Inspection of Code Source

> This document is a work in progress.

This topic describes the comparingReferences sample plugin that creates a custom inspection of Java code. In addition, the sample plugin contains a `JUnit`-based test. Basing on this information, you can develop your own plugins using similar techniques.

About Code Inspections

IntelliJ IDEA provides tools designed for static code analysis (so called code inspections) that help you maintain and clean up your code without actually executing it. In IntelliJ IDEA you will find a set of built-in inspections that are grouped by their goals and sense. For more information about code inspections, see Inspecting Source Code in IntelliJ IDEA Web Help.
You can create custom inspections through the IntelliJ IDEA interface (see Creating Own Inspections). Alternatively, you can develop a plugin to implement a custom inspection.

Used Techniques

The comparingReferences sample plugin illustrates the use of the following techniques:

- How to analyze a PSI tree .
- How to find a Java token of interest in the PSI tree.
- How to inspect Java code in the IntelliJ IDEA editor using the BaseJavaLocalInspectionTool class.
- How to create a `JUnit` test for this plugin using the IdeaTestFixtureFactory class.

Sample Plugin

The comparingReferences sample plugin is available in the <%IDEA project folder%>/community/samples/comparingReferences> directory.
When launched, this plugin adds the '==' or '!=' instead of 'equals()' item to the Probable bugs node in the IDEA Inspections list .

Running the Plugin

To run the sample plugin

1. Start IntelliJ IDEA and open the comparingReferences plugin project saved into the <%IDEA project folder%>/community/samples/comparingReferences directory.
2. Ensure that the project settings are valid for your environment. If necessary, modify the project settings.

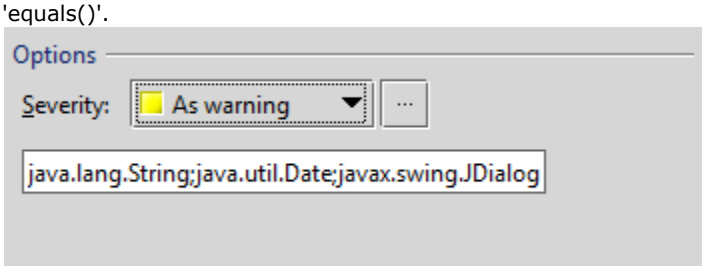
   To view or modify the project settings, on the toolbar, click  , and then complete the Project Structure dialog box that opens.
3. Run the plugin by choosing the Run | Run on the main menu.
   If necessary, change the Run/Debug Configurations .

Configuring the Plugin

Once the plugin is launched, you can set the plugin options. You can specify the Java classes to be participated in the code inspection and the severity level of the found probable bugs.

To configure the sample plugin

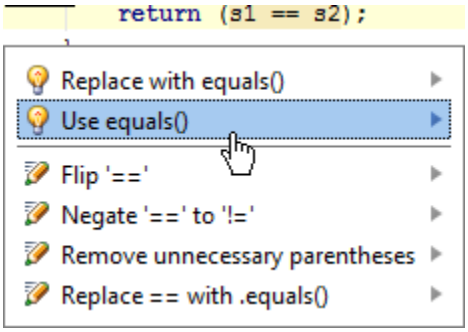
1. On the IDEA main menu, choose File | Settings, and then under Project Settings, click Inspections.
2. In the list of the IntelliJ IDEA inspections, expand the Probable bugs node, and then click '==' or '!=' instead of 'equals()'.



3. Under Options, you can specify the following plugin settings:
    - From the Severity list, select the severity level of probable bugs the plugin will find (such as Warning, Info, etc.)
    - In the text box under Severity, specify the semicolon separated list of Java classes to be participated in this code inspection.
4. When finished, click OK.

How It Works?

The plugin inspects your code opened in the IntelliJ IDEA editor or the code you are typing. The plugin highlights the code fragments where two variables of the reference type are separated by == or != and proposes to replace this code fragment with .equals():



In this example, the s1 and s2 are variables of the `String` type. Clicking Use equals() replaces

```
return (s1==s2);
```

with the code:

```
return (s1.equals(s2));
```

Testing the Plugin

The sample plugin contains the `TestThisPlugin` Java class in the testSource/testPlugin package and the test data in the <plugin directory>/testData directory.
This test adds two test cases to this plugin project. To run test cases, run the `YourTest.test()` or `YourTest.test1()` method, respectively.

For detailed information about testing and all related procedures, refer to Testing and Testing Support in the IntelliJ IDEA Web Help.