2019-03-26 - golang-tools session - meeting notes

Previous session notes

2019-02-26 - golang-tools session - meeting notes

Recording

https://www.youtube.com/watch?v=fJsi85TunPs

Attendees

- Jay Conrod
- Ian Cottrell
- Paul Jolly
- Michael Matloob
- Roger Peppe
- Rebecca Stambler
- Marwan Sulaiman
- Billie Cleek

Notes

gopls support in vim-go

- [Billie] Now on by default for autocompletion, jumping to definitions. Escape hatches may go back to gocode. Would like to be able to use gopls to find out info about identifiers, similar to guru definition. Seems like an LSP deficiency.
- [Ian] There are some ways, like hover. gopls has a guru equivalent mode on the command line. That should show what to do.
- [Billie] Nice foundation, should be easy to hook in additional changes. For autocompletion with gocode, we get full function signature, but gopls doesn't provide that right now.
- [Ian] LSP has signature help that's supposed to drive that sort of thing.
- [Paul] There's a gopls wiki. For people like Billie. Is it worth pulling together some documentation from the editor perspective?
- [Rebecca] Definitely. It would be nice to have a list of the features we support, but a lot of stuff is changing quickly. Feel free to edit. Also file issues about what should be in the different fields.
- [Billie] Would be nice to put the go doc into the documentation field.
- [Paul] Is there any guidance on what contributors should work on?
- [Rebecca] People have messaged me on Slack, probably the best way to reach out. Happy to answer questions and review CLs.
- [Rebecca] There's a gopls label on the issue tracker. Issues for new contributors are marked as "suggested". Issues for me and Ian are marked "fundamental". This should be written in the wiki, too.

- [Ian] We've been focused on making the core functionality, but the details can use more tweaking. Happy to take contributions there.

Demo with gopls in VSCode

- [Rebecca] Will show how to configure in settings. For now you have to enable language server and set gopls to be the default language server. This will eventually be the default.
- [Rebecca] Features have to be enabled. Not all are supported yet. Eventually, they'll all be on by default.
- [Rebecca] Need to turn off buildOnSave and vetOnSave to avoid getting findings from both build and gopls.
- [Rebecca] formatOnSave, organizeImports needed for goimports functionality.
- Completion should show signature, but hitting enter will only insert symbol. Open parenthesis will show signature information. Hover shows full signature.
- Go-to-definition works. Organize imports works on save or as a command. Vet and compile errors show up in Problems pane.
- Hover over type definition shows tags, hard to read. Would be a great starter issue.
- Go-to-definition on an import path doesn't work.
- [Roger] godef goes to the package directory.
- [Ian] It's unclear what the editor should do. Opening a directory in VSCode means opening a new workspace, which is not what we want.
- [?] Does it go-to-definition work inside $GOPATH/pkg/mod?
- [Ian] It should work in modules, but it's broken under some circumstances in GOROOT itself.
- [Rebecca] We're caching now, so speed is much better.
- Wiki has list of every editor that supports gopls.
- [Roger] How much memory does this use in a large project?
- [Rebecca] No information about that right now. Should get some metrics. It was crashing earlier, less of a problem now.
- [Paul] Do you end up firing an instance of gopls per module?
- [Rebecca] One of the starting parameters is the root URI. Opening a different file may cause it to crash. Ian has thoughts, not sure about the answers yet.
- [Marwan] Should we have a unique template for gopls? Might be useful to copy alternate tool settings to make issues more reproducible.
- [Ian] Would like to add a bug verb to the command line, which would submit that information, logs, and such.

Discussion on go/packages

- Adding go/packages to standard library
    - [Michael] About adding go/packages to the standard library. Personally don't think it's ready, prefer to hold off for a few more releases or not at all. Will be sorted out when Russ comes back, interested in hearing other opinions.

    - [Paul] Is there any rush?
    - [Michael] go/build doesn't really work anymore, need to keep working functionality in the standard library. There is an idea that people should be able to write programs that only depend on the standard library, I think that's a mistake.
    - [Paul] In #tools slack, someone was using go/build to determine directory of a package. In go/packages which supports other build systems, there is no concept of a package directory. Moving this into the standard library would lose information compared to go/build.
    - [Michael] A lot of things go/build does aren't necessary. Would prefer people don't make assumptions. Example, someone was writing a generator that depends on vendored code. They can't pass in package paths because with vendoring, imports depend on where you are.
    - Official line is that go/build is meant to be used by the go tool itself during a build. Long term, we'd like people to use go/packages instead.
- Splitting mode bits: Unifying LoadSyntax and LoadAllSyntax
    - [Michael] When Alan was creating this, there was an idea of a linear order of modes, each mode would provide more information. Most people calling LoadSyntax don't care about dependencies, but they get all the dependency information because LoadSyntax > LoadImports. So we'd like to ask people for specifically which bits they want.
    - CL 162140 is this proposal. So users can ask for fields on the packages they ask for, but there's no way to say what fields you get on dependencies. So you get the same information on dependencies right now.
    - Happy to have more feedback and comments on this.
    - I'd love to add a v2, but we're stuck in this crazy world of versions. Opens a rats nest of issues, and we won't be adding explicit versions for a while.
    - [Paul] How bad would it be to break compatibility?
    - [Michael] If we did a v2, we'd remove the load modes and just use the bits. This is mainly a performance question. (The strict compatibility guarantee is another reason not to put it in the standard library).
    - [Ian] If we did move it, we'd make these changes on the way in.
    - [Michael] Breaking backward compatibility is not an option now though. We already got in trouble for making changes we said we'd make when it was marked experimental.
    - [Roger] Unfortunate that go/packages shells out to the API.
    - [Jay] Necessary because gopls may not be built with the same version of Go as is being used in the workspace. If the question is about performance, I'm working on performance improvements for 1.13, on the order of 30-50% for "go list", hopefully more.

Fact support in analysis

- [Michael] API makes distinction between facts and results. Facts are passed between the same analysis on different packages. Results are passed between different analyses on same package.
- Fact API is convenient, and request is to make facts available as results.
- Had discussion with Alan. This might make the API a little more convenient, but it muddles the distinction.
- Once we make a change to the API, we can't reverse it. We should be very conservative.
- [Jay] Does this means fact types would be a new API for anlayses? So it also imposes compatibility constraints on analysis authors/
- [Michael] The idea was that authors could opt in or out by exporting the type.

Other business

- [Ian] There are some recent issues with repositories that have no go.mod in the root folder, but multiple go.mod in subfolders. Can also have problems for a replacement module being interpreted as a separate module, not within a child workspace.
- gopls could allow editors to make more decisions. We cache heavily, meaning we need full dependency graph. Each cache is specific to a main module / workspace / VSCode window.
- We have a concept called a view that the cache lives inside. At the moment, cache is tied to the init call in the root dir.
- LSP has concept of workspace folders, and we could start a new view per workspace folder. One instance of gopls could support multiple caches and multiple views.