GraphQL

- GraphQL
- Implementation to Spring Boot
- Testing

github.com/gurkanucar

What is the GraphQL?

GraphQL is a **query language** for APIs and is similar to REST. GraphQL is developed by Facebook and is **open source**. GraphQL uses a **single endpoint** for receiving requests. You have to write your **controller name** to your query. And also you can select specific fields for your response. Your **response** will be in **JSON format.**

There are three basic operations of GraphQL (like GET, POST, PUT...)

- Query
- Mutation
- Subscriptions

Query

If you want to get something, you can use “**query**”.

```
query{
  getAllUsers {
    id
    username
    email
    createdAt
    name
  }
}
```

Controller name

Fields for response

```
query{
  getByID(id: 1){
    id
    username
    createdAt
    role
  }
}
```

Controller name

Parameter

Fields for response

github.com/gurkanucar

Mutation

If you want to mutate/change/update your data or create a new one, you can use “**mutation**”.

```
mutation {                                   Controller
                                             name
 createUser(
   user: {username: "grkn", email: "mail@mail.com", role: ADMIN,
name: "Gurkan", surname: "UCAR"}                           Parameters
 ) {
   id                                Fields for
                                     response
   username
  }
}
```

github.com/gurkanucar

Subscription

“**Subscription**” can be used for real-time operations. This operation usually implementing with WebSockets.

```
subscription {
   messages{
     content
     sender {
               name
       }
   }
 }
```

GraphQL Scheme

Schemas include your **models**, **queries**, **mutations** and etc. You have to define all related things in your **.graphqls** file. For instance, you can define a model using **type** keyword.

Some other keywords:

- input : input parameter models
- enum
- scalar : for custom types

Example
`type`

```
type UserDTO {
    id: ID
    username: String
    createdAt: DateTime
    email: String
    name: String
    role: Role
}
```

*For more information check the resources

Example GraphQL Scheme (.graphqls file)

```
scalar DateTime

type Query {
    getAllUsers  : [UserDTO]
    getByID(id:  ID!) : UserDTO
}

type Mutation  {
    createUser(user:  UserCreateRequest):  UserDTO
    updateUser(user:  UserUpdateRequest):  UserDTO
    deleteUser(id:  ID!): Boolean
}

type UserDTO  {
    id: ID
    username:  String
    createdAt: DateTime
    updatedAt:  DateTime
    email: String
    name:  String
    surname:  String
    role: Role
}

enum Role {
    ADMIN
    USER
}

input UserUpdateRequest{
    id:ID!
    username:  String!
    email: String!
    name:  String
    surname:  String
    role: Role
}

input UserCreateRequest{
    username:  String!
    email: String!
    name:  String
    surname:  String
    role: Role
}
```

Pros and Cons of GraphQL

Pros

- Single endpoint
- Fetch specific fields from data (Reduce response size)

Cons

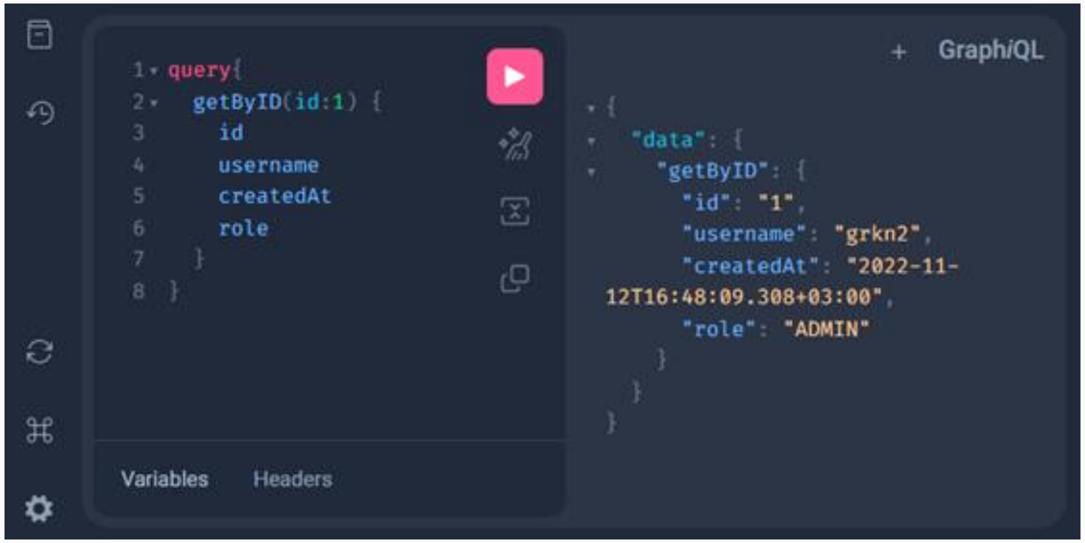
- Complex queries
- Caching implementation

Testing - Graphiql

You can use **Graphiql** console for execute queries.

*In spring boot, you have to enable it before:

- spring.graphql.graphiql.enabled=true

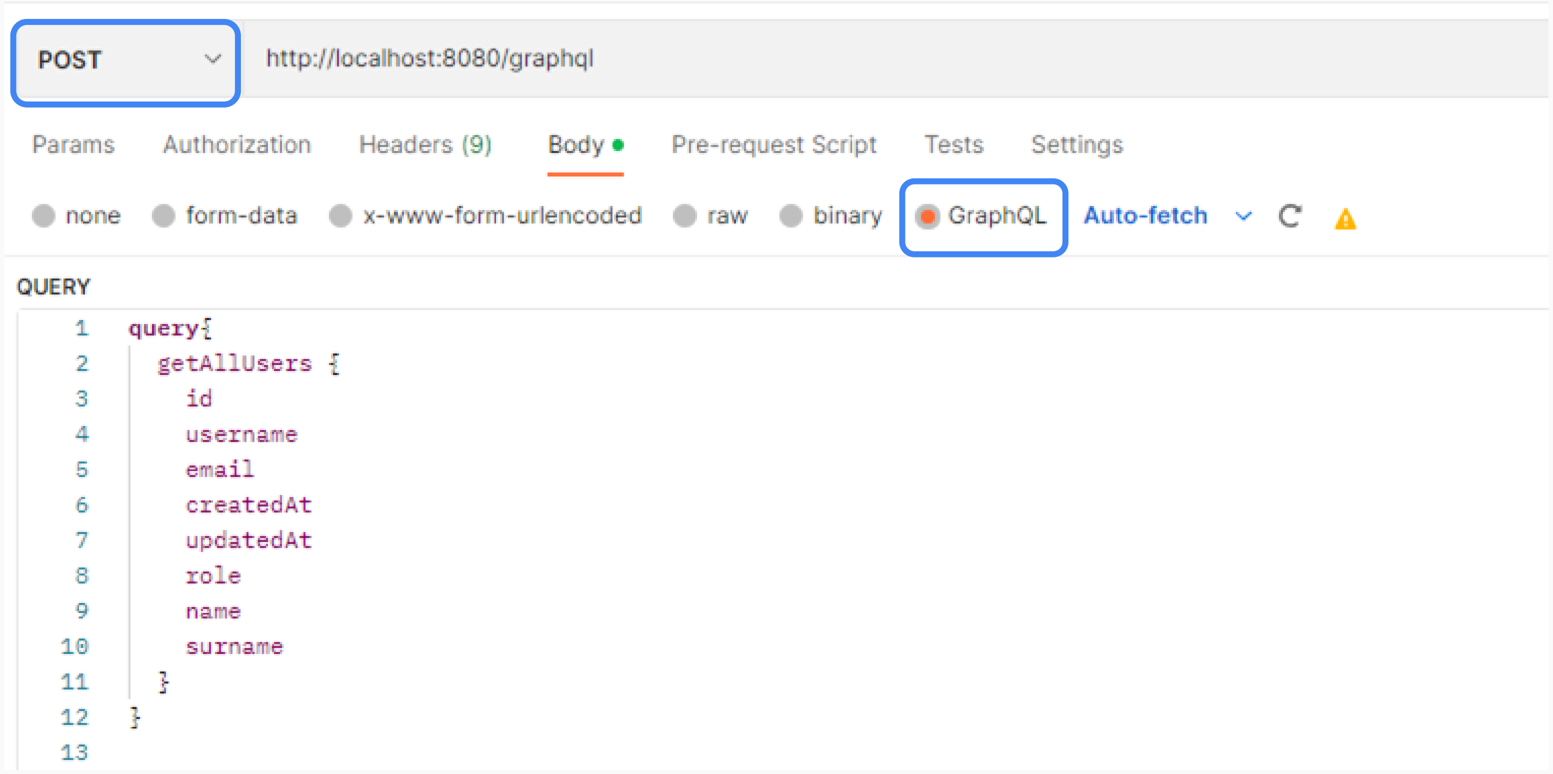


```
query{
  getByID(id:1) {
    id
    username
    createdAt
    role
  }
}
```

```
{
  "data": {
    "getByID": {
      "id": "1",
      "username": "grkn2",
      "createdAt": "2022-11-12T16:48:09.308+03:00",
      "role": "ADMIN"
    }
  }
}
```

Testing - Postman

Select **POST** request and send your query in **graphql** section.



```
query{
  getAllUsers {
    id
    username
    email
    createdAt
    updatedAt
    role
    name
    surname
  }
}
```

Spring Boot Kotlin implementation

Add these dependencies to your pom.xml file:

```
 <dependency>
   <groupId>org.springframework.graphql</groupId>
   <artifactId>spring-graphql-test</artifactId>
   <scope>test</scope>
</dependency>

<dependency>
   <groupId>com.graphql-java</groupId>
   <artifactId>graphql-java-extended-scalars</artifactId>
   <version>19.0</version>
</dependency>

 <dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-graphql</artifactId>
</dependency>
```

Spring Boot Kotlin implementation - Controller

```
@Controller
class UserController(private val userService: UserService) {

    @QueryMapping
    fun getAllUsers() = userService.getAllUsers().map { it.toDTO() }

    @QueryMapping
    fun getByID(@Argument id: Long)  = userService.getUserByID(id).toDTO()

    @MutationMapping
    fun updateUser(@Argument user: UserUpdateRequest) = userService.updateUser(user).toDTO()

    @MutationMapping
    fun createUser(@Argument user: UserCreateRequest) = userService.createUser(user).toDTO()

    @MutationMapping
    fun deleteUser(@Argument id: Long) = userService.deleteUser(id)
}
```

Must be the **same name** with **query** and mutation **definitions** in **.graphqls** file

```
type Query {
   getAllUsers : [UserDTO]
   getByID(id: ID!) : UserDTO
}
```

```
type Mutation {
   createUser(user: UserCreateRequest): UserDTO
   updateUser(user: UserUpdateRequest): UserDTO
   deleteUser(id: ID!): Boolean
}
```

Spring Boot Kotlin implementation - Error Handling

```
@Component
class GlobalExceptionHandler : DataFetcherExceptionResolverAdapter()  {


   override fun resolveToSingleError(e: Throwable, env: DataFetchingEnvironment): GraphQLError? {
      return when (e) {
         is UserNotFoundException -> toGraphQLError(e)
         is Exception -> toGraphQLError(e)
         else -> super.resolveToSingleError(e, env)
      }
   }

   private fun toGraphQLError(e: Throwable): GraphQLError? {
      return
GraphqlErrorBuilder.newError().message(e.message).errorType(ErrorType.DataFetchingException).build()
   }
}
```

Spring Boot Kotlin implementation - Integration Tests

```
/ /other annotations
@AutoConfigureGraphQlTester
internal class UserControllerTest(
   @Autowired private val graphQlTester: GraphQlTester
) {

   @Test
   fun `should return all users`() {
      val query: String = """
     Query{ getAllUsers { id username email role name surname createdAt updatedAt } }
    """
      graphQlTester.document(query)
          .execute()
          .path("getAllUsers")
          .entityList(UserDTO::class.java)
          .hasSize(2)
   }
}
```

Thanks

Full Project: https://github.com/gurkanucar/spring-boot-kotlin-graphql

Resources

https://graphql.org

https://www.baeldung.com/spring-graphql

https://www.javatpoint.com/graphql-advantages-and-disadvantages

https://docs.spring.io/spring-graphql/docs/current/reference/html

https://dzone.com/articles/error-handling-in-spring-for-graphql

https://medium.com/supercharges-mobile-product-guide/graphql-server-using-spring-boot-part-i-722bdd715779

https://refactorfirst.com/spring-boot-with-graphql