# Desing Patterns

Last updated by | GUSTAVO SOUZA | 25 de jan. de 2022 at 13:59 BRT

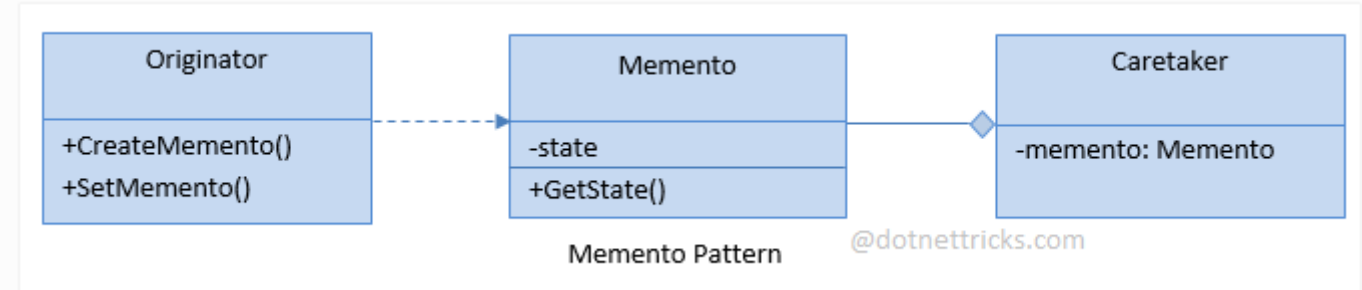
**Memento**

- *Definição*
  Memento é um padrão de projeto de software documentado no Catálogo Gang of Four, sendo considerado como um padrão comportamental. Ele permite armazenar o estado interno de um objeto em um determinando momento, para que seja possível retorná-lo a este estado, sem que isso cause problemas com o encapsulamento.

- *Histórico*
  Quando o estado interno de um objeto é alterado (como os seus atributos), se "perde" os valores armazenados anteriormente. Como se a aplicação "esquecesse" dos valores anteriores. Com isso a proposta do Memento é justamente o inverso: permitir que a aplicação se lembre dos estados anteriores de um objeto.
  Criando a possibilidade de restaurar um estado anterior. Em termos mais técnicos, compare o Memento com a operação de desfazer (Ctrl + Z), utilizada com bastante frequência no dia-a-dia. Ao desfazer uma ação, o sistema restaura o estado anterior dos dados, ou seja, antes da última alteração.

- *Para que serve e Formas de utilização*
  É um padrão de design comportamental que permite salvar e restaurar o estado anterior de um objeto sem revelar os detalhes de sua implementação.

  O cliente solicita um Memento do objeto de origem quando precisa verificar o estado do objeto de origem. O objeto de origem inicializa o Memento com uma caracterização de seu estado. O cliente é o "care-taker" do Memento, mas apenas o objeto de origem pode armazenar e recuperar informações do Memento (o Memento é "opaco" para o cliente e todos os outros objetos). Se o cliente precisar subsequentemente "reverter" o estado do objeto de origem, ele entregará o Memento de volta ao objeto de origem para reintegração.




- Originador é o objeto cujo estado se deseja capturar.

- Memento é o objeto definido dentro da classe
  Originador, com modificador de acesso privado, cujo estado do objeto originador será armazenado.

- Cliente é o objeto que acessará o originador, e deseja desfazer qualquer mudança efetuada, caso necessário.

- *Padrões similares*
  Por mais que não seja similar, algumas vezes o Prototype pode ser uma alternativa mais simples a um Memento. Isso funciona se o objeto, o estado no qual você quer armazenar na história, é razoavelmente intuitivo e não tem ligações para recursos externos, ou as ligações são fáceis de se restabelecer.

**Chain of Responsibility**

- *Definição*

O Chain of Responsibility é um padrão de projeto comportamental que permite que você passe pedidos por uma corrente de handlers. Ao receber um pedido, cada handler decide se processa o pedido ou o passa adiante para o próximo handler na corrente.

- *Para que serve*

Serve para evitar o acoplamento do remetente de uma solicitação ao seu receptor, ao dar a mais de um objeto a oportunidade de tratar essa solicitação. Encadear os objetos receptores, passando a solicitação ao longo da cadeia até que um objeto a trate.

- *Padrão Similar*

1. **Middleware** - O padrão de Middleware implementado pelo express já é bem conhecido e tem sido usado por desenvolvedores. A implementação representa um pipeline de processamento onde handlers, units e filters são funções. Essa funções são conectadas criando uma sequência de processamento assíncrona que permite pré-processamento, processamento e pós-processamento de qualquer tipo de dado. Uma das principais vantagens desse pattern é a facilidade de adicionar plugins de maneira não intrusiva.

**Composite**

- *Definição*
  O Composite é um padrão de projeto estrutural que permite que você componha objetos em estruturas de árvores e então trabalhe com essas estruturas como se elas fossem objetos individuais.

- *Pra que serve e formas de utilização*
  Compor objetos em estruturas de árvore para representar
  hierarquias, ele permite aos clientes tratarem de maneira uniforme objetos
  individuais e composições de objetos.

  Utilize o padrão Composite quando você tem que implementar uma estrutura de objetos tipo árvore e quando você quer que o código cliente trate tanto os objetos simples como os compostos de forma uniforme.

**Mercado de Trabalho**
Conhecer Design Patterns é algo de extrema importância no desenvolvimento de qualquer software. A utilização desses padrões nos ajuda a desenvolver de forma mais rápida frente a desafios semelhantes, fornece uma linguagem comum durante a documentação e discussões técnicas além de nos auxiliar a organizar o código fonte do software que estamos desenvolvendo.

**Fontes**

1. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (1994). Design Patterns: Elements of Reusable Object-Oriented Software. [S.l.]: Addison Wesley. pp. 283ff.
2. https://www.andrecelestino.com/delphi-design-patterns-memento/
3. https://medium.com/xp-inc/design-patterns-parte-20-memento-6b30ad75b12f#:~:text=É um padrão de design,os detalhes de sua implementação.
4. https://imasters.com.br/back-end/padrao-de-projetos-com-memento
5. https://www.opus-software.com.br/design-patterns/