

SSR Cookies



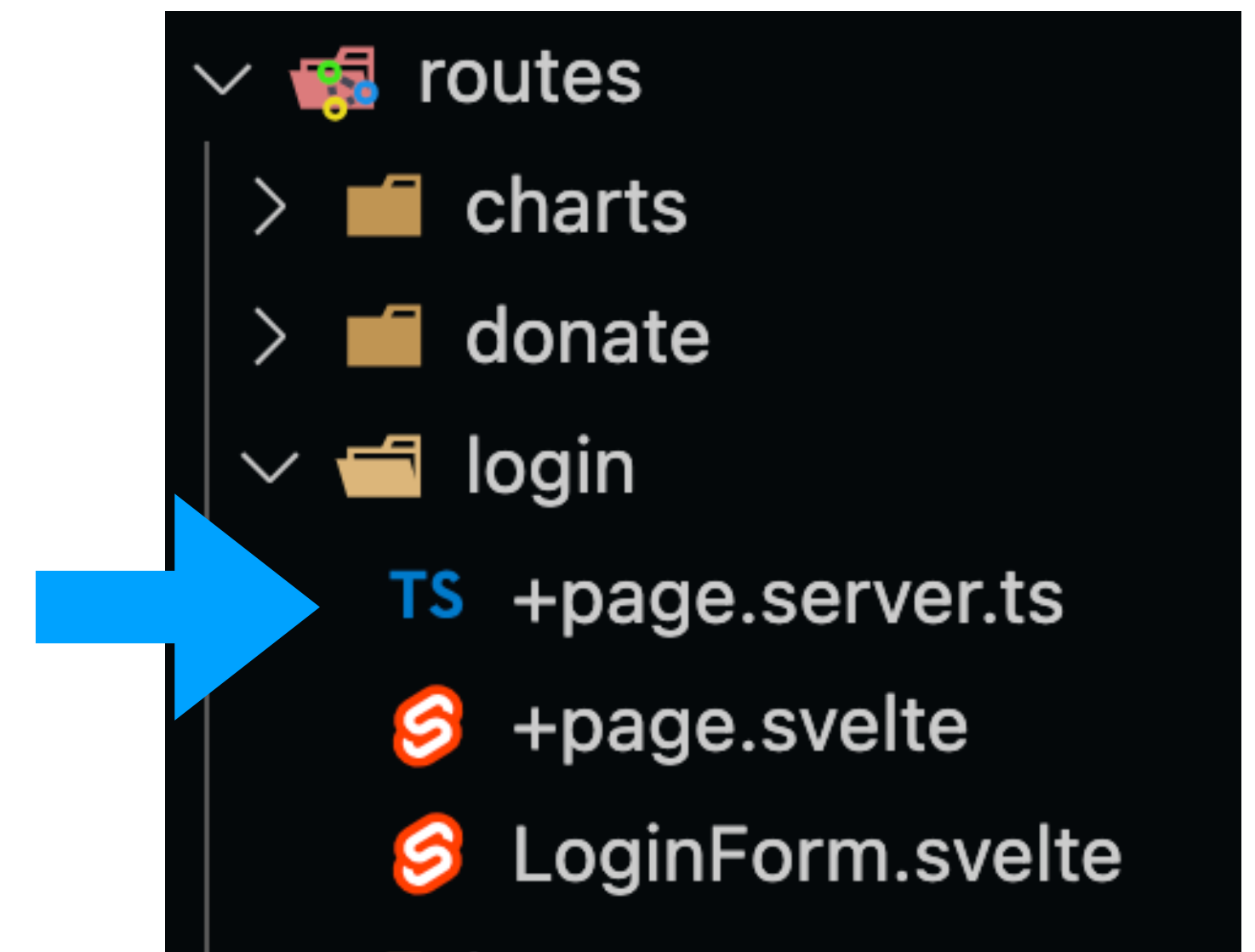
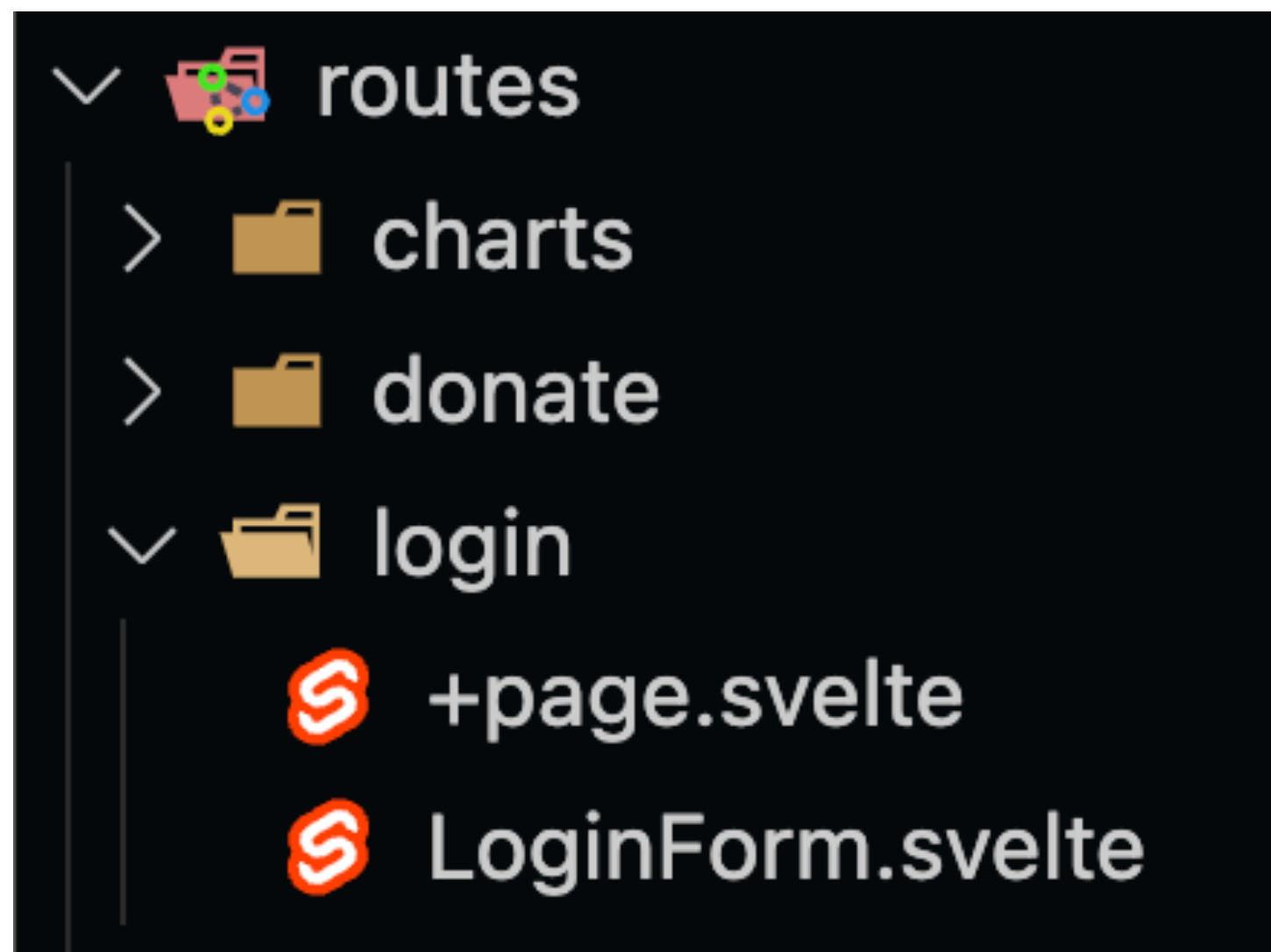
Setting & Getting Cookies
in SvelteKit

SSR - Form Handling - Authentication

- With SSR engaged, we can incorporate form handling in a similar manner to conventional node applications (Hapi)
- i.e. the Form data can be 'Posted' to the server, and processed there.
- Specifically, this processing can include conventional cookie based session creation + tracking, replacing the store/local storage approach

Server Page Component

- Place +page.server.ts module in the login route
- This will always run on the server



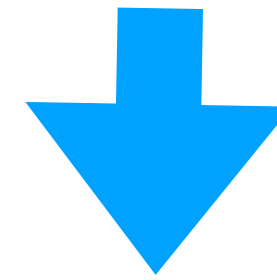
+page.server.ts

- Can export named function called **actions**
- Equivalent to Node/Hapi controller methods
- I.e. run on the server
- May be associated with route associated with a form (POST route)

```
export const actions = {  
  login: async () => {  
  
  }  
};
```

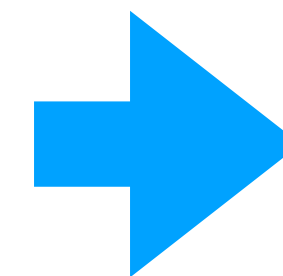
+page.svelte

- Instead of handling the login event locally (in the browser)...
- ... POST the request to the server



```
<form method="POST" action="?/login">
  <UserCredentials />
  <button class="button is-success is-fullwidth">Log In</button>
</form>
```

- This will be routed to
+page.server.ts
- Executing always on the server



```
export const actions = {
  login: async () => {
  }
};
```

login server: action

```
import { donationService } from "$lib/services/donation-service";
import { redirect } from "@sveltejs/kit";

export const actions = {
  login: async ({ request }) => {
    const form = await request.formData();
    const email = form.get("email") as string;
    const password = form.get("password") as string;
    if (email === "" || password === "") {
      throw redirect(307, "/");
    } else {
      console.log(`attempting to log in email: ${email} with password: ${password}`);
      const session = await donationService.login(email, password);

      console.log(session);
    }
  }
};
```

- Process the login form on the server
- Contact the database, verify user is valid + password matches.

login server: form data

```
import { donationService } from "$lib/services/donation-service";
import { redirect } from "@sveltejs/kit";

export const actions = {
  login: async ({ request }) => {
    const form = await request.formData();
    const email = form.get("email") as string;
    const password = form.get("password") as string;
    if (email === "" || password === "") {
      throw redirect(307, "/");
    } else {
      console.log(`attempting to log in email: ${email} with password: ${password}`);
      const session = await donationService.login(email, password);
      console.log(session);
    }
  }
};
```

- Recover the form input fields
- Invoke the login api, generating a session

login server: cookies

```
export const actions = {
  login: async ({ request, cookies }) => {
    const form = await request.formData();
    const email = form.get("email") as string;
    const password = form.get("password") as string;
    if (email === "" || password === "") {
      throw redirect(307, "/");
    } else {
      console.log(`attempting to log in email: ${email} with password: ${password}`);
      const session = await donationService.login(email, password);

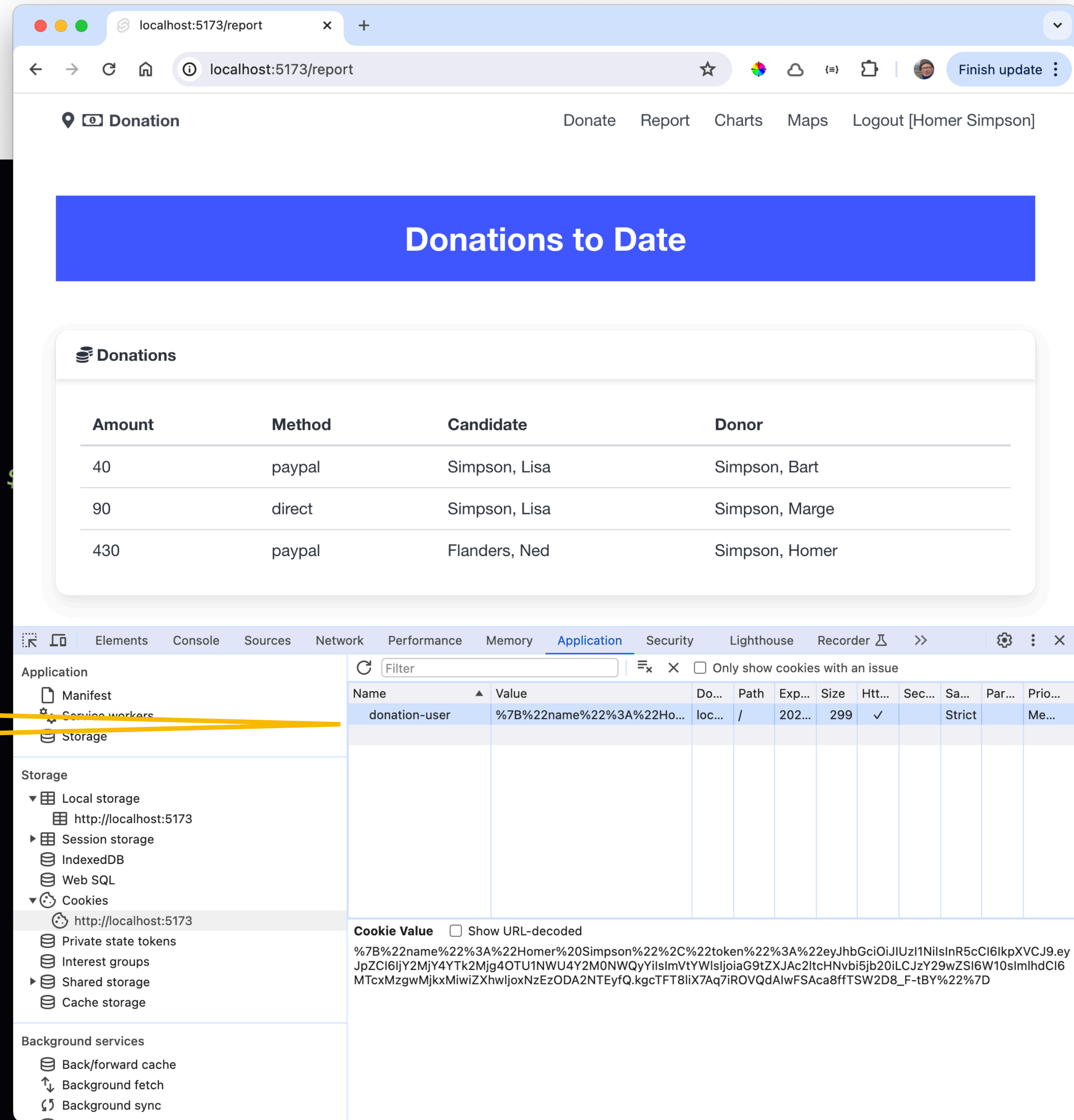
      if (session) {
        const userJson = JSON.stringify(session);
        cookies.set("donation-user", userJson, {
          path: "/",
          httpOnly: true,
          sameSite: "strict",
          secure: !dev,
          maxAge: 60 * 60 * 24 * 7 // one week
        });
        throw redirect(303, "/donate");
      } else {
        throw redirect(307, "/");
      }
    }
  }
};
```

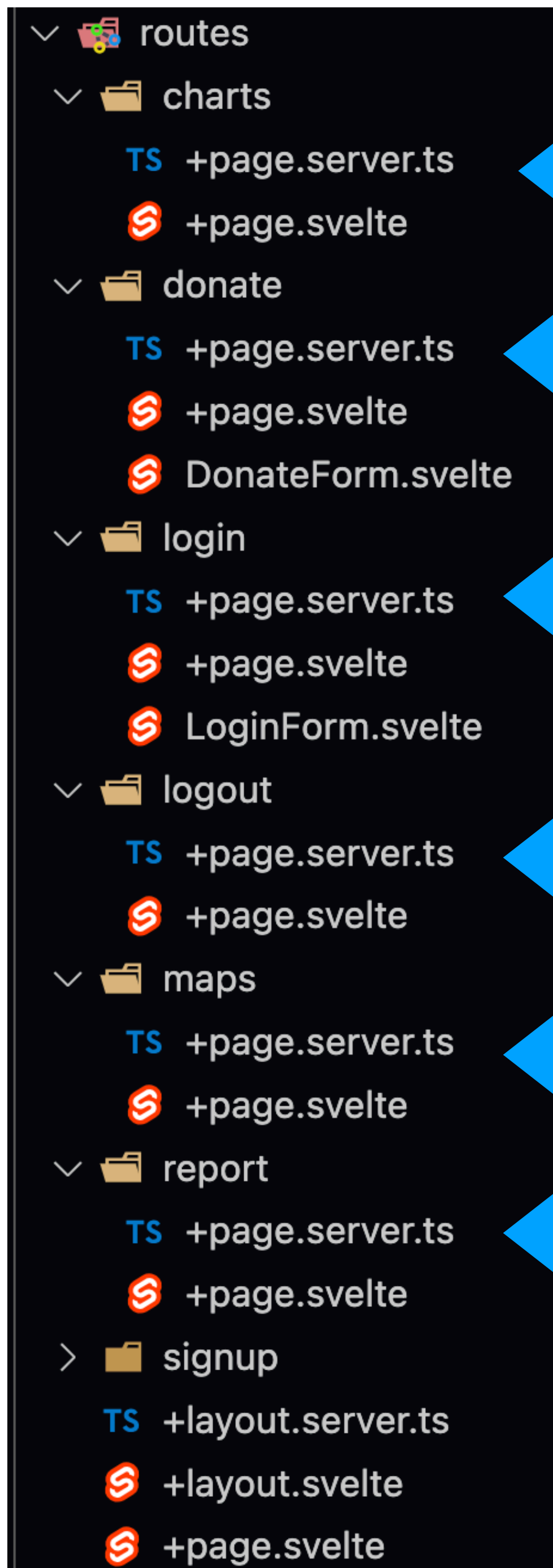
- If a session was created (i.e. name/password valid)
- Create a JSON version of the session
- Create a cookie “donation-user” with this value
- Reroute to “/donate”

login server: cookies

```
export const actions = {
  login: async ({ request, cookies }) => {
    const form = await request.formData();
    const email = form.get("email") as string;
    const password = form.get("password") as string;
    if (email === "" || password === "") {
      throw redirect(307, "/");
    } else {
      console.log(`attempting to log in email: ${email} with password: ${password}`);
      const session = await donationService.login(email, password);

      if (session) {
        const userJson = JSON.stringify(session);
        cookies.set("donation-user", userJson, {
          path: "/",
          httpOnly: true,
          sameSite: "strict",
          secure: !dev,
          maxAge: 60 * 60 * 24 * 7 // one week
        });
        throw redirect(303, "/donate");
      } else {
        throw redirect(307, "/");
      }
    }
  }
};
```





Convert all routes to SSR

- Each route has +page.server.ts:

```
import { donationService } from "$lib/services/donation-service";
import type { Session } from "$lib/types/donation-types";
import type { PageServerLoad } from "./$types";

export const load: PageServerLoad = async ({ cookies }) => {
  const cookieStr = cookies.get("donation-user") as string;
  if (cookieStr) {
    const session = JSON.parse(cookieStr) as Session;
    return {
      donations: await donationService.getDonations(session.token),
      candidates: await donationService.getCandidates(session.token)
    };
  }
};
```


Convert all routes to SSR

- Load the cookie

- Recover the Session

- Access the API

```
import { donationService } from "$lib/services/donation-service";
import type { Session } from "$lib/types/donation-types";
import type { PageServerLoad } from "./$types";

export const load: PageServerLoad = async ({ cookies }) => {
  const cookieStr = cookies.get("donation-user") as string;
  if (cookieStr) {
    const session = JSON.parse(cookieStr) as Session;
    return {
      donations: await donationService.getDonations(session.token),
      candidates: await donationService.getCandidates(session.token)
    };
  }
};
```

layout.svelte

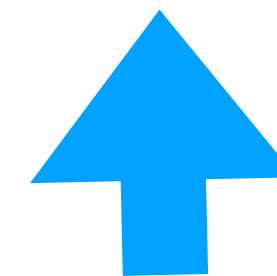
- Current version retrieves session from local storage
- Writes session to currentSession store

```
<script lang="ts">
  import { loggedInUser } from "$lib/runes.svelte";
  import { donationService } from "$lib/services/donation-service";
  import Heading from "$lib/ui/Heading.svelte";
  import Menu from "$lib/ui/Menu.svelte";
  import { onMount } from "svelte";

  onMount(async () => {
    await donationService.restoreSession();
  });
</script>

<div class="container">
  {#if loggedInUser.email}
    <Menu />
    <Heading />
  {/if}
  <slot />
</div>
```

```
async restoreSession() {
  const savedLoggedInUser = localStorage.donation;
  if (savedLoggedInUser) {
    const session = JSON.parse(savedLoggedInUser);
    loggedInUser.email = session.email;
    loggedInUser.name = session.name;
    loggedInUser.token = session.token;
    loggedInUser._id = session._id;
  }
  await this.refreshDonationInfo();
},
```



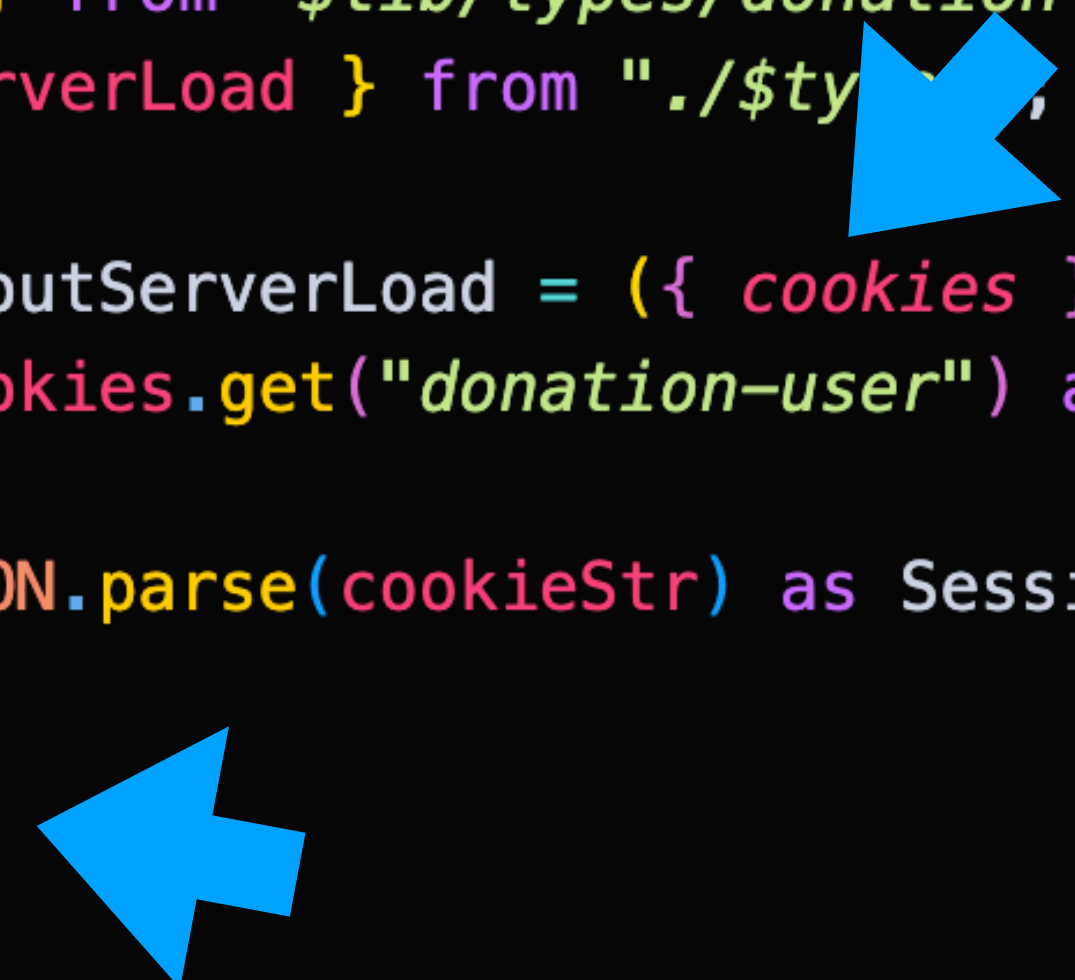
- We are no longer using local storage so this will have to change

layout.server.ts

- Always runs on the server
- Has access to cookies interface

```
import type { Session } from "$lib/types/donation-types";
import type { LayoutServerLoad } from "../types";

export const load: LayoutServerLoad = ({ cookies }) => {
  const cookieStr = cookies.get("donation-user") as string;
  if (cookieStr) {
    const session = JSON.parse(cookieStr) as Session;
    return {
      session: session
    };
  }
};
```



- Retrieve the cookie & return contents as session to the client

layout.svelte

```
<script lang="ts">
  import Heading from "$lib/ui/Heading.svelte";
  import Menu from "$lib/ui/Menu.svelte";
  import { loggedInUser } from "$lib/runes.svelte";

  export let data: any;
  if (data.session) {
    loggedInUser.email = data.session.email;
    loggedInUser.name = data.session.name;
    loggedInUser.token = data.session.token;
    loggedInUser._id = data.session._id;
  } else {
    loggedInUser.email = "";
    loggedInUser.name = "";
    loggedInUser.token = "";
    loggedInUser._id = "";
  }
</script>

<div class="container">
  {#if loggedInUser.token}
    <Menu />
    <Heading />
  {/if}
  <slot />
</div>
```


layout.server.ts

```
import type { Session } from "$lib/types/donation-types";
import type { LayoutServerLoad } from "./$types";

export const load: LayoutServerLoad = ({ cookies }) => {
  const cookieStr = cookies.get("donation-user") as string;
  if (cookieStr) {
    const session = JSON.parse(cookieStr) as Session;
    return {
      session: session
    };
  }
};
```

- Read the cookie in a session object in the server
- Pass to the client side, and write the session into the loggedInUser rune

- Recompute local state using donations/candidates retrieved from server



```
<script lang="ts">
  import { curentDataSets } from "$lib/runes.svelte";
  // @ts-ignore
  import Chart from "svelte-frappe-charts";
  import Card from "$lib/ui/Card.svelte";
  import type { PageProps } from "$types";

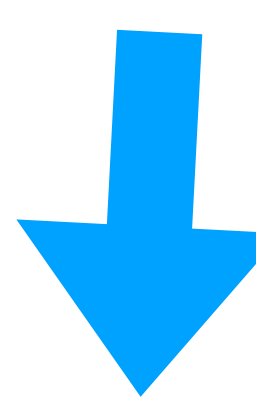
  import { refreshDonationState } from "$lib/services/donation-utils";

  let { data }: PageProps = $props();
  refreshDonationState(data.donations, data.candidates);
</script>

<div class="columns">
  <div class="column">
    <Card title="Donations By Method">
      <Chart data={curentDataSets.donationsByMethod} type="bar" />
    </Card>
  </div>
  <div class="column has-text-centered">
    <Card title="Donations By Method">
      <Chart data={curentDataSets.donationsByCandidate} type="pie" />
    </Card>
  </div>
</div>
```

routes/charts/+page.svelte

- Recover the session
- Request the donations/candidates
- Return to +page.svelte



```
import { donationService } from "$lib/services/donation-service";
import type { Session } from "$lib/types/donation-types";
import type { PageServerLoad } from ".$types";

export const load: PageServerLoad = async ({ cookies }) => {
  const cookieStr = cookies.get("donation-user") as string;
  if (cookieStr) {
    const session = JSON.parse(cookieStr) as Session;
    return {
      donations: await donationService.getDonations(session.token),
      candidates: await donationService.getCandidates(session.token)
    };
  }
};
```

15 routes/charts/+page.server.ts

+layout.server.ts

- Recover the session
- Pass session to the view

```
import type { Session } from "$lib/types/donation-types";
import type { LayoutServerLoad } from "./$types";

export const load: LayoutServerLoad = ({ cookies }) => {
  const cookieStr = cookies.get("donation-user") as string;
  if (cookieStr) {
    const session = JSON.parse(cookieStr) as Session;
    return {
      session: session
    };
  }
};
```

- Recover the session from the "parent"
- The parent (on the server) is the layout.server.ts

```
import { donationService } from "$lib/services/donation-service";
import type { PageServerLoad } from "./$types";


export const load: PageServerLoad = async ({ parent }) => {
  const { session } = await parent();
  if (session) {
    return {
      donations: await donationService.getDonations(session.token),
      candidates: await donationService.getCandidates(session.token)
    };
  }
};
```


+layout.server.ts

```
import type { Session } from "$lib/types/donation-types";
import type { LayoutServerLoad } from "./$types";

export const load: LayoutServerLoad = ({ cookies }) => {
  const cookieStr = cookies.get("donation-user") as string;
  if (cookieStr) {
    const session = JSON.parse(cookieStr) as Session;
    return {
      session: session
    };
  }
};
```

- +page.server no longer responsible for reading cookie
- This is carried out in +layout.server.ts



```
import { donationService } from "$lib/services/donation-service";
import type { PageServerLoad } from "./$types";

export const load: PageServerLoad = async ({ parent }) => {
  const { session } = await parent();
  if (session) {
    return {
      donations: await donationService.getDonations(session.token),
      candidates: await donationService.getCandidates(session.token)
    };
  }
};
```

SSR Cookies



Setting & Getting Cookies
in SvelteKit