



R a i s i n g t h e b a r

Angular HttpClient

Mục tiêu

- Tạo được service sử dụng HttpClient.
- Lấy được dữ liệu từ back-end API.
- Đưa được dữ liệu lên back-end.
- Sử dụng được các HTTP Method khác nhau.

Project mẫu

- Project mẫu tại folder: angular-http
- Hoặc bạn có thể tạo mới project với câu lệnh:

```
ng new angular-http --style scss
```

Project mẫu

- Nếu bạn tạo mới project, bạn sẽ cần cài đặt thêm thư viện sau.
 - Các bạn chạy lệnh **npm i typescript@2.9.2 -D** để cài đặt thư viện typescript **nếu phiên bản hiện tại < 2.9.2**
 - Update file styles.scss

Project mẫu

```
* , *::before, *::after {  
    box-sizing: border-box;  
}  
  
body {  
    font-family: -apple-system, BlinkMacSystemFont,  
    'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open  
    Sans', 'Helvetica Neue', Arial, sans-serif;  
}
```

Angular HttpClient

Angular HttpClient là gì?

- Angular cung cấp sẵn một thư viện để bạn có thể communicate với Backend API.
- Công việc cần thiết mà bạn cần làm để có thể sử dụng HttpClient là import HttpClientModule vào root NgModule.

Import HttpClientModule

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

@NgModule({
  imports: [
    BrowserModule,
    ReactiveFormsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Import các NgModule khác

Chuẩn bị

- Xây dựng ứng dụng todo, có kết nối với Rest API tại địa chỉ:
<http://jsonplaceholder.typicode.com/todos>
- Khởi tạo todo component, todo service, todo interface.

Todo Interface

```
export interface ITodo {  
    id: number;  
    title: string;  
    completed: boolean;  
}
```

Todo Service

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class TodoService {
  private readonly API_URL = 'http://jsonplaceholder.typicode.com/todos';
  constructor(private http: HttpClient) { }
}
```

Inject HttpClient service để call API

Todo Component

```
import { TodoService } from './todo.service';
import { FormControl } from '@angular/forms';

export class TodoComponent implements OnInit {
  todoList: ITodo[] = [];
  inputControl = new FormControl();
  constructor(private todoService: TodoService) { }
  ngOnInit() {}
  toggleTodo(i) {}
  addTodo() {}
  deleteTodo(i) {}
}
```

Todo Component

```
<h2>Todo App with HttpClient</h2>
```

```
<div>
  <input type="text"
    [formControl]="inputControl" (keydown.enter)="addTodo()">
</div>
```

Todo Component

```
<ul>
  <li *ngFor="let todo of todoList; index as i"
      [class.completed]="todo.completed">
    <div (click)="toggleTodo(i)">
      Id: {{todo.id}}
      <br>
      {{todo.title}}
    </div>
    <button (click)="deleteTodo(i)">delete</button>
  </li>
</ul>
```

Tạo get request để lấy dữ liệu

```
export class TodoService {  
    getTodos(count = 10): Observable<ITodo[]> {  
        return this.http.get<ITodo[]>(this.API_URL).pipe(  
            map(data => data.filter((todo, i) => i < count))  
        );  
    }  
}
```

Kiểu dữ liệu trả về của hàm này
là một Observable

Sử dụng method get để call HTTP GET
request

Chuyển đổi data từ dạng này sang dạng khác,
sử dụng RxJs operator

Tạo get request để lấy dữ liệu

```
export class TodoComponent implements OnInit {  
  ngOnInit() {  
    this.todoService.getTodos().subscribe(next => {  
      this.todoList = next;  
    }, error => {  
      console.log(error);  
    }, () => {  
      console.log('complete');  
    });  
  }  
}
```

Tạo get request để lấy dữ liệu

- Method `this.todoService.getTodos()` khi gọi sẽ trả về một Observable, nên chúng ta sẽ trigger Observable này bằng cách call method subscribe của nó.
- Method subscribe nhận đầu vào là 3 callback lần lượt để handle khi có dữ liệu đến, khi bị phát sinh lỗi, và khi complete
- Ngoài ra method subscribe cũng nhận đầu vào là một object có chứa 3 callback được gọi là observer.
- Không nhất thiết phải có đủ cả 3 callback, nhưng riêng với dạng nhận vào function thì thứ tự function phải đồng nhất (`nextFn`, `errorFn`, `completeFn`), nếu bạn không muốn truyền vào thì truyền `null/undefined` để tránh bị sai vị trí.

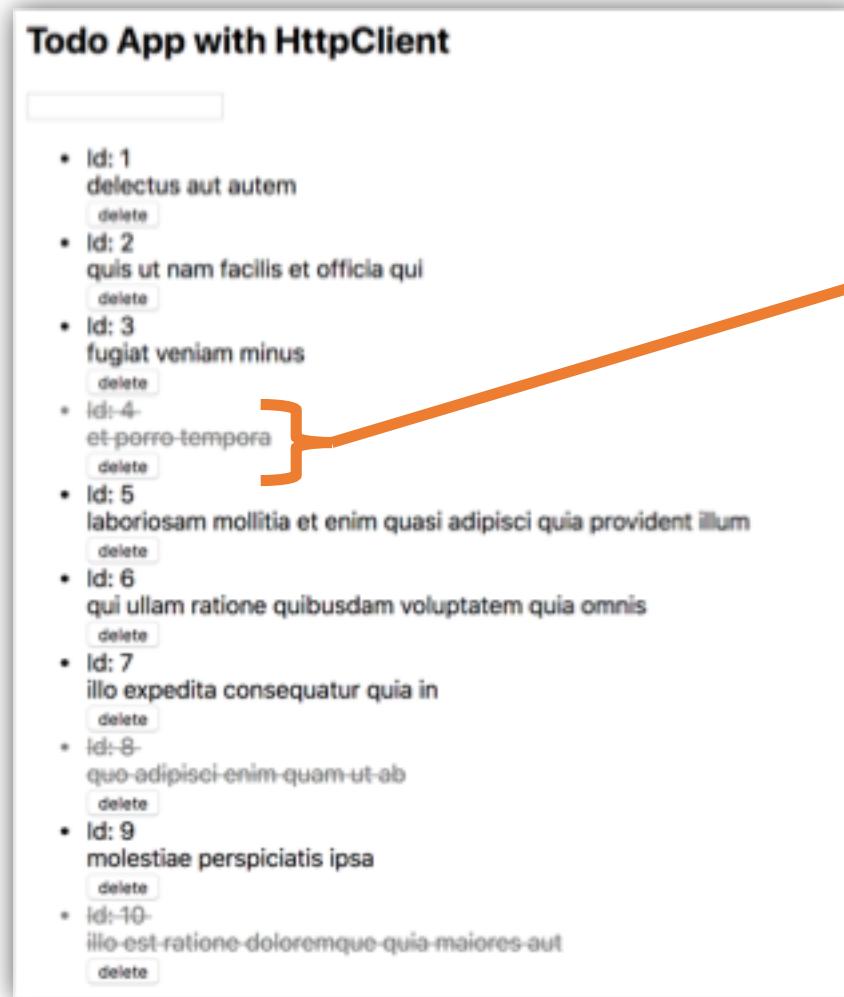
Tạo get request để lấy dữ liệu

```
const observer = {  
    next: (data) => {},  
    error: (error) => {},  
    complete: () => {}  
}
```

Tạo get request để lấy dữ liệu

Trong hàm ngOnInit chúng ta đã xử lý việc call API, khi có dữ liệu trả về, thuộc tính trong class TodoComponent sẽ được gán và Angular sẽ hiển thị kết quả cho chúng ta biết.

Tạo get request để lấy dữ liệu



Phần tử này đã completed, nên nó được áp dụng css cho gạch ngang text

Update một dữ liệu

- Thông thường, phía Server sẽ quy định các dữ liệu cần thiết để update, các HTTP method tương ứng.
- Ví dụ với Restful API, method PUT thường được dùng để “replace”, và method PATCH thường được dùng để update một phần.
- Giả sử chúng ta cần toggle trạng thái completed của một todo, lúc này những phần dữ liệu khác dữ nguyên thì chúng ta có thể dùng PATCH để update một phần.
- Lưu ý rằng có những hệ thống không rõ ràng việc chia method tương ứng với resource, nên có thể có hệ thống sẽ dùng POST cho cả add, update.

Update một dữ liệu

```
export class TodoService {  
    updateTodo(todo: ITodo): Observable<ITodo> {  
        return this.http.patch<ITodo>(  
            `${this.API_URL}/${todo.id}`, todo);  
    }  
}
```

Update một dữ liệu

```
export class TodoComponent {  
  toggleTodo(i) {  
    const todo = this.todoList[i];  
    const todoData = {  
      ...todo,  
      completed: !todo.completed  
    };  
    this.todoService.updateTodo(todoData).subscribe(next => {  
      this.todoList[i].completed = next.completed;  
    });  
  }  
}
```

Clone sang một object khác, khi nào update thành công thì update lại phần tử tương ứng trong list

Update một dữ liệu

- Id: 1
delectus aut autem
[delete](#)
- Id: 2
quis ut nam facilis et officia qui
[delete](#)
- Id: 3
fugiat veniam minus
[delete](#)

Khi click vào phần text này, ứng dụng sẽ gửi một request để thay đổi trạng thái của todo

Todo App with HttpClient

- Id: 1
delectus aut autem
[delete](#)
- Id: 2
quis ut nam facilis et officia qui
[delete](#)
- Id: 3
fugiat veniam minus
[delete](#)

jsonplaceholder...
1
jsonplaceholder...

General

Request URL: http://jsonplaceholder.typicode.com/todos/1
Request Method: PATCH
Status Code: 200 OK
Remote Address: 104.27.181.202:80
Referrer Policy: no-referrer-when-downgrade

Response Headers

Delete một dữ liệu

```
export class TodoService {  
    deleteTodo(id: number): Observable<any> {  
        return this.http.delete(  
            `${this.API_URL}/${id}`  
        );  
    }  
}
```

Delete một dữ liệu

```
export class TodoComponent {  
  deleteTodo(i) {  
    const todo = this.todoList[i];  
    this.todoService.deleteTodo(todo.id).subscribe(  
      () => {  
        this.todoList = this.todoList.filter(  
          t => t.id !== todo.id  
        );  
      });  
  }  
}
```

Sau khi delete thành công
thì xóa phần tử khỏi list

Delete một dữ liệu

- Id: 1
delectus aut autem
delete
- Id: 2
quis ut nam facilis et officia qui
delete
- Id: 3
fugiat veniam minus
delete
- Id: 4
et porro tempora
delete

Todo App with HttpClient

- Id: 1
delectus aut autem
delete
- Id: 3
fugiat veniam minus
delete
- Id: 4
et porro tempora
delete

Name	Headers	Preview	Response	Timing
 2 jsonplaceholder...			<p>▼ General</p> <p>Request URL: http://jsonplaceholder.typicode.com/todos/2</p> <p>Request Method: DELETE</p> <p>Status Code: 200 OK</p> <p>Remote Address: 104.27.180.202:80</p> <p>Referrer Policy: no-referrer-when-downgrade</p>	
 2 jsonplaceholder...				

Thêm mới dữ liệu

```
export class TodoService {  
    createTodo(todo: Partial<ITodo>): Observable<ITodo> {  
        return this.http.post<ITodo>(this.API_URL, todo);  
    }  
}
```

Thêm mới dữ liệu

```
export class TodoComponent {  
    addTodo() {  
        const todo: Partial<ITodo> = {  
            title: this.inputControl.value,  
            completed: false  
        };  
        this.todoService.createTodo(todo).subscribe(next => {  
            this.todoList.unshift(next);  
            this.inputControl.setValue('');  
        });  
    }  
}
```

Partial là built-in class của TypeScript, cho phép convert tất cả các property của class/interface về optional

Sau khi add thành công thì thêm phần tử vào đầu list và reset form

Thêm mới dữ liệu

Todo App with HttpClient

- Id: 1
delectus aut autem
delete
- Id: 2
quis ut nam facilis et officia qui
delete
- Id: 3
fugiat veniam minus
delete
- Id: 4
~~et perro tempora~~
delete

Todo App with HttpClient

- Id: 201
Learning Angular
delete
- Id: 1
delectus aut autem
delete
- Id: 2
quis ut nam facilis et officia qui
delete
- Id: 3
fugiat veniam minus
delete

Thêm mới dữ liệu

Name	Headers	Preview	Response	Timing
<input type="checkbox"/> todos jsonplaceholder...	General Request URL: http://jsonplaceholder.typicode.com/todos Request Method: POST Status Code: 201 Created Remote Address: 104.27.180.202:80 Referrer Policy: no-referrer-when-downgrade			
<input type="checkbox"/> todos jsonplaceholder...				

Name	Headers	Preview	Response	Timing
<input type="checkbox"/> todos jsonplaceholder...	Headers Server: cloudflare Vary: Origin, X-HTTP-Method-Override, Accept-Encoding Via: 1.1 vegur X-Content-Type-Options: nosniff X-Powered-By: Express			
<input type="checkbox"/> todos jsonplaceholder...	Request Headers ⚠ Provisional headers are shown Accept: application/json, text/plain, */* Content-Type: application/json DNT: 1 Origin: http://localhost:4200 Referer: http://localhost:4200/ User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X L, like Gecko) Chrome/68.0.3440.106 Safari/537.36			
	Request Payload view source → {title: "Learning Angular", completed: false} completed: false title: "Learning Angular"			
	2 requests 1.4 KB t			

Intercepting requests and responses

- Angular HttpClient cho phép bạn intercept request và response, giúp dễ dàng sửa đổi request, handle error.
- Ví dụ: mỗi lần call API bạn phải gửi kèm token vào header để xác thực người dùng.

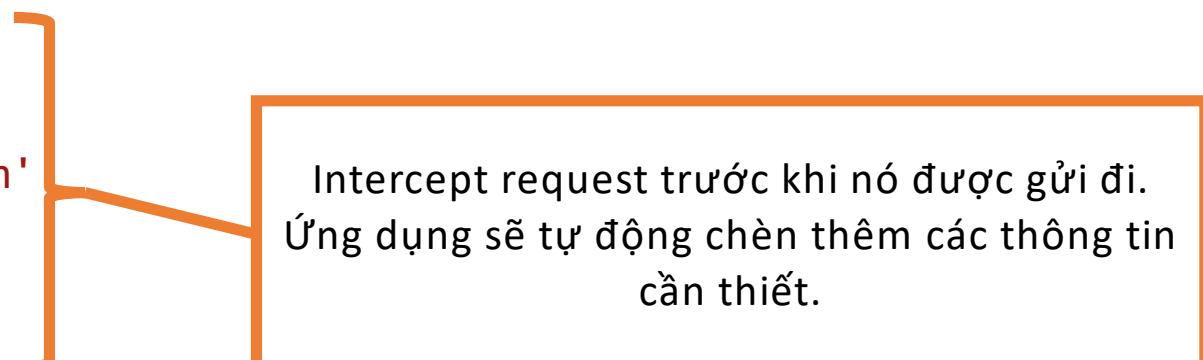
Intercepting requests and responses

- Tạo mới file src/app/token.interceptor.ts
- Thêm các đoạn code cài đặt sau đây

Intercepting requests and responses

```
import { HttpInterceptor, HttpHandler, HttpRequest, HttpEvent } from '@angular/common/http';
import { Observable } from 'rxjs';

export class TokenInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    req = req.clone({
      setHeaders: {
        Authorization: 'Bearer token'
      }
    });
    return next.handle(req);
  }
}
```



Intercept request trước khi nó được gửi đi.
Ứng dụng sẽ tự động chèn thêm các thông tin cần thiết.

Intercepting requests and responses

```
@NgModule({  
  providers: [  
    {  
      provide: HTTP_INTERCEPTORS,  
      useClass: TokenInterceptor,  
      multi: true  
    },  
  ],  
})  
export class AppModule { }
```



Đăng ký DI token với class chúng ta
vừa khởi tạo

Intercepting requests and responses

Name	Headers	Preview	Response	Timing
1 jsonplaceholder.com	X-Content-Type-Options: nosniff X-Powered-By: Express			
1 jsonplaceholder.com	Request Headers ⚠ Provisional headers are shown Accept: application/json, text/plain, */* Authorization: Bearer token } Content-Type: application/json DNT: 1 Origin: http://localhost:4200 Referer: http://localhost:4200/			

Interceptor sẽ tự động thêm header này chặng hạn

Link tham khảo

- <https://angular.io/guide/http>
- https://medium.com/@ryanchenkie_40935/angular-authentication-using-the-http-client-and-http-interceptors-2f9d1540eb8
- <https://blog.angularindepth.com/insiders-guide-into-interceptors-and-httpclient-mechanics-in-angular-103fbdb397bf>
- <https://www.tiepphan.com/rxjs-reactive-programming/>

CODEGYM

CODEGYM

R a i s i n g t h e b a r