

Integrate Language-Server with VS-Code extension



Manserpatrice · Follow

Published in Nerd For Tech · 4 min read · Nov 11, 2022



```
let lc: LanguageClient;

export function activate(context: ExtensionContext) {
  // The server is a started as a separate app and listens on port 5007
  let connectionInfo = {
    port: 5007
  };
  let serverOptions = () => {
    // Connect to language server via socket
    let socket = net.connect(connectionInfo);
    let result: StreamInfo = {
      writer: socket,
      reader: socket
    };
    return Promise.resolve(result);
  };

  let clientOptions: LanguageClientOptions = {
    documentSelector: ['gui'],
    synchronize: {
      fileEvents: workspace.createFileSystemWatcher( globPattern: '**/*.gui')
    }
  };
};
```

Medium Search Write

Eclipse Language Plugin which was based on XText. In order to be able to use the Language Server on other IDEs than Eclipse, we created a new Language Server based on Microsofts Language Server Protocol (LSP).

My part was to integrate the newly created Language-Server with VS-Code. I chose this Editor over IntelliJ IDEA since it is known to have an easier and better documented integration for Language-Servers than IntelliJ has.

Communication between Client and Language-Server

First of all, when the user opens a document, the client sends a notification with the content of the document to the Language Server.

The server sends a notification with the Diagnostics of the current document back to the client.

If the user edits a document, the client sends a new notification with the current url of the document and the differences compared to the old version. This way it does not have to upload the whole document after every change.

If the user executes “Go to definition” he sends a Request to the server with the current documentURI and the position of the cursor.

The server sends back a response with the location of the linked class, method or variable.

When the user finally closes the document, it sends a notification with the documentURI to the server so that it can remove the document from its current context.

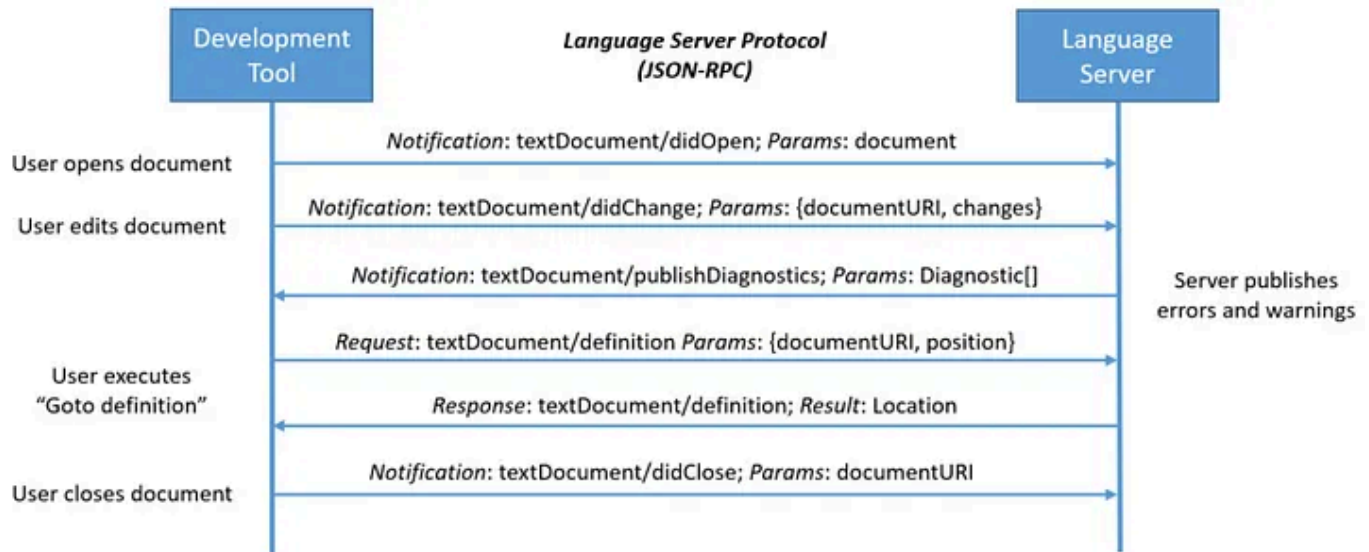


Figure from <https://microsoft.github.io>

The default port for Language-Servers is port 5007. In this example we will also start our Language-Server on this port.

You can read more about the communication between server and client on <https://code.visualstudio.com/api/language-extensions/language-server-extension-guide>

Creating the extension

To create a language extension, you need to create a project with the following project structure:

/package.json

/`<extension>`.configuration.json

```
/src/extension.ts  
/src/tsconfig.json
```

Where `<extension>` should be the extension of the new language that you want to support.

In the following sections of this tutorial, I will continue using “**gui**” as the extension of the language files.

The content of the `package.json` file looks like this:

Keep an eye on the parts from line 22 til 35 since this is the extension related part. It is really important that you add the dot before the extension name on line 33.

The `gui.configuration.json` file will be used for local support of the language like auto-closing brackets and comments.

All these configurations are needed to improve the workflow of the enduser and providing basic functionalities for autocompletion.

The basic `/src/tsconfig.json` file looks like this:

Now the really interesting part is located in the `extension.ts` file.

A `LanguageClient` gets created with a connection to port 5007 on localhost.

On line 27 the gui extension is needed, so that the server knows which files he should track.

The fileEvents that should notify the server are defined on line 29. With this configuration the client only notifies the server if a “.gui” file changed.

On line 37 the client is started.

The deactivate() function is needed so that the server can close the connection if the extension gets disabled.

Running the extension

To run the extension, you need to open this project in VS-Code, if you are not already working with it, and press “Debug/Run”. This will open a new VS-Code window with you custom extension already installed and enabled. Now you can create a new file with the defined extension. You should now be able to use autocomplete functionality and semantic checking of your LSP server.

Further information can be found on these sites:

- <https://code.visualstudio.com/api/language-extensions/language-server-extension-guide>
- <https://learn.microsoft.com/en-us/visualstudio/extensibility/adding-an-lsp-extension?view=vs-2022>
- <https://idiomaticsoft.com/post/2022-04-29-create-vscode-client/>

Reflection

What went good

I think that the actual implementation of the extension was not that hard and I got it working in a pretty short time.

What needs improvement

The hardest part in this small project was to understand what exactly the Language Server Protocol is and how it works. At the beginning I did not read that much in the official docs but rather tried to find answers to specific questions I had at that moment on StackOverflow. The Problem with that approach was that I never had a full understanding of the topic itself and therefore ran into issues that I would not have had if I just informed myself a little bit better about that topic beforehand.

Vscode

Vscode Extension

Lsp

Language Server Protocol

Intellisense



Written by Manserpatrice

Follow

100 Followers · Writer for Nerd For Tech

Developer Experience Engineer

More from Manserpatrice and Nerd For Tech




 Manserpatrice in Nerd For Tech

Log4J2 and JUnit 5 | How to test logs with JUnit 5

Recently I had the problem, that I needed to log some warnings for my application. Since ...

Mar 30, 2021  9  1




 Olenin Slava in Nerd For Tech

Why Zig programmers are making so much money

Are you wondering why Zig programmers make so much money? Guess what? It is not...

★ Aug 5  723  10



 Mayank Sharma in Nerd For Tech

I discovered this amazing IntelliJ feature after 6 years.

How a random debugging session helped me discover this amazing feature.

★ Aug 7  212  6



 Manserpatrice in Nerd For Tech

Install old versions of Apps with Ansible and Homebrew

Sometimes, there are new software releases, that have new requirements to your IT-...


May 17, 2021  27



See all from Manserpatrice


See all from Nerd For Tech


Recommended from Medium


 Alexander Nguyen in Level Up Coding


The resume that got a software engineer a \$300,000 job at Google.


1-page. Well-formatted.

 Jun 1

 18.3K


 298





 Andrew Zuo


Async Await Is The Worst Thing To Happen To Programming

I recently saw this meme about async and await.


 Jun 22

 2.9K

 164




Lists




Staff Picks

716 stories · 1234 saves




Self-Improvement 101

20 stories · 2583 saves



Stories to Help You Level-Up at Work

19 stories · 754 saves



Productivity 101

20 stories · 2227 saves


 Kenichi Sasagawa

Why I Discontinued the Windows Version of Easy-ISLisp

Question

Aug 12  4




 Olenin Slava in Nerd For Tech

Why Zig programmers are making so much money

Are you wondering why Zig programmers make so much money? Guess what? It is not...

★ Aug 5  723  10




 Mohit Vaswani

6 Mac Apps That Are So Good, They Feel Illegal

If you are a Mac user looking for the best apps to enhance your productivity, speed up your...

★ Aug 13  465  9



 Nov Tech in Mac O'Clock

Here are the Reasons I Regret Buying the MacBook M3 Pro.

I just returned my new MacBook M3 after 3 days. It's a trap and well-played, Apple.

★ Aug 12  437  20



See more recommendations