

SPARQL, DBPedia, and SPARQL extensions

Inigo Surguy

Oxfordshire SWIG, 6th March 2008

What is SPARQL?



```
SELECT distinct ?chalkboard_gag WHERE {  
  ?episode skos:subject ?season .  
  ?season rdfs:label ?season_title .  
  ?episode dbpedia2:blackboard ?chalkboard_gag .  
  FILTER (regex(?season_title, "The Simpsons episodes,  
season")) .  
  FILTER (lang(?chalkboard_gag)="en")  
}  
ORDER BY ?season
```



Based on an example by Bob DuCharme

" "I will not waste chalk""@en
" "I will not call my teacher 'hot cakes'""@en
" "I will not skateboard in the halls""@en
" "They are laughing at me, not with me""@en
" "I will not instigate revolution""@en
" "I will not yell 'Fire' in a crowded classroom. ""@en
" "I will not draw naked ladies in class""@en
" "Garlic gum is not funny""@en
""I did not see Elvis"; one line reads "I did see Elvis""@en
" "I will not burp in class""@en
" "I am not a licensed hairstylist. ""@en



```
SELECT ?artist_name ?album_title WHERE {  
  ?album dbpedia2:artist ?artist  
  . ?album rdfs:label ?album_title  
  . ?artist rdfs:label ?artist_name  
  . ?artist dbpedia2:origin ?origin  
  . FILTER (?origin = <http://dbpedia.org/resource/Oxford> )  
  . FILTER (lang(?artist_name)="en")  
}
```



"Ride (band)"@en "Ride (EP)"@en
"Ride (band)"@en "Smile (Ride)"@en
"Ride (band)"@en "Tarantula (album)"@en
"Ride (band)"@en "Today Forever"@en
"Ride (band)"@en "Waves (BBC Radio One sessions)"@en
"Supergrass"@en "Alright/Time"@en
"Supergrass"@en "Caught by the Fuzz"@en
"Supergrass"@en "Diamond Hoo Ha"@en
"Supergrass"@en "Rush Hour Soul (song)"@en
"Supergrass"@en "Seen the Light"@en
"Supergrass"@en "St. Petersburg (song)"@en
"Supergrass"@en "Supergrass (album)"@en
"Supergrass"@en "Supergrass Is 10"@en
"Swervedriver"@en "Mezcal Head"@en
"Swervedriver"@en "Raise (album)"@en



```
SELECT ?subject ?label ?lat ?long WHERE {
  dbpedia:Radcliffe_Camera geo:lat ?myLat .
  dbpedia:Radcliffe_Camera geo:long ?myLong .
  ?subject geo:lat ?lat.
  ?subject geo:long ?long.
  ?subject rdfs:label ?label.
  FILTER(xsd:float(?lat) - xsd:float(?myLat) <= 0.05 &&
         xsd:float(?myLat) - xsd:float(?lat) <= 0.05 &&
         xsd:float(?long) - xsd:float(?myLong) <= 0.05 &&
         xsd:float(?myLong) - xsd:float(?long) <= 0.05 &&
         lang(?label) = "en").
} LIMIT 20
```

Based on an example from the DBPedia wiki

SPARQL results:

subject	label
:Oxford_Science_Park	"Oxford Science Park"@en
:Kassam_Stadium	"Kassam Stadium"@en
:South_Hinksey	"South Hinksey"@en
:Littlemore	"Littlemore"@en
:Rose_Hill%2C_Oxfordshire	"Rose Hill, Oxfordshire"@en
:Iffley_Lock	"Iffley Lock"@en
:Iffley	"Iffley"@en
:Cowley%2C_Oxfordshire	"Cowley, Oxfordshire"@en
:North_Hinksey	"North Hinksey"@en
:Donnington_Bridge	"Donnington Bridge"@en
:New_Hinksey	"New Hinksey"@en
:Jesus_College_Boat_Club_%28Oxford%29	"Jesus College Boat Club (Oxford)"@en
:Grandpont%2C_Oxford	"Grandpont, Oxford"@en
:St_Bartholomew%27s_Chapel%2C_Oxford	"St Bartholomew's Chapel, Oxford"@en
:Folly_Bridge	"Folly Bridge"@en
:St_Hilda%27s_College%2C_Oxford	"St Hilda's College, Oxford"@en
:Campion_Hall%2C_Oxford	"Campion Hall, Oxford"@en
:Botley%2C_Oxfordshire	"Botley, Oxfordshire"@en
:Christ_Church_Cathedral%2C_Oxford	"Christ Church Cathedral, Oxford"@en
:Pembroke_College%2C_Oxford	"Pembroke College, Oxford"@en



**What if you don't know
the co-ordinates?**

A (property) extension function

```
public class GeocodeExtensionFn extends PFuncAssignToObject {
    private final Geocoder geocoder;

    public GeocodeExtensionFn(Geocoder geocoder) {
        this.geocoder = geocoder;
    }

    public Node calc(Node node) {
        try {
            String s = node.getLiteralValue().toString();
            Placemark placemark = geocoder.lookup(s);
            return Node.createLiteral(
                placemark.getLatitude()+" "+placemark.getLongitude());
        } catch (Exception e) {
            return Node.createLiteral("");
        }
    }
}
```

In normal Java use:

```
PropertyFunctionRegistry.get().  
    put("http://surguy.net/sparql/functions#latLngFn",  
        GeocodeExtensionFn.class) ;
```

For Joseki

```
joseki:initialization  
[ module:implementation  
    [ module:className <java:net.surguy.rdf.joseki.JosekiInitializer> ;  
      rdfs:label "Property function initializer" ; ]  
] .
```

```
public class JosekiInitializer implements ServerInitialization {  
    public void init(Resource service, Resource implementation) {  
        PropertyFunctionRegistry.get().  
            put("http://surguy.net/sparql/functions#latLngFn",  
                GeocodeExtensionFn.class);  
    }  
}
```

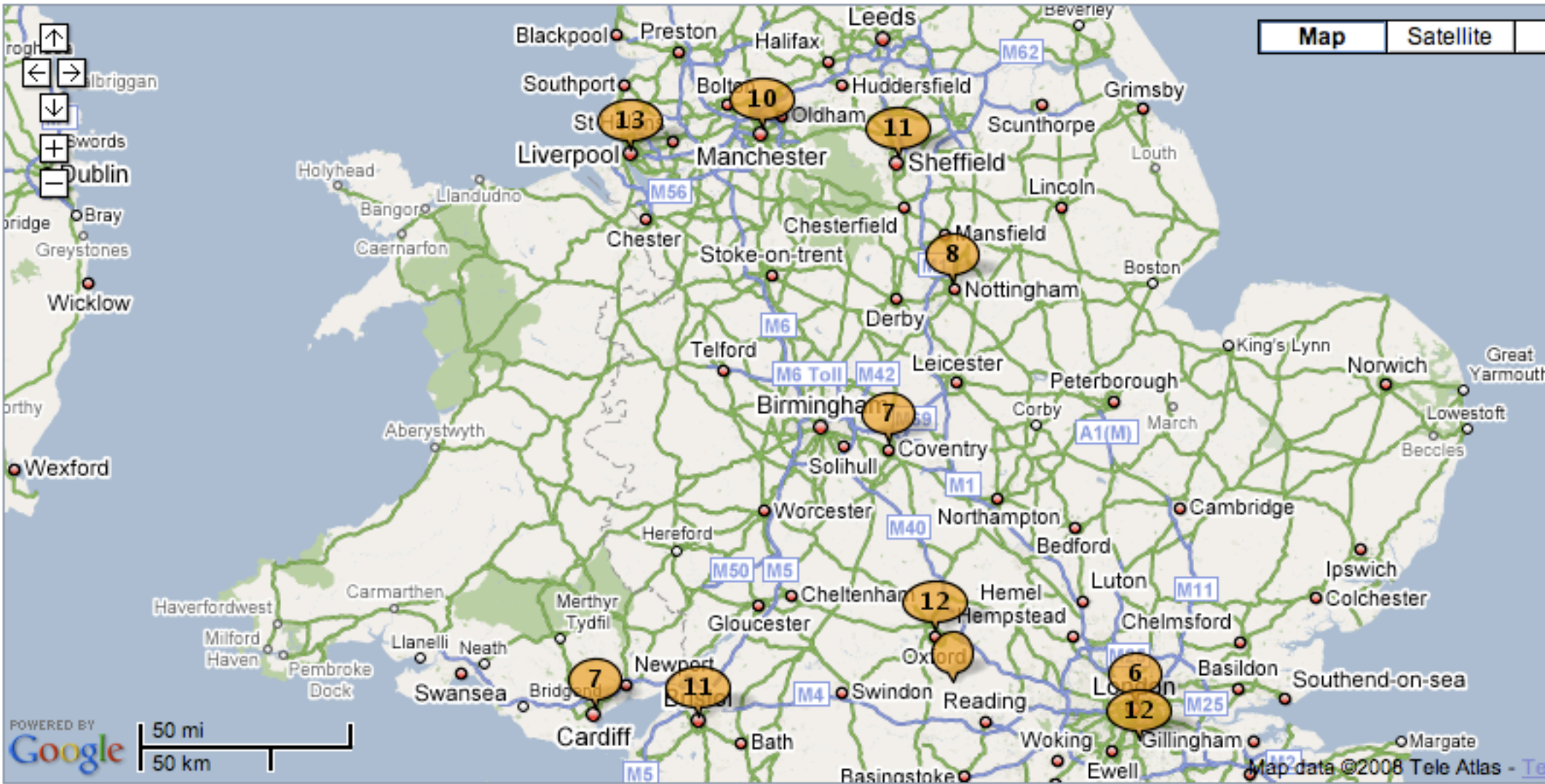
PREFIX myfn: <<http://surguy.net/sparql/functions#>>

```
SELECT * WHERE {  
  "Oxford" myfn:latLngFn ?result  
} LIMIT 10
```

```
select * where {  
  ?id_account a vocab:accounts  
  .?id_account vocab:accounts_name ?label_name  
  .?id_account vocab:accounts_billing_address_city ?geo_city  
  .?geo_city myfn:latLngFn ?addressLatLng  
}
```

TILES · MAP

141 account



What common SPARQL functions do we need?

- For XSLT, the EXSLT library has dates, dynamic, math, random, regex, sets, strings
- ARQ has filter functions for strings, maths, RDF graph, booleans
- ARQ has property functions for lists, text search, bags
- What about other SPARQL implementations?
- What can't be done in standard SPARQL?
- What else do we need?