Create Angular Starter Apps Lab

Angular CLI is a tool to help you generate Angular projects.

**Goals**

* Install the Angular CLI tool
* Create a new Angular project
* Run the generated Angular project
* Modify the Angular source code

1. Install the Angular CLI Tool

In this section, you install Angular CLI and verify the installation.

1. Open a new terminal window.
2. Install the Angular CLI tool:

npm install -g @angular/cli

1. Verify the installation:

ng --version

1. Verify that Help is available:

ng help

2. Create a New Angular Project

In this section, you use the Angular CLI tool to create a new Angular project.

1. Move to the **angular-training** directory that you created in a previous lab:

cd angular-training

1. Create a new **my-angular-demo** project:

ng new my-angular-demo

* + The command may take several minutes to complete.

3. Run the Generated Angular Project

In this section, you run the generated Angular project.

1. Move into the new project directory:

cd my-angular-demo

1. Run the app:

ng serve

1. Wait until you see the following text:
2. ng serve
3. \*\* NG Live Development Server is running on http:*//localhost:4200 \*\**
4. Hash: 719df4eb209182a9a4fc
5. Time: 10090ms
6. chunk {0} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 157 kB {4} [initial] [rendered]
7. chunk {1} main.bundle.js, main.bundle.js.map (main) 4.05 kB {3} [initial] [rendered]
8. chunk {2} styles.bundle.js, styles.bundle.js.map (styles) 9.77 kB {4} [initial] [rendered]
9. chunk {3} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.69 MB [initial] [rendered]
10. chunk {4} inline.bundle.js, inline.bundle.js.map (inline) 0 bytes [entry] [rendered]

webpack: Compiled successfully.

* + Keep this terminal window running. It contains the server that is listening on port 4200.

1. Click this link to go to [http://localhost:4200](http://localhost:4200/).
   * Expect to see a screenshot similar to this one:

4. Explore the Angular Project

1. Open the Visual Studio code text editor.

4.1. View the App Component

1. Open and review **src/app/app.component.ts**:
2. import { Component } from '@angular/core';
3. @Component({
4. selector: 'app-root',
5. templateUrl: './app.component.html',
6. styleUrls: ['./app.component.css']
7. })
8. export class AppComponent {
9. title = 'app works!';

}

* + This component defines the selector for **app-root**. The HTML template is stored in **app.component.html**. Supporting CSS styles are in **app.component.css**.

1. Open and view these additional files:
   * **app.component.html**
   * **app.component.css**
2. Note the **title** property.
   * The HTML template performs data binding with this property, as shown below.
   * <h1>
   * {{title}}

</h1>

4.2. View the App Module

The app module is a collection of support classes that your application uses, including any dependencies and bootstrap classes.

* Open and review **src/app/app.module.ts**:
* import { BrowserModule } from '@angular/platform-browser';
* import { NgModule } from '@angular/core';
* import { FormsModule } from '@angular/forms';
* import { HttpModule } from '@angular/http';
* import { AppComponent } from './app.component';
* @NgModule({
* declarations: [
* AppComponent
* ],
* imports: [
* BrowserModule,
* FormsModule,
* HttpModule
* ],
* providers: [],
* bootstrap: [AppComponent]
* })

export class AppModule { }

4.3. View the Main Bootstrap Code

On application startup, the bootstrap code loads and compiles your module. The code uses a just-in-time compiler to compile the code dynamically in the client-side browser.

* Open and review **src/main.ts**:
* import { enableProdMode } from '@angular/core';
* import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
* import { AppModule } from './app/app.module';
* import { environment } from './environments/environment';
* if (environment.production) {
* enableProdMode();
* }

platformBrowserDynamic().bootstrapModule(AppModule);

4.4. View the HTML Page

The HTML file has a placeholder for the component selector named **app-root**. When the app is rendered, the selector **app-root** is replaced with the HTML code of **AppComponent**.

* Open and review **src/index.html**:
* <body>
* <app-root>Loading...</app-root>

</body>

5. Modify the Angular Source Code

Now that you understand the source code, make some modifications to it and review the results.

1. Open **src/app/app.component.ts**.
2. Change the value of **title** to **Hello World of TypeScript!** and save the file.
3. View your browser to see the updated results:

Remember, the ng serve command automatically reloads the code.

5.1. Add New Fields

Modify the code to display a first name and last name. Save your files after you modify them.

1. Add the following fields to **src/app/app.component.ts**:(Typescript class)
2. firstName = "Sasirekha";

lastName = "Vijayan";

1. Add the following template to **src/app/app.component.html**:
2. <h2>
3. {{firstName}} {{lastName}}

</h2>

1. View the updated results in your web browser:

5.2. Add Support for an Address Field

Modify the code to record a user’s address: street, city, and state. Save each file after you modify it.

1. To the end of **src/app/app.component.ts**, add a new class for address:
2. class Address {
3. street : string;
4. city : string;

}

* + This **Address** class defines two fields: **street** and **city**.

1. After the **lastName** field of the **AppComponent** class, add the following code:
2. location : Address = {
3. "street" : "1007 Mountain Drive",
4. "city" : "Gotham"

};

* + This code defines a new field named **location** of type **Address**, the class created in a previous step.
  + The **location** field is initialized using a JSON object.
  + The JSON object is a collection of name-value pairs that correspond to the **Address** class.

1. Verify that your final code looks like this:
2. import { Component } from '@angular/core';
3. @Component({
4. selector: 'app-root',
5. templateUrl: './app.component.html',
6. styleUrls: ['./app.component.css']
7. })
8. export class AppComponent {
9. title = 'Hello World of TypeScript!';
10. firstName = "Bruce";
11. lastName = "Wayne";
12. location : Address = {
13. "street" : "1007 Mountain Drive",
14. "city" : "Gotham"
15. };
16. }
17. class Address {
18. street : string;
19. city : string;

}

1. At the end of **src/app/app.component.html**, add the following HTML template code:
2. <hr>
3. <p>
4. Street: {{location.street}}
5. <br>
6. City: {{location.city}}

</p>

* + This template accesses a field of the **AppComponent** class. It accesses the **location** field (of type **Address**), which has **street** and **city** fields.

1. View the updated results in your web browser: