C1- Code & Go

Mean Pool

Express

Day 02

CODING ACADEMY
BY EPITECH

Express

Foreword

This day is devoted to the Express framework. You must use it in each exercise.
Heavily advised resource ===> http://expressjs.com/ && http://expressjs.com/en/4x/api.html

PLEASE MAKE SURE TO READ INSTALL_EXPRESS.TXT CAREFULLY

You will create a repository named **MEAN_Pool_Day_02/**

Each Exercise will have its own directory.

E.g.: Exercise 01 sources will be in *ex_01*

Cheers ;)

Exercise 01

1 pt

File to hand in: ex_01.js, app.js

Restriction: Use Express

Express Foundations

As you may have seen on the first day, configuring a basic server in Node requires a fair amount of code and is not easy to use and maintain. Plus, it's quite easy to do it in the wrong way.

Express comes in and makes things a lot easier.

Use Express functionalities to implement a **start** method in an **app** module. This method will take a **port** as parameter and will launch an Express application listening on that port.

You will also make sure that the home of your site displays "Greetings Traveler!" in plain text.

ex_01.js is only to be used to launch your app.

Exercise 02

2 pts

File to hand in: ex_02.js, app.js, index.html, image.html, form.html

Restriction: Use Express, Path module

Express basic serving

Now add the *index.html*, *image.html* and *form.html* pages to your *ex_02*. Use Express functionalities to implement a **start** method in a **app** module. This method will take a **port** as parameter and will launch an Express application listening on that port.

Use Express methods in a way that:

- *index.html* file must be accessible from paths / and **/index**
- *image.html* file from **/image** path.
- *form.html* file from **/form** path.

If a user tries to access a page that does not exist, you will return a 404 error with the following message "Error 404: Page not found." in plain text.

Tip: No need to improvise something for non-existing files. Express already has an easy to use functionality for this.

ex_02.js is only to be used to launch your app.

Exercise 03

2 pts

File to hand in: ex_03.js, app.js, index.html, image.html, form.html

Restriction: Use Express, FS, & Path module

Work on your path

Copy your *app.js*, *index.html*, *image.html* and *form.html* from the previous exercise.

Add another page to your express application, accessible from address **localhost:port/student/X**.

Your page will display in simple text "Greetings Student Number X!".
If no number is given or if X is not a number, you will display a page error 404.

Example: accessing **localhost:port/student/50** will display "Greetings Student Number 50!".

ex_03.js is only to be used to launch your app.

Exercise 04

3 pts

File to hand in: ex_04.js, app.js, index.html, image.html, form.html, student.ejs

Restriction: Use Express, FS & Path module

Templating, because you know that's how we roll

Copy your *app.js*, *index.html*, *image.html* and *form.html* from the previous exercise.

Up until now you displayed your pages by spitting HTML at the browser! Spitting is rude! Really that's not a way of behaving yourself. You are better than that, we want you to dynamically generate HTML.

Thankfully, Express has your back covered, and a number of different view engines are available.

Install "EJS" engine by using npm.

A request to "**/student/X**" will now display the page *student.ejs* that implements the following example:

```
       => localhost:port/student/5?name=Martin
-------------------------------------------------------------------------------------
<!DOCTYPE html>
<html>
       <head>
              <meta charset="utf-8" />
              <title>Student</title>
       </head>
       <body>
              <p>Greetings, Martin, student number 5!</p>
       </body>
</html>
```

Note: If the parameter "name" is not entered, it must not be displayed.

You will also ensure that an access to path "/" or "**/index**" still displays the *index.html* page.

ex_04.js is only to be used to launch your app.

Exercise 05

2 pts

File to hand in: ex_05.js, app.js, index.html, image.html, form.html, student.ejs

Restriction: Use Express, FS, Path and cookie-parser

Middleware not middle-earth

Copy your *app.js*, *index.html*, *image.html* and *form.html* from the previous exercise.

Let's add a new path to the express application:

/memory that renders a page displaying last access data to page **/student/X**.

- An access to **/student/999?name=rico** and then **/memory** will display "rico, student number 999 was here" in plain text.
- An access to **/student/999** then to **/memory** display "student number 999 was here".
- A direct access to **/memory** displays nothing (you don't have to remove the cookie).

In order to save these parameters even if the user closes his browser, you will save them in a cookie named "name" and a cookie named "number".

ex_05.js is only to be used to launch your app

Exercise 06

3 pts

File to hand in: ex_06.js, app.js, index.ejs, show.ejs

Restriction: Use Express, FS, Path, body-parser and express-helpers

Templating is the name of the game

Use Express to implement a **start** method in an **app** module. This method will take a **port** as parameter, and will launch an Express application listening on that port.

Your express app will serve two pages.

The first will be the index page, accessible from path / and **/index**.
The index will contain a **POST** form sending to the **/show** page.

The form will contain the three following elements:

- A field text with a name **Name**
- A selection field with a **Gender** label with the following choice:
  - ''Male'', with value ''Male'', the default choice
  - ''Female'', that will have value ''Female''
- A **submit** button to send the information

Your *index.ejs* page must not contain input or select tags. Use express-helpers

Your *show.ejs* page will display the received information in plain text like this:

- >''[name] is a [gender].''.

If the page **/show** is not being accessed by a Post request or if Name or Gender are not provided, it will redirect the user on the index page with a redirection 302.

ex_06.js is only to be used to launch your app

Exercise 07

2 pts

File to hand in: ex_07.js, app.js, index.ejs

Restriction: Use Express, FS, Path, socket.io and jQuery

Time to get real

One of the main interests of Express/Node is the possibility for real-time communication between clients and server.

You will create an **app** module with a **start** method. (You know the drill by now)

As socket.io will be used it should obviously be installed.
Your app will render an *index.ejs* page, accessible from paths / and **/index**.

When a user accesses the page, your server will display, thanks to sockets, "New user connected" on the standard output followed by a new line.

Your *index.ejs* page must contain a tag ** **.

Upon connection, your server will send an event named **welcome** that contains the message "Welcome on my server!".

Upon receipt of this event, your client will display the content in a **ServerAnsw**. Use jQuery.

ex_07.js is only to be used to launch your app

Exercise 08

3 pts

File to hand in: ex_08.js, app.js, chat.js, index.html, style.css

Restriction: Use Express, FS, Path and socket.io

Let's chat! Shall we?

You will now implement a small chat.

Your **app** module will serve an *index.html* on path **/** and **/index**. This file is provided as well as a *style.css* file.

Implement in your *chat.js* the functionalities - included in *index.html* - and *server.js*.

The text input with id **username** lets the user change his username.

He will send the information when he presses the button with id **send_username** thanks to an event that you will name **change_username**.

The text input with id **message** lets the user send messages when the button with id **send_message** is pressed. The event will be called **new_message**.

When the server receives a new message, it must return "[username]: [message]" to all clients with an event named **new_message**.

Upon receipt of this event, the message will be added in the tag ** ** with the following format: ** message **.

ex_08.js is only to be used to launch your app

Exercise 09

2 pts

File to hand in: ex_09.js, app.js, index.html

Restriction: Use Express, FS and Path

Download this!

To conclude this glorious day, you are going to create a small file download system. The user should be able to download a file by simply typing its name in the URL like in the example below.

Example: **localhost:host/some_file.txt** will download the file *some_file.txt*.

The only files your **app** module can offer to download are to be found in a **files** directory. You will obviously create this directory.

- If the file doesn't exist, you must return an error 404 with the message "Error: Page not found".
- If it's a directory outside of the **files** directory, you will return an error 403 with the message "Error: Forbidden".

In addition, your server will have a *index.html* accessible from / and **/index**.

This page will contain links to each file contained in "files". In the case of file "private.txt", you will display a link with the following format:

private.txt

Note: you MUST NOT create a page for each file. As a matter of fact, if files are added when the server is started, it should still work as it supposed to.

ex_09.js is only to be used to launch your app