**Creating Records**

1. Create a prisma client to access our database.
2. Create a form in SnippetCreatePage
3. Define a *Server Action*. This is a function that will be called when the form is submitted.
4. In the Server Action, validate the users input then create a new snippet.
5. Redirect the user to the Home Page, which lists all the snippets.

**Create a prisma client to access our database.**

Create a new folder in src folder: db

Create a new file in db folder: index.ts

src 🡪 db 🡪 index.ts

import { PrismaClient } from "@prisma/client";

export const db = new PrismaClient();

**Create a form in SnippetCreatePage**

src 🡪 app 🡪 snippets 🡪 new 🡪 page.tsx

export default function SnippetCreatePage() {

    return (

        <form>

            <h3 className="font-bold m-3">Create a Snippet</h3>

            <div className="flex flex-col gap-4">

                <div className="flex gap-4">

                    <label className="w-12" htmlFor="title">Title</label>

                    <input

                        name="title"

                        className="border rounded p-2 w-full"

                        id="title"

                    />

                </div>

            </div>

            <div className="flex flex-col gap-4">

                <div className="flex gap-4">

                    <label className="w-12" htmlFor="code">Code</label>

                    <textarea

                        name="code"

                        className="border rounded p-2 w-full"

                        id="code"

                    />

                </div>

                <button type="submit" className="rounded p-2 bg-blue-200">

                    Create

                </button>

            </div>

        </form>

    )

}

**Define a *Server Action*. This is a function that will be called when the form is submitted.**

**Server Action**

* Number one way to change data (create, update, delete records) in a next app.
* Super close integration with HTML forms.
* Server actions are functions that will be called with the values a user entered into a form.

src 🡪 app 🡪 snippets 🡪 new 🡪 page.tsx

import { redirect } from "next/navigation";

import { db } from "@/db";

export default function SnippetCreatePage() {

    // To use this createSnippet function, place it in the FORM ACTION below.

    async function createSnippet(formData: FormData) {

        // This needs to be a SERVER ACTION

        'use server';

        // Check the user's inputs and make sure they are valid

        const title = formData.get('title') as string; // From the Name attribute of form in return statement below.

        const code = formData.get('code') as string;  // From the NAME attribute of form in the return statement below.

        // Createa new record in the database

        const snippet = await db.snippet.create({

            data: {

                title: title,

                code: code

            }

        })

        // Redirect the user back to the root route

        redirect('/');  // Redirect back to the HOME screen.

    }

    return (

        <form action={createSnippet}>

**In the Server Action, validate the users input then create a new snippet.**

**Redirect the user to the Home Page, which lists all the snippets.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Fetching Data**

* Create a *server component* – component that doesn’t have ‘use client’ at the top.
* Mark the component as ‘async’
* Make an HTTP request or directly access a database to get your data.
* Render your data directly or pass it to a child component.

**Create a server component – component that doesn’t have ‘use client’ at the top.**

src 🡪 app 🡪 page.tsx

export default function Home() {

  return (

    <div>Home Page</div>

  )

}

**Mark the component as ‘async’**

src 🡪 app 🡪 page.tsx

export default async function Home() {

  return (

    <div>Home Page</div>

  )

}

**Make an HTTP request or directly access a database to get your data.**

In this case, we are not going to make an HTTP request.

We are going to access or database (db) to get our data.

src 🡪 app 🡪 page.tsx

import { db } from "@/db";

export default async function Home() {

  const snippets = await db.snippet.findMany();

  return (

    <div>Home Page</div>

  )

}

**Render your data directly or pass it to a child component.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Dynamic Paths**

Create a folder in the snippets folder: [id]

The [id] folder is a wildcard. It allows us to use dynamic paths.

src 🡪 app 🡪 snippets 🡪 [id]

**Now we will go through the Fetching Data process again:**

**Create a server component – component that doesn’t have ‘use client’ at the top.**

Now create a page.tsx file in the [id] folder

src 🡪 app 🡪 snippets 🡪 [id] 🡪 page.tsx

export default function SnippetShowPage(props: any) {

    return (

        <div>Show a snippet</div>

    )

}

**Mark the component as ‘async’**

src 🡪 app 🡪 snippets 🡪 [id] 🡪 page.tsx

export default async function SnippetShowPage(props: any) {

    return (

        <div>Show a snippet</div>

    )

}

**Make an HTTP request or directly access a database to get your data.**

src 🡪 app 🡪 snippets 🡪 [id] 🡪 page.tsx

import { db } from "@/db";

interface SnippetShowPageProps {

    params: {

        id: string

    }

}

export default async function SnippetShowPage(props: SnippetShowPageProps) {

    const snippet = await db.snippet.findFirst({

        where: { id: parseInt(props.params.id) }

    })

    return (

        <div>Show a snippet</div>

    )

}