NodeJS

EXPRESS 102: CRUD AND MVC

After setting up the skeleton for your project it's time to set up the database. As usual, there's quite a bit of background information that you will find useful as you progress.

Learning Outcomes

By the end of this lesson, you should be able to do the following:

- Explain CRUD and how it correlates to HTTP methods in Express.
- Describe MVC and how it correlates to Express.
- Describe databases and ORMs as well as how to use them with Node/Express apps.
- Design and create your own models using Mongoose.
- Declare object schema and models.
- Describe the main field types and basic validation.
- List a few ways to access model data.
- Test models by creating a number of instances (using a standalone script).

CRUD

CRUD is a concept that comes up a lot in web development, and it's the type of thing that might show up in interview questions so it's worth taking a little time to make sure you understand what it refers to. Thankfully, it's a relatively simple concept.

CRUD stands for: *Create, Read, Update* and *Delete*. These are the four basic functions that you will be building into your database driven apps. Put simply, if you are designing a CRUD interface that means that users can expect to be able to do these 4 things to items in the database (providing they have the appropriate permissions of course).

In your library example then this simply means that we are going to be building the ability for users to `create` entries (add books, authors or genres to the database), `read` entries (or, retrieve lists of books and other things from the database), `update` entries (edit details of an entry), and `delete` entries (remove them from the database).

Of course, this is simply a concept and not some sort of rule that must be followed. You may not want to allo users to do all of these actions, or you may want to limit which users can do what at any given time. For

example, if you are creating a social networking site, you might only allow users to `read` the profile information of their friends or connections, and you might not want to allow people to `delete` things at all.

The CRUD operations roughly correlate to the HTTP methods that you can employ in an express app. This definition can be somewhat flexible, but in general `create` correlates to `POST` (or `app.post()` in an express app), `read` correlates to `GET` (`app.get()`), `update` to `PUT` (`app.put()`) and `delete` to `DELETE` (`app.delete()`)

MVC

MVC is another common concept in web development and also something that is likely to come up in an interview question. MVC stands for *Model, View, Controller* and refers to the architecture of your code. Basically, it is a way to organize your application by separating all of the actions into 3 main components: Models, Views and Controllers.

Models are the basic building blocks of your database. So for every entry in your DB (books, authors, etc. in our Library Project), you'll create a model that holds the details of that entry. Models define the types of information that get used by your views and controllers.

Views are, of course, the component that generates the UI for your application. In our case, we've selected a templating engine that uses data supplied by a controller to display the desired information.

Controllers are the components that decide what view to display and what information is going to be put into it.

MVC Example

Without digging into the code prematurely, consider a very simple photo-uploading site. Users can upload and then view photos that are all listed on an index somewhere. In this case, we'll have a model for our photos that would define how our photos are stored in the database (DB). The model might specify that photos should be objects that have a `filename`, a `URL` and a `date-created` field.

We'll need two views, 1) the index, and 2) the display-photo view which will just display a single photo.

Our controller then would be called by Express whenever we get an `app.get()` and would then use the details of the request to determine which view is shown, and which image is displayed depending on whether the user is requesting the index or a specific photo's page.

If this is a little confusing at this point, don't worry about it too much. You will be creating models, views, and controllers in the tutorial and it will all become much clearer once you see them in use.

Which database should I choose?

One final note before diving back into the tutorial. Express does not care about which database you use. The lesson lists a few options but ultimately uses MongoDB. In this case, the actual DB you use matters little. If you later decide that you would rather use SQL or something else, you should be able to pick it up fairly easily by reading the documentation. At this point, Mongo is probably the most popular choice to use with Express so recommend just sticking with that for now.

Assignment

1. Continue where we left off with the MDN library tutorial (Part 3)!

Additional Resources

This section contains helpful links to other content. It isn't required, so consider it supplemental for if you need to dive deeper into something.

- For a deeper explanation of MVC you could read this article from freeCodeCamp.

development learning experience?

Get started