

# TESTING QUERIES IN GRAPHIQL

in this GraphQL tutorial I'll be explaining how we can use that front-end querying tool called Graphiql, to test out our GraphQL server.



app.js

```
const express = require('express');
const { graphqlHTTP } = require('express-graphql');
const schema = require('./schema/schema');

const app = express();

// bind express with graphql
app.use('/graphql', graphqlHTTP({
  schema,
  graphiql: true
}));


app.listen(4000, () => {
  console.log('now listening for requests on port 4000');
});
```

Insert  
code




Go to <http://localhost:4000/graphql>

GraphiQL



PrettifyMergeCopyHistory



< Docs

```
1 | # Welcome to GraphiQL
2 | #
3 | # GraphiQL is an in-browser tool for writing, validating, and
4 | # testing GraphQL queries.
5 | #
6 | # Type queries into this side of the screen, and you will see intelligent
7 | # typeaheads aware of the current GraphQL type schema and live syntax and
8 | # validation errors highlighted within the text.
9 | #
10 | # GraphQL queries typically start with a "{" character. Lines that start
11 | # with a # are ignored.
12 | #
13 | # An example GraphQL query might look like:
14 | #
15 | #   {
16 | #     field(arg: "value") {
17 | #       subField
18 | #     }
19 | #   }
20 | #
21 | # Keyboard shortcuts:
22 | #
23 | #   Prettify Query:  Shift-Ctrl-P (or press the prettify button above)
```

QUERY VARIABLES

Go to <http://localhost:4000/graphql>

GraphiQL

Prettify

Merge

Copy

History

```
1 # Welcome to GraphiQL
2 query
3 # An in-browser tool for writing, validating, and executing GraphQL queries.
4 mutation
5 subscription
6 fragment
7 {
8 # Be aware of the current GraphQL type schema on the right. Errors highlighted within the text.
9 Self descriptive.
10 # GraphQL queries typically start with a "{" character.
11 # Comments starting with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 # {
16 #   field(arg: "value") {
17 #     subField
18 #   }
19 # }
20 #
21 # Keyboard shortcuts:
22 #
23 # Prettify Query: Shift-Ctrl-P (or press the button)
```

QUERY VARIABLES

Documentation Explorer

Search Schema...

A GraphQL schema provides a root type for each kind of operation.

ROOT TYPES

query: RootQueryType

Go to <http://localhost:4000/graphql>

GraphiQL

Prettify

Merge

Copy

History

1 # Welcome to GraphiQL

2 #

3 # GraphiQL is an in-browser tool for writing, validating,

4 # testing GraphQL queries.

5 #

6 # Type queries into this side of the screen, and you'll see them

7 # typeahead as you type, and the current GraphQL type schema

8 # validation errors highlighted within the text.

9 #

10 # GraphQL queries typically start with a "{" character, and queries

11 # with a # are ignored.

12 #

13 # An example GraphQL query might look like:

14 #

15 # {

16 # field(arg: "value") {

17 # subField

18 # }

19 # }

20 #

21 # Keyboard shortcuts:

22 #

23 # Prettify Query: Shift-Ctrl-P (or press the prettify button)

QUERY VARIABLES

< Schema

RootQueryType

×

🔍 Search RootQueryType...


No Description

FIELDS

book(id: String): Book

Go to <http://localhost:4000/graphql>

GraphiQL



Prettify

Merge

Copy

History

```
1 {  
2   book(id: "1"){  
3     name  
4     genre  
5     id  
6   }  
7 }
```

```
{  
  "data": {  
    "book": {  
      "name": "Name of the Wind",  
      "genre": "Fantasy",  
      "id": "1"  
    }  
  }  
}
```

QUERY VARIABLES

< RootQueryType

Book

×

🔍 Search Book...

No Description

FIELDS


id: String

name: String

genre: String

Go to <http://localhost:4000/graphql>

GraphiQL



Prettify

Merge

Copy

History

```
1 {
2   book(id: "1"){
3     name
4     genre
5   }
6 }
```

```
{
  "data": {
    "book": {
      "name": "Name of the Wind",
      "genre": "Fantasy"
    }
  }
}
```

QUERY VARIABLES

< RootQueryType

Book

×

🔍 Search Book...

No Description

FIELDS

id: String

name: String

genre: String