



DEVELOPERS LIVE

Headless GraphQL with Content Fragments

Jabran Asghar | Senior Software Engineer

Agenda

Content delivery & management

Content Fragment Models & GraphQL types

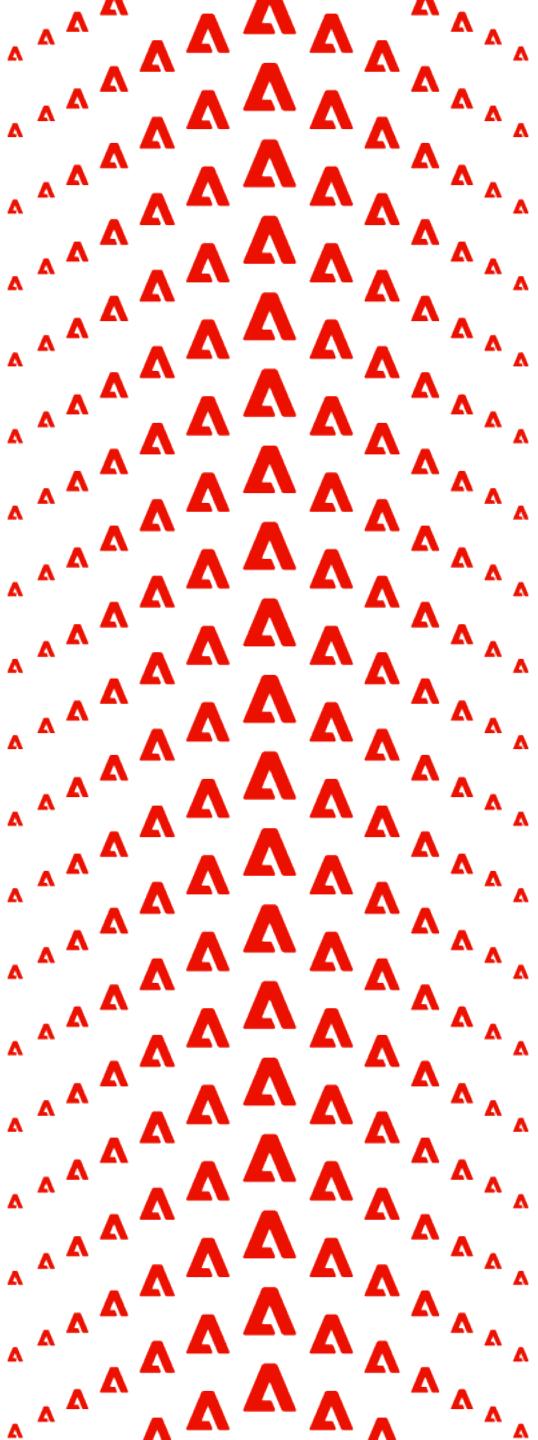
Content Fragment Modeling enhancements

Content Fragment editor enhancements

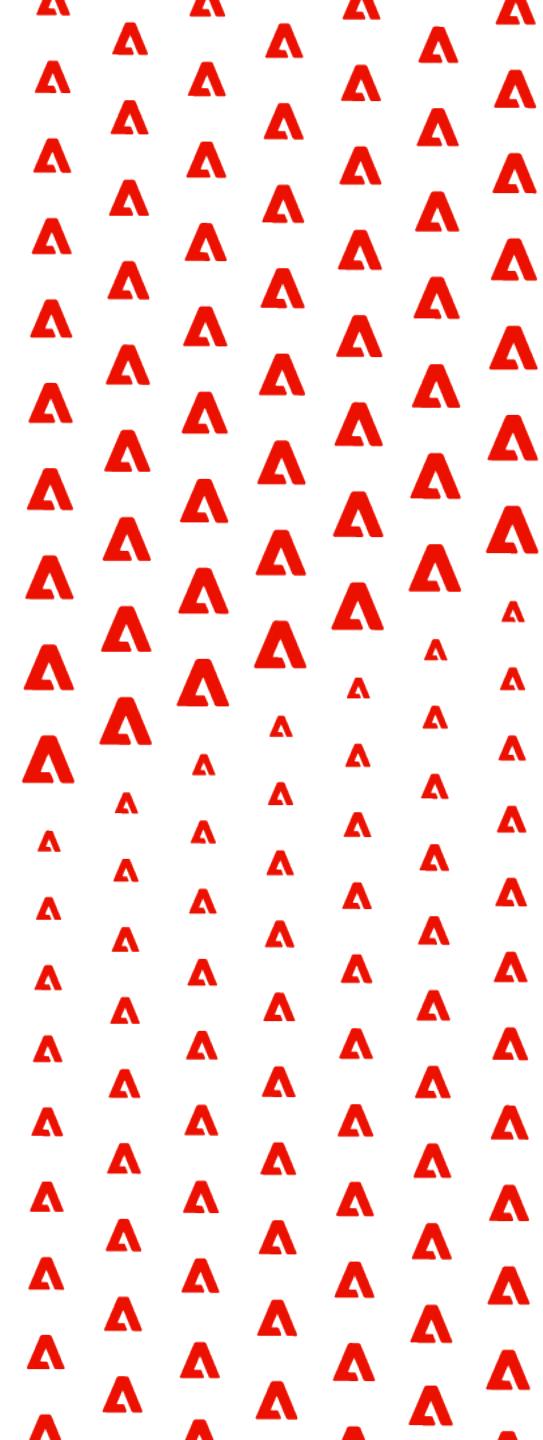
Headless – AEM GraphQL API Demo

Wrap up

What's next?

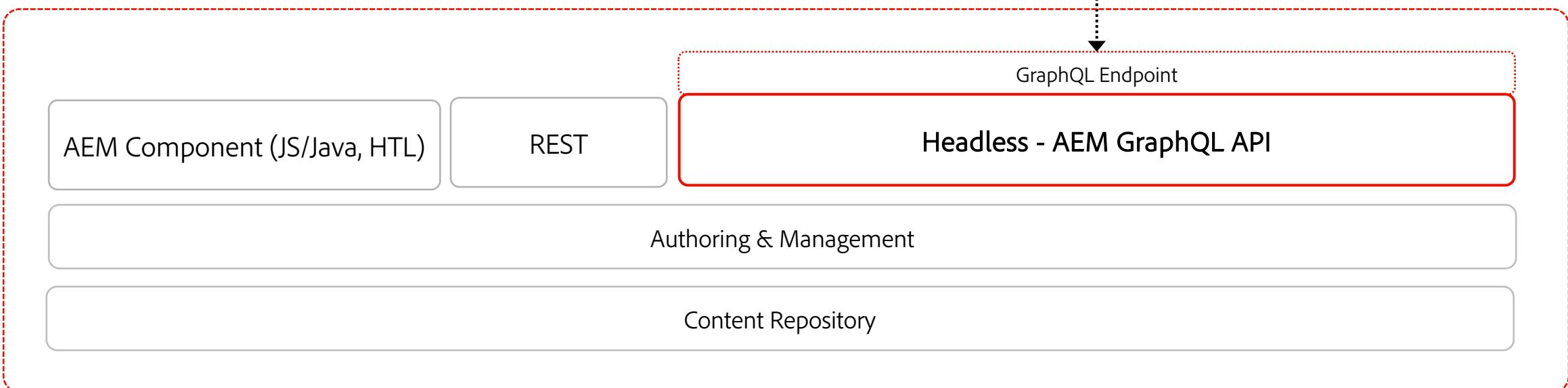
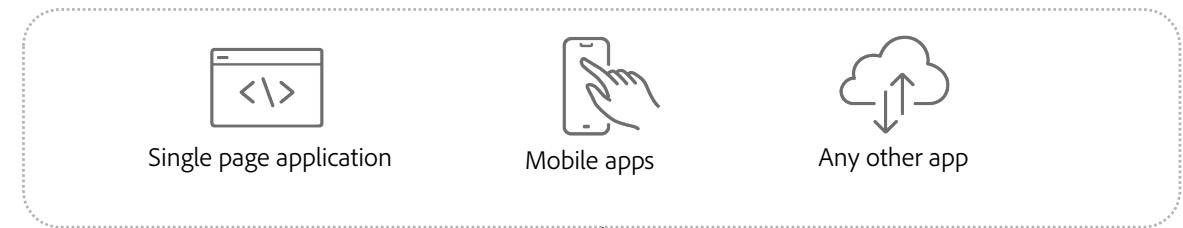


AEM content delivery & management

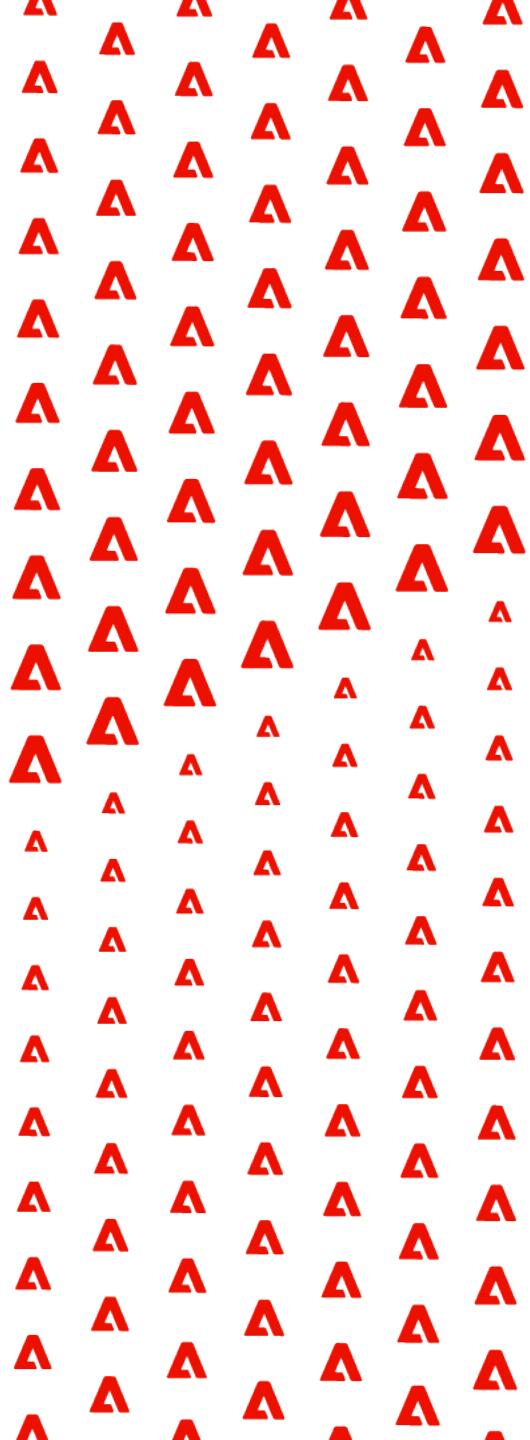


AEM: Content Delivery & Management

Manage experiences centrally, deliver across channels via GraphQL API



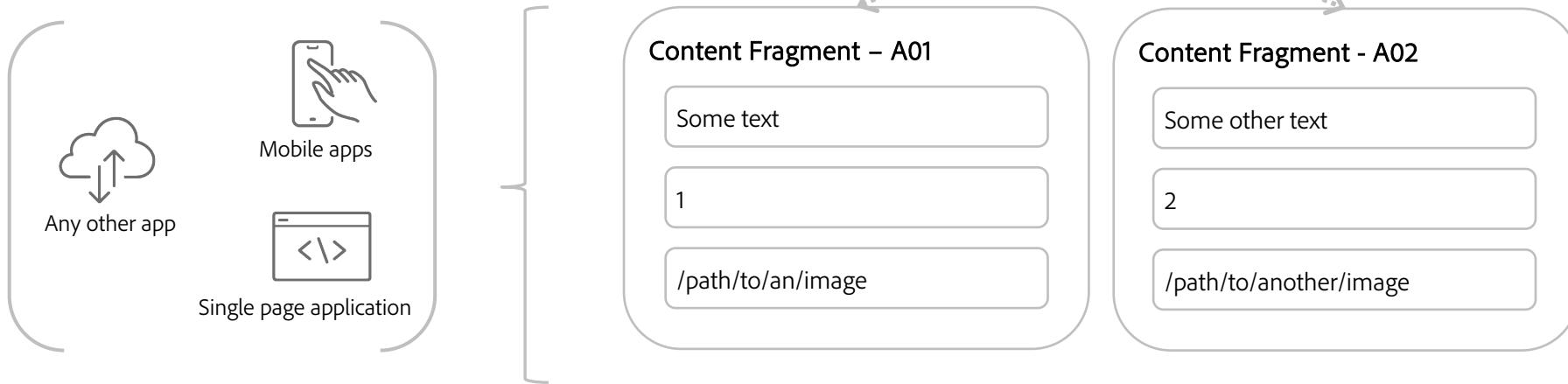
Content Fragment Models & GraphQL types



Content Fragment Models: What?

Content schemas, to templatize the creation of raw content by AEM authors

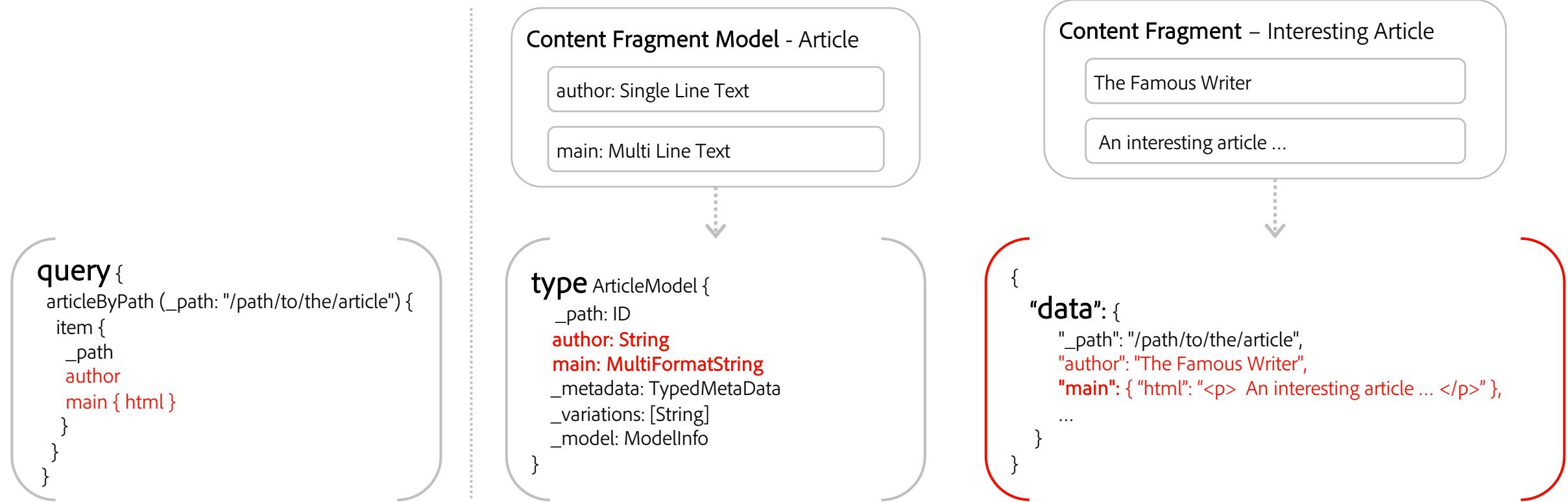
- Structured
- Presentation-agnostic
- Intended for multi-channel use



Content Fragment Models: GraphQL Types

AEM APIs for headless capabilities converts Content Fragment Models to GraphQL types

- Content Fragments are retrieved via GraphQL queries, in JSON format



AEM headless: GraphQL query features

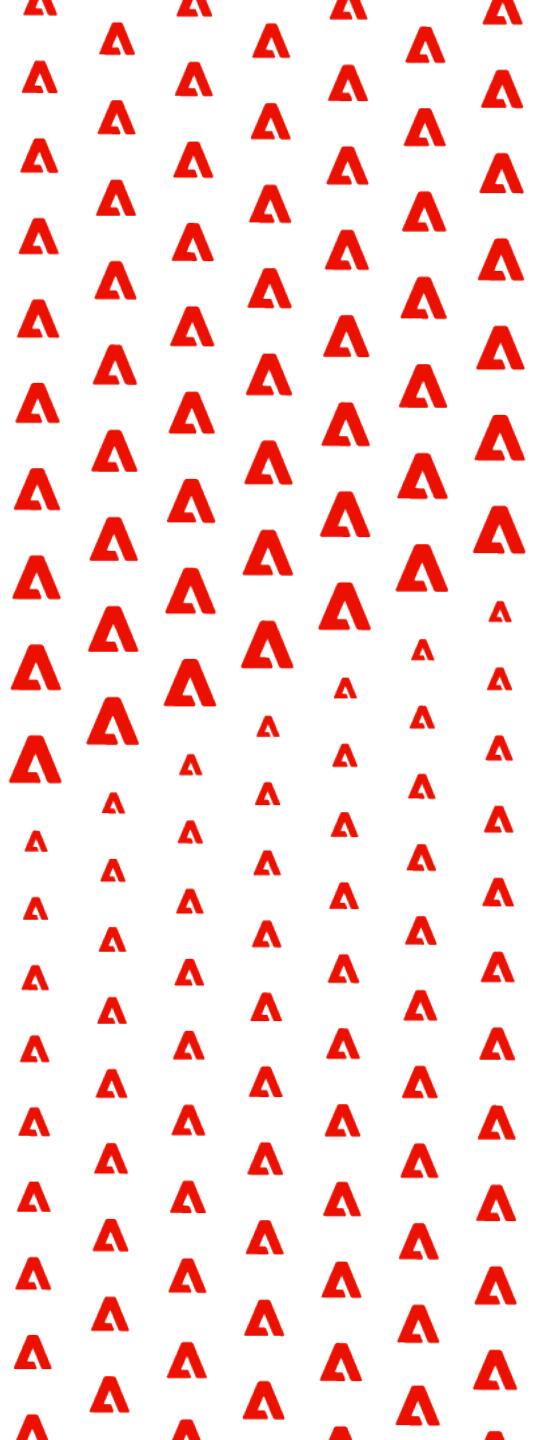
Supported GraphQL features

- Aliases
- Arguments
- Fragments
- Input types
- Introspection
- Meta fields
- Nested fragments
- Union types
- Variables

Not supported (yet)

- Mutations
- Subscriptions
- Paging/Sorting (for GraphQL JSON results)

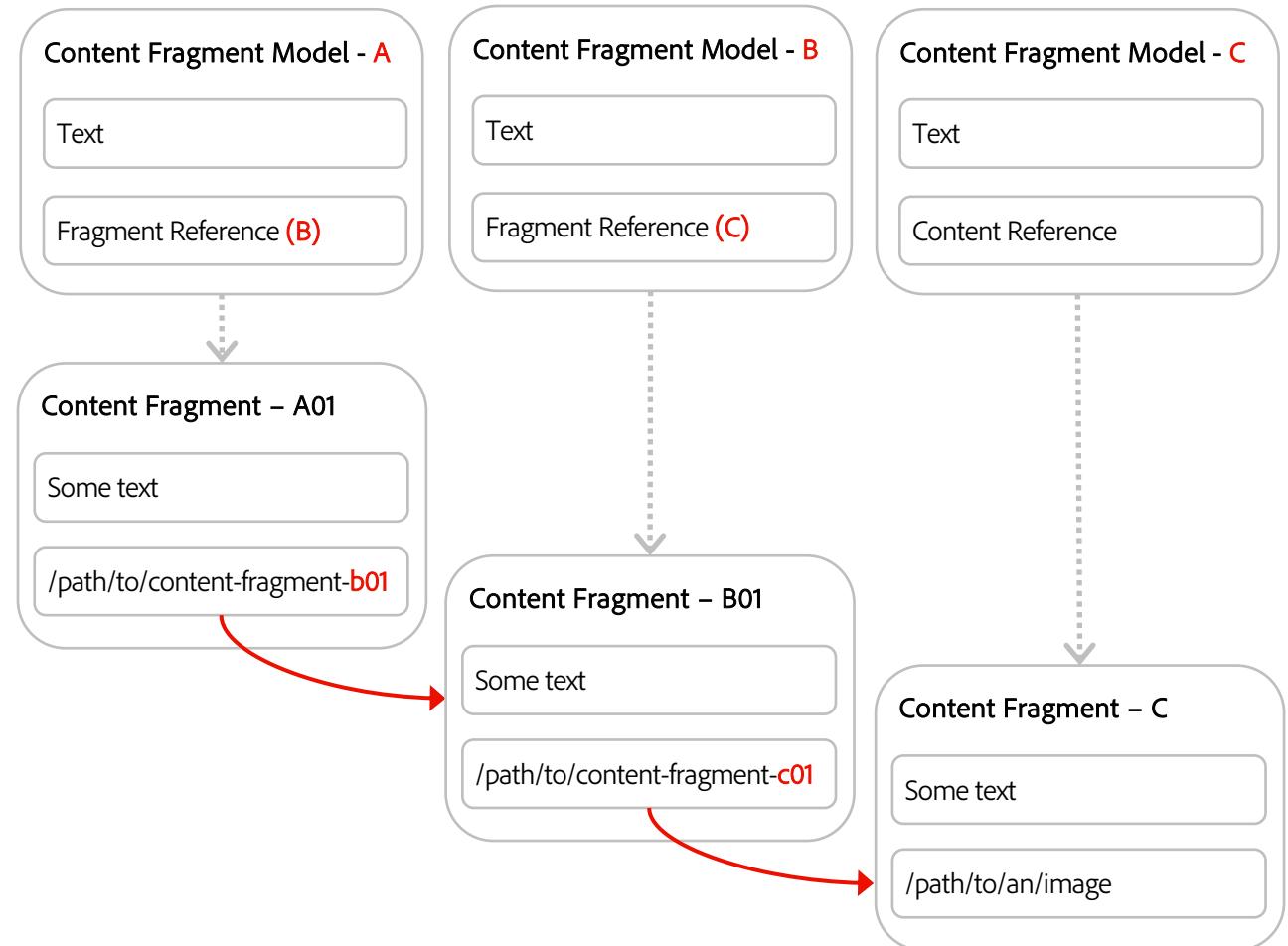
Content Fragment Modeling enhancements



Content Fragment Model editor: New data types

Fragment Reference type

- References another fragment, dependent on a specific Content Fragment Model
- Used to define additional layers of structure when defining a model
- Can be nested over multiple levels
- Enables to retrieve nested Content Fragments for a prime Content Fragment



Content Fragment Model editor: New data types

JSON data type

- Enables you to enter a JSON value in a field
- The JSON value is returned as a JSON scalar in the GraphQL result

```
query {  
  bookByPath (_path: "/path/to/the/book") {  
    item {  
      author  
      info  
    }  
  }  
}
```

Content Fragment Model - Book

author: Single Line Text

info: JSON

```
type BookModel {  
  _path: ID  
  author: String  
  info: Json  
  _metadata: TypedMetaData  
  _variations: [String]  
  _model: ModelInfo  
}
```

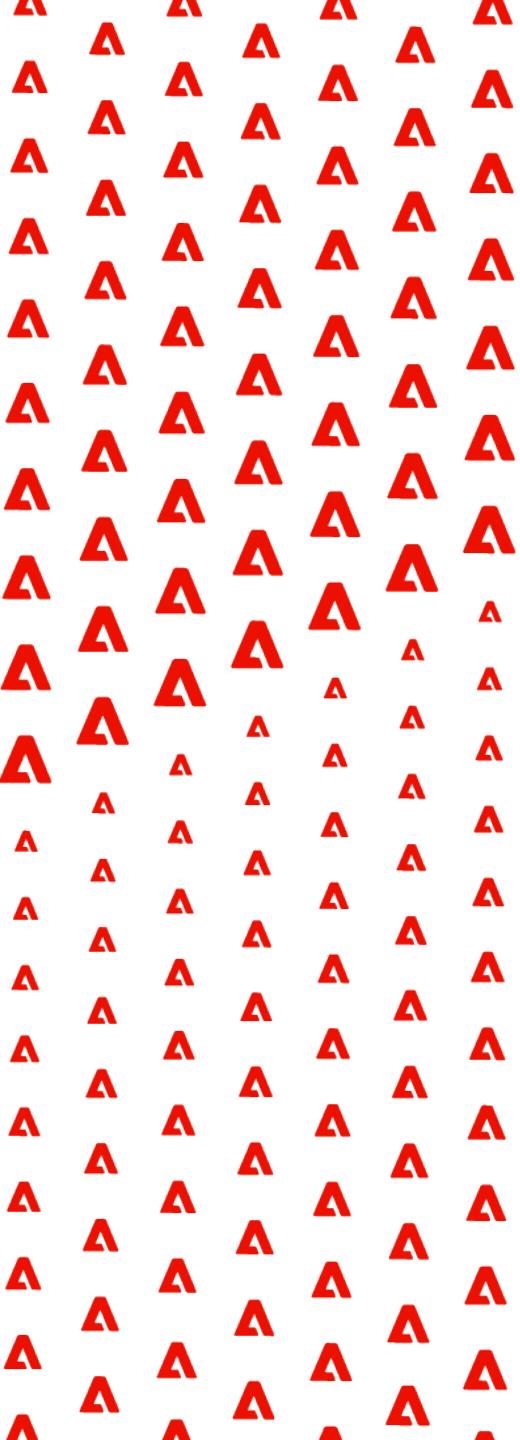
Content Fragment – Nice Book

The Famous Writer

{"importantProperty": " Important value"}

```
{  
  "data": {  
    "author": "The Famous Writer",  
    "info": {  
      "importantProperty": " Important value"  
    }  
  }  
}
```

Content Fragment Editor enhancements



Content Fragment editor enhancements

Structure tree

- Easily navigate through Content Fragment structure

Direct access to corresponding model

Direct editing of nested Content Fragments

The screenshot shows the Content Fragment editor interface. On the left, a 'Structure Tree' panel lists the fragment structure with nodes like 'author: Author', 'main: Main', and 'referencearticle: Reference article'. A red box highlights the 'referencearticle' node. Below it, the main content area displays the 'referencearticle' content, which includes the author 'Sofia Sjöberg', the main text 'My backpack feels heavier for every meter, slowly deforming my spine down my shoulders. My mouth tastes like blood, while salty pearls roll their way to my eyes. I look back up, I'm not there yet, just a few hundred more steps...', and various editing tools. A red box also highlights the 'Ski Touring Article' tab at the top right.

This screenshot shows a detailed view of the 'referencearticle' fragment. The left sidebar shows the 'alaskan-adventures' node selected. The main content area displays the text: 'dollar corporations charge us big money to ride down over-crowded slopes, many of us are returning to the original way of skiing. Exchanging lift queues for solitude, quantity for quality, and apres-ski beer for trail mix, a new generation of skiers are rediscovering the virtue of earning one's turns, the silence of the mountains, and how far into the unknown a little off-season cardio training can get you.' Below the text, a 'Reference article' section shows the URL '/content/dam/wknd/en/magazine/alaska-adventure/alaskan-adventures' and two buttons: 'Edit Content Fragment' (highlighted with a red box) and '+ New Content Fragment'.

Content Fragment editor enhancements

JSON Preview

- Generates a JSON output for of the current content fragment

Direct publishing

- To save & publish content fragment instantly*

The screenshot shows the Content Fragment editor interface. On the left is a sidebar with icons for edit, structure tree, comments, files, information, and search. The main area has tabs for Structure Tree, Preview, and Publish. The Preview tab shows a rich text editor with a toolbar and a preview of the content. The Publish tab shows basic metadata: 'Modified 17 seconds ago by Administrator' and 'Ski Touring Article'. A red box highlights the 'Publish' button. The search icon in the sidebar is also highlighted with a red box.

The screenshot shows the JSON Preview feature. It displays a JSON representation of the content fragment. The JSON code is as follows:

```
1  {
2   "data": {
3     "articleByPath": {
4       "item": {
5         "_path": "/content/dam/wknd/en/magazine/skitouring/skitouring",
6         "author": "Sofia Sjöberg",
7         "main": {
8           "html": "<p>My backpack feels heavier for every meter, slowly",
9           "markdown": "My backpack feels heavier for every meter, slowly",
10          "plaintext": "My backpack feels heavier for every meter, slowly",
11          "json": [
12            {
13              "nodeType": "paragraph",
14              "content": [
15                {
16                  "nodeType": "text",
17                  "value": "My backpack feels heavier for every meter, slowly"
18                }
19              ]
20            }
21          ]
22        }
23      }
24    }
25  }
```

* Upcoming feature, post GA.

GraphQL

Develop and test GraphQL queries

- Syntax highlighting
- Auto completion
- Schema documentation

The screenshot shows the GraphiQL interface with the following components:

- Toolbar:** Includes buttons for "GraphiQL" (with a play icon), "Prettify", "Merge", "Copy", and "History".
- Query Editor:** Displays a GraphQL query:

```
1 #all adventures
2 {
3   adventureList {
4     items {
5       _path
6       adventureTitle
7       adventurePrice
8       adventureTripLength
9       adventurePrimaryImage {
10      ... on ImageRef {
11        _path
12        mimeType
13        width
14        height
15      }
16    }
17  }
18}
```
- Result Panel:** Shows the JSON response from the query:

```
{
  "data": {
    "adventureList": {
      "items": [
        {
          "_path": "/content/dam/wknd/en/adventures/beervana-portland/beervana-in-portland",
          "adventureTitle": "Beervana in Portland",
          "adventurePrice": "$300 USD",
          "adventureTripLength": "1 Day",
          "adventurePrimaryImage": {
            "_path": "/content/dam/wknd/en/adventures/beervana-portland/AdobeStock_200192344.jpeg",
            "mimeType": "image/jpeg",
            "width": 1760,
            "height": 1175
          }
        },
        {
          "_path": "/content/dam/wknd/en/adventures/climbing-new-zealand/climbing-new-zealand",
          "adventureTitle": "Climbing New Zealand",
          "adventurePrice": "$900 USD",
          "adventureTripLength": "2 Days",
          "adventurePrimaryImage": {
            "_path": "/content/dam/wknd/en/adventures/climbing-new-zealand/AdobeStock_140634652.jpeg",
            "mimeType": "image/jpeg",
            "width": 1293,
            "height": 862
          }
        },
        {
          "_path": "/content/dam/wknd/en/adventures/cycling-tuscany/cycling-tuscany",
          "adventureTitle": "Cycling Tuscany",
          "adventurePrice": "$4500 USD",
          "adventureTripLength": "4 Days",
          "adventurePrimaryImage": {
            "_path": "/content/dam/wknd/en/adventures/cycling-tuscany/AdobeStock_141786166.jpeg"
          }
        }
      ]
    }
  }
}
```
- Variables Panel:** Shows a single variable "1" under the "QUERY VARIABLES" section.
- Documentation Panel:** On the right, it shows schema documentation for the "AdventureModel" type, including fields like `_logOp`, `_path`, `adventureTitle`, etc., each with a corresponding filter type (e.g., `LogOp` is `AND`, `_path` is `IDFilter`, etc.).

Note: GraphQL content-package is downloadable at: <https://experience.adobe.com/.../aem-graphql/graphiql-0.0.4.zip>

AEM headless: Persisted queries (sneak-peek)*

"GET" your "PUT" queries

- PUT your GraphQL query to save as a persisted query
- GET (execute) the persisted query for result
- Critical for high performance and scalability
- Support HTTP or CDN caching for queries and results

Persisted queries can be enabled via AEM configuration console*

1. Create persisted query through PUT

```
curl -X PUT \  
-H 'authorization: Basic YWRtaW46YWRtaW4=' \  
-H "Content-Type: application/json" \  
"https://<aem-cloud-service-host>/graphql/persist.json/wknd/your-query-name" \  
-d '{your-graphql-query}'
```

2. Check success result

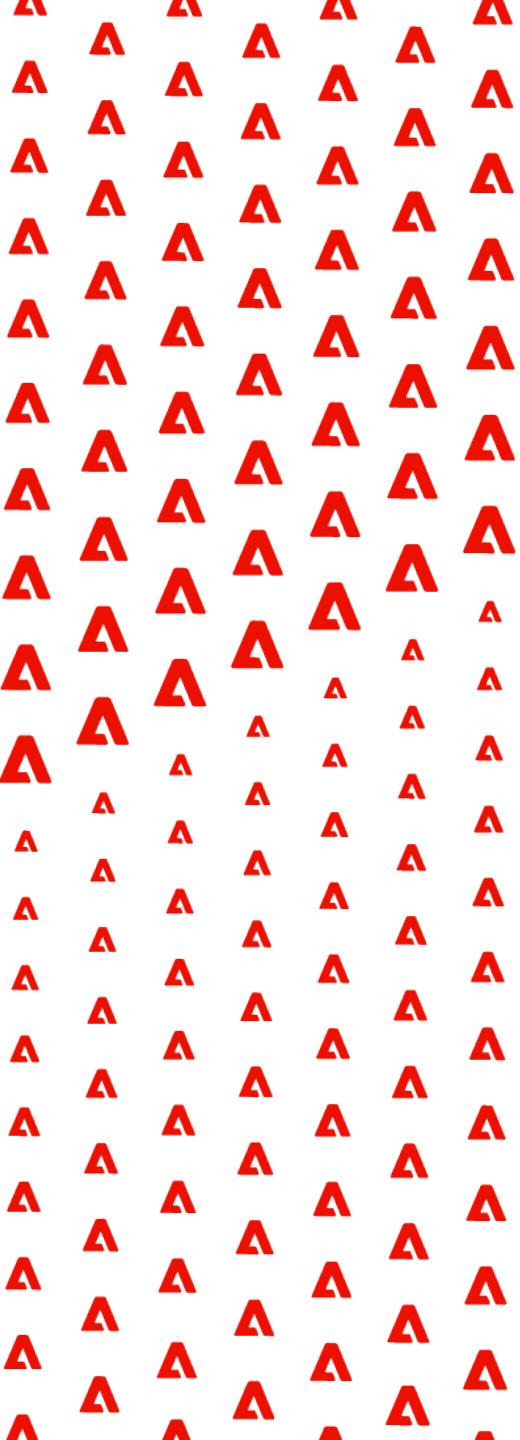
```
{  
  "action": "create",  
  "configurationName": "wknd",  
  "name": "your-query-name",  
  "shortPath": "/wknd/your-query-name",  
  "path": "/conf/wknd/settings/graphql/persistentQueries/your-query-name"  
}
```

3. Execute the persisted query using GET

```
curl -X GET \  
https://<aem-cloud-service-host>/graphql/execute.json/wknd/your-query-name
```

* Upcoming feature, post GA.

Headless – AEM GraphQL API Demo



AEM & headless: Wrap up

AEM headless capabilities **modernize and simplify** building new experiences

- **Enhances** existing content fragment **modeling capabilities**
- **Streamlines AEM UX** for authoring content fragments
- **Industry standard** for querying contents headless-ly



AEM & headless: What's next?

Jump start with headless features of AEM!

- Read more in the docs
 - [Headless and AEM](#)
 - [Explore AEM GraphQL APIs](#)



& start playing with AEM, headless-ly!

