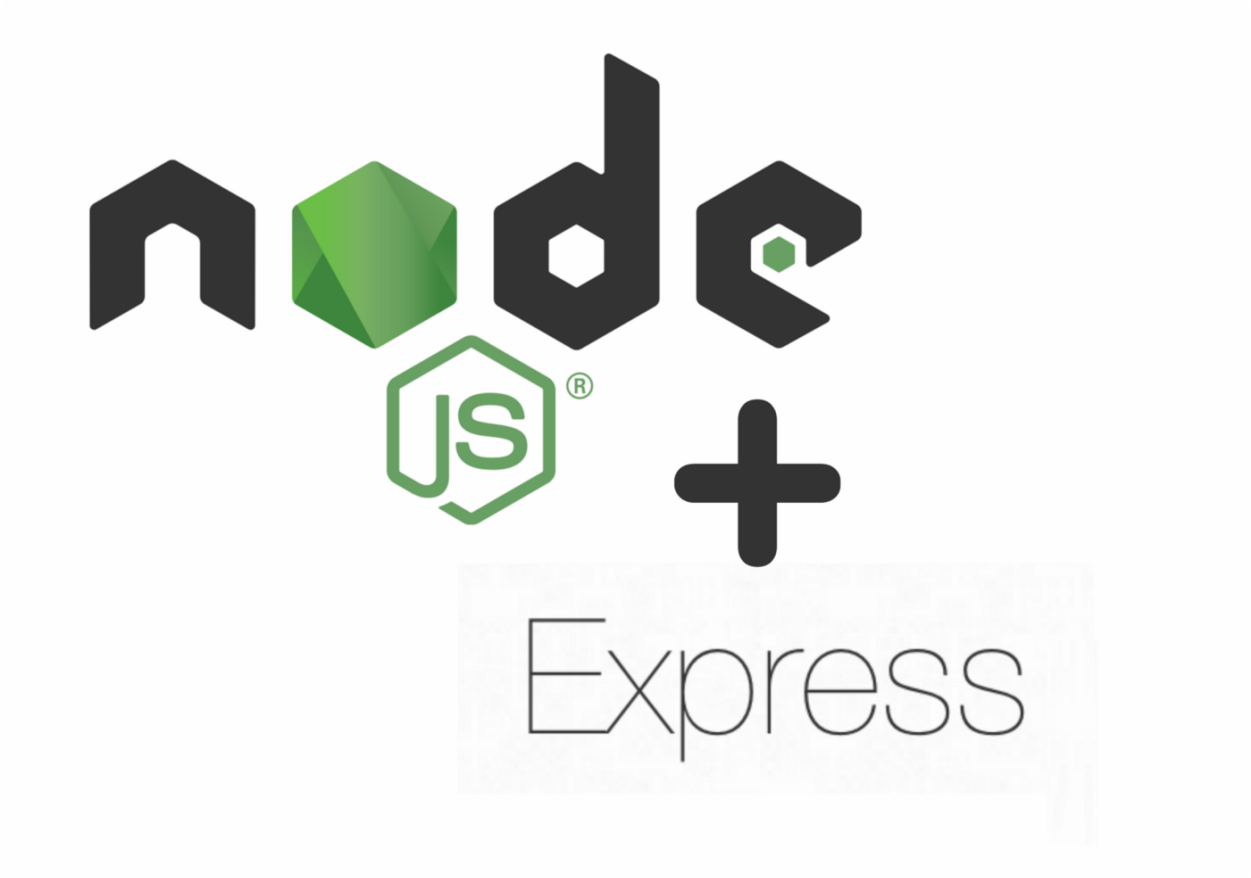
06 - Node & Express - PT



NODE.JS AND EXPRESS

Express.js is a web application server framework for Node.js, designed specifically for building single-page (SPA), multi-page, and hybrid web applications. It has become the default framework for Node.js.

Express is the backend part of something known as the MEAN stack. MEAN is a free and open source JavaScript software stack for building dynamic websites and web applications that encompasses the following components:

- MongoDB - A NoSQL database Express.js - A web
- application framework
- Angular.js - A JavaScript MVC framework used for web applications Node.js -
- Framework used for scalable server and network applications.

The Express.js framework makes it easy to develop an application that can be used to handle various types of requests, such as GET, PUT, POST, and DELETE.

INSTALL THE EXPRESS MODULE

Express is installed through the Node package manager. This can be done by running on the command line:

```
npm install express
```

The above command asks the Node package manager to download and install the required express modules.

Let's use our newly installed Express framework and create a simple "Hello World" application.

In our application we will create a simple server module that will listen on port 3000.

In this example, if a request is made through the browser on this port, then the server application will send a 'Hello' World' response to the client.

HELLO WORLD WITH EXPRESS

Create a folder named “express” and a new file named “expresshello.js”. Write the following code:

```
var express = require('express');
var app = express();
app.get('/', function (req, res) {
    res.send('Hello World!');
});
var server = app.listen(3000, function () { });
```

CODE EXPLANATION

```
var express = require('express');
var app = express();
app.get('/', function (req, res) {
    res.send('Hello World!');
});
var server = app.listen(3000, function () { });
```

In the first line of code, we use the require function to include the “express” module.


```
var express = require('express');
var app = express();
app.get('/', function (req, res) {
    res.send('Hello World!');
});
var server = app.listen(3000, function () { });
```


Before we can start using the express module, we need to create an object. Next, we create a “callback” function. This function will be called whenever someone navigates to the root of our web application located at http://localhost:3000 .


```
var express = require('express');
var app = express();
app.get('/', function (req, res) {
    res.send('Hello World!');
});
var server = app.listen(3000, function () { });
```


The callback function will be used to send the string 'Hello World' to the web page.

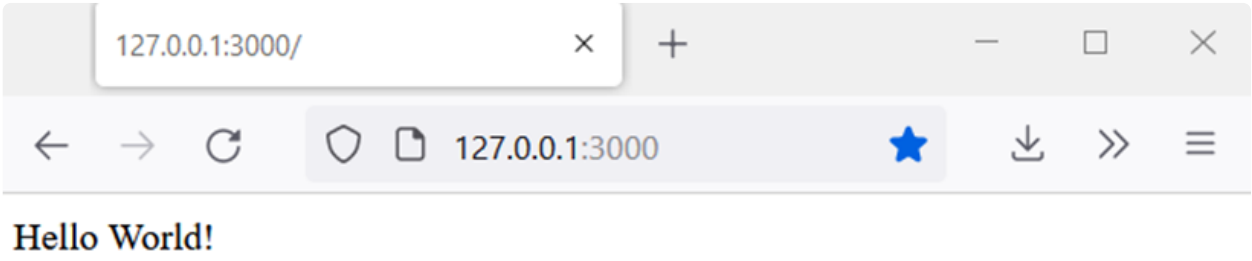

```
var express = require('express');
var app = express();
app.get('/', function (req, res) {
    res.send('Hello World!');
});
var server = app.listen(3000, function () { });
```


In the callback function, we send the string “Hello World” to the client. The 'res' parameter is used to send content to the web page. This 'res' parameter is provided by the 'request' module which allows sending content to the web page.

```
var express = require('express');
var app = express();
app.get('/', function (req, res) {
    res.send('Hello World!');
});
var server = app.listen(3000, function () { });
```

Using the listen function to make our application listen for client requests on port 3000. You can specify any available port.

If the command is executed successfully, the output will be displayed in your browser.



WHAT ARE ROUTES?

Routing determines how an application responds to a client request to a specific endpoint.

For example, a client might make a GET, POST, PUT, or DELETE request to multiple URLs, like the examples below:

- http://localhost:3000/students
- http://localhost:3000/teachers

In the examples above, if a GET request is made to the first URL, the response should be a list of students.

If the GET request is made to the second URL, the response should ideally be a list of teachers.

Therefore, based on the URL that is accessed, a different functionality on the web server will be invoked and consequently the response will be sent to the client.

This is the concept of routing. Each route can have one or more functions, which are executed when the route is matched.

The general syntax for a route is shown below:

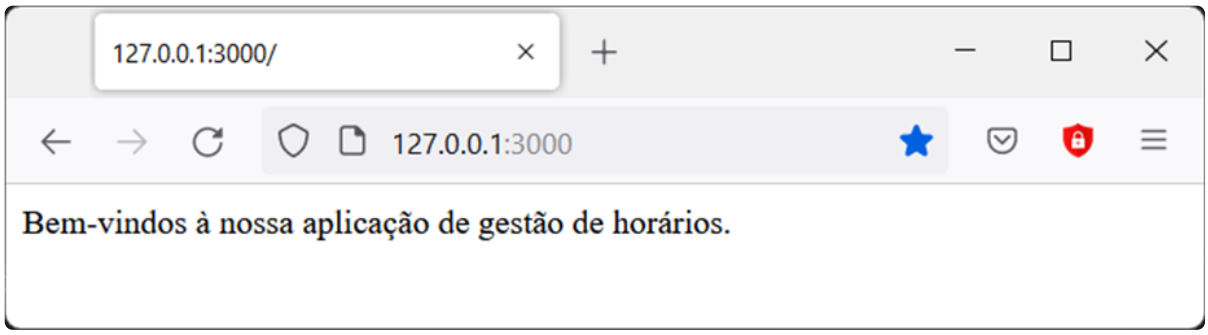
```
app.METHOD(PATH, HANDLER)
```

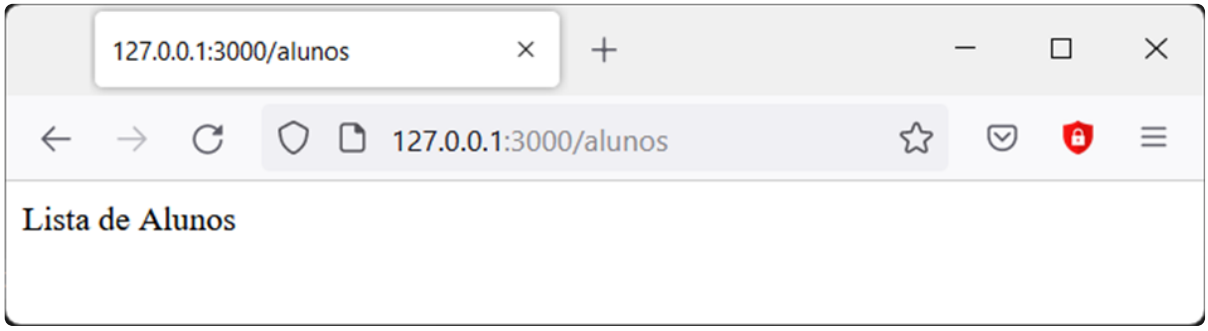
IMPLEMENTING ROUTES

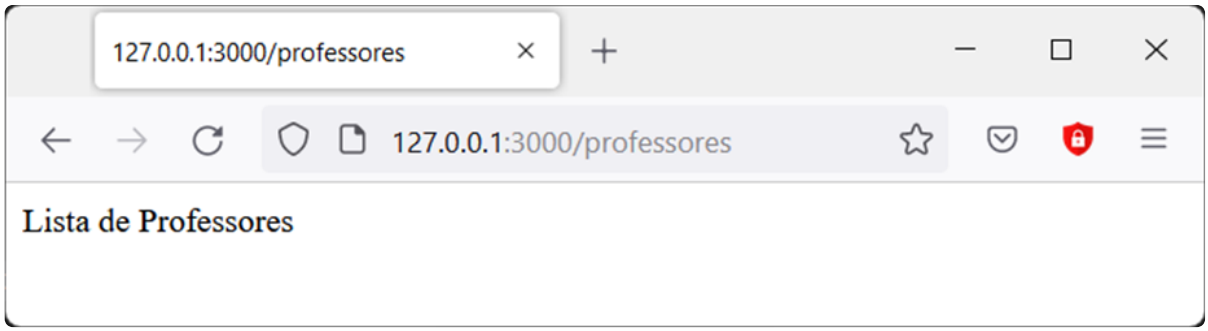
Create a file inside the project folder called index.js and insert the following code:

```
var express = require('express');
var app = express();
app.route('/alunos').get(function (req, res) {
    res.send("Lista de Alunos");
});
app.route('/professores').get(function (req, res) {
    res.send("Lista de Professores");
});
app.get('/', function (req, res) {
    res.send('Bem-vindos à nossa aplicação de gestão de horários.');
});
var server = app.listen(3000, function () { });
```

RESULT OF APPLICATIONS







EXAMPLE OF WEB SERVER WITH EXPRESS.JS

In the above example, we saw how we can decide which output to show based on the chosen route. This type of routing is used in most modern web applications.

Another important component is templates in NodeJs. When creating fast and dynamic Node applications, a quick and easy way is to use templates.

There are several frameworks available on the market for creating templates. In our case, we will use the Jade framework.

Jade is installed through the Node package manager. This can be done by running the following command line:

```
npm install jade
```

The above command asks the Node package manager to download the required jade modules and then install them.

TEMPLATING WITH JADE

Create a folder named “views” inside your project folder, and inside that folder create a new file named index.jade. Insert the code below:

```
html
    head
        title=title
        body
            h1=message
```

CREATE THE SERVER

Now let's create our server. In the project folder, create a new file named “indexjade.js”. Insert the following code:

```
var express = require('express');
var app = express();
app.set('view engine', 'jade');
app.get('/', function (req, res) {
    res.render('index',
        { title: 'IslaGaia', message: 'Bem-vindos' })
});
var server = app.listen(3000, function () { });
```

CODE EXPLANATION

```
var express = require('express');
var app = express();
app.set('view engine', 'jade');
app.get('/', function (req, res) {
    res.render('index',
        { title: 'IslaGaia', message: 'Bem-vindos' })
});
var server = app.listen(3000, function () { });
```

The first thing to specify in the application is the “view engine” that will be used to render the templates.

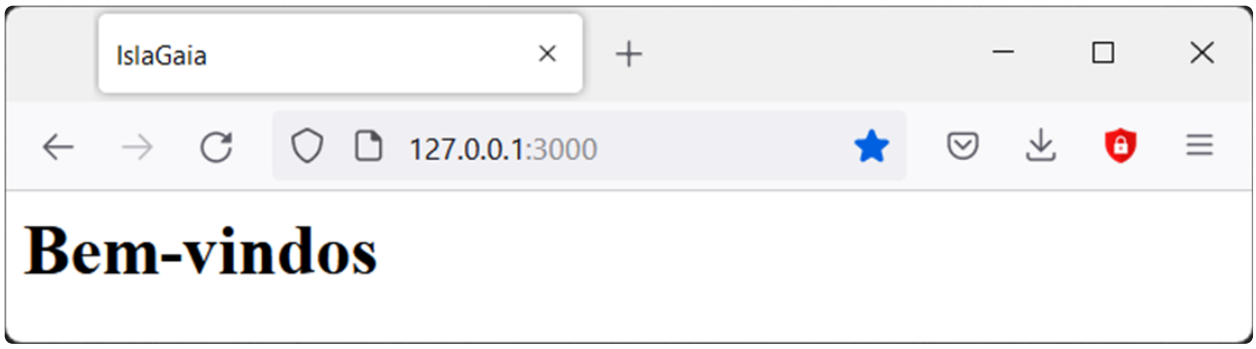
```
var express = require('express');
var app = express();
app.set('view engine', 'jade');
app.get('/', function (req, res) {
    res.render('index',
        { title: 'IslaGaia', message: 'Bem-vindos' })
});
var server = app.listen(3000, function () { });
```


The render function is used to render a web page. In our example, we are rendering the template (index.jade) that was created earlier.

```
var express = require('express');
var app = express();
app.set('view engine', 'jade');
app.get('/', function (req, res) {
    res.render('index',
        { title: 'IslaGaia', message: 'Bem-vindos' })
});
var server = app.listen(3000, function () { });
```


Then we pass the values of “IslaGaia” and “Welcome” to the parameters “title” and “message” respectively.

These values will be replaced by the 'title' and 'message' parameters declared in the index.jade template and the page is generated.



SUMMARY

- The express framework is the most common framework for developing Nodejs applications.
- This framework is built on top of the node.js framework itself and helps in rapid development of server-based applications.
- Routes are used to direct users to different parts of web applications based on the request made.
- The response for each route may vary depending on what needs to be shown to the user.
- Templates can be used to inject content efficiently. Jade is one of the most popular templating engines for Node.js.

CHALLENGE

Based on what you have learned so far, create an application with nodeJs, mySql and Jade that queries the student and teacher tables and displays their respective records.