TELEIOS 3rd User Community Workshop
12 - 14 June 2013

Session 3 - Query Builder User Interface

Konstantinos Kravvaritis

Spatial Information Management

Fraunhofer Institute for Computer Graphics Research IGD
Session Outline

- Introduction
- Basic Concept - Demonstration
- System Overview
- Roadmap
- Questions & Discussion
Introduction - What we had
DLR's EOWEB

Queries available EO products by selecting...

...bounding rectangle
...acquisition time range
...classes / types from EO product type taxonomy

technically speaking:

- Requires matching Java plugin version available and installed
- Certain browsers / platforms have issues even if correct plugin version is installed and enabled
- What about organizational security policies: disabled plugins, firewalls, (filtering-)proxies, ... ?
Introduction - What we're also working on: Real Time Fire Monitoring

Designed primarily with specific NOA Use Case in mind

UI provides facilities that allow to easily perform a complex but rather limited set of queries

What if (other) users actually (also) want to do something else?

- Add query constraints involving other aspects of the data?
- Have more control on the structure of the queries performed?
- Run the HMI (or parts thereof) on different data sets without a lot of effort?
- Add and use additional data sets within the system without a lot of effort?
Among the User Stories: “I want to...”

- “...query the products based on their semantic content...”
- “...save the parameters of a query and share them with my colleagues...”
- “...retrieve information about fires using semantic, geographical and temporal constraints such that I can mash-up the retrieved information with information from other systems...”

Among derived requirements:

Develop various new user interfaces while...

“These user interfaces shall be adaptable to the particular needs of the individual users and their conjectures and shall provide extended EOWEB portal capabilities.”
Introduction - What we're also working on: “Semantic Technology”

For NOA and DLR Use Case:

- Ontologies that “capture” semantic structure of the available meta-data have been developed
- Existing meta-data can be converted into stRDF
- Meta-data is stored in Strabon and can be queried via stSPARQL

Well, couldn't we exploit all that for a HMI approach which allows end users to query EO product meta-data in a more flexible way?
Let's design a new HMI component...
...the Query Builder

- Impose as few restrictions as possible on query formulation against semantically structured, spatial data sets
- Maintain usability for end users
- Let meta-data ontologies “drive” large parts of the UI facilities
- Stick to HTML5, avoid plugins, be cross-browser compliant
Basic Concept – What we came up with: ...connecting some dots...

- Allow users to...
  - reuse and adapt predefined queries
  - save and share queries
  - express complex spatial / topological constraints

- Provide different interfaces for different users:
  - “Graph-based editor”
  - text based stSPARQL editor
Basic Concept: Filter Facet Boxes

provide Facet Box Types for:

- taxonomies
- spatial location / extent
- “kinds of topologies”
- generic box type for primitive properties
Basic Concept:
Graph-Based Query Editor

Select all patches corresponding to water in a specified region

```
SELECT ?g 
WHERE {
  ?p dlr:hasGeometry ?g .
  ?p dlr:hasLabel ?l .
  ?l dlr:correspondsTo dlr:Water .
  FILTER (strdf:intersects(?g, "POLYGON ((12.301451 45.40493,12.398127 45.416817,12.386066 45.46502,12.289288 45.45313,12.301451 45.40493))"^^strdf:WKT)) .
}
```
Basic Concept: Graph-Based Query Editor

Select all patches corresponding to all subclasses of water in a specified region

```
SELECT ?g
WHERE {
  ?p dlr:hasGeometry ?g .
  ?p dlr:hasLabel ?l .
  ?annotation rdfs:subClassOf dlr:Water .
  FILTER (strdf:intersects(?g, "POLYGON ((12.301451 45.40493,12.398127 45.416817,12.386066 45.46502,12.289288 45.45313,12.301451 45.40493))"^^strdf:WKT)) .
}
Basic Concept: Graph-Based Query Editor

Select hotspots in Athens, derived from MSG1_RSS, produced by DynamicThresholds, for 2012-08-19T17:30:00

```sparql
SELECT ?h
WHERE {
  ?h rdf:type noa:Hotspot ;
  noa:isDerivedFromSensor "MSG1_RSS"^^xsd:string ;
  noa:hasAcquisitionTime "2012-08-19T17:30:00"^^xsd:dateTime ;
  noa:producedFromProcessingChain "DynamicThresholds" .
  gag:hasMunicipality Athens .
}
```
System Overview

Webbrowser
- Query Builder UI
- JS API calls
- QB adapter
- other HMI components

Query Builder Server
- User Management / Access Control
- Meta-data ontology to filter facet box configuration mapper
- Query model to stSPARQL mapper
- Alignments, queries, presets

LDAP Server
- User Credentials, Groups, Roles

Strabon
- EO product meta-data, meta-data ontologies, additional GIS datasets

UIDL / HTTP

LDAP

stSPARQL
System Overview – other involved technologies / frameworks

Client:

- Vaadin (+ addons)
- GWT
- OpenLayers
- Web-Browser

Server:

- Main application is OSGi RFC 66 compliant Web Application Bundle (WAB)
- Entirely structured as set of OSGi bundles
- Pure Java

- JavaScript Application
- Exposes JavaScript API
- XS-support
- Developed mostly in Java

- Vaadin (+ addons)
- Eclipse Gemini Web
- TinkerPop
- Apache Tomcat
- Eclipse Blueprint
- Neo4J
- Eclipse Equinox
- Java Runtime Environment
Roadmap

**Focus on:** Transformation from Graph-based queries to text-based

**Next Milestone:** a complete iteration of the following steps
graph-based query creation → text-based query transformation → getting back results from Strabon

- Persistence for Queries, Presets

**TBD:**

- Query Sharing
- LDAP Integration
- Implementation of other Query Editors (hierarchical filter facets, text based)
- Linked Data Integration
Open Questions & Discussion

Questions?

Suggestions?