C# Design Patterns: Strategy

STRATEGY PATTERN

Filip Ekberg
PRINCIPAL CONSULTANT & CEO

@fekberg fekberg.com

Course Overview

Understanding and implementing the strategy pattern

Identifying and leveraging existing implementations

Understanding the benefits and tradeoffs

Strategy Pattern Characteristics

Context

Has a reference to a strategy and invokes it

IStrategy

Defines the interface for the given strategy

Strategy

A concrete implementation of the strategy

Strategy Pattern Characteristics

Context

Calls
IStrategy.GetTaxFor(order)

IStrategy

Defines the contract:
GetTaxFor(Order order)

Strategy

SwedenSalesTaxStrategy
USAStateSalesTaxStrategy

Strategy Pattern Characteristics

Context

Calls
IStrategy.CreateInvoice(...)

IStrategy

Defines the contract:
CreateInvoice(Order order)

Strategy

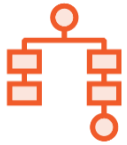
PDFInvoiceStrategy
EmailInvoiceStrategy
PrintInvoiceStrategy

Select an implementation at runtime based on user input without having to extend the class

Demo

Strategy pattern: First look

What Did We Achieve?



A more extensible, object oriented and dynamic implementation

Easily add new strategies without affecting existing ones

Cleaner approach with single responsibility in mind

Decouple your code and achieve a cleaner, more extensible code base

Example: Tax Calculation

```
if (destination == "sweden")
{

    if (destination == ShippingDetails.OriginCountry.ToLowerInvariant())
    {
        return  TotalPrice * 0.25m;
    }

     return 0;

}

if (destination == "us")
{
    switch (ShippingDetails.DestinationState.ToLowerInvariant())
    {
        case "la":  return  TotalPrice * 0.095m;
        case "ny":  return  TotalPrice * 0.04m;
        case "nyc": return  TotalPrice * 0.045m;
        default :   return  0m;
    }
}
```

Example: Tax Calculation

Leveraging the Strategy Pattern

Setting the strategy

```
var order = new Order
{

    SalesTaxStrategy = new
                SwedenSalesTaxStrategy()
};
```

Using the strategy

```
if (SalesTaxStrategy == null)
{
    return 0m;
}

return SalesTaxStrategy.GetTaxFor(this);
```

Demo

Strategy pattern: An alternative approach

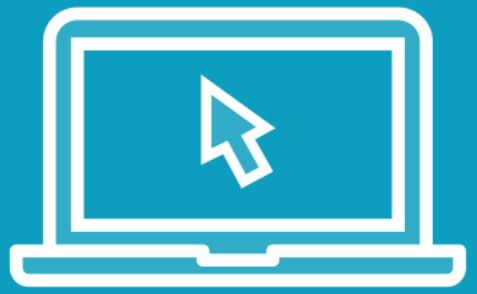
Demo

Example: Creating an invoice

Demo

Example: Using different shipping providers

Demo



Existing implementations: Array.Sort

Whenever you inject an interface to allow change of behavior you are leveraging the strategy pattern

Summary

One of the most commonly used patterns

Decouple the context and the concrete implementation

Allows for a cleaner implementation in the context

Easily extend with additional strategies without affecting current implementations

Makes testing a lot easier as you can write mocked implementations to inject

Identify existing implementations and where you have used the pattern before