© Victor Tan 2024

Angular
Lab 2
Parent and Child Components

1 Lab setup

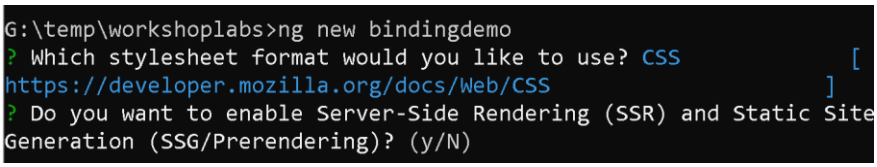
The setup here is identical to that in Lab 1

2 Generating a new Angular project

In a separate command prompt (or shell terminal) or in an embedded terminal in Visual Studio Code inside the main folder on your machine for Angular projects, create a new Angular project with:

```
ng new parentchilddemo
```

Press enter to accept the default values for all the question prompts that follow regarding stylesheet format (CSS) and enabling Server side rendering (No)



Navigate into the root project folder from the terminal with:

```
cd parentchilddemo
```

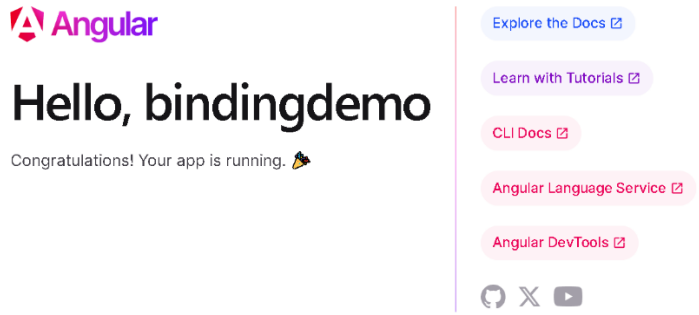
Build the app and serve it via Angular's live development server by typing:

```
ng serve
```

The final output line from this should indicate that the compilation process was successful and identify the port that the live development server is currently serving up the Angular app dynamically from (by default this will be port 4200)

Open a browser tab at:
http://localhost:4200/

You should be able to see the default landing page for all new autogenerated Angular projects.



To stop the development server, type Ctrl+C in the terminal window that it is running in. You can restart again anytime by typing `ng serve` in the project root folder.

3 Generating and using a component as a child

The source code for this lab can be found in the `changes` subfolder of `Parent-Child-Demo`

Open a new command prompt (separate from the one that you are running `ng serve` in) in the root folder of the project created earlier. Type this command to generate a new component:

```
ng generate component firstChild
```

NOTE: Often, the command above is provided in its short form format where only the first letter of the command is specified, for e.g.

```
ng g c firstChild
```

By default, this command creates the following:

- A folder named after the component
- A component file, `<component-name>.component.ts`
- A template file, `<component-name>.component.html`
- A CSS file, `<component-name>.component.css`
- A testing specification file, `<component-name>.component.spec.ts`

The folder created to place the component-related files will be by default inside `src/app` (the folder holding the root component of the app and its associated files).

If camel-case naming is used for the component name in the `generate` command, then each of the component related file names will consist of the individual names of the camel case separated by a hyphen, for e.g.

```
ng generate component TryThisApple
CREATE    src/app/try-this-apple/try-this-apple.component.html    (29
bytes)
CREATE  src/app/try-this-apple/try-this-apple.component.spec.ts  (670
bytes)
…
…
```

The new component class itself (`first-child.component.ts`) will have the specified name in the `generate` command suffixed with `Component` (for e.g. `FirstChildComponent`). Like AppComponent, all newly generated components are stand alone components by default in Angular 17 and later.

Modify the following files in `src/app` with the latest update from `changes`:

`app.component-v1.html`

`app.component-v1.ts`

Notice that AppComponent needs to explicit import FirstChildComponent and specify it in its `imports` property of @Component decorator in order to use it.
Placing the CSS selector for the newly generated component in the template of the root component (AppComponent) results in the template for the new component to be embedded in place of this selector when rendering the complete app view. This is functionally identical to the case of placing the CSS selector for AppComponent in the main `index.html`.

References:

https://angular.io/guide/component-overview#creating-a-component

4 Transferring data from parent to child via @Input and property binding

Modify the following files in `src/app` with the latest update from `changes`:

`app.component-v2.ts`

`app.component-v2.html`

Modify the following files in `src/app/first-child` with the latest update from `changes`:

`first-child.component-v2.ts`

`first-child.component-v2.html`

To transfer data from the parent to the child component, in the child component we:

- Import the @Input module from @angular/Core Library
- Declare and decorate a property with the @Input decorator. This property will receive data sent from the parent

In the parent component, we

- Bind the @Input property in the child with a parent property in the parent template within the selector tag of the child component

References:

https://angular.io/guide/inputs-outputs#sending-data-to-a-child-component

https://www.tektutorialshub.com/angular/angular-passing-data-child-component

5 Transferring data from child to parent via @Output and event binding

Modify the following files in `src/app` with the latest update from `changes`:

`app.component-v3.ts`

`app.component-v3.html`

Modify the following files in `src/app/first-child` with the latest update from `changes`:

`first-child.component-v3.ts`

`first-child.component-v3.html`

To transfer data from the child back to the parent component, in the child component:

- Declare a property of type EventEmitter and mark it with a @Output Decorator
- At a particular point (typically in response to an event in the child template), raise an event on this property via the emit method

In the parent component:

- Provide an event binding for the event issued from the child property. The name of the event is the same as the child property and the $event object will contain be of the type of the value emitted from this property
- Typically, the template statement for the event binding will call a method with the $event object to perform further processing.

5.1 Update property transferred from parent to child within the child component

In this example, the event emitted does not affect the property originally transferred from the parent component (`parentCounter`) to the child (`childCounter`). However, it is possible to also emit a value from the child component that involves some update to this property (`childCounter`) which is then transmitted back to the parent component to change the original property, which in turn results in a change in the child property (`childCounter`) through property binding in the usual manner.

Modify the following files in `src/app` with the latest update from `changes`:

`app.component-v4.ts`

`app.component-v4.html`

Modify the following files in `src/app/first-child` with the latest update from `changes`:

`first-child.component-v4.ts`

`first-child.component-v4.html`

References:

https://angular.io/guide/inputs-outputs#sending-data-to-a-parent-component
https://www.tektutorialshub.com/angular/angular-pass-data-to-parent-component/

6 Accessing child component via template reference variable

Modify the following files in `src/app` with the latest update from `changes`:

`app.component-v5.html`

`app.component-v5.ts`

Modify the following files in `src/app/first-child` with the latest update from `changes`:

`first-child.component-v5.ts`

We can access properties / methods in the child component via a template variable. This provides an alternative to using event binding with @Output in certain situations.

Here we remove the event binding for the `textChanged` event that was previously emitted from the child component:
`(textChanged)="processChangeFromChild($event)"`
as we can now directly access the child property `childText` using the template variable `myChild`