



Service



Agenda

- Призначення сервісів
- Observable
- RxJS
- httpClient
- json-server

Стандартні завдання сервісів

- Сервіс може сам зберігати дані в пам'яті, або для отримання даних може звертатися до будь-якого джерела даних, наприклад, до сервера.
- Сервіс може представляти канал взаємодії між окремими компонентами програми
- Сервіс може інкапсулювати бізнес-логіку, різні обчислювальні завдання, завдання логування. Тим самим код компонентів буде зосереджений безпосередньо на роботі з шаблоном. Крім того, ми також можемо вирішити проблему повторення коду, якщо нам буде потрібно виконати одну і ту ж задачу в різних компонентах і класах
- створення `service(cli) ng g s <name>`

Зберігання даних у сервісі

- дані зберігаються в сервісі
- з компонентів ми маємо доступ до методів для роботи з даними
- перед застосуванням сервісу потрібно його “заінджектувати”(підключити)

```
- providers: [ DataService ] // додати в app.module
- @Injectable({                // вказати у сервісі до якого
    модуля підключити
    providedIn: 'root',        // root - app.module
  })
```

- app.component

```
import { DataService } from './data.service'; // підключаємо service

constructor(                                     // отримуємо доступ
до об'єкта service
  private data: DataService
) {}

ngOnInit() {
  this.allProduct = this.data.getAll(); // викликаємо метод service
}
```

- data.service

```
products = [{                                     // дані
  ...
}];

getAll() {                                       // метод повертає всі продукти
  return this.products;
}
```

[Demo](#)

Observable

механізм, який використовується в Angular для програмування асинхронних потоків даних. Він дозволяє створювати об'єкт, про зміну якого ми хотіли б дізнаватися в різних частинах програми. Щоб отримати зміни потрібно на нього підписатися.

Кожен раз, коли спостережуваний об'єкт змінюється, підписники відловлюють цю подію і отримують нове значення.

Аналогічно, ми в будь-якій частині коду можемо змінювати значення об'єкта, і посилати подію про цю зміну.

Observer

onNext

onError

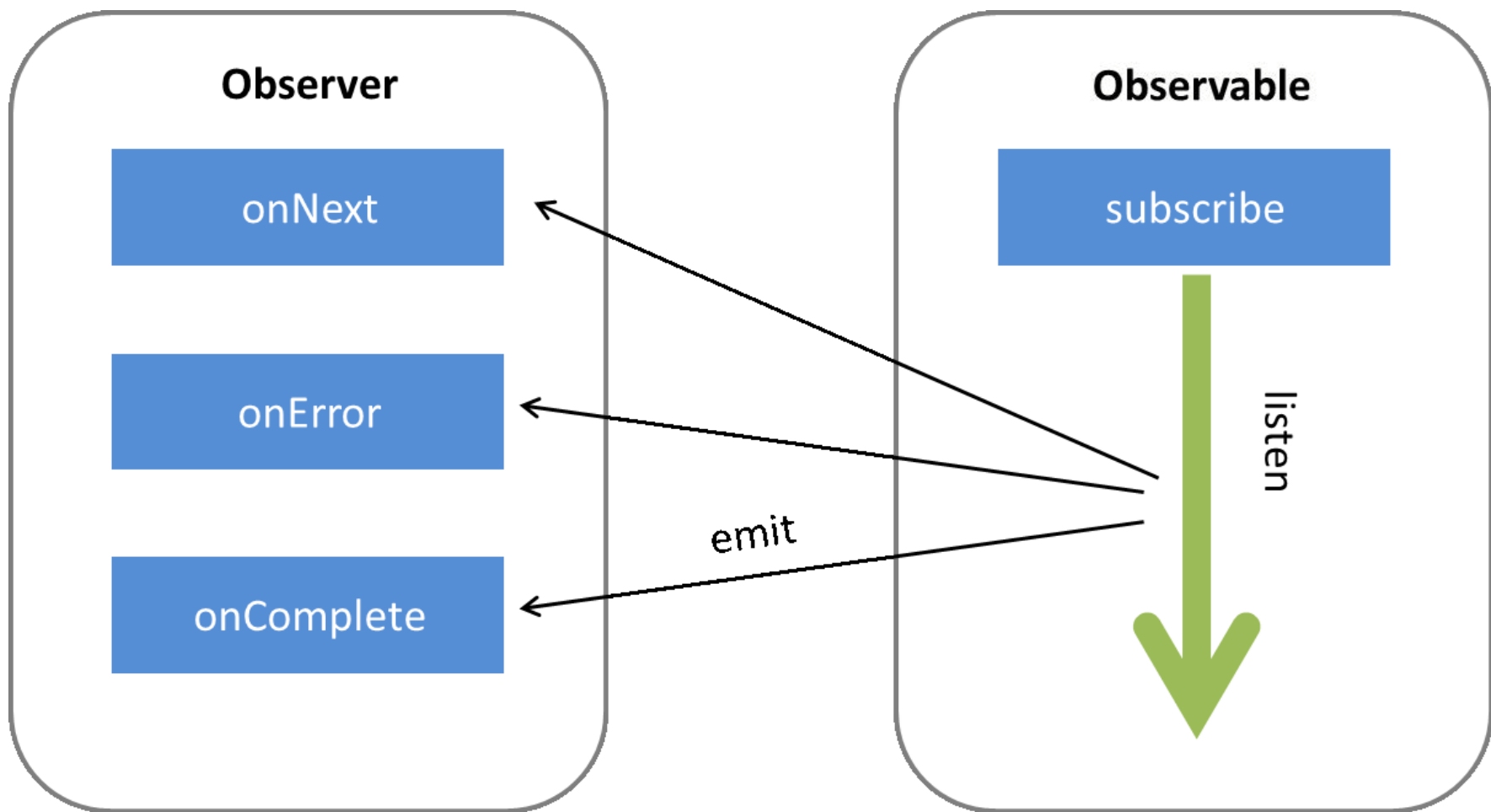
onComplete

Observable

subscribe

listen

emit



RxJS

Реактивне програмування - це парадигма асинхронного програмування, що стосується потоків даних. RxJS - це бібліотека для реактивного програмування з використанням Observables, що полегшує використання асинхронного коду. Ці функції утиліти можна використовувати для:

- Перетворення існуючого коду для операцій async
- Перебирати значення в потоці
- Перетворення значень на різні типи
- Фільтрування потоків
- Складання декількох потоків

<https://rxjs-dev.firebaseapp.com/guide/overview>

Приклад

```
import { Observable, interval } from 'rxjs'; // імпортуємо бібліотеку

getTime(): Observable<number> { // метод повертає Observable
    return interval(1000);
}

this.time.getTime().subscribe(time => { // підписуємся на змінні
    console.log(time);
})
```

[Demo](#)

Subject в RxJS

Subject (далі тут буде для кращого розуміння називатися сабж) - це одночасно глядач і видовище (subscriber and observable). Інакше кажучи, сабж має методи як видовища, так і глядача.

Ми можемо генерувати події в сабже, використовуючи `.next()`, успішно завершувати сабж за допомогою `.complete()` і завершувати сабж помилкою допомогою `.error()`

[Demo](#)

Приклад

- toggle.service

```
import { Subject } from 'rxjs';           // імпортуємо об'єкт Subject з бібліотеки RxJS
subject = new Subject();                  // створюємо новий об'єкт
```

- app.component

```
private sub: ToggleService                // підключаємо сервіс

this.sub.subject.subscribe(x => console.log(x)); // підписуємось на зміни
this.sub.subject.next(1);                  // передаємо нове
значення в сервіс
```

http

Більшість прикладних програм спілкуються з серверними службами через протокол HTTP. Сучасні браузеры підтримують два різних API для створення HTTP-запитів: інтерфейс XMLHttpRequest і API fetch ().

HttpClient в @angular/common/http пропонує спрощений клієнтський HTTP-API для Angular-додатків, що спирається на інтерфейс XMLHttpRequest, підключений браузерами. Додаткові переваги HttpClient включають функції тестування, типізовані об'єкти запитів і відповідей, перехоплення запитів і відповідей, Observable apis і спрощену обробку помилок.

- app.module

```
import { HttpClientModule } from '@angular/common/http'; // імпортуємо http модуль
imports: [
  ...
  HttpClientModule,
    // підключаємо до проекту
],
```

- post.service

```
import { HttpClient } from '@angular/common/http'; // імпортуємо і підключаєм
httpClient

constructor(private http: HttpClient) { }

getPost() {
    // запит до API
    return this.http.get('https://jsonplaceholder.typicode.com/posts');
}
```

- app.component

```
this.data.getPost().subscribe( (posts: IPost[]) => { // підписуємось на метод
    this.allPost = posts;
})
```

json-server

- fake REST API
- можна розгорнути API з даними які зберігаються на локалі

```
npm install -g json-server    // інсталуємо
```

```
db.json                      // створюємо файл з даними
```

```
json-server --watch db.json  // запускаємо сервер
```

<https://github.com/typicode/json-server>

Приклад

- реалізація CRUD(create read update delete -базові функції управління даними «створення, зчитування, зміна і видалення».)
- <https://github.com/shev4uk/crud-post>

Firestore

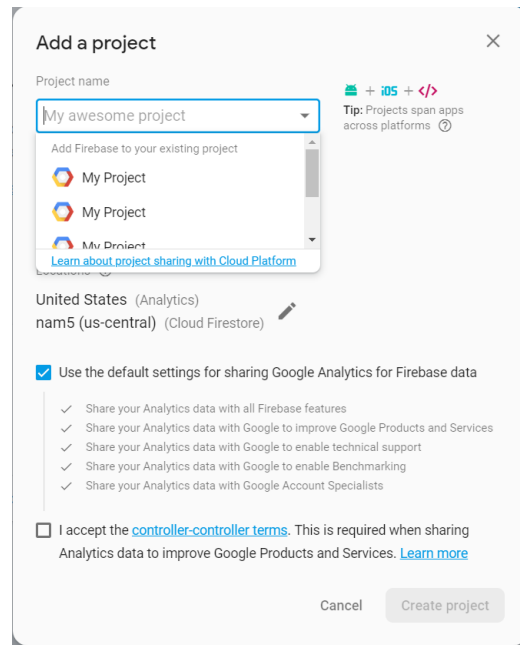
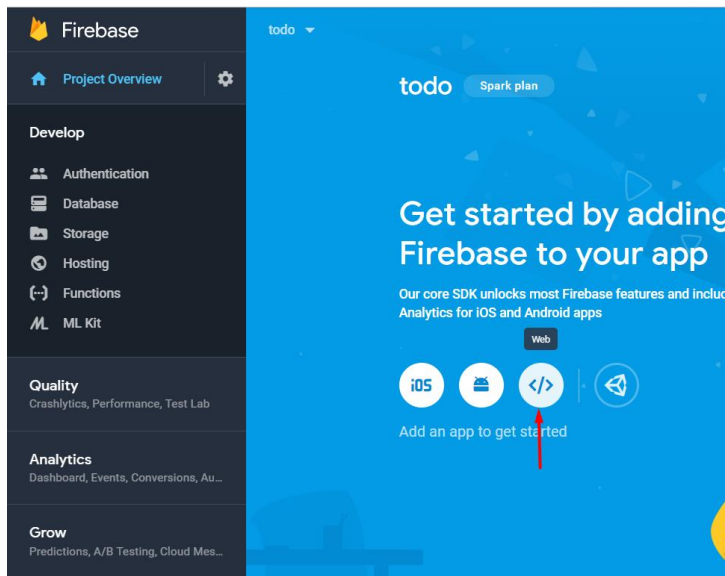
це хороший інструмент, який допомагає швидко створювати програми, не керуючи інфраструктурою. Це потужне рішення DBaaS, яке надає масштабовану хмарна базу даних NoSQL для зберігання та синхронізації інформації для розробки клієнтської та серверної сторін.

Майже кожне додаток потребує зберігання даних, оскільки зміст є сутністю комунікації та взаємодії з користувачами. Зокрема, вам потрібно буде зберігати інформацію для підтримки вашої бізнес-логіки, а також вам знадобиться якийсь сервер для обробки автентифікації користувача.

<https://firebase.google.com/>

Get started

- Додаємо новий проект
- Заповнюємо форму
- Додаємо в нашу аплікацію для web



Скопіювати налаштування

Добавление Firebase в веб-приложение

✓ Зарегистрируйте приложение

2 Добавление Firebase SDK

Скопируйте и вставьте эти скрипты в конец тега <body> перед сервисами Firebase:

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/7.12.0/firebase-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
https://firebase.google.com/docs/web/setup#available-libraries -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: [REDACTED],
    authDomain: [REDACTED],
    databaseURL: [REDACTED],
    projectId: [REDACTED],
    storageBucket: [REDACTED],
    messagingSenderId: [REDACTED],
    appId: [REDACTED]
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

Чтобы получить более подробную информацию о Firebase для веб-приложений, перейдите по следующим ссылкам: [Начало работы](#), [Справочник по Web SDK API](#), [Примеры](#)

Налаштування ангуляр

Відкриваємо файл `src/environments/environment.ts`

```
export const environment = {  
  production: false,  
  firebase: {
```

firebase

```
    apiKey: YOUR_API_KEY,  
    authDomain: YOUR_AUTH_DOMAIN,  
    databaseURL: YOUR_DATABASE_URL,  
    projectId: YOUR_PROJECT_ID,  
    storageBucket: "",  
    messagingSenderId: YOUR_MESSAGING_SENDER_ID  
  }  
};
```

// копіюємо параметри з

Додаємо модулі для роботи з firebase

```
ng add @angular/fire@next --save          // встановлюємо firebase
```

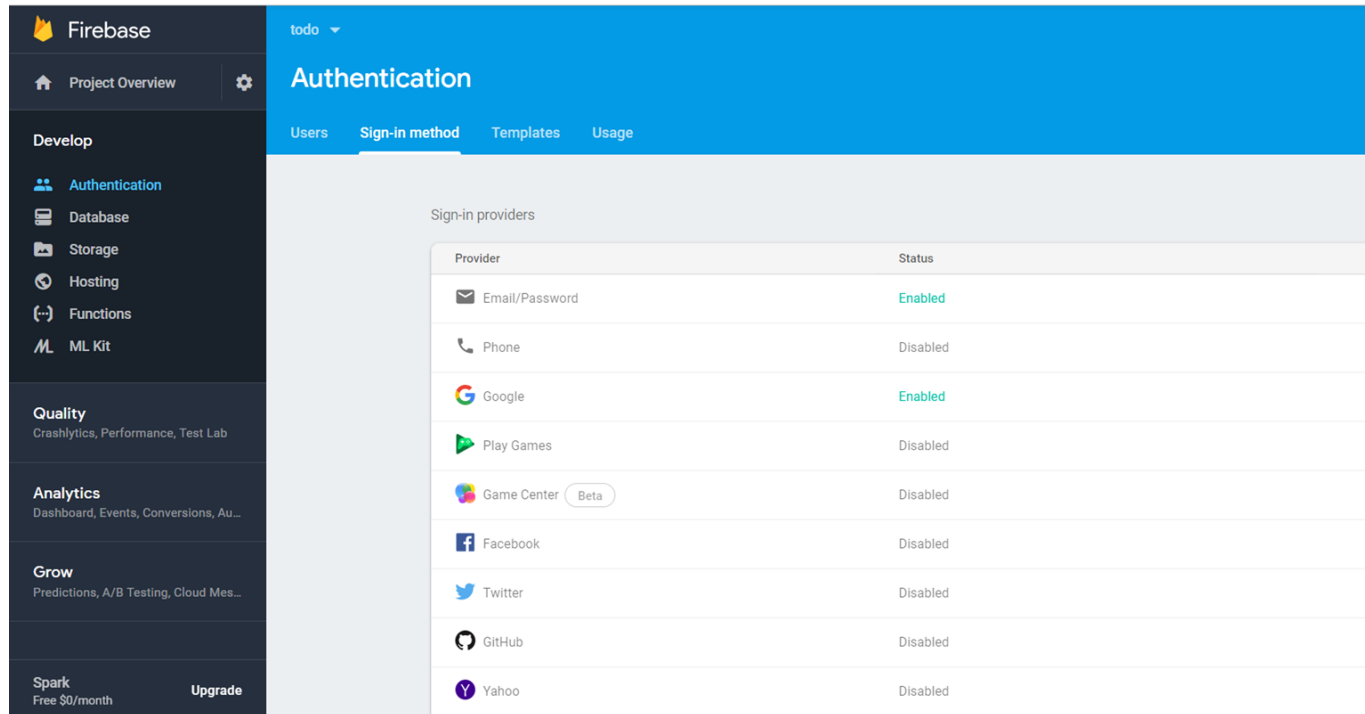
- app.module

```
import { AngularFireModule } from '@angular/fire';          // імпортуємо  
import { AngularFirestoreModule } from '@angular/fire/firestore';  
import { environment } from '../environments/environment';  
  
imports: [  
  ...  
  AngularFireModule.initializeApp(environment.firebase),  // ініціалізуємо проект  
  AngularFirestoreModule,  
],
```

<https://github.com/angular/angularfire/blob/master/docs/install-and-setup.md>

Authentication

Включаємо системи для логінування



The screenshot displays the Firebase Authentication console. On the left is a dark sidebar with navigation links for Firebase, Project Overview, and various development tools like Authentication, Database, Storage, Hosting, Functions, and ML Kit. Below these are sections for Quality, Analytics, and Grow. The main content area has a blue header with 'Authentication' and tabs for Users, Sign-in method (selected), Templates, and Usage. Under the 'Sign-in method' tab, there's a 'Sign-in providers' section with a table listing various providers and their status.

Provider	Status
Email/Password	Enabled
Phone	Disabled
Google	Enabled
Play Games	Disabled
Game Center (Beta)	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Yahoo	Disabled

Підключення

- app.module

```
import { AngularFireAuthModule } from '@angular/fire/auth'; // імпортуємо і підключаємо  
модуль для authentication
```

Посилання

<https://habr.com/ru/post/429342/>

<https://metanit.com/web/angular2/4.1.php>

<https://www.learnrxjs.io/>

<https://angular.io/guide/rx-library>

<https://medium.com/@kosmogradsky/subject-%D0%B2-rxjs-%D0%BA%D1%80%D0%B0%D1%82%D0%BA%D0%BE%D0%B5-%D0%B2%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5-c9099231be6d>

<http://www.front-nika.ru/ru/angular-2-servis-s-observable/>

<http://reactivex.io/rxjs/class/es6/Observable.js~Observable.html>

<https://www.sitepoint.com/angular-rxjs-create-api-service-rest-backend/>

<https://angular.io/guide/http>

<https://webdraftt.com/tutorial/rxjs/about>

Example

<https://stackblitz.com/edit/angular-rxjs-subject-and-behaviorsubject-g6qhwt>

Відео

https://www.youtube.com/watch?v=BBG9-BV-udo&list=PLVfMKQXDAhGW12JY3SfeDnEx7S5_tn11j&index=10

Посилання(firebase)

<https://angular-templates.io/tutorials/about/angular-crud-with-firebase>

<https://fireship.io/lessons/angularfire-google-oauth/>

<https://angular-templates.io/tutorials/about/firebase-authentication-with-angular>

<https://www.techiediaries.com/angular-firebase-authentication-email-google/>

<https://fireship.io/lessons/angularfire-google-oauth/>

<https://angularfirebase.com/lessons/multi-step-signup-firebase-email-password-auth-angular-reactive-forms/>

<https://codelabs.developers.google.com/codelabs/firestore-web/#11>

<https://angularfirebase.com/lessons/managing-firebase-user-relationships-to-database-records/>

<https://github.com/angular/angularfire2>

<https://stackoverflow.com/questions/56417164/angular-compilation-warnings-with-angular-material-declarations>

structure data

<https://www.airpair.com/firebase/posts/structuring-your-firebase-data>

<https://howtofirebase.com/firebase-data-modeling-939585ade7f4>

<https://firebase.google.com/docs/database/web/structure-data?hl=ru>

<https://firebase.google.com/docs/firestore/manage-data/structure-data> !!!

Відео(firebase)

https://www.youtube.com/watch?v=RxLI9_ub6PM&list=PL0vfts4VzfNg7nTsEkiWCCNB8BvoZcPz4&index=6

structure data

<https://www.youtube.com/watch?v=haMOUb3KVSo>