

Patrons de conception 03

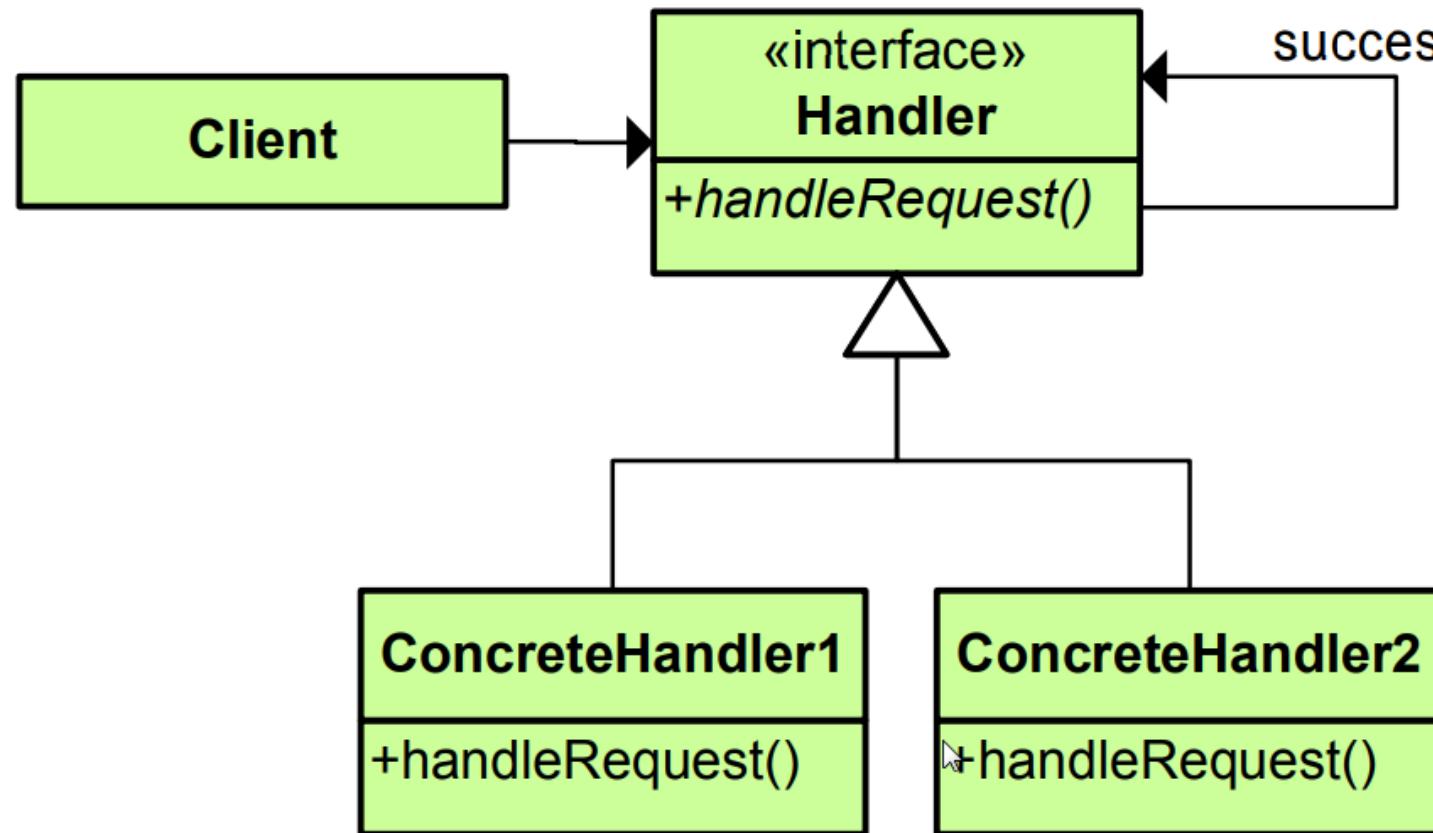


Objectifs

- Comprendre le patron de conception « Chaine de responsabilités »
- Implanter ce patron en C#



Chaine de responsabilité



Chain of Responsibility

Type: Behavioral

What it is:

Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.

Implantation en C# - Exemple 1

- Dans cet exemple, nous allons créer une chaîne de responsabilités qui peut cumuler les actions suivantes : mettre en majuscule / minuscule, enlever les diacritiques, ajouter un préfixe et un suffixe :
 - Le traitement prend une chaîne en paramètres et renvoie une chaîne
 - On effectue le traitement demandé
 - S'il y a un traitement suivant on l'exécute à partir du résultat précédent
 - On renvoie le résultat

Implantation en C# - Exemple 1

```
public interface ITraitementChaineCaracteres
{
    public ITraitementChaineCaracteres Suivant { get; set; }
    public string TraiterChaineCaracteres(string p_chaine);
}
```

```
public class TraitementChaineCaracteresMettreEnMinuscules :
ITraitementChaineCaracteres {
    public ITraitementChaineCaracteres Suivant { get; set; }

    public string TraiterChaineCaracteres(string p_chaine) {
        string res = p_chaine.ToLower();

        if (Suivant != null) {
            res = this.Suivant.TraiterChaineCaracteres(res);
        }

        return res;
    }
}
```

```

// tcc1 : MetteEnMajuscules -> SupprimerDiacritics -> AjouterPrefixeSuffixe
ITraitementChaineCaracteres tcc1 = new TraitementChaineCaracteresAjouterPrefixeSuffixe("<p>", "</p>");
tcc1 = new TraitementChaineCaracteresSupprimerDiacritiques() { Suivant = tcc1 };
tcc1 = new TraitementChaineCaracteresMettreEnMajuscules() { Suivant = tcc1 };

// tcc2 : AjouterPrefixeSuffixe -> SupprimerDiacritics
ITraitementChaineCaracteres tcc2 = new TraitementChaineCaracteresSupprimerDiacritiques();
tcc2 = new TraitementChaineCaracteresAjouterPrefixeSuffixe("<p>", "</p>") { Suivant = tcc2 };

List<string> chaines = new List<string> () {
    "éèàçÉÈÀÇ",
    "La vanité, encore qu'elle fleurisse, ne graine pas.",
    "Celui qui a passé le gué sait combien la rivière est profonde."
};

chaines.ForEach(s => Console.Out.WriteLine($"{s} :{Environment.NewLine} - tcc1 : {tcc1.TraiterChaineCaracteres(s)}{Environment.NewLine} - tcc2 : {tcc2.TraiterChaineCaracteres(s)}"));

```

éèàçÉÈÀÇ :

- tcc1 : <p>EEACEEAC</p>
- tcc2 : <p>eeacEEAC</p>

La vanité, encore qu'elle fleurisse, ne graine pas. :

- tcc1 : <p>LA VANITE, ENCORE QU'ELLE FLEURISSE, NE GRAINE PAS.</p>
- tcc2 : <p>La vanite, encore qu'elle fleurisse, ne graine pas.</p>

...

Implantation en C# - Exemple 2

- Dans cet exemple, nous allons créer une chaîne de responsabilités qui doit traiter une requête de dessin de formes
 - Si l'objet courant peut le traiter, il le fait et arrête la chaîne
 - Sinon il le passe à l'objet suivant

Implantation en C# - Exemple 2

```
public interface IDessinForme {
    public IDessinForme Suivant { get; set; }
    public void Dessiner(IFormeGeometrique p_forme);
}
```

```
public class DessinerFormeEllipse : IDessinForme {
    public IDessinForme Suivant { get; set; }

    public void Dessiner(IFormeGeometrique p_forme) {
        if (p_forme is Ellipse) {
            Console.Out.WriteLine($"Je dessine une ellipse...");
        }
        else {
            this.Suivant?.Dessiner(p_forme);
        }
    }
}
```

Implantation en C# - Exemple 2

```
public class DessinerFormeSentinelle : IDessinForme {
    public IDessinForme Suivant {
        get {
            return null;
        }
        set {
            throw new InvalidOperationException();
        }
    }

    public void Dessiner(IFormeGeometrique p_forme) {
        throw new ArgumentOutOfRangeException(
            nameof(p_forme),
            $"Aucun traitement pour la forme {p_forme.GetType().Name}"
        );
    }
}
```

```
// Segment -> Rectangle -> Ellipse
IDessinForme dfSansSentinelle = new DessinerFormeEllipse();
dfSansSentinelle = new DessinerFormeRectangle() { Suivant = dfSansSentinelle };
dfSansSentinelle = new DessinerFormeSegment() { Suivant = dfSansSentinelle };

// Segment -> Rectangle -> Ellipse -> Sentinel
IDessinForme dfAvecSentinelle = new DessinerFormeSentinelle();
dfAvecSentinelle = new DessinerFormeEllipse() { Suivant = dfAvecSentinelle };
dfAvecSentinelle = new DessinerFormeRectangle() { Suivant = dfAvecSentinelle };
dfAvecSentinelle = new DessinerFormeSegment() { Suivant = dfAvecSentinelle };

List<IFormeGeometrique> formes = new List<IFormeGeometrique>() {
    new Ellipse(), new Rectangle(), new Segment(), new Triangle()
};

Console.Out.WriteLine("Chaine sans sentinelle : ");
formes.ForEach(f => dfSansSentinelle.Dessiner(f));

Console.Out.WriteLine();
Console.Out.WriteLine("Chaine avec sentinelle : ");
formes.ForEach(f => {
    try {
        dfAvecSentinelle.Dessiner(f);
    }
    catch (Exception ex) {
        ConsoleColor backColor = Console.BackgroundColor;
        Console.BackgroundColor = ConsoleColor.Red;
        Console.Error.WriteLine(ex.Message);
        Console.BackgroundColor = backColor;
    }
});
```

Chaine sans sentinelle :
Je dessine une ellipse...
Je dessine un rectangle...
Je dessine un segment...

Chaine avec sentinelle :
Je dessine une ellipse...
Je dessine un rectangle...
Je dessine un segment...

Aucun traitement pour la forme Triangle (Parameter 'p_forme')