httptest

package | standard library

Version: go1.20.1 Latest | Published: Feb 14, 2023 | License: BSD-3-Clause | Imports: 18 | Imported by: 9,648

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

Package httptest provides utilities for HTTP testing.

Index

func (s *Server) StartTLS()

Examples

NewTLSServer
ResponseRecorder
Server
Server (HTTP2)

Constants

View Source

```
const DefaultRemoteAddr = "1.2.3.4"
```

DefaultRemoteAddr is the default remote address to return in RemoteAddr if an explicit DefaultRemoteAddr isn't set on ResponseRecorder.

Variables

This section is empty.

Functions

func NewRequest

added in go1.7

```
func NewRequest(method, target string, body io.Reader) *http.Request
```

NewRequest returns a new incoming server Request, suitable for passing to an http.Handler for testing.

The target is the RFC 7230 "request-target": it may be either a path or an absolute URL. If target is an absolute URL, the host name from the URL is used. Otherwise, "example.com" is used.

The TLS field is set to a non-nil dummy value if target has scheme "https".

The Request.Proto is always HTTP/1.1.

An empty method means "GET".

The provided body may be nil. If the body is of type *bytes.Reader, *strings.Reader, or *bytes.Buffer, the Request.ContentLength is set.

NewRequest panics on error for ease of use in testing, where a panic is acceptable.

To generate a client HTTP request instead of a server request, see the NewRequest function in the net/http package.

Types

type ResponseRecorder

```
type ResponseRecorder struct {
    // Code is the HTTP response code set by WriteHeader.
    //
    // Note that if a Handler never calls WriteHeader or Write,
    // this might end up being 0, rather than the implicit
    // http.StatusOK. To get the implicit value, use the Result
    // method.
    Code int

    // HeaderMap contains the headers explicitly set by the Handler.
    // It is an internal detail.
    //
    // Deprecated: HeaderMap exists for historical compatibility
    // and should not be used. To access the headers returned by a handler,
    // use the Response.Header map as returned by the Result method.
    HeaderMap http.Header

    // Body is the buffer to which the Handler's Write calls are sent.
    // If nil, the Writes are silently discarded.
    Body *bytes.Buffer

    // Flushed is whether the Handler called Flush.
    Flushed bool
    // contains filtered or unexported fields
}
```

ResponseRecorder is an implementation of http.ResponseWriter that records its mutations for later inspection in tests.

▶ Example

func NewRecorder

```
func NewRecorder() *ResponseRecorder
```

NewRecorder returns an initialized ResponseRecorder.

func (*ResponseRecorder) Flush

```
func (rw *ResponseRecorder) Flush()
```

Flush implements http.Flusher. To test whether Flush was called, see rw.Flushed.

func (*ResponseRecorder) Header

```
func (rw *ResponseRecorder) Header() http.Header
```

Header implements http.ResponseWriter. It returns the response headers to mutate within a handler. To test the headers that were written after a handler completes, use the Result method and see the returned

Response value's Header.

func (*ResponseRecorder) Result

added in go1.7

```
func (rw *ResponseRecorder) Result() *http.Response
```

Result returns the response generated by the handler.

The returned Response will have at least its StatusCode, Header, Body, and optionally Trailer populated. More fields may be populated in the future, so callers should not DeepEqual the result in tests.

The Response.Header is a snapshot of the headers at the time of the first write call, or at the time of this call, if the handler never did a write.

The Response.Body is guaranteed to be non-nil and Body.Read call is guaranteed to not return any error other than io.EOF.

Result must only be called after the handler has finished running.

func (*ResponseRecorder) Write

```
func (rw *ResponseRecorder) Write(buf []byte) (int, error)
```

Write implements http.ResponseWriter. The data in buf is written to rw.Body, if not nil.

func (*ResponseRecorder) WriteHeader

```
func (rw *ResponseRecorder) WriteHeader(code int)
```

WriteHeader implements http.ResponseWriter.

func (*ResponseRecorder) WriteString

added in go1.6

```
func (rw *ResponseRecorder) WriteString(str string) (int, error)
```

WriteString implements io.StringWriter. The data in str is written to rw.Body, if not nil.

type Server

```
type Server struct {
    URL      string // base URL of form http://ipaddr:port with no trailing slash
    Listener net.Listener

    // EnableHTTP2 controls whether HTTP/2 is enabled
    // on the server. It must be set between calling
    // NewUnstartedServer and calling Server.StartTLS.
    EnableHTTP2 bool

    // TLS is the optional TLS configuration, populated with a new config
    // after TLS is started. If set on an unstarted server before StartTLS
```

```
    // is called, existing fields are copied into the new config.
    TLS *tls.Config

    // Config may be changed after calling NewUnstartedServer and
    // before Start or StartTLS.
    Config *http.Server
    // contains filtered or unexported fields
}
```

A Server is an HTTP server listening on a system-chosen port on the local loopback interface, for use in end-to-end HTTP tests.

▶ Example

▶ Example (HTTP2)

func NewServer

```
func NewServer(handler http.Handler) *Server
```

NewServer starts and returns a new Server. The caller should call Close when finished, to shut it down.

func NewTLSServer

```
func NewTLSServer(handler http.Handler) *Server
```

NewTLSServer starts and returns a new Server using TLS. The caller should call Close when finished, to shut it down.

▶ Example

func NewUnstartedServer

```
func NewUnstartedServer(handler http.Handler) *Server
```

NewUnstartedServer returns a new Server but doesn't start it.

After changing its configuration, the caller should call Start or StartTLS.

The caller should call Close when finished, to shut it down.

func (*Server) Certificate

added in go1.9

```
func (s *Server) Certificate() *x509.Certificate
```

Certificate returns the certificate used by the server, or nil if the server doesn't use TLS.

func (*Server) Client

added in go1.9

```
func (s *Server) Client() *http.Client
```

Client returns an HTTP client configured for making requests to the server. It is configured to trust the server's TLS test certificate and will close its idle connections on Server.Close.

func (*Server) Close

```
func (s *Server) Close()
```

Close shuts down the server and blocks until all outstanding requests on this server have completed.

func (*Server) CloseClientConnections

```
func (s *Server) CloseClientConnections()
```

CloseClientConnections closes any open HTTP connections to the test Server.

func (*Server) Start

```
func (s *Server) Start()
```

Start starts a server from NewUnstartedServer.

func (*Server) StartTLS

```
func (s *Server) StartTLS()
```

StartTLS starts TLS on a server from NewUnstartedServer.

Source Files

View all

httptest.go recorder.go server.go

Connect

Twitter

GitHub

Slack

r/golang

Meetup

Golang Weekly

Report an Issue

Google