INF3580 – Semantic Technologies – Spring 2010

Lecture 13: Publishing RDF Data on the Web

Martin Giese

11th May 2010

Department of Informatics

University of Oslo

Today's Plan

Outline

RDF on the Web

- RDF data exists in many forms:

RDF on the Web

- RDF data exists in many forms:
  - In RDF files, downloadable with HTTP, FTP, etc.

RDF on the Web

- RDF data exists in many forms:
    - In RDF files, downloadable with HTTP, FTP, etc.
        - FOAF profiles

RDF on the Web

- RDF data exists in many forms:
  - In RDF files, downloadable with HTTP, FTP, etc.
    - FOAF profiles
    - data files from dbpedia.org, geonames, etc.

RDF on the Web

- RDF data exists in many forms:
  - In RDF files, downloadable with HTTP, FTP, etc.
    - FOAF profiles
    - data files from dbpedia.org, geonames, etc.
    - In RSS 1.0 feeds

RDF on the Web

- RDF data exists in many forms:
    - In RDF files, downloadable with HTTP, FTP, etc.
        - FOAF profiles
        - data files from dbpedia.org, geonames, etc.
        - In RSS 1.0 feeds
    - As data model behind SPARQL query endpoints

RDF on the Web

- RDF data exists in many forms:
  - In RDF files, downloadable with HTTP, FTP, etc.
    - FOAF profiles
    - data files from dbpedia.org, geonames, etc.
    - In RSS 1.0 feeds
  - As data model behind SPARQL query endpoints
    - for instance dbpedia.org, dblp, and others

RDF on the Web

- RDF data exists in many forms:
  - In RDF files, downloadable with HTTP, FTP, etc.
    - FOAF profiles
    - data files from dbpedia.org, geonames, etc.
    - In RSS 1.0 feeds
  - As data model behind SPARQL query endpoints
    - for instance dbpedia.org, dblp, and others
  - Embedded in HTML, as RDFa

RDF on the Web

- RDF data exists in many forms:
  - In RDF files, downloadable with HTTP, FTP, etc.
    - FOAF profiles
    - data files from dbpedia.org, geonames, etc.
    - In RSS 1.0 feeds
  - As data model behind SPARQL query endpoints
    - for instance dbpedia.org, dblp, and others
  - Embedded in HTML, as RDFa
  - Embedded in PDF as XMP metadata

RDF on the Web

- RDF data exists in many forms:
  - In RDF files, downloadable with HTTP, FTP, etc.
    - FOAF profiles
    - data files from dbpedia.org, geonames, etc.
    - In RSS 1.0 feeds
  - As data model behind SPARQL query endpoints
    - for instance dbpedia.org, dblp, and others
  - Embedded in HTML, as RDFa
  - Embedded in PDF as XMP metadata
  - ...

RDF on the Web

- RDF data exists in many forms:
  - In RDF files, downloadable with HTTP, FTP, etc.
    - FOAF profiles
    - data files from dbpedia.org, geonames, etc.
    - In RSS 1.0 feeds
  - As data model behind SPARQL query endpoints
    - for instance dbpedia.org, dblp, and others
  - Embedded in HTML, as RDFa
  - Embedded in PDF as XMP metadata
  - ...
- How do I *find* data about something?

RDF on the Web

- RDF data exists in many forms:
  - In RDF files, downloadable with HTTP, FTP, etc.
    - FOAF profiles
    - data files from dbpedia.org, geonames, etc.
    - In RSS 1.0 feeds
  - As data model behind SPARQL query endpoints
    - for instance dbpedia.org, dblp, and others
  - Embedded in HTML, as RDFa
  - Embedded in PDF as XMP metadata
  - ...
- How do I *find* data about something?
  - Announcement of a cool new SPARQL endpoint

RDF on the Web

- RDF data exists in many forms:
  - In RDF files, downloadable with HTTP, FTP, etc.
    - FOAF profiles
    - data files from dbpedia.org, geonames, etc.
    - In RSS 1.0 feeds
  - As data model behind SPARQL query endpoints
    - for instance dbpedia.org, dblp, and others
  - Embedded in HTML, as RDFa
  - Embedded in PDF as XMP metadata
  - ...
- How do I *find* data about something?
  - Announcement of a cool new SPARQL endpoint
  - Semantic Web indices and search engines (Google to find some!)

RDF on the Web

- RDF data exists in many forms:
    - In RDF files, downloadable with HTTP, FTP, etc.
        - FOAF profiles
        - data files from dbpedia.org, geonames, etc.
        - In RSS 1.0 feeds
    - As data model behind SPARQL query endpoints
        - for instance dbpedia.org, dblp, and others
    - Embedded in HTML, as RDFa
    - Embedded in PDF as XMP metadata
    - ...
- How do I *find* data about something?
    - Announcement of a cool new SPARQL endpoint
    - Semantic Web indices and search engines (Google to find some!)
    - Links from HTML pages to RDF data

RDF on the Web

- RDF data exists in many forms:
  - In RDF files, downloadable with HTTP, FTP, etc.
    - FOAF profiles
    - data files from dbpedia.org, geonames, etc.
    - In RSS 1.0 feeds
  - As data model behind SPARQL query endpoints
    - for instance dbpedia.org, dblp, and others
  - Embedded in HTML, as RDFa
  - Embedded in PDF as XMP metadata
  - ...
- How do I *find* data about something?
  - Announcement of a cool new SPARQL endpoint
  - Semantic Web indices and search engines (Google to find some!)
  - Links from HTML pages to RDF data
  - “Linked Open Data” (LOD)

Outline

URIs

- URIs in RDF can have many different forms:

URIs

- URIs in RDF can have many different forms:
    - `http://www.google.com/` – a web page

URIs

- URIs in RDF can have many different forms:
  - `http://www.google.com/` – a web page
  - `mailto:jsmith@example.com` – a mailbox

URIs

- URIs in RDF can have many different forms:
  - `http://www.google.com/` – a web page
  - `mailto:jsmith@example.com` – a mailbox
  - `http://dbpedia.org/resource/Oslo` – a town

URIs

- URIs in RDF can have many different forms:
  - `http://www.google.com/` – a web page
  - `mailto:jsmith@example.com` – a mailbox
  - `http://dbpedia.org/resource/Oslo` – a town
  - `http://heim.ifi.uio.no/martingi/foaf.rg#me` – a person

URIs

- URIs in RDF can have many different forms:
  - `http://www.google.com/` – a web page
  - `mailto:jsmith@example.com` – a mailbox
  - `http://dbpedia.org/resource/Oslo` – a town
  - `http://heim.ifi.uio.no/martingi/foaf.rg#me` – a person
  - `tel:+47-22852737` – a telephone number

URIs

- URIs in RDF can have many different forms:
  - `http://www.google.com/` – a web page
  - `mailto:jsmith@example.com` – a mailbox
  - `http://dbpedia.org/resource/Oslo` – a town
  - `http://heim.ifi.uio.no/martingi/foaf.rg#me` – a person
  - `tel:+47-22852737` – a telephone number
  - `urn:isbn:0-395-36341-1` – a book

URIs

- URIs in RDF can have many different forms:
  - `http://www.google.com/` – a web page
  - `mailto:jsmith@example.com` – a mailbox
  - `http://dbpedia.org/resource/Oslo` – a town
  - `http://heim.ifi.uio.no/martingi/foaf.rg#me` – a person
  - `tel:+47-22852737` – a telephone number
  - `urn:isbn:0-395-36341-1` – a book
- Two basic types

URIs

- URIs in RDF can have many different forms:
  - `http://www.google.com/` – a web page
  - `mailto:jsmith@example.com` – a mailbox
  - `http://dbpedia.org/resource/Oslo` – a town
  - `http://heim.ifi.uio.no/martingi/foaf.rg#me` – a person
  - `tel:+47-22852737` – a telephone number
  - `urn:isbn:0-395-36341-1` – a book
- Two basic types
  - "information resources": downloadable documents

URIs

- URIs in RDF can have many different forms:
  - `http://www.google.com/` – a web page
  - `mailto:jsmith@example.com` – a mailbox
  - `http://dbpedia.org/resource/Oslo` – a town
  - `http://heim.ifi.uio.no/martingi/foaf.rg#me` – a person
  - `tel:+47-22852737` – a telephone number
  - `urn:isbn:0-395-36341-1` – a book
- Two basic types
  - "information resources": downloadable documents
  - "non-information resources": other entities

URIs

- URIs in RDF can have many different forms:
  - `http://www.google.com/` – a web page
  - `mailto:jsmith@example.com` – a mailbox
  - `http://dbpedia.org/resource/Oslo` – a town
  - `http://heim.ifi.uio.no/martingi/foaf.rg#me` – a person
  - `tel:+47-22852737` – a telephone number
  - `urn:isbn:0-395-36341-1` – a book
- Two basic types
  - "information resources": downloadable documents
  - "non-information resources": other entities
- Some provide a download protocol, but the resources don't exist

URIs

- URIs in RDF can have many different forms:
  - `http://www.google.com/` – a web page
  - `mailto:jsmith@example.com` – a mailbox
  - `http://dbpedia.org/resource/Oslo` – a town
  - `http://heim.ifi.uio.no/martingi/foaf.rg#me` – a person
  - `tel:+47-22852737` – a telephone number
  - `urn:isbn:0-395-36341-1` – a book
- Two basic types
  - "information resources": downloadable documents
  - "non-information resources": other entities
- Some provide a download protocol, but the resources don't exist
- Others are not dereferencable

URIs

- URIs in RDF can have many different forms:
  - `http://www.google.com/` – a web page
  - `mailto:jsmith@example.com` – a mailbox
  - `http://dbpedia.org/resource/Oslo` – a town
  - `http://heim.ifi.uio.no/martingi/foaf.rg#me` – a person
  - `tel:+47-22852737` – a telephone number
  - `urn:isbn:0-395-36341-1` – a book
- Two basic types
  - “information resources”: downloadable documents
  - “non-information resources”: other entities
- Some provide a download protocol, but the resources don’t exist
- Others are not dereferencable
- From the RDF standpoint, all are OK

URIs

- URIs in RDF can have many different forms:
  - `http://www.google.com/` – a web page
  - `mailto:jsmith@example.com` – a mailbox
  - `http://dbpedia.org/resource/Oslo` – a town
  - `http://heim.ifi.uio.no/martingi/foaf.rg#me` – a person
  - `tel:+47-22852737` – a telephone number
  - `urn:isbn:0-395-36341-1` – a book
- Two basic types
  - “information resources”: downloadable documents
  - “non-information resources”: other entities
- Some provide a download protocol, but the resources don’t exist
- Others are not dereferencable
- From the RDF standpoint, all are OK
- In practice, software wants to locate information

URIs

- URIs in RDF can have many different forms:
  - `http://www.google.com/` – a web page
  - `mailto:jsmith@example.com` – a mailbox
  - `http://dbpedia.org/resource/Oslo` – a town
  - `http://heim.ifi.uio.no/martingi/foaf.rg#me` – a person
  - `tel:+47-22852737` – a telephone number
  - `urn:isbn:0-395-36341-1` – a book
- Two basic types
  - "information resources": downloadable documents
  - "non-information resources": other entities
- Some provide a download protocol, but the resources don't exist
- Others are not dereferencable
- From the RDF standpoint, all are OK
- In practice, software wants to locate information
  - Protocols like http, ftp, etc. are an advantage

The Problem

- Need to differentiate between:
  - A web page or RDF file about Berlin
  - The city of Berlin
- e.g. the city was "created" around 1200...
- A URI for Berlin should not be an existing HTTP resource (why?)
- Need another way to retrieve information about a resource

$\neq$

Two Solutions

- The problem:

Two Solutions

- The problem:
    - Need to locate information *about* a resource

Two Solutions

- The problem:
    - Need to locate information *about* a resource
    - The URI cannot denote a *downloadable* resource

Two Solutions

- The problem:
  - Need to locate information *about* a resource
  - The URI cannot denote a *downloadable* resource
- Two W3C-recommended solutions:

Two Solutions

- The problem:
  - Need to locate information *about* a resource
  - The URI cannot denote a *downloadable* resource
- Two W3C-recommended solutions:
  - The hash-namespace solution

Two Solutions

- The problem:
    - Need to locate information *about* a resource
    - The URI cannot denote a *downloadable* resource
- Two W3C-recommended solutions:
    - The hash-namespace solution
    - The slash-namespace solution (aka HTTP 303 redirects)

Two Solutions

- The problem:
  - Need to locate information *about* a resource
  - The URI cannot denote a *downloadable* resource
- Two W3C-recommended solutions:
  - The hash-namespace solution
  - The slash-namespace solution (aka HTTP 303 redirects)
- To fully understand them, we need to have a look at HTTP!

HTTP

- HTTP Server listens to “requests” (usually on TCP/IP port 80)

HTTP

- HTTP Server listens to “requests” (usually on TCP/IP port 80)
- An HTTP client sends requests to the server and obtains responses

HTTP

- HTTP Server listens to “requests” (usually on TCP/IP port 80)
- An HTTP client sends requests to the server and obtains responses
- A typical request: `http://heim.ifi.uio.no/martingi/`

HTTP

- HTTP Server listens to “requests” (usually on TCP/IP port 80)
- An HTTP client sends requests to the server and obtains responses
- A typical request: `http://heim.ifi.uio.no/martingi/`
    - Connect to port 80 on `heim.ifi.uio.no`

HTTP

- HTTP Server listens to “requests” (usually on TCP/IP port 80)
- An HTTP client sends requests to the server and obtains responses
- A typical request: `http://heim.ifi.uio.no/martingi/`
  - Connect to port 80 on `heim.ifi.uio.no`
  - Send:

    ```
    GET /martingi/ HTTP/1.1
    User-Agent:  Mozilla/5.0 (X11; U; Linux i686; ...
    Accept:  text/html,application/xhtml+xml,...
    Accept-Language:  no, en
    Host:  heim.ifi.uio.no
    ...
    ```

    followed by a blank line

HTTP

- HTTP Server listens to “requests” (usually on TCP/IP port 80)
- An HTTP client sends requests to the server and obtains responses
- A typical request: `http://heim.ifi.uio.no/martingi/`
  - Connect to port 80 on `heim.ifi.uio.no`
  - Send:

    ```
    GET /martingi/ HTTP/1.1
    User-Agent:  Mozilla/5.0 (X11; U; Linux i686; ...
    Accept:  text/html,application/xhtml+xml,...
    Accept-Language:  no, en
    Host:  heim.ifi.uio.no
    ...
    ```

    followed by a blank line
- Other “methods”: `HEAD`, `POST`, `PUT`,. . .

HTTP (cont.)

- A typical response to the GET request:

```
HTTP/1.1 200 OK
Date:  Wed, 05 May 2010 14:15:24 GMT
Server:  Apache/2.2.14 (Unix) ...
Content-Length:  14348
Content-Type:  text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
    ...
```

HTTP (cont.)

- A typical response to the GET request:

```
HTTP/1.1 200 OK
Date: Wed, 05 May 2010 14:15:24 GMT
Server: Apache/2.2.14 (Unix) ...
Content-Length: 14348
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
    ...
```

- Result may vary depending on the `Accept:` choices in request

HTTP (cont.)

- A typical response to the GET request:

```
HTTP/1.1 200 OK
Date:  Wed, 05 May 2010 14:15:24 GMT
Server:  Apache/2.2.14 (Unix) ...
Content-Length:  14348
Content-Type:  text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
    ...
```

- Result may vary depending on the `Accept:` choices in request
- `200 OK` is not the only possible response ("status code")

HTTP (cont.)

- A typical response to the GET request:

```
HTTP/1.1 200 OK
Date:  Wed, 05 May 2010 14:15:24 GMT
Server:  Apache/2.2.14 (Unix) ...
Content-Length:  14348
Content-Type:  text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
    ...
```

- Result may vary depending on the `Accept:` choices in request
- `200 OK` is not the only possible response (“status code”)
  - `404 Not Found`

HTTP (cont.)

- A typical response to the GET request:

```
HTTP/1.1 200 OK
Date:  Wed, 05 May 2010 14:15:24 GMT
Server:  Apache/2.2.14 (Unix) ...
Content-Length:  14348
Content-Type:  text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
    ...
```

- Result may vary depending on the `Accept:` choices in request
- `200 OK` is not the only possible response ("status code")
  - `404 Not Found`
  - `401 Unauthorized`

HTTP (cont.)

- A typical response to the GET request:

```
HTTP/1.1 200 OK
Date:  Wed, 05 May 2010 14:15:24 GMT
Server:  Apache/2.2.14 (Unix) ...
Content-Length:  14348
Content-Type:  text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
    ...
```

- Result may vary depending on the `Accept:` choices in request
- `200 OK` is not the only possible response ("status code")
  - `404 Not Found`
  - `401 Unauthorized`
  - `303 See Other`

Fragment identifiers

- A *fragment identifier* is the part after # in a URI

```
http://en.wikipedia.org/wiki/Fragment_identifier#Examples
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
```

Fragment identifiers

- A *fragment identifier* is the part after # in a URI

  `http://en.wikipedia.org/wiki/Fragment_identifier#Examples`
  `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`

- HTTP specifies that fragment identifiers are processed client-side:

Fragment identifiers

- A *fragment identifier* is the part after # in a URI

  ```
  http://en.wikipedia.org/wiki/Fragment_identifier#Examples
  http://www.w3.org/1999/02/22-rdf-syntax-ns#type
  ```

- HTTP specifies that fragment identifiers are processed client-side:
  - GET request is sent without the fragment identifiers:

    ```
    GET /wiki/Fragment_identifier HTTP/1.1
    ```

Fragment identifiers

- A *fragment identifier* is the part after # in a URI

  `http://en.wikipedia.org/wiki/Fragment_identifier#Examples`
  `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`

- HTTP specifies that fragment identifiers are processed client-side:
  - GET request is sent without the fragment identifiers:

    ```
    GET /wiki/Fragment_identifier HTTP/1.1
    ```

  - fragment identifier is processed by client

Fragment identifiers

- A *fragment identifier* is the part after # in a URI

  ```
  http://en.wikipedia.org/wiki/Fragment_identifier#Examples
  http://www.w3.org/1999/02/22-rdf-syntax-ns#type
  ```

- HTTP specifies that fragment identifiers are processed client-side:
  - GET request is sent without the fragment identifiers:

    ```
    GET /wiki/Fragment_identifier HTTP/1.1
    ```

  - fragment identifier is processed by client
- For HTML or XHTML:

Fragment identifiers

- A *fragment identifier* is the part after # in a URI

  ```
  http://en.wikipedia.org/wiki/Fragment_identifier#Examples
  http://www.w3.org/1999/02/22-rdf-syntax-ns#type
  ```

- HTTP specifies that fragment identifiers are processed client-side:
  - GET request is sent without the fragment identifiers:

    ```
    GET /wiki/Fragment_identifier HTTP/1.1
    ```

  - fragment identifier is processed by client
- For HTML or XHTML:
  - Elements (sections titles, paragraphs, etc.) can have *id* attributes

    ```
    <h2 id="Examples">Examples</h2>
    ```

Fragment identifiers

- A *fragment identifier* is the part after # in a URI

  ```
  http://en.wikipedia.org/wiki/Fragment_identifier#Examples
  http://www.w3.org/1999/02/22-rdf-syntax-ns#type
  ```

- HTTP specifies that fragment identifiers are processed client-side:
  - GET request is sent without the fragment identifiers:

    ```
    GET /wiki/Fragment_identifier HTTP/1.1
    ```

  - fragment identifier is processed by client
- For HTML or XHTML:
  - Elements (sections titles, paragraphs, etc.) can have *id* attributes

    ```
    <h2 id="Examples">Examples</h2>
    ```

  - Browser will jump to element identified by fragment identifier

Fragment identifiers

- A *fragment identifier* is the part after # in a URI

  ```
  http://en.wikipedia.org/wiki/Fragment_identifier#Examples
  http://www.w3.org/1999/02/22-rdf-syntax-ns#type
  ```

- HTTP specifies that fragment identifiers are processed client-side:
  - GET request is sent without the fragment identifiers:

    ```
    GET /wiki/Fragment_identifier HTTP/1.1
    ```

  - fragment identifier is processed by client
- For HTML or XHTML:
  - Elements (sections titles, paragraphs, etc.) can have *id* attributes

    ```
    <h2 id="Examples">Examples</h2>
    ```

  - Browser will jump to element identified by fragment identifier
- Various uses with JavaScript (AJAX), PDF viewers, etc.

Hash namespaces

- For RDF served over HTTP: fragment identifiers identify resources:

Hash namespaces

- For RDF served over HTTP: fragment identifiers identify resources:
  - `http://bla.bla/bla#resource` is a resource

Hash namespaces

- For RDF served over HTTP: fragment identifiers identify resources:
  - `http://bla.bla/bla#resource` is a resource
  - `http://bla.bla/bla` is a document describing the resource

Hash namespaces

- For RDF served over HTTP: fragment identifiers identify resources:
  - `http://bla.bla/bla#resource` is a resource
  - `http://bla.bla/bla` is a document describing the resource
- E.g. FOAF files:

Hash namespaces

- For RDF served over HTTP: fragment identifiers identify resources:
  - `http://bla.bla/bla#resource` is a resource
  - `http://bla.bla/bla` is a document describing the resource
- E.g. FOAF files:
  - `http://heim.ifi.uio.no/martingi/foaf.rdf#me` - a person

Hash namespaces

- For RDF served over HTTP: fragment identifiers identify resources:
  - `http://bla.bla/bla#resource` is a resource
  - `http://bla.bla/bla` is a document describing the resource
- E.g. FOAF files:
  - `http://heim.ifi.uio.no/martingi/foaf.rdf#me` - a person
  - `http://heim.ifi.uio.no/martingi/foaf.rdf` - an RDF/XML file

Hash namespaces

- For RDF served over HTTP: fragment identifiers identify resources:
  - `http://bla.bla/bla#resource` is a resource
  - `http://bla.bla/bla` is a document describing the resource
- E.g. FOAF files:
  - `http://heim.ifi.uio.no/martingi/foaf.rdf#me` - a person
  - `http://heim.ifi.uio.no/martingi/foaf.rdf` - an RDF/XML file
- *by convention* the RDF file contains some triples involving resources identified by its fragments.

Hash namespaces

- For RDF served over HTTP: fragment identifiers identify resources:
  - `http://bla.bla/bla#resource` is a resource
  - `http://bla.bla/bla` is a document describing the resource
- E.g. FOAF files:
  - `http://heim.ifi.uio.no/martingi/foaf.rdf#me` - a person
  - `http://heim.ifi.uio.no/martingi/foaf.rdf` - an RDF/XML file
- *by convention* the RDF file contains some triples involving resources identified by its fragments.
- Can use the part of the URI until # as namespace

```
@prefix myfoaf:  <http://.../martingi/foaf.rdf#>
myfoaf:me foaf:givenname "Martin" .
```

Hash namespaces

- For RDF served over HTTP: fragment identifiers identify resources:
  - `http://bla.bla/bla#resource` is a resource
  - `http://bla.bla/bla` is a document describing the resource
- E.g. FOAF files:
  - `http://heim.ifi.uio.no/martingi/foaf.rdf#me` - a person
  - `http://heim.ifi.uio.no/martingi/foaf.rdf` - an RDF/XML file
- *by convention* the RDF file contains some triples involving resources identified by its fragments.
- Can use the part of the URI until # as namespace

```
@prefix myfoaf:  <http://.../martingi/foaf.rdf#>
myfoaf:me foaf:givenname "Martin" .
```

- This is known as a "hash namespace"

Hash namespaces – pros and cons

- Hash namespaces solve our problem:

Hash namespaces – pros and cons

- Hash namespaces solve our problem:
  - Resources are separate from documents about them

Hash namespaces – pros and cons

- Hash namespaces solve our problem:
  - Resources are separate from documents about them
  - It is possible to find a document given a resource URI

Hash namespaces – pros and cons

- Hash namespaces solve our problem:
  - Resources are separate from documents about them
  - It is possible to find a document given a resource URI
- Moreover:

Hash namespaces – pros and cons

- Hash namespaces solve our problem:
    - Resources are separate from documents about them
    - It is possible to find a document given a resource URI
- Moreover:
    - Fetching the right document is done automatically by HTTP

Hash namespaces – pros and cons

- Hash namespaces solve our problem:
  - Resources are separate from documents about them
  - It is possible to find a document given a resource URI
- Moreover:
  - Fetching the right document is done automatically by HTTP
  - It is enough to publish the RDF file on an HTTP server

Hash namespaces – pros and cons

- Hash namespaces solve our problem:
    - Resources are separate from documents about them
    - It is possible to find a document given a resource URI
- Moreover:
    - Fetching the right document is done automatically by HTTP
    - It is enough to publish the RDF file on an HTTP server
    - Very low tech and fool proof, in other words!

Hash namespaces – pros and cons

- Hash namespaces solve our problem:
  - Resources are separate from documents about them
  - It is possible to find a document given a resource URI
- Moreover:
  - Fetching the right document is done automatically by HTTP
  - It is enough to publish the RDF file on an HTTP server
  - Very low tech and fool proof, in other words!
- However:

Hash namespaces – pros and cons

- Hash namespaces solve our problem:
  - Resources are separate from documents about them
  - It is possible to find a document given a resource URI
- Moreover:
  - Fetching the right document is done automatically by HTTP
  - It is enough to publish the RDF file on an HTTP server
  - Very low tech and fool proof, in other words!
- However:
  - All data published this way about all entities in a hash namespace needs to be stored in the same RDF file

    `http://brreg.no/bedrifter.rdf#974760673`

Hash namespaces – pros and cons

- Hash namespaces solve our problem:
  - Resources are separate from documents about them
  - It is possible to find a document given a resource URI
- Moreover:
  - Fetching the right document is done automatically by HTTP
  - It is enough to publish the RDF file on an HTTP server
  - Very low tech and fool proof, in other words!
- However:
  - All data published this way about all entities in a hash namespace needs to be stored in the same RDF file

    `http://brreg.no/bedrifter.rdf#974760673`
  - URI says much about data organization. RDF file name baked in!

Hash namespaces – pros and cons

- Hash namespaces solve our problem:
  - Resources are separate from documents about them
  - It is possible to find a document given a resource URI
- Moreover:
  - Fetching the right document is done automatically by HTTP
  - It is enough to publish the RDF file on an HTTP server
  - Very low tech and fool proof, in other words!
- However:
  - All data published this way about all entities in a hash namespace needs to be stored in the same RDF file

    `http://brreg.no/bedrifter.rdf#974760673`
  - URI says much about data organization. RDF file name baked in!
  - No way to change the organization without changing URIs

HTTP Redirection

- Reminder: HTTP responses start with a “status code”

HTTP Redirection

- Reminder: HTTP responses start with a “status code”
    - Usually “200 OK”, if the document was found and can be served

HTTP Redirection

- Reminder: HTTP responses start with a “status code”
  - Usually “200 OK”, if the document was found and can be served
  - “404 Not Found”, if the document does not exist

HTTP Redirection

- Reminder: HTTP responses start with a “status code”
  - Usually “200 OK”, if the document was found and can be served
  - “404 Not Found”, if the document does not exist
- One of the possible status codes is “303 See Other”

HTTP Redirection

- Reminder: HTTP responses start with a “status code”
  - Usually “200 OK”, if the document was found and can be served
  - “404 Not Found”, if the document does not exist
- One of the possible status codes is “303 See Other”
- Always comes with a `Location:` field in the response

HTTP Redirection

- Reminder: HTTP responses start with a “status code”
  - Usually “200 OK”, if the document was found and can be served
  - “404 Not Found”, if the document does not exist
- One of the possible status codes is “303 See Other”
- Always comes with a `Location:` field in the response
- Tells the client to submit a “GET” request to that location

HTTP Redirection

- Reminder: HTTP responses start with a “status code”
  - Usually “200 OK”, if the document was found and can be served
  - “404 Not Found”, if the document does not exist
- One of the possible status codes is “303 See Other”
- Always comes with a `Location:` field in the response
- Tells the client to submit a “GET” request to that location
- Also known as “303 redirection”

HTTP Redirection

- Reminder: HTTP responses start with a “status code”
  - Usually “200 OK”, if the document was found and can be served
  - “404 Not Found”, if the document does not exist
- One of the possible status codes is “303 See Other”
- Always comes with a `Location:` field in the response
- Tells the client to submit a “GET” request to that location
- Also known as “303 redirection”
- Followed by all modern HTTP clients

HTTP Redirection

- Reminder: HTTP responses start with a “status code”
    - Usually “200 OK”, if the document was found and can be served
    - “404 Not Found”, if the document does not exist
- One of the possible status codes is “303 See Other”
- Always comes with a `Location:` field in the response
- Tells the client to submit a “GET” request to that location
- Also known as “303 redirection”
- Followed by all modern HTTP clients
- Often used when URIs have changed

Example of 303 Redirection

- User requests `http://www.sun.com/`

Example of 303 Redirection

- User requests `http://www.sun.com/`
- Client sends request to `www.sun.com`

```
GET / HTTP/1.1
Host:  www.sun.com
```

Example of 303 Redirection

- User requests `http://www.sun.com/`
- Client sends request to `www.sun.com`

```
GET / HTTP/1.1
Host:  www.sun.com
```

- Sun was bought by Oracle. . . Server responds:

```
HTTP/1.1 303 See Other
Location:  http://www.oracle.com/
```

Example of 303 Redirection

- User requests `http://www.sun.com/`
- Client sends request to `www.sun.com`

```
GET / HTTP/1.1
Host:  www.sun.com
```

- Sun was bought by Oracle. . . Server responds:

```
HTTP/1.1 303 See Other
Location:  http://www.oracle.com/
```

- Client sends new request to `www.oracle.com`:

```
GET / HTTP/1.1
Host:  www.oracle.com
```

Example of 303 Redirection

- User requests `http://www.sun.com/`
- Client sends request to `www.sun.com`

```
GET / HTTP/1.1
Host:  www.sun.com
```

- Sun was bought by Oracle… Server responds:

```
HTTP/1.1 303 See Other
Location:  http://www.oracle.com/
```

- Client sends new request to `www.oracle.com`:

```
GET / HTTP/1.1
Host:  www.oracle.com
```

- Server at `www.oracle.com` responds:

```
HTTP/1.1 200 OK
Content-Type:  text/html
...
```

303 Redirection for RDF

- Find information about `http://dbpedia.org/resource/Oslo`

303 Redirection for RDF

- Find information about `http://dbpedia.org/resource/Oslo`
- Send "GET" request to server `dbpedia.org`:

```
GET /resource/Oslo HTTP/1.1
Accept:  application/rdf+xml
```

303 Redirection for RDF

- Find information about `http://dbpedia.org/resource/Oslo`
- Send “GET” request to server `dbpedia.org`:

```
GET /resource/Oslo HTTP/1.1
Accept:  application/rdf+xml
```

- Server `dbpedia.org` recognizes this as a non-information resource

303 Redirection for RDF

- Find information about `http://dbpedia.org/resource/Oslo`
- Send “GET” request to server `dbpedia.org`:
```
GET /resource/Oslo HTTP/1.1
Accept:  application/rdf+xml
```
- Server `dbpedia.org` recognizes this as a non-information resource
- Redirects to a file with data about the city of Oslo:
```
HTTP/1.1 303 See Other
Location:  http://dbpedia.org/data/Oslo.xml
```

303 Redirection for RDF

- Find information about `http://dbpedia.org/resource/Oslo`
- Send “GET” request to server `dbpedia.org`:

```
GET /resource/Oslo HTTP/1.1
Accept:  application/rdf+xml
```

- Server `dbpedia.org` recognizes this as a non-information resource
- Redirects to a file with data about the city of Oslo:

```
HTTP/1.1 303 See Other
Location:  http://dbpedia.org/data/Oslo.xml
```

- Browser can now send a new request for that location:

```
GET /data/Oslo.xml HTTP/1.1
Accept:  application/rdf+xml
```

303 Redirection for RDF

- Find information about `http://dbpedia.org/resource/Oslo`
- Send "GET" request to server `dbpedia.org`:
  ```
  GET /resource/Oslo HTTP/1.1
  Accept:  application/rdf+xml
  ```
- Server `dbpedia.org` recognizes this as a non-information resource
- Redirects to a file with data about the city of Oslo:
  ```
  HTTP/1.1 303 See Other
  Location:  http://dbpedia.org/data/Oslo.xml
  ```
- Browser can now send a new request for that location:
  ```
  GET /data/Oslo.xml HTTP/1.1
  Accept:  application/rdf+xml
  ```
- This time the server responds with the requested document:
  ```
  HTTP/1.1 200 OK
  Content-Type:  application/rdf+xml
  ...
  ```

Slash Namespaces

- Common to use URIs with a slash (/) as last non-identifier character:

  `http://dbpedia.org/resource/Oslo`

Slash Namespaces

- Common to use URIs with a slash (/) as last non-identifier character:

  ```
  http://dbpedia.org/resource/Oslo
  ```

- Can use URI up to last slash as namespace:

  ```
  @prefix dbpedia:  <http://dbpedia.org/resource/>
  dbpedia:Oslo dbprop:maySnowCm "0" .
  ```

Slash Namespaces

- Common to use URIs with a slash (/) as last non-identifier character:

```
http://dbpedia.org/resource/Oslo
```

- Can use URI up to last slash as namespace:

```
@prefix dbpedia:  <http://dbpedia.org/resource/>
dbpedia:Oslo dbprop:maySnowCm "0" .
```

- Known as a “slash namespace”

Slash Namespaces

- Common to use URIs with a slash (/) as last non-identifier character:

  ```
  http://dbpedia.org/resource/Oslo
  ```

- Can use URI up to last slash as namespace:

  ```
  @prefix dbpedia:  <http://dbpedia.org/resource/>
  dbpedia:Oslo dbprop:maySnowCm "0" .
  ```

- Known as a “slash namespace”
- Advantages over hash namespaces:

Slash Namespaces

- Common to use URIs with a slash (/) as last non-identifier character:

```
http://dbpedia.org/resource/Oslo
```

- Can use URI up to last slash as namespace:

```
@prefix dbpedia:  <http://dbpedia.org/resource/>
dbpedia:Oslo dbprop:maySnowCm "0" .
```

- Known as a “slash namespace”
- Advantages over hash namespaces:
  - Whole URI is sent to server, so...

Slash Namespaces

- Common to use URIs with a slash (/) as last non-identifier character:

```
http://dbpedia.org/resource/Oslo
```

- Can use URI up to last slash as namespace:

```
@prefix dbpedia:  <http://dbpedia.org/resource/>
dbpedia:Oslo dbprop:maySnowCm "0" .
```

- Known as a “slash namespace”
- Advantages over hash namespaces:
  - Whole URI is sent to server, so...
  - Possible to redirect different resources to different documents

Slash Namespaces

- Common to use URIs with a slash (/) as last non-identifier character:

  ```
  http://dbpedia.org/resource/Oslo
  ```

- Can use URI up to last slash as namespace:

  ```
  @prefix dbpedia:  <http://dbpedia.org/resource/>
  dbpedia:Oslo dbprop:maySnowCm "0" .
  ```

- Known as a “slash namespace”
- Advantages over hash namespaces:
  - Whole URI is sent to server, so. . .
  - Possible to redirect different resources to different documents
  - Possible to change redirection without changing URIs

Slash Namespaces

- Common to use URIs with a slash (/) as last non-identifier character:

  ```
  http://dbpedia.org/resource/Oslo
  ```

- Can use URI up to last slash as namespace:

  ```
  @prefix dbpedia:  <http://dbpedia.org/resource/>
  dbpedia:Oslo dbprop:maySnowCm "0" .
  ```

- Known as a "slash namespace"
- Advantages over hash namespaces:
  - Whole URI is sent to server, so...
  - Possible to redirect different resources to different documents
  - Possible to change redirection without changing URIs
- Requires some more server configuration

Slash Namespaces

- Common to use URIs with a slash (/) as last non-identifier character:

  `http://dbpedia.org/resource/Oslo`

- Can use URI up to last slash as namespace:

  ```
  @prefix dbpedia:  <http://dbpedia.org/resource/>
  dbpedia:Oslo dbprop:maySnowCm "0" .
  ```

- Known as a “slash namespace”
- Advantages over hash namespaces:
  - Whole URI is sent to server, so. . .
  - Possible to redirect different resources to different documents
  - Possible to change redirection without changing URIs
- Requires some more server configuration
- See recipes at `http://www.w3.org/TR/swbp-vocab-pub/`

Slash Namespaces

- Common to use URIs with a slash (/) as last non-identifier character:

  ```
  http://dbpedia.org/resource/Oslo
  ```

- Can use URI up to last slash as namespace:

  ```
  @prefix dbpedia:  <http://dbpedia.org/resource/>
  dbpedia:Oslo dbprop:maySnowCm "0" .
  ```

- Known as a “slash namespace”
- Advantages over hash namespaces:
  - Whole URI is sent to server, so. . .
  - Possible to redirect different resources to different documents
  - Possible to change redirection without changing URIs
- Requires some more server configuration
- See recipes at `http://www.w3.org/TR/swbp-vocab-pub/`
- See also `http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/`

Serving Vocabularies

- What about classes and properties?

Serving Vocabularies

- What about classes and properties?
- Identified by URIs:

```
http://xmlns.com/foaf/0.1/Person
http://xmlns.com/foaf/0.1/knows
http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
```

Serving Vocabularies

- What about classes and properties?
- Identified by URIs:

```
http://xmlns.com/foaf/0.1/Person
http://xmlns.com/foaf/0.1/knows
http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
```

- What should be served in response to these?

Serving Vocabularies

- What about classes and properties?
- Identified by URIs:

```
http://xmlns.com/foaf/0.1/Person
http://xmlns.com/foaf/0.1/knows
http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
```

- What should be served in response to these?
  - A description of the “vocabulary” defining the term

Serving Vocabularies

- What about classes and properties?
- Identified by URIs:

  ```
  http://xmlns.com/foaf/0.1/Person
  http://xmlns.com/foaf/0.1/knows
  http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement
  http://www.w3.org/1999/02/22-rdf-syntax-ns#type
  ```

- What should be served in response to these?
  - A description of the “vocabulary” defining the term
  - Often an RDF file with RDFS or OWL/RDF content

Serving Vocabularies

- What about classes and properties?
- Identified by URIs:

```
http://xmlns.com/foaf/0.1/Person
http://xmlns.com/foaf/0.1/knows
http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
```

- What should be served in response to these?
  - A description of the “vocabulary” defining the term
  - Often an RDF file with RDFS or OWL/RDF content
  - Sometimes (FOAF) just an HTML page with documentation

Serving Vocabularies

- What about classes and properties?
- Identified by URIs:

```
http://xmlns.com/foaf/0.1/Person
http://xmlns.com/foaf/0.1/knows
http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
```

- What should be served in response to these?
  - A description of the “vocabulary” defining the term
  - Often an RDF file with RDFS or OWL/RDF content
  - Sometimes (FOAF) just an HTML page with documentation
- Mechanisms are the same as for “ordinary” RDF data

Serving Vocabularies

- What about classes and properties?
- Identified by URIs:

```
http://xmlns.com/foaf/0.1/Person
http://xmlns.com/foaf/0.1/knows
http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
```

- What should be served in response to these?
  - A description of the “vocabulary” defining the term
  - Often an RDF file with RDFS or OWL/RDF content
  - Sometimes (FOAF) just an HTML page with documentation
- Mechanisms are the same as for “ordinary” RDF data
- A single RDF file (hash namespace) is usually OK

Serving Vocabularies

- What about classes and properties?
- Identified by URIs:

```
http://xmlns.com/foaf/0.1/Person
http://xmlns.com/foaf/0.1/knows
http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
```

- What should be served in response to these?
  - A description of the “vocabulary” defining the term
  - Often an RDF file with RDFS or OWL/RDF content
  - Sometimes (FOAF) just an HTML page with documentation
- Mechanisms are the same as for “ordinary” RDF data
- A single RDF file (hash namespace) is usually OK
- Should also serve the vocabulary description for the “vocabulary URI”:

```
http://xmlns.com/foaf/0.1/
http://www.w3.org/1999/02/22-rdf-syntax-ns#
```

HTTP Content Type Negotiation

- In HTTP, data formats are identified by “internet media types”

HTTP Content Type Negotiation

- In HTTP, data formats are identified by “internet media types”
  - Previously known as MIME types

HTTP Content Type Negotiation

- In HTTP, data formats are identified by “internet media types”
  - Previously known as MIME types
  - `text/html, image/jpeg, application/pdf,...`

HTTP Content Type Negotiation

- In HTTP, data formats are identified by “internet media types”
  - Previously known as MIME types
  - `text/html, image/jpeg, application/pdf,...`
- RDF media types:

HTTP Content Type Negotiation

- In HTTP, data formats are identified by "internet media types"
  - Previously known as MIME types
  - `text/html, image/jpeg, application/pdf,...`
- RDF media types:
  - RDF/XML: `application/rdf+xml`

HTTP Content Type Negotiation

- In HTTP, data formats are identified by "internet media types"
  - Previously known as MIME types
  - `text/html`, `image/jpeg`, `application/pdf`,...
- RDF media types:
  - RDF/XML: `application/rdf+xml`
  - Turtle: `text/turtle` (registration pending)

HTTP Content Type Negotiation

- In HTTP, data formats are identified by “internet media types”
    - Previously known as MIME types
    - `text/html`, `image/jpeg`, `application/pdf`,...
- RDF media types:
    - RDF/XML: `application/rdf+xml`
    - Turtle: `text/turtle` (registration pending)
    - N3: `text/rdf+n3` (not registered)

HTTP Content Type Negotiation

- In HTTP, data formats are identified by "internet media types"
  - Previously known as MIME types
  - `text/html`, `image/jpeg`, `application/pdf`,...
- RDF media types:
  - RDF/XML: `application/rdf+xml`
  - Turtle: `text/turtle` (registration pending)
  - N3: `text/rdf+n3` (not registered)
- Client sends accepted media types in `Accept:` header:

HTTP Content Type Negotiation

- In HTTP, data formats are identified by “internet media types”
  - Previously known as MIME types
  - `text/html, image/jpeg, application/pdf,...`
- RDF media types:
  - RDF/XML: `application/rdf+xml`
  - Turtle: `text/turtle` (registration pending)
  - N3: `text/rdf+n3` (not registered)
- Client sends accepted media types in `Accept:` header:
  - `Accept:  text/html, text/plain`

HTTP Content Type Negotiation

- In HTTP, data formats are identified by “internet media types”
  - Previously known as MIME types
  - `text/html, image/jpeg, application/pdf,...`
- RDF media types:
  - RDF/XML: `application/rdf+xml`
  - Turtle: `text/turtle` (registration pending)
  - N3: `text/rdf+n3` (not registered)
- Client sends accepted media types in `Accept:` header:
  - `Accept:  text/html, text/plain`
  - Can additionally add “quality factors” to specify preference

HTTP Content Type Negotiation

- In HTTP, data formats are identified by “internet media types”
  - Previously known as MIME types
  - `text/html`, `image/jpeg`, `application/pdf`,...
- RDF media types:
  - RDF/XML: `application/rdf+xml`
  - Turtle: `text/turtle` (registration pending)
  - N3: `text/rdf+n3` (not registered)
- Client sends accepted media types in `Accept:` header:
  - `Accept:  text/html, text/plain`
  - Can additionally add “quality factors” to specify preference
- Server chooses sent media type:

HTTP Content Type Negotiation

- In HTTP, data formats are identified by “internet media types”
  - Previously known as MIME types
  - `text/html, image/jpeg, application/pdf,...`
- RDF media types:
  - RDF/XML: `application/rdf+xml`
  - Turtle: `text/turtle` (registration pending)
  - N3: `text/rdf+n3` (not registered)
- Client sends accepted media types in `Accept:` header:
  - `Accept:  text/html, text/plain`
  - Can additionally add “quality factors” to specify preference
- Server chooses sent media type:
  - Picks the preferred one among available types

HTTP Content Type Negotiation

- In HTTP, data formats are identified by “internet media types”
  - Previously known as MIME types
  - `text/html`, `image/jpeg`, `application/pdf`,...
- RDF media types:
  - RDF/XML: `application/rdf+xml`
  - Turtle: `text/turtle` (registration pending)
  - N3: `text/rdf+n3` (not registered)
- Client sends accepted media types in `Accept:` header:
  - `Accept: text/html, text/plain`
  - Can additionally add “quality factors” to specify preference
- Server chooses sent media type:
  - Picks the preferred one among available types
  - Sends the media type of the response in the header

HTTP Content Type Negotiation

- In HTTP, data formats are identified by “internet media types”
  - Previously known as MIME types
  - `text/html, image/jpeg, application/pdf,...`
- RDF media types:
  - RDF/XML: `application/rdf+xml`
  - Turtle: `text/turtle` (registration pending)
  - N3: `text/rdf+n3` (not registered)
- Client sends accepted media types in `Accept:` header:
  - `Accept: text/html, text/plain`
  - Can additionally add “quality factors” to specify preference
- Server chooses sent media type:
  - Picks the preferred one among available types
  - Sends the media type of the response in the header
  - `Content-Type: text/html`

Content Type Negotiation for RDF

- Given the URI of a non-information resource. . .

Content Type Negotiation for RDF

- Given the URI of a non-information resource. . .
  - A semantic web applications wants RDF data, as discussed

Content Type Negotiation for RDF

- Given the URI of a non-information resource. . .
  - A semantic web applications wants RDF data, as discussed
  - A regular WWW browser wants HTML, human readable

Content Type Negotiation for RDF

- Given the URI of a non-information resource. . .
  - A semantic web applications wants RDF data, as discussed
  - A regular WWW browser wants HTML, human readable
- This can be achieved using HTTP content type negotiation!

Content Type Negotiation for RDF

- Given the URI of a non-information resource. . .
  - A semantic web applications wants RDF data, as discussed
  - A regular WWW browser wants HTML, human readable
- This can be achieved using HTTP content type negotiation!
- Semantic web client:

Content Type Negotiation for RDF

- Given the URI of a non-information resource. . .
  - A semantic web applications wants RDF data, as discussed
  - A regular WWW browser wants HTML, human readable
- This can be achieved using HTTP content type negotiation!
- Semantic web client:
  - Requests RDF, e.g. `Accept:  application/rdf+xml, text/turtle`

Content Type Negotiation for RDF

- Given the URI of a non-information resource...
  - A semantic web applications wants RDF data, as discussed
  - A regular WWW browser wants HTML, human readable
- This can be achieved using HTTP content type negotiation!
- Semantic web client:
  - Requests RDF, e.g. `Accept: application/rdf+xml, text/turtle`
  - Server uses e.g. 303 redirection to an RDF file

Content Type Negotiation for RDF

- Given the URI of a non-information resource...
  - A semantic web applications wants RDF data, as discussed
  - A regular WWW browser wants HTML, human readable
- This can be achieved using HTTP content type negotiation!
- Semantic web client:
  - Requests RDF, e.g. `Accept: application/rdf+xml, text/turtle`
  - Server uses e.g. 303 redirection to an RDF file
- HTML web client:

Content Type Negotiation for RDF

- Given the URI of a non-information resource...
  - A semantic web applications wants RDF data, as discussed
  - A regular WWW browser wants HTML, human readable
- This can be achieved using HTTP content type negotiation!
- Semantic web client:
  - Requests RDF, e.g. `Accept:  application/rdf+xml, text/turtle`
  - Server uses e.g. 303 redirection to an RDF file
- HTML web client:
  - Requests text, e.g. `Accept:  text/html, text/plain`

Content Type Negotiation for RDF

- Given the URI of a non-information resource. . .
  - A semantic web applications wants RDF data, as discussed
  - A regular WWW browser wants HTML, human readable
- This can be achieved using HTTP content type negotiation!
- Semantic web client:
  - Requests RDF, e.g. `Accept:  application/rdf+xml, text/turtle`
  - Server uses e.g. 303 redirection to an RDF file
- HTML web client:
  - Requests text, e.g. `Accept:  text/html, text/plain`
  - Server uses e.g. 303 redirection to an HTML file

Content Type Negotiation for RDF

- Given the URI of a non-information resource. . .
  - A semantic web applications wants RDF data, as discussed
  - A regular WWW browser wants HTML, human readable
- This can be achieved using HTTP content type negotiation!
- Semantic web client:
  - Requests RDF, e.g. `Accept: application/rdf+xml, text/turtle`
  - Server uses e.g. 303 redirection to an RDF file
- HTML web client:
  - Requests text, e.g. `Accept: text/html, text/plain`
  - Server uses e.g. 303 redirection to an HTML file
- Also possible with hash namespaces, see `http://www.w3.org/TR/swbp-vocab-pub/`

Example: dbpedia.org

- Requesting the URI `http://dbpedia.org/resource/Oslo`

Example: dbpedia.org

- Requesting the URI `http://dbpedia.org/resource/Oslo`
- From an HTML web browser:

Example: dbpedia.org

- Requesting the URI `http://dbpedia.org/resource/Oslo`
- From an HTML web browser:
    - Sends `Accept: text/html` in request

Example: dbpedia.org

- Requesting the URI `http://dbpedia.org/resource/Oslo`
- From an HTML web browser:
  - Sends `Accept: text/html` in request
  - Server returns:

```
HTTP/1.1 303 See Other
Location: http://dbpedia.org/page/Oslo
```

Example: dbpedia.org

- Requesting the URI `http://dbpedia.org/resource/Oslo`
- From an HTML web browser:
  - Sends `Accept: text/html` in request
  - Server returns:

    ```
    HTTP/1.1 303 See Other
    Location: http://dbpedia.org/page/Oslo
    ```

  - Client requests `http://dbpedia.org/page/Oslo`

Example: dbpedia.org

- Requesting the URI `http://dbpedia.org/resource/Oslo`
- From an HTML web browser:
  - Sends `Accept: text/html` in request
  - Server returns:

    ```
    HTTP/1.1 303 See Other
    Location: http://dbpedia.org/page/Oslo
    ```

  - Client requests `http://dbpedia.org/page/Oslo`
  - Server sends HTML document:

    ```
    HTTP/1.1 200 OK
    Content-Type: text/html
    ```

Example: dbpedia.org (cont.)

- Requesting the URI `http://dbpedia.org/resource/Oslo`

Example: dbpedia.org (cont.)

- Requesting the URI `http://dbpedia.org/resource/Oslo`
- From a semantic web browser:

Example: dbpedia.org (cont.)

- Requesting the URI `http://dbpedia.org/resource/Oslo`
- From a semantic web browser:
  - Sends `Accept:  application/rdf+xml` in request

Example: dbpedia.org (cont.)

- Requesting the URI `http://dbpedia.org/resource/Oslo`
- From a semantic web browser:
    - Sends `Accept: application/rdf+xml` in request
    - Server returns:

```
HTTP/1.1 303 See Other
Location:  http://dbpedia.org/data/Oslo.xml
```

Example: dbpedia.org (cont.)

- Requesting the URI `http://dbpedia.org/resource/Oslo`
- From a semantic web browser:
    - Sends `Accept: application/rdf+xml` in request
    - Server returns:

      ```
      HTTP/1.1 303 See Other
      Location: http://dbpedia.org/data/Oslo.xml
      ```

    - Client requests `http://dbpedia.org/data/Oslo.xml`

Example: dbpedia.org (cont.)

- Requesting the URI `http://dbpedia.org/resource/Oslo`
- From a semantic web browser:
  - Sends `Accept: application/rdf+xml` in request
  - Server returns:

    ```
    HTTP/1.1 303 See Other
    Location: http://dbpedia.org/data/Oslo.xml
    ```

  - Client requests `http://dbpedia.org/data/Oslo.xml`
  - Server sends RDF/XML document:

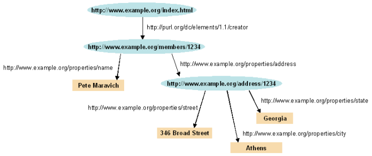
    ```
    HTTP/1.1 200 OK
    Content-Type: application/rdf+xml
    ```

Outline

The Problem

- The HTML web contains lots of human-readable information
- How can clients discover the location of corresponding machine-readable information?

$$\xrightarrow{?}$$


http://www.example.org/index.html
http://purl.org/dc/elements/1.1/creator
http://www.example.org/members/1234
http://www.example.org/properties/name
http://www.example.org/properties/address
Pete Maravich
http://www.example.org/address/1234
http://www.example.org/properties/street
http://www.example.org/properties/state
Georgia
346 Broad Street
http://www.example.org/properties/city
Athens

Embedding RDF/XML in (X)HTML

- First idea: Embed RDF/XML in HTML or XHTML:

```
<html>
  <head>
    <title>My Homepage</title>
    <rdf:RDF>
      <rdf:Description rdf:about="#me">
        <foaf:name>Martin Giese</foaf:name>
        ...
```

Embedding RDF/XML in (X)HTML

- First idea: Embed RDF/XML in HTML or XHTML:

```
<html>
  <head>
    <title>My Homepage</title>
    <rdf:RDF>
      <rdf:Description rdf:about="#me">
        <foaf:name>Martin Giese</foaf:name>
        ...
```

- Not recommended:

Embedding RDF/XML in (X)HTML

- First idea: Embed RDF/XML in HTML or XHTML:

```
<html>
  <head>
    <title>My Homepage</title>
    <rdf:RDF>
      <rdf:Description rdf:about="#me">
        <foaf:name>Martin Giese</foaf:name>
        ...
```

- Not recommended:
- Does not fit HTML or XHTML DTDs

Embedding RDF/XML in (X)HTML

- First idea: Embed RDF/XML in HTML or XHTML:

```
<html>
  <head>
    <title>My Homepage</title>
    <rdf:RDF>
      <rdf:Description rdf:about="#me">
        <foaf:name>Martin Giese</foaf:name>
        ...
```

- Not recommended:
- Does not fit HTML or XHTML DTDs
- No satisfactory solution, due to flexible RDF vocabulary

Embedding RDF/XML in (X)HTML

- First idea: Embed RDF/XML in HTML or XHTML:

```
<html>
  <head>
    <title>My Homepage</title>
    <rdf:RDF>
      <rdf:Description rdf:about="#me">
        <foaf:name>Martin Giese</foaf:name>
        ...
```

- Not recommended:
- Does not fit HTML or XHTML DTDs
- No satisfactory solution, due to flexible RDF vocabulary
- B.t.w. there *is* a `metadata` element in SVG for this!

HTML LINK elements

- `LINK` occur inside HTML `HEAD` elements

HTML LINK elements

- `LINK` occur inside HTML `HEAD` elements
- relate a document to other documents

HTML LINK elements

- `LINK` occur inside HTML `HEAD` elements
- relate a document to other documents
  - CSS style sheets

HTML LINK elements

- `LINK` occur inside HTML `HEAD` elements
- relate a document to other documents
  - CSS style sheets
  - Alternative languages

HTML LINK elements

- `LINK` occur inside HTML `HEAD` elements
- relate a document to other documents
  - CSS style sheets
  - Alternative languages
  - Next, previous, index, etc.

HTML LINK elements

- `LINK` occur inside HTML `HEAD` elements
- relate a document to other documents
    - CSS style sheets
    - Alternative languages
    - Next, previous, index, etc.
- Can contain attributes:

HTML LINK elements

- `LINK` occur inside HTML `HEAD` elements
- relate a document to other documents
  - CSS style sheets
  - Alternative languages
  - Next, previous, index, etc.
- Can contain attributes:
  - `rel` – the kind of relation

HTML LINK elements

- `LINK` occur inside HTML `HEAD` elements
- relate a document to other documents
  - CSS style sheets
  - Alternative languages
  - Next, previous, index, etc.
- Can contain attributes:
  - `rel` – the kind of relation
  - `type` – the media type of the related document

HTML LINK elements

- `LINK` occur inside HTML `HEAD` elements
- relate a document to other documents
  - CSS style sheets
  - Alternative languages
  - Next, previous, index, etc.
- Can contain attributes:
  - `rel` – the kind of relation
  - `type` – the media type of the related document
  - `href` – the URL of the other document

HTML LINK elements

- `LINK` occur inside HTML `HEAD` elements
- relate a document to other documents
  - CSS style sheets
  - Alternative languages
  - Next, previous, index, etc.
- Can contain attributes:
  - `rel` – the kind of relation
  - `type` – the media type of the related document
  - `href` – the URL of the other document
  - `title` – the title of the other document

HTML LINK elements

- `LINK` occur inside HTML `HEAD` elements
- relate a document to other documents
  - CSS style sheets
  - Alternative languages
  - Next, previous, index, etc.
- Can contain attributes:
  - `rel` – the kind of relation
  - `type` – the media type of the related document
  - `href` – the URL of the other document
  - `title` – the title of the other document
  - (and some more)

HTML LINK elements

- `LINK` occur inside HTML `HEAD` elements
- relate a document to other documents
  - CSS style sheets
  - Alternative languages
  - Next, previous, index, etc.
- Can contain attributes:
  - `rel` – the kind of relation
  - `type` – the media type of the related document
  - `href` – the URL of the other document
  - `title` – the title of the other document
  - (and some more)
- E.g. a style sheet:

```
<html>
  <head>
    <title>My Homepage</title>
    <link rel="stylesheet" type="text/css" href="style.css">
```

LINKing to RDF

- To link to an RDF representation:

```
<LINK rel="meta"
      type="application/rdf+xml"
      title="RDF/XML version"
      href="http://dbpedia.org/data/Oslo.xml">
```

LINKing to RDF

- To link to an RDF representation:

```
<LINK rel="meta"
      type="application/rdf+xml"
      title="RDF/XML version"
      href="http://dbpedia.org/data/Oslo.xml">
```

- Also: `rel="alternate"`

LINKing to RDF

- To link to an RDF representation:

```
<LINK rel="meta"
      type="application/rdf+xml"
      title="RDF/XML version"
      href="http://dbpedia.org/data/Oslo.xml">
```

- Also: `rel="alternate"`
  - Note: difference between meta-data and alternative representation

LINKing to RDF

- To link to an RDF representation:

```
<LINK rel="meta"
      type="application/rdf+xml"
      title="RDF/XML version"
      href="http://dbpedia.org/data/Oslo.xml">
```

- Also: `rel="alternate"`
  - Note: difference between meta-data and alternative representation
- Various web browser plugins exist to detect these LINKs

HTTP `Link:` response headers

- Non-standardized proposal, originally by Berners-Lee, 1992

HTTP `Link:` response headers

- Non-standardized proposal, originally by Berners-Lee, 1992
- Generated by a few servers, recognized by a few clients

HTTP `Link:` response headers

- Non-standardized proposal, originally by Berners-Lee, 1992
- Generated by a few servers, recognized by a few clients
- Same information as in LINK HTML element, but as HTTP header:

```
Link: <foaf.rdf>; rel="meta"; type="application/rdf+xml"
```

HTTP `Link:` response headers

- Non-standardized proposal, originally by Berners-Lee, 1992
- Generated by a few servers, recognized by a few clients
- Same information as in LINK HTML element, but as HTTP header:

  ```
  Link:  <foaf.rdf>; rel="meta"; type="application/rdf+xml"
  ```
- Advantage: can be sent also with non-HTML data

Outline

Once More: Embedding RDF in (X)HTML

- Directly embedding RDF/XML in (X)HTML does not work well

Once More: Embedding RDF in (X)HTML

- Directly embedding RDF/XML in (X)HTML does not work well
- Use a different “serialization” that blends well with (X)HTML!

Once More: Embedding RDF in (X)HTML

- Directly embedding RDF/XML in (X)HTML does not work well
- Use a different "serialization" that blends well with (X)HTML!

From the RDFa specification (`http://www.w3.org/TR/rdfa-syntax/`)

The aim of RDFa is to allow a single RDF graph to be carried in various types of document mark-up.

Once More: Embedding RDF in (X)HTML

- Directly embedding RDF/XML in (X)HTML does not work well
- Use a different "serialization" that blends well with (X)HTML!

From the RDFa specification (`http://www.w3.org/TR/rdfa-syntax/`)

The aim of RDFa is to allow a single RDF graph to be carried in various types of document mark-up.

- XHTML in spec., but works with HTML and other XML

Once More: Embedding RDF in (X)HTML

- Directly embedding RDF/XML in (X)HTML does not work well
- Use a different "serialization" that blends well with (X)HTML!

From the RDFa specification (`http://www.w3.org/TR/rdfa-syntax/`)

The aim of RDFa is to allow a single RDF graph to be carried in various types of document mark-up.

- XHTML in spec., but works with HTML and other XML
- RDFa adds a *fixed* set of attributes to (X)HTML

Once More: Embedding RDF in (X)HTML

- Directly embedding RDF/XML in (X)HTML does not work well
- Use a different "serialization" that blends well with (X)HTML!

From the RDFa specification (`http://www.w3.org/TR/rdfa-syntax/`)

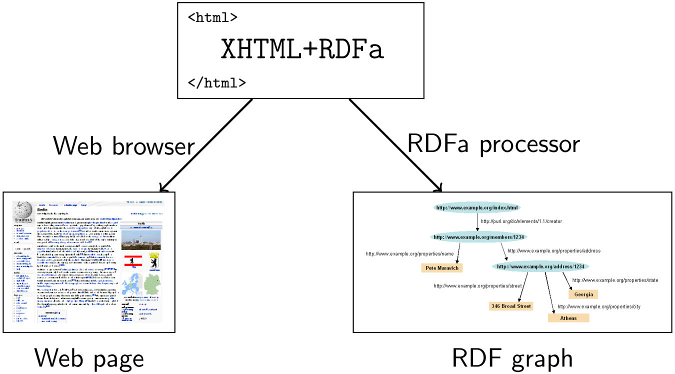
The aim of RDFa is to allow a single RDF graph to be carried in various types of document mark-up.

- XHTML in spec., but works with HTML and other XML
- RDFa adds a *fixed* set of attributes to (X)HTML
- Document type:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
    "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
```

RDFa Processing

- Web browsers ignore RDFa attributes
- RDFa processors extract a *single* RDF graph from a document


<html>
XHTML+RDFa
</html>
Web browser
RDFa processor
Berlin
http://www.example.org/index.html
http://purl.org/dc/elements/1.1/creator
http://www.example.org/members/1234
http://www.example.org/properties/name
http://www.example.org/properties/address
Pete Maravich
http://www.example.org/address/1234
http://www.example.org/properties/state
http://www.example.org/properties/street
Georgia
346 Broad Street
http://www.example.org/properties/city
Athens
Web page
RDF graph

RDFa Concepts

- RDFa adds semantic annotations to

RDFa Concepts

- RDFa adds semantic annotations to
    - hyper-links (`href`)

RDFa Concepts

- RDFa adds semantic annotations to
    - hyper-links (`href`)
    - textual content

RDFa Concepts

- RDFa adds semantic annotations to
  - hyper-links (`href`)
  - textual content
- RDFa attributes can appear in (almost) any element

RDFa Concepts

- RDFa adds semantic annotations to
  - hyper-links (`href`)
  - textual content
- RDFa attributes can appear in (almost) any element
- As the XHTML is processed, there is always a “current subject” that generated triples refer to

RDFa Concepts

- RDFa adds semantic annotations to
    - hyper-links (`href`)
    - textual content
- RDFa attributes can appear in (almost) any element
- As the XHTML is processed, there is always a “current subject” that generated triples refer to
- The current subject starts as the base URI of the document, but can change on the way

Reminder: (X)HTML Meta and Link

- Links and metadata in HTML header:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Page 507</title>
    <meta name="author" content="Sigrid Undset" />
    <link rel="prev" href="page506.html" />
    <link rel="next" href="page508.html" />
  </head>
  <body>...</body>
</html>
```

Reminder: (X)HTML Meta and Link

- Links and metadata in HTML header:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Page 507</title>
    <meta name="author" content="Sigrid Undset" />
    <link rel="prev" href="page506.html" />
    <link rel="next" href="page508.html" />
  </head>
  <body>...</body>
</html>
```

- Meaning of `name` and `rel` informal

Reminder: (X)HTML Meta and Link

- Links and metadata in HTML header:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Page 507</title>
    <meta name="author" content="Sigrid Undset" />
    <link rel="prev" href="page506.html" />
    <link rel="next" href="page508.html" />
  </head>
  <body>...</body>
</html>
```

- Meaning of `name` and `rel` informal
- Only a few values defined by the standard

RDFa `property` and `rel`

- “semantic” `meta` and `link` in RDFa:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
      xmlns:dc="http://purl.org/dc/elements/1.1/">
  <head>
    <title>MG’s home page</title>
    <meta property="dc:creator" content="Martin Giese" />
    <link rel="foaf:topic" href="foaf.rdf#me" />
  </head>
  <body>...</body>
</html>
```

RDFa `property` and `rel`

- "semantic" `meta` and `link` in RDFa:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
      xmlns:dc="http://purl.org/dc/elements/1.1/">
  <head>
    <title>MG's home page</title>
    <meta property="dc:creator" content="Martin Giese" />
    <link rel="foaf:topic" href="foaf.rdf#me" />
  </head>
  <body>...</body>
</html>
```

- Extracted triples: (`<>` is base URI!)

```
<> dc:creator "Martin Giese" .
<> foaf:topic <foaf.rdf#me> .
```

Attribute `rel` on `A` elements

- Any hyper-link can be given a “meaning”:

```
This document is licensed under a
<a xmlns:cc="http://creativecommons.org/ns#"
   rel="cc:license"
   href="http://creativecommons.org/licenses/by-nc-nd/3.0/">
  Creative Commons License
</a>.
```

Attribute `rel` on `A` elements

- Any hyper-link can be given a “meaning”:

```
This document is licensed under a
<a xmlns:cc="http://creativecommons.org/ns#"
   rel="cc:license"
   href="http://creativecommons.org/licenses/by-nc-nd/3.0/">
  Creative Commons License
</a>.
```

- Extracted triple:

```
  <> cc:license <http://creativecommons.org/.../3.0/> .
```

Attribute `rel` on `A` elements

- Any hyper-link can be given a “meaning”:

```
This document is licensed under a
<a xmlns:cc="http://creativecommons.org/ns#"
   rel="cc:license"
   href="http://creativecommons.org/licenses/by-nc-nd/3.0/">
  Creative Commons License
</a>.
```

- Extracted triple:

```
  <> cc:license <http://creativecommons.org/.../3.0/> .
```

- Can use `rev` instead of `rel` to swap subject and object

The property attribute

- rel is for resource objects, property for literal objects:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:dc="http://purl.org/dc/elements/1.1/">
  <head>...</head>
  <body>
    <h1 property="dc:title">Kransen</h1>
    Written in <span property="dc:created">1920</span>
  </body>
</html>
```

The `property` attribute

- `rel` is for resource objects, `property` for literal objects:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:dc="http://purl.org/dc/elements/1.1/">
  <head>...</head>
  <body>
    <h1 property="dc:title">Kransen</h1>
    Written in <span property="dc:created">1920</span>
  </body>
</html>
```

- Extracted triples:

```
  <> dc:title "Kransen" ; dc:created "1920" .
```

The property attribute

- `rel` is for resource objects, `property` for literal objects:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:dc="http://purl.org/dc/elements/1.1/">
  <head>...</head>
  <body>
    <h1 property="dc:title">Kransen</h1>
    Written in <span property="dc:created">1920</span>
  </body>
</html>
```

- Extracted triples:

```
  <> dc:title "Kransen" ; dc:created "1920" .
```

- Can also use `content` attribute together with `property`:

```
<span property="dc:created" datatype="xsd:dateTime"
      content="2007-09-16T16:00:00-05:00">
  September 16th at 4pm
</span>
```

Changing the Subject

- about changes subject of contained rel and property annotations:

```
<div about="http://.../foaf.rdf#me"
     xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <p property="foaf:name">Martin Giese</p>
  <p> Email:
    <a rel="foaf:mbox" href="mailto:mg@mail.no">
      mg@mail.no</a></p>
  <p> Phone:
    <a rel="foaf:phone" href="tel:+47-31415926">
      31 41 59 26</a></p>
</div>
```

Changing the Subject

- about changes subject of contained rel and property annotations:

```
<div about="http://.../foaf.rdf#me"
     xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <p property="foaf:name">Martin Giese</p>
  <p> Email:
    <a rel="foaf:mbox" href="mailto:mg@mail.no">
      mg@mail.no</a></p>
  <p> Phone:
    <a rel="foaf:phone" href="tel:+47-31415926">
      31 41 59 26</a></p>
</div>
```

- Extracted triples:

```
<http://.../foaf.rdf#me> foaf:name "Martin Giese" ;
                         foaf:mbox <mailto:mg@mail.no> ;
                         foaf:name <tel:+47-31415926> .
```

Types and Blank Nodes

- `typeof` adds an `rdf:type` triple

Types and Blank Nodes

- `typeof` adds an `rdf:type` triple
- Missing URIs can lead to blank nodes:

```
<div typeof="foaf:Person"
     xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <p property="foaf:name">Martin Giese</p>
  <p> Email:
    <a rel="foaf:mbox" href="mailto:mg@mail.no">
      mg@mail.no</a></p>
</div>
```

Types and Blank Nodes

- `typeof` adds an `rdf:type` triple
- Missing URIs can lead to blank nodes:

```
<div typeof="foaf:Person"
     xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <p property="foaf:name">Martin Giese</p>
  <p> Email:
    <a rel="foaf:mbox" href="mailto:mg@mail.no">
     mg@mail.no</a></p>
</div>
```

- Extracted triples:

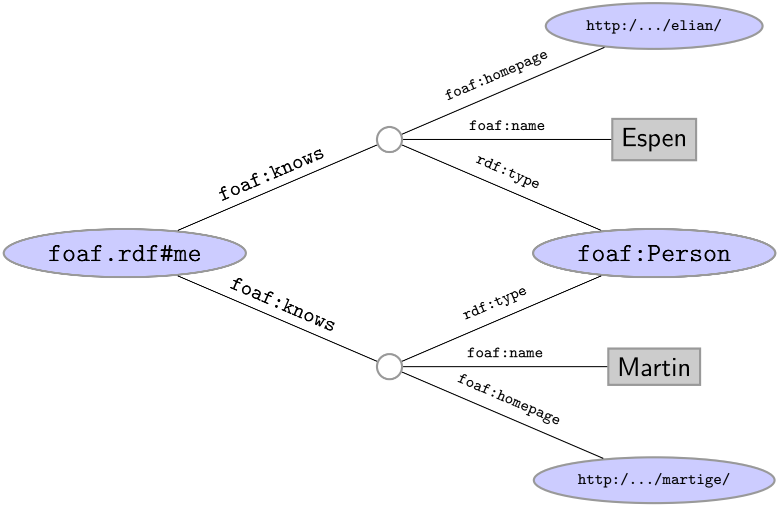
```
  [] a foaf:Person ;
     foaf:name "Martin Giese" ;
     foaf:mbox <mailto:mg@mail.no> ;
```

Know Your Friends

- Missing objects collected from contained elements (chaining):

```
<div xmlns:foaf="http://xmlns.com/foaf/0.1/"
     about="foaf.rdf#me" rel="foaf:knows">
  <ul>
    <li typeof="foaf:Person">
      <a property="foaf:name" rel="foaf:homepage"
         href="http://heim.ifi.uio.no/elian/">Espen</a>
    </li>
    <li typeof="foaf:Person">
      <a property="foaf:name" rel="foaf:homepage"
         href="http://heim.ifi.uio.no/martige/">Martin</a>
    </li>
  </ul>
</div>
```

Triples From Chaining Example



RDFa Summary

- Allows to “hide” an RDF graph in an XHTML document

RDFa Summary

- Allows to “hide” an RDF graph in an XHTML document
    - XHTML processor can ignore RDFa

RDFa Summary

- Allows to “hide” an RDF graph in an XHTML document
    - XHTML processor can ignore RDFa
    - RDFa processor can extract RDF graph

RDFa Summary

- Allows to “hide” an RDF graph in an XHTML document
    - XHTML processor can ignore RDFa
    - RDFa processor can extract RDF graph
- Treat links and text as subjects/objects and literals

RDFa Summary

- Allows to “hide” an RDF graph in an XHTML document
  - XHTML processor can ignore RDFa
  - RDFa processor can extract RDF graph
- Treat links and text as subjects/objects and literals
- Many, many more details!

RDFa Summary

- Allows to “hide” an RDF graph in an XHTML document
  - XHTML processor can ignore RDFa
  - RDFa processor can extract RDF graph
- Treat links and text as subjects/objects and literals
- Many, many more details!
  - Specification hardly less complicated than RDF/XML

RDFa Summary

- Allows to “hide” an RDF graph in an XHTML document
  - XHTML processor can ignore RDFa
  - RDFa processor can extract RDF graph
- Treat links and text as subjects/objects and literals
- Many, many more details!
  - Specification hardly less complicated than RDF/XML
  - See spec. at `http://www.w3.org/TR/rdfa-syntax/`

RDFa Summary

- Allows to "hide" an RDF graph in an XHTML document
  - XHTML processor can ignore RDFa
  - RDFa processor can extract RDF graph
- Treat links and text as subjects/objects and literals
- Many, many more details!
  - Specification hardly less complicated than RDF/XML
  - See spec. at `http://www.w3.org/TR/rdfa-syntax/`
- *Nothing* you couldn't do with a LINK and an RDF file

RDFa Summary

- Allows to “hide” an RDF graph in an XHTML document
    - XHTML processor can ignore RDFa
    - RDFa processor can extract RDF graph
- Treat links and text as subjects/objects and literals
- Many, many more details!
    - Specification hardly less complicated than RDF/XML
    - See spec. at `http://www.w3.org/TR/rdfa-syntax/`
- *Nothing* you couldn’t do with a LINK and an RDF file
- Can be convenient to have information in one place

Next Lecture

- How to publish a relational DB as RDF with D2R
- Maybe Ontology-based Data Access