Intellij Idea plugin development

Wprost zwracają uwagę na to, że dokumentacja nie jest pełna i pisanie pluginów może wymagać przeglądania kodu źródłowego Intellij Platform   
<https://github.com/JetBrains/intellij-community>   
Lub innych pluginów  
<https://plugins.jetbrains.com/intellij-platform-explorer/extensions>

Zaleca się dbanie o dobry UX. Składa się na to łatwość w użytkowaniu, stabilność pluginu, wydajność, jego testowanie, niewielki rozmiar, tłumaczenie, itp.

Istnieje plugin który wspomaga developowanie pluginów (Plugin DevKit). Wymaga użycia Gradle! Używa pluginu gradlowego do zarządzania niezbędnymi zależnościami, uruchomieniem IDE z nowym pluginem, publikowania pluginu do Marketplace.

Dodaje w new project opcję stworzenia nowego pluginu do Intellij.

Klasa AnAction

Dwie metody

update –decyduje czy użycie akcji jest dopuszczalne. Defaultowo jest zawsze dopuszczalne więc nie wymaga implementacji. Intellij Platform co jakiś czas woła ją żeby zweryfikować. Dlatego musi być ultra szybka żeby nie zablokować Intellij.

actionPerformed – logika co się dzieje po wykonaniu akcji przez usera. Wymagana implementacja.

Klasa AnActionEvent

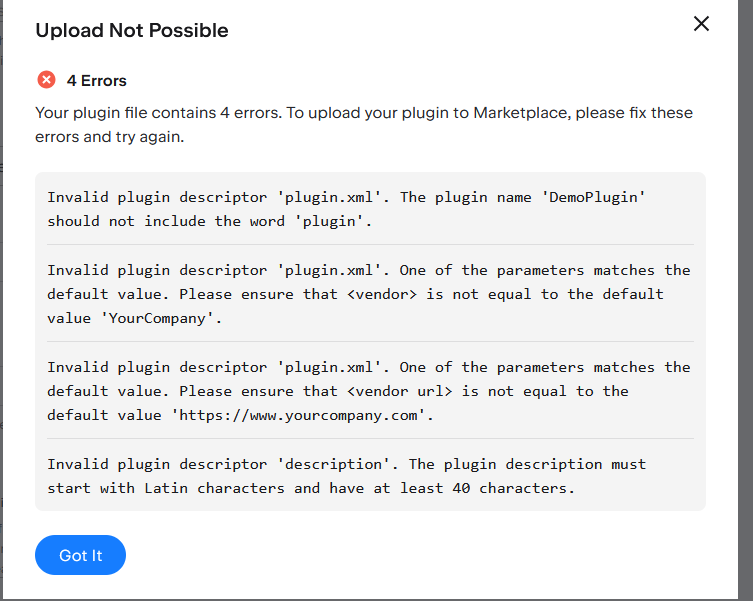
Zawiera licznie informacje o kontekście, np. co akurat jest zaznaczone kursorem.

Publikacja pluginu

Można to zrobić na Intellij Marketplace.

Wymagania:

1. Konto JetBrains;
2. Podpis – wymagany jest podpis cyfrowy który gwarantuje że pliki pluginu nie zostały zmodyfikowane. Task gradlowy automatyzuje to, trzeba tylko ręcznie wygenerować parę kluczy prywatny i publiczny;
3. Wgranie – pierwszy plugin musi być wgrany ręcznie;





Dziwne błędy:

1. Jeśli pracuje się na najnowszej wersji plugina kotlin to wymagane jest wprost określenie w naszym pluginie że wspiera Kotlin K2. Tego w tutorialu nie ma powiedzianego;
2. Plugin DevKit utworzył projekt z pluginem ze ścieżkami do kotlina mimo wskazania że chcę używać javy. Trzeba było zrobić rename, inaczej gradle nie pakował klas do jarki i rzucało ClassNotFoundException;