Comparison between GraphQL and REST

Wanyi Liu (wl49)

With the rise of server-side SOA and client Ajax, communication extensions become more and more important. Therefore, REST has been widely used and RESTful API has become the mainstream. However, REST also faces severe challenges with the rapid development of front-end and mobile applications. A large number of concurrent requests and secondary requests for supplementary data have caused a lot of trouble for mobile application, especially on the response time. In the process of finding a better solution, Facebook engineers stopped to preconceive the data as RESTful set, and considered the relationship between the data. Therefore, GraphQL came into being.

I. Introduction

1. REST

Representational State Transfer (REST) describes an architecture-style network system, usually a web application. It was first proposed by Roy Thomas in his doctoral thesis. Compares with SOAP and XML-RPC, Rest tend to provide people a more simple and lightweight way on URL processing and Payload encoding. Rest is a design style rather than a protocol, discussing how to use web standard correctly, such as HTTP and URI. Any applications designed to fulfill REST principles can be called "RESTful web service", also known as "RESTful Web API".

REST use Caching for HTTP to avoid re-fetching resources and identify if two resources are the same. [1] Caching save some performance problems for REST. REST does not have a real problem on scalability as a kind of resource modeling. However, there are some problems cannot be solved using REST, such as performance optimization and resource classification on page display.

2. GraphQL

GraphQL is not a query database for graph database, but a data abstraction layer, including data format, data association, query definition and implementation of a package of things. GraphQL is not a specific back-end programming framework. If the REST can be treat as a simple logic for the query style, then GraphQL can be treated as a separate abstraction layer, providing a combination of more complex and varied way of query through a number of REST-style simple interfaces.

2.1. Query, Mutation and Execution

Compared with REST, GraphQL defines a more rigorous, scalable, maintainable data query.
A GraphQL query looks like following:

```
{
    user(id: 400) {
        id,
        name
    }
}
```

And the result may be as following:

```
{
    "user": {
        "id": 400,
        "name": "Wanyi"
    }
}
```

The query in GraphQL has exactly the same shape as the result. And GraphQL use mutation to update data in the databases. When we execute a query, each field on each type is returned by a function called the resolver defined on the server. If a field produces a scalar value, then the execution will complete. If a field produces an object then the query continues fetch the fields in the object until scalar values are reached. In the server, there will be a Root type that represents all possible entry points into the API. [1]

2.2. Type system and Introspection

GraphQL has a type system. Every service should define a set of types which describe the data clients can query on that service. With the type system, we can predetermine whether a query is valid or not without having runtime checks, by calling validation function. If the clients don't know what types are available on the server, we can use introspection to access the documentation about the type system. The __schema field in the introspection system can return all types in the server, including types defined by user, the built-in scalar types, and the types to support the introspection system. [4]

II. Comparison

1. GraphQL request complex data at once

The efficiency problem is usually caused by HTTP request. It is because of that the

cost of HTTP request is very high, especially on the mobile end. Using RESTful API, people need to classify the various categories of resources into different APIs. In a complex application, we usually have to request a variety of resources so that we need to merge many APIs for different pages. Additionally, if we choose not to merge resources, the performance loss may become even more serious. It is because of that the resources to be merged are often related to each other. For example, people usually have to ask for user information first, and then render the articles after the API returning the user information. In this case, the requests are serial processed instead of parallel processed, which will double our HTTP request time.

Using the query format in GraphQL, we can ask for a variety of resources at one time, which will save time on HTTP request. For example, we can use the query given below to require the profile, follower and article data of a user in one query:

```
{
    user(id: 400) {
        id,
        profile {
            email,
            zipcode,
            dob,
            follower{
                name
            }
        },
        article {
            _id,
            text,
            author
        }
    }
}
```

The meaning of GraphQL, as the name suggests, is a graph query language. Different from the usual request, the realization of GraphQL server is to receive the request, although the HTTP or a request, and to call the various resources resolver according to the structure of the query recursively. And finally the query returns a JSON Graph back to the client. So you can easily express a complex data relationship in one query.

2. Static type system and un-typed API

The RESTful API expresses the resource through the URL, which is un-typed. With the development of technology, we already have a lot of very powerful static type language, they have a very powerful development tools to help us check the error. However, these powerful tools are powerless on the API of our system. [4]

Different from REST, GraphQL is a static type system. The GraphQL queries have automatic completion and type error corrections. And people can add more types to describe different resources when defining GraphQL Schema. Developers can see signatures and descriptions of different types in the “Docs” panel of GraphQL. This "Docs" does not have to be manually written. It is automatically generated based on the server code. The information in this panel is from the GraphQL query itself, which is called Introspection by Facebook.

3. GraphQL gives more freedom in client development

Using REST, it is difficult to make the RESTful API so generic that the API can be used by anything applications, such as iOS app, Android app and web app. Some basic solutions are creating custom endpoints, creating custom representations, or using custom APIs.

However, using GraphQL allows client developers to write the queries in any language as they like, without asking the API provider to rewrite the API in different languages.

4. Security issues in GraphQL

Although GraphQL provides a powerful query capability to the client, it also means that there is a risk of being misused by the client. Repeating simple request which load all Github users may make their servers directly hang. GraphQL increased the risk of being DDoS. Therefore, in the use of GraphQL process, we have to pay more attention to security issues. [4]

5. Need new cache strategy

REST does cause some performance problems, but it also uses HTTP Cache to solve a lot of performance problems. For GraphQL operation, Cache strategy may need to be completed in client application; otherwise it may need to reload data from databases repeatedly.

III. Conclusion

It is difficult to say which one is better, but we may need to use them in different situations. However, GraphQL would be a better choice if the developers need an API with high query capability. Sometimes, there will different clients which request many tiny and different data. GraphQL can make the cost of restructure the data for queries become lower. And people can even combine GraphQL and REST, by adding one

more endpoint for GraphQL in the RESTful APIs. [2]

IV. References

[1] http://graphql.org/
[2] https://philsturgeon.uk/api/2017/01/24/graphql-vs-rest-overview/
[3] https://blog.startifact.com/posts/graphql-and-rest.html
[4] https://baijiahao.baidu.com/po/feed/share?wfr=spider&for=pc&context=%7B%22sourceFrom%22%3A%22bjh%22%2C%22nid%22%3A%22news_4115722374514428500%22%7D