|  |  |
| --- | --- |
| **Eclipse** | **IntelliJ IDEA** |
| A number of projects, a workspace | Project |
| Project | Module |
| Project-specific JRE | Module SDK |
| User library | Global library |
| Classpath variable | Path variable |
| Project dependency | Module dependency |
| Library | Module library |

You can create a new [project](http://www.jetbrains.com/idea/webhelp/project.html) or [module](http://www.jetbrains.com/idea/webhelp/module.html) by importing an Eclipse project or workspace into IntelliJ IDEA. In IntelliJ IDEA each imported Eclipse project is represented by a module.

Alternatively, you can create a project by opening an Eclipse project or workspace in IntelliJ IDEA. In comparison with importing, this way of creating a project is quicker, however, you are offered less options for configuring the resulting project.

IntelliJ IDEA can keep the [module files](http://www.jetbrains.com/idea/webhelp/module.html#module_file) and the corresponding Eclipse project files synchronized. This is useful when a development team uses both IntelliJ IDEA and Eclipse.

* [Importing an Eclipse project or workspace into IntelliJ IDEA](http://www.jetbrains.com/idea/webhelp/importing-eclipse-projects-into-intellij-idea.html#d1158675e427)
* [Opening an Eclipse project](http://www.jetbrains.com/idea/webhelp/importing-eclipse-projects-into-intellij-idea.html#d1158675e555)

See also, [IntelliJ IDEA vs Eclipse Terminology](http://www.jetbrains.com/idea/webhelp/intellij-idea-vs-eclipse-terminology.html).

**To import an Eclipse project into IntelliJ IDEA**

1. Do one of the following:
   * If you are going to create a new project: click **Import Project** on the [Welcome screen](http://www.jetbrains.com/idea/webhelp/welcome-screen.html) or select **File | Import Project**.
   * If you are going to import an Eclipse project into an existing IntelliJ IDEA project: [open the project](http://www.jetbrains.com/idea/webhelp/opening-reopening-and-closing-projects.html) if interest, and select **File | Import Module**.
2. In the [dialog that opens](http://www.jetbrains.com/idea/webhelp/select-path-dialog.html), select the necessary Eclipse project or workspace directory, or the corresponding .project or .classpath file. Click **OK**.

As a result, the **Import Project** or the **Import Module** wizard opens. (See [Import Project or Module Wizard](http://www.jetbrains.com/idea/webhelp/import-project-or-module-wizard.html).)

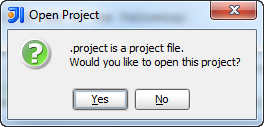
1. On the [first page of the wizard](http://www.jetbrains.com/idea/webhelp/origin-of-the-sources.html), select **Import project from external model** or **Import module from external model**, then select**Eclipse** and click **Next**. (This page is not shown if IntelliJ IDEA has guessed that you are importing an Eclipse project or workspace.)
2. On the [next page of the wizard](http://www.jetbrains.com/idea/webhelp/import-from-eclipse-page-1.html), specify:
   * The directory that contains the Eclipse project or workspace to be imported.
   * Where the IntelliJ IDEA [project](http://www.jetbrains.com/idea/webhelp/project.html#ProjectFiles) and [module files](http://www.jetbrains.com/idea/webhelp/module.html#module_file) should be created.
   * The IntelliJ IDEA [project format](http://www.jetbrains.com/idea/webhelp/project.html#ProjectFiles).
   * Whether the IntelliJ IDEA [module file](http://www.jetbrains.com/idea/webhelp/module.html#module_file) or files should be kept synchronized with the corresponding Eclipse project file or files.
   * Whether and how test sources are to be looked for.

Click **Next**.

1. If you are importing more than one Eclipse project (e.g. an Eclipse workspace), select the Eclipse projects you want to import on the [next page of the wizard](http://www.jetbrains.com/idea/webhelp/import-from-eclipse-page-2.html), and click **Next**.
2. If suggested, [specify the project SDK](http://www.jetbrains.com/idea/webhelp/import-existing-sources-project-sdk.html) and click **Next**.
3. The [next page of the wizard](http://www.jetbrains.com/idea/webhelp/import-existing-sources-frameworks.html) shows the frameworks and technologies detected in your Eclipse project or projects. Select the ones that you want to be enabled and click **Finish**.

**To open an Eclipse project**

1. If no project is currently open in IntelliJ IDEA, click **Open Project** on the [Welcome screen](http://www.jetbrains.com/idea/webhelp/welcome-screen.html). Otherwise, select **Open** from the **File** menu.
2. In the [dialog that opens](http://www.jetbrains.com/idea/webhelp/select-path-dialog.html), select the necessary Eclipse workspace or project folder, or the corresponding .project or .classpath file. Click**OK**.
3. If the following dialog is displayed:



Click **Yes** to create and open the project. Click **No** to open the file in the editor.

---------------------------------------------------------------------------

Project

Whatever you do in IntelliJ IDEA, you do that in the context of a project. A project is an organizational unit that represents a complete software solution.

Your finished product may be decomposed into a series of discrete, isolated modules, but it's a project definition that brings them together and ties them into a greater whole.

Projects don't themselves contain development artifacts such as source code, build scripts, or documentation. They are the highest level of organization in the IDE, and they define [project-wide settings](http://www.jetbrains.com/idea/webhelp/project-settings.html) as well as collections of what IntelliJ IDEA refers to as [modules](http://www.jetbrains.com/idea/webhelp/module.html) and [libraries](http://www.jetbrains.com/idea/webhelp/library.html).

On this page:

* [Project format and files](http://www.jetbrains.com/idea/webhelp/project.html#d1244689e186)
* [Directory-based format](http://www.jetbrains.com/idea/webhelp/project.html#d1244689e201)
* [File-based format](http://www.jetbrains.com/idea/webhelp/project.html#d1244689e238)

**Project format and files**

IntelliJ IDEA stores the configuration data for projects and their components in plain text XML files making it easy to manage and share project configuration data with others.

Two formats are available for storing the project configuration data: [directory-based](http://www.jetbrains.com/idea/webhelp/project.html#newProjectFormat) and [file-based](http://www.jetbrains.com/idea/webhelp/project.html#file_based_format).

**Directory-based format**

When the directory-based format is used, there is a .idea directory in the project directory.

The .idea directory contains a set of configuration files (.xml). Each file contains only a portion of configuration data pertaining to a certain functional area which is reflected in the name of a file, for example, compiler.xml, encodings.xml, modules.xml.

Almost all of the files contain information core to the project itself, such as names and locations of its component modules, compiler settings, etc. Thus, these files may (and should) be kept under version control.

The exception is the file workspace.xml. It stores your personal settings such as placement and positions of your windows, your VCS and History settings, and other data pertaining to the development environment. So it's unlikely that you'd want to share this file with your colleagues.

**File-based format**

If the file-based format is used, there are two configuration files in the project directory. One of the files has the .ipr extension and stores the core project information. The other file has the .iws extension and stores your personal workspace settings.

The .iws file shouldn't be placed under version control while the .ipr file should.

The file-based format can be converted into the directory-based, see [Converting Projects Into Directory-Based Format](http://www.jetbrains.com/idea/webhelp/converting-projects-into-directory-based-format.html).

---------------------------------------------------------------------------

Module

A module is a discrete unit of functionality which you can compile, run, test and debug independently.

Modules contain everything that is required for their specific tasks: source code, build scripts, unit tests, deployment descriptors, and documentation. However, modules exist and are functional only in the context of a [project](http://www.jetbrains.com/idea/webhelp/project.html).

Configuration information for a module is stored in a .iml module file. By default, such a file is located in the module's [content root folder](http://www.jetbrains.com/idea/webhelp/content-root.html).

Development teams, normally, share the .iml module files through version control.

---------------------------------------------------------------------------

Depending on the IntelliJ IDEA project format (.ipr file based or .idea directory based), you should put the following IntelliJ IDEA project files under the version control:

.ipr file based format

Share the project .ipr file and and all the .iml module files, don't share the .iws file as it stores user specific settings.

.idea directory based format

Share all the files under .idea directory in the project root except the workspace.xml and tasks.xml files which store user specific settings, also share all the .iml module files.

I put it in my .gitignore:

#Project

workspace.xml

tasks.xml

Module dependencies are the entities that a [module](http://www.jetbrains.com/idea/webhelp/module.html) uses and, consequently, depends on. Thus, the term **module dependency** may refer to:

* The [SDK](http://www.jetbrains.com/idea/webhelp/sdk.html) associated with the module
* [Libraries](http://www.jetbrains.com/idea/webhelp/library.html) used in the module
* Other modules belonging to the same project

The term **module dependency** may also be used in a narrower sense meaning only the modules that a module depends on.