Design Patterns em PHP 8

IXC Talk 2021

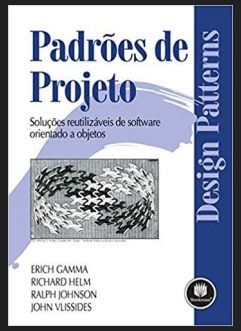
Leonardo Tumadjian

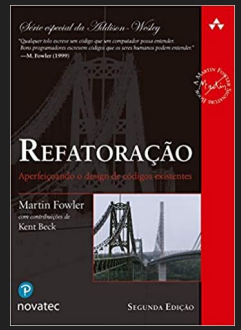
Sobre mim

- Atuação com PHP/Web desde 2009
- Tech Lead @ HeroSpark
- Evangelista PHP
- Professor
- Palestrante
- Linkedin e Youtube - youtube.com/differdev

Referências

Livros:





Sites:

https://refactoring.guru/pt-br

https://github.com/DesignPatternsPHP/DesignPatternsPHP

Sobre Design

“projeto” ou “desenho”

O que são padrões?

I see dead people!

Breve história

É uma solução geral para um problema que ocorre com frequência dentro de um determinado contexto no projeto de software

- Wikipédia

Importante:
Orientação a Objetos

1. Herança
2. Polimorfismo
3. Encapsulamento
4. Abstração

Quando usar?

Cuidado com a patternite...

Composição:
Associada e Agregada

SOLID e Testes

Temas importantes

1. Reaproveitamento
2. Organização
3. Padronização
4. Adaptação
5. Previsão
6. Antecipação
7. Substituição
8. Adição
9. Separação
10. Criação

Grupos de Design Patterns

Criacionais

Factory Method*

Abstract Factory

Builder

Prototype

Singleton (Não use diretamente)

Estruturais

Adapter

Bridge

Composite

Decorator*

Façade

Flyweight

Proxy

Comportamentais

Chain of responsability*

Command

Iterator*

Mediator

Memento

Observer*

State

Strategy*

Template Method*

Visitor

Arquiteturais

MVC

MVVM

MVP

Onion (Hexagonal, Clean Architecture, Domain Driven Design)

Muitos outros mais...

Mapa de resumo

Factory Method

Problema: Criar objetos diretamente impacta em reescrita quando fazemos alteração em composição ou parametrização.

Strategy

Problema: Programamos uma funcionalidade, mas identificamos que precisará de outra implementações completamente diferente no futuro, para não impactar mudança direta em código cliente, criamos as implementações separadas e mudamos o comportamento conforme necessário.

Template Method

Problema: Quando fazemos reutilização em uso de objetos diretos não temos como saber quais métodos são utilizados em determinada ação, por isso definir o template é importante, sendo ele uma sequência de métodos definidos por um template.

Iterator

Problema: Precisamos lidar com um conjunto de objetos de uma família de classes, sem um iterador não conseguimos acesso ao grupo de objetos, a solução é implementar uma classe que auxilie nessa comunicação.

Observer

Problema: Ao lidar com um grupo de objetos, ao decorrer da execução não conseguimos disparar ações ao conjunto de objetos sem que seja feito uma interação por um laço de repetição, ao centraliza-los, conseguimos que eles sejam comunicados por evento.

Some Live Code!

Objetivo: Implementação(simulada) de um envio de mensagens a clientes:

Email?
SMS?
Whats?
Telegram?

Obrigado!

Perguntas?

Obrigado **IXC Soft!**

Contato: tumadjian@gmail.com
@tumadjian
Leonardo Tumadjian (linkedin)
youtube.com/DifferDev