FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

THỰC HỌC – THỰC NGHIỆP

LẬP TRÌNH FRONT-END FRAMEWORK 1

SERVICE VÀ HTTP SERVICE

http://www.poly.edu.vn

Mục tiêu

- Angular giao tiếp với Backend
- Sử dụng module http service để làm việc với Backend
- Tạo và sử dụng service cho yêu cầu với http

NỘI DUNG

- Cách thức Angular giao tiếp với BackEnd
- Thực hiện kết nối với Backend thông qua http service
- Thực hiện tốt với GET,POST, DELETE, PUT
- Hiểu, cài đặt sử dụng service trong Angular
- Sử dụng service kết nối các component
- Quản lý lỗi khi sử dụng http service

PHẦN 1: HTTP SERVICE

Angular giao tiếp với BackEnd như thế thế nào?

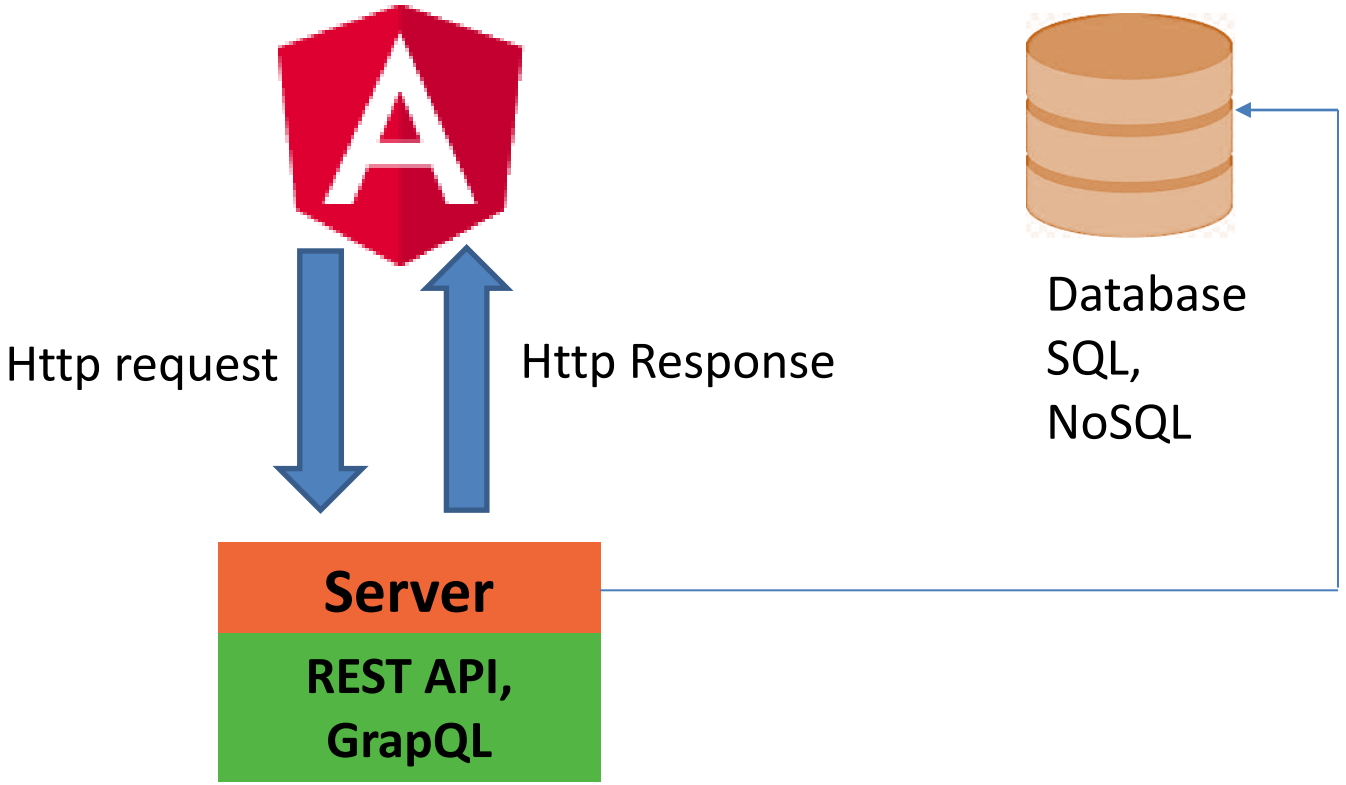
Angular giao tiếp với BackEnd như thế thế nào?

Không thể truy cập trực tiếp database

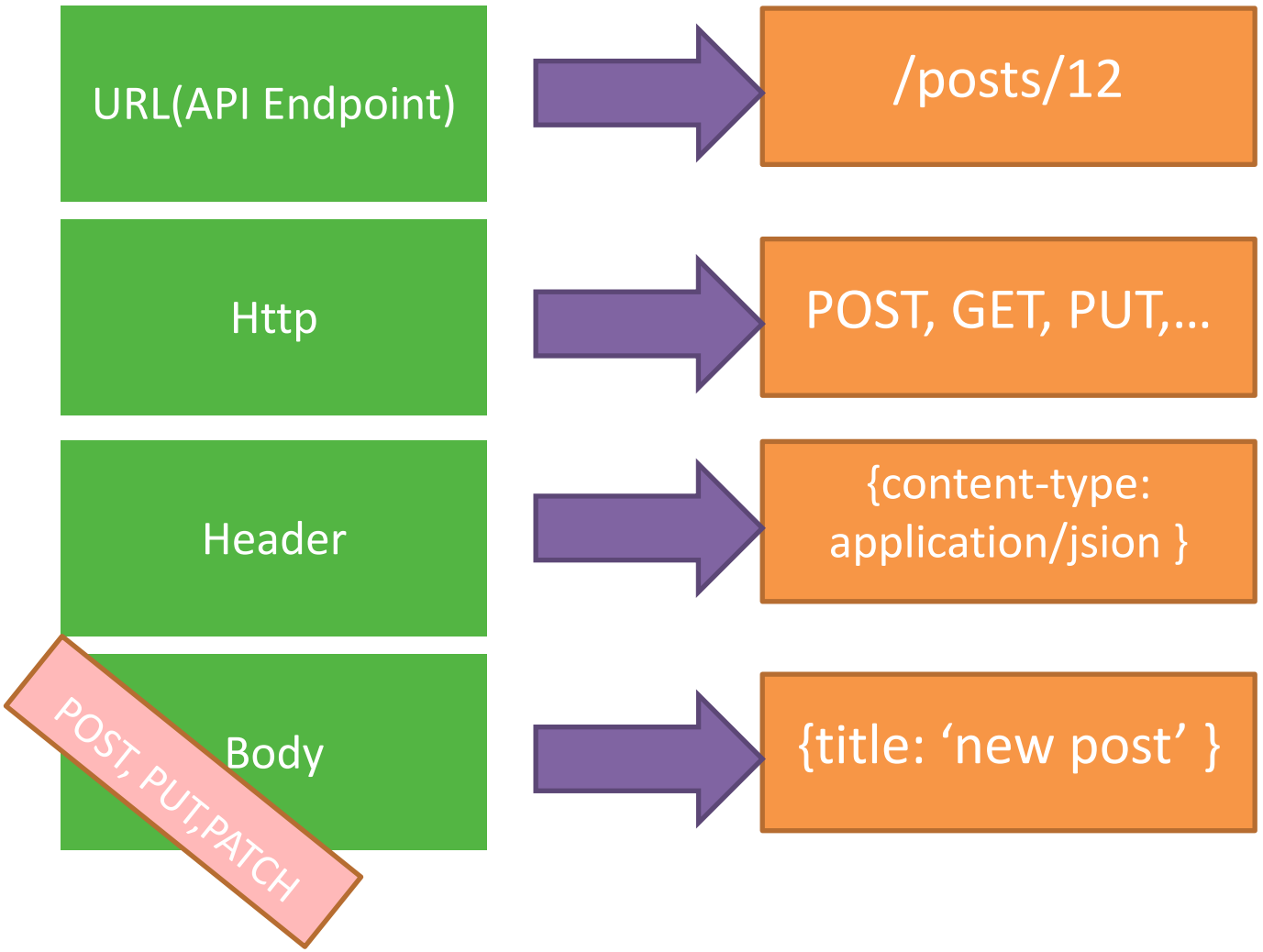
File upload

Phân tích

...



Phân tích Http Request



Cài đặt module http

```
import {HttpClientModule} from '@angular/common/http';
imports: [
BrowserModule,
AppRoutingModule,
FormsModule,
ReactiveFormsModule,
HttpClientModule
]
```

Sử dụng Http với lớp HttpClient

Tại class muốn sử dụng Http request, thực hiện import lớp HttpClient và tạo đối tượng này tại constructor

```
import { HttpClient } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
export class BlogComponent implements OnInit {
constructor(private httpService:HttpClient) { }
ngOnInit(): void {
}
}
```

Sử dụng Http với lớp HttpClient

Thực hiên đọc dữ liệu từ server thông qua phương thức GET của đối tượng HttpClient

```
export class BlogComponent implements OnInit {
url='http://127.0.0.1:3000/blog/';
posts:IPost[]=[];
constructor(private httpService:HttpClient) { }
ngOnInit(): void {
this.getAllPost('posts');
}
getAllPost(endPoint:string){
  this.httpService.get(this.url+endPoint).subscribe(data=>{
  this.posts=data.posts;
  console.log(this.posts);
})
}
}
```

Sử dụng Http với lớp HttpClient

Thực hiện thêm dữ liệu tới server thông qua phương thức POST của đối tượng HttpClient

```
onCreatePost(dataPost:IPost){
this.httpService.post(this.url,dataPost).subscribe(p=>{
  this.getAllPost();
})
}
```

Sử dụng Http với lớp HttpClient

Thực hiện xoá dữ liệu thông qua phương thức DELETE của đối tượng HttpClient

```
onDelete(id:number){
  this.httpService.delete(this.url+id).subscribe(p=>{
  this.getAllPost();
})
}
```

Sử dụng Http với lớp HttpClient

Hiển thị kết quả truy vấn dữ liệu

```
<tbody *ngIf="posts.length>=1">
  <tr *ngFor="let post of posts">
        <td>{{post.postId}}</td>
        <td>{{post.title}}</td>
        <td>{{post.content}}</td>
        <td>Xoá</td>
  </tr>
</tbody>
```

THỰC HIỆN ĐỌC/THÊM/XOÁ/SỬA

PHẦN 2: SERVICE

SERVICE

- Để thực hiện code sạch, rõ ràng, thường ta nên tách các chức năng xử lý thành một tầng riêng.
- Trong Angular ta có thể sử dụng service cho http request

Tại sao phải dùng service

- Component không nên truy xuất và thao tác trực tiếp với dữ liệu mà nó chỉ là nơi trình bày dữ liệu.
- Service là một cách chia sẻ thông tin giữa các lớp tách biệt với nhau

Tạo service

ng generate service post

```
import { Injectable } from '@angular/core';
@Injectable({
providedIn: 'root'
})
export class PostService {
constructor() { }
}
```

Tạo service

Viết các hàm thực hiện http request

```
export class PostService {
  url='http://127.0.0.1:3000/blog/posts/';
  constructor(private httpService:HttpClient) { }
  storePost(postData:IPost){
    this.httpService.post(this.url,postData).subscribe(p=>{
    console.log(p);
    })
  }
  getAllPost(){
    return this.httpService.get(this.url);
  }
  deletePost(id:number){
    return this.httpService.delete(this.url+id);
  }
}
```

Sử dụng service

post.component.ts

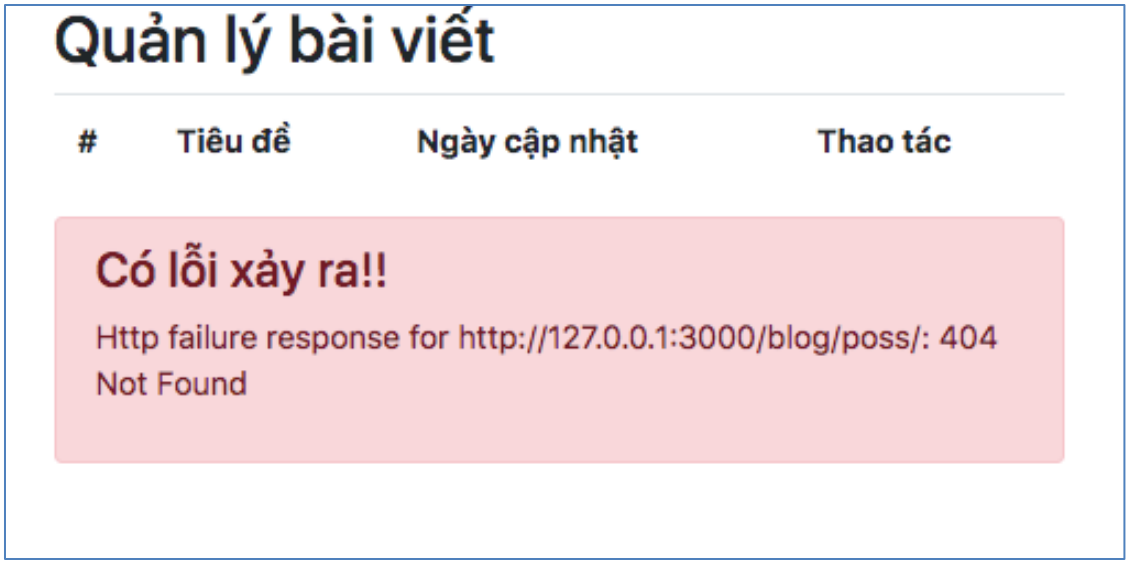
```
import { PostService } from '../post.service';

constructor(private httpService:HttpClient,private postService:PostService)

Các hàm sử dụng postService khi cần thiết
```

Quản lý lỗi: Trong trường hợp giao tiếp với BackEnd có lỗi xảy ra, ta có thể sử dụng tham số thứ 2 của subscribe để nhận thông tin lỗi

```
this.postService.getAllPost().subscribe(data=>{
this.posts=data.posts;
console.log(this.posts);
},error=>{
this.error=error.message;
})
}
```

Quản lý lỗi: Hiển thị lỗi lên trên giao diện

```
<div class="alert alert-danger" *ngIf="error">
<h4>Có lỗi xảy ra!!</h4>
<p>{{error}}</p>
</div>
```



Quản lý lỗi: Thông báo lỗi chính xác dựa vào status của đối tượng error

```
onDelete(id:number){
this.postService.deletePost(id).subscribe(p=>{
console.log(p);
this.getAllPost();
}, (error:Response)=>{
if(error.status===404){
  alert('Không tim thấy!!')
}else{
  alert('Lỗi gì chưa biết!!')
}
})
}
```

Thông báo trạng thái loading…

- Sử dụng biến nhận biết đang thực hiện fetch dữ liệu

```
getAllPost(){
  this.isFetching=true;
    this.postService.getAllPost().subscribe(data=>{
      this.isFetching=false;
        this.posts=data.posts;
        console.log(this.posts);
      },error=>{
        this.error=error.message;
      })
  }
```

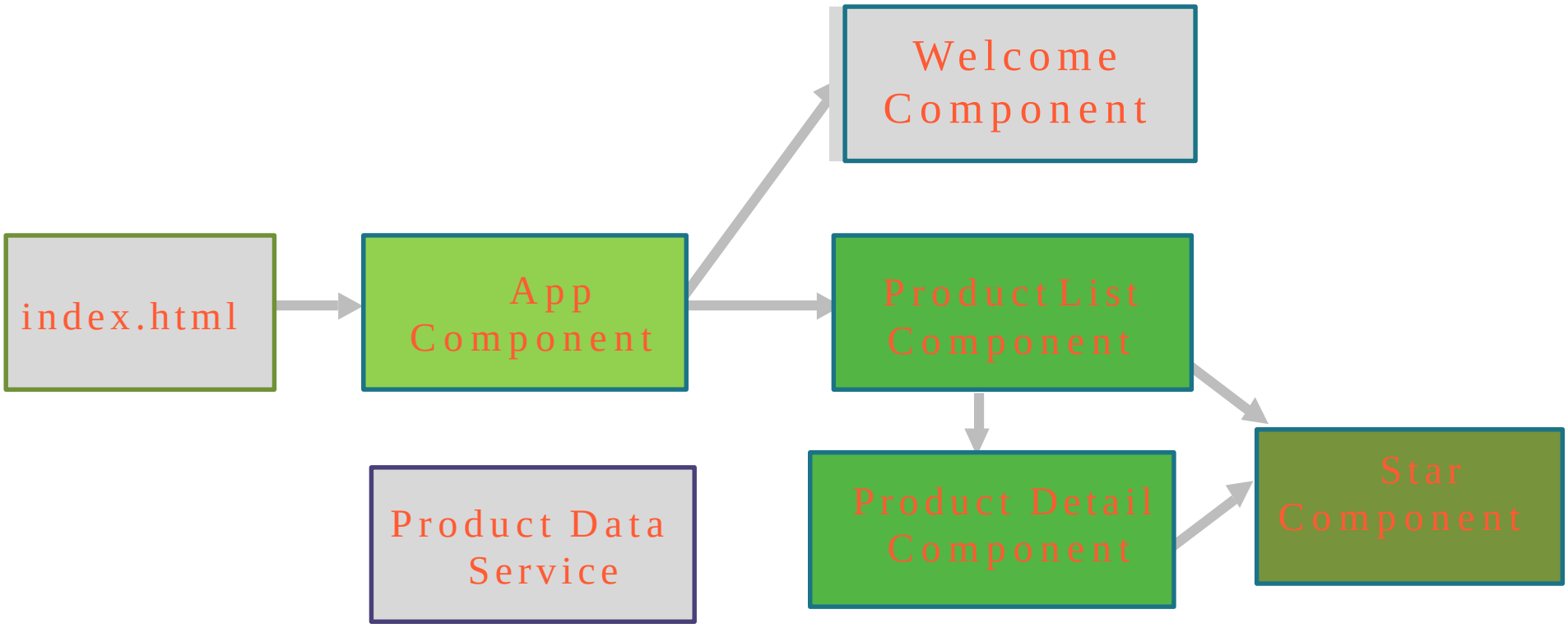
Xử lý lỗi với catchError

- Import đối tượng catchError từ module rxjs/operators

```
getAllPost(){
    this.isFetching=true;
    this.postService.getAllPost().subscribe(data=>{
        this.isFetching=false;
        this.posts=data.posts;
        console.log(this.posts);
    },catchError(error=>{
        throwError(error);
    })
}
```

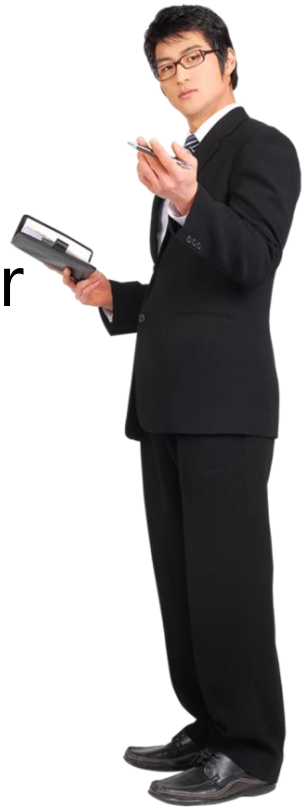
CÁC THAO TÁC ĐỌC, THÊM, XOÁ, SỬA VỚI SERVICE
QUẢN LÝ LỖI

KIẾN TRÚC CỦA ANGULAR



SUMARRY

- ☑ Cách thức Angular giao tiếp với BackEnd
- ☑ Thực hiện kết nối với Backend thông qua http service
- ☑ Thực hiện tốt với GET,POST, DELETE, PUT
- ☑ Hiểu, cài đặt sử dụng service trong Angular
- ☑ Sử dụng service kết nối các component
- ☑ Quản lý lỗi khi sử dụng http service




thank
you!