Robot Intellij Plugin

Adding Custom Language support to Intellij

Charles Capps

Table of Contents

What is it, and where can I get it?

What is the Robot Plugin?

- A Custom Language Plugin for Intellij
- Adds support for a new language, the Robot Framework
- Only compatible with Intellij 13 and above
- Go to Preferences → Plugins → Browse Repositories
- Search for “Robot plugin”
- As of writing, newest version is 1.4.5

How to get it?

Sort by: name

IntelliBot
CUSTOM LANGUAGES
8,770
4 weeks ago

Robot Framework Support
TOOLS INTEGRATION
4,160
5 days ago

Robot Plugin
CUSTOM LANGUAGES
1,647
2 days ago

CUSTOM LANGUAGES

Robot Plugin

1647 downloads

Updated 7/26/14 ver 1.4.5

The Robot Plugin provides integration with Robot Framework Test Case Files (http://www.robotframework.org). This plugin provides integration with Robot keywords implemented in Java that are in the same IntelliJ project as your Robot test case files. This plugin also provides many features supporting coding in the Robot Framework that are useful whether or not you use Java.

Features include:

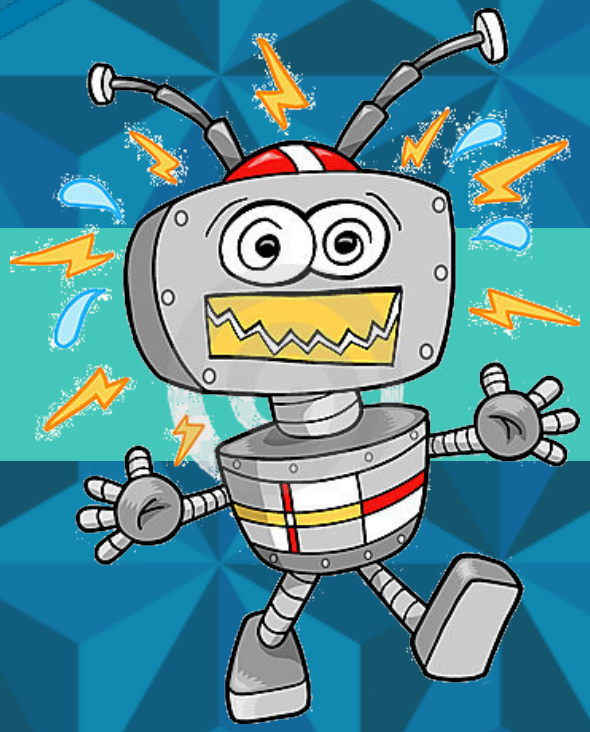
- Syntax highlighting
- Rename refactoring for Robot Keywords, whether defined in Java or Robot files.
- Structure View support for Robot files
- Invalid Robot syntax is underlined in red, so you can catch errors before committing your code.
- Go to declaration of a Keyword from the point of use. Jump to Java or Robot file Keyword definition.
- Find usages of Keywords defined in Java or Robot files
- Autocomplete Keywords and Test Case tags
- Autocomplete Variables that are in scope
- Find Robot Keywords and Test Cases by symbol name (ctrl+shift+alt+n) or (cmd+shift+alt+n) for Mac
- Go to Resource file by ctrl+click resource setting.
- Create a Robot Test Case file from a template
- ...and much more

by Charles Capps, an engineer at Jive Software

Change Notes

Version 1.4.5- rename refactoring for Robot Files. Resource Settings including a file will be refactored when you rename a Robot File.

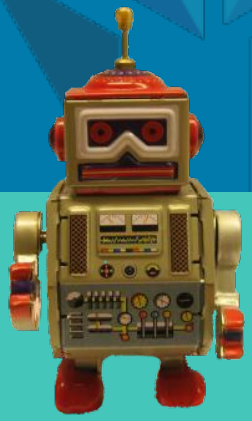
Why do such a crazy thing?



Why write a custom language plugin?

- The Robot Framework is used for our UI Test automation
- Back in February, there were no Intellij plugins for Robot
- Even now, the Robot Plugin is the only one with Java support
- Makes coding so much more efficient and less error-prone!
- Search features help you find Test Cases and Keywords
- Refactoring support saves time, and saves Jive money!

Demos of Features



Resolving References to Declaration

- Resolve Keyword to declaration in Java or Robot
- Resolve Variable to local declaration or Variables table
- Resolve Resource Files

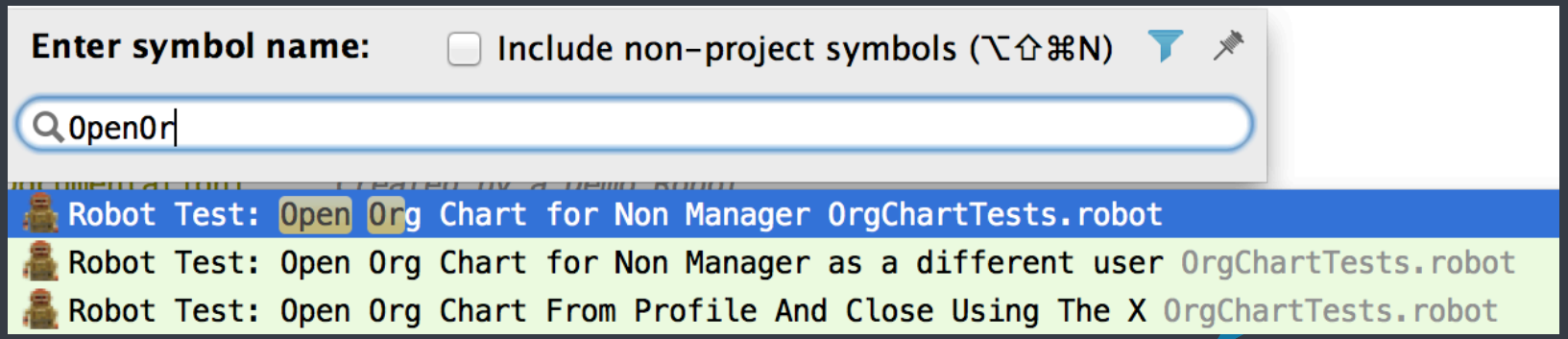
```
*** Testcases ***
####### Sort tests #######
# Given - 2 documents are created, and we wait for the list engine to index the documents
# When - Content Browse Page is loaded and we sort by "Date created: newest first"
# The Newest document shows up first
Sort content by date created newest first
    [tags]                  pri1
    [Documentation]         Created by Charles Capps
    ${personID}=                        Create New User And Login
    ${documentID1}=                     Create Minimal Document API        ${personID}
    ${documentID2}=                     Create Minimal Document API        ${personID}
```

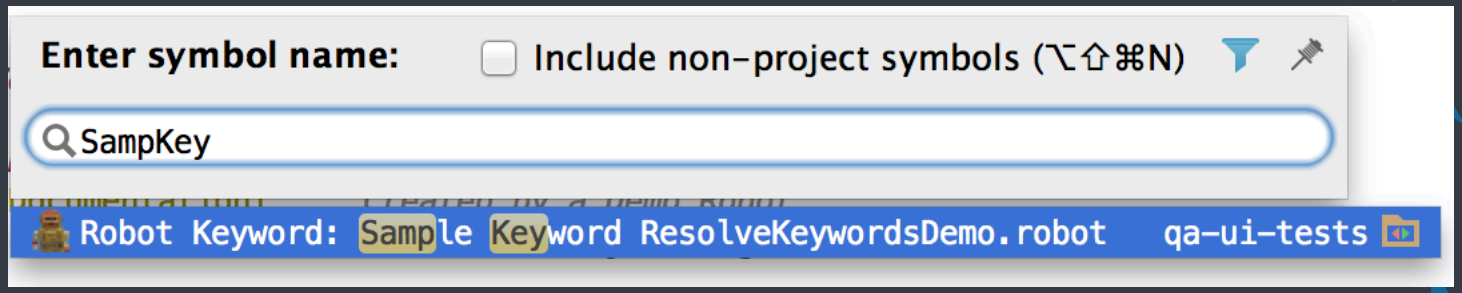
LoginKeywords
public String createNewUserAndLogin ()
throws Exception

Resolve Reference - Live demo time!

Searching for Test Cases and Keywords

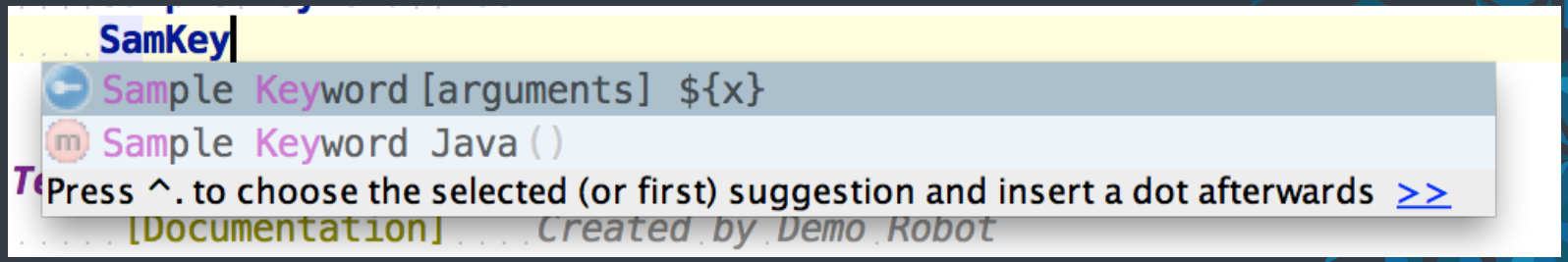
- Previously, you could only search by exact text occurrences
- Now, can use Navigate → Symbol or ctrl+shift+alt+n





Autocomplete Keywords, Tags, and Variables

- Start typing in a context where a Robot Keyword can be used
- Results will pop-up automatically!
- Made a conscious decision for Keyword autocomplete to use all keywords in the project, not just Keywords in scope (give explanation)
- Variables – complete from variables in local scope, or included in Resource files.

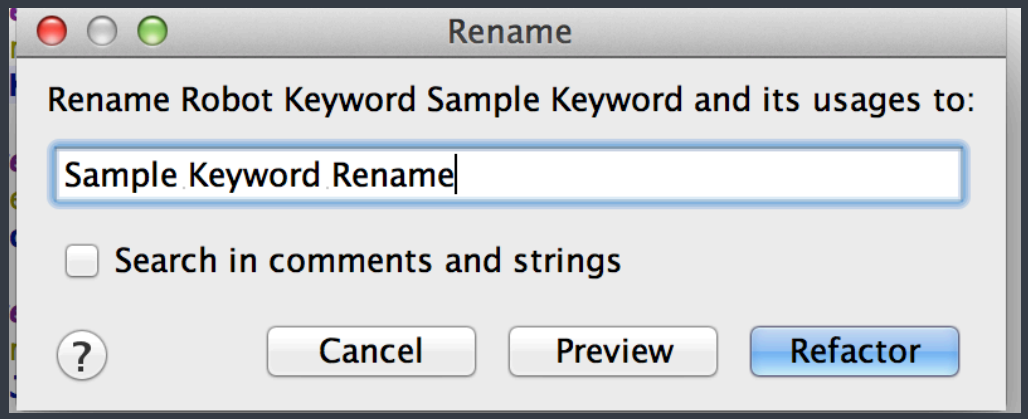


Autocomplete - Live demo time!

Rename Refactoring

- Rename Java Keywords or Robot Keywords with Shift+F6
- All usages will be renamed
- Rename Variables – will smartly rename usages for the right scope
- Rename Robot Files – Resource Settings will be renamed

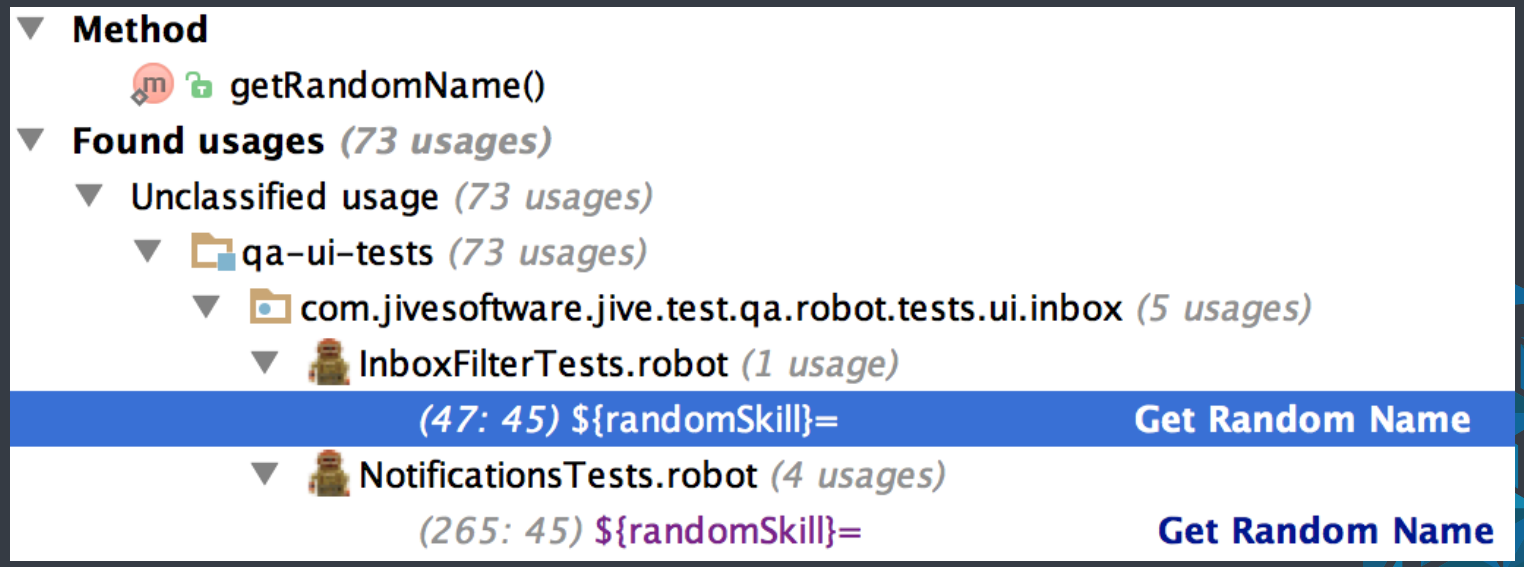
```
@RobotKeyword
public String SampleRenameKeywordJava() {
    return "foo";
}
```



Rename Refactoring - Live demo time!

Find Usages of Keywords and Variables

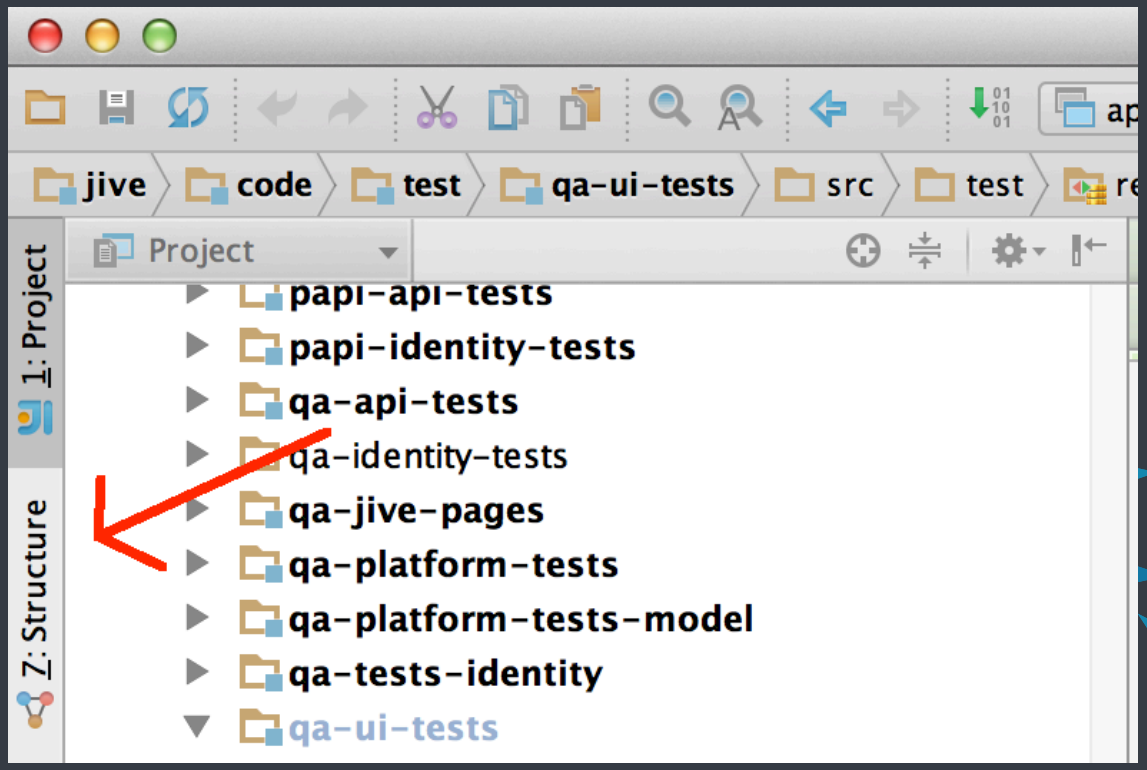
- To Find Usages: Right-click → Find Usages, or Ctrl+F7
- Keep in mind, Robot is case-insensitive and doesn't care about spaces and underscores.



Find Usages - Live demo time!

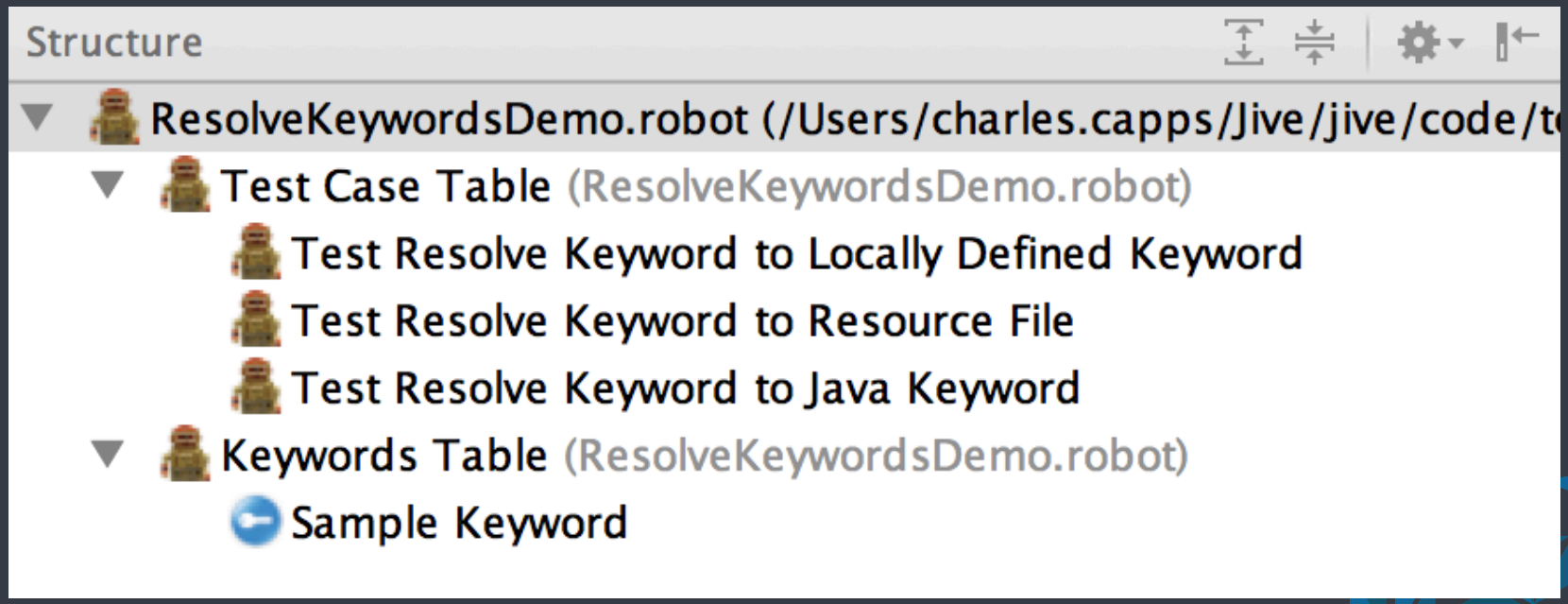
Structure View

- See at a glance what Test Cases and Keywords are in a Robot File



Structure View

- Displays all Test Case Tables and Keywords Tables



Under the Hood – How does it work?

Step 1. Implement a Lexer

Step 1. Implement a Lexer

- A lexer converts plain text into a sequence of Tokens.
- Tokens determine Syntax Highlighting.
- Intellij requires you to implement interface com.intellij.lexer.FlexLexer
- The Grammar Kit Plugin comes in handy.
- The Grammar Kit generates a Lexer from a JFlex file, .flex extension.

Example Lexing

```
*** Settings ***

Resource          SampleResource.robot
Force Tags        MyTag
```

=

[SETTINGS_TABLE_HEADING, NEWLINE,
NEWLINE,
RESOURCE_SETTING, COLUMN_SEP, ROBOT_FILE, NEWLINE,
FORCE_TAGS_SETTING, COLUMN_SEP, TAG, NEWLINE]

Step 2. Implement a Parser

Step 2. Implement a Parser.

- Parser converts the Tokens into an Abstract Syntax Tree (AST)
- AST is a tree representation of code.
- Leaves of the tree are Tokens.
- Branches of the tree are Program Structure Interface (PSI) elements
- Intellij requires you to implement com.intellij.lang.PsiParser
- Parser is generated from a BNF file (.bnf) using the Grammar Kit.

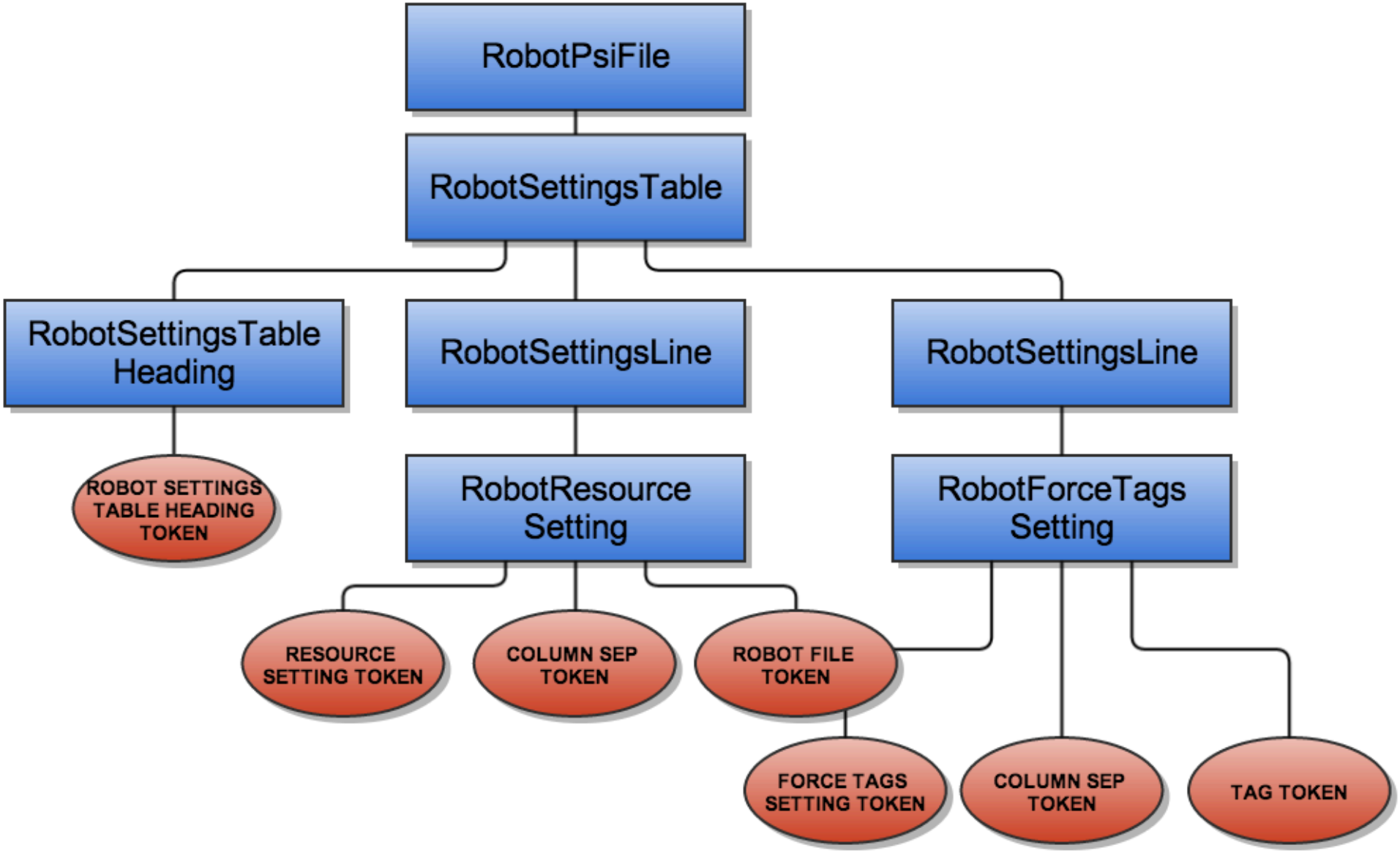
Example AST

```
*** Settings ***

Resource        SampleResource.robot
Force Tags      MyTag
```

=

Example AST


RobotPsiFile
RobotSettingsTable
RobotSettingsTable Heading
RobotSettingsLine
RobotSettingsLine
ROBOT SETTINGS TABLE HEADING TOKEN
RobotResource Setting
RobotForceTags Setting
RESOURCE SETTING TOKEN
COLUMN SEP TOKEN
ROBOT FILE TOKEN
FORCE TAGS SETTING TOKEN
COLUMN SEP TOKEN
TAG TOKEN

More facts about the Parser

- Syntax Errors are highlighted in Red
- Errors are when the Parser can't create an AST from the code
- There are tricks to make the parser resilient to errors
- One syntax error shouldn't cause the whole file to not be parsed

```
Test Invalid Syntax
    [Documentation]    Created by Robot Demo
    ${myVar}=          ${anotherVar}    # Invalid Syntax!

    :FOR   1  IN  @{array}    # Invalid Syntax!
    \  Log  Test Loop
```

Step 3. Implement Extension Points

Step 3. Implement Extension Points

- All Features of a Plugin are Extension Points in the plugin.xml
- Extensions are inside the <extensions> tag:

```
<extensions defaultExtensionNs="com.intellij">

<!-- Put extensions here! -->

</extensions>
```

- The attribute defaultExtensionNs="com.intellij" is the just the base package for the Intellij provided extension points

Extension Points

- File Type Factory – registers the .robot extension

```
<fileTypeFactory implementation="com.jivesoftware.robot.intellij.plugin.lang.RobotFileTypeFactory"/>
```

- Syntax Highlighter – define how Tokens get syntax highlighting

```
<syntaxHighlighter key="robot"
                   implementationClass="com.jivesoftware.robot.intellij.plugin.lang.RobotSyntaxHighlighter"/>
```

- Parser Definition – register your Parser

```
<lang.parserDefinition language="RobotTestFile"
                       implementationClass="com.jivesoftware.robot.intellij.plugin.parser.RobotParserDefinition"/>
```

- Reference Contributor – define how PSI elements resolve to declaration

```
<psi.referenceContributor
        implementation="com.jivesoftware.robot.intellij.plugin.elements.references.RobotReferenceContributor"/>
```

…and a whole lot more!

- Find Usages Provider
- Custom Usages Searcher
- Go to Symbol Contributor
- Completion Contributor
- Rename Psi Element Processor
- …

Links

- Github Page: https://github.com/jivesoftware/robot-intellij-plugin
- Jetbrains page: http://plugins.jetbrains.com/plugin/7430?pr=idea
- My blog: how-to-write-a-custom-language-plugin-for-intellij
- Grammar Kit: https://github.com/JetBrains/Grammar-Kit
- Jetbrains tutorial on Developing Custom Language Plugins

Questions?