Design Pattern in Golang

Design Patterns adalah konsep umum untuk memecahkan masalah pada development suatu app. Ini seperti blueprint / cetakan yang bisa kita customize untuk solving error pada code kita.

Setiap design pattern memiliki use case nya masing masing meskipun ada juga yang mirip mirip. Tujuan utama dari design pattern ini adalah untuk memudahkan kolaborasi dengan developer lain dan mempercepat proses development.

Dalam sesi ini ada 10 design pattern yaitu, 1 Structural, 4 Behavioral, dan 5 Creational

1. Adapter (Structural)

Digunakan ketika kita ingin menghubungkan 2 object yang berbeda implementasinya.

How To:

* Buat struct adaptee dengan fungsionalitas yang diinginkan
* Buat struct adapter yang mengimplementasi interface client yang ingin di adapt namun implementasi nya hanya meneruskan request ke struct adaptee

1. State (Behavioral)

Digunakan ketika kita ingin mengubah perilaku suatu object sesuai state object tersebut.

How To:

* Buat interface state
* Per state dibuat struct nya dan implement interface state
* Buat struct client dimana client memiliki field state
* Buat method di struct client untuk menggunakan field state

1. Strategy (Behavioral)

Digunakan ketika kita memiliki multiple algorithms untuk tugas yang spesifik dan client dapat menentukan implementasi mana yang mau dipakai saat runtime

Struktur code nya mirip dengan state, namun ada perbedaan nya yaitu, state biasanya aware dengan keberadaan state lainnya bahkan biasanya ada transisi pergantian state. sedangkan strategi tidak tahu keberadaan strategi lainnya

How To:

* Buat interface strategy
* Per strategy dibuat struct nya dan implement interface strategy
* Buat struct client dimana client memilikii field strategy
* Buat method di struct client untuk menggunakan field strategy

1. Observer (Behavioral)

Digunakan ketika kita ingin memberitahu / notify object object yang subscribe bahwa ada sesuatu atau update pada object utama

How To:

* Buat interface Publisher
* Buat interface Subscriber
* Buat implementasi dari publisher
* Buat implementasi dari subscriber

1. Prototype (Creational)

Digunakan ketika kita ingin membuat clone object tersebut (deepcopy)

How To:

* Buat interface model yang ingin dibuat (pastikan ada func clone)
* Implementasi method clone dimana method tersebut membuat object baru dengan data dari object yang di clone

1. Singleton (Creational)

Digunakan untuk membuat variable global, memastikan object tersebut hanya ada satu, contohnya object config

How To:

* Buat variable global
* Initiate data / instance lalu assign ke variable global

1. Factory (Creational)

Digunakan untuk membuat suatu object hanya dengan memberi tahu factory instance apa yang perlu dibuat tanpa mengetahui logic pembuatan instance tersebut.

How To:

* Buat interface berisi method yang akan dijalankan di factory
* Buat struct / object yang akan dibuat lewat factory
* Buat struct factory, lalu buat method untuk membuat object

1. Object pool (Creational)

Digunakan untuk melimit maksimal object bisa dibuat / active karna object tersebut merupakan object yang berat atau mahal sehingga jika tidak dilimit bisa membuat memory usage tinggi, contohnya limit goroutine dalam memproses sesuatu dalam satu waktu

How To:

* Buat struct / object yang mau di pool
* Buat objectpool dimana memiliki field object yang sudah kita buat sebelumnya

1. Builder (Creational)

Digunakan ketika kita ingin membuat sebuah object yang kompleks, object dibuat step by step sehingga lebih mudah dibaca / dimengerti dan bisa di reuse untuk membuat object yang lain

How To:

* Buat struct utama dengan field field yang akan dipakai di builder
* Buat struct builder memiliki field struct yang sudah dibuat sebelumnya
* Implementasi setter pada setiap field di struct utama
* Implementasi method build pada struct builder yang berfungsi untuk membuild object

1. Chain of Responsibility (Behavioral)

Design pattern yang memungkinkan untuk suatu handler melanjutkan request ke handler berikutnya atau tidak. Contoh nya seperti middleware pada REST API

How To:

* Buat app rest api
* Buat func middleware, lalu gunakan middleware tersebut di setiap incoming request