Using Server-side Rendering (SSR)

Introduction

Server-side Rendering (SSR) is one of Gatsby's rendering options and allows you to pre-render a page with data that is fetched when a user visits the page. While it is recommended to use Static Site Generation (SSG) or Deferred Static Generation (DSG) over SSR you might have use cases that require it, e.g. dynamic personalization, authenticated data, A/B testing, configurability based on location or user data. If you don't need to pre-render the page you can use client-only routes.

In this guide, you'll learn how to use `getServerData`, fetch an image from a dog API, and display it dynamically on the page.

For full documentation on all options, see the reference guide.

Prerequisites

Before you begin, you should already have:

- An existing Gatsby site. (Need help creating one? Follow the Quick Start.)

Directions

The general process for using SSR looks like this:

1. Adding `getServerData` function
2. Requesting data inside `getServerData` & displaying it

To follow this guide, create a new page at `src/pages/ssr.js`.

Step 1: Adding `getServerData` function

By adding an async function called `getServerData` to your page, you tell Gatsby to choose the SSR rendering option. Add it to your new page:

```
import * as React from "react"

const SSRPage = () => (
  <main>
```

```
    <h1>SSR Page with Dogs</h1>
  </main>
)

export default SSRPage

export async function getServerData() {} // highlight-line
```

Step 2: Requesting data inside `getServerData` & displaying it

You can execute anything you want inside the `getServerData` function, but you need to return an object containing `props`. You then can access the data as a `serverData` prop inside your page component (similarly to how page queries automatically pass in a `data` prop to page components).

Use `fetch` to pull data from the dog.ceo API:

```
// The rest of the page

export async function getServerData() {
  try {
    const res = await fetch(`https://dog.ceo/api/breeds/image/random`)

    if (!res.ok) {
      throw new Error(`Response failed`)
    }

    return {
      props: await res.json(),
    }
  } catch (error) {
    return {
      status: 500,
      headers: {},
      props: {}
    }
  }
}
```

Every time a user visits the page now the URL `https://dog.ceo/api/breeds/image/random` is requested and the response available as `serverData` to the page. The API gives back the response in the shape of `{ "message": "img-url", "status": "" }`.

Display the image using the data from the API now:

```
import * as React from "react"
```

```
const SSRPage = ({ serverData }) => ( // highlight-line
  <main>
    <h1>SSR Page with Dogs</h1>
    {/* highlight-next-line */}
    <img alt="Happy dog" src={serverData.message} />
  </main>
)

export default SSRPage

export async function getServerData() {
  try {
    const res = await fetch(`https://dog.ceo/api/breeds/image/random`)

    if (!res.ok) {
      throw new Error(`Response failed`)
    }

    return {
      props: await res.json(),
    }
  } catch (error) {
    return {
      status: 500,
      headers: {},
      props: {}
    }
  }
}
```

If you haven't already, start `gatsby develop` and visit `http://localhost:8000/ssr`. Refreshing the page should give you a new dog image on every refresh.

Additional Resources

- API Reference Guide
- Conceptual Guide