

拉勾教育

— 互联网人实战大学 —

《前端高手进阶》

朱德龙 前中兴软创主任工程师

— 拉勾教育出品 —

第17讲：前后端如何有效沟通？

GET <https://lagou.com/a>

RPC（Remote Procedure Call，远程过程调用）常用于后端服务进程之间的通信

“远程”指的是不同服务器上的进程

“过程调用”里的“过程”可以理解为“函数”

这种接口设计和函数命名很相似，名称为**动宾结构短语**

RPC—远程过程调用

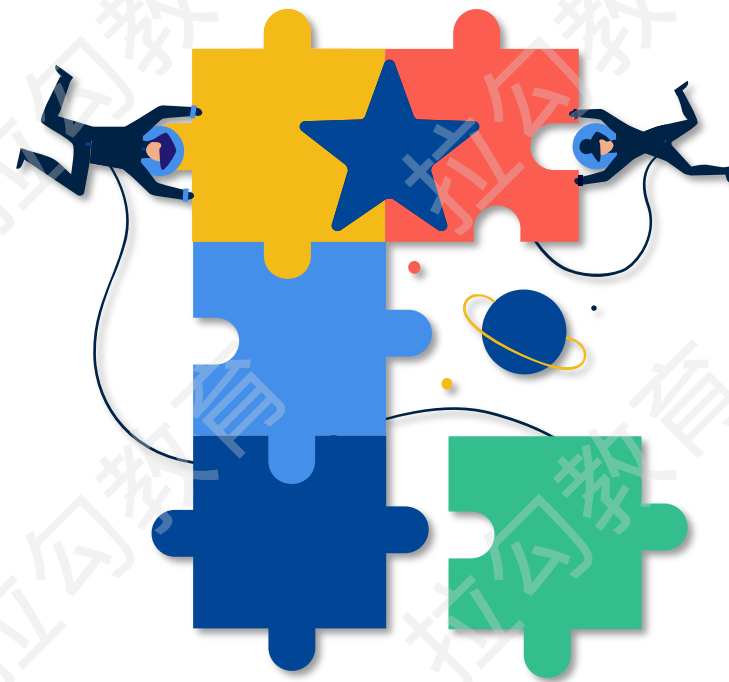
拉勾教育

— 互联网人实战大学 —

GET /getUsers
POST /deleteUser
POST /createUser

RPC 这种设计规范对前端工程师而言是**不够友好**的

- **紧耦合**：当前端工程师需要获取或修改某个数据时
他有可能需要先调用接口 A，再调用接口 B
这种调用需要对系统非常熟悉
让前端工程师熟悉后端逻辑和代码显然是难以办到的
- **冗余**：把执行动作写在 URL 上实际是冗余的
因为 HTTP 的 Method 头部可以表示不同的动作行为



REST—表现层状态转换

拉勾教育

— 互联网人实战大学 —

REST (Representational State Transfer) 即表现层状态转换

什么是“表现层”？

拉勾教育

— 互联网人实战大学 —

在理解“**表现层**”之前，我们先理解另一个概念“**资源**”

资源指的是一个实体信息

一个文本文件、一段 JSON 数据都可以称为资源

而一个资源可以有不同的呈现形式

比如一份数据可以是 XML 格式，也可以是 JSON 格式

这种呈现形式叫做“**表现层 (Representation)**”



什么又是“状态转移”？

拉勾教育

— 互联网人实战大学 —

HTTP 本身是无状态的，如果客户端想要操作服务器

则必须通过某种手段让服务器发生“状态转移（State Transfer）”

而这种转移是建立在表现层之上的，即“**表现层状态转移**”



什么又是“状态转移”？

拉勾教育

— 互联网人实战大学 —

/orgs

/orgs/123asdf12d

/orgs/ss1212sdf/users

/orgs/ss1212sdf/users/111asdl234l

什么又是“状态转移”？

拉勾教育

— 互联网人实战大学 —

/createUser
/samples/export

什么又是“状态转移”？

拉勾教育

— 互联网人实战大学 —

GET (SELECT) : 获取资源

POST (CREATE) : 新建一个资源

PUT (UPDATE) : 更新资源

DELETE (DELETE) : 从服务器删除资源

什么又是“状态转移”？

拉勾教育

— 互联网人实战大学 —

- 弱约束

REST 定义请求路径和方法

但对非常重要的请求体和响应体并没有给出规范和约束

这就意味着需要借助工具来重新定义和校验这些内容

而不同工具之间的定义格式和校验方式都不相同

给工程师带来了一定的学习负担



什么又是“状态转移”？

拉勾教育

— 互联网人实战大学 —

- 接口松散

REST 风格的数据粒度一般都非常小

前端要进行复杂查询的时候可能会涉及多个 API 查询

那么会产生多个网络请求，很容易造成性能问题

通常的解决方案是

通过类似 API 网关的中转服务器来实现对接口的聚合和缓存



什么又是“状态转移”？

拉勾教育

— 互联网人实战大学 —

- 数据冗余

前端对网络请求性能是比较敏感的，所以传输的数据量尽可能小

但 REST API 在设计好之后，返回的字段值是固定的

所以很容易出现这样一个场景

对于后端工程师而言

为了减少代码修改，会尽可能地在返回结果中添加更多的字段

对于前端工程师而言

使用数据的场景往往是多变的

即使是调用同一个 API，在不同场景下也只会用到某些特定的字段

所以不可避免地产生数据冗余，从而造成带宽浪费，影响用户体验



将关注点从资源转移到 API 的调用者上

从调用者的角度来思考 API 设计

对于调用者而言最关心的不是资源和方法

而是响应内容

在前后端的交互中，请求体和响应内容一般都采用 JSON 格式


```
{  
  "id": 1296269,  
  "stargazers_count": 80,  
  "name": "Hello-World",  
  "full_name": "octocat/Hello-World",  
  "owner": {  
    "login": "octocat",  
    "id": 1?  
  },  
  "avatar_url": "https://github.com/images/error/octocat_happy.gif"  
}
```

```
{  
  "type": "object",  
  "properties": {  
    "id": {  
      "name": "id",  
      "type": "number"  
    },  
    "stargazers_count": {  
      "name": "stargazers_count",  
      "type": "number"  
    },  
    "name": {
```

```
"name": "name",  
"type": "string",  
},  
"full_name": {  
  "name": "full_name",  
  "type": "string",  
},  
"owner": {  
  "name": "owner",  
  "type": "object",  
  "properties": {  
    "login": {
```

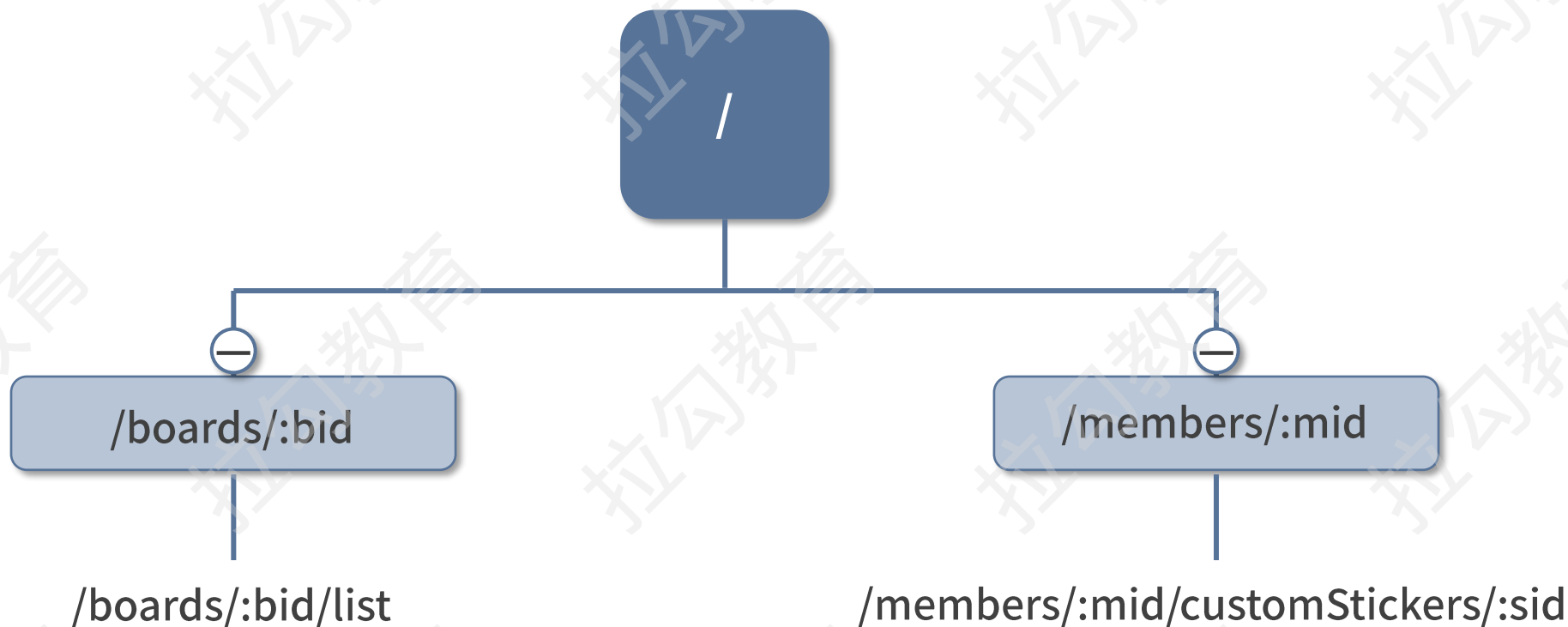
```
"name": "login",  
"type": "string"  
},  
"id": {  
  "name": "id",  
  "type": "number"  
},  
"avatar_url": {  
  "name": "avatar_url",  
  "type": "string"  
}  
},
```

```
"required": [
  "login",
  "id",
  "avatar_url"
],
"required": [
  "id",
  "stargazers_count",
  "name",
  "full_name"
```

```
      "avatar_url"  
    ]  
  }  
}  
  
"required": [  
  "id",  
  "stargazers_count",  
  "name",  
  "full_name",  
  "owner"  
]
```

```
{  
  "id",  
  "stargazers_count",  
  "name",  
  "full_name",  
  "owner": {  
    "login",  
    "id"?  
    "avatar_url"  
  }  
}
```

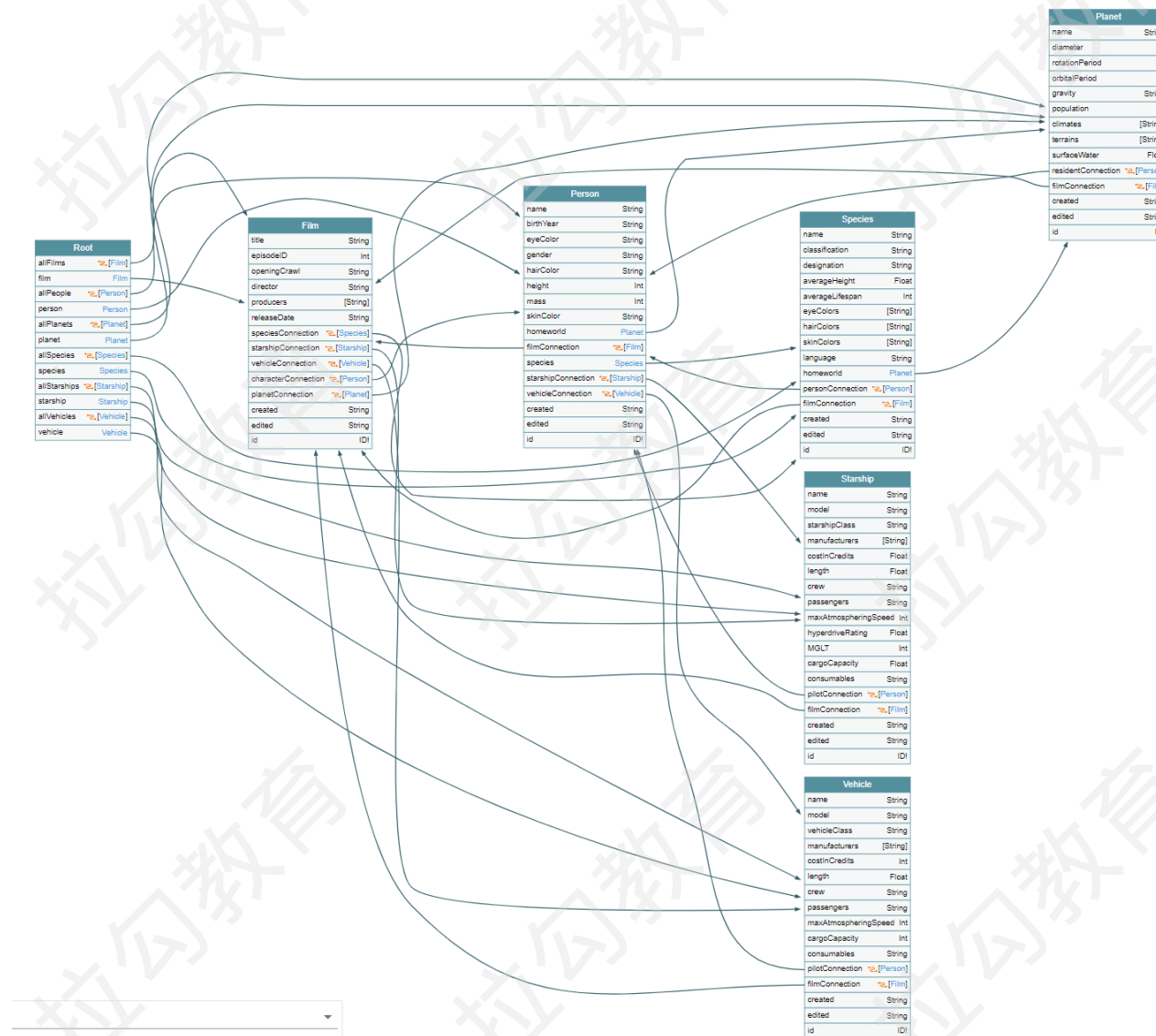
```
{  
  id  
  stargazers_count  
  name  
  full_name  
  owner {  
    login  
    id  
    avatar_url  
  }  
}
```

GraphQL—图表查询语言

拉勾教育

— 互联网人实战大学 —



hahet ☒ Skin Relay ☒ Skin deprecated ☒ Show leaf fields

L / A / G / O / U

GraphQL—图表查询语言

拉勾教育

— 互联网人实战大学 —

GraphQL 的查询语句提供了 3 种操作

查询 (Query)

变更 (Mutation)

订阅 (Subscription)



别名 (Aliases)

拉勾教育

— 互联网人实战大学 —

别名看上去是一个锦上添花的功能，但在开发中也会起到非常重要的作用

前端通过请求 GET /user/:uid 获取一个关于用户信息的 JSON 对象

并使用了返回结果中的 name 字段

如果后端调整了接口数据，将 name 字段改成了 username

那么对于前端来说只能被动地修改代码

而如果使用 **GraphQL**只需要修改查询的别名即可



别名 (Aliases)

拉勾教育

— 互联网人实战大学 —



The screenshot shows the GraphQL IDE interface. The top bar contains the 'GraphiQL' logo, a play button, and buttons for 'Prettify', 'History', and 'Explorer'. The main area is split into two panels. The left panel displays a GraphQL query with line numbers 1 through 9. The query defines a 'repository' field with an alias 'createdTime' for the 'createdAt' field. The right panel shows the JSON response, which includes the 'data' object with a 'repository' field containing the 'createdAt' and 'createdTime' values.

```
1 {  
2   repository(owner: "yalishizhude", name: "rxwx") {  
3     # 原生字段  
4     createdAt,  
5     # 别名  
6     createdTime: createdAt  
7   }  
8 }  
9
```

```
{  
  "data": {  
    "repository": {  
      "createdAt": "2017-09-20T13:24:24Z",  
      "createdTime": "2017-09-20T13:24:24Z"  
    }  
  }  
}
```

片段 (Fragments)

拉勾教育

— 互联网人实战大学 —

```
1 {
2   repository(owner: "yalishihude", name: "rxwx") {
3     firstUser: stargazers(first: 1) {
4       edges{
5         ...edge
6       }
7     }
8     lastUser: stargazers(last: 1) {
9       edges{
10        ...edge
11      }
12    }
13  }
14 }
15 fragment edge on StargazerEdge {
16   node {
17     id
18     name
19   }
20 }
21
```

```
{
  "data": {
    "repository": {
      "firstUser": {
        "edges": [
          {
            "node": {
              "id": "MDQ6VXNlcjEwMTIzMTY3",
              "name": "MarvinWilliam"
            }
          }
        ]
      },
      "lastUser": {
        "edges": [
          {
            "node": {
              "id": "MDQ6VXNlcjQ1MDIzMDQy",
              "name": "Real Liu"
            }
          }
        ]
      }
    }
  }
}
```

内省 (Introspection)



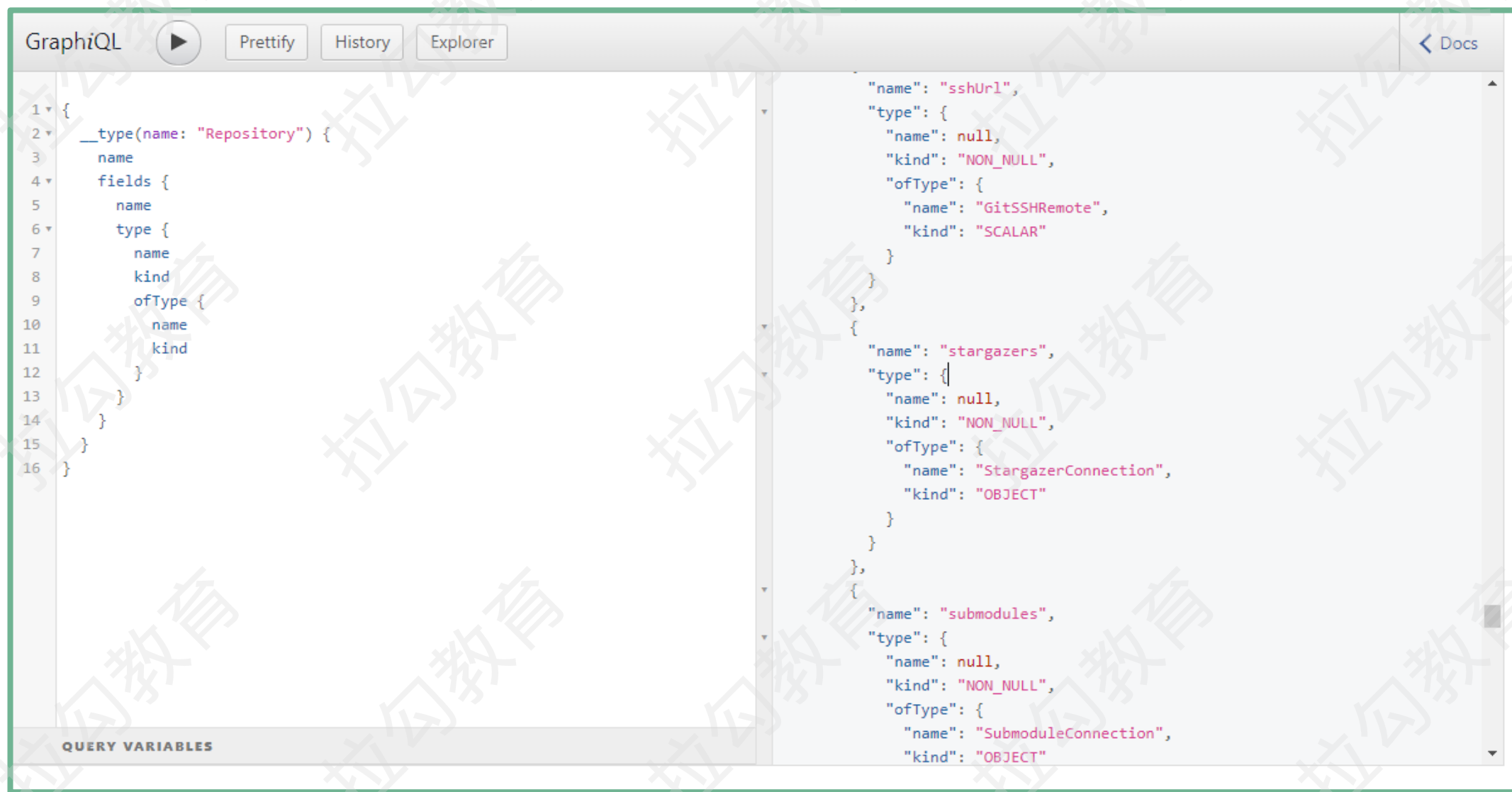
内省 (Introspection)

拉勾教育

— 互联网人实战大学 —

```
{  
  "name": "Repository",  
  "description": "A repository contains the content for a project."  
},
```

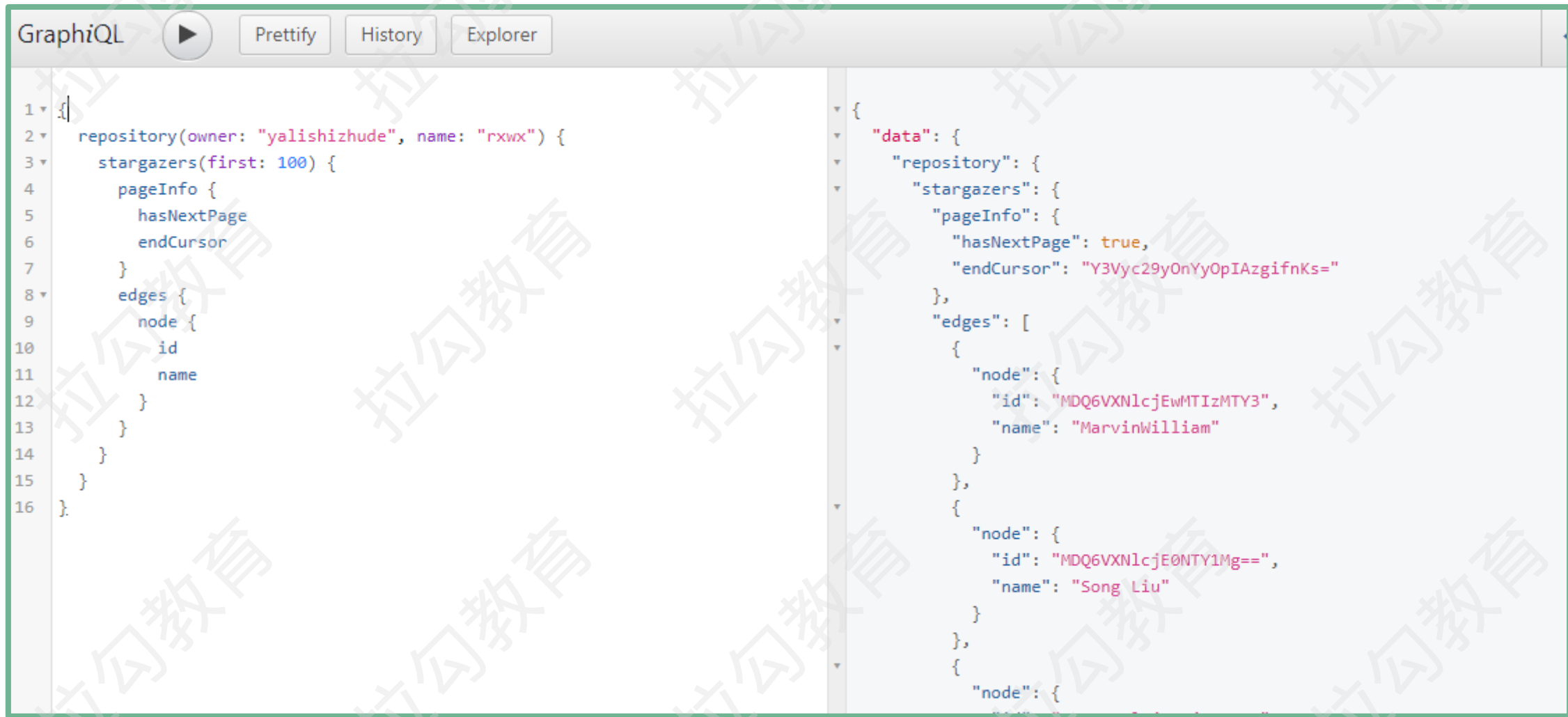

内省 (Introspection)



内省 (Introspection)

拉勾教育

— 互联网人实战大学 —



The screenshot shows the GraphQL IDE interface with a query on the left and its JSON response on the right.

Query (Left Panel):

```
1 {
2   repository(owner: "yalishizhude", name: "rxwx") {
3     stargazers(first: 100) {
4       pageInfo {
5         hasNextPage
6         endCursor
7       }
8       edges {
9         node {
10          id
11          name
12        }
13      }
14    }
15  }
16 }
```

Response (Right Panel):

```
{
  "data": {
    "repository": {
      "stargazers": {
        "pageInfo": {
          "hasNextPage": true,
          "endCursor": "Y3Vyc29yOnYyOpIAzgifnKs="
        },
        "edges": [
          {
            "node": {
              "id": "MDQ6VXNlcjEwMTIzMTY3",
              "name": "MarvinWilliam"
            }
          },
          {
            "node": {
              "id": "MDQ6VXNlcjE0NTY1Mg==",
              "name": "Song Liu"
            }
          },
          {
            "node": {
              "id": "MDQ6VXNlcjE0NTY1Mg==",
              "name": "Song Liu"
            }
          }
        ]
      }
    }
  }
}
```

内省 (Introspection)

拉勾教育

— 互联网人实战大学 —

```
type User{  
  id: ID!  
  name: String!  
  books: [Book!]!  
}
```

内省 (Introspection)

拉勾教育

— 互联网人实战大学 —

```
union Owner = User | Organization
```

```
interface Member {
```

```
  id: ID!
```

```
  name: String
```

```
type User implements Member {
```

```
  ...
```

```
}
```

内省 (Introspection)

拉勾教育

— 互联网人实战大学 —

```
name: String
```

```
}
```

```
type User implements Member {
```

```
}
```

```
type Organization implements Member {
```

```
...
```

内省 (Introspection)

拉勾教育

— 互联网人实战大学 —

```
const schemaStr = `
type Hero {
  id: String!
  name: String!
}

# 根类型
type Query {
  hero(id: String!, name: String!): [Hero]
}
```

内省 (Introspection)

拉勾教育

— 互联网人实战大学 —

```
const resolver = {  
  hero({id='hello', name='world'}) {  
    if(id && name) {  
      return [...data.hero, {id, name}]  
    }  
    return data.hero  
  }  
}
```

内省 (Introspection)

拉勾教育

— 互联网人实战大学 —

- 高聚合

GraphQL 提倡将系统所有请求路径都聚合在一起形成一个统一的地址
并使用 POST 方法来提交查询语句

比如 GitHub 使用的请求地址就是: <https://api.github.com/graphql>

- 无冗余

后端会根据查询语句来返回值, 不会出现冗余字段



内省 (Introspection)

拉勾教育

— 互联网人实战大学 —

- 类型校验

由于有模式的存在，可以轻松实现对响应结果及查询语句进行校验

- 代码即文档

内省功能可以直接查询模式

无需查询文档也可以通过命名及描述信息来进行查询



内省 (Introspection)

拉勾教育

— 互联网人实战大学 —

GraphQL 提供了一种基于特定语言的查询模式
让前端可以随心所欲地获得想要的数据类型，是相当友好的
而对于后端而言
把数据的查询结果编写成 REST API 还是 GraphQL 的解析器
工作量相差不大
最大的问题是**带来的收益可能无法抵消学习和改造成本**
这在很大程度上增加了 GraphQL 的推广难度



内省 (Introspection)

拉勾教育

— 互联网人实战大学 —

一类是**前端工程师主导的新项目**

后端采用 Node.js 来实现

用 GraphQL 来替代 REST

另一类就是**将 Node.js 服务器作为中转服务器**

为前端提供一个 GraphQL 查询

但实际上仍然是调用后端的 REST API 来获取数据



总结

拉勾教育

— 互联网人实战大学 —

从 **RPC** 到 **REST** 再到 **GraphQL**

可以看到 API 规范上的一些明显变化

- 关注点发生了明显的转移
- 语义化的特性更加明显
- 带来的副作用和约束更多，实现起来更加复杂



总结

拉勾教育

— 互联网人实战大学 —

- 对于 **RPC 风格**，了解即可
- 对于 **REST API** 需要重点理解它通过路径指向资源
以及利用 HTTP 方法来指代动作的特性
- 对于 **GraphQL**
应该从 API 调用者和 API 提供者两个角度来分别学习查询语句和模式



你还能找到常见的不符合 REST 规范的例子吗



Next：第18讲：《你是怎么理解“组件”这个概念的？》

拉勾教育

— 互联网人实战大学 —



下载「拉勾教育App」
获取更多内容