Senparc 盛派®

密级：内部

使用ASP.NET Core 搭建GraphQL服务

2019.12.4 · 苏州盛派网络科技有限公司

主持 / 分享：伏允坤

Senparc 盛派®

阅读以下内容或参与相关会议即表示你已同意不可撤销的信息安全规定：

以下展示所有内容未经苏州盛派网络科技有限公司(以下简称：盛派网络）书面许可或官方发布，不得进行任何形式的拍摄、录音、转载。

Senparc 盛派®

GraphQL是什么

一种用于 API 的查询语言
https://graphql.cn/

GraphQL是什么

Senparc 盛派®

- GraphQL 是一个用于 API 的查询语言，是一个使用基于类型系统来执行查询的服务端运行时（类型系统由你的数据定义），使得客户端能够准确地获得它需要的数据，而且没有任何冗余。
- GraphQL与数据库、开发语言、框架无关可以用于任何平台或语言。
- GraphQL和SQL一样是一套规范。
- GraphQL的数据源可以是各种各样得API，可以是各种服务，甚至可以是数据库。

Senparc 盛派®

GraphQL历史

Facebook
https://graphql.cn/

GraphQL历史

Senparc 盛派®

- 来源Facebook
- 2012年Facebook开始开发，2015年开源

Senparc 盛派®

GraphQL和RESTful区别

国内官网

https://graphql.cn/

RESTful

Senparc 盛派®

- 数据过度获取|数据获取不足
- 产品快速迭代需要频繁修改数据

GraphQL

Senparc 盛派®

```
{
  hero {
    name
    height
  }
}
```

```
{
  "hero": {
    "name": "Luke Skywalker",
    "height": 1.72
  }
}
```

请求你所要的数据
不多不少

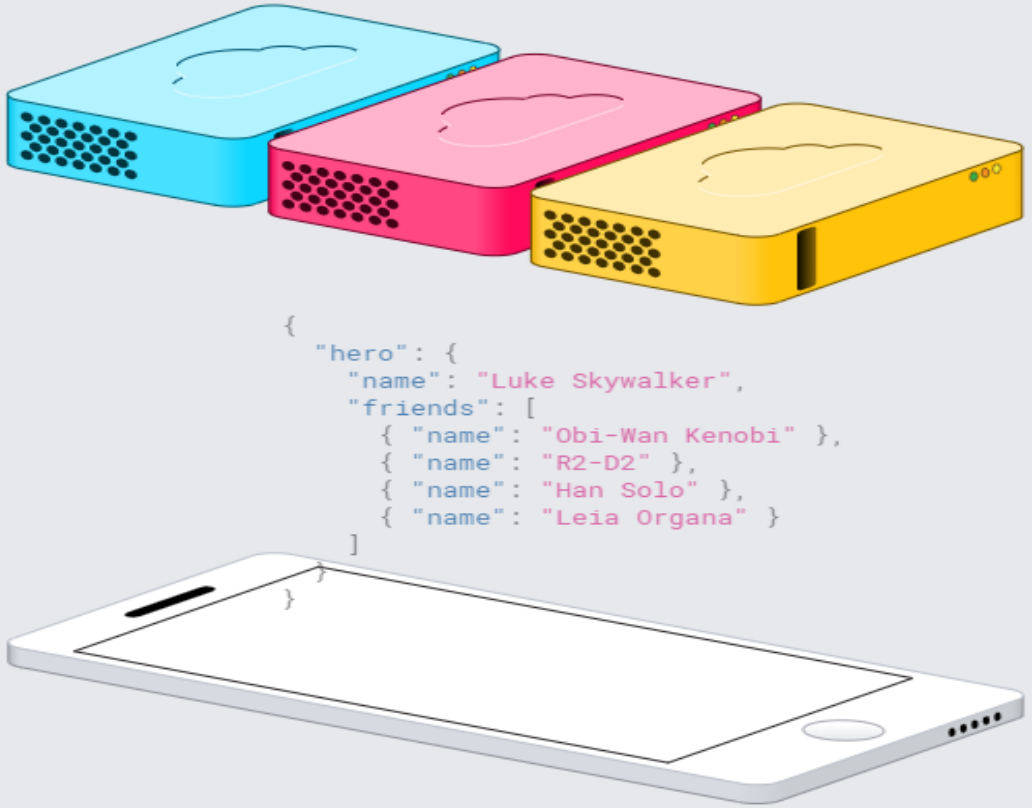
向你的 API 发出一个 GraphQL 请求就能准确获得你想要的数据，不多不少。 GraphQL 查询总是返回可预测的结果。使用 GraphQL 的应用可以工作得又快又稳，因为控制数据的是应用，而不是服务器。

GraphQL()

Senparc 盛派®

获取多个资源
只用一个请求

GraphQL 查询不仅能够获得资源的属性，还能沿着资源间引用进一步查询。典型的 REST API 请求多个资源时得载入多个 URL，而 GraphQL 可以通过一次请求就获取你应用所需的所有数据。这样一来，即使是比较慢的移动网络连接下，使用 GraphQL 的应用也能表现得足够迅速。



RESTful

Senparc 盛派®

① ② ③

HTTP GET

```
{
  "user": {
    "id": "er3tg439frjw"
    "name": "Mary",
    "address": { … },
    "birthday": "July 26, 1982"
  }
}
```

HTTP GET

```
{
  "posts": [{
    "id": "ncwon3ce89hs"
    "title": "Learn GraphQL today",
    "content": "Lorem ipsum … ",
    "comments": [ … ],
  }]
}
```

```
{
  "followers": [{
    "id": "leo83h2dojsu"
    "name": "John",
    "address": { … },
    "birthday": "July 26, 1982"
  },
  …]
}
```

HTTP GET

/users/<id>

/users/<id>/posts

/users/<id>/followers

/users/<id>

/users/<id>/posts

/users/<id>/followers

/users/<id>

/users/<id>/posts

/users/<id>/followers

GraphQL解决了什么问题

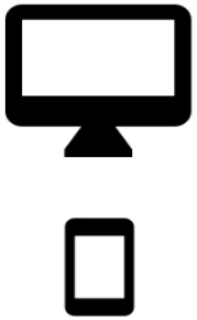
Senparc 盛派®

- API字段定制化，按需获取字段
- 数据聚合，一次请求拿到所需数据
- 完备的类型校验机制

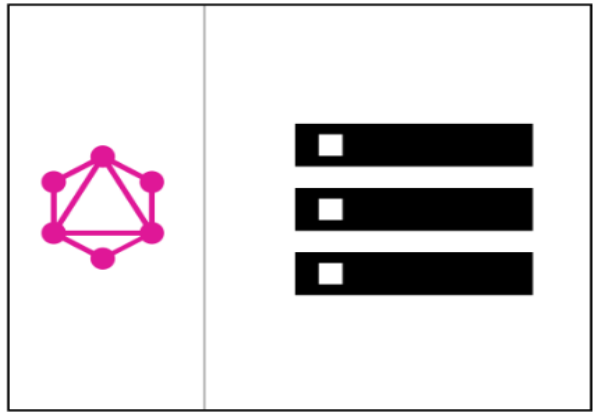
GraphQL

Senparc 盛派®

```
query {
  User(id: “er3tg439frjw”) {
    name
    posts {
      title
    }
    followers(last: 3) {
      name
    }
  }
}
```



HTTP POST



```
{
  “data”: {
    “User”: {
      “name”: “Mary”,
      “posts”: [
        { title: “Learn GraphQL today” }
      ],
      “followers”: [
        { name: “John” },
        { name: “Alice” },
        { name: “Sarah” },
      ]
    }
  }
}
```

Query(查询)

Senparc 盛派®

```
# Write your query or mutation here
query SchoolQuery($id: Int!) {
  school(id: $id) {
    id
    name
    description
    addTime
  }
}
```

QUERY VARIABLES HTTP HEADERS

```
{
  "id": 9
}
```

Headers Preview Response Timing

General

Request URL: https://localhost:5001/api/graphql
Request Method: POST
Status Code: 200
Remote Address: [::1]:5001
Referrer Policy: no-referrer-when-downgrade

Response Headers

content-type: application/json
date: Tue, 03 Dec 2019 13:35:18 GMT
server: Kestrel
status: 200

Request Headers

:authority: localhost:5001
:method: POST
:path: /api/graphql
:scheme: https
accept: */*
accept-encoding: gzip, deflate, br
accept-language: zh-CN,zh;q=0.9,en;q=0.8,fr;q=0.7,und;q=0.6
content-length: 172
content-type: application/json
origin: https://localhost:5001
referer: https://localhost:5001/ui
sec-fetch-mode: cors
sec-fetch-site: same-origin
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904
afari/537.36

Request Payload view source

{operationName: "SchoolQuery", variables: {id: 9},…}
operationName: "SchoolQuery"
query: "query SchoolQuery($id: Int!) {↵ school(id: $id) {↵   id↵   name↵   description↵   addTime↵
variables: {id: 9}
id: 9

Mutation(修改)

Senparc 盛派®

```
mutation CreateTeacherMutation($inputTeacher: TeacherInputType!) {
  createTeacher(teacher: $inputTeacher) {
    id
    school {
      id
      name

    }
  }
}
```

QUERY VARIABLES HTTP HEADERS

```
{
  "id": 5,
  "inputTeacher": {
    "name": "伏允坤",
    "schoolId": 8,
    "sex": "男",
    "age": "25",
    "education": "教授",
    "phone": "+8615351652552"
  }
}
```

Headers Preview Response Timing

General

Request URL: https://localhost:5001/api/graphql
Request Method: POST
Status Code: 200
Remote Address: [::1]:5001
Referrer Policy: no-referrer-when-downgrade

Response Headers

content-type: application/json
date: Tue, 03 Dec 2019 13:38:24 GMT
server: Kestrel
status: 200

Request Headers

:authority: localhost:5001
:method: POST
:path: /api/graphql
:scheme: https
accept: */*
accept-encoding: gzip, deflate, br
accept-language: zh-CN,zh;q=0.9,en;q=0.8,fr;q=0.7,und;q=0.6
content-length: 533
content-type: application/json
origin: https://localhost:5001
referer: https://localhost:5001/ui
sec-fetch-mode: cors
sec-fetch-site: same-origin
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.39

Request Payload view source

```
{operationName: "CreateTeacherMutation", variables: {id: 5,…},…}
  operationName: "CreateTeacherMutation"
  query: "mutation CreateTeacherMutation($inputTeacher: TeacherInputType!) {↵  createTeacher(teacher: $i
  variables: {id: 5,…}
    id: 5
    inputTeacher: {name: "伏允坤", schoolId: 8, sex: "男", age: "25", education: "教授", phone: "+86153516
```

GraphQL工具

Senparc 盛派®

- GraphiQL
- Playground
- Voyager

Query和Mutation

Senparc 盛派®

- 字段(Fields)
- 参数(Arguments)
- 片段(Fragments)
- 操作名称（Operation name）
- 变量（Variables）

Schema 和类型

Senparc 盛派®

- 什么是Schema
- 对象类型和字段（Object Types and Fields）
- 标量类型（Scalar Types）
- 输入类型（Input Types）
- 接口（Interfaces）

什么是Schema

Schema定义了GraphQL API的类型系统，Schema是一套类型定义用来对我们所需要的数据的确切描述，它完整描述了客户端可以访问的所有数据（对象、成员变量、关系、任何类型）。

服务器都会根schema验证并执行查询和修改，schema存放于GraphQL API服务器。

Schema设计

Senparc 盛派®

- Schema First 前后端在数据类型上保持一致
- 使用SDL（Schema Definition Language）语言定义Schema它引入了一套类型系统来对模型进行约束
- SDL和无关开发语言或框架
- Schema定义文件得后缀通常是.graphql

- 对象类型和字段（Object Types and Fields）

Senparc 盛派®

```
#枚举类型
enum Episode {
  NEWHOPE
  EMPIRE
  JEDI
}
```

```
type Character {
  name: String!
  appearsIn: [Episode]!
}
```

标量类型（Scalar Types）

Senparc 盛派®

- Int：有符号 32 位整数。
- Float：有符号双精度浮点值。
- String：UTF-8 字符序列。
- Boolean：true 或者 false。
- ID：ID 标量类型表示一个唯一标识符，通常用以重新获取对象或者作为缓存中的键。

输入类型（Input Types）

Senparc 盛派®

```
input ReviewInput {
  stars: Int!
  commentary: String
}
```

```
mutation CreateReviewForEpisode($review: ReviewInput!) {
  createReview(review: $review) {
    stars
    commentary
  }
}
```

接口（Interfaces）

Senparc 盛派®

```
interface Character {
  id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
}
```

```
type Human implements Character {
  id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
  starships: [Starship]
  totalCredits: Int
}
```

```
type Droid implements Character {
  id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
  primaryFunction: String
}
```

Nuget

- GraphQL.Server.Transports.AspNetCore
- GraphQL
- GraphQL.Server.Ui.Playground
- GraphQL.Server.Ui.Voyager
- GraphQL.Server.Ui.GraphiQL

谁在使用GraphQL?

Senparc 盛派®

Senparc 盛派®

谢谢!

伏允坤
E-mail: yfu@senparc.com