Pertemuan 14: Design Patterns (Pola Desain)

Overview

Pertemuan ini membahas Design Patterns yang paling umum digunakan dalam PHP OOP, termasuk Creational, Structural, dan Behavioral patterns dengan implementasi praktis.

Learning Objectives

Setelah mengikuti pertemuan ini, peserta akan mampu:

- Memahami konsep dan kegunaan Design Patterns
- Mengimplementasikan Creational Patterns (Singleton, Factory, Builder)
- Menggunakan Structural Patterns (Adapter, Decorator, Facade)
- Menerapkan Behavioral Patterns (Observer, Strategy, Command)
- Memilih pattern yang tepat untuk kebutuhan spesifik
- Mengkombinasikan multiple patterns dalam aplikasi

Materi

1. Introduction to Design Patterns

Apa itu Design Patterns?

Design Patterns adalah solusi umum untuk masalah yang sering muncul dalam software design. Mereka seperti blueprint atau template yang bisa digunakan untuk menyelesaikan masalah desain tertentu.

Keuntungan Design Patterns:

- **Reusability**: Template yang bisa digunakan berulang
- **Communication**: Vocabulary umum antar developer
- **Best Practices**: Solusi yang sudah teruji
- **Maintainability**: Code yang lebih terstruktur
- **Problem Solving**: Framework untuk menyelesaikan masalah umum

Kategori Design Patterns:

1. **Creational Patterns**: Berkaitan dengan cara pembuatan objek
2. **Structural Patterns**: Berkaitan dengan komposisi class dan objek
3. **Behavioral Patterns**: Berkaitan dengan interaksi dan tanggung jawab objek

2. Creational Patterns

A. Singleton Pattern

Memastikan hanya ada satu instance dari class tertentu dan menyediakan global access point.

Kapan menggunakan:

- Database connection
- Logger
- Configuration manager
- Cache manager

Struktur:

```php
class Singleton
{
    private static ?self $instance = null;

    private function __construct() {}

    public static function getInstance(): self
    {
        if (self::$instance === null) {
            self::$instance = new self();
        }
        return self::$instance;
    }

    private function __clone() {}
    public function __wakeup() {}
}
```

B. Factory Pattern

Menyediakan interface untuk membuat objek tanpa menentukan class exact-nya.

Kapan menggunakan:

- Ketika pembuatan objek kompleks
- Ketika perlu memilih implementasi berdasarkan kondisi
- Ketika ingin memisahkan pembuatan objek dari penggunaannya

Jenis Factory:

- Simple Factory
- Factory Method
- Abstract Factory

C. Builder Pattern

Memisahkan konstruksi objek kompleks dari representasinya.

Kapan menggunakan:

- Objek dengan banyak parameter opsional
- Proses pembuatan objek yang kompleks
- Perlu membuat objek dengan konfigurasi berbeda

3. Structural Patterns

A. Adapter Pattern

Mengizinkan interface yang incompatible untuk bekerja sama.

Kapan menggunakan:

- Mengintegrasikan library/API lama dengan code baru
- Wrapper untuk external services
- Mengkonversi interface

B. Decorator Pattern

Menambahkan behavior baru ke objek secara dinamis tanpa mengubah strukturnya.

Kapan menggunakan:

- Menambah functionality secara runtime
- Alternative untuk inheritance
- Kombinasi berbagai behavior

C. Facade Pattern

Menyediakan interface sederhana untuk subsystem yang kompleks.

Kapan menggunakan:

- Menyederhanakan complex system
- Decoupling client dari subsystem
- Layered architecture

4. Behavioral Patterns

A. Observer Pattern

Mendefinisikan dependency one-to-many antar objek, sehingga ketika satu objek berubah, semua dependents-nya otomatis terupdate.

Kapan menggunakan:

- Event handling
- Model-View architecture
- Notification system

B. Strategy Pattern

Mendefinisikan family algorithms, encapsulate masing-masing, dan membuatnya interchangeable.

Kapan menggunakan:

- Multiple algorithms untuk tugas yang sama

- Conditional statements yang kompleks
- Runtime algorithm selection

C. Command Pattern

Encapsulate request sebagai objek, memungkinkan parameterisasi clients dengan requests yang berbeda.

Kapan menggunakan:

- Undo operations
- Queuing operations
- Logging operations
- Macro commands

5. Real-World Applications

E-Commerce System dengan Multiple Patterns

Implementasi sistem e-commerce yang menggunakan kombinasi berbagai design patterns:

1. **Singleton**: Database connection, Logger
2. **Factory**: Product creation, Payment processor
3. **Observer**: Order notifications, Inventory updates
4. **Strategy**: Pricing strategies, Shipping methods
5. **Decorator**: Product features, Order modifications
6. **Facade**: Payment processing, Order management

6. Pattern Selection Guidelines

Memilih Pattern yang Tepat:

1. **Identifikasi masalah**: Apa masalah spesifik yang ingin diselesaikan?
2. **Analisis konteks**: Apakah masalah ini umum terjadi?
3. **Pertimbangkan trade-offs**: Kompleksitas vs benefit
4. **Think about future**: Apakah pattern ini akan membantu scalability?

Anti-Patterns (Hindari):

- **Over-engineering**: Menggunakan pattern ketika solusi sederhana sudah cukup
- **Golden hammer**: Menggunakan satu pattern untuk semua masalah
- **Pattern abuse**: Menggunakan pattern dengan salah

7. Best Practices

Implementation Guidelines:

1. **Start simple**: Jangan langsung menggunakan pattern yang kompleks
2. **Refactor gradually**: Introduce pattern ketika dibutuhkan
3. **Document decisions**: Jelaskan kenapa pattern tertentu dipilih
4. **Consider alternatives**: Selalu evaluasi opsi lain

5. **Test thoroughly**: Pattern shouldn't break functionality

Code Quality:

- Maintain SOLID principles
- Use meaningful names
- Keep it simple and readable
- Provide comprehensive documentation
- Write unit tests

8. Advanced Topics

Pattern Combinations:

- Factory + Strategy
- Observer + Command
- Decorator + Adapter
- Builder + Factory

Modern PHP Features:

- Typed properties dalam patterns
- Attributes untuk metadata
- Match expressions untuk factory logic
- Union types untuk flexibility

Praktikum

Latihan 1: Singleton Pattern

Implementasikan Database connection manager menggunakan Singleton pattern.

Latihan 2: Factory Pattern

Buat notification factory yang dapat membuat email, SMS, dan push notification objects.

Latihan 3: Observer Pattern

Implementasikan event system dimana multiple listeners dapat merespon user actions.

Latihan 4: Strategy Pattern

Buat payment processing system dengan multiple payment methods (Credit Card, PayPal, Bank Transfer).

Latihan 5: Decorator Pattern

Implementasikan coffee shop ordering system dimana customers dapat menambah extras (milk, sugar, etc.).

Latihan 6: Comprehensive Project

Buat mini e-commerce system yang menggunakan minimal 5 design patterns yang berbeda.

Tugas

Tugas Praktik:

1. Implementasikan user authentication system menggunakan Strategy pattern untuk berbagai authentication methods
2. Buat logging system menggunakan kombinasi Singleton dan Observer patterns
3. Develop caching system dengan Factory pattern untuk berbagai cache strategies
4. Create notification center menggunakan Observer dan Command patterns

Tugas Analisis:

1. Analisis existing codebase dan identifikasi dimana design patterns bisa diterapkan
2. Bandingkan implementasi dengan dan tanpa design patterns
3. Dokumentasikan trade-offs dari setiap pattern yang digunakan

Studi Kasus

Case Study 1: Content Management System

Analisis bagaimana berbagai design patterns diterapkan dalam CMS modern:

- Factory untuk content types
- Observer untuk plugin system
- Strategy untuk themes
- Decorator untuk content filtering

Case Study 2: API Gateway

Implementasi API Gateway menggunakan:

- Facade untuk external services
- Adapter untuk protocol conversion
- Chain of Responsibility untuk request processing
- Command for API versioning

Resources Tambahan

Books:

- "Design Patterns: Elements of Reusable Object-Oriented Software" - Gang of Four
- "Head First Design Patterns" - Freeman & Freeman
- "PHP Objects, Patterns, and Practice" - Matt Zandstra

Online Resources:

- Refactoring.Guru Design Patterns
- PHP The Right Way - Design Patterns
- SourceMaking Design Patterns

PHP-Specific:

- Laravel Service Container (Dependency Injection)
- Symfony Event Dispatcher (Observer)
- Doctrine ORM (Data Mapper, Unit of Work)

Next Session Preview

Pertemuan selanjutnya (Pertemuan 15) akan membahas **"Aplikasi CRUD Lengkap"** dimana kita akan:

- Mengintegrasikan semua konsep OOP yang telah dipelajari
- Membangun aplikasi web complete dengan PHP OOP
- Implementasi MVC architecture
- Database operations dengan PDO dan prepared statements
- Form validation, security, dan error handling
- Authentication dan authorization
- File upload dan management
- Testing dan deployment

Pertemuan 15 akan menjadi culmination dari seluruh course, dimana semua konsep dari basic OOP hingga advanced patterns akan digunakan dalam real-world application!