

# Angular 12

(Part-4)

by

Anand Kulkarni

[anand.kulkarni1@zensar.com](mailto:anand.kulkarni1@zensar.com)

# Contents

Module	Topic
Module 1	Observables / RxJS
Module 2	Routes

# Angular communication with server

There are 3 ways to handle REST call:

- **Callbacks:** Handling REST call using callback function is very preliminary way of communication. Once REST call sends response, suitable success / error function is called where developer can take certain action on response.
- **Promises:** A promise represents a value that we can handle at some point in the future. Promises allow to execute the action multiple times for a single event & hence Promises are preferred over Callbacks.
- **Observables:** Observables open up a continuous channel of communication in which multiple values of data can be emitted over time. Every observable is a promise plus advance features.

# Angular support for REST communication

Angular provides a separate module called 'HttpClientModule' in order to communicate with server.

Here are the steps to include http support in Angular application:

1. Include HttpClientModule into app.module.ts

```
import { HttpClientModule } from '@angular/common/http';  
  
imports: [ BrowserModule, HttpClientModule ],
```

2. Inject Http service in your service class

```
import { HttpClient, Response } from '@angular/common/http';  
  
export class ProductService {  
  constructor(private _http: HttpClient) {  
  }  
}
```

3. Invoke REST call using http service: `this._http.get('http://localhost:8000/product');`

# What is an Observable?

- Observable isn't an Angular specific feature, but rather a proposed standard for managing async data that will be included in the release of ES7.
- Observable is a sequence of items that arrive async over the time. However, with 'http' service calls it is always a single item also known as `http_response`.
- Since, Observable feature is not available in ES6 specification, we use Observable from a third party library called RxJS.

# RxJS

- RxJS stands for Reactive Extensions for JavaScript.
- Observable feature is not available in ES6 specification. Hence, we use Observable from a third party library called RxJS.
- In order to download RxJS in angular application, make sure you package.json has below dependency added-

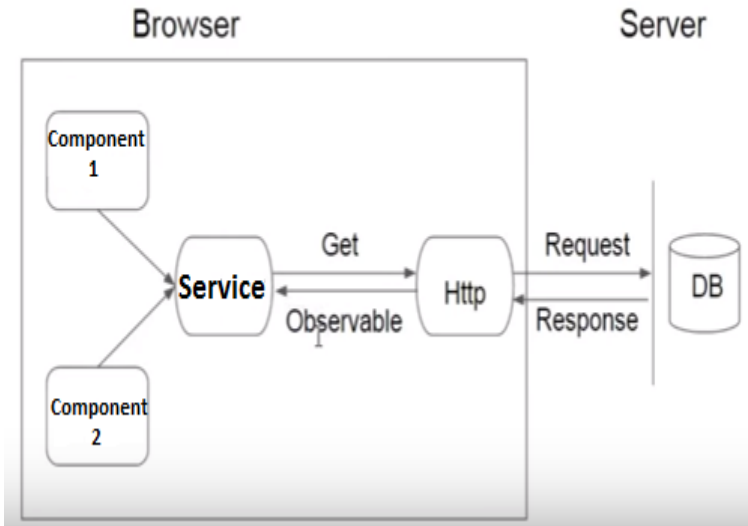
```
"dependencies": {
```

```
  "rxjs": "6.6.0"
```

```
}
```

# Using Observable in Angular App

Http



## ***Steps to use Observable in Angular App:***

1. Make http call from Service.
2. Receive observable.
3. Subscribe to observable in your components.
4. Render the view using received data.

# Observables & Async streams



- Angular service is responsible to send REST request to server & return Observable to respective components.
- Angular component will retrieve Observable from service & subscribe to it.
- It is possible that for one observable there can be multiple subscribers.



# Using Observable in Angular App

**app.module.ts**

```
import { HttpClientModule } from '@angular/common/http';
```

```
@NgModule({
```

```
  imports: [ BrowserModule, FormsModule, ReactiveFormsModule, HttpClientModule ],
```

```
})
```

# Using Observable in Angular App

## ProductService.ts

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

@Injectable()
export class ProductService {
    constructor(private _http: HttpClient) {
    }
    getProducts():Observable<Object>{
const headers = { 'content-type': 'application/json','Access-Control-Allow-Origin': '*'};
        return this._http.get(this.url , {headers: headers});
    }
}
```

# Using Observable in Angular App

## ProductComponent.ts

```
import { ProductService } from './products.service';
@Component({
    providers: [ProductService]
})
export class ProductComponent {
    public products;
    constructor(productService: ProductService) {
        productService.getProducts().subscribe((data)=> {
            this.products=data;
            console.log('Received products: ', this.products)
        });
    }
}
```

# Using Observable for POST call

## ProductService.ts

```
createProduct(product: Product) {  
  const headers = { 'content-type': 'application/json', 'Access-Control-Allow-Origin': '*' };  
  return this._httpClient.post(URL, JSON.stringify(product), {headers: headers});  
}
```

# Promises vs Observable

Sr. No.	Promise	Observable
1.	Promise cannot be cancelled.	Observable can be cancelled.
2.	Promise cannot be retried.	Observables can be retried using <code>retry()</code> or <code>retryWhen()</code> functions.
3.	Promise is a request with single return value.	Observable is a request that can return multiple response as an async stream.

# What is a Router?

An Angular application is a collection of multiple components & you need to switch from one component to another based upon action performed by end user. Thus, Angular Router will help to navigate from one angular component to another.

# Steps to introduce Router in Angular App

- Set the `<base>` tag into `index.html`. It will help your application to understand how to construct url's while navigation.

`<head>`

`<base href="/">`

- Import `RouterModule` into `AppModule` & mention routing details.

```
import { RouterModule } from '@angular/router';  
imports: [ BrowserModule,  
           RouterModule.forRoot(  
             {path: 'first', component: FirstComponent},  
             {path: 'second', component: SecondComponent}  
           )  
        ]
```

# Steps to introduce Router in Angular App

- Add 'active' class in styles.css

```
nav a.active {  
    color: orange;  
}
```

- Finally in the navigation component, provide the links & specify router outlet to render the required component.

```
<nav>  
    <a routerLink="/first" routerLinkActive="active" >First</a>  
    <a routerLink="/second" routerLinkActive="active" >Second</a>  
</nav>  
<router-outlet></router-outlet>
```



# Route parameters

When navigating from one to other component, the current component may wish to send few parameters to another component. Here are the steps to make it possible:

- Register route url with parameter into app.module.ts

```
RouterModule.forRoot([  
    {path: 'fourth/:name', component: FourthComponent}    ])
```

- Pass the parameter while source component is navigating to target component:

```
import { Router } from '@angular/router';  
export class ThirdComponent {  
    constructor(private router: Router) {}  
    onClick(){        this.router.navigate(['/fourth', 'Anand']);    }  
}
```

# Route parameters continue...

Finally, read the supplied parameter into target component:

```
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: '<fourth-comp>',
  template: '<h3>Fourth Component says Hello {{name}}</h3>'
})
export class FourthComponent implements OnInit {
  name: string;
  constructor(private activatedRoute: ActivatedRoute) {}

  ngOnInit() {
    this.name = this.activatedRoute.snapshot.params['name'];
  }
}
```

*Thank you!!*