

CSCI4140

Open-Source Software Project Development

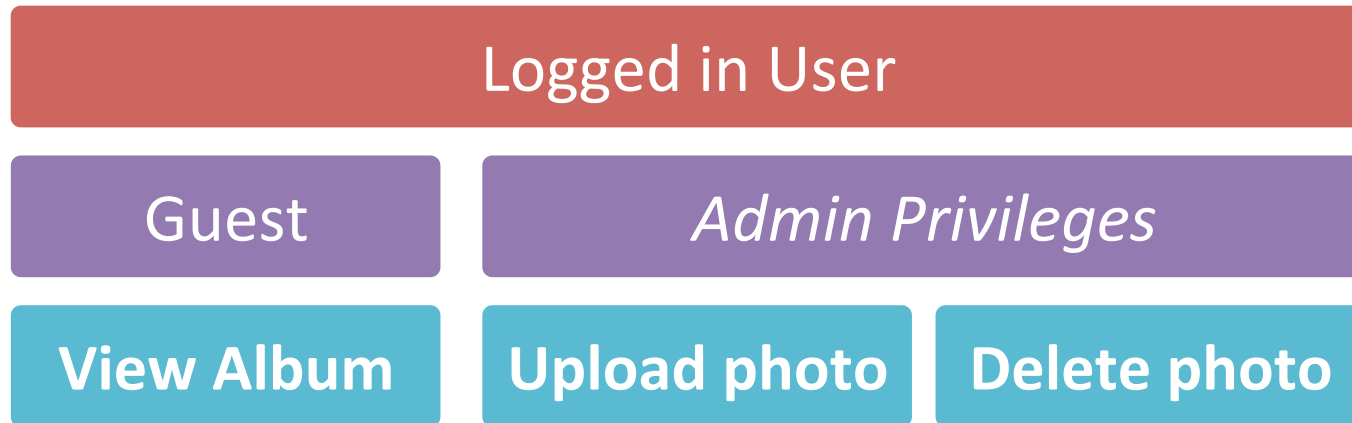
Tutorial 2

Assignment 1
HTML and Perl
File Upload: Part (1)

Assignment 1 – Overview

- Web-based Image Management System
 - a.k.a. Photo Album

- Features



Assignment 1 – Requirement

- Languages
 - HTML
 - Perl
 - CGI & DBI module

NO JavaScript is allowed

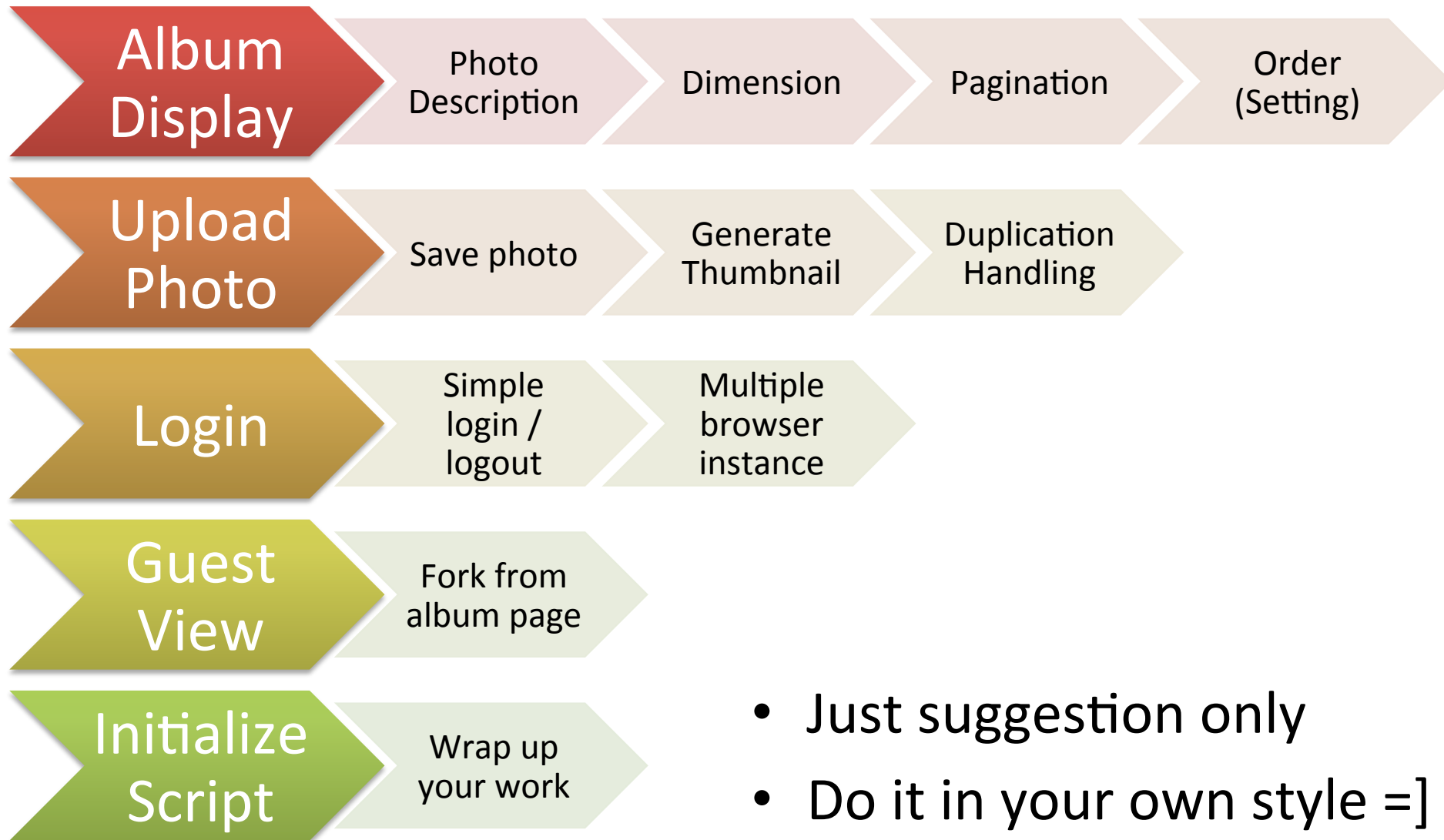
- Permanent Storage
 - MySQL database
 - Photo storage directory

Assignment 1 – Testing and Demo

- OS
 - Mac, Linux / Windows
 - Actually it does not matter ...
- Browser
 - Mozilla Firefox 4.0 or above; or
 - Google Chrome 32.0 (latest stable ver.)



Assignment 1 – Recommended Workflow



- Just suggestion only
- Do it in your own style =]

WEB PROGRAMMING

HTML and Perl

HTML – Basics

- Browser retrieve HTML files (and related files) from web server
- Browser then render the HTML code to a page
 - Show text with style
 - Show images
 - Enable user input

HTML – Elements

- HTML file contain HTML elements



- Content can be text, HTML elements or mixture
- Note the order

`<tag1>Hello <tag2>World</tag2>!</tag1>`

- Element with empty content
`
`

HTML – Attribute

- Additional information to an element
- Name-value pairs: `name="value"`
`Index Page`



HTML – Basics

- Some common HTML tags

HTML Tags	Usage
<code><html></code>	Define root of the HTML document
<code><body></code>	Define start of page body
<code>
</code>	Line break
<code><table></code>	Define a table (use with <code><tr></code> and <code><td></code>)
<code><div></code>	A division / section of page
<code></code>	Show an image (src attribute to define location)

- These tags just define page structure
 - Use `class` / `id` attribute and CSS to define style
- How a page accept user input and send to server ?

For more: <http://www.w3schools.com/tags/>

HTML – Forms

- Send data (user input) to server
- Server receive data and process

<code><form></code>	Declare HTML a form	
<code><input></code>	Checkbox, text box ... (type attribute)	<input type="text"/>
<code><button></code>	Button	<input type="button" value="Button"/>
<code><textarea></code>	Text input field	<div>Content</div>
<code><select></code> <code><option> ...</code>	Drop-down list	<div>Option 1 ▾</div>

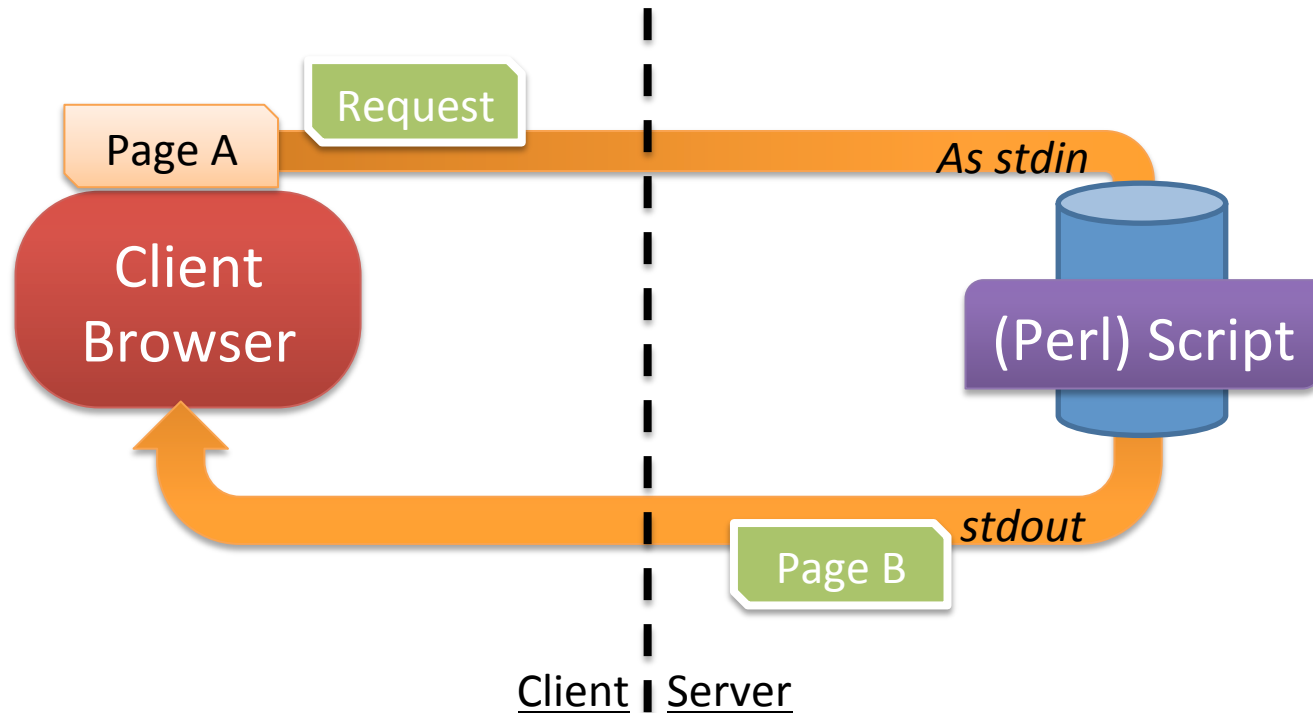
- Use name attribute to distinguish and access from script

For more: http://www.w3schools.com/html/html_forms.asp

[tutorial2/html_form.html](#)

Dynamically Generating Webpage

- Generating HTML page using script



- Content of page can be controlled by script

Perl: CGI Module

- Library for handling CGI request and response
- Functions
 - Receive form request
 - Handle file upload
 - Access cookies
 - Process HTTP header
 - ...

Perl: CGI Module

- **Function 1:** Print HTTP header
 - First, load the CGI module

```
use CGI;
```

- Create a CGI object

```
$q = CGI -> new;
```

- Print the HTTP standard header

```
print $q -> header();
```

See: <http://perldoc.perl.org/CGI.html#CREATING-A-STANDARD-HTTP-HEADER>

[tutorial2/cgi_example1.cgi](#)

Perl: CGI Module

- **Function 2: Page Redirection**

- Print the redirection header using

```
print $q -> redirect('target-page.html');
```

- You only need redirection header OR standard header, but **not both**
 - How to do conditional redirection?

See: <http://perldoc.perl.org/CGI.html#GENERATING-A-REDIRECTION-HEADER>

`tutorial2/cgi_example2.cgi`

Perl: CGI Module

- **Function 3:** Retrieve parameters
 - Get the GET and POST parameters using `param()`

```
$q -> param( 'foo' );
```

- In rare case, you may use `url_param()` for GET parameters ...

See: <http://perldoc.perl.org/CGI.html#FETCHING-THE-VALUE-OR-VALUES-OF-A-SINGLE-NAMED-PARAMETER>
`tutorial2/cgi_example3.cgi`

Perl: CGI Module

- Powerful CGI module
 - Save you time on tedious coding
- More can be achieved using CGI module

For more: <http://perldoc.perl.org/CGI.html>

PHOTO UPLOAD

File Upload and Image Handle

- File upload using HTTP POST request
- We need (at least) two page to handle upload
 - Form accepting user's input and send request
 - Process user's request on server, and generate result page (e.g. Upload finish confirm)
 - Duplication handling page ?

- HTML Form

```
4 <form enctype="multipart/form-data" action="upload.cgi" method="POST">
5   Choose an image (.jpg .gif .png):
6   <input type="file" name="pic" accept="image/gif, image/jpeg, image/png" />
7   <br />
8   Description (50 bytes max)
9   <input type="text" name="desc" maxlength="50" /><br />
10  <input type="submit" value="Upload" />
11 </form>
```

Choose an image (.jpg .gif .png): No file selected.

Description (50 bytes max)

- Using encoding type multipart/form-data
- Post request to processing script (upload.cgi)

- HTML Form

```
4 <form enctype="multipart/form-data" action="upload.cgi" method="POST">
5   Choose an image (.jpg .gif .png):
6   <input type="file" name="pic" accept="image/gif, image/jpeg, image/png" />
7   <br />
8   Description (50 bytes max)
9   <input type="text" name="desc" maxlength="50" /><br />
10  <input type="submit" value="Upload" />
11 </form>
```

Choose an image (.jpg .gif .png): image.png

Description (50 bytes max)

- File input type
- Showing only specific file type for user to select
 - You still have to check if content match extension*

[tutorial2/upload_form.html](#)

- HTML Form

```
4 <form enctype="multipart/form-data" action="upload.cgi" method="POST">
5   Choose an image (.jpg .gif .png):
6   <input type="file" name="pic" accept="image/gif, image/jpeg, image/png" />
7   <br />
8   Description (50 bytes max)
9   <input type="text" name="desc" maxlength="50" /><br />
10  <input type="submit" value="Upload" />
11 </form>
```

Choose an image (.jpg .gif .png): image.png

Description (50 bytes max)

- Using `maxlength` to restrict length of **description**
 - Don't confuse with file size limit!

- How server handle received file ?
 - Apache saved it to somewhere
- Uploaded file is saved by apache already
 - Access as file stream

```
$q -> upload('pic')
```

- Get the filename using

```
$q -> param('pic')
```

- Write the data from stream to file

- Use **persistent directory** to save uploaded images
 - Local changes (include uploaded file) will be flushed when you push your code
- Get the path to persistent storage from environment variable

OPENSIFT_REPO_DIR	Repository directory path
OPENSIFT_DATA_DIR	Persistent directory path
OPENSIFT_TMP_DIR	Temporary directory path

- Get the path as following:

```
$datadir = $ENV{'OPENSIFT_DATA_DIR'}
```

More: <https://www.opensift.com/page/opensift-environment-variables>

- Persistent directory cannot access directly from browser
- Create symbolic link from public directory
 - Public directory: `${OPENSIFT_REPO_DIR}/perl`
- When to create symbolic link ?
 - By running initialize script ?
 - Still a local change
 - Link will be flushed when pushing

More: <https://www.openshift.com/developers/deploying-and-building-applications>

- Action Hooks
 - Scripts run before / after deployment (or other events)
 - Kind of initialize script
 - Changes done will still be flush in next deployment
 - But the script will run again
- Detail procedure

```
[02:25:43] jimmy@JimmyHMac action_hooks $ pwd
/Users/jimmy/tutorial/asg1/.openshift/action_hooks
[02:25:44] jimmy@JimmyHMac action_hooks $ cat > deploy
ln -s ${OPENSIFT_DATA_DIR} ${OPENSIFT_REPO_DIR}/perl/data
[02:26:25] jimmy@JimmyHMac action_hooks $ chmod +x deploy
[02:26:35] jimmy@JimmyHMac action_hooks $ git add deploy
```

More: http://openshift.github.io/documentation/oo_user_guide.html#action-hooks

Photo Upload ... To be continue

- In this tutorial ...
 - HTML Form to accept file
 - Write accepted file to persistent directory in server
 - Access your uploaded file on server **from browser**
- More in next tutorial
 - Check file type
 - Thumbnail generation
 - Duplication handling

Debug Tips

- Check if file saved by your script
 - SSH to OpenShift server
 - Access persistent directory path by
`cd ${OPENSHIFT_DATA_DIR}`
- Detail of SSH into OpenShift will be discussed in next tutorial
 - SSH config in tutorial 1 (OpenShift)

Tutorial Preview

- Tutorial 3 (6 Feb)
 - File Upload: Part (2)
 - DBI Module in Perl
 - Debugging Perl scripts
- Tutorial 4 (13 Feb)
 - Cookies and Session Management
 - Assignment 1 Hints
- Preview of Slides:
<https://appsrv.cse.cuhk.edu.hk/~ltsinn/csci4140/>
 - Slides will keep updating

Recommendation

- W3School: <http://www.w3schools.com/html>
 - Provided tutorial on HTML (and more ...)
 - Just try their examples
 - Clear reference
- Inspect Elements of Browser
 - Built in for Chrome
 - Use Firebug (<https://getfirebug.com/>) for Firefox
 - Allow instant change of HTML (/JavaScript) code

END

Question ?

- *Jimmy Sinn (Office: SHB 115 / Email: [Itsinn\(at\)cse.cuhk.edu.hk](mailto:Itsinn@cse.cuhk.edu.hk))*
- *Ask on Facebook group*

Perl: Here Document (Bonus)

- Here-Document
 - Multi-line strings
- Handy when hardcoding long string / HTML code
- Variables inside can be interpolated

Reference: <http://perlmaven.com/here-documents>

Perl: Here Document (Bonus)

- Example

```
3 print << "___END_MSG" ;  
4 String is start at next line ...  
5 $my_var <- can be interpolated!  
6 ...  
7 And ends when I see the next line:  
8 ___END_MSG |  
9  
10 # Continue my script ...
```

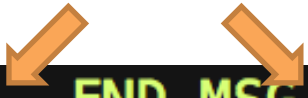
Whole
string
(document)

- An **arbitrary string** followed by <<
- Document starts at the next line
- The **string** used to specify the end of document
 - No indentation!

Reference: <http://perlmaven.com/here-documents>

Perl: Here Document (Bonus)

- Example



```
3 print << '___END_MSG' ;
4 String is start at next line ...
5 $my_var <- don't interpolated
6 ...
7 And ends when I see the next line:
8 ___END_MSG
9
10 # Continue my script ...
```

- Using single quote to NOT interpolate variables inside
- Double quote (or no quote, not recommended) will interpolate included variables

Reference: <http://perlmaven.com/here-documents>

Perl: Here Document (Bonus)

- Assignment of here document to variable

```
3 my $var = <<"__END_MSG";  
4 ...
```

- The indentation is also included in message
 - If you don't want those space ...
 - Use regular expression to filter
 - Do not indent for the message
 - Or just leave them (e.g. HTML code)

Reference: <http://perlmaven.com/here-documents>