DESIGN PATTERN

Huilson José Lorenzi

Orientação a Objetos II

4° Semestre

DESIGN PATTERNS

* Também chamados de padrões de projeto, estão relacionados à arquitetura de software.
* Estão ligados aos paradigmas ao de Orientação ao Objeto.
* Eles são soluções, estruturadas e testadas, de problemas já experienciados em vários softwares desenvolvidos.

DESIGN PATTERNS

* Eles surgiram depois da elaboração de vários projetos de software, principalmente depois que os softwares começaram a ser desenvolvidos em linguagens orientadas a objetos.

DESIGN PATTERNS

* Erich Gamma, Richard Helm, Ralph Johson e John Vlissides, dividiram os design patterns em **três categorias**:
  * *Creational Patterns*: Esses tratam da construção e da referência a um objeto. Voltado a programas com amplo uso de interfaces.

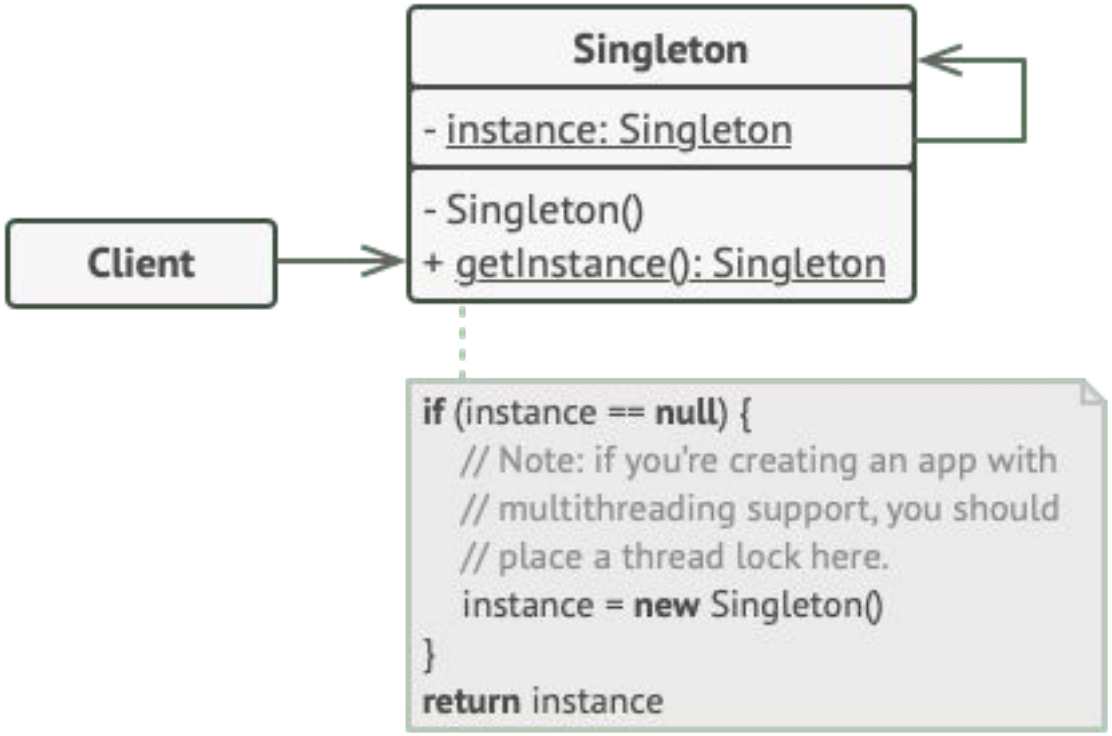
DESIGN PATTERNS

* Exemplos de *Creational Patterns*:
  * Abstract Factory.
  * Builder.
  * Factory Method.
  * Prototype.
  * **Singleton**.

DESIGN PATTERNS - SINGLETON

* **Singleton:**
  * Esse padrão garante uma única instância de uma classe,
  * Ele também provê acesso global da instância,
  * Mas como?
    * Implementar o método getInstance();
    * Construtor privado;
    * Atributo estático da própria classe do Singleton.

DESIGN PATTERNS - SINGLETON



DESIGN PATTERNS

* *Structural Patterns*: Esses tratam da relação e da interação entre os objetos, na formação de grande e complexos objetos.

* *Behavior Patterns*: Esses tratam da comunicação e da responsabilidade dada entre os objetos e do algoritmo.

DESIGN PATTERNS

* Exemplos de *Structural Patterns*:
  * **Adapter**.
  * Bridge.
  * Composite.
  * Decorator
  * Facade.
  * Flyweight.
  * Proxy.

DESIGN PATTERNS - ADAPTER

* **Adapter:**
  * Quando uma classe não implementa a mesma interface de uma classe requerida, o adapter faz essas classes trabalhar de forma conjunta.
  * Mas como?
    * Uma classe (Adapter) chama os métodos das classe inacessíveis (Adaptee) que estão sendo requerida.

DESIGN PATTERNS - ADAPTER

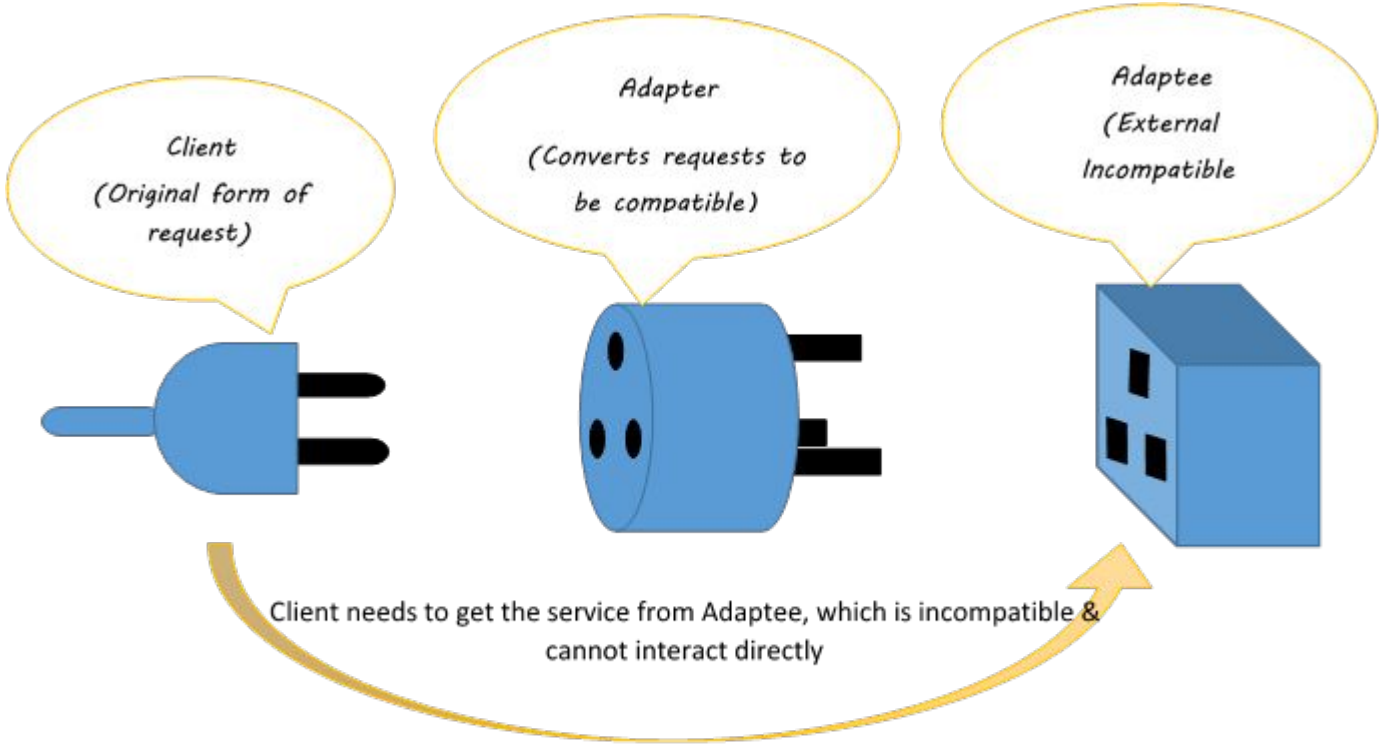
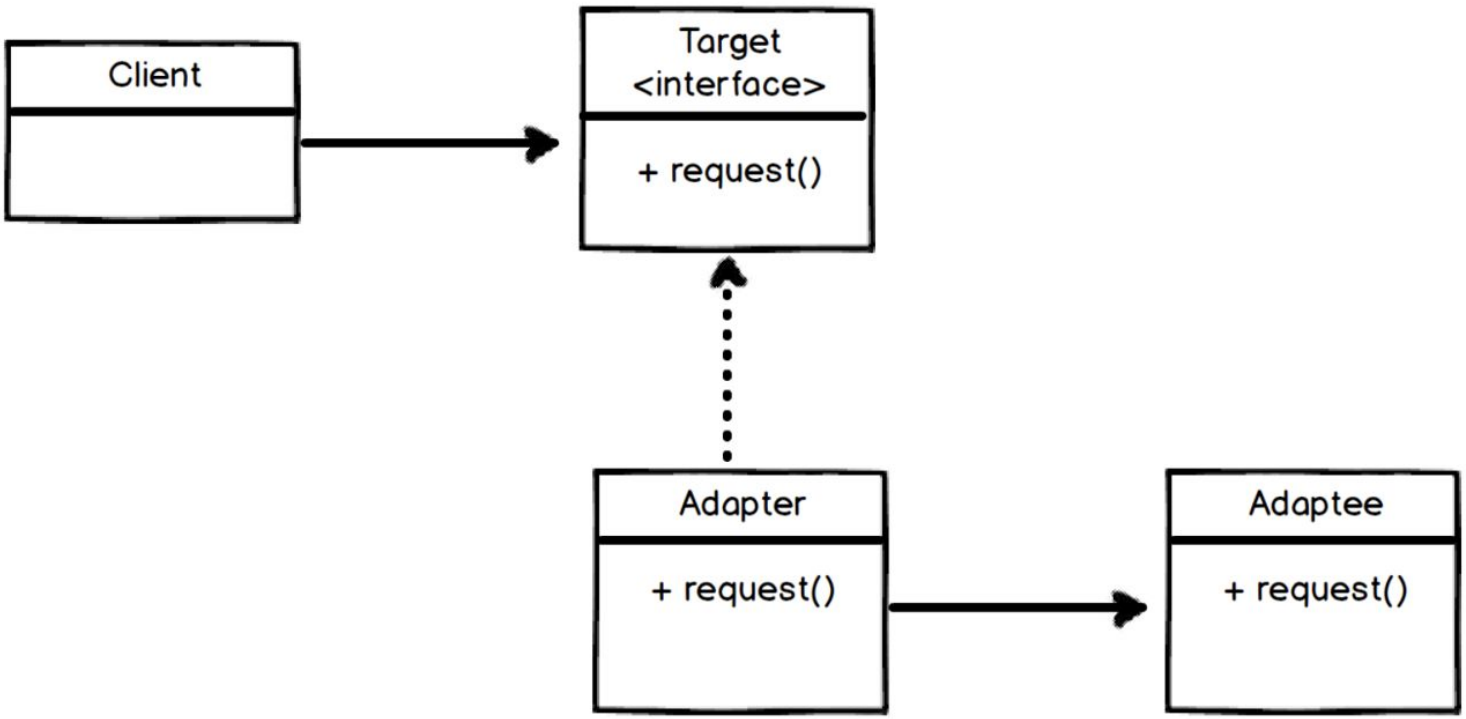


Figure 1-Adapter Pattern Concept

DESIGN PATTERNS - ADAPTER



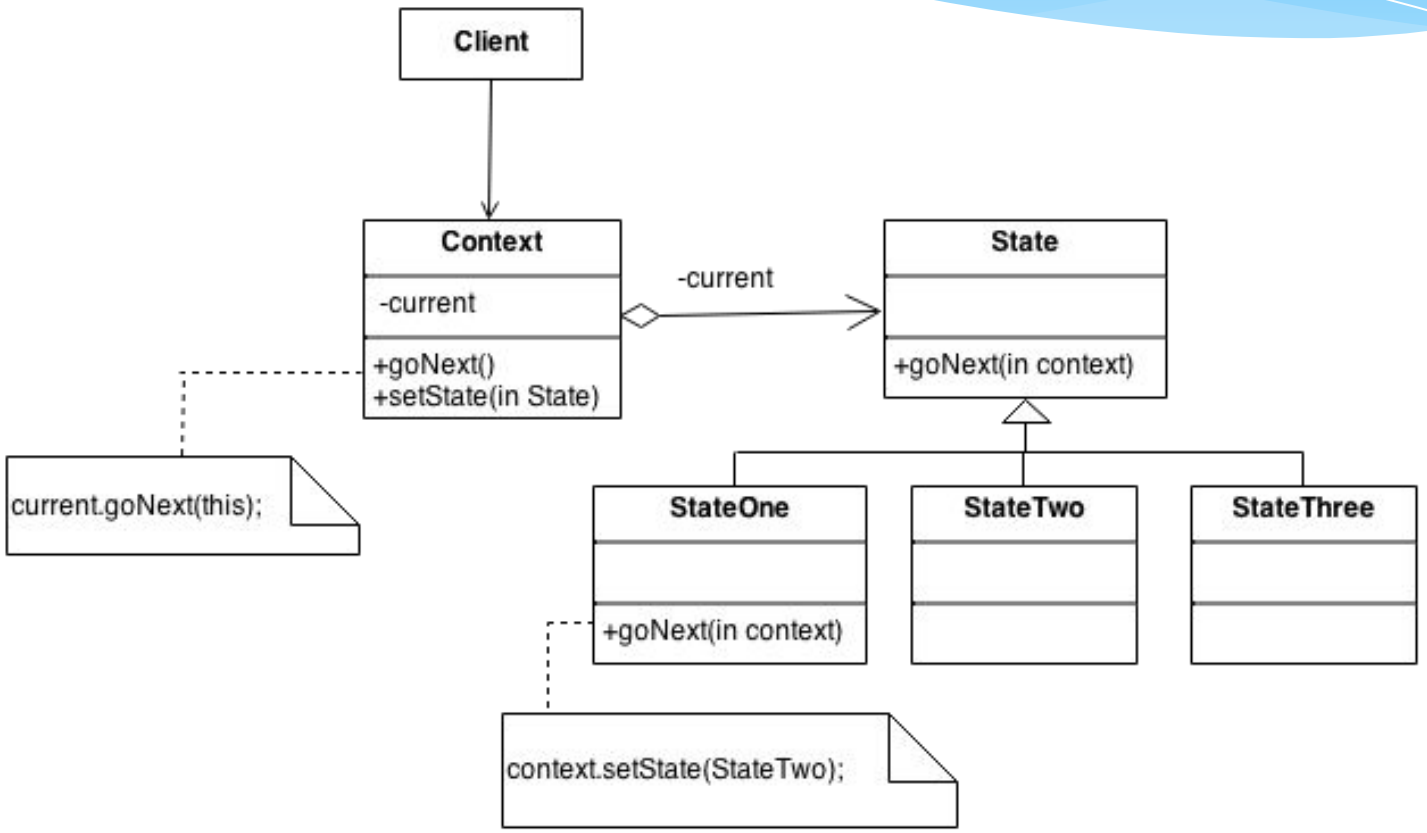
DESIGN PATTERNS

* Exemplos de *Behavior Patterns*:
  * Chain of Responsibility.
  * Command.
  * Interpreter.
  * Iterator.
  * Mediator.
  * Memento.
  * Observer.
  * **State**.
  * Strategy.
  * Template Method.
  * Visitor.

DESIGN PATTERNS - STATE

* **State:**
  * Permite que um objeto altere seu comportamento quando seu estado interno muda, como se objeto estivesse “mudando” de classe.
  * Mas como?
    * Evitando uso de condicionais.
    * As condicionais acabam virando classes.
    * Não se alteram os estados de contexto (superclasse).
    * Herança e polimorfismo são as chaves desse padrão.

DESIGN PATTERNS - STATE



DESIGN PATTERNS

* Simplificando, os design patters focam na reutilização de código, da organização da implementação, e na refatoração do projeto.

* Um bom uso de design pattern permitirá que seu software evolua de mais suavemente, torna a manutenção mais eficaz, além de padronizar e organizar melhor o código.

MVC

* A sigla vem do inglês Model-View-Controller, traduzindo, Modelo-Visão-Controle.

* Sabemos que o Padrão MVC (Model View Controller) não é um Design Pattern, mas sim padrão de arquitetura. Ele também não é um padrão em camadas, pois este diz como agrupar componentes enquanto o MVC diz como os componentes interagem. O MVC utiliza alguns padrões.

MVC

* Muito utilizado no desenvolvimento web, tendo uma abrangência enorme em diversas comunidades e empresas.

* O princípio básico do MVC é dividir a aplicação em 3 “camadas”, separando a interface das regras de negócio.

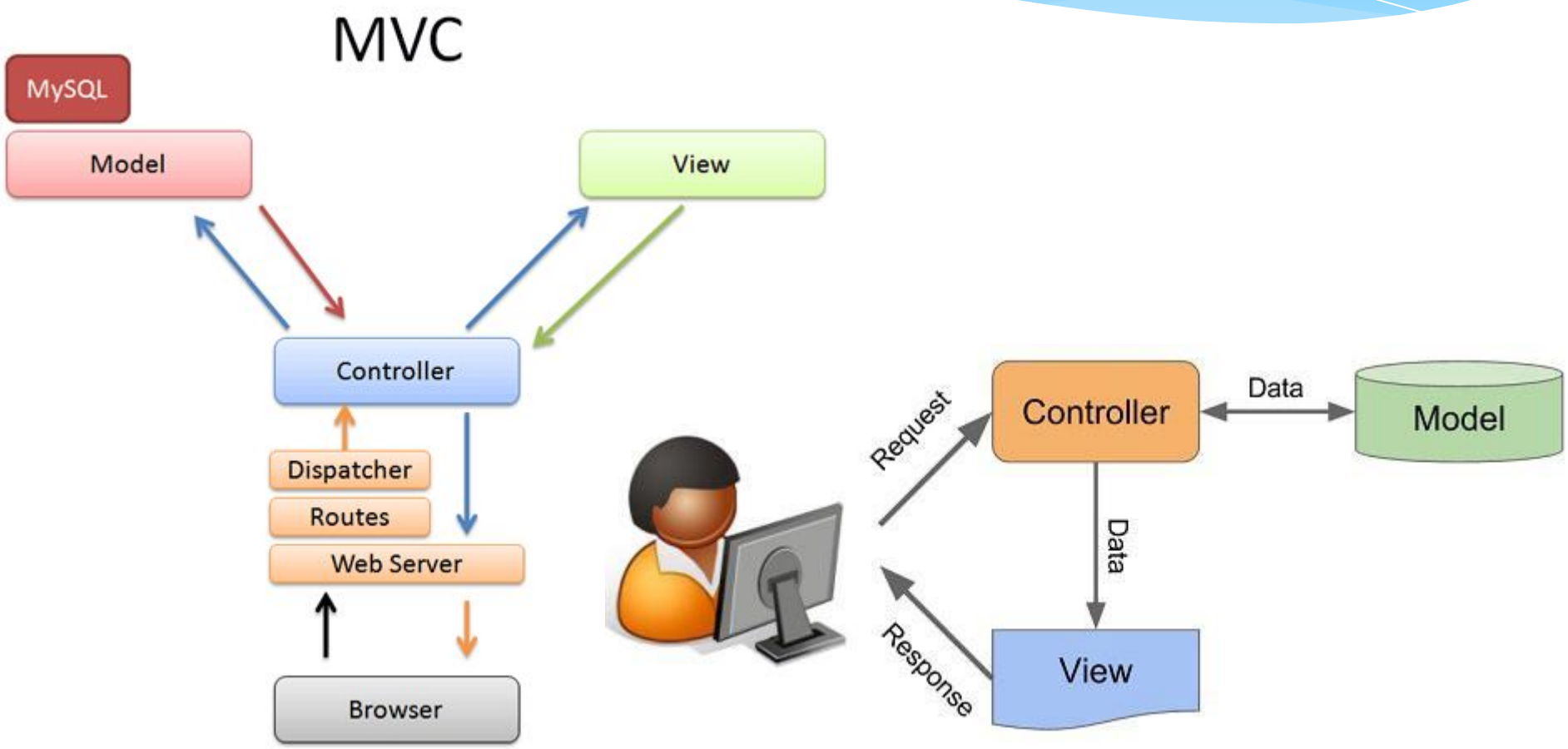
MVC

* **Model**: Essa camada é responsável pelo acesso e manipulação de dados. *Exemplo: Consultas ao BD, comunicação com aplicações externas*.

* **View**: Essa camada é responsável pela interface do usuário. *Exemplo: Páginas HTML e CSS, ou templates*.

* **Controller:** Essa camada é responsável por fazer a ligação entre o Model (busca de dados) e o View (requisições e resposta).

MVC


MVC
MySQL
Model
View
Controller
Dispatcher
Routes
Web Server
Browser
Request
Controller
Data
Model
Data
Response
View

FRAMEWORKS MVC

* Java:
  * Spring
  * Struts

* PHP:
  * Symfony
  * Laravel
  * CodeIgniter
  * CakePHP

FRAMEWORKS MVC

* Java Script:
  * React
  * Backbone
  * Angular

* C#:
  * Asp.net

FRAMEWORKS MVC

* Ruby:
  * Merb

* Python:
  * Django
  * Cherrypy

TAREFA - DESIGN PATTERN

* Crie um programa onde um usuário precisa fazer uma entrega para seu amigo que mora longe. **Regras:**
  * O pacote precisa de *nome* e *endereço* do emissor e do destinatário.
  * O pacote *só pode ser enviado* se a taxa de transporte for paga.
  * Quem define o transporte é o correio, do qual *valida* se o destinatário e emissor *existem*, e se o pagamento foi *aprovado*.
  * O usuário é *informado do estado do pacote* (enviado ou reprovado), ele deve ter uma *segunda chance* para corrigir o problema do pacote.
  * O pacote *deve demorar 1~5 minutos para ser entregue*, enquanto isso o correio *deve aceitar outros pacotes* (Dica: use threads).

REFERÊNCIA

Calebe Oliveira. **Alternativas ao MVC para aplicações Web**. *Stackoverflow*. 2014.

Felipe Leite. **Design Patterns - Conheça o Singleton**. 2018.

Leonardo Vilarinho. **MVC ¬ framework: usando a arquitetura sem código de terceiros**. 2018.

Otávio Miranda. **State Teoria.** 2020.

Vanessa Weber, Gabriel Fróes. **MVC**. 2018.

-. **Design Patterns**. 2018.

Wesley Willians. **Design Pattern Adapter**. 2020