Command Design Pattern in C#

The Command Design Pattern is a behavioral pattern that encapsulates a request as an object, thereby allowing for parameterization of clients with different requests, queuing of requests, and logging of requests. It is especially useful in scenarios where the sender of the request should be decoupled from the object that handles the request.

@lijotech

Usage Scenarios

- Undo/Redo functionality: In applications like text editors or drawing tools.
- Queuing commands: For batch processing or scheduling.
- Parameterizing objects: With different operations, decoupling the requester and executor.
- Fire-and-Forget Scenario: Useful for scenarios, where a request is sent to be processed later without waiting for an immediate response.
- Macro Commands: Group a set of commands into one.
- Transaction Management: Implement rollback or commit functionality.
- Remote-Control Systems: Commands can represent actions for controlling devices like drones, robots, or machinery.
- Workflow Systems: Implement a workflow engine where each step in the workflow is a command that executes some logic and has undo capabilities.

@lijotech

Example: Remote Control for a Smart Home

Command Interface

```
public interface ICommand
{
    void Execute();
    void Undo();
}
```

Receiver

```
public class Light
{
    public void TurnOn()
    {
        Console.WriteLine("The light is ON.");
    }

    public void TurnOff()
    {
        Console.WriteLine("The light is OFF.");
    }
}
```

@lijotech

Concrete Commands

```
public class TurnOnLightCommand : ICommand
{
    private readonly Light _light;
    public TurnOnLightCommand(Light light) => _light = light;
    public void Execute() => _light.TurnOn();
    public void Undo() => _light.TurnOff();
}

public class RemoteControl
{
    private readonly Stack<ICommand> _history = new Stack<ICommand>();
    public void Invoke(ICommand command) { command.Execute(); _history.Push(command); }
    public void Undo() { if (_history.Any()) _history.Pop().Undo(); }
}
```

Concrete Commands

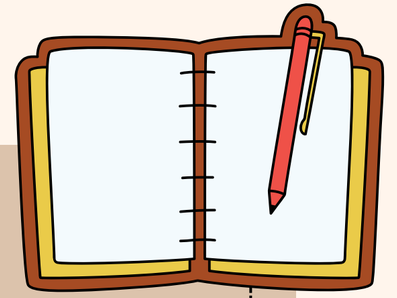
Invoker

@lijotech

Client

```
public class Program
{
    public static void Main(string[] args)
    {
        Light livingRoomLight = new Light();
        ICommand turnOnLight = new TurnOnLightCommand(livingRoomLight);
        RemoteControl remoteControl = new RemoteControl();
        Console.WriteLine("Executing command:");
        remoteControl.Invoke(turnOnLight);

        Console.WriteLine("Undoing command:");
        remoteControl.Undo();
    }
}
```



Stay Focused & Happy Coding

Follow Lijo Sebastian for more coding tips

@lijotech

@lijotech