WOTD

51CTO

全球软件开发技术峰会

World Of Tech 2017

2017年12月1日-2日 • 深圳中洲万豪酒店

DEVELOPMENT

WOTD

51CTO

使用GraphQL构建API网关思考

陈国兴

百安居&前端架构师

API网关的发展

01 Nginx

02 中间层

03 API网关

01 Nginx优劣

- 安全：身份验证、IP限制
- 限流
- 路由转发
- 日志
- 灵活性不够，难以实现复杂功能

02 API中间层

- 为适配而生
- 单一类型设备
- 功能简单

03 API网关

- 公共服务层
- 多设备
- 微服务的必然产物
- Zuul

From Rest to GraphQL

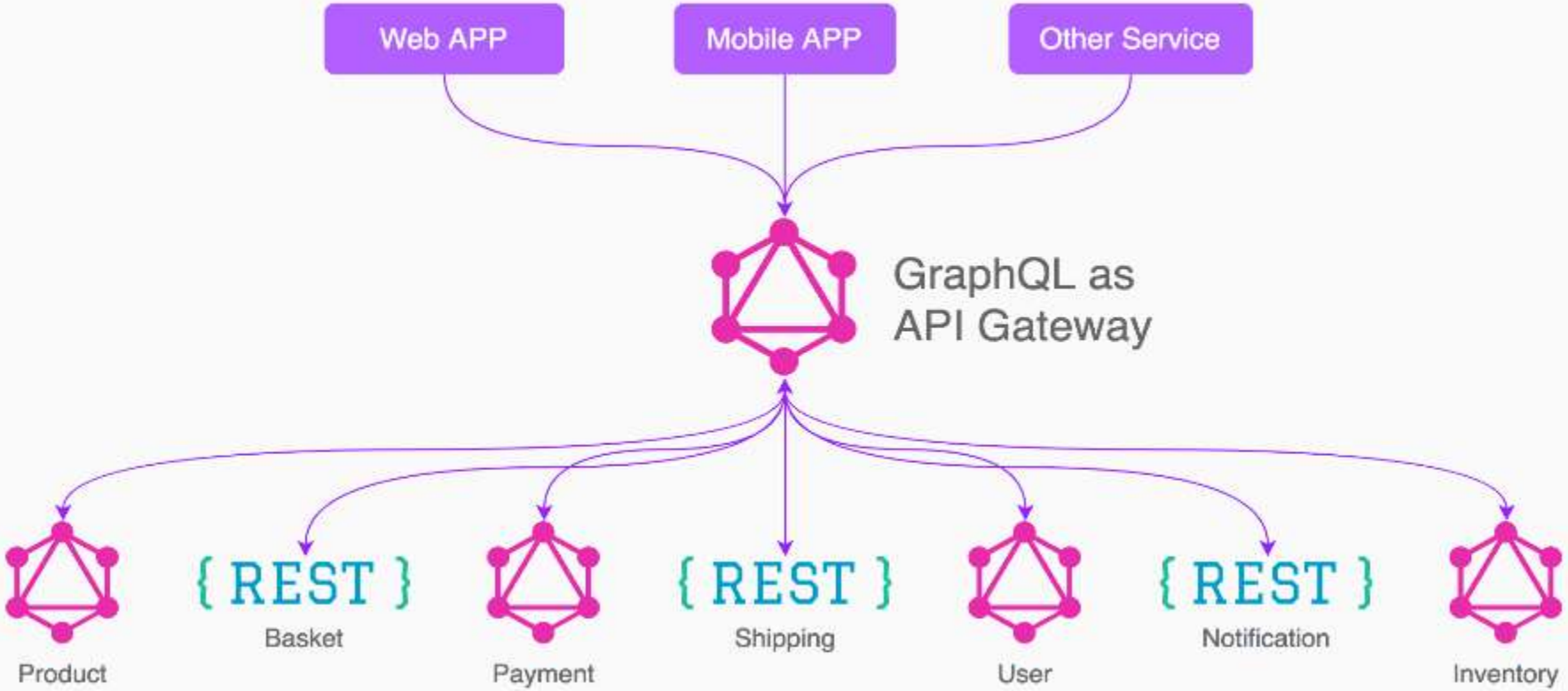
01 标准Rest的问题

02 基于页面设计接口的问题

03 GraphQL解决了什么问题

标准REST

非纯净的rest


Web APP
Mobile APP
Other Service
GraphQL as
API Gateway
{ REST }
Basket
{ REST }
Shipping
{ REST }
Notification
Product
Payment
User
Inventory

GraphQL想要解决什么问题？

你说对了就给你所要的

GraphiQL ▶ Prettify

```
query Main {
  viewer {
    article(id: "1") {
      title
      id
      name
    }

  }
}
```

```
{
  "errors": [
    {
      "message": "Cannot query field \"name\" on type \"article\".",
      "locations": [
        {
          "line": 6,
          "column": 7
        }
      ]
    }
  ]
}
```

输入一个不存在的字段

GraphiQL ▶ Prettify

```
query Main {
  viewer {
    article(id: "1") {
      title
      id
    }

  }
}
```

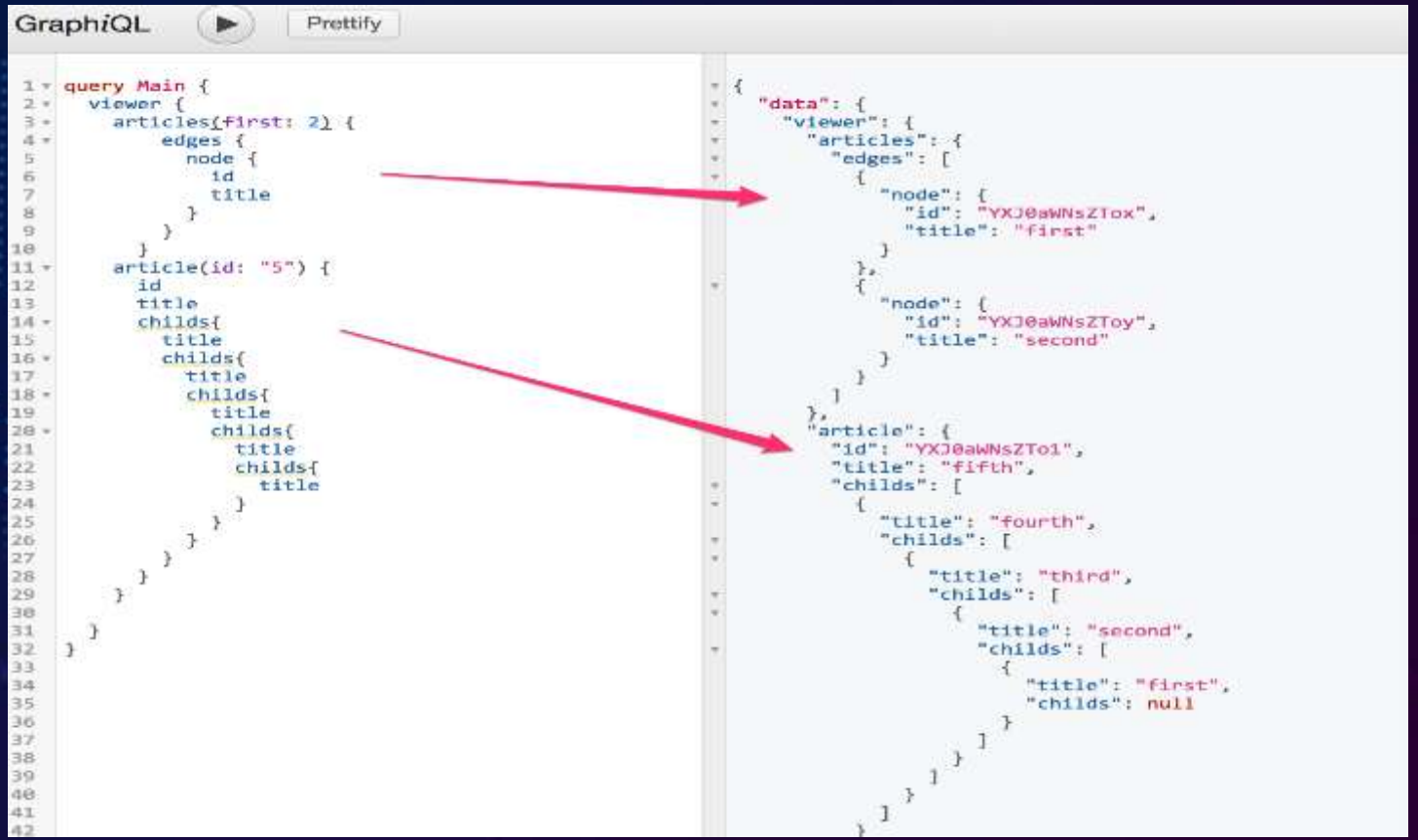
```
{
  "data": {
    "viewer": {
      "article": {
        "title": "first",
        "id": "YXJ0aWNsZTox"
      }
    }
  }
}
```

字段正确

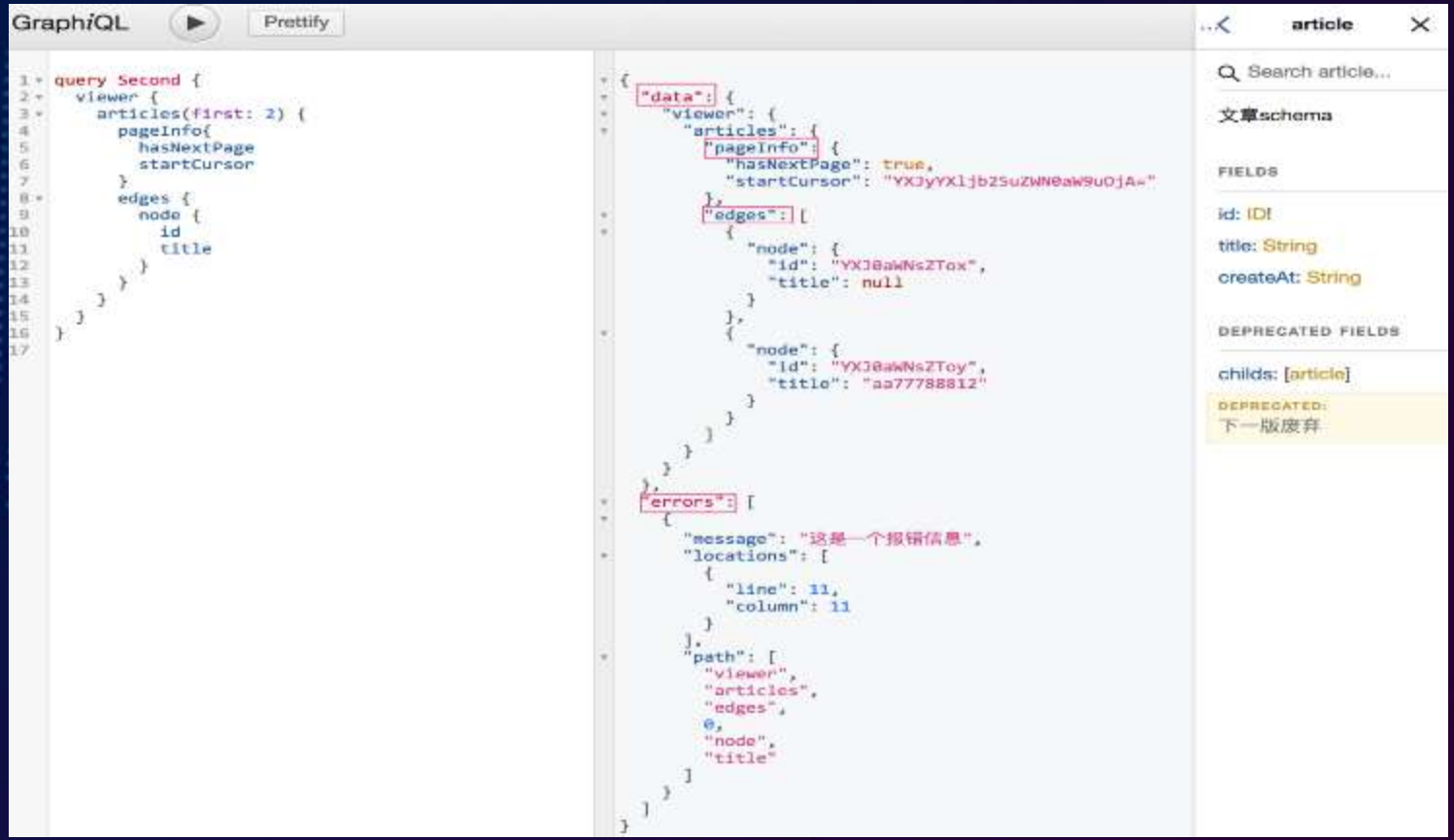
你要什么由你决定


GraphiQL
Prettify
query Main {
viewer {
article(id: "1") {
title
}
}
}
{
"data": {
"viewer": {
"article": {
"title": "first"
}
}
}
}
article
Search article...
文章schema
FIELDS
id: ID!
title: String
createAt: String
DEPRECATED FIELDS
childs: [article]
DEPRECATED:
下一版废弃

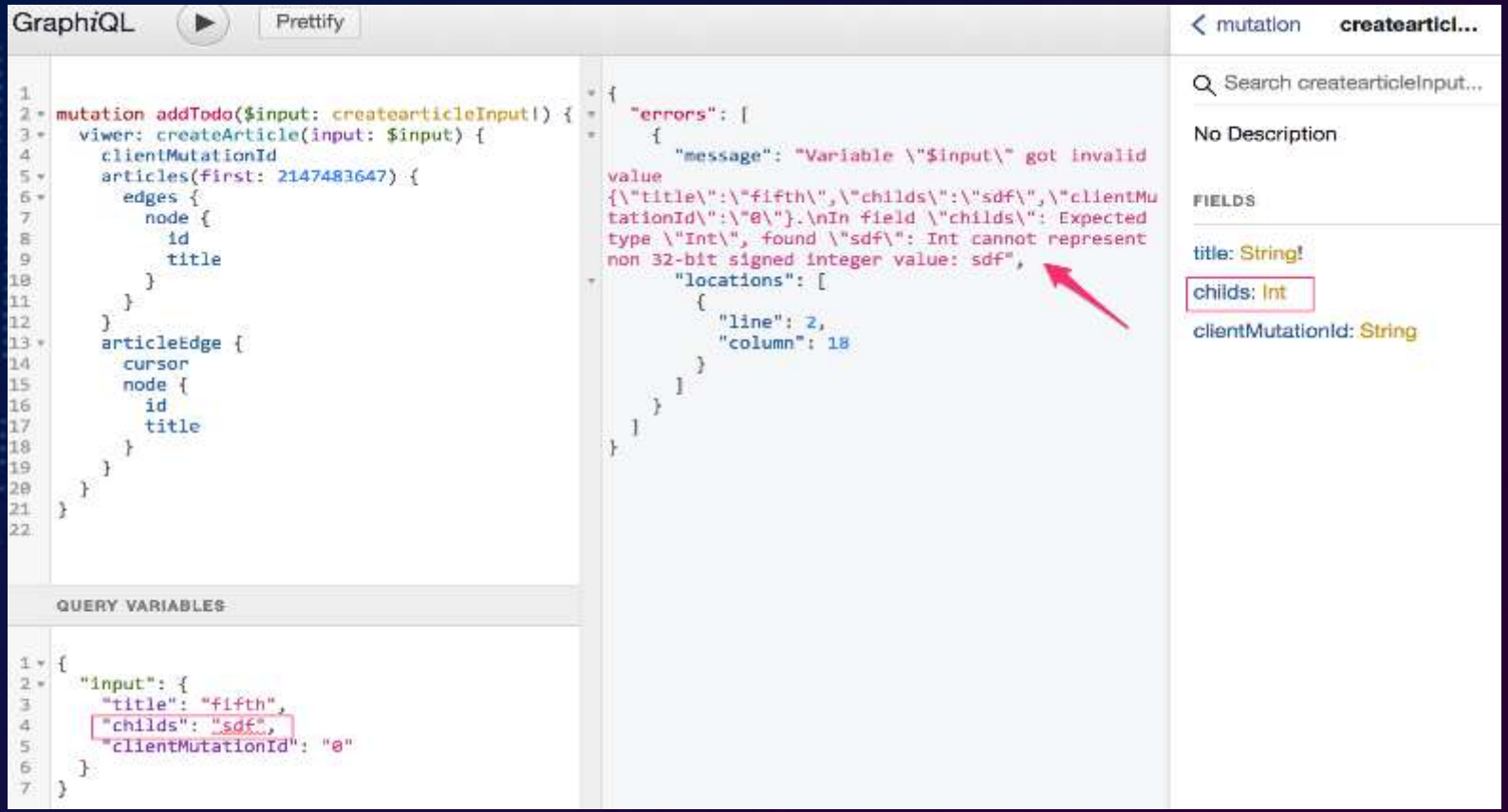
只说一次就够了


GraphiQL
Prettify
query Main {
viewer {
articles(first: 2) {
edges {
node {
id
title
}
}
}
article(id: "5") {
id
title
childs{
title
childs{
title
childs{
title
childs{
title
childs{
title
}
}
}
}
}
}
}
}
{
"data": {
"viewer": {
"articles": {
"edges": [
{
"node": {
"id": "YXJ0aWNsZTox",
"title": "first"
}
},
{
"node": {
"id": "YXJ0aWNsZToy",
"title": "second"
}
}
]
},
"article": {
"id": "YXJ0aWNsZTo1",
"title": "fifth",
"childs": [
{
"title": "fourth",
"childs": [
{
"title": "third",
"childs": [
{
"title": "second",
"childs": [
{
"title": "first",
"childs": null
}
]
}
]
}
]
}

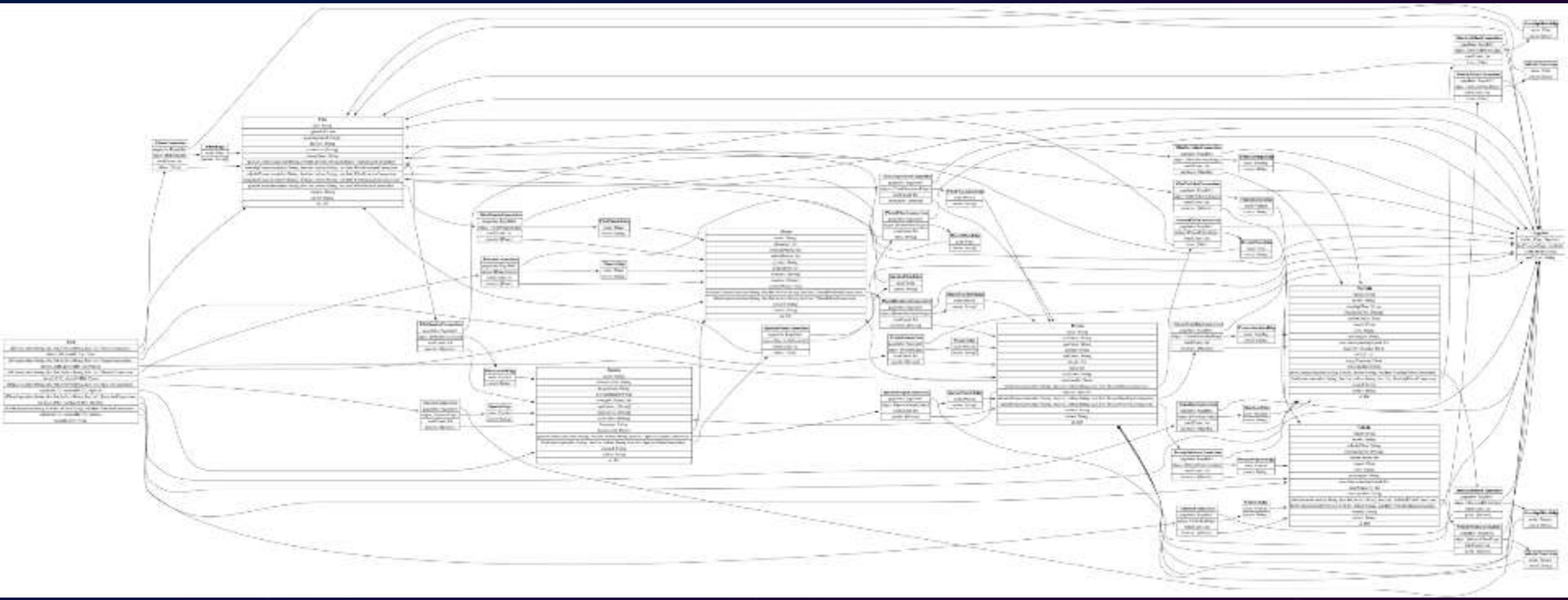
统一响应格式


GraphiQL
Prettify
query Second {
viewer {
articles(first: 2) {
pageInfo{
hasNextPage
startCursor
}
edges {
node {
id
title
}
}
}
}
}
{
"data": {
"viewer": {
"articles": {
"pageInfo": {
"hasNextPage": true,
"startCursor": "YXJyYXljbzSuZWN0aW9uOjA="
},
"edges": [
{
"node": {
"id": "YXJ0aWNsZTox",
"title": null
}
},
{
"node": {
"id": "YXJ0aWNsZToy",
"title": "aa77788812"
}
}
]
}
}
},
"errors": [
{
"message": "这是一个报错信息",
"locations": [
{
"line": 11,
"column": 11
}
],
"path": [
"viewer",
"articles",
"edges",
0,
"node",
"title"
]
}
]
}
article
Search article...
文章schema
FIELDS
id: ID!
title: String
createAt: String
DEPRECATED FIELDS
childs: [article]
DEPRECATED:
下一版废弃

强类型的接口


GraphiQL
Prettify
mutation addTodo($input: createarticleInput!) {
viwer: createArticle(input: $input) {
clientMutationId
articles(first: 2147483647) {
edges {
node {
id
title
}
}
}
articleEdge {
cursor
node {
id
title
}
}
}
}
QUERY VARIABLES
{
"input": {
"title": "fifth",
"childs": "sdf",
"clientMutationId": "0"
}
}
{
"errors": [
{
"message": "Variable \"$input\" got invalid value {\"title\":\"fifth\",\"childs\":\"sdf\",\"clientMutationId\":\"0\"}.\nIn field \"childs\": Expected type \"Int\", found \"sdf\": Int cannot represent non 32-bit signed integer value: sdf",
"locations": [
{
"line": 2,
"column": 18
}
]
}
]
}
mutation
createarticl...
Search createarticleInput...
No Description
FIELDS
title: String!
childs: Int
clientMutationId: String

GraphQL Schema接口设计浅谈



一张GraphQL schema的ER图

把数据库查询的灵活性搬到前端

Graph化（关联）Schema

select * from table

使用缓存解决N+1问题

```
const IntentType = new GraphQLObjectType({
  name: 'Intent',
  description: '意向单',
  fields: ()=>({
    id: globalIdField(),
    address: {
      type: GraphQLString,
      description: '详细地址'
    },
    floorPics: {
      type: new GraphQLList(PicUrlType),
      description: '地面布置图集合',
      resolve: (root,args, {data}) => {
        return data.intent.detail(root.id).then(res=>res.floorPics)
      }
    },
    measurePics: {
      type: new GraphQLList(PicUrlType),
      description: '原始测量图集合',
      resolve: (root,args, {data}) => {
        return data.intent.detail(root.id).then(res=>res.measurePics)
      }
    },
    roofPics: {
      type: new GraphQLList(PicUrlType),
      description: '顶部布置图集合',
      resolve: (root,args, {data}) => {
        return data.intent.detail(root.id).then(res=>res.roofPics)
      }
    },
    planPics: {
      type: new GraphQLList(PicUrlType),
      description: '平面图集合',
      resolve: (root,args, {data}) => {
        return data.intent.detail(root.id).then(res=>res.planPics)
      }
    },
```

使用批量查询优化请求

```
{
  "user": {
   name: "user1",
     posts: [1,3,5,7,9]
  }
}
```

```
select * from posts where id = 1
select * from posts where id = 3
select * from posts where id = 5
select * from posts where id = 7
select * from posts where id = 9
```

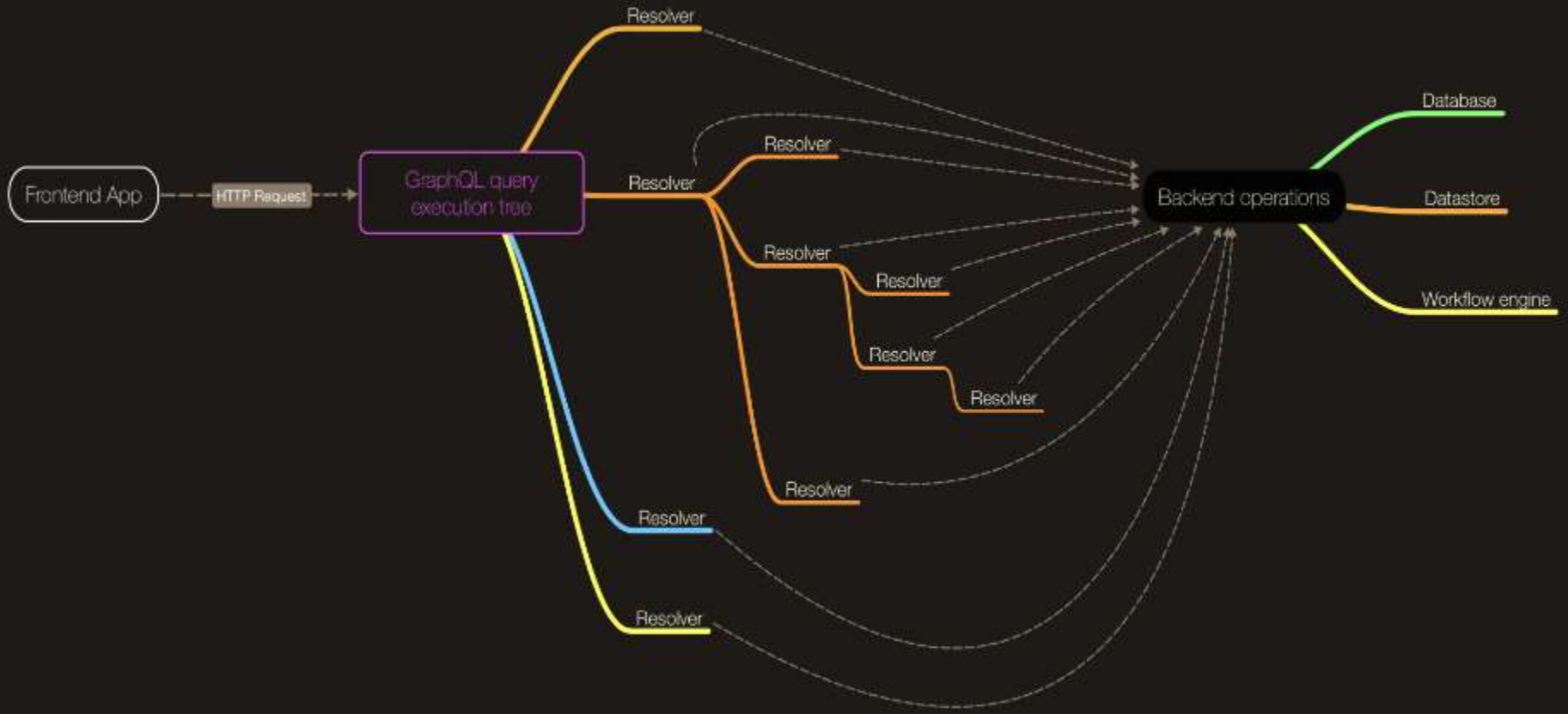
```
select * from posts where id in [1,3,5,7,9]
```

安全

认证和授权

限制不同的客户端的可查询资源

复杂查询管理


Frontend App
HTTP Request
GraphQL query execution tree
Resolver
Resolver
Resolver
Resolver
Resolver
Resolver
Resolver
Resolver
Resolver
Resolver
Backend operations
Database
Datastore
Workflow engine

查询超时限制

```
{
  user(id: 1) {
    name
    email
    friends {
      name
      friends {
        name
        friends {
          name
          friends{
            ......
          }
        }
      }
    }
    posts{
      title
    }
  }
```

超时

查询结果

```
{
  "data": {
    "user": {
      "name": "user1",
      "email": "user1@email.com",
      "friends": [{
        "name": "user2",
        "friends": [{
          "name": "user2",
          "friends": null
        }]
      }],
      posts: null
    }
  }
}
```

查询复杂性成本限制

标准类型
对象类型
列表类型

```
query {
  viewer {
    repositories(first: 50) { # 50
      edges {  # 2
        repository:node { # 2
          name # 1
          pullRequests(first: 20) { # 20*50
            edges {  # 2
              node { # 2
                title # 1
              }
            }
          }
        }
      }
    }
  }
}
# total:  50+2+2+1+100+2+2+1 = 160
```

节点限制

```
query {
  viewer {
    repositories(first: 50) {
      edges {
        repository:node {
          name
          pullRequests(first: 20) {
            edges {
              pullRequest:node {
                title
                comments(first: 10) {
                  edges {
                    comment:node {
                      bodyHTML
                    }
                  ......
}
```

node总数

50 = 50 repositories
+
50 x 20 = 1,000 pullRequests
+
50 x 20 x 10 = 10,000 pullRequest comments
= 11050 total nodes

服务限流

```
query {
  viewer {
    repositories(first: 50) {
      edges {
        repository:node {
          name
          pullRequests(first: 20) {
            edges {
              pullRequest:node {
                title
                comments(first: 10) {
                  edges {
                    comment:node {
                      bodyHTML
                    }
                  ......
}
```

request总数

1(50) = 1 request
+
50 x 1(20) = 50 request
+
50 x 20 x 1 = 1,000 request
= 1050 request

问题

01 开发成本（前期高，后期小）

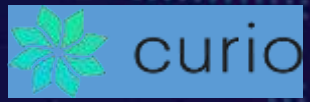
02 学习成本

03 现有系统迁移问题

04 安全问题（成本）

Who's using GraphQL







Thank you!