

Svelte 5 + Hono

Part 3: Auth page (login,register,logout)



www.youtube.com/@GetterSethya

Create client context

```
export class Client {  
  static readonly key = Symbol('client');  
  static readonly value = {  
    user: hc<typeof userController>('/api/user'),  
    authentication: hc<typeof authController>('/api/auth')  
  };  
  
  static setCtx() {  
    return setContext(Client.key, writable(Client.value));  
  }  
  
  static getCtx() {  
    return getContext<ReturnType<typeof this.setCtx>>(Client.key);  
  }  
}
```

Update vite.config.ts

```
export default defineConfig({
  plugins: [sveltekit()],
  server: {
    port: 1437,
    proxy: {
      '/api': {
        target: PROD ? '/' : 'http://localhost:3000/',
        changeOrigin: true
      },
    },
  },
});
```

Set client context in root layout

```
<script lang="ts">
  import { Client } from '@context/client';
  import '../app.css';
  let { children } = $props();

  Client.setCtx();
</script>

{@render children()}
```

Setup auth route on client

```
(auth)
  login
    +page.svelte
    +page.ts
    register-form.svelte
  register
    +page.svelte
    +page.ts
    login-form.svelte
```

/register/+page.ts

```
export const load: PageLoad = async () => {  
  return {  
    form: await superValidate(zod(registerValidator))  
  };  
};
```

```
npx shadcn-svelte@next add form
```

/register/register-form.svelte

```
<script lang="ts">
  type RegisterFormProps = {
    data: SuperValidated<Infer<typeof registerValidator>>;
  };

  let { data }: RegisterFormProps = $props();
  const client = Client.getCtx();

  const form = superForm(data, {
    validationMethod: 'auto',
    validators: zodClient(registerValidator),
    SPA: true,
    onUpdate: async ({ form: fd }) => {
      if (fd.valid) {
        // do register
      }
    }
  });

  const { form: formData, enhance } = form;
</script>
```

/register/register-form.svelte

```
<form method="post" use:enhance>
  <Form.Field {form} name="email">
    <Form.Control>
      {#snippet children({ props })}
        <Form.Label>Email</Form.Label>
        <Input {...props} bind:value={$formData.email} />
      {/snippet}
    </Form.Control>
    <Form.FieldErrors />
  </Form.Field>
  <Form.Button class="w-full">Register</Form.Button>
</form>
```


/register/+page.svelte

```
<script lang="ts">
  import RegisterForm from './register-form.svelte';

  let { data } = $props();
</script>

<div class="flex min-h-screen items-center justify-center">
  <RegisterForm data={data.form} />
</div>
```

Svelte tanstack query

```
npm i @tanstack/svelte-query
```

Setup query provider in root layout

```
<script lang="ts">
  import { Client } from '@context/client';
  import '../app.css';
  import { QueryClient, QueryClientProvider } from '@tanstack/svelte-query';
  let { children } = $props();

  const queryClient = new QueryClient();
  Client.setCtx();
</script>

<QueryClientProvider client={queryClient}>
  {@render children()}
</QueryClientProvider>
```

Client response error

```
export class ResponseError<Result extends unknown> extends Error {  
  status: number;  
  results: Result;  
  
  constructor(status: number, results: Result, message: string) {  
    super(message);  
    this.results = results;  
    this.status = status;  
  }  
}
```

Install sonner

```
npx shadcn-svelte@next add sonner
```

```
<script lang="ts">
  import { Client } from '@context/client';
  import '../app.css';
  import { QueryClient, QueryClientProvider } from '@tanstack/svelte-query';
  import { Toaster } from '@components/ui/sonner';
  let { children } = $props();

  const queryClient = new QueryClient();
  Client.setCtx();
</script>

<QueryClientProvider client={queryClient}>
  <Toaster richColors />
  {@render children()}
</QueryClientProvider>
```

Register mutation

```
const registerMutation = createMutation({
  mutationFn: async (data: z.infer<typeof registerValidator>) => {
    const response = await $client.authentication.register.$post({
      form: data
    });

    const resData = await response.json();

    if (response.status !== 201)
      throw new ResponseError(
        response.status,
        null,
        resData.message ? resData.message : 'Register failed'
      );

    if (!resData.result)
      throw new ResponseError(
        response.status,
        resData.result,
        resData.message ? resData.message : 'Register failed'
      );

    return resData.result;
  },
  onSuccess: () => {
    toast.success('Register success');
    goto('/login');
  },
  onError: (error) => {
    if (error instanceof ResponseError) {
      toast.error(error.message);
      return;
    }

    toast.error('Something went wrong');
  }
});
```

Register mutation

```
onUpdate: async ({ form: fd }) => {  
  if (fd.valid) {  
    $registerMutation.mutate(fd.data);  
  }  
}
```

Do the same for login page

Create protected group & move index to protected

```
routes
  (auth)
  (protected)
    +layout.svelte
    +page.svelte
  +layout.svelte
  +layout.ts
```


Protected layout load

```
export const load: LayoutLoad = async () => {
  const accessToken = localStorage.getItem(JWT.ACCESS_TOKEN);
  if (!accessToken) {
    goto('/login');
    return;
  }
  const currentUser = await Client.value.user.current_user.$get(undefined, {
    init: { headers: { Authorization: accessToken } }
  });

  const resData = await currentUser.json();
  if (!resData || !resData.result) {
    return;
  }

  return {
    user: resData.result
  };
};
```

Local user context

```
export class LocalUser {  
  static readonly key = Symbol('local-user');  
  
  static setCtx(data: UserEntity | undefined) {  
    return setContext(LocalUser.key, writable(data));  
  }  
  
  static getCtx() {  
    return getContext<ReturnType<typeof this.setCtx>>(LocalUser.key);  
  }  
}
```

Protected layout.svelte

```
<script lang="ts">
  import { LocalUser } from '@context/localUser';

  let { data, children } = $props();

  LocalUser.setCtx(data.user);
</script>

{@render children()}
```

Protected +page.svelte

```
<script lang="ts">
  import Button from '@components/ui/button/button.svelte';
  import { LocalUser } from '@context/localUser';

  const user = LocalUser.getCtx();
</script>

<pre class="text-foreground">
  {JSON.stringify($user, null, 2)}
</pre>
<h1>Welcome to SvelteKit</h1>
<p>Visit <a href="https://svelte.dev/docs/kit">svelte.dev/docs/kit</a> to read the
documentation</p>
<Button>button</Button>
```

Logout mutation

```
const logoutMutation = createMutation({
  mutationFn: async () => {
    const response = await $client.authentication.logout
      .$delete(undefined, {
        init: { headers: { Authorization:
localStorage.getItem(JWT.ACCESS_TOKEN)! } }
      })
    .then((r) => {
      localStorage.removeItem(JWT.ACCESS_TOKEN);
      localStorage.removeItem(JWT.REFRESH_TOKEN);
      return r;
    });

    return response.json();
  },
  onSuccess: () => {
    goto('/login');
  }
});
```

(auth) +layout.ts

```
export const load: LayoutLoad = async () => {  
  const accessToken = localStorage.getItem(JWT.ACCESS_TOKEN);  
  if (accessToken) goto('/');  
};
```