

- por Rildo Ferreira dos Santos (rildosan@uol.com.br) -

Uma breve introdução:

Padrões de Projetos, quando são aplicados ao desenvolvimento de aplicações, fornecem meios de se descrever soluções comuns para problemas comuns.

Resultando em redução de tempo gasto com o desenvolvimento e melhoria da qualidade da aplicação.

Os padrões são organizados em grupos, assim temos os seguintes grupos:

Padrões Estruturais:	
Descreve como os objetos interagem entre si. Em outras palavras, eles abordam o framework estrutural dos objetos	
Nome	Breve Descrição
Bridge	Separa uma abstração de sua implementação, de modo que as duas possam variar independentemente.
Adapter	Permite que dois objetos se comuniquem mesmo que tenha interfaces incompatíveis.
Composite	Agrupa os objetos em estrutura de árvore para representar a hierarquia do tipo partes - todo. O Composite permite que os clientes tratem objetos individuais e composições de objetos de maneira uniforme.
Decorator	Atribui responsabilidade adicionais a um objeto dinamicamente. O Decorator fornece uma alternativa flexível a subclasses para a extensão da funcionalidade.
Façade	Fornecer uma interface única para um subsistema com diversas interfaces. O Façade define uma interface de nível mais alto que torna o subsistema mais fácil de usar.
Flyweight	Usa compartilhamento para dar suporte inúmeros objetos, de granularidade fina, de forma eficiente
Proxy	Fornecer um objeto representante ou marcador de outro objeto, para controlar o acesso ao mesmo.

Padrões Comportamentais:	
Descreve como podemos usar os objetos para alterar o comportamento de um sistema em tempo de execução.	
Nome	Breve Descrição
Observer	Define um uma dependência um-para-muitos entre objetos, de modo que, quando um objeto muda de estado todos os seus dependentes são notificados e atualizados.
Mediator	Cria um objeto que age como um mediador, controlando a interação entre um conjunto de objetos
Chain of Responsibility	Evita o dependência do remetente (cliente) de uma requisição ao seu destinatário, dando a oportunidade de mais de objeto tratar a requisição.
Command	Encapsula uma requisição como um objeto, desta forma permitindo que você parametrize clientes de diferentes solicitações.
Interpreter	É usado para ajudar uma aplicação a entender uma declaração de linguagem natural e executar a funcionalidade da declaração.
Iterator	Oferece um método (um meio) para fazer acesso aos elementos de uma coleção
Visitor	É usado Quando as operações necessitam ser realizadas sobre

	inúmeros elementos de um modelo de objeto.
Template Method	Define o esqueleto de uma algoritmo em uma operação, adiando a definição de alguns passos para as subclasses. O Template Method permite que as subclasses redefinam certos passos de um algoritmo sem mudar sua estrutura.
Memento	Possibilita armazenar o estado de um objeto de modo que o mesmo possa ser recuperado
State	Permite que um objeto altere seu comportamento quando se estado interno muda.
Strategy	Define uma família de algoritmos, encapsular cada um deles e fazê-los intercambiáveis. O Strategy permite que o algoritmo varie independente dos clientes que o utilizam

Padrões Criação	
Usado na criação de objetos	
Nome	Breve Descrição
Abstract Factory	Provê uma interface para criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.
Factory Method	Define uma interface para criar um objeto, mas deixa as subclasses decidirem qual classe a ser instanciada. O Factory Method permite a uma subclasse adiar a instanciação às subclasses.
Singleton	Garante que uma classe tenha somente uma "instance" (objeto)
Builder	Separa a construção de um objeto complexo da sua representação, de modo que o mesmo processo de construção possa criar diferentes representações.
Prototype	Especifica os tipos de objetos a serem criados usando uma "instance" prototípica e criar novos objetos copiando este protótipo.

O padrão Command:

Apresentaremos o padrão Command que pertence ao grupo de Padrões Comportamentais.

Propósito:

Encapsular uma requisição como um objeto, permitindo que os clientes parametrizem diferentes requisições, filas ou fazer o registro de log de requisições e dar suporte operações que podem ser desfeitas.

Também conhecido como:

Action ou Transaction

Motivação:

As vezes precisamos executar uma operação sem se preocupar com qual objeto que vai realiza-la ou simplesmente não conhecemos qual objeto vai receber a delegação para executar tal operação.

Em uma aplicação Client/Servidor geralmente temos o componente Menu que é composto de vários itens. Cada item do menu equivale uma operação, como salvar um arquivo, ler arquivo, apagar arquivo, selecionar a paleta de cores e etc..

Quando selecionamos um item do menu uma operação deve ser realizada. Esta operação pode ser encapsulada em um objeto, assim reduziremos o acoplamento entre o objeto menu e o objeto que executa a operação.

Um outro cenário que podemos ter. Em uma aplicação Web de três camadas, podemos fazer uma requisição para processar dos dados de um formulário esta requisição é delegada para um objeto. Esta operação pode ser encapsulada em um objeto, assim reduziremos o acoplamento entre o as camadas.

Aplicação:

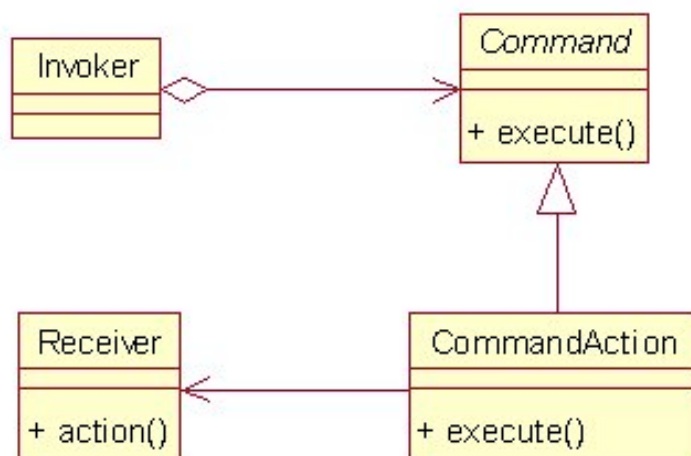
Use o padrão Command quando você precisa:

- Parametrizar objetos por uma ação a ser executada
- **Especificar, enfileirar e executar solicitações em tempos diferentes.** Um objeto Command poder ter o ciclo de vida independente da requisição do cliente.
- **Suporte para desfazer operações.** A operação "execute" do Command, pode armazenar estados para reverter seus efeitos no próprio comando. Basta acrescentar na interface Command uma operação chamada "Unexecute", que terá a responsabilidade de desfazer a operação realizada pelo "execute". Os comandos realizados podem ser armazenados em lista histórica.
- **Estruturar um sistema em torno de operações de alto nível, como transações por exemplo.** Uma transação encapsula um conjunto de mudanças nos dados. O padrão Command provê uma maneira de modelar transações. O Command tem uma interface comum, assim podemos chamar todas as transações do mesmo jeito.
- Reduzir acoplamento entre as requisições dos clientes e o objetos que as executam.

Solução:

Implemente o padrão Command para encapsular as operações e reduzir o acoplamento entre as requisições e os objetos que as executam. Para facilitar implantação de novas operações e tornar mais simples a manutenção das operações.

Estrutura:



Participantes	Descrição
Command	Classe abstrata ou Interface para execução de uma operação. Deve implementar o método abstrato chamado execute
CommandAction	Classe concreta que implementa o método execute, ela deve ser uma subclasse se Command for uma classe abstrata. Se Command for uma interface então ela deve realizar esta interface
Invoker	Objeto que faz uma requisição ao Command para que execute uma operação
Receiver	Objeto responsável por executar a operação

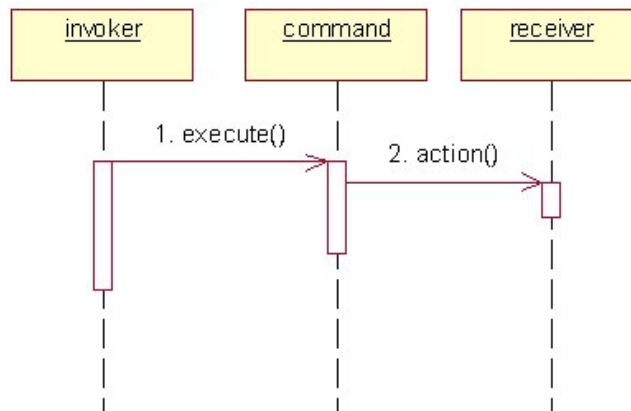


Diagrama de Seqüência:

Em tempo de execução o objeto Invoker chama o método execute() do objeto Command que delega ao método action() do objeto receiver que executará a operação.

Estratégias:

Para implementar o padrão Command pode usar algumas estratégias:

- Em uma aplicação Client/Servidor baseada em janelas podemos utilizar a classe AbstractAction do pacote: **javax.swing.AbstractAction**, para implementar o padrão Command

Classe	Descrição
javax.swing.AbstractAction	Fornecer implementação padrão para interface Action
TestCommand	A classe que implementa o objeto Menu e MenuItem que representa o invoker, que faz a requisição da operação.
ExitAction, ShowDialogAction e ShowJColorChooserAction	Classe concreta que implementa o actionPerformed da interface ActionListener, este é método responsável por executar ou delegar a operação.

- Em aplicação Web podemos utilizar o **framework Struts** (<http://struts.apache.org/>)*

Classe e Arquivos	Descrição
web.xml	Arquivo responsável por conter as configurações da aplicação
action.xml	Arquivo responsável por mapear a requisição e classe que vai atendê-la
SimpleAction	Classe concreta que implementa o método "execute" responsável por realizar ou delegar uma operação.
CounterBean	Bean responsável por manter o estado do atributo contador.
test.jsp	Página JSP que possui um link para fazer requisição
forwardPage.jsp	Página JSP que apresenta a resposta da requisição.

- Outra opção é fazer escrever todas as classes e interfaces necessárias para implementar o padrão.

Classe	Descrição
ActionCommand	Classe abstrata que implementa o método execute
ClienteCommand	A classe que representa o invoker, que faz a requisição da operação.
AddAction, DeleteAction e UpdateAction	Classe que concreta que implementa o método "execute" responsável por realizar uma operação

Conseqüências:

O padrão Command tem as seguintes conseqüências:

- Command reduz o acoplamento (dependência) entre o objeto que chama a operação e o objeto que executa a operação.
- Command é objeto que pode ser estendido e manipulado por outros objetos
- É mais fácil de acrescentar novas operações ou novos comandos, pois você não tem que mudar as classes existentes.

Padrões Relacionados:

O padrão Command pode ser combinado com outros padrões como:

- **Composite:** Use o Composite para implementar MacroCommands
- **Memento:** Use para manter o estado do receiver dentro do padrão Command para suportar o desfazer das operações.
- **Prototype:** Use para copiar o Command antes adicioná-lo na lista de histórica de comandos
- **Singleton:** Em alguma aplicações, a lista histórica é implementada como um Singleton.

Para ir além:

Padrões de Projetos (Soluções Reutilizáveis de Software Orientado a Objetos)

Autores: Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides
 Editora Bookman, ISBN: 85-7307-610-0

Framework Struts (<http://struts.apache.org/>)

Este framework implementa o padrão MVC (Model View Controller) para aplicações Web

