Previous Next

Introduction to APIs and Single Page Apps

Estimated Time: 1 hour

In this Unit, you'll use Python and Flask to build APIs which are accessible from client side code. This style of web application is called a **single-page app**, or SPA. In fact, you are currently using an SPA – the Thinkful curriculum app.

How do SPAs work?

In the previous unit the apps you made construct entire HTML pages on the server-side. These pages are then sent to the client in a single response for the client to render. When you visit a new page, a whole new HTML page is generated and rendered.

Perhaps you can start to spot some unnecessary overhead in this process. For example, when you visit a new page your header should probably stay the same. So why are we bothering to regenerate, and re-render it? SPAs try to eliminate this overhead by sending requests for new content (or changes to existing content) in the background, and then only updating the relevant part of the page. These background requests are made in JavaScript, and are commonly known as AJAX requests.

SPAs provide a number of advantages:

- SPAs load and render less content, so feel faster and more responsive
- Whole pages aren't refreshed to load new content, so the user experience is smoother
- The server has a lower workload - this means cheaper bills!

How do APIs work?

Most SPAs are built on top of an API. An API is simply a group of URLs (known as endpoints), which are used to manage an app's content. Rather than returning HTML, the server-side of your app will return some form of structured data. Notice how the role of the server is getting smaller and smaller - first you aren't returning whole HTML pages, and now we aren't returning HTML at all!

So how does this data get turned into something which you can view in your browser? This is where the front-end takes over, taking the data and running it through a templating engine similar to Jinja. The whole process looks something like this:

- You visit your app, which returns a single static HTML page, and (generally quite a lot of) JavaScript
- The front-end makes AJAX requests to the server to ask for the state of the app
- The server responds with structured data describing the state of the app
- The front-end takes this data and converts it into HTML
- The new HTML is inserted into the static HTML page
- You can now see what is going on!

A similar process takes place when you interact with an SPA - a new request is sent to the back-end describing the change you are trying to make, and the server responds with data describing the new structure.

Exercise

Try looking through the top 10 websites which you visit most frequently. See if you can identify any single-page apps amongst them. Can you identify any reasons why the sites have been chosen to work in that way? Are there any problems with the approach?

How does a Single Page Application (SPA) work?

> *You serve a single HTML page to the end user, and the user never navigates away from this page for the duration of their visit to the application. The HTML page makes AJAX calls to API endpoints on your server to generate dynamic content based on user actions.*

What are some of the advantages of SPAs?

> *Fewer round trip calls to the server, and therefore more performant apps. Backend can support multiple UXs with the same endpoints. Emulate "native app" feel.*

☆ ☆ ☆ ☆ ☆ · Report a typo or other issue

✔ Mark as completed

Previous Next