(/)

Expose GraphQL Field with Different Name

Last modified: October 7, 2022

Written by: baeldung (https://www.baeldung.com/author/baeldung)

Web Services (https://www.baeldung.com/category/web-services)

GraphQL (https://www.baeldung.com/tag/graphql)

1. Overview

GraphQL (/graphql) has been widely used as a pattern of communication in web services. **The basic premise of GraphQL is to be flexible in use by client-side applications.**

In this tutorial, we'll look into another aspect of flexibility. We'll also explore how a GraphQL field can be exposed with a different name.

2. GraphQL Schema

Let us take an example of a blog (/spring-graphql) having *Posts* by different *Authors*. The GraphQL schema looks something like this:

```
query {
    recentPosts(count: 1, offset: 0){
        id
        title
        text
        category
        author {
            id
            name
            thumbnail
        }
    }
}

type Post {
    id: ID!
    title: String!
    text: String!
    category: String
    authorId: Author!
}

type Author {
    id: ID!
    name: String!
    thumbnail: String
    posts: [Post]!
}
```

Here we can fetch recent posts. **Every *post* will be accompanied by its *author*.** The result of the query is as follows:

```
{
    "data": {
        "recentPosts": [
            {
                "id": "Post00",
                "title": "Post 0:0",
                "text": "Post 0 + by author 0",
                "category": null,
                "author": {
                    "id": "Author0",
                    "name": "Author 0",
                    "thumbnail": "http://example.com/authors/0"
                }
            }
        ]
    }
}
```


3. Exposing GraphQL Field with a Different Name

A client-side application may require to use of the field *first_author*. Right now, it's using the *author*. To accommodate this requirement, we have two solutions:

- Change the definition of the schema in the GraphQL server

- Make use of the concept of **Aliases (https://graphql.org/learn/queries/#aliases)** in GraphQL

Let's look at both one by one.

3.1. Changing Schema

Let's update the schema definition of the *post*:

```
type Post {
    id: ID!
    title: String!
    text: String!
    category: String
    first_author: Author!
}
```

The *author* isn't a trivial field. It's a complex one. We'll also have to update the handler method to accommodate this change.

The method a*uthor(Post post), marked with @SchemaMapping* in *PostController,* will need to be updated to *getFirst_author(Post post)*. Alternatively, the *field* attribute has to be added in the @SchemaMapping to reflect the new field name.

Here's the query:

```
query{
    recentPosts(count: 1,offset: 0){
        id
        title
        text
        category
        first_author{
            id
            name
            thumbnail
        }
    }
}
```

The result of the above query is as follows:

```
{
    "data": {
        "recentPosts": [
            {
                "id": "Post00",
                "title": "Post 0:0",
                "text": "Post 0 + by author 0",
                "category": null,
                "first_author": {
                    "id": "Author0",
                    "name": "Author 0",
                    "thumbnail": "http://example.com/authors/0"
                }
            }
        ]
    }
}
```

This solution has two main issues:

- It is introducing changes to the schema and server-side implementation
- It is forcing other client-side applications to follow this updated schema definition

These issues contradict the flexibility feature GraphQL offers.

3.2. GraphQL Aliases

Aliases, in GraphQL, let us rename the result of a field to anything we want without changing the schema definition. To introduce an alias in a query, the alias and colon symbol (:) have to precede the GraphQL field.

Here's the demonstration of the query:

```
query {
    recentPosts(count: 1,offset: 0) {
        id
        title
        text
        category
        first_author:author {
            id
            name
            thumbnail
        }
    }
}
```

The result of the above query is as follows:

freestar.com
paign=branding&
n=stickyFooter&
baeldung.com&
aeldung_adhesion)

```
{
    "data": {
        "recentPosts": [
            {
                "id": "Post00",
                "title": "Post 0:0",
                "text": "Post 0 + by author 0",
                "category": null,
                "first_author": {
                    "id": "Author0",
                    "name": "Author 0",
                    "thumbnail": "http://example.com/authors/0"
                }
            }
        ]
    }
}
```

Let's notice that the query itself is requesting the first post. Another client-side application may request to have *first_post* instead of *recentPosts.* Again, Aliases will come to the rescue.

```
query {
    first_post: recentPosts(count: 1,offset: 0) {
        id
        title
        text
        category
        author {
            id
            name
            thumbnail
        }
    }
}
```

The result of the above query is as follows:

```
{
    "data": {
        "first_post": [
            {
                "id": "Post00",
                "title": "Post 0:0",
                "text": "Post 0 + by author 0",
                "category": null,
                "author": {
                    "id": "Author0",
                    "name": "Author 0",
                    "thumbnail": "http://example.com/authors/0"
                }
            }
        ]
    }
}
```

These two examples clearly show how flexible it is to work with GraphQL. Every client-side application can update itself according to the requirement. Meanwhile, the server-side schema definition and implementation stay the same.

4. Conclusion

In this article, we've looked into two ways of exposing a graphQL field with a different name. We've introduced the concept of Aliases with examples and explained how it is the right approach.

As always, the example code for this article is available over on GitHub (https://github.com/eugenp/tutorials/tree/master/spring-boot-modules/spring-boot-graphql).

COURSES

>> CHECK OUT THE COURSE (/ls-course-end)

ALL COURSES (/ALL-COURSES)

ALL BULK COURSES (/ALL-BULK-COURSES)

SPRING REACTIVE TUTORIALS (/SPRING-REACTIVE-GUIDE)

Learning to build your API **with Spring**?

Download the E-book (/rest-api-spring-guide)

ABOUT

ABOUT BAELDUNG (/ABOUT)

THE FULL ARCHIVE (/FULL_ARCHIVE)

EDITORS (/EDITORS)

JOBS (/TAG/ACTIVE-JOB/)

OUR PARTNERS (/PARTNERS)

PARTNER WITH BAELDUNG (/ADVERTISE)

Comments are closed on this article!

TERMS OF SERVICE (/TERMS-OF-SERVICE)

PRIVACY POLICY (/PRIVACY-POLICY)

COMPANY INFO (/BAELDUNG-COMPANY-INFO)

CONTACT (/CONTACT)

freestar.com
paign=branding&
n=stickyFooter&
baeldung.com&
aeldung_adhesion)