# Creating a Plugin Project

The IntelliJ Platform SDK - the primary source of documentation for extending the IntelliJ Platform by creating plugins, custom language support, or building a custom IDE.

But the IntelliJ Platform's real power comes from the Program Structure Interface (PSI). It is a set of functionalities used to parse files, build rich syntactic and semantic models of the code, and build indexes from this data. PSI powers a lot of functionality, from quick navigating to files, types, and symbols, to the contents of code completion windows and find usages, code inspections, and code rewriting, for quick fixes or refactorings, as well as many other features.
The IntelliJ Platform includes parsers and a PSI model for many languages, and its extensible nature means that it is possible to add support for other languages.

## Creation of agile development environment.

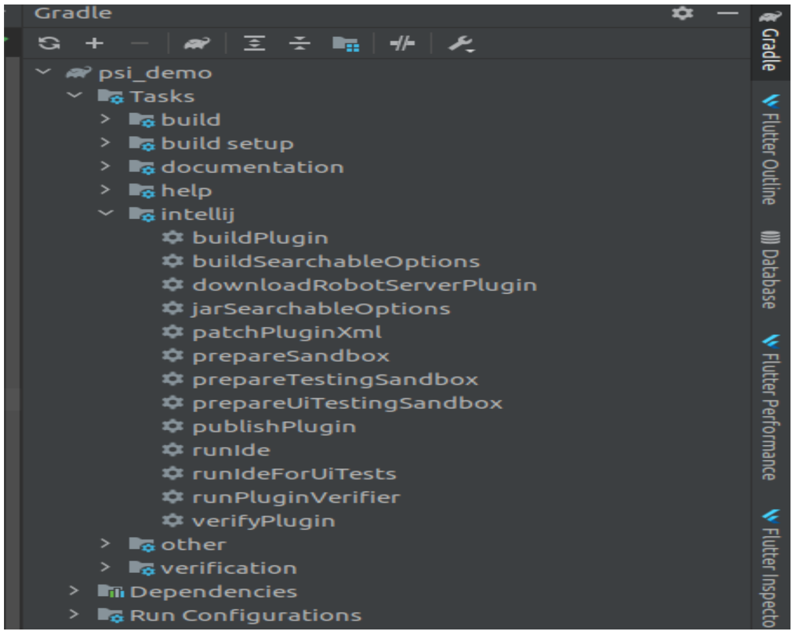
- CRUD .
- Standardized code.
- Reusable code.

## Tasks

Plugin introduces the following tasks

| Task | Description |
|---|---|
| buildPlugin | Assembles plugin and prepares zip archive for deployment. |
| patchPluginXml | Collects all plugin.xml files in sources and fill since/until build and version attributes. |
| downloadRobotServerPlugin | Downloads robot-server plugin which is needed for ui tests running. |
| prepareSandbox | Creates proper structure of plugin, copies patched plugin xml files and fills sandbox directory with all of it. |
| prepareTestingSandbox | Prepares sandbox that will be used while running tests. |
| prepareUiTestingSandbox | Prepares sandbox that will be used while running ui tests. |

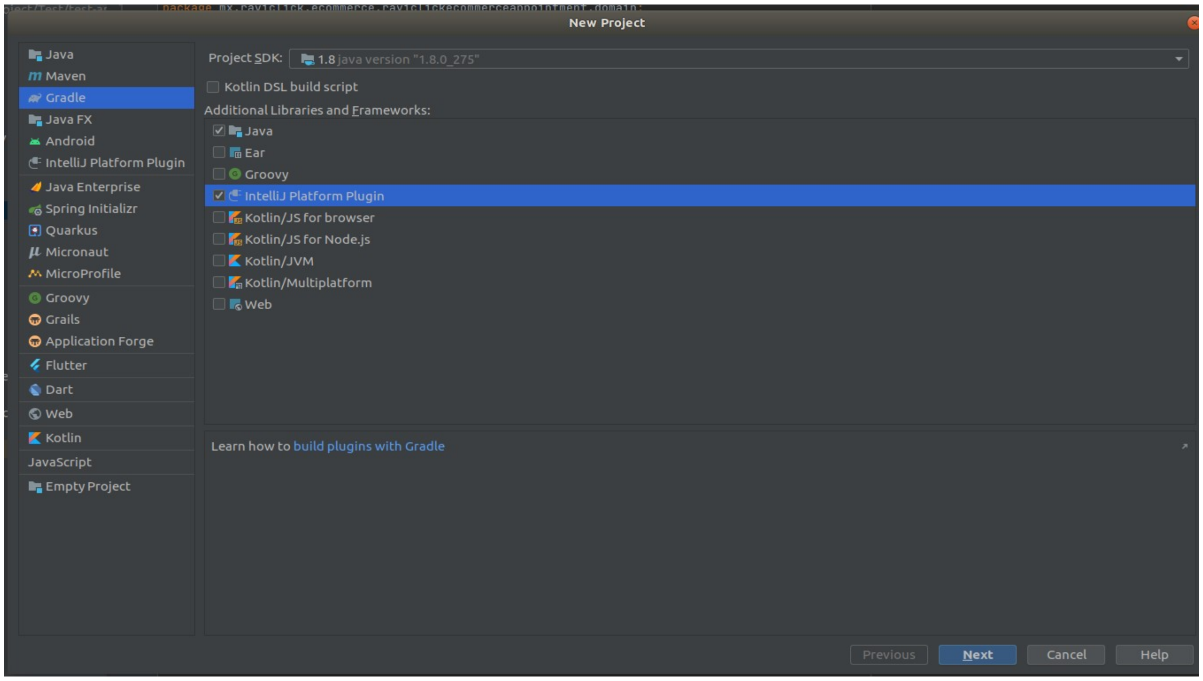
| Task | Description |
|---|---|
| buildSearchableOptions | Builds an index of UI components (a.k.a. searchable options) for the plugin by running a headless IDE instance.<br>Note, that this is a runIde task with predefined arguments and all properties of runIde task are also applied to buildSearchableOptions tasks. |
| jarSearchableOptions | Creates a jar file with searchable options to be distributed with the plugin. |
| runIde | Executes an IntelliJ IDEA instance with the plugin you are developing. |
| runIdeForUiTests | Executes an IntelliJ IDEA instance ready for ui tests run with the plugin you are developing. See intellij-ui-test-robot project to know more |
| publishPlugin | Uploads plugin distribution archive to https://plugins.jetbrains.com. |
| runPluginVerifier | Runs the IntelliJ Plugin Verifier tool to check the binary compatibility with specified IntelliJ IDE builds. |
| verifyPlugin | Validates completeness and contents of plugin.xml descriptors as well as plugin's archive structure. |


```
Gradle
psi_demo
  Tasks
    build
    build setup
    documentation
    help
    intellij
      buildPlugin
      buildSearchableOptions
      downloadRobotServerPlugin
      jarSearchableOptions
      patchPluginXml
      prepareSandbox
      prepareTestingSandbox
      prepareUiTestingSandbox
      publishPlugin
      runIde
      runIdeForUiTests
      runPluginVerifier
      verifyPlugin
    other
    verification
  Dependencies
  Run Configurations
```

## To Create an IntelliJ Platform Plugin Project:

1. On the main menu, choose File | New | Project. ...
2. Set **IntelliJ** Platform **Plugin** project type.
3. Click Next.

4. Set the desired project name.
5. Click Finish to generate project structure files.
6. Go to File | Project Structure to customize project settings if required.



Project SDK: 1.8 java version "1.8.0_275"

[ ] Kotlin DSL build script

Additional Libraries and Frameworks:

[✓] Java
[ ] Ear
[ ] Groovy
[✓] IntelliJ Platform Plugin
[ ] Kotlin/JS for browser
[ ] Kotlin/JS for Node.js
[ ] Kotlin/JVM
[ ] Kotlin/Multiplatform
[ ] Web

Learn how to build plugins with Gradle

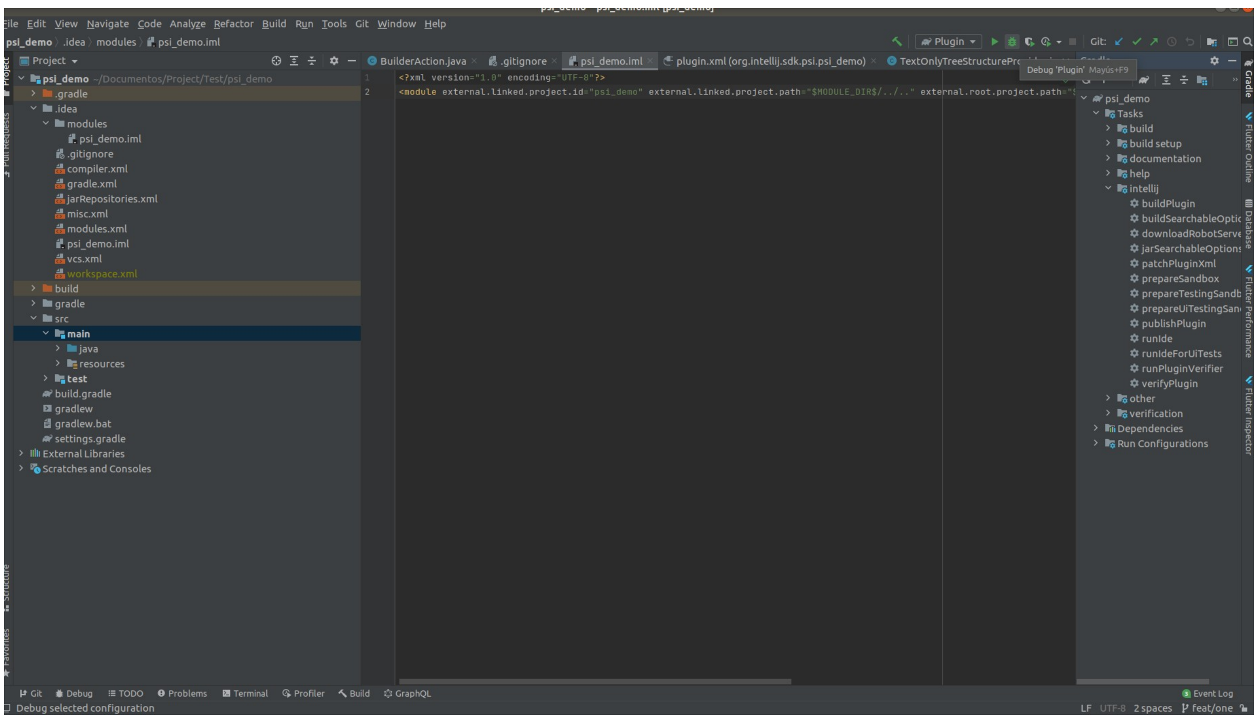
## Running and Debugging a Plugin

It's possible to run and debug a plugin directly from the IntelliJ IDEA. You need a configured special profile (a *Plugin* Run/Debug configuration) that specifies the plugin module, VM parameters, and other specific options. When you run such a profile, it launches the IDE with your plugin installed.

Using IntelliJ IDEA's debugger, you can find out the origin of the run-time errors and exceptions.

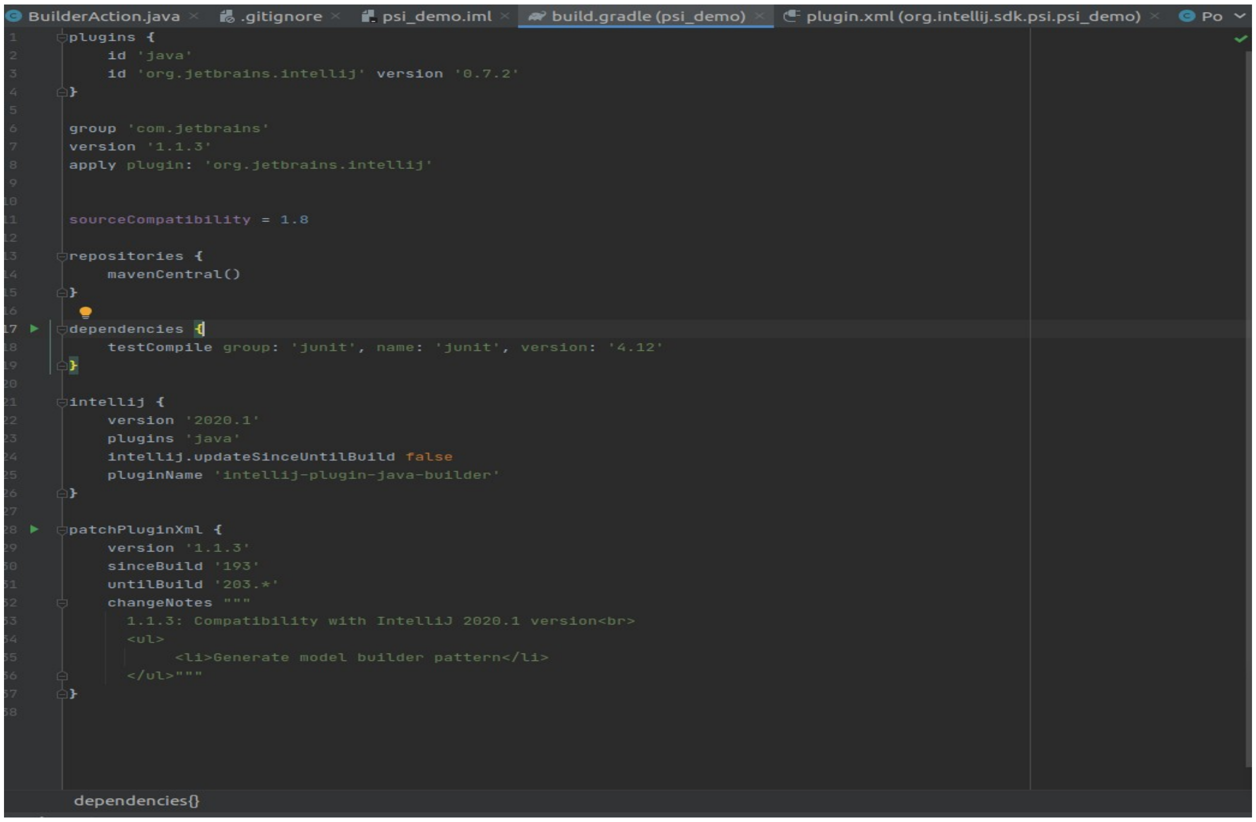
**To debug a plugin**

•Select **Run | Debug** in the main menu, or press Shift + F9.
**To run a plugin**
•Select **Run | Run** in the main menu, or press Shift + F10.



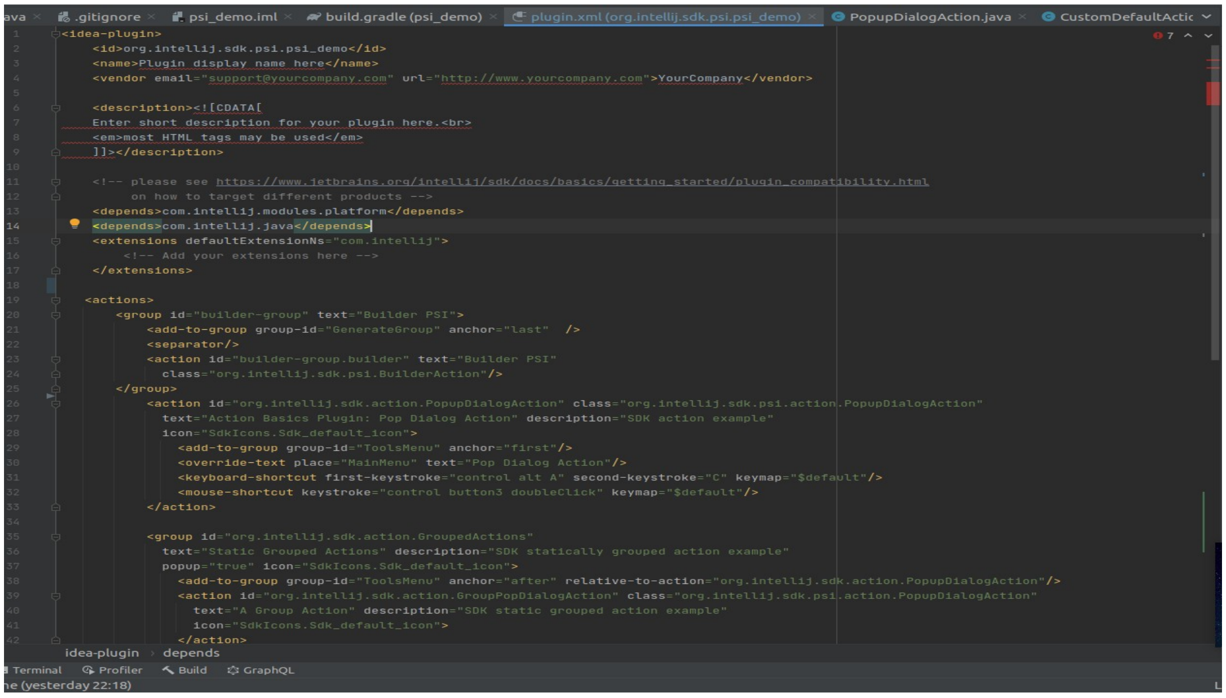
```
<?xml version="1.0" encoding="UTF-8"?>
<module external.linked.project.id="psi_demo" external.linked.project.path="$MODULE_DIR$/../.." external.root.project.path="$
```

## build.gradle



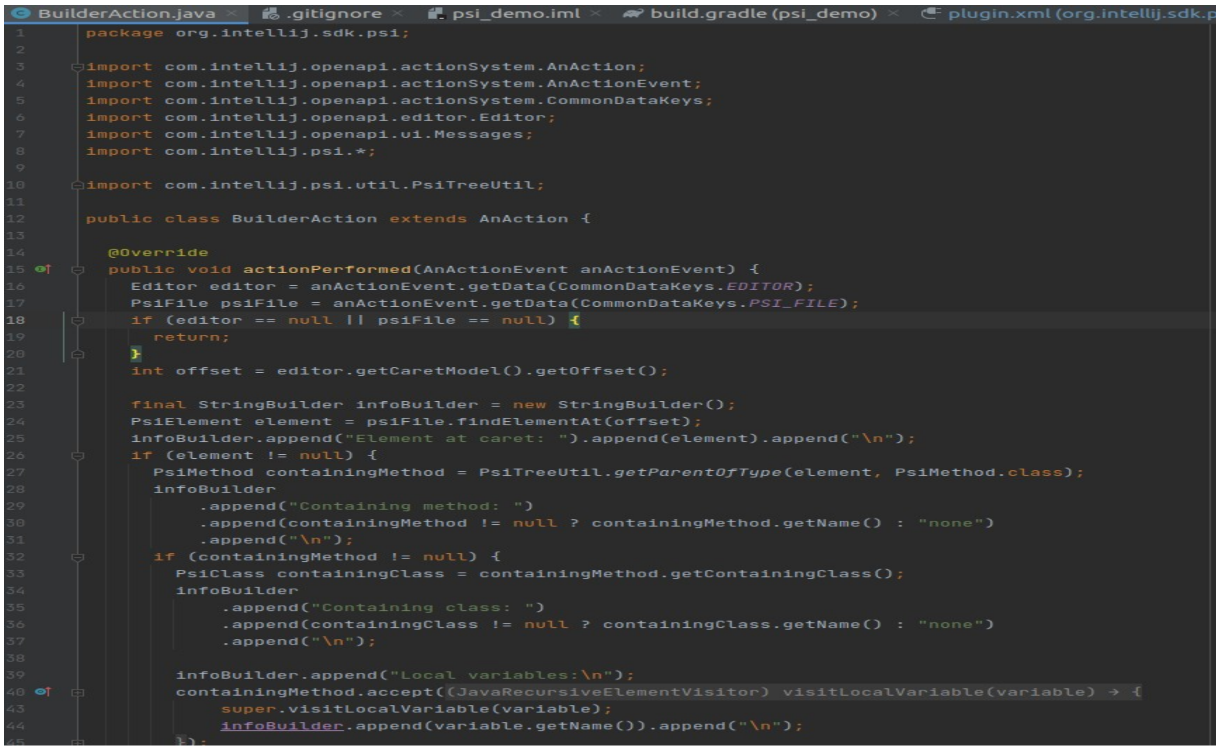
```
plugins {
    id 'java'
    id 'org.jetbrains.intellij' version '0.7.2'
}

group 'com.jetbrains'
version '1.1.3'
apply plugin: 'org.jetbrains.intellij'


sourceCompatibility = 1.8

repositories {
    mavenCentral()
}

dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.12'
}

intellij {
    version '2020.1'
    plugins 'java'
    intellij.updateSinceUntilBuild false
    pluginName 'intellij-plugin-java-builder'
}

patchPluginXml {
    version '1.1.3'
    sinceBuild '193'
    untilBuild '203.*'
    changeNotes """
      1.1.3: Compatibility with IntelliJ 2020.1 version<br>
      <ul>
           <li>Generate model builder pattern</li>
      </ul>"""
}
```

## plugin.xml



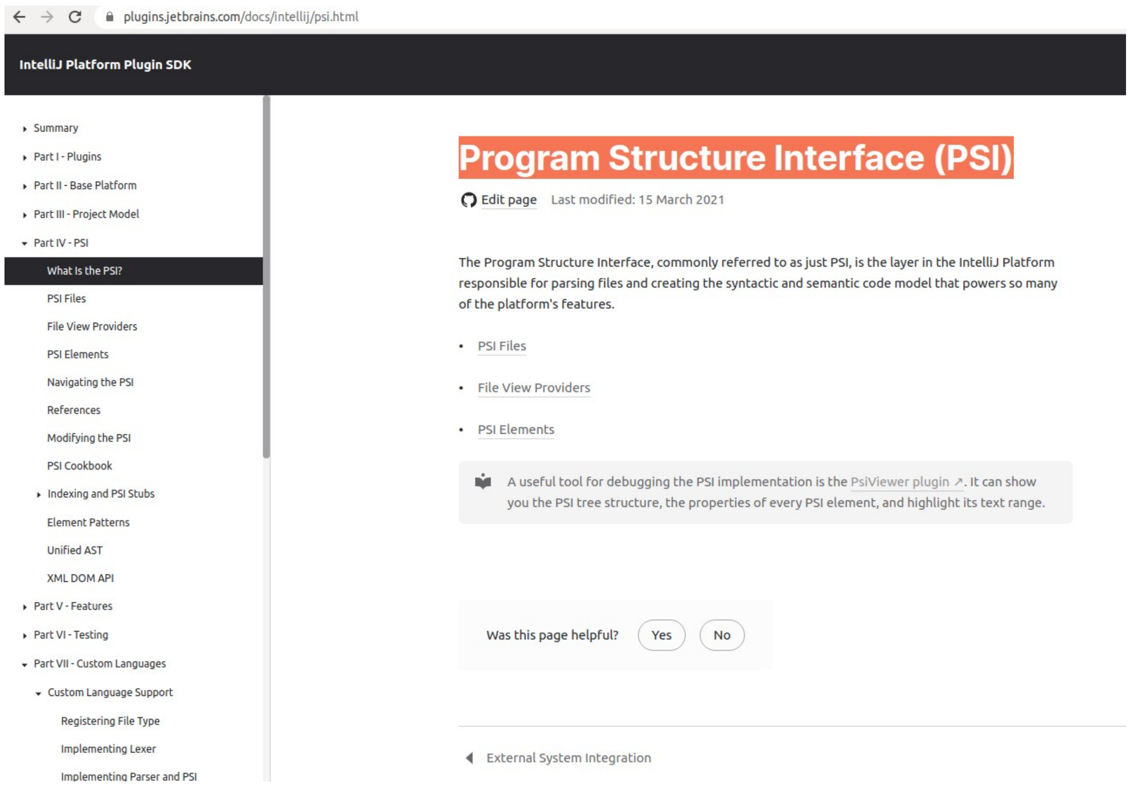
```xml
<idea-plugin>
    <id>org.intellij.sdk.psi.psi_demo</id>
    <name>Plugin display name here</name>
    <vendor email="support@yourcompany.com" url="http://www.yourcompany.com">YourCompany</vendor>

    <description><![CDATA[
    Enter short description for your plugin here.<br>
    <em>most HTML tags may be used</em>
    ]]></description>

    <!-- please see https://www.jetbrains.org/intellij/sdk/docs/basics/getting_started/plugin_compatibility.html
         on how to target different products -->
    <depends>com.intellij.modules.platform</depends>
    <depends>com.intellij.java</depends>
    <extensions defaultExtensionNs="com.intellij">
        <!-- Add your extensions here -->
    </extensions>

   <actions>
       <group id="builder-group" text="Builder PSI">
           <add-to-group group-id="GenerateGroup" anchor="last"  />
           <separator/>
           <action id="builder-group.builder" text="Builder PSI"
             class="org.intellij.sdk.psi.BuilderAction"/>
       </group>
           <action id="org.intellij.sdk.action.PopupDialogAction" class="org.intellij.sdk.psi.action.PopupDialogAction"
             text="Action Basics Plugin: Pop Dialog Action" description="SDK action example"
             icon="SdkIcons.Sdk_default_icon">
               <add-to-group group-id="ToolsMenu" anchor="first"/>
               <override-text place="MainMenu" text="Pop Dialog Action"/>
               <keyboard-shortcut first-keystroke="control alt A" second-keystroke="C" keymap="$default"/>
               <mouse-shortcut keystroke="control button3 doubleClick" keymap="$default"/>
           </action>

           <group id="org.intellij.sdk.action.GroupedActions"
             text="Static Grouped Actions" description="SDK statically grouped action example"
             popup="true" icon="SdkIcons.Sdk_default_icon">
               <add-to-group group-id="ToolsMenu" anchor="after" relative-to-action="org.intellij.sdk.action.PopupDialogAction"/>
               <action id="org.intellij.sdk.action.GroupPopDialogAction" class="org.intellij.sdk.psi.action.PopupDialogAction"
                 text="A Group Action" description="SDK static grouped action example"
                 icon="SdkIcons.Sdk_default_icon">
               </action>
```

## Action



```java
package org.intellij.sdk.psi;

import com.intellij.openapi.actionSystem.AnAction;
import com.intellij.openapi.actionSystem.AnActionEvent;
import com.intellij.openapi.actionSystem.CommonDataKeys;
import com.intellij.openapi.editor.Editor;
import com.intellij.openapi.ui.Messages;
import com.intellij.psi.*;

import com.intellij.psi.util.PsiTreeUtil;

public class BuilderAction extends AnAction {

  @Override
  public void actionPerformed(AnActionEvent anActionEvent) {
    Editor editor = anActionEvent.getData(CommonDataKeys.EDITOR);
    PsiFile psiFile = anActionEvent.getData(CommonDataKeys.PSI_FILE);
    if (editor == null || psiFile == null) {
      return;
    }
    int offset = editor.getCaretModel().getOffset();

    final StringBuilder infoBuilder = new StringBuilder();
    PsiElement element = psiFile.findElementAt(offset);
    infoBuilder.append("Element at caret: ").append(element).append("\n");
    if (element != null) {
      PsiMethod containingMethod = PsiTreeUtil.getParentOfType(element, PsiMethod.class);
      infoBuilder
          .append("Containing method: ")
          .append(containingMethod != null ? containingMethod.getName() : "none")
          .append("\n");
      if (containingMethod != null) {
        PsiClass containingClass = containingMethod.getContainingClass();
        infoBuilder
            .append("Containing class: ")
            .append(containingClass != null ? containingClass.getName() : "none")
            .append("\n");

        infoBuilder.append("Local variables:\n");
        containingMethod.accept((JavaRecursiveElementVisitor) visitLocalVariable(variable) → {
            super.visitLocalVariable(variable);
            infoBuilder.append(variable.getName()).append("\n");
        });
```

## Program Structure Interface (PSI)



plugins.jetbrains.com/docs/intellij/psi.html

IntelliJ Platform Plugin SDK

- Summary
- Part I - Plugins
- Part II - Base Platform
- Part III - Project Model
- Part IV - PSI
  - What Is the PSI?
  - PSI Files
  - File View Providers
  - PSI Elements
  - Navigating the PSI
  - References
  - Modifying the PSI
  - PSI Cookbook
  - Indexing and PSI Stubs
  - Element Patterns
  - Unified AST
  - XML DOM API
- Part V - Features
- Part VI - Testing
- Part VII - Custom Languages
  - Custom Language Support
    - Registering File Type
    - Implementing Lexer
    - Implementing Parser and PSI

**Program Structure Interface (PSI)**

Edit page Last modified: 15 March 2021

The Program Structure Interface, commonly referred to as just PSI, is the layer in the IntelliJ Platform responsible for parsing files and creating the syntactic and semantic code model that powers so many of the platform's features.

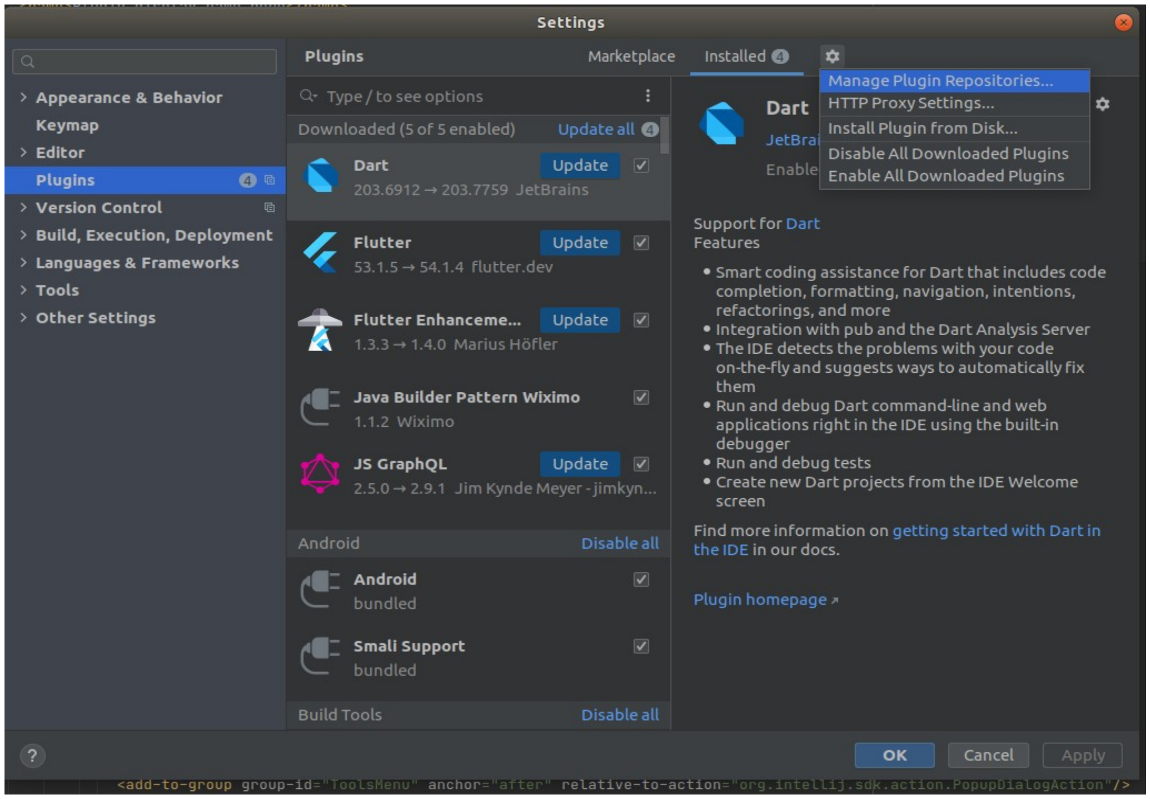
- PSI Files
- File View Providers
- PSI Elements

A useful tool for debugging the PSI implementation is the PsiViewer plugin ↗. It can show you the PSI tree structure, the properties of every PSI element, and highlight its text range.

Was this page helpful? Yes No

External System Integration

## Install Plugin from Disk...



Settings

- Appearance & Behavior
- Keymap
- Editor
- Plugins 4
- Version Control
- Build, Execution, Deployment
- Languages & Frameworks
- Tools
- Other Settings

Plugins Marketplace Installed 4

Type / to see options

Downloaded (5 of 5 enabled) Update all 4

| | | |
|---|---|---|
| Dart | 203.6912 → 203.7759 JetBrains | Update ✓ |
| Flutter | 53.1.5 → 54.1.4 flutter.dev | Update ✓ |
| Flutter Enhanceme... | 1.3.3 → 1.4.0 Marius Höfler | Update ✓ |
| Java Builder Pattern Wiximo | 1.1.2 Wiximo | ✓ |
| JS GraphQL | 2.5.0 → 2.9.1 Jim Kynde Meyer - jimkyn... | Update ✓ |

Android Disable all

| | | |
|---|---|---|
| Android | bundled | ✓ |
| Smali Support | bundled | ✓ |

Build Tools Disable all

Manage Plugin Repositories...
HTTP Proxy Settings...
Install Plugin from Disk...
Disable All Downloaded Plugins
Enable All Downloaded Plugins

Dart

JetBrai

Enable

Support for Dart
Features

- Smart coding assistance for Dart that includes code completion, formatting, navigation, intentions, refactorings, and more
- Integration with pub and the Dart Analysis Server
- The IDE detects the problems with your code on-the-fly and suggests ways to automatically fix them
- Run and debug Dart command-line and web applications right in the IDE using the built-in debugger
- Run and debug tests
- Create new Dart projects from the IDE Welcome screen

Find more information on getting started with Dart in the IDE in our docs.

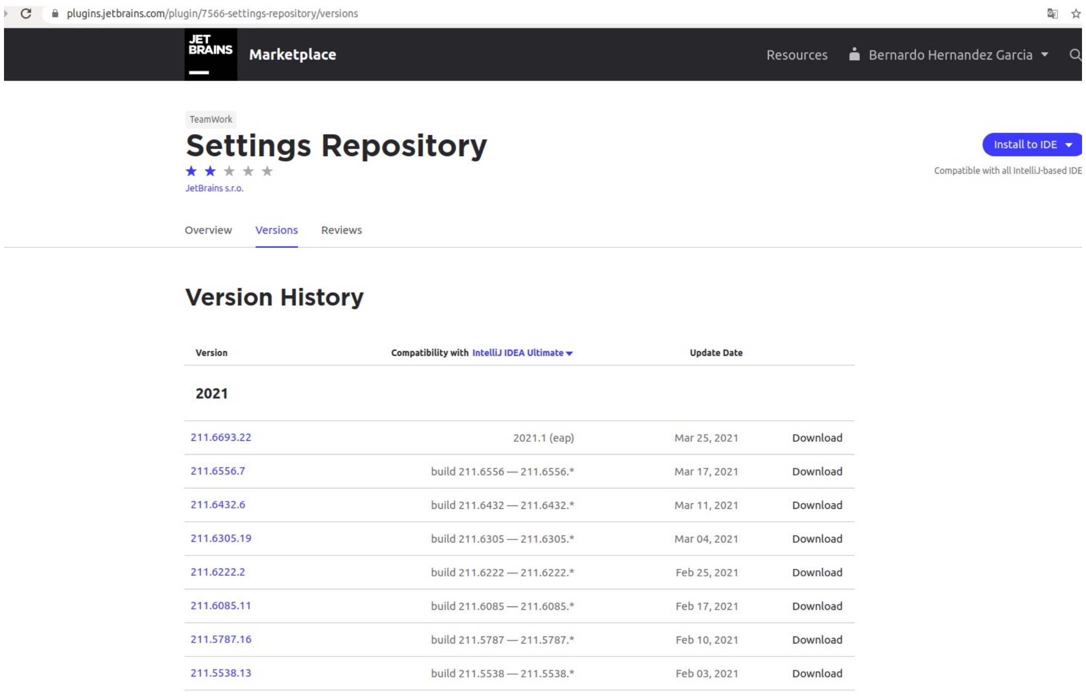
Plugin homepage ↗

OK Cancel Apply

```
<add-to-group group-id="ToolsMenu" anchor="after" relative-to-action="org.intellij.sdk.action.PopupDialogAction"/>
```

## Repositroy Jetbrains

https://plugins.jetbrains.com/plugin/7566-settings-repository/versions


plugins.jetbrains.com/plugin/7566-settings-repository/versions
JET BRAINS
Marketplace
Resources
Bernardo Hernandez Garcia
TeamWork
Install to IDE
Compatible with all IntelliJ-based IDE
JetBrains s.r.o.
Overview
Versions
Reviews

Settings Repository

Version History

| Version | Compatibility with IntelliJ IDEA Ultimate | Update Date | |
|---|---|---|---|
| 2021 | | | |
| 211.6693.22 | 2021.1 (eap) | Mar 25, 2021 | Download |
| 211.6556.7 | build 211.6556 — 211.6556.* | Mar 17, 2021 | Download |
| 211.6432.6 | build 211.6432 — 211.6432.* | Mar 11, 2021 | Download |
| 211.6305.19 | build 211.6305 — 211.6305.* | Mar 04, 2021 | Download |
| 211.6222.2 | build 211.6222 — 211.6222.* | Feb 25, 2021 | Download |
| 211.6085.11 | build 211.6085 — 211.6085.* | Feb 17, 2021 | Download |
| 211.5787.16 | build 211.5787 — 211.5787.* | Feb 10, 2021 | Download |
| 211.5538.13 | build 211.5538 — 211.5538.* | Feb 03, 2021 | Download |