GraphQL Factory Documentation

Release 2.1.0

Branden Horiuchi

Jul 28, 2018

Contents

CHAPTER 1

Quickstart

Install graphql-factory and graphql in your project

```
npm install --save graphql graphql-factory
```

Import graphql-factory and graphql, then create a new factory

```
import * as graphql from 'graphql'
import GraphQLFactory from 'graphql-factory'

const factory = GraphQLFactory(graphql)
```

Create a factory definition

```
const definition = {
    types: {
        User: {
            name: 'User',
            fields: {
                id: { type: 'ID', nullable: false },
                name: { type: 'String', nullable: false },
                email: { type: 'String' }
            }
        },
        UsersQuery: {
            listUsers: {
                type: ['User'],
                args: {
                    search: { type: 'String' }
                },
                resolve (source, args, context, info) {
                    return context.db
                        .table('user')
                        .filter(args)
                }
```

(continues on next page)

(continued from previous page)

```
            }
        }
    },
    schemas: {
        Users: {
            query: 'UsersQuery'
        }
    }
}
```

Make a library object from the definition

```
const lib = factory.make(definition)
```

Make a request

```
lib.Users(`
    query Query ($search: String) {
        listUsers (search: $search) {
            id
            name
            email
        }
    }
`, {}, { db }, {
    search: 'john'
})
.then(users => {
    // process results
})
```

CHAPTER 2

Definitions

CHAPTER 3

Middleware

Middleware in graphql-factory is used to wrap resolver functions in global functions. These functions can be run before or after the resolver. These functions can both modify arguments before the resolver is run or modify results after.

Example

```
const lib = factory.make(definition, {
    beforeResolve: [
        function (args, next) {
            // first before middleware
            next()
        },
        function (args, next) {
            // second before middleware
            next()
        }
    ],
    afterResolve: [
        function (args, result, next) {
            // first after middleware, modify prop1
            result.prop1 = null
            next(result)
        },
        function (args, result, next) {
            // second after middleware
            next()
        }
    ],
    beforeTimeout: 10000,
    afterTimeout: 2000
})
```

CHAPTER 4

Plugins

Plugins extend a graphql factory definition with additional types, schemas, functions, global values, and middleware. They are essentially definitions that are merged into the main definition and have an optional `install` function that gives access to the definition to add middleware

```
const MyPlugin = {
    types: {
        MyType: {
            fields: {
                id: { type: 'ID', nullable: false },
                foo: { type: 'String' }
            }
        }
    },
    install (definition) {
        definition.beforeMiddleware(
            function (params, next) {
                // do something
                return next()
            }
        )
    }
}
```

CHAPTER 5

API

5.1 `GraphQLFactory` instance

GraphQLFactory requires graphql to be passed as its only parameter to create a factory instance:

```
const factory = GraphQLFactory(graphql)
```

Hint: When using `import` to load `graphql` all exports should be imported with `import * as graphql from 'graphql'`

5.2 `factory` prototype

The factory prototype is used to create a library

5.2.1 `make()`

Signature: `make(Object definition [, Object options])`

Creates a new library

```
factory.make(definition, {
    plugin: [
        new TypesPlugin(),
        new SubscriptionPlugin()
    ],
    beforeMiddleware: [
        function (params, next) {
            const { source, args, context, info } = params
            args.before = true
```

(continues on next page)

(continued from previous page)

```
            return next()
        }
    ],
    afterMiddleware: [
        function (params, result, next) {
            result.after = true
            return next(result)
        }
    ]
})
```

Options

- `compile` When **false** skips the compile process
- `plugin` Array of plugins
- `beforeMiddleware` Array of before middleware
- `afterMiddleWare` Array of after middleware
- `beforeTimeout` Timeout for all before middleware to complete
- `afterTimeout` Timeout for all after middleware to complete

Returns `GraphQLFactoryLibrary`

5.3 `GraphQLFactoryLibrary` instance

The GraphQLFactoryLibrary prototype is used to make graphql requests and access the definition. Each schema in the definition will have its own method in the library

5.3.1 `<schemaName>()`

Signature: `<schemaName>(String request [, Object root, Object context, Object variables, String operation])`

Makes a new graphql request

```
lib.Users(`
    query Query {
        listUsers {
            id
            name
            email
        }
    }
`)
```

Options:

- `request` graphql requestString
- `root` graphql rootValue
- `context` graphql context
- `variables` graphql variableValues

- `operation` graphql operationName

Index