

- » About IntelliJ IDEA
- » User Interface
- » Code Completion
- » Editor Basics
- » Navigation and more...

IntelliJ IDEA

By Andrey Chepstov

ABOUT INTELLIJ IDEA

From the get go, IntelliJ IDEA has followed a simple, overriding principle: if it doesn't make a Java developer more productive, it doesn't go in the product. Period. With a tight focus on what a professional Java developer needs, IntelliJ IDEA can make you as productive as you're capable of being. This Refcard is about unlocking that power. Focused on IntelliJ IDEA 15, which will have a full release in fall of 2015, most of the tips, tricks, and shortcuts in this Refcard will work for you regardless of your version of the IDE.

USER INTERFACE

The IntelliJ IDEA editor is unique in a number of ways, most notably that you can invoke almost any IDE feature without leaving it, which allows you to organize a layout where you have more screen space because auxiliary controls like toolbars and windows are hidden.

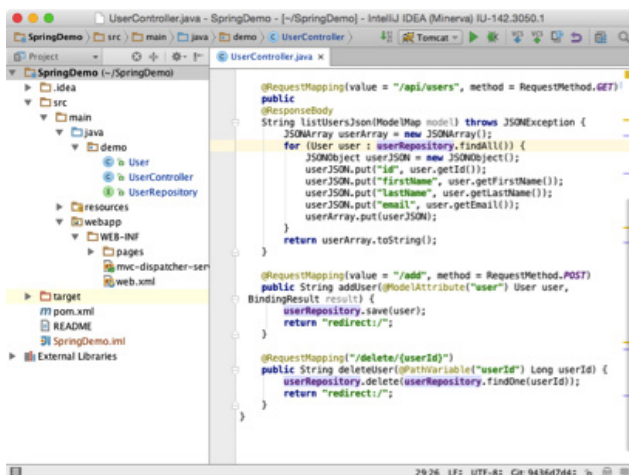


Figure 1: The IntelliJ IDEA UI

Accessing a tool window via its shortcut moves the input focus to it, so you can use all keyboard commands in its context. When you need to go back to the editor, press **Esc**.

ACTION	WINDOWS	OS X
Project	Alt+1	⌘1
Changes	Alt+9	⌘9
Run	Alt+4	⌘4
Debug	Alt+5	⌘5

ACTION	WINDOWS	OS X
Terminal	Alt+F12	⌘F12
Editor	Esc	Esc

When you need to just focus on the code, try the Distraction Free Mode. It removes all toolbars, tool windows, and editor tabs. To switch to this mode, select **View → Enter Distraction Free Mode**.

An alternative to Distraction Free Mode is hiding all tool windows by pressing **Ctrl+Shift+F12** (**Shift+Cmd+F12** for OS X). Restore the layout to default by using the shortcut again.

The Navigation Bar is a compact alternative to the Project tool window. To access the Navigation Bar, press **Alt+Home** (**Cmd+Up** for OS X).



Figure 2: The IntelliJ IDEA Navigation Bar

Most components in IntelliJ IDEA (both tool windows and popups) provide speed search. This feature allows you to filter the list or navigate to a particular item by using a search query.

EXCEL AT **ENTERPRISE**
& MOBILE
DEVELOPMENT
 WITH A **SMARTER IDE**



IntelliJ IDEA
 The most intelligent Java IDE

GET IT NOW

A free and open-source version is included www.jetbrains.com/idea

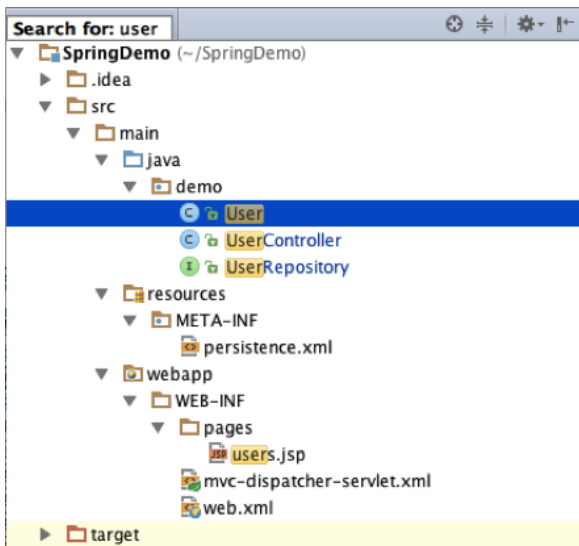


Figure 3: IntelliJ IDEA Speed Search

HOT TIP

When you don't know the shortcut for an action, try using the Find action feature by pressing **Ctrl+Shift+A** (**Shift+Cmd+A** for OS X). Start typing to find an action by its name, see its shortcut, or call it.

CODE COMPLETION

Accessing Basic Completion by pressing **Ctrl+Space** offers basic suggestions for variables, types, methods, expressions, and so on. When you call Basic Completion twice, it shows you more results, including private members and non-imported static members.

The Smart Completion feature is aware of the expected type and data flow and offers the option relevant to the context. To call Smart Completion, press **Shift+Ctrl+Space**. When you call **Smart completion** twice, it shows you more results, including chains.

To overwrite the identifier at the caret, instead of just inserting the suggestion, press **Tab**. This is helpful if you're editing part of an identifier, such as a file name.

To let the IDE complete a statement for you, press **Shift+Ctrl+Enter** (**Shift+Cmd+Enter** for OS X). Statement Completion will automatically add missing parentheses, brackets, braces, and necessary formatting.

PARAMETER INFO

If you want to see the suggested parameters for any method or constructor, just press **Ctrl+P** (**Cmd+P** for OS X). IntelliJ IDEA shows parameter info for every overloaded method or constructor and highlights the best match for the parameters already typed.

The Postfix Completion feature lets you transform an already

typed expression to another one based on the postfix you type after a period, the expression type, and its context.

EDITOR BASICS

Since in IntelliJ IDEA you can undo refactorings and revert changes from Local History, it makes no sense to ask you to save your changes every time—IntelliJ IDEA autosaves your changes for you.

Editor shortcuts include:

ACTION	WINDOWS	OS X
Move the current line of code	Shift+Ctrl+Up Shift+Ctrl+Down	⇧⌘↑ ⇧⌘↓
Duplicate a line of code	Ctrl+D	⌘D
Remove a line of code	Ctrl+Y	⌘⌫
Comment or uncomment a line of code	Ctrl+ /	⌘ /
Comment a block of code	Shift+Ctrl+ /	⇧⌘ /
Find in the currently opened file	Alt+F3	⌘F
Find and replace in the current file	Ctrl+R	⌘R
Next occurrence	F3	⌘G
Previous occurrence	Shift+F3	⇧⌘G
Navigate between opened tabs	Alt+Left Alt+Right	⌘← ⌘→
Navigate back/forward	Ctrl+Alt+Left Ctrl+Alt+Right	⌘[⌘]
Expand or collapse a code block in the editor	Ctrl+NumPad+ Ctrl+NumPad-	⌘+ ⌘-

Other useful editor actions:

ACTION	WINDOWS	OS X
Create new...	Ctrl+N	⌘N
Surround with...	Ctrl+Alt+T	⌘⌘T
Highlight usages of a symbol	Shift+Ctrl+F7	⇧⌘F7

To expand a selection based on grammar, press **Ctrl+W** (**Cmd+W** for OS X). To shrink it, press **Shift+Ctrl+W** (**Shift+Cmd+W** for OS X).

HOT TIP

It's worth knowing that IntelliJ IDEA is able to select more than one piece of code at a time. You can select/deselect any piece of code via **Alt+J** (**Ctrl+G** for OS X) / **Shift+Alt+J** (**Shift+Ctrl+G** for OS X) or by clicking the code selection and pressing **Shift+Alt**.

NAVIGATION

RECENT FILES

Most of the time you work with a certain set of files and need to switch between them quickly. A real time-saver here is an action called Recent Files invoked by pressing **Ctrl+E** (**Cmd+E** for OS X). By default, the focus is on the last accessed file. You can open any tool window through this action.

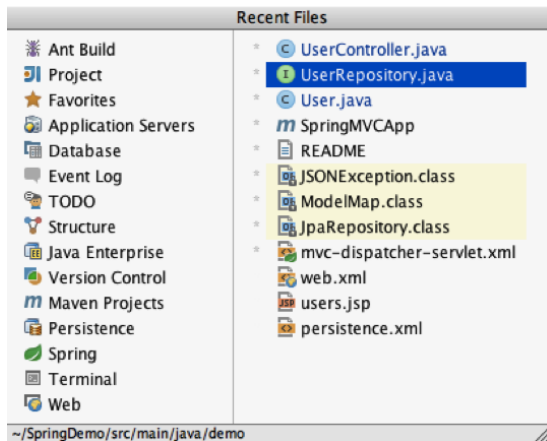


Figure 4: IntelliJ IDEA Recent Files

Navigate to Class is available by pressing **Ctrl+N** (**Cmd+O** for OS X) and supports sophisticated expressions, including “camel humps,” path, line navigate to, middle name matching, and many more. If you call it twice, it shows you the results out of the project classes.

Navigate to File works similarly by pressing **Ctrl+Shift+N** (**Cmd+Shift+O** for OS X) but is used for files and folders. To navigate to a folder, end your expression with the “/” character.

Navigate to Symbol is available by pressing **Ctrl+Alt+Shift+N** (**Alt+Cmd+O** for OS X) and allows you to find a method or field by its name.

STRUCTURE

When you are not switching between files you are probably navigating within a file. The shortest way to do this is to press **Ctrl+F12** (**Cmd+F12** for OS X). The popup shows you the structure of a file and allows you to quickly navigate to any item.

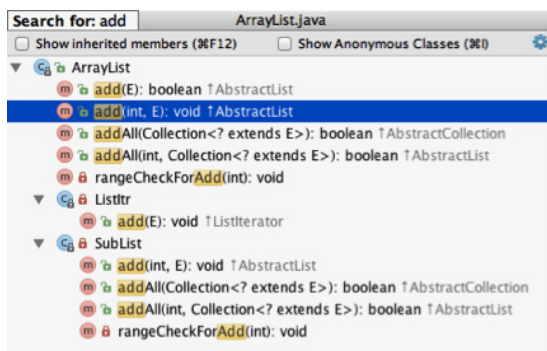


Figure 5: IntelliJ IDEA File Structure Navigation

SELECT IN

If you need to open the file in a particular tool window (or Finder/Explorer) you can do it via the Select In action by pressing **Alt+F1**.

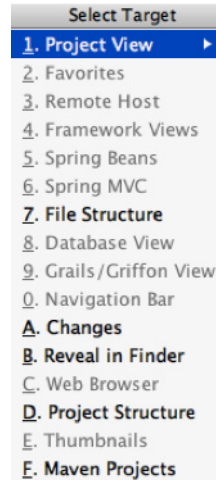


Figure 6: IntelliJ IDEA Select In

Navigation shortcuts include:

ACTION	WINDOWS	OS X
Search everywhere	Double Shift	⇧ x 2
Navigate to class	Ctrl+N	⌘O
Navigate to file	Ctrl+Shift+N	⇧ ⌘O
Navigate to symbol	Ctrl+Shift+Alt+N	⌘⇧O
Recent files	Ctrl+E	⌘E
File structure	Ctrl+F12	⌘F12
Select in	Alt+F1	⌘F1
Navigate to declaration	Ctrl+B	⌘B
Navigate to type hierarchy	Ctrl+H	^H
Show UML popup	Ctrl+Alt+U	⌘⇧U

QUICK POP-UPS

Quick Pop-ups are helpful for checking additional information related to the symbol at the caret. Here is a list of pop-ups you should know if you want to be more productive.

ACTION	WINDOWS	OS X
Documentation	Ctrl+Q	^J
Quick definition	Shift+Ctrl+I	⌘Space
Show usages	Alt+Ctrl+F7	⌘⇧F7
Show implementation	Ctrl+Alt+B	⌘⇧B

Quick Pop-ups are available for symbols in the editor; however, they are also available for items in any other list via the same shortcuts.

REFACTORING BASICS

IntelliJ IDEA offers a comprehensive set of automated code refactorings that lead to significant productivity gains when used correctly. You don't need to select anything before you apply a refactoring. IntelliJ IDEA is smart enough to figure out what statement you're going to refactor, and only asks for confirmation if there are several possible choices.

HOT TIP

To undo the last refactoring, switch your focus to the **Project** tool window and press **Ctrl+Z** (**Cmd+Z** for OS X).

ACTION	WINDOWS	OS X
Rename	Shift+F6	⇧F6
Extract variable	Ctrl+Alt+V	⌘⌥V
Extract field	Ctrl+Alt+F	⌘⌥F
Extract constant	Ctrl+Alt+C	⌘⌥C
Extract method	Ctrl+Alt+M	⌘⌥M
Extract parameter	Ctrl+Alt+P	⌘⌥P
Inline	Alt+Ctrl+N	⌘⌥N
Copy	F5	F5
Move	F6	F6
Refactor this	Ctrl+Shift+Alt+T	⇧⌘⌥T

A real time-saver is the ability to extract part of a string expression with the help of the Extract... refactorings. Just select a string fragment and apply a refactoring to replace all of the selected fragment usages with the introduced constant or variable.

FINDING USAGES

Find Usages helps you quickly find all pieces of code referencing the symbol at the caret (cursor), no matter if the symbol is a class, method, field, parameter, or another statement. Just press **Alt+F7** to get a list of references grouped by usage type, module, and file.

If you want to set custom options for the Find Usages algorithm, press **Shift+Alt+Ctrl+F7** (**Shift+Alt+Cmd+F7** for OS X) or click the first button on the right panel with search results. If you're looking for plain text, use Find in Path by pressing **Shift+Ctrl+F** (**Shift+Cmd+F** for OS X).

INSPECTIONS

Inspections are built-in static code analysis tools that help you find probable bugs, locate dead code, detect performance issues, and improve the overall code structure.

Most inspections not only tell you where a problem is, but provide quick fixes to deal with them right away. Press **Alt+Enter** to choose a quick-fix.

The editor lets you quickly navigate between the highlighted problems via keyboard shortcuts. Press **F2** to go to the next problem and **Shift+F2** to go to the previous one.

Inspections that are too complex to be run on the fly are available when you perform code analysis for the entire project. You can do this one of two ways: by selecting **Analyze → Inspect Code** from the menu, or by selecting **Analyze → Run Inspection by Name** to run the inspection by its name.

While inspections provide quick-fixes for code that has potential problems, intentions help you apply automatic changes to code that is correct. To get a list of intentions applicable to the code at the caret, press **Alt+Enter**.

CODE STYLE AND FORMATTING

IntelliJ IDEA automatically applies a code style you've configured in the **Settings → Code Style** menu as you edit, and in most cases you don't need to call the Reformat Code action explicitly.

Useful formatting shortcuts:

ACTION	WINDOWS	OS X
Reformat code	Ctrl+Alt+L	⌘⌥L
Auto-indent lines	Ctrl+Alt+I	⌘⌥I
Optimize imports	Ctrl+O	⌘⌥O

HOT TIP

By default, IntelliJ IDEA uses regular spaces for indents instead of tabs. If you have files with lots of indents, you may want to optimize their size by enabling the **Use tab character** option.

VERSION CONTROL BASICS

To check out a project from a Version Control System (VCS), click **Checkout from Version Control** on the Welcome Screen or in the **VCS** menu.

HOT TIP

To quickly perform a VCS operation on the current file, directory, or the entire project, use the **VCS operations popup** by pressing **Alt+`** (**Ctrl+V** for OS X).

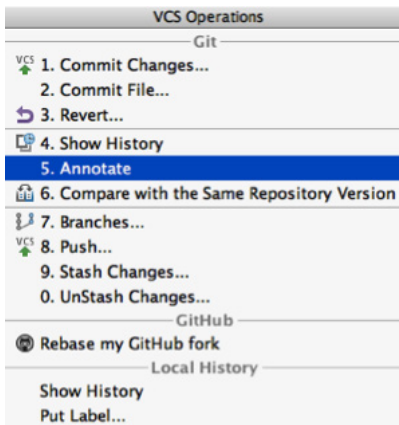


Figure 7: IntelliJ IDEA Version Control Operations

Once you’ve configured the VCS settings, you’ll see the Version Control tool window. You can switch to it anytime by pressing **Alt+9** (Cmd+9 for OS X).

The Local Changes tab of the Version Control tool window shows your local changes, both staged and unstaged.

Useful VCS shortcuts:

ACTION	WINDOWS	OS X
Version control tool window	Alt+9	⌘9
VCS operations popup	Alt+`	^V
Commit changes	Ctrl+K	⌘K
Update project	Ctrl+T	⌘T
Push commits	Ctrl+Shift+K	⇧⌘K

HOT TIP

Annotation (available from both the quick list, and the main and context menus) allows you to see who changed a line of code for any file and when it changed.

BRANCHES

To perform an operation on a branch, use either **Branches** from the main or context menu, the **VCS operations popup**, or the widget on the right of the status bar.

For multiple repositories, IntelliJ IDEA performs all VCS operations on all branches simultaneously, so you don’t need to switch between them manually.

Shelves, Stashes, and Patches help you when you need to store some of the local changes without committing them to the repository. You can then switch to the repository versions of the files, and then come back to your changes later.

MAKE

By default, IntelliJ IDEA doesn’t automatically compile projects

on saving. To compile a project, select **Build → Make Project** and press **Ctrl+F9** (Cmd+F9 for OS X).

RUNNING AND DEBUGGING

Once you’ve created a **Run/Debug configuration** by selecting **Run → Edit Configurations...**, you’re able to run and debug your code.

ACTION	WINDOWS	OS X
Run	Shift+F10	F10
Debug	Shift+F9	⇧F10

While in the debug mode, you can evaluate any expression by using the **Evaluate expression** tool, which is accessed by pressing **Alt+F8**. This tool provides code completion just as in the editor, so it’s easy to enter any expression.

Sometimes you may want to step into a particular method but not the first one which will be invoked. In this case, use **Smart step into** by pressing **Shift+F7** (Cmd+F7 for OS X) to choose a particular method.

ACTION	WINDOWS	OS X
Toggle breakpoint	Ctrl+F8	⌘F8
Step into	F7	F7
Smart step into	Shift+F7	⇧F7
Step over	F8	F8
Step out	Shift+F8	⇧F8
Resume	F9	⌘R
Evaluate Expression	Alt+F8	⌘F8

If you want to “rewind” while debugging you can do it via the **Drop Frame** action. This is particularly helpful if you mistakenly stepped too far. This will not revert the global state of your application, but will at least let you revert to a previous stack frame.

HOT TIP

Any breakpoint can be quickly disabled by clicking on the gutter while holding **Alt**. To change breakpoint details (e.g. conditions), press **Shift+Ctrl+F8** (Shift+Cmd+F8 for OS X).

RELOADING CHANGES AND HOT SWAPPING

Sometimes you need to insert minor changes into your code without shutting down the process. Since the Java VM has a HotSwap feature, the IDE handles these cases automatically when you call **Make**.

APPLICATION SERVERS

To deploy your application to a server:

1. Configure your artifacts by selecting **Project Structure** → **Artifacts** (done automatically for Maven and Gradle projects).
2. Configure an application server by selecting **Settings** → **Application Servers**.
3. Create a run configuration by selecting **Run** → **Edit Configurations...**, then specify the artifacts to deploy and the server to deploy it to.

You can always ask the IDE to build/rebuild your artifacts (once they're configured) by selecting **Build** → **Build Artifacts...**


When you need to apply changes in the code to a running application, in addition to **Make** you can use the **Update** action by pressing **Ctrl+F10** (Cmd+F10 for OS X). This action is only available for the Exploded artifact type. Based on your choice, it can update resources or update classes and resources. When the **Update** action is applied in the **Debug** mode, it uses **HotSwap**; otherwise, it uses **Hot redeployment**.

WORKING WITH BUILD TOOLS (MAVEN/GRADLE)

Once you've imported/created your Maven/Gradle project, you are free to edit its pom.xml/build.gradle files directly via the editor. Any changes to the underlying build configuration will eventually need to be synced with the project model in IntelliJ IDEA.

If you want the IDE to synchronize your changes immediately, perform the following:

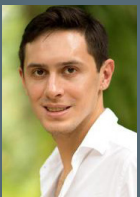
- For pom.xml, enable the corresponding option in **Settings** → **Build, Execution, Deployment** → **Build Tools** → **Maven** → **Importing** → **Import Maven projects automatically**.
- For build.gradle, enable the corresponding option in **Settings** → **Build, Execution, Deployment** → **Build Tools** → **Gradle** → **Use auto-import**.

For manual synchronization, use the corresponding action on the Maven/Gradle tool window toolbar: .

HOT TIP

Any goal or task can be attached to be run before a Run Configuration.

ABOUT THE AUTHOR



Andrey Cheptsov has been professionally writing software for more than 10 years. His interests and expertise include web development and JVM languages. He's passionate about programming and productivity—and the tools that help you achieve these. Andrey is an avid learner, an IntelliJ IDEA guru, and a fan of the Kotlin language.

CREDITS

G. Ryan Spain	Editor
Yassee Mohebbi	Designer
Chris Smith	Production
Chris Brumfield	Sponsor Relations
Chelsea Bosworth	Marketing

BROWSE OUR COLLECTION OF 250+ FREE RESOURCES, INCLUDING:

RESEARCH GUIDES: Unbiased insight from leading tech experts

REFCARDZ: Library of 200+ reference cards covering the latest tech topics

COMMUNITIES: Share links, author articles, and engage with other tech experts

JOIN NOW



DZone communities deliver over 6 million pages each month to more than 3.3 million software developers, architects and decision makers. DZone offers something for everyone, including news, tutorials, cheat sheets, research guides, feature articles, source code and more.

"DZONE IS A DEVELOPER'S DREAM," SAYS PC MAGAZINE.

Copyright © 2015 DZone, Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.

DZONE, INC.
 150 PRESTON EXECUTIVE DR.
 CARY, NC 27513
 888.678.0399
 919.678.0300

REFCARDZ FEEDBACK WELCOME
 refcardz@dzone.com

SPONSORSHIP OPPORTUNITIES
 sales@dzone.com



VERSION 1.0 \$7.95

EXCEL AT ENTERPRISE & MOBILE DEVELOPMENT WITH A SMARTER IDE



IntelliJ IDEA
The most intelligent Java IDE

GET IT NOW