

Svelte Kit

...

Eko Kurniawan Khannedy

Eko Kurniawan Khannedy

- Technical architect at one of the biggest ecommerce company in Indonesia
- 14+ years experiences
- www.programmerzamannow.com
- youtube.com/c/ProgrammerZamanNow



Eko Kurniawan Khannedy

- Telegram : [@khannedy](#)
- Linkedin : <https://www.linkedin.com/company/programmer-zaman-now/>
- Facebook : fb.com/ProgrammerZamanNow
- Instagram : instagram.com/programmerzamannow
- Youtube : youtube.com/c/ProgrammerZamanNow
- Telegram Channel : t.me/ProgrammerZamanNow
- Tiktok : <https://tiktok.com/@programmerzamannow>
- Email : echo.khannedy@gmail.com

Sebelum Belajar

- Kelas HTML, CSS dan JavaScript
- Kelas NodeJS
- Kelas Vite
- Kelas Svelte Dasar

Pengenalan

Pengenalan Svelte Kit

- Sebelumnya kita sudah belajar tentang Svelte, yaitu adalah framework untuk membuat Component
- Namun, pada kenyataannya, saat membuat aplikasi web, tidak hanya diperlukan pembuatan Component, masih banyak hal lain yang diperlukan
- Svelte Kit adalah framework untuk membuat aplikasi Svelte agar production-ready

Fitur Svelte Kit

- Svelte Kit menyelesaikan banyak masalah ketika kita membuat web, seperti :
- Routing
- Server-side render
- Data fetching
- Prerender
- Single-page application
- Library packaging
- Deploying application
- Dan lain-lain

Struktur Project

- Svelte Kit juga memiliki struktur project standard, sehingga tiap programmer tidak perlu membuat struktur project sendiri-sendiri
- <https://svelte.dev/docs/kit/project-structure>

Tipe Project

- Svelte Kit mendukung banyak tipe Project, sehingga bisa kita sesuaikan dengan kebutuhan saat kita membuat aplikasi
- <https://svelte.dev/docs/kit/project-types>

Web Standard

- Svelte Kit menggunakan Web Standard, sehingga kita yang sudah terbiasa dengan Web Standard di JavaScript tidak perlu lagi belajar hal baru
- Seperti Request, Response, URL, dan lain-lain. Menggunakan Web Standard
- <https://svelte.dev/docs/kit/web-standards>

Membuat Project

Membuat Project

```
npx sv create belajar-svelte-kit
```

Routing

Routing

- Svelte Kit menggunakan filesystem-based router. Artinya pembuatan routing pada URL Path akan dipetakan dalam filesystem
- Misal, jika kita ingin membuat URL path /about
- Maka kita perlu membuat file `src/routes/about/+page.svelte`
- Hal ini akan memudahkan, karena kita tidak perlu lagi melakukan pemetaan secara manual, cukup simpan file Svelte pada folder sesuai dengan URL path yang kita inginkan

Tugas

- Buat komponen untuk route /counter

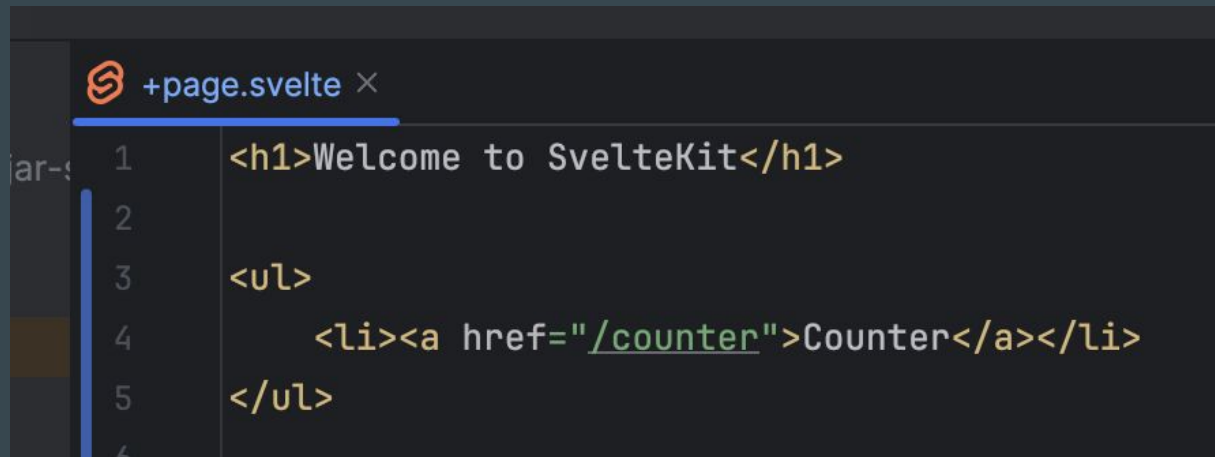
Kode : src/routes/counter/+page.svelte

```
+page.svelte x
1 <script>
2   let counter = $state(0);
3
4   function increment() {
5     counter++;
6   }
7 </script>
8
9 <h1>Counter : {counter}</h1>
10
11 <button onclick={increment}>Increment</button>
12
```


Navigation

- Saat pertama kali URL path diakses via browser, maka Svelte Kit akan melakukan SSR (Server Side Render)
- Namun ketika kita melakukan navigasi perpindahan halaman, maka akan melakukan CSR (Client Side Render)
- Svelte Kit tidak menggunakan komponen khusus untuk melakukan navigasi perpindahan halaman, kita masih tetap menggunakan element `<a>`, berbeda dengan React atau Vue yang perlu menggunakan komponen seperti `<Link>`

Kode : src/routes/+page.svelte

A screenshot of a code editor with a dark theme. The editor has a tab at the top labeled with the Svelte logo, '+page.svelte', and a close button 'x'. The code is written in Svelte syntax. Line 1 contains '<h1>Welcome to SvelteKit</h1>'. Line 3 contains ''. Line 4 contains an indented 'Counter'. Line 5 contains ''. A vertical blue line on the left indicates the current cursor position is at the start of line 4. The file name 'src/routes/+page.svelte' is partially visible on the far left.

```
1  <h1>Welcome to SvelteKit</h1>
2
3  <ul>
4    <li><a href="/counter">Counter</a></li>
5  </ul>
```

Layout

Layout

- Saat membuat halaman web, kadang kita membuat halaman yang memiliki tata letak (layout) yang mirip
- Jika kita membuat komponen tiap halaman, maka itu bukan hal yang efektif
- Svelte memiliki fitur yang bisa kita gunakan untuk membuat Layout

Membuat Layout

- Untuk membuat Layout, kita bisa membuat file `+layout.svelte` pada routes
- Secara default halaman akan dibuat dalam snippet bernama `children`
- Sehingga kita bisa menggunakan `@render children()` untuk me-render halamannya

Tugas

- Buat Layout yang berisikan menu menuju halaman /counter

Kode : +layout.svelte



+layout.svelte ×

```
1  <script>
2      const {children} = $props();
3  </script>
4
5  <ul>
6      <li><a href="/">Home</a></li>
7      <li><a href="/counter">Counter</a></li>
8  </ul>
9
10  {@render children()}
11
```


Nested Layout

- Layout bisa Nested (bertingkat).
- Kadang, misal kita ingin membuat halaman yang bertingkat, misal `/profile/user`, `/profile/address`, `/profile/wallet`, dan lain-lain
- Misal, Layout untuk halaman tersebut ingin berbeda dengan Layout utama
- Maka kita bisa membuat Layout baru misal pada `/routes/profile/+layout.svelte`
- Layout pada `/routes/profile/+layout.svelte` akan ditampilkan dalam `children()` layout `/routes/+layout.svelte`

Tugas

- Buat layout baru pada `/routes/profile/+layout.svelte`
- Buat halaman `/profile/user`
- Buat halaman `/profile/address`
- Buat halaman `/profile/wallet`

Kode : profile/+layout.svelte

 +layout.svelte ×

```
1  <script>
2    const {children} = $props();
3  </script>
4
5  <ul>
6    <li><a href="/profile/user">Profile</a></li>
7    <li><a href="/profile/wallet">Wallet</a></li>
8    <li><a href="/profile/address">Address</a></li>
9  </ul>
10
11  {@render children()}|
```

Profile Pages

```

  routes
    counter
      +page.svelte
    profile
      address
        +page.svelte
      user
        +page.svelte
      wallet
        +page.svelte
        +layout.svelte
        +layout.svelte
        +page.svelte
  app.html
```

Load Data

Load Data

- Sebelum `+page.svelte` ditampilkan, kadang kita butuh mengambil data
- Kita bisa melakukan ini dengan cara membuat load function
- Load function bisa dibuat di Page atau di Layout

Page Data

- Jika kita ingin membuat load function untuk Page, kita bisa membuat file +page.js
- Load function bisa mengembalikan data, dan data tersebut bisa kita ambil di Page dengan menggunakan attribute data via `$props()`

Tugas


- Kita akan update halaman `/profile/user` untuk menampilkan informasi user dari load data

Kode : /routes/profile/user/+page.js

JS +page.js ×

```
1 export function load() {  
2     return {  
3         user: "khannedy",  
4         name: "Eko Kurniawan Khannedy",  
5         email: "eko@test.com"  
6     }  
7 }
```


Kode : /routes/profile/user/+page.svelte

 +page.svelte ×  +page.js

```
1  <script>
2      const {data} = $props();
3  </script>
4
5  <h1>User</h1>
6
7  <ul>
8      <li>User : {data.user}</li>
9      <li>Name : {data.name}</li>
10     ⚡ <li>Email : {data.email}</li>
11 </ul>
```

Layout Data

- Selain pada Page, kita juga bisa menambahkan Load Data pada Layout
- Kita bisa membuat load function pada file +layout.js
- Jika terdapat Layout Data dan Page Data, maka pada Page, kita juga bisa mengakses data dari Layout Data
- Namun perlu diperhatikan, jika attribute pada object sama, bisa jadi Layout Data akan tertimpa oleh attribute pada Page Data

Tugas

- Tambahkan di data di Layout Profile, agar semua halaman Profile memiliki data user

Kode : /routes/profile/+layout.js

JS +layout.js ×

```
1 export function load() {  
2   return {  
3     user: "khannedy"  
4   }  
5 }
```

JS +layout.js

Svelte +layout.svelte ×

```
1 <script>  
2   const {children, data} = $props();  
3 </script>  
4  
5 <ul>  
6   <li>Hello {data.user}</li>  
7   <li><a href="/profile/user">Profile</a></li>  
8   <li><a href="/profile/wallet">Wallet</a></li>  
9   <li><a href="/profile/address">Address</a></li>  
10 </ul>  
11  
12 {@render children()}]
```

Kode : Profile Page

+page.svelte x

```
1 <script>
2   const {data} = $props();
3 </script>
4   ⚡
5 <h1>Address {data.user}</h1>
```

wallet/+page.svelte x

```
1 <script>
2   const {data} = $props();
3 </script>
4   ⚡
5 <h1>Wallet {data.user}</h1>
```

Load Parameter

Load Parameter

- Load function pada Page atau Layout memiliki parameter object, dimana object tersebut memiliki banyak attribute yang berisi informasi data yang bisa kita gunakan
- <https://svelte.dev/docs/kit/@sveltejs-kit#RequestEvent>

Tugas

- Tambahkan informasi URL yang sedang diakses pada layout `/routes/+layout.svelte`

Kode : /routes/+layout.js

JS +layout.js ×  +layout.svelte

```
1 export function load({url}) {  
2     return {  
3         pathname: url.pathname  
4     }  
5 }  
6
```

Kode : /routes/+layout.svelte

JS +layout.js

+layout.svelte ×

```
1  <script>
2    const {children, data} = $props();
3  </script>
4    ⚡
5  <p>{data.pathname}</p>
6
7  <ul>
8    <li><a href="/">Home</a></li>
9    <li><a href="/counter">Counter</a></li>
10   <li><a href="/profile/user">Profile</a></li>
11 </ul>
12
13 {@render children()}
14
```


Page Information

- Kadang, mengambil informasi dari Load Parameter agak sedikit bertele-tele, apalagi jika hanya ingin mengambil informasi dan menyimpannya di data
- Untungnya, Svelte menyediakan helper untuk mendapatkan informasi page tanpa harus menggunakan Load function
- Kita bisa menggunakan page di `$app/state`
- [https://svelte.dev/docs/kit/\\$app-state#page](https://svelte.dev/docs/kit/$app-state#page)

Tugas

- Kita akan ubah pathname di layout data menjadi menggunakan page object di `$app/state`
- Silahkan hapus `/routes/+layout.js`

Kode : /routes/+layout.svelte

 +layout.svelte ×

```
1  <script>
2      import {page} from '$app/state';
3      const {children} = $props();
4  </script>
5
6  <p>{page.url.pathname}</p>
7
8  <ul>
9      <li><a href="/">Home</a></li>
10     <li><a href="/counter">Counter</a></li>
11     <li><a href="/profile/user">Profile</a></li>
12 </ul>
13
14 {@render children()}
```

Route Parameter

Route Parameter

- Saat membuat halaman web, kadang kita membutuhkan URL yang dinamis, misal `/products/123`, dimana 123 adalah data dinamis sesuai dengan kode product
- Pada kasus seperti ini, kita bisa menggunakan fitur Route Parameter di Svelte
- Caranya sangat mudah, kita bisa buat folder dengan format `[nama]`, misal `/routes/products/[id]`, yang artinya id bisa kita dapat nilainya dari props
- Route Parameter bisa lebih dari satu, misal `/routes/categories/[category]/products/[product]`, artinya terdapat props `category` dan `product`
- Untuk mendapatkan informasi route parameter, kita bisa menggunakan attribute `params` di parameter load function

Tugas

- Kita akan coba buat halaman untuk menampilkan data product berdasarkan id

Kode : /static/api/products/*.json

{ 1.json { 2.json ×

```
r-s 1  {
      2     "id": 2,
      3     "name": "Apple iPhone 16 Pro",
      4     "description": "Apple iPhone 16 Pro",
      5     "price": 200000000
      6 }
```

Kode : /routes/products/[id]/+page.js

JS +page.js ×

```
1 export function load({params}) {  
2     return {  
3         id: params.id  
4     }  
5 }
```

Kode : /routes/products/[id]/+page.svelte

+page.svelte ×

```
1  <script>
2    const {data} = $props()
3
4    async function getProduct() {
5      const response = await fetch(`/api/products/${data.id}.json`);
6      return response.json();
7    }
8  </script>
9
10  {#await getProduct()}
11    <p>Loading product</p>
12  {:then product}
13    <h1>{product.id} - {product.name}</h1>
14    <p>{product.description}</p>
15    <p>{product.price}</p>
16  {:catch error}
17    <p>Error: {error.message}</p>
18  {/await}
```

Invalidating

Cache

- Svelte secara default akan melakukan cache data Load Function yang sudah dikunjungi
- Misal, kita mengunjungi halaman /products, lalu mengunjungi /products/1, lalu kembali ke /products dan kembali lagi ke /products/1
- Load Function pada /products dan /product/1, akan di load setiap kita berpindah-pindah halaman, namun jika ternyata /products dan /products/1 menggunakan Layout yang sama, maka Load Function untuk Layout tidak akan dipanggil ulang, hal ini karena Svelte menganggap tidak ada terjadi perubahan sama sekali pada Layout
- Kadang, mungkin hal ini tidak kita inginkan

Tugas

- Sekarang kita akan coba tambahkan waktu pada halaman /products dan /products/[id]
- Dan kita akan tambahkanlah Layout yang sama

Kode : Layout /products

JS +layout.js ×

```
1 export function load() {  
2     return {  
3         date: new Date(),  
4     }  
5 }
```

Svelte +layout.svelte ×

```
1 <script>  
2     const {data, children} = $props();  
3 </script>  
4  
5 <h2>Date : {data.date}</h2>  
6  
7 {@render children()}  
8
```

Kode : /products

JS +page.js ×

```
1 export function load() {  
2   return {  
3     products: ["1", "2", "3"]  
4   }  
5 }
```

+page.svelte ×

```
1 <script>  
2   const {data} = $props();  
3 </script>  
4  
5 <h1>Products</h1>  
6  
7 <ul>  
8   {#each data.products as product}  
9     <li>  
10      <a href="/products/{product}">{product}</a>  
11    </li>  
12  {/each}  
13 </ul>
```


Kode /products/[id]

Js +page.js ×

```
1 export function load({params}) {  
2   return {  
3     id: params.id  
4   }  
5 }
```

+page.svelte ×

```
1 <script>  
2   const {data} = $props()  
3  
4   async function getProduct() {  
5     const response = await fetch(`/api/products/  
6     return response.json();  
7   }  
8 </script>  
9  
10 <a href="/products">Back to Products</a>  
11
```

Invalidating

- Jika kita ingin menjalankan kembali Load Function, maka kita bisa melakukan invalidating
- Terdapat dua function yang bisa kita gunakan untuk melakukan invalidating
- `invalidate(url)` jika kita ingin menjalankan kembali semua Load Function yang tergantung ke parameter url di halaman yang sedang aktif :
[https://svelte.dev/docs/kit/\\$app-navigation#invalidate](https://svelte.dev/docs/kit/$app-navigation#invalidate)
- `invalidateAll()` jika kita ingin menjalankan kembali semua Load Function di halaman yang sedang aktif :
[https://svelte.dev/docs/kit/\\$app-navigation#invalidateAll](https://svelte.dev/docs/kit/$app-navigation#invalidateAll)

Kode : Invalidating

+page.svelte x

```
1 <script>
2   import {onMount} from "svelte";
3   import {invalidateAll} from "$app/
4
5   const {data} = $props()
6
7   async function getProduct() {
8     const response = await fetch(`/a
9     return response.json();
10  }
11
12  onMount(() => {
13    invalidateAll()
14  });
15 </script>
```

+page.svelte x

```
1 <script>
2   import {onMount} from "svelte";
3   import {invalidateAll} from "$app/
4
5   const {data} = $props();
6
7   onMount(() => {
8     invalidateAll();
9   })
10 </script>
```

Fetch Request

Fetch Request

- Saat kita membuat Load Function, mungkin kita akan melakukan Fetch pada Load Function
- Namun perlu diperhatikan, bahwa saat pertama kali halaman di akses, Svelte akan menjalankan mode SSR (Server Side Render), selanjutnya baru CSR (Client Side Render)
- Fetch biasanya digunakan di CSR untuk melakukan Ajax Request, hal ini bisa bermasalah ketika kita jalankan di SSR, karena Fetch akan berjalan di Server, bukan di Client (Browser)

Kode : /products/[id] (Error)

JS +page.js ×

```
1 export async function load({params}) {  
2   const response = await  
3     fetch(`/api/products/${params.id}.json`);  
4   return await response.json();  
5 }
```

Svelte +page.svelte ×

```
1 <script>  
2   import {onMount} from "svelte";  
3   import {invalidateAll} from "$app/navigation";  
4  
5   const {data} = $props()  
6  
7   onMount(() => {  
8     invalidateAll()  
9   });  
10 </script>  
11  
12 <a href="/products">Back to Products</a>  
13  
14 <h1>{data.id} - {data.name}</h1>  
15 <p>{data.description}</p>  
16 <p>{data.price}</p>
```

Error

```
at process.processTicksAndRejections (node:internal/process/task_queues:105:5)
at async eval (/Users/khannedy/Developments/SVELTE/belajar-svelte-kit/node_modules/@sveltejs/kit/src/
Error: Cannot use relative URL (/api/products/3.json) with global fetch – use `event.fetch` instead: http://localhost:3000/api/products/3.json
at globalThis.fetch (file:///Users/khannedy/Developments/SVELTE/belajar-svelte-kit/node_modules/@sveltejs/kit/src/runtime/server/fetch.js:105:11)
at load (/Users/khannedy/Developments/SVELTE/belajar-svelte-kit/src/routes/products/[id]/+page.js:2:2)
at load_data (/Users/khannedy/Developments/SVELTE/belajar-svelte-kit/node_modules/@sveltejs/kit/src/runtime/server/page/load_data.js:25:11)
at process.processTicksAndRejections (node:internal/process/task_queues:105:5)
at async eval (/Users/khannedy/Developments/SVELTE/belajar-svelte-kit/node_modules/@sveltejs/kit/src/
Error: Cannot use relative URL (/api/products/3.json) with global fetch – use `event.fetch` instead: http://localhost:3000/api/products/3.json
at globalThis.fetch (file:///Users/khannedy/Developments/SVELTE/belajar-svelte-kit/node_modules/@sveltejs/kit/src/runtime/server/fetch.js:105:11)
at load (/Users/khannedy/Developments/SVELTE/belajar-svelte-kit/src/routes/products/[id]/+page.js:2:2)
at load_data (/Users/khannedy/Developments/SVELTE/belajar-svelte-kit/node_modules/@sveltejs/kit/src/runtime/server/page/load_data.js:25:11)
at process.processTicksAndRejections (node:internal/process/task_queues:105:5)
at async eval (/Users/khannedy/Developments/SVELTE/belajar-svelte-kit/node_modules/@sveltejs/kit/src/
Error: Cannot use relative URL (/api/products/3.json) with global fetch – use `event.fetch` instead: http://localhost:3000/api/products/3.json
at globalThis.fetch (file:///Users/khannedy/Developments/SVELTE/belajar-svelte-kit/node_modules/@sveltejs/kit/src/runtime/server/fetch.js:105:11)
at load (/Users/khannedy/Developments/SVELTE/belajar-svelte-kit/src/routes/products/[id]/+page.js:2:2)
at load_data (/Users/khannedy/Developments/SVELTE/belajar-svelte-kit/node_modules/@sveltejs/kit/src/runtime/server/page/load_data.js:25:11)
at process.processTicksAndRejections (node:internal/process/task_queues:105:5)
at async eval (/Users/khannedy/Developments/SVELTE/belajar-svelte-kit/node_modules/@sveltejs/kit/src/
```

Svelte Fetch Function

- Untuk mengatasi masalah ini, Svelte menyediakan Fetch Function sendiri, dimana Fetch Function ini penggunaannya sama seperti Fetch Function pada Browser
- Svelte Fetch Function ini bisa tahu ketika berjalan di mode SSR atau CSR
- Untuk menggunakan Svelte Fetch Function, kita bisa menambahkan attribute `fetch` pada Load Function

Kode : /products/[id]

JS +page.js ×

```
1 export async function load({params, fetch}) {  
2     const response = await fetch(`/api/products/${params.id}.json`);  
3     return await response.json();  
4 }  
5
```

Parent Data

Parent Data

- Secara default semua data Load Function akan digabungkan dan bisa digunakan pada Page
- Namun, pada beberapa kasus, mungkin kita ingin menggunakan data dari Load Function (Parent) di Load Function (Child), misal kita ingin menggunakan data Load Function Layout di Load Function Page
- Untuk menggunakan data di Load Function (Parent), kita bisa menggunakan attribute parent
- parent merupakan Promise function

Tugas

- Di halaman Load Function `/profile/wallet/+page.js`, kita akan coba ambil data user yang berasal dari Load Function `/profile/+layout.js`

Kode : /profile/wallet

JS +page.js ×

```
1 export async function load({parent}) {  
2   const data = await parent();  
3  
4   return {  
5     description: `Wallet ${data.user}`,  
6     balance: 1000000  
7   }  
8 }
```

Svelte +page.svelte ×

```
1 <script>  
2   const {data} = $props();  
3 </script>  
4  
5 <h1>Wallet {data.user}</h1>  
6  
7 <p>Description : {data.description}</p>  
8 <p>Balance : {data.balance}</p>
```

Server Load Data

Server Load Data

- Svelte juga mendukung Load data di Server
- Salah satu kelebihan Load data di Server adalah, kita bisa mengakses data di Server, misal data di Database misalnya
- Untuk membuat Server Load Function, kita bisa menggunakan file `+page.server.js` untuk Page data, dan `+layout.server.js` untuk Layout Data
- Semua parameter yang bisa digunakan di Load Function, bisa digunakan juga pada Server Load Function

Load Function vs Server Load Function

- Load Function bisa berjalan di Client atau di Server (untuk akses pertama kali), sedangkan Server Load Function hanya bisa berjalan di Server
- Jika dalam satu Page terdapat Load Function dan Server Load Function, maka yang pertama kali akan dieksekusi adalah Server Load Function
- Server Load Function bisa menggunakan semua parameter yang ada di Load Function, namun ada tambahan parameter yang bisa digunakan pada Server Load Function
- Jika kita membuat Load Function dan Server Load Function, kita bisa mendapatkan hasil dari Server Load Function di Load Function dalam attribute data

Tugas

- Kita akan membuat halaman /todos, dimana kita akan buat data dari server dan dari client

Kode : /routes/todos

JS +page.server.js ×

```
1 export async function load() {
2   return {
3     todos: [
4       "From Server"
5     ]
6   }
7 }
```

JS +page.js ×

```
1 export async function load({data}) {
2   return {
3     todos: ["From Client", ...data.todos]
4   }
5 }
```

Kode : /routes/todos/+page.svelte



+page.svelte ×

```
1 <script>
2   const {data} = $props();
3 </script>
4   ⚡
5 <ul>
6     {#each data.todos as todo}
7       <li>{todo}</li>
8     {/each}
9 </ul>
```

Server Load Parameter

Server Load Parameter

- Server Load Function memiliki parameter dengan tipe RequestEvent
- <https://svelte.dev/docs/kit/@sveltejs-kit#RequestEvent>
- RequestEvent memiliki banyak attribute tambahan yang bisa kita gunakan untuk memanipulasi Response seperti cookies, setHeaders, dan lain-lain

Tugas

- Kita akan coba buat halaman untuk Login, dimana data login akan kita simpan di cookie
- Setelah login, kita akan simpan datanya di Cookie
- Jika logout, kita akan hapus datanya dari Cookie

Kode : /users/login

JS +page.server.js ×

```
1 import {redirect} from "@sveltejs/kit";
2
3 export async function load({cookies, url}) {
4   const user = url.searchParams.get("user")
5   if (user) {
6     cookies.set("user", user, {path: "/"})
7     redirect(303, "/users/dashboard")
8   }
9
10  if (cookies.get("user")) {
11    redirect(303, "/users/dashboard")
12  }
13
14  return {}
15 }
```

Svelte +page.svelte ×

```
1 <form action="/users/login" method="get">
2   <label for="user">User</label>
3   <input type="text" id="user" name="user"/>
4   ⚡ <button>Login</button>
5 </form>
```

Kode : /users/dashboard

JS +page.server.js ×

```
1 import {redirect} from "@sveltejs/kit";
2
3 export async function load({cookies, url}) {
4   if (url.searchParams.get("logout")) {
5     cookies.delete("user", {path: "/"})
6     redirect(303, "/users/login")
7   }
8
9   if (!cookies.get("user")) {
10     redirect(303, "/users/login")
11   }
12
13   return {
14     user: cookies.get("user")
15   }
16 }
```

Svelte +page.svelte ×

```
1 <script>
2   const {data} = $props();
3 </script>
4
5 <h1>Hello {data.user}</h1>
6
7 <a href="/users/dashboard?logout=true">Logout</a>
```


Navigation

Navigation

- Sebelumnya untuk melakukan navigasi dari satu halaman ke halaman lain, kita bisa menggunakan elemen anchor `<a>`
- Namun bagaimana jika kita ingin melakukan navigasi secara programmatic di kode JavaScript?
- Kita bisa memanfaatkan function goto pada Svelte
- [https://svelte.dev/docs/kit/\\$app-navigation#goto](https://svelte.dev/docs/kit/$app-navigation#goto)

Tugas

- Kita akan coba buat halaman navigation untuk berpindah dari satu halaman ke halaman yang lain
- Pada halaman tersebut kita akan buat form dengan input, dan kita akan melakukan navigasi sesuai dengan input nya

Kode : /navigation

+page.svelte ×

```
1 <script>
2   import {goto} from "$app/navigation";
3
4   let location = $state("")
5
6   function go() {
7     goto(location)
8   }
9 </script>
10
11 <div>
12   <label for="location">Location</label>
13   <input type="text" id="location" bind:value={location}>
14   <button onclick={go}>Go</button>
15 </div>
```

API Route

API Route

- Tidak hanya membuat Page, Svelte juga membolehkan kita membuat API response
- Caranya kita bisa menggunakan file `+server.js`
- Kita bisa membuat function sesuai dengan jenis Http Method yang digunakan, misal GET, POST, PUT, PATCH, DELETE, OPTIONS dan HEAD
- API Route Function memiliki parameter `RequestEvent` dan mengembalikan Response object
- <https://svelte.dev/docs/kit/@sveltejs-kit#RequestEvent>
- <https://developer.mozilla.org/en-US/docs/Web/API/Response>
- Perlu diingat bahwa API Route akan berjalan di Server, bukan di Client

Tugas

- Kita akan coba pindahkan Server Load Function di `/users/login` dan `/users/dashboard` menjadi API Route

Kode : /api/users/login

JS +server.js ×

```
1 import {json} from "@sveltejs/kit";
2
3 export async function POST({request, cookies}) {
4   const body = await request.json()
5   cookies.set("user", body.user, {path: "/"})
6
7   return json({
8     user: body.user
9   }, {
10     status: 200
11   })
12 }
```


Kode : /api/users/current

JS +server.js x

```
1 import {json} from "@sveltejs/kit";
2
3 export async function GET({cookies}) {
4   if (cookies.get("user")) {
5     return json({
6       user: cookies.get("user")
7     }, {
8       status: 200
9     })
10  } else {
11    return json({
12      user: null
13    }, {
14      status: 401
15    })
16  }
17 }
```

```
18
19 export async function DELETE({cookies}) {
20   cookies.delete("user", {path: "/"})
21   return json({
22     user: null
23   }, {
24     status: 200
25   })
26 }
```

Kode : /users/login

+page.svelte x

```
1 <script>
2   import {goto} from "$app/navigation";
3
4   let user = $state("")
5
6   async function login(e) {
7     e.preventDefault()
8     const response = await fetch("/api/users/login", {
9       method: "POST",
10      body: JSON.stringify({user}),
11    })
12
13    if (response.ok) {
14      await goto("/users/dashboard")
15    }
16  }
17 </script>
```

```
<form>
  <label for="user">User</label>
  <input type="text" id="user" name="user"
    bind:value={user}/>
  <button onclick={login}>Login</button>
</form>
```

Kode : /users/dashboard

+page.svelte x

```
1 <script>
2   import {goto} from "$app/navigation";
3
4   const {data} = $props();
5
6   async function current() {
7     const response = await fetch('/api/users/current');
8     if (response.ok) {
9       return response.json();
10    } else {
11      await goto("/users/login")
12    }
13  }
14
```

```
  async function logout() {
    await fetch('/api/users/current', {
      method: 'DELETE'
    })
    await goto("/users/login")
  }
</script>

<#await current()>
  Loading user
{:then data}
  <h1>Hello {data.user}</h1>
  <a href="/users/dashboard" onclick={logout}>Logout</a>
</await>
```

Error Page


Error Page

- Saat terjadi error ketika Load Function dieksekusi, secara default Svelte akan menampilkan halaman error
- Namun, kadang-kadang, kita ingin membuat halaman error sesuai dengan yang kita mau
- Kita bisa membuat halaman error di Page yang dengan menggunakan file `+error.svelte`
- Saat kita menambahkan halaman `+error.svelte`, secara otomatis sub path nya akan menggunakan menggunakan halaman error tersebut
- Misal kita buat `/users/+error.svelte`, maka ketika terjadi error di `/users/login` atau `/users/dashboard`, maka otomatis menggunakan `/users/+error.svelte`

Tugas

- Kita akan ubah halaman `/users/dashboard`, dimana jika user tidak login, kita akan throw error di Load Function sehingga halaman error akan ditampilkan

Kode : /users/+error.svelte

 +error.svelte ×


```
1  <h1>Unauthorized</h1>
2
3  <p>You are not authorized to view this page.</p>
4
5  <a href="/users/login">Login</a>
```

Kode : /users/dashboard

Js +page.js x

```
1 export async function load({fetch}) {
2   const response = await fetch('/api/users/current');
3   if (!response.ok) {
4     throw new Error(response.statusText);
5   }
6   
7   const body = await response.json();
8
9   return {
10    user: body.user
11  }
12 }
```

Svelte +page.svelte x

```
1 <script>
2   import {goto} from "$app/navigation";
3
4   const {data} = $props();
5
6   async function logout() {
7     await fetch('/api/users/current', {
8       method: 'DELETE'
9     })
10    await goto("/users/login")
11  }
12 </script>
13
14 <h1>Hello {data.user}</h1>
15 
16 <a href="/users/dashboard" onClick={logout}>Logout</a>
```


Redirect

Redirect

- Sebelumnya, kita sudah pernah menggunakan function `redirect()`
- Redirect merupakan function yang bisa kita gunakan untuk melakukan redirect halaman
- Namun function `redirect()` ini hanya bisa digunakan pada Server, jika kita ingin melakukan redirect pada Client, kita bisa menggunakan function `goto()`
- <https://svelte.dev/docs/kit/@sveltejs-kit#redirect>

Tugas

- Sekarang kita akan coba ubah `/profile/+layout.js` menjadi `/profile/+layout.server.js`
- Kita akan cek, jika tidak ada cookie user, kita akan lakukan redirect ke halaman login

Kode : /profile/+layout.server.js

JS +layout.server.js ×

```
1 import {redirect} from "@sveltejs/kite";
2
3 export function load({cookies}) {
4     const user = cookies.get("user");
5     if (!user) {
6         redirect(302, "/users/login")
7     }
8
9     return {
10         user: user
11     }
12 }
```

Form Action

Form Action

- Svelte juga memiliki fitur untuk menerima membuat route yang menerima input POST Form Action
- Kita bisa buat variable tipe data object dengan nama actions pada +page.server.js
- Lalu kita bisa tambahkan method pada object tersebut dengan nama default()
- Parameter pada default() method sama seperti pada Server Load Data, kita bisa menggunakan RequestEvent
- Setelah actions dieksekusi, selanjutnya halaman akan di render seperti biasanya, sehingga Load Function akan dipanggil seperti biasa setelah selesai eksekusi action

Tugas

- Sekarang kita akan coba update halaman todolist yang sebelumnya kita buat
- Kita akan tambahkan form untuk menambah data todo, dan menggunakan action untuk menerima input data todo baru nya

Kode : /todos

```
+page.svelte × JS +page.server.js

1 <script>
2   const {data} = $props();
3 </script>
4
5 <form method="post">
6   <label for="todo">
7     <input type="text" id="todo" name="todo">
8   </label>
9   <button type="submit">Add</button>
10 </form>
11
12 <ul>
13   {#each data.todos as todo}
14     <li>{todo.id} - {todo.name}</li>
15   {/each}
16 </ul>
```

```
+page.svelte JS +page.server.js ×

1 let lastId = 0;
2 const todos = [];
3
4 export async function load() {
5   return {
6     todos: todos
7   }
8 }
9
10 export const actions = {
11
12   default: async ({request}) => {
13     const data = await request.formData();
14     const todo = data.get("todo")
15     todos.push({
16       id: lastId++,
17       name: todo
18     })
19   }
}
```


Named Action

- Kadang-kadang, dalam satu URL path, kita ingin bisa menerima lebih dari satu jenis Form Action
- Svelte mendukung ini dengan cara membuat method selain default() method
- Caranya kita bisa tambahkan pada element form action `*/namaMethod`
- Jika kita menggunakan nama selain default(), maka kita tidak bisa menggunakan default() lagi, kita harus ganti nama method nya menjadi nama method lain

Tugas

- Kita akan coba untuk menambahkan form action untuk menghapus todo

Kode : /todos/+page.svelte

```
5  <form method="post" action="?/add">
6    <label for="todo">
7      <input type="text" id="todo" name="todo">
8    </label>
9    <button type="submit">Add</button>
10 </form>
11
12 <ul>
13   {#each data.todos as todo}
14     <li>
15       {todo.id} - {todo.name}
16       <form method="post" action="?/delete" style="display: inline-block">
17         <input type="hidden" name="id" value={todo.id}>
18         <button type="submit">Delete</button>
19       </form>
20     </li>
21   {/each}
```

Kode : /todos/+page.server.js

```
10 export const actions = {
11
12   add: async ({request}) => {
13     const data = await request.formData();
14     const todo = data.get("todo")
15     todos.push({
16       id: lastId++,
17       name: todo
18     })
19   },
20
21   delete: async ({request}) => {
22     const data = await request.formData();
23     const id = data.get("id")
24     todos = todos.filter(item => item.id !== Number(id))
25   }
}
```

Validation Error

- Svelte menyediakan function `fail(status, data)` yang bisa kita gunakan untuk menambahkan validation error
- Function `fail()` akan mengembalikan response dengan data yang kita inginkan
- Pada template, kita bisa menggunakan attribute `form` pada `$props()` untuk mendapatkan data pada function `fail()`
- <https://svelte.dev/docs/kit/@sveltejs-kit#fail>

Tugas

- Kita akan coba tambahkan validation error pada form untuk input data todo

Kode : /todos/+page.svelte

 +page.svelte ×  +page.server.js

```
1 <script>
2   const {data, form} = $props();
3 </script>
4
5 <form method="post" action="?/add">
6   {#if form?.error}
7     <label for="todo" style="color: red;">{form?.message}</label>
8   {/if}
9   <input type="text" id="todo" name="todo">
10  <button type="submit">Add</button>
11 </form>
12
```

Kode : /todos/+page.server.js

```
11
12  export const actions = {
13
14    add: async ({request}) => {
15      const data = await request.formData();
16      const todo = data.get("todo")
17
18      if (todo.trim() === "") {
19        return fail(400, {error: true, message: "Todo can not be empty"})
20      }
21
22      todos.push({
```


Page Option

Page Option

- Secara default, SvelteKit akan melakukan render halaman pertama kali di server (SSR), selanjutnya baru akan dilakukan di client (CSR)
- Hal ini memastikan bahwa komponen yang kita buat harus bisa berjalan baik itu di Server maupun di Client. Maka dari itu kita selalu memisahkan logic aplikasi di kode JavaScript di file yang berbeda
- Namun, SvelteKit memberi kita kebebasan untuk menentukan opsi yang kita mau, jika tidak ingin menggunakan default opsi yang dilakukan oleh SvelteKit
- <https://svelte.dev/docs/kit/page-options>

Prerender

- Pada beberapa kasus, mungkin halaman web kita bisa kita generate langsung dalam bentuk file HTML, sehingga tidak perlu dibuat ulang secara realtime ketika request diterima pertama kali, atau kita sebut prerender
- Jika kita ingin memaksa SvelteKit membuatkan file HTML yang sudah jadi (prerender), kita bisa menggunakan flag prerender menjadi true
- <https://svelte.dev/docs/kit/page-options#prerender>

Tugas

- Tambahkan prerender pada halaman /counter

Kode : /counter

JS +page.js ×

ar-s 1 export const prerender = true;

Build App

- Coba lakukan build app menggunakan perintah :
`npm run build`
- Kita bisa melihat hasil HTML prerender di folder `.svelte-kit/output/prerendered`

Client dan Server Side Render

- Seperti yang dibahas sebelumnya, request pertama pada SvelteKit akan dilakukan dalam bentuk SSR, selanjutnya baru CSR
- Namun, jika kita ingin mengubahnya, kita bisa mengubah flag untuk csr dan ssr sesuai dengan keinginan kita
- <https://svelte.dev/docs/kit/page-options#ssr>
- <https://svelte.dev/docs/kit/page-options#csr>

State Management

State Management

- Materi state management sebenarnya sudah kita bahas di materi Svelte Dasar
- Namun kita tahu bahwa State Management di Svelte itu berjalan di Client (Browser), artinya jangan berpikir bahwa data State yang kita buat menggunakan `$state()` itu akan bisa diakses di Server

Shared State

- Menggunakan Shared State antara Client dan Server sebenarnya sangat berbahaya, hal ini karena artinya satu data yang sama di Server bisa diakses oleh semua Client
- Contoh, kita pernah membuat Shared State di halaman /todos
- Kita menyimpan data todos pada variable todos di Server
- Coba buka di beberapa Browser, maka semua Browser akan membuka data yang sama, oleh karena itu penggunaan Shared State harus hati-hati

Advanced Routing

Advanced Routing

- Sebelumnya kita sudah membahas tentang Routing dan Route Parameter
- Di materi ini kita akan bahas fitur lanjutan dari Routing

Optional Parameter

- Sebelumnya saat kita tambahkan Route Parameter, maka parameter tersebut wajib kita isi
- Namun, jika ada kasus kita ingin menambahkan Route Parameter yang Optional, atau tidak wajib, maka kita bisa lakukan dengan mengubah nama folder dari `[param]` menjadi `[[param]]`
- Hati hati ketika menggunakan Optional Parameter, karena bisa bentrok dengan route lain, misal
- `[[lang]]/home`, mungkin bisa bentrok dengan `/home`

Tugas

- Kita akan coba buat halaman /home yang mendukung multi bahasa, namun jika parameter bahasa tidak ada, maka kita akan gunakan default bahasa Indonesia

Kode /[[lang]]/home

```
+page.svelte × JS +page.js
1 <script>
2   const {data} = $props();
3 </script>
4   ⚡
5 <h1>{data.hello}</h1>
```

```
+page.svelte JS +page.js ×
1 export async function load({params}) {
2   const lang = params.lang || "id";
3
4   const hello = {
5     "en": "Hello",
6     ⚡ "id": "Halo",
7     "fn": "Hola"
8   }
9
10  return {
11    hello: hello[lang]
12  }
13 }
```

Rest Parameter

- Kadang kita ingin membuat Route Parameter yang tidak terbatas, biasanya dilakukan di bagian akhir
- Misal kita ingin membuat route `/files/...`, dimana `...` bisa tidak terbatas, contoh `/files/1.png`, `/files/folder/lagi/2.png`, dan lain-lain
- Pada kasus ini, kita bisa menggunakan Rest Parameter
- Untuk membuat Rest Parameter, kita bisa gunakan folder `[...param]`, diawali titik tiga sebelum nama parameter

Tugas


- Kita akan membuat halaman `/files/...`, dimana misal saja digunakan untuk mengambil file

Kode : /files/[...file]

JS +page.js ×

```
1 export async function load({params}) {  
2   return {  
3     file: params.file  
4   }  
5 }
```

Svelte +page.svelte ×

```
1 <script>  
2   const {data} = $props();  
3 </script>  
4   
5 <h1>File {data.file}</h1>
```

Parameter Matchers

- Secara default, data di Route Parameter bisa kita input dengan format apapun
- Namun, kadang-kadang kita ingin memberi batasan format input nya
- Kita bisa menggunakan `match(param)` function pada file JavaScript di folder `/src/params`, dimana `match(param)` function tersebut mengembalikan boolean, jika kita anggap valid maka return true, jika tidak valid maka return false
- Selanjutnya, pada Route Parameter, kita harus ubah nama folder menjadi `[param=file]`, dimana file merupakan file JavaScript yang berisikan `match(param)` function di folder `src/params`

Tugas

- Misal, kita akan tambahkan `match(param)` function pada halaman `/[[lang]]/home` agar pengguna tidak bisa memasukkan lang yang tidak kita support
- Kita akan buat `match()` function di file `/src/params/lang.js`
- Dan kita akan ubah folder `[[lang]]` menjadi `[[lang=lang]]`

Kode : src/params/lang.js

JS lang.js ×

```
1 export function match(lang) {  
2     const value = lang || "id"  
3     return ["en", "id", "fn"].includes(value);  
4 }
```

Advanced Layout

Advanced Layout

- Sebelumnya kita sudah tahu bahwa Layout di SvelteKit itu dibuat secara hirarki, artinya dari yang paling atas ke yang paling bawah
- Namun, pada kasus tertentu, mungkin kita tidak ingin menggunakan format Layout hirarki seperti itu
- Contoh, kisa memiliki layout yang sama untuk halaman /dashboard dan /account menggunakan Layout yang sama, namun misal untuk halaman /about ingin menggunakan Layout yang berbeda
- Jika menggunakan Layout hirarki, ini agak sulit, karena jika membuat layout di Route / , maka semua akan menggunakan Layout yang sama

Route Group

- Pada kasus seperti ini, kita bisa membuat Route Group, caranya kita bisa membuat Route Group dengan menggunakan folder (group), dimana group adalah nama group
- Route di dalam group yang sama, akan menggunakan Layout hirarki yang sama, sehingga misal kita bisa buat group berbeda jika kita ingin menggunakan Layout yang berbeda

Tugas

- Kita akan buat (default) group, lalu kita akan pindahkan semua route yang sudah kita buat ke (default) group
- Selanjutnya kita akan buat (guest) group, dimana kita akan coba gunakan Layout yang berbeda untuk halaman route di group tersebut

Struktur Folder

JS lang.js

✓ folder routes

✓ folder (default)

> folder [[lang=lang]]

> folder api

> folder counter

> folder files

> folder navigation

> folder products

> folder profile

> folder todos

> folder users

+layout.svelte


+page.svelte

folder (guest)

<> app.html

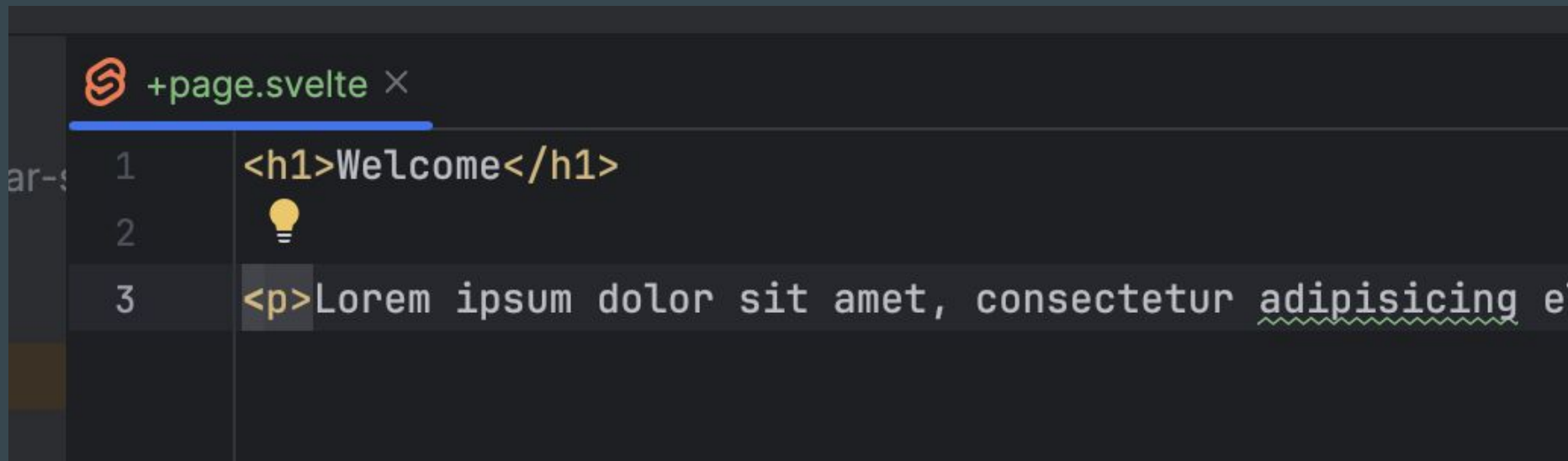
> folder static

Kode : /src/routes/(guest)/+layout.svelte

 +layout.svelte ×

```
1  <script>
2      const {children} = $props();
3  </script>
4
5  <h1>Hi Guest</h1>
6
7  {@render children()}
```

Kode : /src/routes/(guest)/about/+page.svelte



```
+page.svelte x
1 <h1>Welcome</h1>
2 
3 <p>Lorem ipsum dolor sit amet, consectetur adipisicing e
```

Break Layout Hierarchy

- Kadang, mungkin ada kasus kita ingin merusak Layout hirarki, misal ketika kita ingin menggunakan Nested Route , namun tidak ingin menggunakan Layout di atasnya
- Contoh kita punya :
`(guest)/+layout.svelte`
`(guest)/a/+layout.svelte`
`(guest)/a/b/+layout.svelte`
`(guest)/a/b/c/+page.svelte`
- Misal, di Route `/a/b/c`, kita tidak ingin menggunakan Layout b atau layout a

+page@

- Untuk merusak Layout hirarki, kita bisa gunakan @ pada file +page, misal jika kita menggunakan file :
- (guest)/a/b/c/+page@b.svelte, artinya kita menggunakan Layout b/+layout.svelte
- (guest)/a/b/c/+page@a.svelte, artinya kita menggunakan Layout a/+layout.svelte
- (guest)/a/b/c/+page@(guest).svelte, artinya kita menggunakan Layout (guest)/+layout.svelte
- (guest)/a/b/c/+page@.svelte, artinya kita menggunakan Layout /src/routes/+layout.svelte

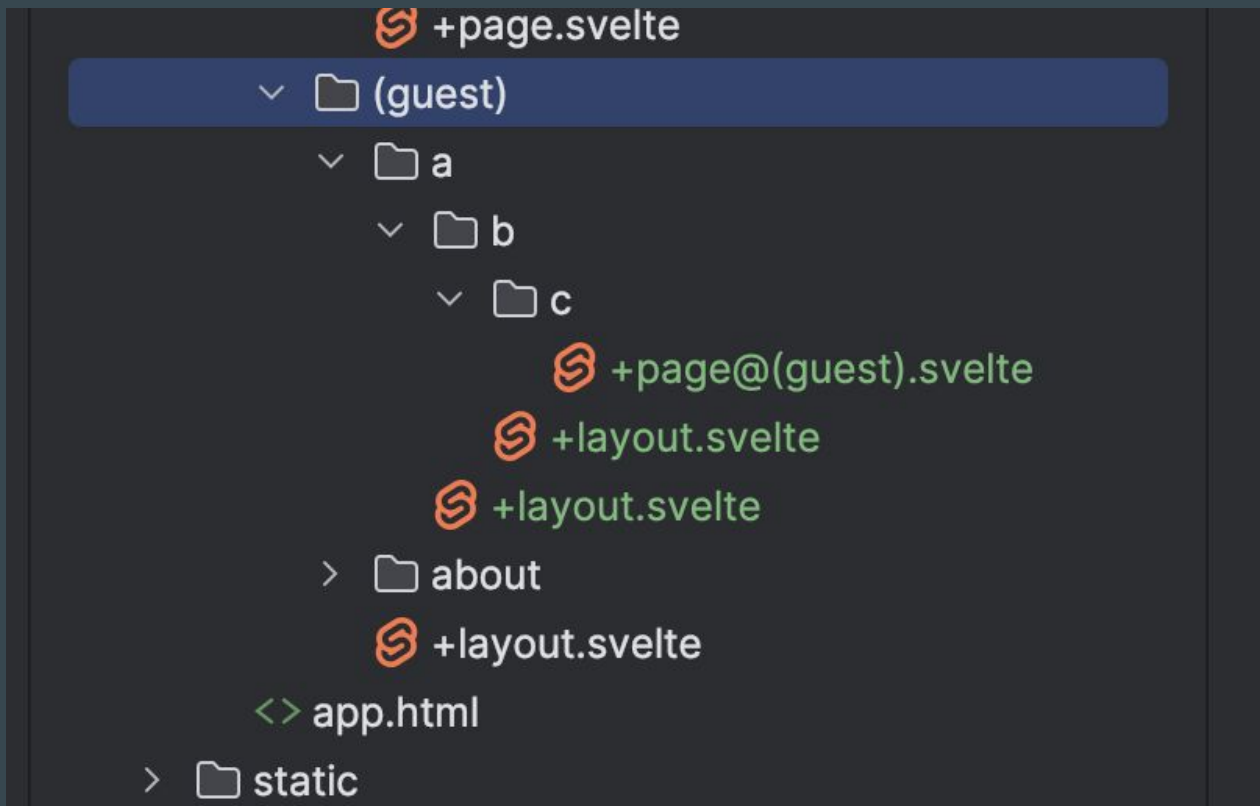
+layout@

- Selain pada +page.svelte, kita juga bisa merusak Layout hirarki pada file +layout.svelte
- Caranya sama saja menggunakan +layout@, dan penggunaannya sama seperti pada +page@

Tugas

- Silahkan buat layout /a, /a/b, dan coba-coba ubah halaman /a/b/c/+page dengan Layout yang berbeda-beda

Struktur Folder



Hooks

Hooks

- Hooks adalah fitur di SvelteKit yang akan dieksekusi ketika ada kejadian tertentu
- SvelteKit memiliki beberapa Hooks
- Server Hooks, yang disimpan di `src/hooks.server.js`
- Client Hooks, yang disimpan di `src/hooks.client.js`
- dan Shared Hooks (berjalan di Server dan Client), yang disimpan di `src/hooks.js`

Server Hooks

- Server Hooks akan dijalankan setiap SvelteKit menerima request, baik itu ketika aplikasi berjalan, ataupun ketika prerender
- Server Hooks dibuat dengan membuat function `handle(request)`, dimana parameter request merupakan object berisi attribute `event` dan `resolve`
- Kita bisa mengubah response dengan mengembalikan object `Response`, atau jika ingin meneruskan request ke aplikasi, kita bisa menggunakan attribute `resolve` pada request
- Function `handle(request)` tidak bisa menggunakan `fetch`, jika kita butuh `fetch`, kita bisa menggunakan `handleFetch(request)`

Tugas

- Kita akan coba menambahkan Server Hooks untuk melakukan log semua request yang diterima oleh Server SvelteKit

Kode : src/hooks.server.js

JS hooks.server.js ×

```
1 export async function handle({event, resolve}) {  
2     console.log('receive request', event.url.href)  
3     return await resolve(event)  
4 }
```

Shared Hooks

- Sebelumnya function `handle(request)` hanya bisa digunakan di Server Hooks, namun terdapat function lain yang bisa kita gunakan di Server Hooks (`hooks.server.js`) atau di Client Hooks (`hooks.client.js`), yaitu function `handleError()` dan function `init()`
- Function `handleError(param)` akan dipanggil ketika error terjadi sewaktu proses loading atau prerender halaman. Param di `handleError` memiliki attribute `error`, `event`, `status code` dan `message`
- Function `init()` akan dipanggil ketika pertama kali aplikasi SvelteKit berjalan. Jika di Server artinya akan dijalankan sekali ketika app SvelteKit berjalan. Jika di Client artinya akan dijalankan ketika pertama kali diload oleh Browser

Tugas

- Sekarang kita akan coba tambahkan `handleError()` dan `init()` di Client Hooks
- Dan juga `init()` di Server Hooks

Kode : src/hooks.server.js

JS hooks.server.js ×

```
1 export async function handle({event, resolve}) {  
2   console.log('receive request', event.url.href)  
3   return await resolve(event)  
4 }  
5  
6 export async function init() {  
7   console.log('server init');  
8 }
```

Kode : src/hooks.client.js

JS hooks.client.js ×

```
1 export async function init() {  
2   console.info('client init');  
3 }  
4
```

```
5 export async function handleError(params) {  
6   console.error(`Ups`, params);  
7 }
```

Universal Hooks

- Universal Hooks merupakan Hooks yang akan dijalankan di Server dan di Client
- Function `reroute(param)` dijalankan sebelum `handle()`, dan bisa digunakan untuk mengubah url

Tugas

- Misal kita akan coba secara otomatis mengubah `/home` menjadi `/id/home` menggunakan `reroute()`

Kode : /src/hooks.js

JS hooks.js ×

```
1 export async function reroute({url}) {  
2     if (url.pathname === "/home") {  
3         return "/id/home"  
4     }  
5 }
```

Kode : /[lang]/home

Js +page.js ×

```
1 export async function load({params}) {  
2     const lang = params.lang;  
3  
4     const hello = {  
5         "en": "Hello",  
6         "id": "Halo",  
7         "fn": "Hola"  
8     }  
9  
10    return {  
11        hello: hello[lang]  
12    }  
13 }
```

Errors

Errors

- Sebelumnya kita sudah tahu bahwa jika terjadi error di SvelteKit, maka SvelteKit akan menampilkan komponen `+error.svelte`
- Sekarang kita akan bahas lebih detail tentang error di SvelteKit
- SvelteKit membagi error menjadi 2 jenis, `expected error` dan `unexpected error`
- `Expected error` artinya error yang memang sengaja kita buat, sedangkan `Unexpected error` merupakan error yang tidak terduga, biasanya karena ada kesalahan logic di aplikasi SvelteKit kita


Unexpected Error

- Unexpected error terjadi ketika terjadi error yang tidak terduga.
- Saat terjadi unexpected error, maka secara otomatis SvelteKit akan memanggil throw error dengan page.status 500, dan page.error berisi object :
`{ "message": "Internal Error" }`

Tugas

- Saat ini jika kita membuka halaman `/products/[id]`, dan jika id tidak ditemukan, maka akan terjadi Unexpected error
- Sekarang kita akan coba buat halaman error di `/products/[id]`

Kode : /products/[id]

 +error.svelte ×

```
1  <script>
2    import {page} from "$app/state";
3  </script>
4
5  <h1>Error : {page.status}</h1>
6
7  <p>Message : {page.error.message}</p>
8
```

Expected Error

- Expected error biasanya dibuat secara sengaja menggunakan function `error()`
- <https://svelte.dev/docs/kit/@sveltejs-kit#error>
- Function `error()` secara otomatis akan throw Error dan menyebabkan proses berhenti, sehingga SvelteKit secara otomatis akan menampilkan komponen `+error.svelte`

Tugas

- Kita akan coba ubah halaman `/products/[id]`, dimana jika `[id]` tidak tersedia, kita akan buat menjadi error 404

Kode : /products/[id]

JS +page.js ×

```
1 import {error} from "@sveltejs/kit";
2
3 export async function load({params, fetch}) {
4     const response = await fetch(`/api/products/${params.id}.json`);
5
6     if (response.status !== 200) {
7         error(404, {
8             message: "Product not found"
9         })
10    }
11
12    return await response.json();
13 }
```

Link Option

Link Option

- Sebelumnya kita sudah bahas bahwa SvelteKit menggunakan element `<a>` sebagai navigasi
- Secara default, link di SvelteKit memiliki opsi yang bisa kita ubah
- Jika kita perhatikan, secara default saat kita melakukan hover ke link, SvelteKit akan melakukan load data
- Ini bisa mempercepat proses perpindahan halaman sehingga tidak terjadi blank page ketika berpindah halaman
- Ada banyak opsi yang bisa kita gunakan untuk link
- <https://svelte.dev/docs/kit/link-options>

Tugas

- Silahkan coba ubah link pada `/routes/(default)/+layout.svelte`

Kode : /routes/(default)/+layout.svelte

+layout.svelte ×

```
1  <script>
2    import {page} from '$app/state';
3    const {children} = $props();
4  </script>
5
6  <p>{page.url.pathname}</p>
7
8  <ul data-sveltekit-preload-data="tap">
9    <li><a href="/">Home</a></li>
10   <li><a href="/counter">Counter</a></li>
11   <li><a href="/profile/user">Profile</a></li>
12   <li><a href="/error">Error</a></li>
13 </ul>
14
15 {@render children()}
16
```

Lib

Lib

- SvelteKit secara otomatis akan membuat semua file didalam src/lib bisa diakses menggunakan alias \$lib
- Misal kita bisa tempatkan komponen-komponen pada folder src/lib

Tugas


- Kita akan buat component `src/lib/component/Counter.svelte`
- Dan kita akan import komponen Counter tersebut di halaman `/counter`

Kode : src/lib/components/Counter.svelte

Counter.svelte ×

```
1  <script>
2    let counter = $state(0);
3
4    let {data} = $props()
5    console.log(data);
6
7    function increment() {
8      counter++;
9    }
10 </script>
11
12 <h1>Counter : {counter}</h1>
13
14 <button onclick={increment}>Increment</button>
```

Kode : /counter

 +page.svelte ×

```
1 <script>
2   import Counter from "$lib/components/Counter.svelte";
3 </script>
4
5 <Counter/>
6
```

Environment

Environment

- Saat kita membuat aplikasi, kita sering membuat konfigurasi yang dinamis di luar aplikasi (tidak di hardcode), dan biasanya kita simpan dalam environment
- Terdapat dua jenis data environment, static di file .env, dan dinamis yang kita set secara runtime

Static Environment

- Untuk mengakses static environment, kita bisa menggunakan dua library, untuk private environment kita bisa gunakan :
[https://svelte.dev/docs/kit/\\$env-static-private](https://svelte.dev/docs/kit/$env-static-private)
- Untuk public environment, yang diawali dengan PUBLIC_, kita bisa gunakan :
[https://svelte.dev/docs/kit/\\$env-static-public](https://svelte.dev/docs/kit/$env-static-public)

Tugas

- Kita akan coba tambahkan .env ke aplikasi lalu kita akan coba tampilkan pada init() server

Kode : .env

≡ .env ×

.i* You are editing a file that is ignored

1 DB_HOST=localhost

2 DB_PORT=3306

3

4 PUBLIC_NAME="Belajar SvelteKit"

Kode : src/hooks.server.js

JS hooks.server.js ×

```
1 import {DB_HOST, DB_PORT} from "$env/static/private";
2 import {PUBLIC_NAME} from "$env/static/public";
3
4 export async function init() {
5   console.log('server init');
6   console.log(DB_HOST)
7   console.log(DB_PORT)
8   console.log(PUBLIC_NAME)
9 }
10
```

Dynamic Environment

- Berbeda dengan static environment, dynamic environment biasanya di set secara runtime ketika aplikasi dijalankan
- Sama seperti static environment, kita bisa buat private dan public
- Untuk private dynamic environment, kita bisa gunakan :
[https://svelte.dev/docs/kit/\\$env-dynamic-private](https://svelte.dev/docs/kit/$env-dynamic-private)
- Untuk public dynamic environment, kita bisa gunakan :
[https://svelte.dev/docs/kit/\\$env-dynamic-public](https://svelte.dev/docs/kit/$env-dynamic-public)

Tugas

- Sebelum menjalankan npm run dev, kita akan coba tambahkan environment variable menggunakan perintah :
export HELLO=Eko
export PUBLIC_HELLO=Eko
- Jika menggunakan Windows, kita tidak menggunakan perintah export, melainkan menggunakan set

Kode : src/hooks.server.js

JS hooks.server.js ×

```
1 import {DB_HOST, DB_PORT} from "$env/static/private";
2 import {PUBLIC_NAME} from "$env/static/public";
3 import { env } from '$env/dynamic/private';
4 import { env as publicEnv } from '$env/dynamic/public';
5
6 export async function init() {
7   console.log('server init');
8   console.log(DB_HOST)
9   console.log(DB_PORT)
10  console.log(PUBLIC_NAME)
11  console.log(env.HELLO)
12  console.log(publicEnv.PUBLIC_HELLO)
13 }
```


Server Only Modules

Server Only Modules

- Hal yang kadang membingungkan saat menggunakan SvelteKit adalah, ketika kita membuat kode yang bisa digunakan di Server dan Client
- Oleh karena itu, jika kita ingin membuat kode JavaScript yang hanya bisa digunakan di Server, jangan lupa tambahkan nama `.server` pada file tersebut
- Atau kita bisa simpan di folder `src/lib/server`

Tugas

- Kita akan coba tambahkan kode contoh misal koneksi ke database di `src/lib/server`

Kode : src/lib/server/database.js

JS database.js ×

```
1 import {DB_HOST, DB_PORT} from "$env/static/private";  
2  
3 export function connect() {  
4     console.log(`connect to database : ${DB_HOST}:${DB_PORT}`)  
5 }
```

Kode : src/hooks.server.js

JS hooks.server.js ×

```
1 import {PUBLIC_NAME} from "$env/static/public";
2 import { env } from '$env/dynamic/private';
3 import { env as publicEnv } from '$env/dynamic/public';
4 import {connect} from "$lib/server/database.js";
5
6 export async function init() {
7   console.log('server init');
8   connect();
9   console.log(PUBLIC_NAME)
10  console.log(env.HELLO)
11  console.log(publicEnv.PUBLIC_HELLO)
12 }
```

Building App

Building App

- Setelah aplikasi SvelteKit yang kita buat selesai, selanjutnya kita perlu build aplikasinya agar menjadi file yang siap untuk didistribusikan
- SvelteKit menggunakan Vite, jadi untuk build aplikasinya, kita bisa gunakan perintah “vite build”, atau “npm run build”
- Vite akan membuat kode yang optimal dari kode aplikasi kita, baik itu kode server dan kode client (browser)
- Jika ada prerender, maka halaman prerender juga akan dibuat ketika proses build

Build Stage

- Perlu diperhatikan, saat proses build ini, semua kode di page dan layout, server maupun client akan di load oleh SvelteKit untuk melakukan analisis
- Oleh karena itu, jika ada kode yang seharusnya tidak di load pada waktu build, seharusnya harus kita tandai
- Contoh misal jika kita menambahkan perintah load ke database di Load Function
- Mungkin ada baiknya jangan dieksekusi pada proses build
- Kita bisa gunakan flag building untuk mengecek apakah ini proses build atau bukan
- [https://svelte.dev/docs/kit/\\$app-environment#building](https://svelte.dev/docs/kit/$app-environment#building)

Tugas

- Kita akan coba memanggil `connect()` di `database.js` pada Load Function di halaman `/counter`

Kode : /counter

JS +page.server.js ×

```
1  import {connect} from "$lib/server/database.js";
2  import {building} from "$app/environment";
3
4  if (!building) {
5      connect();
6  }
7
8  export async function load() {
9      return {}
10 }
```

Preview

- Setelah kita melakukan build aplikasi SvelteKit, kita bisa lihat hasilnya dengan cara menjalankannya menggunakan mode preview
- Kita bisa gunakan perintah “vite preview” atau “npm run preview”
- Karena preview itu akan menjalankan hasil build, maka tidak ada fitur hot reload seperti “vite run”

Tugas

- Silahkan build aplikasi yang sudah kita buat, dan coba preview hasilnya

Error : Dynamic Environment di Prerender

```
.svelte-kit/output/client/_app/immutable/chunks/QKuYQ17H.js 32.47 kB | gzip: 12.64 kB
```

```
✓ built in 330ms
```

```
server init
```

```
node:internal/event_target:1101
```

```
  process.nextTick(() => { throw err; });
```

```
    ^
```

```
Error: Cannot read values from $env/dynamic/private while prerendering (attempted to read env.HELLO). Use $env/static/private
    at Object.get (file:///Users/khannedy/Developments/SVELTE/belajar-svelte-kit/.svelte-kit/output/server/index.js:299:11)
    at Object.init (file:///Users/khannedy/Developments/SVELTE/belajar-svelte-kit/.svelte-kit/output/server/chunks/hooks.js:1:1)
    at file:///Users/khannedy/Developments/SVELTE/belajar-svelte-kit/.svelte-kit/output/server/index.js:3042:24
    at async Server.init (file:///Users/khannedy/Developments/SVELTE/belajar-svelte-kit/.svelte-kit/output/server/index.js:1:1)
    at async prerender (file:///Users/khannedy/Developments/SVELTE/belajar-svelte-kit/node_modules/@sveltejs/kit/src/runtime/server/prerender.js:1:1)
    at async MessagePort.<anonymous> (file:///Users/khannedy/Developments/SVELTE/belajar-svelte-kit/node_modules/@sveltejs/kit/src/runtime/server/prerender.js:1:1)
```

```
Node.js v22.13.0
```

```
~/Developments/SVELTE/belajar-svelte-kit git:[main]
```

Prerender Error

- Saat dibuat materi ini, SvelteKit masih tidak mendukung menggunakan dynamic environment untuk halaman prerender
- Oleh karena itu, kita harus ubah menjadi static environment, atau tidak menggunakan feature prerender
- Misal pada kasus ini, saya akan coba ubah dynamic environment menjadi static environment

Kode : src/hooks.server.js

JS hooks.server.js ×

```
1 import {PUBLIC_HELLO, PUBLIC_NAME} from "$env/static/public";
2 import {connect} from "$lib/server/database.js";
3 import {HELLO} from "$env/static/private";
4
5 export async function init() {
6   console.log('server init');
7   connect();
8   console.log(PUBLIC_NAME)
9   console.log(HELLO)
10  console.log(PUBLIC_HELLO)
11 }
```

Adapter

Adapter

- Setelah kita build aplikasi SvelteKit yang kita buat, kita tidak bisa langsung menjalannya. Dan vite preview bukanlah cara untuk menjalankan aplikasi SvelteKit di production
- Kita harus menggunakan Adapter untuk menjalankan aplikasi SvelteKit yang sudah kita buat
- Ada banyak adapter yang didukung oleh SvelteKit, tapi pada materi ini kita akan menggunakan NodeJS
- <https://svelte.dev/docs/kit/adapters>

Node Server

- Untuk menambahkan adapter NodeJS, kita perlu tambahkan dependency yang dibutuhkan :
`npm i -D @sveltejs/adapter-node`
- Selanjutnya kita perlu mengubah konfigurasi SvelteKit untuk menggunakan NodeJS Adapter
- <https://svelte.dev/docs/kit/adapter-node>

Tugas

- Ikuti tahapan pada halaman Node Server Adapter untuk menjalankan aplikasi SvelteKit yang sudah kita buat
- <https://svelte.dev/docs/kit/adapter-node>

Single Page Application

Single Page Application

- Salah satu yang biasa dilakukan ketika membuat Web Frontend adalah membuat SPA (Single Page Application)
- SvelteKit bisa digunakan untuk membuat aplikasi SPA
- Namun, syaratnya kita tidak boleh menggunakan fitur Server, jadi harus dipastikan bahwa tidak ada kode Server di project kita
- <https://svelte.dev/docs/kit/single-page-apps>

Referensi

Referensi

- <https://svelte.dev/docs/kit/introduction>
- <https://svelte.dev/tutorial/kit/introducing-sveltekit>
- <https://github.com/sveltejs/kit>

Materi Selanjutnya

Materi Selanjutnya

- Mulai membuat web frontend menggunakan Svelte
- Perbanyak studi kasus menggunakan Svelte