SPARQL Micro-Services

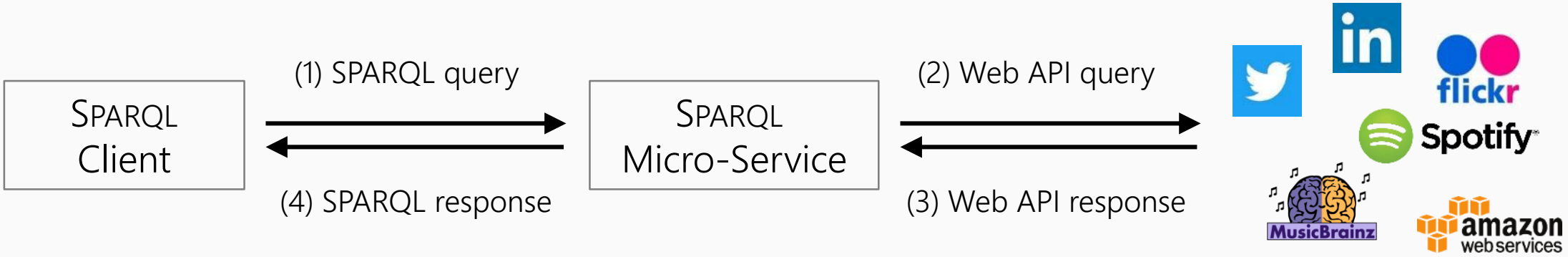
hands-on

Franck MICHEL

Main concepts

The SPARQL Micro-Service Architecture

Lightweight method to **query a Web API with SPARQL**, and assign dereferenceable URIs to Web API resources



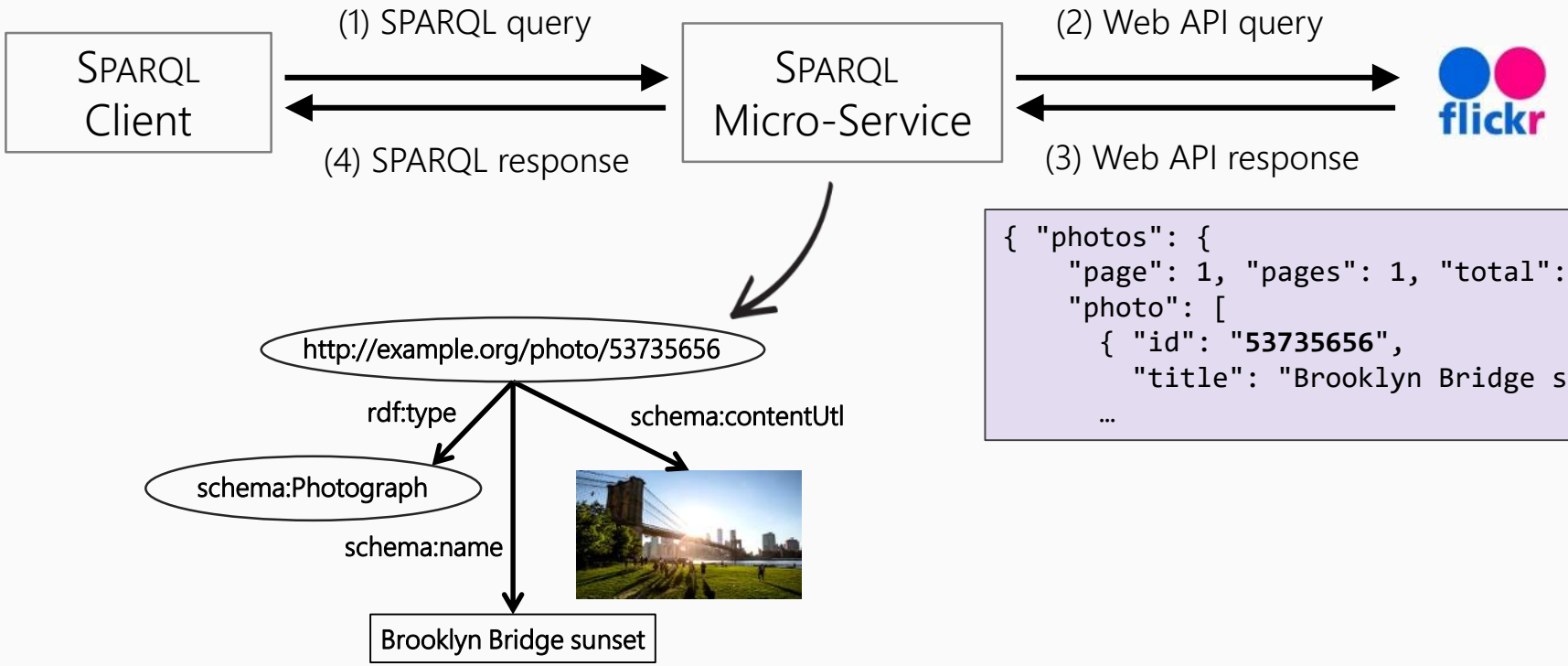
https://github.com/frmichel/sparql-micro-service

Details - Arguments passed **as HTTP parameters**

```
SELECT * WHERE {
  ?photo a schema:Photograph;
    schema:name       ?title;
    schema:contentUrl ?img.
}
```

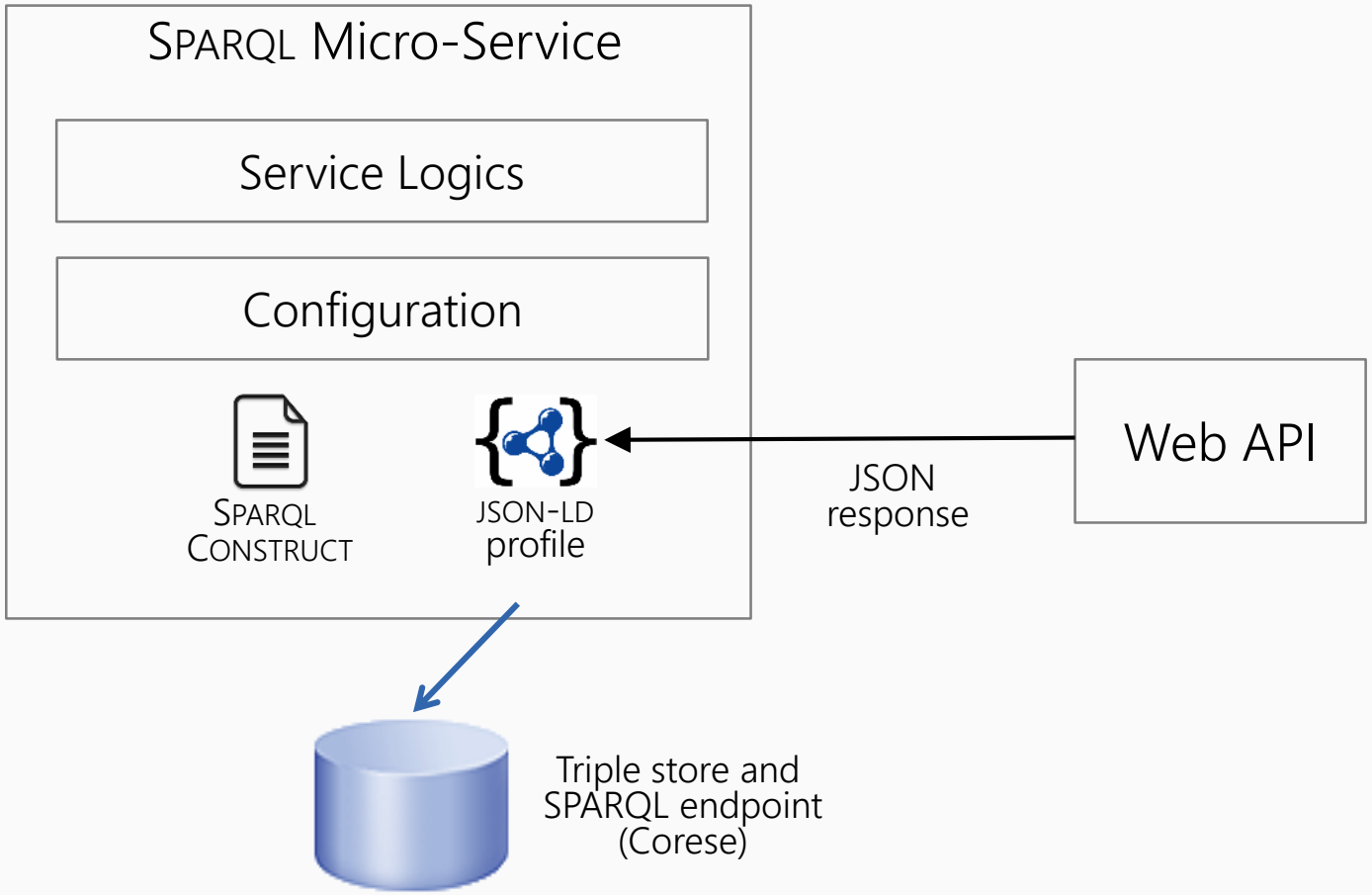
Endpoint: http://example.org/flickr/getPhotosByTag?tag=bridge

https://api.flickr.com/services/rest/?
method=flickr.photos.search&
format=json&per_page=100&tags=bridge&...




```
{ "photos": {
    "page": 1, "pages": 1, "total": "189", …
    "photo": [
      { "id": "53735656",
        "title": "Brooklyn Bridge sunset", … },
      …
```

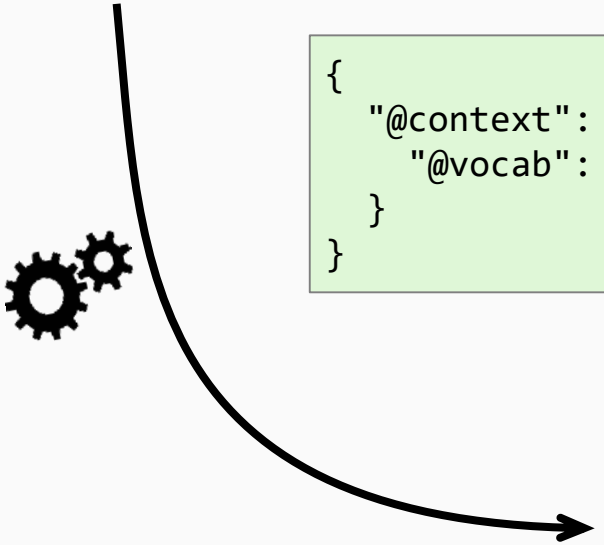
Translating the Web API response to RDF



First translation with a JSON-LD context

API response

```
{
  "id": "53735656",
  "title": "Brooklyn Bridge sunset",
}
```



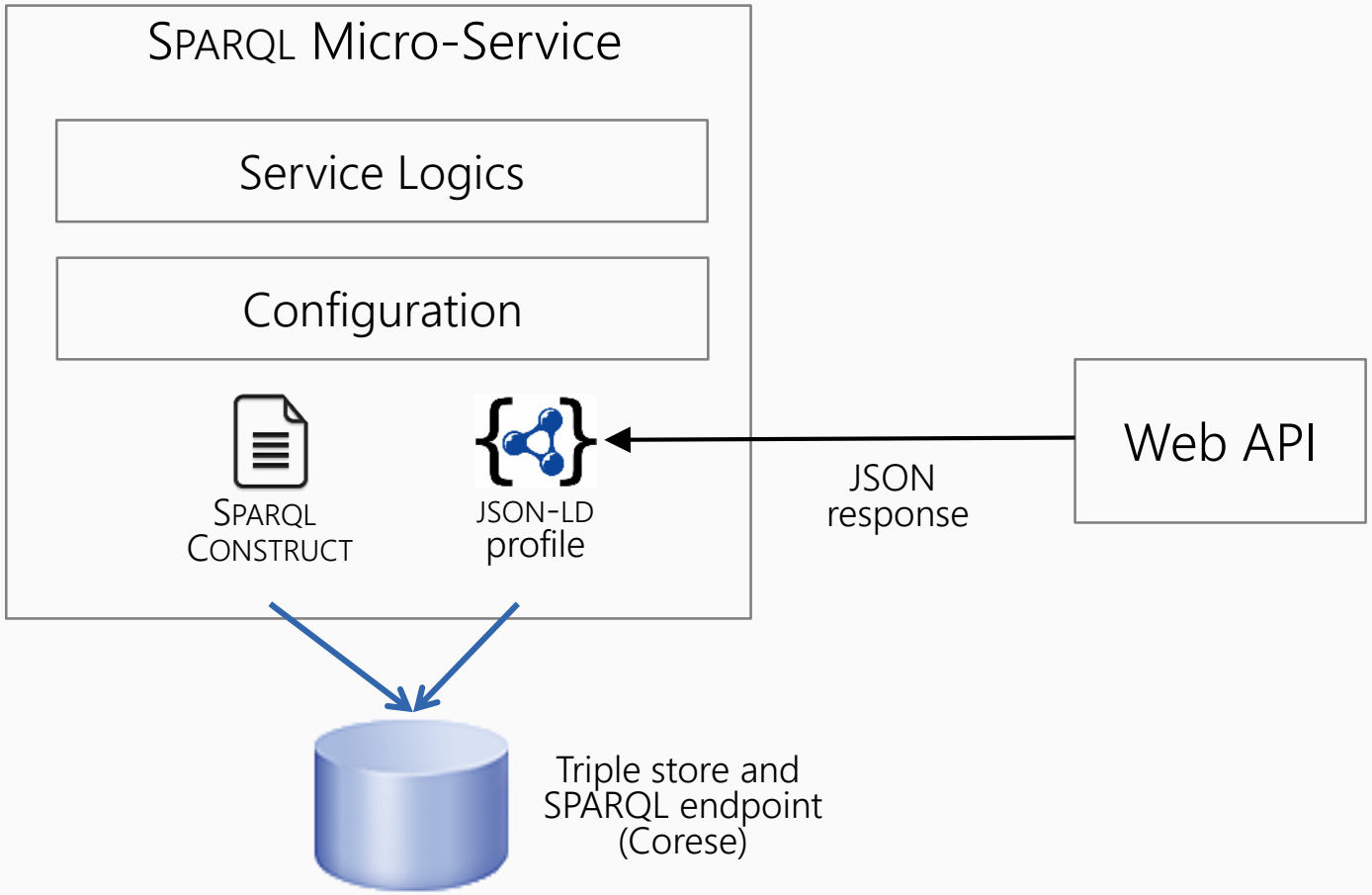
profile.jsonld

```
{
  "@context": {
    "@vocab": "http://ns.inria.fr/sparql-micro-service/api#"
  }
}
```

```
@prefix api: <http://ns.inria.fr/sparql-micro-service/api#>

[] api:id      "53735656";
   api:title   "Brooklyn Bridge sunset";
```

Translating the Web API response to RDF



Advanced mapping with SPARQL (optional)

```
@prefix api: <http://ns.inria.fr/sparql-micro-service/api#>

[] api:id      "53735656";
   api:title   "Brooklyn Bridge sunset";
```

construct.sparql

```
PREFIX schema: <http://schema.org/>
CONSTRUCT {
  ?photoUri
   a            schema:Photograph;
   schema:name  ?title;
}
WHERE {
  ?result
    api:id      ?photoId;
    api:title   ?title;

  BIND(IRI(concat("http://example.org/ld/photo/", ?photoId)) AS ?photoUri)
}
```

```
@prefix schema: <http://schema.org/>

<http://example.org/ld/photo/53735656>
    a           schema:Photograph;
    schema:name "Brooklyn Bridge sunset".
```

Quick-start guide

Main approach to create a SPARQL µ-service

Read the API documentation:
find out the API service that does what you want to do

Create a basic SPARQL micro-service:
a JSON-LD profile translates the API response into “raw” RDF (api namespace)

Decide of a mapping to an RDF vocabulary:
figure out an appropriate vocabulary for your use case

Write the mapping in a CONSTRUCT query
and test the service.

My first SPARQL micro-service

Goal: find music albums on Deezer, that match a keyword

- Find out the right API query
- Create a basic SPARQL micro-service
- Decide of a mapping to an RDF vocabulary
- Write the mapping in a CONSTRUCT query

Check the Web API documentation

https://developers.deezer.com/api/search#connections

Choose a service to be fulfilled by the SPARQL micro-service

Query music albums by keyword (search > Search Methods)

Find out the right query to do this

https://api.deezer.com/search/album?q=eminem

My first SPARQL micro-service

Documentation:
https://s.42l.fr/smsdoc

- Find out the right API query
- **Create a basic SPARQL micro-service**
- Decide of a mapping to an RDF vocabulary
- Write the mapping in a CONSTRUCT query

1. CD to the directory of deployed services.

2. Create directory `deezer/findAlbums`

3. In `deezer/findAlbums`, create file `config.ini` with 2 properties:

```
api_query = "https://api.deezer.com/search/album?q={keyword}"
custom_parameter[] = keyword
```

Create file profile.jsonld

```
{ "@context": {
    "@vocab": "http://ns.inria.fr/sparql-micro-service/api#"
}}
```

Query the SPARQL micro-service (using Yasgui):

Endpoint URL:

```
http://localhost/service/deezer/findAlbums?keyword=eminem
```

Query:

```
select * where { ?s ?p ?o. }
```

My first SPARQL micro-service

- Find out the right API query
- Create a basic SPARQL micro-service
- **Decide of a mapping to an RDF vocabulary**
- Write the mapping in a CONSTRUCT query

Find appropriate vocabularies

- Schema.org https://schema.org/docs/full.html
- Wikidata https://wikidata.org
- LOV (Linked Open Vocabularies) https://lov.linkeddata.es/dataset/lov/
- Specialized ontology portals…

Schema.org

Thing > CreativeWork > MusicPlaylist > MusicAlbum

Thing > Organization > PerformingGroup > MusicGroup

Thing > CreativeWork > MusicRecording

…

My first SPARQL micro-service

- Find out the right API query
- Create a basic SPARQL micro-service
- Decide of a mapping to an RDF vocabulary
- **Write the mapping in a CONSTRUCT query**

```
PREFIX schema: <http://schema.org/>
CONSTRUCT {
   []
     a                  schema:MusicAlbum;
     schema:name        ?albumTitle;
     schema:image       ?imageUri;
     schema:byArtist    ?artistName
}
WHERE {
  ?album
     api:title   ?albumTitle;
     api:cover   ?image;
     api:artist [ api:name ?artistName ].

     bind(iri(?image) as ?imageUri)
}
```

Query the SPARQL micro-service with Yasgui

Query ✕ SµS ✕ +

http://localhost/service/deezer/findAlbums?keyword=orelsan

```
PREFIX schema: <http://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX api: <http://ns.inria.fr/sparql-micro-service/api#>
SELECT *  WHERE {
  ?s ?p ?o.
}
```

Table | Response | 100 results in 0.129 seconds | Simple view ☐ | Ellipse ☑ | b3788 | Page size: 50

	s	p	o
1	b3788	schema:byArtist	"Orelsan"^^<http://www.w3.org/2001/XMLSchema#string>
26	b3788	schema:image	<https://api.deezer.com/album/270762122/image>
51	b3788	schema:name	"Civilisation"^^<http://www.w3.org/2001/XMLSchema#string>
76	b3788	rdf:type	schema:MusicAlbum

Showing 1 to 4 of 4 entries (filtered from 100 total entries) < 1 >

Query the SPARQL micro-service with Yasgui

Query SµS +

http://localhost/service/deezer/findAlbums?keyword=orelsan

```
PREFIX schema: <http://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/0
PREFIX rdfs: <http://www.w3.org/2000/
PREFIX api: <http://ns.inria.fr/sparc
SELECT *  WHERE {
  ?s ?p ?o.
}
```

Table Response 100 results in 0.129 seco

	s	p	o
1	b3788	schema:byArtist	"Orelsan"^^
26	b3788	schema:image	<https://api.
51	b3788	schema:name	"Civilisatio
76	b3788	rdf:type	schema:Mu

Showing 1 to 4 of 4 entries (filtered from 100 total entries

Query SµS +

http://localhost/service/deezer/findAlbums?keyword=orelsan

```
PREFIX schema: <http://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX api: <http://ns.inria.fr/sparql-micro-service/api#>
SELECT *  WHERE {
  ?s a schema:MusicAlbum;
    schema:name ?name.
}
```

Table Response 25 results in 0.132 seconds Simple view ☐ Ellipse ☑ Filter query results Page size: 50

	s	name
1	b3687	"N'importe comment (The Remixes)"^^<http://www.w3.org/2001/XMLSchema#string>
2	b3688	"Potentiel (feat. Orelsan)"^^<http://www.w3.org/2001/XMLSchema#string>
3	b3689	"Enfant lune"^^<http://www.w3.org/2001/XMLSchema#string>
4	b3690	"Millions"^^<http://www.w3.org/2001/XMLSchema#string>
5	b3691	"Tous les jours dimanche"^^<http://www.w3.org/2001/XMLSchema#string>
6	b3692	"Double Vie - Single"^^<http://www.w3.org/2001/XMLSchema#string>

Query the SPARQL micro-service with Yasgui

Query ✕ SµS ✕ +

http://localhost/service/deezer/findAlbums?keyword=orelsan

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX api: <http://ns.inria.fr/sparql-micro-service/api#>
CONSTRUCT   WHERE {
  ?s ?p ?o.
}
```

Table | Response | 100 results in 0.122 seconds

```
@prefix ns1: <http://schema.org/> .
@prefix api: <http://ns.inria.fr/sparql-micro-service/api#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .


_:b3906 ns1:byArtist "Orelsan" ;
ns1:image <https://api.deezer.com/album/1261738/image> ;
ns1:name "Suicide Social - Single" ;
rdf:type ns1:MusicAlbum .

_:b3905 ns1:byArtist "Vald" ;
ns1:image <https://api.deezer.com/album/291023182/image> ;
ns1:name "V" ;
rdf:type ns1:MusicAlbum .
```

Docker environment

Instructions	https://github.com/frmichel/sparql-micro-service/tree/master/deployment/docker
SPARQL µ-services URL	http://localhost/service/<api>/<service>?...
YASGUI	file:///home/user/yasgui.html