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**.

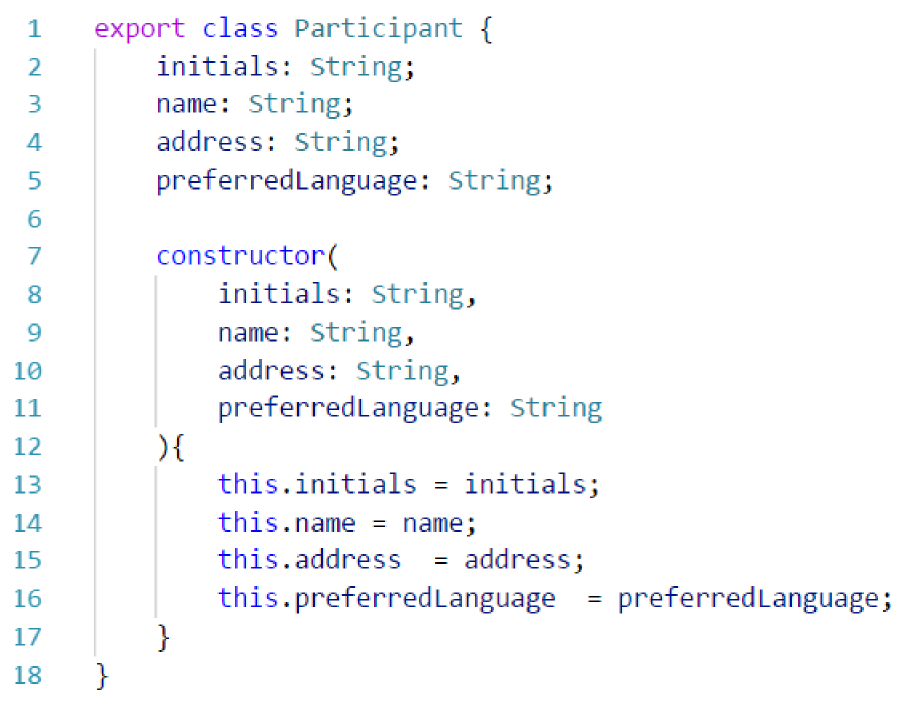
[1] In the rest of the document this indication will be omitted, whenever you see `CLI>` before any sentence, it means type over the CLI.
[2] It may take a while.
[3] Whenever you see a check-mark, please do not go ahead until you successfully complete that step.

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]:



```
export class Participant {
    initials: String;
    name: String;
    address: String;
    preferredLanguage: String;

    constructor(
        initials: String,
        name: String,
        address: String,
        preferredLanguage: String
    ){
        this.initials = initials;
        this.name = name;
        this.address  = address;
        this.preferredLanguage  = preferredLanguage;
    }
}
```

3. Create the components.

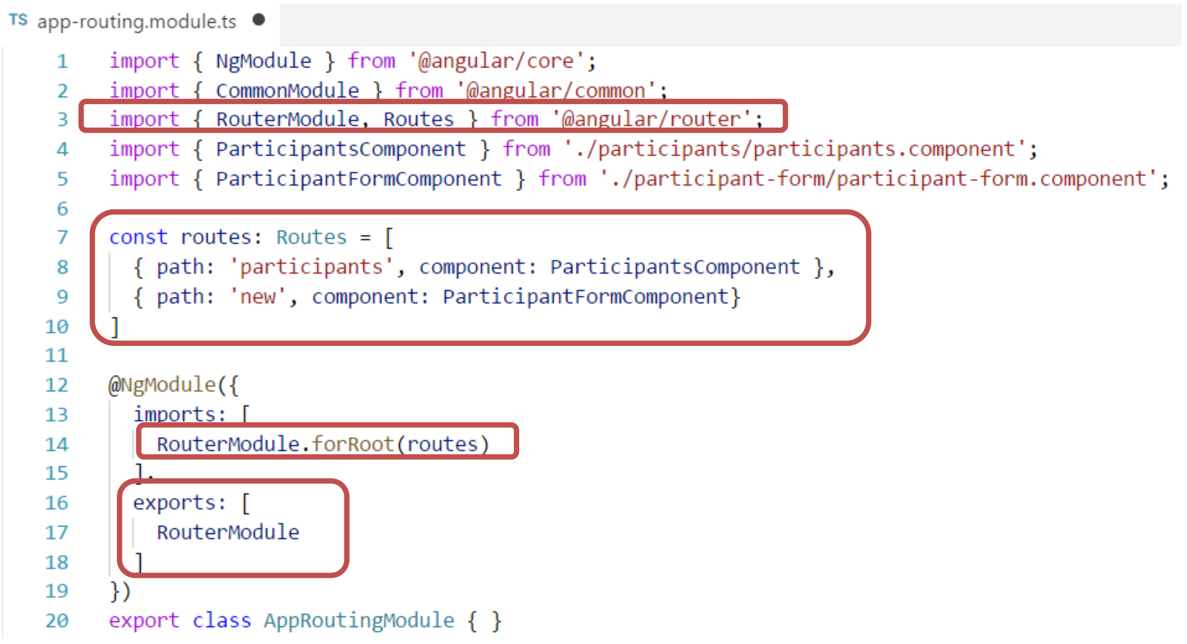
```
CLI> ng generate component participants
CLI> ng generate component participant-form
CLI> ng generate component name-list
CLI> ng generate component detail
```

4. Add the routing.

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

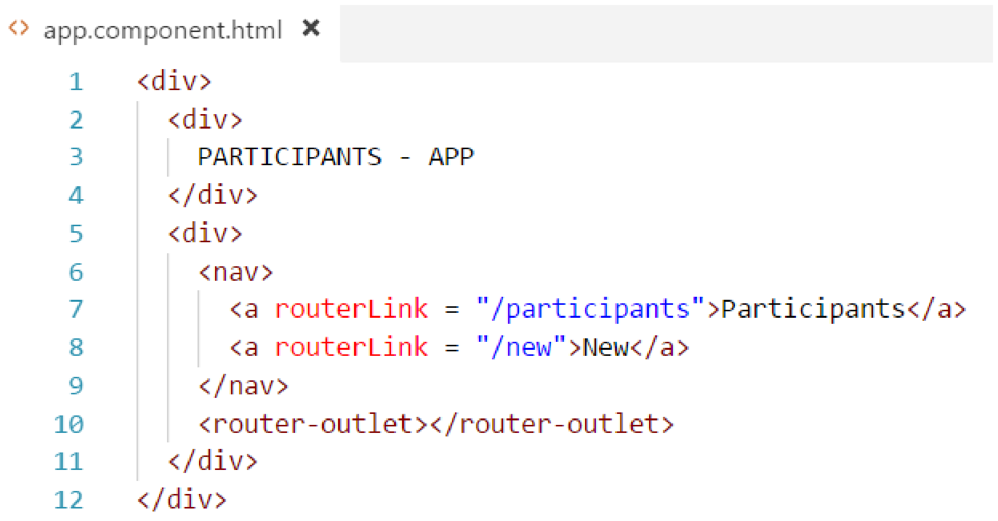
[4] Note that the file name is lower case.

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

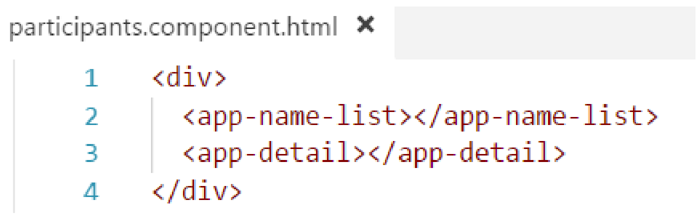

app-routing.module.ts

```
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.

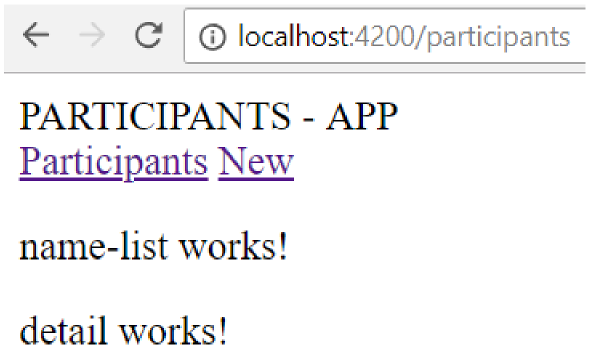

app.component.html

```
<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

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

☑ Verify your application works as expected.

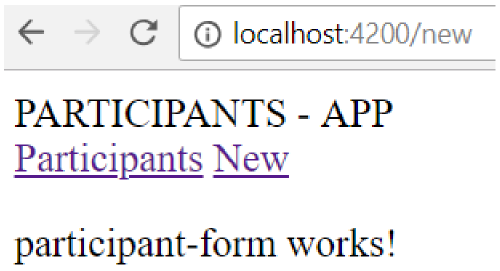


localhost:4200/participants

PARTICIPANTS - APP
Participants New

name-list works!

detail works!



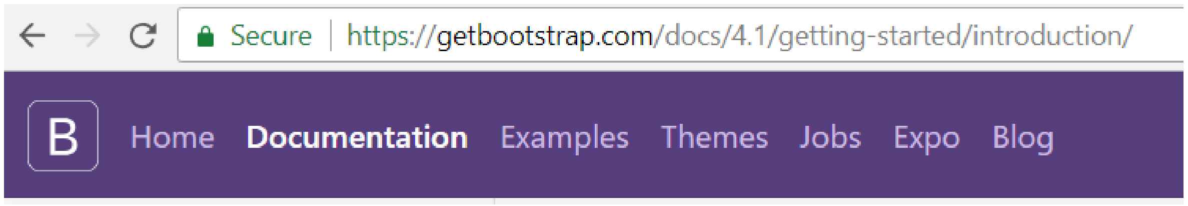
localhost:4200/new

PARTICIPANTS - APP
Participants New

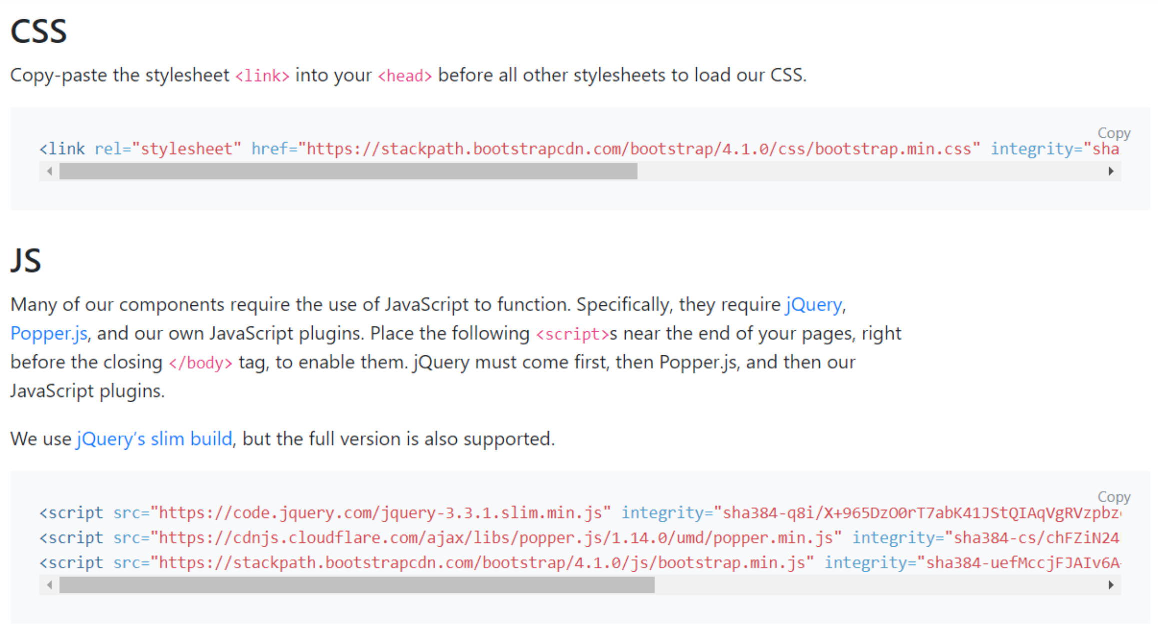
participant-form works!

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.



Secure | https://getbootstrap.com/docs/4.1/getting-started/introduction/



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
```

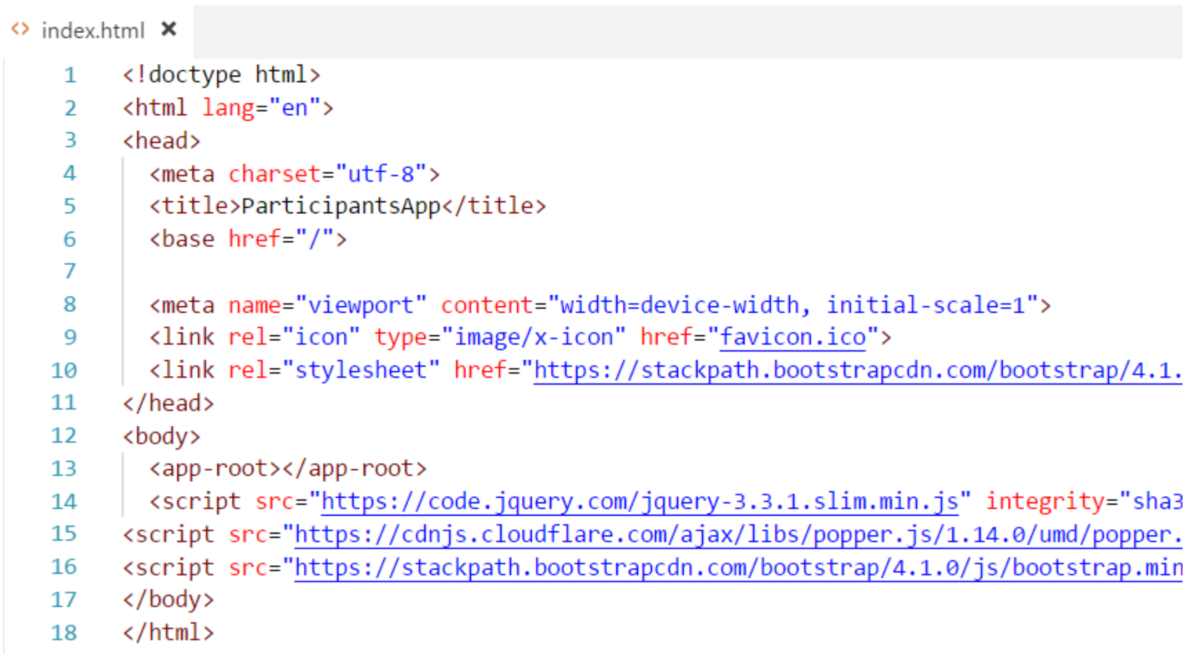
JS

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.

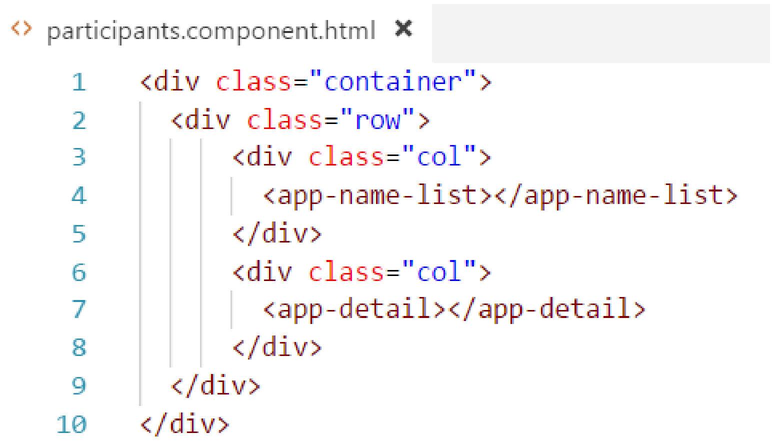
```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbz
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.min.js" integrity="sha384-cs/chFZiN24
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.min.js"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

The index.html file should look like this.



```
<!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>
```

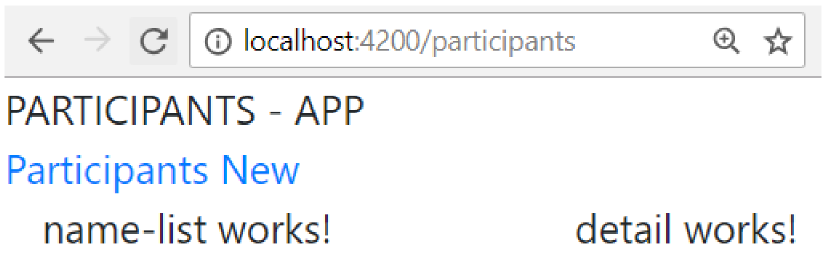
Use “`container`”, “`row`“ and “`col`“ classes to organize the components.



participants.component.html

```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.

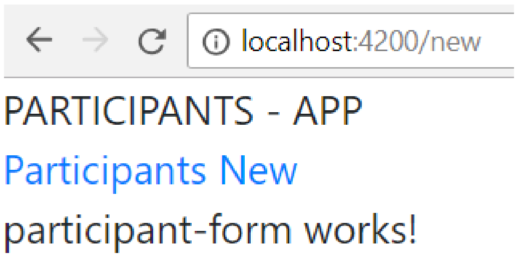


```
localhost:4200/participants
```

PARTICIPANTS - APP

Participants New

name-list works!   detail works!



```
localhost:4200/new
```

PARTICIPANTS - APP

Participants New

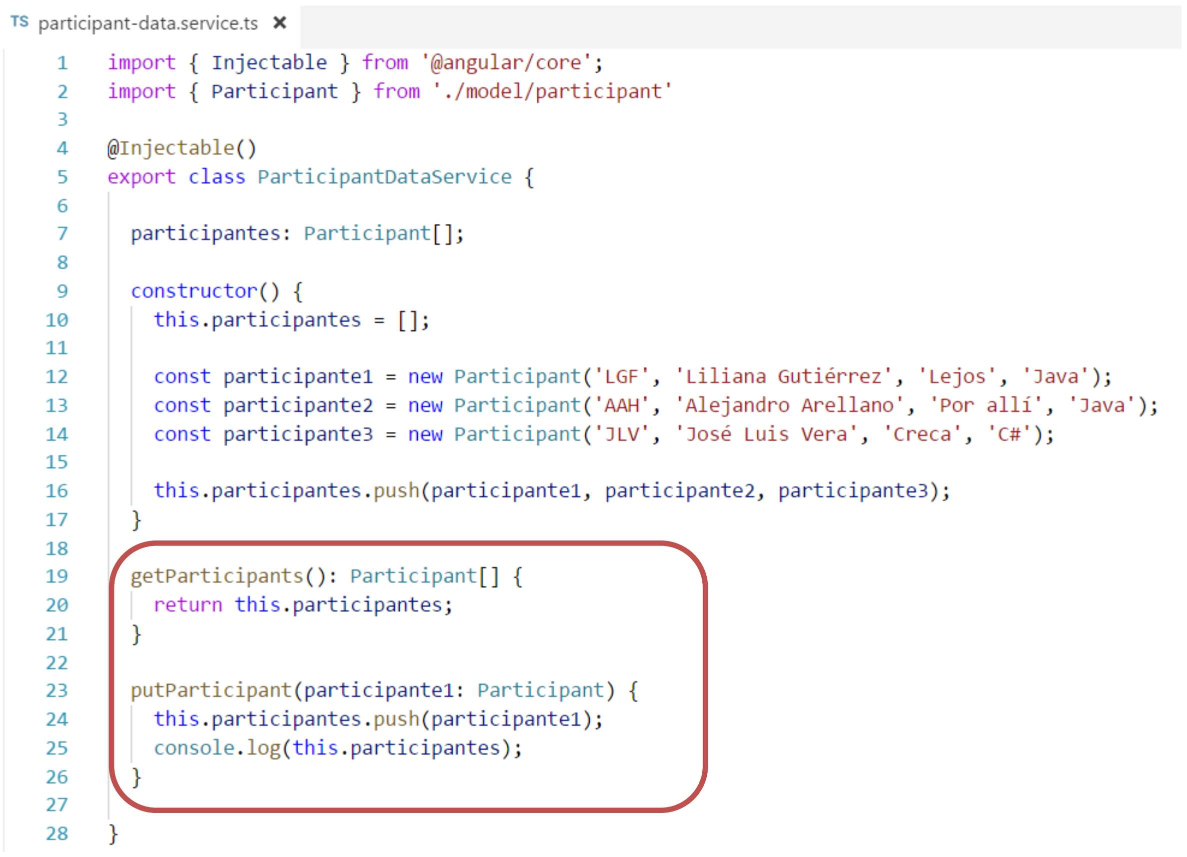
participant-form works!

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
```

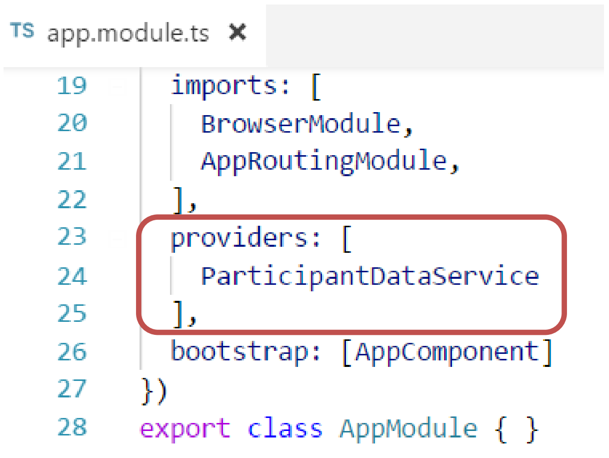
Modify the service to add the `getParticipants()` and `putParticipant()` methods.



participant-data.service.ts

```typescript
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);
  }

}
```

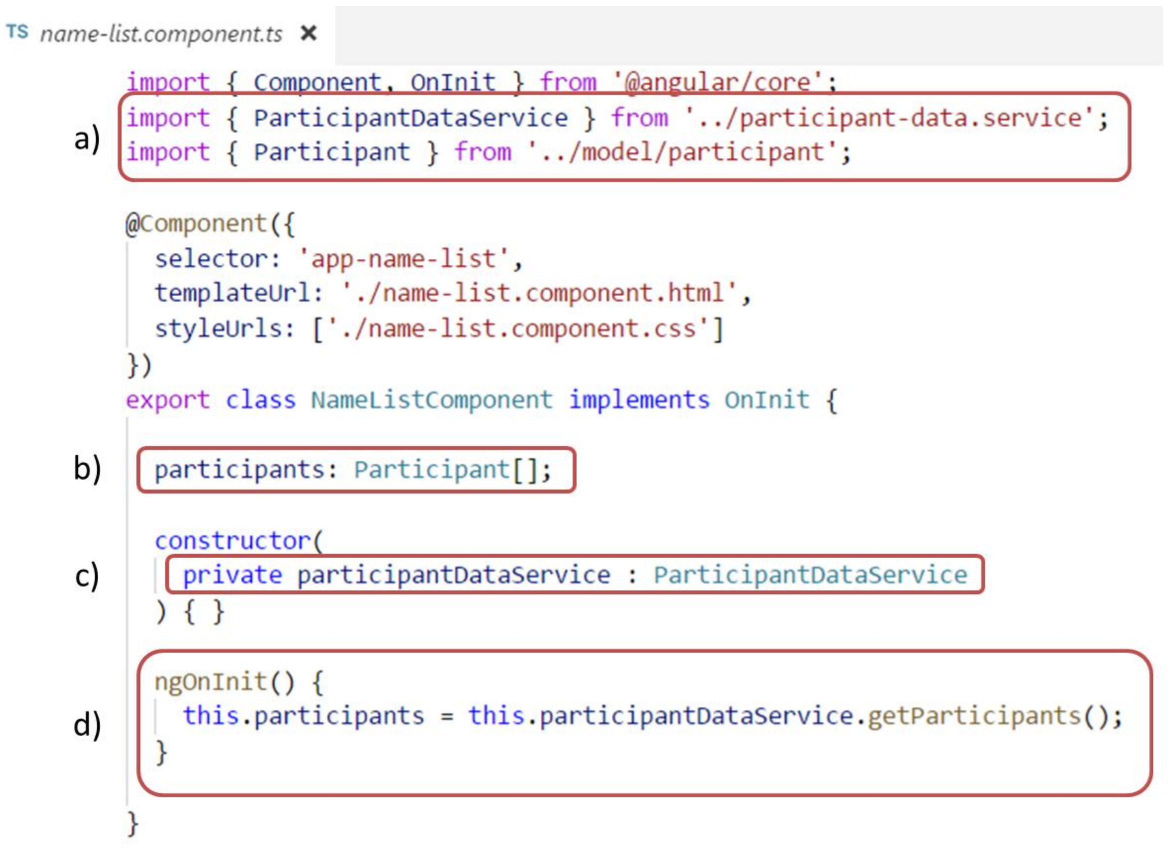
Add the service as a provider in the `app.module.ts` file.


TS app.module.ts

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

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

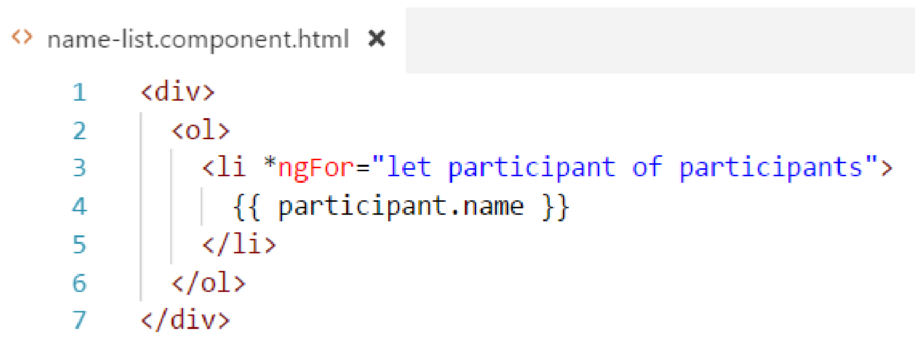
a) Import the Participant model and the ParticipantDataService in the `name-list.component.ts` file.
b) Declare the participants array.
c) Instantiate the service on the constructor.
d) Get the participant list from the service.


TS name-list.component.ts

a)
b)
c)
d)

```
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 {

  participants: Participant[];

  constructor(
    private participantDataService : ParticipantDataService
  ) { }

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

}
```

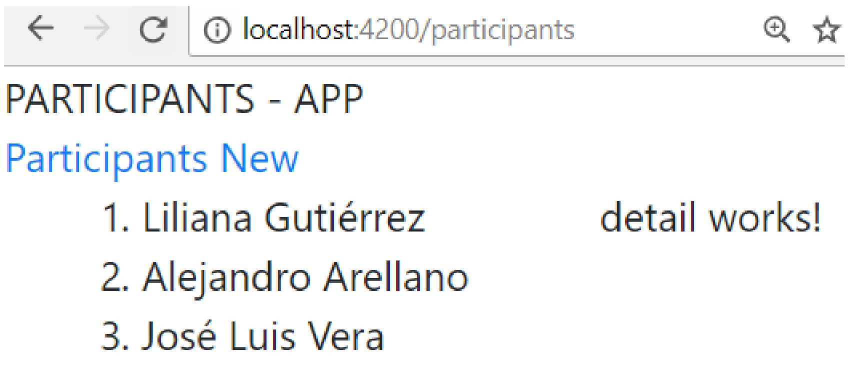
e) Display the `participants` array in the html file.



name-list.component.html

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

☑ Verify your application works as expected.



localhost:4200/participants

PARTICIPANTS - APP

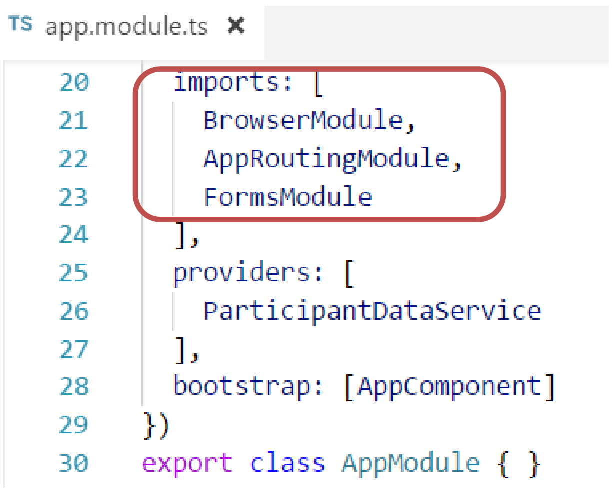
Participants New

1. Liliana Gutiérrez
2. Alejandro Arellano
3. José Luis Vera

detail works!

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.

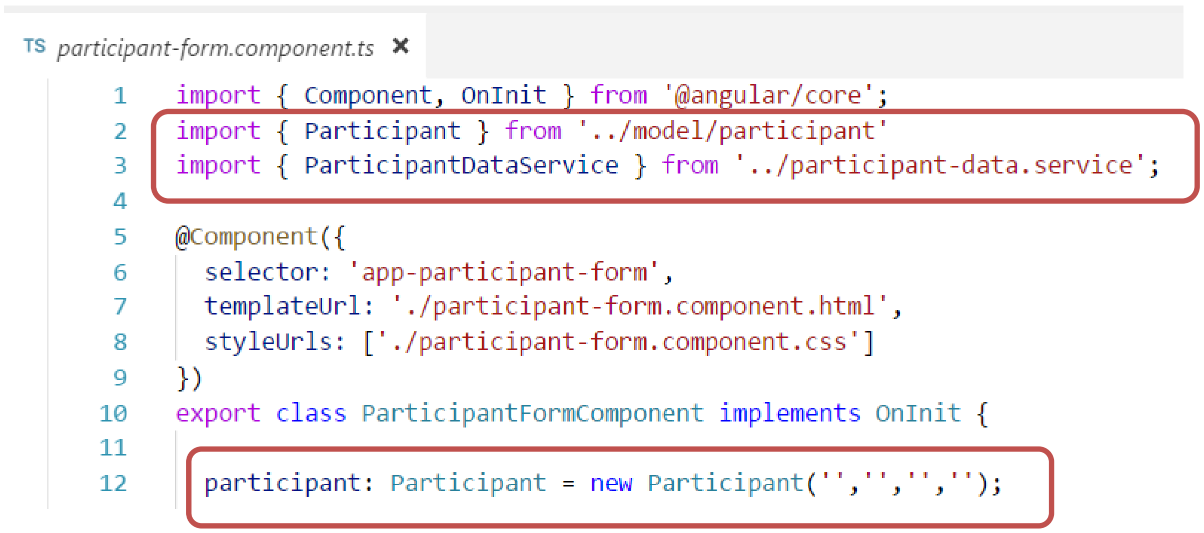


app.module.ts

```
  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.



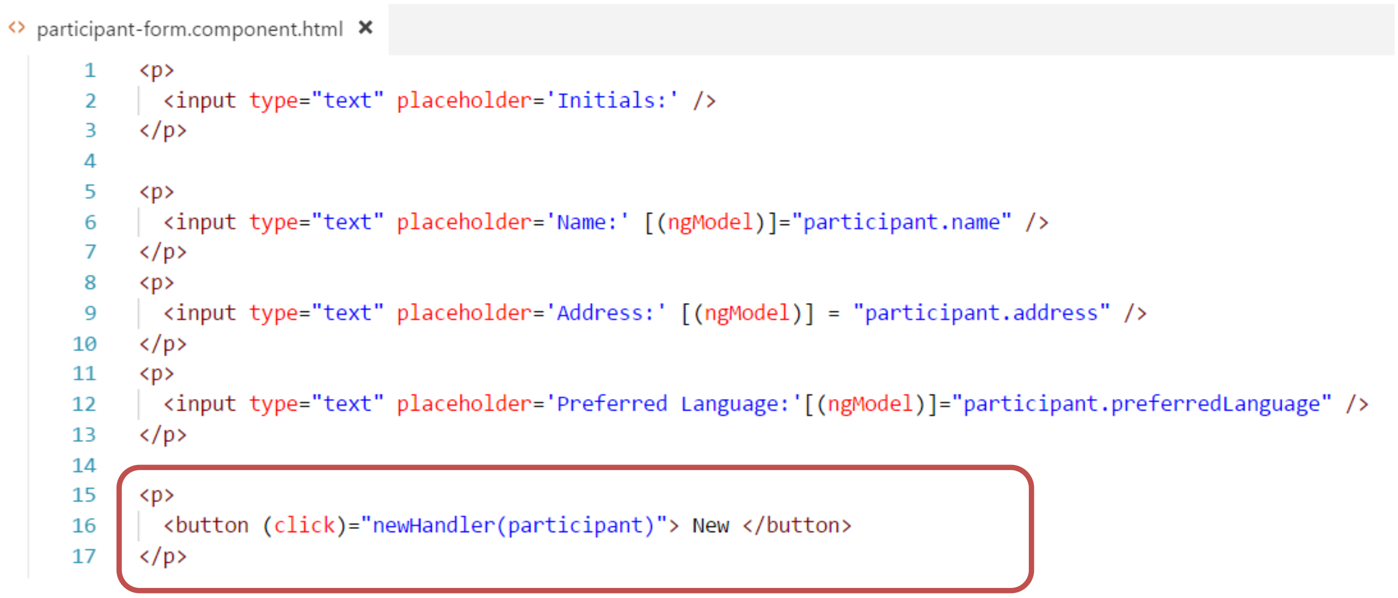
participant-form.component.ts

```
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.

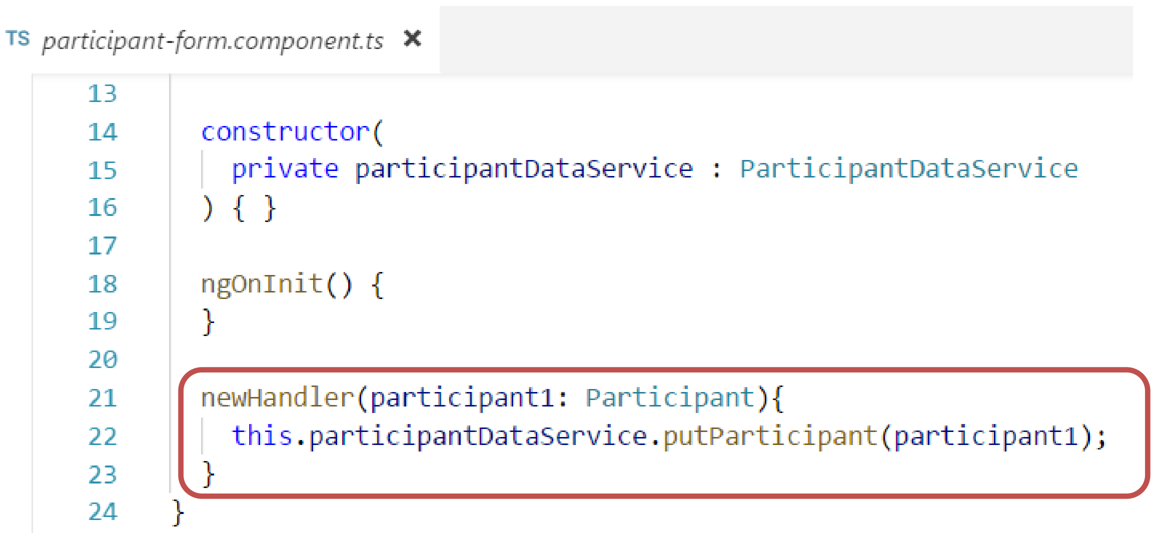
f) Add a button and use the (click) to catch the click.



participant-form.component.html

```
<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>
```

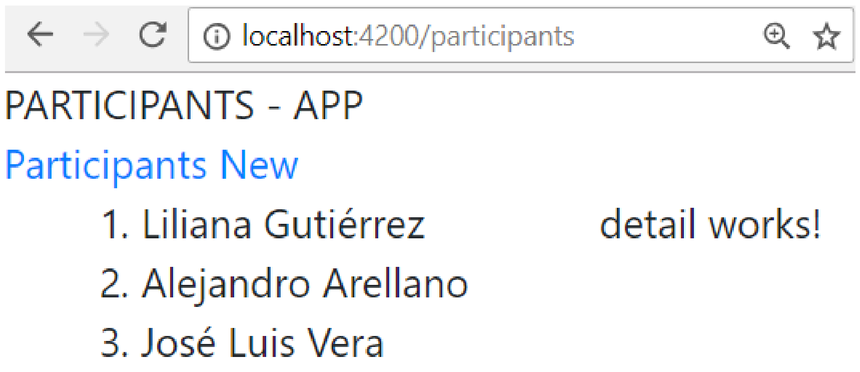
g) Instantiate the service on the constructor.
h) Implements the `newHandler` method and use the service to put the data.



participant-form.component.ts

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

  ngOnInit() {
  }

  newHandler(participant1: Participant){
    this.participantDataService.putParticipant(participant1);
  }
}
```

✓ Verify your application works as expected.



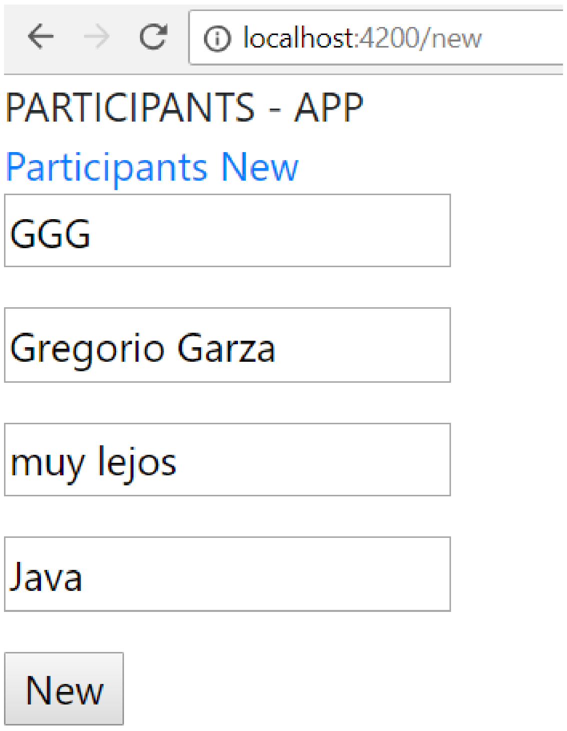
localhost:4200/participants

PARTICIPANTS - APP

Participants New

1. Liliana Gutiérrez
2. Alejandro Arellano
3. José Luis Vera

detail works!


localhost:4200/new

PARTICIPANTS - APP

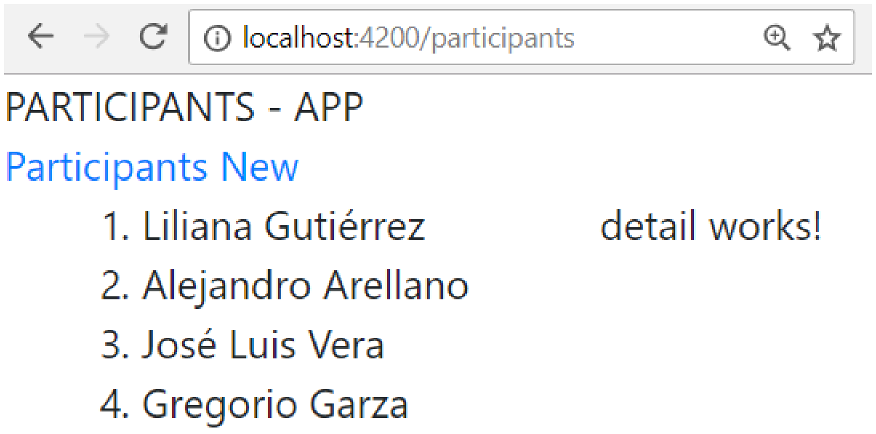
Participants New

GGG

Gregorio Garza

muy lejos

Java


localhost:4200/participants

PARTICIPANTS - APP

Participants New

1. Liliana Gutiérrez
2. Alejandro Arellano
3. José Luis Vera
4. Gregorio Garza

detail works!

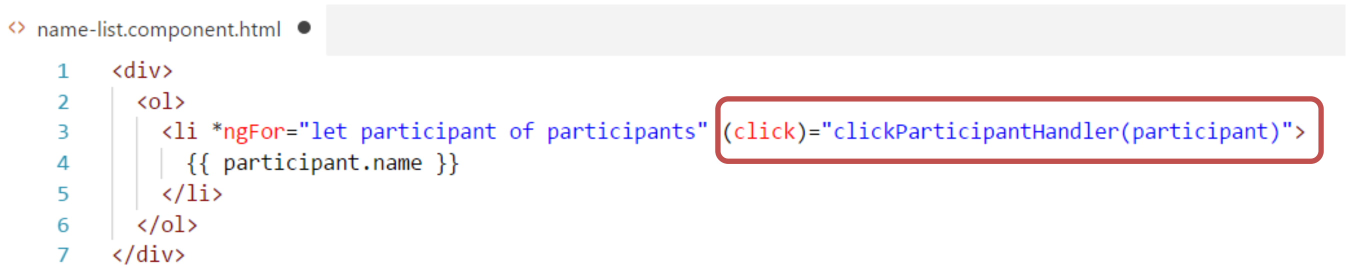
10. Display data from another component.

Emit data from child to parent.

Child: name-list.componet

Parent: participants.componet

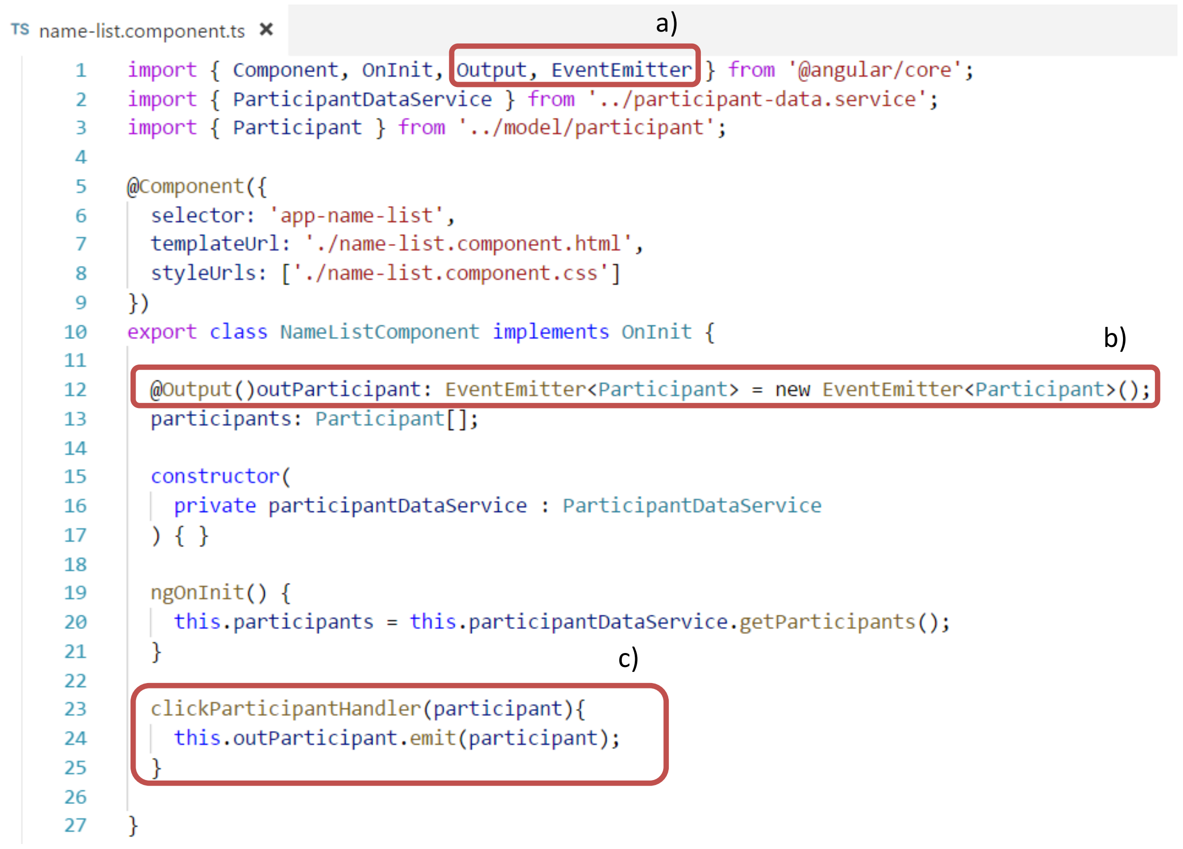
Add the (click) directive to name-list.component.html file (child) to catch the participant clicked.



name-list.component.html

```html
<div>
  <ol>
    <li *ngFor="let participant of participants" (click)="clickParticipantHandler(participant)">
      {{ participant.name }}
    </li>
  </ol>
</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.



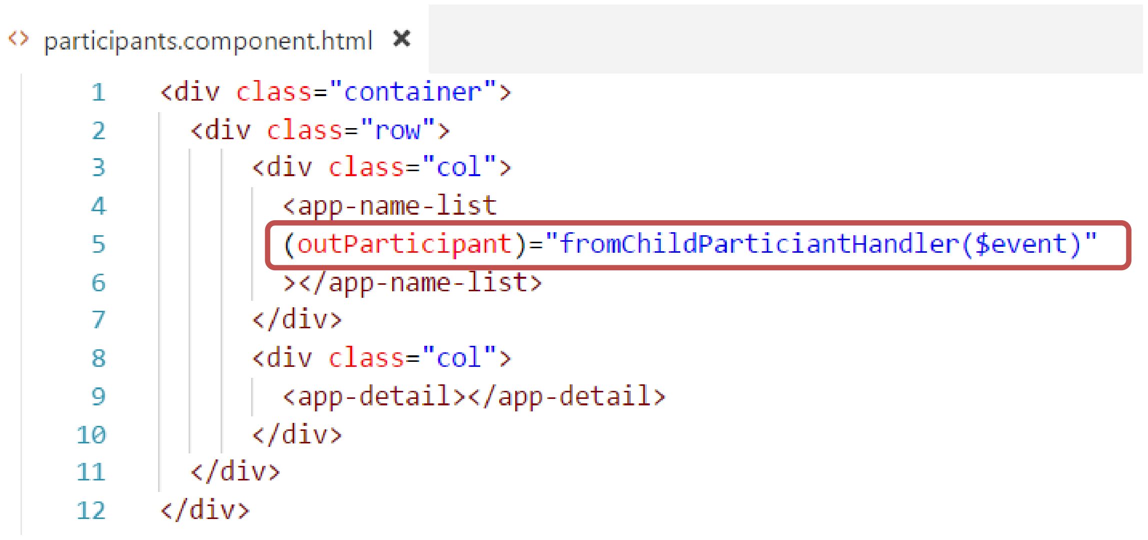
```
import { Component, OnInit, Output, EventEmitter } from '@angular/core';
import { ParticipantDataService } from '../participant-data.service';
import { Participant } from '../model/participant';

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

  @Output()outParticipant: EventEmitter<Participant> = new EventEmitter<Participant>();
  participants: Participant[];

  constructor(
    private participantDataService : ParticipantDataService
  ) { }

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

  clickParticipantHandler(participant){
    this.outParticipant.emit(participant);
  }

}
```

<u>Receive data from child.</u>

Child: name-list.componet

Parent: participants.componet

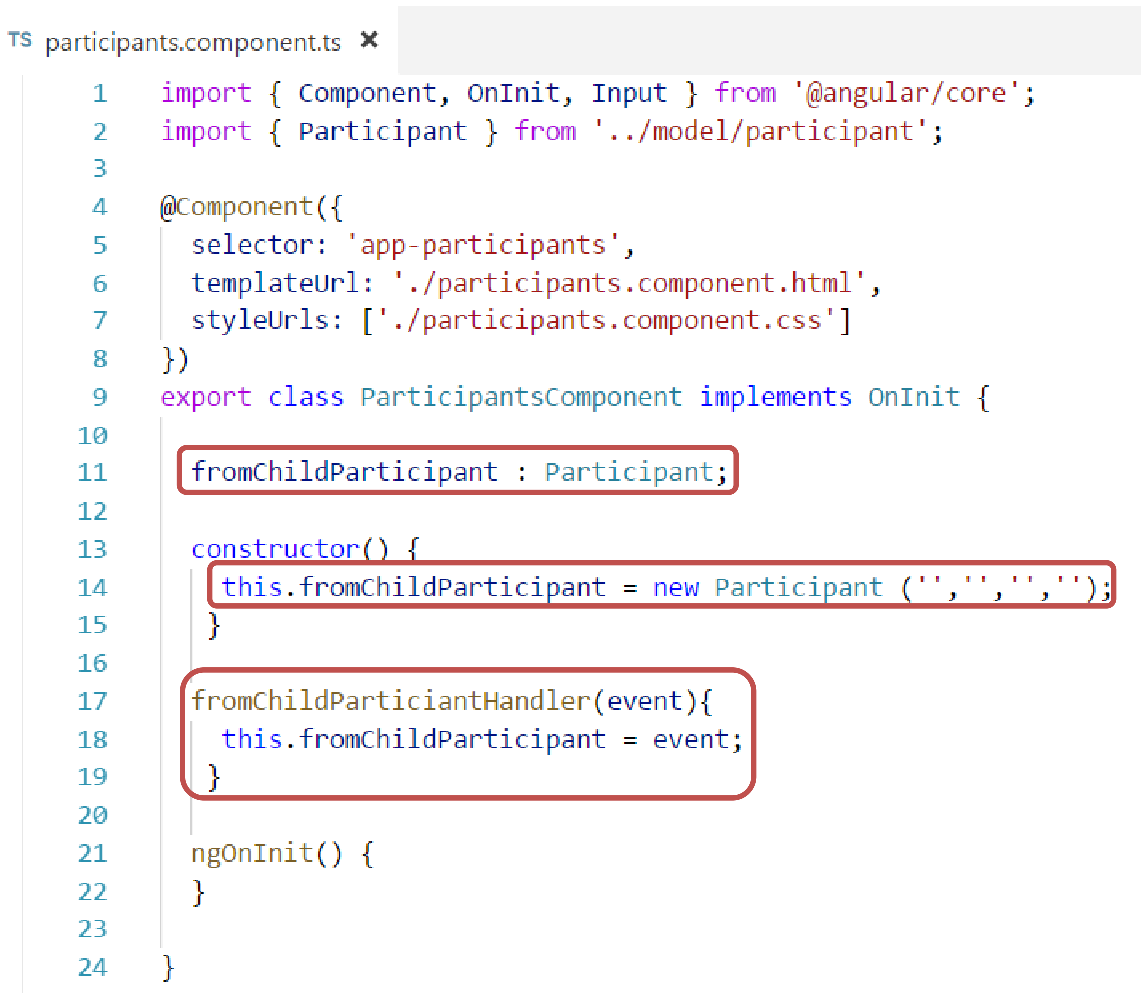
Receive the data in the html file and catch the event.



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></app-detail>
      </div>
  </div>
</div>
```

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



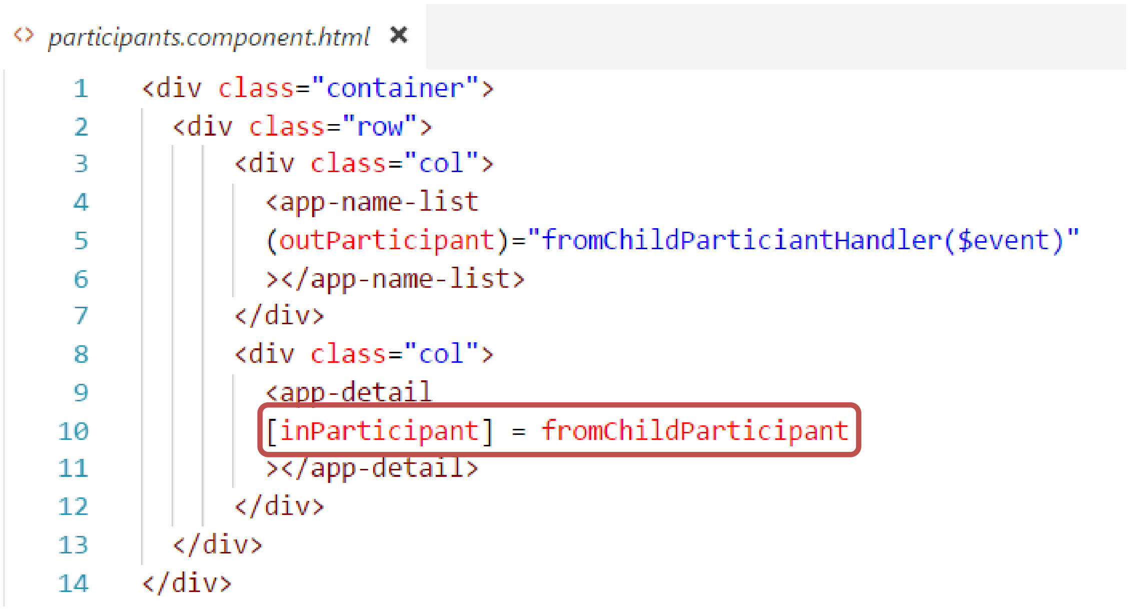
participants.component.ts

```
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() {
  }

}
```

<u>Pass data from parent to child.</u>

Child: detail.component

Parent: participants.componet



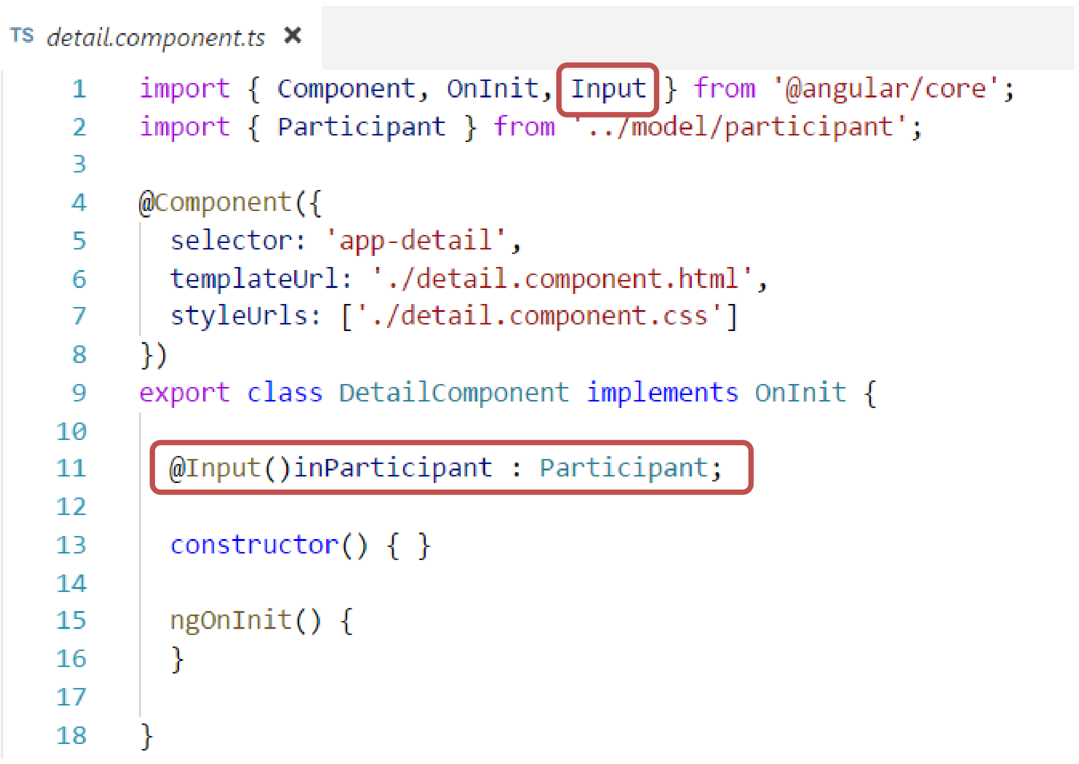
participants.component.html

```
<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>
```

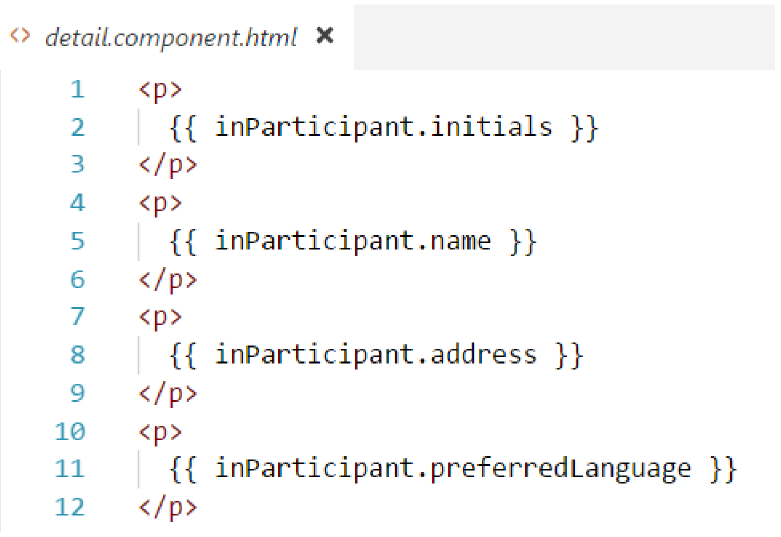
<u>Receive and use the data in the child.</u>

Child: detail.component

Parent: participants.componet


detail.component.ts

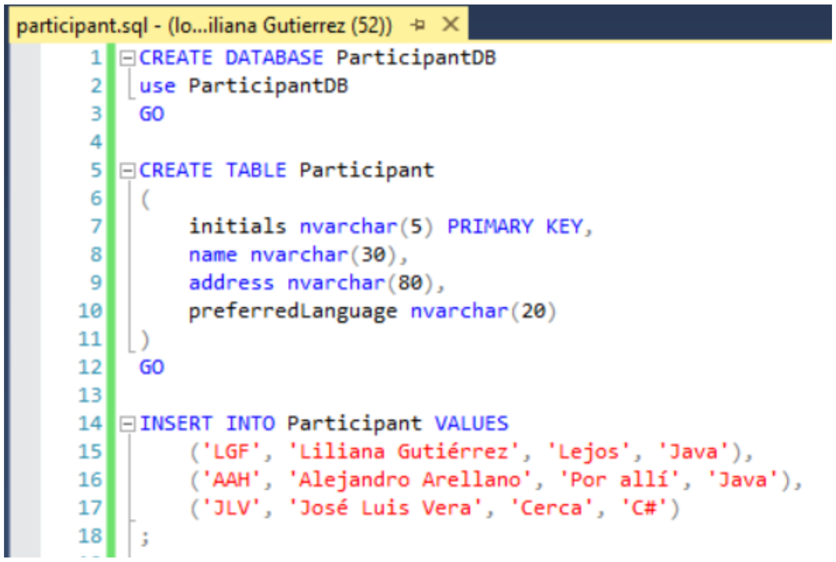
```
import { Component, OnInit, Input } from '@angular/core';
import { Participant } from '../model/participant';

@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

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

✓ Verify your application works as expected.

11. Use SSMS to create the ParticipantDB database and the Participant table.

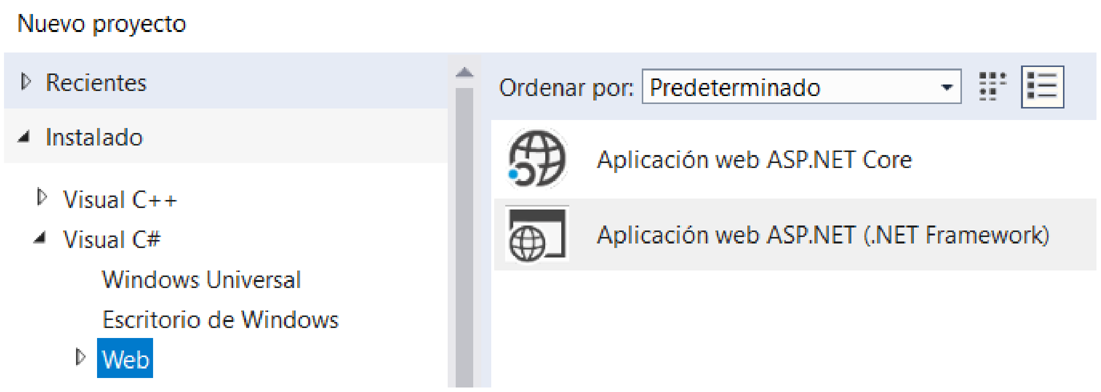


participant.sql - (lo...iliana Gutierrez (52))

```sql
CREATE DATABASE ParticipantDB
use ParticipantDB
GO

CREATE TABLE Participant
(
    initials nvarchar(5) PRIMARY KEY,
    name nvarchar(30),
    address nvarchar(80),
    preferredLanguage nvarchar(20)
)
GO

INSERT INTO Participant VALUES
    ('LGF', 'Liliana Gutiérrez', 'Lejos', 'Java'),
    ('AAH', 'Alejandro Arellano', 'Por allí', 'Java'),
    ('JLV', 'José Luis Vera', 'Cerca', 'C#')
;
```

12. Use MSVS to create a Web API to get info from the DB.

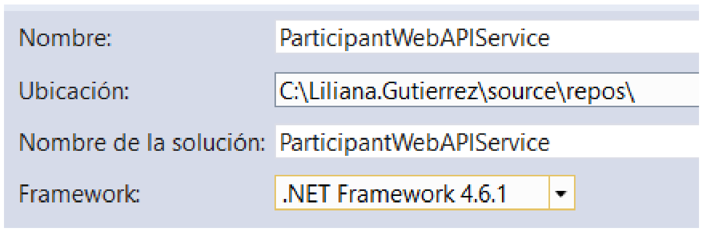
a. Create the ASP.NET Application



Nuevo proyecto

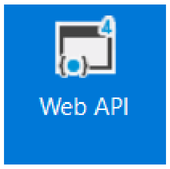
Recientes
Instalado
Visual C++
Visual C#
Windows Universal
Escritorio de Windows
Web
Ordenar por: Predeterminado
Aplicación web ASP.NET Core
Aplicación web ASP.NET (.NET Framework)

Name it ParticipantWebAPIService



Nombre:	ParticipantWebAPIService
Ubicación:	C:\Liliana.Gutierrez\source\repos\
Nombre de la solución:	ParticipantWebAPIService
Framework:	.NET Framework 4.6.1

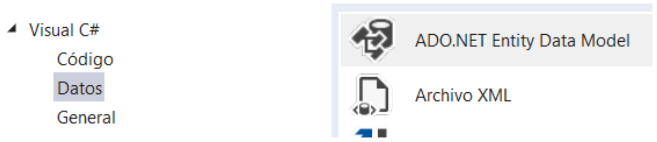
Select Web API



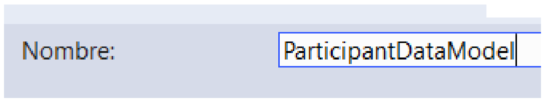
a. Create the model

Add a new ADO.NET Entity Data Model:

- (right-click) the models folder, Add, New Item…
- Select Data and ADO.NER Entity Data Model

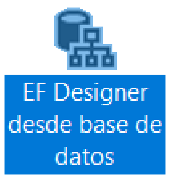


- Name it ParticipantDataModel

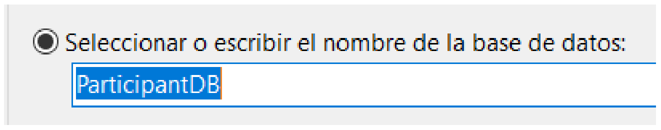


Nombre: ParticipantDataModel

- Select EF Designer from database



- Connect to the local server (.) and select de ParticipantDB you just create.



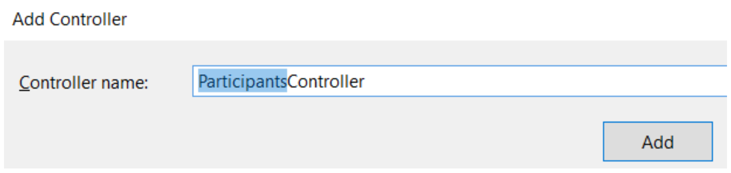
(●) Seleccionar o escribir el nombre de la base de datos:
ParticipantDB

- Select the Participant table to include it in the model.7

b. Add a controller

(rigth-click) the Controllers folder and select Add, and then select Web API 2 Controller Empty

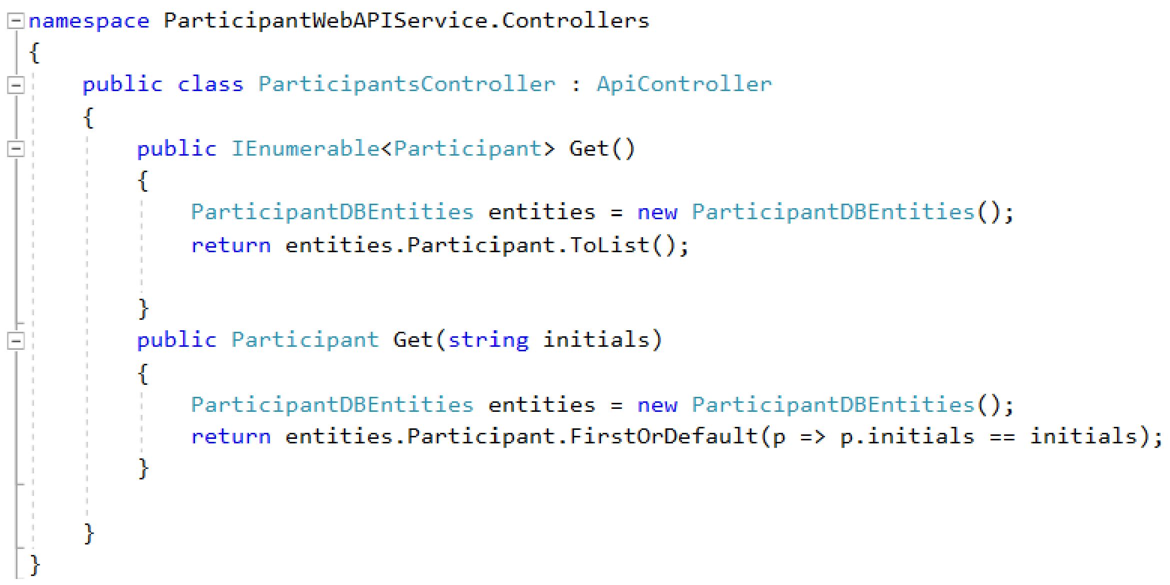
Name it ParticipantsController



Add Controller

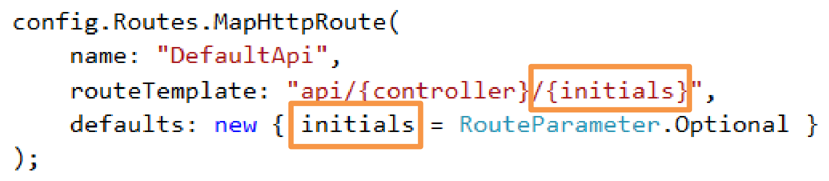
Controller name: ParticipantsController

The code of the ParticipantsController should look like this



```
namespace ParticipantWebAPIService.Controllers
{
    public class ParticipantsController : ApiController
    {
        public IEnumerable<Participant> Get()
        {
            ParticipantDBEntities entities = new ParticipantDBEntities();
            return entities.Participant.ToList();

        }
        public Participant Get(string initials)
        {
            ParticipantDBEntities entities = new ParticipantDBEntities();
            return entities.Participant.FirstOrDefault(p => p.initials == initials);
        }

    }
}
```

Change the default routes in the file WebApiConfig.cs from `id` to `initials`.

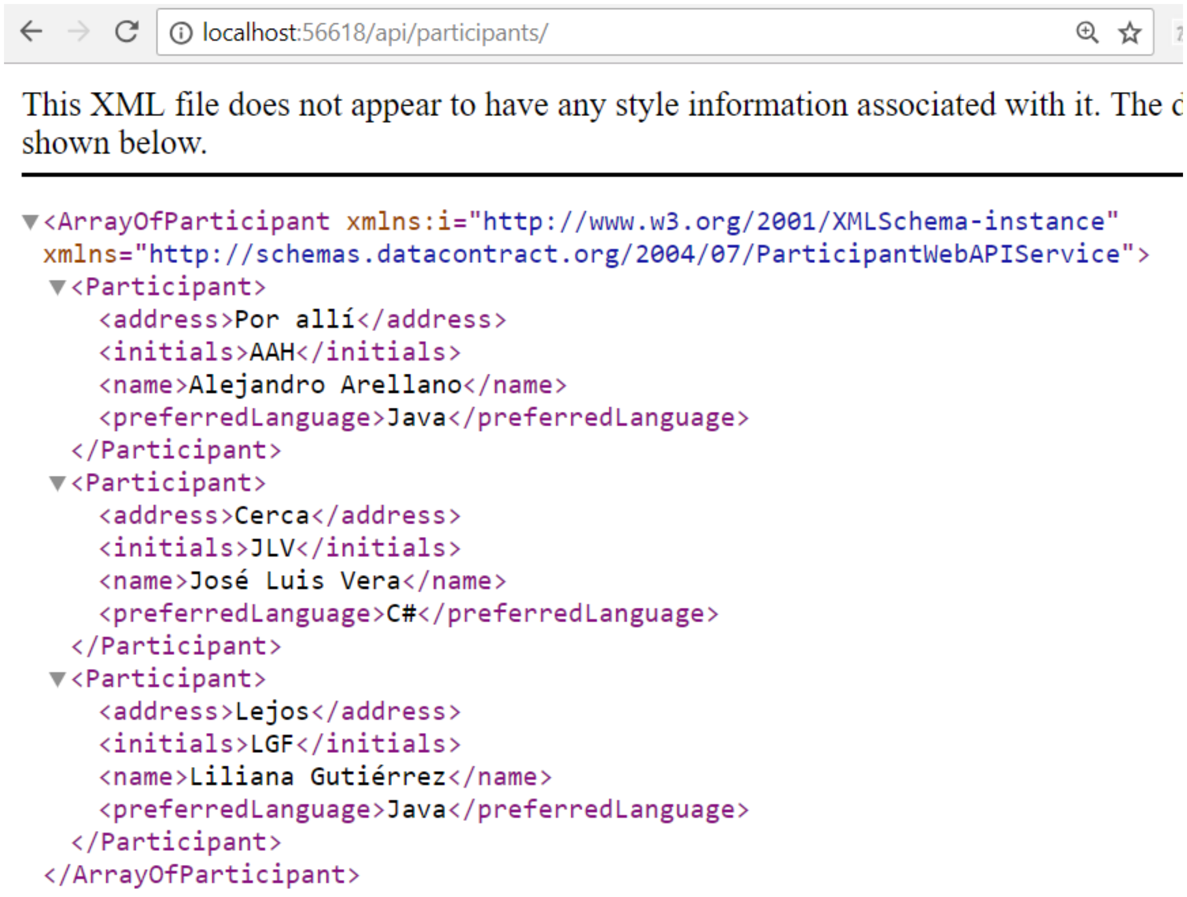


```
config.Routes.MapHttpRoute(
    name: "DefaultApi",
    routeTemplate: "api/{controller}/{initials}",
    defaults: new { initials = RouteParameter.Optional }
);
```

Add the following to the Web.config file

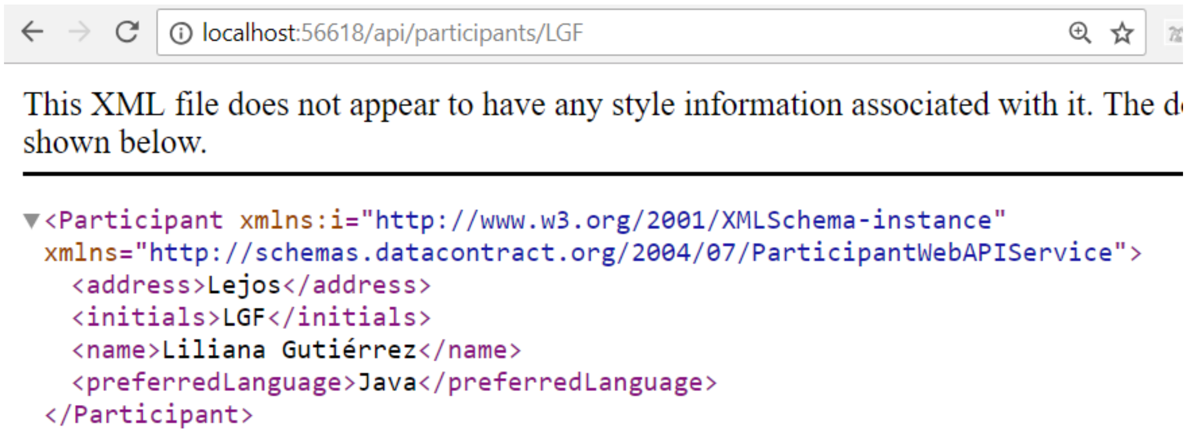
```
<system.webServer>
  <httpProtocol>
  <customHeaders>
   <add name="Access-Control-Allow-Origin" value="*" />
   <add name="Access-Control-Allow-Headers" value="Content-Type" />
   <add name="Access-Control-Allow-Methods" value="GET, POST, PUT, DELETE" />
 </customHeaders>
 </httpProtocol>
```

☑ Use the browser to verify your application works as expected.


localhost:56618/api/participants/

This XML file does not appear to have any style information associated with it. The d
shown below.

```
▼<ArrayOfParticipant xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://schemas.datacontract.org/2004/07/ParticipantWebAPIService">
  ▼<Participant>
     <address>Por allí</address>
     <initials>AAH</initials>
     <name>Alejandro Arellano</name>
     <preferredLanguage>Java</preferredLanguage>
   </Participant>
  ▼<Participant>
     <address>Cerca</address>
     <initials>JLV</initials>
     <name>José Luis Vera</name>
     <preferredLanguage>C#</preferredLanguage>
   </Participant>
  ▼<Participant>
     <address>Lejos</address>
     <initials>LGF</initials>
     <name>Liliana Gutiérrez</name>
     <preferredLanguage>Java</preferredLanguage>
   </Participant>
 </ArrayOfParticipant>
```


localhost:56618/api/participants/LGF

This XML file does not appear to have any style information associated with it. The d
shown below.

```
▼<Participant xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://schemas.datacontract.org/2004/07/ParticipantWebAPIService">
   <address>Lejos</address>
   <initials>LGF</initials>
   <name>Liliana Gutiérrez</name>
   <preferredLanguage>Java</preferredLanguage>
 </Participant>
```

13. **Use the HTTP module to call the ASP.NET Web API service.**

a. **Import the HttpModule in the app.module.ts file**

```
import { HttpModule } from '@angular/http';

imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    HttpModule
],
```

In the participant-data.service.ts file.

b. **Do the following imports.**

```
import { Http, Response } from '@angular/http';
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/operator/map'
```

c. **Use the constructor to inject[6] the service.**

```
constructor(
    private _http: Http
) { }
```

Remember that this short-notation means:

- The creation of the private variable _http
- The creation the constructor parameter with the same name.
- The initialization of this._http with the parameter.

d. **Use the _http to issue Web-service calls.**

Modify the `getParticipants()` method, so it now returns an observable.

[6] Inject the service so far means, to have an instance of that service.

```
@Injectable()
export class ParticipantDataService {

  urlWebAPI= "http://localhost:56618/api/participants/";
  headers: Headers = new Headers({
    'Content-Type': 'application/json'
   });
  options = new RequestOptions({ headers: this.headers });

  constructor(
    private _http: Http
  ) { }

  getParticipants(): Observable<Participant[]> {
    let observableParticipants = this._http.get(this.urlWebAPI)
        .map((response: Response) => <Participant[]>response.json());
    return observableParticipants;
  }
```

Comment by now the `putParticipant` method.

e. Subscribe to the service in the name-list.component.ts.

```
ngOnInit() {
  this.participantDataService.getParticipants()
    .subscribe((participantsData)=>this.participants = participantsData);
}
```

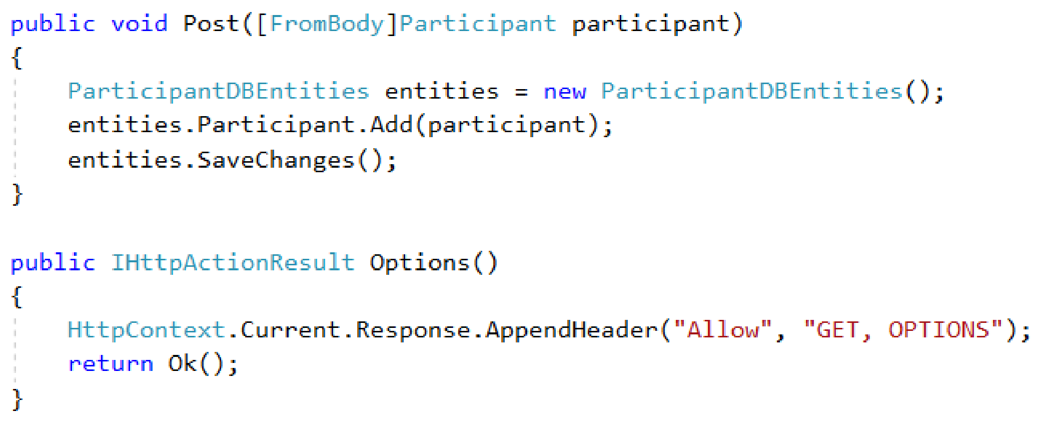
f. Comment by now the use of the putParticipant() method in the participant-form.component.ts.

```
newHandler(participant1: Participant){
  //this.participantDataService.putParticipant(participant1);
}
```

☑ Run the Web API and make sure your angular app is served. Then Insert a participant record directly in the DB and verify you get the data in the browser.

14. **Modify the Web API to post info to the DB.**

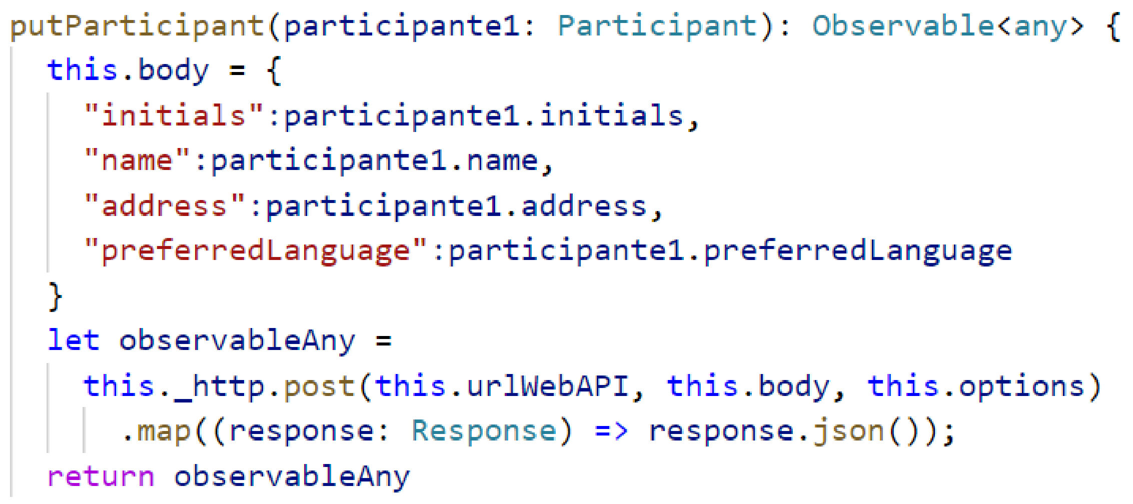
   a. **Add the following to the ParticipantsController**



```
public void Post([FromBody]Participant participant)
{
    ParticipantDBEntities entities = new ParticipantDBEntities();
    entities.Participant.Add(participant);
    entities.SaveChanges();
}

public IHttpActionResult Options()
{
    HttpContext.Current.Response.AppendHeader("Allow", "GET, OPTIONS");
    return Ok();
}
```

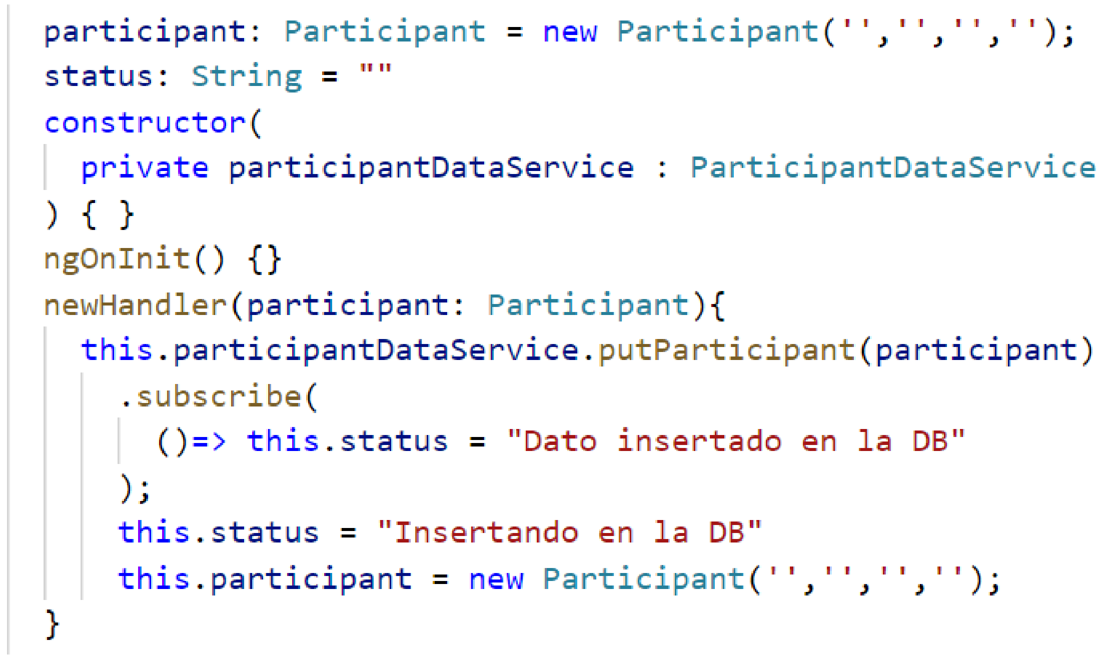
15. **In the angular application,**

   a. **modify the putParticipant method in the participant-data.service.ts file.**



```
putParticipant(participante1: Participant): Observable<any> {
  this.body = {
    "initials":participante1.initials,
    "name":participante1.name,
    "address":participante1.address,
    "preferredLanguage":participante1.preferredLanguage
  }
  let observableAny =
    this._http.post(this.urlWebAPI, this.body, this.options)
      .map((response: Response) => response.json());
  return observableAny
```

   b. **Subscribe to the service in the participant-form.component.ts.**



```
participant: Participant = new Participant('','','','');
status: String = ""
constructor(
  private participantDataService : ParticipantDataService
) { }
ngOnInit() {}
newHandler(participant: Participant){
  this.participantDataService.putParticipant(participant)
    .subscribe(
      ()=> this.status = "Dato insertado en la DB"
    );
    this.status = "Insertando en la DB"
    this.participant = new Participant('','','','');
}
```