rootpd#blog

Articles Talks About

Setting up Intellij IDEA for your Golang project

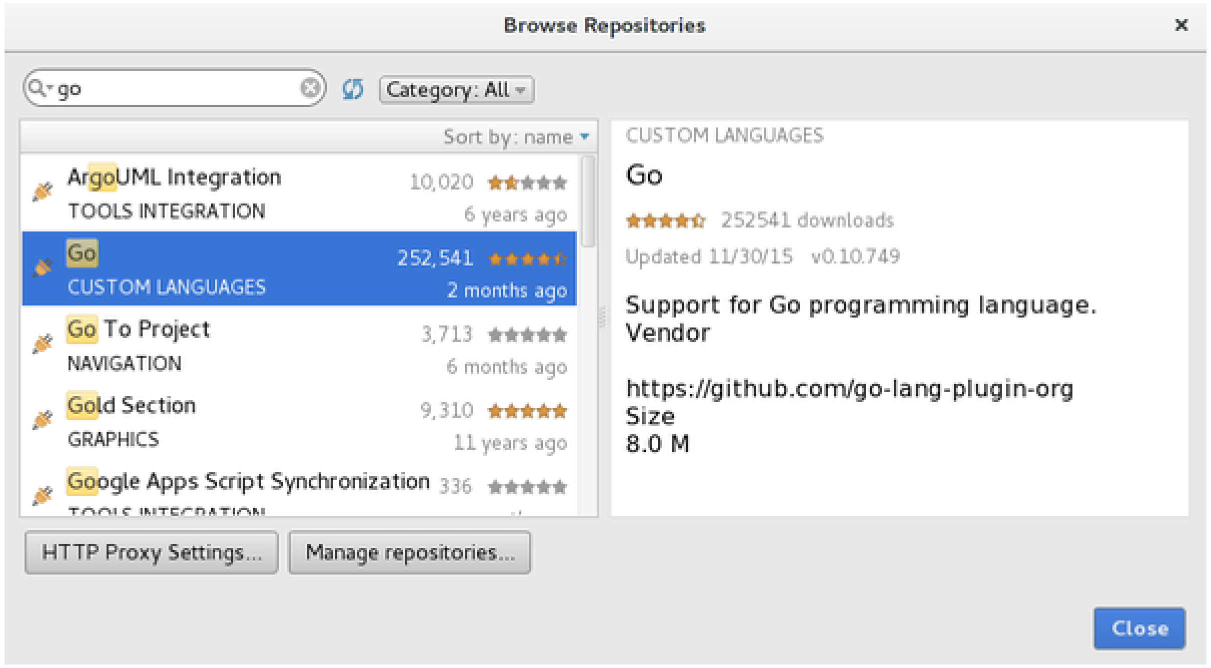
2016-02-04

After two full months of testing Golang plugin in Intellij IDEA, I'm happy I can say it's sufficient and there are no real limits in using it - no more Sublimes, Atoms, LiteIDEs. If you're not a Vim user and want to try real IDE with Golang, "go" for this. Here're the steps to make your live easier too.

Installing Go and Golang plugin

I assume you have the Golang downloaded. If not, please download it first here.

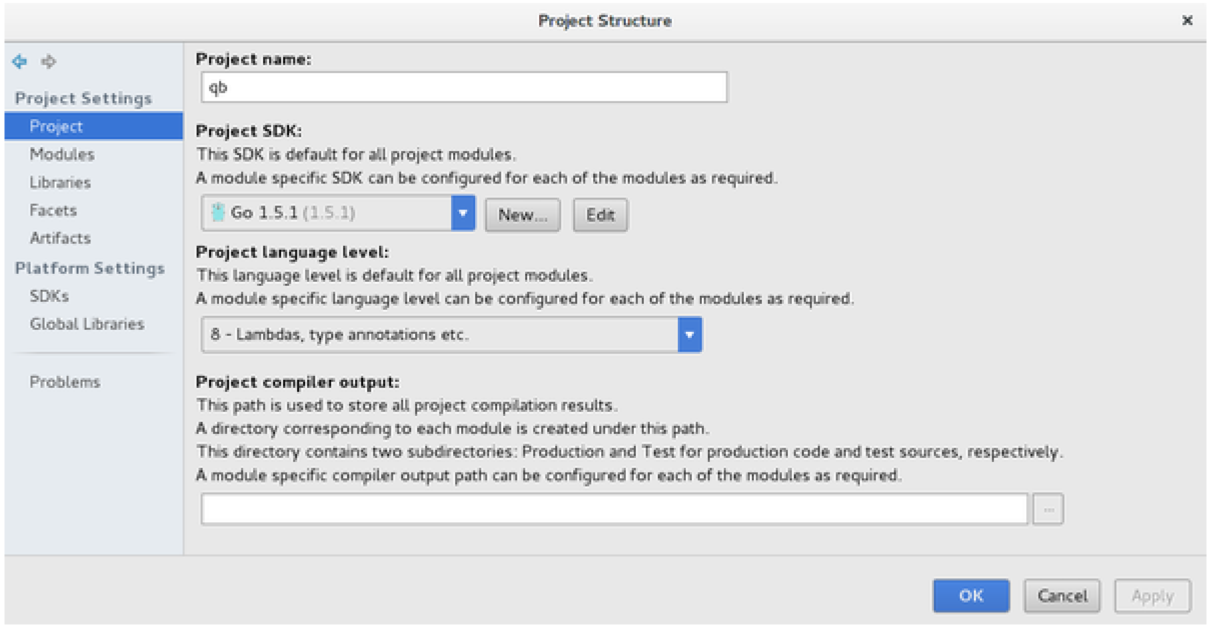
In Intellij Idea go to the plugin list and browse the repository for Go plugin. You can't miss it. After you install it, restart your IDE.



ArgoUML Integration	10,020	6 years ago
TOOLS INTEGRATION		
Go	252,541	2 months ago
CUSTOM LANGUAGES		
Go To Project	3,713	6 months ago
NAVIGATION		
Gold Section	9,310	11 years ago
GRAPHICS		
Google Apps Script Synchronization	336	

CUSTOM LANGUAGES
Go
252541 downloads
Updated 11/30/15 v0.10.749
Support for Go programming language.
Vendor
https://github.com/go-lang-plugin-org
Size
8.0 M

In case you're working on the existing (imported/opened) project, check whether you have proper SDK selected. Idea probably asked you about it first time you opened it via notification, but if it's not set correctly, some things below wouldn't work.


Project Structure

Project Settings: Project, Modules, Libraries, Facets, Artifacts

Platform Settings: SDKs, Global Libraries

Problems

Project name: qb

Project SDK:
This SDK is default for all project modules.
A module specific SDK can be configured for each of the modules as required.
Go 1.5.1 (1.5.1) New... Edit

Project language level:
This language level is default for all project modules.
A module specific language level can be configured for each of the modules as required.
8 - Lambdas, type annotations etc.

Project compiler output:
This path is used to store all project compilation results.
A directory corresponding to each module is created under this path.
This directory contains two subdirectories: Production and Test for production code and test sources, respectively.
A module specific compiler output path can be configured for each of the modules as required.

OK Cancel Apply

Integrating Go tools

There are couple of preconditions you should meet before we proceed – mostly to avoid scenarios that something is working in the terminal and not in IDEA and vice versa:

- `$GOPATH` is set in your environment (via `~/.profile`)
- `$GOPATH/bin` is in your `$PATH` (via `~/.profile`)
- If you use v1.5.x of Golang and want vendor experiment enabled, set `$GO15VENDOREXPERIMENT=1` in your environment (via `~/.profile`)
- If you use custom Golang build, `$GOROOT` is set in your environment (via `~/.profile`)

Now add $GOPATH into global libraries so it nicely resolves all your dependencies:

- File - Settings - Languages & Frameworks - Go - Go libraries
- Add resolved `$GOPATH` into global libraries

GoImports

GoImports is a tool that automatically manages your imports - removes unused, adds unimported in case it's resolvable and also formats your source.

In case you want to only format your code, use `GoFmt` instead.

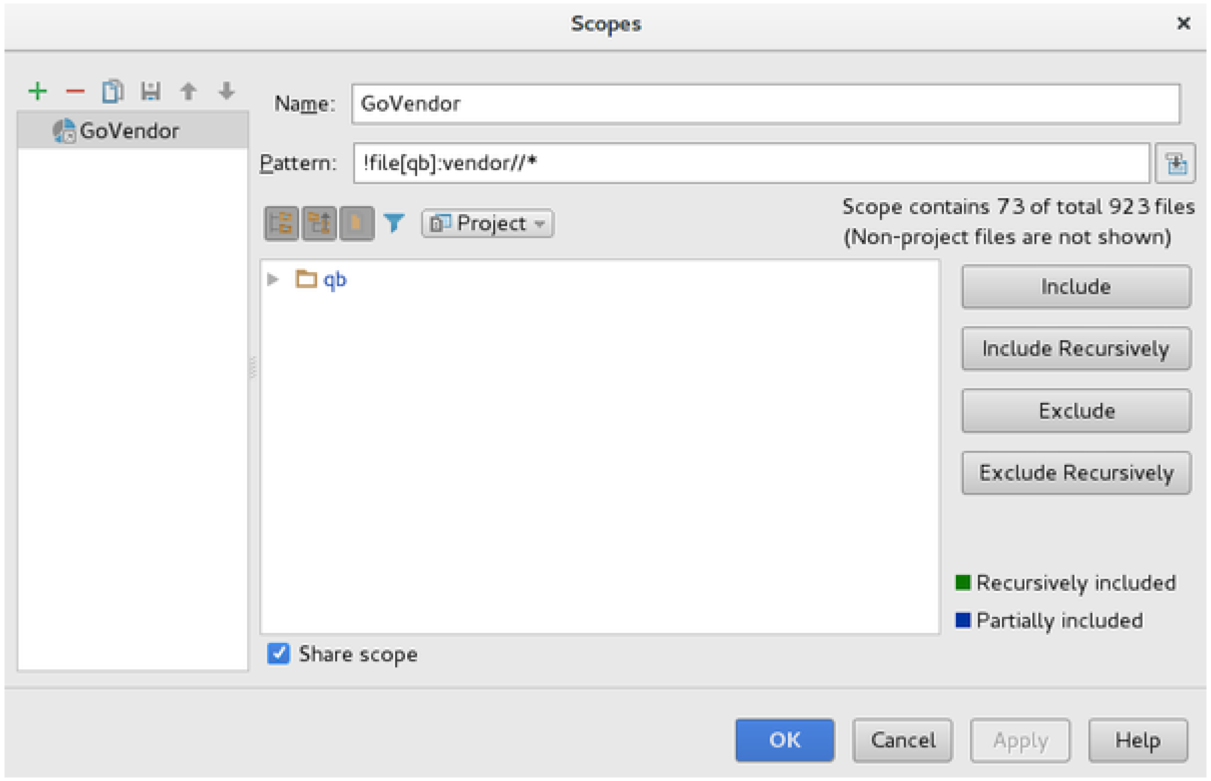
In terminal:

- Run `go get golang.org/x/tools/cmd/goimports`

In Intellij Idea:

- File - Settings - Tools - File Watchers - Green plus
- Fill the form
  - Name, Description: are up to you
  - Disable Immediate file synchronization*
  - File type: Go
  - Scope (skip if you're not using Vendor experiment):
    - Open custom scopes dialog by clicking three-dot button right to the selectbox

        - Create new custom scope called "GoVendor"
        - Put following as a pattern: !file[your-project-name]:vendor//*
        - Use newly created scope as a filewatcher scope, so it doesn't touch a vendor folder
    - Program: full resolved path to goimports (`$GOPATH/bin/goimports`)
    - Arguments: `-w $FilePath$`
    - Environment variables (skip if you're not using Vendor experiment): `GO15VENDOREXPERIMENT=1`
- Confirm (rest can be kept with default values)



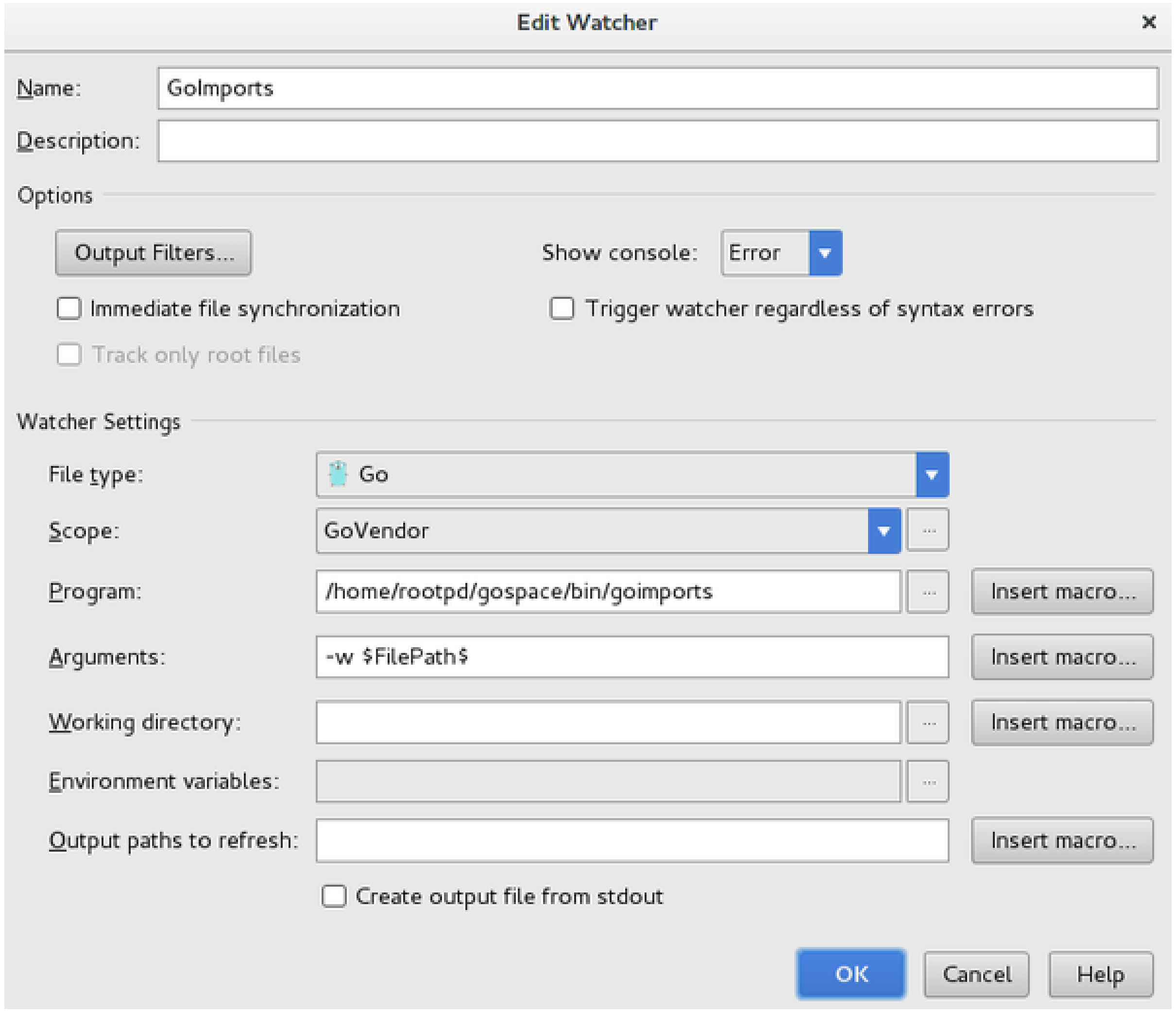
Name: GoVendor

Pattern: !file[qb]:vendor//*

Scope contains 73 of total 923 files
(Non-project files are not shown)

🟩 Recursively included
🟦 Partially included

☑ Share scope



Edit Watcher

Name: GoImports

Description:

Options

Show console: Error

☐ Immediate file synchronization

☐ Trigger watcher regardless of syntax errors

☐ Track only root files

Watcher Settings

File type: Go

Scope: GoVendor

Program: /home/rootpd/gospace/bin/goimports

Arguments: -w $FilePath$

Working directory:

Environment variables:

Output paths to refresh:

☐ Create output file from stdout

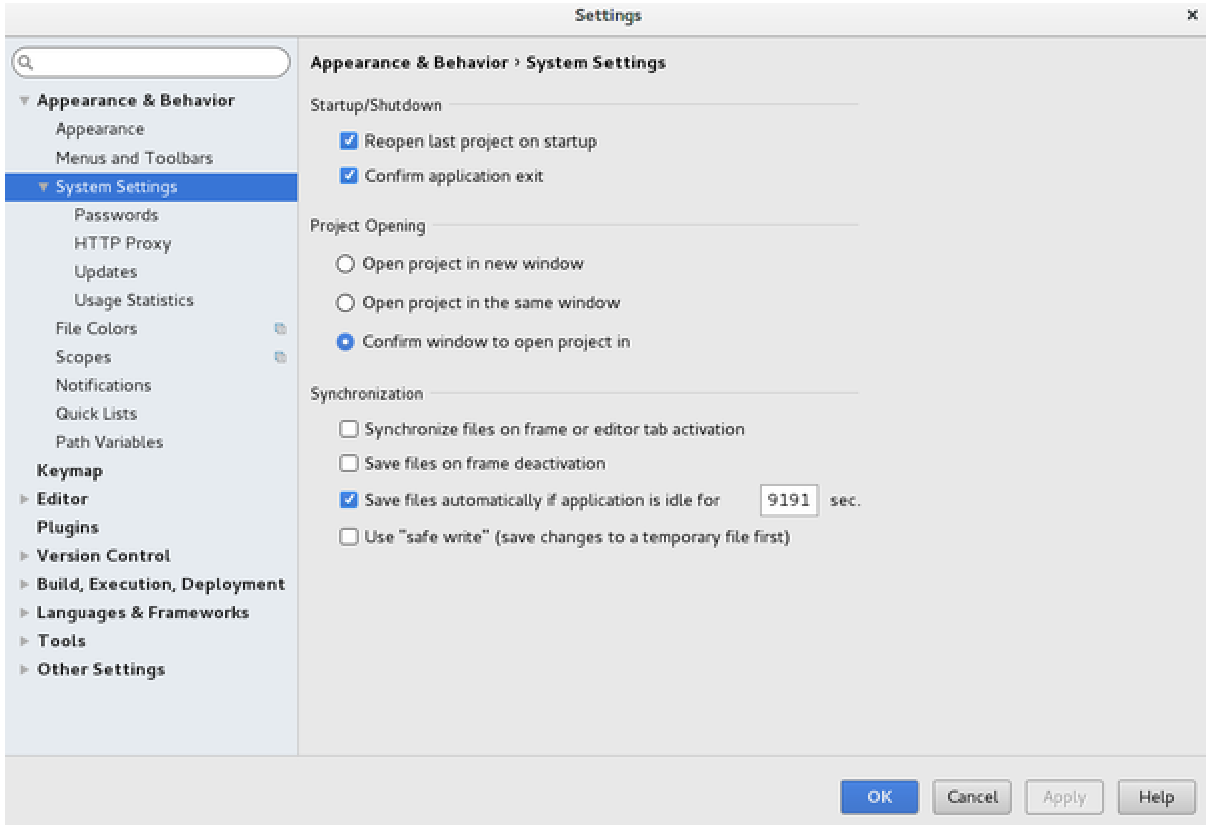
After every save, GoImports will get triggered and your code will be formated and altered as necessary.

We need to one more fix related to saving. The problem here is that if autosave was trigerred every second, your caret would move around randomly all the time when GoImports is run and you would go nuts very quickly.

> **Files are being altered by external tool and IDE would move the caret around every second because of autosave -> external change -> reload cycle. Disabling autosave is mandatory to be able to work with the GoImports.*

To fix it:

- File - Settings - Appearance & Behavior - System settings
- Edit synchronization section
  - Disable synchronize files on frame or editor tab activation
  - Enable save files on frame deactivation
  - Enable and put a big number in Save files automatically if application is idle for X seconds (disabling this will cause autosave being triggered again)
  - Disable safe write to temporary file



Appearance & Behavior › System Settings

Startup/Shutdown

- [✓] Reopen last project on startup
- [✓] Confirm application exit

Project Opening

- ○ Open project in new window
- ○ Open project in the same window
- ● Confirm window to open project in

Synchronization

- [ ] Synchronize files on frame or editor tab activation
- [ ] Save files on frame deactivation
- [✓] Save files automatically if application is idle for 9191 sec.
- [ ] Use "safe write" (save changes to a temporary file first)

GoLint

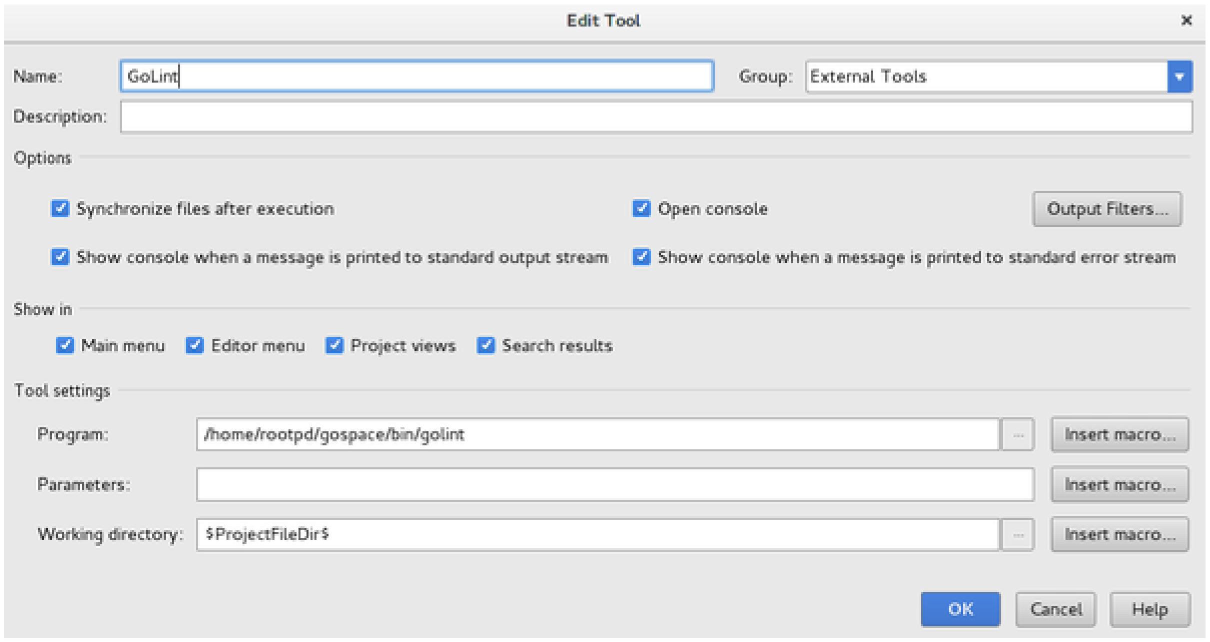
GoLint is a linter tool that will make you write and document your code properly. The sooner you start with it, the nicer code you'll produce.

In terminal:

- Run `go get github.com/golang/lint/golint`

In Intellij Idea:

- Run – Edit configurations – Defaults – Go application
- Before launch – Green plus – Run external tool – Green plus (just once, next time you'll reuse if necessary)
- Fill the form
  - Name, Description: are up to you
  - Group: external tools
  - Mark all 4 option checkboxes enabled: Synchronize files after execution, Open console, Sow console when a message is printed to stdout/stderr
  - Program: full resolved path to golint (`$GOPATH/bin/golint`)
  - Parameters: empty
  - Working directory: `$ProjectFileDir$`
- Confirm


Edit Tool

Name: GoLint
Group: External Tools
Description:

Options
- [✓] Synchronize files after execution
- [✓] Open console
- [✓] Show console when a message is printed to standard output stream
- [✓] Show console when a message is printed to standard error stream

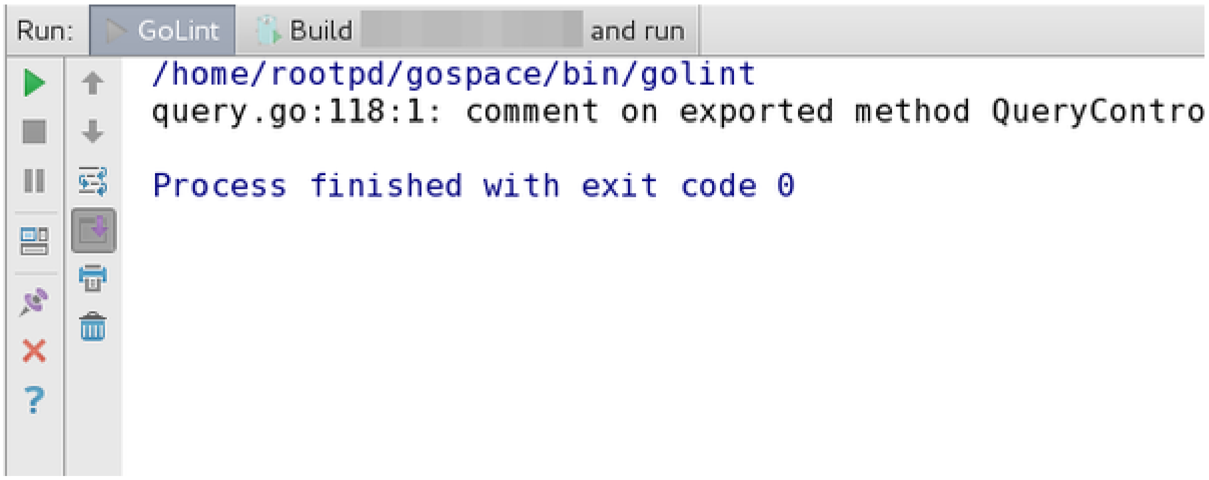
Show in
- [✓] Main menu
- [✓] Editor menu
- [✓] Project views
- [✓] Search results

Tool settings

Program: /home/rootpd/gospace/bin/golint
Parameters:
Working directory: $ProjectFileDir$

Unfortunately there's no way how to stop build if GoLint has any kind of objections. It's always returning exit code 0 – see the GitHub issue.

Linter will now be triggered with every build in separate tab.



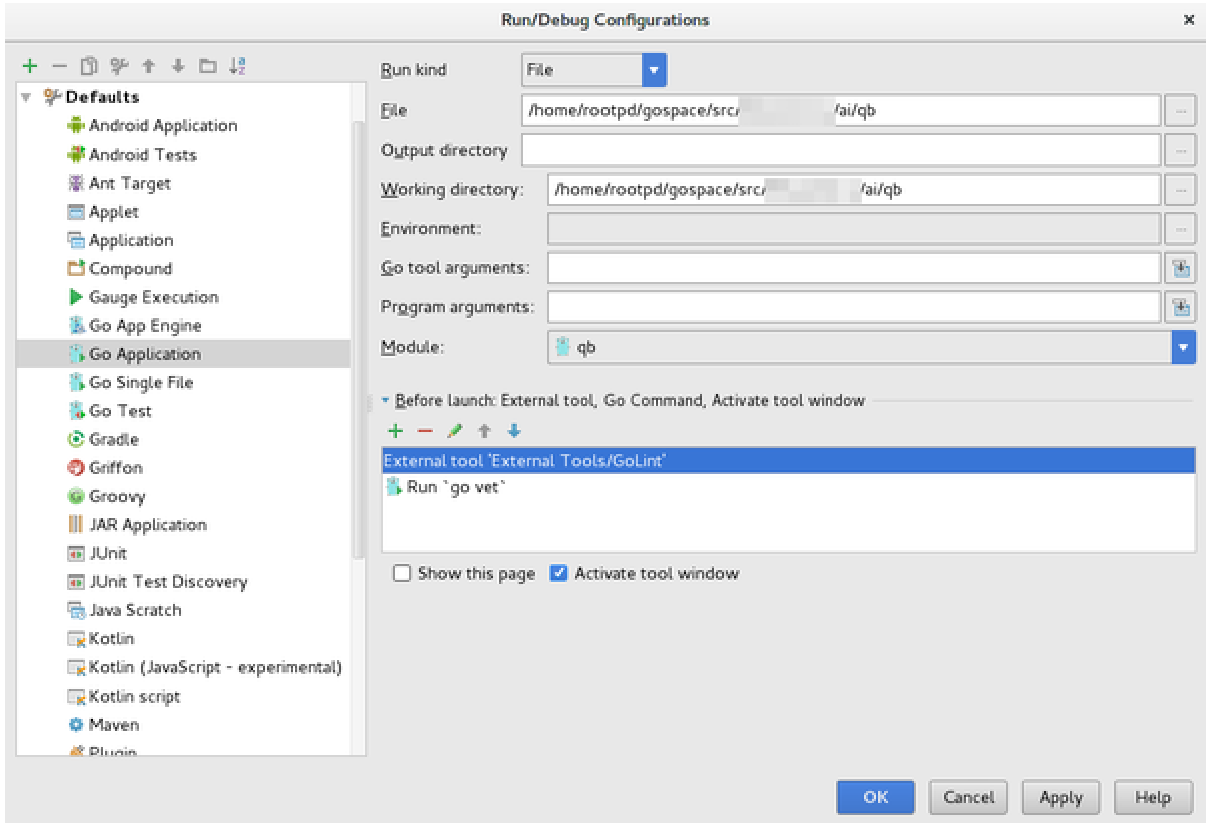
```
/home/rootpd/gospace/bin/golint
query.go:118:1: comment on exported method QueryContro

Process finished with exit code 0
```

Go Vet

Vet is a tool that tries to find logical issues in your source – e.g. checking if your `Sprintf()` string references the same amount of variables as you provided afterwards.

In Intellij Idea:

- Run – Edit configurations – Defaults – Go application
- Before launch – green plus – Go command – Type "vet"



Run kind: File

File: /home/rootpd/gospace/src/[redacted]/ai/qb

Output directory:

Working directory: /home/rootpd/gospace/src/[redacted]/ai/qb

Environment:

Go tool arguments:

Program arguments:

Module: qb

Before launch: External tool, Go Command, Activate tool window

External tool 'External Tools/GoLint'

Run `go vet`

☐ Show this page ☑ Activate tool window

In case go vet finds something, it will stop the build as it returns non-zero exit code.

If you already had some run/debug configurations created prior these settings, you'll have to add "before launch" commands into those configurations too. All new configurations will be created with `Go Vet` automatically.

Vendor experiment

To get more information about vendor experiment in Go 1.5, visit the official post. Vendoring is enabled by default since Go 1.6.

- Run - Edit configurations - Defaults - Go application
- Environment (three dots)
- Add variable with name `GO15VENDOREXPERIMENT` and value `1`

Conclusion

Because of the way how we set things up, all of the tools are also working from within IDE terminal which is nice. With the doc integration that IDEA provides by default (`CTRL+Q`), you now have really neat environment for developing in Go.

Intellij IDEA is also the only IDE I was able to use debugger in, which is a huge plus when you're trying to catch a bug. The only problem I found currently is that when you kill the app, it kills debugger but binary keeps still running - you have to manually kill the process of your app.

The plugin is still in heavy development and things might change in the future (vendoring is not fully supported, interface implementations are not being recognized), but it's on the way.