Angular 16 Online Training

sahosoft
Online Learning Platform

By: Chandan Kumar

sahosoft
Online Learning Platform

Angular 16 : Online Class

Observables and Subscribe

By: Sahosoft Solutions

Presented by : Chandan Kumar

sahosoft
Online Learning Platform

Observables and Subscribe

Observable belongs to **RxJS** library. To perform asynchronous programming in Angular application we can use Observable.

When we send and receive data over HTTP, we need to deal it asynchronously because fetching data over HTTP may take time. Observable is **subscribed** by using **async pipe** or by using **subscribe** method.

Observable is a class of RxJS library. **RxJS** is ReactiveX library for JavaScript that performs reactive programming.

Observable plays most basic role in reactive programming with RxJS.

Some methods of Observable class are subscribe, map, mergeMap, of, retry, catch, throw etc.

Observables and Subscribe

To use RxJS library in Angular programming we need to ensure that we have installed RxJS. In case we are using Angular CLI, it installs RxJS by default.

We can ensure it by checking package.json. If RxJS is not there in package.json, we can install it as following.

```
npm install rxjs –save
```

To use Observable in our Angular application, we need to import it as following.

```
import { Observable } from 'rxjs/Observable';
```

Observables and Subscribe

Angular HttpClient performs HTTP requests for the given URL. HttpClient works with Observable.

Find some of its methods.

1. get(url: string, options: {...}): Observable<any>
2. post(url: string, body: any | null, options: {...}): Observable<any>
3. put(url: string, body: any | null, options: {...}): Observable<any>
4. delete(url: string, options: {...}): Observable<any>

Observables and Subscribe

The http class provide the get() method to getting a resource, Post() for creating it, Put for updating it, delete for deleting resource and head for getting meta data regarding resource.

```
getBooksFromStore(): Observable<Book[]> {
 return this.http.get<Book[]>(this.bookUrl);
}
```

```
getsoftBooks() {
this.bookService.getBooksFromStore().subscribe(books => this.softBooks = books);
}
```

Angular 16 : Online Class

Observables and Subscribe with web API

By: Sahosoft Solutions

Presented by : Chandan Kumar

sahosoft
Online Learning Platform

Angular In-Memory Web API

To test our application we need Web Service URL. Angular provides **In-Memory Web API** that will provide Web Service URL.

We can configure URLs with dummy data using In-Memory Web API. Find the steps to use Angular In-Memory Web API.

To install Angular In-Memory Web API, run following command.

```
npm i angular-in-memory-web-api
```

Angular In-Memory Web API

Create a class that will implement InMemoryDbService. Define createDb() method with some dummy data.

```
import { InMemoryDbService } from 'angular-in-memory-web-api';
export class TestData implements InMemoryDbService {
createDb() {
    let bookDetails = [
      { id: 101, name: 'Angular by Sahosoft', category: 'Angular' },
     { id: 102, name: .net core by Sahosoft', category: 'Java' },
     { id: 103, name: 'NgRx by Sahosoft', category: 'Angular' }
   ];
 let writerDetails = { writerId: 11, writerName: 'Mahesh',
          books: [
{ id: '103', name: 'Angular Tutorial', category: 'Angular', year: '2016' },
{ id: '104', name: 'Core Java Tutorial', category: 'Java', year: '2016' }
] };

let welcomeMsg = "Welcome to the Angular world!";

return { books: bookDetails, writer: writerDetails, message: welcomeMsg };
   }
 }
```

We will get following Web Service URLs.

1. /api/books
2. /api/writer
3. /api/message

Angular In-Memory Web API

3. Before using In-Memory Web API we need to import InMemoryWebApiModule in application module and configure TestData class as following.

```
import { InMemoryWebApiModule } from 'angular-in-memory-web-api';
import { TestData } from './test-data';

@NgModule({
imports: [
------
InMemoryWebApiModule.forRoot(TestData)
],
------
})
export class AppModule { }
```

Angular 16 : Online Class

Observable + Async Pipe + NgFor

By: Sahosoft Solutions

Presented by : Chandan Kumar

sahosoft
Online Learning Platform

Observable + Async Pipe + NgFor

Angular async pipe subscribes to Observable and returns its last emitted value.

Step-1: We will create a method to fetch data over HTTP using Angular HttpClient in our service, suppose in BookService

```
getBooksFromStore(): Observable<Book[]> {
 return this.http.get<Book[]>(this.bookUrl);
}
```

Method will return Observable<Book[]>

Observable + Async Pipe + NgFor

Step-2: In our component we will create a property.

allBooks : Observable<Book[]>

Now we will call the service method and assign value to the allBooks property in our component.

```
getBooks() {
 this.allBooks = this.bookService.getBooksFromStore();
}
```

Call the above method in ngOnInit.

```
ngOnInit() {
  this.getBooks();
 }
```

Observable + Async Pipe + NgFor

Step-3: Now we will subscribe our Observable variable i.e. allBooks using async pipe. We need to keep in mind that HTTP hit will take place only when we subscribe our Observable variable. In our HTML template we will subscribe our Observable variable i.e. allBooks using async pipe with ngFor.

```
<ul>
  <li *ngFor="let book of allBooks | async" >
  Id: {{book.id}}, Name: {{book.name}}, Category: {{book.category}}
 </li>
</ul>
```

Angular 16 : Online Class

Observable + Async Pipe + NgIf

By: Sahosoft Solutions

Presented by : Chandan Kumar

sahosoft
Online Learning Platform

Observable + Async Pipe + NgIf

We will see how to use Observable using async pipe with ngIf. We will fetch data over HTTP and will display it. We will show loading image while data is being fetched over HTTP.

Step-1: We have created a method in service, suppose in BookService, to fetch data over HTTP for the demo.

```
getFavBookFromStore(id: number): Observable<Book> {
  return this.http.get<Book>(this.bookUrl + "/" + id);
}
```

Above method will return Observable<Book>.

Observable + Async Pipe + NgIf

Step-2: In component we have created a property of Observable type.

```
favBook: Observable<Book>
```

Assign value to **favBook** as following.

```
getFavBook() {
    this.favBook = this.bookService.getFavBookFromStore(101);
   }
```

calling the above method in ngOnInit

```
ngOnInit() {
   this.getFavBook();
}
```

Observable + Async Pipe + Nglf

Step-3: Now we will display data in HTML template. We will subscribe Observable variable favBook$ using asyncpipe.

```
<div *ngIf="favBook | async as book; else loading">
  Id: {{book.id}}, Name: {{book.name}}, Category: {{book.category}}
</div>
<ng-template #loading>
  <img src="assets/images/loading.gif">
</ng-template>
```