jahia Day

Extending GraphQL schemas using Jahia

Serge Huber, Jahia CTO & Co-Founder

jahia Day

Context

OSGi

jahia Day

Java framework

Hot deployment of modules

Used in many domains

https://www.osgi.org

GraphQL

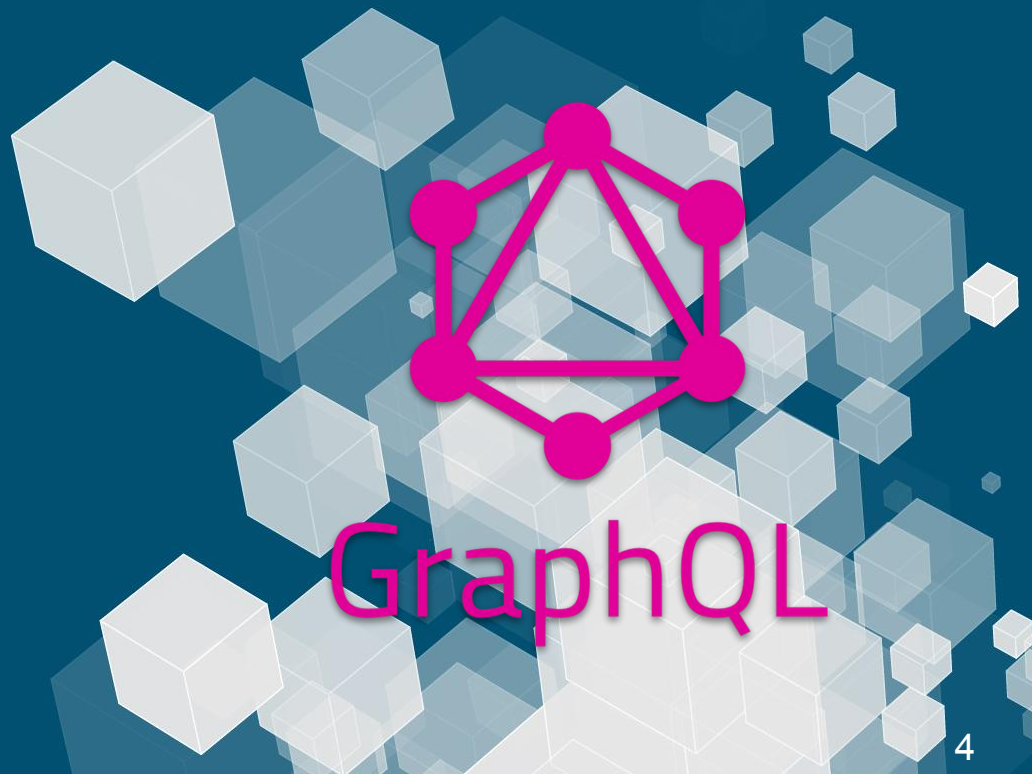
jahia Day

Open Source

Query language for APIs

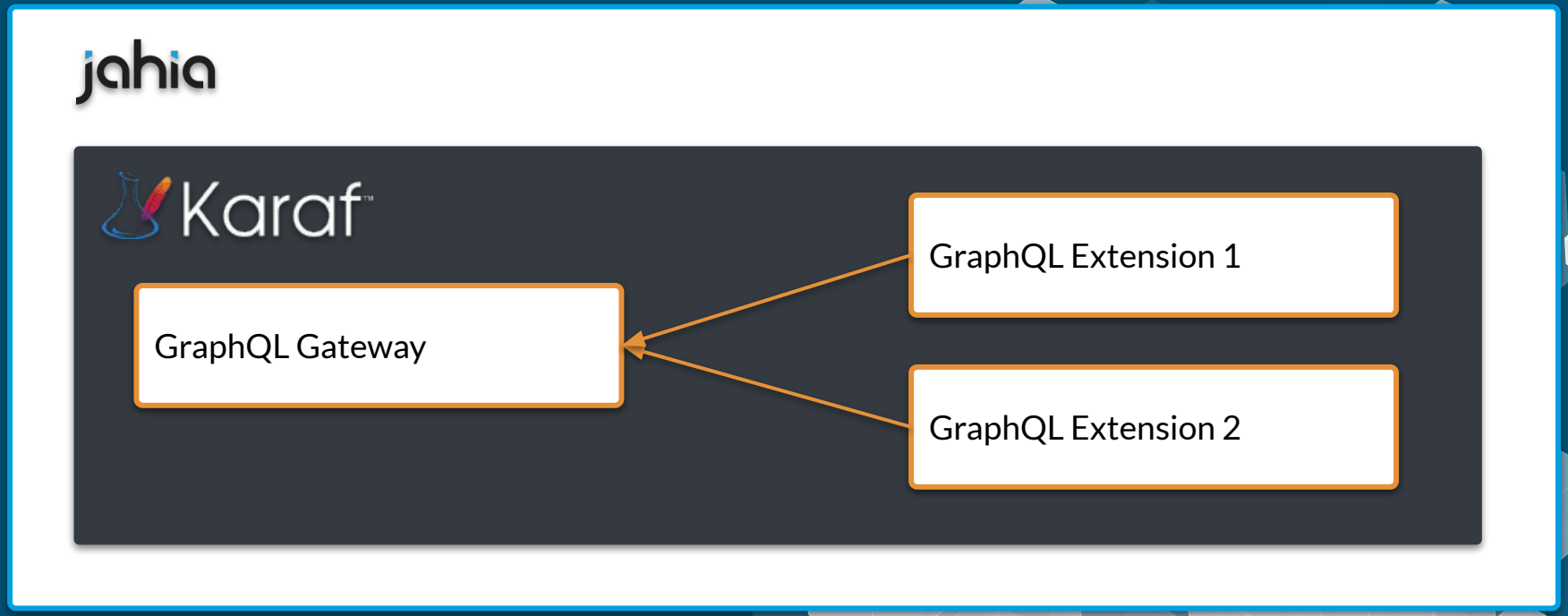
Flexible

https://graphql.org/



How does this work together

jahia Day



jahia Day

Federation VS Stitching

What is schema stitching?

jahia Day

“ Schema stitching is the process of creating a single GraphQL schema from multiple underlying GraphQL APIs. “

So, what's the problem?

jahia Day

- Monolithic
- Multiple requests
- "Glue" code
- Extend query only

Maybe better with a little example

jahia Day

```
type Movie {
    id: ID!
    title: String
    overview: String
}
type Query {
    movieById(id: ID!): Movie
}
```

Let's extend it

jahia Day

```
type Actor {
    id: ID!
    name: String
}
type Query {
    actorById(id: ID!): Actor
    actorsByMovieId(id: ID!): [Actor]
}
```

Result

jahia Day

```
type Query {
    movieById(id: ID!): Movie
    actorById(id: ID!): Actor
    actorsByMovieId(id: ID!): [Actor]
}
```

What is schema federation?

jahia Day

The same… but different!

- Build declarative graph
- Code can be split
- Graph are easy to consume

Back to our example

jahia Day

```
type Movie {
    id: ID!
    title: String
    overview: String
}
type Query {
    movieById(id: ID!): Movie
}
```

With its extension

jahia Day

```
extend type Movie {
    actors: [Actor]
}
type Actor {
    id: ID!
    name: String
}
```

jahia Day

Federation with Java and OSGi

Why?

jahia Day

Because of dynamic federation:

OSGi modules with support for GraphQL federation make it possible to evolve a GraphQL schema at runtime.

GraphQL schema evolution rules

jahia Day

- Never remove a field
- Never change a field's arguments
- Add new fields if new arguments or semantics of a field must change
- No need to version GraphQL schemas, unless major rewrites happen
- Don't split schemas, prefer aggregation layers instead

Let get cracking!

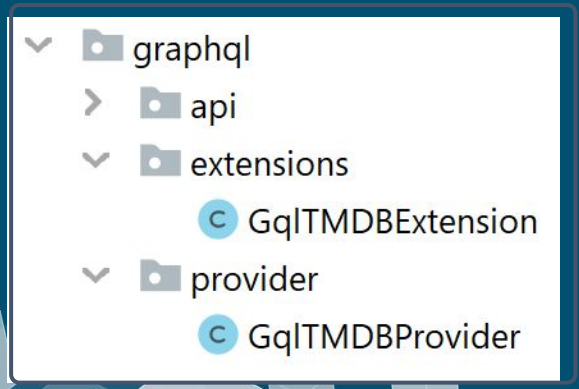
jahia Day

3 modules retrieving movie data from TMDB and IMDB:

- **Team A** provides a new entry tmdb in query and add a new type Movie
- **Team B** needs to extend the Movie type with an Actor type to get the actors list of a movie
- **Team C** needs to enrich the movie data with information coming from another API

Common part

Registering the extension




```
/**
 * Main GraphQL extension provider for TMDB
 */
@Component(immediate = true, service = DXGraphQLExtensionsProvider.class)
public class GqlTMDBProvider implements DXGraphQLExtensionsProvider {
    @Override
    public Collection<Class<?>> getExtensions() { return Arrays.<~>asList(GqlTMDBExtension.class); }
}
```

Common part

jahia Day

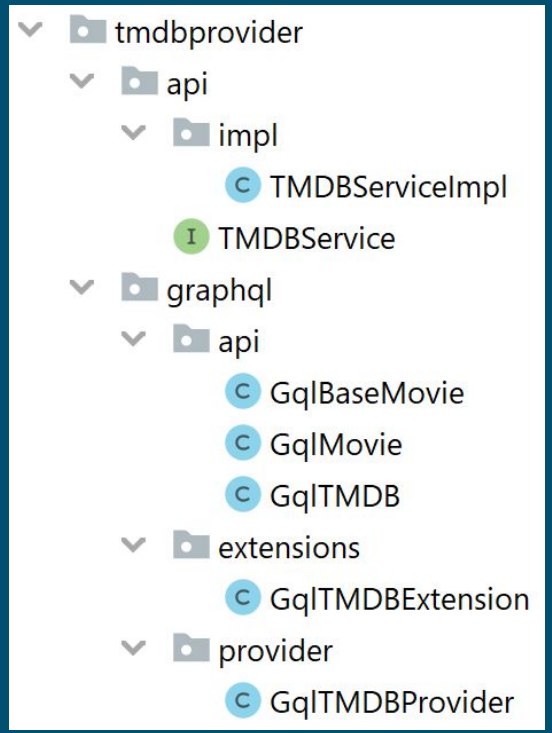
Declaring the extension

```
/**
 * This extension to the Query is where the TMDB GraphQL API is made available
 */
@GraphQLTypeExtension(DXGraphQLProvider.Query.class)
public class GqlTMDBExtension {

    @GraphQLField
    @GraphQLName("tmdb")
    @GraphQLDescription("Main access field to the Jahia GraphQL TMDB API")
    public static GqlTMDB getTmdb() { return new GqlTMDB(); }

}
```

Team A




```java
/**
 * The root class for the GraphQL TMDB API
 */
public class GqlTMDB {

    private TMDBService tmdbService = null;

    public GqlTMDB() {
        this.tmdbService = BundleUtils.getOsgiService(TMDBService.class, filter: null);
        tmdbService.init();
    }

    @GraphQLField
    @GraphQLName("movie")
    @GraphQLDescription("Get a movie")
    public GqlMovie getMovie (
        @GraphQLName("id")
        @GraphQLNonNull
        @GraphQLDescription("The primary identifier of the movie we want to retrieve")
                int id) {
        try {
            Movie movie = tmdbService.getMovie(id);
            if (movie != null) {
                return new GqlMovie(movie);
            }
            return null;
        } catch (IOException e) {
            throw new DataFetchingException(e);
        }
    }
```

Team A

```java
/**
 * The Movie representation for the GraphQL API
 */
@GraphQLName("Movie")
public class GqlMovie {
    private Movie tmdbMovie;

    public GqlMovie(Movie tmdbMovie) { this.tmdbMovie = tmdbMovie; }

    public Movie getTmdbMovie() { return tmdbMovie; }

    @GraphQLField
    @GraphQLDescription("Retrieve the name of the Movie")
    public String getTitle() { return tmdbMovie.title; }

    @GraphQLField
    @GraphQLDescription("Retrieve the overview of the movie")
    public String getOverview() {
        return tmdbMovie.overview;
    }
}
```

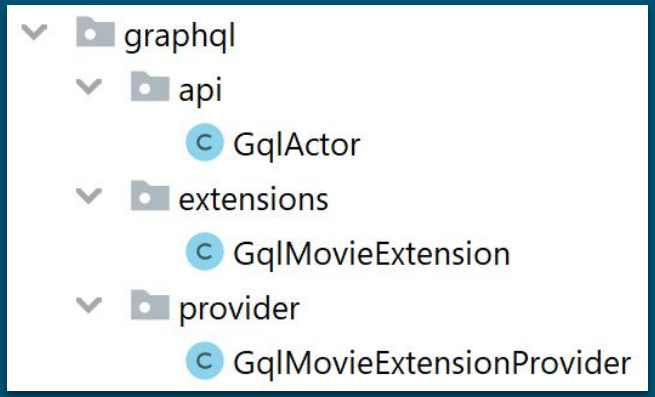
Team A

jahia Day

```
query {
  tmdb {
    movie(id: 429617) {
      title,
      overview
    }
  }
}
```


```
{
  "data": {
    "tmdb": {
      "movie": {
        "title": "Spider-Man: Far from Home",
        "overview": "Peter Parker and his friends go on a summer trip to Europe. However, they will hardly be able to rest - Peter will have to agree to help Nick Fury uncover the mystery of creatures that cause natural disasters and destruction throughout the continent."
      }
    }
  }
}
```

Team B




```java
@GraphQLTypeExtension(GqlMovie.class)
public class GqlMovieExtension {
    private GqlMovie gqlMovie;

    public GqlMovieExtension(GqlMovie gqlMovie) {
        this.gqlMovie = gqlMovie;
    }

    @GraphQLField
    @GraphQLDescription("Retrieve the list of actors of a movie")
    public Collection<GqlActor> getActors() {
        List<GqlActor> gqlActors = new ArrayList<>();
        if (gqlMovie.getTmdbMovie().credits != null
                && gqlMovie.getTmdbMovie().credits.cast != null
                && !gqlMovie.getTmdbMovie().credits.cast.isEmpty()) {
            for (CastMember castMember : gqlMovie.getTmdbMovie().credits.cast) {
                gqlActors.add(new GqlActor(castMember));
            }
        }
        return gqlActors;
    }
}
```

Team B

jahia Day

```java
@GraphQLName("Actor")
public class GqlActor {
    private CastMember castMember;

    public GqlActor(CastMember castMember) { this.castMember = castMember; }

    @GraphQLField
    @GraphQLDescription("Retrieve the name of an actor")
    public String getName() {
        return castMember.name;
    }
}
```

Team B

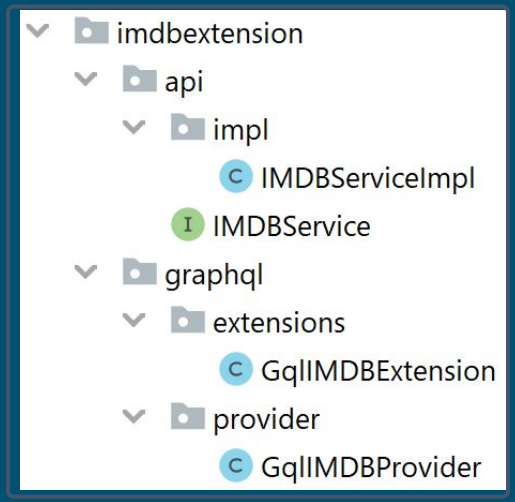
jahia Day

```
query {
  tmdb {
    movie(id: 429617) {
      title,
      overview,
      actors {
        name
      }
    }
  }
}
```


```
{
  "data": {
    "tmdb": {
      "movie": {
        "title": "Spider-Man: Far from Home",
        "overview": "Peter Parker and his friends go on a summer trip to Europe. However, they will hardly be able to rest - Peter will have to agree to help Nick Fury uncover the mystery of creatures that cause natural disasters and destruction throughout the continent.",
        "actors": [
          {
            "name": "Tom Holland"
          },
          {
            "name": "Jake Gyllenhaal"
          },
          {
            "name": "Zendaya"
          },
          {
            "name": "Jon Favreau"
          },
          {
            "name": "Samuel L. Jackson"
          },
```

Team C

jahia Day




```java
@GraphQLTypeExtension(GqlMovie.class)
public class GqlIMDBExtension {
    private GqlMovie gqlMovie;
    private IMDBService imdbService;

    public GqlIMDBExtension(GqlMovie gqlMovie) {
        this.gqlMovie = gqlMovie;
        this.imdbService = BundleUtils.getOsgiService(IMDBService.class, filter: null);
    }

    @GraphQLField
    @GraphQLDescription("Retrieve the IMDB rating of a movie")
    public Double getIMDBRating() {
        return imdbService.getMovieRating(gqlMovie.getTmdbMovie().imdb_id);
    }
}
```

Team C

jahia Day

```
query {
  tmdb {
    movie(id: 429617) {
      title,
      overview,
      iMDBRating,
      actors {
        name
      }
    }
  }
}
```


```
{
  "data": {
    "tmdb": {
      "movie": {
        "title": "Spider-Man: Far from Home",
        "overview": "Peter Parker and his friends go on a summer trip to Europe. However, they will hardly be able to rest - Peter will have to agree to help Nick Fury uncover the mystery of creatures that cause natural disasters and destruction throughout the continent.",
        "iMDBRating": 7.7,
        "actors": [
          {
            "name": "Tom Holland"
          },
          {
```

Summary

jahia Day

With schema federation and OSGi, we are able to:

- Extend an existing schema
- Not overload the query type
- Enrich an existing scheme with data from another API
- Split the code

Resources

https://www.jahia.com/jcontent

https://github.com/Jahia/graphql-core

https://github.com/DameniMilo/graphql-tmdb-provider

https://github.com/DameniMilo/graphql-tmdb-extension

https://github.com/DameniMilo/graphql-imdb-extension

Useful links

https://en.wikipedia.org/wiki/OSGi

https://karaf.apache.org/

https://unomi.apache.org/

https://blog.apollographql.com/apollo-federation-f260cf525d21

https://www.apollographql.com/docs/graphql-tools/schema-stitching/

jahia Day

THANK YOU!