File Uploads and Cookies

Pat Morin

COMP 2405

Carleton
UNIVERSITY
Canada's Capital University

Outline

- File upload
- Cookies

Carleton
UNIVERSITY
Canada's Capital University

File Uploads

- CGI scripts can accept file uploads
- This is done using an `input` with
  - `type "file"`
  - `action "POST"`
  - encoding type `"multipart/form-data"`

```
<form method="POST"
  enctype="multipart/form-data"
  action="cgi-bin/fup.cgi">
 <p>
   File to upload:
   <input type="file" name="upfile">
 </p>
 <p><input type="submit" value="Press to
upload">
</form>
```

File Upload (Cont'd)

- On the server side, the contents of the uploaded file will be assigned to the name "upload"

```
#!/usr/bin/perl
require "./cgi-lib.pl";

my %in; ReadParse(\%in);
print("Content-type: text/html\n\n");

print <<EOT;
<html><head><title>File Upload Page</title></head>
<body><pre>
  $in{'upfile'}
</pre></body>
EOT
```

File Upload Parameters

- Some global variables in `cgi-lib.pl` control various parameters of the file upload
  - `cgi_lib::writefiles` – the name of a directory in which to store uploaded files
  - `cgi_lib::maxdata` – the maximum file size to accept
- When `cgi_lib::writefiles` is the defined, the value assigned to the upload control name is the file name, not its contents

Carleton UNIVERSITY
Canada's Capital University

Cookies

- A *cookie* is a piece of information that a web server stores on a user's computer that is retrieved later
- Cookies can be used to get around the fact that HTTP is a *connectionless* protocol
- The most common use of cookies is to identify a particular user amongst a set of users
  - Customized pages
  - Automatic logins ("Remember Me" buttons)
  - Visitor tracking and statistics systems
  - Site optimization software

Carleton UNIVERSITY
Canada's Capital University

Cookies – Pros and Cons

- Pros
  - An easy way to keep track of a user among many users or over several visits
- Cons
  - Requires a browser that accepts cookies
  - Requires a user willing to accept cookies

Cookie Properties

- Cookies are very simple:
  - name – the name of the cookie (like a variable name)
  - value – a 4000 character string
  - expiry time (optional)
  - path and domain (optional)
- A website sets a cookie's name and value and can later refer to it by name
  - Other websites can not access the cookie (even with the name)

The Expiration Date

- Websites may refer to two kinds of cookies
- *Session cookies* have not expiration time
  - These cookies are deleted when the browser session ends (the browser is closed by the user)
- *Persistent cookies* have an expiration time
  - These cookies are deleted (or stop getting sent) when the expiration time is up

Cookie Security

- Although given a lot of bad press, cookies do not present much of a security problem for users
  - They allow the tracking of a particular user through a site, possibly over multiple visits
- Web developers should assume that the name and value of a cookie are public information
  - Do not store long-term secrets in a cookie
- Real secrets should be kept in a safe spot on the server

Setting a Cookie

- Cookies are set with the Set-Cookie field in an HTTP response header

```
HTTP/1.1 200 OK
Date: Mon, 26 Mar 2007 14:46:52 GMT
Server: Apache/2.2.3 (Fedora)
X-Powered-By: PHP/5.1.6
Set-Cookie: name=Value; expires=Mon, 26-Mar-2007 15:46:52 GMT
Content-Length: 307
Connection: close
Content-Type: text/html; charset=UTF-8

...
```

Setting a Cookie (in PHP)

- Use the `setcookie` function

```
<?php
  setcookie("name", "Value", time()+3600);
?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
...
```

Setting a Cookie (in Perl, using Date::Manip)

```
my $expires=DateCalc("today", "+ 60minutes");
$expires = UnixDate($expires, "%a, %d-%b-%Y %H:%M:%S %z");
print("Set-Cookie: name=Value; expires=$expires\n");
print("Content-type: text/html\n\n");
...
```

Receiving a Cookie

- Once a cookie is set, the user agent will send the cookie with every request when they visit the site

```
GET /~morin/cookies/set.php HTTP/1.1
Host: localhost:5678
User-Agent: Mozilla/5.0 (X11; ...
...
Keep-Alive: 300
Connection: keep-alive
Cookie: name=Value

...
```

Receiving a Cookie (in PHP)

- Use the $_COOKIE variable

```
...
<p>This web page has set a cookie
named 'name' with data value
<q><?php
  if (isset($_COOKIE["name"])) {
   echo $_COOKIE["name"];
  } else {
   echo "undefined";
  }
?></q>
...
```

Carleton UNIVERSITY
Canada's Capital University

Receiving a Cookie (in Perl)

```
sub ParseCookies(\%) {
  my $cookies = shift;
  my $cookie_string = $ENV{'HTTP_COOKIE'};
  my @pairs = split(/[\|; ]/, $cookie_string);
  for (my $i = 0; $i < scalar(@pairs); $i++) {
    if (my ($name, $value) = $pairs[$i] =~ /(\w+)=(\w+)/) {
      $cookies->{$name} = $value;
    }
  }
}

my %cookies;
ParseCookies(\%cookies);
my $name = $cookies{'name'};
```

Deleting a Cookie

- To delete a cookie, simply set the cookie with an expiry time in the past

```
HTTP/1.1 200 OK
Date: Mon, 26 Mar 2007 14:46:52 GMT
Server: Apache/2.2.3 (Fedora)
X-Powered-By: PHP/5.1.6
Set-Cookie: name=Value; expires=Mon, 26-Mar-2007 11:46:52 GMT
Content-Length: 307
Connection: close
Content-Type: text/html; charset=UTF-8

...
```

Deleting a Cookie (in PHP)

```
<?php
  setcookie("name", "", time()-3600);
?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
...
```

Deleting a Cookie (in Perl, using Date::Manip)

```
my $expires=DateCalc("today", "- 60minutes");
$expires = UnixDate($expires, "%a, %d-%b-%Y %H:%M:%S %z");
print("Set-Cookie: name=Value; expires=$expires\n");
print("Content-type: text/html\n\n");
...
```

Other Cookie Properties

- Cookies can also have two other properties
  - path
  - domain
  - A browser should only send a cookie when the path and domain match the document being accessed
- The example below sets a cookie for
  - cg.scs.carleton.ca/~morin
  - www.scs.carleton.ca/~morin/index.html
- But not
  - www.scs.carleton.ca/index.html

Carleton UNIVERSITY
Canada's Capital University

```
Set-Cookie: name=newvalue; expires=date; path=/~morin;
domain=.carleton.ca
```

Cookies in JavaScript

- The document.cookie property is a pseudo-variable that allows for getting and setting cookies
- Setting a cookie:
  - document.cookie = "name=value; expires=date; path=/; domain=.scs.carleton.ca"
- Reading all cookies:
  - var pairs = document.cookie.split(';');
- Note: this modifies the browser's cookie database only

Application: Login Cookies

- Keeping track of a user name
  - When processing the login screen (a form) set two session cookies:
    - `username` whose value is the user name
    - `secret` whose value is a randomly generated string
  - Store the association `username => secret` on the server
  - On all subsequent pages, use `username` to identify the user and `secret` to make sure this is not a scam

Login Cookies (Cont'd)

- The secret cookie should have a short lifetime
  - It can be re-set every time the user visits another page or reloads the current page
  - The 'remember me' login option is implemented by making these cookies into persistent cookies

Application: User Tracking

- Knowing how users navigate through a site is information that can be used to improve the site
- **Problem:** Many users using the site at once and no login required
- Solution: On each access:
  - Check if the cookie `UID` is set
    - If not then set it to a new random number
    - If yes, then we know which user this is and can store this info
- We can now sort all pages accesses by UID to determine access patterns

Carleton UNIVERSITY
Canada's Capital University

Alternatives to Cookies

- Cookies are handy but there are other ways of achieving the same goal
- On CGI sites, we can modify URLs and/or add hidden fields to forms

```
<a href="page.cgi?username=patrick&secret=ks...">
  Proceed to page
</a>
```

Cookies and Security

- A web browser only sends a cookie to the domain that set that cookie
  - or that delivered a script that set the cookie
- A JavaScript script can only read and set cookies set by the same domain that produced the script
- This should mean some amount of privacy
  - A webserver *X* can monitor your actions over the domain of *X* (which it could also do without cookies)
  - A webserver *X* can not see where you go when you leave the domain of *X*
  - This works as long as we don't have many domains whose pages contain content (e.g., images) from a common site

The DoubleClick Trick

- Some companies host content for many domains
  - In the case below, adsite.com can set a cookie that uniquely identifies you
  - Everytime you visit a page that has an adsite.com image your browser loads the image and sends
    - 1. your unique identifier (cookie)
    - 2. The referer (the page you are looking at)

```
http://x.com/page.html contains:
<img src="http://adsite.com/mxadd.gif">
```

```
http://y.com/page.html contains:
<img src="http://adsite.com/myadd.gif">
```

Summary

- Cookies are a small amount of information
  - Server stores a cookie on the client
  - Client sends the cookie to the server with every subsequent request
- Cookies allow for
  - Username/password authentication
  - Customized web page
  - Web server statistics gathering
  - Tracking usage patterns across site (can be bad)
- Disclaimer
  - Some uses of cookies may be illegal in some districts
  - Consult your local lawyer