CSCI4140
Open-Source Software Project Development

Tutorial 4

Cookies and Sessions Management
More about Assignment 1
Submission Guideline

Update on 18 Feb 2014
Fix wording in installation script (pg 15)

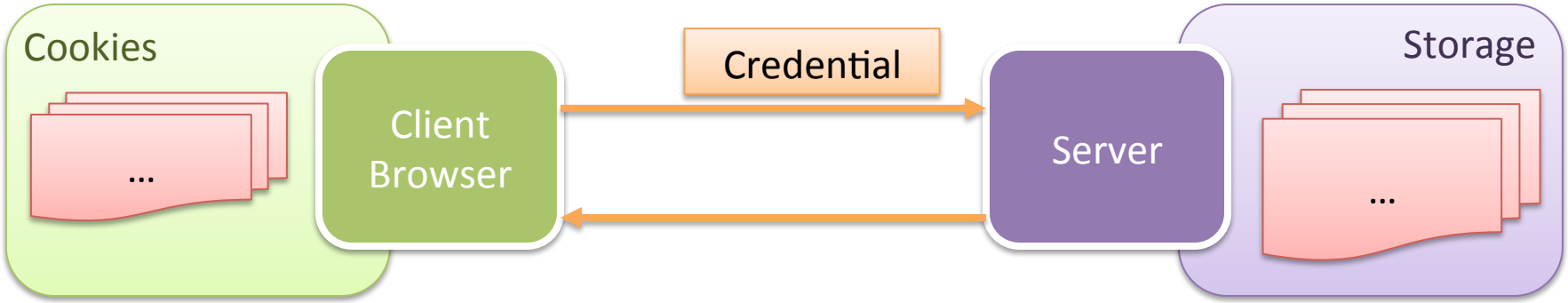
Update on 23 Feb 2014
Fix expires of cookies

Cookies and Session

- How to distinguish 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

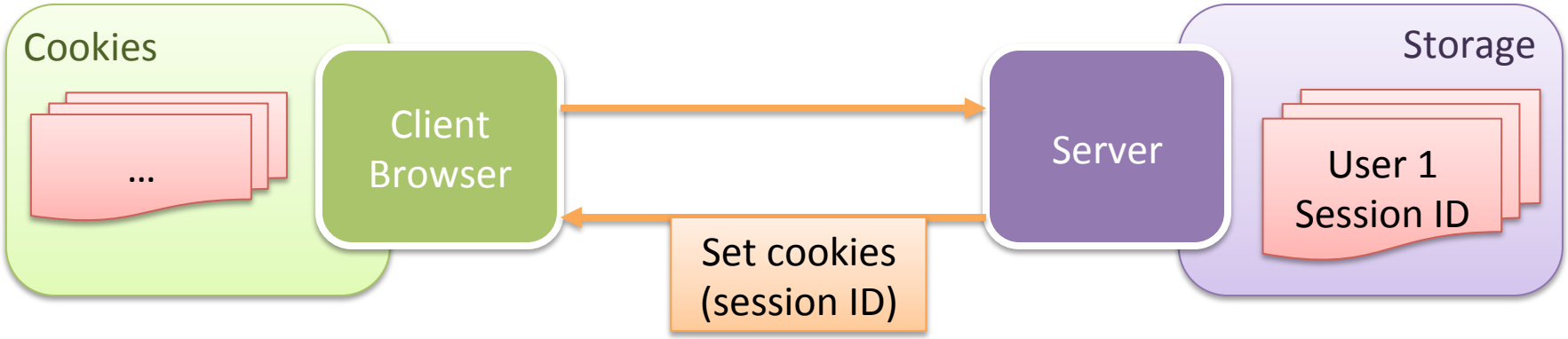
Cookies and Session

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



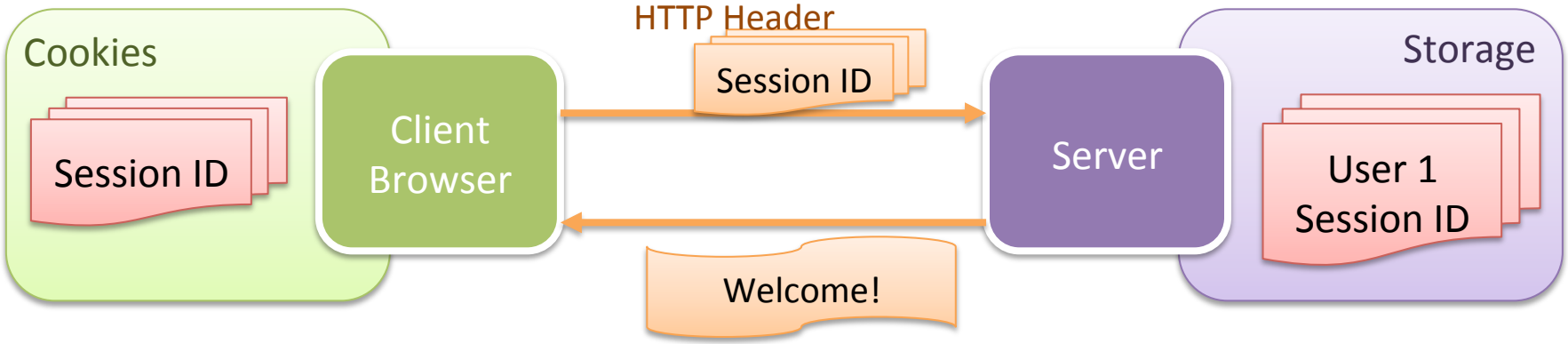
Cookies and Session

- 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) etc
  - Session ID embed in HTTP header of response to client



Cookies and Session

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



Cookies in Perl

- CGI module provides cookies management functions
  - Checking cookies
  - Embed cookies in header

More: http://perldoc.perl.org/CGI.html#HTTP-COOKIES

Perl: Cookies using CGI module

- Setting cookies
  - Create cookie object(s)

```
 6 $cookie1 = $q -> cookie(
 7     -name => 'user',
 8     -value => 'tywong',
 9     -expires => '+1h'
10 );
```

Please notice the 's' here
(Update on 23 Feb)

  - Embed cookie object(s) to header

```
18 print $q -> header(
19     -cookie => [$cookie1, $cookie2]
20 );
```

    - Applicable to standard header and redirection header

tutorial4/cookies_set.cgi

Perl: Cookies using CGI module

- Retrieve Cookies
  - Get from CGI object …

```
6 $val1 = $q -> cookie('user');
7 $val2 = $q -> cookie('session');
```

- Unset cookies
  - Use expire time of cookies

tutorial4/cookies_get.cgi

Session

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

Verifying Active Session

- Check cookie from client browser
- Match with entry stored in server
  - If matched, then this is active session
  - If not matched, ask user to login

Session Requirement in Assignment 1

- "kick-you-out"
  - Redirect user to login page if s/he is not logged in
- "welcome-back"
  - Redirect user to display panel if s/he is logged in
- "multiple-login"
  - More than one browser instance
- "logout"

More about Assignment 1

Modularize your code

- Include a file (library) to a script
  - Less copy-and-paste

```
require './include.pl';
```

- Remember add a return statement at the end of included file

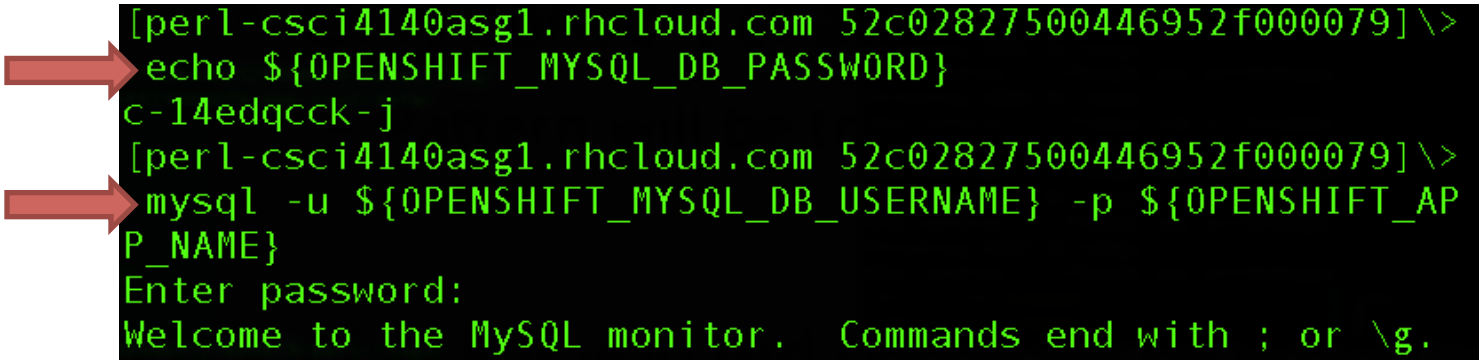
```
# Your library functions

return 1;
```

- e.g. Cookies checking, Database connection, …

Reserved Characters

- Substitute reserved characters in description
  - Use regular experssion
- Only convert characters when upload
  - Pattern will be translated by browser when browsing
- Check if you did correctly by string in database
  - Use mysql command instead of phpMyAdmin!



More: http://www.w3schools.com/charsets/ref_html_ascii.asp

Installation Script: Reset your Album

- Provide a page to re-initialize your album

- Executing the page will reset the album to empty
  - Remove all photo from database (and storage)
  - Clear sessions
  - Remove all users
  - At least one user exist ~~left~~
  - Create any needed directory in persistent directory

Some more Hints (1)

- **File Size** Limit
  - Do not believe `$CGI::POST_MAX`
- **File Type / Extension** Checking
  - Check extension match with actual file type
  - Not a valid filename if mismatch
- How should photos and thumbnails **stored** ?
  - Up to you
  - Don't mix those up!

Some more Hints (2)

- **Filename** will only consist lower cases / digits / underscores
  - When testing, you can use `lc` to convert filename to lowercase
- **Animated GIF** is not required
- Use HTML(5) attribute to **validate** other input
  - e.g. Photo description length, Album array dimension etc.
  - *Server side validation* for file upload only

Some more Hints (3)

- Login **Credential**
  - Store in database
  - Hardcode in script is also acceptable

- Testing **Multiple Login**
  - Using Chrome + Firefox simultaneously
  - Igconito mode (Chrome) / Private Browsing (Firefox)

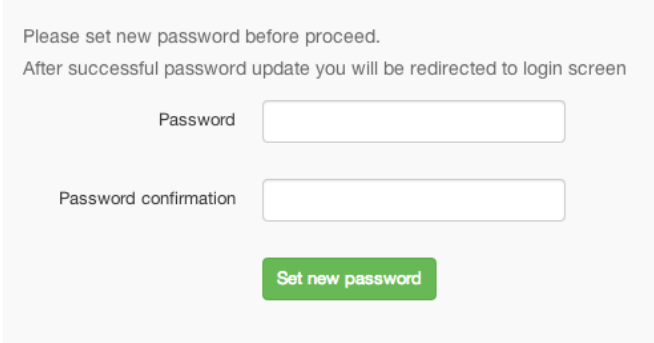
SUBMISSION GUIDELINE

Gitlab

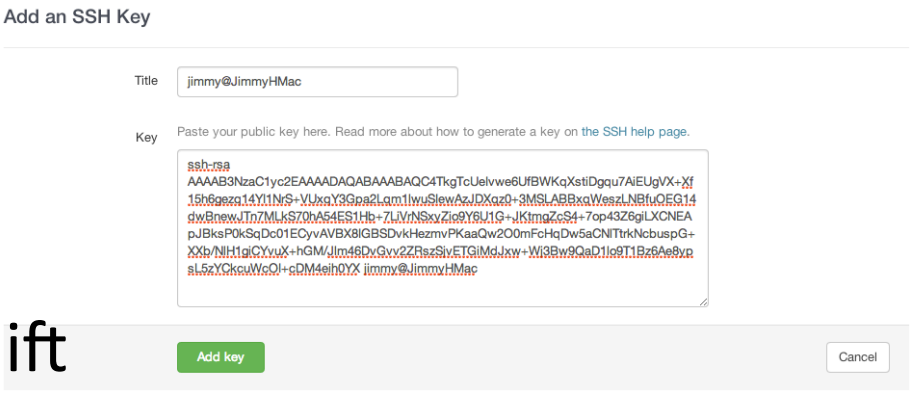
- We have setup a gitlab to receive your assignment
  - Github-like application
- Submit your assignment using git push
  - Via SSH
- Web Interface
  - https://pc89074.cse.cuhk.edu.hk/gitlab
  - Only accessible in CSE network
    - CSE VPN

Web Interface

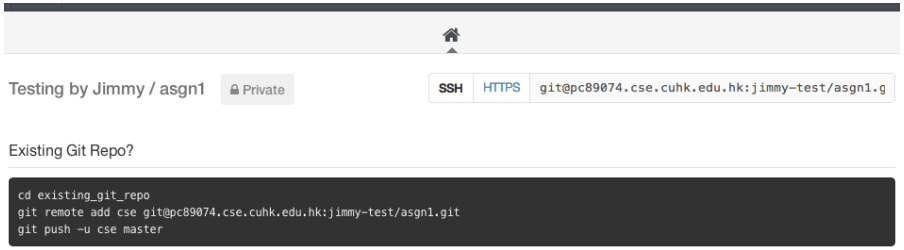
- Login using credential from our mail
  - Change password at first login
  - Re-login after changing password



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



- Get URL to repository



Submit your Work

- Add one more remote repository to git
  - ssh-config for easy access (refer to tutorial 1)
  - Use ProxyCommand when outside department network
- Push your assignment

```
[02:55:39] jimmy@JimmyHMac asgn1 $ git status
# On branch master
nothing to commit, working directory clean
[02:55:42] jimmy@JimmyHMac asgn1 $ git remote add submit 4140submit-test:jimmy-test/asgn1.git
[02:55:49] jimmy@JimmyHMac asgn1 $ git remote
origin
submit
[02:55:52] jimmy@JimmyHMac asgn1 $ git push -u submit master
Counting objects: 40, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (35/35), done.
Writing objects: 100% (40/40), 30.75 KiB | 0 bytes/s, done.
Total 40 (delta 1), reused 40 (delta 1)
Killed by signal 1.
To 4140submit-test:jimmy-test/asgn1.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from submit.
```

Confirm your Submission

- No mail will be sent

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

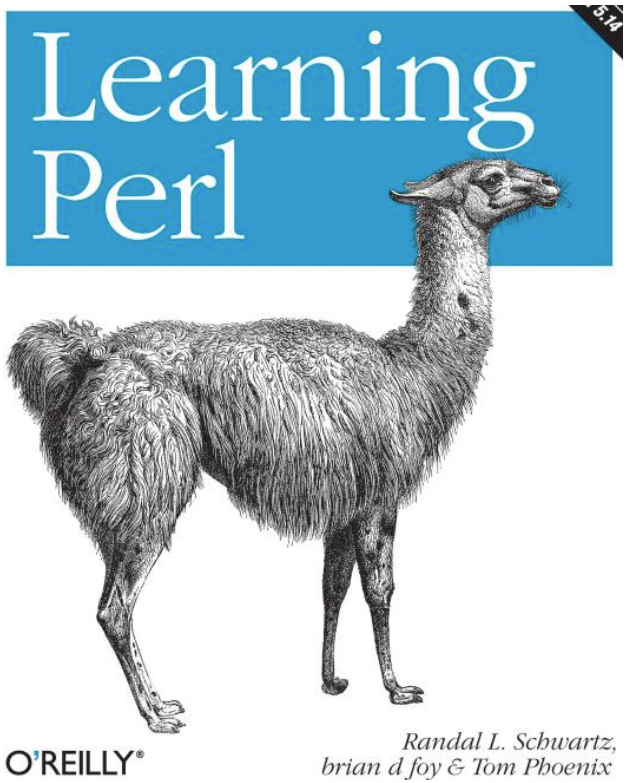
Submission Reminder

- Don't submit to wrong repository
  - Repository name: `asgn1`
  - Branch Name on remote repository: `master`
  - Only latest commit will be counted
- Push may need some time
  - Don't submit at very last minute
- Try submission at anytime
  - Only latest commit will be counted

Finally …

- Check your assignment by checklist!
- Deploy your code to OpenShift and test before submission
- Ask question on Facebook group

ENJOY THE ASSIGNMENT



Deadline: 28 Feb (Fri)

Bonus: SSH ProxyCommand

- Connect to servers in CSE network via CSE Gateway

- ProxyCommand + ssh forward

```
Host csegw
     hostname gw.cse.cuhk.edu.hk
     user ltsinn
     identityfile ~/.ssh/id_rsa

Host 4140submit-test
     hostname pc89074.cse.cuhk.edu.hk
     user git
     identityfile ~/.ssh/4140gitlab
     ProxyCommand ssh csegw -W %h:22
```