[FE-3] Data Fetching External API dengan Next JS

🛠️ 1. Persiapan

Untuk mengikuti pembelajaran ini kamu perlu menyiapkan:

- Install **Node JS** (https://nodejs.org/en)
- **VS Code** untuk text editor, atau editor kesukaanmu (https://code.visualstudio.com/). Pastikan menginstall Plugin:
  - **ES7+ React/Redux/React-Native snippets**
  - **Tailwind CSS IntelliSense**
  - **Prettier - Code formatter**

> Semua contoh code dapat dilihat juga di:
>
> https://github.com/anjarmath/belajar_nextjs

🚀 2. Pendahuluan

🔷2.1 Mengenal Data Fetching & API Call

Saat membangun aplikasi berbasis web, kita sering kali perlu mengambil atau mengirim data dari dan ke sumber eksternal seperti database atau REST API. **Data fetching** adalah proses mengambil data ini dan menampilkannya di aplikasi.

🔑 **Konsep penting:**

- **Client-Side Fetching:** Data diambil di browser setelah halaman dimuat. Biasanya menggunakan `fetch()` atau `useEffect()`.
- **Server-Side Fetching:** Data diambil langsung di server sebelum halaman dikirim ke pengguna, sehingga lebih cepat dan SEO-friendly.

🔷 2.2 Mengenal Server Action

Next.js 14 memperkenalkan **Server Actions**, fitur baru yang memudahkan kita untuk memanggil API langsung dari server tanpa perlu API routes tambahan.

🔥 **Keunggulan Server Actions:**

✅ Bisa dipanggil langsung dari komponen React, tanpa perlu API routes tambahan.

✅ Bisa digunakan untuk **GET** maupun **POST** request.

✅ Lebih aman karena kode tetap berjalan di server, bukan di browser.

Nantinya, kita akan menggunakan **Server Actions** untuk:

🔷 Login ke aplikasi dengan memanggil API DummyJSON.

🔷 Mengambil data pengguna yang sudah login.

> 💡 **Spoiler:** Nanti kita bakal pakai API dari **DummyJSON** sebagai sumber data! API ini menyediakan berbagai data dummy seperti pengguna, produk, postingan, dan lainnya—cocok untuk latihan tanpa perlu setup backend sendiri.

🔑 3. Implementasi Login

📝 3.1 Membuat Form Login

Kita buat halaman baru di `(auth)/login/page.jsx`

- Buat Schema

```
const formSchema = z.object({
  username: z.string().min(2, "Username harus minimal 2 karakter"),
  password: z.string().min(5, "Password harus minimal 5 karakter"),
});
```

- Panggil Hook Form dan Handler

```
const form = useForm({
    resolver: zodResolver(formSchema),
    defaultValues: {
      username: "",
      password: "",
    },
  });

const onSubmit = (data) => {
    console.log("Data Form:", data);
};
```

- Buat UI

```
<div className=" max-w-xl mx-auto px-4 py-6">
      <Form {...form}>
        <form onSubmit={form.handleSubmit(onSubmit)} className="space
-y-4">
          {/* Input Username */}
          <FormField
            control={form.control}
            name="username"
```

```
          render={({ field }) => (
            <FormItem>
              <Label>Username</Label>
              <FormControl>
                <Input {...field} placeholder="Masukkan username" />
              </FormControl>
              <FormMessage />
            </FormItem>
          )}
        />

        {/* Input Password */}
        <FormField
          control={form.control}
          name="password"
          render={({ field }) => (
            <FormItem>
              <Label>Password</Label>
              <FormControl>
                <Input
                  {...field}
                  placeholder="Masukkan Password"
                  type="password"
                />
              </FormControl>
              <FormMessage />
            </FormItem>
          )}
        />

        {/* Tombol Submit */}
        <Button type="submit">Login</Button>
      </form>
    </Form>
  </div>
```

🔄 3.2 Contoh Memanggil API Login di Client Component

Ubah handler submit menjadi seperti berikut:

```
const onSubmit = (data) => {
   fetch("https://dummyjson.com/user/login", {
     method: "POST",
     headers: { "Content-Type": "application/json" },
     body: JSON.stringify(data),
   })
     .then((res) => res.json())
     .then(console.log);
 };
```

🔄 3.2 Implementasi Memakai Server Action Login

```
export async function loginAction(user) {
  const c = await cookies();

  const res = await fetch("https://dummyjson.com/auth/login", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(user),
  });

  const data = await res.json();

  //  Jika response dari API tidak 200 OK kita kembalikan pesan errornya
  if (!res.ok) {
    return { error: data.message };
  }

  //  Kita perlu simpan token ke cookies
  c.set("accessToken", data.accessToken);
  //  Sampai sini asumsikan login berhasil, maka kita redirect langsung ke hala
man dashboard
```

```
  redirect("/dashboard");
}
```

👤 4. Implementasi Get User

🔍 4.1 Membuat Server Action Get User

```
export async function getUserAction() {
  const c = await cookies();
  //   Kita ambil token yang sebelumnya kita simpan
  const token = c.get("accessToken");

  const res = await fetch("https://dummyjson.com/user/me", {
    method: "GET",
    headers: {
      Authorization: `Bearer ${token}`,
    },
  });

  const data = await res.json();

  //   Jika response dari API tidak 200 OK kita kembalikan pesan errornya
  if (!res.ok) {
    return { error: data.message };
  }

  return { user: data };
}
```

🖥️ 4.2 Menampilkan Data User di Halaman Dashboard

Ubah file `page.jsx` untuk halaman `dashboard` untuk memanggil server action tersebut dan menampilkan dalam bentuk card

```
export default async function Page() {
  const { user, error } = await getUserAction();
  return (
    <div>
      {error}
      {user && (
        <Card className="w-full max-w-sm p-4 shadow-lg">
          <CardHeader className="flex flex-col items-center gap-2">
            <Image
              src={user.image}
              width={150}
              height={150}
              alt={user.firstName}
            />
            <h2 className="text-lg font-semibold">
              {user.firstName} {user.lastName}
            </h2>
          </CardHeader>
          <CardContent className=" space-y-2">
            <p className="text-sm text-gray-600">🗓️ {user.age} tahun</p>
            <p className="text-sm text-gray-600">📧 {user.email}</p>
          </CardContent>
        </Card>
      )}
    </div>
  );
}
```

Jika kamu mendapati error untuk menampilkan gambar, kamu perlu menampahkan domain ke option pada `next.config.mjs`

```
const nextConfig = {
  images: {
```

```
    remotePatterns: [{ hostname: "dummyjson.com" }],
  },
};
```

🏁 5. Penutup

Selamat! 🎉 Kamu telah berhasil memahami dan mengimplementasikan cara **memanggil external API dengan Next.js 14**, mulai dari **data fetching**, **menggunakan Server Action**, hingga **menampilkan data user** di UI.