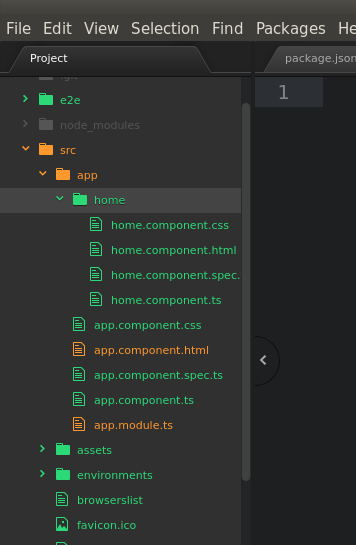
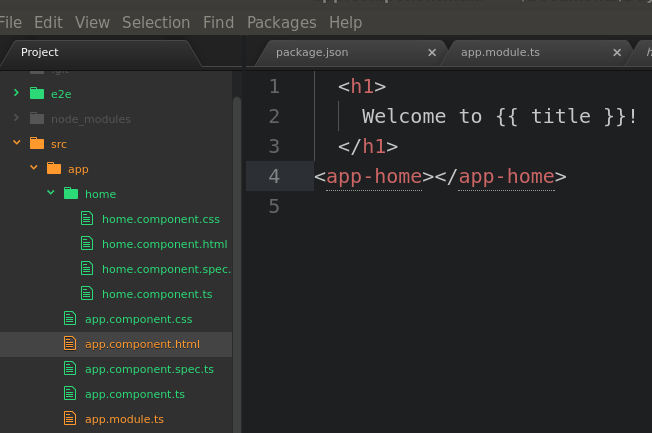
Day04 Introduction to NG 7

part 1 – NG7 Setup

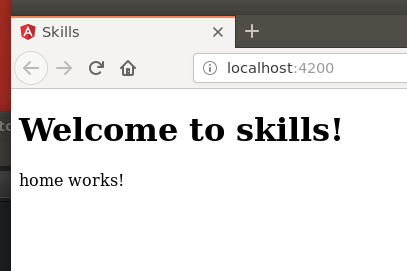
1. From Documents/Day02, open in terminal window to that folder and type the command **ng new skills**
2. Choose **Y** for routing and **CSS** for styles, once done CD into the skills folder and type **ng serve** to run the basic app. You can then open a browser to localhost:4200 to see the basic app. Note, to choose CSS use the arrow keys on the keyboard.
3. CD into the skills directory and open a terminal window pointing to that folder. After typing **ng serve** in the terminal window, notice the word compiling, the type script (ts) code is compiled in order to run successfully.
4. Open the application in Atom and most of our work will be in the **app** folder, which acts like the parent folder.
5. View the code of app.component.html, this is the file that feeds the default page that shows up on the browser at 4200. Remove everything except the **h1** tags and content. Notice the browser refreshes automatically.
6. In the skills folder, open in terminal window in order to start a new custom component. Once there, type **ng generate component home**
7. Back on atom, you should see a new folder called “home”  
   

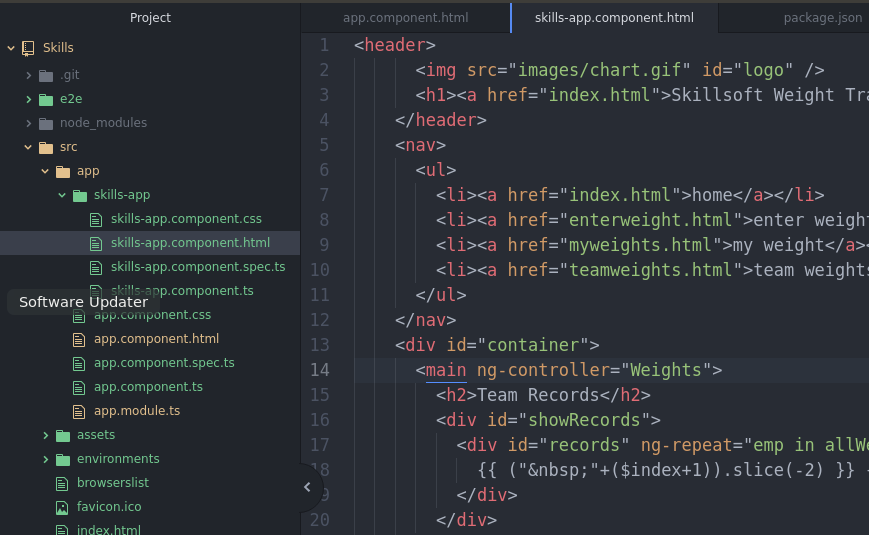
part 2 – Configuring New App

1. Open home.component.html file and notice that a line of code was already inserted automatically
2. Go back to the parent folder and look at the file app.component.ts, there is a **selector** property with a value of **app-root**. Now open index.html under the original app folder and notice that between the body tags, this name appears between angle brackets.
3. Now go back to our component and inside of home.component.ts file there is a **selector** with the name **app-home**, lets insert this name in the app.component.html file with angle brackets.



1. Once you save, the browser will reload the new page



1. We will use the original index.html file from Day01 as an example for the next step. Open that page in an editor and copy everything between the body tags except the script tags. Paste all this code in the home.component.html file replacing the original text that was there.  
   
2. Copy the image from the original images folder into the assets folder of the new NG7 app. Change the image link in home.component.html file accordingly

|  |
| --- |
| **<img src="./assets/chart.gif" id="logo" />** |

1. Copy the original css file and replace the NG7 css file in the src folder
2. Finally for this part, remove the *Welcome to Skills* text from the parent app, app.component.html

part 3 – Routing

At this point we do not have true routing, we hard coded our home page to show up on the first hit to :4200, we will fix this by changing app-routing.ts

1. In app-routing.ts import the HomeComponent component

|  |
| --- |
| **import { NgModule } from '@angular/core';**  **import { Routes, RouterModule } from '@angular/router';**  **import { HomeComponent } from './home/home.component';**  **const routes: Routes = [];** |

1. Enter the home route in the Routes[] array as a JSON object

|  |
| --- |
| **import { HomeComponent } from './home/home.component';**  **const routes: Routes = [**  **{ path: 'home', component: HomeComponent }**  **];** |

1. Now we can remove the <app-home> element from app.component.html. so this file should only have the <router-outlet> element, nothing else. However when the Angular home page refreshes the content is gone. So go to <http://localhost:4200/home> and it should return
2. While we are on this topic lets add a default route, so if the user goes to just :4200, they should see the home page. So back in app-routing.module.ts add the default path as shown below:

|  |
| --- |
| **import { HomeComponent } from './home/home.component';**  **const routes: Routes = [**  **{ path: '', redirectTo: '/home', pathMatch: 'full' },**  **{ path: 'home', component: HomeComponent }**  **];** |

Notice the comma after the first path. Also it is above the home path.

1. Lets add a component and route for teamweights as this is the easiest file to configure. Back in the terminal window, make sure you are in the skills folder and run the same ng generate component command to generate the team weights component. So **n g generate component teamweights**
2. Go into app-routing.ts module and import the teamweights component and add it to the routes array. Spelling is important so sometimes we can go into the app.module.ts file and copy the import statement from there

|  |
| --- |
| **import { HomeComponent } from './home/home.component';**  **import { TeamweightsComponent } from './teamweights/teamweights.component';**  **const routes: Routes = [**  **{ path: '', redirectTo: '/home', pathMatch: 'full' },**  **{ path: 'home', component: HomeComponent },**  **{ path: 'teamweights', component: TeamweightsComponent }**  **];**  **@NgModule({** |

Note the spelling, for the import statement and the path statement.

1. At this point, if we go to 4200/teamweights, the browser will state *teamweights works!,* so we now have to replace the content of that teamweights.components.html page with the content from Day01
2. Open teamweights.html from day01 and copy all the HTML content between the **body** tags, except any links to JavaScript. Paste that code into teamweights.component.html file.
3. Copy the image statement from home.component.html around line 2 and paste it in the same position in the teamweights.component.html file

|  |
| --- |
| **<header>**  **<img src="./assets/chart.gif" id="logo" />**  **<h1><a href="index.html">Skillsoft Weight Tracker</a></h1>** |

1. Now we have to change the menu to be able to send the browser to these various pages. We have to add RouterLink directives on the anchor tags give the router control over those elements. Add the following lines of code to the nav->ul structure on both component html files

|  |
| --- |
| **<ul>**  **<li><a routerLink="/home">home</a></li>**  **<li><a routerLink="/enterweight">enter weight</a></li>**  **<li><a routerLink="/myweight">my weight</a></li>**  **<li><a routerLink="/teamweights">team weights</a></li>**  **</ul>** |

Now you should be able to navigate between home and team weights. The same procedure will be applied to any new component built from this point.

part 4 – API Calls

1. We would need the http module, so in the parent app.module.ts file import this module

|  |
| --- |
| **import { BrowserModule } from '@angular/platform-browser';**  **import { NgModule } from '@angular/core';**  **import { HttpClientModule } from '@angular/common/http';**  **import { AppComponent } from './app.component';**  **import { SkillsAppComponent } from './home/home.component';** |

1. Import the module at the imports section:

|  |
| --- |
| **AppComponent,**  **SkillsAppComponent**  **],**  **imports: [**  **BrowserModule,**  **HttpClientModule**  **],**  **providers: [],**  **bootstrap: [AppComponent]** |

(remember to insert a comma at the line above)

1. Back in the child component, go into the teamweights.component.ts file and import the **HttpClient** like we did in the parent ts file

|  |
| --- |
| **import { Component, OnInit } from '@angular/core';**  **import { HttpClient } from '@angular/common/http';**  **@Component({** |

1. Now in the constructor, pass in the HttpClient

|  |
| --- |
| **export class SkillsAppComponent implements OnInit {**  **constructor(private http: HttpClient) { }**  **ngOnInit() {** |

1. Within that class, create a new property call it allweights:

|  |
| --- |
| **export class SkillsAppComponent implements OnInit {**  **allWeights: any;**  **constructor(private http: HttpClient) { }** |

1. Once the app starts, the method ngOnInit() will fire, in here we can use our http object to make the call

|  |
| --- |
| **allWeights: any;**  **constructor(private http: HttpClient) { }**  **ngOnInit() {**  **this.http.get('http://localhost:8000/getallrecords');**  **}** |

1. Assign the observable returned from the get() to allWeights

|  |
| --- |
| **ngOnInit() {**  **this.allWeights = this.http.get('http://localhost:8000/getallrecords');**  **}** |

1. Now back on the **html** file, insert this code

|  |
| --- |
| **<h2>Team Records</h2>**  **<div id="showRecords">**  **<div id="records" \*ngFor="let emp of allWeights | async">**  **</div>**  **</div>**  **</main>** |

We extract each document and pipe it to an async method just in case the content changes

1. Now we have access to the individual fields

|  |
| --- |
| **<h2>Team Records</h2>**  **<div id="showRecords">**  **<div id="records" \*ngFor="let emp of allWeights | async">**  **{{emp.empName | json}} weighed {{emp.empWeight}} on [date].**  **</div>**  **</div** |

