

# La programmation dynamique

Depuis ses débuts, C# est un langage fortement et statiquement typé, ce qui signifie que le compilateur est capable de connaître le type de chaque variable utilisée dans une application. Ceci lui permet de savoir comment traiter les appels de méthodes et de propriétés sur chaque objet, et permet de générer des erreurs de compilation lorsqu'une opération est invalide.

L'arrivée de C# 4 et de l'environnement .NET en version 4.0 a introduit un nouveau type de donnée singulier : `dynamic`. Ce type s'adresse à des problématiques très particulières car il permet d'utiliser des variables dont le type n'est connu qu'au moment de l'exécution.

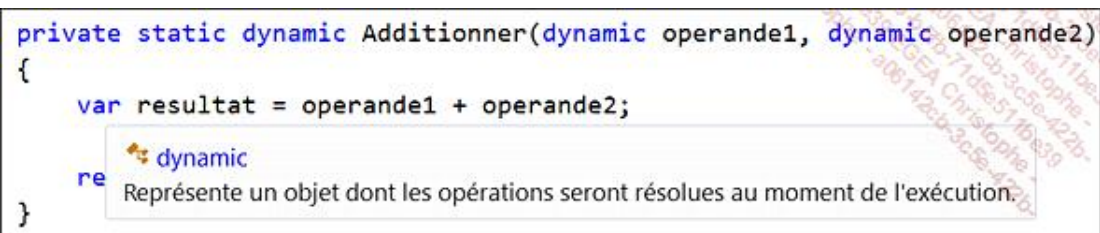
Le compilateur n'effectue ainsi pas de vérifications pour les opérations effectuées sur des variables `dynamic`. Elles ne sont effectuées qu'au moment de l'exécution de l'application : toute opération détectée comme invalide à ce stade génère une erreur d'exécution.

La fonction ci-dessous utilise deux paramètres de type `dynamic` et leur applique l'opérateur `+`.

```
private static dynamic Additionner(dynamic operande1, dynamic
operande2)
{
    var resultat = operande1 + operande2;

    return resultat;
}
```

Le type de retour de la fonction est `dynamic` car il n'est pas déterminable avant la compilation. L'utilisation de l'inférence de type dans Visual Studio le montre très bien en plaçant le curseur de la souris sur le mot-clé `var` :



```
private static dynamic Additionner(dynamic operande1, dynamic operande2)
{
    var resultat = operande1 + operande2;
    re
}
```

Les opérandes étant de type `dynamic`, IntelliSense est incapable de proposer de méthodes ou propriétés utilisables.

Le code suivant permet de vérifier le fonctionnement de la méthode `Additionner` :

```
int entier1 = 10;
int entier2 = 3;
Console.WriteLine(Additionner(entier1, entier2));

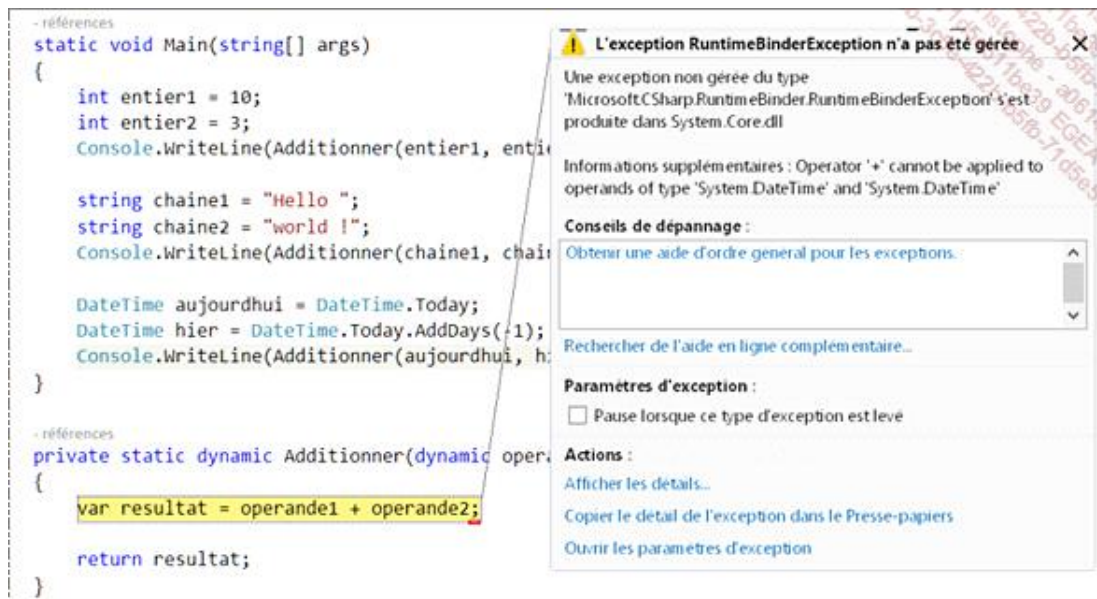
string chaine1 = "Hello ";
string chaine2 = "world !";
Console.WriteLine(Additionner(chaine1, chaine2));

DateTime aujourd'hui = DateTime.Today;
DateTime hier = DateTime.Today.AddDays(-1);
```

```
Console.WriteLine(Additionner(aujourd'hui, hier));
```

Ce code compile sans problème, malgré un problème que l'on peut déceler : il est impossible d'additionner deux objets de type `DateTime`.

L'exécution de ce code se passe sans encombre pour les deux premières additions. En revanche, comme on pouvait s'y attendre, l'opération utilisant deux objets `DateTime` génère une erreur d'exécution :



Il convient donc d'être particulièrement prudent avec l'utilisation du type `dynamic`, celle-ci pouvant générer des erreurs non détectables à la compilation.

En pratique, ce type sert souvent dans des contextes d'interopérabilité avec des objets COM ou des langages dynamiques (IronPython ou IronRuby).

- L'utilisation d'objets dynamiques implique une perte de performance lors de l'exécution. Il convient donc de n'utiliser le type `dynamic` que s'il est vraiment indispensable pour effectuer un traitement.