

Découvrez le top 15 des raccourcis IntelliJ IDEA

Par [Trisha Gee](#)

Date de publication : 9 mars 2020

Pour réagir au contenu de ce tutoriel, un espace de dialogue vous est proposé sur le forum.
Commentez

| | |
|--|----|
| I - Introduction..... | 3 |
| II - Les raccourcis..... | 3 |
| II-1 - Alt + Entrée..... | 3 |
| II-2 - F2..... | 3 |
| II-3 - #1 ou Alt+1..... | 4 |
| II-4 - Échap..... | 4 |
| II-5 - #E ou Ctrl+E..... | 5 |
| II-6 - #B ou Ctrl+B..... | 5 |
| II-7 - Alt+F7..... | 6 |
| II-8 - Ctrl Ctrl..... | 6 |
| II-9 - #↑ / #↓ ou Ctrl + W / Maj + Ctrl + W..... | 7 |
| II-10 - #/ ou Ctrl + /..... | 7 |
| II-11 - ### ou Maj+Ctrl+Entrée..... | 8 |
| II-12 - ##L ou Ctrl+Alt+L..... | 8 |
| II-13 - #T ou Maj+Ctrl+Alt+T..... | 9 |
| II-14 - ##A ou Maj+Ctrl+A..... | 9 |
| II-15 - Maj Maj..... | 10 |
| III - Plus d'informations..... | 10 |
| IV - Remerciements Developpez.com..... | 11 |

I - Introduction

IntelliJ IDEA dispose de raccourcis clavier pour la plupart de ses commandes les plus fréquemment utilisées, notamment pour les actions liées à l'édition, à la navigation, à la refactorisation et au débogage. L'apprentissage et l'utilisation régulière de ces raccourcis permettent de les intégrer à la **mémoire musculaire** et aident à rester en « **état de flow** ».

Vous trouverez une **vidéo présentant tous ces raccourcis en action** sur la **chaîne YouTube IntelliJ IDEA**.

Cliquer sur ce lien pour lancer l'animation

II - Les raccourcis

II-1 - Alt + Entrée

Commençons par le plus connu d'entre eux. Ce raccourci clavier peut être utilisé pour corriger à peu près tout, en vous présentant les actions adaptées au contexte dans lequel vous vous trouvez.

Lorsque vous voyez une erreur dans votre code, placez le curseur sur cette erreur et appuyez sur Alt+Entrée pour obtenir une liste de suggestions de corrections.

Vous pouvez également utiliser Alt+Entrée là où vous voyez des avertissements et des suggestions et choisir d'accepter l'une de ces suggestions.

Vous pouvez même utiliser Alt+Entrée sur du code qui ne comporte pas d'erreur, d'avertissement ou de suggestion : cela affichera les **actions d'intention** (l'une de mes préférées est l'ajout d'importations statiques) et les **inspections** qui sont activées, mais non configurées pour vous avertir.

```
private final String[] stringArray = new String[] { "intellij" };
private List<Converter> converters;

private void error() {
    List<String> strings = new HashMap<Integer, String>();
}

public void lambdas() {
    //Remove redundant types
    Function<Function, Function> f3 =
        (Function function) -> function.compose(function)

    //Lambda can be replaced with method reference
}
```

II-2 - F2

Si vous ne voulez pas utiliser la souris pour naviguer entre les erreurs et avertissements dans l'éditeur, alors utilisez **F2 pour passer à la prochaine erreur (ou avertissement ou suggestion)**. En combinant ceci avec Alt+Entrée, vous pourrez voir toutes les suggestions et en choisir une, ou choisir d'appliquer la première suggestion avec Maj+Alt+Entrée.

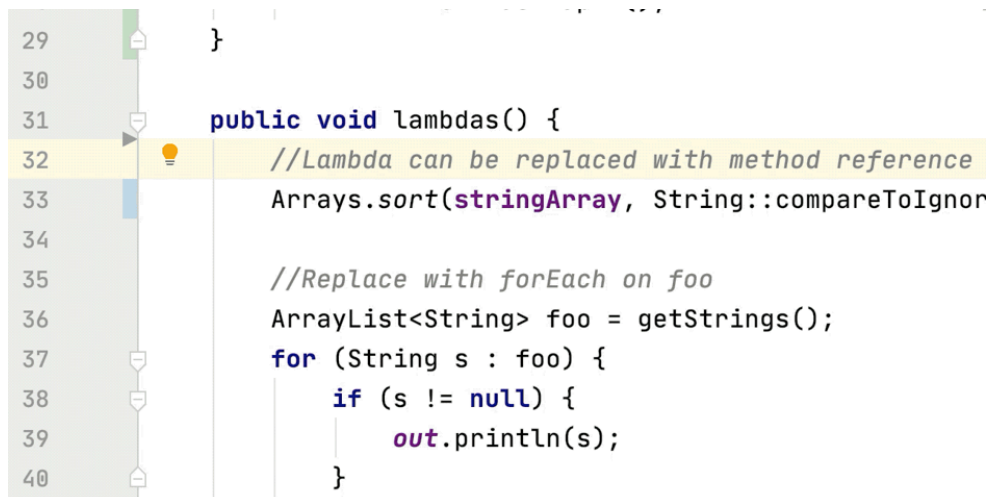
```
private void error() {
    List<String> strings
    = new HashMap<Integer, String>();
}

public void lambdas() {
    //Lambda can be replaced with method reference
    Arrays.sort(stringArray, (s1, s2) -> s1.compareToIgnoreCa

    //Replace with forEach on foo
    ArrayList<String> foo = getStrings();
    for (String s : foo) {
```

II-3 - #1 ou Alt+1

Vous n'avez pas non plus besoin de la souris pour ouvrir les fenêtres d'outils. #1 (MacOS) ou Alt+1 (Windows/Linux) permet d'ouvrir la fenêtre du projet et d'y placer le focus. Vous pouvez naviguer dans l'arborescence en utilisant les touches flèches et effectuer une recherche par saisie.

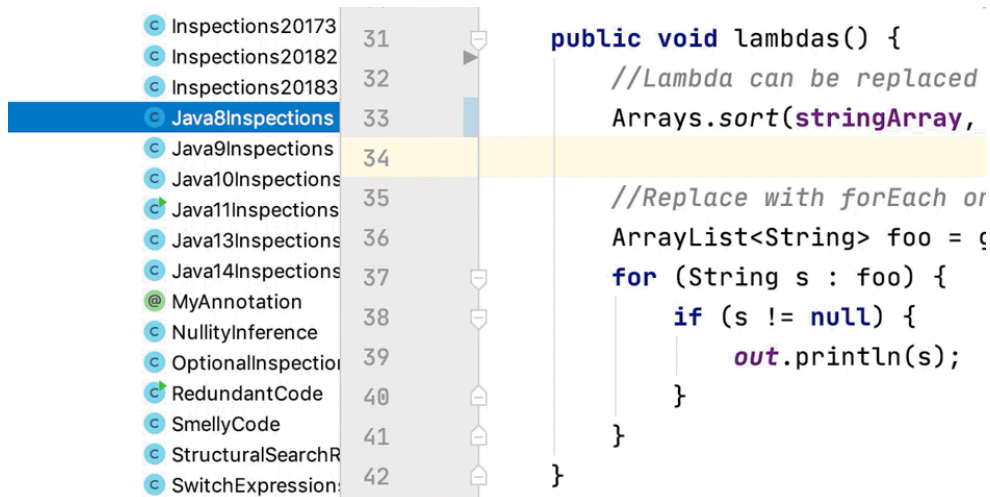


```
29 }
30
31 public void lambdas() {
32     //Lambda can be replaced with method reference
33     Arrays.sort(stringArray, String::compareToIgnor
34
35     //Replace with forEach on foo
36     ArrayList<String> foo = getStrings();
37     for (String s : foo) {
38         if (s != null) {
39             out.println(s);
40         }
41     }
```

II-4 - Échap

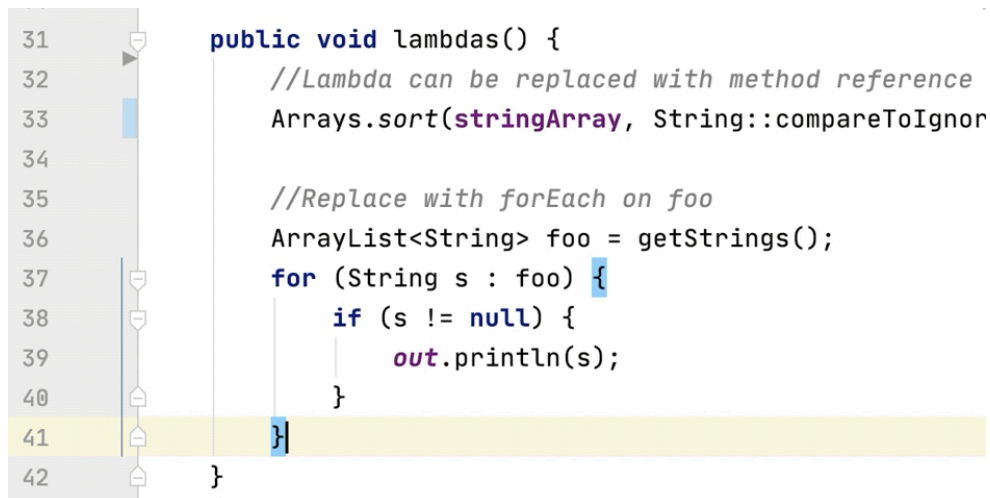
Pour remettre le focus sur l'éditeur, appuyez sur la touche Échap. Quelle que soit la fenêtre d'outils ouverte, cela vous permettra de revenir à l'éditeur et de continuer à travailler sur le code. En fait, la touche Échap permet de fermer toute fenêtre contextuelle sans appliquer de modification.

Pour plus de conseils sur la navigation vers et à partir de l'éditeur, consultez les [principes de base de l'éditeur](#).



II-5 - #E ou Ctrl+E

La fenêtre projet n'est pas forcément le meilleur moyen de naviguer vers le fichier que vous recherchez. Vous pouvez **visualiser les derniers fichiers ouverts** en utilisant #E (MacOS) ou Ctrl+E (Windows/Linux). La fenêtre des fichiers récents s'ouvre alors et vous pouvez y naviguer à l'aide des touches fléchées. Vous pouvez également ouvrir des fenêtres d'outils à partir de là, y compris celles qui n'ont pas de raccourci clavier. Comme dans toute fenêtre d'IntelliJ IDEA, il est possible d'effectuer une recherche par saisie.



II-6 - #B ou Ctrl+B

Nous souhaitons fréquemment pouvoir naviguer simplement dans le code. #B (MacOS) ou Ctrl+B (Windows/Linux) permettent d'**aller à la déclaration** d'un symbole. Par exemple, en pressant ces touches sur un champ, le curseur se positionne sur la déclaration du champ. En pressant ces touches sur le nom d'une classe, on accède au fichier de la classe. Si vous appuyez sur ##B (MacOS) ou Ctrl+Alt+B (Windows/Linux), vous pouvez **naviguer vers une implémentation** à la place.

Consultez **Navigation dans le code source** pour plus de conseils de ce type.

```

        repository = new CustomerRepositoryStub();
    }

    @Test
    public void shouldSaveCustomer() {
        Customer customer = new Customer();
        repository.save(customer);

        //TODO: assert customer saved
    }

    private static class CustomerRepositoryStub implements Cu

```

II-7 - Alt+F7

Plutôt que de trouver la déclaration, nous voulons souvent **trouver où quelque chose est utilisé**. C'est possible avec Alt+F7. Par exemple, en appuyant sur Alt et F7 sur ce nom d'interface, la fenêtre de recherche vous montrera tous les endroits où l'interface est utilisée, qu'il s'agisse d'une déclaration de champ ou d'une classe qui implémente cette interface.

```

import ...

public interface CustomerRepository {
    void save(Customer customer);
    Customer getById(int id);
    List<Customer> getAll();
}

```

II-8 - Ctrl Ctrl

En appuyant deux fois sur la touche Ctrl vous pouvez **exécuter** toute action depuis tout emplacement. Peu importe où vous vous trouvez dans l'EDI ou quel fichier est ouvert, le raccourci double Ctrl permet d'ouvrir la fenêtre Run Anything et d'afficher par défaut une liste des dernières **configurations exécutées**. Mais vous pouvez également saisir le nom de quelque chose à exécuter pour rechercher d'autres configurations d'exécution.

```
public class DataFlow {

    private int id;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    void modifyId() {
```

II-9 - #↑ / #↓ ou Ctrl + W / Maj + Ctrl + W

Vous pouvez sélectionner des sections de code ascendantes ou descendantes près du curseur avec # et les flèches haut ou bas (MacOS) et Ctrl+W ou Ctrl+Maj+W (Windows/Linux). Lors de **l'extension de la sélection**, IntelliJ IDEA sélectionne automatiquement la prochaine expression valide dans les sections croissantes. #↓ (MacOS) ou Ctrl+Maj+W (Windows/Linux), réduira à nouveau la sélection jusqu'au curseur.

```
sed/
ic class ExtendSelection {
    private Set<Map.Entry<String, Long>> exampleCodeForExtendAndS
    return getStrings().stream()
        .collect(groupingBy(stringValue -> stringValue,
            counting()))
        .entrySet();
}

private List<String> getStrings() {
    return new ArrayList<>();
}
```

II-10 - #/ ou Ctrl + /

En appuyant sur #/ (MacOS) ou Ctrl+/ (Windows/Linux) n'importe où sur une **ligne passera cette ligne de code en commentaire** avec un commentaire de ligne. Le même raccourci permet d'annuler le commentaire si la ligne est déjà commentée.

Vous pouvez aussi utiliser ## (MacOS) ou Maj+Ctrl+/ (Windows/Linux) sur un bloc de code entier pour **y ajouter un commentaire**. En appuyant à nouveau sur ce raccourci avec le curseur placé dans le bloc de code, le commentaire de bloc sera supprimé.

```
/unused/
public class MultiLineStrings {

    public void multiLineStrings() {
        String someText = "This code can be on a single " +
            "line or it can be on multiple lines " +
            "and you can type and hit Enter and " +
            "get the + sign inserted automatically";
    }
}
```

II-11 - ### ou Maj+Ctrl+Entrée

Complete current statement, ### (MacOS) ou Maj+Ctrl+Entrée (Windows/Linux), est l'un des raccourcis les plus utiles lorsque vous codez. Si nous avons l'habitude d'utiliser la saisie automatique de la déclaration pour écrire du code, la plupart du temps, cela ajoutera simplement un point-virgule à la fin du code. Mais cela fonctionne pour du code plus complexe. Par exemple, si vous l'utilisez lorsque vous écrivez une boucle "for", IntelliJ IDEA ajoutera les accolades et placera votre curseur à l'intérieur du bloc. Dans une déclaration "if", il peut ajouter les parenthèses et les accolades et placer à nouveau votre curseur au bon endroit. Même si l'EDI n'a pas besoin d'ajouter de code supplémentaire pour terminer votre déclaration, ce raccourci est utile pour placer le curseur là où vous en aurez besoin ensuite.

```
private List<String> replaceWithCollect() {
    List<String> result = getStrings();
    return result;
}
```

Helper methods

II-12 - ##L ou Ctrl+Alt+L

Pour **formater le fichier actuel** selon les normes du projet facilement, utilisez ##L (MacOS), ou Ctrl+Alt+L (Windows/Linux). Vous pouvez choisir de formater uniquement les lignes qui ont changé dans le fichier ou le fichier entier. Le formatage peut **même ajouter des accolades** si cela est requis par les normes. Vous pouvez modifier la portée du formatage. Par exemple, appuyez sur ####L (MacOS) ou Maj+Ctrl+Alt+L (Windows/Linux) et choisissez de reformater l'ensemble du fichier.


```

    }

    public void horriblyFormattedMethod (){
        System.out.println("First line");
        System.out.println("Second line");
        System.out.println("Third line");
        for (int i = 0; i < 3; i++)
            System.out.println("Where?");
    }
}

```

II-13 - #T ou Maj+Ctrl+Alt+T

Dans IntelliJ IDEA, la plupart des refactorisations automatisées **ont leur propre raccourci**, mais il est aussi possible d'accéder à toutes les refactorisations avec un seul raccourci : #T (MacOS) ou Maj+Ctrl+Alt+T (Windows/Linux). En utilisant ce raccourci sur un symbole ou une sélection, vous obtenez **un affichage des options de refactorisation disponibles**. Sélectionnez ensuite une refactorisation, avec les touches fléchées et Entrée ou via le numéro à gauche de la refactorisation de votre choix. La boîte de dialogue affiche également le raccourci clavier de cette refactorisation, s'il existe, afin que vous puissiez l'utiliser directement une prochaine fois.

```

String hotkey = "Ctrl/Cmd+Alt+M";

String[] steps = new String[5];
steps[0] = "First select a block of code";
steps[1] = "Then press " + hotkey;
steps[2] = "Give the method a fullName";
steps[3] = "Assign it a visibility";
steps[4] = "Apply the refactoring";

System.out.println(s);

```

II-14 - ##A ou Maj+Ctrl+A

Pas besoin de mémoriser tous ces raccourcis. Utilisez **Find Action**, ##A (MacOS) ou Maj+Ctrl+A (Windows/Linux) pour rechercher n'importe quelle action dans IntelliJ IDEA. Le menu déroulant affichera non seulement les actions, mais aussi leur raccourci. Find Action permet aussi de rechercher les paramètres afin de pouvoir les modifier, et de rechercher et d'ouvrir des fenêtres d'outils.

```
package com.jetbrains.refactoring;

public class ExtractMethod {

    public static final String SHORTCUT = "Ctrl/Cmd+Alt+M";

    public static void main(String[] args) {
        String s = "How to extract a method: ";

        String[] steps = new String[5];
        steps[0] = "First select a block of code";
        steps[1] = "Then press " + SHORTCUT;
        steps[2] = "Give the method a fullName";
        steps[3] = "Assign it a visibility";
        steps[4] = "Apply the refactoring";

        System.out.println(s);

        //Extract this loop
        printSteps(steps);
    }

    private static void printSteps(String[] steps) {
        for(String step : steps) {
            System.out.println(step);
        }
    }
}
```

II-15 - Maj Maj

Le raccourci ultime est celui de **search everywhere**. Appuyez deux fois sur la touche Maj pour ouvrir une boîte de recherche qui vous permet de rechercher tout ce que vous souhaitez. De la même manière qu'avec Find Action, vous pouvez aussi rechercher et modifier les paramètres. La boîte de recherche affiche par défaut les fichiers récents, et peut donc être utilisée à la place de #E / Ctrl+E. Lorsque vous saisissez quelque chose à rechercher, vous pouvez voir les résultats des classes, fichiers, symboles et **actions**. Search everywhere prend également en charge les commandes, vous pouvez ainsi **rechercher des paramètres pour l'éditeur** par exemple.

```
public class ExtractMethod {

    public static final String SHORTCUT = "Ctrl/Cmd+Alt+M";

    public static void main(String[] args) {
        String s = "How to extract a method: ";

        String[] steps = new String[5];
        steps[0] = "First select a block of code";
        steps[1] = "Then press " + SHORTCUT;
        steps[2] = "Give the method a fullName";
        steps[3] = "Assign it a visibility";
        steps[4] = "Apply the refactoring";

        System.out.println(s);

        //Extract this loop
        printSteps(steps);
    }

    private static void printSteps(String[] steps) {
        for(String step : steps) {
            System.out.println(step);
        }
    }
}
```

III - Plus d'informations

Consultez la section **Maîtriser les raccourcis clavier** pour retrouver ces conseils et bien d'autres. N'oubliez pas que vous pouvez vérifier et modifier votre **configuration clavier**, qui est un autre endroit pour trouver des raccourcis clavier utiles et **créer les vôtres**. Vous pouvez également **télécharger et imprimer la configuration clavier par défaut** et placer ce document à un endroit où il sera toujours visible pendant que vous codez.

Il est aussi pratique de **créer des abréviations** pour trouver rapidement ce que vous utilisez fréquemment. Nous recommandons aux utilisateurs d'Ubuntu en particulier de consulter la section sur les **conflits avec les raccourcis du système d'exploitation**.

Un dernier conseil : pour vous entraîner à utiliser le clavier au lieu de la souris, essayez le **plugin Key Promoter X**. Lorsque vous utiliserez la souris au lieu du clavier pour réaliser une action, le raccourci clavier pour cette action se mettra à clignoter – un excellent rappel pour vous aider à mémoriser ce raccourci !

Connaître les principaux raccourcis clavier pour IntelliJ IDEA vous aidera à être plus productif et à conserver un niveau de concentration optimal.

Bon développement !

IV - Remerciements Developpez.com

Nous tenons à remercier **Malick** pour la mise au gabarit et **ClaudeLELOUP** pour la relecture orthographique.