

Améliorez la qualité de votre code avec ReSharper

La qualité de code est au cœur de l'activité du développeur. Pour cela, il convient d'appliquer un certain nombre de transformations sur notre code, apportées par les pratiques de développement comme le Clean Code ou les nouvelles fonctionnalités des langages de développement (C# notamment).

En tant que coach craftsmanship chez Cdiscount, j'accompagne les équipes de développement dans leur quotidien afin qu'elles améliorent la qualité de leur base de code. Ces pratiques nécessitent d'être évangélisées et leur intégration directement dans les outils de développement permet de les intégrer plus facilement dans le quotidien du développeur. Il peut ainsi détecter si son code contient des défauts (appelés plus couramment code smells) et les corriger automatiquement.

Présentation de ReSharper

Resharper (R# pour les intimes) est un outil de productivité pour les développeurs .NET qui s'intègre comme une extension de l'IDE Visual Studio (une autre possibilité est d'utiliser directement l'IDE Rider de JetBrains qui offre quasiment les mêmes fonctionnalités) et améliore considérablement l'expérience développeur sur de nombreux aspects.

Une des puissances de ReSharper provient du fait que toutes ses actions peuvent être directement effectuées avec des raccourcis clavier.

ReSharper fournit deux schémas de raccourcis clavier par défaut :

- Visual Studio : ce schéma remplace les raccourcis classiques de Visual Studio lorsqu'ils existent. A préconiser pour les développeurs habitués aux raccourcis de l'IDE Microsoft.
- IntelliJ IDEA : ce schéma réutilise les raccourcis des autres IDE JetBrains. A préférer pour les utilisateurs de ces IDE (IntelliJ, WebStorm, Rider...).

Quel que soit le schéma initialement choisi, vous pouvez toujours modifier les raccourcis clavier de manière individuelle ultérieurement.

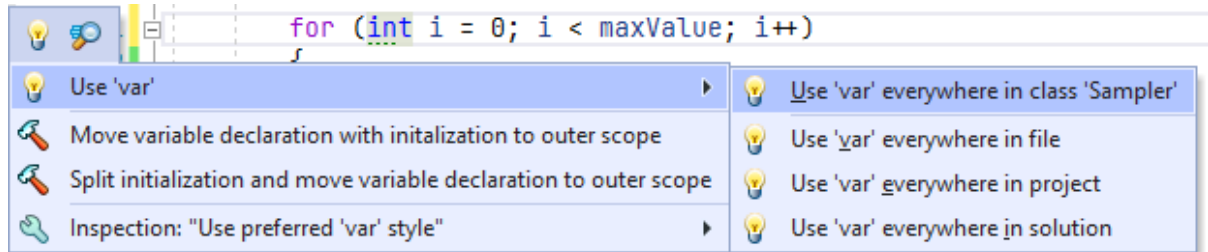
Les raccourcis évoqués dans cet article sont pour les plus courants communs aux 2 schémas. A défaut, le raccourci précisé est celui de Visual Studio (et celui d'IntelliJ entre parenthèses).

Analyse statique de code en temps réel

Souvent quand vous développez, vous voulez effectuer une action mais vous ne vous souvenez plus du raccourci correspondant. ReSharper analyse votre code au fur et à mesure que vous l'écrivez et le lisez. En cliquant sur l'ampoule ou via le raccourci universel *Alt + Entrée*, il ouvrira un menu contextuel vous proposant les corrections ou optimisations appropriées à votre contexte courant.

Ce raccourci sous forme de point d'entrée unique est idéal pour se laisser guider et découvrir des subtilités du langage que l'on ne connaissait pas encore.

Même lorsque votre curseur ne se trouve pas sur une ligne de code, *Alt + Entrée* vous permet d'accéder à l'ensemble des actions possibles en tapant juste quelques lettres. La plupart des correctifs proposés par Resharper s'appliquent au problème courant mais peuvent également être appliqués sur un scope plus large (au niveau du fichier, du projet ou de la solution).



Facilité de refactoring et de nettoyage

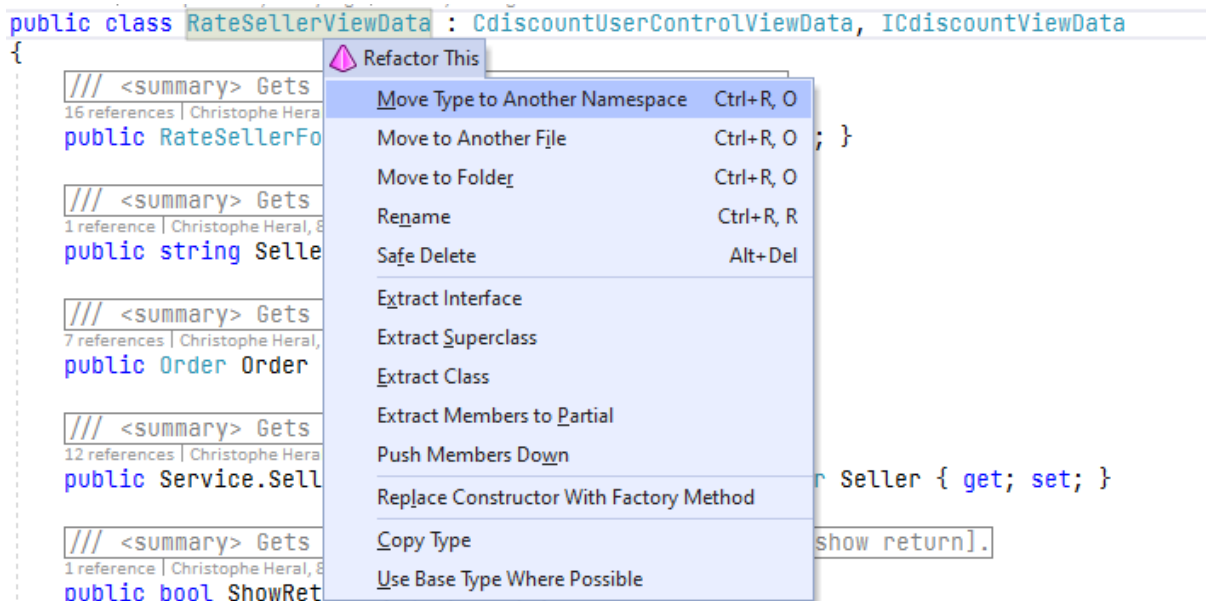
Pour obtenir du code propre, il ne faut pas avoir peur de refactorer l'existant, surtout lorsque l'on travaille sur du code legacy. Pour rappel, le refactoring consiste à retravailler le code existant pour en améliorer la lisibilité et la maintenance, ceci sans ajouter de nouvelle fonctionnalité, ni corriger d'anomalie existante.

Le catalogue de refactoring de Martin Fowler (<https://www.refactoring.com/catalog/>) est une bonne base des différents refactorings possibles.

Visual Studio propose déjà certains d'entre eux. Resharper est optimisé pour cela et va beaucoup plus loin dans les options offertes pour vous faciliter la vie au quotidien dans ces tâches :

- Vous avez une méthode trop longue avec différents traitements isolés dans des régions ? Il vous propose d'extraire des méthodes privées. Vous pouvez également supprimer en un clic les tags *#region* maintenant inutiles dans votre code.
- Un nombre magique a été dispatché à plusieurs endroits dans le code ? Il vous propose d'en extraire une constante et de remplacer toutes les occurrences. La même chose est valable pour des variables.
- Un paramètre n'est plus utilisé par une méthode ? Il le supprime à votre place et supprime les arguments passés à l'ensemble des appelants.
- Vous souhaitez extraire une interface d'une classe ? Pas de panique, il le fait pour vous.
- Vous préférez convertir un if en ternaire ? Cela se fait tout seul.
- Vous avez une expression LINQ difficile à lire ? Il peut la convertir automatiquement en code; ou réciproquement transformer une boucle for simple en une expression LINQ plus concise.
- ...

Toutes ces actions sont disponibles via le menu contextuel "*Refactor this*" dès que vous sélectionnez du code et que vous appuyez sur *Ctrl + Shift + R*.

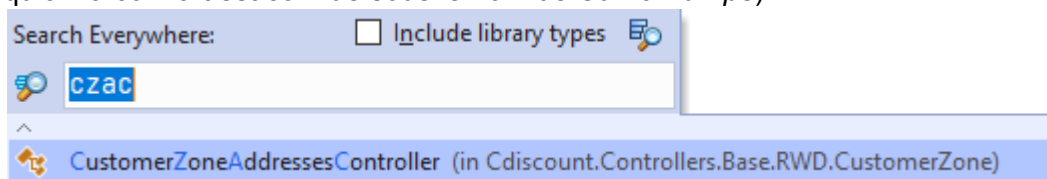


Vous pouvez également définir la liste des différentes règles que vous souhaitez appliquer sur l'ensemble de votre code et passer la totalité de votre projet à la moulinette de ce nettoyage que vous aurez vous-même configuré.

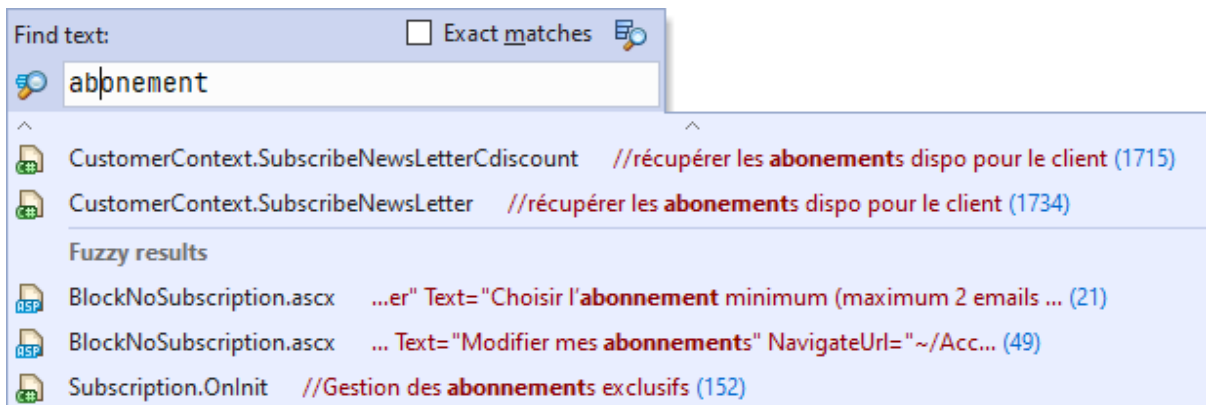
Amélioration de la navigation et de la recherche

Resharper vous permet de naviguer facilement dans l'ensemble de votre code. Le raccourci `Ctrl + T` (`Ctrl + N`) vous propose alternativement 3 possibilités (selon le nombre de fois que vous cliquez) :

- “*Search Anywhere*” : R# vous propose l'ensemble des éléments dans votre code (classe, méthode, variable...) qui correspond à votre saisie. Les suggestions incluent en priorité les fichiers et les méthodes vues récemment. Pour gagner du temps, vous pouvez uniquement taper la première lettre de chaque mot-clé en CamelCase (cette fonctionnalité dont on ne peut plus se passer une fois qu'on la connaît est connue sous le nom de *CamelHumps*).



- “*Go To Type*” : la recherche est alors plus ciblée sur le nom d'un type uniquement.
- “*Go To Text*” : la recherche s'applique ici aux occurrences particulières d'un texte. Par défaut, la recherche floue s'applique et permet de retrouver du code même si le libellé n'est pas totalement exact.



D'autres raccourcis de navigation sont également très puissants :

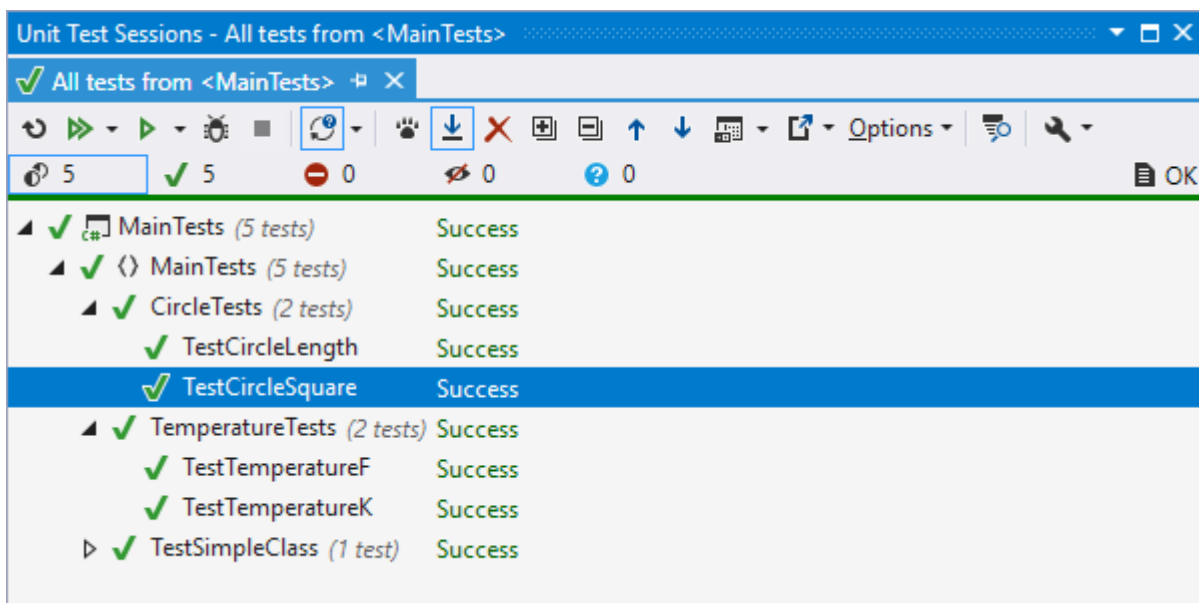
- *Ctrl + Alt + Droite (Ctrl + W)* : Étendre la sélection du bloc de code courant à partir du curseur sur le mot, la ligne, la boucle, la méthode...
- *Ctrl + Alt + Gauche (Ctrl + Shift + W)* : Réduire la sélection.
- *Alt + PageDown (F12)* : Naviguer jusqu'à l'erreur/warning suivant(e) dans le fichier.
- *Shift + Ctrl + BackSpace* : Revenir à la modification précédente (utile après avoir navigué dans plusieurs fichiers).

L'autocomplétion de Resharper remplace celle par défaut de Visual Studio et prend le dessus sur le raccourci *Ctrl + Space*. C'est également le cas pour la coloration syntaxique.

Support des tests unitaires

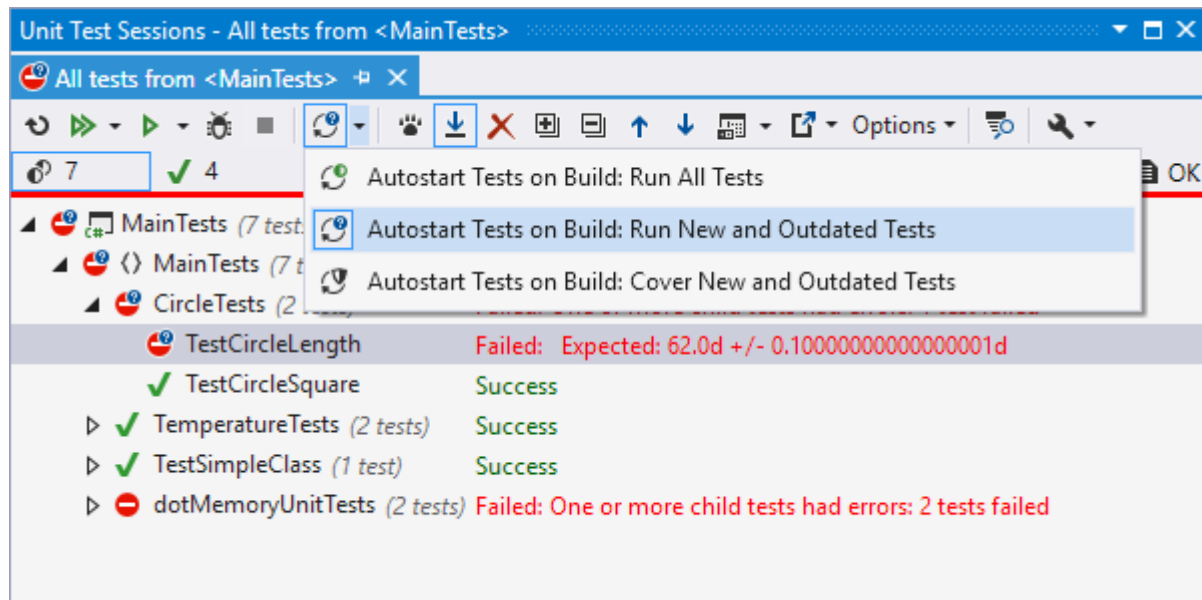
ReSharper détecte automatiquement des tests unitaires basés sur NUnit, xUnit et MSTest dans vos projets .NET et .NET Core afin de les exécuter et de les déboguer. Sélectionnez simplement un fichier, dossier, projet ou solution, et appuyez sur *Ctrl+U, R (Ctrl + T, R)*.

Vous pouvez également exécuter une session de tests unitaires préalablement enregistrée et composée de n'importe quelle combinaison de tests.



Avec dotCover (également disponible avec ReSharper Ultimate), vous pouvez facilement visualiser et analyser le niveau de couverture de code de votre solution.

Il est également possible de profiter des tests continus : dotCover découvre à la volée quels tests unitaires sont concernés par vos dernières modifications de code et exécute automatiquement ces tests pour vous. Cela est particulièrement utile lorsque l'on effectue du TDD avec le cycle Red / Green / Refactor.



Génération de code

Resharper vous propose de générer du code pour vous : des constructeurs, des méthodes et des propriétés, des vérifications d'égalité, etc.

Par exemple, lorsque vous faites du TDD, vous aurez tendance à utiliser une méthode, une propriété ou même une classe avant sa déclaration. ReSharper suggérera alors un correctif rapide pour générer la déclaration correspondant à cette utilisation et ajustera intelligemment la déclaration en fonction du contexte de l'utilisation. Exemple dans le cas d'une méthode : il créera la méthode et détectera également les types de ses paramètres et son type de retour afin d'obtenir une signature la plus cohérente possible.

Resharper enrichit également les snippets proposés par Visual Studio. Par exemple, en tapant *ctorfp* depuis la liste de complétion, il génère un constructeur qui initialise les champs et propriétés :

```

public class Person
{
    private string age;
    1 reference | 0 changes | 0 authors, 0 changes
    public string Name { get; set; }

    0 references | 0 changes | 0 authors, 0 changes
    public Person(string age, string name)
    {
        this.age = age;
        Name = name;
    }
}

```

Support en continu des nouveautés de C# et .NET

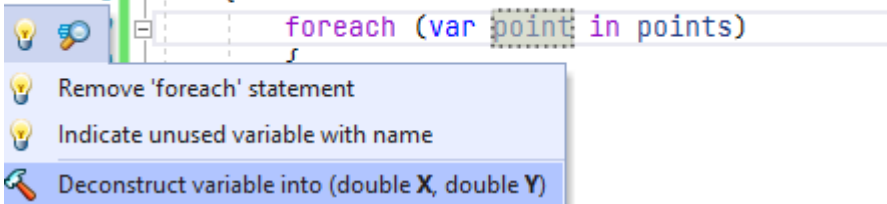
Resharper sort de nouvelles releases tout au long de l'année et vous accompagne dans la découverte des nouveautés implémentées par C# au fil de ses versions (notamment 8 et 9 pour les dernières).

Si vous utilisez par exemple un positional record, Resharper va vous proposer de le déconstruire en plusieurs variables dans une boucle où il est utilisé :

```

1 using System.Collections.Generic;
2
3 public record Point(double X, double Y);
4
5 0 references | 0 changes | 0 authors, 0 changes
6 public class Sampler
7 {
8     0 references | 0 changes | 0 authors, 0 changes
9     public void Draw(List<Point> points)
10    {
11        foreach (var point in points)

```



Il offre également un support pour tout l'écosystème .NET (par ex les routes en ASP.NET).

Conclusion

S'équiper en tant que développeur de Resharper a certes un coût. Et il faut évidemment garder en tête que maîtriser les pratiques de Clean Code est plus important que maîtriser un outil de développement, quel qu'il soit.

Pour progresser sur ces pratiques, vous pouvez vous appuyer sur les ouvrages suivants :

- *Refactoring: improving the design of existing code* de Martin Fowler
- *Working effectively with legacy code* de Michael Feathers
- *Clean Code* de Robert Martin

Comme tout bon artisan, un développeur doit posséder les bons outils et les maîtriser afin d'améliorer son efficacité et sa productivité. Pour C#, Resharper fait clairement partie de ceux-là et propose un ensemble de tutoriels pour découvrir plus en détails l'ensemble de ces fonctionnalités.

Christophe HERAL - @ChrisHeral
Artisan Logiciel Indépendant & Coach Technique Agile