# INF3580/4580 – Semantic Technologies – Spring 2017

## Lecture 15: Publishing RDF Data on the Web

Leif Harald Karlsen

22nd May 2017

Department of Informatics


University of Oslo

# Repetition

- 29th May is reserved for "Repetition"
- No fixed lecture material
- You, the students, say what you want to hear
- Let us know by Friday (26.05), so we are prepared.
- If he recieves no mail, there will be no repretition.
- So drop a mail to `martingi@ifi.uio.no`.

# Today's Plan

1. Relevant highlights from RDF lecture
2. Linked (Open) Data
   - Examples
3. Linking RDF to HTML
4. RDFa
5. Conclusion

# Outline

# RDF

- Why URIs?
  - URIs naturally have a "global" scope, unique throughout the web.
  - URLs are also addresses.
  - *"A web of data."*

# RDF

- Why URIs?
  - URIs naturally have a "global" scope, unique throughout the web.
  - URLs are also addresses.
  - *"A web of data."*
- Why triples?
  - Any information format can be transformed to triples.
  - Relationships are made explicit and are elements in their own right
  - Again, *"A web of data"*.

# RDF on the web: Where is it?

- In files:

# RDF on the web: Where is it?

- In files:
  - In some serialisation format: XML/RDF, Turtle, ...

# RDF on the web: Where is it?

- In files:
  - In some serialisation format: XML/RDF, Turtle, ...
  - Typically small RDF graphs, i.e., max. a few 100 triples, e.g.,

# RDF on the web: Where is it?

- In files:
  - In some serialisation format: XML/RDF, Turtle, ...
  - Typically small RDF graphs, i.e., max. a few 100 triples, e.g.,
    - Vocabularies: `http://xmlns.com/foaf/spec/index.rdf`.

# RDF on the web: Where is it?

- In files:
  - In some serialisation format: XML/RDF, Turtle, ...
  - Typically small RDF graphs, i.e., max. a few 100 triples, e.g.,
    - Vocabularies: `http://xmlns.com/foaf/spec/index.rdf`.
    - Tiny datasets: `http://folk.uio.no/martingi/foaf.rdf`.

# RDF on the web: Where is it?

- In files:
  - In some serialisation format: XML/RDF, Turtle, ...
  - Typically small RDF graphs, i.e., max. a few 100 triples, e.g.,
    - Vocabularies: `http://xmlns.com/foaf/spec/index.rdf`.
    - Tiny datasets: `http://folk.uio.no/martingi/foaf.rdf`.
- "Behind" *SPARQL endpoints*:

# RDF on the web: Where is it?

- In files:
  - In some serialisation format: XML/RDF, Turtle, ...
  - Typically small RDF graphs, i.e., max. a few 100 triples, e.g.,
    - Vocabularies: `http://xmlns.com/foaf/spec/index.rdf`.
    - Tiny datasets: `http://folk.uio.no/martingi/foaf.rdf`.
- "Behind" *SPARQL endpoints*:
  - Data kept in a *triple store*, i.e., a database.

# RDF on the web: Where is it?

- In files:
  - In some serialisation format: XML/RDF, Turtle, ...
  - Typically small RDF graphs, i.e., max. a few 100 triples, e.g.,
    - Vocabularies: `http://xmlns.com/foaf/spec/index.rdf`.
    - Tiny datasets: `http://folk.uio.no/martingi/foaf.rdf`.
- "Behind" *SPARQL endpoints*:
  - Data kept in a *triple store*, i.e., a database.
  - RDF is served from endpoint as results of *SPARQL queries*.

# RDF on the web: Where is it?

- In files:
  - In some serialisation format: XML/RDF, Turtle, ...
  - Typically small RDF graphs, i.e., max. a few 100 triples, e.g.,
    - Vocabularies: `http://xmlns.com/foaf/spec/index.rdf`.
    - Tiny datasets: `http://folk.uio.no/martingi/foaf.rdf`.
- "Behind" *SPARQL endpoints*:
  - Data kept in a *triple store*, i.e., a database.
  - RDF is served from endpoint as results of *SPARQL queries*.
  - Exposes data (in different formats)

# RDF on the web: Where is it?

- In files:
  - In some serialisation format: XML/RDF, Turtle, ...
  - Typically small RDF graphs, i.e., max. a few 100 triples, e.g.,
    - Vocabularies: `http://xmlns.com/foaf/spec/index.rdf`.
    - Tiny datasets: `http://folk.uio.no/martingi/foaf.rdf`.
- "Behind" *SPARQL endpoints*:
  - Data kept in a *triple store*, i.e., a database.
  - RDF is served from endpoint as results of *SPARQL queries*.
  - Exposes data (in different formats)
    - with endpoint frontends, e.g., `http://dbpedia.org/resource/Norway`, or

# RDF on the web: Where is it?

- In files:
  - In some serialisation format: XML/RDF, Turtle, ...
  - Typically small RDF graphs, i.e., max. a few 100 triples, e.g.,
    - Vocabularies: `http://xmlns.com/foaf/spec/index.rdf`.
    - Tiny datasets: `http://folk.uio.no/martingi/foaf.rdf`.
- "Behind" *SPARQL endpoints*:
  - Data kept in a *triple store*, i.e., a database.
  - RDF is served from endpoint as results of *SPARQL queries*.
  - Exposes data (in different formats)
    - with endpoint frontends, e.g., `http://dbpedia.org/resource/Norway`, or
    - by direct SPARQL query: `http://dbpedia.org/sparql`.

# Publishing RDF on the web

- If URIs of resources are dereferencable...

# Publishing RDF on the web

- If URIs of resources are dereferencable...
- ...clients can use URIs to request a description of the resource.

# Publishing RDF on the web

- If URIs of resources are dereferencable...
- ...clients can use URIs to request a description of the resource.
- Make data available in different formats. Typically:

# Publishing RDF on the web

- If URIs of resources are dereferencable...
- ...clients can use URIs to request a description of the resource.
- Make data available in different formats. Typically:
  - HTML for humans,

# Publishing RDF on the web

- If URIs of resources are dereferencable...
- ...clients can use URIs to request a description of the resource.
- Make data available in different formats. Typically:
  - HTML for humans,
  - RDF for computers.

# Publishing RDF on the web

- If URIs of resources are dereferencable...
- ...clients can use URIs to request a description of the resource.
- Make data available in different formats. Typically:
  - HTML for humans,
  - RDF for computers.
- This is called *content negotiation*.

# Publishing RDF on the web

- If URIs of resources are dereferencable...
- ...clients can use URIs to request a description of the resource.
- Make data available in different formats. Typically:
  - HTML for humans,
  - RDF for computers.
- This is called *content negotiation*.
- Endpoint frontends will do all of this for you.

# Publishing RDF on the web

- If URIs of resources are dereferencable...
- ...clients can use URIs to request a description of the resource.
- Make data available in different formats. Typically:
  - HTML for humans,
  - RDF for computers.
- This is called *content negotiation*.
- Endpoint frontends will do all of this for you.
- In this lecture, we look at some of the technicalities.

# 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://folk.uio.no/martingi/foaf#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://folk.uio.no/martingi/foaf#me` – a person
  - `tel:+47-22855050` – a telephone line

# 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://folk.uio.no/martingi/foaf#me` – a person
  - `tel:+47-22855050` – a telephone line
  - `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://folk.uio.no/martingi/foaf#me` – a person
  - `tel:+47-22855050` – a telephone line
  - `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://folk.uio.no/martingi/foaf#me` – a person
  - `tel:+47-22855050` – a telephone line
  - `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://folk.uio.no/martingi/foaf#me` – a person
  - `tel:+47-22855050` – a telephone line
  - `urn:isbn:0-395-36341-1` – a book
- Two basic types
  - "information resources": downloadable documents.
  - "non-information resources": other entities.

# The Problem and Two Solutions

- The problem:

# The Problem and Two Solutions

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

# The Problem and Two Solutions

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

# The Problem and Two Solutions

- The problem:
  - Need to locate information *about* a resource.
  - The same URI cannot denote a *downloadable* resource.
- Example: Need to differentiate between:

# The Problem and Two Solutions

- The problem:
  - Need to locate information *about* a resource.
  - The same URI cannot denote a *downloadable* resource.
- Example: Need to differentiate between:
  - A web page or RDF file about Berlin.

# The Problem and Two Solutions

- The problem:
  - Need to locate information *about* a resource.
  - The same URI cannot denote a *downloadable* resource.
- Example: Need to differentiate between:
  - A web page or RDF file about Berlin.
  - The city of Berlin.

$\neq$

# The Problem and Two Solutions

- The problem:
  - Need to locate information *about* a resource.
  - The same URI cannot denote a *downloadable* resource.
- Example: Need to differentiate between:
  - A web page or RDF file about Berlin.
  - The city of Berlin.

$\neq$

- Two W3C-recommended solutions:

# The Problem and Two Solutions

- The problem:
  - Need to locate information *about* a resource.
  - The same URI cannot denote a *downloadable* resource.
- Example: Need to differentiate between:
  - A web page or RDF file about Berlin.
  - The city of Berlin.

$\neq$

- Two W3C-recommended solutions:
  - The hash-namespace solution.

# The Problem and Two Solutions

- The problem:
  - Need to locate information *about* a resource.
  - The same URI cannot denote a *downloadable* resource.
- Example: Need to differentiate between:
  - A web page or RDF file about Berlin.
  - The city of Berlin.

$\neq$

- Two W3C-recommended solutions:
  - The hash-namespace solution.
  - The slash-namespace solution (aka HTTP 303 redirects).

# The Problem and Two Solutions

- The problem:
  - Need to locate information *about* a resource.
  - The same URI cannot denote a *downloadable* resource.
- Example: Need to differentiate between:
  - A web page or RDF file about Berlin.
  - The city of Berlin.

$\neq$

- 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://folk.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://folk.uio.no/martingi/`.
  - Connect to port 80 on `folk.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://folk.uio.no/martingi/`.
  - Connect to port 80 on `folk.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: folk.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://folk.uio.no/martingi/`.
  - Connect to port 80 on `folk.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: folk.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> ...  </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> ...  </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> ...  </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> ...  </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> ...  </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> ...  </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.

# 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://folk.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://folk.uio.no/martingi/foaf.rdf#me` - a person
  - `http://folk.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://folk.uio.no/martingi/foaf.rdf#me` - a person
  - `http://folk.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://folk.uio.no/martingi/foaf.rdf#me` - a person
  - `http://folk.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://folk.uio.no/martingi/foaf.rdf#me` - a person
  - `http://folk.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`
  - Too tight coupling of URI schema (name design) and physical storage (file name).

# 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://linkeddatabook.com/`.

# 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`.

# 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`.
  - N3: `text/n3`.

# 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`.
  - N3: `text/n3`.
- 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`.
  - N3: `text/n3`.
- 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`.
  - N3: `text/n3`.
- Client sends accepted media types in `Accept:` header:
  - `Accept: text/html, text/plain`
- 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`.
  - N3: `text/n3`.
- Client sends accepted media types in `Accept:` header:
  - `Accept: text/html, text/plain`
- 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`.
  - N3: `text/n3`.
- Client sends accepted media types in `Accept:` header:
  - `Accept: text/html, text/plain`
- 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`.
  - N3: `text/n3`.
- Client sends accepted media types in `Accept:` header:
  - `Accept: text/html, text/plain`
- 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
```

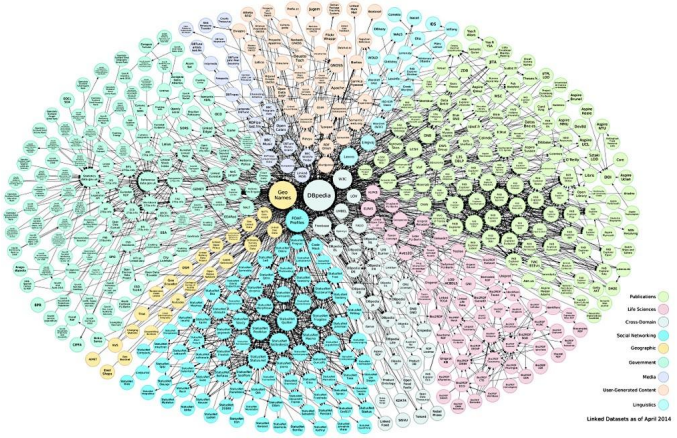
  - Client requests `http://dbpedia.org/data/Oslo.xml`
  - Server sends RDF/XML document:

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

# Examples of Linked Open Data

- http://babelnet.org
- http://en.wikipedia.org/wiki/SNOMED_CT and http://browser.ihtsdotools.org/
- http://dbpedia.org

# The Linked Open Data Cloud



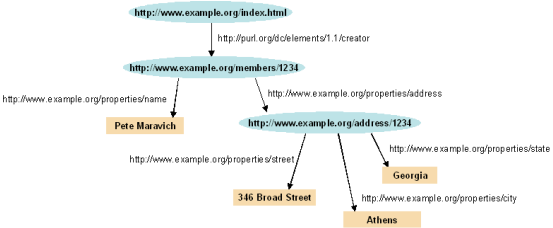

http://lod-cloud.net/

# Outline

# The Problem

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


?



# 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` may occur inside HTML `HEAD` elements

# HTML LINK elements

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

# HTML LINK elements

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

# HTML LINK elements

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

# HTML LINK elements

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

# HTML LINK elements

- `LINK` may 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` may 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` may 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` may 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` may 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` may 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` may 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="alternate"
      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="alternate"
      type="application/rdf+xml"
      title="RDF/XML version"
      href="http://dbpedia.org/data/Oslo.xml">
```

- Also: `rel="meta"`

# LINKing to RDF

- To link to an RDF representation:

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

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

# LINKing to RDF

- To link to an RDF representation:

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

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

```
      type="text/turtle; charset=UTF-8"
```

# LINKing to RDF

- To link to an RDF representation:

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

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

```
      type="text/turtle; charset=UTF-8"
```

- Various web browser plugins exist to detect these LINKs

# HTTP `Link:` response headers

- Problems with `<LINK>` elements:

# HTTP `Link:` response headers

- Problems with `<LINK>` elements:
  - Only works with HTML data, not PDF, Images, etc.

# HTTP `Link:` response headers

- Problems with `<LINK>` elements:
  - Only works with HTML data, not PDF, Images, etc.
  - Need to download HTML content and search `LINK`.

# HTTP `Link:` response headers

- Problems with `<LINK>` elements:
  - Only works with HTML data, not PDF, Images, etc.
  - Need to download HTML content and search `LINK`.
- Idea: put information in `HTTP` response header.

# HTTP `Link:` response headers

- Problems with `<LINK>` elements:
  - Only works with HTML data, not PDF, Images, etc.
  - Need to download HTML content and search `LINK`.
- Idea: put information in `HTTP` response header.
- Non-standardized proposal, originally by Berners-Lee, 1992

# HTTP `Link:` response headers

- Problems with `<LINK>` elements:
  - Only works with HTML data, not PDF, Images, etc.
  - Need to download HTML content and search `LINK`.
- Idea: put information in `HTTP` response header.
- Non-standardized proposal, originally by Berners-Lee, 1992
- Generated by a few servers, recognized by a few clients

# HTTP `Link:` response headers

- Problems with `<LINK>` elements:
  - Only works with HTML data, not PDF, Images, etc.
  - Need to download HTML content and search `LINK`.
- Idea: put information in `HTTP` response header.
- 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="alternate"; type="application/rdf+xml"
```

# HTTP `Link:` response headers

- Problems with `<LINK>` elements:
  - Only works with HTML data, not PDF, Images, etc.
  - Need to download HTML content and search `LINK`.
- Idea: put information in `HTTP` response header.
- 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="alternate"; type="application/rdf+xml"
```

- Advantages:

# HTTP `Link:` response headers

- Problems with `<LINK>` elements:
  - Only works with HTML data, not PDF, Images, etc.
  - Need to download HTML content and search `LINK`.
- Idea: put information in `HTTP` response header.
- 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="alternate"; type="application/rdf+xml"
```

- Advantages:
  - can be sent also with non-HTML data

# HTTP `Link:` response headers

- Problems with `<LINK>` elements:
  - Only works with HTML data, not PDF, Images, etc.
  - Need to download HTML content and search `LINK`.
- Idea: put information in `HTTP` response header.
- 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="alternate"; type="application/rdf+xml"
```

- Advantages:
  - can be sent also with non-HTML data
  - requires only HEAD request

# 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/`)**

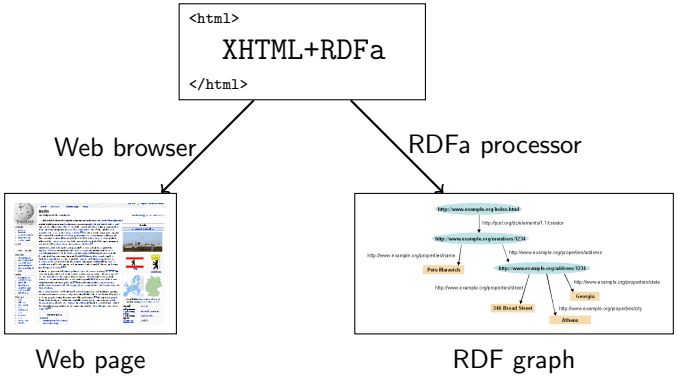
The aim of RDFa is to allow a single RDF graph to be carried in various types of document mark-up.

- 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



# 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:

```
Made by <a rev="foaf:made" href="http://.../foaf#me">me</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/> .
```

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

```
Made by <a rev="foaf:made" href="http://.../foaf#me">me</a>.
```

- Extracted triple:

```
<http://.../foaf#me> foaf:made <> .
```

# 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:phone <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:

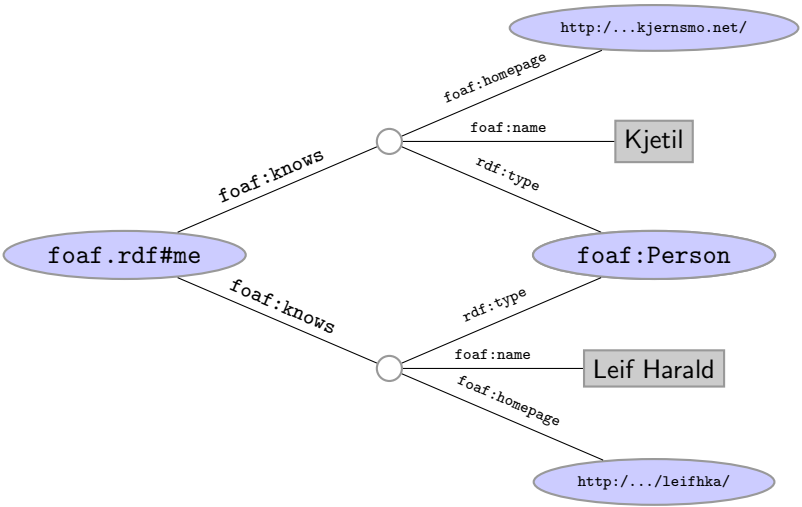
```
[] a foaf:Person ;
   foaf:name "Martin Giese" ;
   foaf:mbox <mailto:mg@mail.no> ;
```

# 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://www.kjetil.kjernsmo.net/">Kjetil</a>
    </li>
    <li typeof="foaf:Person">
      <a property="foaf:name" rel="foaf:homepage"
         href="http://heim.ifi.uio.no/leifhka/">Leif Harald</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

# 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
- Used by Google as one data format for "Rich Snippets"
  `https://developers.google.com/structured-data/`

# 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
- Used by Google as one data format for "Rich Snippets"
  `https://developers.google.com/structured-data/`
- NOTE: this lecture was about RDFa 1.0. Search the web for RDFa 1.1!

# Outline

# Topics Covered

- RDF, principles, Turtle syntax

# Topics Covered

- RDF, principles, Turtle syntax
- The Jena API for RDF

# Topics Covered

- RDF, principles, Turtle syntax
- The Jena API for RDF
- The SPARQL Query Language

# Topics Covered

- RDF, principles, Turtle syntax
- The Jena API for RDF
- The SPARQL Query Language
- Basics of the RDFS and OWL ontology languages

# Topics Covered

- RDF, principles, Turtle syntax
- The Jena API for RDF
- The SPARQL Query Language
- Basics of the RDFS and OWL ontology languages
- Basics of model semantics and reasoning

# Topics Covered

- RDF, principles, Turtle syntax
- The Jena API for RDF
- The SPARQL Query Language
- Basics of the RDFS and OWL ontology languages
- Basics of model semantics and reasoning
- Linked Open Data, RDFa

# Topics Covered

- RDF, principles, Turtle syntax
- The Jena API for RDF
- The SPARQL Query Language
- Basics of the RDFS and OWL ontology languages
- Basics of model semantics and reasoning
- Linked Open Data, RDFa
- Publishing Databases as RDF

# Topics *Not* Covered

- Rule Languages (SWRL, RIF, Jena rules, etc.)

# Topics *Not* Covered

- Rule Languages (SWRL, RIF, Jena rules, etc.)
- SW application structures

# Topics *Not* Covered

- Rule Languages (SWRL, RIF, Jena rules, etc.)
- SW application structures
- Semantic Web Services

# Topics *Not* Covered

- Rule Languages (SWRL, RIF, Jena rules, etc.)
- SW application structures
- Semantic Web Services
- Details of RDF/RDFS model semantics

# Topics *Not* Covered

- Rule Languages (SWRL, RIF, Jena rules, etc.)
- SW application structures
- Semantic Web Services
- Details of RDF/RDFS model semantics
- Some details of OWL

# Topics *Not* Covered

- Rule Languages (SWRL, RIF, Jena rules, etc.)
- SW application structures
- Semantic Web Services
- Details of RDF/RDFS model semantics
- Some details of OWL
- Details of OWL 2 profiles

# Topics *Not* Covered

- Rule Languages (SWRL, RIF, Jena rules, etc.)
- SW application structures
- Semantic Web Services
- Details of RDF/RDFS model semantics
- Some details of OWL
- Details of OWL 2 profiles
- Logical theory: Soundness, Completeness,...

# Topics *Not* Covered

- Rule Languages (SWRL, RIF, Jena rules, etc.)
- SW application structures
- Semantic Web Services
- Details of RDF/RDFS model semantics
- Some details of OWL
- Details of OWL 2 profiles
- Logical theory: Soundness, Completeness,...
  - (You ain't seen nothing yet :-)

# Topics *Not* Covered

- Rule Languages (SWRL, RIF, Jena rules, etc.)
- SW application structures
- Semantic Web Services
- Details of RDF/RDFS model semantics
- Some details of OWL
- Details of OWL 2 profiles
- Logical theory: Soundness, Completeness,...
  - (You ain't seen nothing yet :-)
- And many more!

# Help! I Can't Get Enough!

- For more information on theory:

# Help! I Can't Get Enough!

- For more information on theory:
  - Book on Foundations of SW Technologies

# Help! I Can't Get Enough!

- For more information on theory:
  - Book on Foundations of SW Technologies
  - Take a course in logic or automated reasoning

# Help! I Can't Get Enough!

- For more information on theory:
  - Book on Foundations of SW Technologies
  - Take a course in logic or automated reasoning
- For more information on practical questions:

# Help! I Can't Get Enough!

- For more information on theory:
  - Book on Foundations of SW Technologies
  - Take a course in logic or automated reasoning
- For more information on practical questions:
  - Book on Semantic Web Programming

# Help! I Can't Get Enough!

- For more information on theory:
  - Book on Foundations of SW Technologies
  - Take a course in logic or automated reasoning
- For more information on practical questions:
  - Book on Semantic Web Programming
  - Standards texts on W3C Web pages

# Help! I Can't Get Enough!

- For more information on theory:
  - Book on Foundations of SW Technologies
  - Take a course in logic or automated reasoning
- For more information on practical questions:
  - Book on Semantic Web Programming
  - Standards texts on W3C Web pages
  - Google

# Help! I Can't Get Enough!

- For more information on theory:
  - Book on Foundations of SW Technologies
  - Take a course in logic or automated reasoning
- For more information on practical questions:
  - Book on Semantic Web Programming
  - Standards texts on W3C Web pages
  - Google
- Still not enough?

# Help! I Can't Get Enough!

- For more information on theory:
  - Book on Foundations of SW Technologies
  - Take a course in logic or automated reasoning
- For more information on practical questions:
  - Book on Semantic Web Programming
  - Standards texts on W3C Web pages
  - Google
- Still not enough?
  - Contact us for possible MSc topics!


That's all Folks!