Todo – App

Ref : angular Hero App

Learning Coverage

- Sensible Project Structure
- Data Binding
- Master/ Detail
- Services
- Dependency Injection
- Navigation
- Remote Data Acces

Purpose

The purpose of this todo app is to show you most of the features that you will be using in your real apps.

Purpose of this app is:
Display the list of todos
Editing the selected todo
Navigating to different views

You'll use **built-in directives** to show and hide elements and display lists of todos data. You'll create **components** to display todo details and show an array of todos. You'll use **one-way data binding** for read-only data. You'll add editable fields to update a **model** with two-way data binding. You'll bind component methods to user events, like keystrokes and clicks. You'll enable users to select a todo from a **master** list and edit that todo in the **details** view. You'll format data with pipes. You'll create a shared **service** to assemble the todos. And you'll use **routing** to navigate among different **views** and their components.

Todo-App Setup

- Follow the Setup process shown in the first slide.
- Put the quick_start in todo_app folder
- Keep the app running
    - Go inside the todo_app folder
    - Run **npm start** command, and let the command prompt open
    - Open the todo_app folder in either ATOM or Visual Studio

Show todo

Add two properties to the AppComponent: title (for the title of the app), todo (to display one todo)

app.component.ts (AppComponent class)

export class AppComponent {
 title='Todo - App';
 todo = 'Create Angular 2 Todo app';
 }

Todo-App Setup

Now update the template in the @Component decorator for these two properties

app.component.ts (@Component)

```
template: `
<h1>{{title}}</h1>
<p>{{todo}}</p>
`,
```

Save the changes, browser will refresh automatically, and changes will be reflected.

Double curly braces are the interpolation syntax. It is used to bind the property values.

Todo-App Setup

Todo Object

todo need more properties, convert todo from string to a class.
Create a Todo class with id and name properties. Add these properties near the top of the app.component.ts file, just below the import statement.

app.component.ts (Todo class)

export class Todo{
id:number;
name: string;
}

In the AppComponent class, refactor the component's todo property to be of type Todo, then initialize it with an id of 1 and the name ”Create Angular 2 Todo app”.

```
todo: Todo = {   id:1,   name:'Create Angular 2 Todo app' }
```

Update the binding in the template to refer to todo’s id and name

```
<p>{{todo.id}} : {{todo.name}}</p>
```

Todo-App Setup

Match the code

```
import { Component } from '@angular/core';
export class Todo{
id:number;
name: string;
}

@Component({
selector: 'my-app',
template: `
<h1>{{title}}</h1>
<h2>{{todo.name}} - details!</h2>
<div><label>id: </label>{{todo.id}}</div>
<div><label>name: </label>{{todo.name}}</div>
`,
})
export class AppComponent {
title='Todo - App';
todo: Todo = {
id:1,
name:'Create Angular 2 Todo app'
}
}
```

Todo-App Setup

Edit the todo name

Users should be able to edit the todo name in an <input> textbox. The textbox should both *display* the todo's name property and *update* that property as the user types.
You need a two-way binding between the <input> form element and the todo.name property.

Two way data binding

Refactor the code for todo.name, so that code should look like this.

```
<div>
<label>name: </label>
<input [(ngModel)]="todo.name" placeholder="add task">
</div>
```

[(ngModel)] is the Angular syntax to bind the todo.name property to the textbox. Data flows *in both directions:* from the property to the textbox, and from the textbox back to the property.
immediately after this change, the application breaks. If you looked in the browser console, you'd see Angular complaining that "ngModel ... isn't a known property of input."

Todo-App Setup

Although NgModel is a valid Angular directive, it isn't available by default. It belongs to the optional FormsModule. You must opt-in to using that module.

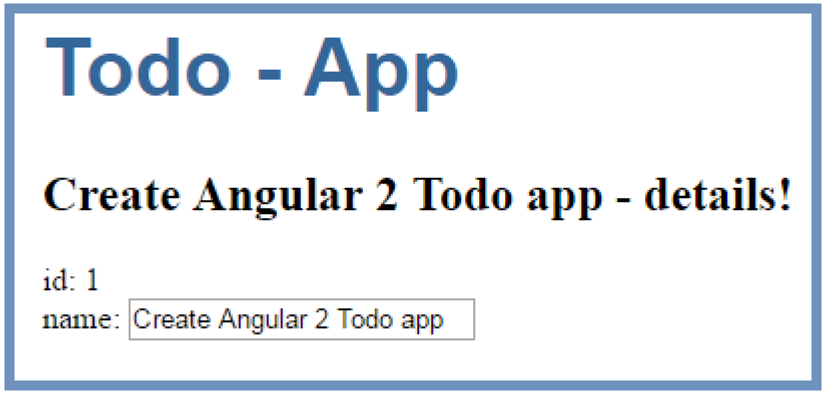
Imports the FormsModule

Open the **app.module.ts** file and import the **FormsModule** symbol from the **@angular/forms** library. Then add the FormsModule to the **@NgModule** metadata's imports array, which contains the list of external modules that the app uses.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';
@NgModule({
imports: [ BrowserModule, FormsModule ],
declarations: [ AppComponent ],
bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Todo-App Setup

Now save the changes and look for the changes on browser. Application should look like this.


Todo - App

Create Angular 2 Todo app - details!

id: 1
name: Create Angular 2 Todo app

Thank You!

Email: info@yash.com
Web: www.yash.com