

Geodata access and retrieval

Spatial Data Analysis and Simulation Modelling,
2020, Simon Scheider



Outline

- Interoperability
- Basic geodata formats
- Web service architectures and RESTful APIs
- Geosocial media: Foursquare and Open Street Map (OSM)
- Linked data principles
- Open Geographic Information Standards
 - Spatial Data on the Web best practices
 - ISO standards
 - Simple Features Model and WKT (Spatial Schema)
 - OGC service specifications
 - Example: WMS
 - Example: WFS

Interoperability

- Technical interoperability is the ability of different software systems to communicate and interact via shared interfaces
- Semantic interoperability is the ability to search, find and make use of data from different sources (providers, organizations, measurements)

Technical interoperability for data and software can be solved via

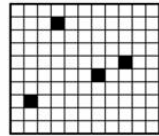

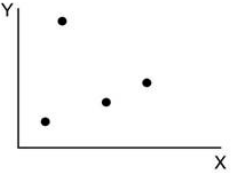
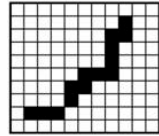

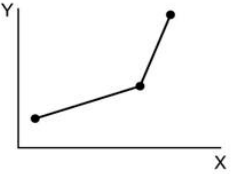
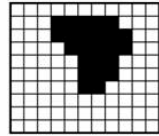

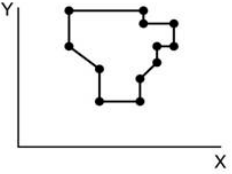
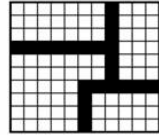
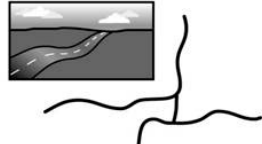
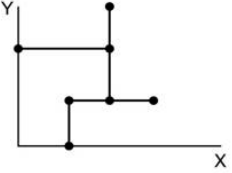
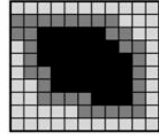

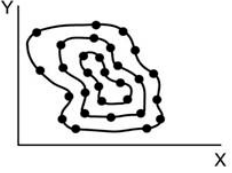
- **standardized data formats**
- **standardized Web services**
- **standardized interfaces (APIs)**

Basic geodata formats: raster and vector

- Raster encodes space into *georeferenced cells*
- Vector encodes space into *georeferenced points, lines and areas*

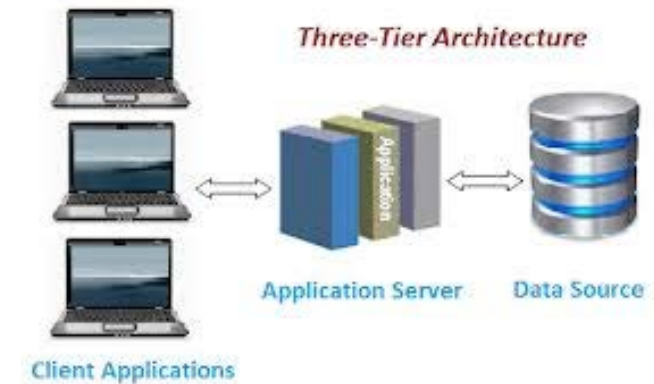
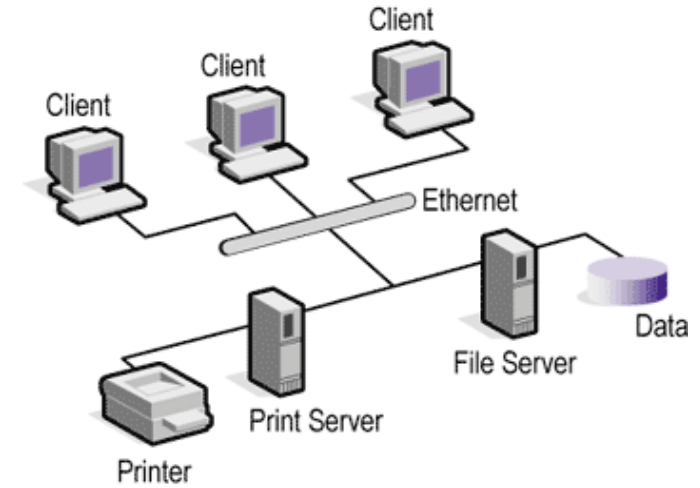
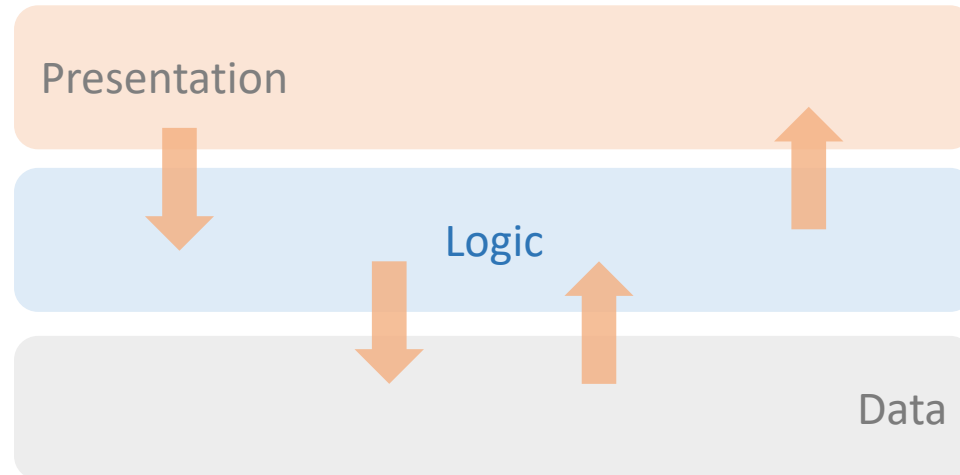
Important data formats (discussed later):

- Vector: shp (shapefile, ESRI), WKT (well known text), GML (Geography Markup language), GeoJSON
- Raster: GeoTIFF, netCDF, IMG, GRID (ESRI)

The raster view of the world	Happy Valley spatial entities	The vector view of the world
	 Points: hotels	
	 Lines: ski lifts	
	 Areas: forest	
	 Network: roads	
	 Surface: elevation	

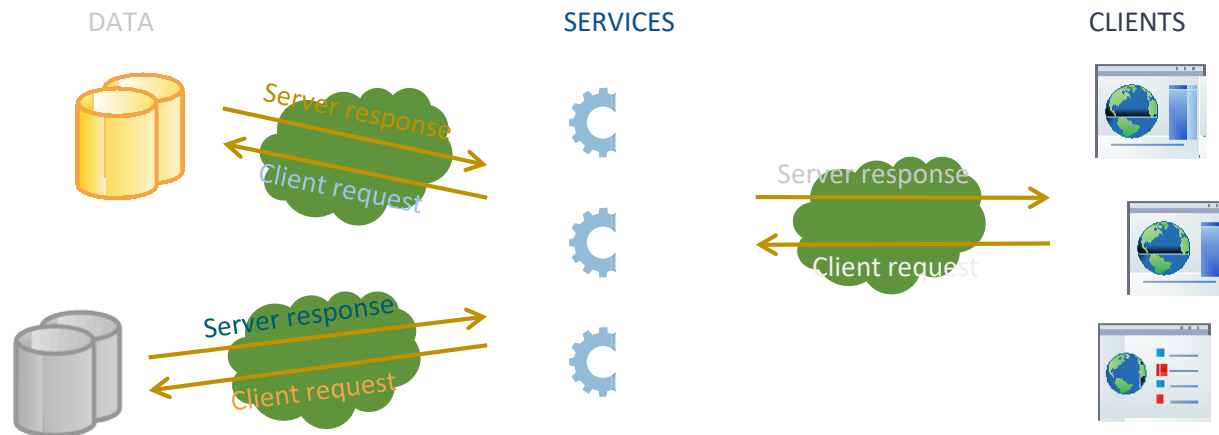
Typical software architectures

- Client-server
- 3-tier
 - Presentation: user interface
 - Application logic: calculations, decisions, transformations, etc
 - Data: storage and retrieval of data



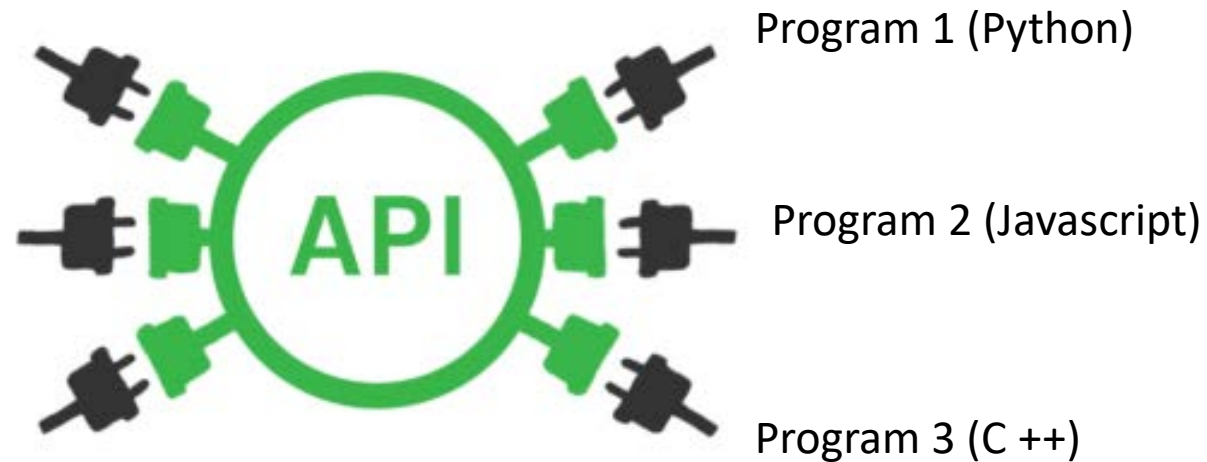
Distribute tiers on the Web

- Web applications
 - n-tier architecture: the presentation layer is a web page/site
 - many possible requests and server responses
 - many alternative implementations with varying requirements



What are Web API's?

- API “Application Programming Interface”
- Exchange of data between a Web service and **a program**
- Use of Web services from within **a program**



http requests

http requests
are fired
against an
endpoint
(computing
device
communicating
back and forth)

HTTP MODEL

Client



GET / HTTP/1.1



HTTP/1.1 200 OK



Server



POST /login HTTP/1.1



HTTP/1.1 401 Unauthorized



Client



HEADER

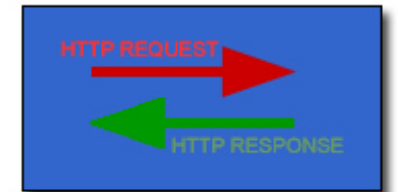
BODY



HTTP REQUEST



HTTP RESPONSE



http requests

```
GET /doc/test.html HTTP/1.1
```

```
Host: www.test101.com
```

```
Accept: image/gif, image/jpeg, */*
```

```
Accept-Language: en-us
```

```
Accept-Encoding: gzip, deflate
```

```
User-Agent: Mozilla/4.0
```

```
Content-Length: 35
```

```
bookId=12345&author=Tan+Ah+Teck
```

Request Line

Request Headers

Request
Message
Header

A blank line separates header & body

Request Message Body

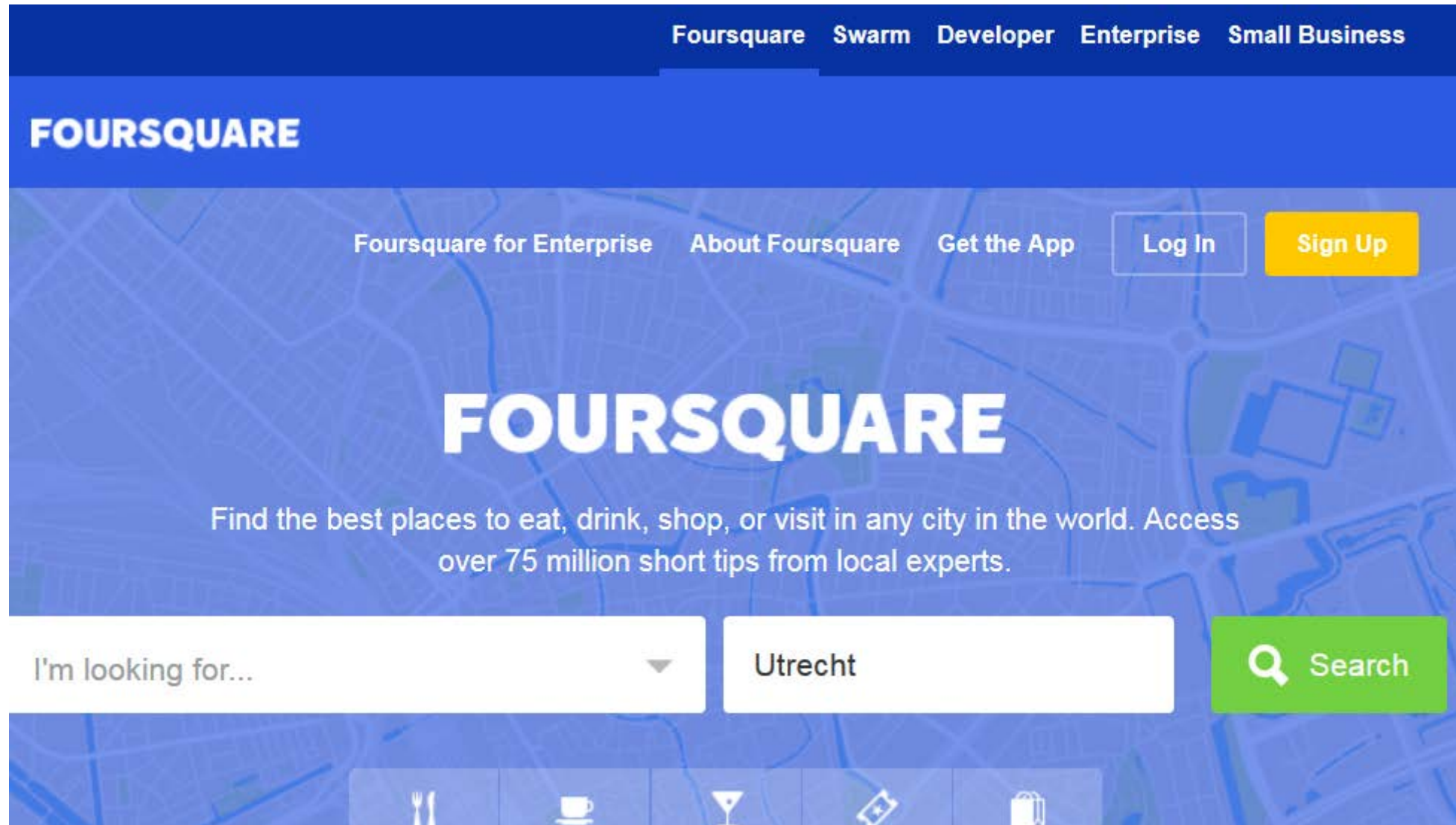
Restful services

RESTful (“Representational state transfer”):

- Uniform interface:
 - http GET request to receive data
 - http POST request to push data
- Stateless: No client state being stored on the server between requests. Each request from any client contains all the information necessary to service the request
- Individual resources are identified using [URIs](#) in Web-based REST systems

Makes services easy to access via *standard clients* (URIs and http) as opposed to the old SOAP (Simple Object Access Protocol) protocol

Geosocial media: Foursquare



Foursquare API

- EXPLORE endpoint

```
import json, requests
url = 'https://api.foursquare.com/v2/venues/explore'

params = dict(
    client_id='CLIENT_ID',
    client_secret='CLIENT_SECRET',
    v='20180323',
    ll='40.7243,-74.0018',
    query='coffee',
    limit=1
)

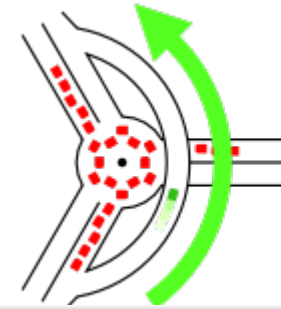
resp = requests.get(url=url, params=params)
data = json.loads(resp.text)
```

```
"venue": {
  "id": "49b6e8d2f964a52016531fe3",
  "name": "Russ & Daughters",
  "location": {
    "address": "179 E Houston St",
    "crossStreet": "btwn Allen & Orchard St",
    "lat": 40.72286707707289,
    "lng": -73.98829148466851,
    "labeledLatLngs": [
      {
        "label": "display",
        "lat": 40.72286707707289,
        "lng": -73.98829148466851
      }
    ],
    "distance": 130,
    "postalCode": "10002",
    "cc": "US",
    "city": "New York",
    "state": "NY",
    "country": "United States",
    "formattedAddress": [
      "179 E Houston St (btwn Allen & Orchard St)",
      "New York, NY 10002",
      "United States"
    ]
  },
  "categories": [
    {
      "id": "4bf58dd8d48988d1f5941735",
      "name": "Gourmet Shop",
```

Problems: Service quality of social media APIs

- Rate limits: the amount of data you can get is usually limited
 - Read the social media platform for details
 - Low quantities are usually free, large quantities can still be obtained with fees
 - Frequency and amount of queries are restricted (denial of service)
- API changes: Companies are free to change syntax or stop services
 - Your code may need to be adapted once service is changed
- Legal: Restrictions in data usage for certain purposes
- Data quality: social media data is far from an unbiased sample

Open Street Map




Overpass
API

Run Share Export Wizard Save Load Settings Help overpass turbo Map Data

```
1  /*
2  This is an example Overpass query.
3  Try it out by pressing the Run button above!
4  You can find more examples with the Load tool.
5  */
6  node
7    [amenity=drinking_water]
8    ({{bbox}});
9  out;
```

<https://overpass-turbo.eu/>

The screenshot shows the Overpass Turbo web application. On the left is a code editor with a sample Overpass query. The main area displays a map of Rome, centered on the Colosseum. The map includes various street names, landmarks like the Colosseum and the Arco di Costantino, and a search bar at the top. The interface also features a toolbar with navigation and editing tools.

Linked open data (LOD) principles:

Link data on the Web, not (only) documents

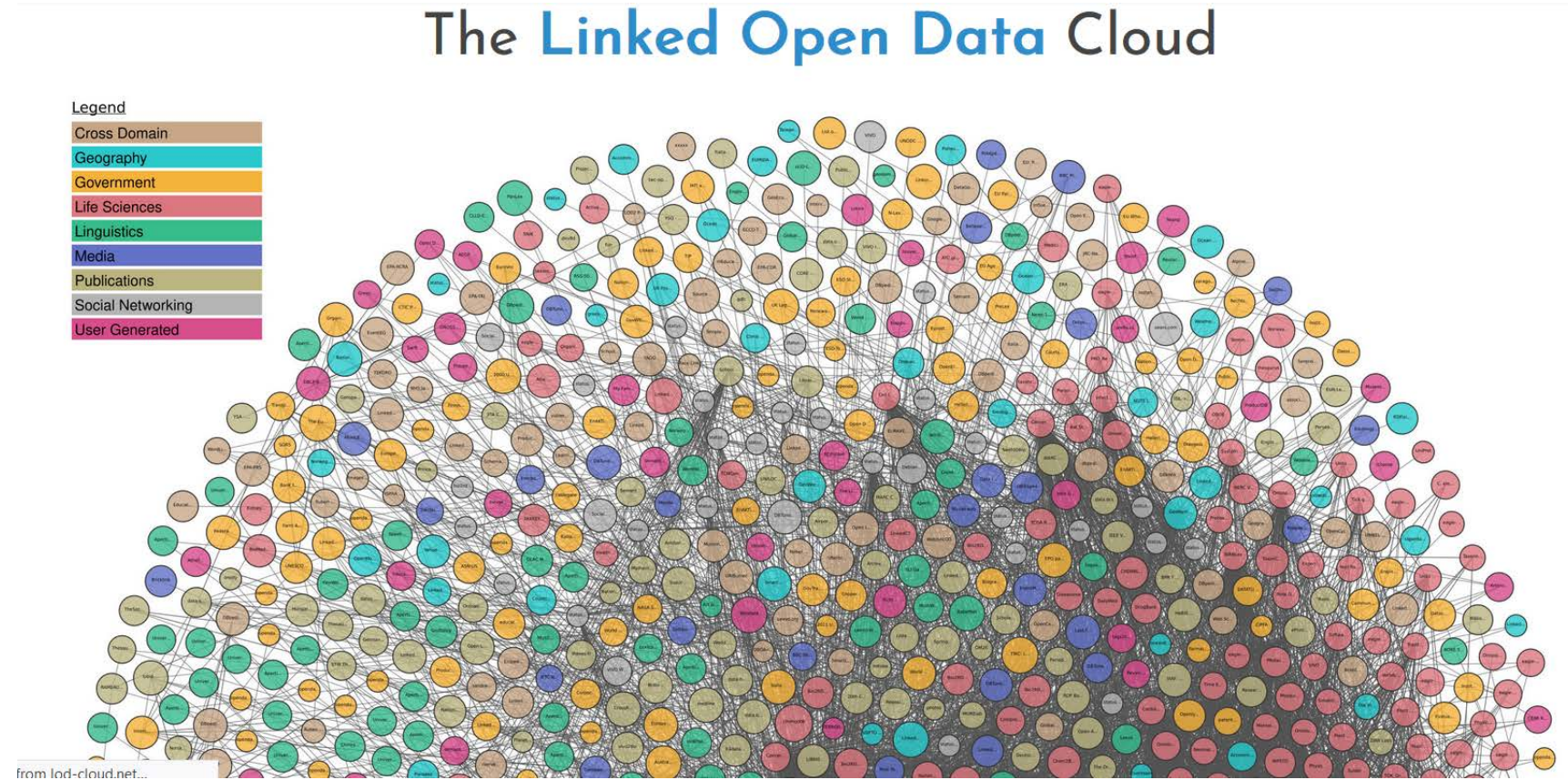
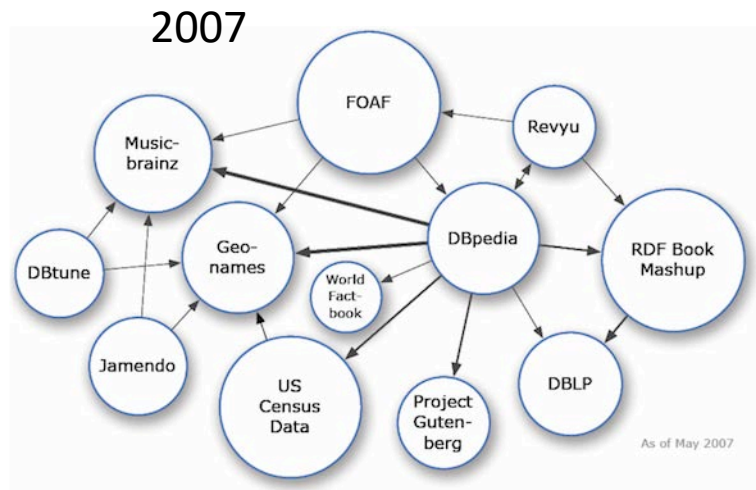
- Use [URIs to denote things.](#)
- Use [HTTP URIs so that these things can be referred to and looked up \("dereferenced"\) by people and user agents.](#)
- Provide useful information about the thing when its URI is dereferenced, leveraging standards such as [RDF, SPARQL.](#)
- Include [links](#) to other related things (using their URIs) when publishing data on the Web.

(Tim Berners Lee, 2008)



The Linked Open Data Cloud

<https://lod-cloud.net/>



Open Geographic Information standards

Open standards are a result of consensus delivered by international organizations

For geographic information

- ISO standards
- Open Geospatial Consortium (OGC) specifications

For Web standards:

- W3C (Html, RDF, XML, HTTP,....)

OGC/W3C Spatial Data on the Web

Best practices

<https://www.w3.org/TR/sdw-bp/>

Best Practices Summary

[Best Practice 1](#): Use globally unique persistent HTTP URIs for Spatial Things

[Best Practice 2](#): Make your spatial data indexable by search engines

[Best Practice 3](#): Link resources together to create the Web of data

[Best Practice 4](#): Use spatial data encodings that match your target audience

[Best Practice 5](#): Provide geometries on the Web in a usable way

[Best Practice 6](#): Provide geometries at the right level of accuracy, precision, and size

[Best Practice 7](#): Choose coordinate reference systems to suit your user's applications

[Best Practice 8](#): State how coordinate values are encoded

[Best Practice 9](#): Describe relative positioning

[Best Practice 10](#): Use appropriate relation types to link Spatial Things

[Best Practice 11](#): Provide information on the changing nature of spatial things

[Best Practice 12](#): Expose spatial data through 'convenience APIs'

[Best Practice 13](#): Include spatial metadata in dataset metadata

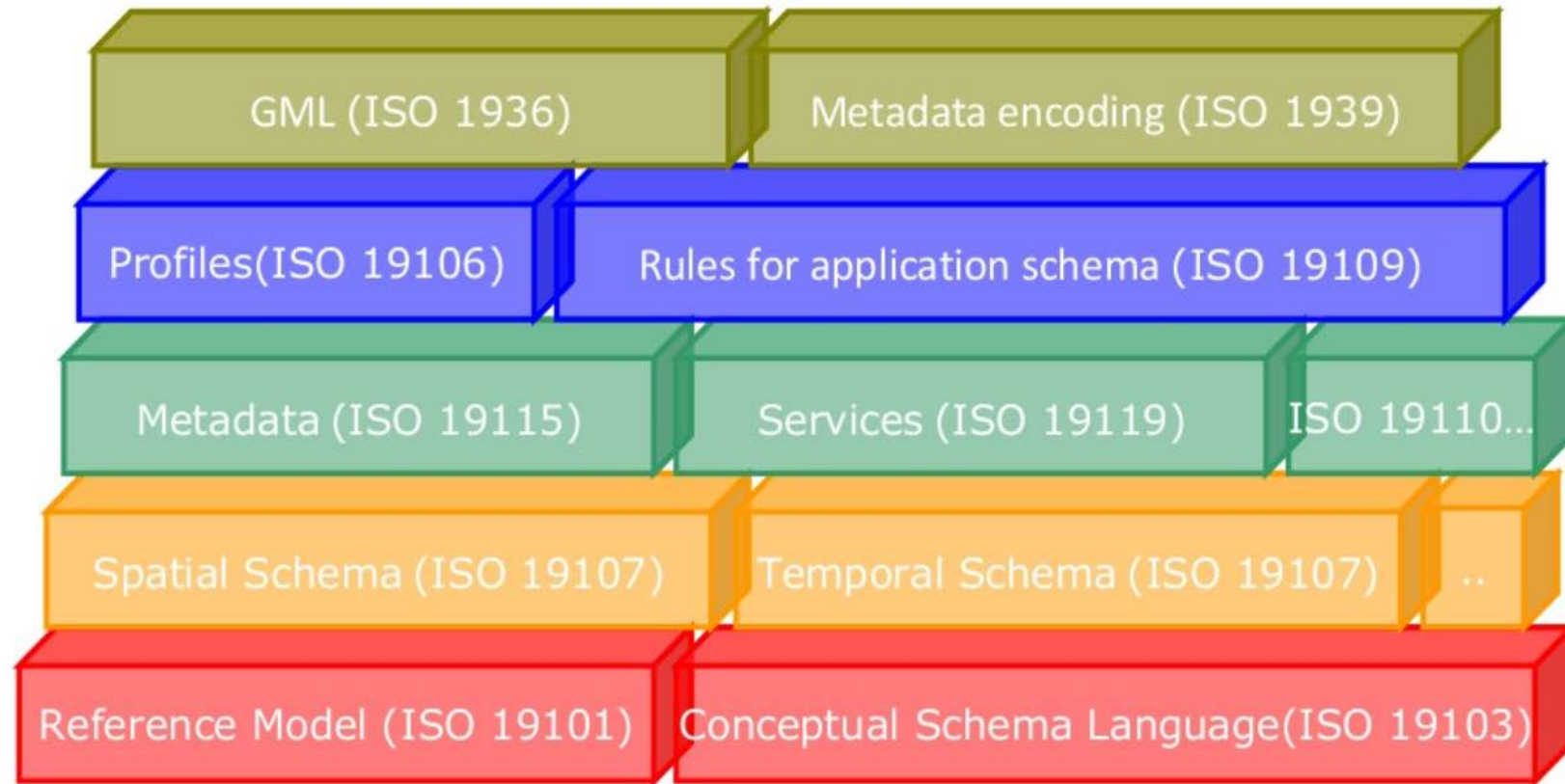
[Best Practice 14](#): Describe the positional accuracy of spatial data

Note:

Currently, most geodata services do *not yet* adhere to such LOD/RESTful standards

GeoServices are often still based on old SOAP standards

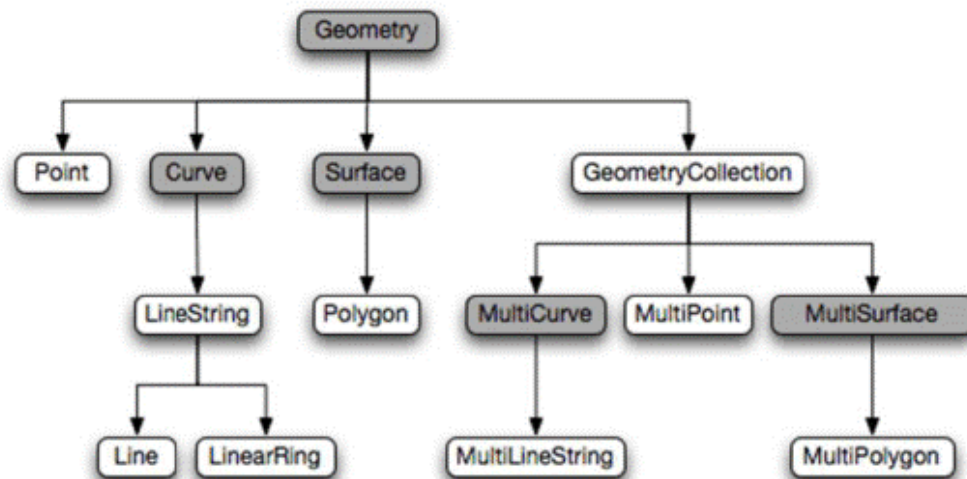
ISO standards for geographic information



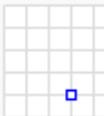
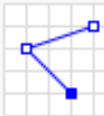
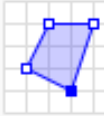
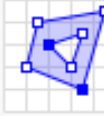
Simple Features Model

OGC Simple Features (ISO19125)
is encoded either as GML or WKT
(Well known text)

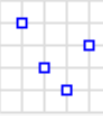
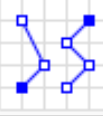
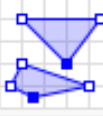
Datentypen:



Geometry primitives (2D)

Type	Examples	
Point	POINT (30 10)	
LineString	LINESTRING (30 10, 10 30, 40 40)	
Polygon	POLYGON ((30 10, 10 20, 20 40, 40 40, 30 10))	
	POLYGON ((35 10, 10 20, 15 40, 45 45, 35 10), (20 30, 35 35, 30 20, 20 30))	

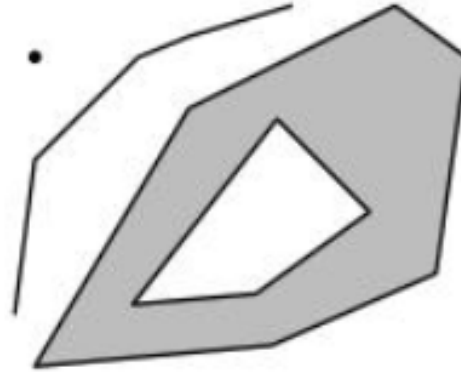
Multipart geometries (2D)

Type	Examples	
MultiPoint	MULTIPOINT ((10 40), (40 30), (20 20), (30 10))	
	MULTIPOINT (10 40, 40 30, 20 20, 30 10)	
MultiLineString	MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))	
MultiPolygon	MULTIPOLYGON (((30 20, 10 40, 45 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))	
	MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)), ((20 35, 45 20, 30 5, 10 10, 10 30, 20 35), (30 20, 20 25, 20 15, 30 20)))	

Well known text (WKT)

- WKT is an OGC and ISO standard for representing **geometries**, **coordinate reference systems**, and **transformations** between coordinate reference systems.
- WKT is specified in **OpenGIS Simple Feature Access - Part 1: Common Architecture** standard which is the same as the **ISO 19125-1** standard. Download from http://portal.opengeospatial.org/files/?artifact_id=25355.
- This standard concentrates on **simple features**: features with all spatial attributes described piecewise by a straight line or a planar interpolation between sets of points.

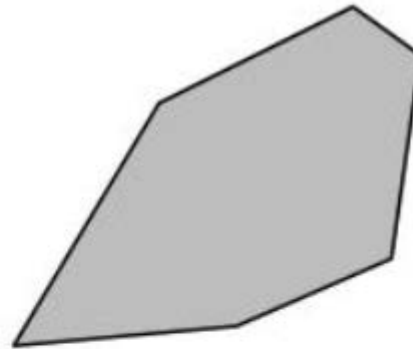
WKT example



WKT representation:

```
GeometryCollection(  
  Point(5 35),  
  LineString(3 10,5 25,15 35,20 37,30 40),  
  Polygon((5 5,28 7,44 14,47 35,40 40,20 30,5 5),  
          (28 29,14.5 11,26.5 12,37.5 20,28 29))  
)
```

GML example



GML representation:

```
<gml:Polygon gml:id="p3" srsName="urn:ogc:def:crs:EPSG:6.6:4326">
  <gml:exterior>
    <gml:LinearRing>
      <gml:coordinates>
        5,5 28,7 44,14 47,35 40,40 20,30 5,5
      </gml:coordinates>
    </gml:LinearRing>
  </gml:exterior>
</gml:Polygon>
```

GeoJSON

The [coordinate reference system](#) for all GeoJSON [[RFC7946](#)] coordinates is a geographic coordinate reference system, using the World Geodetic System 1984 (WGS 84) [WGS84] [datum](#), with [longitude](#) and [latitude](#) units of decimal degrees.

```
HTTP/1.1 200 OK
Date: Sun, 05 Mar 2017 17:12:35 GMT
Content-length: 543
Connection: close
Content-type: application/geo+json
```

```
{
  "type": "Feature",
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [ [4.884235, 52.375108], [4.884276, 52.375153],
        [4.884257, 52.375159], [4.883981, 52.375254],
        [4.883850, 52.375109], [4.883819, 52.375075],
        [4.884104, 52.374979], [4.884143, 52.374965],
        [4.884207, 52.375035], [4.884263, 52.375016],
        [4.884320, 52.374996], [4.884255, 52.374926],
        [4.884329, 52.374901], [4.884451, 52.375034],
        [4.884235, 52.375108] ]
    ]
  },
  "properties": {
    "name": "Anne Frank's House"
  }
}
```


Geodata on the Web (formats)

	WKT	GML	KML	GeoJSON	HTML
Based on	WKT	XML	XML	JSON	HTML
Media type	text/plain	application/gml+xml	application/vnd.google-earth.kml+xml , application/vnd.google-earth.kmz	application/geo+json	text/html
Usage	Representation of 0D-2D geometries, CRS and CRS transformation	Representation of Spatial Things and 0D-3D geometries. Comprehensive and supporting many use cases.	Representation of Spatial Things and 0D-3D geometries. Main focus on spatial data visualization and interaction	Representation of Spatial Things and 0D-2D geometries	Description of Spatial Things and geometries can be embedded by using mechanisms as [HTML-RDfA] , [MICRODATA] , [JSON-LD] , using vocabularies as [SCHEMA-ORG]
Tool support	Widely supported in GIS tools Supported by some Web libraries, usually converted in GeoJSON [RFC7946] Supported by most triple stores	Widely supported in GIS tools Supported by some Web libraries, usually converted in GeoJSON [RFC7946] , but not when the geometry is 3-dimensional (volumes) Supported only by triple stores supporting [GeoSPARQL]	Mainly supported by Earth browsers, as Google Earth	Supported in some GIS tools Widely supported in Web libraries and mapping APIs	Optimal for Web publication and discovery

OGC service specifications

- *Web Map Service (WMS)*: interface to display online maps
- *Web Map Tile Service (WMTS)*: interface for fast display of map tiles
- *Web Feature Service (WFS)*: interface for retrieval of vector data (Simple Features Model)
- *Web Coverage Service (WCS)*: interface for retrieval of raster data
- *Catalogue Interface*: interface for searching geographic meta-data
- *Coordinate transformation service*: Transform coordinates

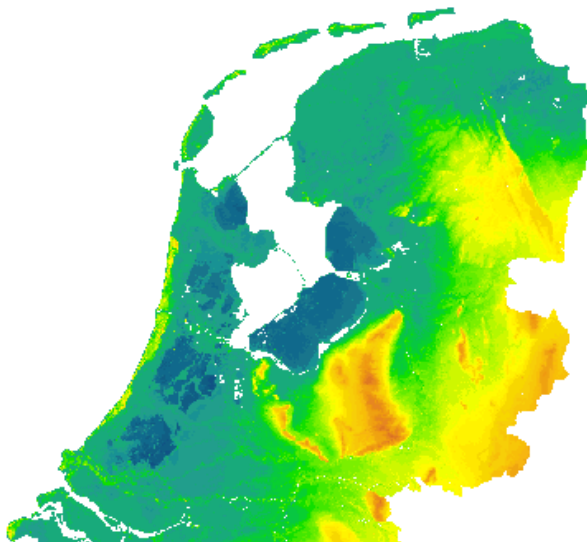
WMS service requests

- *GetCapabilities*: Returns a description of the available map layers
- *GetMap*: Returns a static image of a map at the specified extent and zoom level
- *GetFeatureInfo*: Retrieves information about features at some specified coordinate point

WMS example: AHN2 (Actueel Hoogtebestand Nederland)

Get capabilities request

```
http://geodata.nationaalgeoregister.nl/ahn2/wms?  
service=WMS&  
request=GetCapabilities
```

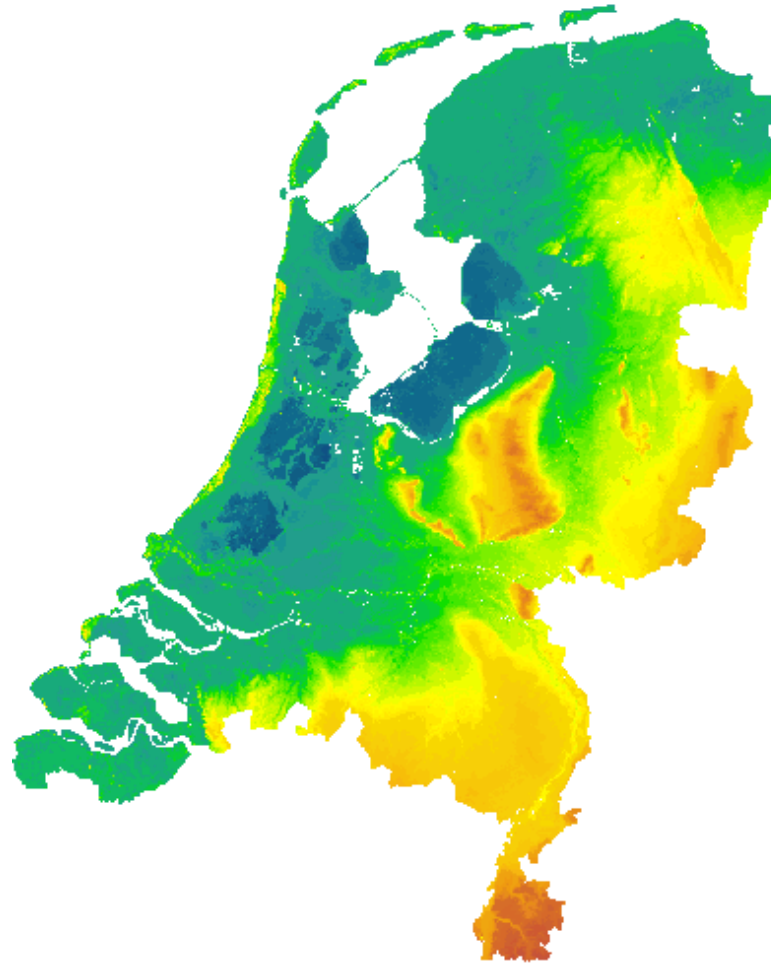


```
<WMS_Capabilities xmlns="http://www.opengis.net/wms" xmlns:xlink="http://www.w3.org/1999  
  
<Service>...</Service>  
<Capability>  
  <Request>  
    <GetCapabilities>...</GetCapabilities>  
    <GetMap>...</GetMap>  
    <GetFeatureInfo>...</GetFeatureInfo>  
  </Request>  
  <Exception>...</Exception>  
  <Layer>  
    <Title>Actueel Hoogtebestand Nederland 2</Title>  
    <Abstract>Actueel Hoogtebestand Nederland 2</Abstract>  
    ...  
    <CRS>EPSG:28992</CRS>  
    <CRS>EPSG:3857</CRS>  
    <CRS>EPSG:4326</CRS>  
    ...  
    <EX_GeographicBoundingBox>...</EX_GeographicBoundingBox>  
    <BoundingBox CRS="EPSG:4326" minx="50.72814376700224" miny="3.2012587672031283" ...  
    ...  
    <Layer queryable="1" opaque="0">...</Layer>  
    <Layer queryable="1" opaque="0">  
      <Name>ahn2_5m</Name>  
      <Title>ahn2_5m</Title>  
      <Abstract/>  
      <KeywordList>...</KeywordList>  
      <CRS>EPSG:28992</CRS>  
      <CRS>CRS:84</CRS>  
      <EX_GeographicBoundingBox>...</EX_GeographicBoundingBox>  
      <BoundingBox CRS="CRS:84" minx="3.2012587672391843" miny="50.72814376700224" ...  
      <BoundingBox CRS="EPSG:28992" minx="10000.0" miny="306250.0" maxx="280000.0" ...  
      ...  
      <MetadataURL type="TC211">...</MetadataURL>  
      <Style>...</Style>  
    </Layer>  
  <Layer queryable="1" opaque="0">...</Layer>
```

WMS example: AHN2 (Actueel Hoogtebestand Nederland)

GetMap request

```
http://geodata.nationaalgeoregister.nl/ahn2/wms?  
service=WMS&  
request=GetMap&  
layers=ahn2_5m&  
bbox=13014,306243,286599,623492&  
width=400&  
height=500&  
format=image/png&  
srs=EPSG:28992
```



WFS service requests

- *GetCapabilities*: Returns a description of the service including the available Feature Types
- *DescribeFeatureType*: Returns a description of one or more feature types
- *GetFeature*: Retrieves features with specified attributes of some specified feature type for some specified spatial extent . Output format can also be specified (GML, SHP, JSON, KML, CSV)

WFS example: requesting BAG (Basisregistratie adressen en gebouwen)

Get capabilities request

```
http://geodata.nationaalgeoregister.nl/bag/wfs?  
service=WFS&  
request=GetCapabilities
```



```
<wfs:WFS_Capabilities version="2.0.0" xsi:schemaLocation="http://www.opengis.net/wfs/2.0 http://geodata.nat  
<ows:ServiceIdentification>...</ows:ServiceIdentification>  
<ows:ServiceProvider>...</ows:ServiceProvider>  
<ows:OperationsMetadata>...</ows:OperationsMetadata>  
<FeatureTypeList>  
  <FeatureType></FeatureType>  
  <FeatureType>  
    <Name>bag:pand</Name>  
    <Title>pand</Title>  
    <Abstract>pand</Abstract>  
    <ows:Keywords>  
      <ows:Keyword>pand</ows:Keyword>  
      <ows:Keyword>features</ows:Keyword>  
    </ows:Keywords>  
    <DefaultCRS>urn:ogc:def:crs:EPSG::28992</DefaultCRS>  
    <ows:WGS84BoundingBox>  
      <ows:LowerCorner>3.2800546964714012 50.748745396375774</ows:LowerCorner>  
      <ows:UpperCorner>7.224161199744223 53.48515806526503</ows:UpperCorner>  
    </ows:WGS84BoundingBox>  
    <MetadataURL xlink:href="http://www.nationaalgeoregister.nl/geonetwork/srv/dut/xml.metadata.get  
  </FeatureType>  
  <FeatureType>...</FeatureType>  
  <FeatureType>...</FeatureType>  
  <FeatureType>...</FeatureType>  
</FeatureTypeList>  
<fes:Filter_Capabilities>...</fes:Filter_Capabilities>  
</wfs:WFS_Capabilities>
```

WFS example: requesting BAG (Basisregistratie adressen en gebouwen)

Get feature request

```
http://geodata.nationaalgeoregister.nl/bag/wfs?  
service=WFS&  
request=GetFeature&  
typeName=bag:pand&  
count=100&  
startIndex=0&  
outputFormat=json
```



```
{  
  "type": "FeatureCollection",  
  "features": [  
    {  
      "type": "Feature",  
      "id": "pand.7871844",  
      "geometry_name": "geometrie",  
      "properties": {  
        "identificatie": 856100000350209,  
        "bouwjaar": 1941,  
        "status": "Pand in gebruik",  
        "gebruiksdoel": "woonfunctie",  
        "oppervlakte_min": 147,  
        "oppervlakte_max": 147,  
        "aantal_verblijfsobjecten": 1,  
        "actualiteitsdatum": null  
      },  
      "geometry": {  
        "type": "Polygon",  
        "coordinates": [  
          [  
            179753.466,  
            405278.319  
          ],  
          ]  
        }  
      }  
    ]  
  }  
}
```


WFS example

<https://geodata.nationaalgeoregister.nl/wijkenbuurten2016/wfs?request=GetCapabilities>

Statistics Netherlands (CBS) publishes their [Neighborhoods statistics data](#) as a [\[WFS\]](#) service. The capabilities of that service can be requested in the following way:

```
https://geodata.nationaalgeoregister.nl/wijkenbuurten2016/wfs?request=GetCapabilities
```

For example, the following request returns the data for neighborhoods within the specified bounding box. The bounding box is specified using [EPSG:28992](#) ("Amersfoort / RD New") and indicates an area of 100 square meters.

```
https://geodata.nationaalgeoregister.nl/wijkenbuurten2016/wfs?request=GetFeature&  
typename=wijkenbuurten2016:cbs_wijken_2016&version=2.0.0&service=WFS&  
bbox=120000,480000,130000,490000
```

Questions?
(Q&A session)

References

- Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, 205-227.
- <https://www.w3.org/TR/sdw-bp/>
- <https://www.ogc.org/standards/wfs>
- <https://www.ogc.org/standards/wms>
- <https://www.ogc.org/standards/wmts>
- <https://www.ogc.org/standards/sfa>