Day-8

Learned about web1.0, web2.0 and web3.0

URI vs URN vs URL

RDF using TURTLE

Did a free course on how to search on google

URI, or Uniform Resource Identifier, is the foundation for naming and identifying resources on the internet and beyond. It's a broad term encompassing various types of identifiers

Here are some examples of URIs to illustrate the different types and components:

- **Web page URL:**
- https://www.example.com/about.html
    - This URI identifies a web page on the website "example.com".
    - "https" is the scheme, indicating the Hypertext Transfer Protocol (secure) used to access the resource.
    - "www.example.com" is the authority, specifying the domain name.
    - "/about.html" is the path, indicating the specific web page file named "about.html".
- **Image URI:**
- ftp://ftp.archive.org/images/earth.jpg
    - This URI identifies an image file named "earth.jpg".
    - "ftp" is the scheme, indicating the File Transfer Protocol used to access the file.
    - "ftp.archive.org" is the authority, specifying the server hosting the image.
    - "/images/earth.jpg" is the path, indicating the location of the image file within the server's directory structure.
- **Mailto URI:**
- mailto:info@wikipedia.org
    - This URI specifies an email address to send a message to.
    - "mailto" is the scheme, indicating it's used for email.
    - "info@wikipedia.org" acts as both the authority and path in this case, defining the recipient's email address.
- **URN:**
- urn:isbn:0-302-38261-4
    - This URI identifies a book using its ISBN number.
    - "urn" is the scheme, indicating it's a Uniform Resource Name.
    - "isbn" is a specific naming scheme for books.
    - "0-302-38261-4" is the unique identifier for this particular book assigned by the ISBN system.

While the specific format can vary depending on the URI scheme, most URIs follow a general structure:

- **Scheme:** This initial part indicates the protocol used to access the resource. Common schemes include http (web pages), ftp (file transfer), mailto (email addresses), and tel (phone numbers).
- **Authority:** This section specifies the location of the resource. For web pages, it includes the domain name and sometimes a port number.
- **Path:** This part identifies the specific resource within the location. In a website URL, it would be the path to a particular file or directory.
- **Query:** This optional section provides additional information used to retrieve the resource. It's often used in web searches or forms, where it appears after a question mark.
- **Fragment:** Another optional part, the fragment identifies a specific section within a resource, like a named section of a web page. It's denoted by a hash mark (#).

Types of URIs:

As mentioned before, URIs come in two main flavors:

- **URL (Uniform Resource Locator):** This is the most familiar type of URI. It acts as both an identifier and a locator, specifying not only what the resource is but also how to access it using a specific protocol. The web addresses you type into your browser are all URLs.
- **URN (Uniform Resource Name):** This type focuses solely on identifying a resource and doesn't provide location information. Think of it as a unique and persistent name assigned to a resource, independent of where it's located. URNs are useful for resources that might change location over time, such as scholarly articles or legal documents.

URN for a book by ISBN:

urn:isbn:0-302-38261-4

- **Explanation:** This URN identifies a specific book using its International Standard Book Number (ISBN).
- **Breakdown:**
    - urn: Identifies it as a Uniform Resource Name.
    - isbn: Indicates the specific naming scheme used, which is ISBN in this case.
    - 0-302-38261-4: The unique identifier assigned to this particular book following the ISBN standard
    -
- **URN for an RFC document:**
- urn:ietf:rfc:3986

- **Explanation:** This URN identifies a document published by the Internet Engineering Task Force (IETF) with the identifier RFC 3986. RFCs are technical documents that define internet protocols and standards.
- **Breakdown:**
  - urn: Identifies it as a Uniform Resource Name.
  - ietf: Namespace indicating the IETF organization that issues this URN.
  - rfc: Further specifies the type of resource within the IETF namespace (Request for Comments document).
  - 3986: The unique identifier assigned to this specific RFC document.

Remember, URNs focus on permanent and unique identification, and don't necessarily tell you how to access the resource. You might need to search for the resource using its URN alongside a search engine or a specific database that recognizes the URN format.

A Turtle document allows writing down an RDF graph in a compact textual form.
An RDF graph is made up of triples consisting of a subject, predicate and object.

Comments may follow a # that is not part of another lexical token

Turtle, or Terse RDF Triple Language, is a textual way to write RDF triples. RDF (Resource Description Framework) is the foundation for representing information in a way that machines can understand, using a data model based on triples.

Here are some examples of writing RDF using Turtle, showcasing different scenarios:

1. Basic Information about a Book:

Code snippet

```
@prefix dc: <http://purl.org/dc/elements/1.1/>  # Dublin Core vocabulary prefix

<http://example.org/books/MobyDick>  # Subject (URI of the book)
  dc:title "Moby Dick" ;          # Predicate and Object (literal for title)
  dc:author <http://example.org/people/HermanMelville> .  # Predicate and Object (URI of author)
```

Use code with caution.

content_copy

This example describes a book titled "Moby Dick" by Herman Melville. It uses Dublin Core vocabulary (dc:title and dc:author) to represent the book's title and author.

2. Describing a Person with Multiple Properties:

Code snippet

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>  # FOAF vocabulary prefix
@prefix dcterms: <http://purl.org/dc/terms/>  # Dublin Core terms prefix

ex:Jane foaf:name "Jane Doe" ;  # Subject, predicate, and object (literal for name)
      foaf:mbox <jane.doe@example.com> ;  # Predicate and object (literal for email)
      dcterms:description "Software Engineer" .  # Predicate and object (literal for
description)
```

Use code with caution.

content_copy

Here, we describe a person named "Jane Doe" using FOAF (foaf:name, foaf:mbox) and Dublin Core Terms (dcterms:description) vocabularies. We use a predicate list to group related properties for Jane.

3. Representing Relationships between Resources:

Code snippet

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>

ex:John foaf:knows ex:Jane ;  # Subject, predicate, and object (URI of another person)
      foaf:knows ex:Alice .  # Another predicate-object pair using a different URI
```

Use code with caution.

content_copy

This example shows how to represent relationships between people using the foaf:knows predicate. John knows both Jane (identified by her URI ex:Jane) and Alice (identified by her URI ex:Alice).

4. Using Blank Nodes for Anonymous Resources:

Code snippet

```
ex:article
  dc:title "The Semantic Web" ;  # Subject, predicate, and object (literal for title)
  dc:creator [ foaf:name "John Doe" ] .  # Predicate and object (blank node with nested
statements)
```

Use code with caution.

content_copy

Here, we describe an article titled "The Semantic Web" with an anonymous creator. We use a blank node [] to hold nested statements about the creator's name ("John Doe") using the foaf:name predicate.

5. Utilizing Prefixes for Readability:

Code snippet

```
@prefix ex: <http://example.org/>  # Custom prefix for our vocabulary

ex:book1 ex:hasTitle "The Lord of the Rings" ;  # Using prefixes for subject and predicate
        ex:hasAuthor ex:Tolkien .  # Using prefixes for subject, predicate, and object (URI)
```
Use code with caution.

content_copy

This example demonstrates using prefixes to shorten long URIs in your code. We define a custom prefix ex: for our vocabulary and use it with subject (ex:book1), predicate (ex:hasTitle, ex:hasAuthor), and object (URI) to improve readability.

These examples showcase the flexibility of Turtle for representing various types of information in the Semantic Web. Remember to choose appropriate vocabularies and structures based on the data you're describing.