Spring for GraphQL

October 2022

SHOULD WE REPLACE ALL OF OUR REST APIS WITH GRAPHQL?

IT DEPENDS...

ABOUT ME

- Husband & Father
- Cleveland
- Software Development 20+ Years
- Spring Developer Advocate
- Content Creator (www.danvega.dev)
  - Blogger
  - YouTuber
  - Course Creator
- @therealdanvega

spring

AGENDA

- Why GraphQL?
- What is GraphQL?
- GraphQL Java
- Spring for GraphQL
- Security
- Testing
- Q&A

Spring
for GraphQL

GraphQL Java

spring

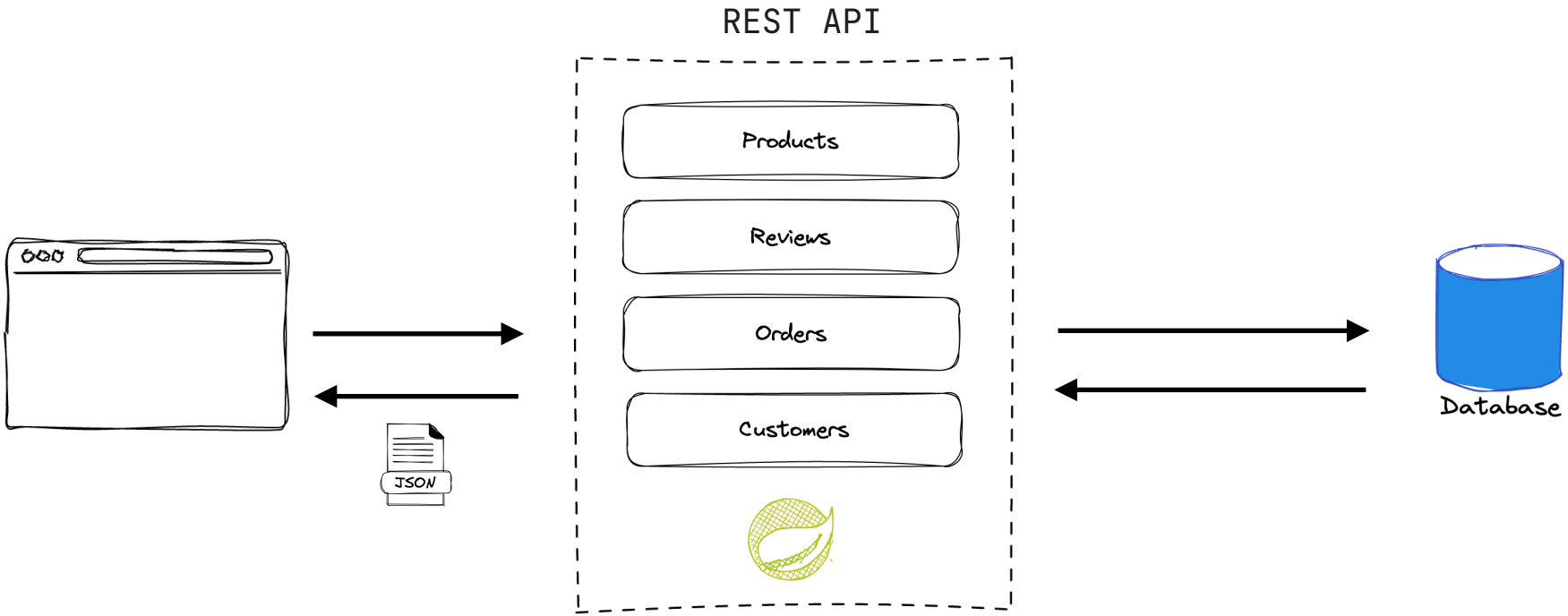
WAFFLECORP

- WaffleCorp Products
    - Direct to Consumer
    - Strategic Partnerships
- WaffleCorp e-commerce service
    - Monolithic Spring Boot MVC
    - REST API & Vue Frontend
- Open up our REST APIs
    - Partner Microservices
    - iOS and Android applications
    - IoT Applications (Smart Toaster)

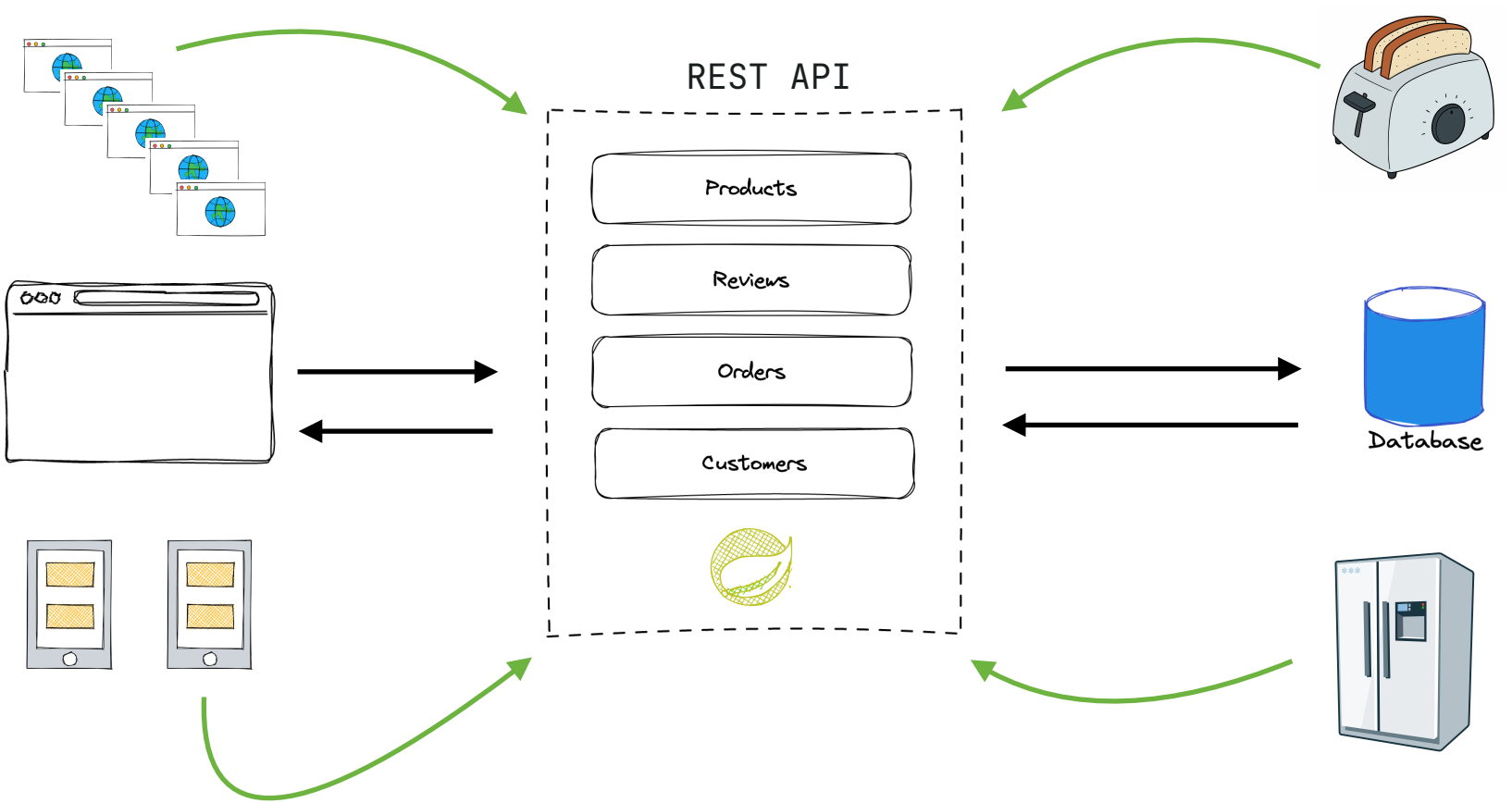
spring

WHY GRAPHQL?

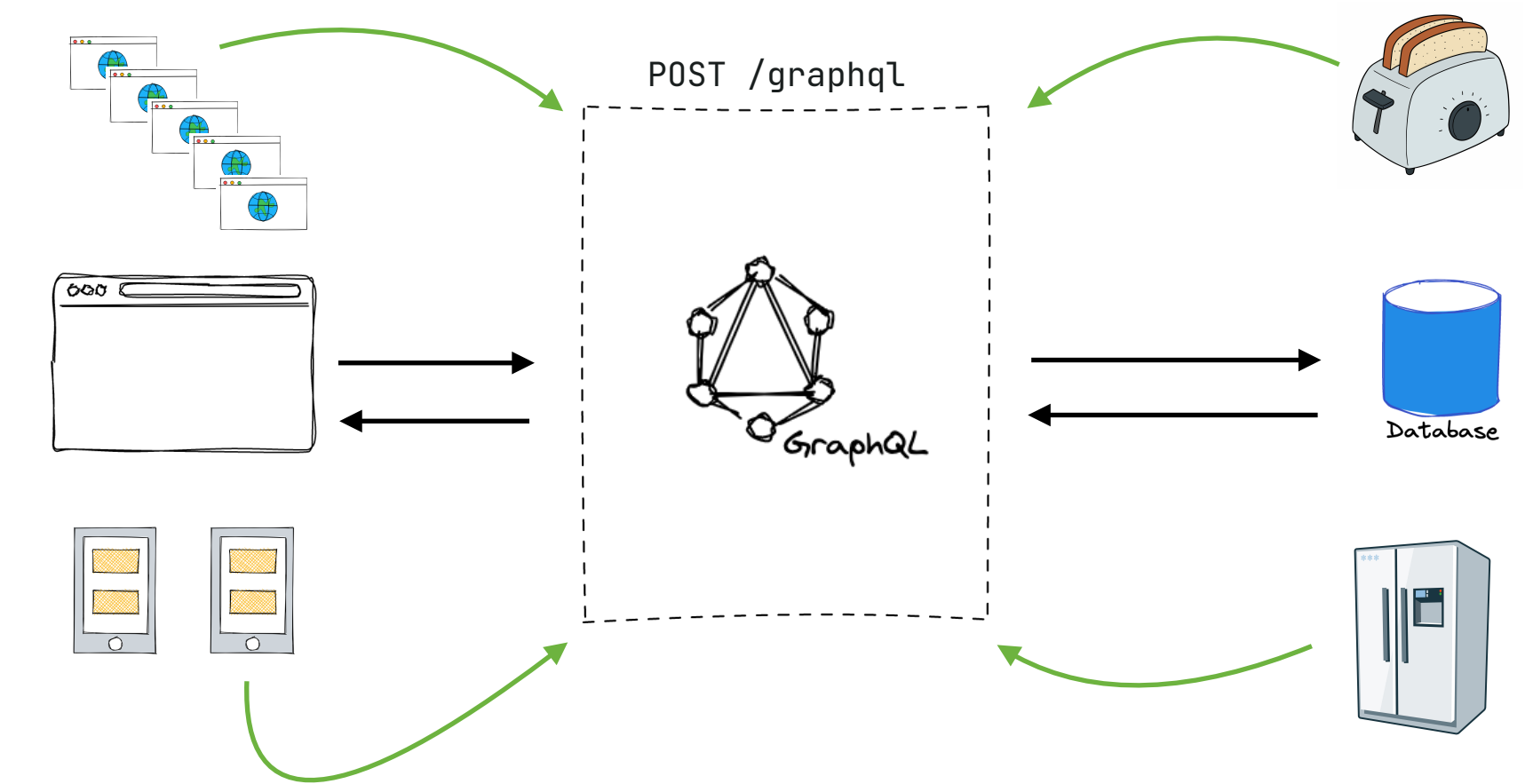
CURRENT ARCHITECTURE



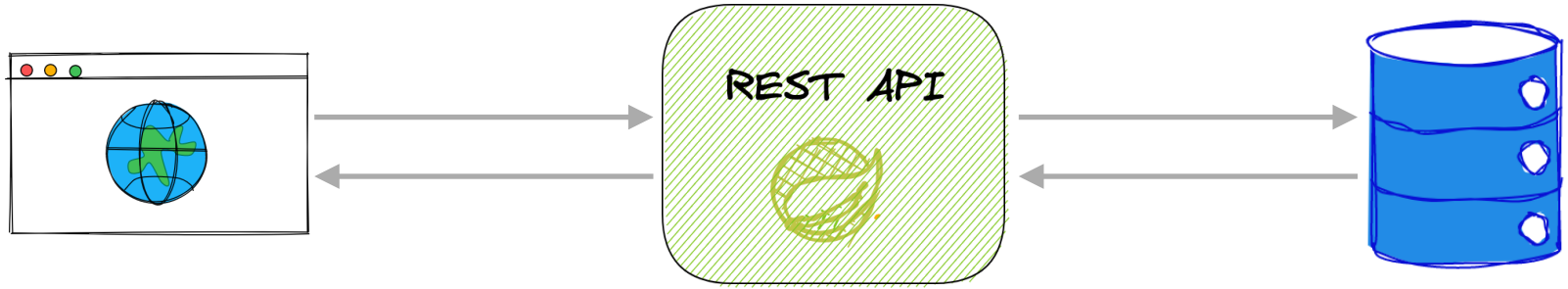
CURRENT ARCHITECTURE



CURRENT ARCHITECTURE



CURRENT ARCHITECTURE



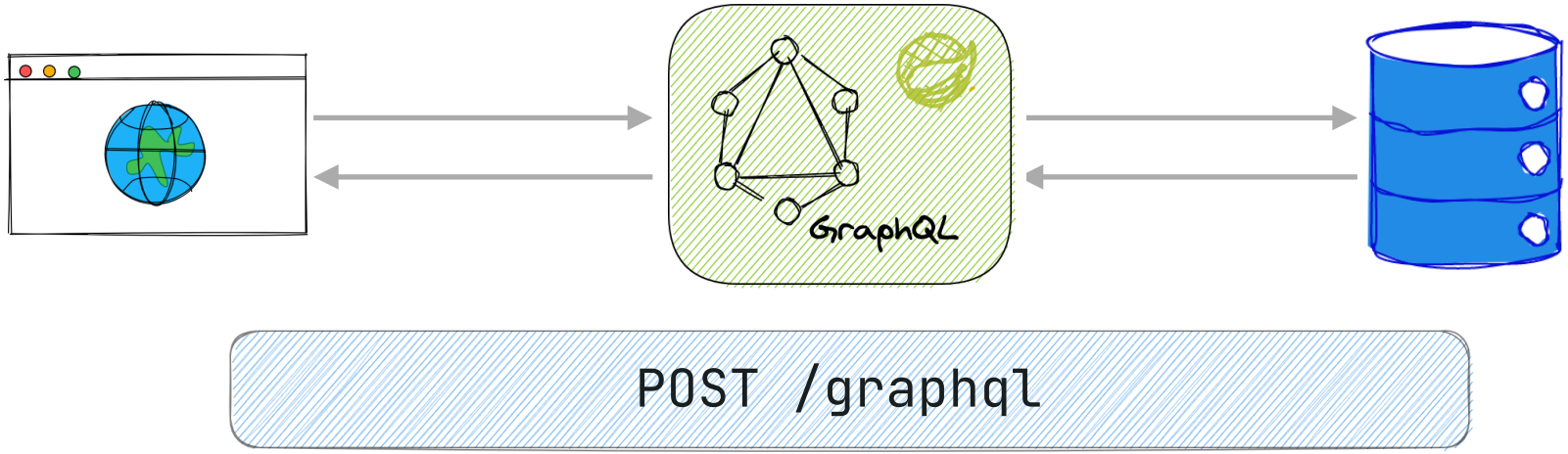
GET /products/123

GET /products/123/related

GET /products/123/reviews

spring

CURRENT ARCHITECTURE



WHAT IS GRAPHQL?

GraphQL is a query language for your API, and a server-side runtime for executing queries using a type system you define for your data. GraphQL **isn't tied to any specific database or storage engine** and is instead backed by your existing code and data.

WHAT IS GRAPHQL?

GraphQL is a technology for client server data exchange

- Alternative to REST APIs
- Provides a really great developer experience
- Created by Facebook in 2012
  - Open Sourced in 2015
  - Governed by a neutral foundation today
- Client + Server: 🤝
  - Client Side Query Language
  - Server Side execution engine

CLIENT + SERVER 🤝

Client

- Custom GraphQL Query Language
- Client crafted query based on their needs
- Responses return only what client requests
- Developer Tooling

Server

- GraphQL Schema defines your API
- Schema Definition Language (SDL) is used to describe a schema.
- Schema based on **simple** static type system
- Implemented in specific languages based on the GraphQL specification.

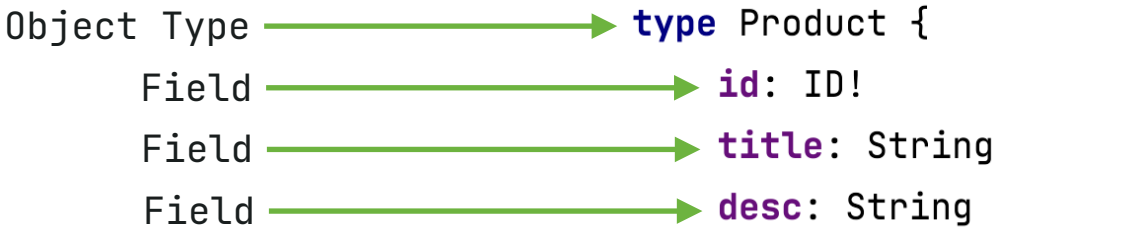
TYPE SYSTEM

```
type Product {
    id: ID!
    title: String
    desc: String
}

type Order {
    id: ID!
    product: Product
    qty: Int
    customer: Customer
    orderedOn: Date
    status: OrderStatus
}
```

Object Type

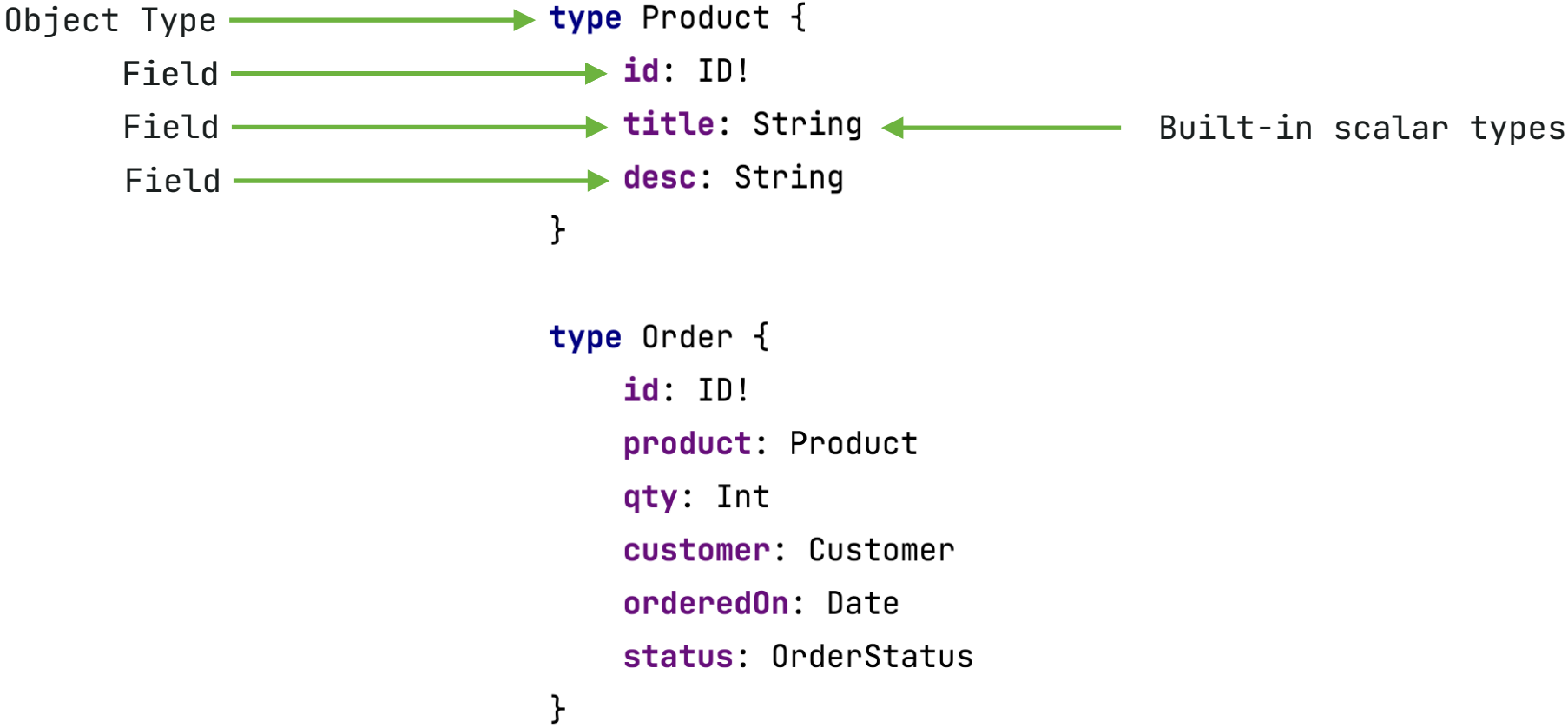
```
type Product {
    id: ID!
    title: String
    desc: String
}

type Order {
    id: ID!
    product: Product
    qty: Int
    customer: Customer
    orderedOn: Date
    status: OrderStatus
}
```

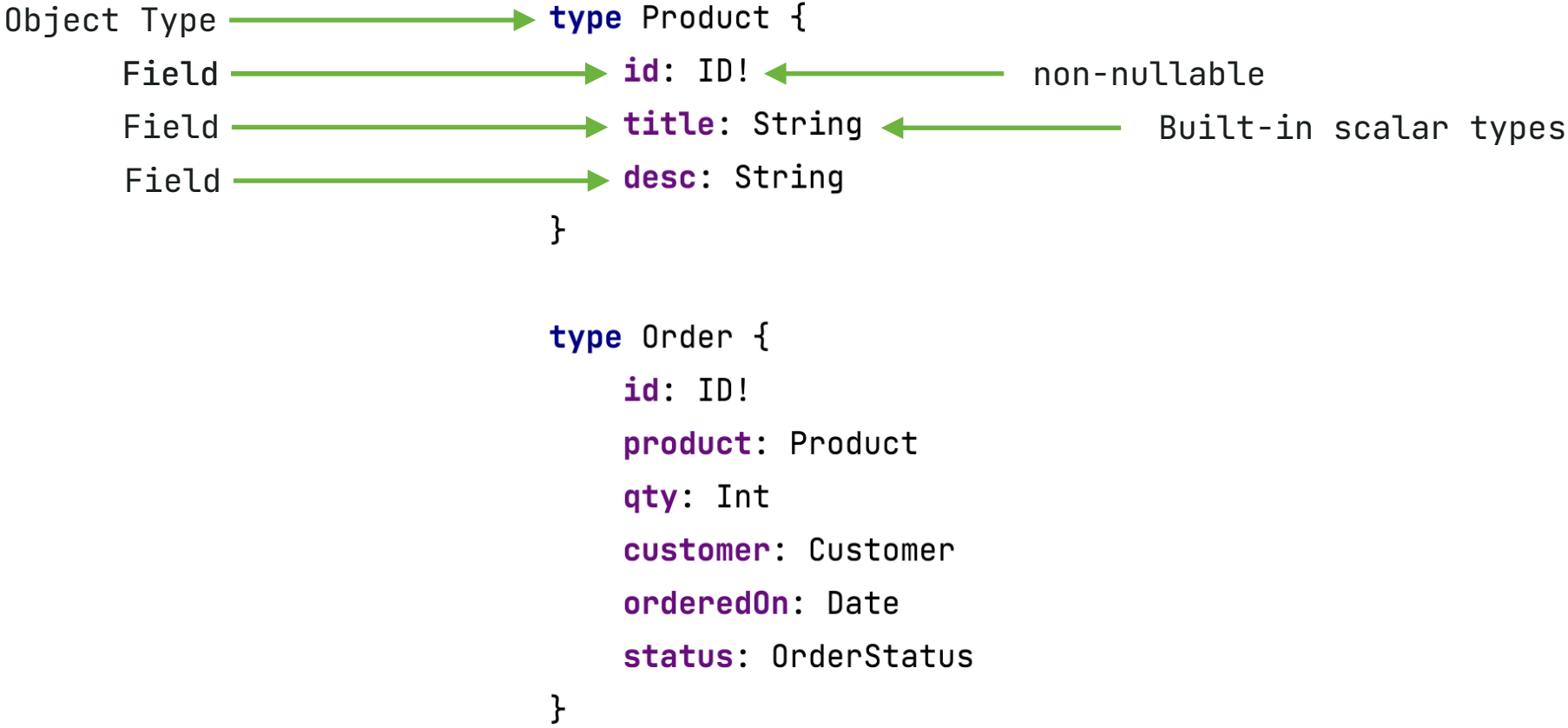
spring



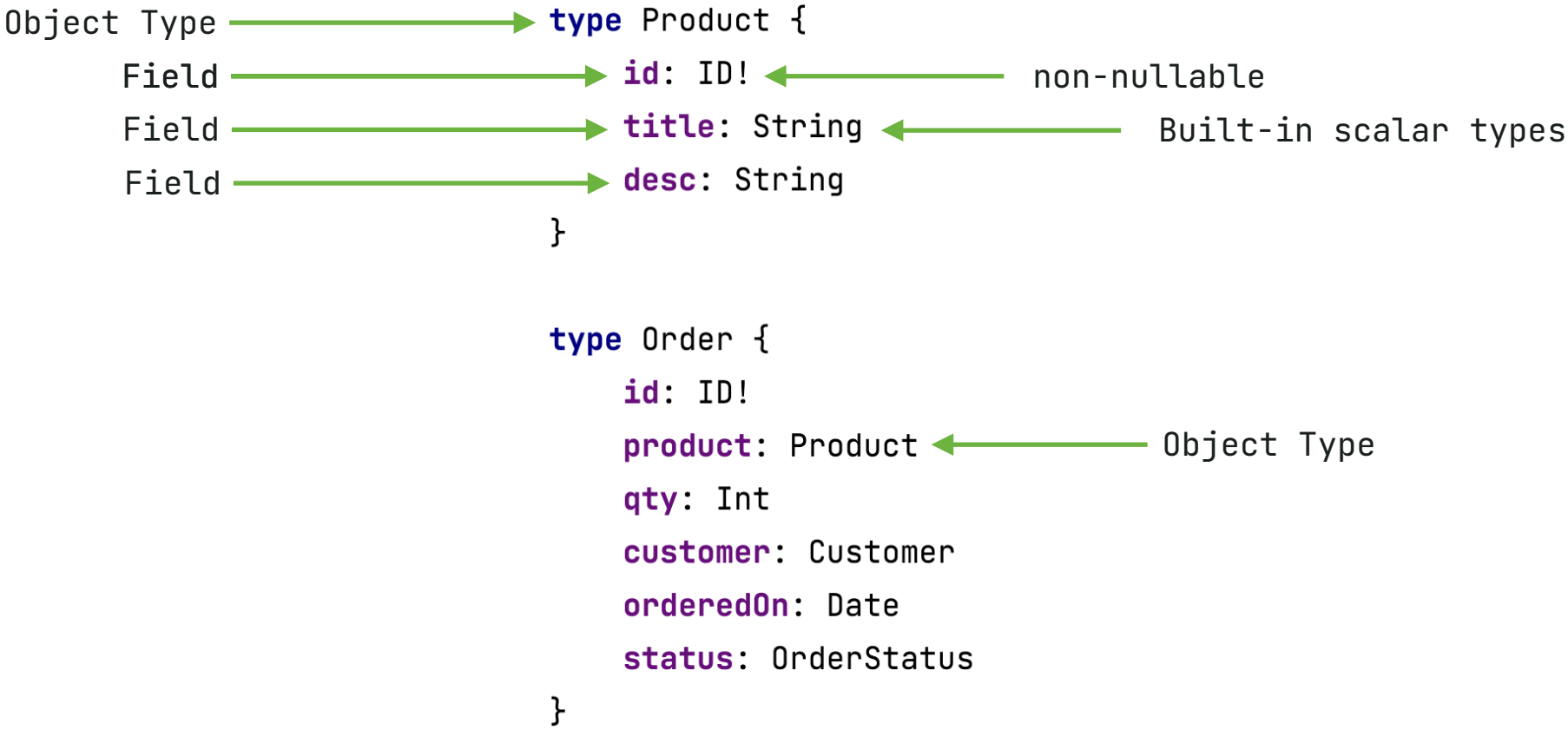

```
type Product {
    id: ID!
    title: String
    desc: String
}

type Order {
    id: ID!
    product: Product
    qty: Int
    customer: Customer
    orderedOn: Date
    status: OrderStatus
}
```

spring

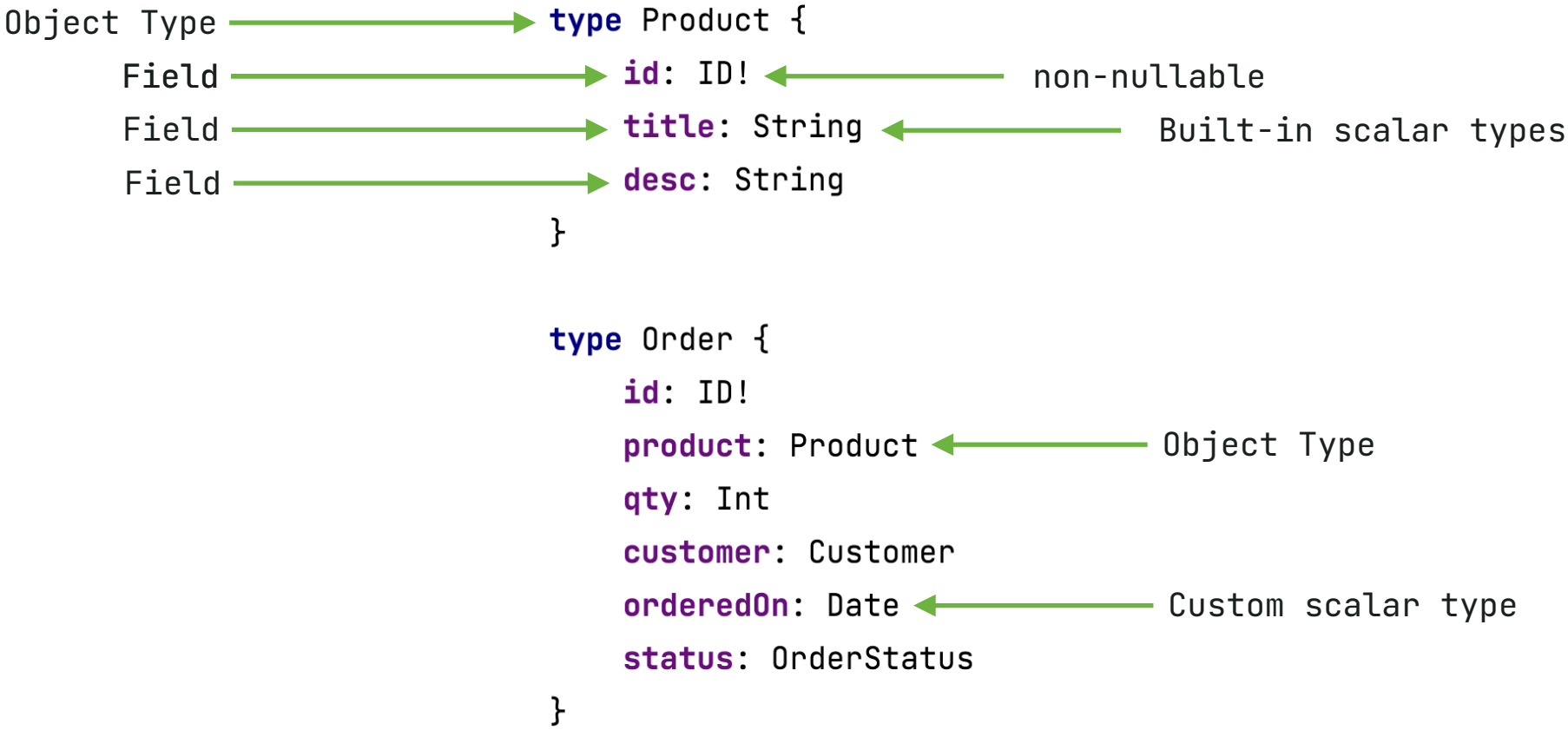

Object Type
Field
Field
Field
type Product {
id: ID!
title: String
desc: String
}
Built-in scalar types
type Order {
id: ID!
product: Product
qty: Int
customer: Customer
orderedOn: Date
status: OrderStatus
}


Object Type
Field
Field
Field
type Product {
id: ID!
title: String
desc: String
}
non-nullable
Built-in scalar types
type Order {
id: ID!
product: Product
qty: Int
customer: Customer
orderedOn: Date
status: OrderStatus
}

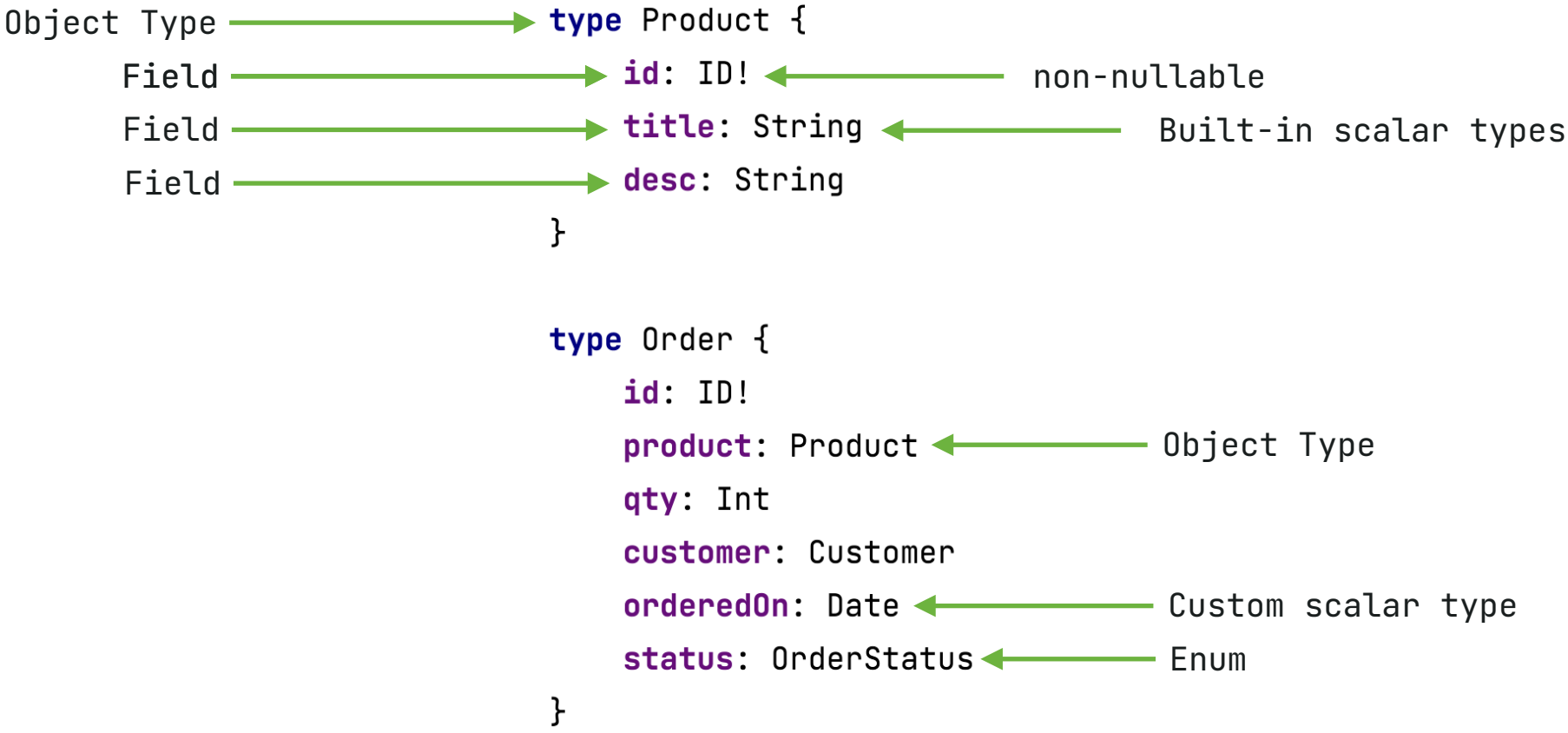
spring


Object Type
Field
Field
Field
type Product {
id: ID!
title: String
desc: String
}
non-nullable
Built-in scalar types
type Order {
id: ID!
product: Product
qty: Int
customer: Customer
orderedOn: Date
status: OrderStatus
}
Object Type

spring


Object Type
Field
Field
Field
type Product {
id: ID!
title: String
desc: String
}
non-nullable
Built-in scalar types
type Order {
id: ID!
product: Product
qty: Int
customer: Customer
orderedOn: Date
status: OrderStatus
}
Object Type
Custom scalar type

spring


Object Type
Field
Field
Field
type Product {
id: ID!
non-nullable
title: String
Built-in scalar types
desc: String
}
type Order {
id: ID!
product: Product
Object Type
qty: Int
customer: Customer
orderedOn: Date
Custom scalar type
status: OrderStatus
Enum
}

spring

GRAPHQL SCHEMA: SCALAR TYPES

GraphQL comes with a set of default scalar types out of the box:

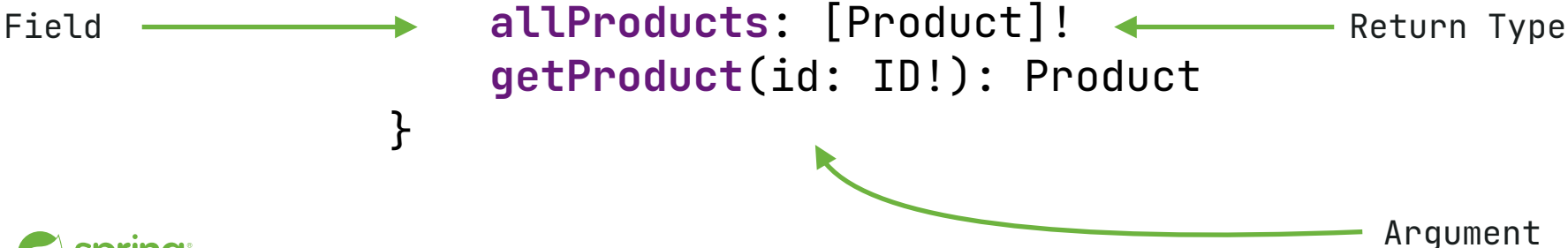
- `Int`: A signed 32-bit Integer.
- `Float`: A signed double-precision floating point value.
- `String`: A UTF-8 character sequence.
- `Boolean`: true or false.
- `ID`: The ID scalar type represents a unique identifier, often used to refetch an object or as the key for a cache. The ID type is serialized in the same way as a String; however, defining it as an `ID` signifies that it is not intended to be human-readable.

OPERATION TYPES

Most types in your schema will just be normal object types, but there are three types that are special within a schema:

```
type Product {
    id: ID!
    title: String
    desc: String
}

type Query {
    allProducts: [Product]!
    getProduct(id: ID): Product
}
```

```
type Product {
    id: ID!
    title: String
    desc: String
}

type Query {
    allProducts: [Product]!
    getProduct(id: ID!): Product
}
```



spring

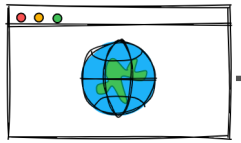
GRAPHIQL DEMO

GRAPHQL JAVA

GRAPHQL JAVA

GraphQL Java is the GraphQL implementation for Java

- Started mid 2015
  - Andreas Marek, Software Architect at Atlassian
  - Bootiful Podcast Interview with Andreas
  - Spring Tips
- Pure Execution Engine: No HTTP or IO. No high level abstractions
- Active on Github
- Adds Custom Scalar Types
- https://www.graphql-java.com/



Client

```
query {
  findCustomerByLastName(last:"Vega"){
    id
    firstName
    lastName
    email
    orders {
      id
      product {
        id
        title
        desc
      }
      qty
      orderedOn
      status
    }
  }
}
```

GraphQL Java Engine

Data Fetcher

Data Fetcher

Data Fetcher

Data Fetcher

Data Fetcher

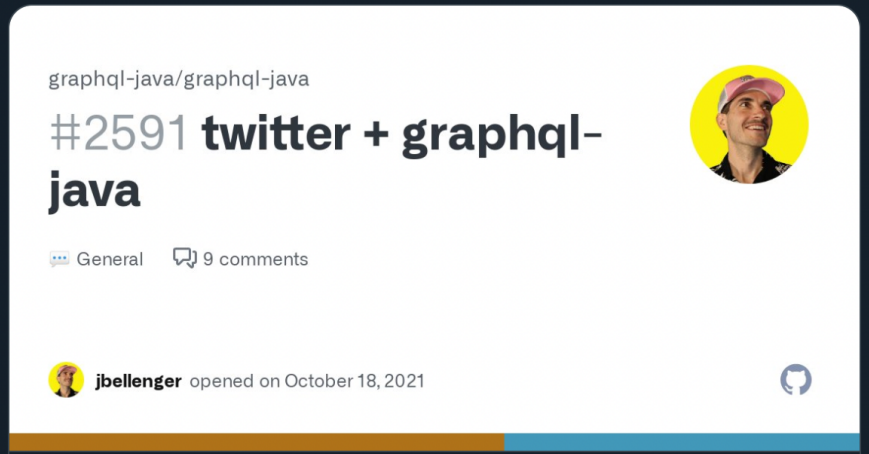
Application

spring

← Tweet

GraphQL Java
@graphql_java

Twitter is now powered by #GraphQL #Java. Serving 500k requests per second. github.com/graphql-java/g…
Thanks a lot for the public feedback @Twitter @jbell



github.com
twitter + graphql-java · Discussion #2591 · graphql-java/graphql-java
Hi team! Twitter is wrapping up its migration to graphql-java and I wanted to share some info about how we use graphql and why we chose to migrate our …

4:24 PM · Oct 18, 2021 · Twitter for iPhone

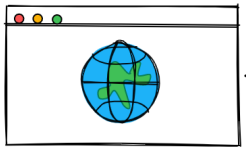
spring

SPRING FOR GRAPHQL

WHAT IS SPRING FOR GRAPHQL?

Spring for GraphQL provides support for Spring applications built on GraphQL Java.

- Joint collaboration between both teams
- GraphQL Java is “limited” on purpose
- Spring Projects - Make complex problems easy
- Requirements
  - JDK8
  - Spring Boot 2.7+
- Current Version: 1.0.2



Client

```
query {
  findCustomerByLastName(last:"Vega"){
    id
    firstName
    lastName
    email
    orders {
      id
      product {
        id
        title
        desc
      }
      qty
      orderedOn
      status
    }
  }
}
```

Single HTTP Endpoint

GraphQL Java Engine

Data Fetcher

Data Fetcher

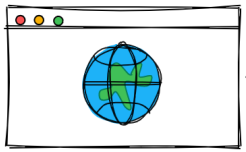
Data Fetcher

Data Fetcher

Data Fetcher

Application

spring



Client

```
query {
  findCustomerByLastName(last:"Vega"){
    id
    firstName
    lastName
    email
    orders {
      id
      product {
        id
        title
        desc
      }
      qty
      orderedOn
      status
    }
  }
}
```

WebSocket Endpoint

GraphQL Java Engine

Data Fetcher

Data Fetcher

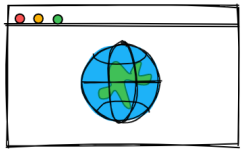
Data Fetcher

Data Fetcher

Data Fetcher

Application

spring



Client

```
query {
  findCustomerByLastName(last:"Vega"){
    id
    firstName
    lastName
    email
    orders {
      id
      product {
        id
        title
        desc
      }
      qty
      orderedOn
      status
    }
  }
}
```

RSocket Endpoint

WebSocket, TCP

GraphQL Java Engine

Data Fetcher

Data Fetcher

Data Fetcher

Data Fetcher

Data Fetcher

Application

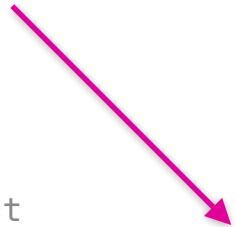
spring

```
type Query {
    allProducts: [Product]!
    getProduct(id: ID!): Product
}
```

```
@Controller
public class ProductController {

    private final ProductRepository repository;

    public ProductController(ProductRepository repository) {
        this.repository = repository;
    }

    @SchemaMapping(typeName = "Query", value = "allProducts")
    public Iterable<Product> allProducts() {
        return repository.findAll();
    }

    @QueryMapping
    public Optional<Product> getProduct(@Argument Integer id) {
        return repository.findById(id);
    }

}
```

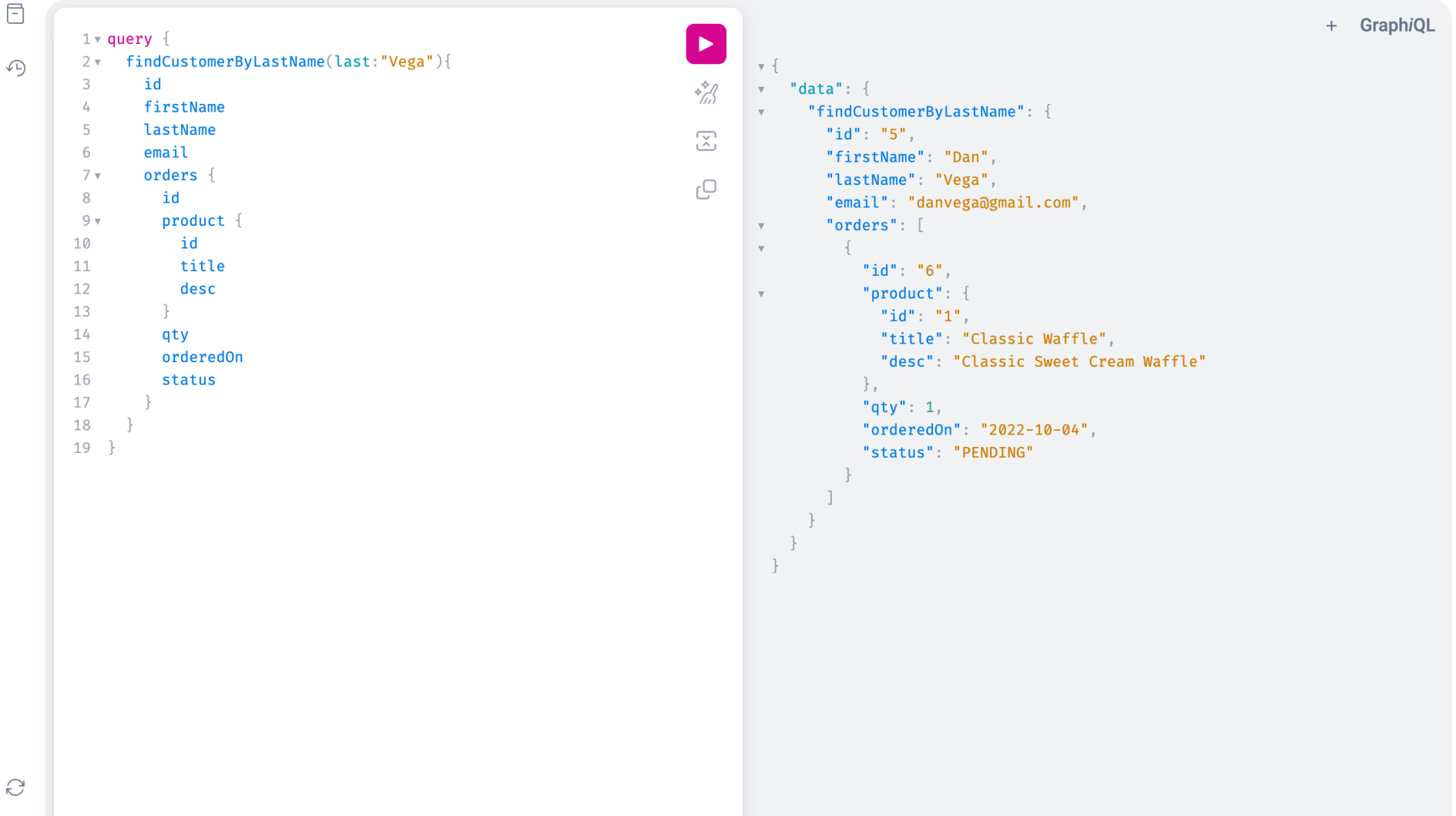
```
type Customer {
    id: ID!
    firstName: String
    lastName: String
    email: String
    orders: [Order]
}

type Order {
    id: ID!
    product: Product
    qty: Int
    customer: Customer
    orderedOn: Date
    status: OrderStatus
}
```

```
type Customer {
    id: ID!
    firstName: String
    lastName: String
    email: String
    orders: [Order]
}

type Order {
    id: ID!
    product: Product
    qty: Int
    customer: Customer
    orderedOn: Date
    status: OrderStatus
}
```



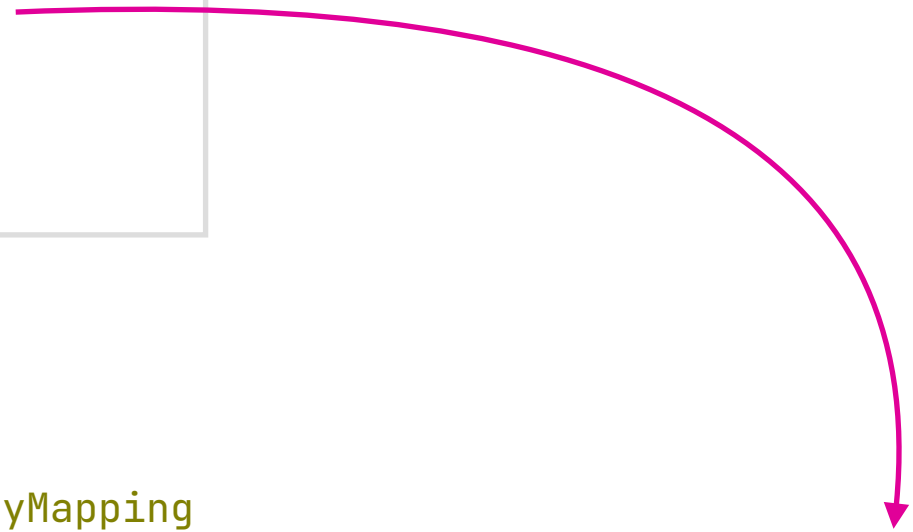
```
@QueryMapping
public List<Customer> allCustomers() {
    return repository.findAll();
}

@QueryMapping
public Customer findCustomerByLastName(@Argument String last) {
    return repository.findByLastName(last);
}

@SchemaMapping
public List<Order> orders(Customer customer) {
    return orderRepository.findByCustomer(customer);
}
```

spring


GraphiQL
query {
findCustomerByLastName(last:"Vega"){
id
firstName
lastName
email
orders {
id
product {
id
title
desc
}
qty
orderedOn
status
}
}
}
{
"data": {
"findCustomerByLastName": {
"id": "5",
"firstName": "Dan",
"lastName": "Vega",
"email": "danvega@gmail.com",
"orders": [
{
"id": "6",
"product": {
"id": "1",
"title": "Classic Waffle",
"desc": "Classic Sweet Cream Waffle"
},
"qty": 1,
"orderedOn": "2022-10-04",
"status": "PENDING"
}
]
}
}
}

GRAPHQL REQUEST ARGUMENTS

```
query {
  getProduct(id:1){
    id
    title
    desc
  }
```



```
@QueryMapping
public Optional<Product> getProduct(@Argument Integer id) {
    return repository.findById(id);
}
```

COMPLEX GRAPHQL ARGUMENTS

```
query {
  findOrders(criteria:{orderedOn:"10/05/2022",status:"PENDING"}) {
    id
    product {
      id
      title
      desc
    }
    qty
    orderedOn
    status
  }
}
```

```
@QueryMapping
public List<Order> findOrders(@Argument OrderCriteria criteria) {
    // ...
}
```

SUPPORTS VALIDATION

```
@QueryMapping
public List<Order> findOrders(@Argument @Valid OrderCriteria criteria) {
    // ...
}
```

Project

- (●) Maven Project
- ( ) Gradle Project

Language

- (●) Java
- ( ) Kotlin
- ( ) Groovy

Spring Boot

- ( ) 3.0.0 (SNAPSHOT)
- ( ) 3.0.0 (M3)
- ( ) 2.7.1 (SNAPSHOT)
- (●) 2.7.0
- ( ) 2.6.9 (SNAPSHOT)
- ( ) 2.6.8

Project Metadata

Group	dev.danvega
Artifact	graphql-store
Name	graphql-store
Description	GraphQL Demo project for Spring Boot
Package name	dev.danvega.graphql-store
Packaging	(●) Jar ( ) War
Java	( ) 18 (●) 17 ( ) 11 ( ) 8

Dependencies

ADD DEPENDENCIES... ⌘ + B

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring for GraphQL WEB

Build GraphQL applications with Spring for GraphQL and GraphQL Java.

GENERATE ⌘ + ↵ | EXPLORE CTRL + SPACE | SHARE...

SPRING BOOT STARTER

The Spring Boot Starter Graphql (spring-boot-starter-graphql)

- Dependency Management
- Auto-configuration
- Properties
- Metrics
- GraphiQL UI and Schema Pages

SPRING FOR GRAPHQL DEMO

https://github.com/danvega/graphql-store

spring

SECURITY

```java
@Configuration
@EnableWebSecurity
@EnableMethodSecurity(securedEnabled = true)
public class SecurityConfig {

    @Bean
    public SecurityFilterChain configure(HttpSecurity http) throws Exception {
        return http
                .csrf(csrf -> csrf.disable())
                .authorizeRequests( auth -> {
                    auth.anyRequest().authenticated();
                })
                .sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
                .httpBasic(withDefaults())
                .build();
    }
```

```java
@Secured("ROLE_USER")
@QueryMapping
public List<Coffee> allCoffee() {
    return coffeeService.findAll();
}

@PreAuthorize("hasRole('ADMIN')")
@MutationMapping
public Coffee createCoffee(@Argument String name, @Argument Size size) {
    return coffeeService.create(name,size);
}
```

```java
@PreAuthorize("hasRole('ADMIN')")
@MutationMapping
public Coffee createCoffee(@Argument String name, @Argument Size size) {
    return coffeeService.create(name,size);
}
```

```java
@Service
public class CoffeeService {

    private List<Coffee> coffees = new ArrayList<>();

    @PreAuthorize("hasRole('ADMIN')")
    public Coffee create(String name, Size size) {
        Coffee coffee = new Coffee(id.incrementAndGet(), name, size);
        coffees.add(coffee);
        return coffee;
    }
```

```java
@Controller
public class OrderController {

    private final OrderService orderService;

    public OrderController(OrderService orderService) {
        this.orderService = orderService;
    }

    @QueryMapping
    public List<Order> findAllOrders(Principal principal) {
        return orderService.findAllByUsername(principal.getName());
    }
}
```

TESTING SUPPORT

AUTO-CONFIGURED SPRING GRAPHQL TESTS

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

GRAPHQL SLICE TEST

```java
@GraphQlTest(ProductController.class)
class ProductControllerTest {

    @Autowired
    private GraphQlTester graphQlTester;

    @MockBean
    private ProductRepository productRepository;

    private List<Product> products;

    // ...
}
```

spring

```java
@Test
void shouldGetAllProductsQueryReturnsAllProducts() {
    // language=GraphQL
    String document = """
        query {
          allProducts {
            id
            title
            desc
          }
        }
    """;

    when(productRepository.findAll()).thenReturn(products);

    graphQlTester.document(document)
            .execute()
            .path("allProducts")
            .entityList(Product.class)
            .hasSize(4);
}
```

```
query findProduct($id:ID) {
    getProduct(id:$id) {
        id
        title
        desc
    }
}
```

src/main/resources/graphql/product.test

```
@Test
void shouldReturnValidProductWithDocumentName() {
    when(productRepository.findById(1)).thenReturn(java.util.Optional.ofNullable(products.get(0)));

    graphQlTester.documentName("product")
            .variable("id",1)
            .execute()
            .path("getProduct")
            .entity(Product.class)
            .satisfies(product -> {
                assertEquals("Classic Waffle",product.getTitle());
                assertEquals("Classic Sweet Cream Waffle",product.getDesc());
            });
}
```

TEST OVER HTTP

MockMVC or WebTestClient
No Server

```java
@SpringBootTest
@AutoConfigureHttpGraphQlTester
public class ProductControllerHttpTest {

    @Autowired
    private GraphQlTester graphQlTester;

    @Test
    void shouldFindAllProducts() {
        graphQlTester.documentName("products")
                .execute()
                .path("allProducts")
                .entityList(Product.class)
                .hasSize(4);
    }

}
```

Test over HTTP

MockMVC or WebTestClient

No server

```
@SpringBootTest
@AutoConfigureHttpGraphQlTester
public class UserTests {

    @Autowired
    private GraphQlTester graphQlTester;

    // ...

}
```

WHAT'S NEXT?

What didn't we have a chance to talk about today?

- GraphQL Challenges (not a silver bullet)
- Subscriptions
- Server Transports
  - WebSocket
  - RSocket
- Operation Caching
- Batch Loading (N+1)
- GraphQL Client
- API Versioning
- Spring Data Integration (QueryDSL,QueryByExample)

Thank you

Contact me at dvega@vmware.com

@therealdanvega