Content Delivery using Content Fragments with GraphQL

With Adobe Experience Manager (AEM) as a Cloud Service, you can use Content Fragments, together with GraphQL, to deliver structured content for use in your applications.

GraphQL - An Overview

GraphQL is:

- "*...a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.*".
  See GraphQL.org
- "*...an open spec for a flexible API layer. Put GraphQL over your existing backends to build products faster than ever before....*".
  See Explore GraphQL. "*Explore GraphQL is maintained by the Apollo team. Our goal is to give developers and technical leaders around the world all of the tools they need to understand and adopt GraphQL.*".

GraphQL allows you to perform (complex) queries on your Content Fragments; with each query being according to a specific model type. The content returned can then be used by your applications.

GraphQL Terminology

GraphQL uses the following:

- **Queries**
- **Schemas and Types**
- **Fields**

See the (GraphQL.org) Introduction to GraphQL for comprehensive details, including the Best Practices.

GraphQL Query Types

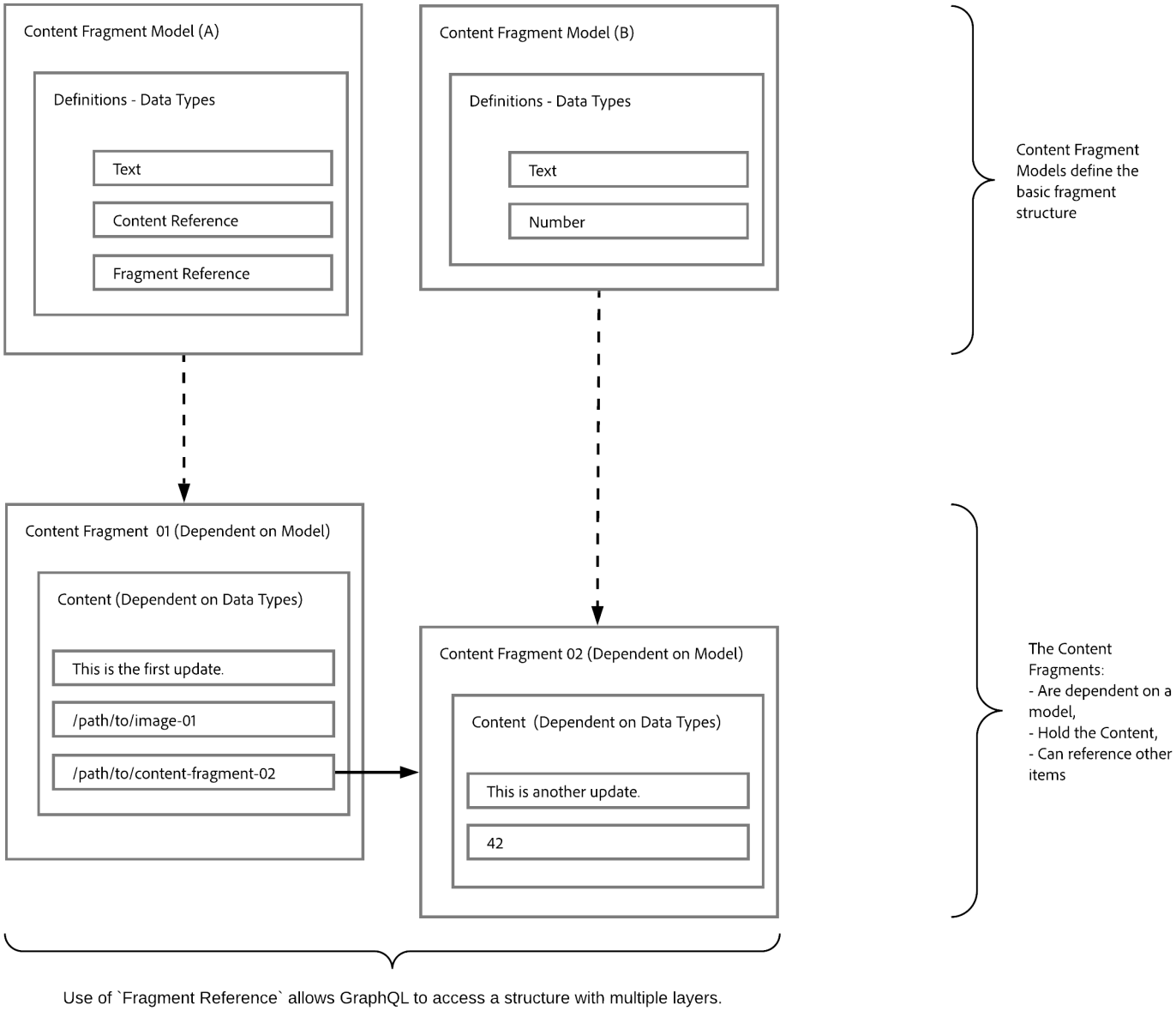
With GraphQL you can perform queries for either:

- **Single entry**
- **List of entries**

Content Fragments for use with GraphQL

Content Fragments can be used as a basis for GraphQL queries as:

- The Content Fragment Models provide the required structure by means of defined data types.
- The Fragment Reference, available when defining a model, can be used to define additional layers of structure.



Content Fragments

Content Fragments:

- Contain structured content.
- They are based on a Content Fragment Model, which predefines the structure for the resulting fragment.

Content Fragment Models

These Content Fragment Models:

- Provide the data types and fields required for GraphQL. They ensure that your application only requests what is possible, and receives what is expected.
- The data type **Fragment References** can be used in your model to reference another Content Fragment, and so introduce additional levels of structure.

Fragment References

The **Fragment Reference**:

- Is of particular interest in conjunction with GraphQL.
- Is a specific data type that can be used when defining a Content Fragment Model.
- References another fragment, dependent on a specific Content Fragment Model.
- Allows you to retrieve structured data.
    - When defined as a **multifeed**, multiple sub-fragments can be referenced (retrieved) by the prime fragment.

See:

- A sample Content Fragment structure
- And some sample GraphQL queries, based on the sample content fragment structure (Content Fragment Models and related Content Fragments).

A Sample Content Fragment Structure for use with GraphQL

For a simple example we need:

- One, or more, Sample Content Fragment Models - form the basis for the GraphQL schemas
- Sample Content Fragments based on the above models

Sample Content Fragment Models (Schemas)

For the sample queries, we will use the following Content Models, and their interrelationships (references ->):

- Company
  -> Person
    -> Award
- City

Company

The basic fields defining the comparny are:

Field Name	Data Type	Reference
Company Name	Single line text	
CEO	Fragment Reference (single)	Person
Employees	Fragment Reference (multifield)	Person

Person

The fields defining a person, who can also be an employee:

Field Name	Data Type	Reference
Name	Single line text	
First name	Single line text	
Awards	Fragment Reference (multifield)	Award

Award

The fields defining an award are:

Field Name	Data Type	Reference
Shortcut/ID	Single line text	
Title	Single line text	

City

The fields for defining a city are:

Field Name	Data Type	Reference
Name	Single line text	
Country	Single line text	
Population	Number	
Categories	Tags	

Sample Content Fragments

The following fragments are used for the appropriate model.

Company

- Apple
  - CEO: Steve Jobs
  - Employees: Duke Marsh and Max Caulfield
- Little Pony Inc.
  - CEO: Adam Smith
  - Employees: Lara Croft and Cutter Slade
- NextStep Inc.
  - CEO: Steve Jobs
  - Employees: Joe Smith and Abe Lincoln

Person

- Abe Lincoln
- Adam Smith
- Cutter Slade
  - Awards: Gameblitz and Gamestar
- Duke Marsh
- Joe Smith
- Lara Croft
  - Awards: Gamestar
- Max Caulfield
  - Awards: Gameblitz
- Steve Jobs

Award

- Gameblitz Award
- Gamestar Award
- Oscar

City

- Basel
  - Country: Switzerland
  - Population: 172258
  - Categories: city:emea
- Berlin
  - Country: Germany
  - Population: 3669491
  - Categories: city:capital and city:emea
- Bucharest
  - Country: Romania
  - Population: 1821000
  - Categories: city:capital and city:emea
- San Francisco
  - Country: USA
  - Population: 883306
  - Categories: city:na
- San Jose
  - Country: USA
  - Population: 102635
  - Categories: city:na
- Stuttgart
  - Country: Germany
  - Population: 634830
  - Categories: city:emea
- Zurich
  - Country: Switzerland
  - Population: 415367
  - Categories: city:capital and city:emea

GraphQL - Sample Queries

Sample Query - All Available Schemas and Datatypes

This will return all available schemas and datatypes.

Sample Query

```
{
  __schema {
    types {
      name
    }
  }
}
```

Sample Result

```
{
  "data": {
    "__schema": {
      "types": [
        {
          "name": "ArrayMode"
        },
        {
          "name": "AwardModel"
        },
        {
          "name": "AwardModelArrayFilter"
        },
        {
          "name": "AwardModelFilter"
        },
        {
          "name": "Boolean"
        },

...more results...

       {
          "name": "__InputValue"
        },
        {
          "name": "__Schema"
        },
        {
          "name": "__Type"
        },
        {
          "name": "__TypeKind"
        }
      ]
    }
  }
}
```


Sample Query - All Cities

This is a straightforward query to return the `name` of all entries in the `city` schema.

Sample Query

```
query {
  citys {
    name
  }
}
```

Sample Results


```
{
  "data": {
    "citys": [
      {
        "name": "Basel"
      },
      {
        "name": "Berlin"
      },
      {
        "name": "Bucharest"
      },
      {
        "name": "San Francisco"
      },
      {
        "name": "San Jose"
      },
      {
        "name": "Stuttgart"
      },
      {
        "name": "Zurich"
      }
    ]
  }
}
```


Sample Query - All Persons that have a name of "Jobs" or "Smith"

This will filter all `persons` for any that have the name `Jobs` or `Smith` .

Sample Query

```
query {
  persons(filter: {
    name: {
      _logOp: OR
      _expressions: [
        {
          value: "Jobs"
        },
        {
          value: "Smith"
        }
      ]
    }
  }) {
    name
    firstName
  }
}
```

Sample Results

```
{
  "data": {
    "persons": [
      {
        "name": "Smith",
        "firstName": "Adam"
      },
      {
        "name": "Smith",
        "firstName": "Joe"
      },
      {
        "name": "Jobs",
        "firstName": "Steve"
      }
    ]
  }
}
```

Sample Query - All cities located in Germany or Switzerland with a population between 400000 and 999999

Here a combination of fields are filtered on. An `AND` (implicit) is used to select the `population` range, while an `OR` (explicit) is used to select the required cities.

Sample Query

```
query {
  citys(filter: {
    population: {
      _expressions: [
        {
          value: 400000
          _operator: GREATER_EQUAL
        }, {
          value: 1000000
          _operator: LOWER
        }
      ]
    },
    country: {
      _logOp: OR
      _expressions: [
        {
          value: "Germany"
        }, {
          value: "Switzerland"
        }
      ]
    }
  }) {
    name
    population
    country
  }
}
```

Sample Results

```
{
  "data": {
    "citys": [
      {
        "name": "Stuttgart",
        "population": 634830,
        "country": "Germany"
      },
      {
        "name": "Zurich",
        "population": 415367,
        "country": "Switzerland"
      }
    ]
  }
}
```

Sample Query for Nested Content Fragments - All companies that have at least one employee that has a name of "Smith"

This query illustrates filtering across two nested fragments - `company` and `employee`.

Sample Query

```
query {
  companys(filter: {
    employees: {
      _match: {
        name: {
          _expressions: [
            {
              value: "Smith"
            }
          ]
        }
      }
    }
  }) {
    name
    ceo {
      name
      firstName
    }
    employees {
      name
      firstName
    }
  }
}
```

Sample Results

```
{
  "data": {
    "companys": [
      {
        "name": "NextStep Inc.",
        "ceo": {
          "name": "Jobs",
          "firstName": "Steve"
        },
        "employees": [
          {
            "name": "Smith",
            "firstName": "Joe"
          },
          {
            "name": "Lincoln",
            "firstName": "Abraham"
          }
        ]
      }
    ]
  }
}
```


Sample Query for Nested Content Fragments - All companies where all employees have won the "Gamestar" award

This query illustrates filtering across three nested fragments - `company`, `employee`, and `award`.

Sample Query

```
query {
  companys(filter: {
    employees: {
      _apply: ALL
      _match: {
        awards: {
          _match: {
            id: {
              _expressions: [
                {
                  value: "GS"
                  _operator:EQUALS
                }
              ]
            }
          }
        }
      }
    }
  }) {
    name
    ceo {
      name
      firstName
    }
    employees {
      name
      firstName
      awards {
        id
        title
      }
    }
  }
}
```

Sample Results

```
{
  "data": {
    "companys": [
      {
        "name": "Little Pony, Inc.",
        "ceo": {
          "name": "Smith",
          "firstName": "Adam"
        },
        "employees": [
          {
            "name": "Croft",
            "firstName": "Lara",
            "awards": [
              {
                "id": "GS",
                "title": "Gamestar"
              }
            ]
          },
          {
            "name": "Slade",
            "firstName": "Cutter",
            "awards": [
              {
                "id": "GB",
                "title": "Gameblitz"
              },
              {
                "id": "GS",
                "title": "Gamestar"
              }
            ]
          }
        ]
      }
    ]
  }
}
```