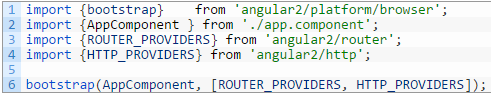
# Angular2 Authentication

Open up index.html and add the below two lines to the <script src> lines so that it looks as shown below.



We are simply importing the router and http js files so that we could use the classes therein in our code.

Open up main.ts file and modify the code to look like this.



ROUTER\_PROVIDERS and HTTP\_PROVIDERS are two constant arrays that contain references to the classes which we will be using in our app.

Open up app.component.ts and add the below code.



I have simply created a couple of route configurations to a login page(which gets loaded by default and a dashboard page).

For more clarity on how the routing actually works, please read my Angular2 – Routing post here.

Now let’s go ahead and create the login and dashboard pages respectively.

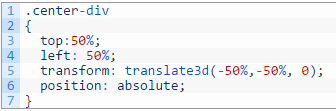
Create a new folder called login and add two files, login.html and login.component.ts inside it.

Open up login.html and add the below code.

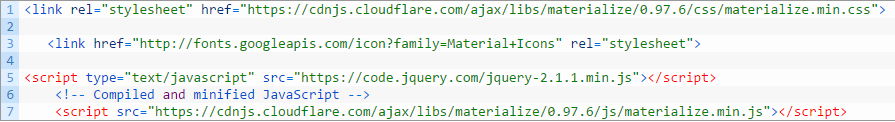


I have used a framework called materializecss to create a fancy login box (or card, as they say it). You can use any framework you desire.

In case you use the same code as above, please add the below code to your styles.css file as well



Don’t forget to add the css files and script files in your index.html



If you use any other frameworks kindly include the src files of that particular framework.

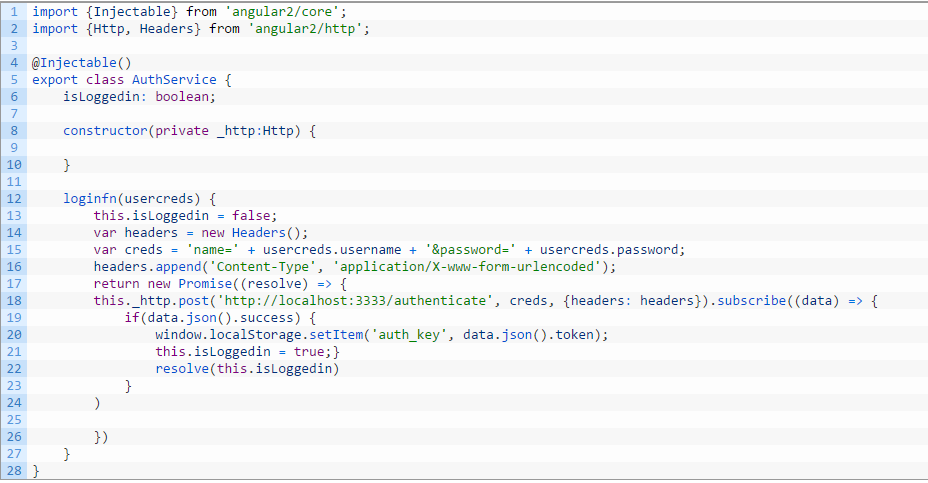
Now open up login.component.ts and add the below code.



Let’s break this down.

1. I have imported a service called authservice in here and used a function loginfn() in that service to authenticate the user.
2. I have imported Router to navigate to the dashboard upon successful authentication.
3. Clear fields is simply a clear values method used to clear the values in the login form.

Now create a new folder called services and then add a authservice.ts file in it.



Let’s break this down.

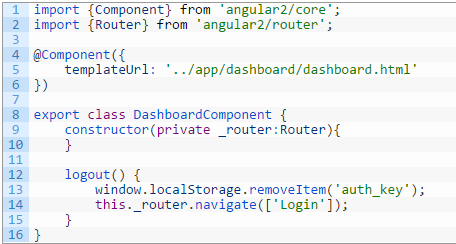
1. Since this is a service and I would be injecting it into other components to use the methods here I have used the @Injectable() decorator here.
2. I am making a http post request to the backend api passing along the creds and headers.
3. If the authentication is successful then I store the authkey which I obtain in the localStorage and the promise is resolved to true.

Now create a new folder called dashboard and inside it create two new files dashboard.html and dashboard.component.ts

Open up dashboard.html and add the below code.



and add the below code in dashboard.component.ts



I simply have a button in that page that is hooked up to the logout method which removes the localStorage key and redirects the user to the login page.

Now if everything has been written fine, then you could go ahead and give the below command.

**npm start**

|  |  |
| --- | --- |
|  |  |

Please ensure that the backend is running before giving this command.

Enter your username and password and your authentication goes fine. Hola..!!

Whoa.. !! Wait.. Even without entering my username and password I am still able to navigate to dashboard page. Where is security in my app ?

Well to provide route security that is to prevent access to authenticated routes you need to do the following.

If you notice we have used a <router-outlet> directive right ? We could tweak this slightly to do our desired authentication and provide security.

Create a new file called authcheck.ts at the same level as app.component.ts and add the below code in it.



Here I have a publicRoutes variable in which I am storing the routes that don’t require authentication. In our case only login.

Then I have an activate method which first checks the url. If the url is not present in the public routes and if there is no local storage item with the key (‘auth\_key) then it simply redirects the user back to login page.

If the url is not present in the public route and there exists an item in the localStorage with the authkey then the navigation happens., i.e., the user is authenticated.

Save the file and try navigating to dashboard now. You will see that you are redirected back to the login page.

This is how authentication can be done in angular2 apps.