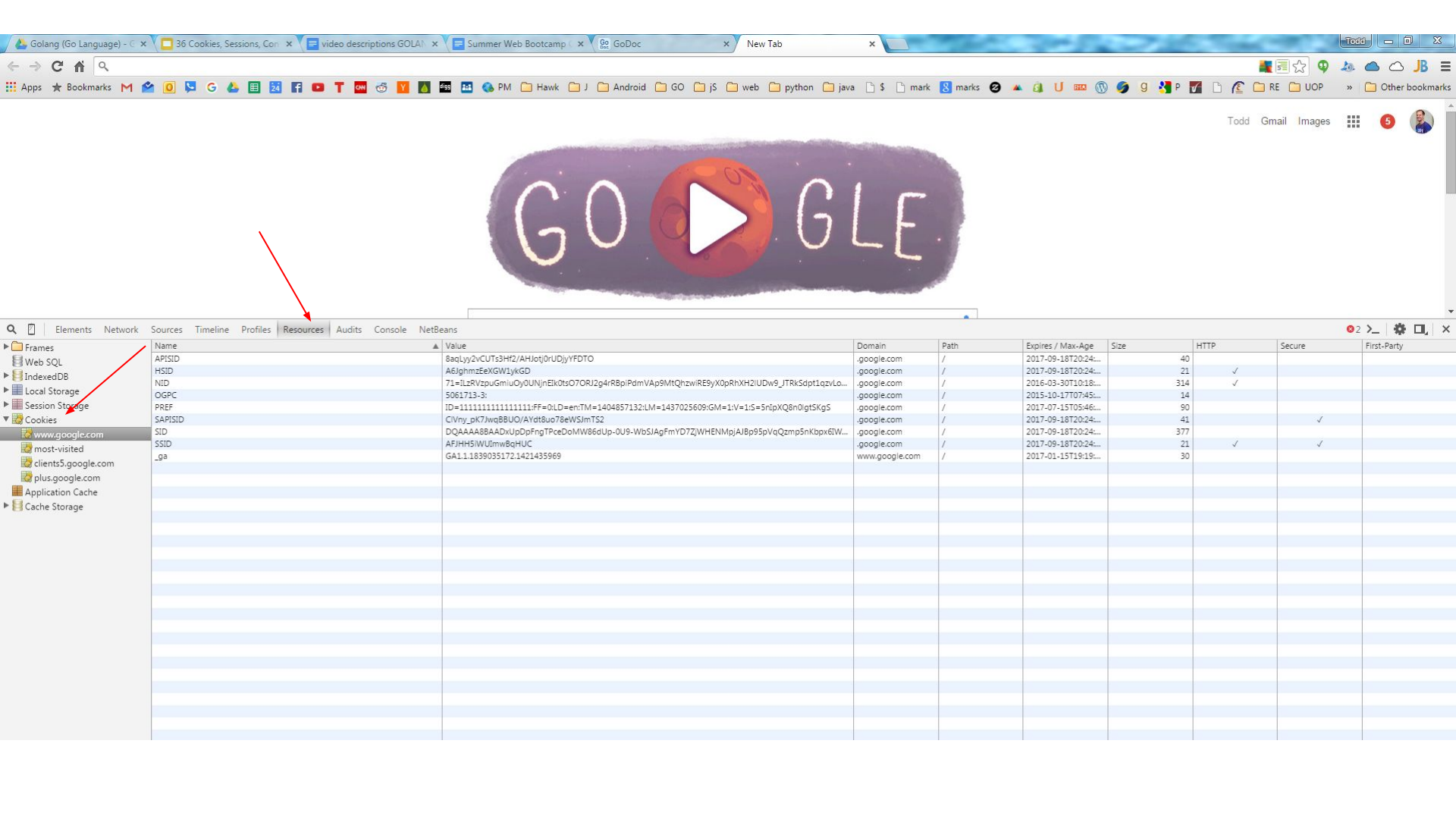
Cookies, Sessions, Context

Cookie

- a small amount of data that we can associate with a user
- every time the user comes back to our site
  - that cookie is persistent
  - it stays between repeat visits to our site
  - “a database stored on the user’s computer that’s passed with every request”
- cookies are a tool for us as web developers
  - we can use cookies to know if we’ve seen the same user before
  - makes it possible for us to implement logins / sessions
  - cookies are the most common way to implement sessions
    - in Europe you have to notify your user that you’re using cookies
- you can see cookies in your chrome browser
  - under dev tools under resources


Golang (Go Language) -
36 Cookies, Sessions, Cor
video descriptions GOLAN
Summer Web Bootcamp
GoDoc
New Tab
Todd
Apps
Bookmarks
PM
Hawk
J
Android
GO
jS
web
python
java
$
mark
marks
RE
UOP
Other bookmarks
Todd
Gmail
Images
5
GO GLE
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
NetBeans
Frames
Web SQL
IndexedDB
Local Storage
Session Storage
Cookies
www.google.com
most-visited
clients5.google.com
plus.google.com
Application Cache
Cache Storage
Name
Value
Domain
Path
Expires / Max-Age
Size
HTTP
Secure
First-Party
APISID
8aqLyy2vCUTs3Hf2/AHJotj0rUDjyYFDTO
.google.com
/
2017-09-18T20:24:...
40
HSID
A6JghmzEeXGW1ykGD
.google.com
/
2017-09-18T20:24:...
21
✓
NID
71=iLzRVzpuGmiuOy0UNjnEIk0tsO7ORJ2g4rRBpiPdmVAp9MtQhzwiRE9yX0pRhXH2IUDw9_JTRkSdpt1qzvLo...
.google.com
/
2016-03-30T10:18:...
314
✓
OGPC
5061713-3:
.google.com
/
2015-10-17T07:45:...
14
PREF
ID=1111111111111111:FF=0:LD=en:TM=1404857132:LM=1437025609:GM=1:V=1:S=5nIpXQ8n0lgtSKgS
.google.com
/
2017-07-15T05:46:...
90
SAPISID
CiVny_pK7JwqBBUO/AYdt8uo78eWSJmTS2
.google.com
/
2017-09-18T20:24:...
41
✓
SID
DQAAAA8BAADxUpDpFngTPceDoMW86dUp-0U9-WbSJAgFmYD7ZjWHENMpjAJBp95pVqQzmp5nKbpx6lW...
.google.com
/
2017-09-18T20:24:...
377
SSID
AFJHH5iWU0mwBqHUC
.google.com
/
2017-09-18T20:24:...
21
✓
✓
_ga
GA1.1.1839035172.1421435969
www.google.com
/
2017-01-15T19:19:...
30

Cookie

- every time a user goes to a domain
  - if there is a cookie from that domain on the user’s computer
  - that cookie is sent along with the user’s request to that server
  - the server can use this information to know the user’s identity

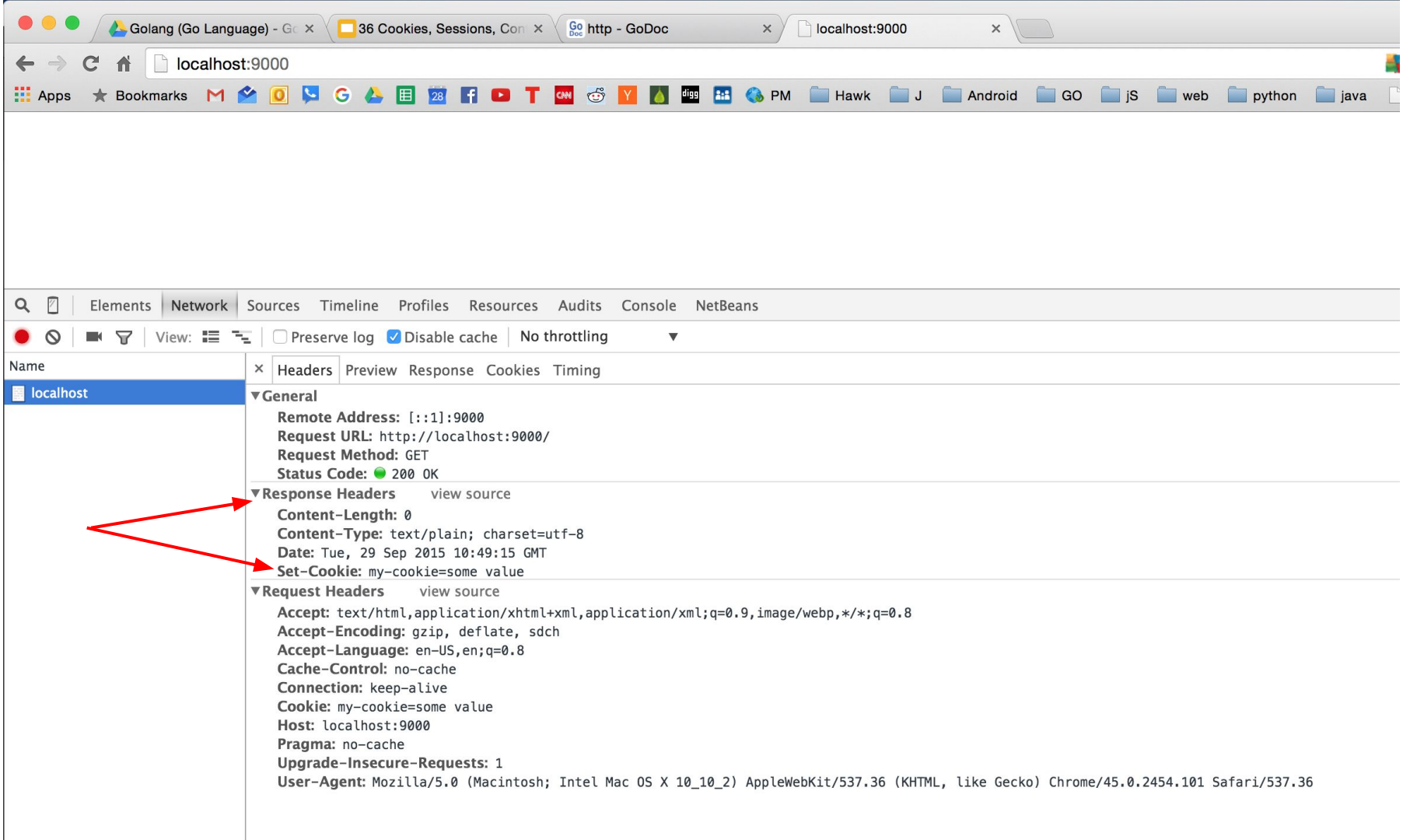
set cookie

func SetCookie

```
func SetCookie(w ResponseWriter, cookie *Cookie)
```

SetCookie adds a Set-Cookie header to the provided ResponseWriter's headers. The provided cookie must have a valid Name. Invalid cookies may be silently dropped.

a cookie is a header


Golang (Go Language) - Go
36 Cookies, Sessions, Con
http - GoDoc
localhost:9000
localhost:9000
Apps
Bookmarks
PM
Hawk
J
Android
GO
jS
web
python
java
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
NetBeans
View:
Preserve log
Disable cache
No throttling
Name
localhost
Headers
Preview
Response
Cookies
Timing
General
Remote Address: [::1]:9000
Request URL: http://localhost:9000/
Request Method: GET
Status Code: 200 OK
Response Headers
view source
Content-Length: 0
Content-Type: text/plain; charset=utf-8
Date: Tue, 29 Sep 2015 10:49:15 GMT
Set-Cookie: my-cookie=some value
Request Headers
view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cache-Control: no-cache
Connection: keep-alive
Cookie: my-cookie=some value
Host: localhost:9000
Pragma: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36

type Cookie

```
type Cookie struct {
    Name  string
    Value string

    Path       string    // optional
    Domain     string    // optional
    Expires    time.Time // optional
    RawExpires string    // for reading cookies only

    // MaxAge=0 means no 'Max-Age' attribute specified.
    // MaxAge<0 means delete cookie now, equivalently 'Max-Age: 0'
    // MaxAge>0 means Max-Age attribute present and given in seconds
    MaxAge   int
    Secure   bool
    HttpOnly bool
    Raw      string
    Unparsed []string // Raw text of unparsed attribute-value pairs
}
```

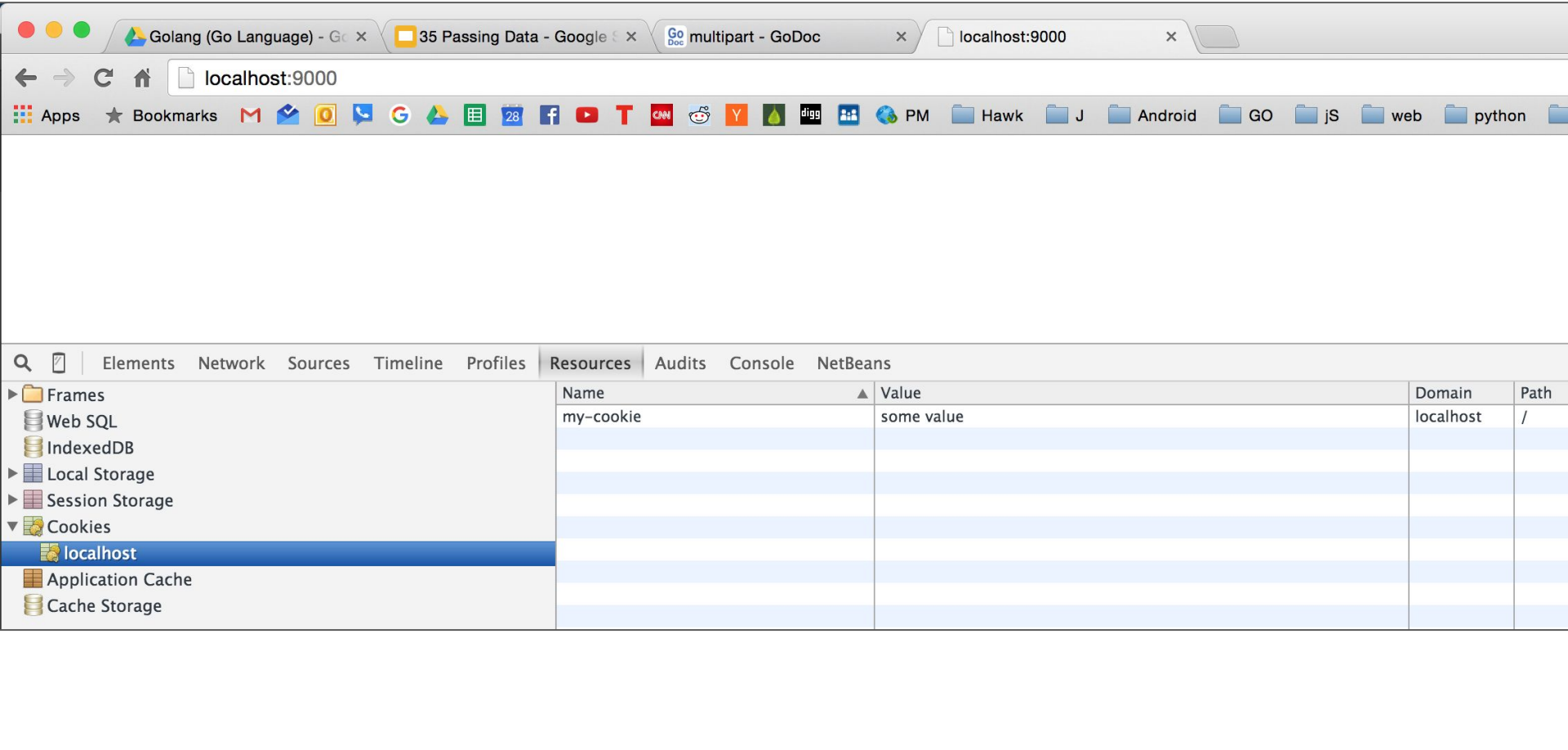
A Cookie represents an HTTP cookie as sent in the Set-Cookie header of an HTTP response or the Cookie header of an HTTP request.

See http://tools.ietf.org/html/rfc6265 for details.

main.go

```go
package main

import (
    "net/http"
)

func main() {
    http.HandleFunc("/", func(res http.ResponseWriter, req *http.Request) {

        http.SetCookie(res, &http.Cookie{
            Name:  "my-cookie",
            Value: "some value",
        })
    })
    http.ListenAndServe(":9000", nil)
}
```

SET COOKIE


Golang (Go Language) - Gc
35 Passing Data - Google S
multipart - GoDoc
localhost:9000
localhost:9000
Apps
Bookmarks
PM
Hawk
J
Android
GO
jS
web
python
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
NetBeans
Frames
Web SQL
IndexedDB
Local Storage
Session Storage
Cookies
localhost
Application Cache
Cache Storage
Name
Value
Domain
Path
my-cookie
some value
localhost
/

get cookie

https://godoc.org/net/http

okmarks PM

- func (err *ProtocolError) Error() string

type Request

- func NewRequest(method, urlStr string, body io.Reader) (*Request, error)
- func ReadRequest(b *bufio.Reader) (req *Request, err error)
- func (r *Request) AddCookie(c *Cookie)
- func (r *Request) BasicAuth() (username, password string, ok bool)
- func (r *Request) Cookie(name string) (*Cookie, error)
- func (r *Request) Cookies() []*Cookie
- func (r *Request) FormFile(key string) (multipart.File, *multipart.FileHeader, error)
- func (r *Request) FormValue(key string) string
- func (r *Request) MultipartReader() (*multipart.Reader, error)
- func (r *Request) ParseForm() error
- func (r *Request) ParseMultipartForm(maxMemory int64) error
- func (r *Request) PostFormValue(key string) string
- func (r *Request) ProtoAtLeast(major, minor int) bool
- func (r *Request) Referer() string
- func (r *Request) SetBasicAuth(username, password string)
- func (r *Request) UserAgent() string
- func (r *Request) Write(w io.Writer) error
- func (r *Request) WriteProxy(w io.Writer) error

func (*Request) Cookie

```
func (r *Request) Cookie(name string) (*Cookie, error)
```

Cookie returns the named cookie provided in the request or ErrNoCookie if not found.

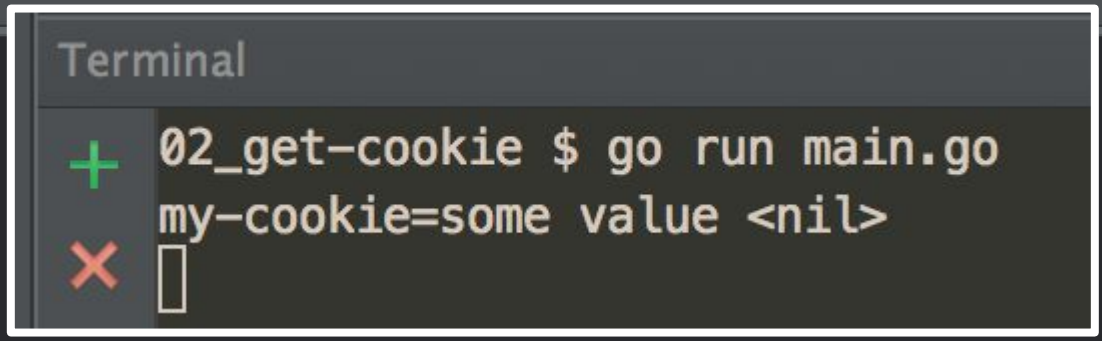
func (*Request) Cookies

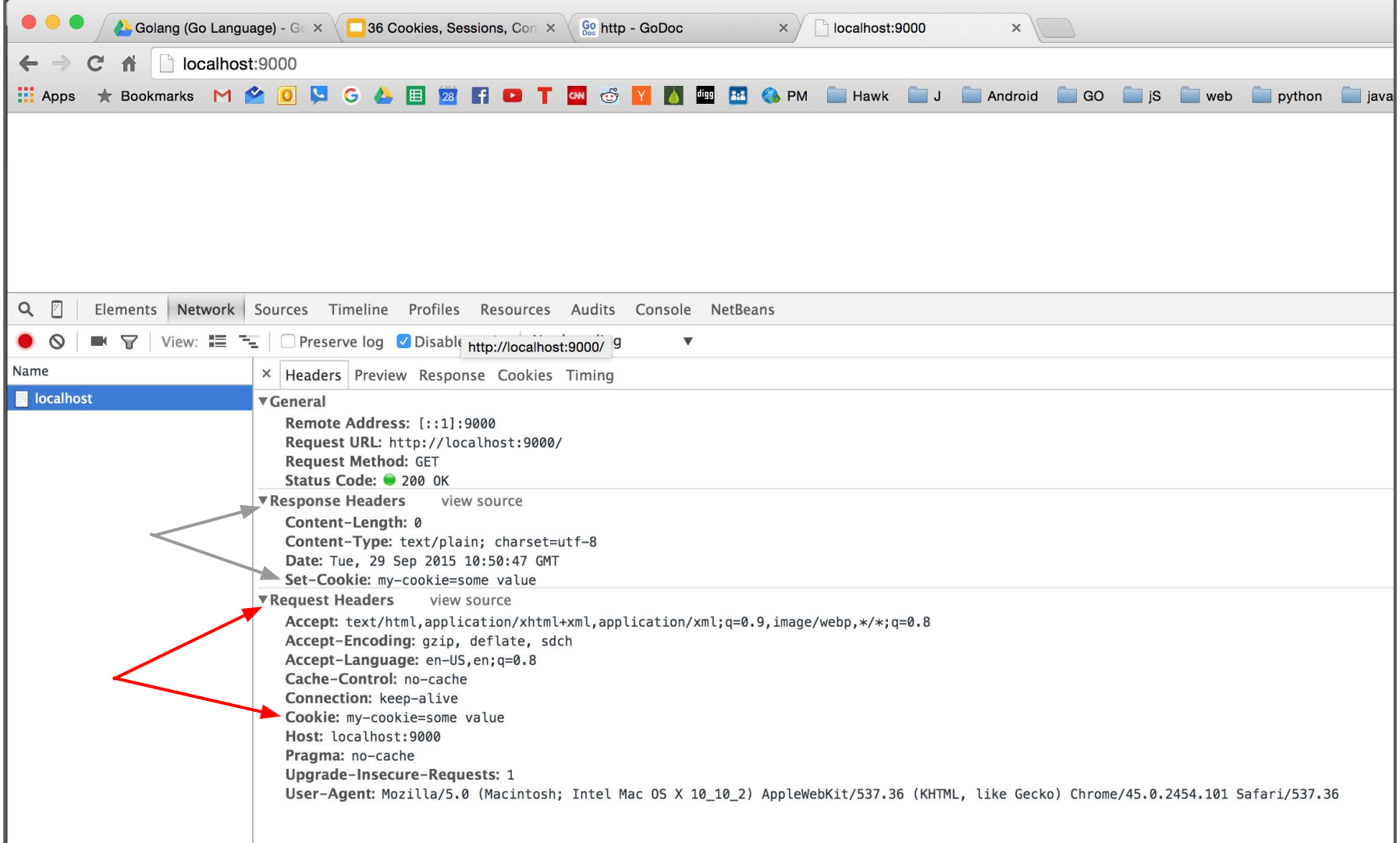
```
func (r *Request) Cookies() []*Cookie
```

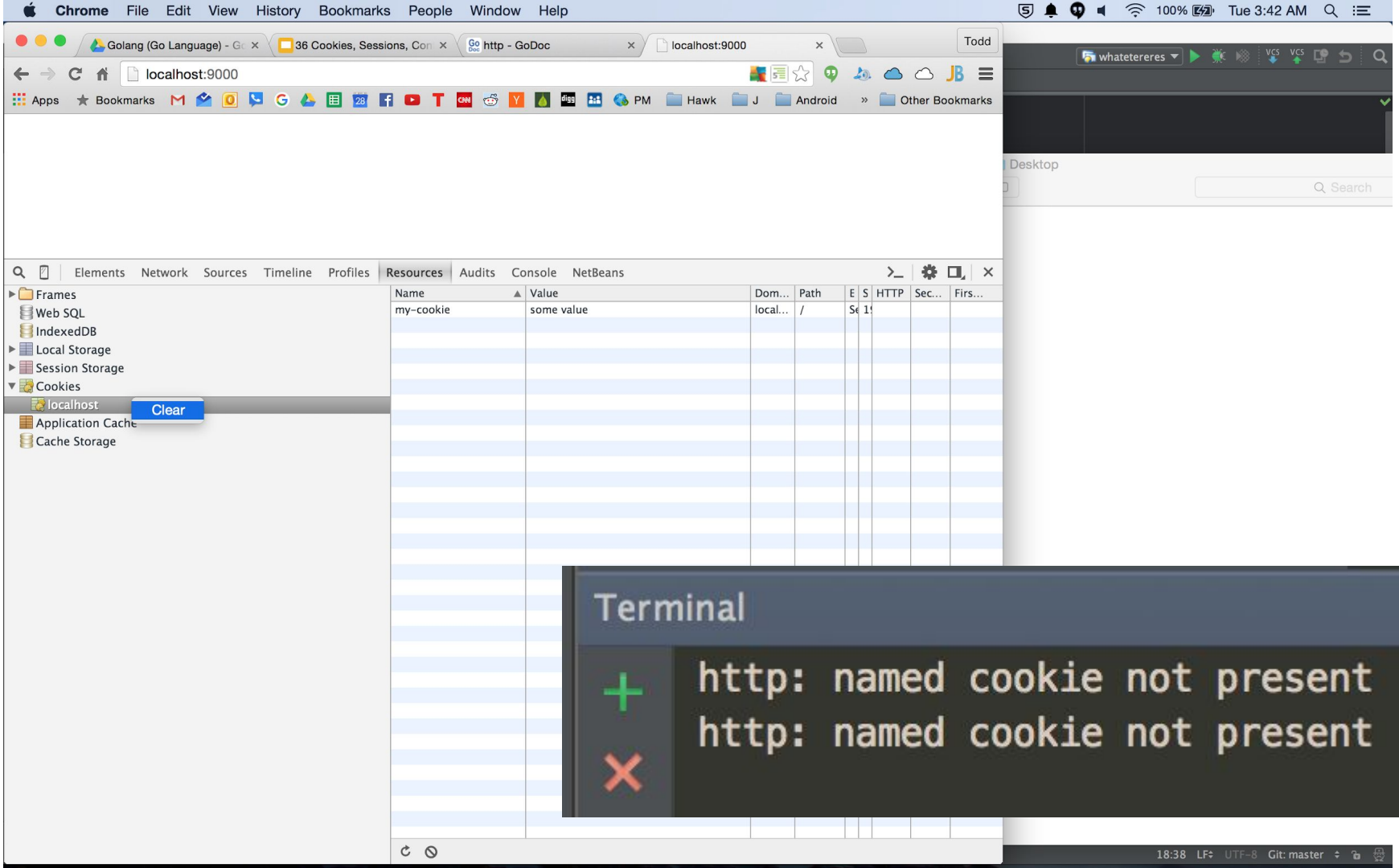
Cookies parses and returns the HTTP cookies sent with the request.

main.go

```go
package main

import (
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/", func(res http.ResponseWriter, req *http.Request) {
        cookie, err := req.Cookie("my-cookie")
        fmt.Println(cookie, err)

        http.SetCookie(res, &http.Cookie{
            Name:  "my-cookie",
            Value: "some value",
        })
    })
    http.ListenAndServe(":9000", nil)
}
```




Golang (Go Language) - Go
36 Cookies, Sessions, Con
http - GoDoc
localhost:9000
localhost:9000
Apps
Bookmarks
PM
Hawk
J
Android
GO
jS
web
python
java
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
NetBeans
View:
Preserve log
Disable
http://localhost:9000/
Name
localhost
Headers
Preview
Response
Cookies
Timing
General
Remote Address: [::1]:9000
Request URL: http://localhost:9000/
Request Method: GET
Status Code: 200 OK
Response Headers
view source
Content-Length: 0
Content-Type: text/plain; charset=utf-8
Date: Tue, 29 Sep 2015 10:50:47 GMT
Set-Cookie: my-cookie=some value
Request Headers
view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cache-Control: no-cache
Connection: keep-alive
Cookie: my-cookie=some value
Host: localhost:9000
Pragma: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36


Chrome File Edit View History Bookmarks People Window Help
Golang (Go Language) - G
36 Cookies, Sessions, Cor
http - GoDoc
localhost:9000
Todd
localhost:9000
Apps Bookmarks PM Hawk J Android Other Bookmarks
Elements Network Sources Timeline Profiles Resources Audits Console NetBeans
Frames
Web SQL
IndexedDB
Local Storage
Session Storage
Cookies
localhost
Clear
Application Cache
Cache Storage
Name Value Dom... Path E S HTTP Sec... Firs...
my-cookie some value local... /
Terminal
http: named cookie not present
http: named cookie not present
whatetereres
Desktop
Search
18:38 LF UTF-8 Git: master

sessions

creating a UUID & setting it in a cookie

Sessions

- a user logs in
  - until they log out, that's their session
  - sessions create "state"
    - we want to know a user is logged in for each of their requests
- UUID
  - universally unique ID
  - find a package
    - go-search.org
    - godoc.org

https://godoc.org/github.com/nu7hatch/gouuid

okmarks

Index

type UUID

```
type UUID [16]byte
```

A UUID representation compliant with specification in RFC 4122 document.

func NewV3

```
func NewV3(ns *UUID, name []byte) (u *UUID, err error)
```

Generate a UUID based on the MD5 hash of a namespace identifier and a name.

func NewV4

```
func NewV4() (u *UUID, err error)
```

Generate a random UUID.

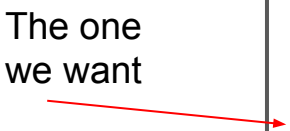
Example

func NewV5

```
func NewV5(ns *UUID, name []byte) (u *UUID, err error)
```

Generate a UUID based on the SHA-1 hash of a namespace identifier and a name.

Example



func NewV4

```
func NewV4() (u *UUID, err error)
```

Generate a random UUID.

Example

Code:

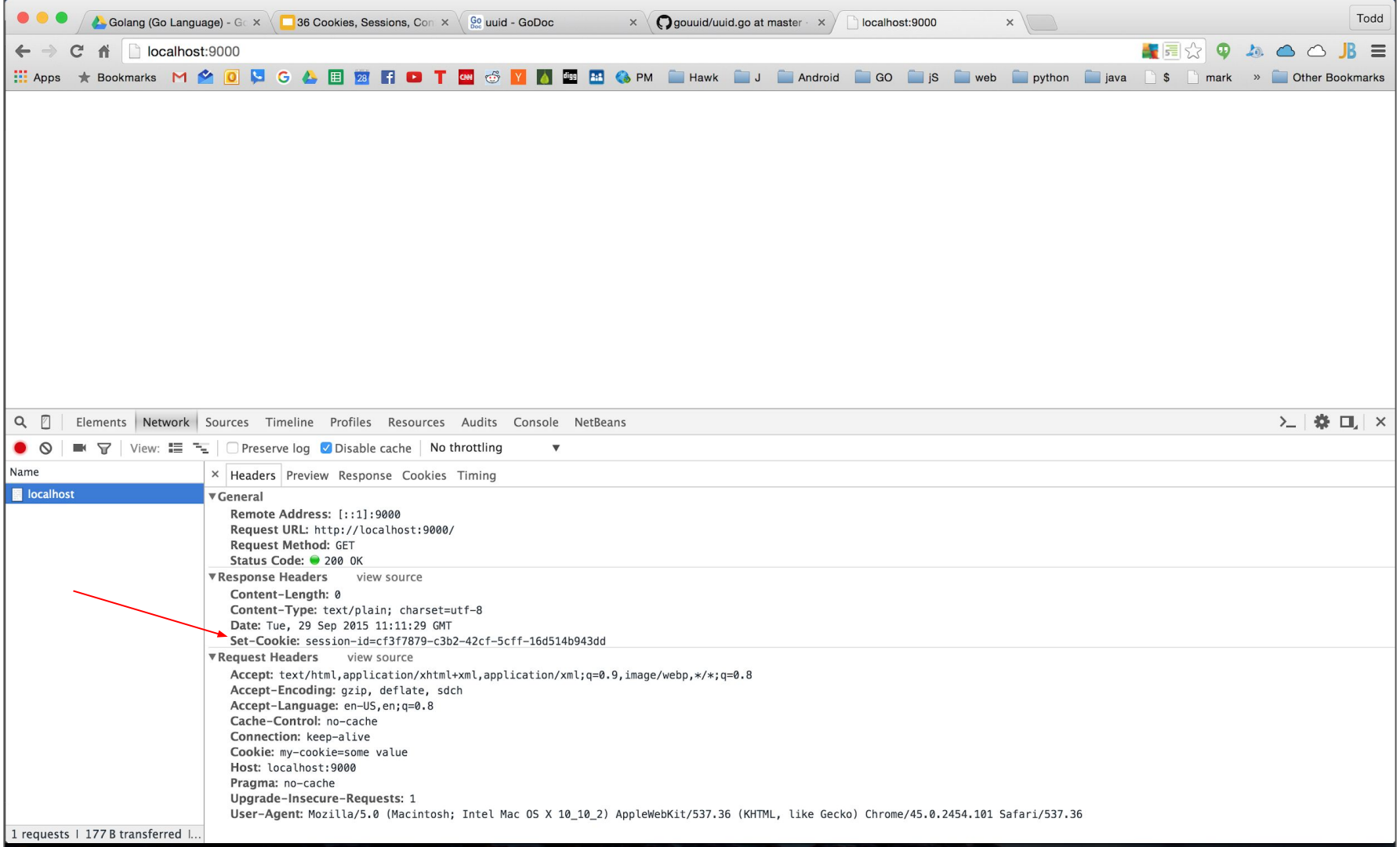
```
u4, err := uuid.NewV4()
if err != nil {
    fmt.Println("error:", err)
    return
}
fmt.Println(u4)
```

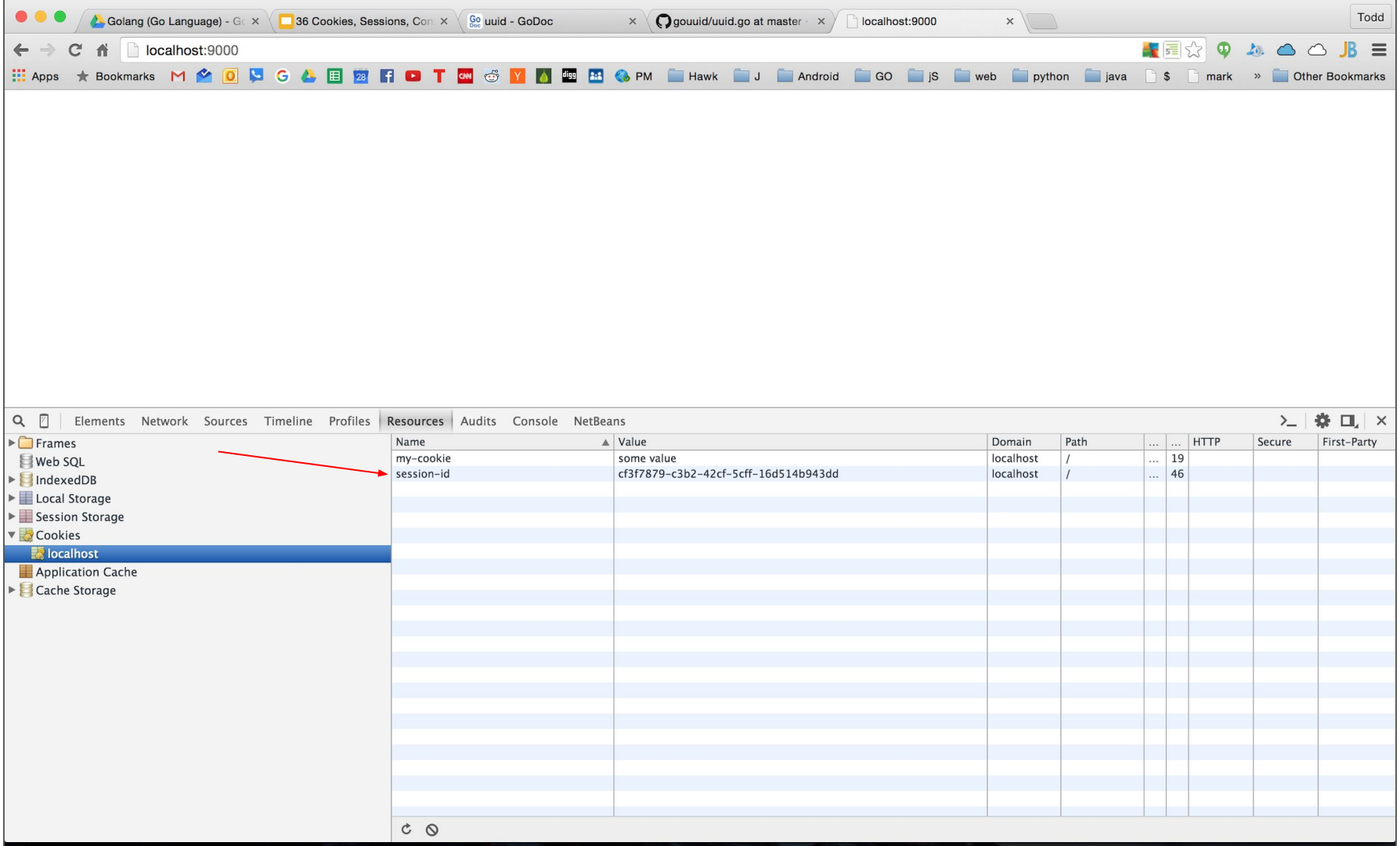
main.go

```go
package main

import (
    "fmt"
    "github.com/nu7hatch/gouuid"
    "net/http"
)

func main() {

    http.HandleFunc("/", func(res http.ResponseWriter, req *http.Request) {
        cookie, err := req.Cookie("session-id")
        // cookie is not set
        if err != nil {
            id, _ := uuid.NewV4()
            cookie = &http.Cookie{
                Name:  "session-id",
                Value: id.String(),
            }
            http.SetCookie(res, cookie)
        }
        fmt.Println(cookie)

    })
    http.ListenAndServe(":9000", nil)
}
```

string() 1 of 1

```
140             u[8] = (u[8] | ReservedMicrosoft) & 0x3F
141         }
142 }
143
144 // Variant returns the UUID Variant, which determines the internal
145 // layout of the UUID. This will be one of the constants: RESERVED_NCS,
146 // RFC_4122, RESERVED_MICROSOFT, RESERVED_FUTURE.
147 func (u *UUID) Variant() byte {
148         if u[8]&ReservedNCS == ReservedNCS {
149                 return ReservedNCS
150         } else if u[8]&ReservedRFC4122 == ReservedRFC4122 {
151                 return ReservedRFC4122
152         } else if u[8]&ReservedMicrosoft == ReservedMicrosoft {
153                 return ReservedMicrosoft
154         }
155         return ReservedFuture
156 }
157
158 // Set the four most significant bits (bits 12 through 15) of the
159 // time_hi_and_version field to the 4-bit version number.
160 func (u *UUID) setVersion(v byte) {
161         u[6] = (u[6] & 0xF) | (v << 4)
162 }
163
164 // Version returns a version number of the algorithm used to
165 // generate the UUID sequence.
166 func (u *UUID) Version() uint {
167         return uint(u[6] >> 4)
168 }
169
170 // Returns unparsed version of the generated UUID sequence.
171 func (u *UUID) String() string {
172         return fmt.Sprintf("%x-%x-%x-%x-%x", u[0:4], u[4:6], u[6:8], u[8:10], u[10:])
173 }
```


Golang (Go Language) - Go
36 Cookies, Sessions, Con
uuid - GoDoc
gouuid/uuid.go at master
localhost:9000
Todd
localhost:9000
Apps
Bookmarks
PM
Hawk
J
Android
GO
jS
web
python
java
$
mark
Other Bookmarks
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
NetBeans
View:
Preserve log
Disable cache
No throttling
Name
localhost
Headers
Preview
Response
Cookies
Timing
General
Remote Address: [::1]:9000
Request URL: http://localhost:9000/
Request Method: GET
Status Code: 200 OK
Response Headers
view source
Content-Length: 0
Content-Type: text/plain; charset=utf-8
Date: Tue, 29 Sep 2015 11:11:29 GMT
Set-Cookie: session-id=cf3f7879-c3b2-42cf-5cff-16d514b943dd
Request Headers
view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cache-Control: no-cache
Connection: keep-alive
Cookie: my-cookie=some value
Host: localhost:9000
Pragma: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36
1 requests | 177 B transferred


Golang (Go Language) - G
36 Cookies, Sessions, Con
uuid - GoDoc
gouuid/uuid.go at master
localhost:9000
Todd
localhost:9000
Apps
Bookmarks
PM
Hawk
J
Android
GO
jS
web
python
java
$
mark
Other Bookmarks
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
NetBeans
Frames
Web SQL
IndexedDB
Local Storage
Session Storage
Cookies
localhost
Application Cache
Cache Storage
Name
Value
Domain
Path
HTTP
Secure
First-Party
my-cookie
some value
localhost
/
19
session-id
cf3f7879-c3b2-42cf-5cff-16d514b943dd
localhost
/
46

Cookies, Sessions, UUIDs

- If you refresh
  - the cookie’s value remains the same
- If you delete the cookie
  - you get a new UUID
- To log a user out
  - delete the cookie

sessions

storing other info in the cookie

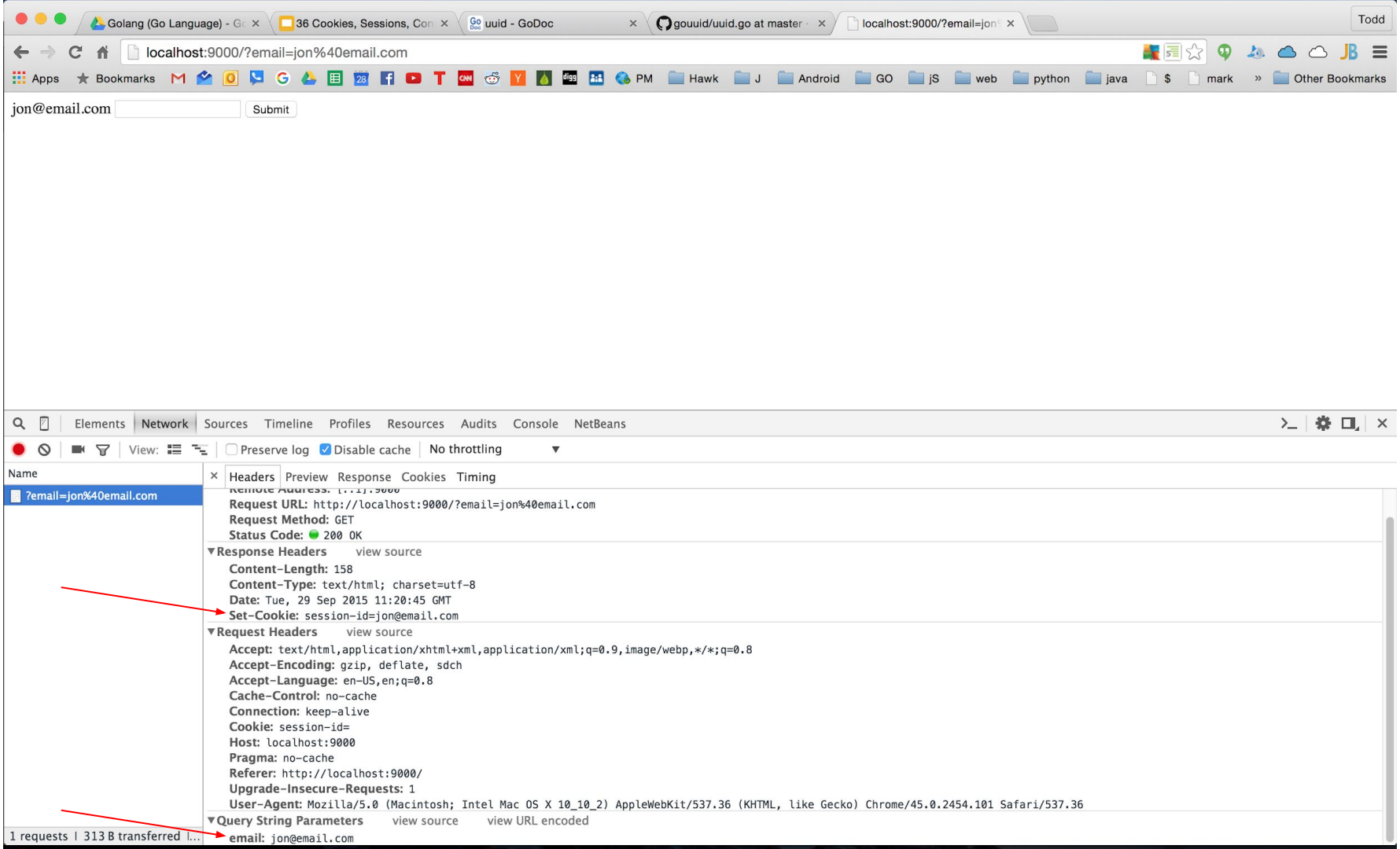
main.go

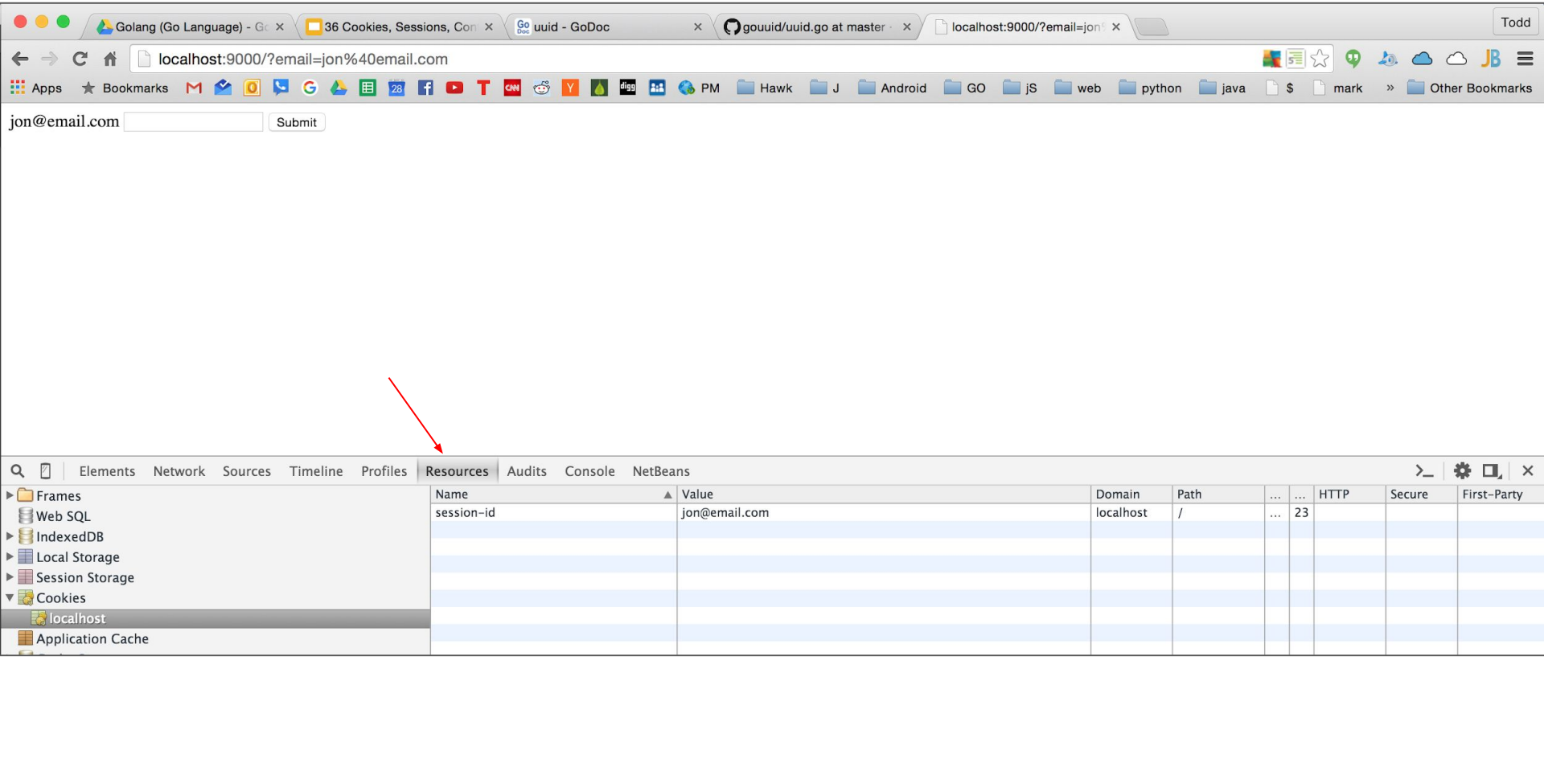
```go
package main

import (
    "fmt"
    "github.com/nu7hatch/gouuid"
    "net/http"
)

func main() {

    http.HandleFunc("/", func(res http.ResponseWriter, req *http.Request) {
        cookie, err := req.Cookie("session-id")
        // cookie is not set
        if err != nil {
            id, _ := uuid.NewV4()
            cookie = &http.Cookie{
                Name:  "session-id",
                Value: id.String() + " email=jon@email.com" + " JSON data" + " Whatever",
            }
            http.SetCookie(res, cookie)
        }
        fmt.Println(cookie)

    })
    http.ListenAndServe(":9000", nil)
}
```

main.go

An example of storing user info in a cookie
note: no UUID used here

```
import (
    "io"
    "net/http"
)

func main() {

    http.HandleFunc("/", func(res http.ResponseWriter, req *http.Request) {
        cookie, err := req.Cookie("session-id")
        // cookie is not set
        if err != nil {
            //id, _ := uuid.NewV4()
            cookie = &http.Cookie{
                Name: "session-id",
            }
        }

        if req.FormValue("email") != "" {
            cookie.Value = req.FormValue("email")
        }

        http.SetCookie(res, cookie)

        io.WriteString(res, `<!DOCTYPE html>
<html>
  <body>
    <form>
    `+cookie.Value+`
      <input type="email" name="email">
      <input type="submit">
    </form>
  </body>
</html>`)

    })
    http.ListenAndServe(":9000", nil)
}
```


Golang (Go Language) - Go
36 Cookies, Sessions, Con
uuid - GoDoc
gouuid/uuid.go at master
localhost:9000/?email=jon
Todd
localhost:9000/?email=jon%40email.com
Apps
Bookmarks
PM
Hawk
J
Android
GO
jS
web
python
java
$
mark
Other Bookmarks
jon@email.com
Submit
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
NetBeans
View:
Preserve log
Disable cache
No throttling
Name
?email=jon%40email.com
Headers
Preview
Response
Cookies
Timing
Request URL: http://localhost:9000/?email=jon%40email.com
Request Method: GET
Status Code: 200 OK
Response Headers
view source
Content-Length: 158
Content-Type: text/html; charset=utf-8
Date: Tue, 29 Sep 2015 11:20:45 GMT
Set-Cookie: session-id=jon@email.com
Request Headers
view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cache-Control: no-cache
Connection: keep-alive
Cookie: session-id=
Host: localhost:9000
Pragma: no-cache
Referer: http://localhost:9000/
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36
Query String Parameters
view source
view URL encoded
email: jon@email.com
1 requests | 313 B transferred


Golang (Go Language) - G
36 Cookies, Sessions, Con
uuid - GoDoc
gouuid/uuid.go at master
localhost:9000/?email=jon
Todd
localhost:9000/?email=jon%40email.com
Apps
Bookmarks
PM
Hawk
J
Android
GO
jS
web
python
java
$
mark
Other Bookmarks
jon@email.com
Submit
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
NetBeans
Frames
Web SQL
IndexedDB
Local Storage
Session Storage
Cookies
localhost
Application Cache
Name
Value
Domain
Path
HTTP
Secure
First-Party
session-id
jon@email.com
localhost
/
23

sessions

HMAC ~ Hash-based Message Authentication Code

Downside of Cookies

- User can change the content
  - you can’t trust the content
- We can protect against this with HMAC
  - HMAC - Hash Based Message Authentication Code
    - prevents cookie tampering
    - Go has a fantastic crypto library

type Hash

```
type Hash uint
```

Hash identifies a cryptographic hash function that is implemented in another package.

```
const (
    MD4         Hash = 1 + iota // import golang.org/x/crypto/md4
    MD5                         // import crypto/md5
    SHA1                        // import crypto/sha1
    SHA224                      // import crypto/sha256
    SHA256                      // import crypto/sha256
    SHA384                      // import crypto/sha512
    SHA512                      // import crypto/sha512
    MD5SHA1                     // no implementation; MD5+SHA1 used for TLS RSA
    RIPEMD160                   // import golang.org/x/crypto/ripemd160
    SHA3_224                    // import golang.org/x/crypto/sha3
    SHA3_256                    // import golang.org/x/crypto/sha3
    SHA3_384                    // import golang.org/x/crypto/sha3
    SHA3_512                    // import golang.org/x/crypto/sha3
    SHA512_224                  // import crypto/sha512
    SHA512_256                  // import crypto/sha512

)
```

Go: crypto/hmac Index | Files

package hmac

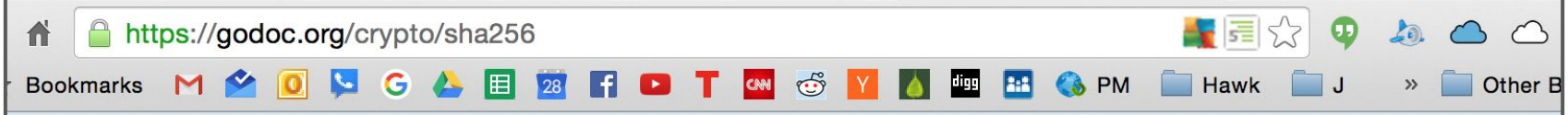
```
import "crypto/hmac"
```

Package hmac implements the Keyed-Hash Message Authentication Code (HMAC) as defined in U.S. Federal Information Processing Standards Publication 198. An HMAC is a cryptographic hash that uses a key to sign a message. The receiver verifies the hash by recomputing it using the same key.

Receivers should be careful to use Equal to compare MACs in order to avoid timing side-channels:

```
// CheckMAC reports whether messageMAC is a valid HMAC tag for message.
func CheckMAC(message, messageMAC, key []byte) bool {
        mac := hmac.New(sha256.New, key)
        mac.Write(message)
        expectedMAC := mac.Sum(nil)
        return hmac.Equal(messageMAC, expectedMAC)
}
```

Index



GoDoc Home Index About Search

Go: crypto/sha256 Index | Files

package sha256

```
import "crypto/sha256"
```

Package sha256 implements the SHA224 and SHA256 hash algorithms as defined in FIPS 180-4.

Index

Package Files

sha256.go sha256block_decl.go

main.go

```
10
11  func getCode(data string) string {
12      h := hmac.New(sha256.New, []byte("ourkey"))
13      io.WriteString(h, data)
14      return fmt.Sprintf("%x", h.Sum(nil))
15  }
16
17  func main() {
18      http.HandleFunc("/", func(res http.ResponseWriter, req *http.Request) {
19          cookie, err := req.Cookie("session-id")
20          // cookie is not set
21          if err != nil {
22              //id, _ := uuid.NewV4()
23              cookie = &http.Cookie{
24                  Name: "session-id",
25              }
26          }
27
28          if req.FormValue("email") != "" {
29              cookie.Value = req.FormValue("email")
30          }
31
32          fmt.Println(getCode(cookie.Value))
33          http.SetCookie(res, cookie)
34          io.WriteString(res, `<!DOCTYPE html>
35  <html>
36    <body>
37      <form method="POST">
38      `+cookie.Value+`
39       <input type="email" name="email">
40       <input type="password" name="password">
41       <input type="submit">
42      </form>
43    </body>
44  </html>`)
45
46      })
47      http.ListenAndServe(":9000", nil)
48  }
49
```

Terminal

```
75bbe8b70010241834d5f8575d5e7bfcbbd7ab11ea108e6c0d21209d30735ae9
```

HMAC

- When a cookie comes in on a request
  - we can verify it hasn’t been tampered with
    - this gets complicated quickly
- we can use prebuilt session libraries instead
  - gorilla sessions

sessions

gorilla sessions

Overview

[Top]

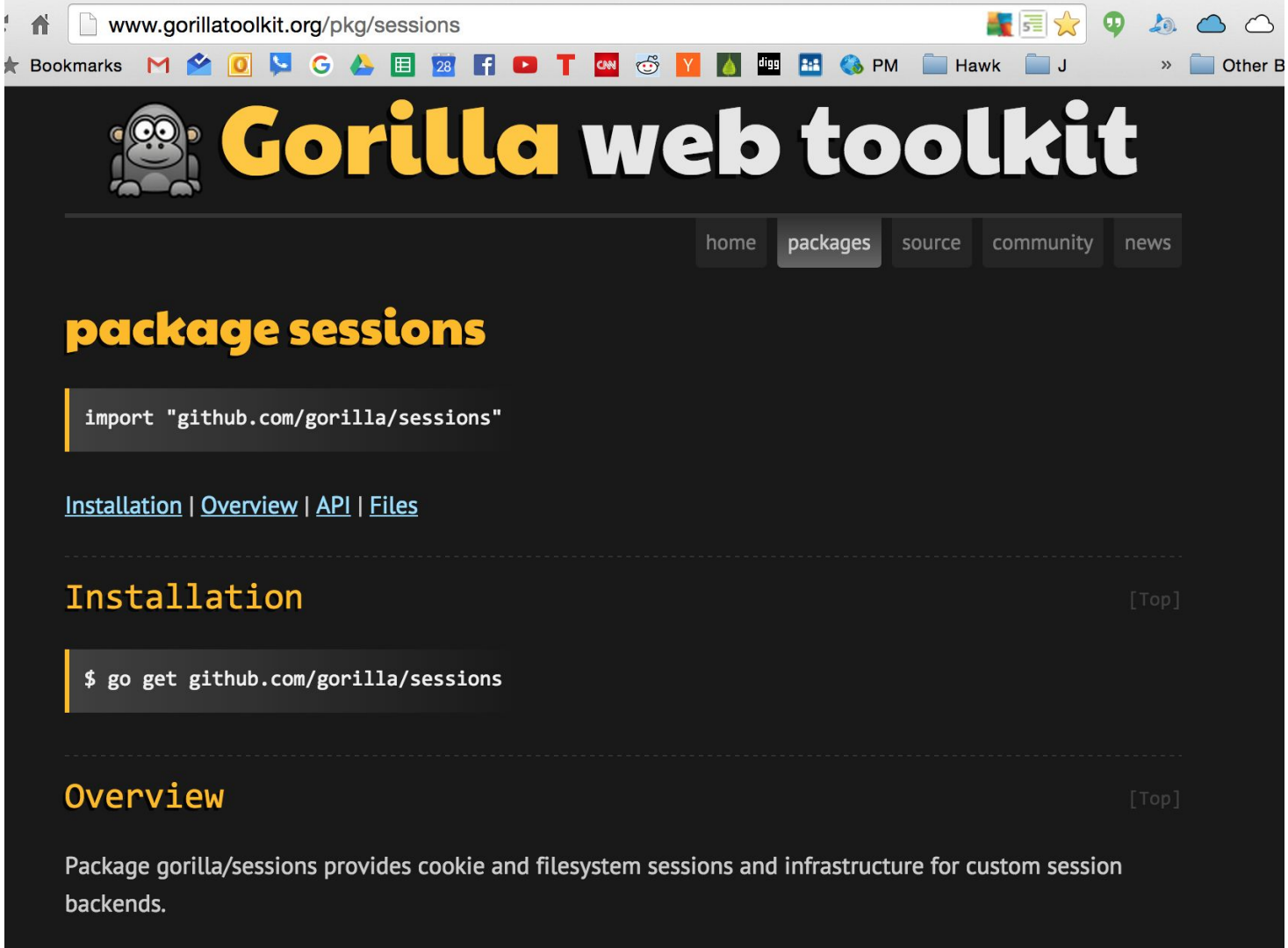
Package gorilla/securecookie encodes and decodes authenticated and optionally encrypted cookie values.

Secure cookies can't be forged, because their values are validated using HMAC. When encrypted, the content is also inaccessible to malicious eyes.

To use it, first create a new SecureCookie instance:

```
var hashKey = []byte("very-secret")
var blockKey = []byte("a-lot-secret")
var s = securecookie.New(hashKey, blockKey)
```

The hashKey is required, used to authenticate the cookie value using HMAC. It is recommended to use a key with 32 or 64 bytes.


www.gorillatoolkit.org/pkg/sessions
Bookmarks
PM
Hawk
J
Other B
Gorilla web toolkit
home
packages
source
community
news
package sessions
import "github.com/gorilla/sessions"
Installation | Overview | API | Files
Installation
[Top]
$ go get github.com/gorilla/sessions
Overview
[Top]
Package gorilla/sessions provides cookie and filesystem sessions and infrastructure for custom session backends.

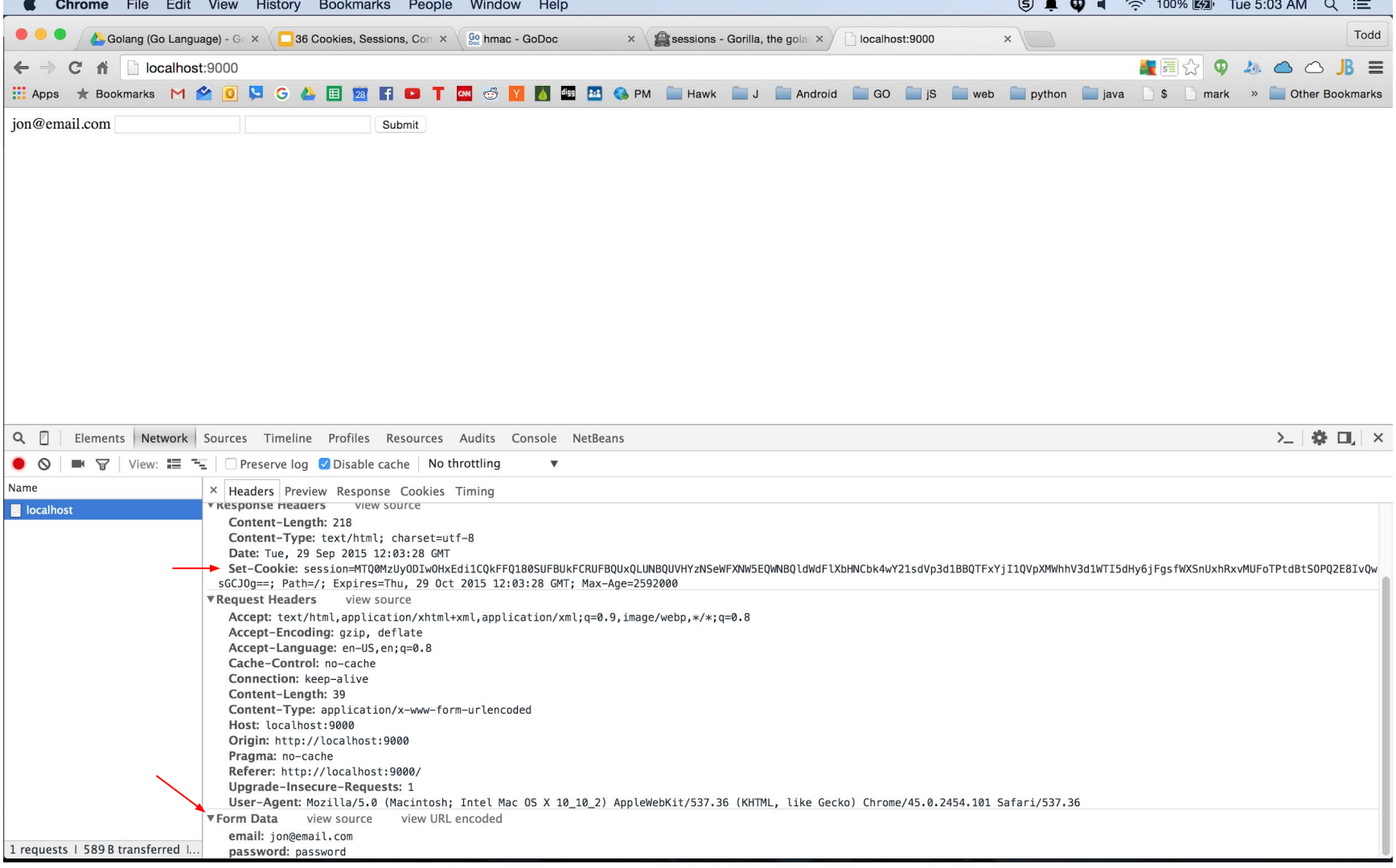
Let's start with an example that shows the sessions API in a nutshell:

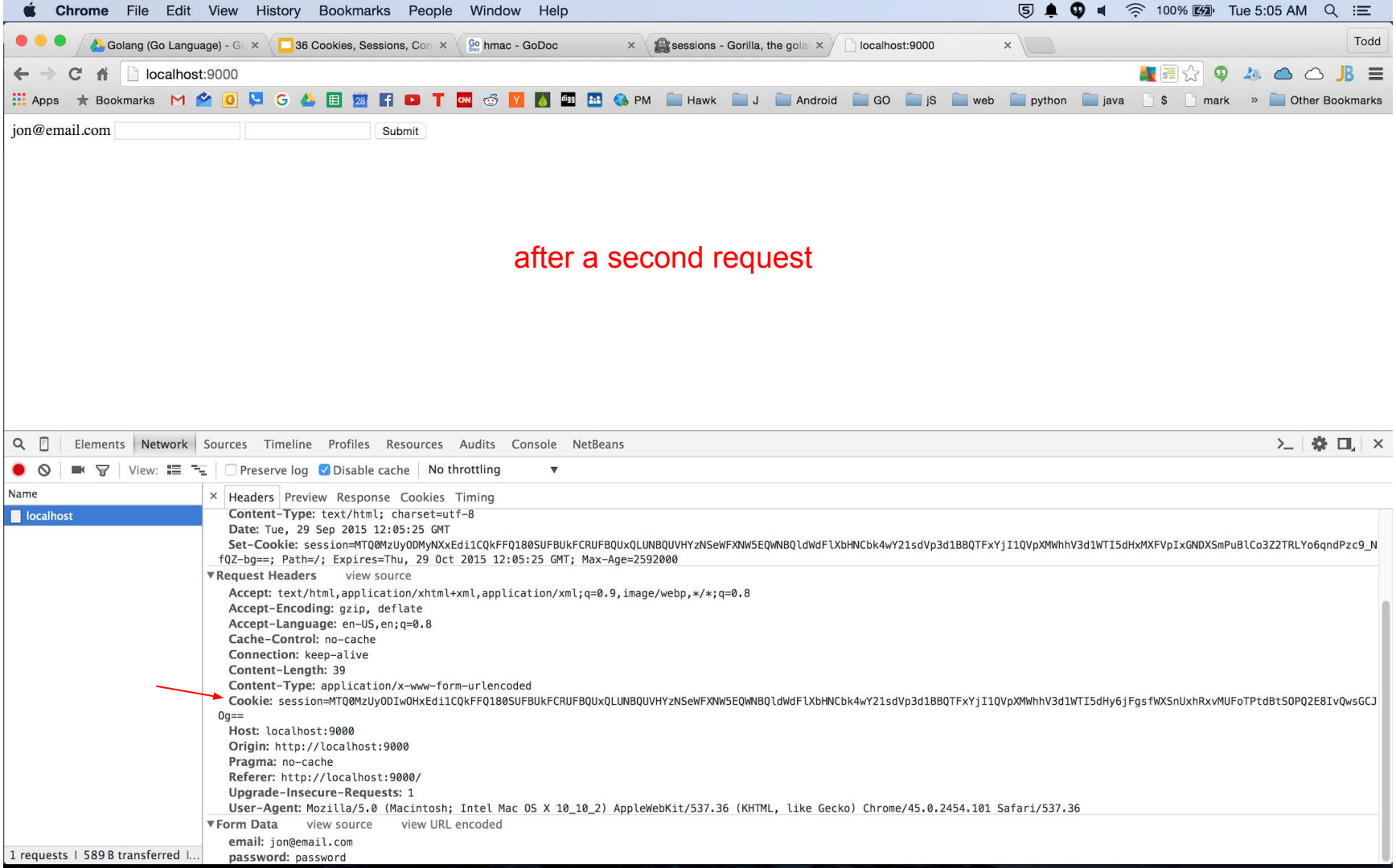
```
import (
    "net/http"
    "github.com/gorilla/sessions"
)

var store = sessions.NewCookieStore([]byte("something-very-secret"))

func MyHandler(w http.ResponseWriter, r *http.Request) {
    // Get a session. We're ignoring the error resulted from decoding an
    // existing session: Get() always returns a session, even if empty.
    session, err := store.Get(r, "session-name")
    if err != nil {
        http.Error(w, err.Error(), 500)
        return
    }

    // Set some session values.
    session.Values["foo"] = "bar"
    session.Values[42] = 43
    // Save it before we write to the response/return from the handler.
    session.Save(r, w)
}
```

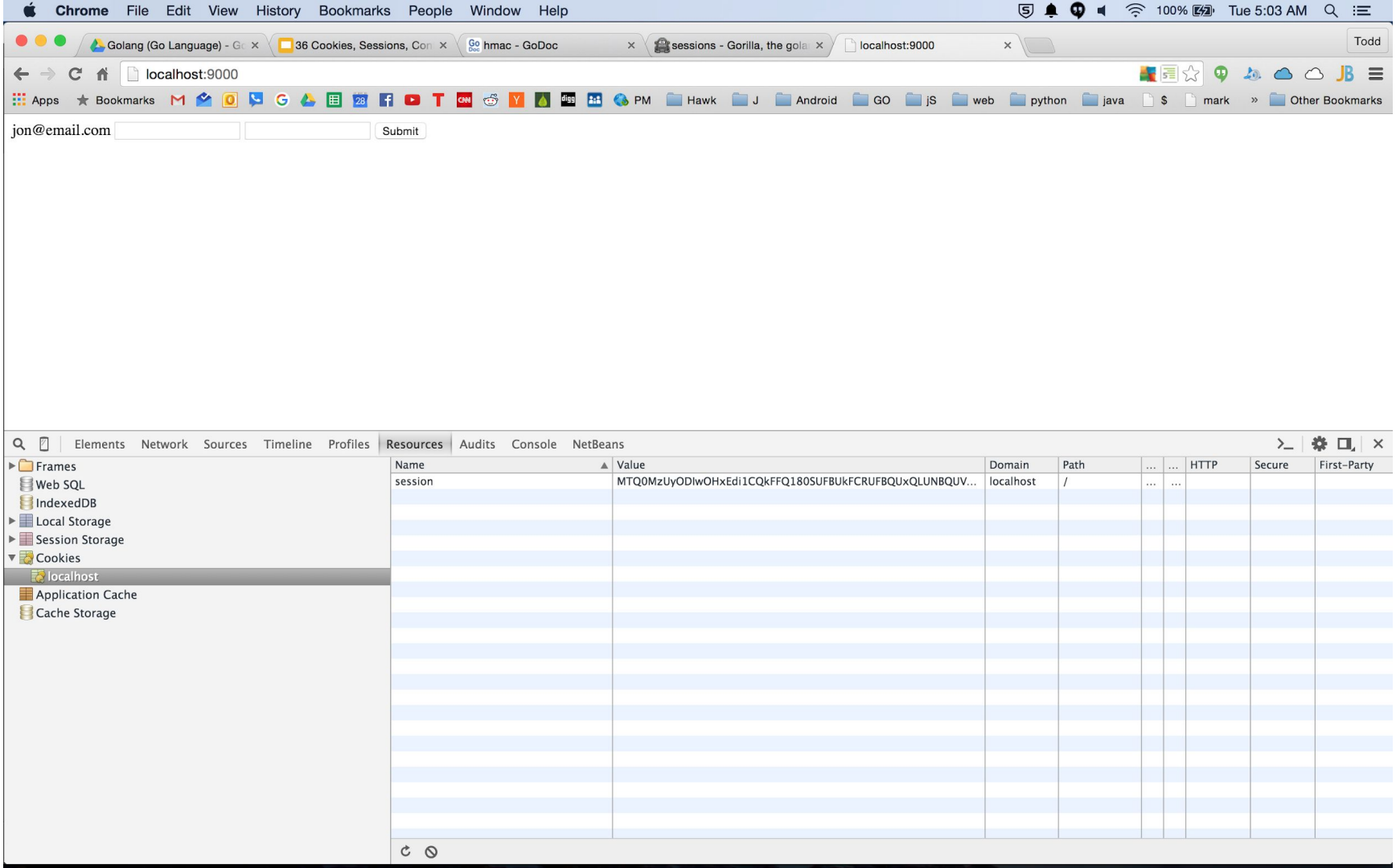
main.go

```go
package main

import (
    "fmt"
    "github.com/gorilla/sessions"
    "io"
    "net/http"
)

var store = sessions.NewCookieStore([]byte("secret-password"))

func main() {
    http.HandleFunc("/", func(res http.ResponseWriter, req *http.Request) {
        session, _ := store.Get(req, "session")
        if req.FormValue("email") != "" {
            // check password
            session.Values["email"] = req.FormValue("email")
        }
        session.Save(req, res)

        io.WriteString(res, `<!DOCTYPE html>
<html>
  <body>
    <form method="POST">
    `+fmt.Sprint(session.Values["email"])+`
      <input type="email" name="email">
      <input type="password" name="password">
      <input type="submit">
    </form>
  </body>
</html>`)

    })
    http.ListenAndServe(":9000", nil)
}
```

We can now create sessions: we store session.values and get session.values; being able to store email or user ID allows us to create a session for a user


Chrome File Edit View History Bookmarks People Window Help
Golang (Go Language) - Go
36 Cookies, Sessions, Cor
hmac - GoDoc
sessions - Gorilla, the gola
localhost:9000
Todd
localhost:9000
Apps Bookmarks PM Hawk J Android GO jS web python java $ mark Other Bookmarks
jon@email.com
Submit
Elements Network Sources Timeline Profiles Resources Audits Console NetBeans
View: Preserve log Disable cache No throttling
Name
localhost
Headers Preview Response Cookies Timing
Response Headers view source
Content-Length: 218
Content-Type: text/html; charset=utf-8
Date: Tue, 29 Sep 2015 12:03:28 GMT
Set-Cookie: session=MTQ0MzUyODIwOHxEdi1CQkFFQ180SUFBUkFCRUFBQUxQLUNBQUVHYzNSeWFXNW5EQWNBQldWdFlXbHNCbk4wY21sdVp3d1BBQTFxYjI1QVpXMWhhV3d1WTI5dHy6jFgsfWXSnUxhRxvMUFoTPtdBtSOPQ2E8IvQwsGCJOg==; Path=/; Expires=Thu, 29 Oct 2015 12:03:28 GMT; Max-Age=2592000
Request Headers view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Cache-Control: no-cache
Connection: keep-alive
Content-Length: 39
Content-Type: application/x-www-form-urlencoded
Host: localhost:9000
Origin: http://localhost:9000
Pragma: no-cache
Referer: http://localhost:9000/
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36
Form Data view source view URL encoded
email: jon@email.com
password: password
1 requests | 589 B transferred |...


Chrome File Edit View History Bookmarks People Window Help
100% Tue 5:05 AM
Golang (Go Language) - G
36 Cookies, Sessions, Cor
hmac - GoDoc
sessions - Gorilla, the gola
localhost:9000
Todd
localhost:9000
Apps Bookmarks PM Hawk J Android GO jS web python java $ mark Other Bookmarks
jon@email.com
Submit
after a second request
Elements Network Sources Timeline Profiles Resources Audits Console NetBeans
Preserve log
Disable cache
No throttling
Name
localhost
Headers Preview Response Cookies Timing
Content-Type: text/html; charset=utf-8
Date: Tue, 29 Sep 2015 12:05:25 GMT
Set-Cookie: session=MTQ0MzUyODMyNXxEdi1CQkFFQ180SUFBUkFCRUFBQUxQLUNBQUVHYzNSeWFXNW5EQWNBQldWdFlXbHNCbk4wY21sdVp3d1BBQTFxYjI1QVpXMWhhV3d1WTI5dHxMXFVpIxGNDXSmPuBlCo3Z2TRLYo6qndPzc9_NfQZ-bg==; Path=/; Expires=Thu, 29 Oct 2015 12:05:25 GMT; Max-Age=2592000
Request Headers view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Cache-Control: no-cache
Connection: keep-alive
Content-Length: 39
Content-Type: application/x-www-form-urlencoded
Cookie: session=MTQ0MzUyODIwOHxEdi1CQkFFQ180SUFBUkFCRUFBQUxQLUNBQUVHYzNSeWFXNW5EQWNBQldWdFlXbHNCbk4wY21sdVp3d1BBQTFxYjI1QVpXMWhhV3d1WTI5dHy6jFgsfWXSnUxhRxvMUFoTPtdBtSOPQ2E8IvQwsGCJOg==
Host: localhost:9000
Origin: http://localhost:9000
Pragma: no-cache
Referer: http://localhost:9000/
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36
Form Data view source view URL encoded
email: jon@email.com
password: password
1 requests | 589 B transferred


Chrome File Edit View History Bookmarks People Window Help
100% Tue 5:03 AM
Golang (Go Language) - G
36 Cookies, Sessions, Cor
hmac - GoDoc
sessions - Gorilla, the gola
localhost:9000
Todd
localhost:9000
Apps Bookmarks PM Hawk J Android GO jS web python java $ mark Other Bookmarks
jon@email.com
Submit
Elements Network Sources Timeline Profiles Resources Audits Console NetBeans
Frames
Web SQL
IndexedDB
Local Storage
Session Storage
Cookies
localhost
Application Cache
Cache Storage
Name Value Domain Path HTTP Secure First-Party
session MTQ0MzUyODIwOHxEdi1CQkFFQ180SUFBUkFCRUFBQUxQLUNBQUV... localhost /

exercises

Cookies

Create an http application that tracks how many times a user accesses your web page with a cookie.

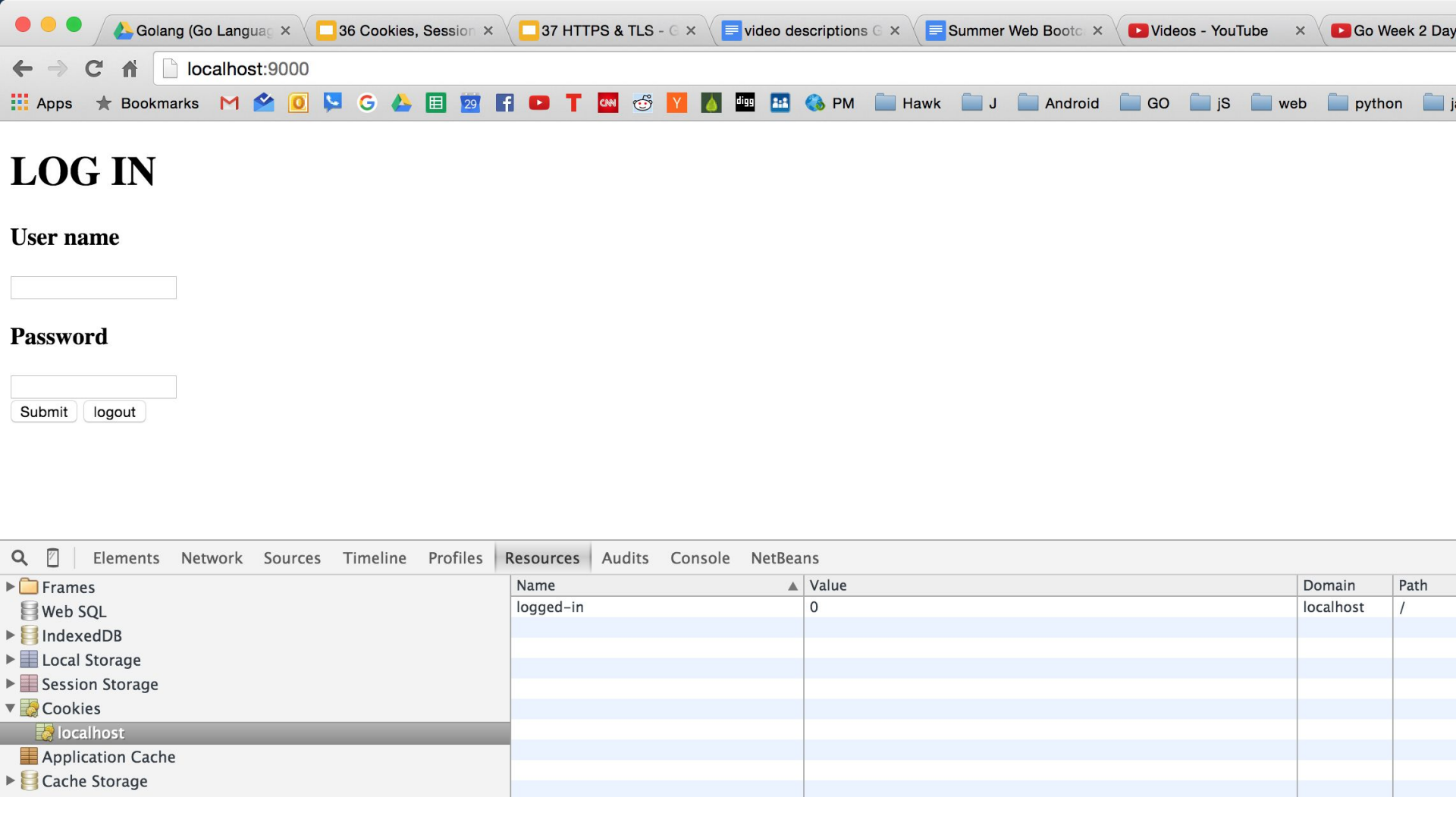
main.go

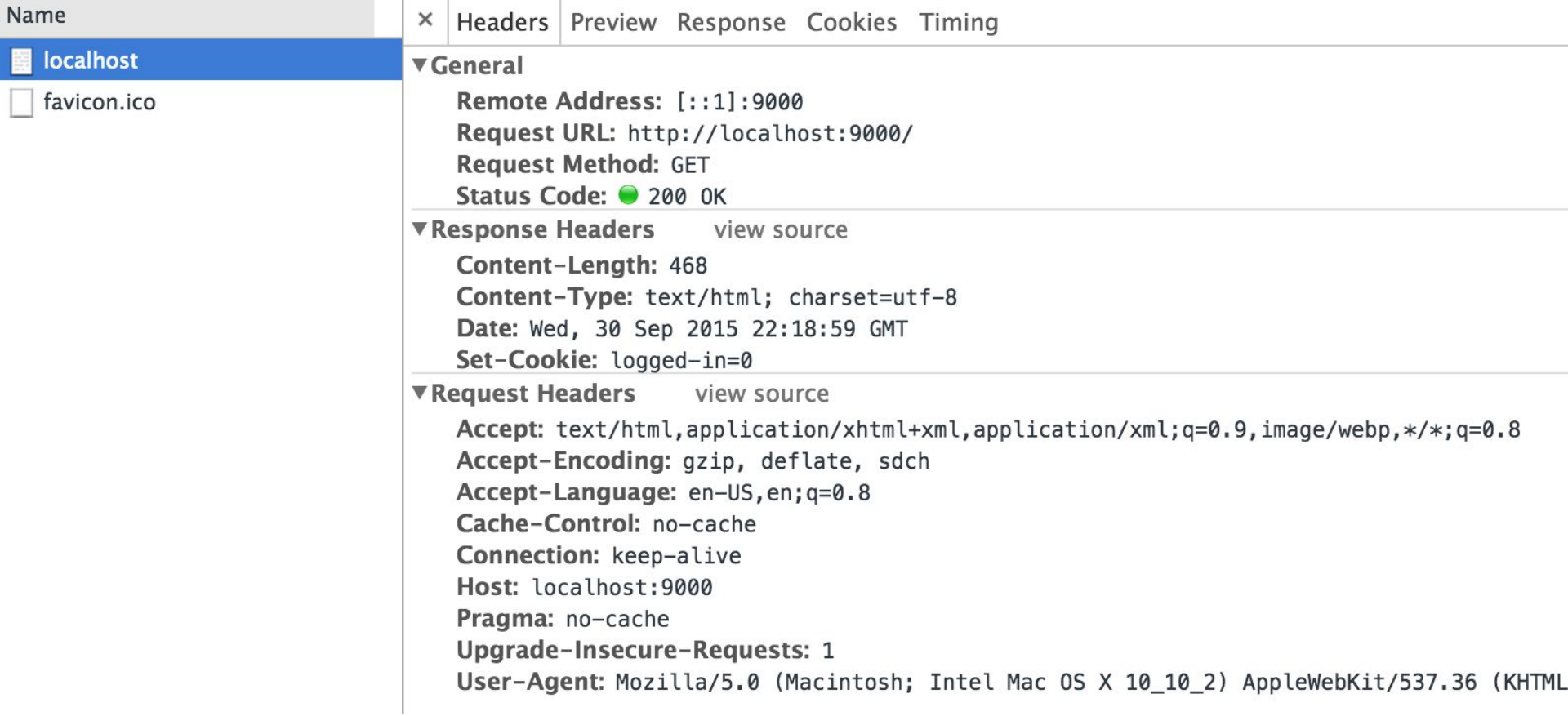
```go
package main

import (
    "io"
    "net/http"
    "strconv"
)

func main() {
    http.HandleFunc("/", func(res http.ResponseWriter, req *http.Request) {
        cookie, err := req.Cookie("my-cookie")
        // there is no cookie
        if err == http.ErrNoCookie {
            cookie = &http.Cookie{
                Name:  "my-cookie",
                Value: "0",
            }
        }
        // there is a cookie
        count, _ := strconv.Atoi(cookie.Value)
        count++
        cookie.Value = strconv.Itoa(count)

        http.SetCookie(res, cookie)

        io.WriteString(res, cookie.Value)
    })
    http.ListenAndServe(":9000", nil)
}

```

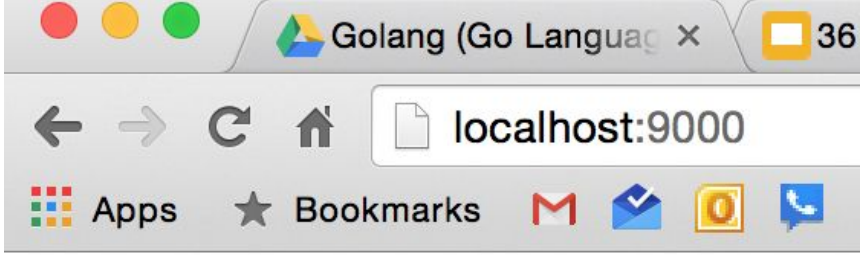
Sessions

- Create an http application that supports at least 2 endpoints: login and logout.
  - login should accept a form and save a cookie. (with at least the username)
  - logout should clear the cookie

```go
func main() {
    http.HandleFunc("/", func(res http.ResponseWriter, req *http.Request) {
        cookie, err := req.Cookie("logged-in")
        // no cookie
        if err == http.ErrNoCookie {
            cookie = &http.Cookie{
                Name:  "logged-in",
                Value: "0",
            }
        }

        // check log in
        if req.Method == "POST" {
            password := req.FormValue("password")
            if password == "secret" {
                cookie = &http.Cookie{
                    Name:  "logged-in",
                    Value: "1",
                }
            }
        }

        // if logout, then logout
        if req.URL.Path == "/logout" {
            cookie = &http.Cookie{
                Name:   "logged-in",
                Value:  "0",
                MaxAge: -1,
            }
        }

        http.SetCookie(res, cookie)
        var html string

        // not logged in
        if cookie.Value == strconv.Itoa(0) {
            html = `
            <!DOCTYPE html>
            <html lang="en">
            <head>
                <meta charset="UTF-8">
                <title></title>
            </head>
            <body>
            <h1>LOG IN</h1>
            <form method="post" action="http://localhost:9000/">
                <h3>User name</h3>
                <input type="text" name="userName" id="userName">
                <h3>Password</h3>
                <input type="text" name="password" id="password">
                <br>
                <input type="submit">
                <input type="submit" name="logout" value="logout">
            </form>
            </body>
            </html>`
        }

        // logged in
        if cookie.Value == strconv.Itoa(1) {
            html = `
            <!DOCTYPE html>
            <html lang="en">
            <head>
                <meta charset="UTF-8">
                <title></title>
            </head>
            <body>
            <h1><a href="http://localhost:9000/logout">LOG OUT</a></h1>
            </body>
            </html>`
        }

        io.WriteString(res, html)
    })
    http.ListenAndServe(":9000", nil)
}
```


Golang (Go Languag
36 Cookies, Session
37 HTTPS & TLS - C
video descriptions C
Summer Web Bootc
Videos - YouTube
Go Week 2 Day
localhost:9000
Apps
Bookmarks
PM
Hawk
J
Android
GO
jS
web
python
LOG IN
User name
Password
Submit
logout
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
NetBeans
Frames
Web SQL
IndexedDB
Local Storage
Session Storage
Cookies
localhost
Application Cache
Cache Storage
Name
Value
Domain
Path
logged-in
0
localhost
/


Name
localhost
favicon.ico
Headers
Preview
Response
Cookies
Timing
General
Remote Address: [::1]:9000
Request URL: http://localhost:9000/
Request Method: GET
Status Code: 200 OK
Response Headers
view source
Content-Length: 468
Content-Type: text/html; charset=utf-8
Date: Wed, 30 Sep 2015 22:18:59 GMT
Set-Cookie: logged-in=0
Request Headers
view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cache-Control: no-cache
Connection: keep-alive
Host: localhost:9000
Pragma: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2) AppleWebKit/537.36 (KHTML



LOG IN

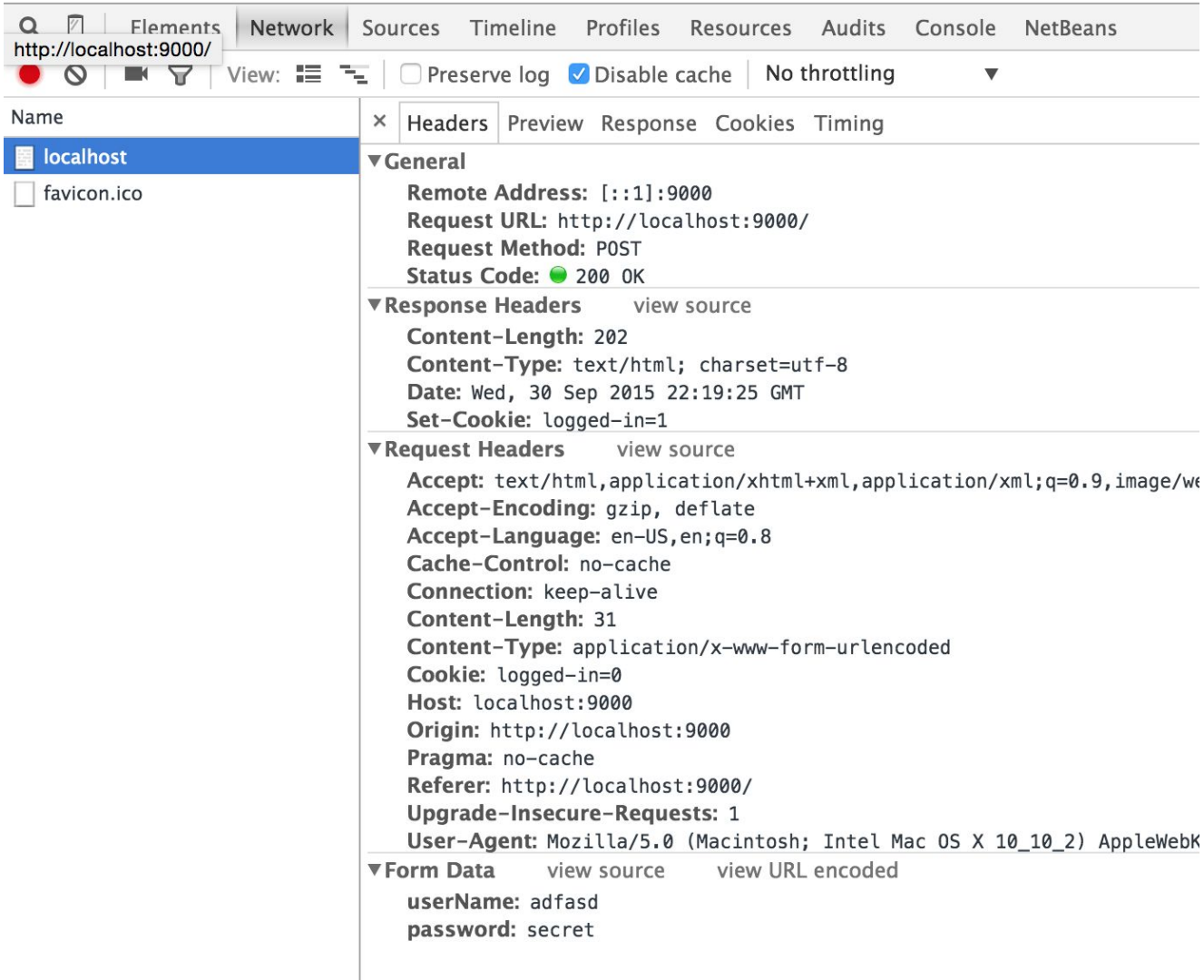
User name

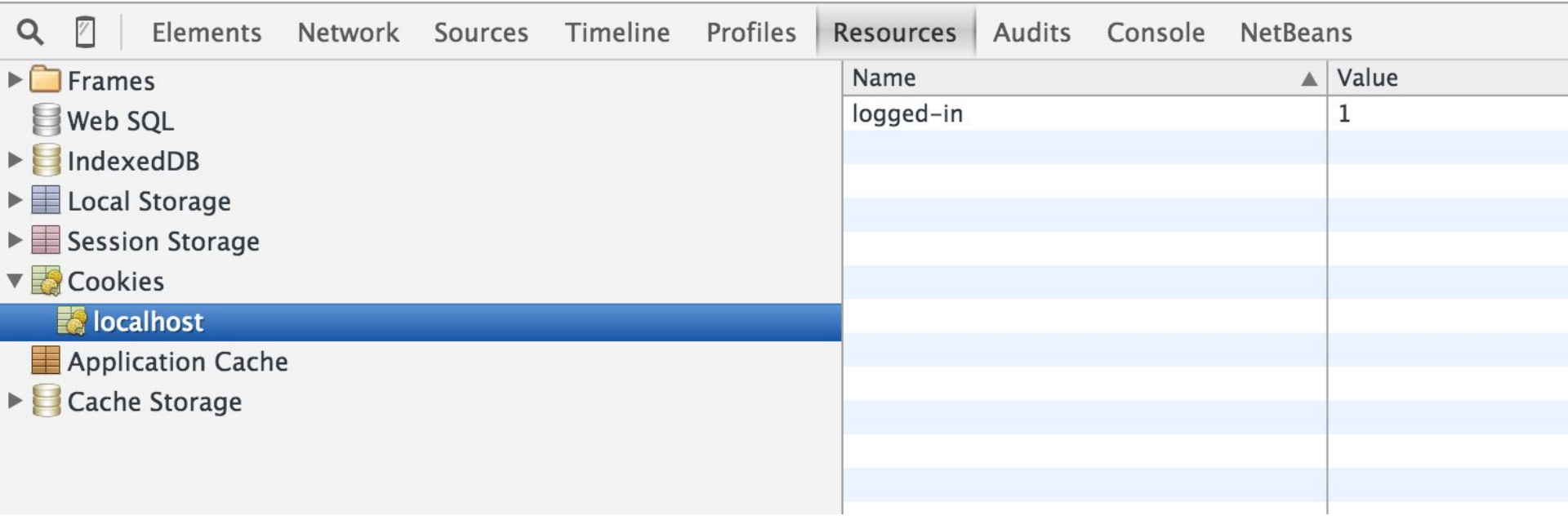
adfasd

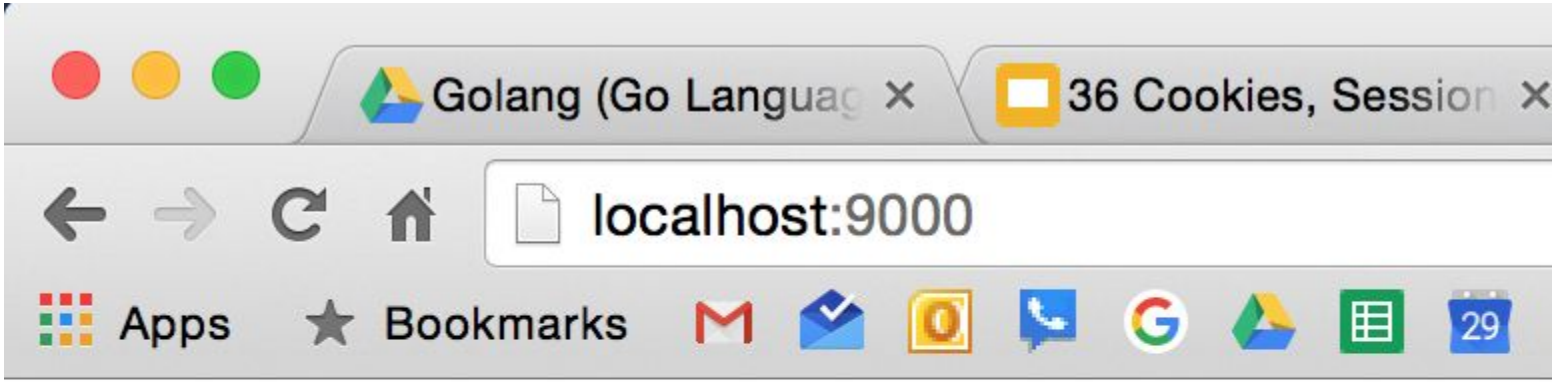
Password

secret

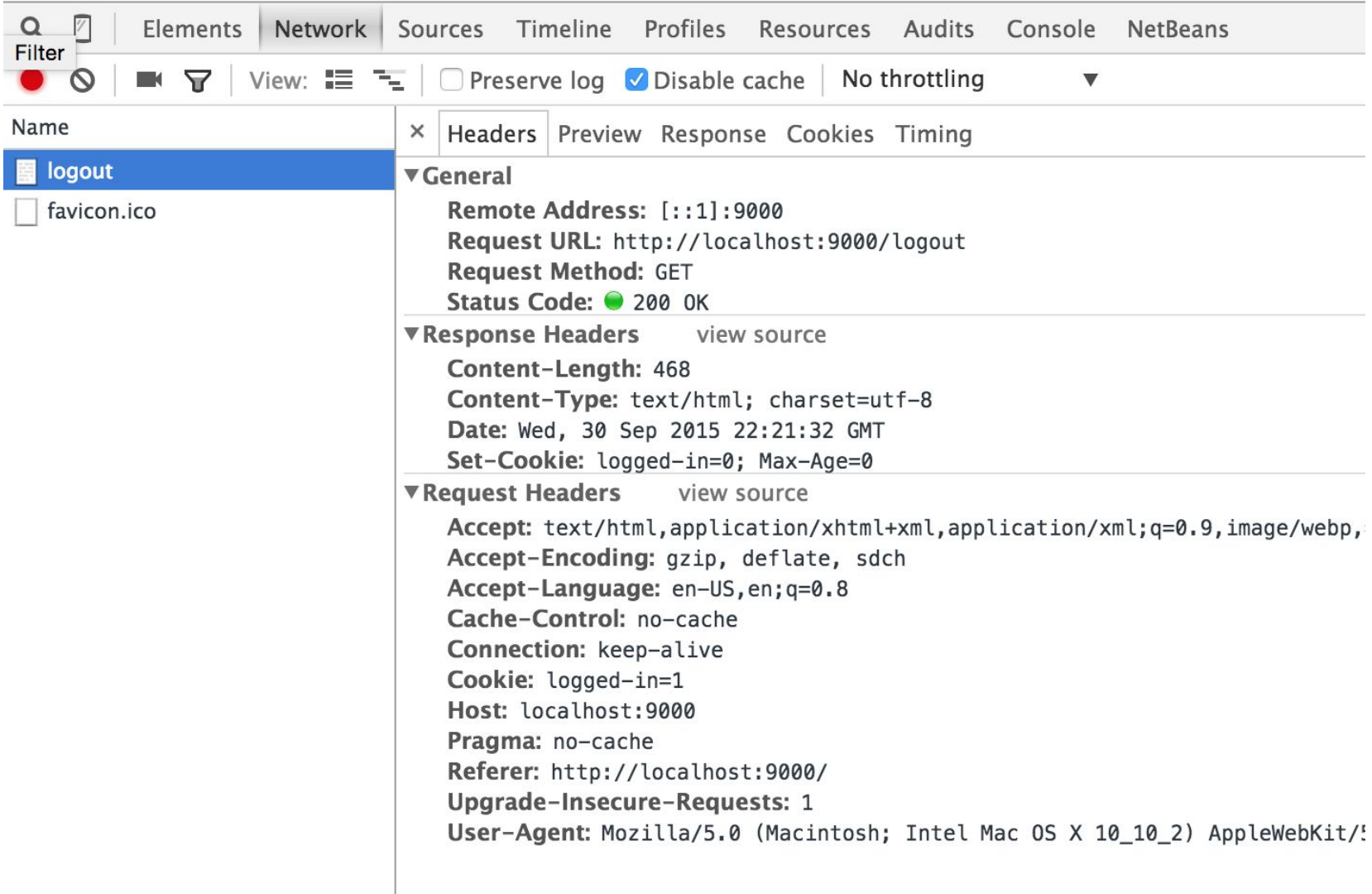
Submit logout

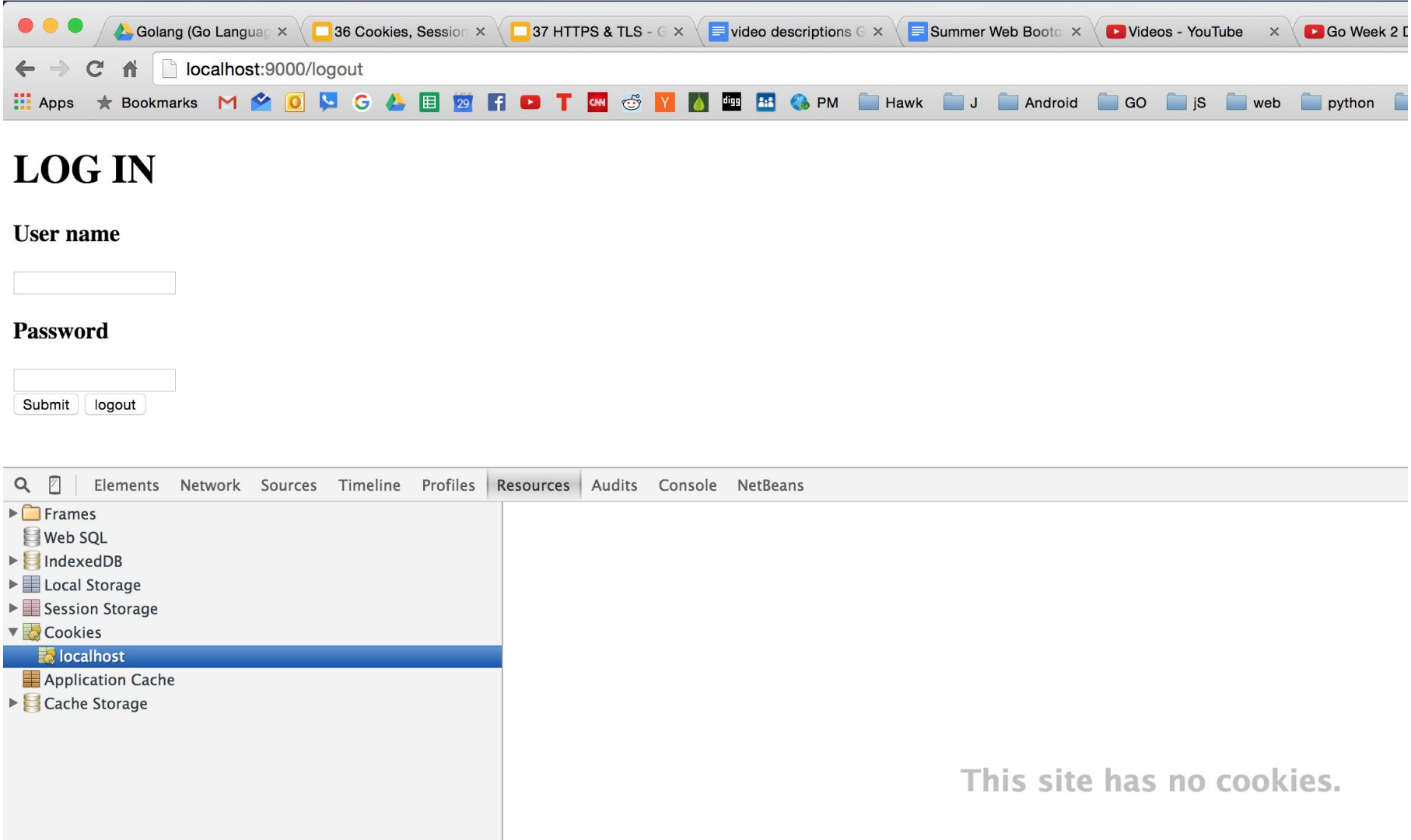

Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
NetBeans
http://localhost:9000/
View:
Preserve log
Disable cache
No throttling
Name
localhost
favicon.ico
Headers
Preview
Response
Cookies
Timing
General
Remote Address: [::1]:9000
Request URL: http://localhost:9000/
Request Method: POST
Status Code: 200 OK
Response Headers
view source
Content-Length: 202
Content-Type: text/html; charset=utf-8
Date: Wed, 30 Sep 2015 22:19:25 GMT
Set-Cookie: logged-in=1
Request Headers
view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/we
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Cache-Control: no-cache
Connection: keep-alive
Content-Length: 31
Content-Type: application/x-www-form-urlencoded
Cookie: logged-in=0
Host: localhost:9000
Origin: http://localhost:9000
Pragma: no-cache
Referer: http://localhost:9000/
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2) AppleWebK
Form Data
view source
view URL encoded
userName: adfasd
password: secret


Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
NetBeans
Frames
Web SQL
IndexedDB
Local Storage
Session Storage
Cookies
localhost
Application Cache
Cache Storage
Name
Value
logged-in
1


Golang (Go Languag
36 Cookies, Session
localhost:9000
Apps
Bookmarks
29

LOG OUT


Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
NetBeans
Filter
View:
Preserve log
Disable cache
No throttling
Name
logout
favicon.ico
Headers
Preview
Response
Cookies
Timing
General
Remote Address: [::1]:9000
Request URL: http://localhost:9000/logout
Request Method: GET
Status Code: 200 OK
Response Headers
view source
Content-Length: 468
Content-Type: text/html; charset=utf-8
Date: Wed, 30 Sep 2015 22:21:32 GMT
Set-Cookie: logged-in=0; Max-Age=0
Request Headers
view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cache-Control: no-cache
Connection: keep-alive
Cookie: logged-in=1
Host: localhost:9000
Pragma: no-cache
Referer: http://localhost:9000/
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2) AppleWebKit/5


Golang (Go Languag
36 Cookies, Session
37 HTTPS & TLS - C
video descriptions C
Summer Web Bootc
Videos - YouTube
Go Week 2 D
localhost:9000/logout
Apps
Bookmarks
PM
Hawk
J
Android
GO
jS
web
python
LOG IN
User name
Password
Submit
logout
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
NetBeans
Frames
Web SQL
IndexedDB
Local Storage
Session Storage
Cookies
localhost
Application Cache
Cache Storage
This site has no cookies.