Linked Data Fragments Call with Ruben Verborgh 2015-04-14

Meeting on 2015-04-14

Present: Steven Anderson, Corey Harper, Tom Johnson, Trey Terrell, Ruben Verborgh

Corey introduced topic. Caching issues for remote resources. How to do this?

Trey: Caching isn’t hard because triples change a lot, it’s hard because servers are often down.

Tom: DCMI Types -- Hitting cache that sits in ruby, but for bigger data sets, large sizes, this will be a challenge

• DPLA interested in reconciliation endpoint, but also for front-end like Trey’s use case
• LD Fragments servers for others to use as a potential DPLA goal

Trey: Thinks he understands the goal of TPF & LDF, but wants to hear it. What is the problem set.

Ruben:

• Availability on the Web.
• Data Dump & do stuff locally -- or --
• query live.
• Many data sets aren’t queriable.
• Those that are suffer from downtime
• LDF is a conceptual framework to say “This API offers that kind of fragments”
• Data dumps have one fragment: the entire dataset
• SPARQL endpoints have many highly specific fragments, many of which are expensive to compute
• Can we find different types of fragments that divide the workload differently? (Triple Pattern Fragments are an example.)
• Moves some intelligence and business logic to client side.
• Clients solve complex queries by splitting them into smaller queries the server can handle, depending on its interface.

Q from Trey: High availability theory is that server does less work, so easier to keep up

Ruben: That’s part of it. Most APIs on the Web have far less expensive requests than SPARQL endpoints.

• 1st, the TPF API is low-cost for the server.
• 2nd, Web is optimized for caching.
• Overlapping questions can reuse same fragments and be more cacheable
• Publication (http://linkeddatafragments.org/publications/iswc2014.pdf) includes evidence that availability data & cost-data per request, this is cheaper

Pushing this to the client side -- Ability to combine from multiple data streams

• Trey’s Primary use case is “I have stuff, I need labels”
• Can have an interface that says, ask me for a subject, I’ll always give you the label
• Layers of abstraction

Reconciliation:

• Now experimenting with full text search
• Corey: Question about ranking, probabilistic matching.
• Ruben, these examples have some rank, since from Elastic Search
• Could have interfaces that supported explicitly scored
• Corey: Even support “just give me your top match” interfaces
• Some LD Frags might take responsibility for the ranking

• this is powerful, since we don’t trust LC server ranking
• could support different ranking methods
• Reliably combine different data sources

Reusability:

• Support for multiple interfaces, which are composed of interface features
• Allows us to keep a lot of this functionality out of Hydra, have a nice clear interface, separation of concerns goodness
• Figure out which interfaces are useful to whom
• This allows for reusable interfaces

Next steps:

• Hydra folks should think about a ruby implementation.
• Spec exists.
• Implementations in JavaScript, Java, Perl
• Tom’s interested in setting up a geospatial frags server & integrating with Two Fishes
• Reverse geocoding