GANATAN

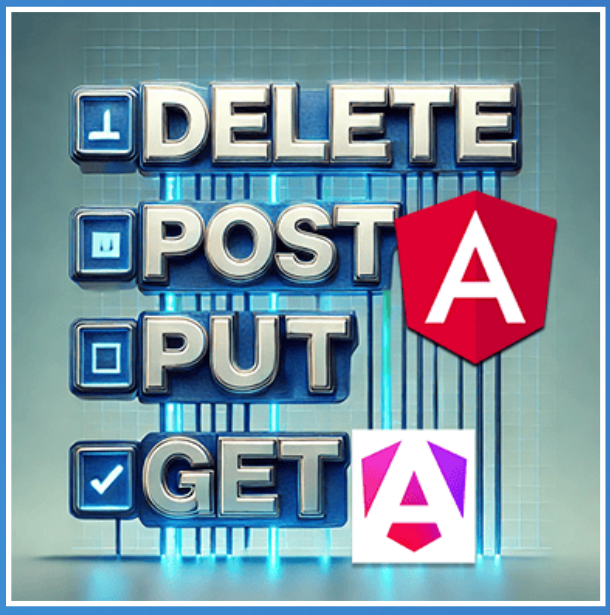
HttpClient avec angular

Guide Complet

Mise à jour du 2 février 2025

Nous allons rajouter dans notre projet le mécanisme **HttpClient** pour communiquer avec un serveur HTTP.

Nous utiliserons le framework javascript **Angular** version **19.2.0**



www.ganatan.com

Qu’allons nous faire ?

Il s'agit de **l'étape 9** de notre guide Angular
qui nous permettra d'obtenir une **Application Web** de type **PWA**.
Le projet Angular de base que nous utiliserons dispose déjà des caractéristiques suivantes

- Généré avec **Angular CLI**
- Le **Routing**
- Le **Lazy Loading**
- Le framework CSS **Bootstrap**
- **Server Side Rendering**
- **Progressive Web App**
- **Search Engime optimization**

Tous les sources créés sont indiqués en fin de tutoriel.
L' application est à l'adresse suivante

- https://angular.ganatan.com

Installation

Notre application Angular représente notre partie **Frontend**.

Nous allons utiliser le module Httpclient d’angular pour accèder à une **API REST** qui représentera notre partie **Backend.**

L’objet de ce tutoriel n’étant pas de développer un backend, nous allons donc choisir arbitrairement une API disponible sur le web.

JSONPlaceholder est une API REST gratuite idéale pour les tutoriels.

Elle est accessible avec le lien suivant https://jsonplaceholder.typicode.com/

Les étapes seront les suivantes

- Créer un module items
- Modifier l'interface visuelle du frontend
- Effectuer le routage

- Adapter le module pour faire appel à l'API REST

Commençons par créer notre module Items, celui-ci étant spécifique à notre application sera créé dans le répertoire **modules/application**.

```
# Création du module items
ng generate component pages/application/example-items/items --flat --module=app
ng generate module pages/application/example-items/items --flat --routing

# Création du module items (Méthode 2)
ng g c pages/application/example-items/items --flat --module=app
ng g m pages/application/example-items/items --flat --routing
```

Le répertoire pages/**application/example-items** est automatiquement créé par Angular CLI

Modifions les fichiers suivants

- **app.routes.ts**
- **items.components.ts**
- **items-routing.module.ts**
- **items.module.ts**

src/app/app.routes.ts

```
 {
   path: 'httpclient',
   loadChildren: () => import('./pages/application/example-items/items.module')
     .then(mod => mod.ItemsModule)
 },
```

src/app/pages/application/example-items/items.components.ts

```
import  { Component } from '@angular/core';

@Component({
  selector: 'app-items',
  templateUrl: './items.component.html',
  styleUrl: './items.component.css'})
export class ItemsComponent {

}
```

src/app/pages/application/example-items/items-routing.module.ts

```
import  { NgModule } from '@angular/core';
import  { RouterModule, Routes } from '@angular/router';

import  { ItemsComponent } from './items.component';

const  routes: Routes = [{ path: '', component: ItemsComponent }];

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

src/app/pages/application/example-items/items.module.ts

```
import  { NgModule } from '@angular/core';
import  { CommonModule } from '@angular/common';

import  { ItemsComponent } from './items.component';
import  { ItemsRoutingModule } from './items-routing.module';

@NgModule({
  imports: [CommonModule, ItemsRoutingModule],
  declarations: [ItemsComponent],
  exports: [ItemsComponent]
})
export  class ItemsModule {}
```

A ce stade nous pouvons tester le script de debugage pour contrôler l'architecture de notre projet.

- npm run start

Puis taper l'url

- http://localhost:4200/httpclient

Integration

Il ne reste plus qu'à intégrer la gestion **httpclient** sur notre partie Items.

Pour respecter les best practices angular toutes les fonctionnalités seront ajoutées dans le répertoire **app/modules/items.**

Nous créerons **un service** pour gérer l'accès à l'api.

Les étapes sont les suivantes.

- Créer un service via un fichier **items.service.ts**
- Modifier le fichier **items.service.ts** pour accèder à l'api
- Modifier le fichier **items.component.ts** pour appeler le service
- Modifier le fichier **items.module.ts** pour appeler le module angular nécessaire
- Modifier les fichiers de test **items.component.spec.ts** et **items.service.spec.ts**
- Modifier le fichier **items.component.html** pour personaliser l'affichage du résultat
- Modifier le fichier **home.component.ts** pour permettre le lien avec httpclient

Commencons par créer le service avec la commande angular-cli correspondante puis modifions les fichiers cités.

```
# Création du service items
ng generate service pages/application/example-items/items --flat

# Création du service items (Méthode 2)
ng g s pages/application/example-items/items --flat
```

src/app/modules/application/example-items/items.service.ts

```
import  { Injectable } from '@angular/core';
import  { HttpClient, HttpHeaders } from '@angular/common/http';
import  { Observable } from 'rxjs';

const httpOptions = {
  headers: new HttpHeaders(
    {
      'Content-Type': 'application/json',
    }
  )
};

 @Injectable({
   providedIn: 'root'})
 export class ItemsService {

   constructor(private http: HttpClient) { }
                                                 getItems(url: string): Observable<object> {
     return this.http.get(url, httpOptions);
   }

 }
```

src/app/modules/application/example-items/items.component.ts

```
import  { Component, OnInit } from '@angular/core';
import  { ItemsService } from './items.service';

import  { isPlatformBrowser } from '@angular/common';
import  { PLATFORM_ID, APP_ID, Inject } from '@angular/core';

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


  // eslint-disable-next-line  @typescript-eslint/no-explicit-any
  items: any;
  loaded: boolean;
  constructor(
    @Inject(PLATFORM_ID) private platformId: object,
    @Inject(APP_ID) private appId: string,
    private ItemsService: ItemsService
  ) {
    this.loaded = false;
  }

  ngOnInit(): void {
    this.getUsers();
  }

  getUsers(): void {
    this.loaded = false;
    this.ItemsService.getItems('https://jsonplaceholder.typicode.com/users')
      .subscribe(
        items => {
          this.items = items;
          this.loaded = true;
          const platform = isPlatformBrowser(this.platformId) ?
            'in the browser' : 'on the server';
          console.log(`getUsers : Running ${platform} with appId=${this.appId}`);
        });
  }

  resetUsers(): void {
    this.items = null;
    this.loaded = true;
  }

}
```

src/app/pages/application/example-items/items.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';

import { ItemsComponent } from './items.component';
import { ItemsRoutingModule } from './items-routing.module';
import { ItemsService } from './items.service';

@NgModule({
  imports: [CommonModule, ItemsRoutingModule],
  declarations: [ItemsComponent],
  exports: [ItemsComponent],
  providers: [ItemsService]
})
export class ItemsModule {}
```

src/app/pages/application/example-items/items.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { HttpClientModule } from '@angular/common/http';
import { ItemsComponent } from './items.component';

describe ('ItemsComponent', () => {
  let component: ItemsComponent;
  let fixture: ComponentFixture<ItemsComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [
        HttpClientModule
      ],
      declarations: [ItemsComponent]
    }).compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(ItemsComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

src/app/modules/example-items/items.service.spec.ts

```
import  { TestBed } from '@angular/core/testing';
import  { HttpClientModule } from '@angular/common/http';
import  { ItemsService } from './items.service';

describe ('ItemsService', () => {
  let service: ItemsService;

   beforeEach(() => {
     TestBed.configureTestingModule({
       imports: [
         HttpClientModule
       ]
     });
     service = TestBed.inject(ItemsService);
   });

   it('should be created', () => {
     expect(service).toBeTruthy();
   });
 });
```

src/app/modules/example-items/items.component.html

```
<div class="row pb-4">

  <div class="col-12 col-sm-12 col-md-3 col-lg-3 col-xl-3">
    <div class="row pb-4">
      <div class="card" style="width: 18rem;">
        <div class="card-body">
          <h5 class="card-title">Feature : HttpClient</h5>
          <hr>
          <p class="card-text">Use HtppClient</p>
          <button type="button" class="btn btn-primary mr-4" (click)="getUsers()">Get</button>
          <button type="button" class="btn btn-primary" (click)="resetUsers()">Reset</button>
        </div>
      </div>
      <div *ngIf="!loaded">
        <div class="spinner-grow text-primary" role="status">
          <span class="sr-only">Loading...</span>
        </div>
        <div class="spinner-grow text-secondary" role="status">
          <span class="sr-only">Loading...</span>
        </div>
        <div class="spinner-grow text-success" role="status">
          <span class="sr-only">Loading...</span>
        </div>
        <div class="spinner-grow text-danger" role="status">
          <span class="sr-only">Loading...</span>
        </div>
        <div class="spinner-grow text-warning" role="status">
          <span class="sr-only">Loading...</span>
        </div>
        <div class="spinner-grow text-info" role="status">
          <span class="sr-only">Loading...</span>
        </div>
        <div class="spinner-grow text-dark" role="status">
          <span class="sr-only">Loading...</span>
        </div>
      </div>
    </div>
  </div>

  <div class="col-12 col-sm-12 col-md-8 col-lg-8 col-xl-8">
    <div class="row">
      <div class="table-responsive httpclient-table blue-gradient" *ngIf="loaded">
        <table class="table table-hover table-striped table-responsive-md">
          <thead>
            <tr>
              <th scope="col">name</th>
              <th scope="col">username</th>
              <th scope="col">email</th>
            </tr>
          </thead>
          <tbody>
            <tr *ngFor="let item of items">
              <td>{{item.name}}</td>
              <td>{{item.username}}</td>
              <td>{{item.email}}</td>
            </tr>
          </tbody>
        </table>
      </div>
    </div>
  </div>

 </div>
```

items.component.css

```
.httpclient- table {
  border -radius: 10px;
}

.httpclient- table table {
  color : #FFFFFF;
}

.httpclient- table .blue-gradient {
  background : linear-gradient(40deg, #45cafc, #5664bd) !important
}
```

src/app/modules/general/home/home.component.ts

```
{
  name: 'Items',
  description: 'Items',
  icon: 'fab fa-bootstrap',
  link: 'httpclient'       },
```

Erreurs

Quelques modifications finales

src/app/modules/application/example-items/items.module.ts

```
import  { NgModule } from '@angular/core';
import  { CommonModule } from '@angular/common';

import  { ItemsComponent } from './items.component';
import  { ItemsRoutingModule } from './items-routing.module';
import  { ItemsService } from './items.service';

@NgModule({
  imports: [CommonModule, ItemsRoutingModule],
  declarations: [ItemsComponent],
  exports: [ItemsComponent],
  providers: [ItemsService]
})
export  class ItemsModule {}
```

src/app/app.config.ts

```
import { ApplicationConfig, isDevMode } from '@angular/core';
import { provideRouter } from '@angular/router';

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

import { routes } from './app.routes';
import { provideClientHydration } from '@angular/platform-browser';
import { provideServiceWorker } from '@angular/service-worker';

export const appConfig: ApplicationConfig = {
  providers: [
    importProvidersFrom(HttpClientModule),
    provideRouter(routes),
    provideClientHydration(),
     provideServiceWorker('ngsw-worker.js', {
       enabled: !isDevMode(),
       registrationStrategy: 'registerWhenStable:30000'    })]
};
```

Conclusion

Il ne reste plus qu'à tester les différents scripts Angular.

```
# Développement
npm run start
http://localhost:4200/

# Tests
npm run lint
npm run test

# AOT compilation
npm run build

# SSR compilation
npm run build:ssr
npm run serve:ssr
http://localhost:4000/
```