Stefan Đokić • Following
➡️ I help you to improve your .NET knowledge! | Microsoft MVP
1h •

How can you use the Adapter Pattern in .NET?

What is the real-world example? ⬇️

The Problem: A Real-World Story

Imagine you've just moved to a new country. You're excited to set up your home and plug in your laptop, but then you realize the power outlets are completely different from the ones back home.

Your laptop's plug doesn't fit into the wall socket. Panic sets in - your laptop is essential for work!

What can you do?

You visit a local store and discover the solution: a power adapter. The adapter lets your laptop's plug fit seamlessly into the foreign socket.

It doesn't change the wall socket or laptop plug; it simply acts as a bridge, translating one interface into another.

Problem solved!

In the software world, you often encounter mismatched systems. For instance, you're building a modern application that needs to integrate with a legacy payment gateway.

Your application works with REST APIs, while the payment gateway only supports SOAP-based services. They speak different "languages" and can't communicate directly.

Modifying the legacy system to support REST APIs is costly and risky. Similarly, rewriting your application to support SOAP is time-consuming and unnecessary.

How do you bridge the gap?

The Solution: The Adapter Pattern

The Adapter pattern solves this problem by acting as a translator. It allows two incompatible interfaces to work together without changing their existing code.

Here’s how it works:

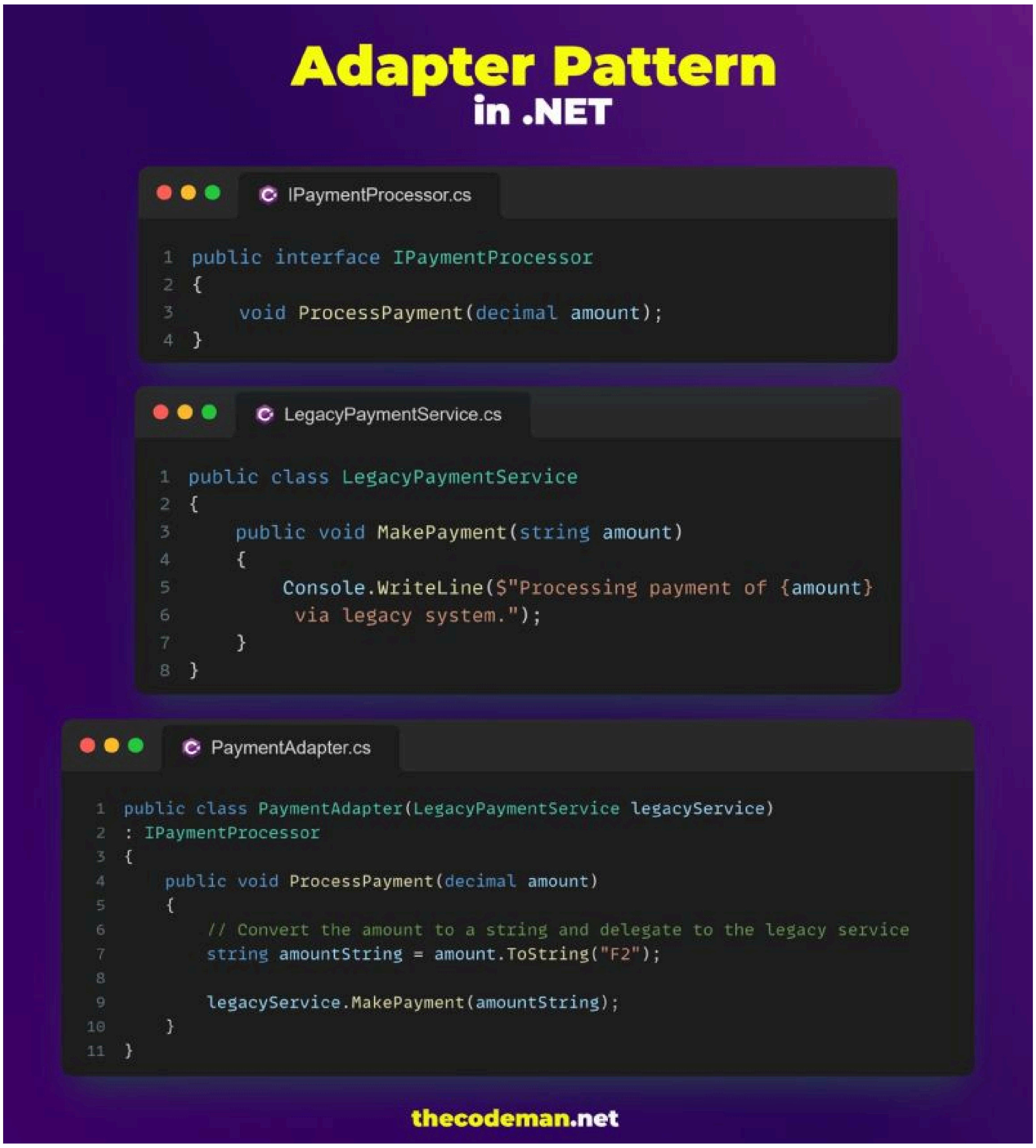
1. Create an Adapter: Build a new class that implements the interface your application expects (e.g., REST APIs).
2. Delegate to the Legacy System: Inside the adapter, translate the REST API calls into the SOAP requests understood by the payment gateway.
3. Return Results in the Expected Format: The adapter translates SOAP responses back into REST API responses for your application.

What is the real case implementation in .NET?
Cloud Providers Integration with Adapter Pattern, for example?

Check the implementation here: **https://lnkd.in/dRZkcJ8S**

—

P.S. Don't be silly not to join **thecodeman.net**



Adapter Pattern in .NET

IPaymentProcessor.cs

```csharp
public interface IPaymentProcessor
{
    void ProcessPayment(decimal amount);
}
```

LegacyPaymentService.cs

```csharp
public class LegacyPaymentService
{
    public void MakePayment(string amount)
    {
        Console.WriteLine($"Processing payment of {amount}
         via legacy system.");
    }
}
```

PaymentAdapter.cs

```csharp
public class PaymentAdapter(LegacyPaymentService legacyService)
: IPaymentProcessor
{
    public void ProcessPayment(decimal amount)
    {
        // Convert the amount to a string and delegate to the legacy service
        string amountString = amount.ToString("F2");

        legacyService.MakePayment(amountString);
    }
}
```

thecodeman.net

47

8 comments 2 reposts