adaptTo()

EUROPE'S LEADING AEM DEVELOPER CONFERENCE

28th – 30th SEPTEMBER 2020

GraphQL services in the AEM world

Mark J. Becker, Markus Haack - Adobe

About us

Markus Haack
Software Engineer
Adobe Germany
@mhaack

Mark J. Becker
Software Engineer
Adobe Basel
@ markjbecker

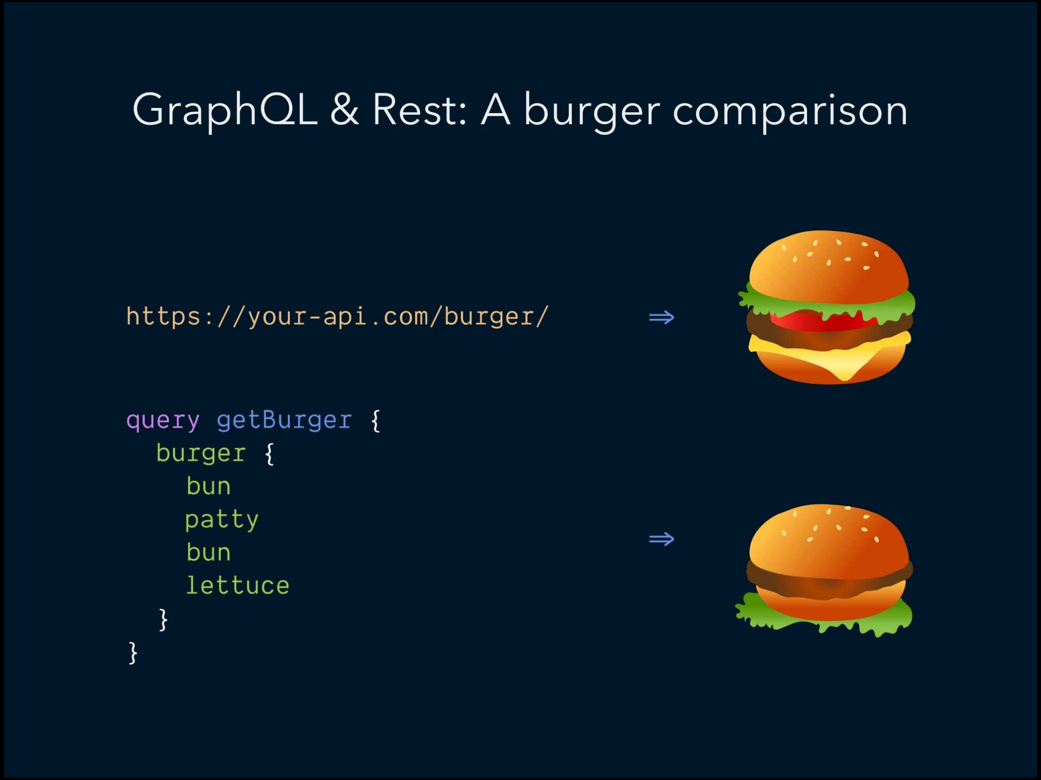
Agenda

- Introduction to CIF GraphQL
- Commerce Core Components
- GraphQL on
  - Server-side
  - Client-side
- GraphQL Integration Layer

adaptTo()

CIF GraphQL

Why GraphQL in AEM projects?



GraphQL & Rest: A burger comparison

```
https://your-api.com/burger/          ⇒
```

```
query getBurger {
  burger {
    bun
    patty
    bun
    lettuce
  }
}
```

Graphic Source: http://twitter.com/NikkitaFTW

Commerce Core Components

Commerce Integration Framework (CIF)

CIF = Adobe Experience Manager +

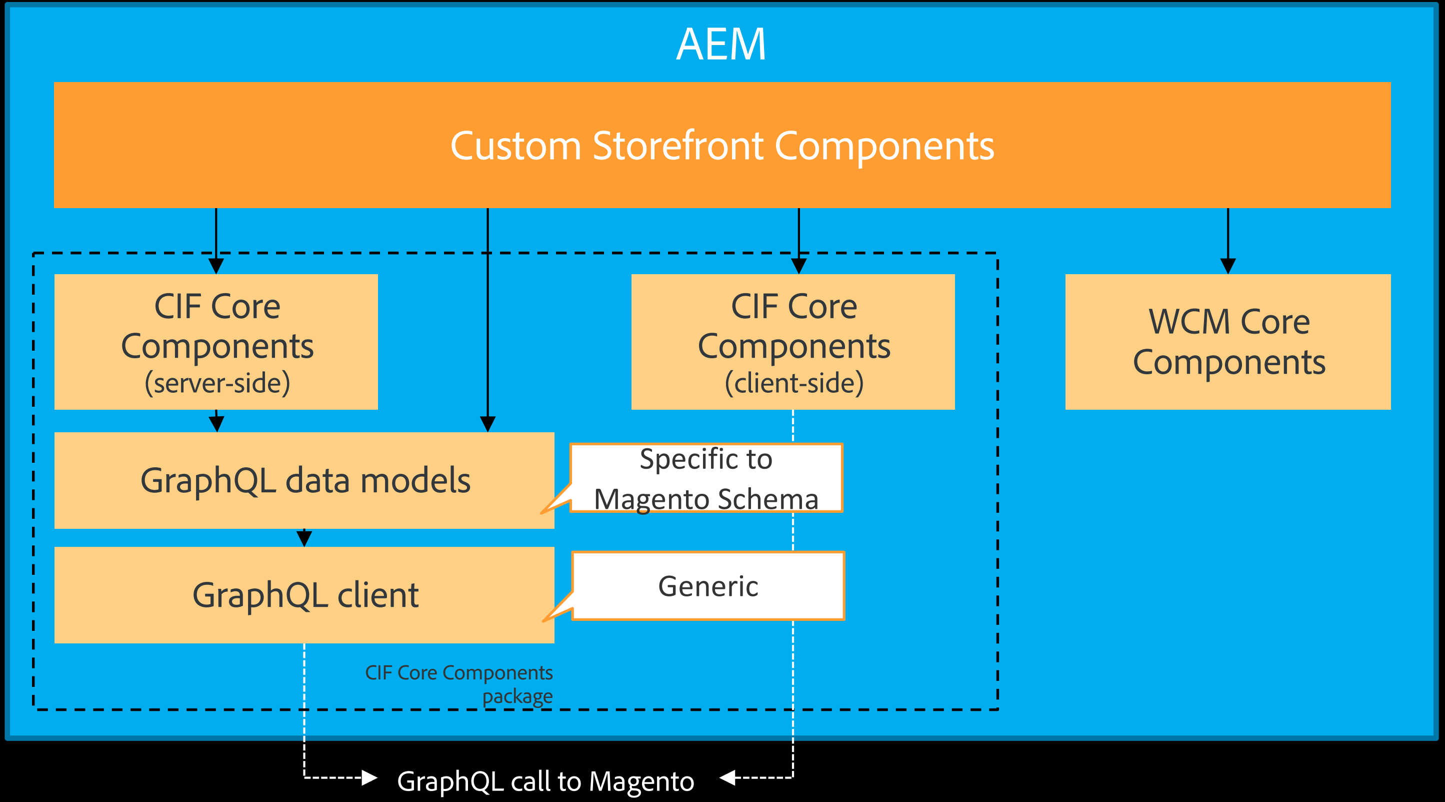
Core Components for Commerce use cases

adaptTo()

Commerce Core Components

- Classic AEM Components using Sling models & HTL templates
- Focus on commerce use cases
  - Retrieve and display product data, search, catalog navigation – server side
  - Cart & checkout flow, user account – client side
- GraphQL client linked to AEM site via Sling Context Aware Configuration
  - Flexible mappings & combinations of AEM site (using MSM) to multiple Magento stores
- CIF Extras:
  - AEM page templates commerce page types
  - SEO helpers for commerce pages

Commerce Core Components & GraphQL usage


AEM
Custom Storefront Components
CIF Core Components (server-side)
CIF Core Components (client-side)
WCM Core Components
GraphQL data models
Specific to Magento Schema
GraphQL client
Generic
CIF Core Components package
GraphQL call to Magento

Demo
Commerce Core Component Library

Server-side GraphQL

GraphQL client for all

- Generic **`GraphqlClient`** OSGI service
- Adapt it from resource, page or get the OSGI service directly
- CIF provides **`MagentoGraphqlClient`** for convenience (configures caching, HTTP headers, etc. special to Magento)
- Optional Java GraphQL models
  - generated from any GraphQL schema
  - provided via GitHub for Magento
  - Model class generator is open source as well

Usage of GraphQL client

- With generic GQL queries

```
query {
  products(search: "shirt", sort: { relevance: DESC }) {
    items {
      id
      sku
      name
      url_key
      updated_at
      thumbnail {
        url
      }
    }
  }
}
```

Usage of GraphQL client

- With generic GQL queries

```
String queryStr = "{products(search:\”shirt\",sort:{relevance:DESC})
{items{id,sku,name,url_key,updated_at,thumbnail{url}}}}";

GraphqlResponse<JsonObject, JsonObject> response = graphqlClient.execute(
    new GraphqlRequest(queryStr), JsonObject.class, JsonObject.class);

JsonObject query = response.getData();
JsonArray products =
    query.getAsJsonObject("products").getAsJsonArray("items");
```

Usage of GraphQL client

- With model classes

```
private SimpleProductQueryDefinition generateProductQuery () {
  return (SimpleProductQuery q) -> {
    q.id().sku().name().urlKey().updatedAt().thumbnail(t -> t.url());
  };
}

String filter = "shirt";
QueryQuery.ProductsArgumentsDefinition searchArgs = s -> s.filter(filter);
ProductsQueryDefinition queryArgs = q -> q.items(generateProductQuery());

GraphqlResponse<Query, Error> response = graphqlClient.execute(query.toString());

Query rootQuery = response.getData();
List<ProductInterface> products = rootQuery.getProducts().getItems();
```

Client-Side GraphQL

Client-Side GraphQL

- Displaying data that is visitor specific, dynamic or difficult to cache
- Enhance server-side components
- React Components with Apollo GraphQL client
- Full SPA/PWA or mixed page
- Same Sling CA config as server-side components

Declarative Data Fetching

- Query state managed by Apollo
- Caching
  - HTTP layer caching (via GET requests)
  - In-Memory caching
- Mutation can update state

```
const { data, loading, error } = useQuery(QUERY_GET_CART);
const [addProduct] = useMutation(QUERY_ADD_PRODUCT);
```

Customization with React Context

- Composition
- Encapsulate component logic in re-useable context

```
const App = () => (
    <CartContext>
        <MyCart />
    </CartContext>);

// MyCart
const [products, { add, remove }] = useCartContext();
```

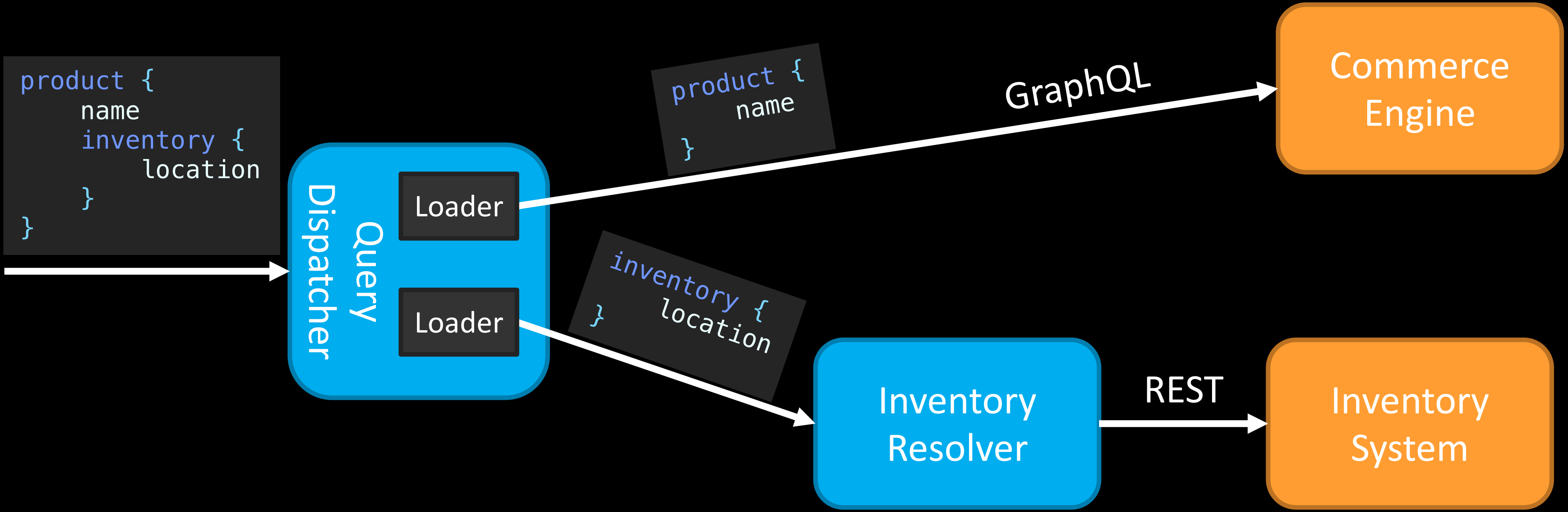
Demo
React Component Customization

GraphQL Integration Layer

Serverless GraphQL Server

- Implement your own GraphQL server to integrate non-GraphQL services
- Runs on Apache OpenWhisk
- Extend, merge or customize existing schemas
- Schema delegation to split server into multiple schemas and actions

Query Execution



Q&A

adaptTo()

References

- CIF Core Components
  https://github.com/adobe/aem-core-cif-components
- CIF GraphQL Client
  https://github.com/adobe/commerce-cif-graphql-client
- Magento GraphQL data models and query builders
  https://github.com/adobe/commerce-cif-magento-graphql
- GraphQL Java Generator
  https://github.com/adobe/graphql-java-generator
- GraphQL Reference Implementation
  https://github.com/adobe/commerce-cif-graphql-integration-reference
- Venia Reference
  https://github.com/adobe/aem-cif-guides-venia
- Magento GraphQL Schema
  https://devdocs.magento.com/guides/v2.4/graphql/
- Serverless GraphQL on Adobe I/O Runtime (Medium)
  https://medium.com/adobetech/serverless-graphql-on-adobe-i-o-runtime-e221d2a8e215