Praktikum Pemrograman Web

1. Design Pattern

Adalah solusi yang dioptimalkan dan dapat diulang untuk masalah yang sering terjadi dalam pengembangan perangkat lunak. Penerapan design pattern membantu meningkatkan keterbacaan kode, memfasilitasi pemeliharaan dan mempromosikan rekayasa perangkat lunak yang baik.

Jenis Design Pattern

1. Creational Patterns

Kumpulan design pattern yang fokus pada proses penciptaan objek. Tujuannya adalah untuk menyembunyikan kompleksitas pembuatan objek, mengisolasi sistem dari cara objek ^{2x} diciptakan, dikombinasikan dan direpresentasikan.

2. Structural Patterns

Kumpulan design yang berfokus pada komposisi struktur kelas dan objek. Tujuannya adalah untuk menyusun objek dan kelas ke dalam struktur yang lebih besar dengan cara yang lebih fleksibel dan efisien.

3. Behavioral Patterns

Kumpulan design pattern yang fokus pada tata cara dan tanggung jawab distribusi antara objek-objek yang berkerjasama. Menggambarkan bagaimana objek-objek tersebut berkomunikasi, berinteraksi dan bertanggung jawab terhadap perilaku bersama.

4. Architectural Patterns.

Pola-pola design tingkat tinggi yang memberikan panduan terhadap organisasi struktural dan fungsional suatu sistem perangkat lunak.

Keuntungan menggunakan Design Pattern.

1. Dapat digunakan kembali
2. Kemampuan menambah kapasitas.
3. Kemudahan pemeliharaan.
4. Fleksibilitas.

5. Keterbacaan kode yang meningkat.
6. Peningkatan kualitas code.
7. Peningkatan Produktivitas.
8. Komunitas Pengembang yang terstandardisasi.

MVC (Model - View - Controller) Design Pattern.

Adalah Pola desain perangkat lunak yang digunakan untuk mengorganisir struktur kode dalam pengembangan aplikasi. Dalam MVC (aplikasi dibagi menjadi 3, yaitu :

1. Model.

Model mewakili data dan logika bisnis aplikasi. Ini bertanggung jawab untuk mengelola data, melakukan validasi dan menyediakan metode untuk mengakses atau memanipulasi data. Model ini terisolasi dari tampilan dan kontrol, memungkinkan perubahan dalam logika bisnis tanpa memengaruhi tampilan.

2. View.

View bertanggung jawab untuk menampilkan data kepada pengguna. Ini menggambarkan antarmuka pengguna dan menerima input dari pengguna. Tampilan mendapatkan informasi yang diperlukan untuk ditampilkan dari model, tetapi tidak memiliki pengetahuan tentang bagaimana data tersebut diperoleh atau diproses.

3. Controller

Controller bertindak sebagai perantara antara model dan view. Ini mengelola alur kontrol aplikasi, menerima input dari pengguna melalui tampilan dan memperbarui model sesuai dengan tindakan yang dilakukan pengguna. Controller juga dapat memperbarui tampilan ketika model berubah. Dengan memisahkan kontrol dari tampilan, aplikasi menjadi lebih modular dan mudah dipelihara.

Proses kerja umumnya :

1. Pengguna berinteraksi dengan View
2. View mengirim input ke Controller.
3. Controller memproses input, memperbarui model, dan memilih tampilan yang sesuai untuk menampilkan hasil.
4. View menampilkan data dari model yang telah diperbarui.