Digital Public Space: Crafting URIs and Publishing Data

BBC Archive Development

Mo McRoberts, April 2012

What's this about?

- This is a (short) guide to publishing linked data on the Web.
- In particular, how to construct good URIs for the things described by that data, and how to make sure that the data can be easily retrieved and processed.
- These are not *rules* — they are guidelines to promote interoperability.

1. Dereferencing* the URI for something should return the data about that thing.

* Attempting to retrieve a *representation* of the resource associated with that URI.
For example, dereferencing an `http:` URI involves performing an HTTP request for it.

Implications

- This means that, in general, it's a good idea to use `http:` or `https:` URIs to identify your things.
- This brings about its own problem: what kind of URIs should refer to documents, and what to things described by those documents?

Use “hash” URIs

- Use a fragment identifier (e.g., “`#thing`”) in the URIs for things *described by* documents; don’t use fragments for the URIs of the documents.
- Fragment identifiers are never sent as part of HTTP requests (to do so is a direct violation of the protocol).
- To a *server,* the “hash” and “hashless” URIs look the same, and so return the same document.
- This is generally the simplest and most straightforward disambiguation mechanism.

For example:

`http://example.com/items/abc1234`

- A document URI

`http://example.com/items/abc1234#thing`

- A thing described by that document

Dereferencing *either* URI will result in the same document — and which can contain distinct descriptions of both the document and thing ("This data was published yesterday" versus "this book was published a year ago").

For example:

`http://example.com/items/abc1234`

- A document URI

`http://example.com/items/abc1234#thing`

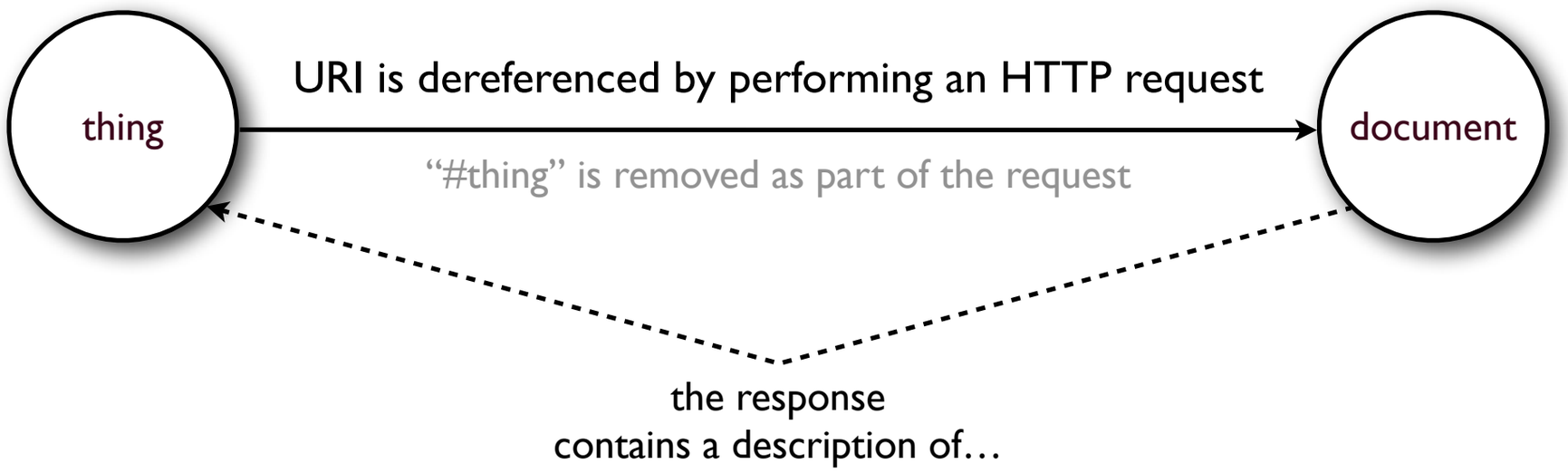
- A thing described by that document

Others linking to your data and referring to the thing you describe can differentiate between them properly (“I like this data” versus “I like this animal”).

For example:

http://example.com/items/abc1234#thing

http://example.com/items/abc1234



II. Be prepared to publish data in multiple formats, but use RDF/XML as a baseline.

Implications

- Employ HTTP Content Negotiation to serve different representations of documents to different clients.
- RDF/XML is the most widely-supported serialisation of RDF in consumers at present.
- The data format landscape doesn't stand still: best practice *will* change over time.
- Other RDF serialisations (and non-RDF formats) exist, and it's good to serve them too if you can.

- Include a `Vary: accept` header in your responses.

- Include a `Content-Location` header in your responses, giving the URL of the *specific representation* being served to the consumer.

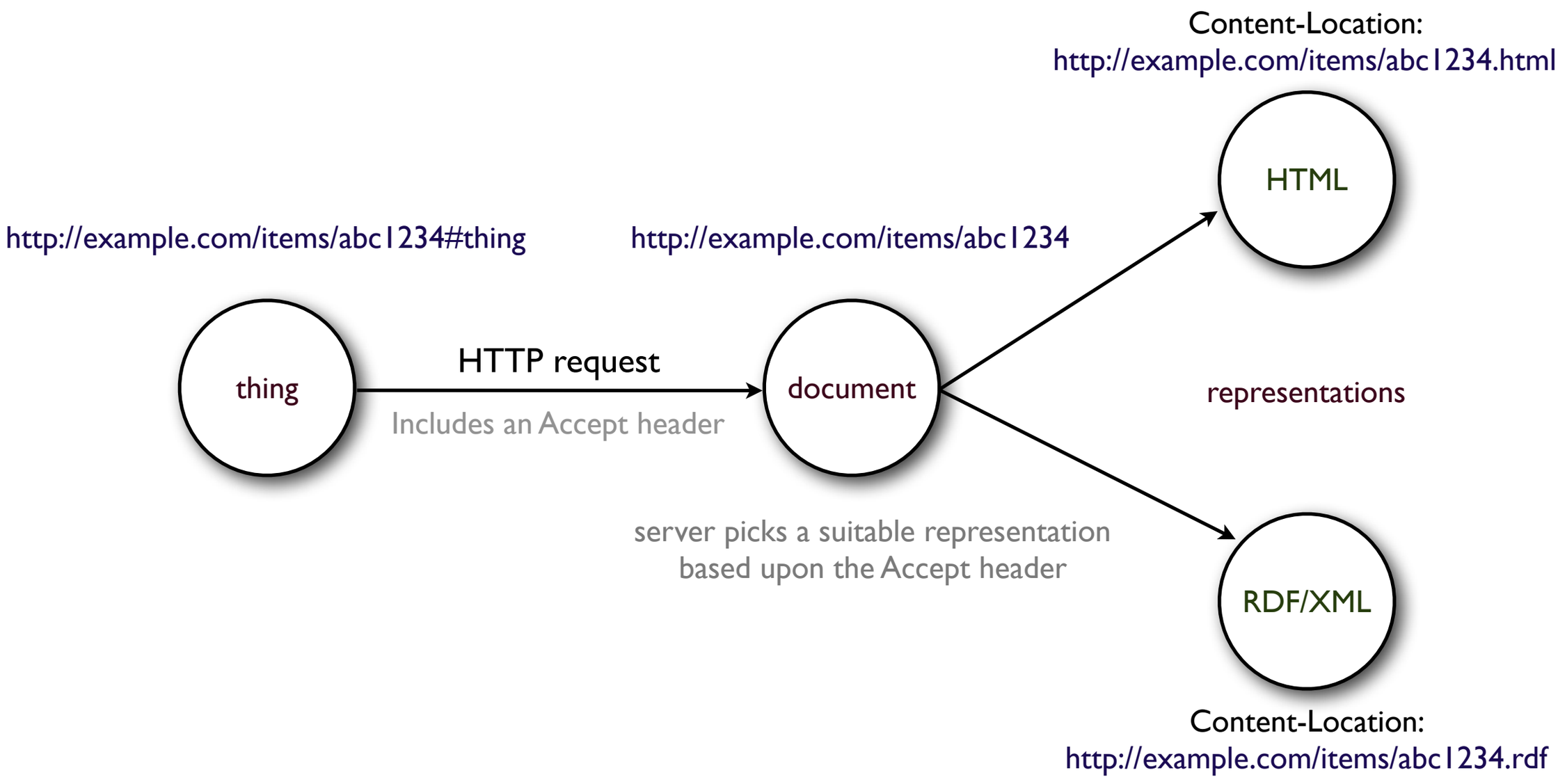
- Include a `Alternates` header in your responses to describe different variants.

- Include `<link rel="alternate" ...>` elements in any HTML representations.

- Use predictable extensions as part of your representation-specific URLs: e.g., `.rdf`, `.json`, `.html` — this doesn't aid software, but does make developers' lives easier.

- If possible, include information about your documents and the various available representations in those documents.

For example:


http://example.com/items/abc1234#thing
thing
HTTP request
Includes an Accept header
http://example.com/items/abc1234
document
server picks a suitable representation
based upon the Accept header
Content-Location:
http://example.com/items/abc1234.html
HTML
representations
RDF/XML
Content-Location:
http://example.com/items/abc1234.rdf

III. Construct URIs so as to minimise risk of change.

BBC Archive Development

Implications

- Avoid using other people's domain names in your canonical URIs.
- Avoid deriving URIs from metadata liable to need correction or otherwise changing over time (such as titles).
- When URIs do eventually need to change, handle this gracefully through 301 redirects and 410 ("Gone") responses.

For example:

`/items/d6707813#thing`

- Derived from an opaque permanent identifier.
- Does not need to change even when corrections or other changes to the data are made.
- Follows a pattern (`/items/:id#thing`), which aids developers' understanding.

IV. Make discovery easy

BBC Archive Development

Implications

- Don't put your "human-facing" HTML representations and your "machine-facing" structured data in two completely different places.
- Use predictable consistent patterns.
- Publish dataset descriptions describing search endpoints, URI patterns, and providing examples.
- Publish a "root" dataset on your domain root, or at `/.well-known/void` (or both).

V. Link and cross-reference

BBC Archive Development

Implications

- It's fine to publish descriptions of things "owned" by other people: create your own identifiers for them and link between them.
- Similarly, if you can, link to other people publishing descriptions of things you "own".
- Use `owl:sameAs` to indicate where two URIs really do refer to the same thing.
- Don't feel the need to generate your own URIs just for things you're referencing, though — linking through other people's URIs is fine (and good).

Resources

BBC Archive Development

- http://patterns.dataincubator.org/book/
  - “Linked Data Patterns” — Leigh Dodds and Ian Davis
- http://www.w3.org/Provider/Style/URI.html
  - “Cool URIs don’t change” — Sir Tim Berners-Lee.
- http://www.iana.org/assignments/media-types/index.html
  - IANA MIME type assignments, includes recommended file extensions.
- http://en.wikipedia.org/wiki/Content_negotiation
  - HTTP Content Negotiation.

- http://httpd.apache.org/docs/current/mod/mod_negotiation.html
  - Configuration content negotiation for static resources with Apache
- http://vocab.deri.ie/void
  - Vocabulary of Interlinked Datasets (VoID)
- http://vocab.deri.ie/void/autodiscovery
  - VoID Autodiscovery via a RFC5785 `.well-known` resource.

- http://dublincore.org/documents/dcmi-type-vocabulary/
  - DCMI Media Types — classes for describing documents
- http://purl.org/NET/mediatypes
  - Linked data for MIME types (for use with `dct:format`)
- http://tools.ietf.org/html/draft-ietf-http-alternates
  - “Alternates” HTTP response header (expired draft)

Case Study:

BBC /programmes

BBC Archive Development

“Thing” URI

`http://www.bbc.co.uk/programmes/b01cpfvb#programme`

- The fragment identifier — `#programme` — is used to differentiate information about the programme itself from the document describing it by giving them distinct URIs which are both dereferenced to the same document.
- Dereferencing this URI results in a request for the URI without the `#programme` fragment identifier.

Document URI

`http://www.bbc.co.uk/programmes/b01cpfvb`

- The server performs HTTP Content Negotiation when requests for this resource are received.
- A successful response will contain a specific representation of this document, and include a `Content-Location` header identifying it.
- You can copy and paste from the browser address bar into a linked data consumer application.

RDF/XML representation URL

`http://www.bbc.co.uk/programmes/b01cpfvb.rdf`

- When clients indicate that they prefer `application/rdf+xml`, they will be delivered this resource which contains a description of both itself, and of the programme (identified by its "thing" URI).

HTML (human-facing) representation URL

`http://www.bbc.co.uk/programmes/b01cpfvb.html`

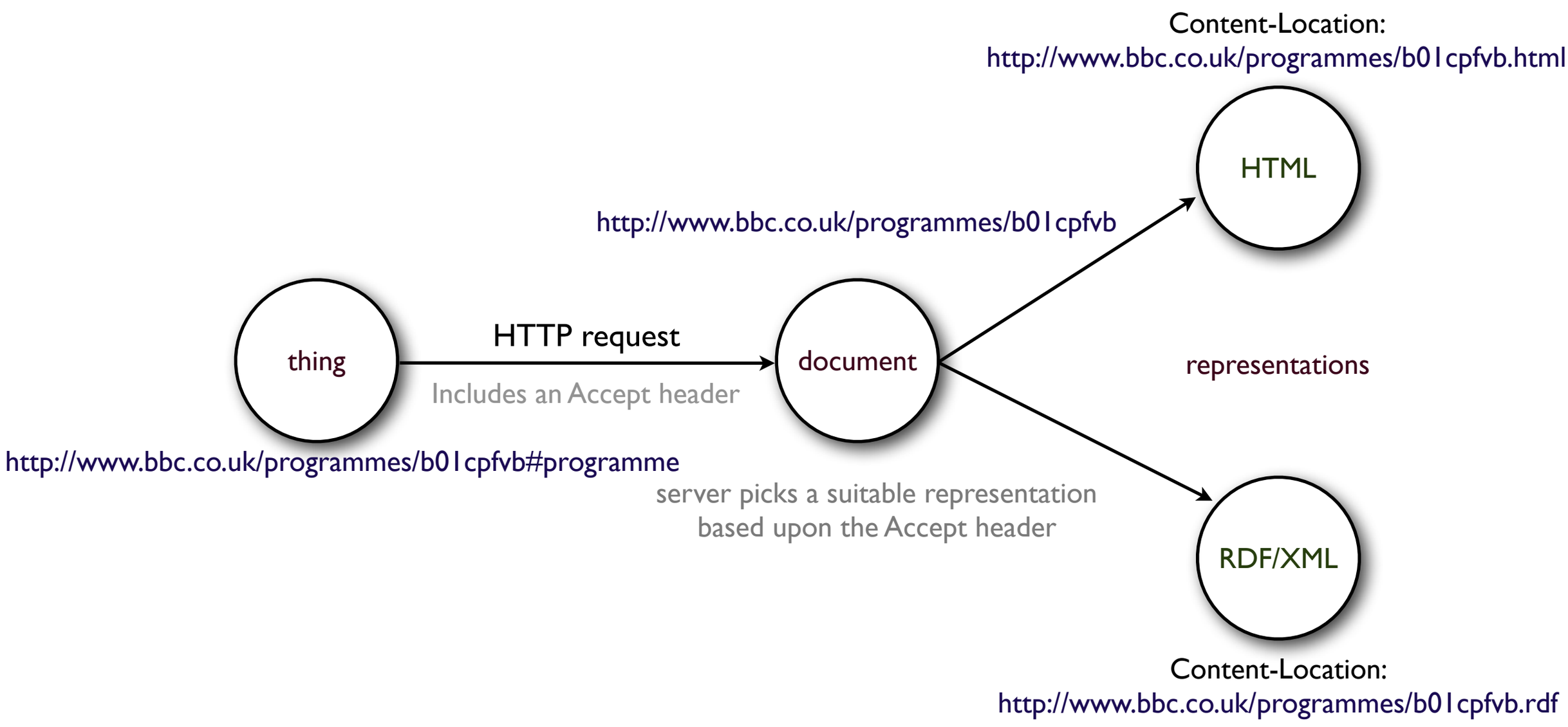
- This representation will generally be provided to ordinary web browsers requesting information about the programme.

JSON representation URL

`http://www.bbc.co.uk/programmes/b01cpfvb.json`

- Not all consumers will want or understand RDF serialisations: you can provide as many different representations as you're able to — JSON, YAML, CSV, XLS.

For example:


Content-Location:
http://www.bbc.co.uk/programmes/b01cpfvb.html
HTML
http://www.bbc.co.uk/programmes/b01cpfvb
thing
HTTP request
Includes an Accept header
document
representations
http://www.bbc.co.uk/programmes/b01cpfvb#programme
server picks a suitable representation
based upon the Accept header
RDF/XML
Content-Location:
http://www.bbc.co.uk/programmes/b01cpfvb.rdf

Try it! (with curl)

```
$ curl -H 'Accept: application/rdf+xml' http://www.bbc.co.uk/programmes/b01cpfvb
> GET http://www.bbc.co.uk/programmes/b01cpfvb HTTP/1.1
> User-Agent: curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4 OpenSSL/0.9.8r
zlib/1.2.5
> Host: www.bbc.co.uk
> Proxy-Connection: Keep-Alive
> Accept: application/rdf+xml
>
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Server: Apache
< Cache-Control: public, max-age=300, s-maxage=300
< Content-Type: application/rdf+xml
< Date: Thu, 05 Apr 2012 09:53:01 GMT
< Expires: Thu, 05 Apr 2012 09:58:00 GMT
< Access-Control-Allow-Origin: *
< X-Bbc-Licence-Url: http://backstage.bbc.co.uk/archives/2005/01/terms_of_use.html
< Accept-Ranges: bytes
< X-Bbc-Licence-Text: Access to and use of this feed is for non-commercial use only and
is covered by the BBC Backstage Terms of Use
< ETag: "307bcc6b5782dcc16c3eadee54bdb336"
< X-Programmes-Host: nolaps402.wtf.nolcontent.net:80
< Content-Length: 2077
* HTTP/1.0 connection set to keep alive!
< Connection: keep-alive
```