CODING NOMADS

9) HTTP & Web Services Lesson

URL Structure in Web Development

10 min to complete · By Ryan Desmond, Jared Larsen

Contents

Another critical component in understanding how HTTP functions is the structure of a Uniform Resource Locator (URL). A URL is not just a random string; it's a structured address that specifies the location of a resource somewhere on the internet, typically on a web server. Understanding its various components is crucial for your path forward in web development.

URL Structure

Generally, URLs consist of the following components:

1. The protocol being used.

2. The location of the server (aka host).
3. (Optional) Port number to access on the server.
4. The location (path) of the resource on the server.
5. Any extra information required to locate the resource.

Info: There are a few other less-common URL components that can be optionally used in specific circumstances.

If you are interested in learning more, check out Wikipedia. Rest assured this lesson covers the most important pieces to you as a software developer.

URL Protocol

The first part of a URL is the protocol. There are many protocols available, which just like HTTP are used to transmit information from client to server. When attempting to access a resource using a URL, you specify which protocol you want to use in the following format:

```
{protocol}://
```

You probably recognize this from your time on the internet. Almost all URLs you access using a browser begin with either **http://** or **https://**.

Location of the Server

Now that you have established the protocol to be used during communication, you need to specify who you want to talk to. This is done directly after the protocol, and can either be in the form of a domain (e.g. `duckduckgo.com`) or an IP address (e.g. `52.142.124.215`). Merging the protocol and server location you end up with something you see very frequently:

```
https://duckduckgo.com
```

Port on the Server

This component is often omitted from URLs, as web browsers usually default to port 80 for HTTP and 443 for HTTPS if no port is specified. However, it can be included to direct the request to a specific port, as you will do very often moving forward to access your

Spring Boot apps running on port 8080. To include a port, place it directly after the server separated by a colon:

```
http://localhost:8080
```

Location of the Resource on the Server

Servers generally store a multitude of resources, and if you were so inclined you can potentially store an *infinite* amount of resources. Therefore, you need a way to tell the server which of its many resources you are attempting to access. This is done using a URL **path**.

URL paths come directly after the domain/IP (or port, if present) and always begin with a forward slash. When you specify a path, you are telling the server the location of the resource you need. Take the following example:

```
https://codingnomads.com/course/java-programming-101
```

Here, you can see that this URL instructs the browser to use HTTPS to request information from the server located at the domain `codingnomads.com` , and that you want the resource found at `/course/java-programming-101` . It's pretty concise when you think about it.

Adding More Information

The URL components mentioned above are already very powerful when used together. However, there are two more ways that you can increase the flexibility and information-carrying capability of a URL:

1. **Path Variables**: allow you to change values (and therefore pass information) in a URL path. Using path variables is also a great way to communicate what resource you're looking for. Take the following URL:

   ```
   https://www.api.com/finduser/{userID}
   ```

   Here, `{userID}` is a path variable. It indicates to the server that you want to find the user with the ID in its place. For example:

   ```
   https://www.api.com/finduser/2034
   ```

As you can see, this is a very elegant way of passing information to the server. Here's another example:

```
https://www.api.com/findmessages/{chatID}/{userID}
```

This link would instruct the server to find messages in the chat with `{chatID}` sent by the user with `{userID}`.

2. **Query Parameters**: make it possible to add key-value pairs to your URL. This gives you a new area to add additional data to your requests. Let's see how this works by changing the two path variables from the example above to use query parameters instead. The first example would end up looking like this:

   ```
   https://www.api.com/find_user?userID={userID}
   ```

   You can see that instead of the `userID` being embedded in the path, a key-value pair is added after a `?`. This question mark indicates that the next part of the URL will be one or more query parameters. Multiple query parameters are separated by ampersands:

   ```
   https://www.api.com/find_messages?chatID={chatID}&userID={userID}
   ```

Info: The URL structure laid out above has an incredibly high character limit. You can essentially make your URLs as long as you want with as many path variables & query parameters as you want, but keep in mind that the longer the URL the harder it becomes to wrap your head around.

If you find yourself using very long URLs, it may be time to migrate some of that information to the HTTP body or headers.

Path Variables vs. Query Parameters

You may be asking yourself if there are any rules about when you should use query parameters over path variables. The short answer is no. Your decision to use a path variable over a query parameter will in no way affect your application's ability to function. That said, the common usage for path variables is to help identify a resource, while query parameters are normally used to filter the identified resources.

Learn by Doing

Location: `Various Websites`

- Visit several websites (shopping, social media, blogs, etc), and take a closer look at the URL structure.
- See if you can discern the various domains, endpoints, path variables, and query parameters!

Summary: URL Structure

Creating a URL requires a few components:

- The protocol being used.
- The location and port of the server.
- The path of the resource on the server.
- Any extra information required to locate the resource.

URLs are used to identify server resources. URLs use this general format:

`{protocol}://{domain/IP}:{port}/{path}/{more_path}/{path_variable}?{query_key}={value}&{another_key}={value}`

Learn Spring Boot

Learn more

Previous Next → Video: URL Structure

Want to go faster with dedicated 1-on-1 support? Enroll in a Bootcamp program.

Learn more

Beginner - Intermediate Courses

Java Programming

Python Programming

JavaScript Programming

Git & GitHub

SQL + Databases

Intermediate - Advanced Courses

Spring Framework

Data Science + Machine Learning

Deep Learning with Python

Django Web Development

Flask Web Development

Career Tracks

Java Engineering Career Track

Python Web Dev Career Track

Data Science / ML Career Track

Career Services

Resources

About CodingNomads

Corporate Partnerships

Contact us

Blog

Discord