**Using MySQL In Express -** [Ryan Schleck](https://ryan-m-schleck.medium.com/?source=post_page-----57f900f5e362--------------------------------) Jul 28, 2022

**Introduction**

An important aspect of creating web applications is learning how to manage data. In a [previous article](https://ryan-m-schleck.medium.com/html-templating-within-express-js-7b668e34482a) I showed how user submitted data can be stored in memory using the Express framework. This of course is not best practice due to all information being lost whenever the server goes down. Two popular options for storing data in web applications are [SQL](https://www.sqlcourse.com/beginner-course/what-is-sql/) which stores information in tables that relate to each other, and [NoSQL](https://www.mongodb.com/nosql-explained) which stores data in key values that stand on their own. In this blog post I will be covering how to create a simple Express application that utilizes SQL to store data. We will be using a [relational database management system](https://www.codecademy.com/article/what-is-rdbms-sql) called [MySQL](https://www.mysql.com/) to do so.

**Creating Our Database**

Because the mascot of MySQL is a dolphin I thought our application might be a form users use to create dolphins. Once this form is completed the created dolphins are displayed on an index page. If you are a Linux user and are new to MySQL feel free to read [my article](https://ryan-m-schleck.medium.com/how-to-install-mysql-in-ubuntu-3945cdbffa16) on setting up MySQL in Ubuntu. The first step after we have MySQL successfully installed is creating our database that our app will interact with. To do this we will use the MySQL command line client. In order to start using this client open a terminal and enter:

sudo mysql

-OR-

If you possibly installed MySQL on Ubuntu via my previously mentioned tutorial you’ll have to enter:

mysql -u root -p

Be prepared to enter your MySQL password if you configured one. Now that you are within the MySQL command line client you should be seeing **mysql>** enter the following to create your database:

CREATE DATABASE IF NOT EXISTS express\_sql\_demo;

After this completes select this database by entering:

use express\_sql\_demo;

Once this is done you should see a confirmation that the selected database has changed. Now it’s time to create the table that will store information regarding the dolphins you create. We will call this table ‘dolphins’. Our dolphins are going to be pretty basic and only have a name, an age, and a mood. Make sure you are still within the MySQL command line client and enter the following:

Text

Description automatically generated

For more information on handy SQL queries checkout [this](https://www.codecademy.com/article/sql-commands) cheat sheet!

You should see another confirmation message. You can now exit the MySQL command line client by entering **exit**. Now that we have our database up and running lets start creating our application that will interact with it!

**Setting Up MySQL In Express**

The first step in this process is to create a new project folder, navigate into it, and open your text editor. Once you have your terminal open within this root directory you are of course going to want to initialize node by entering **npm init**. After the appearance of your package.json file you’ll want to install some packages as dependencies using the following format:

npm i --save *package\_name*

Do this for the following packages:

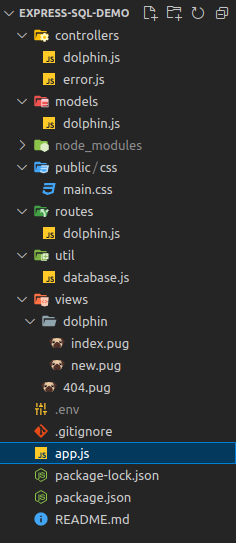
* [body-parser](https://www.npmjs.com/package/body-parser)
* [dotenv](https://www.npmjs.com/package/dotenv)
* [express](https://expressjs.com/)
* [mysql2](https://www.npmjs.com/package/mysql2)
* [pug](https://pugjs.org/api/getting-started.html)
* [nodemon](https://www.npmjs.com/package/nodemon)

You’ll probably also want to add a “start” script to your package.json file that specifies using nodemon. Once this is complete your package.json file should resemble the following:

Text

Description automatically generated

For this app we will be following a [Model View Controller](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) (MVC)pattern like the one I use in [this article](https://ryan-m-schleck.medium.com/mvc-patterns-in-express-c7efa25a7c86). Please construct your file structure similar to the following:



notice that app.js sits within the root of our application and thus is its starting point.

In order for our application to interact with our database we are going to want to supply our MySQL password within the connection configuration. Because of this it is important to create a [.gitignore file](https://www.freecodecamp.org/news/gitignore-what-is-it-and-how-to-add-to-repo/) and within it specify both our **node\_modules** folder that was created when we installed our packages, and a **.env** file you will use to store your password in a global constant. It is best practice to do this in order to prevent your password from being pushed to whatever repo you’re using. Once these two files are configured our first step will be to setup our MySQL connection. Within util/database.js we are going to import our dotenv package to gain access to our .env file along with our mysql2 middleware that will interact with our database. We will then use **mysql.createPool** in order to configure our connection. Once this is done our database.js should resemble this:

Text

Description automatically generated

You can read more about mysql.createPool [here](https://www.npmjs.com/package/mysql2#documentation) (scroll half way down to ‘Using connecion pools’)

Now that we have our database connection configured lets start working on the M in MVC which is our Model. Within **models/dolphin.js** lets define what attributes a dolphin within our application will have. We are also going to have several methods that interact with the database in regard to reading and writing data. For that we will want to import our database configuration up top. Our dolphin model might look something like this:

Text

Description automatically generated

For a closer look at Object Oriented Programming feel free to checkout [my blog post](https://ryan-m-schleck.medium.com/a-brief-look-at-object-oriented-programming-ecccdb08256c) all about it!

Once this is complete we are going to need some controller actions that will handle the logic regarding dolphins.

Within **controllers/dolphin.js** we will import our new Dolphin class and begin constructing some [middleware functions](https://expressjs.com/en/guide/using-middleware.html) that deal with this logic. If you are curious about the views that these functions render please feel free to checkout the [github repository](https://github.com/FluxOfPingIntegers/express-sql-demo" \t "_blank) for this demo app. Once all of this is complete our **controllers/dolphin.js** should resemble this following:

Text

Description automatically generated

If you are curious have a look at the documentation for information regarding **[res.render](https://expressjs.com/en/api.html" \l "res.render" \t "_blank)** and **[res.redirect](https://expressjs.com/en/api.html" \l "res.redirect" \t "_blank)**

In order to specify the routes to which these middleware functions will be applied we need to head over to **routes/dolphin.js** and import our new dolphin middleware. We will also need to create a [router object](https://expressjs.com/en/4x/api.html#router) and use it to specify which types of requests need to be routed to which controller actions. Once this is complete our **routes/dolphin.js** should resemble the following:

Text

Description automatically generated

Finally it is time to pool it all together. Within **app.js** at the top of our app we will need to import path to interact with our file system. We’ll import express because that is the web framework we’re using. We’ll also need to import body-parser in order to allow user data to be available within the request object. Then we will set our view engine to pug, and import/use our dolphin routes. Once this is complete **app.js** should resemble the following:

Text

Description automatically generated

**Conclusion**

With this our application is complete! We should now be able to fire up our server and save persistent dolphins our users create. If you are interested in how to interact with a database in Express without having to directly write SQL queries as we did I suggest checking out the node package called [Sequelize](https://sequelize.org/" \t "_blank). I hope this article has been helpful and thanks for reading!