Atitit 提升效率 查询语言**GraphQL**

目录

[1. 接口能不能不写死，把静态变成动态？ 1](#_Toc5935)

[1.1. GraphQL最大的优势，就是它能够大大提高开发者的效率 1](#_Toc16330)

[1.2. GraphQL和RESTful的区别    2](#_Toc3394)

[2. 框架 2](#_Toc17070)

[2.1. 用GraphQL.js 2](#_Toc11740)

[2.2. graphql-java  2](#_Toc23339)

[3. ref 4](#_Toc4901)

# 接口能不能不写死，把静态变成动态？

****回答是可以的，这就是GraphQL所做的!****

****定制化的数据，完全根据你查什么返回什么结果。这就是GraphQL被称作API查询语言的原因。****

## 数据查询语言(DQL)、 sql graphql

## GraphQL最大的优势，就是它能够大大提高开发者的效率

总结

GraphQL最大的优势，就是它能够大大提高开发者的效率，而且最大化地简化了前端的数据层的复杂性，并且使得前后端对数据的组织观点一致。只是使用时，需要考察scale, performance, tech stack, migration等等方面的要求，做合理的trade-off，否则它可能不仅没能提高开发者效率，反倒制造出更多的问题。

## GraphQL和RESTful的区别

        前面提到GraphQL可以理解为基于RESTful的一种封装，目的在于构建使Client更加易用的服务，可以说GraphQL是更好的RESTful设计。在过去的十多年中，REST已经成为设计web api的标准(虽然只是一个模糊的标准)。它提供了一些很棒的想法，比如无状态服务器和结构化的资源访问。然而REST api表现得过于僵化，无法跟上访问它们的客户的快速变化的需求。 GraphQL的开发是为了应付更多的灵活性和效率，它解决了与REST api交互时开发人员所经历的许多缺点和低效之处。 为了说明在从API获取数据时REST和GraphQL之间的主要区别，让我们考虑一个简单的示例场景：在blog应用程序中，应用程序需要显示特定用户的文章的标题。同一屏幕还显示该用户最后3个关注者的名称。REST和GraphQL如何解决这种情况?

## ****GraphQL是什么?****它是REST API的替代品

GraphQL是Facebook开发的一种数据查询语言，并于2015年公开发布。它是REST API的替代品

## 各种QL已经说明了前端或者说者UI才是数据的实际需求者

这是前端或者说是UI发展的必经阶段，各种QL已经说明了前端或者说者UI才是数据的实际需求者，既然是需求者，就只有自己才知道数据要怎样拿，格式应该怎样。现在前后端分工的不合理，或者前端视角的狭小，或者架构师和后端人员的偏见，导致凡是涉及到业务数据实际获取时都要借助于所谓的后端开发人员之手。  
node.js也好，graphQL也罢，都是前端的一种觉醒。只有自己能够掌握从定义格式，获取数据，到抓出传递，到打包合并等整个过程，才能让UI有更好的性能，更大的灵活。才是真正意义上的UI开发。  
graphQL自助转成SQL，意味着很大程度不再需要后端协助。  
而可预见的未来，必定是前端自己就可以写SQL，自己合并，过滤，分组，排序，组装数据，甚至自己定义传输协议和模式。

如果有可能，尝试让UI开发人员直接写SQL吧，你们会发现，其实SQL真的很美。

作者：kevin zou  
链接：https://www.zhihu.com/question/53562205/answer/135524228  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

# 按照条件查询

### 1. 一次请求获取到所有

一个 GraphQL 服务只暴露一个端点以便客户端能传输必要的查询去检索数据。 使用前面考虑过的相同示例，让我们看看 GraphQL 查询

{

findPost(id: <postId>) {

id

title

content

author

comments {

id

comment

commentedBy

}

}

}

### GraphQL queries

GraphQL查询结构非常简单，易于理解。请看下面的例子：

{

subscribers(publication: "apollo-stack"){

name

email

}

}复制代码

如果我们为[Building Apollo](https://dev-blog.apollodata.com/" \t "_blank)构建了一个API，显而易见这个查询将会返回所有订阅了“apollo-stack”订阅者的name和email。以下是响应的样子：

{

subscribers: [

{ name: "Jane Doe", email: "jane@doe.com" },

{ name: "John Doe", email: "john@doe.com" },

...

]

}

## in查询

profiles(handles: ["zuck", "cocacola"]) { handle ...userFragment ...pageFragment } }

type Query { getAuthor(id: Int): Author getPostsByTitle(titleContains: String): [Post] }

# 框架

## 用GraphQL.js

下面是用GraphQL.js和express-graphql搭建一个的普通GraphQL查询（query）的例子，包括讲解GraphQL的部分类型和参数，已经掌握了的同学可以跳过

## [graphql-java](https://github.com/graphql-java/graphql-java" \t "https://graphql.cn/code/_blank)

一个用于构建 GraphQL API 的 Java 库。

可以执行一个 hello world GraphQL 查询的 graphql-java 代码如下：

import graphql.ExecutionResult;import graphql.GraphQL;import graphql.schema.GraphQLSchema;import graphql.schema.StaticDataFetcher;import graphql.schema.idl.RuntimeWiring;import graphql.schema.idl.SchemaGenerator;import graphql.schema.idl.SchemaParser;import graphql.schema.idl.TypeDefinitionRegistry;

import static graphql.schema.idl.RuntimeWiring.newRuntimeWiring;

public class HelloWorld {

public static void main(String[] args) {

String schema = "type Query{hello: String} schema{query: Query}";

SchemaParser schemaParser = new SchemaParser();

TypeDefinitionRegistry typeDefinitionRegistry = schemaParser.parse(schema);

RuntimeWiring runtimeWiring = newRuntimeWiring()

.type("Query", builder -> builder.dataFetcher("hello", new StaticDataFetcher("world")))

.build();

SchemaGenerator schemaGenerator = new SchemaGenerator();

GraphQLSchema graphQLSchema = schemaGenerator.makeExecutableSchema(typeDefinitionRegistry, runtimeWiring);

GraphQL build = GraphQL.newGraphQL(graphQLSchema).build();

ExecutionResult executionResult = build.execute("{hello}");

System.out.println(executionResult.getData().toString());

// Prints: {hello=world}

}}

# 问题

注入解决，强类型检查

资源消耗，限制查询深度

谨防嵌套查询!该查询允许带有恶意的客户端通过过度复杂的查询执行DoS(拒绝服务)攻击，该攻击将消耗服务器的所有资源:

复杂性分析：分析其深度，收影响行数

元数据查询，要严格限制

# ref

GraphQL 入门介绍 - 不负春光，努力生长 - 博客园.html

执行 \_ GraphQL.html

5个用\_不用GraphQL的理由 - 简书.html

Code \_ GraphQL.html

API 接口设计： GraphQL 和 REST 怎么选择？ - PHP \_ Laravel \_ 全栈 - SegmentFault 思否.html

Facebook开发的一种数据查询语言——GraphQL：安全概述和测试技巧-云栖社区-阿里云.html