(Cool) URIs *

Jens Lechtenbörger

VM Neuland im Internet 2021

Contents

1 Introduction

1.1 Today's Core Questions

- What are URI, URL, URN?
- How are URIs used in the context of Linked Data?

1.2 Learning Objectives

- Explain how URIs can identify "things"
- Explain how hash URIs and URIs with redirection help to resolve ambiguity

2 URIs

2.1 Uniform Resource Identifier (URI)

- Character string to **identify** entities
- RFC 3986
- Examples from RFC 3986 (some containing DNS names)

*This PDF document is an inferior version of an OER HTML page; free/libre Org mode source repository.

  - `ftp : // ftp.is.co.za /rfc/rfc1808.txt`
  - `http : // www.ietf.org /rfc/rfc2396.txt`
  - `ldap : //[2001:db8::7]/c=GB?objectClass?one`
  - `mailto : John.Doe@ example.com`
  - `news : comp.infosystems.www.servers.unix`
  - `tel : +1-816-555-1212`
  - `telnet : //192.0.2.16:80/`
  - `urn : oasis:names:specification:docbook:dtd:xml:4.1.2`

2.1.1 URI Structure

- (Somewhat) Uniform, specified in RFC 3986
  - Different types of IDs under consistent format
  - Green parts on previous slide specify **schemes** for types
    - Our focus is on http(s)
- Syntax for absolute URIs depends on given scheme
  - <scheme>:<scheme-specific-part>
  - Scheme-specific part often structured
    - <scheme>://<authority><path>?<query>#fragment
    - E.g., `https://oer.gitlab.io/oer-courses/vm-neuland/URIs.html?default-navigation#slide-uri-structure`
- URIs subsume URLs and URNs

2.1.2 IRIs

- URIs are limited to ASCII characters
- **Internationalized Resource Identifiers** (IRIs) allow Unicode
  - Defined in RFC 3987
- Translations from IRI to encoded URI and back
- Examples
  - Visit `https://de.wikipedia.org/wiki/Dom%C3%A4ne`
    - Notice how browser translates encoded URI
  - Visit Wikipedia on red-black trees
    - Non-ASCII? Use copy&paste from address bar...

2.2 URLs, URNs

- Clarification in RFC 3305
  - Uniform Resource Locator (URL)
    - "URL is a useful but informal concept"
    - Identification of web resources via primary **access** mechanism
      - Network location, **address** of access point
    - Scalable
    - Address, thus potentially invalid
  - Uniform Resource Name (URN)
    - Permanent, location independent name of web resource
    - Registration of URN and URL for resource with URN-service
    - urn:... (RFC 8141)
    - E.g., urn:nbn:de:1111-200606299
      - Deutsche Nationalbibliothek
      - `https://nbn-resolving.org/`

3 Linked Data and Cool URIs

3.1 Linked Data

- "Linked Data" coined by Tim Berners-Lee, 2006; four rules

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)
4. Include links to other URIs. so that they can discover more things.

3.2 Cool URIs

- "Cool URI" defined in context of Semantic Web
  - Semantic Web: Use standard vocabularies/ontologies to make Web contents machine-readable and -processable
    - E.g., "+1-816-555-1212" is the telephone number of some person Bob
  - Cool URIs come with
    - Two requirements
      1. Be on the Web
      2. Be unambiguous
    - And three properties
      1. Simplicity
      2. Stability
      3. Manageability

3.2.1 Requirements in Detail

1. Be on the Web
   - Obvious
2. Ambiguity is mainly for URIs that do **not** describe Web pages
   - Example: Person vs her homepage

     ```
     <URI-of-alice> a foaf:Person;
       foaf:name "Alice";
       foaf:mbox <mailto:alice@example.com>;
       foaf:homepage <http://www.example.com/people/alice> .
     ```

   - What should `<URI-of-alice>` be?
     - Two solutions: Hash URIs and 303 URIs

3.2.2 Properties in Detail

- Simplicity
  - Short, mnemonic URIs
- Stability
  - URI for resource should persist
  - No implementation-specific bits and pieces such as `.php` and `.asp`
- Manageability, e.g.:
  - Current year in URI path
    - Allows change of URI-schema each year without breaking older URIs
  - Keeping 303 URIs on dedicated subdomain, e.g., `http://id.example.com/alice`

3.3 Hash URIs

- Hash URI = URI with fragment after hash sign
  - E.g., `https://ruben.verborgh.org/profile/#me`
    - URI to identify a person (as abstract concept)
  - Browser strips off fragment before GET request
    - E.g., `GET https://ruben.verborgh.org/profile/`
    - Thus, `https://ruben.verborgh.org/profile/#me` does **not** identify the returned document
  - Returned document contains descriptions for fragment identifiers, e.g.:
    - `:me a foaf:Person`
    - `:me foaf:name "Ruben Verborgh"@en`
    - `:me foaf:img <https://ruben.verborgh.org/images/ruben.jpg>`

3.3.1 Hash URIs with Content Negotiation

- Client sends `Accept` header in HTTP request, e.g.:

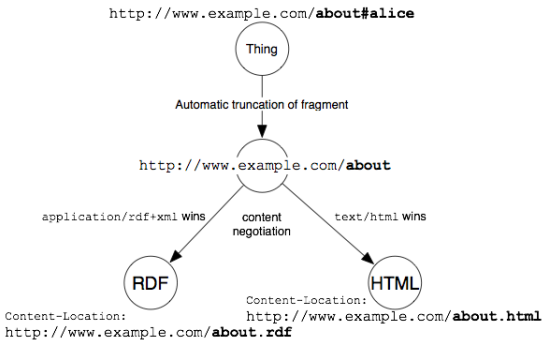

Figure 1: "Hash URI with content negotiation" Copyright © 2008 W3C® (MIT, ERCIM, Keio) under W3C Document License; from W3C

  - Human interpretation: `Accept: text/html`
  - Machine: `Accept: application/rdf+xml`
    - Or `Accept: */*` (server preferences determine content type)

- Server response may include `Content-Location`

3.4 303 URIs

- 303 = Redirection
  - If "thing" is requested, server does not respond with HTTP code `200 OK` but with `303 See other`
  - Two options

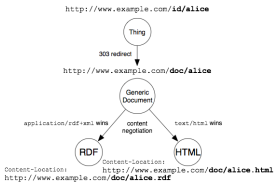

Figure 2: "303 URI with redirect to generic document" Copyright © 2008 W3C® (MIT, ERCIM, Keio) under W3C Document License; from W3C

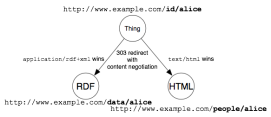

Figure 3: "303 URI with redirect with content negotiation" Copyright © 2008 W3C® (MIT, ERCIM, Keio) under W3C Document License; from W3C

4 Sample Solid URIs

4.1 URIs for People

- Solid WebIDs

  - E.g., `https://lechten.solidcommunity.net/profile/card#me`

- **Your tasks**

  1. Use Web Developer Tools of your browser to see HTTP requests and responses for above WebID.
  2. Verify that hash fragment is not part of `GET` request for WebID.
     - How does the request look like, what content is returned?
     - Beware, Firefox may mislead you
       * Use right-click on request, then copy headers, paste somewhere
  3. Note how content negotiation (different `Accept` header) is used later on with same URI to retrieve Turtle document.

4.2 A Suggestion

- Try out curl as command line tool for data transfer
  - `curl -H "Accept:  text/html" https://lechten.solidcommunity.net/profile/card`
  - `curl -H "Accept:  text/turtle" https://lechten.solidcommunity.net/profile/card`
  - `curl https://lechten.solidcommunity.net/profile/card`

5 Conclusions

5.1 Summary

- URIs identify things
  - URIs are names (at least, they can be)
- URIs are basis of Linked Data
  - Hash URIs and redirection remove ambiguities
  - "Understandable" links based on standardized vocabularies

License Information