



Langage C# - Faut-il abuser de l'inférence de type (var) ?



James RAVAILLE | [Follow](#)

Consultant & Formateur Microso...



12



2



0

Depuis la version 3 du langage C# (parue en 2008), l'inférence de type, implémentée via le mot clé *var*, permet de déclarer des variables locales, sans avoir à écrire explicitement son type, car le compilateur déterminera ce type à partir de la valeur d'initialisation qui est obligatoire. Voici quelques exemples :

```
var Code = "125ER78"; // Variable est de type string
var Identifiant = 1; // Variable est de type int
```

Mais est-ce une bonne pratique ? Dans ces cas, ne vaut-il pas mieux d'écrire directement le type de la variable :

Que nous connaissons ;

Qui peut être plus adapté que le type choisi par le compilateur de manière arbitraire. Par exemple, la variable *Identifiant* est de type *int*, mais peut-être souhaiterions-nous qu'elle soit d'un autre type numérique tel que *long*, *short* ou *sbyte*, voire *uint*, *ulong*, *ushort*, *byte*, ... en fonction de la plage des valeurs possible pour cette variable ?

Et dans cet autre cas :

```
var resultat = this.ExecuterTache("A25");
```

La lecture de cette instruction ne permet pas de connaître le type de la variable *resultat*, il faut lire la signature de la méthode *ExecuterTache* ou mettre le curseur de la souris sur cette variable pour en connaître le type ...

Bref, faut-il utiliser dans ces cas l'inférence de type sous prétexte qu'elle existe ? Je ne le pense pas, pour les raisons suivantes :

Il est important de choisir avec précision le type de la variable.

de faciliter la maintenance.

Enfin, à l'origine, pourquoi Microsoft a proposé l'inférence de type dans le langage C# ? Je vois différentes raisons majeures :

Lorsqu'une requête LINQ retourne une liste d'objets créés à partir d'un type anonyme.

Pour typer les variables d'entrée des expressions lambda.

Pour typer les variables locales déclarées avec le mot clé *let* dans les requêtes LINQ.

Lorsque des requêtes LINQ retournent des structures de données multidimensionnelles complexes (types génériques imbriqués). Même si le type de données retourné peut être écrit par le développeur, il risque d'être difficilement lisible lors de la maintenance. Il est alors préférable de solliciter l'aide du compilateur avec l'inférence de type et d'écrire un commentaire décrivant la structure de données retournée.

Ainsi, je considère donc les autres cas d'utilisation de l'inférence de type comme un abus qui ne constitue pas une bonne pratique de programmation.

Published By



James RAVAILLE

James Consultant & Formateur Microsoft

[Follow](#)

RAVAILLE

C# - Faut-il #abuser de l'#inférence de #type (#var) ?

2 comments

[Sign in](#) to leave your comment



Cédric
DANIEL

Cédric DANIEL

Chef de projets chez Léni

9mo

Bonjour, en effet dans l'exemple donné, le choix d'utiliser l'inférence de type n'est pas pertinent car le nom de la méthode n'indique pas clairement quel sera le résultat du calcul. mais si on prends les exemples suivants, l'utilisation du mot clé var apportera de la lisibilité à mon code car on peut également se retrouver avec d'autres dérives (pour le point 1 ci-dessous, on se retrouve avec une redondance d'informations et un appel un peu verbeux, pour le point 2, le développeur se dit que le fait de nommer sa variable "c" se suffira au moment de l'appel, mais on se retrouve avec une variable mal nommée tout du long du traitement). 1. Customer customer = customerRepository.GetById(id); 2. Customer c = customerRepository.GetById(id); 3. var customer = customerRepository.GetById(id);


Like Reply | 1 Like 1 Reply



Bonjour Cédric, Merci pour ce complément avec lequel je suis aussi d'accord. Je sais que cette question provoque parfois un débat entre les développeurs et je voulais exprimer mon point de vue concernant l'abus de l'inférence de type. Le point 3 est lisible dans l'état actuel. Mais lors de la maintenance, si la méthode GetById retourne un objet d'un autre type que Customer (dans ce cas, c'est peut-être exagéré, car normalement, cette méthode ne doit retourner que des objets de type Customer), les développeurs changeront-ils le nom des variables où cette méthode est utilisée ? Par expérience, j'ai déjà vu que ce n'était pas le cas et cela rend les algorithmes moins lisibles (et donc moins maintenables), même s'ils pourraient le faire à partir d'une fonctionnalité de refactoring de Visual Studio. Alors que s'ils utilisent la première instruction de ton exemple, le compilateur peut les inviter à modifier le type de la variable (suite à une erreur de compilation) et ils peuvent changer le nom en même temps ... Cordialement,


[Like](#) [Reply](#) | [1 Like](#)

More from James RAVAILLE

 C# - Optimiser les performances des opérations ensemblistes avec la classe HashSet


C# - Optimiser les performances des...

March 1, 2019

 C# - La clause « Let » dans les requêtes LINQ

C# - La clause « Let » dans les requêtes LINQ

February 24, 2019

 WPF - Traitement asynchrone modifiant les

WPF - Traitement asynchrone

February 23

© 2019

[User Agreement](#)

[Cookie Policy](#)

[Brand Policy](#)

[Community Guidelines](#)

[About](#)

[Privacy Policy](#)

[Copyright Policy](#)

[Guest Controls](#)

[Language](#) 