participants-app

step-by-step

1. Create the application.

Open VS Code.

Open the folder where you want to create the app [Ctrl-K Ctrl-O].

Open the Angular-CLI [Ctrl-ñ] and type[1]:

```
CLI> ng new participants-app
```
[2]

☑ Verify you can serve and open the app[3].

```
CLI> cd participants-app
CLI> ng serve
```

☑ Verify you open the app.

Type localhost:4200 in the direction-bar of a browser to see the default application you just created.

**note: if you have many apps in the folder you just opened in code, you may want to close that folder and open only the folder that was created during the creation of your new application.*

2. Create the model.

Create a folder to hold all your model classes.

(right-click) over the ./src/app folder, and select New Folder.

Name it **model**.

[1] In the rest of the document this indication will be omitted, whenever you see CLI> before any sentence, it means type over the CLI.

[2] It may take a while.

[3] Whenever you see a check-mark, please do not go ahead until you successfully complete that step.

Create the model for participant.

(right-click) over the model folder, and select New File.

Name it **participant.ts**[4]

The content of the file should be by like this[5]:

```
1   export class Participant {
2       initials: String;
3       name: String;
4       address: String;
5       preferredLanguage: String;
6
7       constructor(
8           initials: String,
9           name: String,
10          address: String,
11          preferredLanguage: String
12      ){
13          this.initials = initials;
14          this.name = name;
15          this.address  = address;
16          this.preferredLanguage  = preferredLanguage;
17      }
18  }
```

3. Create the components.

```
CLI> ng generate component participants
CLI> ng generate component participant-form
CLI> ng generate component name-list
CLI> ng generate component detail
```

4. Add the routing.

```
CLI> ng generate module app-routing --flat --module=app
Modify the file to add the routes
```

[4] Note that the file name is lower case.

[5] In the rest of the document this indication will be omitted unless it gives valuable information.

app-routing.module.ts

```ts
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { RouterModule, Routes } from '@angular/router';
import { ParticipantsComponent } from './participants/participants.component';
import { ParticipantFormComponent } from './participant-form/participant-form.component';

const routes: Routes = [
  { path: 'participants', component: ParticipantsComponent },
  { path: 'new', component: ParticipantFormComponent}
]

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

5. Modify the template for data-less components: AppComponent and ParticipantsComponent.

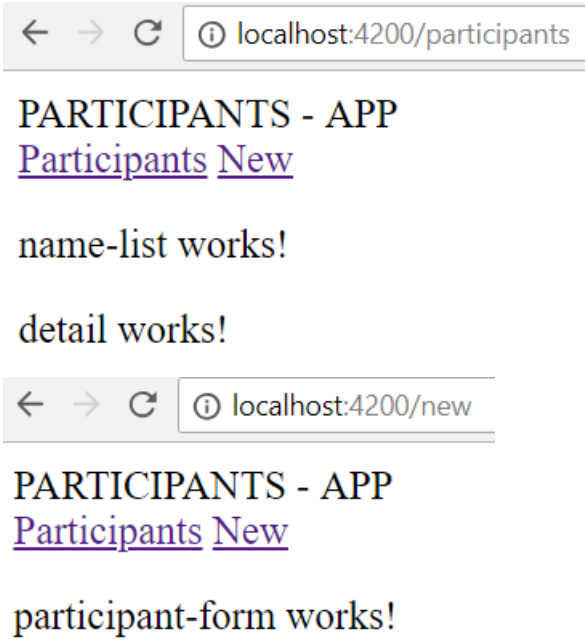
app.component.html

```html
<div>
  <div>
    PARTICIPANTS - APP
  </div>
  <div>
    <nav>
      <a routerLink = "/participants">Participants</a>
      <a routerLink = "/new">New</a>
    </nav>
    <router-outlet></router-outlet>
  </div>
</div>
```

participants.component.html

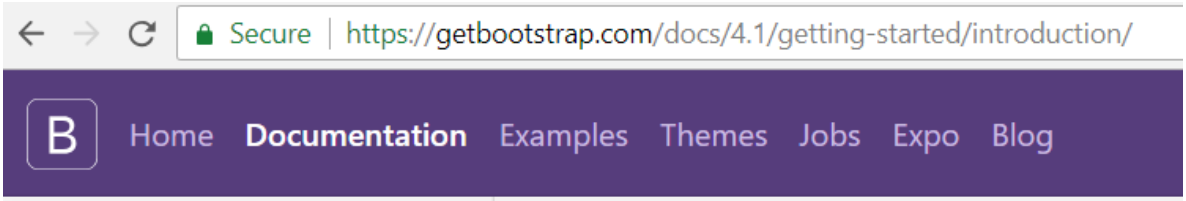
```html
<div>
  <app-name-list></app-name-list>
  <app-detail></app-detail>
</div>
```

☑ Verify your application works as expected.



6. Use the bootstrap-framework.

Copy the stylesheet link from the bootstrap web-site into your <head> index.html file.
Place the scripts from the before the closing </body> tag.



CSS

Copy-paste the stylesheet <link> into your <head> before all other stylesheets to load our CSS.

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css" integrity="sha
```

Copy

JS

Many of our components require the use of JavaScript to function. Specifically, they require jQuery, Popper.js, and our own JavaScript plugins. Place the following <script>s near the end of your pages, right before the closing </body> tag, to enable them. jQuery must come first, then Popper.js, and then our JavaScript plugins.

We use jQuery's slim build, but the full version is also supported.

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbz
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.min.js" integrity="sha384-cs/chFZiN24
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.min.js" integrity="sha384-uefMccjFJAIv6A
```

Copy

The index.html file should look like this.

index.html

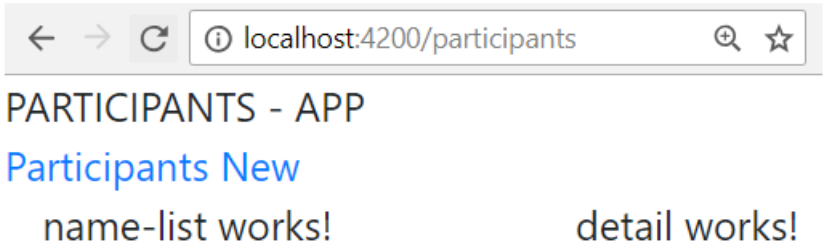
```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>ParticipantsApp</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.
</head>
<body>
  <app-root></app-root>
  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha3
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.min
</body>
</html>
```

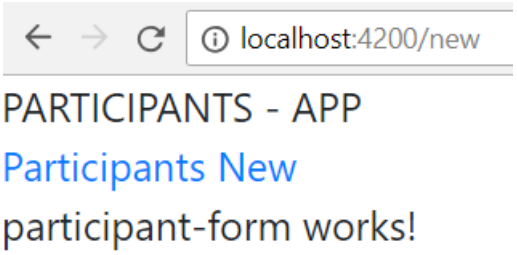
Use “`container`”, “`row`“ and “`col`“ classes to organize the components.

participants.component.html

```
<div class="container">
  <div class="row">
    <div class="col">
      <app-name-list></app-name-list>
    </div>
    <div class="col">
      <app-detail></app-detail>
    </div>
  </div>
</div>
```

☑ Verify your application works as expected.





7. Create the services.

Generate a service to get the participants list from and to put a new participant.

```
CLI> ng generate service participant-data
```

Modify the service to add the `getParticipants()` and `putParticipant()` methods.

participant-data.service.ts

```ts
import { Injectable } from '@angular/core';
import { Participant } from './model/participant'

@Injectable()
export class ParticipantDataService {

  participantes: Participant[];

  constructor() {
    this.participantes = [];

    const participante1 = new Participant('LGF', 'Liliana Gutiérrez', 'Lejos', 'Java');
    const participante2 = new Participant('AAH', 'Alejandro Arellano', 'Por allí', 'Java');
    const participante3 = new Participant('JLV', 'José Luis Vera', 'Creca', 'C#');

    this.participantes.push(participante1, participante2, participante3);
  }

  getParticipants(): Participant[] {
    return this.participantes;
  }

  putParticipant(participante1: Participant) {
    this.participantes.push(participante1);
    console.log(this.participantes);
  }

}
```

Add the service as a provider in the `app.module.ts` file.

TS app.module.ts ✕

```
19    imports: [
20      BrowserModule,
21      AppRoutingModule,
22    ],
23    providers: [
24      ParticipantDataService
25    ],
26    bootstrap: [AppComponent]
27  })
28  export class AppModule { }
```

8. Use the service to get data and display that.

a) Import the Participant model and the ParticipantDataService in the `name-list.component.ts` file.
b) Declare the participants array.
c) Instantiate the service on the constructor.
d) Get the participant list from the service.

TS *name-list.component.ts* ✕

a)
```
import { Component, OnInit } from '@angular/core';
import { ParticipantDataService } from '../participant-data.service';
import { Participant } from '../model/participant';

@Component({
  selector: 'app-name-list',
  templateUrl: './name-list.component.html',
  styleUrls: ['./name-list.component.css']
})
export class NameListComponent implements OnInit {
```

b)
```
  participants: Participant[];
```

c)
```
  constructor(
    private participantDataService : ParticipantDataService
  ) { }
```

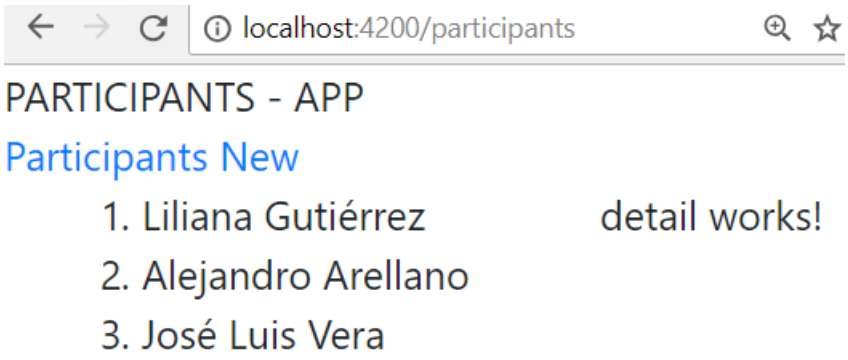
d)
```
  ngOnInit() {
    this.participants = this.participantDataService.getParticipants();
  }
}
```

e) Display the `participants` array in the html file.

name-list.component.html

```
<div>
  <ol>
    <li *ngFor="let participant of participants">
      {{ participant.name }}
    </li>
  </ol>
</div>
```

☑ Verify your application works as expected.



9. Get data from the user and use the service to put that.

a) Add the FormsModule in the imports in the `app.module.ts` file.

app.module.ts

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

b) Import the Participant model and the ParticipantDataService in the `participant-form.component.ts` file.
c) Declare and create a participant object.

participant-form.component.ts

```ts
import { Component, OnInit } from '@angular/core';
import { Participant } from '../model/participant'
import { ParticipantDataService } from '../participant-data.service';

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

  participant: Participant = new Participant('','','','');
```

d) Design a form the get the data from the user.
e) Use the [(ngModel)] to bind the data to the model.
f) Add a button and use the (click) to catch the click.

participant-form.component.html

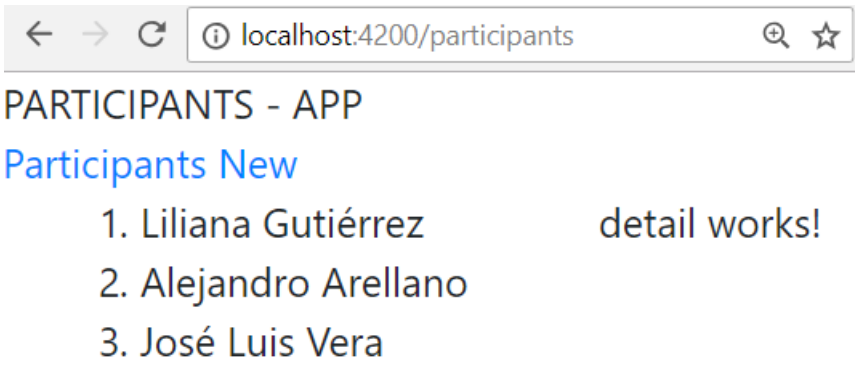
```html
<p>
  <input type="text" placeholder='Initials:' />
</p>

<p>
  <input type="text" placeholder='Name:' [(ngModel)]="participant.name" />
</p>
<p>
  <input type="text" placeholder='Address:' [(ngModel)] = "participant.address" />
</p>
<p>
  <input type="text" placeholder='Preferred Language:'[(ngModel)]="participant.preferredLanguage" />
</p>

<p>
  <button (click)="newHandler(participant)"> New </button>
</p>
```

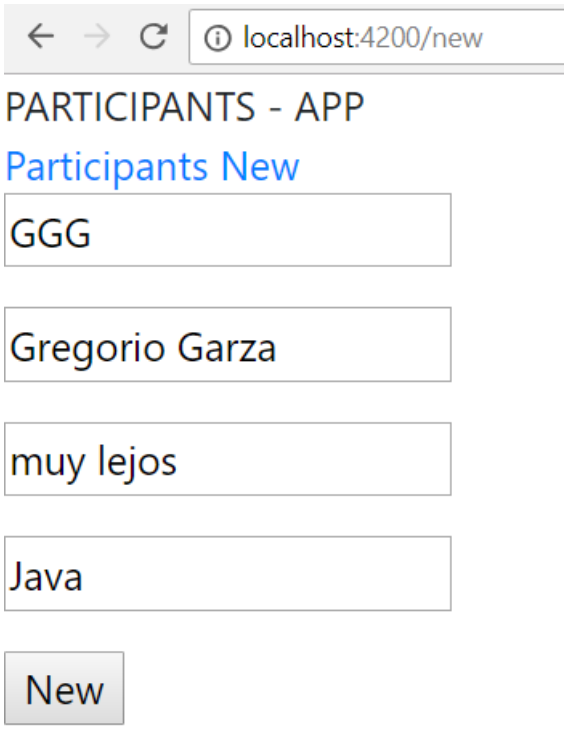
g) Instantiate the service on the constructor.
h) Implements the `newHandler` method and use the service to put the data.

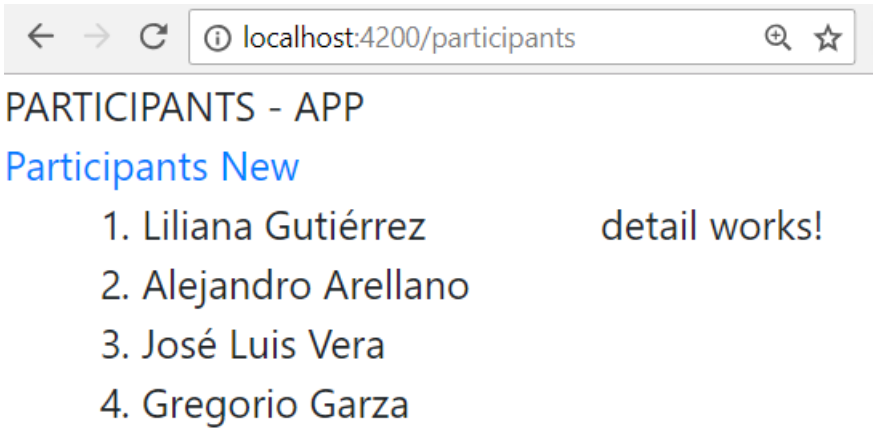
participant-form.component.ts

```
13
14    constructor(
15      private participantDataService : ParticipantDataService
16    ) { }
17
18    ngOnInit() {
19    }
20
21    newHandler(participant1: Participant){
22      this.participantDataService.putParticipant(participant1);
23    }
24  }
```

☑ Verify your application works as expected.




localhost:4200/new
PARTICIPANTS - APP
Participants New
GGG
Gregorio Garza
muy lejos
Java
New


localhost:4200/participants
PARTICIPANTS - APP
Participants New
1. Liliana Gutiérrez
detail works!
2. Alejandro Arellano
3. José Luis Vera
4. Gregorio Garza

10. Display data from another component.

Emit data from child to parent.

Child: name-list.componet

Parent: participants.componet

Add the (click) directive to name-list.component.html file (child) to catch the participant clicked.

name-list.component.html

```
1  <div>
2    <ol>
3      <li *ngFor="let participant of participants" (click)="clickParticipantHandler(participant)">
4        {{ participant.name }}
5      </li>
6    </ol>
7  </div>
```

Import Output and EvenEmitter.
Use the @Output directive to declare the output variable and create it.
Implement the clickParticipantHandler method to emit the participant.

a)

TS name-list.component.ts

```
1  import { Component, OnInit, Output, EventEmitter } from '@angular/core';
2  import { ParticipantDataService } from '../participant-data.service';
3  import { Participant } from '../model/participant';
4
5  @Component({
6    selector: 'app-name-list',
7    templateUrl: './name-list.component.html',
8    styleUrls: ['./name-list.component.css']
9  })
10 export class NameListComponent implements OnInit {
11
12   @Output()outParticipant: EventEmitter<Participant> = new EventEmitter<Participant>();
13   participants: Participant[];
14
15   constructor(
16     private participantDataService : ParticipantDataService
17   ) { }
18
19   ngOnInit() {
20     this.participants = this.participantDataService.getParticipants();
21   }
22
23   clickParticipantHandler(participant){
24     this.outParticipant.emit(participant);
25   }
26
27 }
```

b)

c)

Receive data from child.

Child: name-list.componet

Parent: participants.componet

Receive the data in the html file and catch the event.

participants.component.html

```
1   <div class="container">
2     <div class="row">
3       <div class="col">
4         <app-name-list
5         (outParticipant)="fromChildParticiantHandler($event)"
6         ></app-name-list>
7       </div>
8       <div class="col">
9         <app-detail></app-detail>
10      </div>
11    </div>
12  </div>
```

Create the object where to receive the data.
Implements the `fromChildParticipantHandler` method and receive the data.

participants.component.ts

```ts
import { Component, OnInit, Input } from '@angular/core';
import { Participant } from '../model/participant';

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

  fromChildParticipant : Participant;

  constructor() {
    this.fromChildParticipant = new Participant ('','','','');
    }

  fromChildParticiantHandler(event){
    this.fromChildParticipant = event;
    }

  ngOnInit() {
  }

}
```

Pass data from parent to child.

Child: detail.component

Parent: participants.componet

participants.component.html

```
<div class="container">
  <div class="row">
    <div class="col">
      <app-name-list
      (outParticipant)="fromChildParticiantHandler($event)"
      ></app-name-list>
    </div>
    <div class="col">
      <app-detail
      [inParticipant] = fromChildParticipant
      ></app-detail>
    </div>
  </div>
</div>
```

Receive and use the data in the child.

Child: detail.component

Parent: participants.componet

TS *detail.component.ts* ✖

```ts
import { Component, OnInit, Input } from '@angular/core';
import { Participant } from '../model/participant';

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

  @Input()inParticipant : Participant;

  constructor() { }

  ngOnInit() {
  }

}
```

<> *detail.component.html* ✖

```html
<p>
  {{ inParticipant.initials }}
</p>
<p>
  {{ inParticipant.name }}
</p>
<p>
  {{ inParticipant.address }}
</p>
<p>
  {{ inParticipant.preferredLanguage }}
</p>
```

☑ Verify your application works as expected.