



REFCARDS (/reference-cards) TRENDS (/trendreports) PARTNERS (/partners)

 (/users/login.html)

 (/search)

[Culture and Methodologies](#)

[Data Engineering](#)

[Software Design and Architecture](#)

[Coding](#)

[Testing, Deployment, and Maintenance](#)

[Partner](#)

[\(/culture-and-methodologies\)](#)

[\(/data-engineering\)](#)

[\(/software-design-and-architecture\)](#)

[\(/coding\)](#)

[\(/testing-deployment-and-maintenance\)](#)

[Zones](#)

[NEW] DZone's 2023 "DevOps: CI/CD, Application Delivery, and Release Orchestration" Trend Report

Get the free report ▶

(https://dzone.com/trendreports/devops-4?utm_source=dzone&utm_medium=sitenotification)

Partner Resources

X

GraphQL With Spring Boot

In this article, learn more about GraphQL, see whether GraphQL is better than REST, and go through an example schema.



by [Sunitha Narayana \(/users/4211690/sunithan.html\)](/users/4211690/sunithan.html) · Jan. 24, 20 · Tutorial

 Like (15)
  Comment (6)
  Save
  Tweet
  Share
  41.88K Views



GraphQL – A Query Language for APIs

The basic need of most applications is to fetch data from the backend, which could be a database, a third-party application, NFS, etc. API interface is written to facilitate this.

REST web services are one of the most popular architectures to expose data from the backend. Previously, this was the best fit because the development pace wasn't as fast as it is today. We are all so comfortable with REST, so the question is, "Is GraphQL better than REST?"

REST vs GraphQL

x

- **Network Performance:** When UI invokes a REST API, it will not have any control over the response data, whereas GraphQL provides a mechanism to specify the fields of interest in response. This reduces network overload by fetching



the smallest data possible.

Endpoint: In REST, each resource is identified by a URI. This makes the client obliged to know each endpoint. In GraphQL, all resources are identified by a single endpoint. There is no hassle of testing, deploying, and maintaining multiple URIs.

- **Data Fetching Strategy:** In GraphQL, we have only one endpoint. The client sends a single query with the required field. This helps with improved network performance and avoids the over-fetching/under-fetching data problem.

You might also like: **GraphQL Java Example for Beginners [Spring Boot]**
(<https://dzone.com/articles/a-beginners-guide-to-graphql-with-spring-boot>)

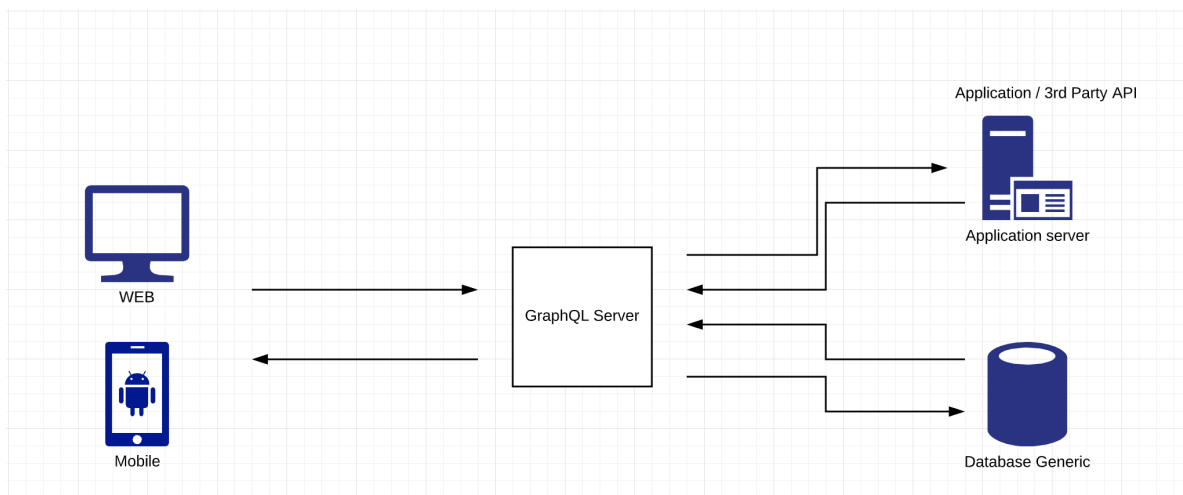
What Is GraphQL?

GraphQL is a query language. It gives control to the user to query what he needs, thus reducing the payload on the network.

The main terms used most commonly in GraphQL are:

- **Schema** — The contract between the GraphQL client and the GraphQL server
- **Query** — Similar to GET call in REST and used by the client to query the fields
- **Mutations** — It is similar to a POST/PUT call in REST and is used by the client for any insert/update operation

GraphQL Architecture:




Schema and Type System

GraphQL uses Schema Definition Language (SDL) to write the schema, which is a contract between client and server. In the schema, we define all the fields and functionalities exposed by APIs.

Once we have this schema in place, both the client and the server can continue their development independently. The client can mock the data until the server is ready.



Here is an example schema for **Student** and his **Address**. This schema contains a type definition for Student and Address. Each type may have one or more fields. In our example, Student and Address has 5 and 3 fields respectively. The non-nullable fields are represented with ! at the end of the field definition.

The relationship between types can also be seen in the below example. The Address type is referred to in the Student type definition.



Java

[REFCARDS \(/refcards/\)](#)
[RECOMMENDATIONS \(/recommendations/\)](#)
[TECHREPORTS \(/techreports/\)](#)
[TRENDS \(/trends/\)](#)
[WEBSNARERS \(/websnarers/\)](#)

 (/users/login.html)
  (/search)

1

type Student {

2

id: ID!

3

name: String!

4

age: Int!

5

phone: String!

6

address: Address!

7

}

8

9

10

type Address {

11

street: String!

12

city: String!

13

zipcode: Int!

14

}

15

16

17

type Query {

18

students(name:String);

19

}

20

21

22

type Mutation {

23

createStudent(name:String!, age:String, phone:String!);

24

}

25

Culture and Methodologies

[Data Engineering \(/data-engineering/\)](#)
[Software Design and Architecture \(/software-design-and-architecture/\)](#)
[Coding \(/coding/\)](#)
[Testing, Deployment, and Maintenance \(/testing-deployment-and-maintenance/\)](#)

Partner

Zones

GraphQL Query

In the below example, the GraphQL query contains the root field i.e. Students and query's payload i.e. name and age. This query also takes an argument student name. This query returns the student's name and their age whose name matches to Smith

```

Java
1 query{
2     students(name:"Smith") {
3         name
4         age
5     }
6 }

```

Mutations

This is similar to a REST POST method. If we need an API to create/modify data, then we can go for mutations. Possible mutations are:

1. Creating new data
2. Modifying existing data

Mutation syntax starts with the keyword mutation. The below mutation creates a student, and the server returns the unique ID of the inserted record.


Example:

```



Java
1 mutation createStudent($req: StudentInput) {
2     createStudent(input: $req)
3     {
4         name
5         address {
6             city
7             zipcode
8         }
9     }

```

X


DZone

[REFCARDS \(/reference-cards/\)](#)
[REPORTS \(/reports/\)](#)
[TECHNICAL WEBSINARS \(/technical-websinars/\)](#)

 (/users/login.html)
  (/search)

[Culture and Methodologies \(/culture-and-methodologies/\)](#)
[Data Engineering \(/data-engineering/\)](#)
[Software Design and Architecture \(/software-design-and-architecture/\)](#)
[Coding \(/coding/\)](#)
[Testing, Deployment, and Maintenance \(/testing-deployment-and-maintenance/\)](#)
[Partner Zones](#)

```

10 }
11 {
12   query: {
13     // ...
14   },
15   mutation: {
16     // ...
17   },
18   subscription: {
19     // ...
20   },
21   // ...
22 }
23
24 --
  
```

Resolvers

Resolvers are functions written in the GraphQL server corresponding to each field in GraphQL query/mutations. When a request comes to the server, it will invoke resolvers' functions corresponding to fields mentioned in the query.

GraphQL With Spring Boot

The complete source code is available at <https://github.com/sunithan09/SampleGraphQLApp> (<https://github.com/sunithan09/SampleGraphQLApp>).

Now that we know the main features of GraphQL, let's go over the details on implementing GraphQL with Spring Boot.

To build a GraphQL server with Spring Boot, all you need are the below dependencies in your pom:

```

XML
1 <dependency>
2   <groupId>com.graphql-java-kickstart</groupId>
3   <artifactId>graphql-spring-boot-starter</artifactId>
4   <version>5.11.1</version>
5 </dependency>
6 <dependency>
7   <groupId>com.graphql-java-kickstart</groupId>
8   <artifactId>graphql-java-tools</artifactId>
9   <version>5.7.1</version>
10
  
```

Once you have these dependencies, Spring Boot downloads the necessary GraphQL handlers to parse the schema, and GraphQL API's are exposed at the endpoint /graphql. This also can be customized in application.properties.

Schema and Resolvers


Schemas must be written in classpath with the extension “.graphqls”. We can have as many schema files as we require. Each type defined in the schema must have field resolvers. FieldResolverError results if the GraphQL server cannot find the resolver for any given field.

Along with field resolvers, it is required to have **Query resolvers** and **Mutation resolvers**.



Spring bean must implement GraphQL Query Resolver.

Query resolver for Students may look like:

x



[REFCARDS \(/REFCARDS\)](#)
[REPORTS \(/REPORTS\)](#)
[TRENDS \(/TRENDS\)](#)
[WEBSNARS \(/WEBSNARS\)](#)

 (/users/login.html)
  (/search)

[Java](#)
[Culture and Methodologies](#)
[Data Engineering](#)
[Software Design and Architecture](#)
[Coding](#)
[Testing, Deployment, and Maintenance](#)
[Partner](#)

[\(/culture-and-methodologies\)](#)
[\(/data-engineering\)](#)
[\(/software-design-and-architecture\)](#)
[\(/coding\)](#)
[\(/testing-deployment-and-maintenance\)](#)
[Zones](#)

```

1 import com.coxautodev.graphql.tools.GraphQLQueryResolver;
2 import com.sample.graphql.modal.Student;
3 import com.sample.graphql.service.StudentService;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.stereotype.Component;
6
7 @Component
8 public class StudentQueryResolver implements GraphQLQueryResolver
9 {
10     @Autowired
11     private StudentService studentService;
12
13     public Student student (String name)
14     {
15         return studentService.getStudentDetailsByName(name);
16     }
17 }

```

And mutation resolver to create Student would be:

```

Java
1 import com.coxautodev.graphql.tools.GraphQLMutationResolver;
2 import com.sample.graphql.modal.Student;
3 import com.sample.graphql.service.StudentService;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.stereotype.Component;
6
7 @Component
8 public class StudentMutationResolver implements GraphQLMutationResolver
9 {
10     @Autowired
11     private StudentService studentService;
12
13     public Student createStudent (Student student) {
14         return studentService.createStudentRecord(student);
15     }
16 }

```

Testing

Now your GraphQL server is ready. You can test your API's using Postman or the GraphiQL UI tool.

GraphiQL

To use the GraphiQL UI tool, you need to have the below dependency:

```

XML
1 <dependency>
2     <groupId>com.graphql-java-kickstart</groupId>
3     <artifactId>graphiql-spring-boot-starter</artifactId>
4     <version>5.11.1</version>
5 </dependency>

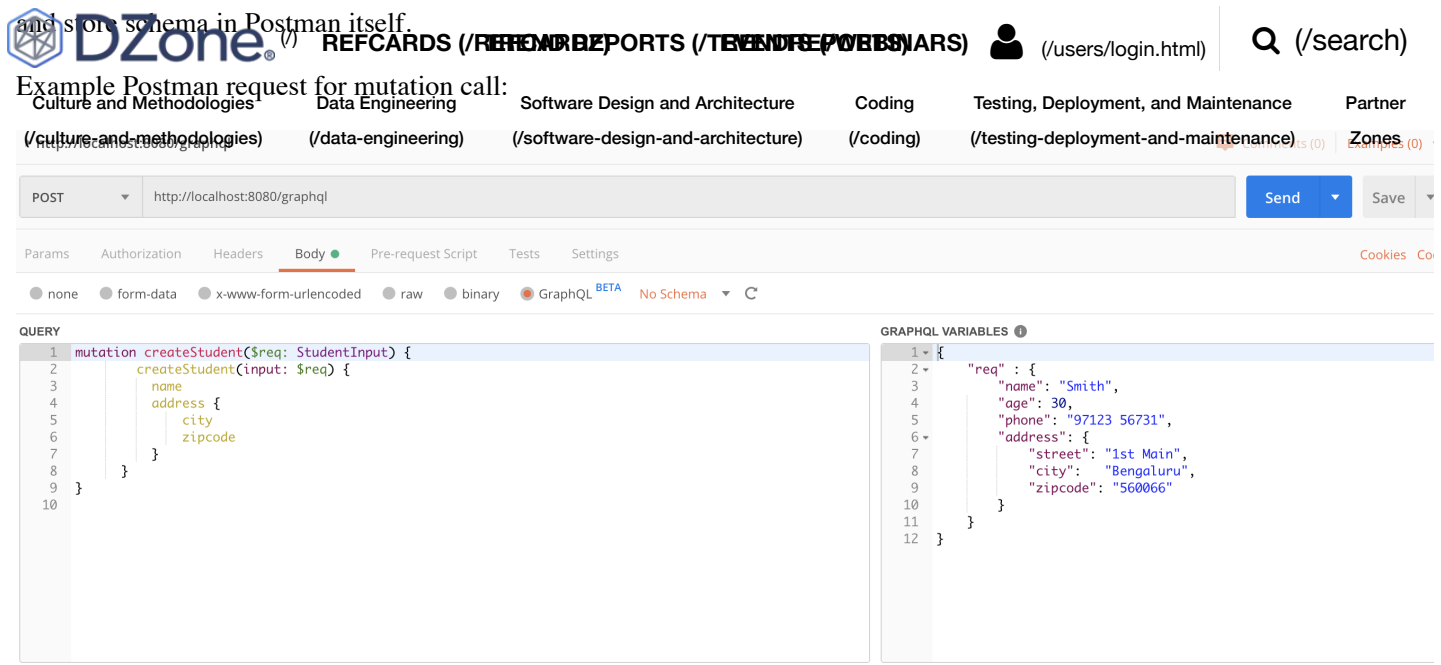
```

GraphiQL is a web tool and is accessible at the root /graphiql.

Postman

We can use Postman to post queries to the GraphQL server. These are POST calls, and the query is passed as body^x with content type set to application/graphql.

Latest Postman versions (V7 onwards) support GraphQL queries where you can pass your GraphQL queries directly or build



Postman request

Limitations

GraphQL is indeed a great way to build and query APIs, but it does have certain limitations. A few of them are:

- The nested queries having multiple fields could lead to performance issues. GraphQL queries has to be carefully designed as the control is with client and it could ask anything.
- Web caching is easier with REST compared to GraphQL, as the latter has a single endpoint.
- Retrieving objects recursively (to infinite length) is not supported in GraphQL. One has to specify to what depth it needs the data to get the recursive data.

Conclusion

Though GraphQL is becoming popular, it is not always the best choice, and it is not the alternative for REST web services. REST has its own advantages over GraphQL in terms of web cache, performance, etc.

Keeping the above limitations in mind will help you choose the right tool for your needs.

Further Reading

GraphQL: Understanding Spring Data JPA/Spring Boot

(<https://dzone.com/articles/graphql-understanding-with-springdatajpaspringboot>)

(<https://dzone.com/articles/graphql-understanding-with-springdatajpaspringboot>)

Microservices With GraphQL (<https://dzone.com/articles/microservices-with-graphql>)

X

GraphQL Spring Framework Database Spring Boot

Opinions expressed by DZone contributors are their own.



Popular on DZone REF CARDS (/REF CARDS) TRENDS (/TRENDS) REPORTS (/REPORTS) ARTICLES (/ARTICLES)

 (/users/login.html)

Q (/search)

React, Angular and Vue.js: What's the Technical Difference? (articles)					
/culture-and-design	/engineering	/software-design-and-architecture	/coding	/testing-deployment-and-maintenance	Zones

- [Deploy a Kubernetes Application With Terraform and AWS EKS \(/articles/deploy-a-kubernetes-application-with-terraform-and-aws-eks?fromrel=true\)](#)
- [GKE Security: Top 10 Strategies for Securing Your Cluster \(/articles/gke-security-top-10-strategies-for-securing-your-c?fromrel=true\)](#)
- [What Is API-First? \(/articles/what-is-api-first?fromrel=true\)](#)

ABOUT US

About DZone (/pages/about)

Send feedback (<mailto:support@dzzone.com>)

Careers (<https://careers.dzone.com/>)

Sitemap (/sitemap)

ADVERTISE

Advertise with DZone (<https://advertise.dzone.com>)

CONTRIBUTE ON DZONE

Article Submission Guidelines (</articles/dzones-article-submission-guidelines>)

Become a Contributor (/pages/contribute)

Visit the Writers' Zone ([/writers-zone](#))

LEGAL

Terms of Service (/pages/tos)

Privacy Policy (/pages/privacy)

CONTACT US

600 Park Offices Drive

Suite 300

Durham, NC 27709

support@dzone.com (mailto:support@dzone.com)

+1 (919) 678-0300 (tel:+19196780300)

Let's be friends:

in

<https://www.linkedin.com>

(/page1?url=http://www.facebook.com

/feed(5)Z0Dz1d4zbr0/)

X