

[Discover Packages](#) > [Standard library](#) > [net](#) > [http](#) > pprof 



pprof

package

standard library

Version: [go1.20.1](#) **Latest** | Published: Feb 14, 2023 | License: [BSD-3-Clause](#) | Imports: [18](#) |Imported by: [19,059](#)



Details

[Valid go.mod file](#)  [Redistributable license](#)  [Tagged version](#) [Stable version](#) [Learn more](#)

Repository

cs.opensource.google/go/go

Links

[Report a Vulnerability](#) Documentation 

<> Documentation

Overview

[Usage examples](#)

Package pprof serves via its HTTP server runtime profiling data in the format expected by the pprof visualization tool.

The package is typically only imported for the side effect of registering its HTTP handlers. The handled paths all begin with `/debug/pprof/`.

To use pprof, link this package into your program:

```
import _ "net/http/pprof"
```

If your application is not already running an http server, you need to start one. Add "net/http" and "log" to your imports and the following code to your main function:

```
go func() {  
    log.Println(http.ListenAndServe("localhost:6060", nil))  
}()
```

By default, all the profiles listed in [runtime/pprof.Profile](#) are available (via [Handler](#)), in addition to the [Cmdline](#), [Profile](#), [Symbol](#), and [Trace](#) profiles defined in this package. If you are not using `DefaultServeMux`, you will have to register handlers with the mux you are using.

Usage examples

Use the pprof tool to look at the heap profile:

```
go tool pprof http://localhost:6060/debug/pprof/heap
```

Or to look at a 30-second CPU profile:

```
go tool pprof http://localhost:6060/debug/pprof/profile?seconds=30
```

Or to look at the goroutine blocking profile, after calling `runtime.SetBlockProfileRate` in your program:

```
go tool pprof http://localhost:6060/debug/pprof/block
```

Or to look at the holders of contended mutexes, after calling `runtime.SetMutexProfileFraction` in your program:

```
go tool pprof http://localhost:6060/debug/pprof/mutex
```

The package also exports a handler that serves execution trace data for the "go tool trace" command. To collect a 5-second execution trace:

```
curl -o trace.out http://localhost:6060/debug/pprof/trace?seconds=5
go tool trace trace.out
```

To view all available profiles, open <http://localhost:6060/debug/pprof/> in your browser.

For a study of the facility in action, visit

```
https://blog.golang.org/2011/06/profiling-go-programs.html
```

Index

```
func Cmdline(w http.ResponseWriter, r *http.Request)
func Handler(name string) http.Handler
func Index(w http.ResponseWriter, r *http.Request)
func Profile(w http.ResponseWriter, r *http.Request)
func Symbol(w http.ResponseWriter, r *http.Request)
func Trace(w http.ResponseWriter, r *http.Request)
```

Constants

This section is empty.

Variables

This section is empty.

Functions

func Cmdline

```
func Cmdline(w http.ResponseWriter, r *http.Request)
```

Cmdline responds with the running program's command line, with arguments separated by NUL bytes. The package initialization registers it as `/debug/pprof/cmdline`.

func Handler

```
func Handler(name string) http.Handler
```

Handler returns an HTTP handler that serves the named profile. Available profiles can be found in [runtime/pprof.Profile](#).

func Index

```
func Index(w http.ResponseWriter, r *http.Request)
```

Index responds with the pprof-formatted profile named by the request. For example, `/debug/pprof/heap` serves the "heap" profile. Index responds to a request for `/debug/pprof/` with an HTML page listing the available profiles.

func Profile

```
func Profile(w http.ResponseWriter, r *http.Request)
```

Profile responds with the pprof-formatted cpu profile. Profiling lasts for duration specified in seconds GET parameter, or for 30 seconds if not specified. The package initialization registers it as `/debug/pprof/profile`.

func Symbol

```
func Symbol(w http.ResponseWriter, r *http.Request)
```

Symbol looks up the program counters listed in the request, responding with a table mapping program counters to function names. The package initialization registers it as `/debug/pprof/symbol`.

func Trace

added in go1.5

```
func Trace(w http.ResponseWriter, r *http.Request)
```

Trace responds with the execution trace in binary form. Tracing lasts for duration specified in seconds GET parameter, or for 1 second if not specified. The package initialization registers it as `/debug/pprof/trace`.

Types

This section is empty.

Source Files

[View all](#) 

[pprof.go](#)

Why Go

[Use Cases](#)

[Case Studies](#)

Get Started

[Playground](#)

[Tour](#)

[Stack Overflow](#)

[Help](#)

Packages

[Standard Library](#)

[About Go Packages](#)

About

[Download](#)

[Blog](#)

[Issue Tracker](#)

[Release Notes](#)

[Brand Guidelines](#)

[Code of Conduct](#)

Connect

[Twitter](#)

[GitHub](#)

[Slack](#)

[r/golang](#)

[Meetup](#)

[Golang Weekly](#)

[Copyright](#)

[Terms of Service](#)

[Privacy Policy](#)

[Report an Issue](#)



Google