Lesson 8:
Node.js - continued (MVC and data management)

Lesson goal

This lesson continues the Node.js topic covering the code organization in the Express framework and the basic means of data persistence and user interaction data management.

The topics directly addressed include:

- Using partials in EJS views to avoid redundancy of the code;
- Separating the request handling functions into their own code files within controllers folder;
- Separating the persistent data definitions as JS classes into their own code files within models folder;
- Using the Express routing features including the URL path parameters;
- Processing the query string of the request;
- MySQL relational database management system installation and accessing it using the mysql2 package;
- Direct interaction with SQL database using the execute() method of the mysql2 package; using the notion of *promise* with its then() and catch() methods in the database API to ease the asynchronous style programming;
- Using the res and req objects to handle cookie creation and consuming on the server;
- Using the express-session package to manage user's session data.

Important details to investigate

- The features for *destructuring* in JS;
- The concept of Promise in JS;
- The optional details of HTTP cookie configuration (including e.g. the Secure and HttpOnly flags, the Expires and Max-Age properties and Domain and Path properties;
- The notion of a session cookie (not to be confused with the common use of cookie as the session implementation constituent)

External resources in English

- (see previous module)
- EJS syntax, including different kinds of markups: https://ejs.co/#docs
- Destructuring: https://codeburst.io/es6-destructuring-the-complete-guide-7f842d08b98f
- Express-session package: https://www.npmjs.com/package/express-session

Assignment

Build a website functionality matching the problem domain of your chosen semester project. The functionality should be implemented using Express according to the MVC pattern separation (partitioning code into models, views and controllers folders is recommended, and support the storing and retrieval of single entity instances (like e.g. Product, Employee or Article). The functionality should cover:

a) an index page listing all the instances by their names
b) an input form page for entering new instances
c) a page handling the instance creation requests
d) a details page showing all attributes for a single instance identified by request parameter (provided as a query string or (preferably) an URL path component.

The data should be persisted (as you choose) either though a JSON file or with a MySQL database.