air Go live reload

CONCEPTS

> air watches your files, runs the build command on each change, then restarts the built binary. It reads .air.toml in the current directory and uses defaults without one.

File change watched ext/dir
↓ delay
build.cmd go build -o ./tmp/main .
↓ stop old, start
entrypoint your app
↓ proxy
Browser reloads

> Air's own flags come first; everything after -- goes to your program.

INSTALL

`go install github.com/air-verse/air@latest`
Install into GOPATH/bin

`export PATH=$PATH:$(go env GOPATH)/bin`
Fix 'command not found: air'

`go get -tool github.com/air-verse/air@latest`
Add as a project-local tool

RUN

`air init`
Create .air.toml with defaults

`air`
Run with .air.toml or defaults

`air -c <file>`
Use a specific config file

`air -d`
Debug mode, print all logs

`air -h`
List every flag

`air server --port 8080`
Run the binary with these args

RUN (cont.)

`air -c <file> -- -h`
Split air's flags from the app's

FLAG OVERRIDES

`air --build.cmd "go build -o ./tmp/main ./cmd/api"`
Set the build command

`air --build.entrypoint ./tmp/main`
Set the binary to run

[BUILD] SETTINGS

`cmd = "go build -o ./tmp/main ."`
Command that builds the app

`entrypoint = ["./tmp/main"]`
Executable plus default args

`include_ext = ["go", "tpl", "html"]`
Extensions to watch

`exclude_dir = ["tmp", "vendor"]`
Directories to ignore

`exclude_regex = ["_test\\.go"]`
Ignore paths by regex

TOP-LEVEL & MISC

`root = "."`
Working directory to watch

`tmp_dir = "tmp"`
Folder for build output and logs

`env_files = [".env"]`
Load .env files; later wins

BROWSER RELOAD & RULES

`[proxy] enabled = true`
Reload the browser after rebuilds

`proxy_port = 8090`
Port you open in the browser

`app_port = 8080`
Port your app listens on