Agenda

- Introduction to GraphQL
- GraphQL Tradeoffs (vs ReST-like APIs)
- GraphQL SDL Need-to-Know
- Show in Spring Boot: Introduction
  - Application Configuration
  - Query, Schema Definition
  - Minimum Spring Boot Application
  - GraphQLQueryResolver

Part 2 (followup: not today)

- Resolvers: Field-level data fetchers demonstration (Image)
- Integration to Repositories (local sources)
- Integration to REST APIs sources (external sources)

GraphQL is...

a query language

designed to provide an intuitive and flexible syntax

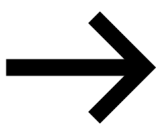
to query application servers that expose data shapes defined to this specification

A GraphQL query and schema...

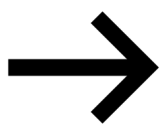
```
schema {
    query: Query
}

type Query {
    cars(): [Car!]!
}

type Car {
    vin: ID!,
    year: Int!
    make: String!
    model: String!
}
```



```
{
  cars {
    vin
    model
  }
}
```



schema

query

result

GraphQL *Comparison* (gql API vs JSON/HTTP API)

Strengths

- API describes how to query a graph of data using a query language spec
- Allows selection of exactly what the caller requires
- Can get many resources (stitched together) in a single request
- Single Endpoint
- Encapsulates location of data - great for relationships spanning multiple resources, rapidly changing shapes

Weaknesses

- Weak(-er) binding to HTTP
- Single Endpoint
- Because of first two, caching isn't as free as other mechanisms
- Client "controls" queries
- Error Handling (200-OK always!)

Cautions

- Need to be smart about understanding how data is accessed.
- Have resolvers, dataloader, parallelization, caches between resolver and source to mitigate