# Development in IntelliJ IDEA

The preconfigured IntelliJ IDEA project with modules for two sample plugins and a batch mode (command-line) tool is provided with a program installation directory. The project can be found in *<modeling tool installation directory>\openapi\ide\intellij.zip.*

Let's use MagicDraw as an example to describe the following procedures.

To setup the IntelliJ IDEA environment for the MagicDraw (or other program according to a modeling tool you are using) development

---

1. In IntelliJ IDEA, add a path variable named MAGIC_DRAW_INSTALL_DIRECTORY and pointing to your version of MagicDraw (CSM, CEA that you have the licence for, as it starts like a normal version and will require an active licence) installation directory.
2. Extract *<MagicDraw installation directory>\openapi\ide\intellij.zip* to a chosen folder.
3. Open the extracted IntelliJ IDEA project *MagicDraw development.ipr* from that folder.

> ⚠ The launch configuration is designed to load plugins from the program installation directory (see step #1) and two plugins from the IDEA project. Thus, if the *md.plugins.dir* java system property is not defined (see Plugins directories), developing plugins are not loaded.

After projects are imported and the launch configurations are prepared, the IDEA project are ready for the source code development and running/debugging.

> ⚠ The libraries (jar files) of the plugin must be added to the development class path throughout the plugin dependency hierarchy if the developing code depends on that plugin.
> For example, if the code depends on plugin A; plugin A depends on plugins B and C; plugin B depends on plugin D, the libraries of all plugins (A, B, C, and D) must be added to the class path.

> ⚠ When you launch your own plugin, you need to add all jar files that are required by your plugin from appropriate plugins. The MagicDraw jar files can be found in MAGIC_DRAW_INSTALL_DIRECTORY/lib and its sub directories, whereas plugins' jar files can be found in MAGIC_DRAW_INSTALL_DIRECTORY/plugins and its sub directories.

> ⓘ Even if the plugin descriptor file contains information about a runtime plugin *.jar* file, it is not necessary to build and deploy this *.jar* file to a plugin directory while a plugin is developed under IntelliJ IDEA.

## Launching tests

A custom test launching mechanism is used to launch MagicDraw tests, because of an Eclipse OSGi environment (which is not officially supported by the recent version of IntelliJ IDEA).

The test launcher classes are defined at the end of *VM options* (without prefix *-D*) and the classpath to those classes is additionally defined as CLASSPATH variable in *Environment variables* section of the run configuration. These settings should be used for any test configuration for MagicDraw on IntelliJ IDEA by default (see the picture below and the provided sample configuration).

An IntelliJ IDEA module that is used for classpath in the test configuration should include all contents of the *<modeling tool installation directory>\lib* directory recursively as it's dependencies.

The test launching mechanism is designed to use IntelliJ IDEA classpath file (autmatically generated) to support long classpaths. It is mandatory to setup Intellij IDEA to create that classpath file for provided test launchers. That can be done by adding or modifying a property *dynamic.classpath* to be *true* in IntelliJ IDEA workspace properties file *<project_name>.iws* (it usually resides beside project file *<project_name>.ipr)* or *.idea/workspace.xml* (if the directory based project structure is used).

**<project_name>.iws or .idea/workspace.xml**

```
...
<component name="PropertiesComponent">
    <property name="recentsLimit" value="5" />
    ...
    <property name="dynamic.classpath" value="true" />
```