2010

Technical Report

SimpleFeatureService

Ing. Andrea Aime

Ing. Simone Giannecchini

GeoSolutions S.A.S.

Date 10/11/2010

Version 0.1

Contents

GeoSolutions S.A.S
Via Poggio Alle Viti 1187
55054 Massarosa (LU) Italy
Tel: +390584962313 Fax: +390584962313
http://www.geo-solutions.it
info@geo-solutions.it

No table of figures entries found.

GeoSolutions S.A.S
Via Poggio Alle Viti 1187
55054 Massarosa (LU) Italy
Tel: +390584962313 Fax: +390584962313
http://www.geo-solutions.it
info@geo-solutions.it

Record of Changes

Version	When	Who	What
0.1	10/11/201	AAime	First Version

GeoSolutions S.A.S
Via Poggio Alle Viti 1187
55054 Massarosa (LU) Italy
Tel: +390584962313 Fax: +390584962313
http://www.geo-solutions.it
info@geo-solutions.it

SimpleFeatureService protocol

The OpenDataStore protocol is a simplified feature protocol using GeoJSON as the encoding format loosely based on RESTful principles (but not fully compliant with them to allow for better performance and handling of large requests).

It is heavily inspired by the MapFish protocol, but with modifications to make it possible to build an efficient GeoTools data store on top of it and cope with more real world use cases.

GeoTools already provides an implementation of a streaming parser/encoder as well as a Mapfish protocol oriented (but still incomplete) data store.

Supported URLs

- http://someserver:port/.../capabilities: used to provide a list of available layers
- http://someserver:port/.../describe/layername: returns a description of the feature type
- http://someserver:port/.../data/layername: used to query the layer features
- http://someserver:port/.../data/layername/fid: used to grab a specific feature by id

Capabilities document

The capabilities document lists the available layers and provides the basic information about the service: name of the layers, their bounding box and the native SRS.

The pattern is as follows:

```
[
    {
    "name": "layer1",
    "bbox" : [      -10,    -40,    30, 80],
    "crs" : "urn:ogc:def:crs:EPSG:4326",
```


GeoSolutions S.A.S
Via Poggio Alle Viti 1187
55054 Massarosa (LU) Italy
Tel: +390584962313 Fax: +390584962313
http://www.geo-solutions.it
info@geo-solutions.it

```
        "axisorder" : “yx”
            },
            {
            "name": "layer2",
            "bbox" : [ 15000000, 49000000, 18000000, 52000000 ],
            "crs" : "urn:ogc:def:crs:EPSG:32632"
            },
        ...
    ]
```


Following the GeoJSON specification suggestions the crs is expressed in urn form. If a layer for any reason has flipped axis it will indicate so by using the axisorder parameter, if that information is missing the layer is supposed to be in x/y, lon/lat order regardless of the notation used to express the crs.

GeoSolutions S.A.S
Via Poggio Alle Viti 1187
55054 Massarosa (LU) Italy
Tel: +390584962313 Fax: +390584962313
http://www.geo-solutions.it
info@geo-solutions.it

Layer description document

The layer description document lists the available fields and their type:

```
[ {"name":"string","quantity":"number", ...} ]
```

For example, the topp:states GeoServer demo layer description would look like:

```
[ {"the_geom":"MultiPolygon", "STATE_NAME":"string",
"STATE_FIPS":"string", "SUB_REGION":"string", "STATE_ABBR":"string",
"LAND_KM":"nunumber", "WATER_KM":"number", "PERSONS":"number",
"FAMILIES":"number", "HOUSHOLD":"number", "MALE":"number",
"FEMALE":"number", "WORKERS":"number", "DRVALONE":"number",
"CARPOOL":"number", "PUBTRANS":"number", "EMPLOYED":"number",
"UNEMPLOY":"number", "SERVICE":"number", "MANUAL":"number",
"P_MALE":"number", "P_FEMALE":"number", "SAMP_POP":"number" } ]
```

GeoJSON by default supports only the basic geometry types, strings and numbers. This specification supports:

- geometric types: Geometry, Point, MultiPoint, LineString, MultiLineString, Polygon, MultiPolygon, Geometry Collection
- field types: string, number, boolean, timestamp

A number can be expressed in integral or floating point notation. A boolean can be ‘true’ or ‘false’

A timestamp is expressed as a Unix date (number of seconds since Jan 1 1970), see also http://weblogs.asp.net/bleroy/archive/2008/01/18/dates-and-json.aspx

GeoSolutions S.A.S
Via Poggio Alle Viti 1187
55054 Massarosa (LU) Italy
Tel: +390584962313 Fax: +390584962313
http://www.geo-solutions.it
info@geo-solutions.it

JSON can in theory describe complex structures in which an attribute is a complex one containing sub-attributes. This first protocol specification won't support that case.

GeoSolutions S.A.S
Via Poggio Alle Viti 1187
55054 Massarosa (LU) Italy
Tel: +390584962313 Fax: +390584962313
http://www.geo-solutions.it
info@geo-solutions.it

Layer querying

This follows closely the MapFish protocol, with one variation, crs in the place of epsg to be consistent with the GeoJSON notation, a output mode to support count and bound operations, and hints to allow for implementation specific parameters to be passed over:

- no_geom=true: so that the returned feature has no geometry ("geometry": null)
- attrs={field1}[,{field2},...]: to restrict the list of properties returned in the feature
- limit={num}: limit the number of features to num features (maxfeatures is an alias to limit)
- offset={num}: skip num features
- order_by={field}: order the features using field
- dir=DESC|ASC: determine the ordering direction (applies only if order_by is specified)
- lon={x}: the x coordinate of the center of the search region, this coord's projection system can be specified with the epsg parameter
- lat={y}: the y coordinate of the center of the search region, this coord's projection system can be specified with the epsg parameter
- tolerance={num}: the tolerance around the center of the search region, expressed in the units of the lon/lat coords' projection system
- box={xmin,ymin,xmax,ymax}: a list of coordinates representing a bounding box, the coords' projection system can be specified with the epsg parameter
- geometry={geojson}: a GeoJSON string representing a geometry, the coords' projection system can be specified with the epsg parameter
- crs={num}: the EPSG code of the lon, lat or box values
- queryable={field1}[,{field2},...]}: the names of the feature fields that can be queried
- {field}__{query_op}={value}: specify a filter expression, field must be in the list of fields specified by queryable, supported query_op's are:

GeoSolutions S.A.S
Via Poggio Alle Viti 1187
55054 Massarosa (LU) Italy
Tel: +390584962313 Fax: +390584962313
http://www.geo-solutions.it
info@geo-solutions.it

  - eq: equal to
  - ne: not equal to
  - lt: lower than
  - lte: lower than or equal to
  - gt: greater than
  - gte: greater than or equal to
  - like
  - ilike

- mode. Can be ‘features’, ‘count’, ‘bounds’. In ‘features’ mode it just returns the features in json format, in ‘count’ mode it returns a count of the features satisfying the filters, in ‘bounds’ mode it returns the bounding box of the features satisfying the filter as a json array
- hints. A map providing implementation specific hints. The expected format is key1:value1;key2:value2;...

As a variant compared to the MapFish protocol the above can also be emitted as a POST request using the application/x-www-form-urlencoded mime type for the contents. This is a departure from restful principles that allows to send large queries (hints can be big) and keep the implementation simple.

The MapFish protocol uses POST to create new features, if we’re going to support writing in the future we’ll make sure the POST request to create/update data is going to use the “application/json” mime type instead.

GeoSolutions S.A.S
Via Poggio Alle Viti 1187
55054 Massarosa (LU) Italy
Tel: +390584962313 Fax: +390584962313
http://www.geo-solutions.it
info@geo-solutions.it

Mapping the protocol onto the GeoTools datastore API

The implementation should be based on the GeoRest store.

DataStore API	Protocol equivalent and notes
DataStore.getTypeNames()	/capabilities
DataStore.getSchema()	/describe/layername Would use both the caps and describe to build the result (crs comes from caps)
FeatureSource.getQueryCapabilities()	Should list the ability to perform sorting and offsets natively
FeatureSource.getFeatures(Query/Filter)	/data/layername?mode=features&... Should take the filter and split it into two parts, the one natively supported, and the one that is performed later in memory. Also, it should pass down the view params contained in the Hints.VIRTUAL_TABLE_PARAMETERS hint.
FeatureSource.getBounds(Query)	/data/layername?mode=bounds&... Should build a ReferencedEnvelope using the CRS attached to the FeatureType geometry. Should also pass down the hints as they might contain out of band extra filtering that cannot be encoded in the simple native filtering abilities

GeoSolutions S.A.S
Via Poggio Alle Viti 1187
55054 Massarosa (LU) Italy
Tel: +390584962313 Fax: +390584962313
http://www.geo-solutions.it
info@geo-solutions.it

FeatureSoruce.getCount(Query)	/data/layername?mode=count&... Should also pass down the hints

GeoSolutions S.A.S
Via Poggio Alle Viti 1187
55054 Massarosa (LU) Italy
Tel: +390584962313 Fax: +390584962313
http://www.geo-solutions.it
info@geo-solutions.it