RESTdesc

Efficient runtime service discovery and consumption.

Ruben Verborgh, Thomas Steiner,
Davy Van Deursen, Rik Van de Walle, Joaquim Gabarró Vallés

Talking with others is difficult when you're different.


The goal is not
to blend in...
© Eric Vest

What makes a Web service stand out from others?

Functionality.

Capturing functionality is vital for automated service consumption.

RESTdesc

Efficient runtime service discovery and consumption.

1. **Why?** – need and challenges
2. **What?** – syntax and structure
3. **So what?** – benefits and potential

RESTdesc

Efficient runtime service discovery and consumption.

1. **Why?** – need and challenges
2. **What?** – syntax and structure
3. **So what?** – benefits and potential

Example service: calculating photo height

photo height service

send a photo, retrieve its height in pixels

Several service description methods exist.

FACT: there are already lots of ways to describe that

- the input is an image;
- the output is a height in pixels.

FACT: none of them really helps… wait?

They don't describe functionality.

What height in pixels?

- The height of the face rectangle in the image?
- The optimal height on a certain mobile device?
- The height of the costume, mysteriously converted to pixels?

Functional descriptions tell the whole story.

What height in pixels?
Only the functional relation tells us this:

send a photo, retrieve its height in pixels

RESTdesc

Efficient runtime service discovery and consumption.

1. **Why?** – need and challenges
2. **What?** – syntax and structure
3. **So what?** – benefits and potential

Ingredients of RESTdesc

- HTTP
  - resources

REST-style HTTP is resource-oriented.

http://example.com/images/wolfie
This is a resource.

http://example.com/images/wolfie/height
This is a resource, too.

They're even the same resources as those on the Semantic Web: "**R**esource **D**escription **F**ramework".

Ingredients of RESTdesc

- **HTTP**
  - resources
- **Notation3 (N3)**
  - small superset of RDF

Everything in RDF has 3 parts.

```
:MyPhoto :a :Photo.
```

subject *verb* *object*

RDF is organized in vocabularies.

```
@prefix : <http://example.org/photo/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dbpedia: <http://dbpedia.org/resource/>.

:MyPhoto rdf:type dbpedia:Photo.
```

Ingredients of RESTdesc

- **HTTP**
  - resources
- **Notation3 (N3)**
  - small superset of RDF
- **your own vocabulary**

How do we describe this functionality?

photo height service

send a photo, retrieve its height in pixels

Photo height service in RESTdesc

```
@prefix : <http://example.org/ontologies/photos#>.
@prefix http: <http://www.w3.org/2006/http#>.
@prefix tmpl: <http://purl.org/restdesc/http-template#>.

{
  ?photo :photoId ?id.
}
=>
{
   _:request http:methodName "GET";
             tmpl:requestURI ("/photos/" ?id "/height");
             http:resp [ tmpl:represents ?pixels ].
   ?photo :height ?pixels.
}.
```

Not a new model.
This is RDF in Notation3,
a Semantic Web language.

Photo height service in RESTdesc

If your photo has identifier "wolfie"…

"wolfie"

```
{
  ?photo :photoId ?id.
}
=>
{
   _:request http:methodName "GET";
             tmpl:requestURI ("/photos/" ?id "/height");
             http:resp [ tmpl:represents ?pixels ].
   ?photo :height ?pixels.
}.
```

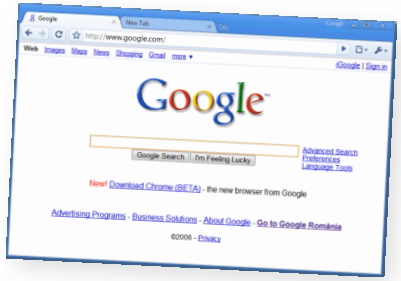
Photo height service in RESTdesc

… then …

```
{
  ?photo :photoId ?id.
}
=>
{
   _:request http:methodName "GET";
             tmpl:requestURI ("/photos/" ?id "/height");
             http:resp [ tmpl:represents ?pixels ].
   ?photo :height ?pixels.
}.
```

Photo height service in RESTdesc

…you can use HTTP to get /photos/wolfie/height…




```
{
  ?photo :photoId ?id.
}
=>
{
   _:request http:methodName "GET";
             tmpl:requestURI ("/photos/" ?id "/height");
             http:resp [ tmpl:represents ?pixels ].
   ?photo :height ?pixels.
}.
```

Photo height service in RESTdesc

…and the response will represent a value…

768

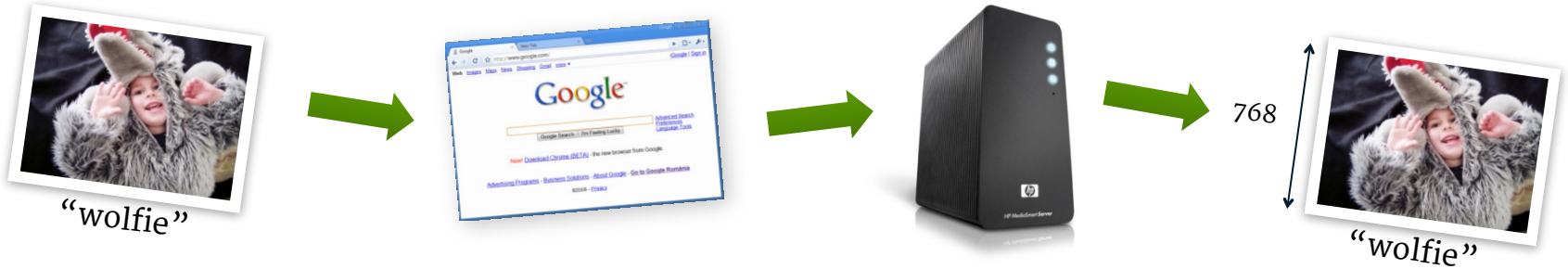
```
{
  ?photo :photoId ?id.
}
=>
{
   _:request http:methodName "GET";
             tmpl:requestURI ("/photos/" ?id "/height");
             http:resp [ tmpl:represents ?pixels ].
   ?photo :height ?pixels.
}.
```

Photo height service in RESTdesc

…which is
that photo's height.


```
{
  ?photo :photoId ?id.
}
=>
{
   _:request http:methodName "GET";
             tmpl:requestURI ("/photos/" ?id "/height");
             http:resp [ tmpl:represents ?pixels ].
   ?photo :height ?pixels.
}.
```

This description contains everything you need.




```
{
  ?photo :photoId ?id.
}
=>
{
   _:request http:methodName "GET";
             tmpl:requestURI ("/photos/" ?id "/height");
             http:resp [ tmpl:represents ?pixels ].
   ?photo :height ?pixels.
}.
```

It's that simple.
And it's that powerful.

RESTdesc

Efficient runtime service discovery and consumption.

1. **Why?** – need and challenges
2. **What?** – syntax and structure
3. **So what?** – benefits and potential

RESTdesc descriptions are simple, because…

They rely on HTTP and HTTP best practices.

- resource-oriented

They use the vocabulary you choose.

- descriptions adapts to your domain

They describe only what you need.

- concepts live in vocabularies
  - “height” is an integer, expresses a length in pixels

RESTdesc descriptions are powerful, because…

They use the Resource Description Framework.

- logically sound
- allows complex expressions
- interoperable with other services and data

They can be composed by generic reasoners.

- all N3 reasoners understand RESTdesc
- enables goal-driven service composition

RESTdesc enables straightforward automated use of services.

Seeing services as resources leads to elegant functional descriptions.

In functional descriptions, services are not defined by what they look like…


…but by
what they do.
http://restdesc.org/