Language Server Protocol implementation for Elixir

Summary

From the idea's page, Language Server Protocol is an attempt to establish a common, cross-language and cross-editor protocol for tools to implement features like auto complete, goto definition, find all references, and alike. It mediates between the editor (the client) and a language smartness provider (the server). This project aims to develop a server that implements the Language Server Protocol specification for the Elixir language.

Benefits

There are several benefits here:

Firstly, the protocol is open, well-defined and designed to be a cross-language and cross-editor(cross-platform maybe?) protocol. The protocol requires that the client and server communicate over JSON-RPC, which is a very simple and lightweight remote procedure call protocol.

Secondly, servers can be used across any client(editor) that implements the client side of the protocol. This will greatly improve the editor tooling as mentioned here and basic requirements like code completion, hovering, jumping to definitions, finding references etc., will become easy to implement and support for several languages across many editors and platforms.

Thirdly, this protocol is being adopted widely, with various language servers and clients implementing the protocol evident from here and here. This project aims to develop a language server for Elixir, which will eventually make Elixir programmers enjoy writing Elixir code.

Deliverables

The end goal of this project would be to have a language server for Elixir which, at minium, provides the following features:

- Code Completion
- Hover
- Jump to definition
- Find all references

Apart from the above, the plan is to test the server by tests that simulate client behavior, and integrate the server with atleast one client(vscode probably). Initially we will start out with using `JSON-RPC` over `TCP` for communication between server and client, and eventually add `JSON-RPC` over `STDIN/STDOUT` as well.

Below I have drawn up an estimated schedule of this project over the summer.

May 5 - May 30 During the community bonding period, I wish to accomplish the following:

- Introduce myself to the community and get to know my mentor and community well
- Carve out a regular schedule and get adjusted to the time difference between timezones
- Read through the protocol specification all over again
- Discuss with antipax regarding developing transport for `JSON-RPC` over `STDIO` for the `jsonrpc2-elixir` library that we plan to use
- Read through implementations of other language servers and also the `alchemist-server`
- Discuss more on the design and resolve any issues that might come up in further discussions before the coding period begins

May 31 - June 10

Begin working on the project. Start by setting up basic `JSON-RPC` communication over `TCP` between server and a simulated client. Also, define the JSON structures as defined in the protocol specification, along with their documentation. Write tests for the communication between server and client. Use `posion` for `JSON` and `jsonrpc2-elixir` for `JSON-RPC` communication. Setup a CI(Travis?) for automated testing.

June 10 - June 20

Continue defining the JSON structures if any are left, and start with working on providing code completion. Borrow ideas from IEx.Autocomplete and refer `alchemist-server` and other language server implementations as well.

June 20 - June 26

Finish the code for providing completions. Write documentation and tests. We have one working feature now, and we should work on integrating the server with a client that implements this protocol. I would like to pick vscode as the client against which we would do the testing, apart from the automated tests that we write.

June 26 - June 30

Work on editor integration and fix bugs and issues and cleanup code for Midterm evaluations.

July 1 - July 15

The next point to start would be on jumping to defintion and hovering. Both of them would go hand in hand to quite some extent and again, we are to follow what is mentioned in the protocol and refer to other server implementations including `alchemist-server`.

July 16 - July 23

Finish with the implementations of hovering and jumping to definitions, and get them to work with vscode.

July 24 - July 28

Write rigorous tests, clean up the code and fix bugs and update the documentation for second evaluation.

July 29 - Aug 5

At this point, all our goals except find all references should be implemented. So, get started with working on finding all references. For this feature, we can refer to, and use `Mix.Tasks.Xref`. While working on this, I also plan to extend this a bit to provide symbols in current workspace/context, which would be a nice plus to the set of features. I believe work on finding all references can be extended easily to do so.

Aug 6 - Aug 13

This week would be focused on writing tests for the implemented features from previous week. Also, we have a basic, functional server at this point. So, we should work on transport now, and get `JSON-RPC` over `STDIO` working.

Aug 14 - Aug 29

Work on adding `STDIO` transport for `JSON-RPC` in our server, and make it as much transport agnostic as possible. Fix bugs and issues, refine and clean up the code, get the tests to pass, update the documentation and prepare the code for final submission.

Post GSoC Period

There is a lot of scope and features that can be added, like linting, formatting etc. So, continue working and improving this as much as possible, and respond to issues and fix them. Be associated with the community and stay active and take the project to its completion, implementing as much of the protocol as possible. Also, get it to work with as many editors as possible.

I haven't mentioned in the above schedule, but I plan to make blog posts at regular intervals as well regarding my experiences.

Related Work

There are several implementations of language servers for different languages as mentioned above. In lieu of my interest in this project, I have been through the `python` implementation. Also, there is a very similar Elixir project with similar goals, with a different API and a different implementation, alchemist-server. I plan to refer to both these repositories to borrow some ideas and generally use them as a reference to implement the server in a best way possible. The fact that there are many existing language servers out there is a great advantage for this project, because when we are stuck or not sure how to proceed, we can refer to their ideas and design, and then decide which would be the right way to go.

Personal Information

I am Nishith Kumar Shah, a linux enthusiast from India(UTC +05:30). I am currently a fourth year undergraduate student of the Department of Mathematics, at Indian Institute of Technology, Kharagpur, where I am studying towards an Integrated M.Sc in Mathematics and Computing. My Github username is nishithshah2211. I can be contacted at nishithshah.2211@gmail.com.

I am a Vim person and work a lot with C, Lua, Python and Bash. I was first introduced to Elixir 2 years ago, during my internship at a startup in Bangalore, India, during the month of December. I had to design a message server that would eventually replace their existing chat backend and Elixir was chosen for this because of its great support for concurrency and frameworks like Phoenix. Apart from that, I haven't had a lot of opportunities to use Elixir in any of my projects since then.

Regarding my experience, last summer, I was a GSoC student under the libvirt ,where I worked on improving the autocompletion module of the `virsh` shell to perform intelligent autocompletion. It involved reading and writing C code that was fairly pointer intensive. You can find the final summary of the project here , and the patches. It was a great summer and I learned a lot about good programming etiquettes and about C itself, as a language and the intricacies involved. Apart from that, I am a part of the team that maintains the code(lua scripts, bash scripts for backups, databases, internal LAN website etc.,) that powers the PtokaX DC++ hub of IIT Kharagpur. You can find more information about it here and my contributions. I have also submitted a minor fix to SymPy, which fixed the `solve` function that was buggy in case of piecewise non-linear functions.

I can work on this project over the summer and beyond and I am very excited for this. My semester finals get over on April 30 and I can easily devote 35+ hours on an average per week. As an Elixir enthusiast, this project is a great opportunity for me to get involved with the Elixir community and I'd love to be the one to be working on the Language Server for Elixir.

Links

- Language Server Protocol's Repository: https://github.com/Microsoft/language-server-protocol
- Description of the Idea: https://github.com/beamcommunity/beamcommunity.github.com/wiki/Project:-Elixir#idea-4-language-server-protocol-implementation-for-elixir
- Protocol specification: https://github.com/Microsoft/language-server-protocol/blob/master/protocol.md
- Implementations of Servers for different languages and Clients supporting the protocol: https://github.com/Microsoft/language-server-protocol/wiki/Protocol-Implementations
- JSON-RPC library for Elixir: https://github.com/fanduel/jsonrpc2-elixir
- alchemist-server, similar to language server: https://github.com/tonini/alchemist-server