Netflix DGS Framework для побудови GraphQL API. Порівняння з Spring for GraphQL

Стельмащук Віталій Володимирович

Львівський національний університет імені Івана Франка
Кафедра інформаційних систем

3 березня 2025 р.

Outline

Netflix DGS Framework

- Netflix DGS (Domain Graph Service) Framework allows for easy development of GraphQL API using JVM-based programming languages (Java, Kotlin)
- Built on top of Spring Boot (Spring Core, MVC, WebFlux, etc.)
- Written in Kotlin
- Developed by Netflix, used firstly only in Netflix internally
- In 2021 Netflix open-sourced its implementation and since then it became widely adopted by many companies, since at that time it was the only framework that allowed creation production-ready GraphQL APIs in Java
- Maintained by Netflix

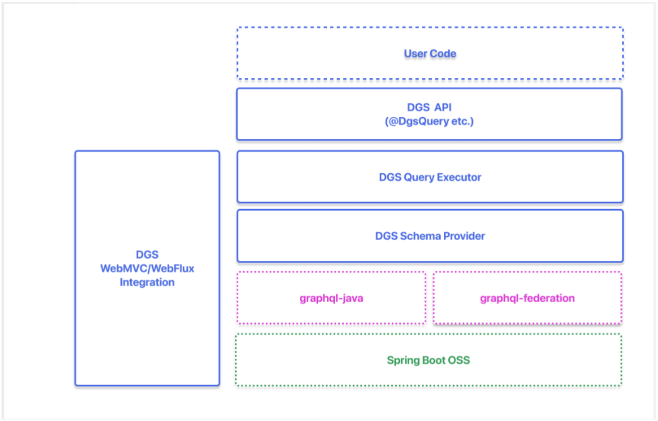
Netflix DGS Framework

- Based on Spring Boot 3, requires at least Java 17
- A competing framework to Spring for GraphQL
- DGS framework follows a **schema-first** development approach. Schema files should be placed in the `src/main/resources/schema` folder.
- Code generation capabilites
- **GraphiQL** tool is available at `/graphiql`
- Getting Started Guide
  Use Spring Initializr and include the Netflix DGS, Spring Web (or Spring Reactive Web), and optionally GraphQL DGS Code Generation dependencies. Gradle or Maven can be used with Java 17 or Kotlin.
- Example (Creating project using Spring Initializr)

Relations with Spring for GraphQL

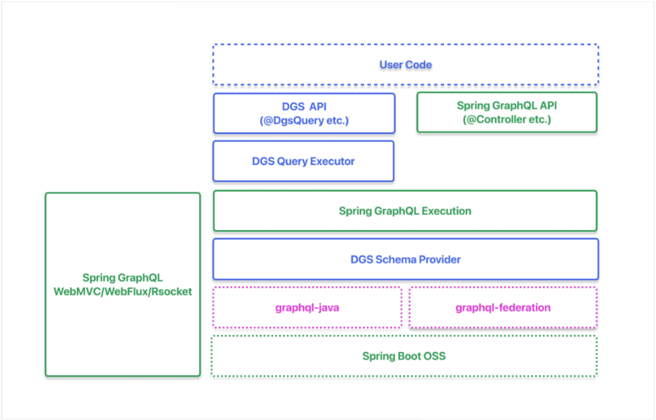
- In 2021 Netflix DGS was the only framework to develop production-ready GraphQL APIs in Java ecosystem.
- After Netflix DGS has been open-sourced in 2021, the Spring team began developing its own GraphQL framework for Spring Boot.
- In its early stages, the Spring for GraphQL project provided a low-level of integration with graphql-java. Over time, Spring for GraphQL has matured to reach feature parity with the DGS Framework.
- DGS Framework now integrates with Spring for GraphQL internally. This integration allows users to leverage features from Spring for GraphQL within the DGS framework.
- **Reasons for not abandoning either framework**: The DGS framework is widely used, including at Netflix, making a migration to Spring-GraphQL costly without significant benefits. Spring Framework benefits from having an out-of-the-box GraphQL offering, similar to its support for REST.

Netflix DGS Framework. Technical implementation



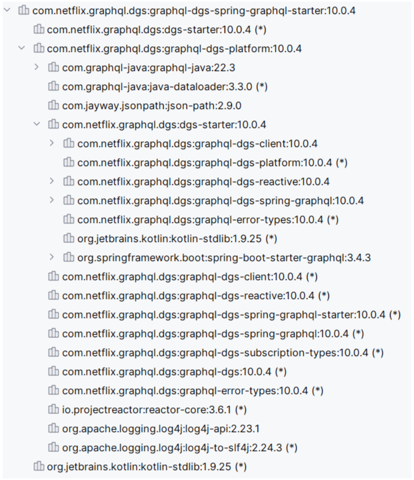
DGS Documentation. Spring GraphQL integration

Netflix DGS Framework. Technical implementation. Spring for GraphQL integration



DGS Documentation. Spring GraphQL integration

Netflix DGS Framework. Dependency tree of the starter



```
com.netflix.graphql.dgs:graphql-dgs-spring-graphql-starter:10.0.4
    com.netflix.graphql.dgs:dgs-starter:10.0.4 (*)
    com.netflix.graphql.dgs:graphql-dgs-platform:10.0.4
        com.graphql-java:graphql-java:22.3
        com.graphql-java:java-dataloader:3.3.0 (*)
        com.jayway.jsonpath:json-path:2.9.0
        com.netflix.graphql.dgs:dgs-starter:10.0.4
            com.netflix.graphql.dgs:graphql-dgs-client:10.0.4
            com.netflix.graphql.dgs:graphql-dgs-platform:10.0.4 (*)
            com.netflix.graphql.dgs:graphql-dgs-reactive:10.0.4
            com.netflix.graphql.dgs:graphql-dgs-spring-graphql:10.0.4
            com.netflix.graphql.dgs:graphql-error-types:10.0.4 (*)
            org.jetbrains.kotlin:kotlin-stdlib:1.9.25 (*)
            org.springframework.boot:spring-boot-starter-graphql:3.4.3
        com.netflix.graphql.dgs:graphql-dgs-client:10.0.4 (*)
        com.netflix.graphql.dgs:graphql-dgs-reactive:10.0.4 (*)
        com.netflix.graphql.dgs:graphql-dgs-spring-graphql-starter:10.0.4 (*)
        com.netflix.graphql.dgs:graphql-dgs-spring-graphql:10.0.4 (*)
        com.netflix.graphql.dgs:graphql-dgs-subscription-types:10.0.4 (*)
        com.netflix.graphql.dgs:graphql-dgs:10.0.4 (*)
        com.netflix.graphql.dgs:graphql-error-types:10.0.4 (*)
        io.projectreactor:reactor-core:3.6.1 (*)
        org.apache.logging.log4j:log4j-api:2.23.1
        org.apache.logging.log4j:log4j-to-slf4j:2.24.3 (*)
    org.jetbrains.kotlin:kotlin-stdlib:1.9.25 (*)
```

Netflix DGS Framework. Request execution

- Spring for GraphQL‘s `ExecutionGraphQlService` handles actual query execution
- DGS‘s `DgsQueryExecutor` is now a proxy on top of `ExecutionGraphQlService`

Netflix DGS Framework. Data fetching annotations

To make a method a data fetcher we can use @DgsData annotation in a class that is marked with @DgsComponent annotation.

```java
@DgsComponent
public class ShowDataFetcher {

        @DgsData(parentType = "Query", field = "shows")
        public List<Show> shows() {

                //Load shows from a database and return
                ↪  the list of Show objects
                return shows;
        }
}
```

The @DgsQuery, @DgsMutation and @DgsSubscription annotations are shorthands to define datafetchers on the Query, Mutation and Subscription types.

Netflix DGS Framework. Child data fetchers

Separate (child) data fetcher for expensive field:

```java
@DgsQuery
public List<Show> shows() {

        //Load shows, which doesn't include "actors"
        return shows;
}

@DgsData(parentType = "Show", field = "actors")
public List<Actor> actors(DgsDataFetchingEnvironment dfe)
↪ {

        Show show = dfe.getSource();

        return actorsService.forShow(show.getId());
}
```

`DgsDataFetchingEnvironment` encapsulates graphql-java‘s `DataFetchingEnvironment` and provides a context info for current GraphQL operation.

Netflix DGS Framework. Code generation

The DGS Code Generation plugin generates code during your project's build process based on your Domain Graph Service's GraphQL schema file. The plugin generates the following:

- Data types for types, input types, enums and interfaces.
- A `DgsConstants` class containing the names of types and fields
- Example data fetchers
- A type safe query API that represents your queries

DGS Framework Documentation. Code generation

Netflix DGS Framework. Code generation plugin configuration

It‘s a Gradle plugin. In `build.gradle` file the following configuration is needed:

```
plugins {
        id "com.netflix.dgs.codegen" version
        ↪  "[PLUGIN_VERSION]"
}
```

```
generateJava{
        schemaPaths =
        ↪  ["${projectDir}/src/main/resources/schema"]
        packageName = 'com.example.packagename' // The
        ↪  package name to use to generate sources
        generateClient = true // Enable generating the
        ↪  type safe query API
}
```

Comparison with Spring for GraphQL

Advantages of Netflix DGS Framework:

- Built-in Apollo Federation support
- Built-in support of code generation (including client code)
- Easier to write unit/integration tests
- Easier due to simpler API and annotations

Advantages of Spring for GraphQL:

- Seamless Spring Security integration
- Part of official Spring ecosystem, strong Spring community

Example

- Netflix DGS example
- Example project on Netflix‘s GitHub

Thank you for your attention!

Questions?