Uber-modern APIs for Sitecore

Kam Figy
April 23, 2018
#sugcon
GraphQL all the things!
GraphQL is a query language for your API

- Designed to support the needs of the frontend
- Offers amazing tooling
- Strongly typed, self-documenting
- Bandwidth efficient
- Easy real-time data
Give your REST API a rest
No more overfetching!

ItemService Response

```
{
  ItemID: "110d559f-dea5-42ea-9c1c-8a5df7e70ef9",
  ItemName: "Home",
  ItemPath: "/sitecore/content/Home",
  ParentID: "0de95ae4-41ab-4d01-9eb0-67441b7c2450",
  TemplateID: "76036f5e-cbce-46d1-af0a-4143f9b557aa",
  TemplateName: "Sample Item",
  CloneSource: null,
  ItemLanguage: "en",
  ItemVersion: "1",
  DisplayName: "Home",
  HasChildren: "True",
  ItemIcon: "/temp/iconcache/network/16x16/home.png",
  ItemMedialUrl: "/-/icon/Network/48x48/home.png.aspx",
  ItemUrl: "~/link.aspx?_id=110D559FDEA542EA9C1C8A5DF7E7F"
  Text: "<p style="line-height: 22px;">From a single common marketing repository, to completely eliminating much of achievable. Learn how the latest version of Sitecore g: lifecycle &dash; the technology foundation absolutely target="_blank" title="Sitecore Documentation site">Sitecore Experience Platform"
  Title: "Sitecore Experience Platform"
}
```

GraphQL Query + Response

```
{
  item(path: "110d559f-dea5-42ea-9c1c-8a5df7e70ef9") {
    name
    title: field(name: "Title") {
      value
    }
  }
}
```

```
"data": {
  "item": {
    "name": "Home",
    "title": {
      "value": "Sitecore Experience Platform"
    }
  }
}
```
Data relationships!

Get all your data in *one* query.
GraphQL is typed!

Allows querying items from the content tree

**TYPE**

*Item*

**ARGUMENTS**

- **path**: `String = null`
The item path or ID to get
- **language**: `String = null`
The item language to request (defaults to the default language)
- **version**: `Int = null`
The item version to request (if not set, latest version is returned)

**FIELDS**

- **id**(format: `String = "N"`): `ID!`
- **language**: `ItemLanguage!`
- **version**: `Int!`
- **database**: `String!`
- **name**: `String!`
- **displayName**: `String!`

/sitecore/templates/JssBasicAppGraphQL/Integrated template (ID: {DA8784A4-9ADF-5DA3-9175-CD9CEC0F41FC}).
GraphQL

Playtime!

Take the red pill.
GraphQL + JSS

A perfect match
JSS 101: Layout Service

```
placeholders: {
  - main: [
    - {
        uid: "{BC8C7AB9-6D40-5393-ABDE-6D7DF27F2D3F}"
      componentName: "Home"
      dataSource: "available-in-connected-mode"
      params: {}
      fields: 
      placeholders: {
        - page-header: [
          - {
              uid: "{B840AB8A-0BB6-5393-B39E-D64F96901D9C}"
                componentName: "Jumbotron"
                dataSource: "available-in-connected-mode"
                params: {
                    titleSize: "1"
                    shade: "dark"
                }},
            fields: {
                titleText: {
                    value: "Hello, stranger!"
                    editable: "Hello, stranger!"
                },
                body: {
                    value: "<p>Start Bootstrap can help you bu"
                    editable: "<p>Start Bootstrap can help you"
                }
            }
        }
    ]

```
**Integrated GraphQL**

- Reshape layout service rendering data without server-side customization

**Connected GraphQL**

- GraphQL HTTP/WebSocket requests initiated by client
- Client libraries such as Apollo Client recommended
query IntegratedPageQuery($datasource: String!, $contextItem: String) {
  # Datasource query
  # $datasource will always be set to the ID of the rendering
  datasource(value: $datasource) {
    ... on IntegratedPage {
      title {
        jss
      }
      text {
        jss
      }
      logoImage {
        jss
      }
    }
  }
}

# Context item query
# $contextItem will always be set to the ID of the current context item (the route item)
contextItem: item(path: $contextItem) {
  id
  children(requirePresentation: true) {
    displayName
    url(options: { disableLanguageEmbedding: true })
  }
}
Putting it all together.
Make your own GraphQL APIs!
• Sitecore GraphQL is an API framework
• Create a *Schema Provider* to build your GraphQL API
• Create *Graph Types* to map C# types to GraphQL types
• Add *Schema Extenders* to change existing graph types
But wait, there’s more!

- Aggregating existing APIs into a single GraphQL schema
- Using `graphql-tools` to create a fake GraphQL server
- Static analysis and validation of `.graphql` files with ESLint
- Query whitelisting & complexity analysis
ONE DOES NOT SIMPLY ADDRESS THE ELEPHANT IN THE ROOM
Sitecore GraphQL currently ships with JSS as a technical preview. It can also operate independently from JSS.

Docs: [https://jss.sitecore.net/#/graphql/graphql](https://jss.sitecore.net/#/graphql/graphql)

GraphQL Resources

- [http://graphql.org/](http://graphql.org/)
- [https://reactjs.org/blog/2015/05/01/graphql-introduction.html](https://reactjs.org/blog/2015/05/01/graphql-introduction.html)
- [https://github.com/chentsulin/awesome-graphql](https://github.com/chentsulin/awesome-graphql)
Thank you!

Twitter: @kamsar