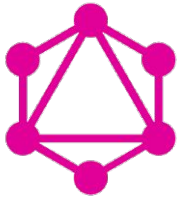
Hands-on GraphQL

Bastien CHARÈS & Thomas SIMONNET

Côté Client...



GraphQL

- Langage de requête
- Débuté par Facebook en 2012 (Open-source depuis 2015)
- Une alternative à REST, permet de récupérer uniquement les données que l’on veut, structurées comme la requête
- Implémentations en différents langages

Quand utiliser GraphQL ?

- Données à récupérer en graphe / avec relations
- API consommée par un Front sur des ressources liées entre elles
- Besoin de limiter le nombre de requêtes côté client
- Pas de surprises dans le format des réponses

L'auteur dont le nom de famille est "King", avec son prénom et les titres de tous ses livres

Requête

```
query {
  author(lastName : "King") {
    firstName,
    books {
      title
    }
  }
}
```

```
{
  author : {
    firstName : "Stephen",
    books : [
      {
        title : "The Shininge"
      },
      {
        title : "The Black
                 House"
      }
  }
}
```

Modifier le titre du livre avec l'ID 1

Requête

```
mutation {
  updateBookTitle(
    id : 1,
    title : “The Shining”
  ) {
    id,
    title
    }
  }
}
```

Modifier le titre du livre avec l'ID 1

Requête

```
mutation {
  updateBookTitle(
    id : 1,
    title : “The Shining”
  ) {
    id,
    title,
     slug
    }
  }
}
```

```
{
  id : 1,
  title : “The Shining”,
  slug : “the-shining”
}
```

Service GraphQL

Une Ressource GraphQL c'est :

- Un Type qui définit une liste d'attributs
- Une fonction (resolver) pour chaque attribut

Toujours pas clair ?

Voyons un exemple simple

```
query {
    author(id : 1) {
        firstName
        lastName
    }
}
```



```
{
  "data": {
    "author": {
      "firstName": "Stephen",
      "lastName": "King"
    }
  }
}
```

```
query {
    author(id : 1) {
        firstName
        lastName
    }
}
```

```
type Query {
  author : Author
}
```

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

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

```
module.exports = new GraphQLObjectType({
 name : "Author",
 fields : () => ({
   firstName : {
     type : GraphQLString,
     resolve(author) {
        return getFirstName(author);
     },
   },
   lastName : {
     type : GraphQLString,
     resolve(author) {
       return author.family_name;
     },
   },
 }),
});
```

```
type Query {
  author : Author
}
```

```
query {
    author {
        firstName
        lastName
    }
}
```

```
const graphQL = require("graphql");
const AuthorType = require("./Author");
const authorClient = require("./author/author.client");

const QueryType = new GraphQLObjectType({
 name : "Query",
 fields : () => ({
   author : {
     type : AuthorType,
     resolve() {
       return authorClient.getRandomAuthor();
     }
   },
 }),
});

module.exports = new GraphQLSchema({
  query : QueryType,
});
```

```
type Query {
  author : Author
}
```

```
query {
    author(id : 1) {
        firstName
        lastName
    }
}
```

```
...
const QueryType = new
graphQL.GraphQLObjectType({
 name : "Query",
 fields : () => ({
   author : {
     type : AuthorType,
     args : {
       id : {
         name : "Author id",
         type : graphQL.GraphQLInt,
       },
     },
     resolve(_, args) {
        return getAuthorById(args.id);
     }
   },
 }),
});
...
```

Partons maintenant de la requête

```
query {
    author(id : 1) {
        ...
    }
}
```

ressource “author”

du type query

```
...graphQL.GraphQLObjectType({
 name : "Query",
 fields : () => ({
   author : {
     type : AuthorType,
     args : {
       id : {
         name : "Author id",
         type : graphQL.GraphQLInt,
       },
     },
     resolve(_, args) {
       return getAuthorById(args.id);
     }
...
```

résultat

```
{
  firstName : "Stephen",
  family_name : "King",
  etc...
}
```

```
query {
    author(id : 1) {
        firstName
        lastName
    }
}
```

```
{
firstName : "Stephen",
family_name : "King",
etc...
}
```

```
... new GraphQLObjectType({
 name : "Author",
 fields : () => ({
   firstName : {
     type : GraphQLString,
     resolve(author) {
       return getFirstName(author);
     },
   },
   lastName : {
     type : GraphQLString,
     resolve(author) {
       return author.family_name;
     },
   },
 ...
```

→ Stephen

→ King

C’est tout !

```
{
    “data” : {
        “author” : {
            “firstName” : “Stephen”
            “lastName” : “King”
        }
    }
}
```

Hands-on

https://github.com/xebia-france/hands-on-graphql