Angular Workshop

From Beginner to Pro, Creating Applications for the Real World

Abdelfattah Ragab

Angular Workshop

From Beginner to Pro, Creating Applications for the Real World

Abdelfattah Ragab

Source code

The source code is available on the book's website
https://books.abdelfattah-ragab.com

Introduction

Welcome to the book “Angular Workshop: From beginner to pro, Building Applications for the real world”.
In this book I will explain to you from scratch how to create a professional Angular application. The interface looks simple, but the functionality is powerful and real.
If you want to familiarise yourself with Angular, I highly recommend this book.
In this hands-on book, we will create a simple application with a shopping cart using the latest Angular v19.2 and NgRx SignalStore.
By the end of this book, you will have all the tools you need to create a real application.
Let us get started.

Install Node.js

Visite https://nodejs.org/en/download to download and install the node.js
After installation open the terminal and execute the following command to verify the installed Node.js version

```
node -v
```

And this command to verify the installed npm version

```
npm -v
```

Install the Angular CLI

To install the Angular CLI, open a terminal window and run the following command:

```
npm install -g @angular/cli
```

Install Visual Studio Code

Visual Studio Code is a code editor that we will use to develop our applications.
Go to https://code.visualstudio.com/ to download and install the application.

Create a new Application

Open a terminal window where you want to create the project and execute the following command:

```
ng new my-app
```

Run the Application

Run the following command

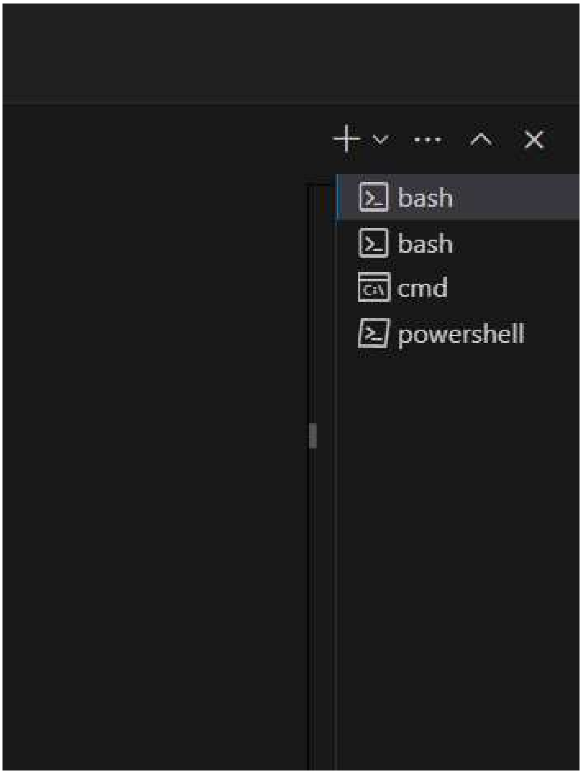
```
ng serve --open
```

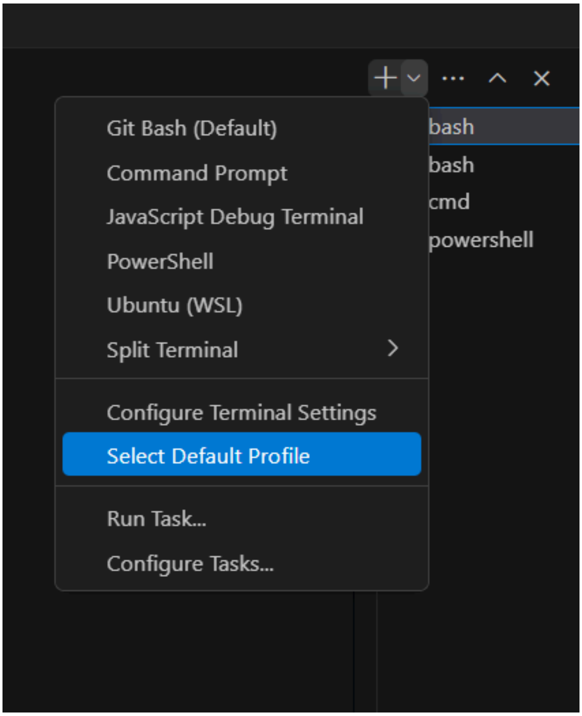
Open the Project in VS Code

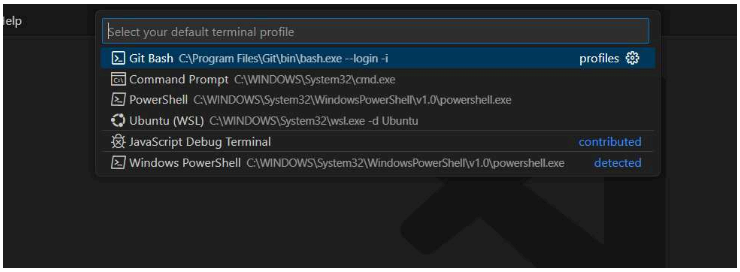
Open Visual Studio Code and choose “Open Folder…”.
Open the folder of your project.

Terminal in Visual Studio Code

You can open the terminal from within the Visual Studio Code.
You can open many terminals and also set the default:


+
bash
bash
cmd
powershell

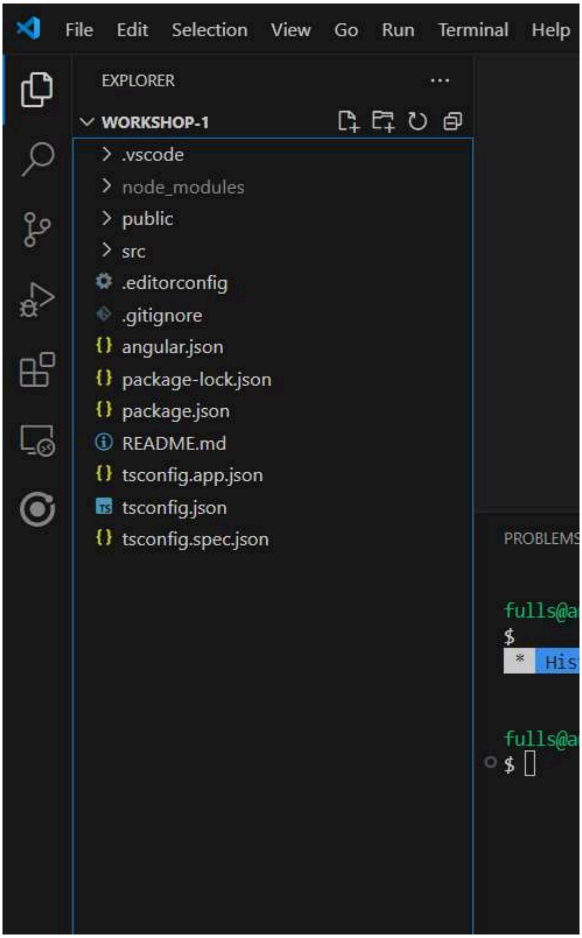




```
Select your default terminal profile
Git Bash  C:\Program Files\Git\bin\bash.exe --login -i                                profiles
Command Prompt  C:\WINDOWS\System32\cmd.exe
PowerShell  C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe
Ubuntu (WSL)  C:\WINDOWS\System32\wsl.exe -d Ubuntu
JavaScript Debug Terminal                                                          contributed
Windows PowerShell  C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe        detected
```

Explorer

To the left you will find the explorer with a tree of the project folders and files



```
.vscode
node_modules
public
src
.editorconfig
.gitignore
angular.json
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
```

```
fulls@a
$
 * His

fulls@a
$
```

index.html

In the “src” folder you will find the “index.html” file.
You can set the application title, the app icon, the SEO details and other metadata.

The “public” folder

We put the static images and so on in the public folder. They will be available to all users without restrictions, they are simply public. Things like logo, background image, and so on.

The “styles.css” file

We put global styles in this file. It will be applied to the whole application.
Things like the styles of the html, the body or the declaration of the :root variables.

The “main.ts” file

This file links the index.html file to the Angular application. Most of the time, there is nothing to do with this file.

The “app.config.ts” file

This is where we configure our application. You will need to visit this file everytime you add a new library. Almost all libraries will ask you to add a new configuration line here in this file to be able to use it.

The “app.routes.ts” file

Here we configure the application routes. Routes are simply the paths in our application that we write in the address bar, “/about” for example.
We define what component Angular should load for the “/about” path.

The “package.json” file

The package.json file is important for managing dependencies and project settings. It contains metadata about the project, such as name and version, and lists the npm packages required for the application and development tools.

The “package-lock.json” file

The package-lock.json file is an automatically generated file that accompanies the package.json. Its primary purpose is to lock the versions of all dependencies and their sub-dependencies to ensure consistent installations across different environments. This file captures the entire dependency tree, including minor and patch versions, which helps maintain stability in the application as it evolves.

The “angular.json” file

The angular.json file is primarily used by the Angular CLI. It contains settings that define the structure and behavior of the project, including information about the project's root directory, build configurations, and the various assets and styles to be included. This file allows developers to manage multiple projects within a single workspace and customize build options for different environments, such as development and production. Essentially, angular.json serves as the main source of configuration for the entire Angular application, guiding how the project is built and served

The “tsconfig.json” file

The tsconfig.json file is a configuration file for TypeScript projects that indicates the root of the project and specifies the compiler options required for compiling TypeScript code. It contains a set of key-value pairs that control various aspects of the compilation process, such as the target ECMAScript version, module system, and strictness settings. By defining these options, the tsconfig.json file helps ensure that the TypeScript code is compiled consistently and correctly, tailored to the specific needs of the project.

The “tsconfig.app.json” file

The tsconfig.app.json file extends the base tsconfig.json file and typically includes configurations that are tailored for the Angular app, such as specifying the root directory for the application source files and the output directory for compiled files. This separation allows for different configurations for the application and other parts of the project, such as tests, ensuring that each part can have its own specific compiler options while still maintaining a common base configuration.

The “tsconfig.spec.json” file

The tsconfig.spec.json file extends the base tsconfig.json file and includes settings tailored for the application's test environment. This configuration typically specifies options such as the output directory for compiled test files, the module system, and the types required for testing, such as Jasmine and Node. By having a dedicated tsconfig.spec.json file, developers can ensure that the TypeScript compiler uses the appropriate settings when compiling test files, separate from the main application code.

The “.gitignore” file

The .gitignore file in an Angular project specifies which files and directories should be ignored by Git, helping to keep the repository clean and free from unnecessary files. It typically includes entries for directories like node_modules/, which contains project dependencies that can be regenerated, and dist/, where compiled output is stored. By using this file, developers ensure that only relevant source code and configuration files are tracked, making collaboration and deployment more efficient.

The “.editorconfig” file

The .editorconfig file is a configuration file that helps maintain consistent coding styles across different editors and IDEs used by team members. It contains a set of rules regarding formatting, such as indentation styles, line endings, and character encoding, which can be applied automatically by compatible text editors. By using this file, developers ensure that their code adheres to a uniform style, improving collaboration and reducing formatting discrepancies in the codebase.

The “.vscode” folder

The .vscode folder contains configuration files specific to Visual Studio Code, which help customize the development environment for that particular project. This folder may include settings such as workspace preferences, debugging configurations, and extensions that enhance the coding experience. While it is not used by Angular itself, it allows developers to maintain consistent settings across different team members' environments, improving collaboration and productivity when working on the project.

The “node_modules” folder

The node_modules folder is the place where all dependencies of the project are stored. When you install packages with npm, the required libraries and modules are downloaded and stored in this folder. This ensures that all necessary dependencies are present so that the project can be executed successfully. The content of the node_modules folder is defined in the package.json file, which lists the libraries required for the application. This folder is essential for the development and functioning of Angular applications, as it contains both third-party libraries and tools that the project relies on. This file is created when we run the `npm install` command, so we don’t keep track of it. It is sufficient to track the package.json and package-lock.json files.

The “dist” folder

The dist folder is the build output directory that contains all the files necessary for deploying the application to a server. When you run the build command, Angular compiles the application and generates optimized files, including transpiled JavaScript, HTML, and CSS, which are placed in the dist folder. This folder is essential for production deployment, as it holds the final version of the application that can be hosted and served to users.

The “.angular” folder

The .angular folder is created by the Angular CLI and is used primarily for caching purposes during development. This folder stores various build artifacts and metadata that help speed up the development process by avoiding unnecessary recompilation of unchanged files. It is typically included in the .gitignore file, meaning it is not tracked by version control, as it is not necessary for the application's functionality and can be regenerated as needed.

The AppComponent

The AppComponent is the root component that serves as the main entry point for the application, responsible for bootstrapping the entire component tree. It typically contains the overall layout and structure of the application, including its template and styles, and can include other child components, allowing for a modular and organized architecture within the application.

What is Component

Component is a fundamental building block of the user interface, encapsulating the application's logic, template, and styles. Each component is defined by a TypeScript class decorated with the @Component decorator, which specifies metadata such as the component's selector, template, and styles. Components enable developers to break down the application into smaller, reusable pieces, facilitating better organization and maintainability of the codebase.
You will have a file for the logic or code behind. Another file for the html template, and one for the styles and finally a fourth file for testing.
All files will have the same component name with different file extensions, for example if we have a component for the home page, it will have four files as follows:

- home.component.css
- home.component.html
- home.component.spec.ts
- home.component.ts

The Hello Placeholder

Delete the hello placeholder (everything) in the "app.component.html" file and leave only the <router-outline /> line.

Create the Home Page

To create a new component, execute the command:

```
ng g c pages/home
```

or

```
ng generate component pages/home
```

There is no "pages" folder yet, but don't worry, Angular will create it and put the new "home" component in it. Pages in Angular are just normal components but what makes them pages is that they are connected to a route. So, the next step is to connect the home page to a route, so when the user types /home in the address bar it should load the home component. Also it would be nice if Angular redirects to the /home page by default when the application starts.

Routes

Go to the "app.routes.ts" and add a new entry to the Routes array as follows:

```
import { Routes } from '@angular/router';
import { HomeComponent } from
'./pages/home/home.component';
```

```
export const routes: Routes = [
  { path: 'home', component: HomeComponent,
title: 'Home - Workshop' },
];
```

By doing this you can type http://localhost:4200/**home** in the address bar and it will load the home component.
To load the "home" path by default, add the following:

```
import { Routes } from '@angular/router';
import { HomeComponent } from
'./pages/home/home.component';

export const routes: Routes = [
  { path: '', redirectTo: 'home', pathMatch:
'full' },
  { path: 'home', component: HomeComponent,
title: 'Home - Workshop' },
];
```

If it is on the base url "/" so it should redirect to the "/home" address.
The pathMatch: 'full', tells Angular to do so only if the entire URL path exactly matches the specified path. Which means in our case to redirect to the home page only if it is exactly on the base url and not any other url. Everytime you try to visit http://localhost:4200/ it will redirect to http://localhost:4200/home
Similar to this you can create pages for about, contact and so on

```
ng g c pages/about
ng g c pages/contact
```