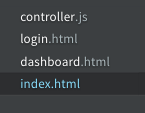
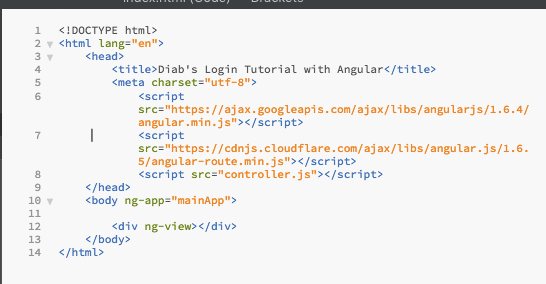
Using AngularJS Login A Database

I chose to use AngularJS as my Exploration. Initially, I made a mongoDB database to correlate with this, but things went south and I started to run out of time. So in this tutorial, I’m going to show you how to create a secure login, and redirect a user back to the login page if they don’t have a login. Essentially, the code is ready to bring in mongoDB database with users and passwords, mind just was not working like I wanted it to so I kept it out.

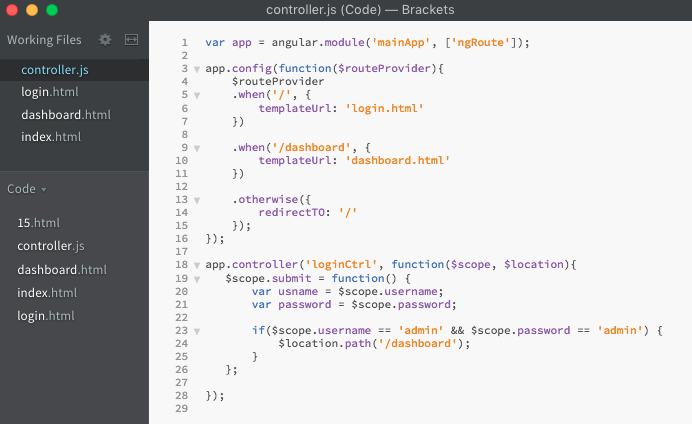
1. Set up all of the files that you are going to need.



1. Set Up Your index.html file And Get your CDN’s to link the AngularJS API, and import controller.js that will control your page functionality using Angular.
   1. I used <https://cdnjs.com/libraries/materialize> to get my CDN’s. I added comments to describe what was what.
   2. Add #mainApp id in order for controller.js to redirect the webpage to that index
   3. Also, create a dashboard.html with a header in order to redirect users after they log in to show that they are actually logged in



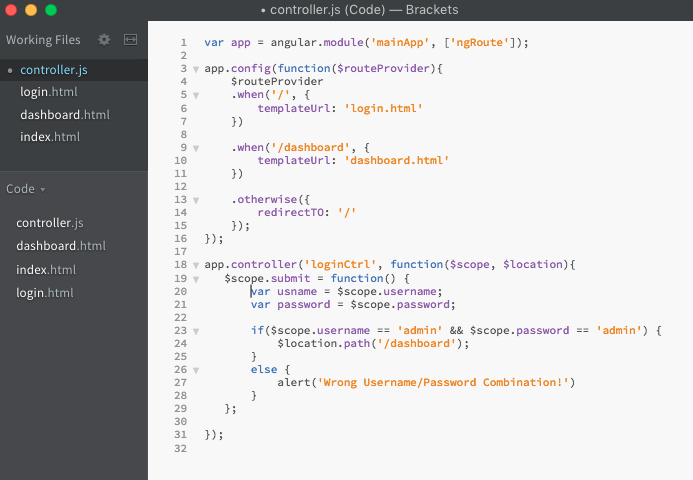
1. Create a Basic Controller Layout To controller.js
   1. I watched a youtube tutorial on how to do this, will explain functionality later on:



1. Set Up Your login.html File To Correlate With Variable and Functions in Controller
   1. We create a form like you normally you, except instead of name, we use ng-model in order for AngularJS to handle the variables from backend

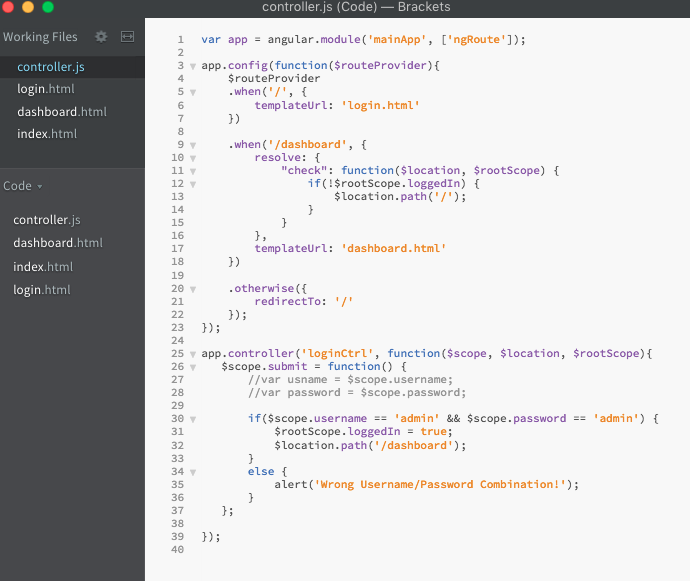


1. Now, We Can Create A Simple Alert If The User Does Not Enter In Correct Credentials



We can now log in as a user with admin as the Username and admin as the Password. But if you notice, you direct yourself straight to the dashboard from the URL, we’re going to fix that now.

1. Preventing Unauthorized Dashboard Viewing With AngularJS
   1. Instead of just using $scope, we can use $rootScope as a global variable in order to use it throughout the controller file.
   2. Before, when a user logged in, it redirected the user to go straight to the Dashboard with no checks. Here, we will create a check to make sure a user is actually logged in. Make sure you add the $rootScope variable to the function call
   3. In the submit function where we create the log in relationship, comment out the old scope variables. Once the username and password is verified, we will set our $rootScope global variable to true. That verifies a user is actually logged in.



Recap:

Overall, AngularJS was easy to use, and really didn’t stray far from a regular php file you could set up. I did research beforehand, and it seems like there is the most info about how to use Angular on the web. Although I didn’t get to implement the database, I got the main view of how it will work with backend SQL or noSQL, and it will be easier for me to use when I get it figured out next time.

The biggest downfall was trying to implement the mongoDB database into my web page. What was happening was I could not establish a connection with mongoDB. It wasn’t showing that I had a database when I clearly did have one, and another downfall was the shortage of documentation of using mongoDB with AngularJS. I found a way to connect straight to it, and then it was not storing any users to my database collection. I tried several different ways, and I meant to get screenshots of it, but I restarted and tried so many different ways that I lost track. I ran out of time, but for the next exploration I will use either AngularJS or Node.js to make a connection with my database in order to create a real database log in.

Side note: This is not a secure login at all. I used this as an example on how to create a LogIn using angularJS. Do not try to use this as a secure login to your web page.