

# ACE Inspiration

Angular (Web Framework)



# Content

---

- ▶ What is a Stored Procedure?
- ▶ CREATE Stored Procedure
- ▶ Stored Procedure Variables
- ▶ Stored Procedure Parameters
- ▶ Conditional Statements
- ▶ Loop Statements
- ▶ Cursor
- ▶ Calling Procedure in Java Application

# What is Angular?

- ▶ Angular is a front-end framework and written in TypeScript.
- ▶ Angular is free and open-source to build dynamic and single-page applications (SPAs).
- ▶ Angular is led by the Angular Team at Google.

# Benefits of Angular

---

- ▶ Gives clean structure application
- ▶ Includes lots of reusable code
- ▶ Makes more testable application

# Angular Versions

---

Angular 1 — also called AngularJS; was released in 2010.

Angular 2 — released in September 2016.

Angular 4 — released in March 2017.

Angular 5 — released in November 2017.

Angular 6 — released in May 2018.

Angular 7 — released in the same year in October 2018.

Angular 8 — released in May 2019.

Angular 9 — released in February 2020.

Angular 10 — released in June 2020.

Angular 11 — released in November 2020.

Angular 12 — released in May 2021.

Angular 13 — released in November 2021.

Angular 14 — released in July 2022.

# TypeScript Versions

TypeScript 0.8 - October 2012

TypeScript 0.9 - June 2013

TypeScript 1.0 - October 2014

TypeScript 2.0 - September 2016

TypeScript 3.0 - July 2018

TypeScript 4.0 - August 2020

TypeScript 4.7.4 - June 2022

# TypeScript Data Types

Data type	Keyword	Description
Number	number	Double precision 64-bit floating point values. It can be used to represent both, integers and fractions.
String	string	Represents a sequence of Unicode characters
Boolean	boolean	Represents logical values, true and false
Void	void	Used on function return types to represent non-returning functions
Null	null	Represents an intentional absence of an object value.
Undefined	undefined	Denotes value given to all uninitialized variables

# Install Angular on Windows

---

## Step 1: Install Node.js

- ▶ to develop your angular app to use necessary tools.
- ▶ to give npm tool that allows to download libraries and packages used in Angular.

## Step 2: Install TypeScript (Optional)

- ▶ can install TypeScript as an NPM package.

## Step 3: Install Angular CLI

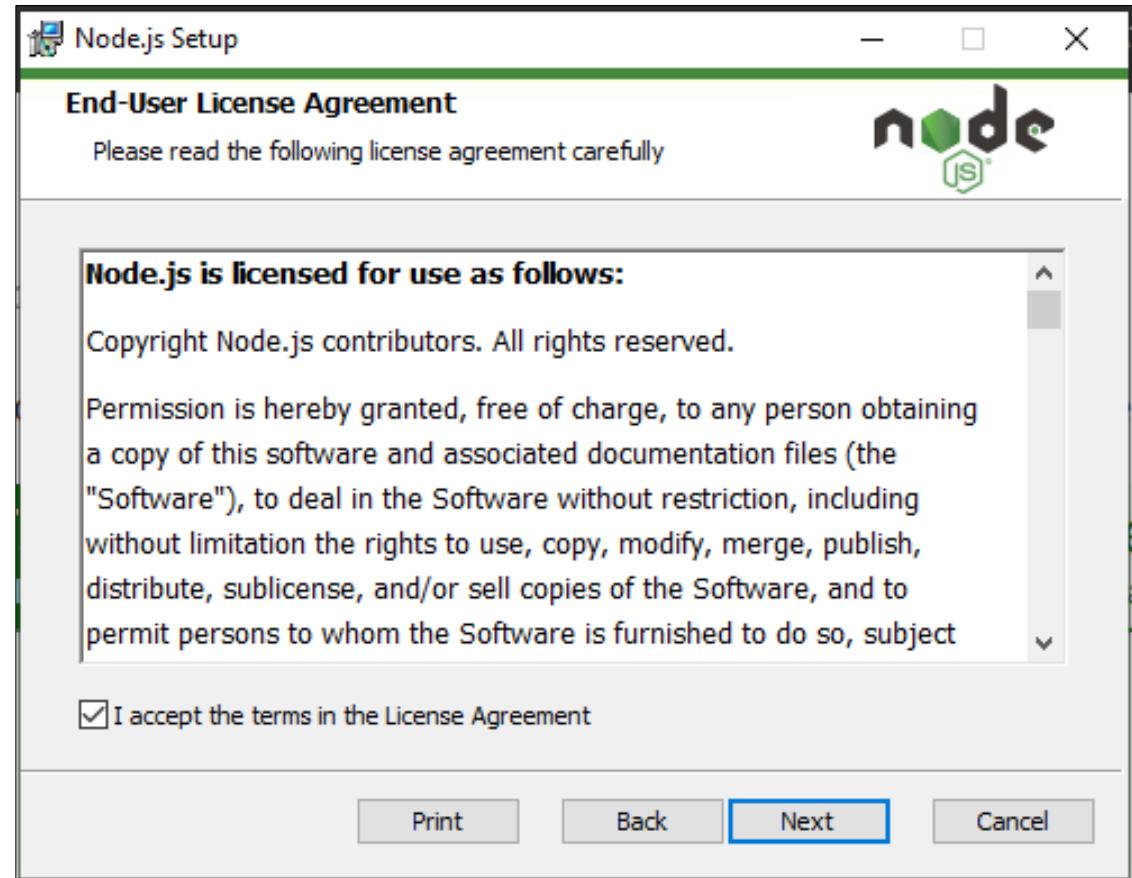
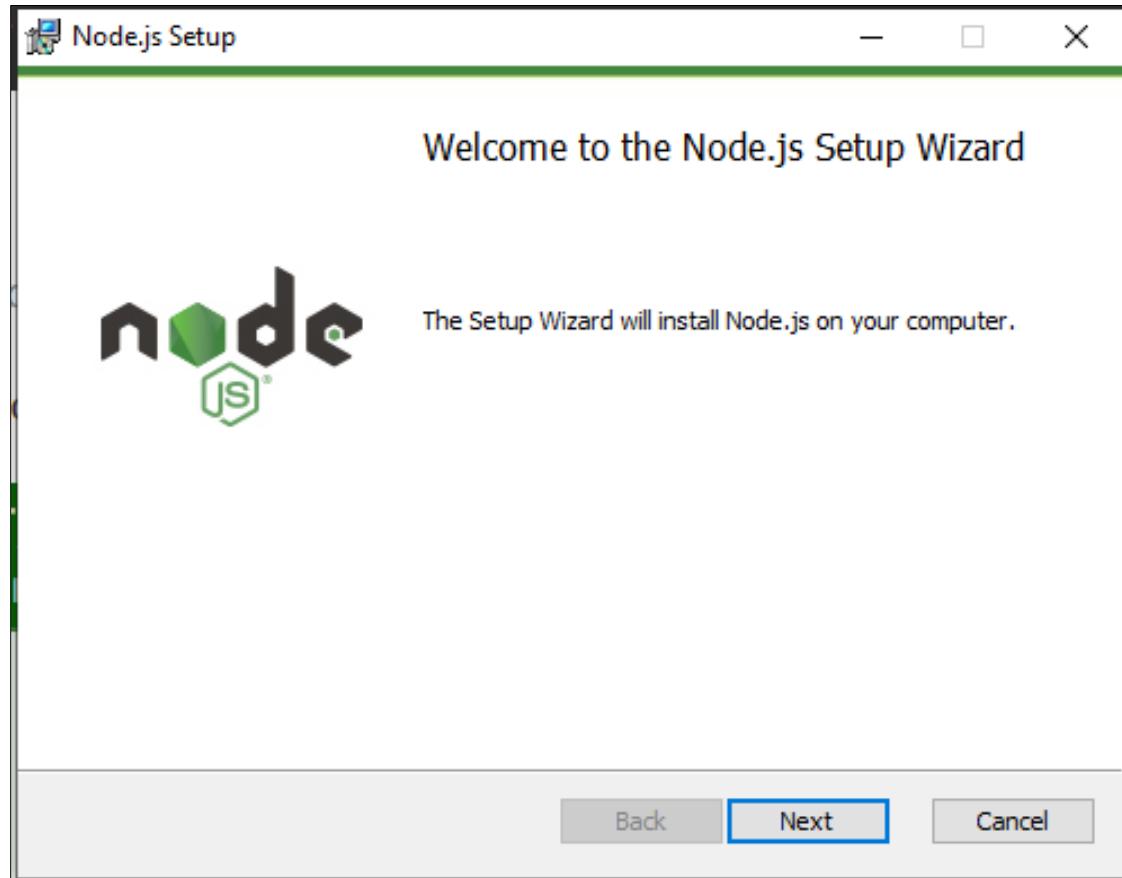
- ▶ to initialize, develop, and manage your Angular applications.

## Step 1: Install Node.js

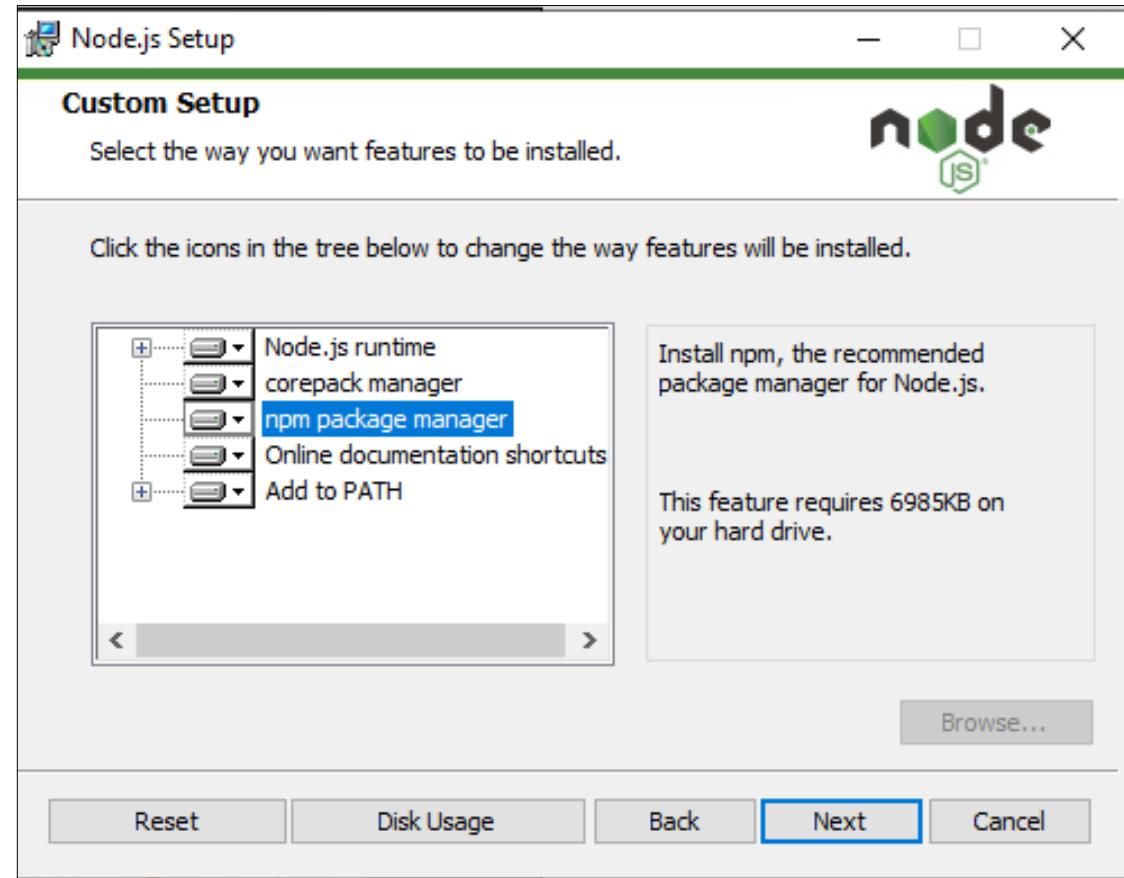
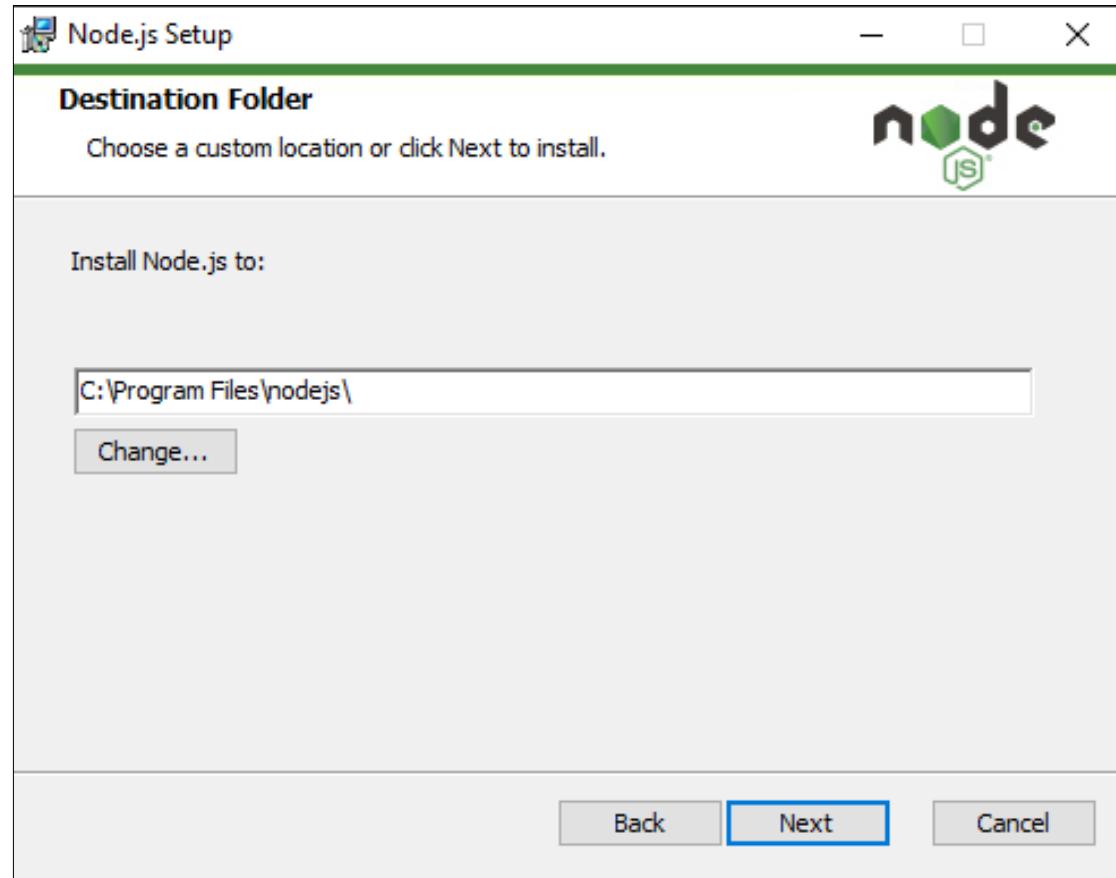
The screenshot shows the Node.js download page at nodejs.org/en/download/. The top navigation bar includes links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, and NEWS. The main content area is titled "Downloads" and features a message about the Latest LTS Version: 16.16.0 (includes npm 8.11.0). Below this, a call-to-action encourages users to "Download the Node.js source code or a pre-built installer for your platform, and start developing today." The page is divided into two main sections: "LTS" (Recommended For Most Users) and "Current" (Latest Features). Under the LTS section, there are links for "Windows Installer" (node-v16.16.0-x64.msi), "macOS Installer" (node-v16.16.0.pkg), and "Source Code" (node-v16.16.0.tar.gz). Under the Current section, there are links for "Windows Binary" (.zip) and "macOS Installer" (.pkg). A red box highlights the "Windows Installer (.msi)" link, which is further divided into 32-bit and 64-bit options. The "macOS Installer (.pkg)" link is also highlighted with a red box.

Platform	Architecture
Windows Installer (.msi)	32-bit
Windows Installer (.msi)	64-bit
Windows Binary (.zip)	32-bit
macOS Installer (.pkg)	64-bit / ARM64

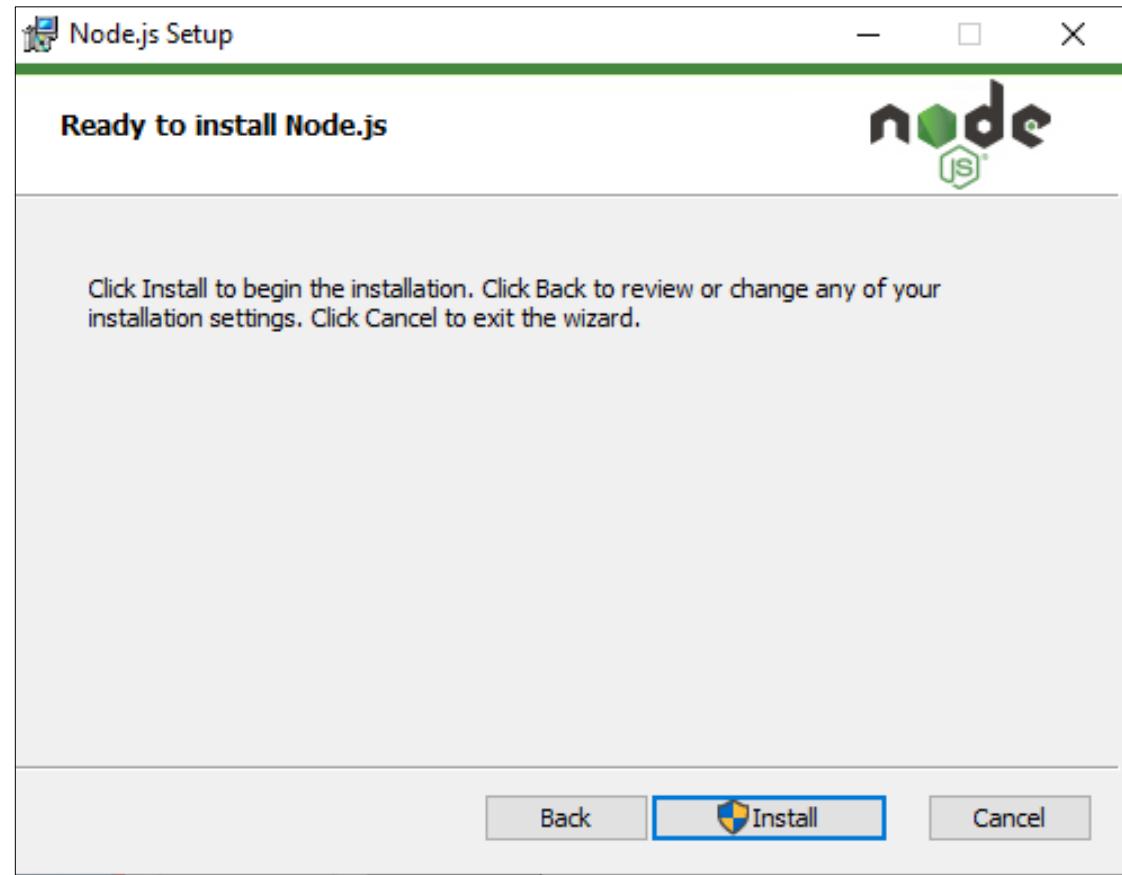
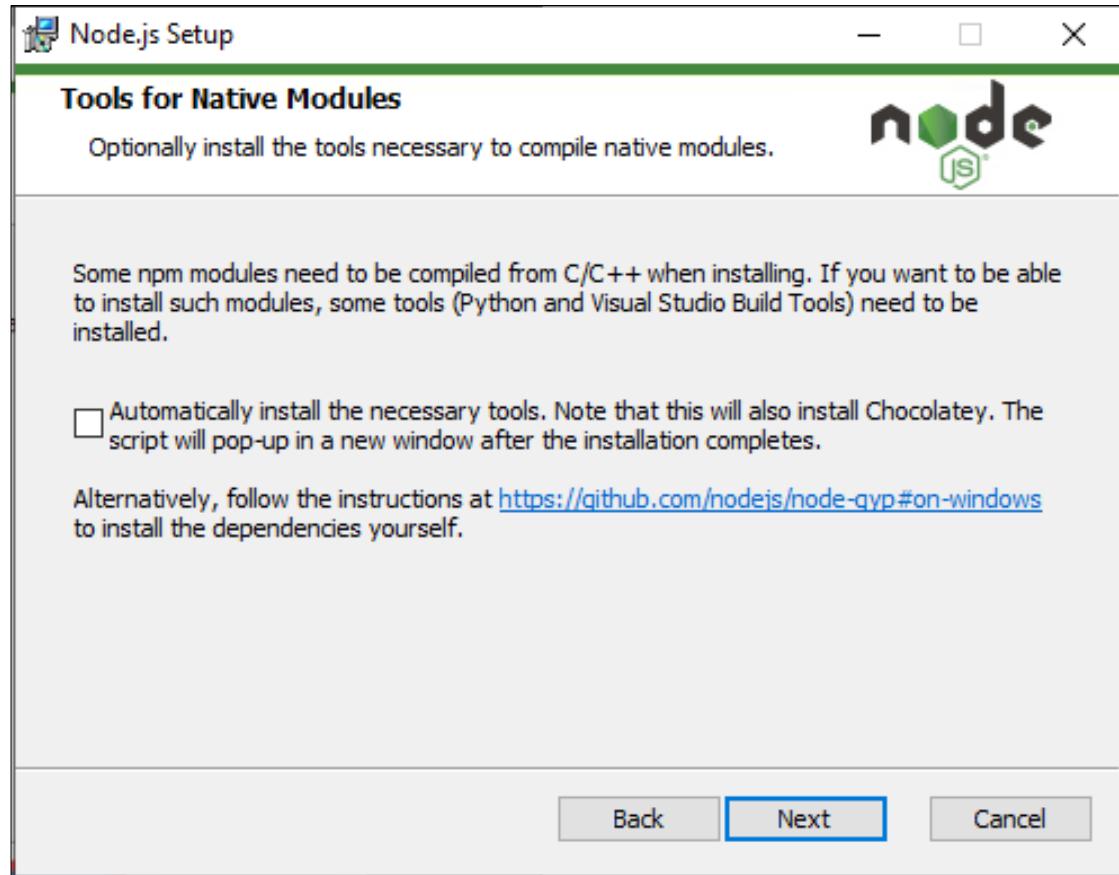
## Step 1: Install Node.js



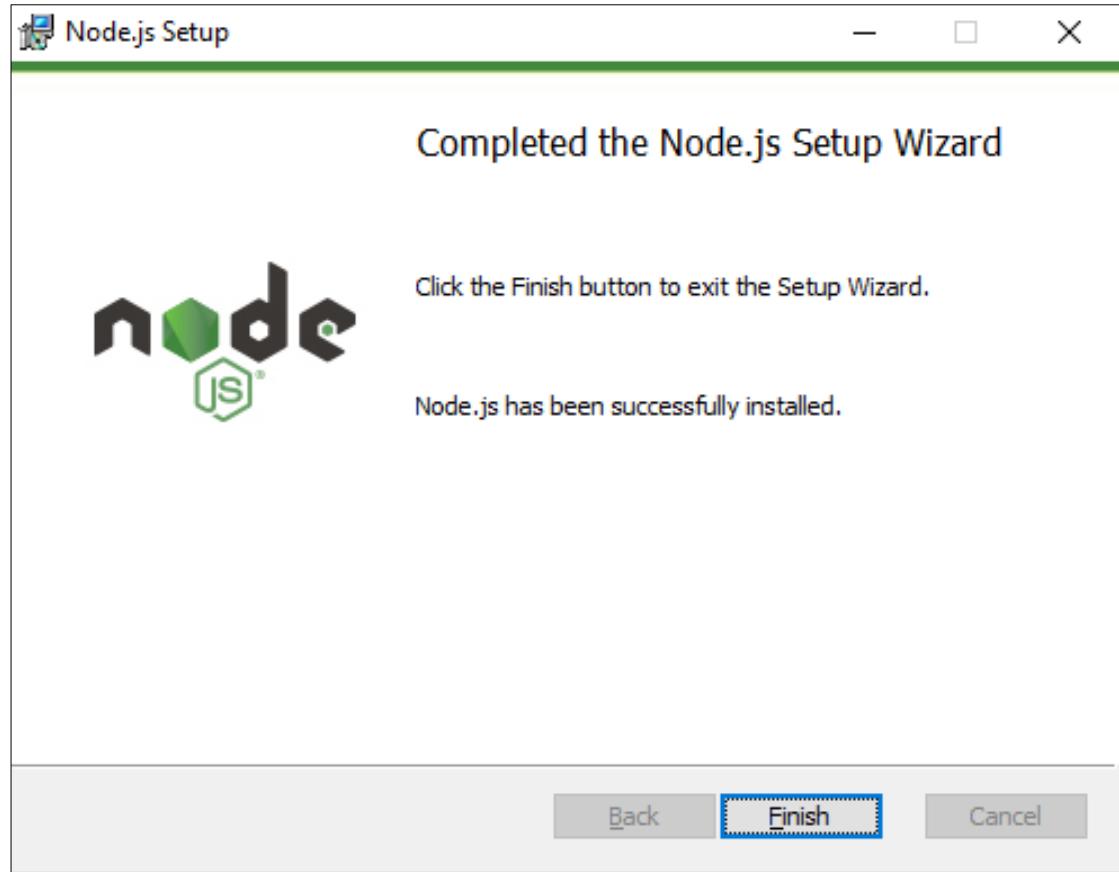
## Step 1: Install Node.js



## Step 1: Install Node.js



## Step 1: Install Node.js



The screenshot shows a Windows Command Prompt window titled "cmd" with the path "C:\WINDOWS\system32\cmd.exe". The output of the command "node -v" is "v16.16.0" and the output of "npm -v" is "8.16.0". The prompt at the end is "C:\Users\Zwei>".

```
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Zwei>node -v
v16.16.0

C:\Users\Zwei>npm -v
8.16.0

C:\Users\Zwei>
```

## Step 3: Install Angular CLI

```
C:\WINDOWS\system32\cmd.exe

C:\Users\Zwei>npm install -g @angular/cli@latest
changed 206 packages, and audited 207 packages in 2m
25 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```

## Step 3: Install Angular CLI

```
C:\WINDOWS\system32\cmd.exe
C:\Users\Zwei>ng version
? Would you like to share anonymous usage data about this project with the Angular Team at
Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. Yes

Thank you for sharing anonymous usage data. Should you change your mind, the following
command will disable this feature entirely:

  ng analytics disable --global

Global setting: enabled
Local setting: No local workspace configuration file.
Effective status: enabled

Angular CLI: 14.1.1
Node: 16.16.0
Package Manager: npm 8.16.0
OS: win32 x64

Angular:
  ...
  ...

  Package          Version
  -----
@angular-devkit/architect    0.1401.1 (cli-only)
@angular-devkit/core         14.1.1 (cli-only)
@angular-devkit/schematics   14.1.1 (cli-only)
@schematics/angular          14.1.1 (cli-only)
```

# Creating Angular Project using VS Code

```
> Angular_20220807
PS D:\AI_20200707\03_SelfCourse\07_Angular\AngularProjects> ng new Angular_20220807
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE Angular_20220807/angular.json (2972 bytes)
CREATE Angular_20220807/package.json (1047 bytes)
CREATE Angular_20220807/README.md (1069 bytes)
CREATE Angular_20220807/tsconfig.json (863 bytes)
CREATE Angular_20220807/.editorconfig (274 bytes)
CREATE Angular_20220807/.gitignore (548 bytes)
CREATE Angular_20220807/.browserslistrc (600 bytes)
CREATE Angular_20220807/karma.conf.js (1433 bytes)
CREATE Angular_20220807/tsconfig.app.json (287 bytes)
CREATE Angular_20220807/tsconfig.spec.json (333 bytes)
CREATE Angular_20220807/.vscode/extensions.json (130 bytes)
CREATE Angular_20220807/.vscode/launch.json (474 bytes)
CREATE Angular_20220807/.vscode/tasks.json (938 bytes)
CREATE Angular_20220807/src/favicon.ico (948 bytes)
CREATE Angular_20220807/src/index.html (301 bytes)
CREATE Angular_20220807/src/main.ts (372 bytes)
CREATE Angular_20220807/src/polyfills.ts (2338 bytes)
CREATE Angular_20220807/src/styles.css (80 bytes)
CREATE Angular_20220807/src/test.ts (749 bytes)
CREATE Angular_20220807/src/assets/.gitkeep (0 bytes)
CREATE Angular_20220807/src/environments/environment.prod.ts (51 bytes)
CREATE Angular_20220807/src/environments/environment.ts (658 bytes)
CREATE Angular_20220807/src/app/app-routing.module.ts (245 bytes)
CREATE Angular_20220807/src/app/app.module.ts (393 bytes)
CREATE Angular_20220807/src/app/app.component.html (23115 bytes)
CREATE Angular_20220807/src/app/app.component.spec.ts (1103 bytes)
CREATE Angular_20220807/src/app/app.component.ts (220 bytes)
CREATE Angular_20220807/src/app/app.component.css (0 bytes)
✓ Packages installed successfully.
'git' is not recognized as an internal or external command,
operable program or batch file.
PS D:\AI_20200707\03_SelfCourse\07_Angular\AngularProjects>
```

# Creating Angular Project using VS Code

Change the directory to project folder

```
PS D:\AI_20200707\03_SelfCourse\07_Angular\AngularProjects> cd Angular_20220807
```

## Compiling Project

```
PS D:\AI_20200707\03_SelfCourse\07_Angular\AngularProjects\Angular_20220807> ng serve
? Would you like to share anonymous usage data about this project with the Angular Team at
Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. No
Global setting: enabled
Local setting: disabled
Effective status: disabled
✓ Browser application bundle generation complete.

Initial Chunk Files | Names           | Raw Size
vendor.js           | vendor          | 2.04 MB
polyfills.js        | polyfills       | 315.35 kB
styles.css, styles.js | styles          | 207.38 kB
main.js             | main            | 50.19 kB
runtime.js          | runtime         | 6.53 kB

| Initial Total | 2.61 MB

Build at: 2022-08-06T18:49:10.581Z - Hash: b9cea07174df2217 - Time: 63370ms

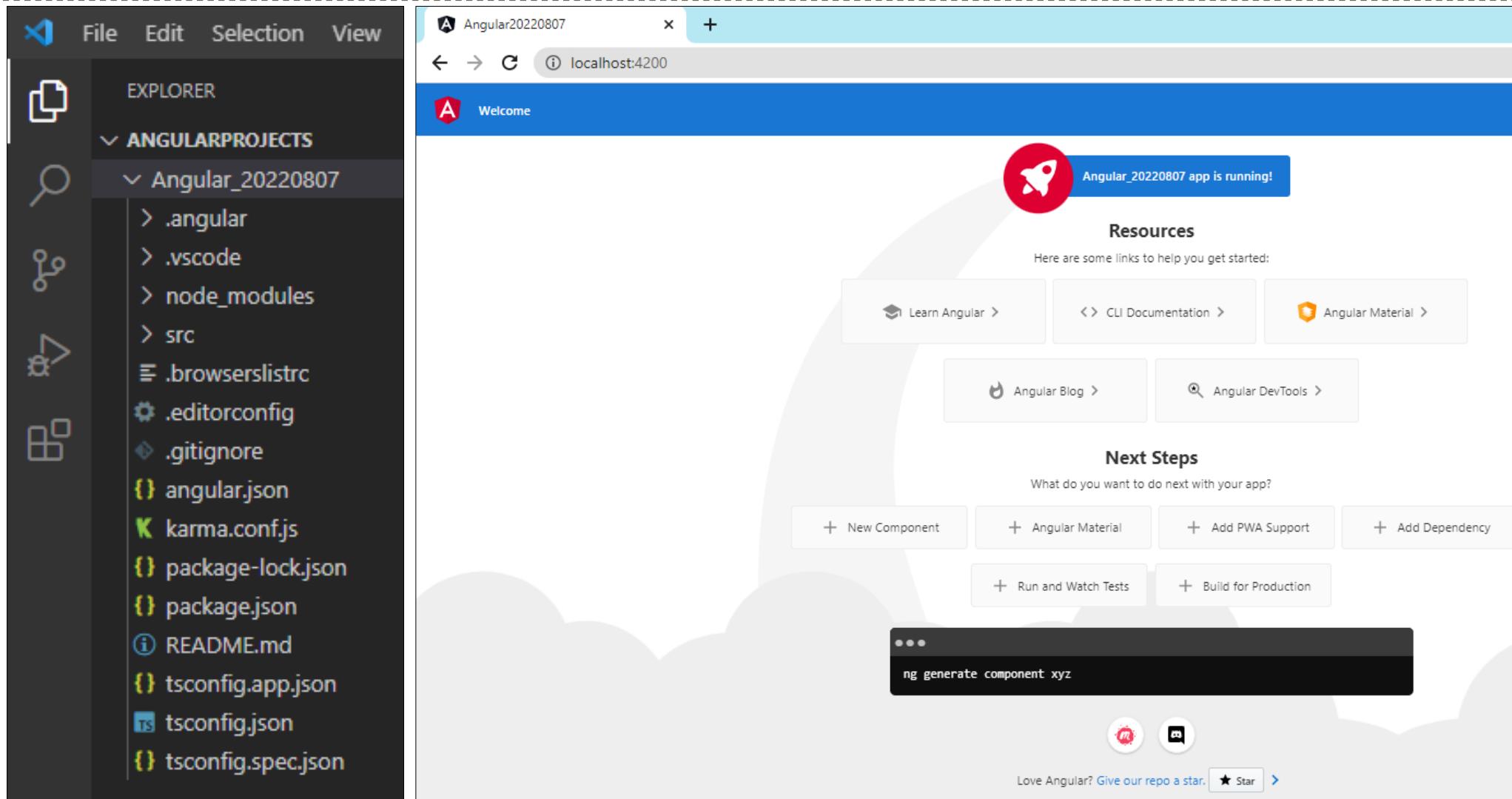
** Angular Live Development Server is listening on localhost:4200, open your browser on http://lo
calhost:4200/ **

✓ Compiled successfully.
```

The ng serve command will do two things

- build the angular application
- start the webserver

# Creating Angular Project using VS Code



# Understanding Angular

---

To understand the capabilities of the Angular framework,

- ▶ Components
- ▶ Templates
- ▶ Directives
- ▶ Dependency injection

# Component

---

The Component is the main building block of an Angular Application.

The Angular Components are plain JavaScript classes and defined using @Component Decorator.

The Component is responsible to provide the data to the view.

The Components consists of three main building blocks

- ▶ Template
- ▶ Class
- ▶ MetaData

# Component : Template(View)

---

The template defines the layout and content of the View.

## Component : Class

---

The Class provides the data & logic to the View.

TypeScript is used to create the class, but JavaScript can be used directly in the class.

Class Contains the Properties & Methods.

The Properties of a class can be bind to the view using Data Binding.

# Component : Metadata

---

Angular uses metadata to process the class.

@Component decorator is used to provide the Metadata to the Component.

## Important Component metadata properties

- ▶ Selector
- ▶ Providers
- ▶ Directives
- ▶ styleUrls
- ▶ templateUrl

# Component : app.component.ts

```
TS app.component.ts ×  
  
anglar2 > src > app > TS app.component.ts > ...  
1 import { Component } from '@angular/core';  
2  
3 @Component({  
4   selector: 'app-root',  
5   templateUrl: './app.component.html',  
6   styleUrls: ['./app.component.css']  
7 })  
8 export class AppComponent {  
9   title = 'anglar2';  
10 }  
11
```

# Template : app.component.html

In Angular, a template is a blueprint for a fragment of a user interface (UI).

Templates are written in HTML, and special syntax can be used within a template to build on many of Angular's features.

```
↳ app.component.html X
angular2 > src > app > ↳ app.component.html > ⚙ div.content
344   <span>{{ title }} app is running!</span>
345
346   <svg id="rocket-smoke" xmlns="http://www.w3.org/2000/svg" w...
347   |   <title>Rocket Ship Smoke</title>
348   |   <path id="Path_40" data-name="Path 40" d="M644.6,1415143.0...
349   |   </svg>
350
351   </div>
352
```

# Angular Module : app.module.ts

```
TS app.module.ts X

angular2 > src > app > TS app.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { AppRoutingModule } from './app-routing.module';
4  import { AppComponent } from './app.component';
5
6  @NgModule({
7    declarations: [
8      AppComponent
9    ],
10   imports: [
11     BrowserModule,
12     AppRoutingModule
13   ],
14   providers: [],
15   bootstrap: [AppComponent]
16 })
17 export class AppModule { }
```

The Angular Module organizes the components, directives, pipes, and services that are related and arrange them into cohesive blocks of functionality.

# Directives

---

The Angular directive helps us to manipulate the DOM.

The appearance, behavior, or layout of a DOM element can be changed by using the Directives.

There are three kinds of directives in Angular:

- ▶ Component Directive
- ▶ Structural directives
- ▶ Attribute directives

# Directives : Structural

---

Structural directives are responsible for the Structure and Layout of the DOM Element.

It is used to hide or display the things on the DOM.

Structural Directives can be easily identified using the '\*'.

Some Directives are as follows:

- ▶ \*ngIf
- ▶ \*ngFor
- ▶ \*ngSwitch

# Directives : Attribute

---

Attribute directives listen to and modify the behavior of other HTML elements, attributes, properties, and components.

Some Directives are as follows:

- ▶ NgClass : Adds and removes a set of CSS classes.
- ▶ NgStyle : Adds and removes a set of HTML styles.
- ▶ NgModel : Adds two-way data binding to an HTML form element.

# Some Events in Angular

Event name	Description
(click)	The click event occurs when an element is clicked.
(change)	The change event is triggered when change occurs on the binding elements, like select, Textarea, input, and other elements.
(dblclick)	The double-click event occurs when an element is clicked two times.
(blur)	The blur event fires when an element has lost focus.
(submit)	The submit event fire when clicking on the button or inputting with type submit.

## Example : Person List

Output:

The image displays two side-by-side browser windows, both titled "Angular2". The left window shows a simple "Person List" page at "localhost:4200". The right window shows a more complex "Person List" page at "localhost:4200/person", which includes filtering logic based on address.

**Person List (localhost:4200):**

Person List

**Person List (localhost:4200/person):**

Person Information based on Conditions

Person List

Name	Age	Address
Mg Mg	20	YGN
Su Su	23	YGN
Ag Ag	25	MDY

**Person List (Address = YGN)**

Mg Mg - 20 - YGN

Su Su - 23 - YGN

**Person List (Address = YGN) using ng-template**

Mg Mg - 20 - YGN

Su Su - 23 - YGN

## Example : Person List

book folder

```
✓ person
  # person.component.css
  ↗ person.component.html
  TS person.component.spec.ts
  TS person.component.ts
  TS person.spec.ts
  TS person.ts
```

book.ts

```
TS person.ts  X
angular2 > src > app > person > TS person.ts > ...
1   export class Person {
2     name! : string;
3     age! : number;
4     address! : string;
5
6   }
7   |
```

# Example : Person List

## book.component.ts

```
TS person.component.ts X
angular2 > src > app > person > TS person.component.ts > PersonComponent
1 import { Component, OnInit } from '@angular/core';
2 import { Person } from './person';
3
4 @Component({
5   selector: 'app-person',
6   templateUrl: './person.component.html',
7   styleUrls: ['./person.component.css']
8 })
9 export class PersonComponent implements OnInit {
10   title: string = "Person Information based on Conditions" ;
11
12   personList: Person[] =[ 
13     {name: 'Mg Mg',age : 20, address:'YGN'},
14     {name: 'Su Su',age : 23, address:'YGN'},
15     {name: 'Ag Ag',age : 25, address:'MDY'}
16   ]
17
18   constructor() { }
19
20   ngOnInit(): void {
21   }
22 }
23
```

# Example : Person List

book.component.html

```
↳ person.component.html ●  
  
angular2 > src > app > person > ↳ person.component.html > ⏺ h2  
1   |

## {{title}} </h2> 2 |<h3>Person List</h3> 3 |<table> 4 | |<thead> 5 | ||<tr> 6 | |||<th> Name</th> 7 | |||<th> Age </th> 8 | |||<th> Address</th> 9 | |</tr> 10 |</thead> 11 |<tbody> 12 | |<tr *ngFor = "let person of personList" > 13 | ||<td> {{ person.name }} </td> 14 | ||<td> {{ person.age }} </td> 15 | ||<td> {{ person.address}} </td> 16 | |</tr> 17 |</tbody> 18 |</table>


```

```
34  |<h3>Person List (Address = YGN)</h3>  
35  |  |<div *ngFor="let person of personList">  
36  |  ||<p *ngIf="person.address=='YGN'">  
37  |  ||| {{ person.name }} - {{person.age}} - {{person.address}}  
38  |  ||</p>  
39  |  |</div>  
40  |<br>  
41  |<h3>Person List (Address = YGN) using ng-template</h3>  
42  |  |<div *ngFor="let person of personList">  
43  |  ||<ng-template [ngIf]= "person.address=='YGN'">  
44  |  |||<div> {{ person.name }} - {{person.age}} - {{person.address}}</div>  
45  |  ||</ng-template>  
46  |</div>
```

# Example : Person List

app-routing.module.ts

```
TS app-routing.module.ts X

angular2 > src > app > TS app-routing.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { PersonComponent } from './person/person.component';
4
5 const routes: Routes = [
6   {path: 'person', component: PersonComponent}
7 ];
8
9 @NgModule({
10   imports: [RouterModule.forRoot(routes)],
11   exports: [RouterModule]
12 })
13 export class AppRoutingModule { }
```

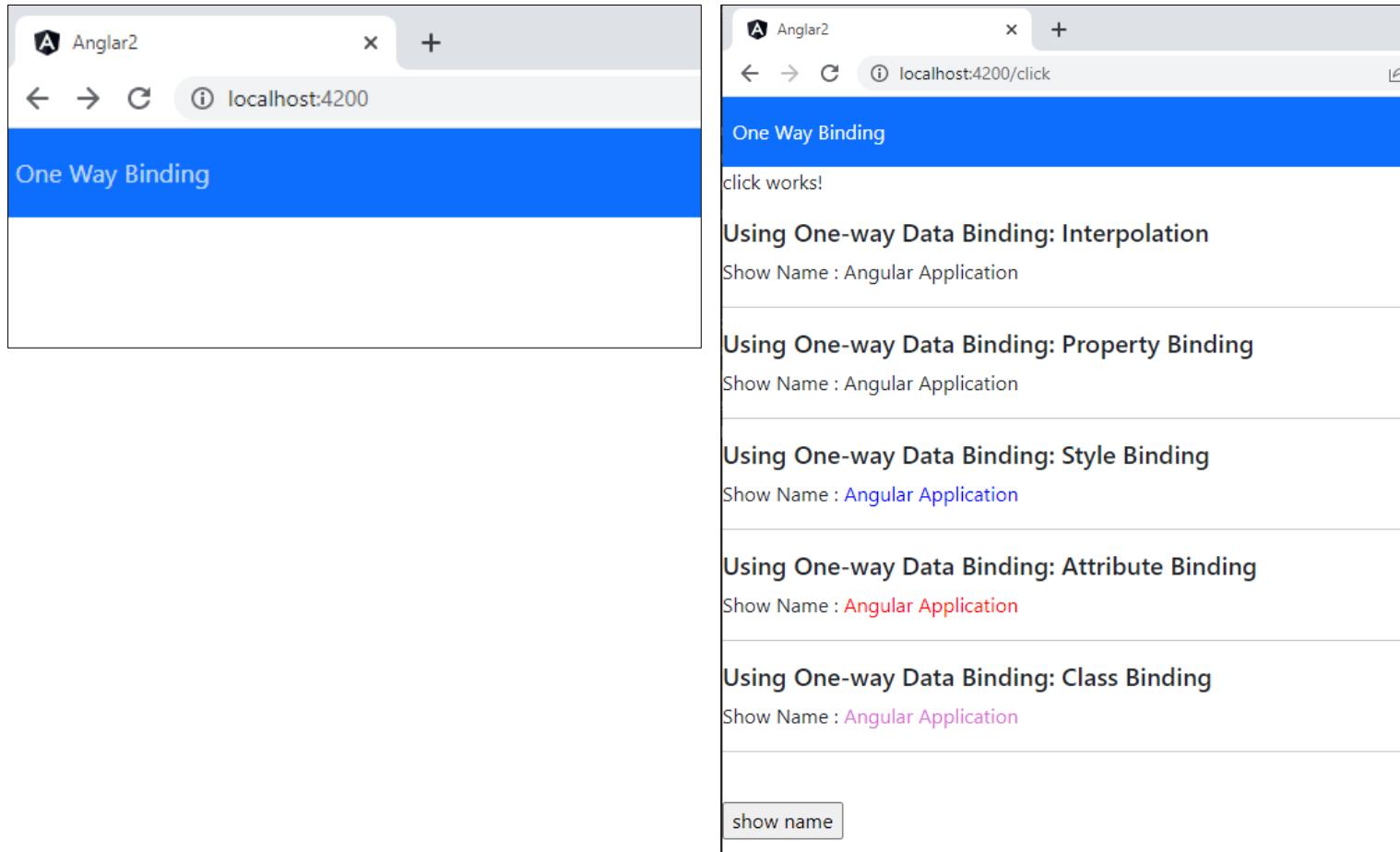
# Example : Person List

## app.module.ts

```
TS app.module.ts X
angular2 > src > app > TS app.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { AppRoutingModule } from './app-routing.module';
4  import { AppComponent } from './app.component';
5  import { PersonComponent } from './person/person.component';
6
7  @NgModule({
8    declarations: [
9      AppComponent,
10     PersonComponent
11   ],
12   imports: [
13     BrowserModule,
14     AppRoutingModule
15   ],
16   providers: [],
17   bootstrap: [AppComponent]
18 })
19 export class AppModule { }
```

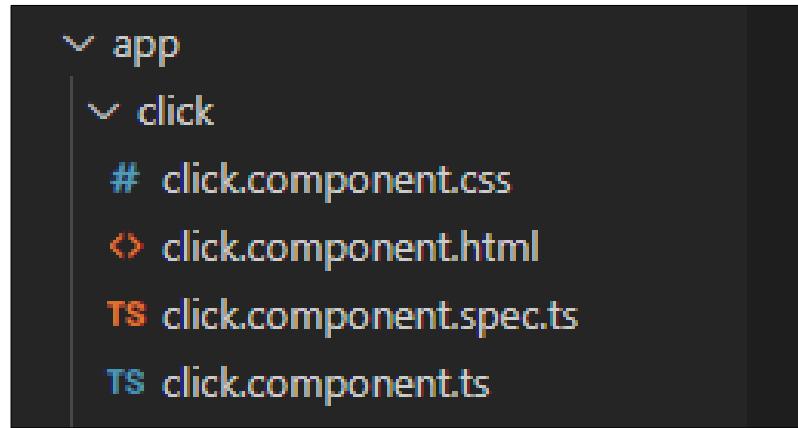
## Example : One-way Binding(DataSource-View)

Output:



# Example : One-way Binding(DataSource-View)

click folder



click.component.ts

```
TS click.component.ts X
angular2 > src > app > click > TS click.component.ts > ClickComponent > 1
  6   styleUrls: ['./click.component.css']
  7 }
  8 export class ClickComponent implements OnInit {
  9   name: string = "";
10
11  constructor() { }
12
13  ngOnInit(): void {
14    }
15
16  changeName(): void{
17    this.name ="Angular Application";
18  }
19
20 }
```

## Example : One-way Binding(DataSource-View)

click.component.html

```
<> click.component.html < />

angular2 > src > app > click > <> click.component.html > h5
1   <p>click works!</p>
2
3   <h5>Using One-way Data Binding: Interpolation</h5>
4   Show Name : {{name}}
5   <hr>
6   <h5>Using One-way Data Binding: Property Binding</h5>
7   Show Name : <span [innerHTML]='name'></span>
8   <hr>
9   <h5>Using One-way Data Binding: Style Binding</h5>
10  Show Name : <span [style.color]="'blue'">{{name}}</span>
11  <hr>
12  <h5>Using One-way Data Binding: Attribute Binding</h5>
13  Show Name : <font [attr.color]="'red'">{{name}}</font>
14  <hr>
15  <h5>Using One-way Data Binding: Class Binding</h5>
16  Show Name : <font class="colorClass">{{name}}</font>
17  <hr>
18  <br>
19  <input type="button" value="show name" (click)="changeName()">
```

styles.css

```
# styles.css < />

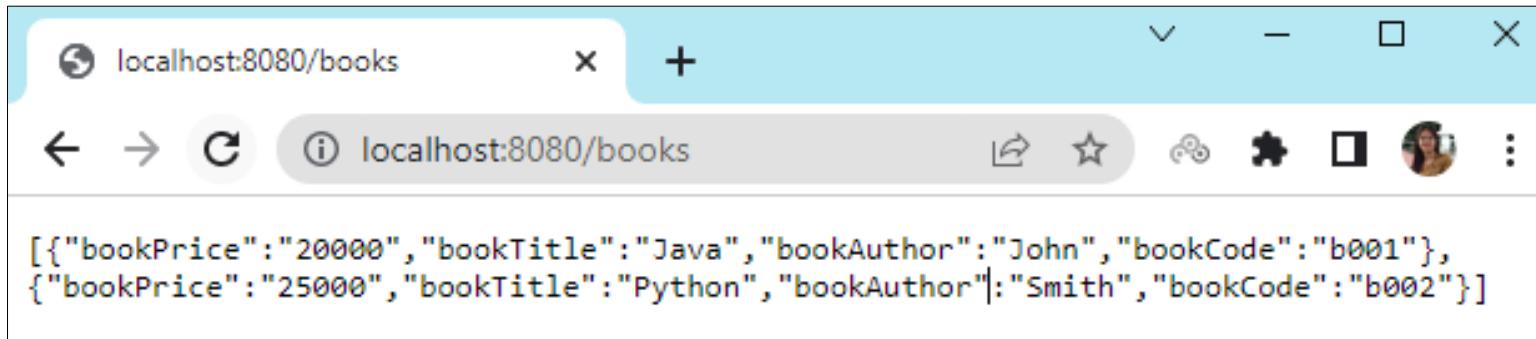
angular2 > src > # styles.css > .colorClass
1   /* You can add global styles to this
2   .boldClass{
3     font-weight:bold;
4   }
5   .italicClass{
6     font-style:italic;
7   }
8   .colorClass{
9     color:orchid;
10 }
```

index.html

```
<link rel="icon" type="image/x-icon" href="favicon.ico">
<link rel="icon" type="text/css" href="/angular2/src/styles.css">
```

# Spring Boot + Angular First Application

Backend Output:



A screenshot of a web browser window. The address bar shows 'localhost:8080/books'. The main content area displays a JSON array of book data:

```
[{"bookPrice": "20000", "bookTitle": "Java", "bookAuthor": "John", "bookCode": "b001"}, {"bookPrice": "25000", "bookTitle": "Python", "bookAuthor": "Smith", "bookCode": "b002"}]
```

# Spring Boot + Angular First Application

## BookBean.java

```
package com.book.model;

public class BookBean {

    private String BookCode;
    private String BookTitle;
    private String BookAuthor;
    private String BookPrice;

    // getters, setters
}
```

# Spring Boot + Angular First Application

## BookBean.java

```
package com.book.model;

public class BookBean {

    private String BookCode;
    private String BookTitle;
    private String BookAuthor;
    private String BookPrice;

    // getters, setters
}
```

# Spring Boot + Angular First Application

## BookController.java

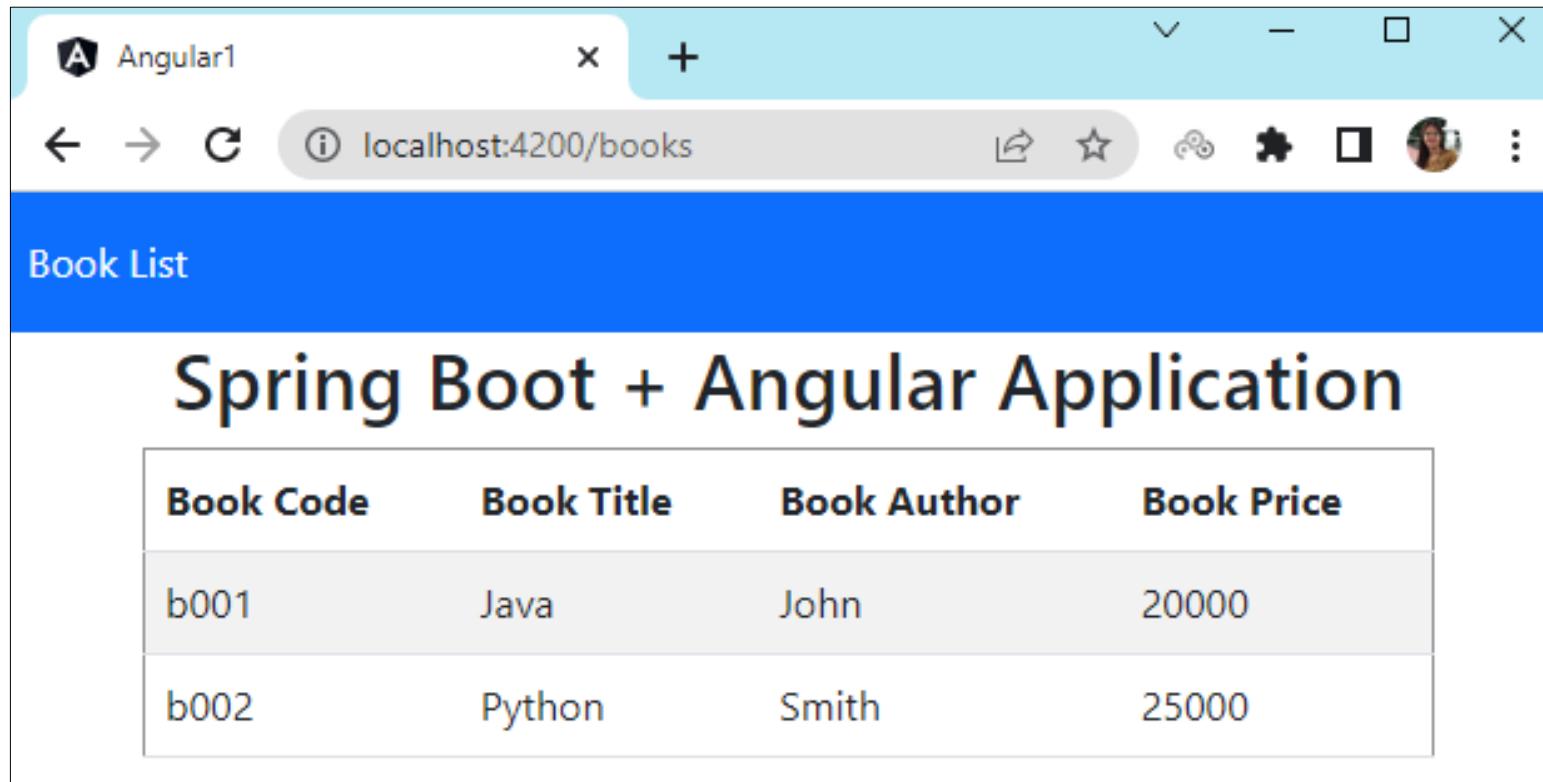
```
package com.book.controller;  
  
//Import statements  
  
@CrossOrigin(origins = "http://localhost:4200")  
  
@RestController  
  
public class BookController {  
  
    private List<BookBean> bookList = createList();  
  
    @RequestMapping(value = "/books", method = RequestMethod.GET, produces = "application/json")  
  
    public List<BookBean> firstPage() {  
  
        return bookList;  
    }  
}
```

# Spring Boot + Angular First Application

```
private static List<BookBean> createList() {  
    List<BookBean> bookList = new ArrayList<BookBean>();  
    BookBean book1 = new BookBean();  
    book1.setBookCode("b001");  
    book1.setBookTitle("Java");  
    book1.setBookAuthor("John");  
    book1.setBookPrice("20000");  
  
    BookBean book2 = new BookBean();  
    book2.setBookCode("b002");  
    book2.setBookTitle("Python");  
    book2.setBookAuthor("Smith");  
    book2.setBookPrice("25000");  
    bookList.add(book1);  
    bookList.add(book2);  
    return bookList;  
}  
}
```

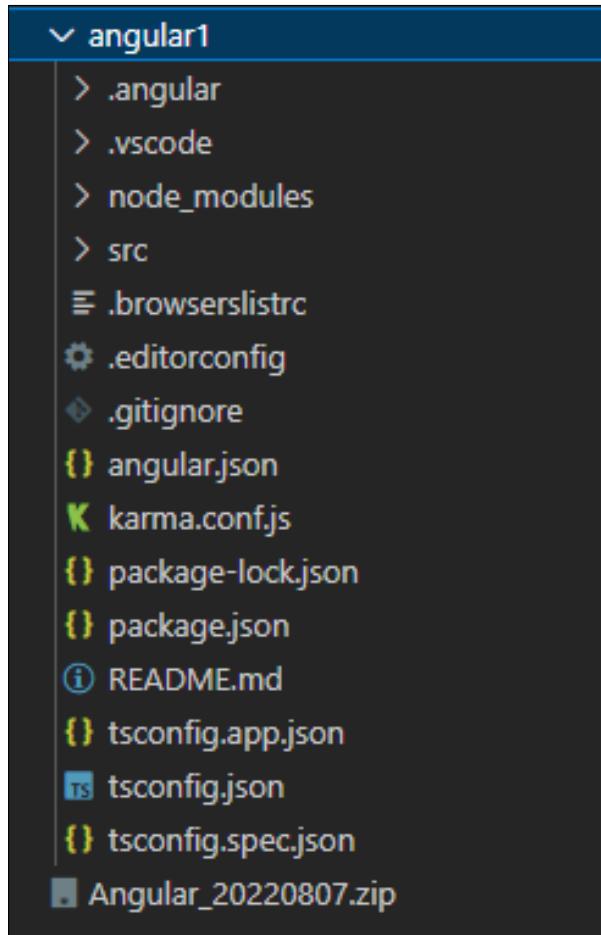
# Spring Boot + Angular First Application

Frontend Output:



# Spring Boot + Angular First Application

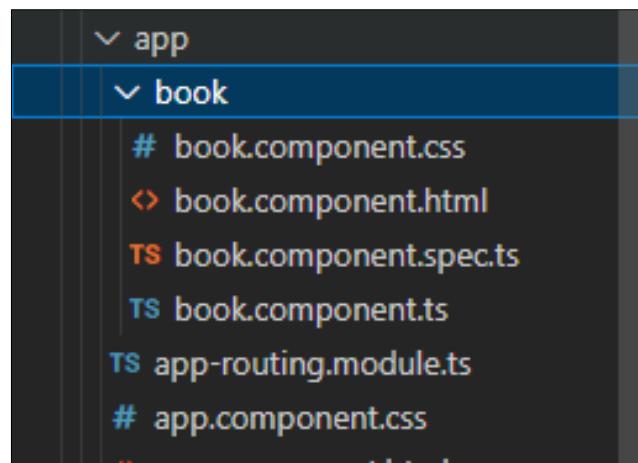
Front-end Angular project structure:



# Spring Boot + Angular First Application

Create Book component:

```
PS D:\VAI_20200707\03_SelfCourse\07_Angular\AngularProjects> cd angular1
PS D:\VAI_20200707\03_SelfCourse\07_Angular\AngularProjects\angular1> ng generate component book
CREATE src/app/book/book.component.html (19 bytes)
CREATE src/app/book/book.component.spec.ts (585 bytes)
CREATE src/app/book/book.component.ts (267 bytes)
CREATE src/app/book/book.component.css (0 bytes)
UPDATE src/app/app.module.ts (467 bytes)
PS D:\VAI_20200707\03_SelfCourse\07_Angular\AngularProjects\angular1> []
```



# Spring Boot + Angular First Application

book.component.html

```
<table border="1" class="table table-striped">
<thead>
<tr>
<th>Book Code</th>
<th>Book Title</th>
<th>Book Author</th>
<th>Book Price</th>
</tr>
</thead>
<tbody>
<tr *ngFor="let book of books">
<td>{{book.bookCode}}</td>
<td>{{book.bookTitle}}</td>
<td>{{book.bookAuthor}}</td>
<td>{{book.bookPrice}}</td>
</tr>
</tbody>
</table>
```

# Spring Boot + Angular First Application

Create Book service:

```
PS D:\AI_20200707\03_SelfCourse\07_Angular\AngularProjects\angular1> ng generate service book
CREATE src/app/book.service.spec.ts (347 bytes)
CREATE src/app/book.service.ts (133 bytes)
```

# Spring Boot + Angular First Application

## book.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Book } from './book';

@Injectable({
  providedIn: 'root'
})
export class BookService {
  private baseURL = "http://localhost:8080/books";
  constructor(private httpClient: HttpClient) { }

  getBookList(): Observable<Book[]>{
    return this.httpClient.get<Book[]>(` ${this.baseURL}`);
  }
}
```

# Spring Boot + Angular First Application

Create Book model:

```
PS D:\AI_20200707\03_SelfCourse\07_Angular\AngularProjects\angular1> ng generate class book
CREATE src/app/book.spec.ts (146 bytes)
CREATE src/app/book.ts (22 bytes)
PS D:\AI_20200707\03_SelfCourse\07_Angular\AngularProjects\angular1> 
```

book.ts

```
export class Book{
    bookCode!: string
    bookTitle!: string;
    bookAuthor!: string;
    bookPrice!: string;
}
```

# Spring Boot + Angular First Application

app.component.html

```
<nav class="navbar navbar-expand-sm bg-primary navbar-dark">
  <ul class = "navbar-nav">
    <li class = "nav-item">
      <a routerLink="books" routerLinkActive="active" class="nav-link" >Book List</a>
    </li>

  </ul>
</nav>

<h1 class="text-center"> {{title}} </h1>
<div class = "container">
  <router-outlet></router-outlet>
</div>
```

# Spring Boot + Angular First Application

## app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BookComponent } from './book/book.component';
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent,
    BookComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

# Spring Boot + Angular First Application

## app-routing.module.ts

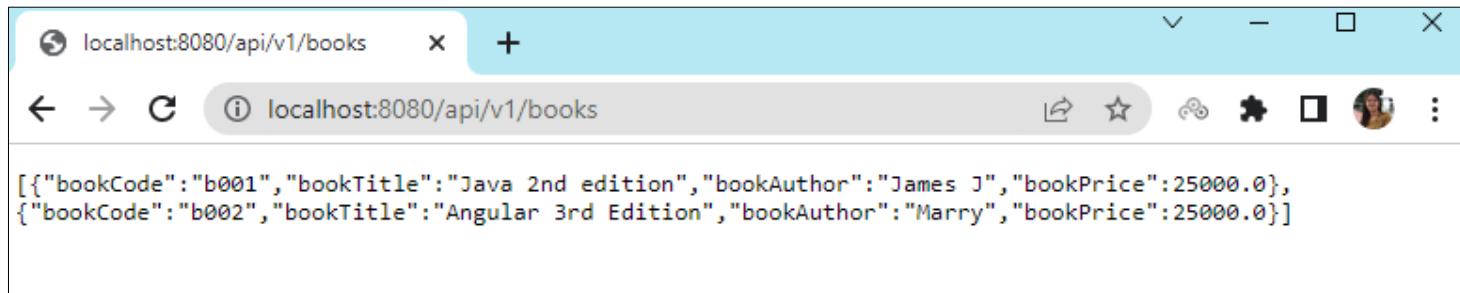
```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { BookComponent } from './book/book.component';

const routes: Routes = [
  { path: 'books', component: BookComponent
}];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

# BookManagement Project

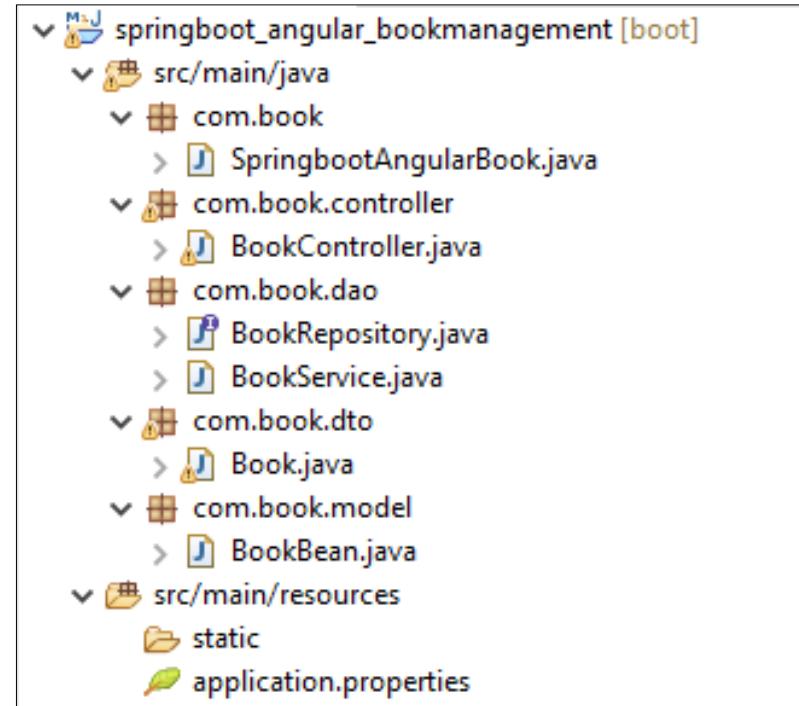
## Backend Output:



A screenshot of a web browser window. The address bar shows "localhost:8080/api/v1/books". The main content area displays the following JSON response:

```
[{"bookCode": "b001", "bookTitle": "Java 2nd edition", "bookAuthor": "James J", "bookPrice": 25000.0}, {"bookCode": "b002", "bookTitle": "Angular 3rd Edition", "bookAuthor": "Marry", "bookPrice": 25000.0}]
```

## Project Structure:



# Spring Boot + Angular First Application

## BookBean.java

```
package com.book.model;

public class BookBean {

    private String BookCode;
    private String BookTitle;
    private String BookAuthor;
    private String BookPrice;

    // getters, setters
}
```

# Spring Boot + Angular First Application

## BookController.java

```
package com.book.controller;  
  
//Import statements  
@CrossOrigin(origins = "http://localhost:4200")  
@RestController  
@RequestMapping("/api/v1/")  
public class BookController {  
  
    @Autowired  
    private BookService bookService;  
  
    @GetMapping(value="/books",produces = "application/json")  
    public List<Book> displayBook(ModelMap model) {  
        return bookService.getAllBook();  
    }  
}
```

# Spring Boot + Angular First Application

```
@PostMapping(value="/books",produces = "application/json")
public Book addbook(@RequestBody Book book) {
    Book dto = new Book();
    dto.setBookCode(book.getBookCode());
    dto.setBookTitle(book.getBookTitle());
    dto.setBookAuthor(book.getBookAuthor());
    dto.setBookPrice(Double.valueOf(book.getBookPrice()));
    return bookService.saveBook(dto);
}

// get employee by id rest api
@GetMapping("/books/{bookCode}")
public ResponseEntity<Book> getBookByBookCode(@PathVariable String bookCode) {
    Book book = new Book();
    book = bookService.getBookByBookCode(bookCode);
    return ResponseEntity.ok(book);
}
```

# Spring Boot + Angular First Application

```
// update employee rest api
@PutMapping(value="/books/{bookCode}",produces = "application/json")
public ResponseEntity<Book> updateBook(@PathVariable String bookCode, @RequestBody Book bookDetails) {
    Book book = new Book();
    book = bookService.getBookByBookCode(bookCode);
    book.setBookTitle(bookDetails.getBookTitle());
    book.setBookAuthor(bookDetails.getBookAuthor());
    book.setBookPrice(bookDetails.getBookPrice());
    Book updatedBook = bookService.update(book, book.getBookCode());
    return ResponseEntity.ok(updatedBook);
}
```

# Spring Boot + Angular First Application

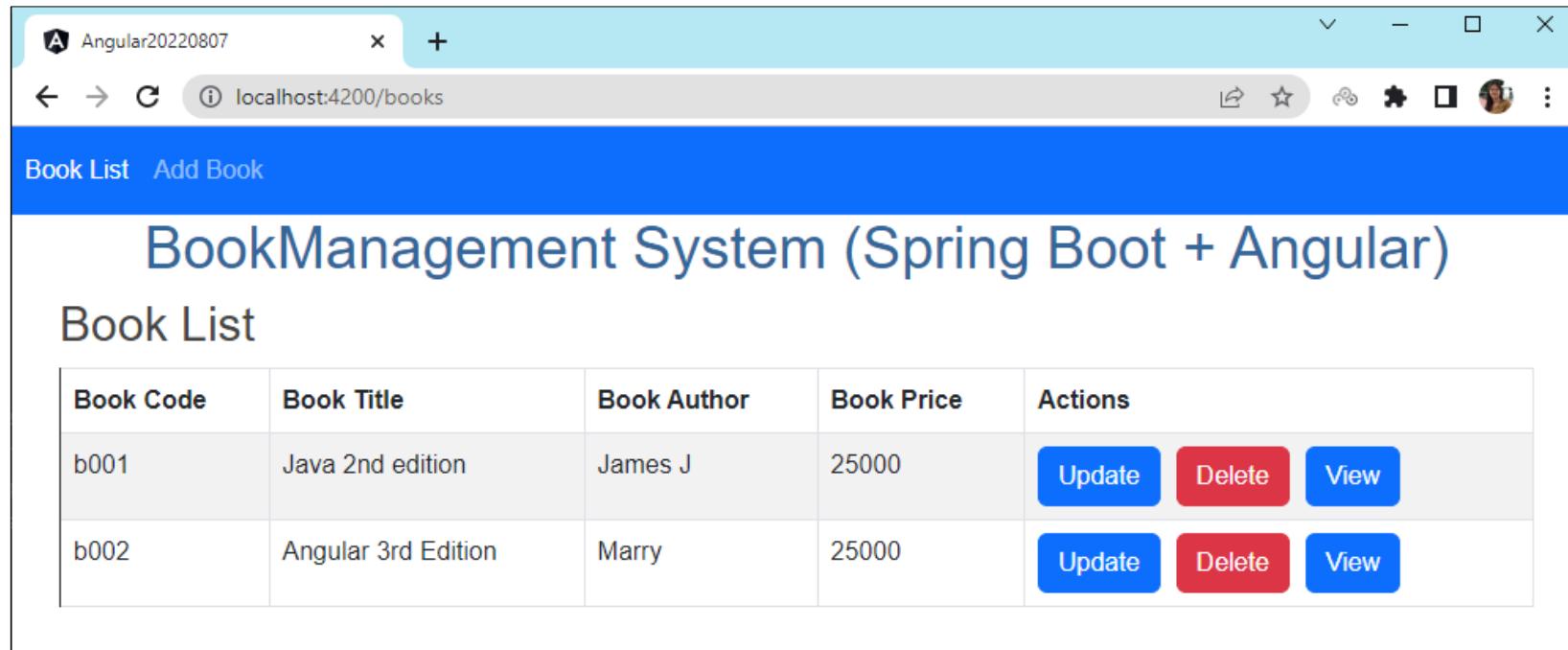
```
// delete employee rest api
@DeleteMapping("/books/{bookCode}")
public ResponseEntity<Map<String, Boolean>> deleteBook(@PathVariable String bookCode) {

    bookService.delete(bookCode);
    Map<String, Boolean> response = new HashMap<>();
    response.put("deleted", Boolean.TRUE);
    return ResponseEntity.ok(response);
}

}
```

# BookManagement Project

Frontend Output: Book List

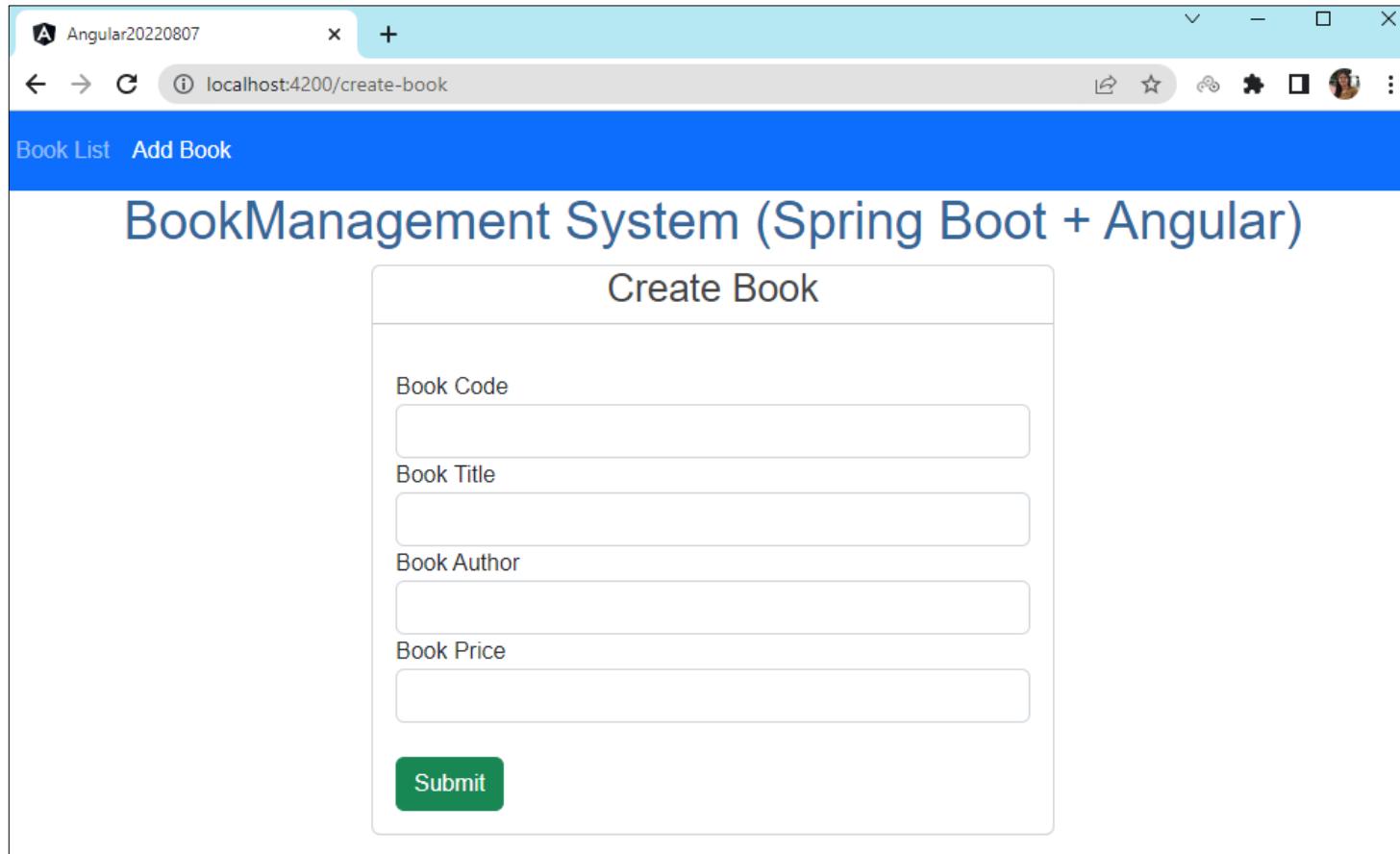


The screenshot shows a web browser window titled "Angular20220807" with the URL "localhost:4200/books". The page has a blue header with the text "Book List" and "Add Book". Below the header, the title "BookManagement System (Spring Boot + Angular)" is displayed in a large, bold font. The main content area is titled "Book List" and contains a table with two rows of book data. The table has columns for Book Code, Book Title, Book Author, Book Price, and Actions. The first row contains the book "Java 2nd edition" by James J. with a price of 25000. The second row contains the book "Angular 3rd Edition" by Marry with a price of 25000. Each row has three action buttons: "Update" (blue), "Delete" (red), and "View" (blue).

Book Code	Book Title	Book Author	Book Price	Actions
b001	Java 2nd edition	James J	25000	<a href="#">Update</a> <a href="#">Delete</a> <a href="#">View</a>
b002	Angular 3rd Edition	Marry	25000	<a href="#">Update</a> <a href="#">Delete</a> <a href="#">View</a>

# BookManagement Project

## Frontend Output: Create Book



A screenshot of a web browser window titled "Angular20220807". The address bar shows "localhost:4200/create-book". The page content is a blue header with "Book List" and "Add Book" buttons, followed by the title "BookManagement System (Spring Boot + Angular)". Below this is a "Create Book" form with four input fields: "Book Code", "Book Title", "Book Author", and "Book Price", each with a corresponding text input field. A green "Submit" button is at the bottom of the form.

Angular20220807

localhost:4200/create-book

Book List Add Book

BookManagement System (Spring Boot + Angular)

Create Book

Book Code

Book Title

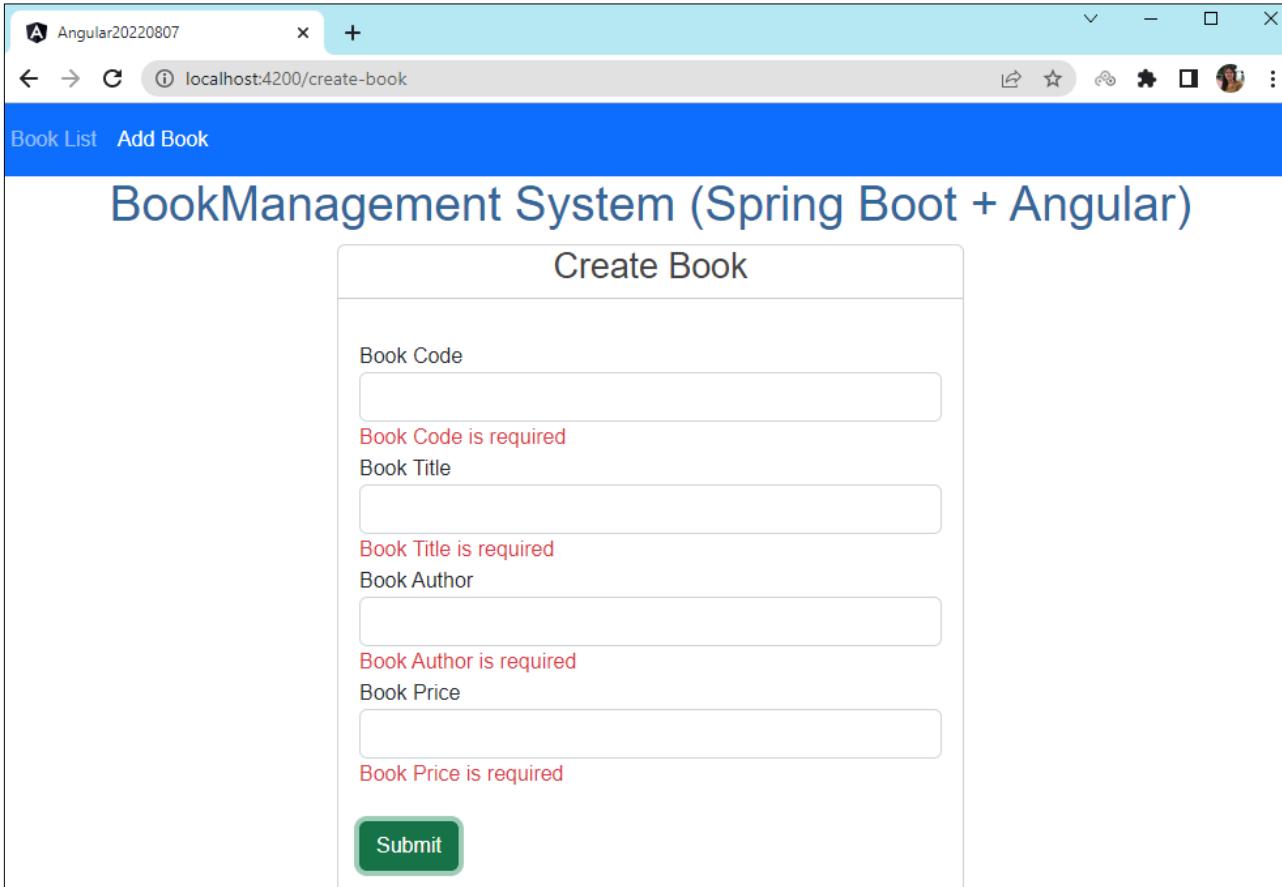
Book Author

Book Price

Submit

# BookManagement Project

Frontend Output: Create Book – Template Driven Validation

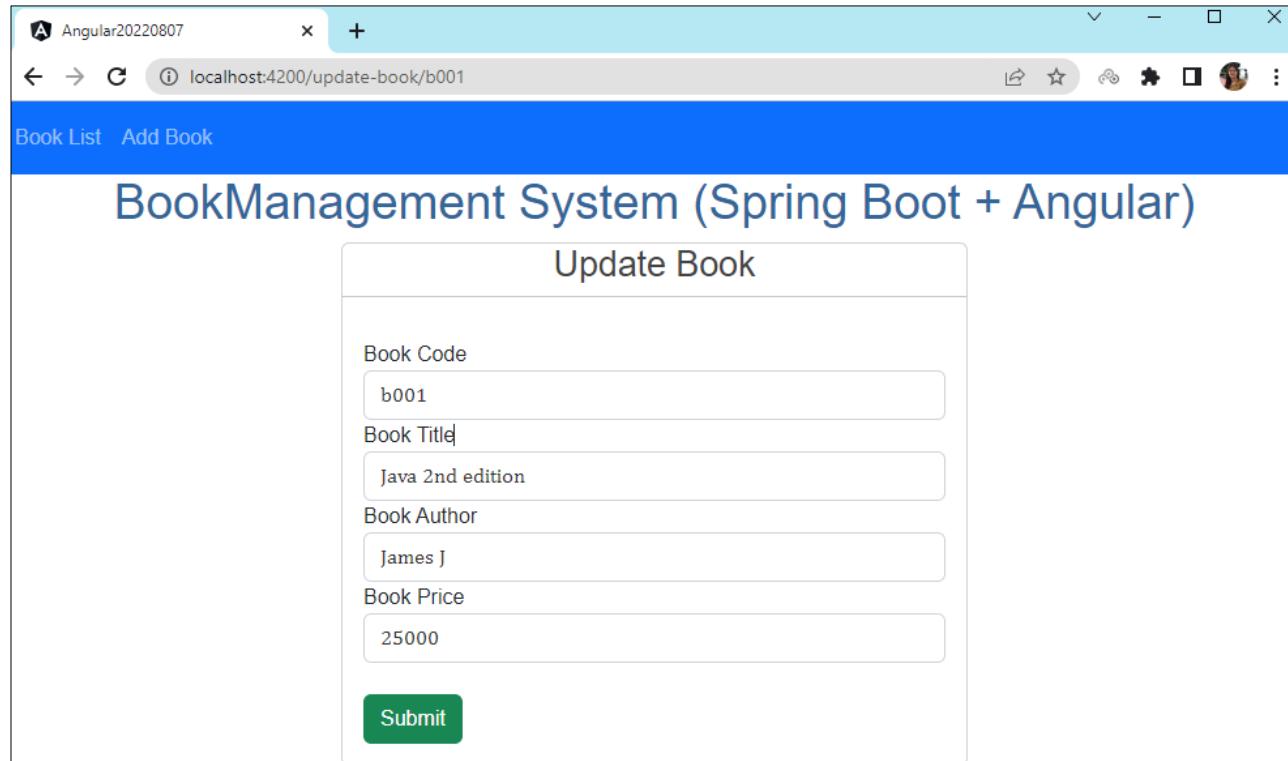


The screenshot shows a browser window titled "Angular20220807" with the URL "localhost:4200/create-book". The page has a blue header with "Book List" and "Add Book" buttons. Below the header, the title "BookManagement System (Spring Boot + Angular)" is displayed. A "Create Book" form is centered on the page, enclosed in a light gray border. The form contains four input fields: "Book Code", "Book Title", "Book Author", and "Book Price". Each field has a red error message below it: "Book Code is required", "Book Title is required", "Book Author is required", and "Book Price is required". At the bottom of the form is a green "Submit" button.

Field	Error Message
Book Code	Book Code is required
Book Title	Book Title is required
Book Author	Book Author is required
Book Price	Book Price is required

# BookManagement Project

## Frontend Output: Update Book



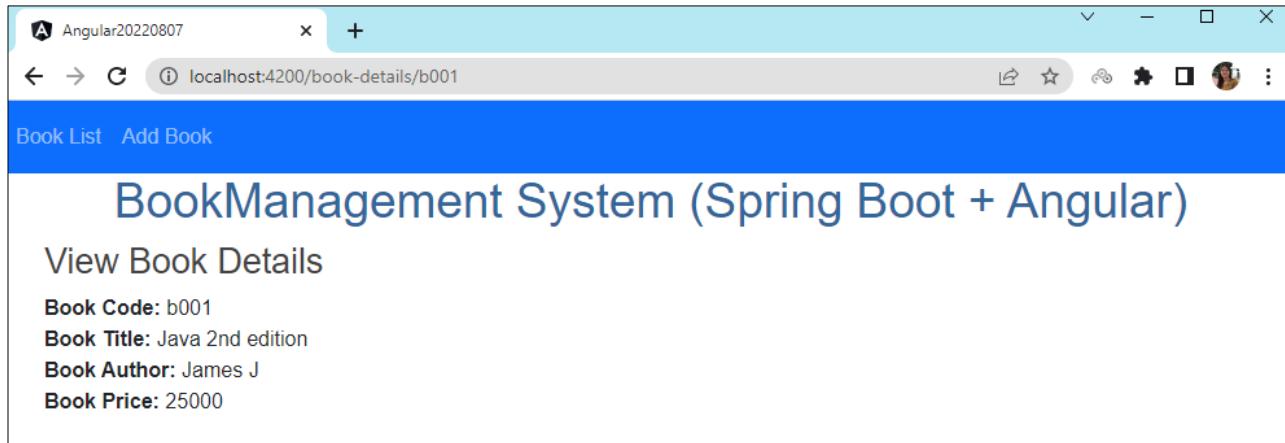
A screenshot of a web browser window titled "Angular20220807". The address bar shows "localhost:4200/update-book/b001". The page content is titled "BookManagement System (Spring Boot + Angular)". It features a form titled "Update Book" with the following fields:

- Book Code: b001
- Book Title: Java 2nd edition
- Book Author: James J
- Book Price: 25000

At the bottom of the form is a green "Submit" button.

# BookManagement Project

## Frontend Output: Book Detail



# BookManagement Project

## Project Structure:

The image shows a file explorer interface with two main sections. The left section displays the root directory 'Angular\_20220807' containing files like '.angular', '.vscode', 'node\_modules', 'src', 'app-routing.module.ts', 'app.component.css', 'app.component.html', 'app.component.spec.ts', 'app.component.ts', 'app.module.ts', 'book.service.spec.ts', 'book.service.ts', 'book.ts', 'assets', 'environments', 'favicon.ico', and 'index.html'. The right section displays the 'app' directory structure, which includes 'book-details', 'book-list', 'create-book', 'update-book', and 'app-routing.module.ts'. Each of these components contains component.css, component.html, component.spec.ts, and component.ts files.

```
Angular_20220807
  .angular
  .vscode
  node_modules
  src
    app
      book-details
      book-list
      create-book
      update-book
      app-routing.module.ts
      # app.component.css
      < app.component.html
      TS app.component.spec.ts
      TS app.component.ts
      TS app.module.ts
      TS book.service.spec.ts
      TS book.service.ts
      TS book.ts
      assets
      environments
      ★ favicon.ico
      < index.html

  app
    book-details
      # book-details.component.css
      < book-details.component.html
      TS book-details.component.spec.ts
      TS book-details.component.ts
    book-list
      # book-list.component.css
      < book-list.component.html
      TS book-list.component.spec.ts
      TS book-list.component.ts
    create-book
      # create-book.component.css
      < create-book.component.html
      TS create-book.component.spec.ts
      TS create-book.component.ts
    update-book
      # update-book.component.css
      < update-book.component.html
      TS update-book.component.spec.ts
      TS update-book.component.ts
    app-routing.module.ts
    # app.component.css
    < app.component.html
    TS app.component.spec.ts
    TS app.component.ts
    TS app.module.ts
    TS book.service.spec.ts
    TS book.service.ts
    TS book.ts
```

# BookManagement Project

book.ts

```
export class Book{  
    bookCode!: string  
    bookTitle!: string;  
    bookAuthor!: string;  
    bookPrice!: string;  
}
```

# BookManagement Project

## book.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http'
import { Observable } from 'rxjs';
import { Book } from './book';

@Injectable({
  providedIn: 'root'
})
export class BookService {

  private baseURL = "http://localhost:8080/api/v1/books";

  constructor(private httpClient: HttpClient) { }
```

# BookManagement Project

```
getBookList(): Observable<Book[]>{
    return this.httpClient.get<Book[]>(`${this.baseURL}`);
}

createBook(book: Book): Observable<Object>{
    return this.httpClient.post(` ${this.baseURL}` , book);
}

getBookByBookCode(bookCode: string): Observable<Book>{
    return this.httpClient.get<Book>(` ${this.baseURL}/ ${bookCode}` );
}

updateBook(bookCode: string, book: Book): Observable<Object>{
    return this.httpClient.put(` ${this.baseURL}/ ${bookCode}` , book);
}

deleteBook(bookCode: string): Observable<Object>{
    return this.httpClient.delete(` ${this.baseURL}/ ${bookCode}` );
}
```

# BookManagement Project

## book-list.component.html

```
<!-- <p>book-list works!</p> -->
<ng-template [ngIf]="check" [ngIfElse]="noData">
  <div class = "row">
    <h2> Book List</h2>
  </div>
  <table class = "table table-striped table-bordered">
    <thead>
      <tr>
        <th> Book Code</th>
        <th> Book Title </th>
        <th> Book Author</th>
        <th> Book Price</th>
        <th> Actions </th>
      </tr>
    </thead>
```

# BookManagement Project

```
<tbody>
  <tr *ngFor = "let book of books" >
    <td> {{ book.bookCode }} </td>
    <td> {{ book.bookTitle }} </td>
    <td> {{ book.bookAuthor}} </td>
    <td> {{ book.bookPrice}} </td>
    <td>
      <button (click) = "updateBook(book.bookCode)" class = "btn btn-primary"> Update</button>
      <button (click) = "deleteBook(book.bookCode)" class = "btn btn-danger" style="margin-left: 10px"> Delete</button>
      <button (click) = "bookDetails(book.bookCode)" class = "btn btn-primary" style="margin-left: 10px"> View</button>
    </td>
  </tr>
</tbody>
</table>

</ng-template>
<ng-template #noData>
<h4>No Book Found!</h4>
</ng-template>
```

# BookManagement Project

## book-list.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Book } from './book'
import { BookService } from './book.service'
import { Router } from '@angular/router';
@Component({
  selector: 'app-book-list',
  templateUrl: './book-list.component.html',
  styleUrls: ['./book-list.component.css']
})
export class BookListComponent implements OnInit {
  books: Book[];
  check = false;
  constructor(private bookService: BookService,
    private router: Router) {
    this.books = [];
  }
}
```

# BookManagement Project

```

ngOnInit(): void {
  this.getBooks();
}

private getBooks(){
  this.bookService.getBookList()
    .subscribe({
      next: (data) => {
        this.books = data;
        if(this.books.length!=0){
          this.check=true;
        }
      },
      error: (e) => console.log(e)
    });
}

bookDetails(bookCode: string){
  this.router.navigate(['book-details', bookCode]);
}

```

```

updateBook(bookCode: string){
  this.router.navigate(['update-book', bookCode]);
}

deleteBook(bookCode: string){
  this.bookService.deleteBook(bookCode)
    .subscribe({
      next: (data) => {
        console.log(data);
        this.getBooks();
      },
      error: (e) => console.log(e)
    });
}

```

# BookManagement Project

## create-book.component.html

```
<div class="row">
  <div class="card col-md-6 offset-md-3 offset-md-3">
    <div class="row">
      <h3 class="text-center"> Create Book </h3>
      <hr />
      <div class="card-body">
        <form #templateDrivenForm = "ngForm" (ngSubmit)="onSubmit(templateDrivenForm)">
          <div class="form-group">
            <label> Book Code</label>
            <input type="text" class="form-control" id="bookCode" [(ngModel)]="book.bookCode"
              name="bookCode" #bookCode="ngModel" required>

            <span class="text-danger" *ngIf="submitted">
              Book Code is required
            </span>
        </div>
```

# BookManagement Project

```
<div class="form-group">
    <label>Book Title</label>
    <input type="text" class="form-control" id="bookTitle" [(ngModel)]="book.bookTitle"
        name="bookTitle" #bookTitle="ngModel" required>
    <span class="text-danger" *ngIf = "submitted">
        Book Title is required
    </span>
</div>

<div class="form-group">
    <label> Book Author</label>
    <input type="text" class="form-control" id="bookAuthor" [(ngModel)]="book.bookAuthor"
        name="bookAuthor" #bookAuthor="ngModel" required>
    <span class="text-danger" *ngIf = "submitted">
        Book Author is required
    </span>
</div>
```

# BookManagement Project

```
<div class="form-group">
    <label> Book Price</label>
    <input type="text" class="form-control" id="bookPrice" [(ngModel)]="book.bookPrice"
        name="bookPrice" #bookPrice="ngModel" required>
    <span class="text-danger" *ngIf = "submitted">
        Book Price is required
    </span>
</div>

<br />
<button class="btn btn-success" type="submit" >Submit</button>
</form>
</div>
</div>
</div>
</div>
```

# BookManagement Project

## create-book.component.ts

```
import { NgForm } from '@angular/forms';
import { Component, OnInit } from '@angular/core';
import { Book } from '../book';
import { BookService } from '../book.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-create-book',
  templateUrl: './create-book.component.html',
  styleUrls: ['./create-book.component.css']
})
export class CreateBookComponent implements OnInit {
  book: Book = new Book();
  message="";
  submitted=false;
```

# BookManagement Project

```
constructor(private bookService: BookService,  
    private router: Router) {}  
  
ngOnInit(): void {}  
  
saveBook(){  
    this.bookService.createBook(this.book)  
        .subscribe(  
            next: (data) =>{  
                console.log(data);  
  
                this.goToBookList();  
                //this.message = 'Book was inserted successfully!';  
            },  
            error: (e) => console.log(e)  
        );  
}
```

# BookManagement Project

```
goToBookList(){  
  this.router.navigate(['/books']);  
}  
  
onSubmit(form: NgForm):void{  
  
  if (form.valid) {  
    this.submitted=false;  
    console.log(form.value);  
    this.saveBook();  
  } else {  
    this.submitted=true;  
    console.log('invalid form');  
  }  
}
```

# BookManagement Project

## update-book.component.html

```
<div class="row">
  <div class="card col-md-6 offset-md-3 offset-md-3">
    <div class="row">
      <h3 class="text-center"> Update Book </h3>
      <hr />
      <div class="card-body">
        <form (ngSubmit)="onSubmit()">

          <div class="form-group">
            <label> Book Code</label>
            <input type="text" class="form-control" id="bookCode" [(ngModel)]="book.bookCode"
                  name="bookCode" readonly>
          </div>
```

# BookManagement Project

```
<div class="form-group">
    <label>Book Title</label>
    <input type="text" class="form-control" id="bookTitle" [(ngModel)]="book.bookTitle"
           name="bookTitle">
</div>

<div class="form-group">
    <label> Book Author</label>
    <input type="text" class="form-control" id="bookAuthor" [(ngModel)]="book.bookAuthor"
           name="bookAuthor">
</div>
```

# BookManagement Project

```
<div class="form-group">
    <label> Book Price</label>
    <input type="text" class="form-control" id="bookPrice" [(ngModel)]="book.bookPrice"
           name="bookPrice">
</div>

<br />
<button class="btn btn-success" type="submit">Submit</button>

</form>
</div>
</div>
</div>
</div>
```

# BookManagement Project

## update-book.component.ts

```
import { Component, OnInit } from '@angular/core';
import { BookService } from '../book.service';
import { Book } from '../book';
import { ActivatedRoute, Router } from '@angular/router';

@Component({
  selector: 'app-update-book',
  templateUrl: './update-book.component.html',
  styleUrls: ['./update-book.component.css']
})
export class UpdateBookComponent implements OnInit {

  bookCode!: string;
  book: Book= new Book();
  message ="";
```

# BookManagement Project

```
constructor(private bookService: BookService,  
private route: ActivatedRoute,  
private router: Router) {}  
  
ngOnInit(): void {  
  this.bookCode = this.route.snapshot.params['bookCode'];  
  
  this.bookService.getBookByBookCode(this.bookCode)  
    .subscribe(  
      next: (data) => {  
        this.book = data;  
      },  
      error: (e) => console.log(e)  
    );  
}
```

# BookManagement Project

```
onSubmit(){
    this.bookService.updateBook(this.bookCode, this.book)
        .subscribe({
            next: (data) => {
                this.goToBookList();
            },
            error: (e) => console.log(e)
        });
}

goToBookList(){
    this.router.navigate(['/books']);
}
```

# BookManagement Project

## book-details.component.html

```
<h3> View Book Details</h3>
<div>
  <div>
    <label> <b> Book Code: </b></label> {{book.bookCode}}
  </div>
  <div>
    <label> <b> Book Title: </b></label> {{book.bookTitle}}
  </div>
  <div>
    <label> <b> Book Author: </b></label> {{book.bookAuthor}}
  </div>
  <div>
    <label> <b> Book Price: </b></label> {{book.bookPrice}}
  </div>
</div>
```

# BookManagement Project

## book-details.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Book } from '../book';
import { ActivatedRoute } from '@angular/router';
import { BookService } from '../book.service';

@Component({
  selector: 'app-book-details',
  templateUrl: './book-details.component.html',
  styleUrls: ['./book-details.component.css']
})
export class BookDetailsComponent implements OnInit {

  bookCode!:string
  book!: Book
  constructor(private route: ActivatedRoute, private bookService: BookService) { }
```

# BookManagement Project

```
ngOnInit(): void {
  this.bookCode = this.route.snapshot.params['bookCode'];

  this.book = new Book();
  this.bookService.getBookByBookCode(this.bookCode)
    .subscribe({
      next: (data)=> {
        this.book = data;
      },
      error: (e) => console.log(e)
    });
}
```

# BookManagement Project

## app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { BookListComponent } from './book-list/book-list.component';
import { CreateBookComponent } from './create-book/create-book.component';
import { UpdateBookComponent } from './update-book/update-book.component';
import { BookDetailsComponent } from './book-details/book-details.component';

const routes: Routes = [
  {path: 'books', component: BookListComponent},
  {path: 'create-book', component: CreateBookComponent},
  {path: '', redirectTo: 'books', pathMatch: 'full'},
  {path: 'update-book/:bookCode', component: UpdateBookComponent},
  {path: 'book-details/:bookCode', component: BookDetailsComponent}
];
```

# BookManagement Project

```
@NgModule({  
  imports:  
  [RouterModule.forRoot(routes)],  
  
  exports: [RouterModule]  
})  
export class AppRoutingModule { }
```

# BookManagement Project

## app.component.html

```
<nav class="navbar navbar-expand-sm bg-primary navbar-dark">
  <ul class = "navbar-nav">
    <li class = "nav-item">
      <a routerLink="books" routerLinkActive="active" class="nav-link" >Book List</a>
    </li>
    <li class = "nav-item">
      <a routerLink="create-book" routerLinkActive="active" class="nav-link" >Add Book</a>
    </li>
  </ul>
</nav>

<h1 class="text-center"> {{title}} </h1>
<div class = "container">
  <router-outlet></router-outlet>
</div>
```

# BookManagement Project

## app.component.ts

```
import { Component } from '@angular/core';
import { ViewChild } from '@angular/core';
import { NgForm } from '@angular/forms';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {
  title = 'BookManagement System (Spring Boot + Angular)';
}
```

# BookManagement Project

## app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { HttpClientModule } from '@angular/common/http'
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BookListComponent } from './book-list/book-list.component';
import { CreateBookComponent } from './create-book/create-book.component';
import { FormsModule} from '@angular/forms';
import { UpdateBookComponent } from './update-book/update-book.component';
import { BookDetailsComponent } from './book-details/book-details.component'

@NgModule({
  declarations: [
    AppComponent,
    BookListComponent,
    CreateBookComponent,
    UpdateBookComponent,
    BookDetailsComponent ],
  imports: [BrowserModule,HttpClientModule, AppRoutingModule]
})
export class AppModule { }
```

# BookManagement Project

```
imports: [
  BrowserModule,
  AppRoutingModule,
  HttpClientModule,
  FormsModule
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule { }
```

# BookManagement Project

## angular.json

```
"styles": [  
    "src/styles.css",  
    "node_modules/bootstrap/dist/css/bootstrap.min.css"  
,  
    "scripts": [  
        "node_modules/jquery/dist/jquery.min.js",  
        "node_modules/bootstrap/dist/js/bootstrap.min.js"  
    ]
```

Thank you!!  
Q&As

