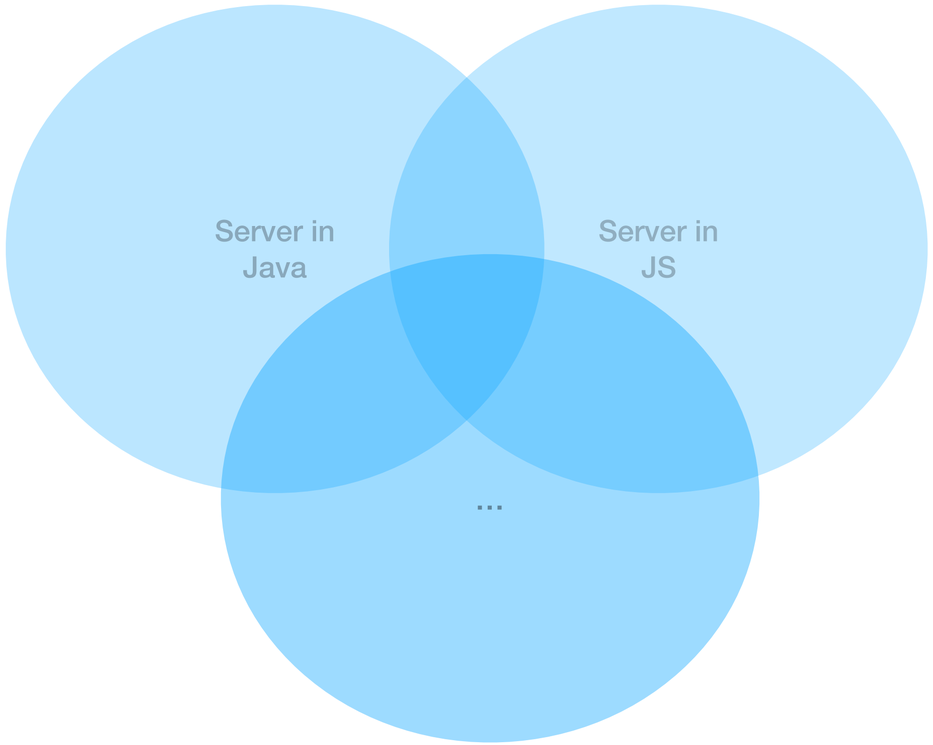
Introduction to GraphQL with Java

Andi Marek

Agenda

- A bit GraphQL
- How to implement a GraphQL server in Java



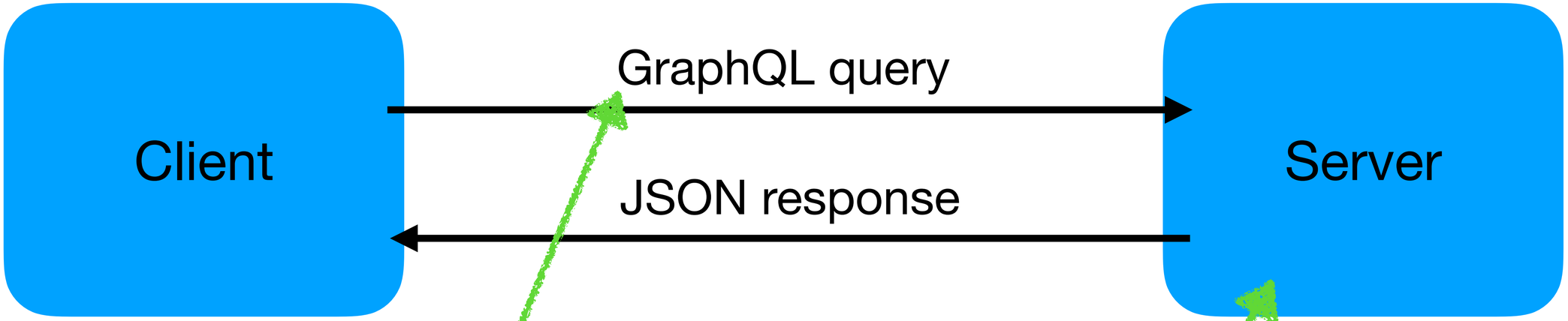
Andi Marek

@andimarek

@graphql_java

ATLASSIAN

What is GraphQL?



GraphQL is a API query language

AND

a server-side runtime for executing queries

GraphQL Playground Demo

All requests go to one URL (POST)

Errors are part of the response.
HTTP status codes are not used.

Clients define the query

You only get the data you ask for

Static type system describes the API

Our example: book details

```
{
  bookById(id: "book-2"){
    id
    name
    pageCount
    author {
      firstName
      lastName
    }
  }
}
```

```
{
  "data": {
    "bookById": {
      "id": "book-2",
      "name": "Moby Dick",
      "pageCount": 635,
      "author": {
        "firstName": "Herman",
        "lastName": "Melville"
      }
    }
  }
}
```

Basics

- We will use Spring Boot
- Full source code for this example app including a tutorial is available online

We need two things

- Schema: Defines how our API looks like
- DataFetchers: Brings the API to life by providing the actual data (also called Resolvers)

Define the schema

- You model your API in GraphQL with a static schema
- Schema design is a subject on its own
- We define “Book” and an “Author” type

Book details schema

```
type Query {
    bookById(id: ID): Book
}

type Book {
    id: ID
    name: String
    pageCount: Int
    author: Author
}

type Author {
    id: ID
    firstName: String
    lastName: String
}
```

GraphQL schema creation code

DataFechters

- A DataFetcher fetches the Data for a field
- If you don’t define a DataFetcher “PropertyDataFetcher” is used
- The data can come from everywhere: another service, database, static values etc

Our DataFetchers

```
type Query {
    bookById(id: ID): Book
}

type Book {
    id: ID
    name: String
    pageCount: Int
    author: Author
}

type Author {
    id: ID
    firstName: String
    lastName: String
}
```

→ BookByIdDataFetcher

→ PropertyDataFetcher
→ PropertyDataFetcher
→ PropertyDataFetcher
→ AuthorDataFetcher

→ PropertyDataFetcher
→ PropertyDataFetcher
→ PropertyDataFetcher

DataFetcher interface

```
@PublicSpi
public interface DataFetcher<T> {

    T get(DataFetchingEnvironment environment) throws Exception;

}
```

Our example DataFetchers

- Our DataFetchers are as simple as possible
- We get data from static values
- Good for this introduction but in real life you call another service or database

DataFetcher implementation code

Lets change our API

```
type Query {
    bookById(id: ID): Book
}

type Book {
    id: ID
    name: String
    pageCount: Int
    author: Author
}

type Author {
    id: ID
    firstName: String
    lastName: String
}
```

```
type Query {
    bookById(id: ID): Book
}

type Book {
    id: ID
    title: String
    pageCount: Int
    author: Author
}

type Author {
    id: ID
    firstName: String
    lastName: String
}
```

DataFetchers maps the API onto your datasources

Thank you

Tutorial with source code:
https://www.graphql-java.com/tutorials/getting-started-with-spring-boot/