Phoenix

Goal

Explore emerging patterns for building systems that manage and distribute content[1].

The jump from wiki software to modern web architecture (for example, a reactive system) is huge and difficult to conceptualize. Experimenting with core patterns enables us to gain insight into the possibilities and potential impossibilities.

Primary focus

Canonical data modeling[2][3] allows content to be understood by people, programs and machines outside the traditional boundaries of MediaWiki. And, as far as possible, allows consumers[4] to request only what they need.

Building this model requires defining boundaries around parts and their interrelationships. For example, a page has parts (sections) and is also part of collections (about the same topic). Our work here includes:

- Define a predictable structure[5] using industry-standard formats like schema.org (to support predictability and reusability)
- Break down prexisting structures (all the content on the Philadephia page) into parts (a section on the History of Philadephia) and establish interrelationships between the parts (to support "only what they need") using hypermedia linking.[6]
- Enhance the structure with contextual information by associating parts with Wikidata (to enable natural collections like US Cities) and indexing collections with Elasticsearch.
- Enable interaction with the structure via API calls. Multiple API calls can be wrapped into a single payload -- or not.

Working draft of our CDM

Note: Honestly, we don't know if it's humanly possible to "structure" Wikipedia documents. We are identifying the Biggest Challenges so we can raise them and resolve them organizationally.

Secondary focus

Loose coupling:[7] New ways to interact with, enhance or process content (capabilities)

operate independently and are built on top of (or adjacent to) the data model.

Event-based interactions:[8] activities in the system happen only when they need to happen (asynchronously) with only the information they need to accomplish their aim.

CQRS:[9] The current structure inside of MediaWiki is left alone. When changes happen in MW, the new system reacts by getting the necessary information and translating it into the canonical data model.

Initial Model of how this works

Implementation

We have purposefully defined the implementation toolset for this PoV. We did this so we can focus on the patterns, which present signification challenges, *before deciding which are valuable long term.* By working in AWS, we don't need to build that toolset.

Our next step is not to put this exact toolset into production. Our next step is to collectively design an infrastructure that supports the value while also considering the tradeoffs.

What about editing?

We are beginning to model editing events now. This approach is designed to work *with* any CMS, as part of a system, not replace it. We will be carefully thinking through which patterns apply, which don't, and which need to be included that aren't here.

Upcoming use case

We've modeled a number of use cases, some in partnership with the Structured Data Across Wikipedia team, that will benefit from these patterns. When our initial work is complete, we will be prototyping with the Mobile team, specifically focused on References.

Definitions and resources

- [1]: Content is the free knowledge / information being shared on a wiki page about a subject.
- [2]: Enterprise Integration Patterns: CDM
- [3]: CDM pitfalls and approaches)
- [4]: Consumers are people, programs (like a front-end application) and machines who are

using content outside the context of MediaWiki. MediaWiki could also be a consumer.
- [5]: Model your Application Domain, not your JSON structures
- [6]: Hypermedia linking
- [7]: Loose coupling is an approach to interconnecting the components in a system so that those componentsdepend on each other to the least extent practicable.
- [8]: Event-driven architecture on English Wikipedia
- [9]: Command Query Responsibility Segregation (CQRS) means that the data model for reading doesn't have to be the same as the model for updating.

Content descriptions

	Description
`common`	Common structures, helpers, etc
`env`	Package for project-wide constants (AWS account & resource information)
`event-bridge`	Send filtered change events to an SNS topic
`lambdas/fetch-changed`	Subscribe to change events and download the corresponding Parsoid HTML to an S3
`lambdas/fetch-schema.org`	Create schema.org JSON-LD output from Wikidata, and upload to S3. Triggered when HTML is added to `incoming/` (see `lambdas/fetch-changed`)
`lambdas/merge-schema.org`	Merge JSON-LD with HTML documents, and upload to S3. Triggered when linked data is added to `schema.org/` (see `lambdas/fetch-schema.org`)