API Design made easy with the GraphQL Whiteboard

Jürgen Albert
Data In Motion Consulting GmbH
About Us

- Founded in 2010
- Located in Jena/Thuringia - Germany
- Consulting, Independen RnD, Development, Training
- Assisted Development on complex and distributed systems
- Wide Range of Industries like, Medical, Transportation, Traffic, Public Sector, Smart City and Industrial IoT
What is GraphQL - History

- Developed by Facebook
- Since 2012, the base for the Facebook mobile App
- Open Source since 2015
- Resides under the GraphQL Foundation since end of 2018/beginning of 2019 as part of the Linux Foundation
- Biggest Members
  - AWS
  - Facebook
  - IBM
  - Twitter
  - PayPal
  - shopify
What is GraphQL

- A Query Language and Runtime for an API
- Alternative to REST, especially if your API needs to provide for a wide range of use cases
- Best suited Backend to Frontend APIs
What is a Use Case

Consider An API for the following Model

class Address{
    attribute id : String[1] {id};
    attribute street : String;
    attribute zipcode : String;
    attribute number : String;
    attribute city : String;
    property residents : Person[*];
}

class Person {
    attribute id : String[1] {id};
    attribute name : String[1];
    property address : Address[1];
    property relatives : Person[*];
}
The Eclipse Start Process

REST API?

- GET /address
- GET /address/{id}
- GET /address/{id}/person
- GET /address/{id}/person/{id}
- GET /person
- GET /person/{id}
- GET /person/{id}/relatives
- GET /person/{id}/relatives/{id}

- A Lot of possible use cases, depending
- How many hierarchy levels do you want to return e.g. with the address?
GraphQL to the rescue

- GraphQL defines a simple schema for the API
  - Entities
    - Fields
    - Data types
    - Validation
  - Operations (Fields with Arguments)
    - Mutations (Data manipulation)
    - Querys
  - Subscriptions
How would it look like?

- Define a Schema, either in Code or with one of the available Schema DSL.

```graphql
type Address {
  id: ID!
  street: String
  number: String
  zipcode: String
  city: String
  residents: [Person]!
}

type Person {
  id: ID!
  name: String!
  relatives: [Person]!
  address: [Address]!
}

type AddressService {
  getAddresses(id: String): [Address]!
}
```
How would it look like?

- Attache data fetchers to every junktion in your Schema

```java
public class RelativesFetcherImpl implements DataFetcher<List<Person>> {

    /*
     * (non-Javadoc)
     * @see graphql.schema.DataFetcher#get(graphql.schema.DataFetchingEnvironment)
     */
    @Override
    public List<Person> get(DataFetchingEnvironment environment) throws Exception {
        Person p = environment.getSource();
        return p.get Relatives();
    }
}
```
How would it look like?

**● Gives you quite a lot of freedom**

```java
public class GetAddressesFetcherImpl implements DataFetcher<List<Address>> {
    @Reference
    AddressQuery addressService;

    /*
     * (non-Javadoc)
     * @see graphql.schema.DataFetcher#get(graphql.schema.DataFetchingEnvironment)
     */
    @Override
    public List<Address> get(DataFetchingEnvironment environment) throws Exception {
        String idArgument = environment.getArgument("id");
        return addressService.getAddresses(idArgument);
    }
}
```
What is the GraphQL White board?

- In OSGi we have Services, that provide Java Classes that execute logic (Datafetchers)
- The Java Classes are the Model that describes the Schema we are working with.

“It was so nice, lets do it twice!” - I don’t think so
What is the GraphQL White board?

Let's see!
Next Steps

● Remove binding to slf4j
● Update to latest GraphQL Version (currently supports 11)
● Better POJO Support
● Tutorials and Documentation
● Templates
● Get it more stable and easier to use
Thanks for listening!

Resources:

Web:  https://www.datainmotion.de
Blog:  https://datainmotion.de/blog
Git:  https://gitlab.com/gecko.io/geckographql
      https://gitlab.com/gecko.io/talks/ece2019_graphql
Repos:  https://devel.data-in-motion.biz/repository/gecko/snapshot/geckoGraphQL/
        https://devel.data-in-motion.biz/nexus/repository/maven-releases/
EVALUATE THE SESSIONS

Sign in and vote using the conference app or eclipsecon.org

-1 0 +1