Exploration — URL

Introduction

We briefly discussed the structure of a URL in an earlier exploration. In this exploration, we look at URL structure in detail.

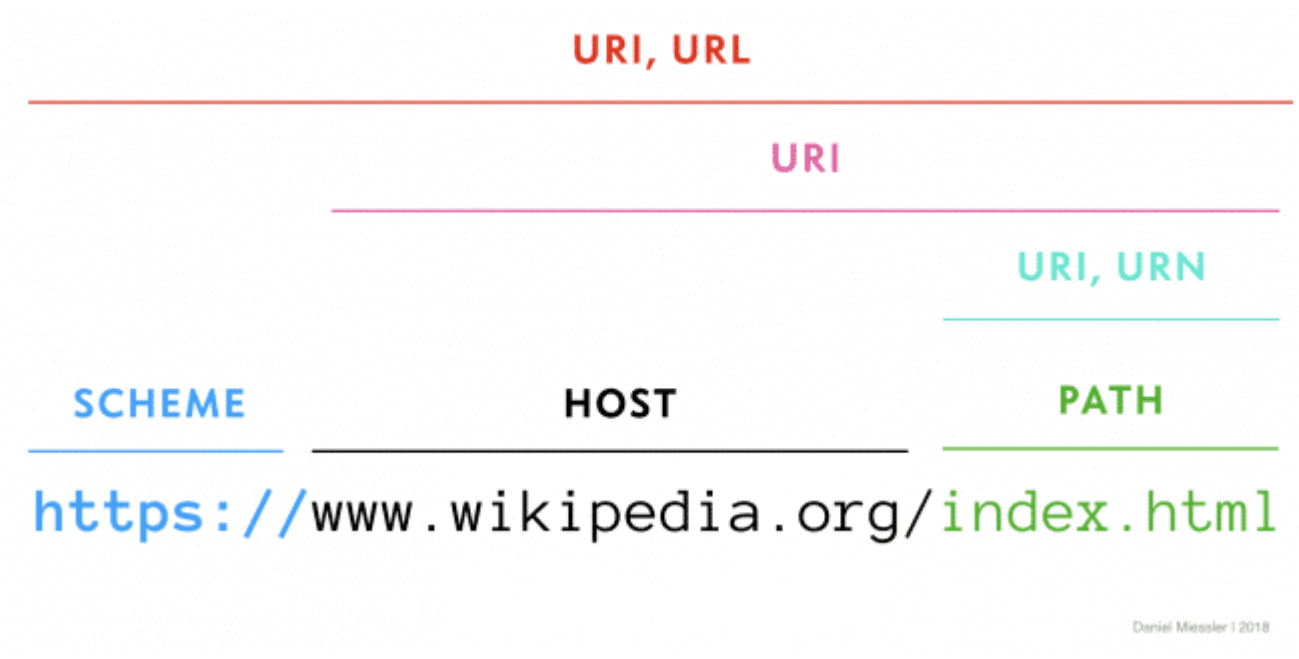
URL, URN and URI

Technically, a URL is a type of a **URI** (**Uniform Resource Identifier)**. The other type of URIs are called **Uniform Resource Names** or **URNs**. The idea is that a URN identifies a resource reliably without reference to its location, while a URL identifies an actual location of the resource. An analogy with books will help explain a URN:

- An ISBN (International Standard Book Number) is a unique identifier that is assigned to each separate edition of a published book.
- For example, the first edition of the book *Modern JavaScript for the Impatient* has the ISBN 978-0-13-650214-2. If another edition of this book is issued, a separate ISBN will be assigned to it.
- My copy of *Modern JavaScript for the Impatient* as well as your copy of this book both have the same ISBN.
- But this ISBN doesn't tell us anything about the location of an actual copy of the book.

So conceptually, a URN is similar to an ISBN. We will not be dealing with URNs in this course and for the purposes of this course, we can consider the term URI and URL to be equivalent. However, be aware that the term URI, in all caps or in a different case (e.g., URI) is used quite frequently in technologies related to the web and we will encounter this term in JavaScript libraries we use in this course.

Parts of a URL



© Meister 2018. Diagram shows relationship between scheme, host, path of a URI, URL, and URN.

A URL contains some required and some optional part. The general structure of a URL is as follows:

```
scheme://server-name:port-number/path-to-resource?param1=value1&param2=value2#link-in-resource
```

To explain the parts, we consider the following URL:

```
https://example.com:443/a/b/c.html?symbol=AMC&duration=6months#section11
```

Scheme

- In our example URL, its value is `https`.
- A URL starts with a scheme which identifies the protocol the web client must use to request the resource.
- The scheme is followed by a colon, i.e., `:`.
- Historically, the scheme for most websites was `http`. However, most websites now use `https` which is similar to `http` but encrypts the data being sent between the client and the server.
- Most browsers also know how to send requests and handle responses in other schemes.

- Other examples of common schemes include `ftp` (unencrypted) and `ftps` (encrypted) to handle file transfers and `sms` for interacting with devices that understand the SMS text messaging service.
- Another example is the scheme `file` which is used to read files from the local machine.
  - E.g., here is the URL displayed in the browser when I open a local HTML file on my machine in a browser `file:///C:/Users/nauman/OneDrive/bookmarks_10_11_20.html`

Server Name

- In our example URL, its value is `example.com`.
- This is the name of the machine where the resource is hosted and to which the request must be sent.
- The server name is preceded by two slashes, i.e., `//`.
- We have already discussed how a web client uses a DNS to map the server name to the IP address to which the request is sent.
- The terms server name, server machine, host name, host machine are used interchangeably.

Server's port number

- In our example URL, its value is `443`.
- If present, the port number preceded by a colon, i.e., `:` , which represents the term: **port**.
- Each host machine on the internet is logically divided into logical port numbers in the range 1 to 65535 (or 64K). This allows multiple servers to run on the same machine.
  - E.g., a program to handle `https` requests may be using port 443, while a different program may be using port 21 to handle `ftp` requests.
- Specific port numbers are reserved for specific purposes.
  - E.g., port `443` is reserved for `https` servers.
- We can omit the port number when a server uses the default port number.
  - E.g., since 443 is the standard port number for `https`, we can omit it from our example URL and instead use `https://example.com/a/b/c.html?symbol=AMC&duration=6months#section11`

Path to Resource

- In our example URL, its value is `/a/b/c.html`.
- This part of the URL identifies the resource on the server.
- For static files (e.g., image files), this hierarchical path usually represents the physical location of a file on the web server.
- For dynamically generated HTML, this is the path name of a program that will be executed to generate the content.
- However, it is completely up to the web server to interpret this identifier.
- Importantly for us, web servers allow developers to add **routing** information to specify how this path should be interpreted.

- As an example, when developing our web application, we may specify that the web server should use `/a/b` to identify the program that the web server should execute for this request and that the web server should pass `c.html` as an argument to this program.
- As a convention, most web servers are configured so that the request for `/` is mapped to the file at the path `/index.html`. This file is called the designated homepage on the web server.

Query Parameters

- In our example, its value is `symbol=AMC&duration=6months`.
- These are an optional list of key-value pairs separated by `&`.
- If present, the query parameters are preceded by a question mark, i.e., `?`.
- Traditionally, the query parameters have been used to provide additional information to the program to be executed for processing the request.
- As a real-world example, here is a URL at the MDN website that uses a query parameter **https://developer.mozilla.org/en-US/search?q=parts+of+a+url (https://developer.mozilla.org/en-US/search?q=parts+of+a+url)**

Link in the Resource

- In our example, its value is `section11`.
- If present, it is preceded by a hash, i.e., `#`.
- This is also called an **anchor, fragment, or hash**.
- It is a link to a location within the resource telling the browser where in the resource should it display the content.
- For our example, the browser will scroll to `section11` in the HTML document.
- When we study HTML in detail, we will see how we can specify links within an HTML document.
- As a real-world example, here is a URL at the MDN website that uses an anchor: **https://developer.mozilla.org/en-US/docs/Web/API/URL#usage_notes (https://developer.mozilla.org/en-US/docs/Web/API/URL#usage_notes)**

Exercise

Identify the different parts of the following URL.
https://ecampus.oregonstate.edu:443/soc/ecatalog/ecourselist.htm?termcode=all&subject=CS (https://ecampus.oregonstate.edu:443/soc/ecatalog/ecourselist.htm?termcode=all&subject=CS)

Does the URL still work the same if you omit `:443` from it? Explain the behavior.

Summary

In this exploration, we studied the different parts of a URL. In later explorations, we will learn how to construct as well as interpret URLs in our programs. We will also learn how to set up routing rules that allow us to map URLs to programs running on the web server, and to specific functions within those programs.

Additional Resources

Here are some references to learn more about the topics we discussed in this exploration.

- A very good explanation of URL structure is given at the MDN page **What is a URL (https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL)**
- The Wikipedia articles on **URI (https://en.wikipedia.org/wiki/Uniform_Resource_Identifier)** and **URN (https://en.wikipedia.org/wiki/Uniform_Resource_Name)** have a good discussion of these terms.