**GraphQL Client Call in with SpringBoot 3-2025**

There is a publicly hosted GraphQL api exposed for testing. We can use that one to make call.

Public URL: <https://countries.trevorblades.com/graphql>

To get Country Details by passing Country Code

Actual GraphQL Query is given below.

**query** Country {

    country(code: "IN") {

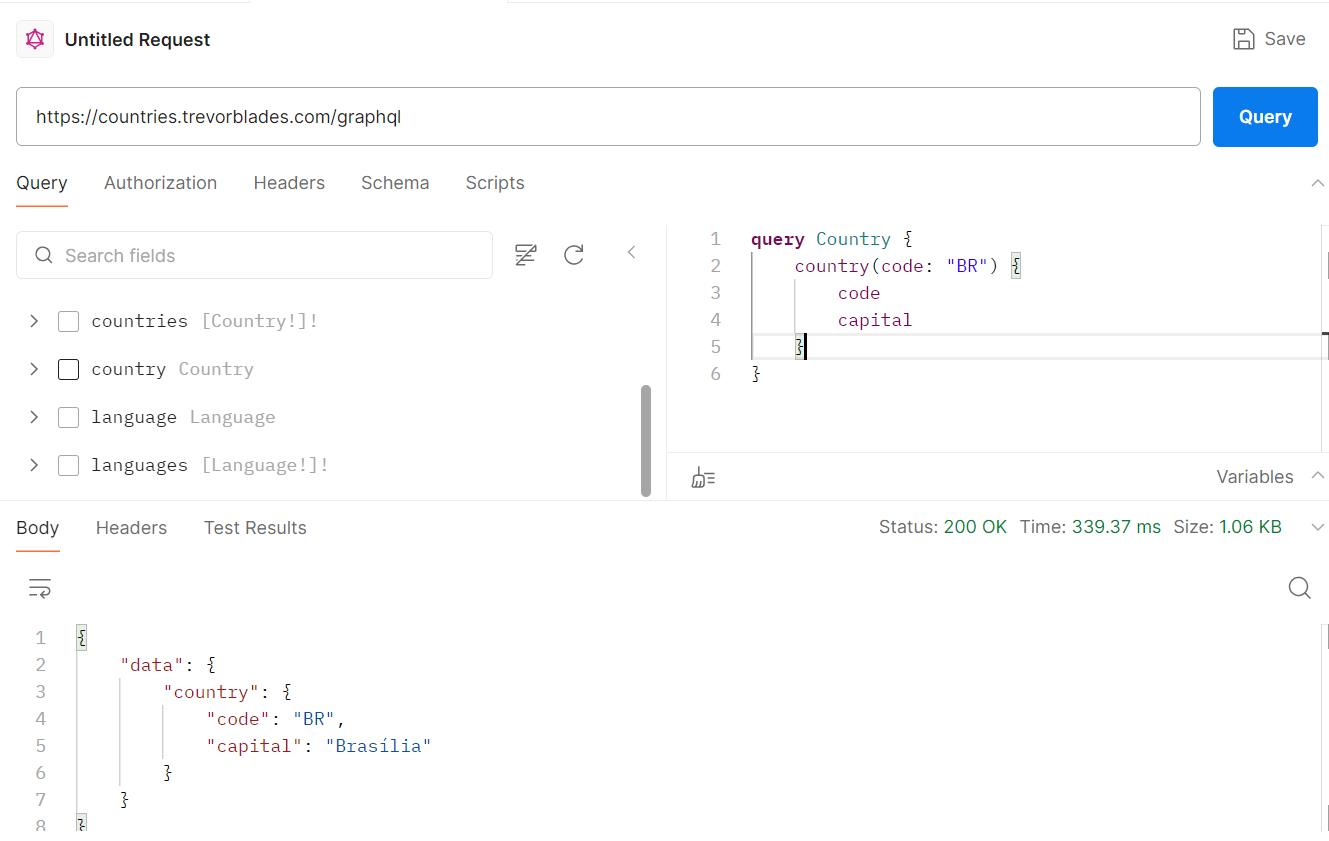
        code

        capital

    }

}

The screenshot is given below.



**Can the GraphQL query be called like normal HTTP ways ? Yes**

**How to Respond to GraphQL requests using an HTTP server**

**query** Country {

    country(code: "IN") {

        code

        capital

    }

}

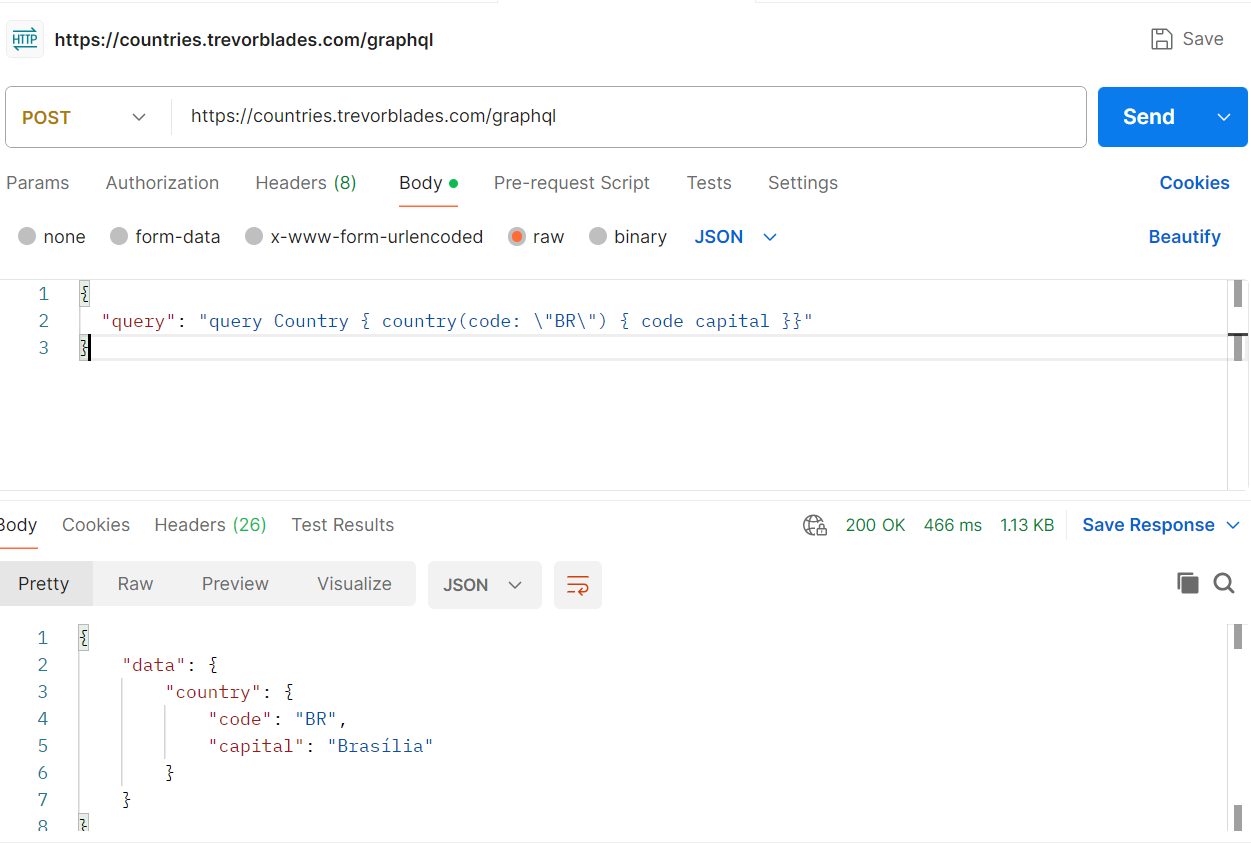
The above graphql query can be written as below.

{

  "query": "query Country { country(code: \"BR\") { code capital }}"

}

In order to make HTTP Post call, you have to make HTTP Post call with the above query and in case of Postman client, paste the above query as **raw**. The screenshot is given below.



In case of dynamic values, the graphql query is given below.

**query($code: ID!) {**

**country(code: $code) {**

**code**

**capital**

**}**

**}**

The above query can be written as below.

{

  "query": "query($code: ID!) { country(code: $code) { code capital } }",

  "variables": {

    "code": "IN"

  }

}

Based upon the above query, you can make HTTP call in PostMan client. The screenshot is given below.

A screenshot of a computer program

AI-generated content may be incorrect.

There is an online tool which can convert Graphql query to Json body.

<https://datafetcher.com/graphql-json-body-converter>

We can make GraphQL query call in various ways. Prominently, we can use **GraphQLWebClient** and **RestTemplate** as of now.

**How to use GraphQLWebClient**

First you can configure the URL for Graphql in application.properties as shown below.

**graphql.client.url**=https://countries.trevorblades.com/graphql

Actual graphql query is given below.

**query Country {**

**country(code: "BR") {**

**code**

**capital**

**}**

**}**

You need to Autowire as shown below.

@Autowired

**private** GraphQLWebClient graphQLWebClient;

The code is given below.

**public** **void** callUsingGraphQLWebClient() {

String queryStr1 = """

query($code: ID!) {

country(code: $code) {

code

capital

}

}

""";

Map<String, Object> variables = **new** HashMap<>();

variables.put("code", "IN");

GraphQLRequest request = GraphQLRequest.*builder*()

.query(queryStr1)

.variables(variables)

.build();

//print requestBody

System.***out***.println(request.toString());

//passing requestBody to server

GraphQLResponse response = graphQLWebClient.post(request).block();

System.***out***.println("Raw Response: "+response.getRawResponse());

}

Mark the difference

query($code: ID!) {

country(code: $code) {

code

capital

}

}

**query Country {**

**country(code: "BR") {**

**code**

**capital**

**}**

**}**

For another Graphql query, it is given below.

**query Countries {**

**countries(filter: { currency: { eq: "USD" } }) {**

**currencies**

**name**

**}**

**}**

For **GraphQLWebClient**, the above can be written as

String queryStr2 = """

query($eq11: String!) {

countries(filter: { currency: { eq: $eq11 } }) {

currencies

name

}

}

""";

The following variables will be used.

Map<String, Object> variables = new HashMap<>();

variables.put("eq11", "USD");

Let us check with a complex query with two dynamic variables.

Actual GraphQL query is given below.

**query** Countries {

    countries(filter: { code: { eq: "VA" }, continent: { eq: "EU" } }) {

        capital

        code

        awsRegion

        languages {

            name

            native

        }

    }

}

The above can be written for Java code as

String queryStr2 =

"""

query($eq1: String!, $eq2: String!) {

countries(filter: { code: { eq: $eq1 }, continent: { eq: $eq2 } }) {

capital

code

awsRegion

languages {

name

native

}

}

}

""";

All the java code for the above is same.

**The trick is given below.**

If the GraphQL query is given like this

**query** Countries {

    countries(filter: { code: { eq: "VA" }, continent: { eq: "EU" } }) {

        capital

        code

        awsRegion

        languages {

            name

            native

        }

    }

}

While writing query for java code execution,

First write as below. There is no need to mention the name of the query in the first line

**query($variableName1: DataType!, $variableName2: DataType!)** {

copy paste the graphql query with the **$variableName**

}

Like query($eq1: String!, $eq2: String!) { }

countries(filter: { code: { eq: $eq1 }, continent: { eq: $eq2 } }) {

capital

code

awsRegion

languages {

name

native

}

}

Final query should look like this.

String queryStr2 =

"""

query($eq1: String!, $eq2: String!) {

countries(filter: { code: { eq: $eq1 }, continent: { eq: $eq2 } }) {

capital

code

awsRegion

languages {

name

native

}

}

}

""";

In case of Mutation call, the complete code is given below.

**public** **void** mutationCall1() {

String queryStr2 =

"""

mutation($fName: String!, $lName: String!, $sta1: MaritalStatus, $sta2: Boolean) {

createAppUser(

inUser: { firstName: $fName, lastName: $lName, mStatus: $sta1, status: $sta2 }

) {

id

firstName

lastName

mStatus

status

salary

phoneNos

}

}

""";

Map<String, Object> variables = **new** HashMap<>();

variables.put("fName", "Hati");

variables.put("lName", "Ghoda");

variables.put("sta1", "MARRIED");

variables.put("sta2", Boolean.***TRUE***);

//setting up requestBody with query and variables

GraphQLRequest request = GraphQLRequest.*builder*()

.query(queryStr2).variables(variables).build();

//print requestBody

System.***out***.println(request.toString());

//passing requestBody to server

GraphQLResponse response = graphQLWebClient.post(request).block();

System.***out***.println(response.getRawResponse());

}

The actual GraphQL query is given below.

**mutation** CreateAppUser {

    createAppUser(

        inUser: {

            firstName: "Hati"

            lastName: "Ghoda"

            mStatus: MARRIED

            status: **true**

        }

    ) {

        id

        firstName

        lastName

        mStatus

        status

        salary

        phoneNos

    }

}

**How to use RestTemplate in GraphQL Call**

Complete Java code is given below.

**public** **void** usingRestTemplatecall1() {

String url = "https://countries.trevorblades.com/graphql";

String queryStr2 = """

query($eq11: String!) {

countries(filter: { currency: { eq: $eq11 } }) {

currencies

name

}

}

""";

RestTemplate restTemplate = **new** RestTemplate();

HttpHeaders headers = **new** HttpHeaders();

headers.setContentType(MediaType.***APPLICATION\_JSON***);

//create requestBody with query

Map<String, Object> requestBody = **new** HashMap<>();

requestBody.put("query", queryStr2);

//create variables

Map<String, Object> variables = **new** HashMap<>();

variables.put("eq11", "USD");

//add variables in the requestBody

requestBody.put("variables",variables);

HttpEntity<Object> entity = **new** HttpEntity<>(requestBody, headers);

ResponseEntity<String> response = restTemplate.postForEntity(url, entity, String.**class**);

System.***out***.println(response.getBody());

}

For the above, GraphQL query is given below.

**query** Countries {

    countries(filter: { currency: { eq: "USD" } }) {

        currencies

        name

    }

}

**How to Use WebClient to make GraphQL Call**

Actual GraphQL Query is given below.

**query** Country {

    country(code: "IN") {

        code

        capital

    }

}

The above graphql query can be written as below for making HTTP call.

{

  "query": "query Country { country(code: \"IN\") { code capital }}"

}

Finally, the above code is written as below for Java

String body = """

{

"query": "query($code: ID!) { country(code: $code) { code capital } }",

"variables": {

"code": "IN"

}

}

""";

Java code is given below.

**public** **void** callUsingWebClient() {

String body = """

{

"query": "query($code: ID!) { country(code: $code) { code capital } }",

"variables": {

"code": "IN"

}

}

""";

WebClient webClient = WebClient.*create*(firstUrl);

String sss = webClient

.post()

.uri(firstUrl)

.contentType(MediaType.***APPLICATION\_JSON***)

.bodyValue(body)

.retrieve()

.bodyToMono(String.**class**)

.block();

System.***out***.println(sss);

}

Check below for Analysis

Actual GraphQL query is given below.

**query** Country {

    country(code: "IN") {

        currency

        name

    }

}

Request Body for Http POST call for the above query is given below.

{

  "query": "query($code11: ID!) { country(code: $code11) { currency name } }",

  "variables": {

    "code11": "IN"

  }

}

References

<https://graphql.org/learn/serving-over-http/#post-request>

<https://datafetcher.com/graphql-json-body-converter>