CSCI4140
Open-Source Software Project Development

Tutorial 4

5 Feb 16:50
Add disabling cache in appendix

5 Feb 17:30
Add hot deploy in appendix

Cookies and Sessions Management
Assignment 1
Submission Guideline

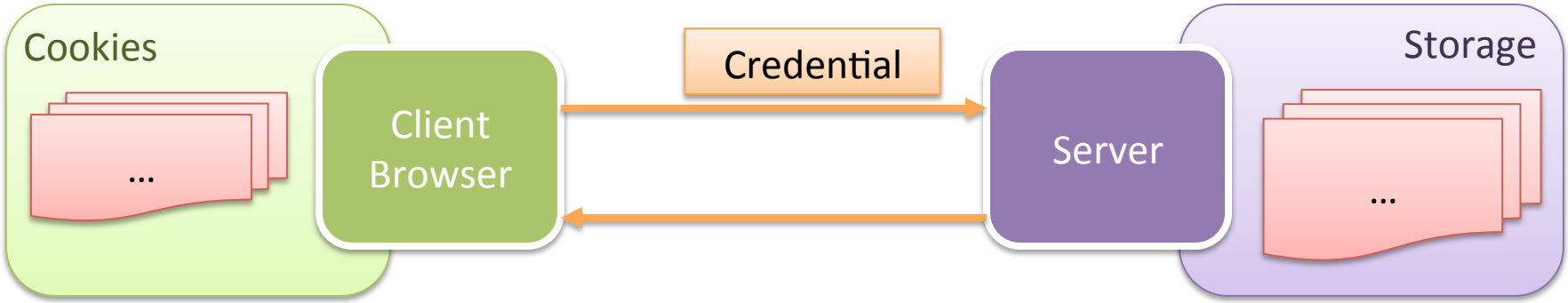
Cookies and Session

- **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

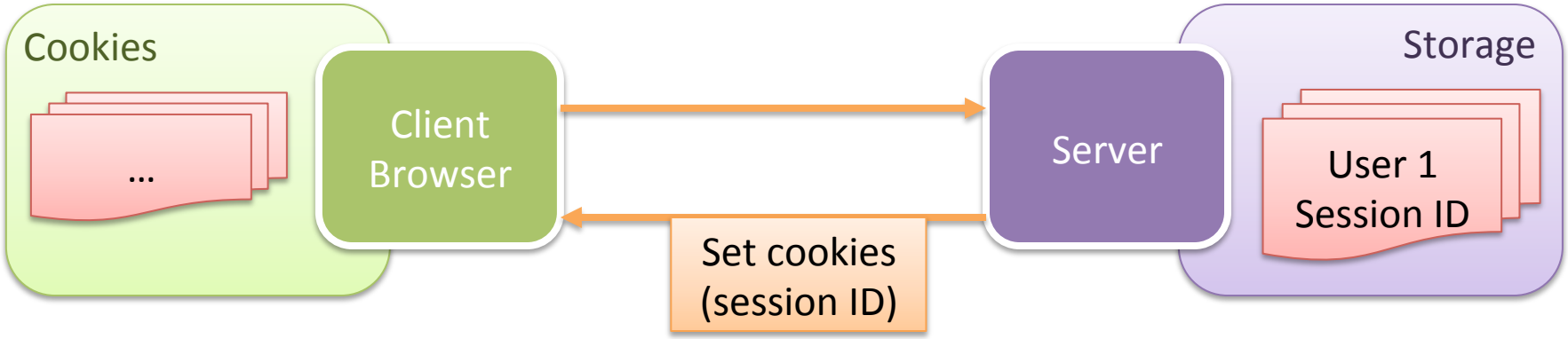
Cookies and Session

- Cookies and session work together
  - Client sent a request to server with credential



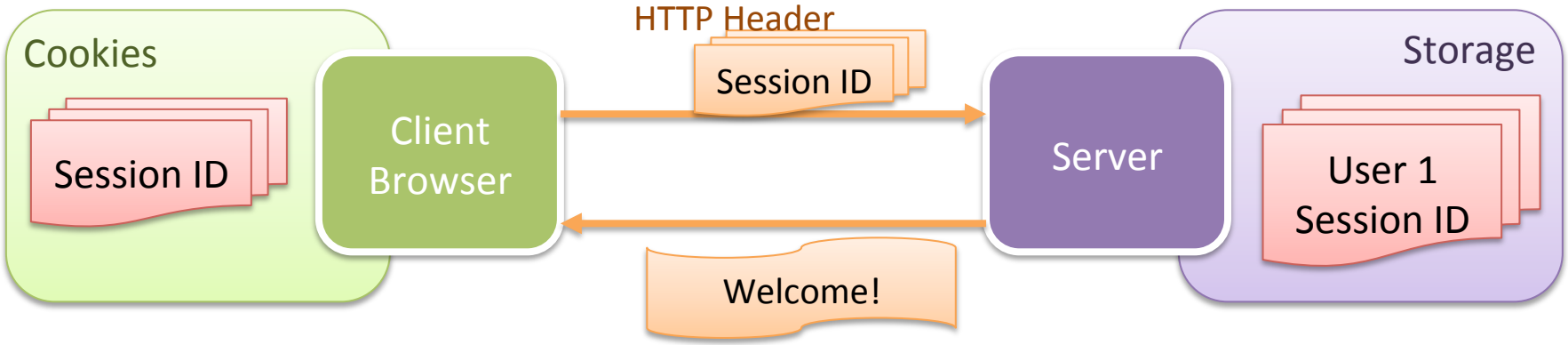
Cookies and Session

- 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

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



Cookies in Python

- Cookies should be included in HTTP header
  - Compute all necessary 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()
```

tutorial4/cookie.cgi

SimpleCookie as a Dictionary

- Access as directory

Key of cookie

```
cookieDict['session']
```

- 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
  - …

tutorial4/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
```

tutorial4/cookie.cgi

Print cookies in HTTP header

- Our common HTTP header is following:

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

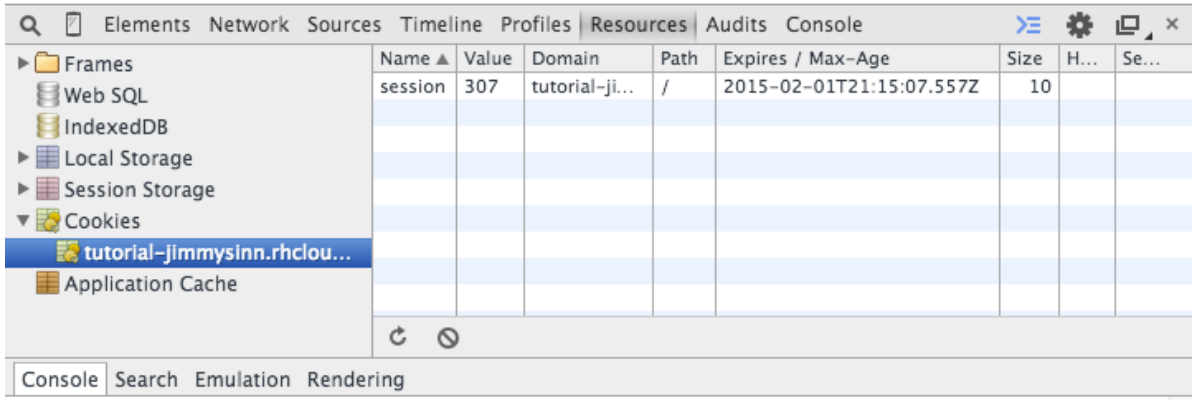
- To set cookie, print the SimpleCookie object before the delimiter

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

tutorial4/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

Modularize your code

- To modularize your code (separate into multiple files)
  - Name your python source file `<module>.py`
  - In your cgi (or main python source file), add `import <module>`
  - Every time you use functions / variables inside module, add `<module>.`

csci4140.py

```
course = 'csci4140'

def foo():
    print 'sosad'
```

main.py

```
import csci4140

csci4140.foo()

print csci4140.course
```

```
>>> import csci4140
>>> csci4140.foo()
sosad
>>> print csci4140.course
csci4140
```

  - `<module>.pyc`: bytecode for python's virtual machine
    - You can ignore it in git repository

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)
```

- Need more? 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

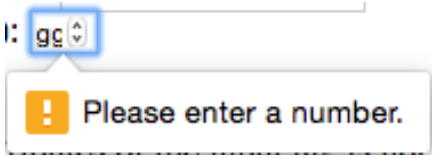
HINTS ON ASSIGNMENT 1

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

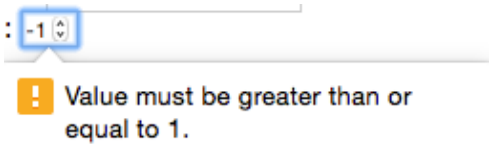


Image Upload (1)

- 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 (2)

- 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 (1)

- 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 (2)

- 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!

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)
- Use **environment variables** to avoid hardcode

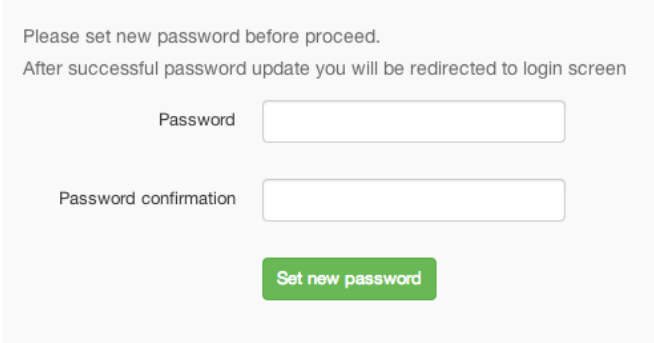
SUBMISSION GUIDELINE

Gitlab

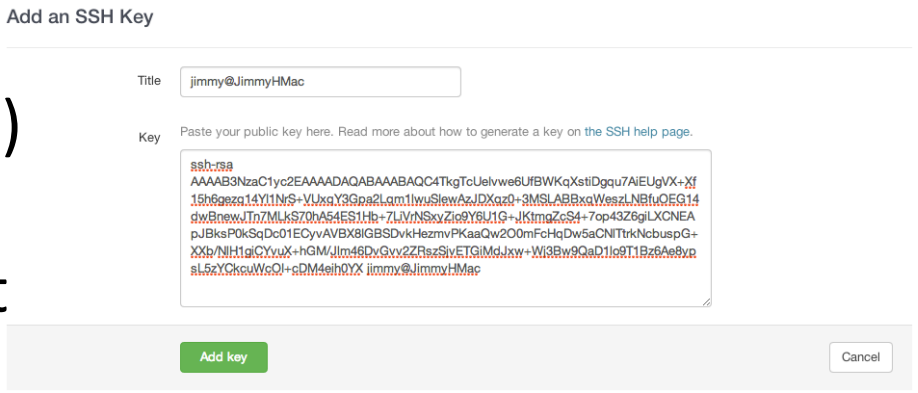
- We have setup a gitlab to receive your assignment
  - Github-like application
- Submit your assignment using `git push`
  - Via SSH or HTTPS
- Web Interface
  - https://pc89074.cse.cuhk.edu.hk/gitlab
  - Only accessible in CUHK network
    - CUHK VPN or CSE VPN

Web Interface

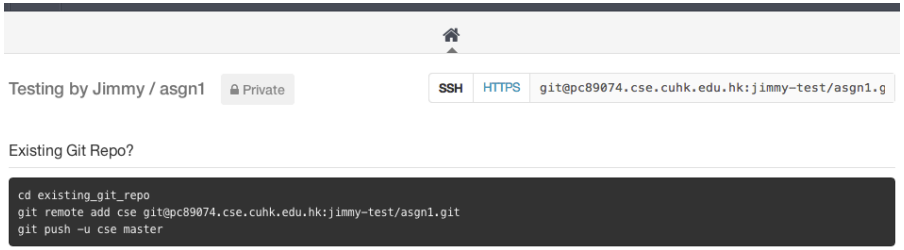
- Login using credential from our mail
  - Change password at first login
  - Re-login after changing password



- Add SSH key to gitlab
  - Profile Settings
    (Logo at the top-right corner)
    → SSH Keys
  - Just like you did in OpenShift



- Get URL to repository
  - HTTPS or SSH



Submit your Work

- Add one more remote repository to git

  `git remote add cse <URL>`

  – If you use HTTPS, please also disable certificate verification

- Push your assignment

  `git push -u cse master`

  – If you use HTTPS, type your username and password

```
✔ 17:59:33 jimmy@JimmyMBA ~/openshift/asg1pl  master git config http.sslVerify false
✔ 17:59:34 jimmy@JimmyMBA ~/openshift/asg1pl  master git remote add cse https://pc89074.cse.cuhk.edu.hk/gitlab/ltsinn/asgn1.git
✔ 17:59:39 jimmy@JimmyMBA ~/openshift/asg1pl  master git push -u cse master
Username for 'https://pc89074.cse.cuhk.edu.hk': ltsinn
Password for 'https://ltsinn@pc89074.cse.cuhk.edu.hk':
Counting objects: 1085, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (642/642), done.
Writing objects: 100% (1085/1085), 500.12 KiB | 0 bytes/s, done.
Total 1085 (delta 706), reused 676 (delta 437)
To https://pc89074.cse.cuhk.edu.hk/gitlab/ltsinn/asgn1.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from cse.
```

HTTPS only

Confirm your Submission

- No confirmation mail

- View the repository on web interface and check
  - Is **latest commit** match your desired submission?
  - **Browse code** for further comfirmation

Submission Requirement

- Submit your OpenShift repository used in development
  - We will simply restore your submission to another OpenShift application by `git push`
  - Include deploy script and files under `.openshift/` directory (created by OpenShift) in your submission
  - Do not create another repository for submission!

Submission Reminder

- Don't submit to wrong repository
  - Repository name: `asgn1` *(for assignment 1)*
  - Branch Name on remote repository: `master`
  - Only latest commit will be used for grading
    - Restoring to specific commit will leads to mark deduction
- Push may need some time
  - Don't submit at very last minute
- Try submission at anytime
  - Only latest commit will be counted

Finally …

- Deploy your code to OpenShift and test before submission
  - If you are not using OpenShift in development
- We will have demo for grading
  - Stay tune for registration announcement
- Ask question on Facebook group
  - Avoid Facebook message … maybe your classmates will have similar question with you

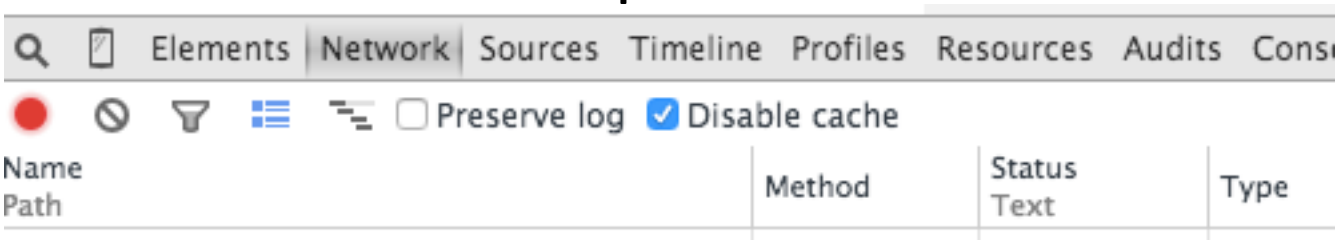
Enjoy the Assignment

Deadline: 12 Feb (Thu)

Btw, there will be tutorial next week ……

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" />
```

Hot Deploy

- Every time you push your code to OpenShift, it will restart the apache, mysql etc.
- If you want to skip the slow restart, add a marker to your repository

```
✔ 17:28:17 jimmy@JimmyMBA ~/openshift/tutorial/.openshift/markers master
touch hot_deploy
✔ 17:28:20 jimmy@JimmyMBA ~/openshift/tutorial/.openshift/markers master
git add hot_deploy
✔ 17:28:24 jimmy@JimmyMBA ~/openshift/tutorial/.openshift/markers master
git commit -am "Add hot deploy marker"
git [master 6410656] Add hot deploy marker
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 .openshift/markers/hot_deploy
✔ 17:28:38 jimmy@JimmyMBA ~/openshift/tutorial/.openshift/markers master
git push
```

```
remote: Not stopping cartridge mysql because hot deploy is enabled
remote: Not stopping cartridge perl because hot deploy is enabled
remote: Not stopping cartridge phpmyadmin because hot deploy is enabled
```