

# LEARNING intellij-idea

Free unaffiliated eBook created from **Stack Overflow contributors.** 

## **Table of Contents**

About	
Chapter 1: Getting started with intellij-idea	2
Remarks	2
Examples	2
Installation or Setup	2
Windows & Linux	2
OS X / macOS	2
Arch Linux	3
Ubuntu	3
Other	4
Hello, World!	4
Migrating from Eclipse	14
Chapter 2: Database Tools	20
Examples	20
Creating new data source	20
Chapter 3: Exporting	26
Examples	26
Building a .jar	26
Chapter 4: Hibernate HQL console and inspections	30
Introduction	30
Examples	30
Configuring HQL Inspections	30
Chapter 5: How to Install Plugins	31
Introduction	31
Examples	31
To download and install a repository plugin	31
To install a plugin from the disk	31
Chapter 6: IdeaVim	34
Remarks	34

Examples	34
Showing line numbers	34
Allocating conflicting keystrokes to IdeaVim	34
Chapter 7: Inspections	36
Introduction	36
Examples	36
@NotNull / @Nullable inspections	36
Chapter 8: Little-known features	38
Examples	38
Language injection	38
Quick Preview	38
Check screencast video on YouTube about this feature	38
Chapter 9: Live Templates	42
Examples	42
Add a test method easily	42
Insert the name of the current class	43
Chapter 10: Optimization	45
Examples	45
Customizing the VM Options	45
Chapter 11: Useful Shortcuts	46
Examples	46
Compile and Run	46
Make project (compile modifed and dependent)	46
Windows: Ctrl + F9	46
OS X / macOS: Cmd + F9	46
Compile selected file, package or module	46
Windows: Ctrl + Shift + F9	46
OS X / macOS: Cmd + Shift + F9	46
Select configuration and run	46
Windows: Alt + Shift + F10	
OS X / macOS: Option + Shift + F10	

Select configuration and debug	46
Windows: Alt + Shift + F9	46
OS X / macOS: Option + Shift + F9	46
Run	46
Debug	47
Run context configuration from editor	47
Windows: Ctrl + Shift + F10	47
OS X / macOS: Cmd + Shift + F10	47
Code Completion	47
Basic code completion (the name of any class, method or variable)	47
Windows: Ctrl + Space	47
OS X / macOS: Cmd + Space	47
Smart code completion (filters the list of methods and variables by expected type)	47
Windows: Ctrl + Shift + Space	47
OS X / macOS: Cmd + Shift + Space	47
Overwriting code with a suggestion	47
Adding code from a completion suggestion	47
Search/Replace	48
Search everywhere	48
Find	48
Windows / Linux: Ctrl + F	48
OS X / macOS: Cmd + F.	48
Find next	48
Find previous	48
Replace	48
Windows / Linux: Ctrl + R	48
OS X / macOS: Cmd + R	48
Find in path	48
Windows / Linux: Ctrl + Shift + F	
OS X / macOS: Cmd + Shift + F	48

Replace in path	48
Windows / Linux: Ctrl + Shift + R	49
OS X / macOS: Cmd + Shift + R	49
Refactoring	49
Copy	49
Move	49
Safe delete	49
Windows / Linux: Alt + Delete	49
OS X / macOS: Cmd + Delete	49
Rename	49
Extract Method	49
Windows / Linux: Ctrl + Alt + M	49
OS X / macOS: Cmd + Option + M.	49
Extract Field	49
Windows / Linux: Ctrl + Alt + F	50
OS X / macOS: Cmd + Option + F.	50
Extract Variable	50
Windows / Linux: Ctrl + Alt + V	50
OS X / macOS: Cmd + Option + V.	50
Extract Constant	50
Windows / Linux: Ctrl + Alt + C	50
OS X / macOS: Cmd + Option + C	50
Extract Parameter	50
Windows / Linux: Ctrl + Alt + P	50
OS X / macOS: Cmd + Option + P.	50
Other	50
Surround with	50
Windows / Linux: Ctrl + Alt + T	50
OS X / macOS: Cmd + Option + T	50
Basic navigation	50

Go to editor (from tool window) 50
Switching focus to corresponding tool window51
Windows: Alt + <tool number="" window="">51</tool>
OS X / macOS: Cmd + <tool number="" window="">51</tool>
For example switching focus to the project window51
Windows: Alt + 151
OS X / macOS: Cmd + 1
Recent files popup51
Windows: Ctrl + E
OS X / macOS: Cmd + E
Find Action51
Windows: Ctrl + Shift + A
OS X / macOS: Cmd + Shift + A
Navigate to
File :51
Windows: Ctrl + Shift + N51
OS X / macOS: Cmd + Shift + N
Class:
Windows: Ctrl + N
OS X / macOS: Cmd + N
Symbol (class/method/variable/constant name):52
Windows: Ctrl + Alt + Shift + N
OS X / macOS: Cmd + Option + Shift + N
Everywhere :
Windows: Shift + Shift
OS X / macOS: Shift + Shift
Go to line number
Windows: Ctrl + G
OS X / macOS: Cmd + L
Go back to last edit location

Windows: Ctrl + Shift + Backspace	53
OS X / macOS: Cmd + Shift + Backspace	53
Usage Search	53
Find usages / Find usages in file	53
Windows / Linux: Alt + F7 / Ctrl + F7	53
OS X / macOS: Option + F7 / Ctrl + F7	53
Highlight usages in file	53
Windows / Linux: Ctrl + Shift + F7	53
OS X / macOS: Cmd + Shift + F7	53
Show usages	53
Windows / Linux: Ctrl + Alt + F7	53
OS X / macOS: Cmd + Option + F7	53
Show Method Parameters	53
Windows / Linux: Ctrl + P.	53
OS X / macOS: Cmd + P.	53
Selection	54
Windows: Ctrl + W	54
OS X / macOS: Cmd + W	54
Windows: Ctrl + Shift + W	54
OS X / macOS: Cmd + Shift + W	54
Windows: Alt	54
OS X / macOS: Opt	54
Windows: Alt + Shift	55
OS X / macOS: Opt + Shift	55
Windows: Alt + J	55
OS X / macOS: ctrl + G.	55
Credits	57

# **About**

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: intellij-idea

It is an unofficial and free intellij-idea ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official intellij-idea.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# Chapter 1: Getting started with intellij-idea

### Remarks

intellij-idea is an IDE made as the spiritual successor for the widely-adopted Eclipse IDE used for Java development. Eclipse, although extremely powerful, is often criticized for being extremely clunky and difficult to use.

Intellij IDEA attempts to build an IDE with similar power to Eclipse, but with a finishing polish on top. Developers would be at an advantage using IDEA because of the many tools and hooks it has to save time on all projects. Smart code completion, native unit test integration and native Gradle management are just a few of the highlights of Jetbrain's Java IDE

## **Examples**

#### **Installation or Setup**

There are two main versions of IntelliJ IDEA: the Community edition and the Ultimate edition. The Community edition is free and is not lacking for features in terms of Java SE development.

## **Windows & Linux**

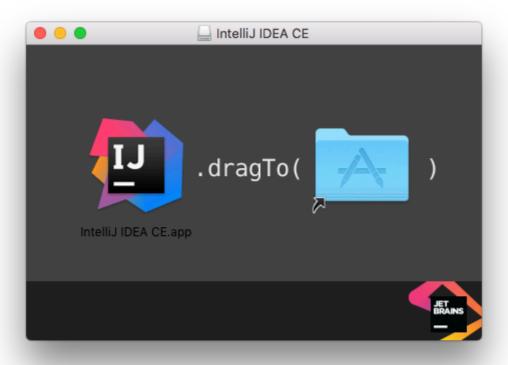
Download IntelliJ IDEA from the JetBrains website, and follow installation procedures. If the Java Development Kit (JDK) is not installed, download and install the JDK. Note that you need the JDK, only having the Java Runtime Environment (JRE) is not enough.

Once IntelliJ IDEA has been downloaded:

- 1. Run the installer
- 2. Press next
- 3. Choose a folder to install IntelliJ IDEA to (In most cases, leave this as the default)
- 4. Choose a start menu folder to crete IntelliJ IDEA shortcuts (In most cases, leave this as the default)
- 5. Choose whether to create a desktop shortcut, and choose whether to associate various Java files with IntelliJ IDEA
- 6. Press next, and wait for it to install

# OS X / macOS

Download IntelliJ IDEA from the JetBrains website, open the disk image (\*.dmg) file downloaded, and drag and drop the application to the alias to your /Applications folder.



## **Arch Linux**

IntelliJ IDEA can be installed on Arch Linux using its package manager, pacman. Open a terminal and enter the following command.

```
sudo pacman -S intellij-idea-community-edition
```

Using sudo is not required if you're running as the root user.

```
@ArchLinux sudo pacman -S intellij-idea-community-edition
[sudo] password for resolving dependencies...
:: There are 2 providers available for java-environment:
:: Repository extra
1) jdk7-openjdk 2) jdk8-openjdk

Enter a number (default=1): 2
looking for conflicting packages...

Packages (7) intellij-idea-libs-2:2016.2-1 java-environment-common-2-2 java-runtime-common-2-2 jdk8-openintellij-idea-community-edition-2:2016.2-1

Total Download Size: 251.91 MiB
Total Installed Size: 593.61 MiB
```

#### \_\_\_\_ Ubuntu

(1) Install ubuntu-make package.

For ubuntu 16.04 and later,

```
sudo apt install ubuntu-make
```

For previous versions of ubuntu,

```
sudo add-apt-repository ppa:ubuntu-desktop/ubuntu-make
sudo apt-get update
sudo apt-get install ubuntu-make
```

(2) After installing Ubuntu Make, do a

```
umake ide idea
```

Default installation path: /home/current-user/.local/share/umake/ide/idea

Follow hello\_world project listed above.

Follow ubuntu-make page to change default installation and to install other IDEs.

# **Other**

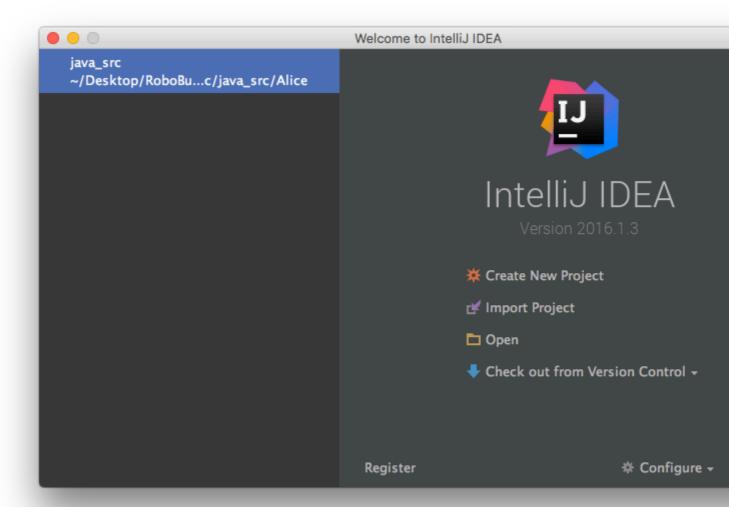
Further installation details can be found here:

https://www.jetbrains.com/help/idea/2016.1/installing-and-launching.html

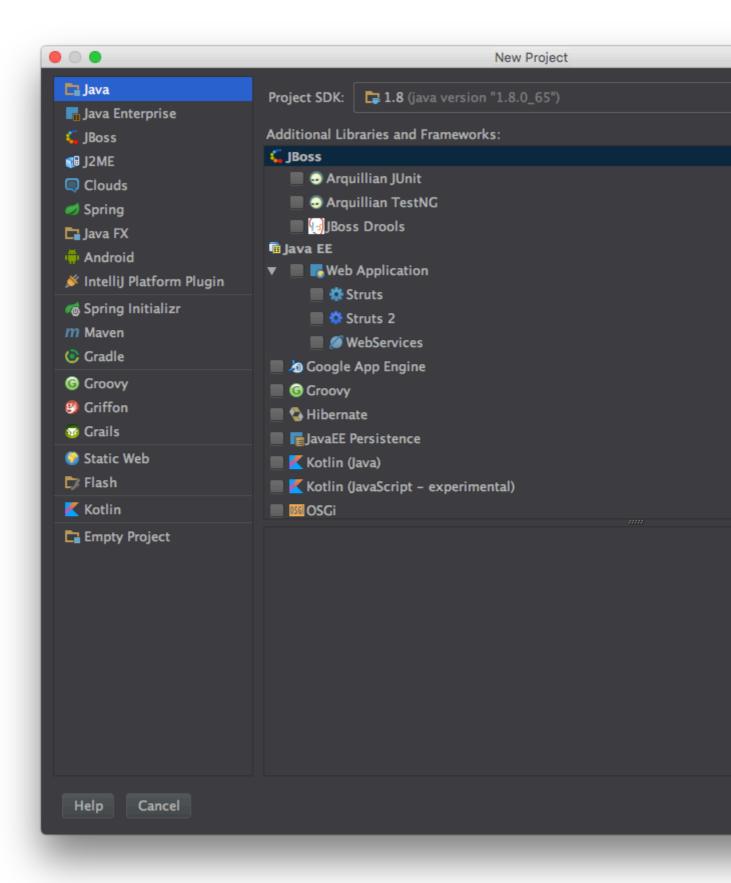
Hello, World!

This will teach you how to make your first project using IDEA.

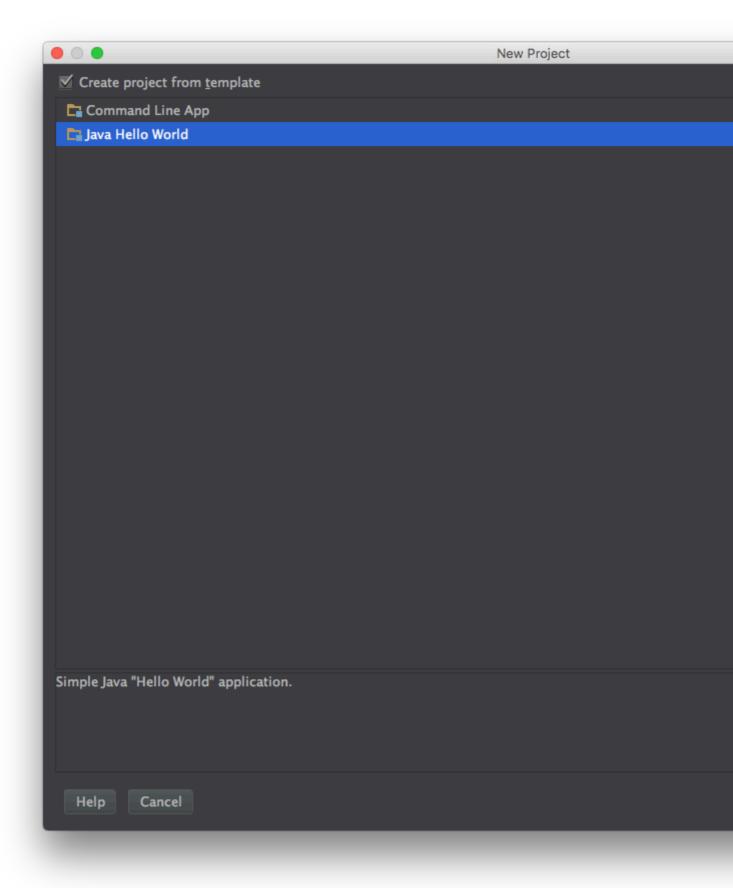
Launch IDEA, and click Create New Project from the startup screen:



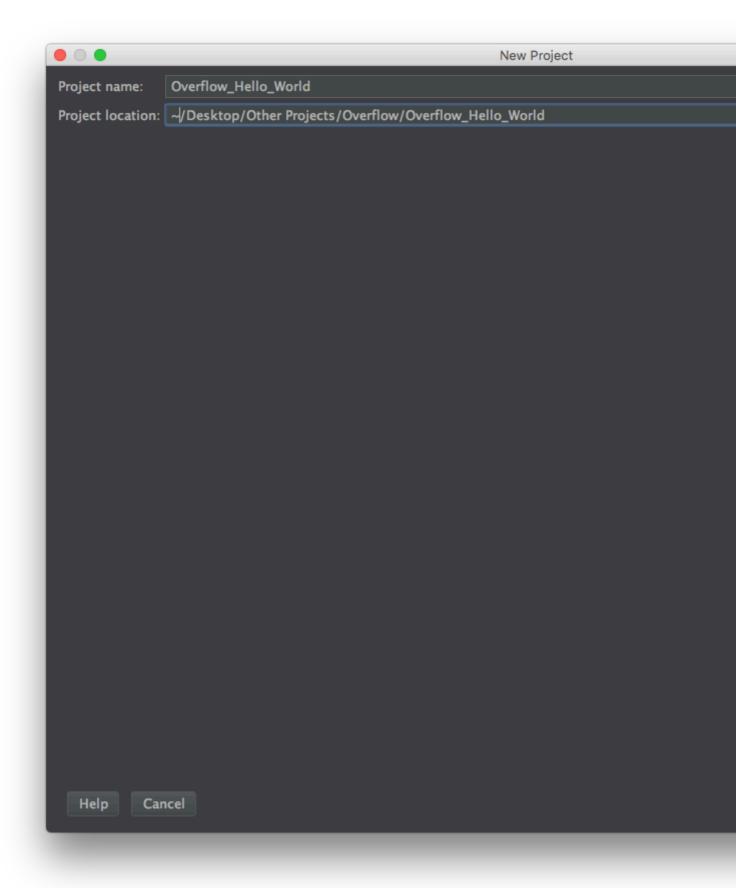
Click  $_{\text{Next}}$  on the next screen. We're creating a simple Java project, so we don't need any addons or extras to this project



Use the next screen to create the Java Hello World template project:



Finally, name your project and select a location on disk, and click  ${\tt Finish}$ :



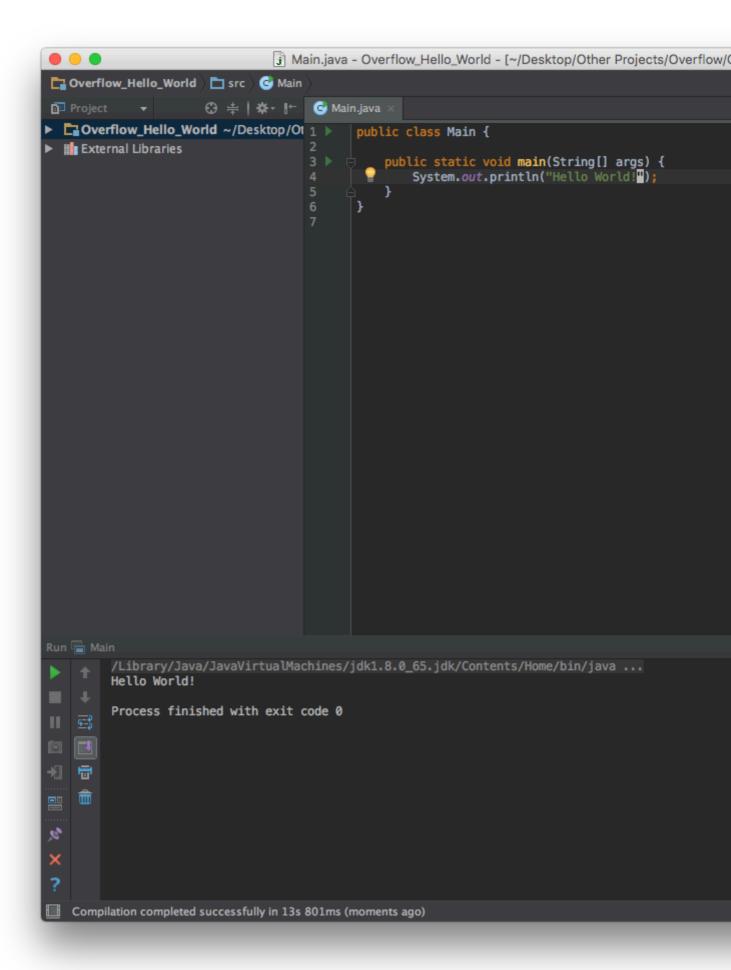
You should end up with a window that looks something like this:

```
i Main.java - Overflow_Hello_World - [~/Desktop/Other Projects/Overflow/G
Overflow_Hello_World
                       Project
▶ ☐ Overflow_Hello_World ~/Desktop/Ot 1 ▶ public class Main {
► III External Libraries
                                                 public static void main(String[] args) {
    System.out.println("Hello World!");
```

t this point, the project is all ready to go, simply click the Run button or go to Run -> Run 'Main'	

```
i Main.java - Overflow_Hello_World - [~/Desktop/Other Projects/Overflow/G
Overflow_Hello_World
                       Project
▶ ☐ Overflow_Hello_World ~/Desktop/Ot 1 ▶ public class Main {
► III External Libraries
                                                 public static void main(String[] args) {
    System.out.println("Hello World!");
```

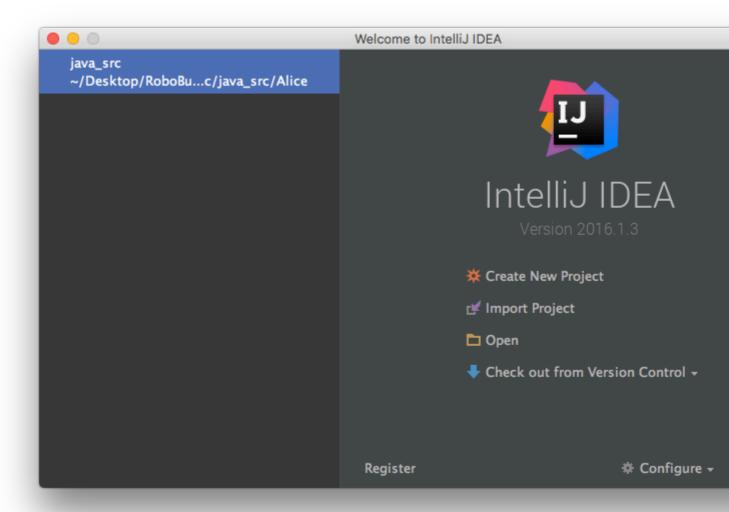
And you're done! The console will automatically pop up, giving it's salutations to the globe!



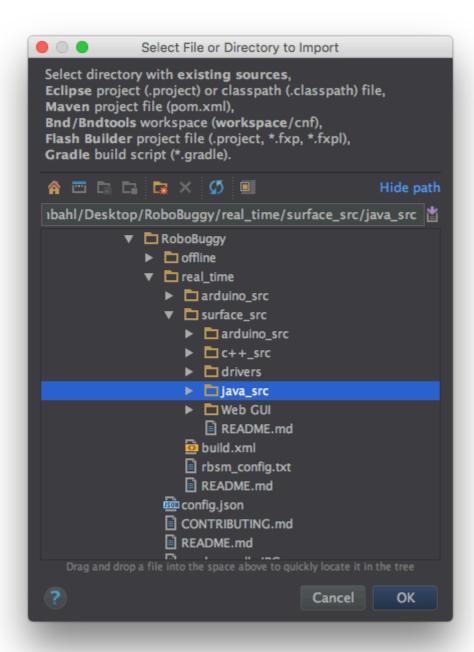
## **Migrating from Eclipse**

Intellij IDEA attempts to appeal to the wide Java fanbase which uses Eclipse for their development by allowing developers to migrate their Eclipse projects over to an IDEA structure with a few simple clicks!

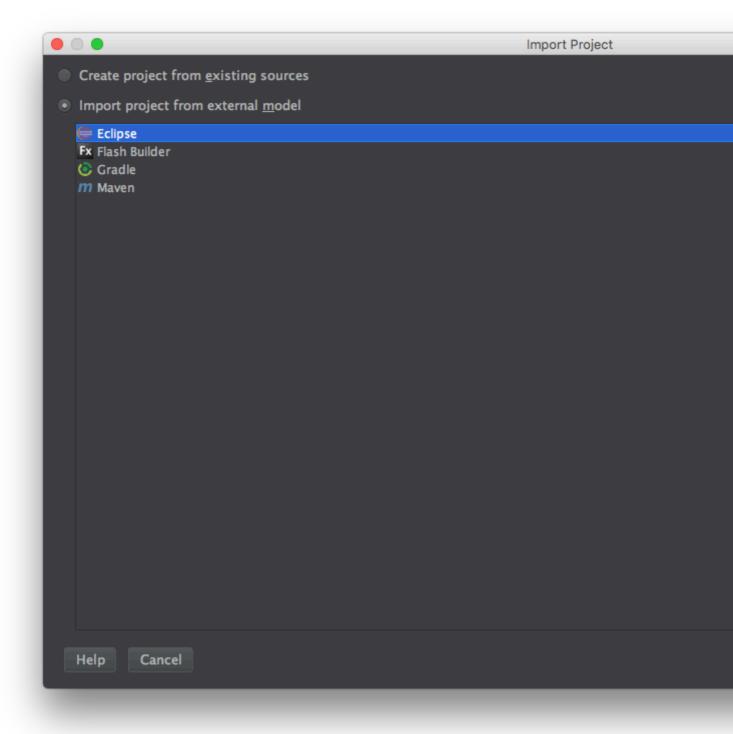
First, start IDEA and click Import Project from the startup window:



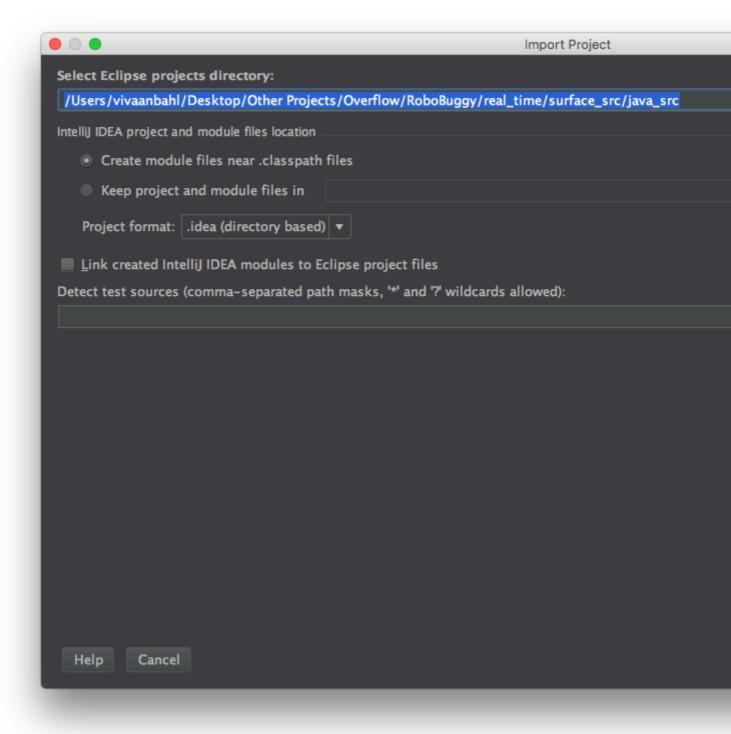
Then, select your Eclipse project using the explorer window



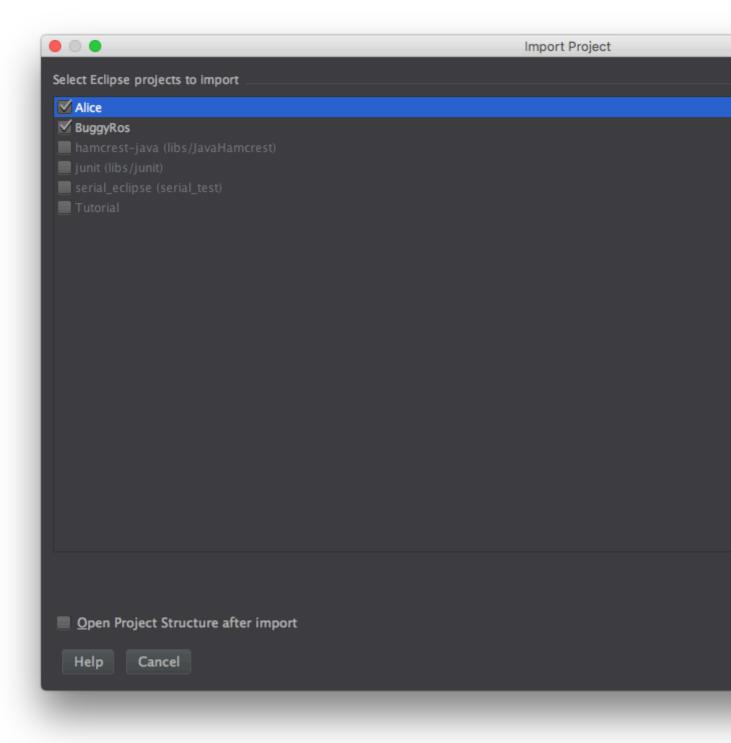
Intellij will prompt you for the model you are importing from, make sure  $\mathtt{Eclipse}$  is selected before clicking  $\mathtt{Next}$ 



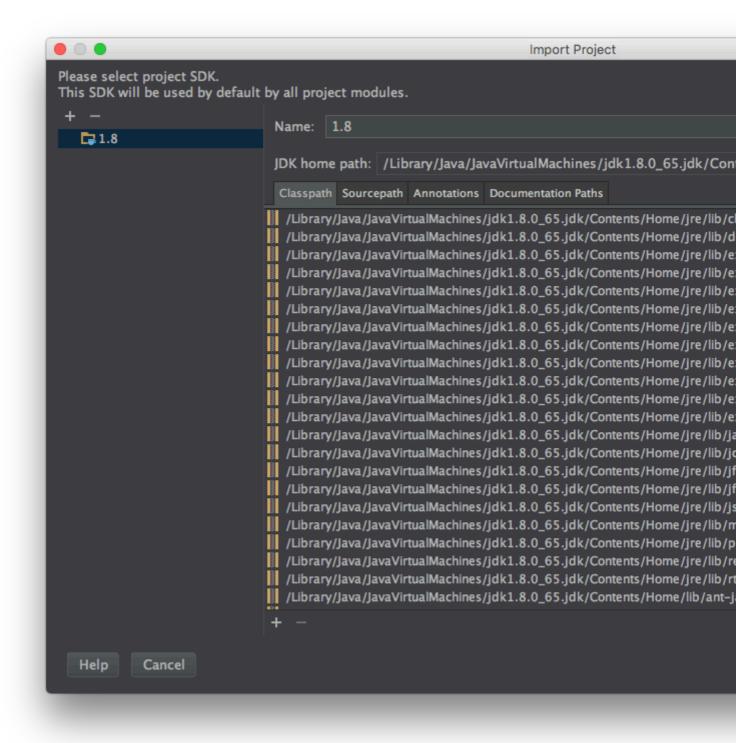
The next screen will show a confirmation of the path you want to import, simply click  $\mathtt{Next}$ :



Next, select the modules you want created. In the particular example project, only the Alice and the BuggyRos projects mattered when working in Eclipse



Finally, make sure the correct version of the JDK is selected before  ${\tt Finish}ing$ 



And the Eclipse project has been fully migrated to Intellij! The project will still open in both IDEs, and will be fully functional in both

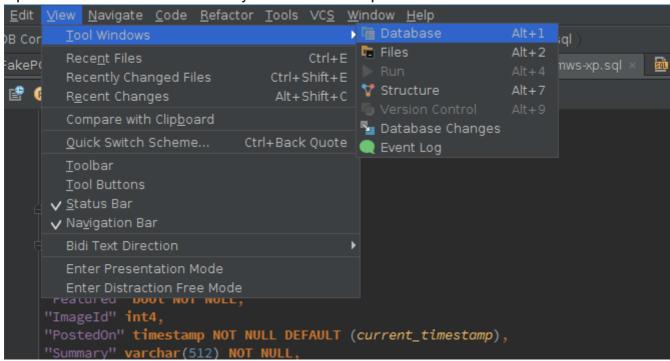
Read Getting started with intellij-idea online: https://riptutorial.com/intellij-idea/topic/1734/getting-started-with-intellij-idea

# **Chapter 2: Database Tools**

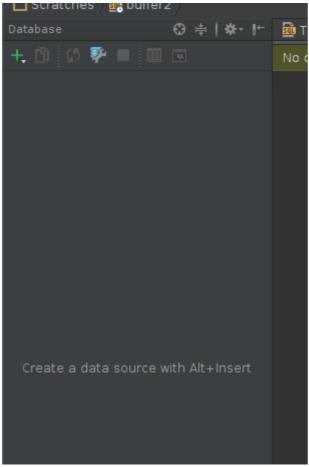
## **Examples**

Creating new data source

1. Open "Database Tool Window" if you don't have it opened:

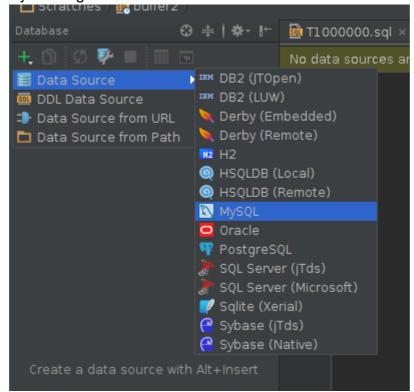


This what it looks like:

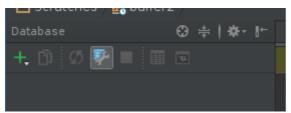


#### 2. Create new data source:

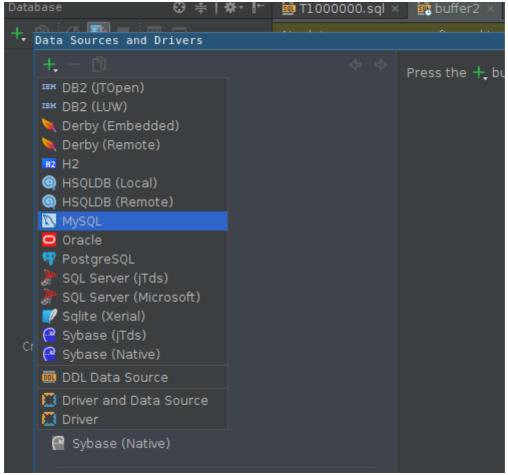
• By clicking on "+" icon:



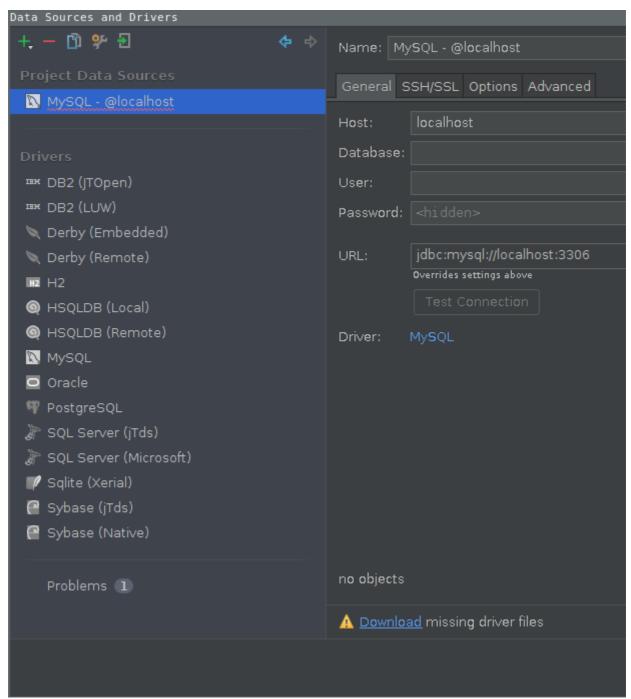
• Or by opening "Data Sources and Drivers" dialog



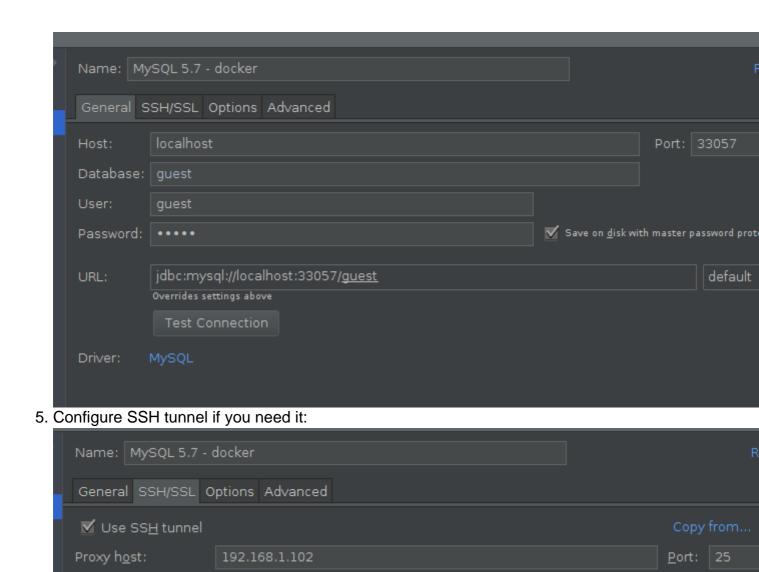
And adding new data source by clicking "+" there



3. If you have not already downloaded JDBC driver, you will be prompted to do it:



4. Enter connection parameters:



Save on disk with master password protection

Key pair (OpenSSH) ▼

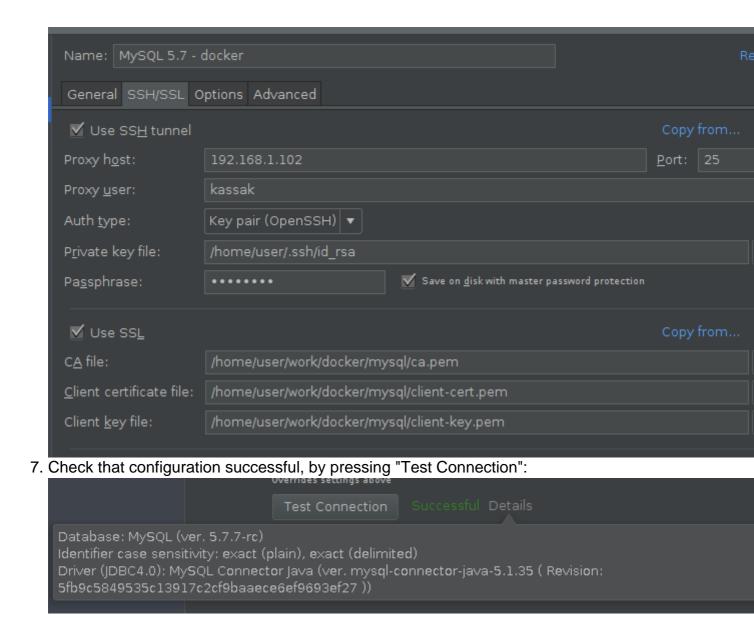
6. Configure SSL if you need it:

Proxy user:

Auth type:

P<u>r</u>ivate key file:

Passphrase:



Read Database Tools online: https://riptutorial.com/intellij-idea/topic/6275/database-tools

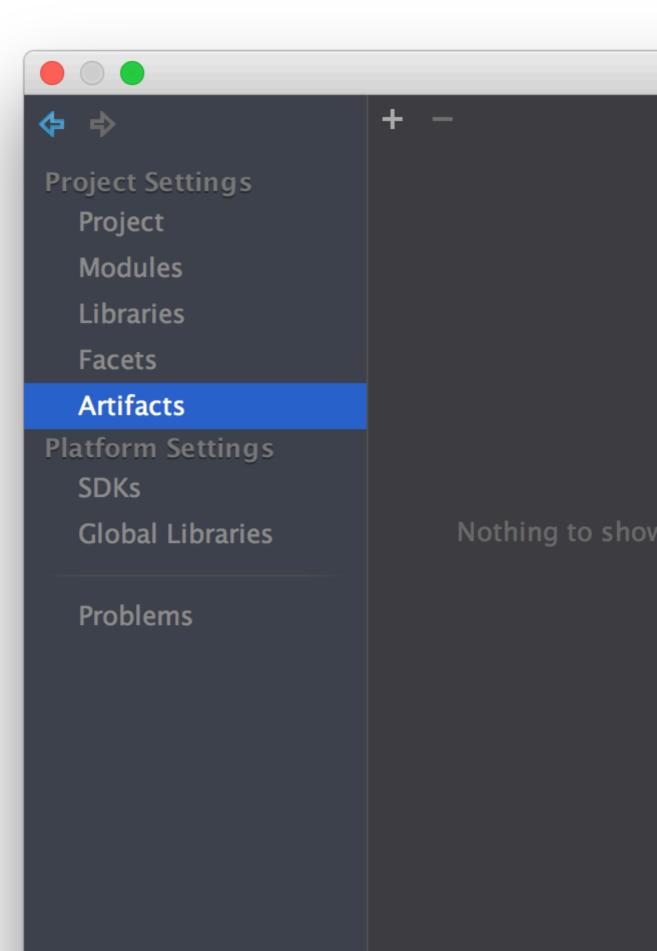
# **Chapter 3: Exporting**

## **Examples**

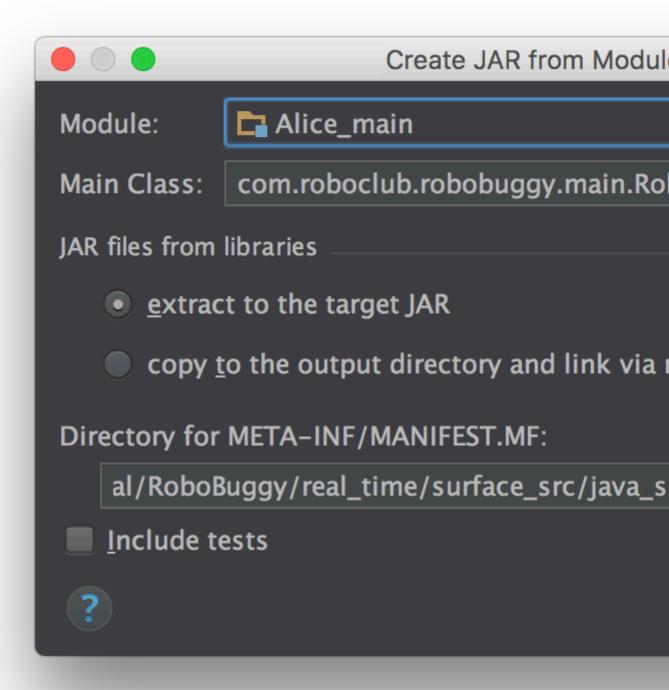
## Building a .jar

Eventually, when you're ready to release a version of your code to production, you'll need a .jar file to distribute. Intellij makes building JARs quite easy.

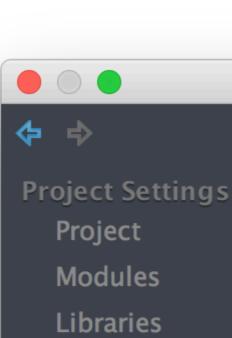
First, navigate to File -> Project Structure and click on Artifacts:



public static void main() method):



Click ok, verify that all the information regarding dependencies is correct, and click ok to finish setting up the artifact.



**Artifacts** 

**Facets** 

Platform Settings SDKs

Global Libraries

**Problems** 



**Alice\_main:jar** 

# **Chapter 4: Hibernate HQL console and inspections**

#### Introduction

Intellij IDEA supports HQL auto completion and running HQL queries on console. This is how you enable that support.

#### **Examples**

#### **Configuring HQL Inspections**

- 1. Go to File -> Project Structure -> Modules.
- 2. Add new Hibernate module.
- 3. Right click on the desired module -> Add -> Hibernate.
- 4. Select the newly created Hibernate configuration option, and click the (+) sign in the right pane to create hibernate.cfg.xml file.
- 5. Go to File -> Project Structure -> Facets, and add new JPA.
- 6. Select the newly created JPA configuration option, and click the (+) sign in the right pane to assign it your Hibernate configuration file.
- 7. Open Persistence window, there you should see the list of your project modules.
- 8. Expand the module name, and assign your data source to the hibernate.cfg.xml file.

Now you can write queries on hibernate console and get HQL auto completion.

Read Hibernate HQL console and inspections online: https://riptutorial.com/intellijidea/topic/9617/hibernate-hql-console-and-inspections

# **Chapter 5: How to Install Plugins**

#### Introduction

Plugins help us to do things easier. Intellij provides a vast range of plugins for various technologies / laguages. There are three ways to install plugin in Intellij.

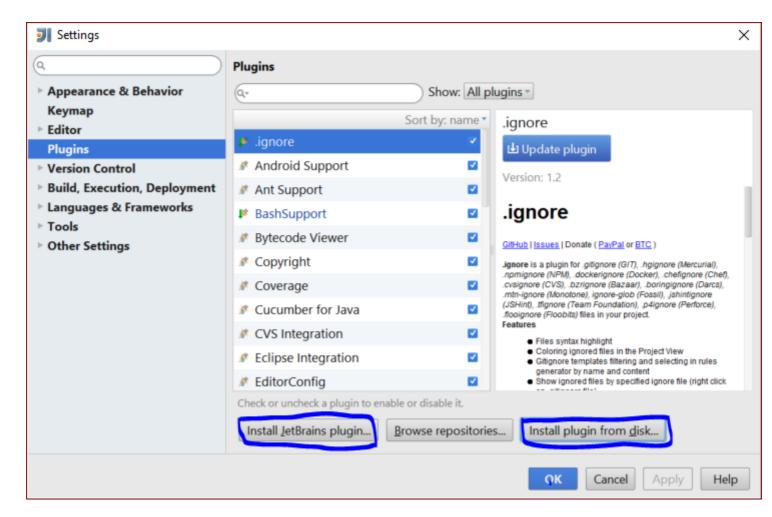
#### **Examples**

#### To download and install a repository plugin

```
    Go to File --> Settings (e.g. Ctrl+Alt+S).
    In the left-hand pane, select Plugins.
    On the Plugins window, click "Install JetBrains plugin" or the "Browse repositories button".
```

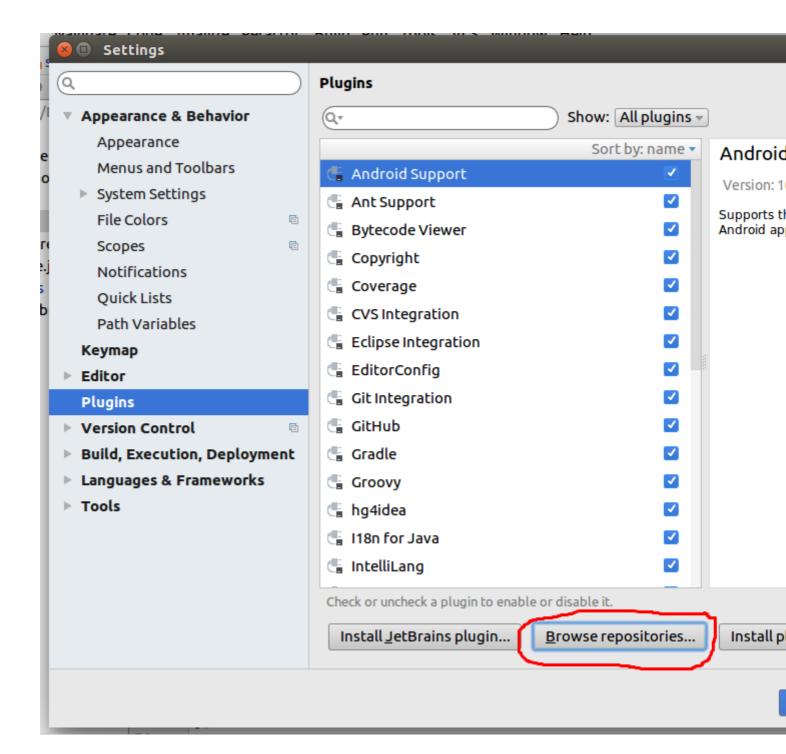
#### To install a plugin from the disk

```
    Go to File --> Settings (e.g. Ctrl+Alt+S).
    In the left-hand pane, select Plugins.
    On the Plugins window, click "Install plugin from disk button".
    Select the desired plugin from your local machine.
    Click Apply button of the Settings/Preferences dialog.
```



Or you can directly add pugins from the other venders by directly searching the repositories.

```
    Go to "Browse Repositories"
    Select the category (on the top of the window) that you need to search(or just search by the name if you know it).
    Install it.
```



Read How to Install Plugins online: https://riptutorial.com/intellij-idea/topic/8069/how-to-install-plugins

# **Chapter 6: IdeaVim**

#### **Remarks**

IdeaVim is a plugin for IDEA products that aims in providing Vim functionality in editor views

#### **Examples**

#### **Showing line numbers**

As of IntelliJ IDEA version 2016.2, and IdeaVim version 0.46, IntelliJ's native option for showing line numbers is ineffective. When clicking *Show line numbers*, the line numbers immediately show and disappear.

This problem is caused by a bug in the IdeaVim plugin, which can be resolved by using the Vim command for showing line numbers:

:set number

#### and

:set nonumber

to hide.

These commands can also be used as the shorthand :set nu and :set nonu.

If you wish to activate the feature which shows relative line numbers instead you can use

:set relativenumber

or a shorthand :set rnu. Remember that you can mix set relativenumber with set number.

#### Allocating conflicting keystrokes to IdeaVim

By default, some keystrokes that are useful in Vim contradict with the keystrokes of IntelliJ.

For example, AR in Vim is 'redo', but in IntelliJ it's the shortcut for Run

To decide which program interprets the keystroke, go to Preferences -> Other Settings -> Vim Emulation and choose which keystrokes to use with IdeaVim and which to use with IntelliJ:



- Appearance & Behavior Keymap
- Editor
  Plugins
- Version Control
- **▶** Build, Execution, Deployment
- Languages & Frameworks
- **▶** Tools
- ▼ Other Settings

Vim Emulation

## Other Settings > Vi

### Shortcut Conflicts for A

	Shortcut
^2	
^企2	
^企6	
^G	
^H	
^M	
^N	
^0	
^V	

# **Chapter 7: Inspections**

#### Introduction

Intellij IDEA provides lots of code inspections, that can significantly simplify writing code.

Inspections parameters can be found in Preferences | Editor | Inspections section. By default, IDEA has lots of them enabled. And lots of inspections support auto-fixing options, that can be seen on pressing Alt + Enter.

To run inspections for your whole project (or some custom scope), you need to select Analyze | Inspect code.

#### **Examples**

#### @NotNull / @Nullable inspections

These inspections are extremely useful for preventing NullPointerExceptions. By default they are disabled. You can find these inspections in Inspections preferences: Java | Probable bugs | Constant conditions & exceptions and @NotNull/@Nullable problems. There you can also configure your annotations. You can use this manual to add JetBrains annotations into your project.

For example, consider this methods:

```
public int getLength() {
    return getString().length();
}

public String getString() {
    return "not null string";
}
```

If getString can't possibly return null, everything is fine. But if we enable our inspections and in some cases it can return null, we will immediately will see inspection triggered:

```
public int getLength() {
    return getString().length();
}

public String getString() {
    if (condition()) {
       return null;
    }
    return "not null string";
}
```

which says 'null' is returned by the method which is not declared as @Nullable. And if we hit Alt + Enter, there will be an option Annotate method as '@Nullable'. If we hit Enter again, our code will look like that:

```
public int getLength() {
    return getString().length();
}

@Nullable
public String getString() {
    if (condition()) {
        return null;
    }
    return "not null string";
}
```

with inspection triggered on <code>length()</code> method saying <code>Method invocation 'length' may produce 'java.lang.NullPointerException'. And if we go further and introduce the result of <code>getString()</code> method as a variable, after hitting Alt + Enter IDEA will suggest a few ways to fix this inspection:</code>

This way you can inspect your code on-the-fly, and fix all potential NullPointerExceptions. If you want to check your whole project (or some random scope), you can use Analyze | Inspect code. Just make sure that your selected inspections profile has all necessary inspections enabled.

Read Inspections online: https://riptutorial.com/intellij-idea/topic/9361/inspections

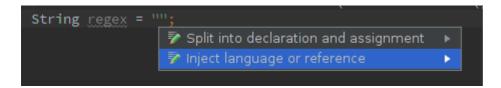
# **Chapter 8: Little-known features**

#### **Examples**

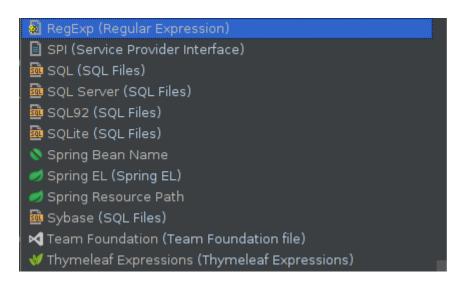
#### Language injection

If you want to write strings containing other languages (JSON, regexes), it's hard to keep up with escaping symbols, and it would be nice to get some code assist.

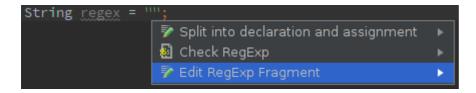
- 1. Put your cursor inside an empty string
- 2. ALT + ENTER
- 3. Pick "Inect language or reference"



4. Pick the desirable language (RegExp in my case) from the pop-up



5. Again use ALT + ENTER and pick Edit regex fragment



6. In the new tool window enter the regex - note how it's automatically mapped to a properly escaped Java string. Similarly for JSON the indents will be placed properly.

**Quick Preview** 

### Check screencast video on YouTube about this feature

IntelliJ provides a quick-preview feature called Viewing Definition. Using this feature allows a user to quickly see the contents of a method/class without navigating into the class itself

```
• Unix / Windows - Ctrl+Shift+I
Example: Looking into Arrays.copyOfRange():
        if (offset > value.length - count) {
            throw new StringIndexOutOfBoundsException
      this.value = Arrays.copyOfRange(value, offset
                                                 Definitio
                   C Arrays.java
     public static char[] copyOfRange(char[] original
         int newLength = to - from;
       if (newLength < 0)</pre>
              throw new IllegalArgumentException(from
        char[] copy = new char[newLength];
        System.arraycopy(original, from, copy, 0,
                            Math.min(original.length -
      return copy;
```

Verifying if you've selected the right class in search:

• OS X - ([]+Y) or ([]+Space)

```
String.java
    public final class String
        implements java.io.Serializable, Comparable<</pre>
      /** The value is used for character storage.
     private final char value[];
ties
      .../** Cache the hash code for the string */
     ....private int hash; // Default to 0
     /** use serialVersionUID from JDK 1.0.2 for i
     private static final long serialVersionUID =
                 String is special cased within the S
                           // Note: of
                                        Enter class n
                             if (offset
                                 throw r
                                         Q string
         String (java.lang)
    😋 🚡 String (com.sun.org.apache.xpath.internal.op
    STRING in DocFlavor (javax.print)
    O String (scala.math.Ordering)
    O String (scala.math.Ordering)
    🔙 🔒 StringAction in OperationFactory (com.sun.co
    G StringAdd (scala.Predef)
    G StringAdd (scala.runtime)
    G StringAdd (scala.runtime)
    🛺 🚡 StringArgument in Connector (com.sun.jdi.cor
Ы
https://ripturgrial.co.stringArgumentImpl in ConnectorImpl (COMO SUR
```

https://riptutorial.com/intellij-idea/topic/3755/little-known-features	

# **Chapter 9: Live Templates**

## **Examples**

Add a test method easily

```
@org.junit.Test
public void should_$name$() {
   $END$
}
```

Make sure to check the **Shorted FQ names** box when creating this template.

```
Abbreviation: should Descript

Template text:

@org.junit.Test
public void should_$name$() {
    $END$
}
```

## Applicable in Java: declaration. Change

When you type "should" (the abbreviation), this will add the necessary import org.junit.Test; statement at the top of the file, and this code:

```
@Test
public void should_() {
}
```

It is thanks to the Shorten FQ names option that @org.junit.Test is reduced to simply @Test.

The \$name\$ variable is irrelevant, it could be named something else. The purpose of that variable is that when the template is inserted in the class, the cursor will be placed in the position of \$name\$, asking you to enter something.

After you entered a value for \$name\$ (effectively the name of the test method), the cursor will finally jump to \$END\$, a built-in variable, so that you can carry on and implement the test case.

#### Insert the name of the current class

Consider the *utility class pattern*: a class with only static methods and no fields. It's recommended to prevent instantiation of such classes by adding a private a constructor.

This live template example makes it easy to add a private constructor to an existing class, using the name of the enclosing class.

```
private $className$() {
    throw new AssertionError("utility class, forbidden constructor");
}
```

Applicable in Java: declaration scope.

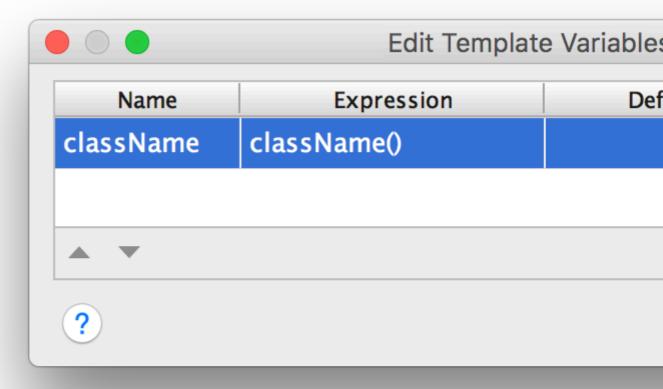
```
Abbreviation: utility_class

Template text:

private $className$() {
    throw new AssertionError("utility class, forbidden)}
```

## Applicable in Java: declaration. Change

Click **Edit variables** to define the className variable as the built-in className () expression, and check the **Skip if defined** box to avoid prompting for a custom name, which is unnecessary in this example.



For example, inside a class like this:

```
class ListUtils {
   // ...
}
```

When you type "utility\_class" (the abbreviation), this will insert a constructor like this:

```
class ListUtils {
    private ListUtils() {
        throw new AssertionError("utility class, forbidden constructor");
    }
    // ...
}
```

Read Live Templates online: https://riptutorial.com/intellij-idea/topic/2703/live-templates

# **Chapter 10: Optimization**

#### **Examples**

#### **Customizing the VM Options**

You can override the default <code>vmoptions</code> with your own personal settings by choosing **Help > Edit Custom VM Options** from the toolbar. This will create a local copy of the file which you are free to edit.

For example, if you double the value set for  $x_{mx}$ , the maximum size of the memory allocation pool will be doubled after you restart the IDE. On many machines this will lead to faster performance.

See the options here for a description of each parameter.

Read Optimization online: https://riptutorial.com/intellij-idea/topic/5305/optimization

# **Chapter 11: Useful Shortcuts**

#### **Examples**

**Compile and Run** 

# Make project (compile modifed and dependent)

Windows: Ctrl + F9

OS X / macOS: cmd + F9

# Compile selected file, package or module

This is useful to know, as when debugging this shortcut can be used to quickly reload / hotswap classes.

Windows: Ctrl + Shift + F9

OS X / macOS: cmd + shift + F9

# Select configuration and run

Windows: Alt + Shift + F10

OS X / macOS: Option + Shift + F10

# Select configuration and debug

Windows: Alt + Shift + F9

OS X / macOS: Option + Shift + F9

#### Run

Shift + F10

# **Debug**

Shift + F9

## Run context configuration from editor

Windows: ctrl + Shift + F10

OS X / macOS: cmd + shift + F10

**Code Completion** 

# Basic code completion (the name of any class, method or variable)

Windows: Ctrl + Space

OS X / macOS: cmd + Space

# Smart code completion (filters the list of methods and variables by expected type)

Windows: Ctrl + Shift + Space

OS X / macOS: Cmd + Shift + Space

# Overwriting code with a suggestion

Tab

## Adding code from a completion suggestion

Enter

Search/Replace

## Search everywhere

Double Shift

## **Find**

Windows / Linux: Ctrl + F

OS X / macOS: cmd + F

## **Find next**

F3

## **Find previous**

Shift + F3

# **Replace**

Windows / Linux: Ctrl + R

OS X / macOS: cmd + R

## Find in path

Windows / Linux: Ctrl + Shift + F

OS X / macOS: cmd + Shift + F

## Replace in path

Windows / Linux: Ctrl + Shift + R

OS X / macOS: cmd + shift + R

Refactoring

# Copy

F5

## Move

F6

# Safe delete

Windows / Linux: Alt + Delete

OS X / macOS: cmd + Delete

Note that the Delete key on OS X / macOS is the equivalent of the Backspace key on other operating systems.

#### Rename

Shift+ F6

## **Extract Method**

Windows / Linux: Ctrl + Alt + M

OS X / macOS: cmd + Option + M

## **Extract Field**

Windows / Linux: Ctrl + Alt + F

OS X / macOS: cmd + Option + F

### **Extract Variable**

Windows / Linux: Ctrl + Alt + v

OS X / macOS: cmd + option + v

## **Extract Constant**

Windows / Linux: Ctrl + Alt + C

OS X / macOS: cmd + Option + c

### **Extract Parameter**

Windows / Linux: Ctrl + Alt + P

OS X / macOS: cmd + Option + P

Other

## **Surround with**

Surrounds a code block with an if, for, <editor-fold ...> and more.

Windows / Linux: Ctrl + Alt + T

OS X / macOS: cmd + option + T

**Basic navigation** 

# Go to editor (from tool window)

# Switching focus to corresponding tool window

Windows: Alt + <tool window number>

OS X / macOS: cmd + <tool window number>

# For example switching focus to the project window

Windows: Alt + 1

OS X / macOS: cmd + 1

# **Recent files popup**

Windows: ctrl + E

OS X / macOS: cmd + E

## **Find Action**

Windows: Ctrl + Shift + A

OS X / macOS: cmd + Shift + A

# **Navigate to**

## File:

Windows: ctrl + Shift + N

OS X / macOS: cmd + Shift + N

## Class:

- Windows: ctrl + N
  - OS X / macOS: cmd + N

# Symbol (class/method/variable/constant name):

- Windows: ctrl + Alt + Shift + N
  - OS X / macOS: cmd + Option + Shift + N

Note that you can use class name to narrow down the method/variable/constant search, for example to find symbol usersCollection in class UserDAO type:

UserDAO.usersCollection

# **Everywhere:**

- Windows: shift + shift
  - OS X / macOS: Shift + Shift

To search for something that has multiple words, e.g., InetAddressCachePolicy you can just type InAddCacPo or something similar that contains parts of words in the whole name.

#### Go to line number

Windows: ctrl + G

OS X / macOS: cmd + L

### Go back to last edit location

Windows: Ctrl + Shift + Backspace

OS X / macOS: Cmd + Shift + Backspace

**Usage Search** 

# Find usages / Find usages in file

Windows / Linux: Alt + F7 / Ctrl + F7

OS X / macOS: Option + F7 / Ctrl + F7

# Highlight usages in file

Windows / Linux: Ctrl + Shift + F7

OS X / macOS: cmd + shift + F7

# **Show usages**

Windows / Linux: Ctrl + Alt + F7

OS X / macOS: cmd + option + F7

**Show Method Parameters** 

Windows / Linux: Ctrl + P

OS X / macOS: cmd + P

Shows what parameters a method and all of its overloads accepts.

```
JButton button = new JButton("Super Amazing Button!");

<no parameters>
|con icon |
| String text |
| Action a |
| String text, Icon icon |
```

#### Selection

#### Selection with increasing scope

This comes handy when you want to select a block to extract a variable / method etc, no need to do a precise bracket matching, just put the caret somewhere in the statement and keep doing this

Windows: ctrl + w

OS X / macOS: cmd + w

Selection with decreasing scope

Windows: Ctrl + Shift + W

OS X / macOS: cmd + shift + w

This feature is also very useful when editing / playing with json documents in your IDE.

#### Vertical selection

Press and hold

Windows: Alt

OS X / macOS: Opt

and select normally using mouse / trackpad (the way you select a word in a row etc)

This is how it should look like

```
"angular": "1.5.7",

"angular-animate": "1.5.7",

"angular-route": "1.5.7",

"angular-resource": "-0.2.11",

"angular-translate": "-2",

"angular-bootstrap": "-0.11",

"angular-timer": "1.3.3",

"angular-cookies": "-1.5.7",
```

#### **Multiple carets**

Press and hold

Windows: Alt + Shift

OS X / macOS: Opt + Shift

and click where all you want to put a caret. You can choose to put multiple carets in a single line or across lines at different positions.

Now you can perform all operations that you would have been able to perform on a single selected word (hold <code>ctrl</code> (windows) or <code>option</code> (mac OS) and use <code>Left</code> or <code>Right</code> keys to jump across words) and all those will affect all caret positions.

You can even cut / paste multiple selections from one place to another.

```
"angular": "1.5.7",

"angular-animate": "1.5.7",

"angular-route": "1.5.7",

"angular-ui-router": "~0.2.11",

"angular-resource": "1.5.7",

"angular-translate": "~2",

"angular-bootstrap": "~0.11",

"angular-timer": "1.3.3",
```

Having multiple carets is very usefult when you want to change the structure of text across many lines / many positions in same line.

#### Selecting duplicate occurences

Select some text and press

Windows: Alt + J

OS X / macOS: ctrl + G

to select the next occurance of the same text.

You get one caret at each of the selected occurrence that could be used to change each occurrence simultaneously.

E.g., I've tried to put an example in this gif, hope it helps

Read Useful Shortcuts online: https://riptutorial.com/intellij-idea/topic/3085/useful-shortcuts	

# **Credits**

S. No	Chapters	Contributors
1	Getting started with intellij-idea	Community, CraftedCart, foxtrot9, Harmelodic, Jeeter
2	Database Tools	kassak
3	Exporting	Jeeter
4	Hibernate HQL console and inspections	Shai Givati
5	How to Install Plugins	Priya, RamenChef, Sudeepa Nadeeshan
6	IdeaVim	Jeeter, Mateusz Piotrowski, mnoronha, sevenforce
7	Inspections	esin88
8	Little-known features	Anton Dozortsev, Jeeter, mszymborski, Prateek
9	Live Templates	janos
10	Optimization	Kevin Cooper
11	Useful Shortcuts	CraftedCart, karel, mszymborski, Nadim Bahadoor, Prateek