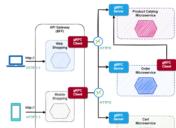
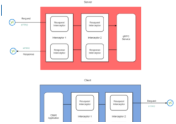
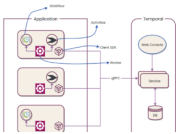


Home > Microservices > GraphQL

Spring for GraphQL: Mutation



by **Pankaj** — August 14, 2022 in **GraphQL** Reading Time: 13 mins read



Recent Articles

gRPC Bidirectional Example
© FEBRUARY 17, 2023

Spring for GraphQL mutation

0

SHARES

2.1k

VIEWS

 Share on Facebook

 Share on Twitter

 Share on LinkedIn

In earlier articles, we talked about GraphQL queries. But, GraphQL is not just about queries.

Like any other API platform, a GraphQL service also needs to provide a way to manipulate data. This is where GraphQL mutation comes into the picture.

In theory, we can use GraphQL queries to modify the state but this will confuse our API consumers.

In REST, we use GET request to retrieve data, and by convention, it should not cause any side-effect. Similarly, we should **not use GraphQL queries to modify the application's state**.

In this article

- [Read more about GraphQL...](#)
- [What is GraphQL Mutation?](#)
 - [GraphQL Mutation Input Type](#)
- [Implementing Mutations in Spring](#)
- [Implementing Mutation with Input Type](#)
- [Code Example](#)
- [Conclusion](#)

Therefore, if we need to provide **APIs that change state, we should use mutation instead**.

[Read more about GraphQL...](#)

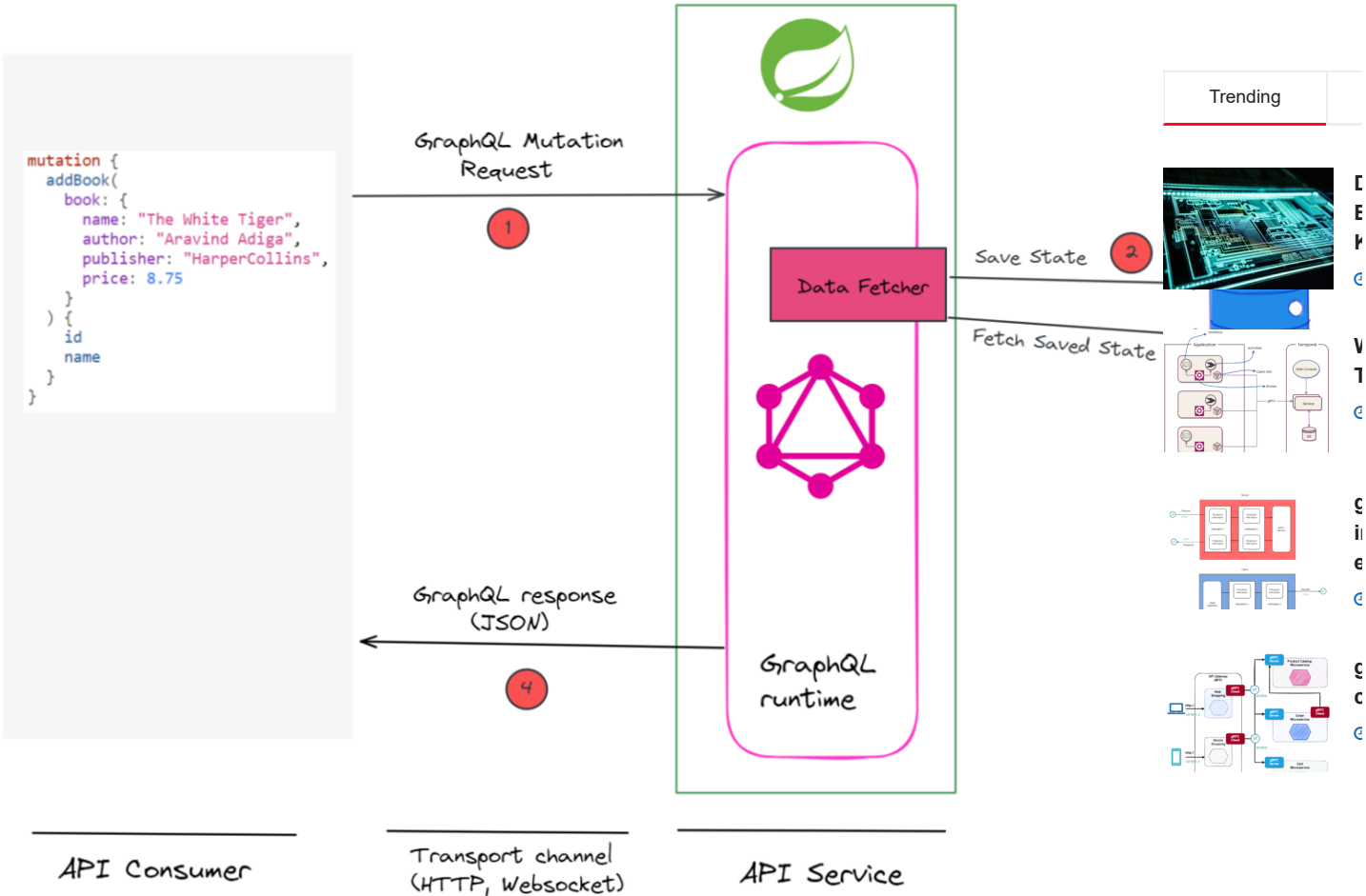
What is GraphQL Mutation?

In GraphQL, a **mutation** is used to **insert, update or delete** data. The mutation API is defined with the type **Mutation** rather than the **Query**.

Recent Articles

gRPC Bidirection:
Example

© FEBRUARY 17, 2023



GraphQL Mutation

An example of a mutation that adds a book to the book catalog.

```
type Mutation {
  addBook(name: String!, author: String!, publisher: String!,price: Float!): String!
}
```

In the above example, `addBook` API is a `mutation` ; it allows you to add/save a book and returns the ID of the book successful save.

gRPC Bidirection: Example
© FEBRUARY 17, 2023

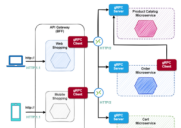
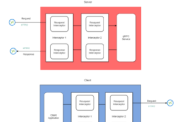
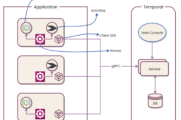
`!` means a required field.

Similarly, we can define API to *update* the book as:

```
type Mutation {
  updateBook(id: ID!, name: String, author: String, publisher: String, price: Float): String!
}
```

Trending

We can even design APIs to return info about added, updated, or deleted books as:



```
type Mutation {
  addBook(name: String!, author: String!, publisher: String!,price: Float!): BookInfo!
  updateBook(id: ID!, name: String, author: String, publisher: String,price: Float): BookInfo!
  deleteBook(id: ID!): BookInfo!
}
```

Where type `BookInfo` is defined as:

```
type BookInfo {
  id : ID
  name : String
  author: String
  publisher: String
  price: Float
}
```

Recent Articles

In GraphQL, you can add a book using mutation as:

GraphiQL ▶ Prettify Merge Copy History Introspect

```
1 mutation {
2   addBook(
3     name: "Sapiens: A Brief History of Humankind"
4     author: "Yuval Noah Harari"
5     publisher: "Random House Uk"
6     price: 18.75
7   ) {
8     id
9     name
10  }
11 }
```

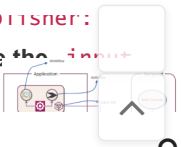
```
{
  "data": {
    "addBook": {
      "id": "6",
      "name": "Sapiens:
    }
  }
}
```

gRPC Bidirection:
Example
© FEBRUARY 17, 2023

Add Book Mutation

GraphQL Mutation Input Type

Instead of defining the API with scalar arguments, for example – `addBook(name: String!, author: String!, publisher: String!,price: Float!)` , you can define a complex object called `input` type. This is **useful if you want to reuse the input type for both updates and inserts.**

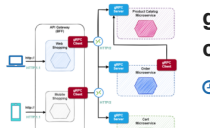
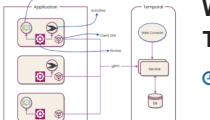
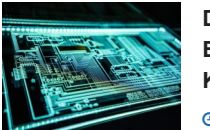


An input type is defined with a keyword `input` instead of `type` as:



- 3 Easy Steps**
- 1. Click 'Start'
 - 2. Activate your account
 - 3. Access your content

Trending



```
input BookInput {
  name : String
  author: String
  publisher: String
  price: Float
}
```

As a result, we can change the API definition as:

```
type Mutation {
  addBook(book: BookInput!): BookInfo!
  updateBook(id: ID!, book: BookInput!): BookInfo!
  deleteBook(id: ID!): BookInfo!
}
```

And, in GraphiQL (or other clients), we can call API as:

GraphiQL

▶ Prettify Merge Copy History Introspect

```
1 mutation {
2   addBook(
3     book: {
4       name: "Sapiens: A Brief History of Humankind",
5       author: "Yuval Noah Harari",
6       publisher: "Random House Uk",
7       price: 18.75
8     }
9   ) {
10    id
11    name
12  }
13 }
```

```
{
  "data": {
    "addBook": {
      "id": "6",
      "name": "Sapiens: A Brie
    }
  }
}
```

Recent Articles

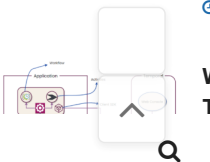
gRPC Bidirection:
Example

© FEBRUARY 17, 2023

Implementing Mutations in Spring

In Spring for GraphQL, we can implement mutation using `@SchemaMapping` or `@MutationMapping` .

If we have defined addBook API as:



```

}
```

Then, we can implement mutation by defining `@SchemaMapping` with `typeName` as `Mutation` as:

```

@SchemaMapping(typeName = "Mutation", field = "addBook")
public String addBook(
    @Argument String author,
    @Argument String name,
    @Argument String publisher,
    @Argument Double price) {
    log.info("Saving book, name {}", name);
    var book = new BookInput(name, author, publisher, price);
    return bookCatalogService.saveBook(book);
}
```

Generally, we can leave the parameter `field` if the method name is the same as the field, as:

```

@SchemaMapping(typeName = "Mutation")
public String addBook(
    @Argument String author,
    @Argument String name,
    @Argument String publisher,
    @Argument Double price) {
    log.info("Saving book, name {}", name);
    var book = new BookInput(name, author, publisher, price);
    return bookCatalogService.saveBook(book);
}
```

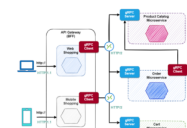
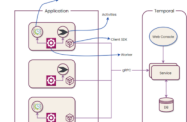
Similar to `@QueryMapping`, Spring also provides a shorthand annotation as `@MutationMapping`. As a result, we can implement `addBook` mutation as:

```

@MutationMapping
public String addBook(
    @Argument String author,
    @Argument String name,
    @Argument String publisher,
    @Argument Double price) {
    log.info("Saving book, name {}", name);
    var book = new BookInput(name, author, publisher, price);
    return bookCatalogService.saveBook(book);
}
```

As always, we can test the API in GraphiQL at <http://localhost:8080/graphiql?path=/graphql>

Trending



Recent Articles

gRPC Bidirectional
Example

© FEBRUARY 17, 2023



GraphiQL

Prettify

Merge

Copy

History

Introspect

1

mutation {

2

addBook({

3

name: "Sapiens: A Brief History of Humankind"

4

author: "Yuval Noah Harari"

5

publisher: "Random House Uk"

6

price: 18.75

7

}) {

8

id

9

name

10

}

11

}

{

"data": {

"addBook": {

"id": "6",


"name": "Sapiens:

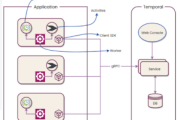
}

}

}

Trending





Implementing Mutation with Input Type

If a mutation is defined as

Previous Post

Next Post

Spring for GraphQL: How to solve the N+1 Problem?

type Mutation {

addBook(book: BookInput!): BookInfo!

}

GraphQL Directive



Pankaj

Then, we can implement mutation by defining @MutationMapping and @Argument as

Software Engineer @ Sonar

Java | Python

gRPC Bidirection: Example

© FEBRUARY 17, 2023

```
@MutationMapping
public BookInfo addBook(@Argument BookInput book) {
    log.info("Saving book, name {}", book.name());
    return bookCatalogService.saveBook(book);
}
```

Related Posts

GRAPHQL

GRAPHQL

GRAPHQL

Conclusion

GraphQL Directive

SEPTEMBER 29, 2022

GraphQL mutation is used to change the application's state (insert, update and delete). In Spring for GraphQL, a mutation is implemented by defining a handler method and using annotation @MutationMapping or @SchemaMapping with the typeName as Mutation .

Tags: [graphql](#) [microservices](#) [spring-boot](#)

MICROSERVICES ▾ SPRING BOOT ▾ gRPC ▾ GRAPHQL ▾ KUBERNETES ▾ JAVA ▾

Spring for GraphQL: How to solve the N+1 Problem?

AUGUST 7, 2022

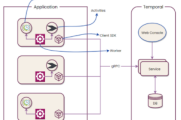
Spring for GraphQL : @SchemaMapping and @QueryMapping

JULY 29, 2022

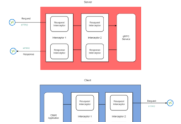
Trending



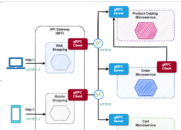
E
E
P
e



V
T
e



G
i
e
e



G
C
e

Recent Articles

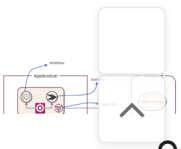
gRPC Bidirection:
Example

© FEBRUARY 17, 2023

G
e

E
M
S
e

V
T
e



Q

GRAPHQL

Getting started with Spring Boot
GraphQL service

© JULY 23, 2022

Discussion about this post

0 Comments

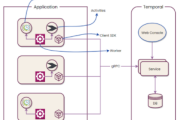


Add a comment...

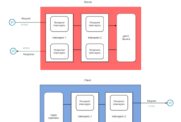
Trending



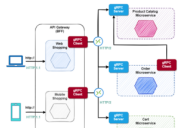
E
E
P
e



V
T
e



G
i
e
e



G
C
e

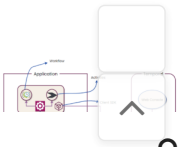
Recent Articles

gRPC Bidirection:
Example

© FEBRUARY 17, 2023

G
e

E
M
S
e

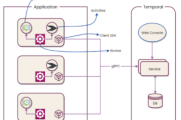


V
T
Q
:

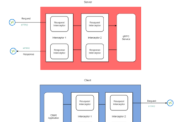
Trending



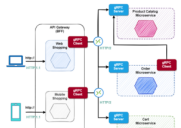
E
E
P
e



V
T
e



G
i
e
e



G
C
e

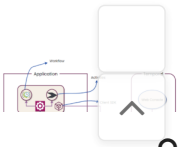
Recent Articles

gRPC Bidirection:
Example

© FEBRUARY 17, 2023

G
e

E
M
S
e

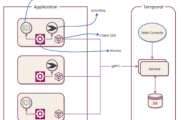


V
T
e

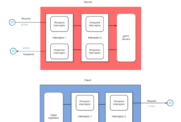
Trending



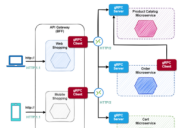
E
E
P
C



V
T
C



C
i
e
C



C
C
C

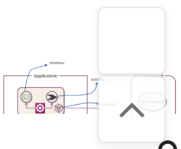
Recent Articles

gRPC Bidirection:
Example

© FEBRUARY 17, 2023

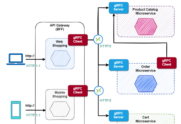
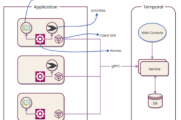
C
C

E
M
S
C



V
T
C

Trending



Recent Articles

f FACEBOOK t TWITTER p PINTEREST

TECHDOZO

Simplifying modern tech stack!

Browse by Category

- Bitesize
GraphQL
gRPC
- Java
Kubernetes
Microservices
- Spring Boot

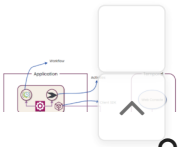
gRPC Bidirectional Streaming with Code Example
© FEBRUARY 17, 2023

Recent Articles

gRPC Bidirectional Streaming with Code Example
© FEBRUARY 17, 2023

gRPC Client Streaming
© JANUARY 20, 2023

MICROSERVICES ▾ SPRING BOOT ▾ gRPC ▾ GRAPHQL ▾ KUBERNETES ▾ JAVA ▾



Trending

© 2023 Techdozo.

