



Raising the bar

Angular HttpClient



Mục tiêu

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



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,
                                     Import các NgModule khác
    ReactiveFormsModule,
    HttpClientModule
  providers: [],
  bootstrap: [AppComponent]
export class AppModule { }
```



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

Inject HttpClient service để call API



Todo Component

```
import { TodoService } from '../todo.service';
import { FormControl } from '@angular/forms';
export class TodoComponent implements OnInit {
  todoList: |Todo[] = [];
  inputControl = new FormControl();
  constructor(private todoService: TodoService) { }
  ngOnInit() {}
  toggleTodo(i) {}
  addTodo() {}
  deleteTodo(i) {}
```



Todo Component



Todo Component

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



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



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



- 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.

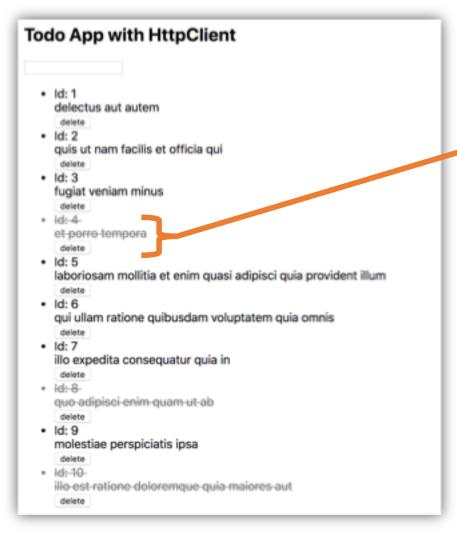


```
const observer = {
  next: (data) => {},
  error: (error) => {},
  complete: () => {}
}
```



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.





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



- 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.





```
export class TodoComponent {
  toggleTodo(i) {
                                                        Clone sang một object khác, khi nào
    const todo = this.todoList[i];
                                                      update thành công thì update lại phần tử
    const todoData = {
                                                             tương ứng trong list
      ...todo,
      completed: !todo.completed
    this.todoService.updateTodo(todoData).subscribe(next => {
      this.todoList[i].completed = next.completed;
    } );
```



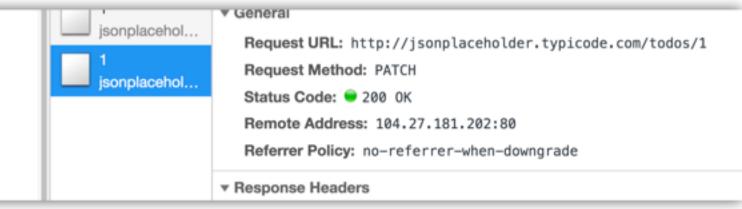
Id: 1
 delectus aut autem

- Id: 2
 quis ut nam facilis et officia qui
 delete
- Id: 3 fugiat veniam minus

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
- Id: 2 quis ut nam facilis et officia qui delete
- Id: 3 fugiat veniam minus delete





Delete một dữ liệu



Delete một dữ liệu

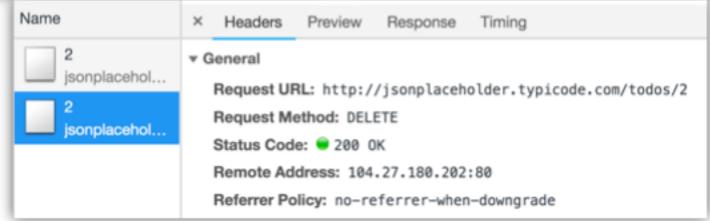
```
export class TodoComponent {
  deleteTodo(i) {
    const todo = this.todoList[i];
    this.todoService.deleteTodo(todo.id).subscribe(
    () => {
      this.todoList = this.todoList.filter(
        t => t.id !== todo.id
```

Sau khi delete thành công thì xóa phần tử khỏi list



Delete một dữ liệu







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



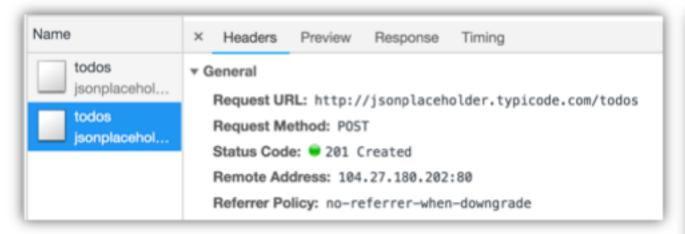
```
export class TodoComponent {
                                                        Partial là built-in class của TypeScript, cho
  addTodo() {
                                                          phép convert tất cả các property của
                                                             class/interface vè optional
    const todo: Partial<ITodo> = {
       title: this.inputControl.value,
       completed: false
    this.todoService.createTodo(todo).subscribe(next => {
       this.todoList.unshift(next);
       this.inputControl.setValue(");
                                                        Sau khi add thành công thì thêm phần tử
    } );
                                                              vào đầu list và reset form
```

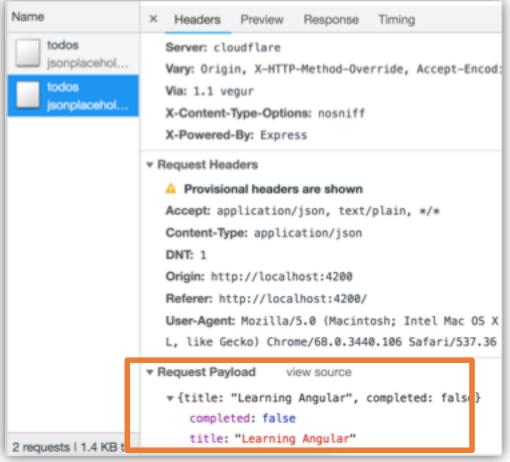


Todo App with HttpClient Id: 1 delectus aut autem delete Id: 2 quis ut nam facilis et officia qui delete Id: 3 fugiat veniam minus delete et porro tempora delete

Todo App with HttpClient ld: 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









- 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.



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



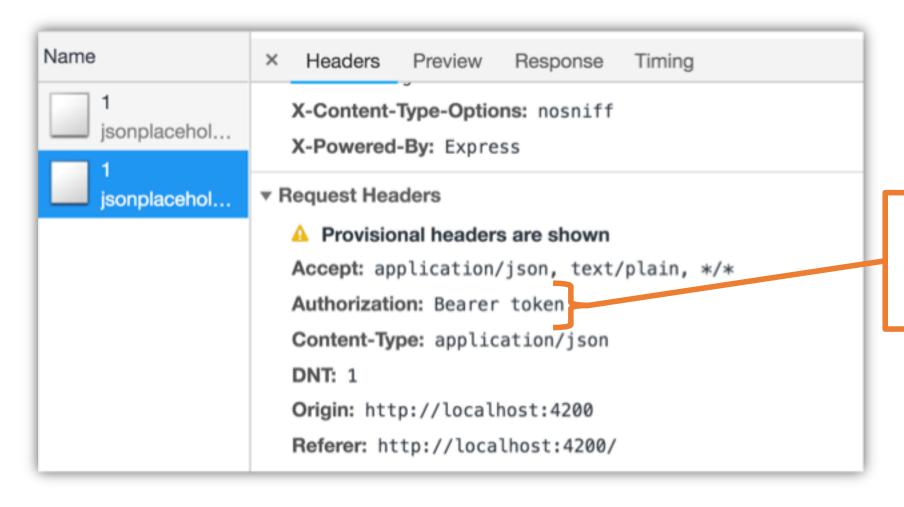
```
import { HttpInterceptor, HttpHandler, HttpRequest, HttpEvent } from '@angular/common/http';
import { Observable } from 'rxis';
export class TokenInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    req = req.clone({
      setHeaders: {
        Authorization: 'Bearer token'
                                                    Intercept request trước khi nó được gửi đi.
                                                   Ứng dụng sẽ tự động chèn thêm các thống tin
                                                                   cần thiết.
    return next.handle(req);
```



```
@NgModule({
  providers: [
      provide: HTTP INTERCEPTORS,
      useClass: TokenInterceptor,
      multi: true
export class AppModule { }
```

Đăng ký DI token với class chúng ta vừa khởi tạo





Interceptor sẽ tự động thêm header này chẳng hạn





Raising the bar