Assignment 1
Cookies and Sessions Management

CSCI 4140 Tutorial 4
Feb. 25
Qin Chuan

Cookies and Session

Cookies is local storage of information in browser

- Key-value pair
- Set by cookies in HTTP response
- Embed in later HTTP request

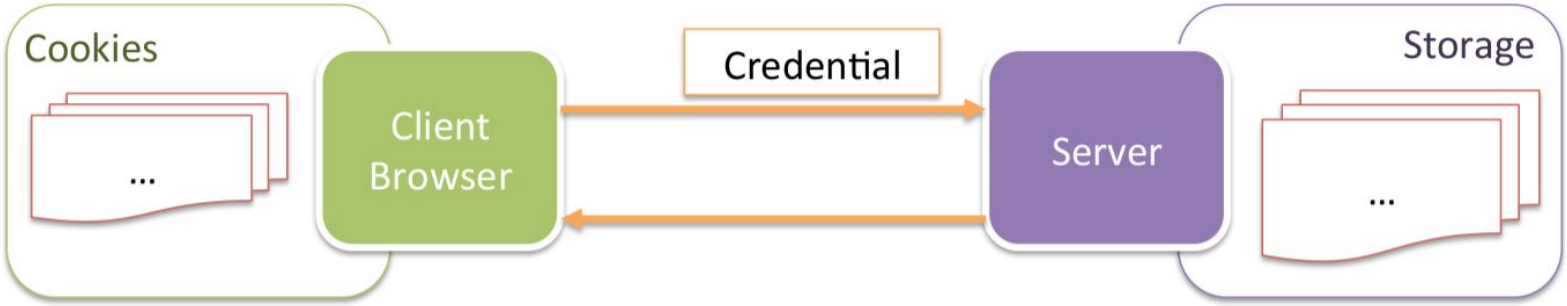
Session is to verify yourself with server

- Identify you as recently logged in user
- Something (e.g. session ID) shared between client browser and server
- Session ID can be stored as cookies in client

Cookies and Session

Cookies and session work together

- Client sent a request to server with credential



Cookies and Session

Cookies and session work together

- Server verify the credential received
- If credential is valid, server generate a session for client
  - Session includes session ID, username, login time (if implementing timeout in server side) etc
- Session information is recorded in server side storage
- Session ID embed in HTTP header of response to client

Cookies and Session

- Client browser store session ID from response as cookie
- When user access the same domain, corresponding cookies (session ID) is embedded in HTTP header
- Server can check if session ID valid, and generate the corresponding response

Cookies in Python

Cookies should be included in HTTP header

- Compute information before end of HTTP header
- Before content / HTML

`Cookie`: Python module to handle cookies

- Parse cookies in request
- Print cookies for HTTP header in response

Initialize SimpleCookie Object

- Cookies in request are stored in environment variable

```
os.environ['HTTP_COOKIE']
```

- Parse the content of environment variable to `SimpleCookie` object

```
cookieDict = Cookie.SimpleCookie(os.environ['HTTP_COOKIE'])
```

- If no cookie in request, create the object without any entry

```
cookieDict = Cookie.SimpleCookie()
```

tutorial5/cookie.cgi

SimpleCookie as a Dictionary

- Access as directory

- Get cookie value if set

```
val = cookieDict['session']
```

  - Raise exception if not set
- Set value of cookie

```
cookieDict['session'] = value
```

- Attribute you may need
  - expires: to specify expire time of cookie
  - If not set, cookie will expire when browser close

tutorial5/cookie.cgi

Computing Expire Time

- Get current time (in Unix timestamp) by

```
time.time()
```

You will need `time` module (`import time`)

Compute expire time from current timestamp

1 Day * 24 hrs * 60 min * 60 sec

```
expireTimestamp = time.time() + 1 * 24 * 60 * 60
```

- Make this timestamp to proper format

```
expireTime = time.strftime("%a, %d-%b-%Y %T GMT",
                           time.gmtime(expireTimestamp))
```

```
expireTimestamp: 1422824533.27
expireTime:      Sun, 01-Feb-2015 21:02:13 GMT
```

- Finally, set this to cookie entry

```
cookieDict['session']['expires'] = expireTime
```

tutorial5/cookie.cgi

Print cookies in HTTP header

- Our common HTTP header is following:

```
print 'Content-Type: text/html'
print
```

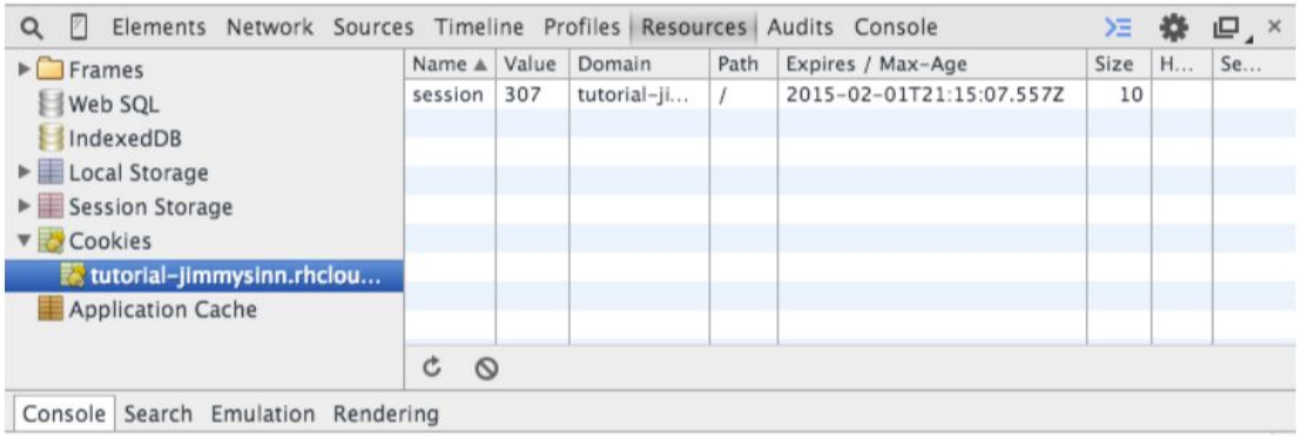
- To set cookie, print the SimpleCookie object before the delimiter

```
print 'Content-Type: text/html'
print cookieDict
print
```

tutorial5/cookie.cgi

Cookies in Python: Summary

- If you do not add cookie in header, the cookie still kept (if not expired)
  - A good practice: always update cookie expire time in response
  - To unset the cookie, set expire time to past
- You can check cookies from inspect element



  - If you see ‘Session’ in expires field, you probably failed to set expire time

Session

- Generate a ‘random’ string (session key) on server
- Store the session key in server
  - Database / Text file
- Send and session key to client browser
  - As cookies

Verifying Active Session

- Check cookies from client browser

- Match with entry stored in server
  - If matched, then this is active session
  - If not matched, reject user

Session / Cookies Requirement in Assignment 1

- Editing: storing current progress in cookies
  - Another way round: using HTML form
  - Check sample code in tutorial 2
- Resume
  - Associate session with filename and current progress in server
  - Set session ID as cookie to browser
  - Expire after a month
  - When browser is **closed and re-open**, cookie is kept
  - If server recognize a valid cookie,
  - Allow resume (Add button in index page!)
  - Use filename and progress to generate the editor page

More about Python & Hints

Some more Utilities

- List all files inside a directory

```
os.listdir(path)
```

  - Return list of files
  - Use for-in loop to loop it

```
for f in os.listdir(path):
    print f
```

- Get modification time of file

```
os.path.getmtime(f)
```

- Find from Google / StackOverflow / Python doc

Redirection

- HTTP redirect header
  - Print the following instead of normal HTTP header

```
print 'Status: 302 Found'
print 'Location: index.cgi'
print
```

Target

- HTML meta tag

```
<meta http-equiv="refresh" content="2; url=index.cgi" />
```

Refresh after 2 seconds

Target

  - Place this tag in head section

Installation Script: Reset the Instagram

- Two scenarios in executing the installation script
  - Perform **clean install**
  - No old database table / directory
  - Re-install / **Reset**
  - Database contains old table
- After confirmation, it will reset the Instagram
  - Remove all photo from database (and storage) [if exist]
  - Clear sessions / resume information [if exist]
  - Create all necessary tables [if not exists]
  - Create any required directory in persistent directory [if not exist]
- Confirmation interface: `init.html`
  - Don’t forget to handle “Go Back”

Input Validation

- Validation is required on client-side only (except file upload)
  - Although it is not secure enough for real application
  - Of course, you can do server-side validation if you want
- We will NOT modify your HTML code using inspect element
  - But we may input anything possible to fields
- Use HTML5 input type / attribute
  - Number
  - Max / Min

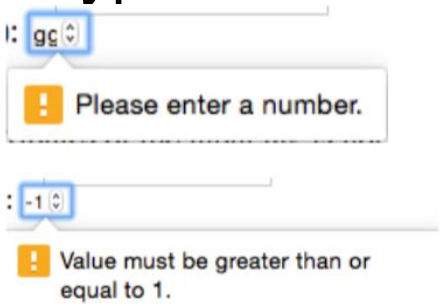


Image Upload

- Validation of uploaded Image
  - Extension **match** actual image type?
  - JPEG in .jpg file, GIF in .gif file, PNG in .png file
  - Check actual image type by ImageMagick
  - Note: we will not test animated GIF

- Trouble in handling result of `identify`?
  - Try `-format` flag

Image Upload

Image filename

- What if file with same name exist already ?
- Maybe editing / finalized
- Special characters contained in filename
- Space character

A call maybe useful: `cgi.escape(str)`

- Help you to escape most needed characters for HTTP
- Or you can simply discard the original filename (but keep the extension)

Photo Editing

- Commands of filter are given
  - Given command are for shell prompt
  - Your job is to adapt the command for subprocess
- Just direct copy for most of the commands
  - Change commands into list of argument in python

- Some characters may need escape in python
  - '%'
- Some characters do NOT need escape in python
  - '!'

Photo Editing

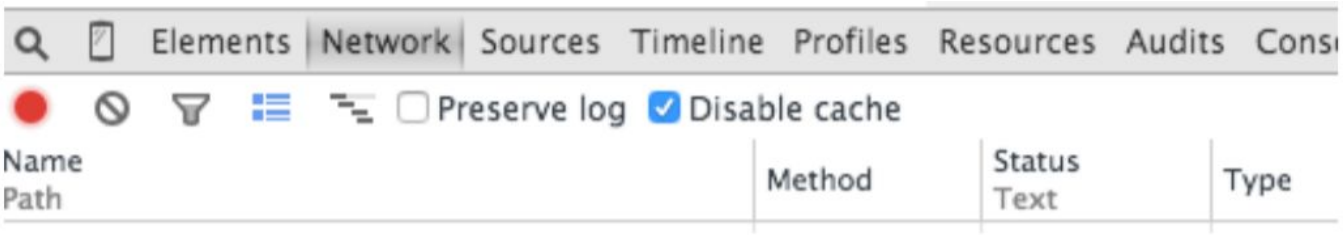
- How to undo?
- You have to support at least 10 steps undo stack
- Once an filter applied, how to revert the change?
- Why not restore the image? =D

Index Page

- Display images in 2x4 dimension
  - Sort based on completion time (NOT upload time!)
- Resized image
  - `-resize` or `-thumbnail` from IM
  - `max-width` / `max-height` of CSS
- Click the resized image can open the image in original size
- Remember to implement the resume button!

Disabling Cache

- Image may be cached in local browser
- If you perform undo then another operation quickly, the page may show old image from cache
  - Force reload (Ctrl+F5)
  - Disable cache in developer tools



  - Disable cache in HTML

```
<meta http-equiv="cache-control" content="max-age=0" />
<meta http-equiv="cache-control" content="no-cache" />
<meta http-equiv="expires" content="0" />
<meta http-equiv="expires" content="Tue, 01 Jan 1980 1:00:00 GMT" />
<meta http-equiv="pragma" content="no-cache" />
```

Final Reminder

- Testing multiple sessions / multiple files editing by different user
  - Using Chrome + Firefox simultaneously
  - Incognito mode (Chrome) / Private Browsing (Firefox)

- Testing resume
  - Terminate the browser (Cmd-Q for Mac)

Thank You