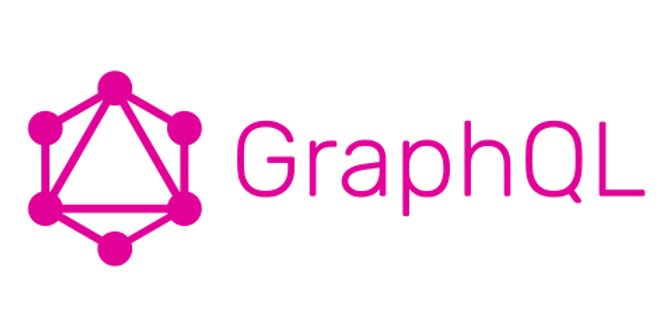
**­GraphQL(Graph Query Language)**



GraphQL is a query language for your API, and a server-side run time for executing queries by using a type system you define for your data.

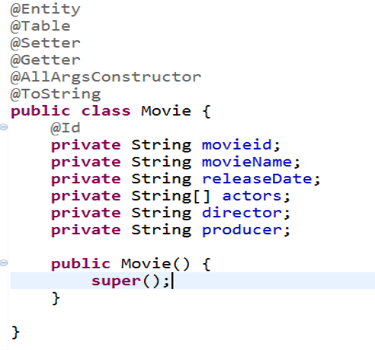
GraphQL invented by Facebook on 2012 support with multiple technology like C#/.NET,Clojure,Elixir,Erlang,Go,Groovy,Java,JavaScript,PHP,Python,Scala,Ruby So we can say this API is technology independent

In sort GraphQL is an alternative to REST for Web APIs.

**Why it called as Alternative of Rest, what is the main purpose to use it?**

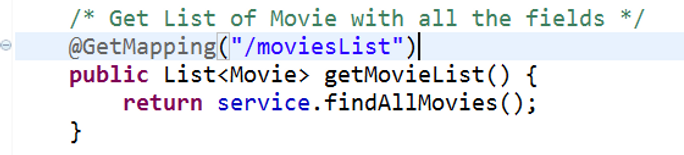
Let’s discuss with small example

Assume I have one Model i.e. Movie with some fields like below

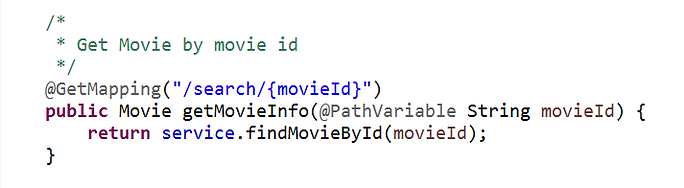


I have 2 requirements and I developed it using REST API Now let’s find out what is drawback In Rest that’s why GraphQL came into picture

1. Fetch List of movie info which stored in Database



2. Fetch Movie Info based on Selected MovieId

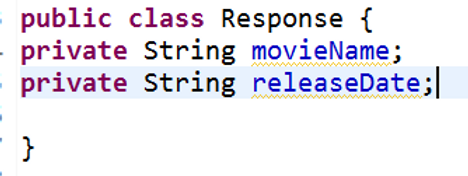


In first requirement my method return type is List<Movie> so it will give me Movies with all the fields like below



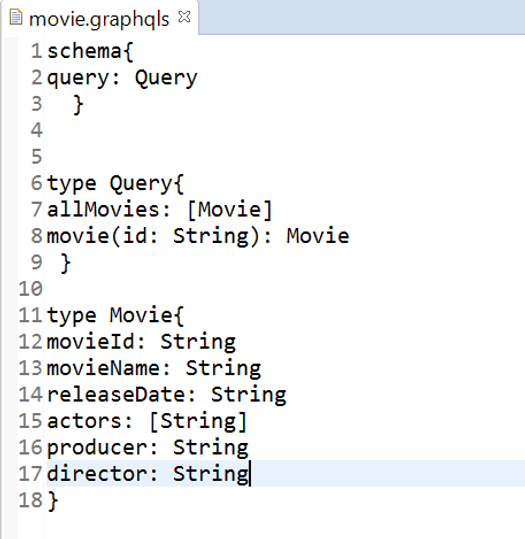
But I don’t want all fields as a part of response which present in Model class means I want only movieName and release Date as part of response so what I can do?

To customize my desired Response I need to create one more separate dto and I need to populate required field to that dto then I have to return that from controller right ?



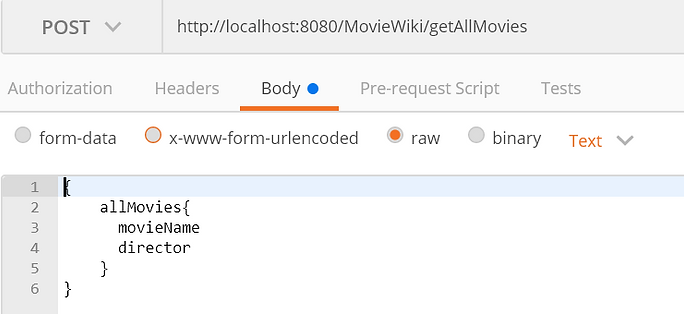
Then I need to return List<Response> from controller instead of List<Movie> right? Assume in future client change his requirement and he raised CR (Change Request) To display actors information along with movieName and releaseDate as part of Response for this API call Then again I need to add one more field in Response class with name actors and again I need to change business logic in service Think if requirement will keep on change then how many times I need to format my Response dto and business logic Same for requirement 2, I don’t want to display all fields of movie as a response I want some specific field like movieName and director and producer information so for that again I need to format my Response dto and business It looks my application is tightly coupled with business so how can we make it generic? We can make it generic using our Rest API also but we need to write few annotation in Response dto and we need to depend on some third party lib like JsonPath to parse desired response which seems bit burden for developer but which can be more easy using GraphQL To overcome this issue GraphQL introduced. Now let’s see how we can ignore above issue using GraphQL Really, it’s a beautiful concept everything u can achieve using type Schema

**Type Schema**: where we need to specify our desired response format like below

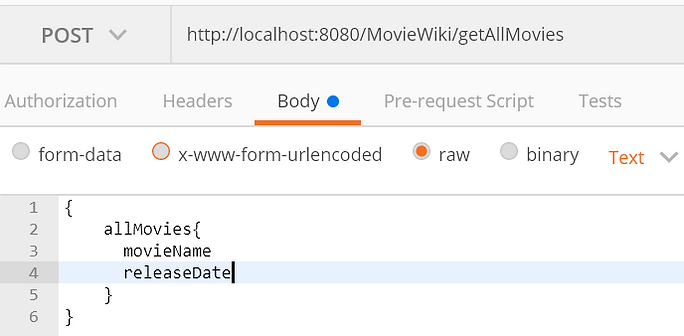


See the line of 6 to 9 who contains our query details like Line no 7: we are indicating hey my query name is all Movies when I call this Query allMovies then please return List<Movie> with desired fields , but you don’t worry I will send request with required fields which I want as part of response like below

**Request Query**: I want List of movie with fields’ movieName and director

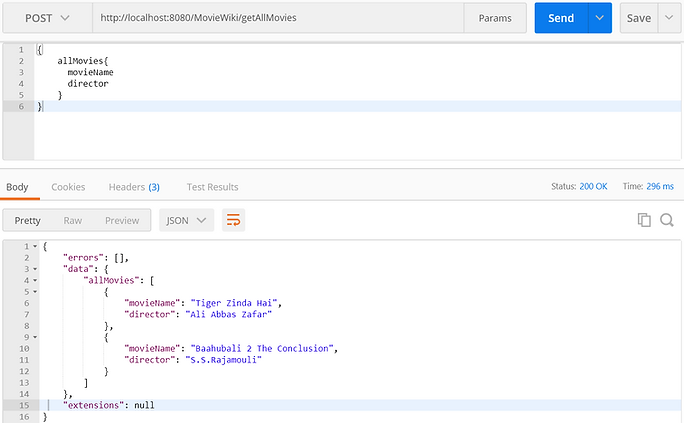


Tomorrow, I want some more field like only movieName and release Date



So, notice both Request properly The fields which we mentioned in Request Query those field data will be fetched from DB and return as Response to us See below complete Request and Response

**Result:**



Line no 8:

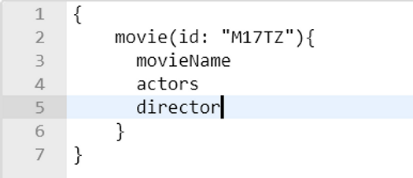
We are informing to GraphQL based on input movieId Give me Movie Object

As I explain above for requirement 1 same way we can get response which we need based on business so in future requirement is changing then we no need to bother about business logic just change the request query pass the field which you want

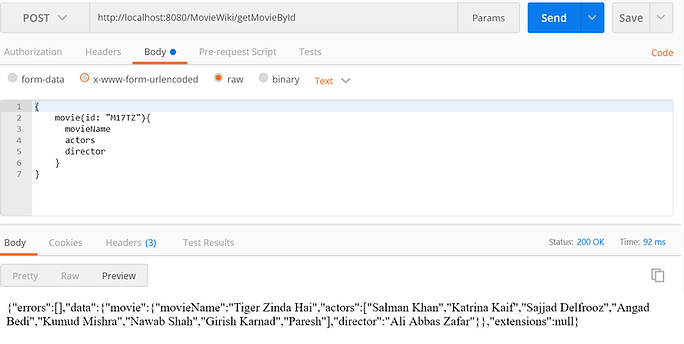
Let’s see one Example of req-2

I want a movie object with field movieName, director and actors info only

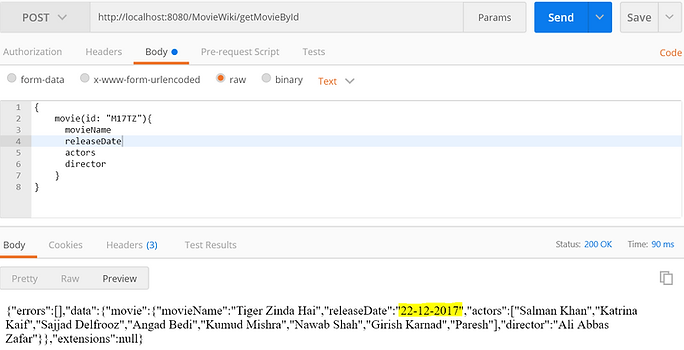
**Request Query**



Result:



In future u want releaseDate as part of response then u can add that field name in Request query so you will get required response in proper JSON format like below

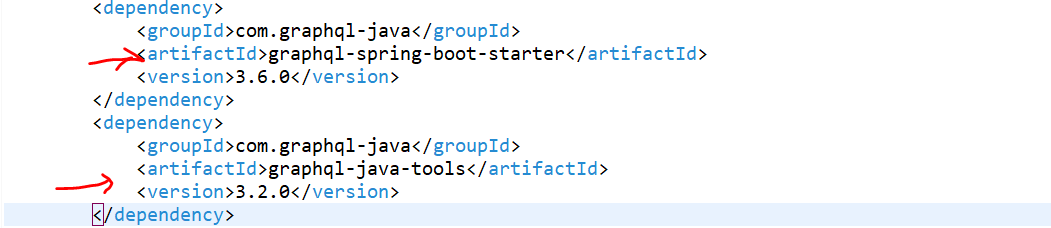


**==============Spring Boot Integration with GraphQL==========**

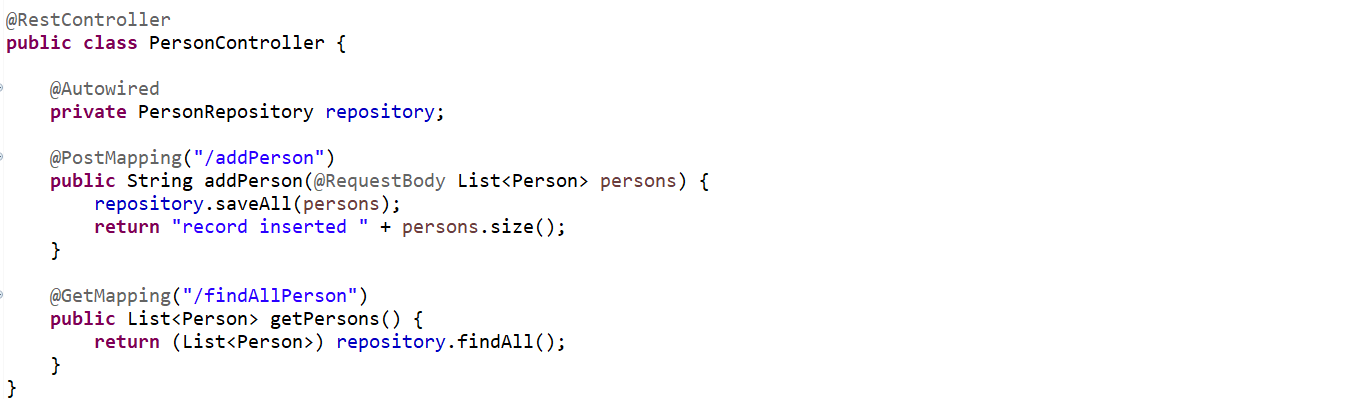
**App Name** - **spring-graphQL**

**Dependencies – Devtool. Lombok. Web, JPA, H2**

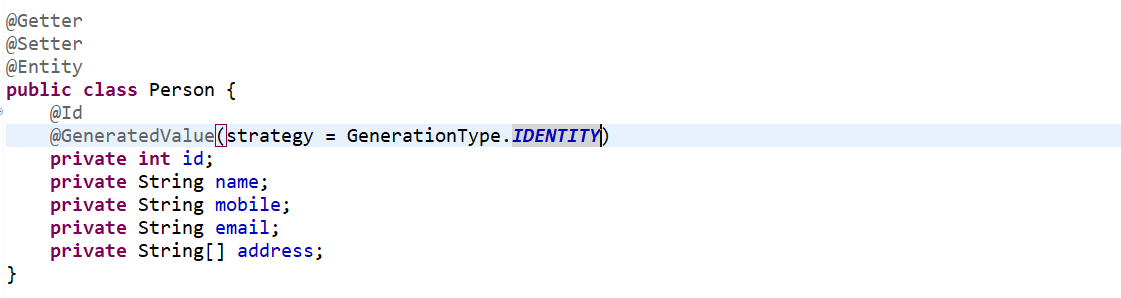
To work with GraphQL we need to work with GraphQL dependencies.



------------------------------------------------------------------------------------------------------------------------------------------



------------------------------------------------------------------------------------------------------------------------------------------



------------------------------------------------------------------------------------------------------------------------------------------

**GET** <http://localhost:8080/findAllPerson>

[

    {

        "id": 1,

        "name": "kaushal singh",

        "mobile": "1234567890",

        "email": "kk@gmail.com",

        "address": [

            "Bangalore",

            "Marathalli",

            "flat No : 502"

        ]

    },

    {

        "id": 2,

        "name": "Bharat Bhushan",

        "mobile": "8765434532",

        "email": "bb@gmail.com",

        "address": [

            "Bangalore",

            "HSR",

            "flat No : 107"

        ]

    },

    {

        "id": 3,

        "name": "Kanishk singh",

        "mobile": "6677885544",

        "email": "kk@gmail.com",

        "address": [

            "Bangalore",

            "Marathalli",

            "flat No : 302"

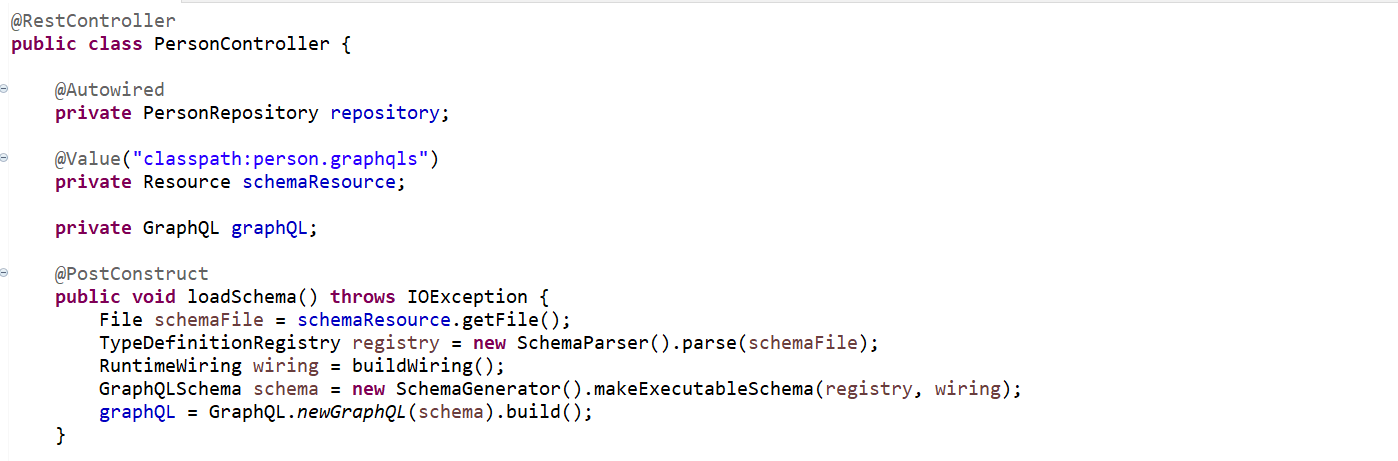
        ]

    }

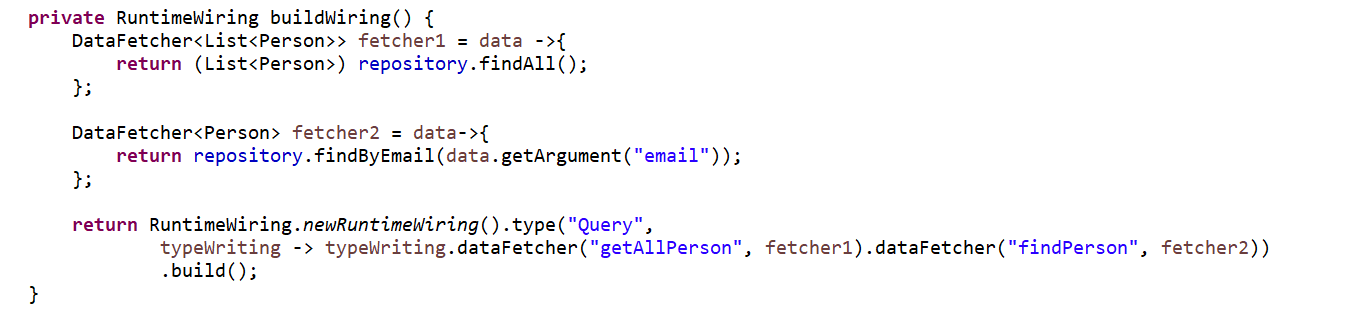
]



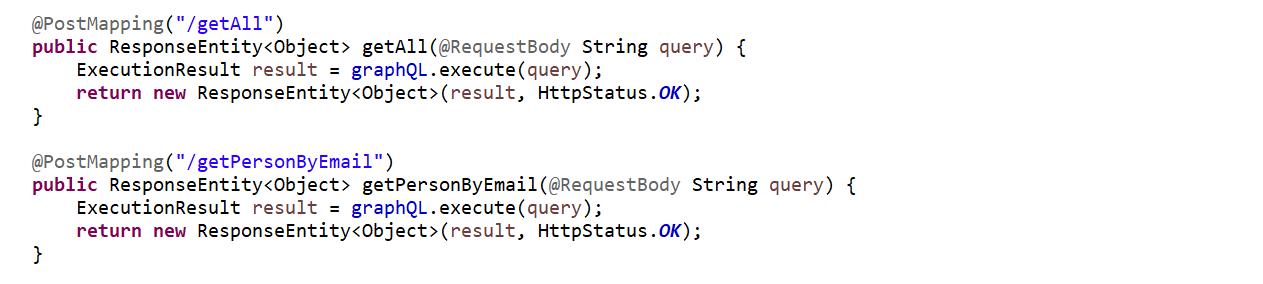
So, we are done with our type script Now, let's load this graph QLS file in my controller.



--------------------------------------------------------------------------------------------------------------------------------------------------------



--------------------------------------------------------------------------------------------------------------------------------------------------------



--------------------------------------------------------------------------------------------------------------------------------------------------------



**POST** [**http://localhost:8080/addPerson**](http://localhost:8080/addPerson)

**Request** Type Json

[{

    "name": "kaushal singh",

    "mobile": "1234567890",

    "email": "kk@gmail.com",

    "address": ["Bangalore", "Marathalli", "flat No : 502"]

},

{

    "name": "Bharat Bhushan",

    "mobile": "8765434532",

    "email": "bb@gmail.com",

    "address": ["Bangalore", "HSR", "flat No : 107"]

},

{

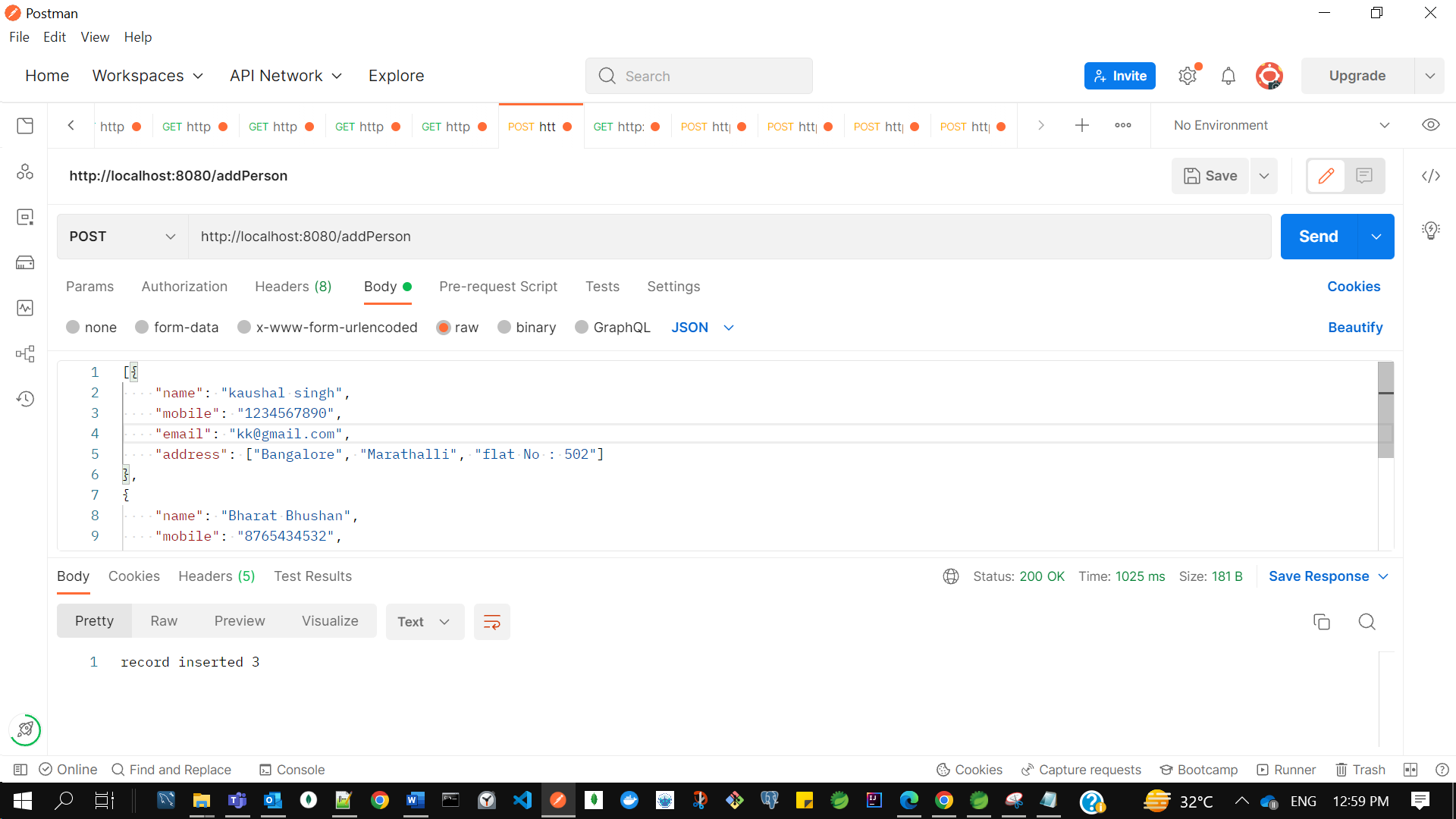
    "name": "Kanishk singh",

    "mobile": "6677885544",

    "email": "kk@gmail.com",

    "address": ["Bangalore", "Marathalli", "flat No : 302"]

}]



**GET** [**http://localhost:8080/findAllPerson**](http://localhost:8080/findAllPerson)

Response-

[

    {

        "id": 1,

        "name": "kaushal singh",

        "mobile": "1234567890",

        "email": "kk@gmail.com",

        "address": [

            "Bangalore",

            "Marathalli",

            "flat No : 502"

        ]

    },

    {

        "id": 2,

        "name": "Bharat Bhushan",

        "mobile": "8765434532",

        "email": "bb@gmail.com",

        "address": [

            "Bangalore",

            "HSR",

            "flat No : 107"

        ]

    },

    {

        "id": 3,

        "name": "Kanishk singh",

        "mobile": "6677885544",

        "email": "kk@gmail.com",

        "address": [

            "Bangalore",

            "Marathalli",

            "flat No : 302"

        ]

    }

]

Graph QL Test-

Request Type - Text

**URL** - <http://localhost:8080/getAll>

**Request –**

{

getAllPerson{

name

email

}

}

**Response –**

{

    "errors": [],

    "data": {

        "getAllPerson": [

            {

                "name": "kaushal singh",

                "email": "kk@gmail.com"

            },

            {

                "name": "Bharat Bhushan",

                "email": "bb@gmail.com"

            },

            {

                "name": "Kanishk singh",

                "email": "kk@gmail.com"

            }

        ]

    },

    "extensions": **null**,

    "dataPresent": **true**

}

**POST** [**http://localhost:8080/getAll**](http://localhost:8080/getAll)

**Request**-

{

getAllPerson{

name

address

}

}

**Response**-

{

    "errors": [],

    "data": {

        "getAllPerson": [

            {

                "name": "kaushal singh",

                "address": [

                    "Bangalore",

                    "Marathalli",

                    "flat No : 502"

                ]

            },

            {

                "name": "Bharat Bhushan",

                "address": [

                    "Bangalore",

                    "HSR",

                    "flat No : 107"

                ]

            },

            {

                "name": "Kanishk singh",

                "address": [

                    "Bangalore",

                    "Marathalli",

                    "flat No : 302"

                ]

            }

        ]

    },

    "extensions": **null**,

    "dataPresent": **true**

}

**POST** [**http://localhost:8080/getAll**](http://localhost:8080/getAll)

**Request-**

{

getAllPerson{

id

email

}

}

**Response-**

{

    "errors": [],

    "data": {

        "getAllPerson": [

            {

                "id": "1",

                "email": "kk@gmail.com"

            },

            {

                "id": "2",

                "email": "bb@gmail.com"

            },

            {

                "id": "3",

                "email": "kk@gmail.com"

            }

        ]

    },

    "extensions": **null**,

    "dataPresent": **true**

}

**POST** [**http://localhost:8080/getPersonByEmail**](http://localhost:8080/getPersonByEmail)

**Request–**

{

findPerson(email: "bb@gmail.com"){

name

address

}

}

**Response -**

{

    "errors": [],

    "data": {

        "findPerson": {

            "name": "Bharat Bhushan",

            "address": [

                "Bangalore",

                "HSR",

                "flat No : 107"

            ]

        }

    },

    "extensions": **null**,

    "dataPresent": **true**

}

Sts –

**Controller**

package com.java.selfdeveloped.graphql.api.controller;

import java.io.File;

import java.io.IOException;

import java.util.List;

import javax.annotation.PostConstruct;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.core.io.Resource;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import com.java.selfdeveloped.graphql.api.dao.PersonRepository;

import com.java.selfdeveloped.graphql.api.entity.Person;

import graphql.ExecutionResult;

import graphql.GraphQL;

import graphql.schema.DataFetcher;

import graphql.schema.GraphQLSchema;

import graphql.schema.idl.RuntimeWiring;

import graphql.schema.idl.SchemaGenerator;

import graphql.schema.idl.SchemaParser;

import graphql.schema.idl.TypeDefinitionRegistry;

@RestController

public class PersonController {

@Autowired

private PersonRepository repository;

@Value("classpath:person.graphqls")

private Resource schemaResource;

private GraphQL graphQL;

@PostConstruct

public void loadSchema() throws IOException {

File schemaFile = schemaResource.getFile();

TypeDefinitionRegistry registry = new SchemaParser().parse(schemaFile);

RuntimeWiring wiring = buildWiring();

GraphQLSchema schema = new SchemaGenerator().makeExecutableSchema(registry, wiring);

graphQL = GraphQL.newGraphQL(schema).build();

}

private RuntimeWiring buildWiring() {

DataFetcher<List<Person>> fetcher1 = data ->{

return (List<Person>) repository.findAll();

};

DataFetcher<Person> fetcher2 = data->{

return repository.findByEmail(data.getArgument("email"));

};

return RuntimeWiring.newRuntimeWiring().type("Query",

typeWriting -> typeWriting.dataFetcher("getAllPerson", fetcher1).dataFetcher("findPerson", fetcher2))

.build();

}

@PostMapping("/addPerson")

public String addPerson(@RequestBody List<Person> persons) {

repository.saveAll(persons);

return "record inserted " + persons.size();

}

@GetMapping("/findAllPerson")

public List<Person> getPersons() {

return (List<Person>) repository.findAll();

}

@PostMapping("/getAll")

public ResponseEntity<Object> getAll(@RequestBody String query) {

ExecutionResult result = graphQL.execute(query);

return new ResponseEntity<Object>(result, HttpStatus.OK);

}

@PostMapping("/getPersonByEmail")

public ResponseEntity<Object> getPersonByEmail(@RequestBody String query) {

ExecutionResult result = graphQL.execute(query);

return new ResponseEntity<Object>(result, HttpStatus.OK);

}

}

Dao

package com.java.selfdeveloped.graphql.api.dao;

import org.springframework.data.repository.CrudRepository;

import com.java.selfdeveloped.graphql.api.entity.Person;

public interface PersonRepository extends CrudRepository<Person, Integer>{

Person findByEmail(String email);

}

**Person**

package com.java.selfdeveloped.graphql.api.entity;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import lombok.Getter;

import lombok.Setter;

@Getter

@Setter

@Entity

public class Person {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private String mobile;

private String email;

private String[] address;

}

Src/main/resource

**person.graphqls**

schema{

query: Query

}

type Query{

getAllPerson : [Person]

findPerson(email: String) : Person

}

type Person{

id: String

name: String

mobile: String

email: String

address: [String]

}

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.13</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.java.selfdeveloped</groupId>

<artifactId>spring-graphQL</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-graphQL</name>

<description>Spring boot with graphql example</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.graphql-java</groupId>

<artifactId>graphql-spring-boot-starter</artifactId>

<version>3.6.0</version>

</dependency>

<dependency>

<groupId>com.graphql-java</groupId>

<artifactId>graphql-java-tools</artifactId>

<version>3.2.0</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>