cloud connect

A cloud native company

Latihan Aplikasi Moden

Jabatan Digital Negara

9 - 12 September 2024

cloud connect

Material Latihan:
https://bit.ly/jdn-lpam

Google Meet:
https://bit.ly/jdn-lpam-meet1

Latihan Hari Keempat

Hari Keempat	
Sesi	Topik
Sesi 1	Ringkasan Latihan Hari Ketiga
Sesi 2	Pembangunan aplikasi web *front-end* menggunakan Nuxt.js versi 3: • *State management with Pinia* • *Environment Variables* • *Server Routes*
Sesi 3	Pengenalan kepada Axios
Sesi 4	*Consuming API Data endpoint using Nuxt 3 and Axios*
Sesi 5	Rumusan Latihan

Ringkasan Latihan Hari Ketiga

cloud connect

Hari Ketiga	
Sesi	Topik
Sesi 1	Ringkasan Latihan Hari Ke Kedua
Sesi 2	Pengenalan kepada Vue.js
Sesi 3	Pengenalan kepada Modern UI Framework
Sesi 4	Pembangunan UI Framework
Sesi 5	Pengenalan kepada Nuxt.js
Sesi 6	Pembangunan aplikasi web front-end menggunakan Nuxt.js versi 3: • Routing, Client-side / Server-side / Universal Rendering, Pages, Layouts, Components

cloud connect

State Management using Pinia

Apa itu *State Management*?

cloud connect

- Vue state management refers to the **process of managing the state or data** of a Vue.js application.
- In Vue.js, **the state is the data that represents the current state of the application**, including **user input, server responses, and other variables** that may change over time.
- Vue state management is an **important aspect of building scalable and maintainable** Vue.js applications.
- Vue state management:
  - Vuex
  - Pinia

Apa itu *Pinia*?

cloud connect

- **Pinia** is a **state management** library for Vue.js applications. A simple and reactive way to manage the state of your application, making it easier to build complex and scalable applications.
- Pinia uses the **composition API** introduced in Vue.js 3 to create **stores**, which are objects that hold the application's state and provide methods to update it. With Pinia, you can create multiple stores to manage different parts of your application's state, and easily share state between components.
- One of the advantages of using Pinia is that it **offers type safety and auto-completion for your state, mutations, and actions**, which can save you time and reduce the chances of introducing errors.
- Overall, Pinia provides a simple and effective way to manage the state of your Vue.js applications, helping you build scalable and maintainable code.

- Pinia also uses the **state**, **getters**, and **actions** concepts, which are equivalent to data, computed, and methods in components:
  - The **state** is defined as a function returning the initial state
  - The **getters** are functions that receive the state as a first argument
  - The **actions** are functions that can be asynchronous

- You can think of **state** as the **data** of the store, **getters** as the **computed properties** of the store, and **actions** as the **methods**.

Pinia

cloud connect

stores/config.js

```
export const useConfigStore = defineStore(
'ConfigStore',{
    state: () => {
        return {
          strapiURL: "http://abc.com:1337",
          identifier: "nama",
          password: "katalaluan",
          jwttoken: null,
        };
    },
    getters: {},
    actions: {},
});
```

Declare Pinia State Management

pages/myvuefile.vue

```
<script setup>
import { useConfigStore } from '~~/stores/config';

const configStore = useConfigStore();

// read
const strapiURL = configStore.strapiURL
const identifier = configStore.identifier
const katalaluan = configStore.password

// update
let configStore.jwttoken = "mytoken"



</script>
```

Using Pinia State Management

cloud connect

{ Latihan 10.7 - *State Management with Pinia* }

cloud connect

Environment Variables in Nuxt

Apa itu *Environment Variables?*

cloud connect

- An **environment variable** is a variable whose value is set outside the program, typically through functionality built into the operating system or microservice.
- An environment variable is made up of a name/value pair, and any number may be created and available for reference at a point in time.
- Use cases:
  - Execution mode (e.g., production, development, staging, etc.)
  - Domain names
  - API URL/URI's
  - Public and private authentication keys (only secure in server applications)
  - Group mail addresses, such as those for marketing, support, sales, etc.
  - Service account names

Environment Variables in Nuxt

cloud connect

- Declare publicly via runtimeConfig in nuxt.config.ts config file.
- Two types
  - Private – visible only during server-side
  - Public – visible only at the client-side
- Can be declared via -
  - .env file
  - Host Server Environment Variables

{ Latihan 10.8 – *Environment Variables* }

Nuxt Server

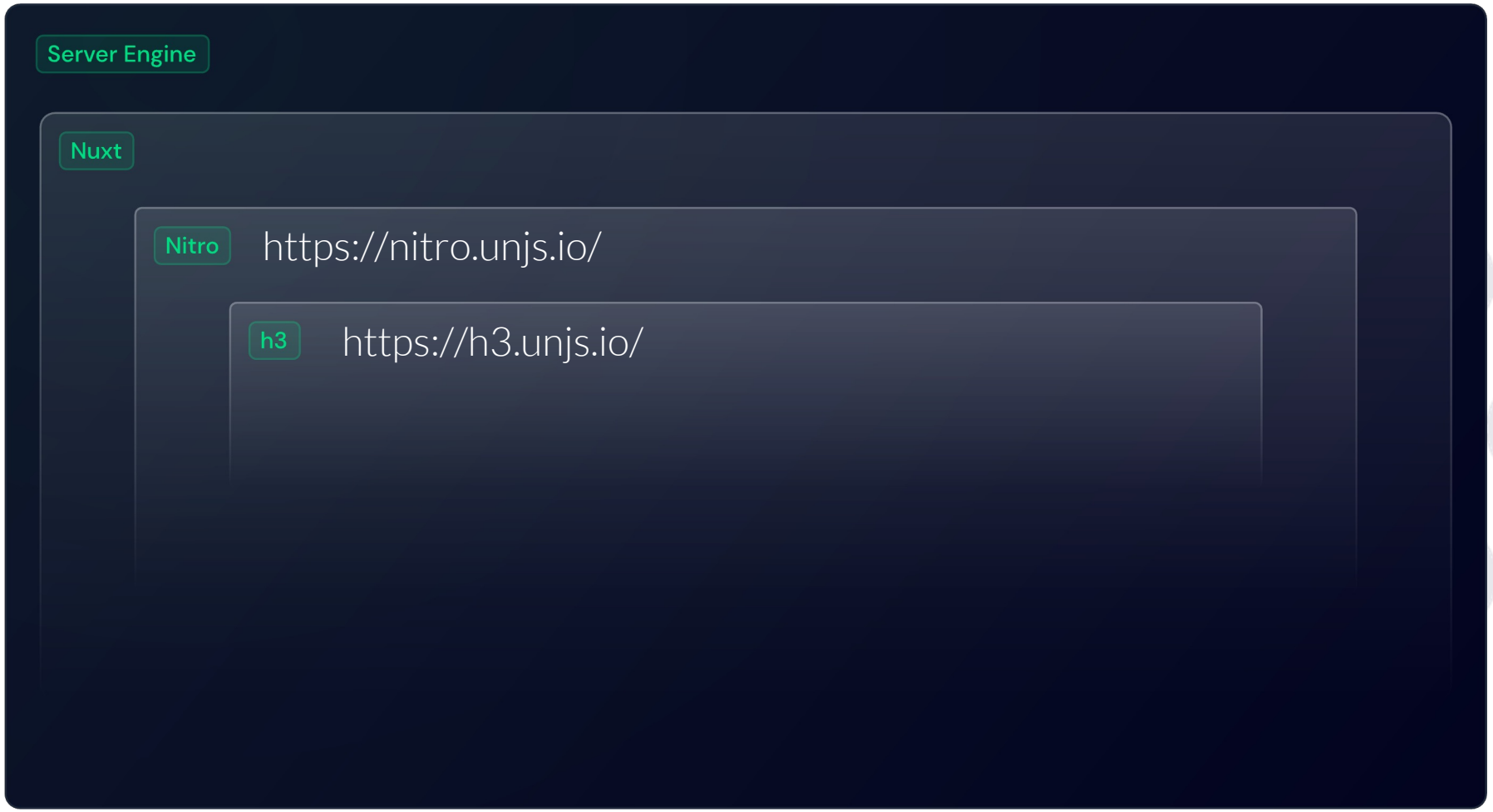
Nuxt Server

cloud connect

- Nuxt 3 is powered by a new server engine, Nitro
- Features:
  - Cross-platform support for Node.js, browsers, service workers and more.
  - Serverless support out-of-the-box.
  - API routes support.
  - Automatic code-splitting and async-loaded chunks.
  - Hybrid mode for static + serverless sites.
  - Development server with hot module reloading.

Nuxt Server Engine

cloud connect



Nuxt Server

cloud connect

Directory structure

```
-| server/
---| api/
-----| hello.ts      # /api/hello
---| routes/
-----| bonjour.ts    # /bonjour
---| middleware/
-----| log.ts        # log all requests
```

Nuxt Server

cloud connect

Server	Descriptions
Server Routes	• Files inside the ~/server/api are automatically prefixed with /api in their route. • To add server routes without /api prefix, put them into ~/server/routes directory.
Server Middleware	• Nuxt will automatically read in any file in the ~/server/middleware to create server middleware for your project. • Middleware handlers will run on every request before any other server route to add or check headers, log requests, or extend the event's request object.
Server Plugins	• Nuxt will automatically read any files in the ~/server/plugins directory and register them as Nitro plugins. This allows extending Nitro's runtime behavior and hooking into lifecycle events.

Nuxt Server

cloud connect

- *Route Parameters*
  - *Server routes can use dynamic parameters within brackets in the file name like /api/hello/[name].ts and be accessed via event.context.params*

TS server/api/hello/[name].ts

```ts
export default defineEventHandler((event) => {
  const name = getRouterParam(event, 'name')

  return `Hello, ${name}!`
})
```

Nuxt Server

cloud connect

- *HTTP Methods*
  - *Handle file names can be suffixed with .get, .post, .put, .delete, ... to match request's HTTP Method.*

TS server/api/test.get.ts

```ts
export default defineEventHandler(() => 'Test get handler')
```

TS server/api/test.post.ts

```ts
export default defineEventHandler(() => 'Test post handler')
```

  - *can also use **foo.[method].ts** inside a directory for structuring your code differently, this is useful to create API namespaces*

TS server/api/foo/index.get.ts | TS server/api/foo/index.post.ts | TS server/api/foo/bar.get.ts

```ts
export default defineEventHandler((event) => {
  // handle GET requests for the `api/foo` endpoint
})
```

Nuxt Server

cloud connect

Params

TS server/api/hello/[name].ts

```ts
export default defineEventHandler((event) => {
  const name = getRouterParam(event, 'name')

  return `Hello, ${name}!`
})
```

Body

TS server/api/submit.post.ts

```ts
export default defineEventHandler(async (event) => {
  const body = await readBody(event)
  return { body }
})
```

Querystring

TS server/api/query.get.ts

```ts
export default defineEventHandler((event) => {
  const query = getQuery(event)

  return { a: query.foo, b: query.baz }
})
```

{ Latihan 10.9 – *Nuxt Server Routes* }

cloud connect

Using Axios to Consume API

Apa itu *Axios*?

cloud connect

- *Axios is a promise-based HTTP client that works both in the browser and in a Node.js environment*
- *Small 2KB*
- *Supports older browsers*
- *Supports the Promise API*
- *Supports upload progress*
- *Built-in CSRF protection and client-side support for protecting against XSRF*
- *Automatic request body serialization to:*
  - *JSON (application/json)*
  - *Multipart / FormData (multipart/form-data)*
  - *URL encoded form (application/x-www-form-urlencoded)*
- *Posting HTML forms as JSON*
- *Automatic JSON data handling in response*

Penggunaan Axios - Request

cloud connect

Method1: Axios request

```
axios.post('/user',
{
  nama: Hanafiah',
  emel: 'hanafiah@cloud-connect.asia'
})
.then(function (response)
{
  console.log(response)
})
.catch(function (error)
{
  // handle error
  console.log(error);
})
```

Method 2: Axios instance & custom config

```
axios({
  method: 'post',
  url: '/user',
  data: {
        nama: Hanafiah',
        emel: hanafiah@cloud-connect.asia'
  }
})
.then(function (response)
{
  console.log(response)
})
.catch(function (error)
{
  // handle error
  console.log(error);
})
```

Penggunaan Axios - Response

cloud connect

Berikut adalah ***response*** setiap panggilan ***Axios***

```
{ // `data` is the response that was provided by the server
  data: {},
  // `status` is the HTTP status code from the server response
  status: 200,
  // `statusText` is the HTTP status message from the server response
  // As of HTTP/2 status text is blank or unsupported.
  // (HTTP/2 RFC: https://www.rfc-editor.org/rfc/rfc7540#section-8.1.2.4)
  statusText: 'OK',
  // `headers` the HTTP headers that the server responded with
  // All header names are lower cased and can be accessed using the bracket notation.
  // Example: `response.headers['content-type']`
  headers: {},
  // `config` is the config that was provided to `axios` for the request
  config: {},
  // `request` is the request that generated this response
  // It is the last ClientRequest instance in node.js (in redirects)
  // and an XMLHttpRequest instance in the browser
  request: {}
}
```

cloud connect

{ Latihan 10.10 – *Using Axios to Consume API* }

{ Rumusan Latihan }

Latihan Hari Pertama

cloud connect

Hari Pertama	
Sesi	Topik
Sesi 1	Pengenalan Latihan
Sesi 2	Pengenalan kepada teknologi aplikasi moden & Jamstack Framework
Sesi 3	Pengenalan kepada Microservices & API
Sesi 4	Pengenalan kepada Fastify
Sesi 5	Fastify Server, Logging, Environment Variables, Errors, Routes

Latihan Hari Kedua

cloud connect

HARI KEDUA	
Sesi	Topik
Sesi 1	Ringkasan Latihan Hari Pertama
Sesi 2	Fastify Validation & Serialization
Sesi 3	Pengenalan kepada PrismaORM
Sesi 4	Pembangunan API untuk MySQL Database sediada
Sesi 5	Pembangunan API untuk MySQL Database baru
Sesi 6	Pengenalan kepada Jamstack Framework
Sesi 7	Pengenalan kepada Vue.js

Latihan Hari Ketiga

cloud connect

Hari Ketiga	
Sesi	Topik
Sesi 1	Ringkasan Latihan Hari Ke Kedua
Sesi 2	Pengenalan kepada Vue.js
Sesi 3	Pengenalan kepada Modern UI Framework
Sesi 4	Pembangunan UI Framework
Sesi 5	Pengenalan kepada Nuxt.js
Sesi 6	Pembangunan aplikasi web front-end menggunakan Nuxt.js versi 3: • Routing, Client-side / Server-side / Universal Rendering, Pages, Layouts, Components

Latihan Hari Keempat

Hari Keempat	
Sesi	Topik
Sesi 1	Ringkasan Latihan Hari Ketiga
Sesi 2	Pembangunan aplikasi web *front-end* menggunakan Nuxt.js versi 3: • *State management with Pinia* • *Environment Variables* • *Server Routes*
Sesi 3	Pengenalan kepada Axios
Sesi 4	*Consuming API Data endpoint using Nuxt 3 and Axios*
Sesi 5	Rumusan Latihan

{ TamaT }