CODING NOMADS

5) Build Your First Flask Apps Lesson

What are URLs and Routes?

11 min to complete · By Brandon Gigous

Contents

In this lesson, you'll take a step back from pure Flask and explore some of the basic building blocks of the web that Flask leverages to work: URLs and routes.

What is a URL?

URL. Does it stand for Unilateral Reasonable Logic? Underwater Respiratory Loophole? Unicycle Redundancy League?

When you're talking about the web, URL stands for Uniform Resource Locator, which is a fancy name for a *web address*, say, like this one:

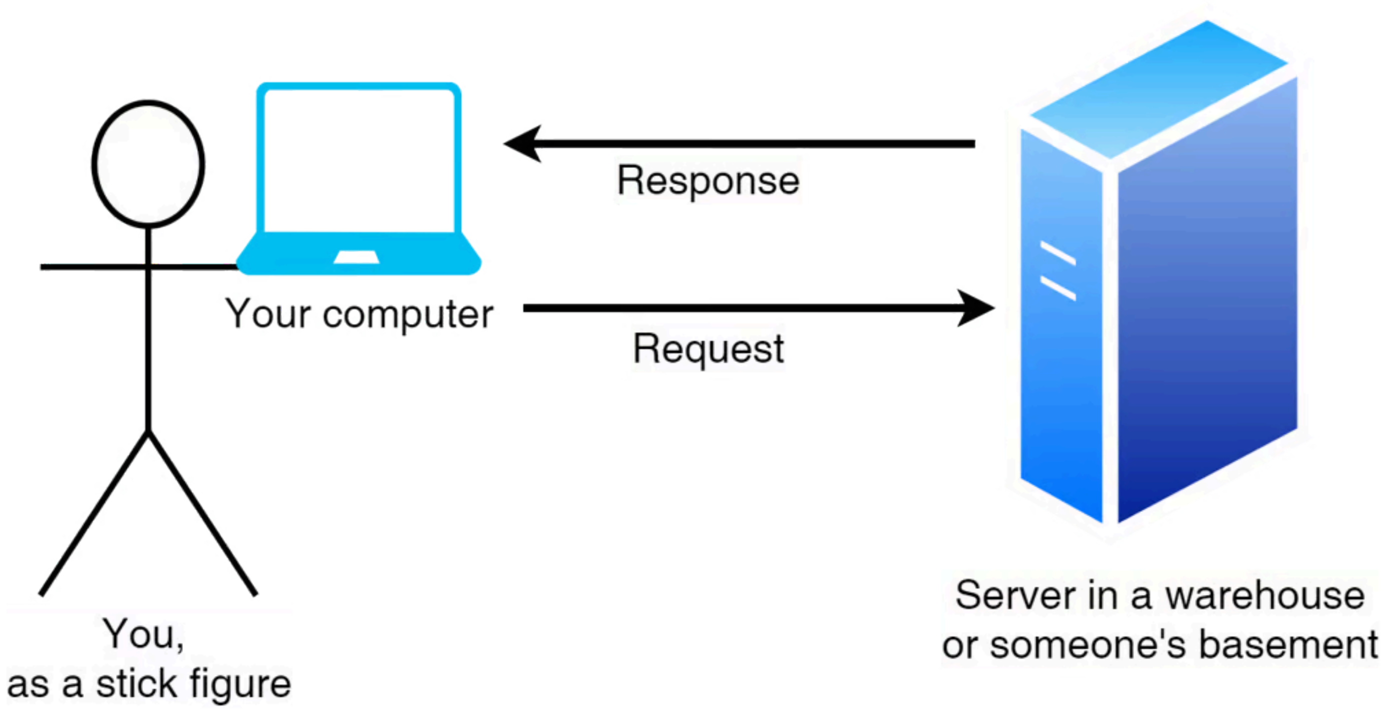
```
http://example.com/
```

That web address lets you see what *web resources* the server has that it can show you. That is, of course, assuming that you have a connection to that server.

Requests and Responses

Let's think of a simple website, our imaginary *example.com*. The website has two main parts: the "front end," which is the part of the website that the user sees, and the "back end," which is everything else behind the scenes. As a Flask web developer, you are mostly concerned with the back end because, ultimately, it's the server that must determine what the user gets to see.

When a user navigates to a webpage, like *example.com*, a **request** is sent from the user to the web server, which says, "Hey web server for *example.com*, I'd like to see what information you have for me at this URL!" The request then gets *processed* by the server, which then provides the user (their browser, likely) with a **response**. Think of it like that game you used to play as a kid with the two tin cans and a string; one of your friends would send a *request* and you'd give them back a *response* after you thought about it.



The first part of the URL, `https://` or `http://`, is the protocol and tells the server whether the request is encrypted or not, respectively. The next part, `example.com`, is the *domain name* or **host** and tells the internet which server to send the request *to*; in other words, the server that handles requests for the website `example.com`. But neither of those really have much effect on *how* a user request is responded to by a server. Wait a sec, so what's left of our web address that *does* have an effect? There's nothing left in our example!

Paths

Well, it turns out there is, and you saw it in the last page. The **path** is the part of the URL that points to the actual location of the web resource. It's like a path on your computer, where */home/you/Pictures/corn_flakes* is the album of your corn flake collection (everyone has one of those, right?). If you go to *http://example.com/about*, the path is `/about`. For our original example, *htpp://example.com/*, the path is just `/` which indicates the root path.

Paths are the determining factor in *how* a request is handled. The path tells the server what the user *really* wants. Well, not what their inner desires are, but rather, it tells the server where it needs to go to get the user the right response. More specifically, it lets the server know which *section of code* should be run to return the correct response. When visiting our imaginary website and going to the "about" page in *example.com/about*, the server finds the code for `about` and runs it.

Routes and View Functions in Flask

The same is true for a Flask web server, and every Flask application instance has an association between the path and the section of code a.k.a. the *handler* (mentioned previously) that will handle the request. Each mapping of a path to its associated handler is called a **route**. So when you use the `app.route` decorator, you're defining a mapping from a path to the **view function** (the decorated function) that will be called when a user visits that URL. If you happened to use Flask to make our example.com website, you could write the following to handle the request for the "about" page:

```
@app.route('/about')
def about_us():
    return "<p>A blurb about this website</p>"
```

Pretty cool, huh? In fact, the `app.route` decorator is a "shorthand" way of using the `app.add_url_rule()` method that Flask also provides. This function takes three arguments. In order, they are the path, the endpoint name, and the view function. The equivalent code using `add_url_rule` would be:

```
def about_us():
    return "<p>A blurb about this website</p>"
```

```
app.add_url_rule('/about', 'about_us', about_us)
```

The endpoint name is the symbolic name you can use to reference the view function from other parts of your app. Flask by default will make the endpoint name the same as the name of the view function it references, as you can tell above.

Did you also notice the `<p>` HTML tags in the response? It's no surprise since that's what websites display for their users! A bunch of HTML.

For this course, you'll mostly be working with `localhost:5000` as your website (where 5000 is the port), so an about page would be at `localhost:5000/about` in your browser.

Summary: What are URLs and Routes?

- URL stands for Uniform Resource Locator, and is a unique *web address*
- A URL has two main parts: a **front end**, which is what the user sees; and a behind-the-scenes **back end**
- When a user navigates to a URL, a request is sent from the user to the web server, the request is then processed by the web server and provides the user with a response.
- The first part of the URL, `https://` or `http://` , is the **protocol** and tells the server whether the request is encrypted or not. The second part of the URL is the **host** `example.com` , which tells the internet which server to send the request to. The third part is the **path** `/about` , which points to location of the web resource.
- Every Flask application instance has an association between a URL path and the section of code, a.k.a., the *handler*, which will handle the request. Each mapping of a path to its associated handler is called a **route**.
- when you use the `app.route` decorator, you're defining a mapping from a path to the **view function** that will be called when a user visits that URL.

```
@app.route('/about')
def about_us():
    return "<p>A blurb about this website</p>"
```

- the `app.route` decorator is a "shorthand" way of using the `app.add_url_rule()` method that Flask also provides. This function takes three arguments. In order, they are the path, the endpoint name, and the view function. The equivalent code using `add_url_rule` would be:

```
def about_us():
    return "<p>A blurb about this website</p>"

app.add_url_rule('/about', 'about_us', about_us)
```

- The endpoint name is the symbolic name you can use to reference the view function from other parts of your app.
- You'll be working with `localhost:5000` as your website (where 5000 is the port), so an about page would be at `localhost:5000/about` in your browser.

Mentorship Makes the Difference!

You don't have to learn alone. Join the CodingNomads' 1-on-1 Mentorship Program and get:

- A team of mentors and advisors dedicated to your success
- Weekly 1-on-1 screen share meetings with your personal, professional mentor
- Constant, personal, in-depth support 7 days a week via Discord
- Accountability, feedback, encouragement, and success