Integration patterns, Magnolia Monolith vs. Microservices

Date

05 Dec 2018

Brief

Magnolia Monolith vs. Microservices vs. Macroservices

Christopher Kaiser

Integration Patterns

Discussion. We work to improve the speed and ease of implementing integrations with Magnolia. We also work to improve their maintainability and User experience.…

Christopher Zimmermann, Product Manager at Magnolia

Attendees

- Christopher Zimmermann
- Christoph Meier
- Mikaël Geljić
- Antti Hietala
- Jan Haderka
- Aleksandr Pchelintcev
- Christopher Kaiser

Discussion

Some move logic out of Magnolia now

(unlike the old days of Magnolia doing everything.)

- Past times; let magnolia do everything (data mgmt, rendering)
    - nowadays, give business logic to a remote service
- Connect with e.g. headless shop systems; is Magnolia the right place to host product data?
    - going domain-driven, Magnolia = Content hub; all integrations rather on the client side not ideal for the editor; choose the product they want, etc.

Macroservice Architecture. Keep external data outside of Magnolia, has benefits.

- Other approach: architecture, 3 macro services:
    - product db
    - magnolia
    - elasticsearch
    - all integration from frontend; (e.g. rendered from hybris)
    - edge-side-includes (header/footer served from cms, main product component from product db), processed w/ varnish
    - Chris mentioned: Magnolia as a "Jupiter" in a macroservice "constellation".
- Similar case from backend systems;
    - how much do you store in the CMS? export your product db or access only when needed
    - call when page is rendered? layer of caching in the CMS?
- Lot of data in (old) PIM system & lot in Magnolia too;
    - sporadic updates, diverging
- Dynamic data (pricing, campaigns); distributed product db
    - integration to the catalog from Magnolia
- Content apps that expose external content (ie they are not JCR-based) solves one side of the problem
- Crossing network boundaries is where painpoint comes. Speed of the various content sources can be an issue.
- Keeping index consistency on author, public, reacting to data changes (JCR observation; hard for external content)

- With macro-service and micro-service world - Orchestration becomes a challenge.

Microservices are popular. They are getting smaller and smaller. But can we have too many?

Contract testing becomes important.

Some avoid using Java where possible because restart of Magnolia required.

- Configure without coding, **Webhooks** would be nice, event descriptors
- Call remote services from freemarker;
    - restfn no go if requires the java service class, recompilation etc.
- Java-class is a killer, needs a restart; pass headers onto the remote system

It would be handy if REST endpoints follows a spec.

- need an REST API w/ a spec (open API, swagger), 2 requirements
    - mock to test in frontend
    - work with real data, safe anywhere
- Also want to easily put data into magnolia via open api endpoint.
- When jendpoint has open-api spec - its "discoverable" and you can interface with it more automatically.

Magnolia's short-term improvement plans:

- building upon Light development, Content Types, REST Delivery API
- Declarative REST clients coming via YAML (baseUrls, connection details, methods proxied) => remove the need for the Java proxy interface
- Content Types will be used to make Magnolia aware of *external* content too (REST datasource definition)
- New UI framework will be able to infer default Content app for *external* content types
- Delivery endpoint may also proxy/serve "remote" content types too, eventually