

1. **GraphQL là gì ?**

GraphQL là một Graph Query Language được dành cho API. Nó được phát triển bởi Facebook và hiện tại nó được duy trì bởi rất nhiều công ty lớn, và mọi cá nhân trên khắp thế giới. GraphQL từ khi ra đời đã gần như thay thế hoàn toàn REST bởi sự hiệu quả, mạnh mẽ và linh hoạt hơn rất nhiều.

GraphQL đối với .NET nói riêng, hiện tại chỉ hỗ trợ cho ASP.NET Core hoặc .NET 5 trở lên

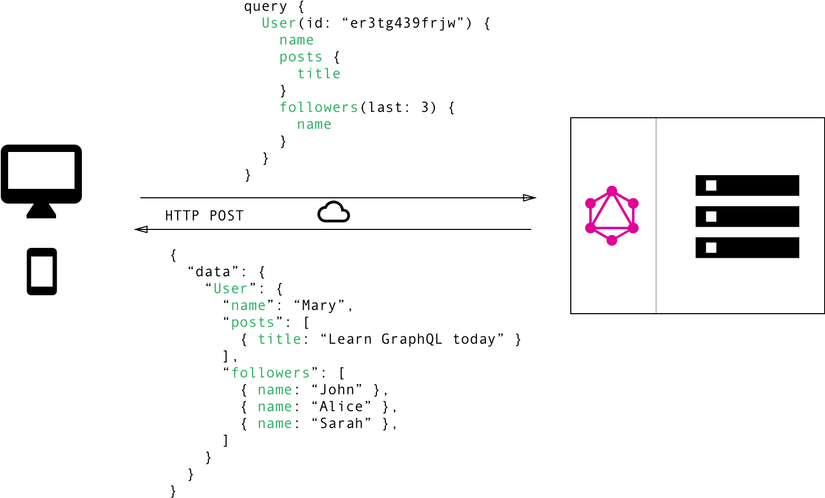
1. **Tại sao GraphQL sẽ thay thế REST trong tương lai ?**

Lấy Facebook làm ví dụ. Facebook phải quản lí vô số data source và API clients mà **REST API** lại lộ khuyết điểm thiếu linh hoạt do tính chất dựa trên tài nguyên cố định, dẫn đến trường hợp “nghẽn cổ chai” thường thấy. Chính vì vậy, thay vì có đến hàng tá “endpoint” dư thừa, Facebook đã nghĩ đến giải pháp chỉ dùng một “endpoint” thông minh với khả năng tiếp thu những Query phức tạp rồi đưa ra output data với loại type tùy theo yêu cầu của client

**REST API:**



**GraphQL:**

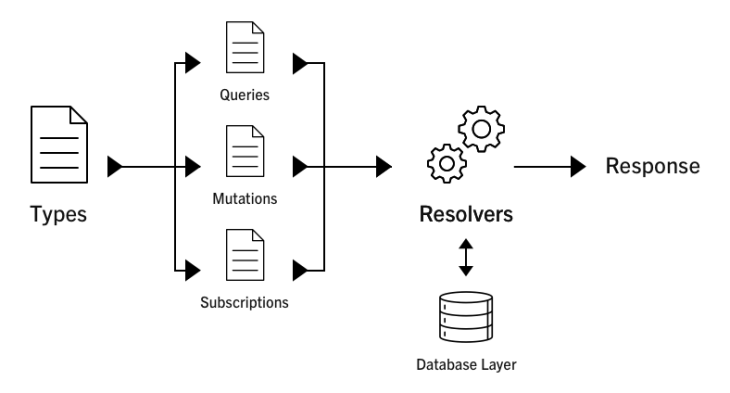


1. **GraphQL có nhược điểm gì không ?**

Tuy rằng GraphQL rất vượt trội so với REST nhưng không có nghĩa nó không có những khuyết điểm thua thiệt hơn so với REST, GraphQL có những khuyết điểm cụ thể như sau:

* GraphQL luôn trả về HTTP status code là 200, bất kể query có success hay không. Nếu query fail, JSON response trả về sẽ có key errors, với các error message và stacktrace. Điều này gây khó khăn hơn trong việc error handling.
* Phải thiết kế Schema trước, sẽ vất vả hơn vì thêm việc mặc dù sau này schema sẽ giúp bạn ngăn chặn nhiều lỗi và đỡ tốn sức hơn khi nâng cấp.
* Khi Client không cần quan tâm đến data lấy từ đâu thì Sự phức tạp được đẩy về phía server, GraphQL không phải là giải pháp tốt cho các ứng dụng đơn giản.
* Vấn đề caching. REST API sử dụng nhiều endpoint nên nó tận dùng HTTP caching để tránh việc phải tìm nạp lại tài nguyên. Với GraphQL, nó sử dụng 1endpoint thay vì theo cơ chế caching của HTTP. Lưu vào bộ nhớ đệm là rất quan trọng vì nó làm giảm sự truy cập vào máy chủ. Với GraphQL, bạn phải sử dụng thư viện khác phía Client để đáp ứng việc caching.

1. **Cấu trúc GraphQL**



GraphQL chia thành 3 phần chính :

* Query:  Các câu lệnh lấy dữ liệu (tương tự method GET trong REST API)
* Mutation: Các câu lệnh để thêm/sửa dữ liệu (tuơng tự method POST/PUT/DELETE trong REST API)
* Subscription: Câu lệnh dùng để lắng nghe sự kiên trên server, khi dữ liệu trên server được thay đổi nó cũng thay đổi theo nhằm cung cấp dữ liệu cho client một cách realtime nhất có thể. Ở điểm này REST API không có cửa để đấu lại với GraphQL.

1. **Bắt đầu với GraphQL ASP.NET core**

Trước khi bắt đầu chúng ta sẽ cài đặt những Framework cần thiết

Tools -> Nuget Package Manager -> Package Manager Console

**Install-Package HotChocolate.AspNetCore**

**Install-Package HotChocolate.AspNetCore.Playground**

**Install-Package HotChocolate.Data.EntityFramework**

**Install-Package Microsoft.EntityFrameworkCore.SqlServer**

**Install-Package Microsoft.EntityFrameworkCore.Tools**

**Install-Package Microsoft.VisualStudio.Web.CodeGeneration.Design**

Sau khi cài đặt chúng ta sẽ kết nối với cơ sở dữ liệu

* Ta vào SQL Server Object Explorer -> chuột phải SQL Server -> Add SQL Server -> nhập vào thông tin Server Name, Username, Password ( nếu có ) sau đó chọn Connect
* Ta tạo thư mục Models trong Project, sau đó ta sẽ generate Models theo câu lệnh sau trong console:

**Scaffold-DbContext “Server=\*\*\*\*\*\*;Database=Inventory;Integrated Security=True” Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models**

Ta có thể thấy trong models sẽ được tạo ra 1 file **xxxContext.cs**, ta sẽ xóa bỏ đi cái hàm **OnConfiguration**

* Sau đó ta vào **appsetting.json** thêm vào ConnectionString như sau:

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"DefaultConnection": "Server=\*\*\*\*\*\*;Database=Inventory;Integrated Security=True"

}

}

* Sau đó ta sẽ vào **Startup.cs** setup kết nối tới Database:

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)

{

services.AddPooledDbContextFactory<BookShellContext>(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

}

* Setup đường dẫn cho GraphQL:

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UsePlayground(new PlaygroundOptions

{

QueryPath = "/api",

Path = "/playground"

});

}

app.UseGraphQL("/api");

}

* Tạo 1 file class **Query.cs** trong project

Trong đây ta sẽ setup Get dữ liệu từ Database

VD mình muốn lấy tất cả sách từ Database, ta sẽ làm như sau

using HotChocolate;

using HotChocolate.Data;

using MyGraphQLs.Models;

using System.Linq;

public class Query

{

[UseDbContext(typeof(BookShellContext))]

[UseProjection]

public IQueryable<Book> GetBooks([ScopedService] BookShellContext db) => db.Books;

}

* [UseProjection] sẽ giúp chúng ta trong việc truy xuất đúng các dữ liệu của GraphQL trên những table khác có mối quan hệ với nhau, hạn chế được Overfetch và UnderFetch
* Để có thể Thêm xóa dữ liệu ta sẽ tạo thêm 1 file class **Mutation.cs**

using HotChocolate;

using HotChocolate.Data;

using MyGraphQLs.Models;

using System.Linq;

* Để có thể thực hiện việc thêm xóa, đầu tiên ta phải tạo 1 models class để lưu trữ dữ liệu được nhập

public class CreateBookInput

{

public string bookName { get; set; }

public string bookAuthor { get; set; }

public double? Price { get; set; }

public string img { get; set; }

}

public class DeleteBookId

{

public int id { get; set; }

}

* Sau đó ta có thể setup hàm Thêm xóa

public class Mutation

{

[UseDbContext(typeof(BookShellContext))]

public bool CreateBook(CreateBookInput model, [ScopedService] BookShellContext db)

{

try

{

var book = new Book

{

BookName = model.bookName,

BookAuthor = model.bookAuthor,

Price = model.Price,

Img = model.img,

};

db.Books.Add(book);

db.SaveChanges();

return true;

}

catch

{

return false;

}

}

[UseDbContext(typeof(BookShellContext))]

public bool DeleteBook(DeleteBookId model, [ScopedService] BookShellContext db)

{

try

{

var BookToDelete = db.Books.Single(x => x.Id == model.id);

db.Books.Remove(BookToDelete);

db.SaveChanges();

return true;

}

catch

{

return false;

}

}

}

* Cuối dùng ta sẽ setup Schema cho GraphQL trong **Setup.cs** thêm vào trongConfigureServices để kết nối với những gì mình tạo ra

services

.AddGraphQLServer()

.AddMutationType<Mutation>()

.AddProjections()

.AddType<Book>()

.AddQueryType<Query>();

Project Mẫu:

<https://github.com/CapooCat/MyGraphQL>

Thông tin chi tiết về GraphQL và các hướng dẫn khác: https://chillicream.com/docs/hotchocolate/