Adding New Language Support

One can add support for other programming languages to Apache NetBeans using its Rich Client Platform API.

Support to a programming language can be considered complete when it provides support for:

1. File type recognition
2. Project type
3. Semantic syntax highlighting and braces matching
4. Code completion
5. Navigation (jump to definition, peek definition, find all references, symbol search)
6. Types and documentation on hover
7. Code formatting
8. Configuration
9. Refactoring (e.g. rename, move)
10. Error squiggles and apply suggestions from errors
11. Debugging
12. Snippets
13. Build tasks

There are a number of ways to add support for a programming language, each one of them having its pros and cons. Most of the above are commonly supported by Apache NetBeans' Rich Client Platform APIs, but for some of them, different APIs have been developed that support one of the following technologies:

- JavaCC
- ANTLR
- Language Server Protocol (LSP)

How to add support in NetBeans to a programming language?

First, you will need Apache NetBeans source code. Please download or clone it from here and build it.

1. File Type Recognition

The first thing to do is for NetBeans to be able to recognize the file type. E.g. if you are adding support for the kotlin programming language you would like NetBeans editor to be able to recognize `.kt` source files.

Use the **`New File  Module Development  File Type`** wizard. A MIME type must be specified. This MIME type will be the key under which other services will be looked up.

See the File Type Integration Tutorial for more details on how to add File Type recognition support. A `DataObject` file will be created with a number of annotations.

Important Note! *Make sure to create your module inside your cloned Apache NetBeans source code. Your module will need some files from `nbbuild` folder.*

Take a look at `java/kotlin.editor/src/org/netbeans/modules/kotlin/editor/KtDataObject.java` and `rust/rust.sources/src/org/netbeans/modules/rust/sources/rs/RustFileDataObject.java` as examples of `DataObjects`.

Hint! To open a source file easily, click on **Window Favorites** and navigate to the source file (e.g. a `.kt` file if you are adding support for kotlin, or a `.rs` source file for Rust)

2. Custom project types

You may want to create a new "Project Type" for your specific language. More explicitly, when you click on **File New Project**, you can customize the *New Project* wizard dialog to create a new project for your language. For instance, "Rust" projects usually have a folder structure defined by the "cargo" tool. `rust/rust.project.api` module allows the user to create a new cargo project.

The NetBeans Project Type Tutorial is a good starting point.

3. Semantic Syntax highlighting and brace matching

Without syntax highlighting, NetBeans opens the source file as a text file. Each technology uses different ways to support syntax highlighting:

- JavaCC
- ANTLR
- Language Server Protocol (LSP)

4. Code Completion

- JavaCC
- ANTLR
- Language Server Protocol (LSP)

5. Navigation

Navigation includes: jump to definition, peek definition, find all references, symbol search etc.

- JavaCC
- ANTLR
- Language Server Protocol (LSP)

6. Types and documentation on hover

7. Code formatting

8. Configuration

NetBeans IDE provides the *Options* window (menu **Tools Options** or **NetBeans Preferences** on MacOS) that allows the user to customize it. You can provide any customizations for your language support in the *Options* window, too. E.g. you could allow the user to provide the path to the Kotlin compiler. See e.g. `rust/rust.cargo` module that allows the user to provide the path to `cargo` for Rust projects.

The NetBeans Options Window Module Tutorial explains how you could do that.

9. Refactoring

Refactorings like e.g. rename, move are supported by all major IDEs.

- NetBeans Refactoring Module Tutorial - Part 1
- NetBeans Refactoring Module Tutorial - Part 2

10. Error squiggles and apply suggestions from errors

11. Debugging

12. Snippets

13. Build tasks

Resources

NetBeans Specific Resources

- Rich Client Programming: Plugging into the NetBeans Platform https://www.amazon.com/Rich-Client-Programming-Plugging-NetBeans/dp/0132354802
- Apache NetBeans Platform for Beginners https://leanpub.com/nbp4beginners
- Lahoda J. (2019), "LSP Client demo - (ba)sh language server", ASF.
- Cardona J.R. (2018), "Quick Start: Creating Language Tools In NetBeans IDE", DZone.
- NetBeans Platform Learning Trail https://netbeans.apache.org/kb/docs/platform/index.html
- Kostaras I. et al. (2020), *Pro Apache NetBeans,* APress, Chapter 11, "Writing a Plugin for NetBeans".

Other resources

1. Clinton J.L. (2021), *Build Your Own Programming Language*, Packt.
2. Nadeeshaan G. & Nipuna M. (2022), *Language Server Protocol and Implementation: Supporting Language-Smart Editing and Programming Tools*, APress.
3. Parr T. (2010), *Language Implementation Patterns*, The Pragmatic Programmer.
4. Stalla A. (2021a), "Converting from JavaCC to ANTLR", Strumenta.
5. Stalla A. (2021b), "Go to Definition in the Language Server Protocol", Strumenta.
6. Singh V., *Basics of Compiler Design,* Anniversary Edition.
7. Tomassetti G., "The ANTLR Mega Tutorial", Strumenta.
8. Watt D.A. & Brown D. F. (2000), *Programming Language Processors in Java,* Prentice Hall.