

Maintaining State

Eleni Stroulia

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

1

Techniques

- cookies
- URL rewriting
- hidden form variables
- server-side state files
- CGI Side Includes
- Netscape Persistent Cookies

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

2

Techniques

- cookies
 - URL rewriting
 - hidden form variables
 - server-side state files
 - CGI Side Includes
 - Netscape Persistent Cookies
- A cookie is a small chunk of data that a server can include in a response for storage by the client.
–<http://www.ietf.org/rfc/rfc2109.txt>
 - Each time the client contacts a server, it includes in the request HTTP headers all of the previously-stored cookies it has received from that server.
 - Session state can be stored in a cookie

Techniques

- cookies
 - URL rewriting
 - hidden form variables
 - server-side state files
 - CGI Side Includes
 - Netscape Persistent Cookies
- URL rewriting is the technique of encoding every URL on a served page to include client-side session state.
 - Any time the browser performs an HTTP GET to the server, the server receives the client-side state as an argument (in the URL).
 - The server may then parse the URL to extract the state and use it to perform the requested service.

Techniques

- cookies
- URL rewriting
- hidden form variables
- server-side state files
- CGI Side Includes
 - A server can encode client-side session state in "hidden" form variables
–<INPUT TYPE="HIDDEN" ... >.
 - These values are included in subsequent HTTP requests from the client, and used by the server as the session state.
 - This mechanism can be encapsulated in a JSP tag to make the programming easier.
- Netscape Persistent Cookies

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

2

Techniques

- cookies
- URL rewriting
- hidden form variables
- server-side state files
- CGI Side Includes
- Netscape Persistent Cookies
 - This is a mechanism by which we embed special tags into the HTML document that pass CGI variables invisibly.

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

2

The session maintenance problem

- Most Web applications enable a browsing session, where a given http transaction remembers data from a previous http transaction
- The middleware (web server) does NOT keep track of data during a browsing session.
- It is up to the application to maintain its own data between transactions in a session.

What's in a “state”

- *Session state data* \sim data which a Web application maintains to keep track of the state of a browsing session
 - relevant to a single session (current shopping cart)
 - different from permanent data (personal data, i.e., name, , credit-card number, address, shopping history)

Hidden variables

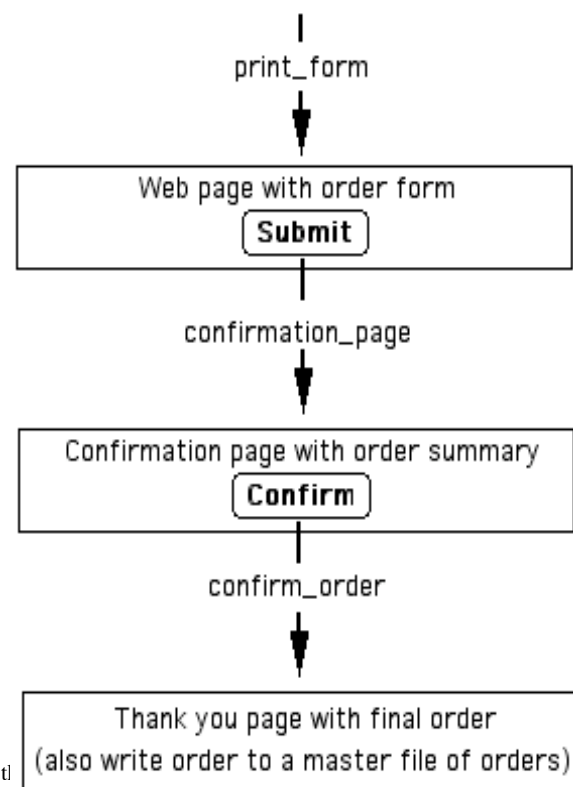
state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

5

pizza3.cgi: a one-step preservation of application state

- Most online order forms give an order summary page, which also serves as an order confirmation page.
- The transaction diagram to the right depicts a Web application which gives such an intermediary confirmation page.
- The arrows represent the three distinct http transactions of the application logic.

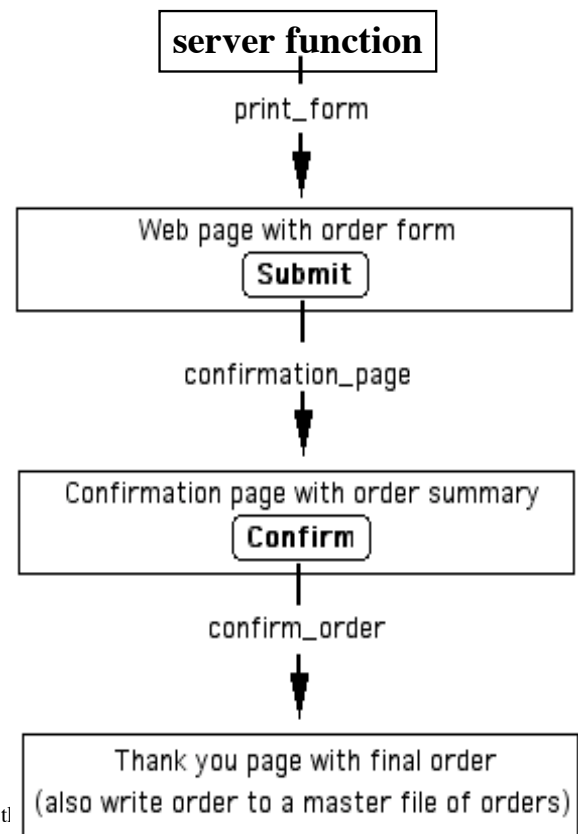
state maintenance (based on notes on ti



pizza3.cgi: a one-step preservation of application state

- Most online order forms give an order summary page, which also serves as an order confirmation page.
- The transaction diagram to the right depicts a Web application which gives such an intermediary confirmation page.
- The arrows represent the three distinct http transactions of the application logic.

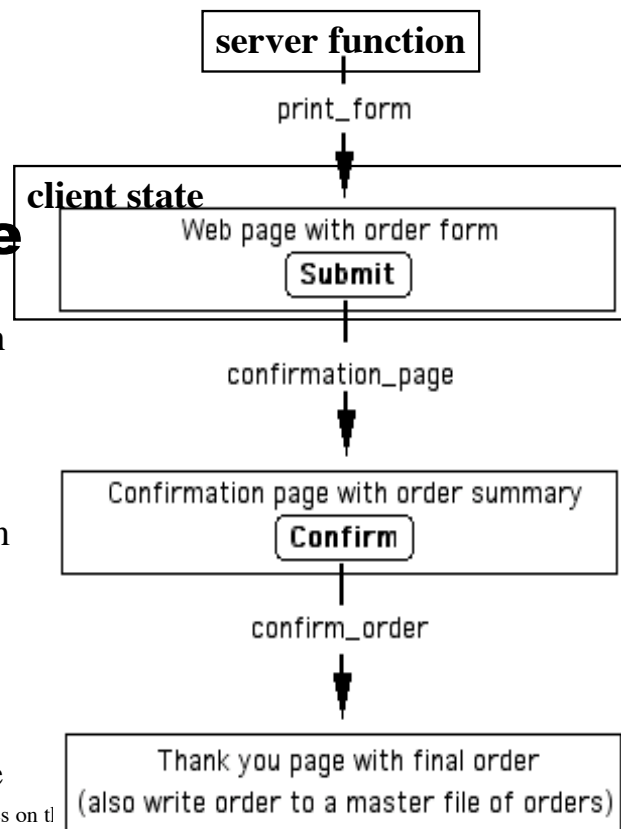
state maintenance (based on notes on ti

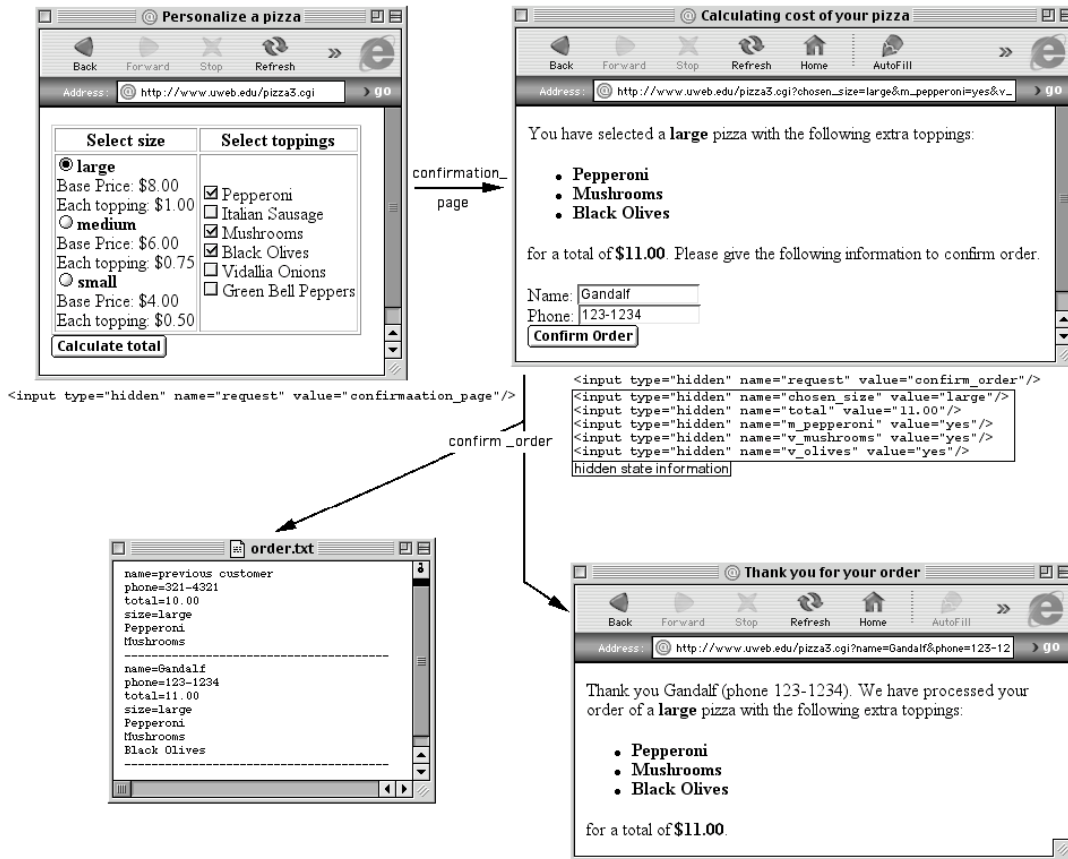


pizza3.cgi: a one-step preservation of application state

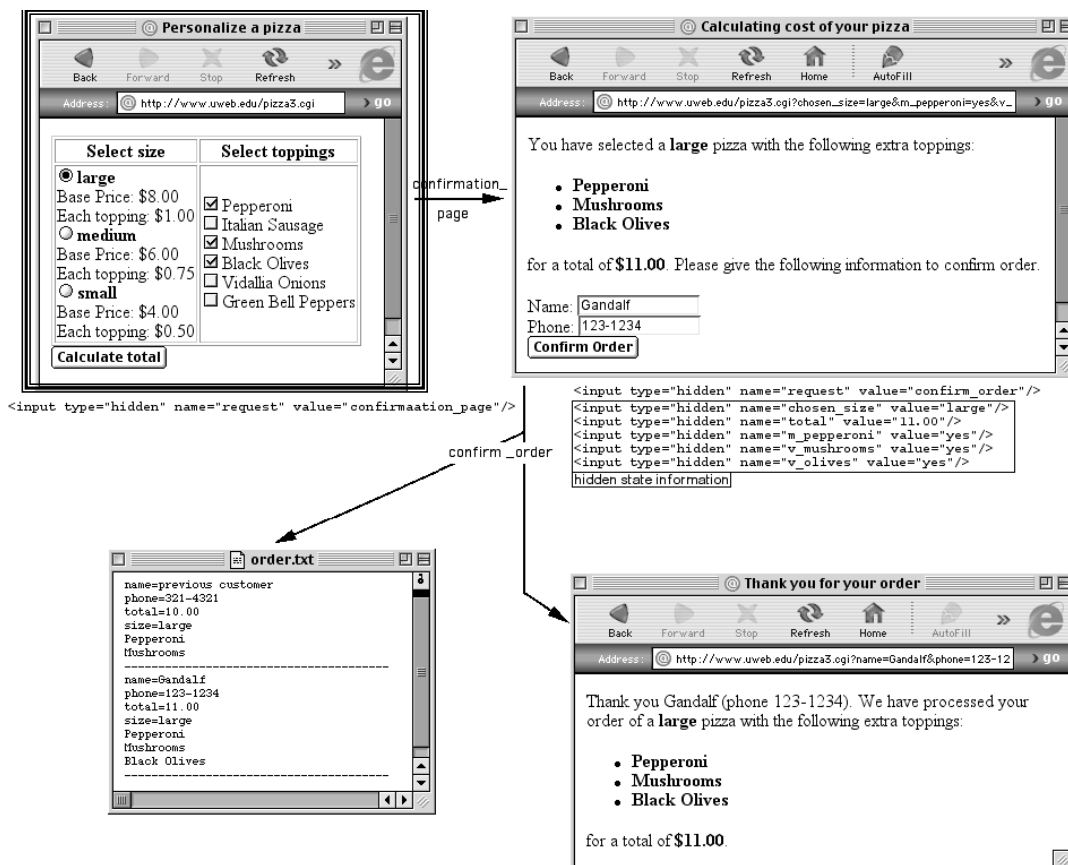
- Most online order forms give an order summary page, which also serves as an order confirmation page.
- The transaction diagram to the right depicts a Web application which gives such an intermediary confirmation page.
- The arrows represent the three distinct http transactions of the application logic.

state maintenance (based on notes on ti

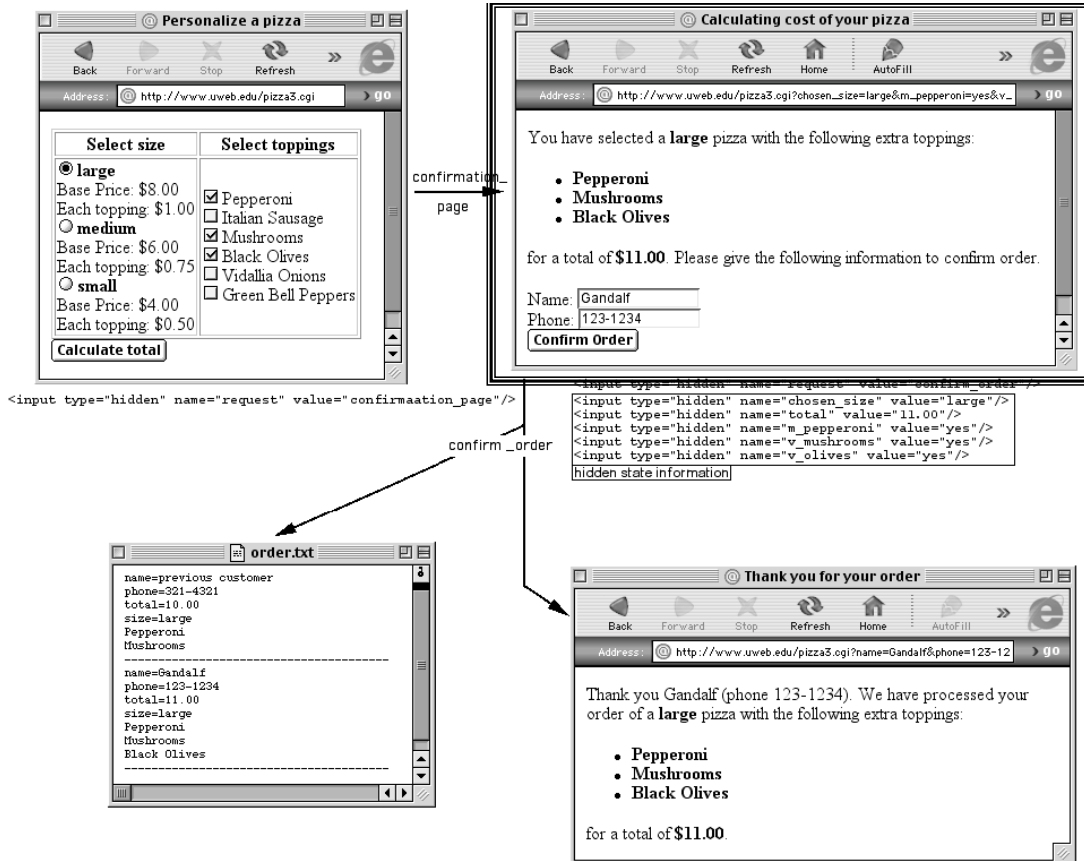




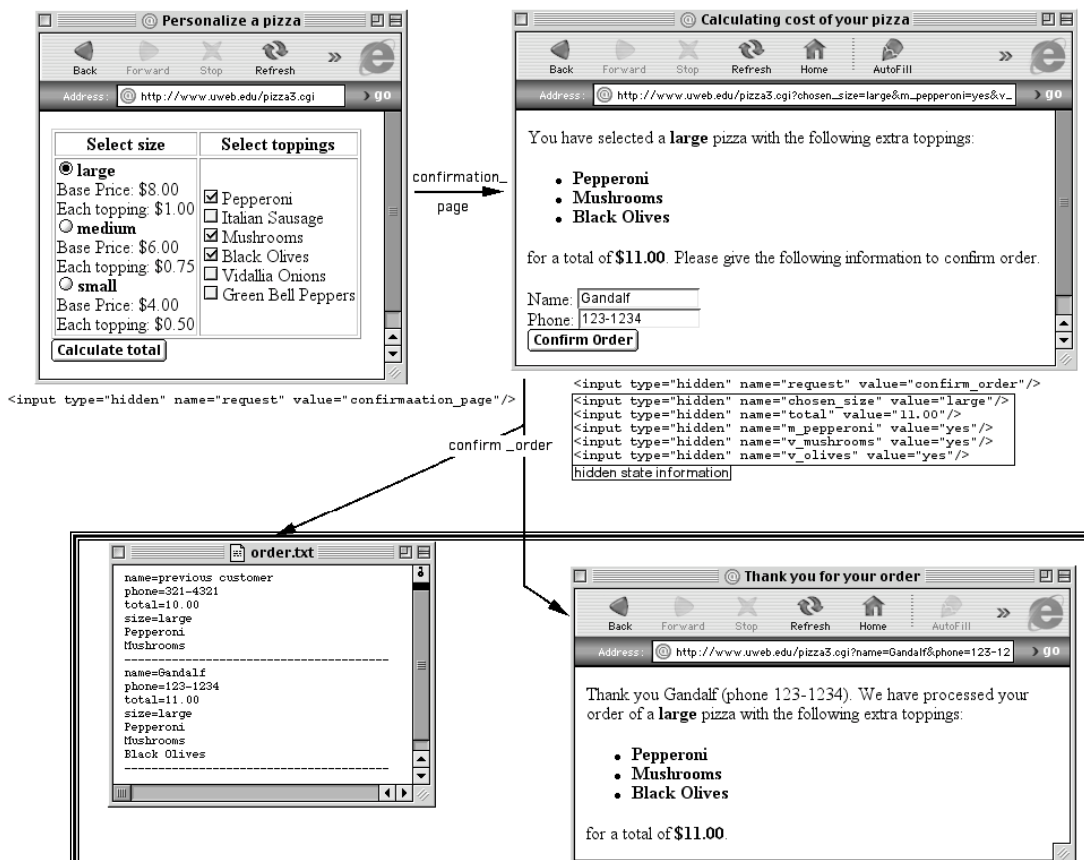
7



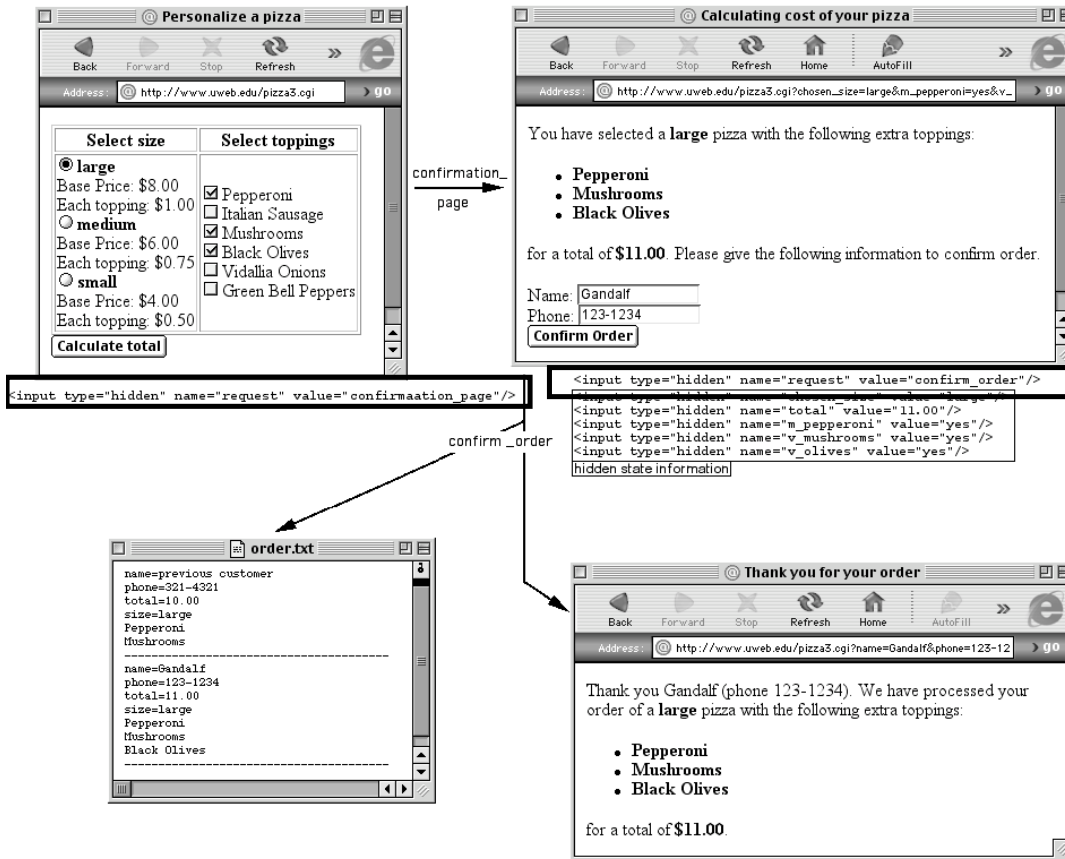
7



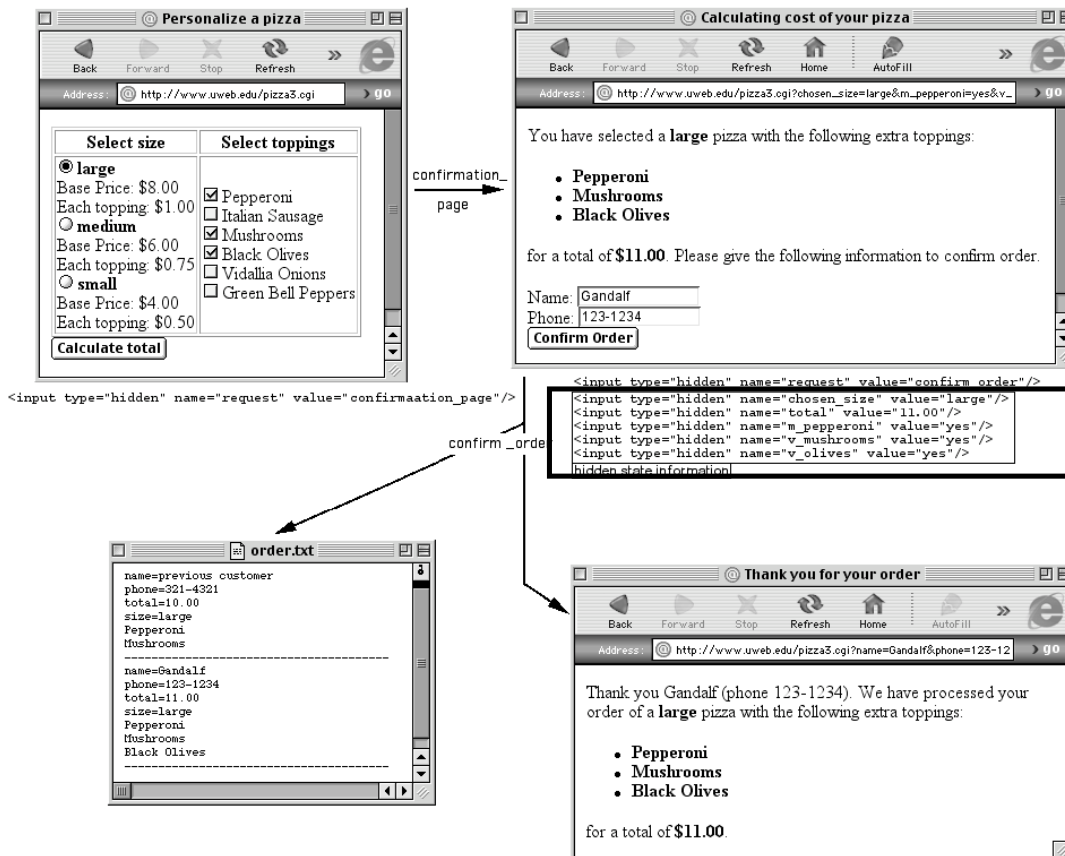
7



7



7



7

pizza3.cgi: how does it work?

- How does pizza3.cgi know which of its three different HTML pages should be sent to the client?
 - Hidden form elements tell the application which page it should generate, i.e. which function should be called.
 - In the original order form `<input type="hidden" name="request" value="confirmation_page"/>`
 - In the intermediate confirmation form: `<input type="hidden" name="request" value="confirm_order" />`
 - The hidden form elements drive the application logic which can be deduced from the transaction diagram.
 - `if($formHash{"request"} eq "confirmation_page") {`
 - `&confirmation_page; }`
 - `elseif($formHash{"request"} eq "confirm_order") {`
 - `&confirm_order; }`
 - `else {`
 - `&print_form; }`

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

8

pizza3.cgi: how does it work?

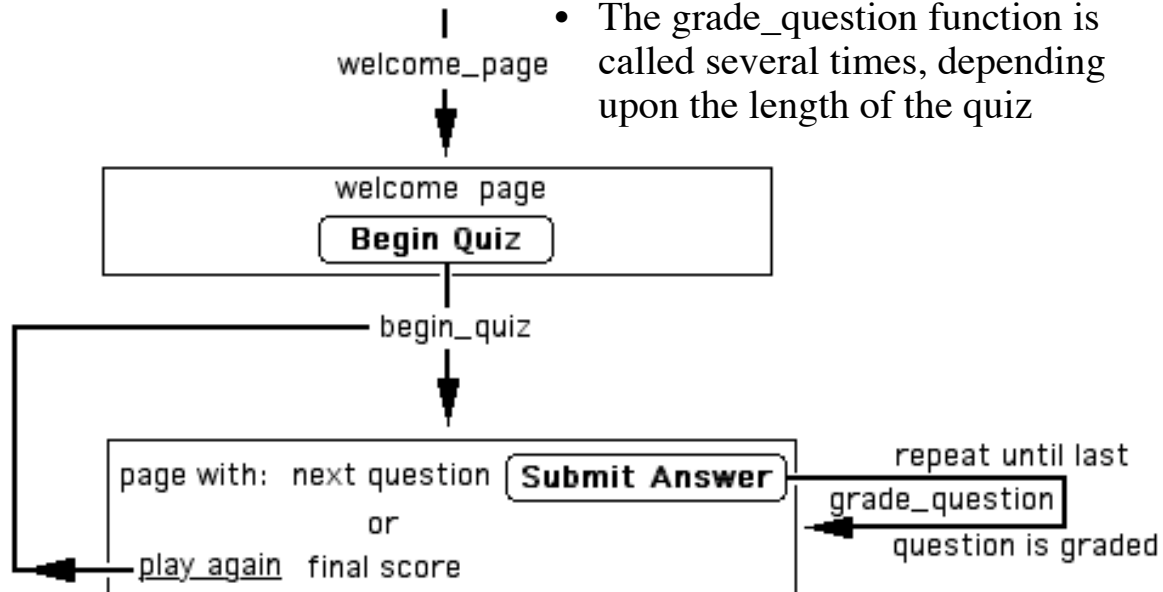
- How does the final “receipt producing” function remember the user's order data submitted in the original order form?
 - The user's order data is hidden in the confirm_order form. Thus, that hidden data is submitted to the server along with the `request=confirm_order` hidden data.
 - `<input type="hidden" name="chosen_size" value="large"/>`
 - `<input type="hidden" name="total" value="11.00"/>`
 - `<input type="hidden" name="m_pepperoni" value="yes"/>`
 - `<input type="hidden" name="v_mushrooms" value="yes"/>`
 - `<input type="hidden" name="v_olives" value="yes"/>`

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

9

quiz1.cgi: a multiple-step state preservation

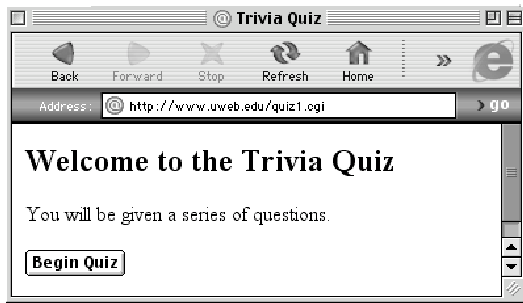
- The quiz application keeps two running counters:
 - the current question being posed
 - the total number of correct answers so far
- The grade_question function is called several times, depending upon the length of the quiz



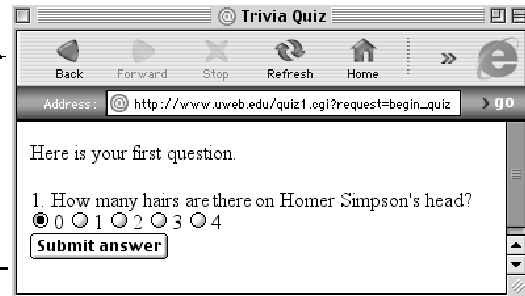
quiz1.cgi: how does it work?

- Three different functions handle the three different types of transactions corresponding to the value of the hidden variable "request"
 - if(\$formHash{"request"} eq "begin_quiz") {
 - &begin_quiz; }
 - elsif(\$formHash{"request"} eq "grade_question") {
 - &grade_question; }
 - else {
 - &welcome_page; }
- The begin_quiz function prints the form for the first question, with the following hidden state data:
 - <input type="hidden" name="number" value="1"/>
 - <input type="hidden" name="correct" value="0"/>
- The grade_question function knows which question it is grading and which one to print next from this hidden data.
- The grade_question function always hides the current state of the quiz in the next question it prints.
 - <input type="hidden" name="number" value="2"/>
 - <input type="hidden" name="correct" value="0"/>

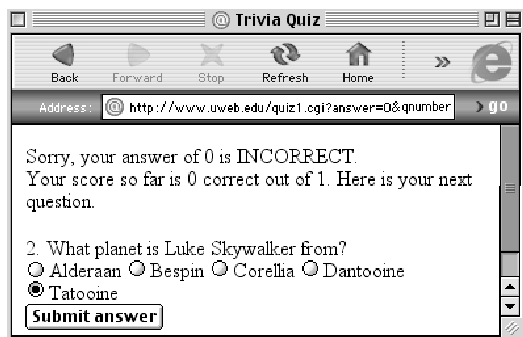
state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)



```
<input type="hidden" name="request" value="begin_quiz"/>
```



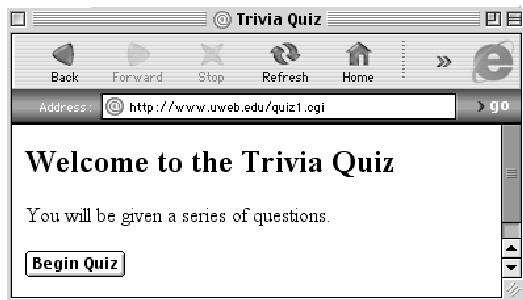
```
<input type="hidden" name="request" value="grade_question"/>
<input type="hidden" name="qnumber" value="1"/>
<input type="hidden" name="correct" value="0"/>
```



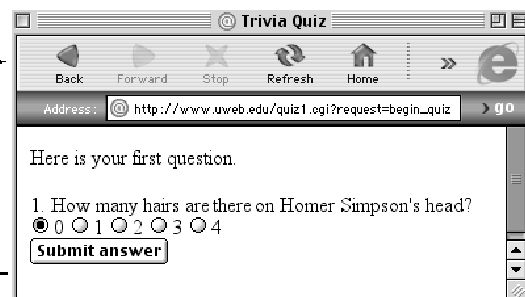
```
<input type="hidden" name="request" value="grade_question"/>
<input type="hidden" name="qnumber" value="2"/>
<input type="hidden" name="correct" value="0"/>
```

<http://www.cknuckles.com/webapps>)

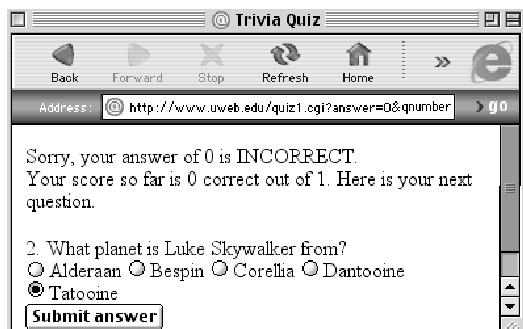
12



```
<input type="hidden" name="request" value="begin_quiz"/>
```



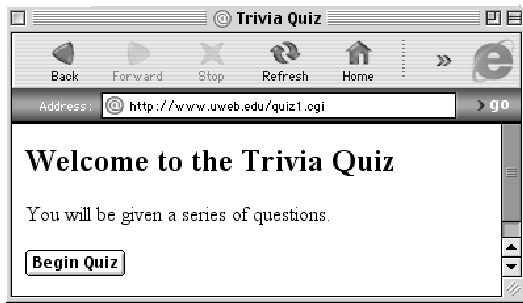
```
<input type="hidden" name="request" value="grade_question"/>
<input type="hidden" name="qnumber" value="1"/>
<input type="hidden" name="correct" value="0"/>
```



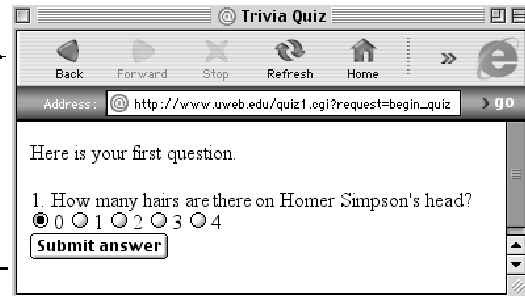
```
<input type="hidden" name="request" value="grade_question"/>
<input type="hidden" name="qnumber" value="2"/>
<input type="hidden" name="correct" value="0"/>
```

<http://www.cknuckles.com/webapps>)

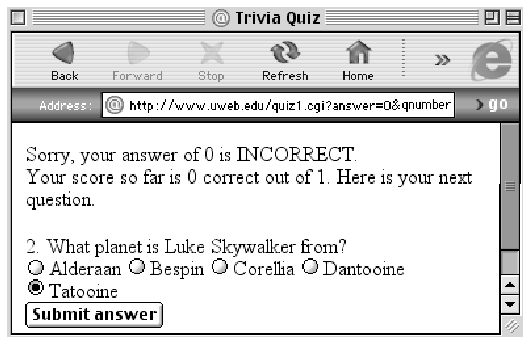
12



```
<input type="hidden" name="request" value="begin_quiz"/>
```



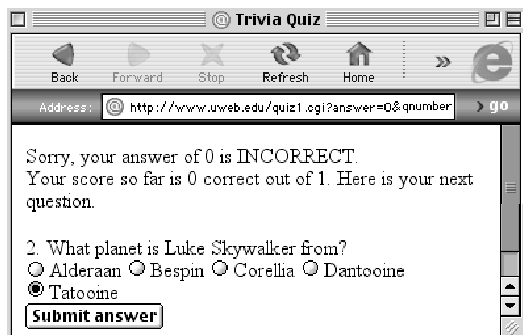
```
<input type="hidden" name="request" value="grade_question"/>
<input type="hidden" name="qnumber" value="1"/>
<input type="hidden" name="correct" value="0"/>
```



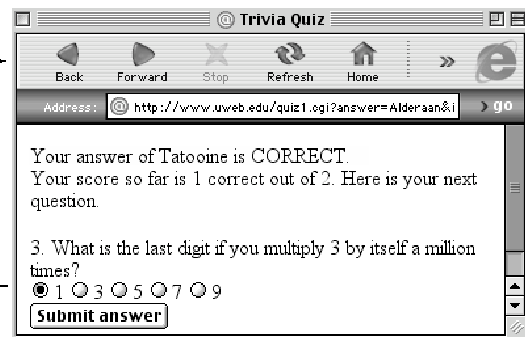
```
<input type="hidden" name="request" value="grade_question"/>
<input type="hidden" name="qnumber" value="2"/>
<input type="hidden" name="correct" value="0"/>
```

<http://www.cknuckles.com/webapps>)

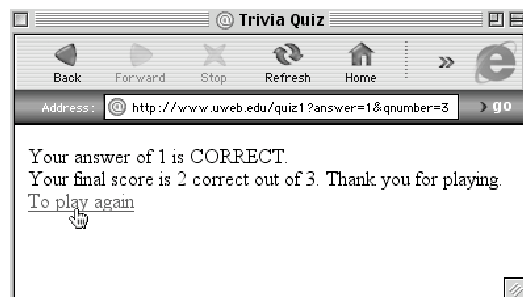
12



```
<input type="hidden" name="request" value="grade_question"/>
<input type="hidden" name="qnumber" value="2"/>
<input type="hidden" name="correct" value="0"/>
```



```
<input type="hidden" name="request" value="grade_question"/>
<input type="hidden" name="qnumber" value="3"/>
<input type="hidden" name="correct" value="1"/>
```



Problems with hidden variables

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

14

Problems with hidden variables

- If state is only at the client, it can be altered.
 - The state data is hidden in Web pages in the browser's cache. Hitting back on the browser effectively pulls up a previous state of the application, so the user can try a step multiple times or execute steps out of order
 - The page can be saved locally; the value of hidden variables (number of correct answers, pizza total cost) can be changed and the “doctored” page can be loaded into a browser and then submitted with the altered state data.

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

14

Problems with hidden variables

- If state is only at the client, it can be altered.
 - The state data is hidden in Web pages in the browser's cache. Hitting back on the browser effectively pulls up a previous state of the application, so the user can try a step multiple times or execute steps out of order
 - The page can be saved locally; the value of hidden variables (number of correct answers, pizza total cost) can be changed and the “doctored” page can be loaded into a browser and then submitted with the altered state data.
- All pertinent state data has to be included in all states whether used in this state or not; context flows from one state to the next only

Server-side state files

A better solution: state files

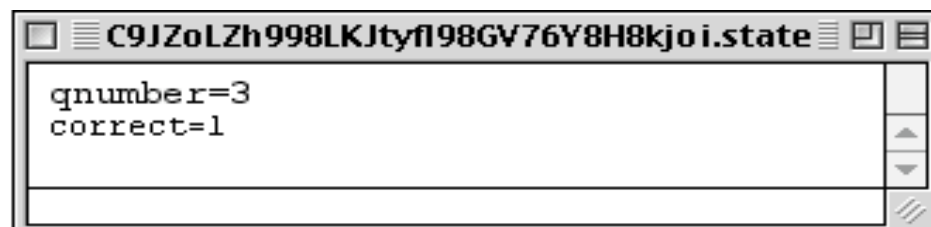
- Create a text file (state file) on the Web server (so that it cannot be easily tampered with) to store state data for each session.
 - Each session that a Web application provides should have a corresponding state file to store state data during the session.
- The application will randomly generate a 32-character string, session ID, for each session (unique, difficult to guess, hidden variable).
 - Example: C9JzoLZh998LKJtyfI98GV76Y8H8kjoI
- The name of session-state file (open for writing) will be
 - ~C9JzoLZh998LKJtyfI98GV76Y8H8kjoI.state
- Each session transaction must submit an id= C9JzoLZh...Y8H8kjoI parameter
 - Possibly hidden, as in
<input type="hidden" name="id" value="C9Jz..." />

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

16

State file format

- Like submitted data from HTML forms, the most convenient way to store state data is in a hash.
 - When a new client request is received, the stored state can be read into a hash in %stateHash
 - As the client request is processed and the state is updated the %stateHash is written back to the state file.
- The order in which the state data is stored in the file doesn't matter and new elements can be added as needed. You can grab a piece of state data using its name (key).
 - For example: \$stateHash{"correct"}



State-file names

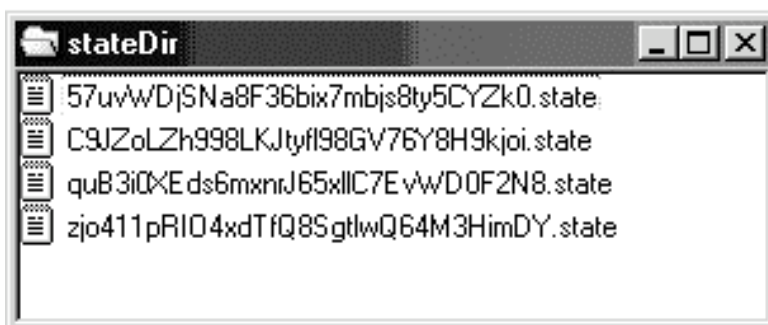
- The state file names should be ensured to be unique so that a new session does not overwrite the state file for a session in progress.
- The session IDs are randomly generated from the characters 0-9 , a-z , A-Z. (62 characters)
- The probability of randomly generating a given 32-digit session ID is $1/62^{32} = 2.3 \times 10^{-57}$.
- The probability of generating the same session ID twice is low enough to ensure uniqueness of IDs across sessions.

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

18

State-file permissions

- The state cache, i.e., the directory containing the state files, **MUST** be given full rwx permission for anyone.
- This is because the Web server software will be the user calling the CGI program which creates/alters/deletes the state files.



**chmod 777
on Unix/Linux**

www.cknuckles.com/webapps)

19

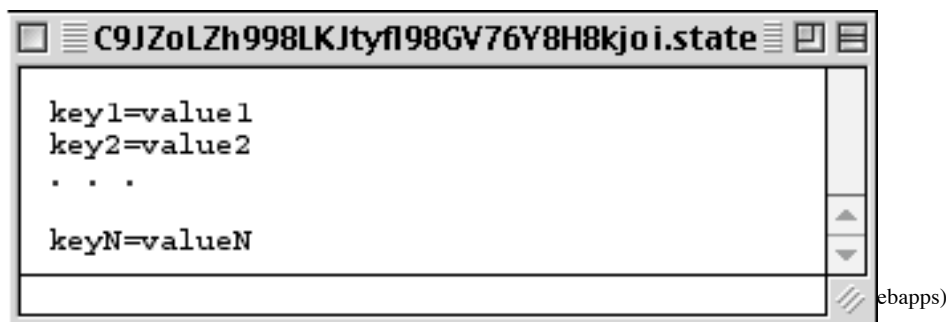
Archiving state files

- After thousands (or tens of thousands) of sessions, the state file cache would contain thousands of state files.
- Possible solutions:
 - The server administrator cleans out the cache periodically,
 - A shell script deletes only those state files that have not been used (modified) lately.
 - The function which creates state files also deletes old ones periodically.

Utilities for state files

- The state data is URL-encoded in the state file.
- The `&write_state` function URL-encodes the state data before writing it to the file.
- The `&read_state` function URL-decodes the state data as it builds the hash it returns.
- One reason for this is to keep unwanted characters (like `=`) in the data from interfering with the structure (delimiting characters) of the state file.
- Another reason is to circumvent a potential security risk which can arise from a cleverly placed `\n` character in the state data. (This is discussed in detail in Section 13.6) .

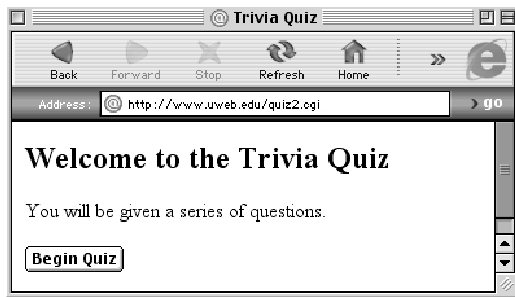
- `&write_state`
 - writes a hash to a state file
 - creates a new state file or overwrites an existing one
- `&write_state($stateDir, $sessionID, %stateHash);`
- `&read_state`
 - returns a hash containing the data found in a state file
- `%stateHash = &read_state($stateDir,$sessionID);`



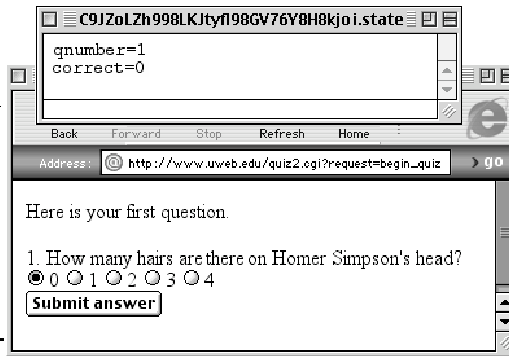
22

quiz2.cgi

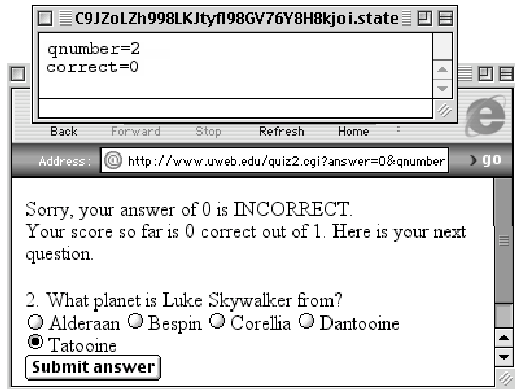
- It behaves the same as `quiz1.cgi`, but uses state files for the session data instead of hiding the state data in the forms. The program simply updates the state file after each question is graded.
- The `sessionID` is hidden in the form for each question to identify the proper state file when the question is submitted.
- The current question number is also hidden in each quiz form to prevent cheating. If the user hits the back button and resubmits a question, the submitted question number won't match the one in the state file.
- Note: It is still necessary to hide the `request=some_function` data in each form in order to drive the app logic.



```
<input type="hidden" name="request" value="begin_quiz">
```



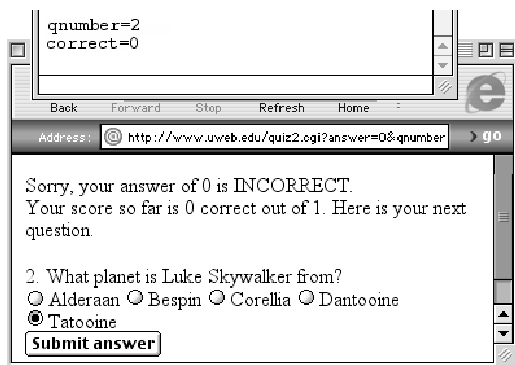
```
<input type="hidden" name="request" value="grade_question"/>
<input type="hidden" name="id" value="C9JZoLZh...Y8H8kjoI"/>
<input type="hidden" name="qnumber" value="1"/>
```



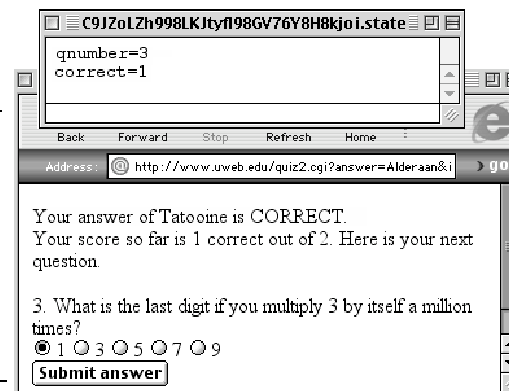
```
<input type="hidden" name="request" value="grade_question"/>
<input type="hidden" name="id" value="C9JZoLZh...Y8H8kjoI"/>
<input type="hidden" name="qnumber" value="2"/>
```

he <http://www.cknuckles.com/webapps>)

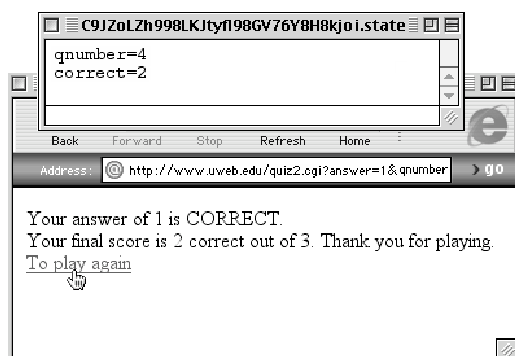
24



```
<input type="hidden" name="request" value="grade_question"/>
<input type="hidden" name="id" value="C9JZoLZh...Y8H8kjoI"/>
<input type="hidden" name="qnumber" value="2"/>
```



```
<input type="hidden" name="request" value="grade_question"/>
<input type="hidden" name="id" value="C9JZoLZh...Y8H8kjoI"/>
<input type="hidden" name="qnumber" value="3"/>
```



25

Policing the state-file cache

- The regular deletion of “stale” session state files should be the responsibility of `get_long_id`.
 - The request for a new session ID should be contingent upon whether the cache has reached some preset limit, the cache limit, on the number of state files it can contain
 - If the cache is "full" (cache limit reached) it is policed before a new session ID is given out.

The “policing” & `get_long_id`

- Obtain the number of files in the cache.
 - If the cache limit is reached, then
 - Delete "old" state files
 - Obtain the number of files in the cache.
 - If the cache limit is still reached
 - Deny the user a session by sending an error page -- site is busy.
 - Notify the site administrator that that the cache still exceeds the cache limit, even after it was policed. The administrator either needs to increase the cache limit to accommodate the load of the application or see if a denial of service attack is occurring.
 - Else, return a long session ID (OK to start session).

Examining directories and files

- Perl's directory functions are used to read in the names (as strings) of all the state files found in the cache.
 - `opendir(D,"path/to/cache/directory/");`
 - `@allfiles = readdir(D);`
 - `closedir(D);`
- The length `$#allfiles` is the number of session state files in the cache
- The files in the cache can be examined by looping over `@allfiles`
 - To test the "age" of state files:
 - `-M filename` returns the time (in days as a real number) since the file was last modified
 - To make sure that what we have is actually a state file: (there could be other directories in the cache, like the hidden ones used by the Unix and Linux file systems.)
 - `-f filename` returns false (0) if it is not a file and true (1) if it is

Password Protection

The password file

- The format of a password file is shown below
 - (`$user`, `$pass`) = `split(/:/, $line, 2)`;
- The password file should be in a directory above the application directory tree



state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

30

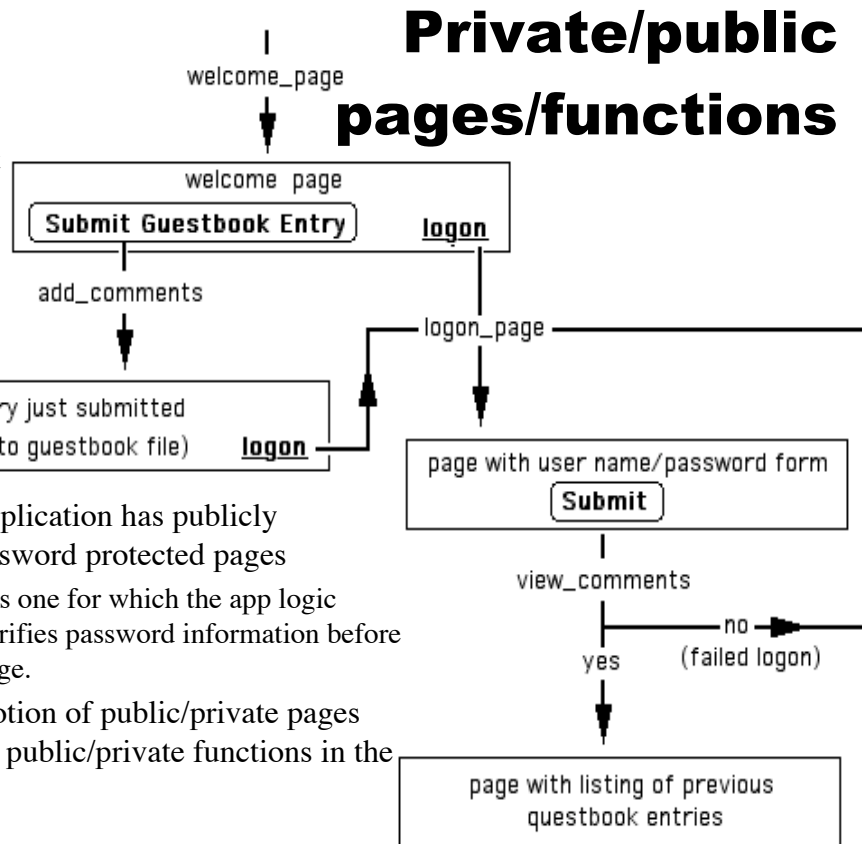
&logon

- Password verification is a common need in Web applications
 - `$result = &logon("pw.txt","user","pass");`
 - If `user:pass` is found on a line in the password file, the function returns the string "yes".
 - Otherwise, it returns a descriptive failure message, "invalid user" for example.

•&view_comments
sends the user back to
the logon page upon
a failed logon attempt

•&logon_page
is publicly accessible

- Usually, a Web application has publicly accessible and password protected pages
 - A private page is one for which the app logic function first verifies password information before returning the page.
- That means, the notion of public/private pages directly equates to public/private functions in the app logic.



comments.cgi (the guestbook)

- `if($formHash{"request"} eq "add_comments") {`
 - `&add_comments;` }
- `elsif($formHash{"request"} eq "view_comments") {`
 - `&view_comments; ## private ##`
- `elsif($formHash{"request"} eq "logon_page") {`
 - `&logon_page;` }
- `else {`
 - `&welcome_page;` }

```

sub view_comments {
    my $result = &logon("$dataDir"."password.txt", $formHash{"user"}, $formHash{"pass"});
    if($result ne "yes") { # failed logon
        &logon_page($result);
        exit; }
    # else print the protected page }
  
```


Anonymous and logged-on state

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

34

Logged-on state

- Logged on state -- the state of prolonged access to private pages (functions) in the application.
- After one successful logon, the entire private portion of the application is available.



state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

35

How to implement logged-on state

- Upon successful logon, create a session ID and state file; hide the session ID in every page subsequently returned to the user.
 - The session ID serves as a validation certificate.
 - The corresponding state file contains information relevant to the status of the logged on user.
 - user name
 - access privileges ...
- A successful logon is the only way to obtain a session ID
 - the existence of the state file is proof that the client has successfully logged on
 - each private function needs to reject a user when called without a validation certificate for logged on state

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

36

Empty state files?

- The mere existence of a state file can be used to verify logged on state.
 - `if (!(-e "path/to/sessionID.state")) {`
 - `&logon_page; # revoke session`
 - `exit; }`
- since successful logon is the only way to obtain a session ID (which is impossible to guess and therefore private) and create a corresponding state file
 - submitting a valid session ID (corresponding file exists) is equivalent to having successfully logged on
 - the file doesn't include who is logged on.

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

37

Named state files?

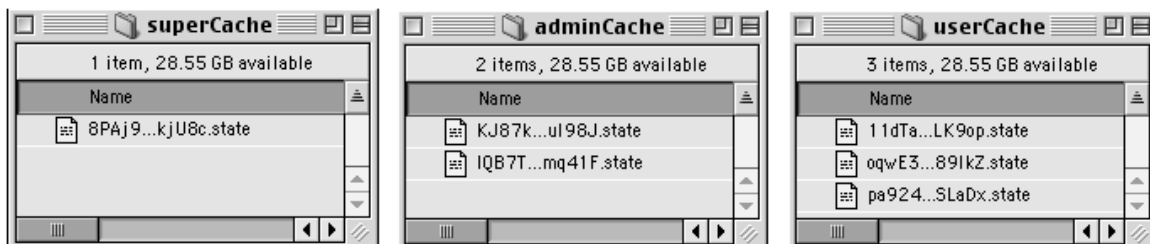
- What if administrator state files were named after the administrator?
 - One may know or try to guess the name of the administrator
 - The administrator could not have two windows open

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

38

Many state files?

- If the state file does not maintain any information, to recognize the level of access privileges it represents, files should be organized in different directories

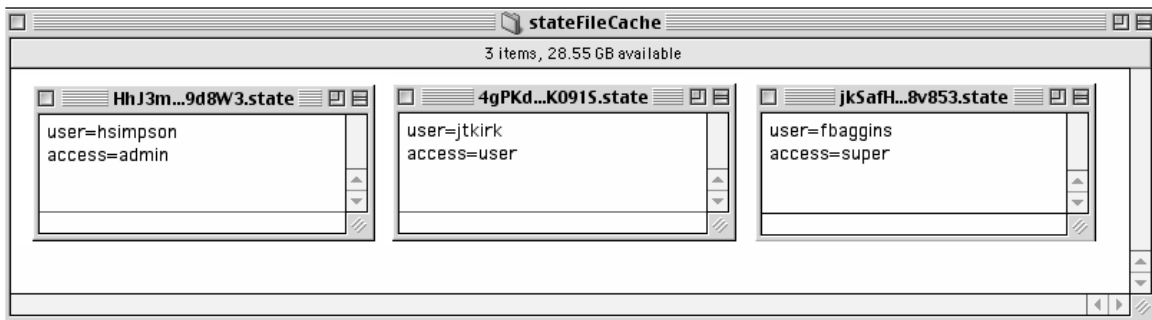


state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

39

One state file?

- In theory, privilege-specific directories for corresponding stet files is no problem.
- However, the built-in state file caching features of most Web programming environments (like PHP, ASP) do not provide for separate file caches.



state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

40

Web-based user-password administration

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

41

```

if($formHash{"request"}
eq "initial_logon") {
    &initial_logon;
## private
## initiate logged on state
}

```

```

elseif($formHash{"request"} eq
"add_user") {
    &add_user;
## private
## verify logged on state
}

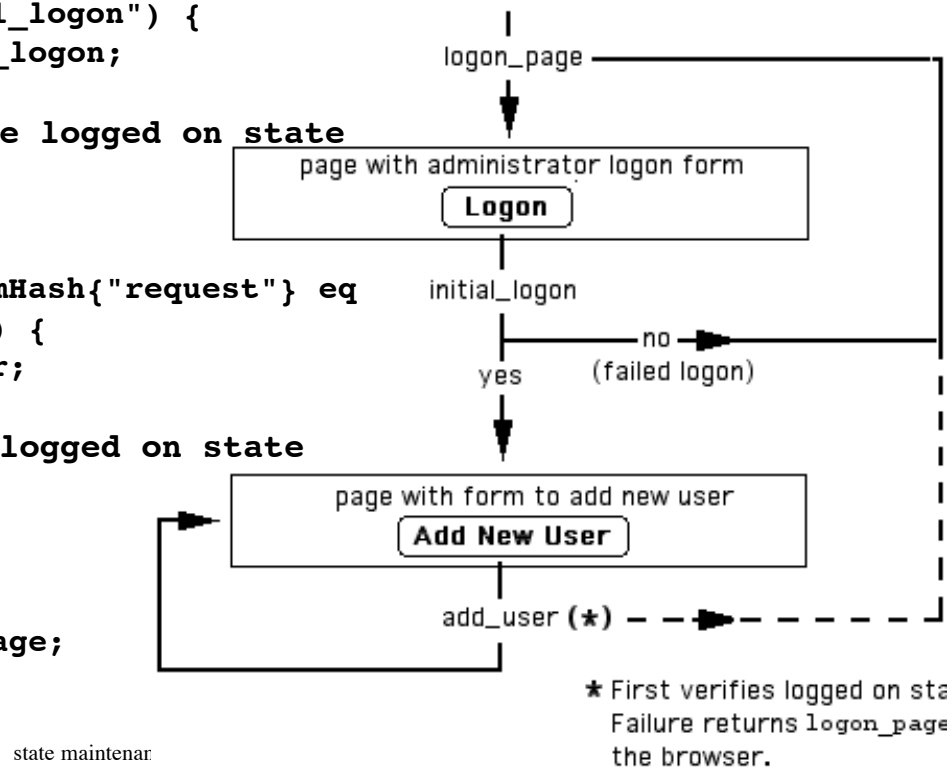
```

```

else {
    &logon_page;
}

```

The workflow



The logic

- The administrator adds users to a password file through a Web interface (i.e. without having manually to edit the password file on the server.)
- Someone must have the administration password, kept in adminpw.txt, in order to have the privilege to append new users to the normal password file, password.txt.
- This application does not actually use the password file. It simply allows an administrator to append lines to it.



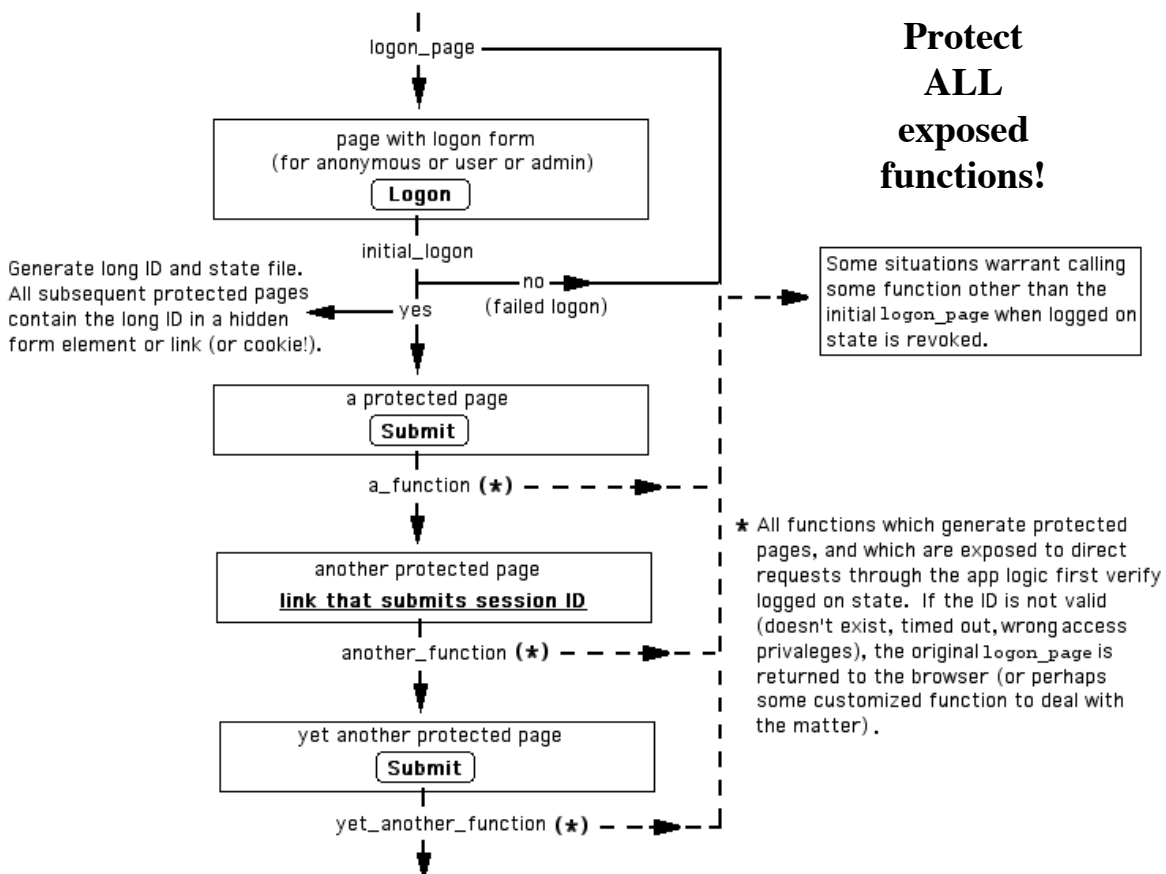
On a “secure” style

- All private functions exposed in the app logic should include security.
- When the application is used as intended, one would first have to complete a successful logon in order to progress to private pages.
- But if a function is exposed in the application logic, it can be called directly by submitting a request=function_name in a query string.

- See if you can hack [http://www.cknuckles.com/webapps/chap07/extras/hackMe/admin\(hackable\).cgi](http://www.cknuckles.com/webapps/chap07/extras/hackMe/admin(hackable).cgi)

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

45



Session Timeouts

- Many Web applications are equipped to revoke sessions after some period of inactivity, maybe an hour or even a day.
- The modified logic of `&read_state`

```
&read_state($dir,$fileroot,$time_out,$time_out_func,$time_out_msg)
```

```
    Read input parameters
```

```
    If only the 2 required input parameters are given, proceed as before
```

```
    Else
```

```
        If the session is timed out or there is no state file
```

```
            If the $time_out_func and $time_out_msg are defined, call it to deal with it
```

```
            Else call the $errorPage function
```

```
        Else "touch" the file and read the state
```

Cookies

- Introduced by Netscape with the release of NN2 in 1996.
- Then they became an official "extension" of the HTTP protocol and supported by all browsers.
- A basic cookie has the form
`www.uweb.edu name=value`

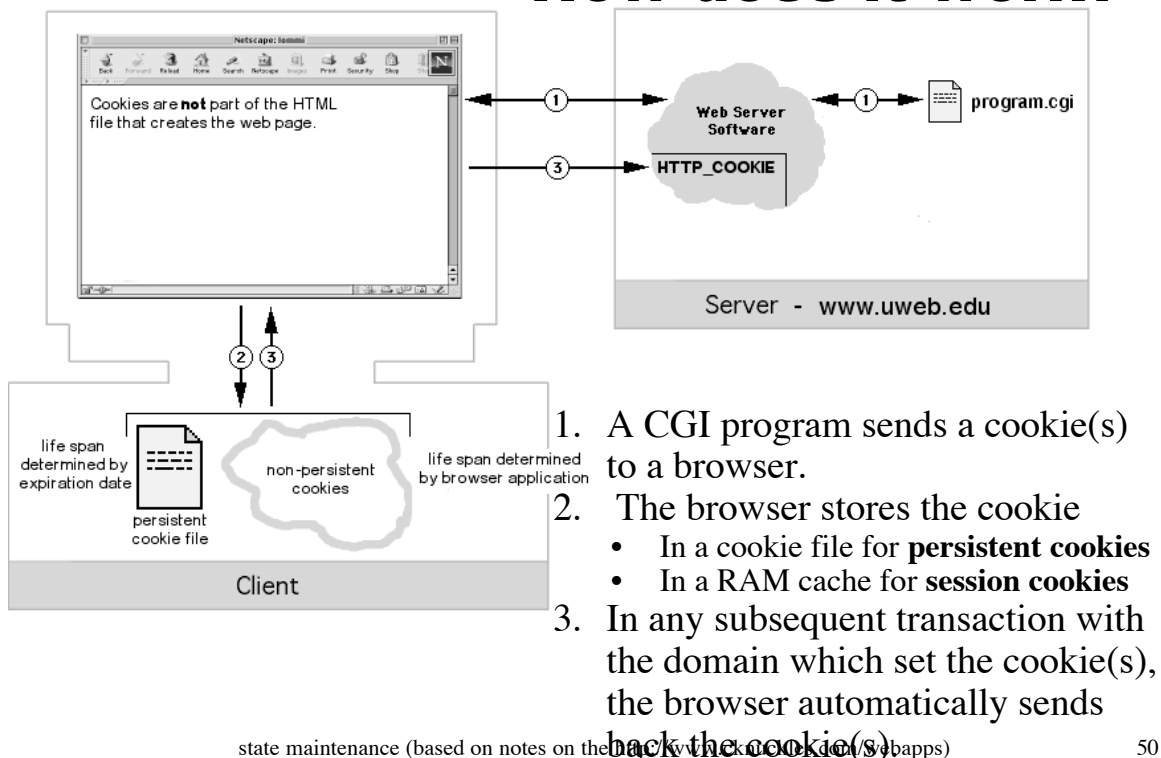
What are cookies good for?

- They can be set using JavaScript.
 - Cookies are the only "writing" that JavaScript can do to the client's file system.
- More importantly, cookies can be set and subsequently used by CGI programs
 - storing state data -- on the client
 - storing session IDs
 - storing user preferences for a given Web site

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

49

How does it work?



state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

50

Cookie types

- Session cookies are meant to live for the duration of a browsing session, like a state file.
 - Depending on the browser, a cache of session cookies might be shared by browser windows or not
- When a cookie is set with an expires field, it becomes a persistent cookie in the cookie file.
`www.uweb.edu name1=value1 expires=Thu, 01-Jan-2005 03:24:33 GMT`
- A browser automatically polices the persistent cookie file, deleting expired ones.

Cookie structure

name=value (mandatory)	The name identifies the cookie. Set a cookie with a name already in use, the new cookie replaces the old one.
expires=date (optional)	Special format for expiration dates: Fri, 03-Nov-2001 15:09:33 GMT
domain=domainname (optional)	The default is to send back only to domain which set the cookie (e.g. <code>www.uweb.edu</code>) Can generalize: <code>domain=uweb.edu</code> Can restrict: <code>domain= x.y.uweb.edu</code>
path=directorypath (optional)	Can also restrict where cookie is returned based on URL path <code>path= /~jones</code>
secure (optional)	Return only if transaction is using https

Manipulating cookies from perl -1

- A cookie is set by placing a Set-cookie line in the HTTP response header
- Simply print the each cookie BEFORE the Content-type line.
 - `print "Set-Cookie: name1=value1\n";`
 - `print "Set-Cookie: name2=value2\n";`
 - `print "Content-type: text/html\n\n";`
- Each cookie must be printed with a following line break to ensure the cookie appears on a separate line in the HTTP header.
- The resulting response:
 - HTTP/1.0 200 OK
 - Date: Fri, 30 Nov 2001 15:24:33 GMT
 - Server: Apache/1.3.1
 - Set-cookie: name1=value1
 - Set-cookie: name2=value2
 - Content-length: 341
 - Content-type text/html
 - ... blank line containing only a newline character
 - ... data returned from the CGI program (i.e. the HTML page)

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

53

Manipulating cookies from perl -2

- The name field carries the data. ALL of the optional fields determine whether a given cookie should be sent back to a given Web server.
- If more fields are needed, they are placed in the print statement, delimited with semi-colons.
 - `print "Set-Cookie: name=value; expires=Thu, 01-Jan-2005 03:24:33 GMT; domain=.uweb.edu; path=/cgi; secure\n";`
- A browser will ONLY send back the above cookie if the request is coming from some sub-domain of uweb.edu and it's in a /cgi directory.
 - `anySubDomain.uweb.edu/cgi`
- Moreover, in the case of that cookie, it must not be expired and the transaction must be using https.

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

54

Manipulating cookies from perl -3

- To access incoming cookies in a CGI program, the HTTP_COOKIE environment variable \$ENV{"HTTP_COOKIE"} must be parsed
- The name=value pairs are in a string delimited by semi-colon followed by a space (kind of weird).
 - name1=value1; name2=value2; name3=value3
- So
 - %cookieHash = ();
 - @nameValuePairs= split(/; /,\$ENV{"HTTP_COOKIE"});
 - foreach \$pair (@nameValuePairs) {
 - (\$name, \$value) = split(/=/, \$pair);
 - \$cookieHash{\$name} = \$value;
 - }

Cookie notes

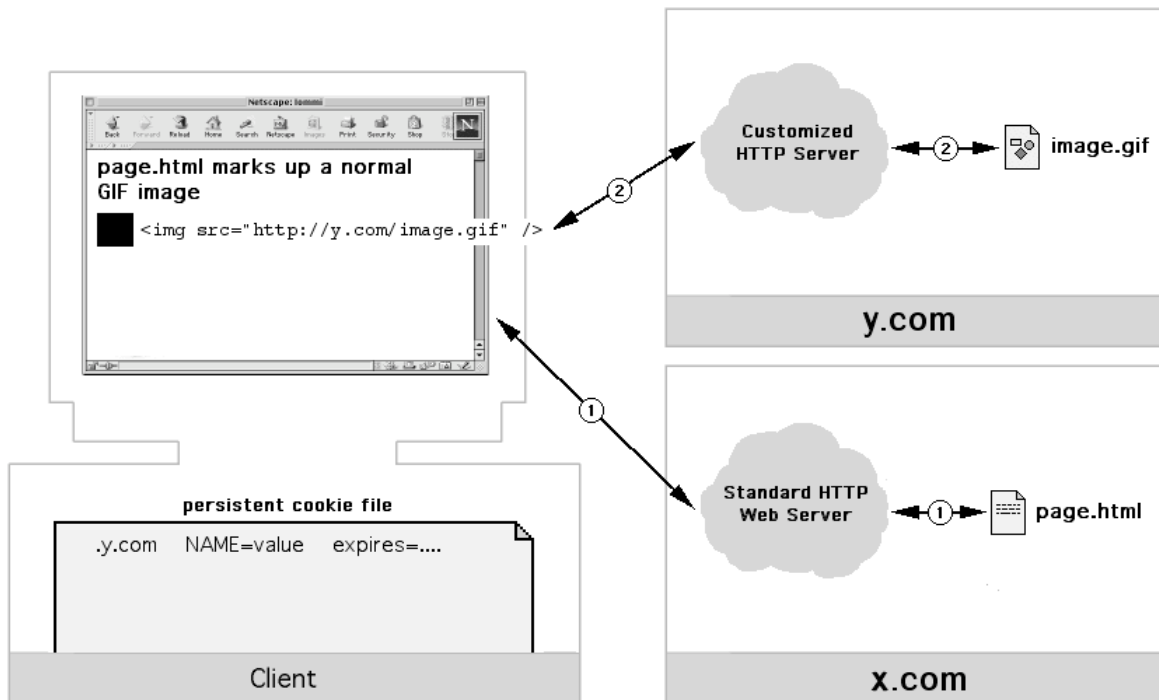
- When a cookie is set on a browser, and one with the same name already exists, the new one overwrites the old one. (Basically, a browser's cookies work like a hash in that respect.)
- To delete a cookie, set a new cookie with the same name but with an expired date.
- When you are writing CGI programs and testing them in an account on a Web server, it is a good idea to include your user name in the path field:
 - path=/~jones
 - That way, Jones only gets his own cookies back (ones set from www.uweb.edu/~jones), not all cookies set by other users on www.uweb.edu
 - Otherwise, if two users are using the same name for a cookie, one user's cookie overwrites the other users cookie.

Cookies versus Embedded Data

- Cookie data is automatically returned by a browser for the life of a session (or longer). Embedded data must be re-embedded in each Web page to propagate a session.
 - If you surf to a different site (without closing the window) and then return to the one which set the cookie, the cookie is sent back and session can resume, ostensibly with no interruption. With embedded data, you would have to go back in the browser's history list and find a cached page with embedded data (a session ID for example) in order to resume a session.
- Sensitive data should NOT be put in a cookie NOR embedded it in a Web page.
 - Old Cookies can be read just as easily as some embedded data cached in an old Web page.
 - Just send a session ID back to the browser and keep sensitive data on the server.
- Cookies are unreliable in that they can be completely disabled in a browser.
- Persistent cookies can be used to arm a particular browser with long-term data, such as site preferences or long term logged on state. Embedded data cannot accomplish that.

Third-party cookies

- There are images in the Web page whose source files reside on a third party server.
- Your browser and the server giving you the Web page are the first two parties - the primary parties in the transaction.
- Your browser reads the src of the HTML image element as it parses the file and makes a secondary request for the graphic from the third party server.
- Normally, this secondary request is made to the server which is serving the HTML page.
- Typically, a third party image is put in a page solely to cause a secondary transaction with a third party server. The purpose of such secondary transactions is to set (and read) cookies.



state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

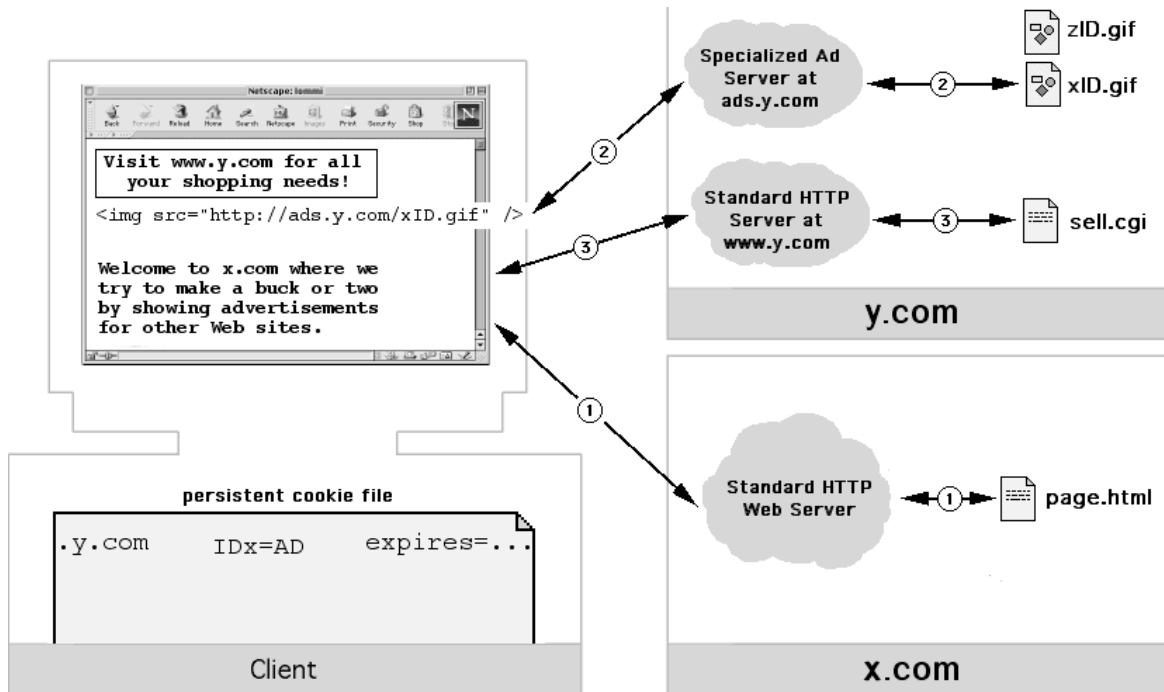
66

Why?

- To set ad banners and monitor how effective they are. That is, third party ad servers can be used to compile statistics during mass marketing campaigns.
- The scenario:
 - y.com sells things online and pays other sites to advertise for them.
 - x.com gets paid by y.com to display their add banner, which is served from y.com's add server.
 - y.com pays a lot of sites to display their ads, not just x.com.
- How does y.com know how many people are seeing their add at x.com as opposed to z.com that is also displaying their ad?
 - Simply, let the name of the requested image contain an ID indicating which site is showing the ad.

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

67



state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

68

E-Marketing terminology

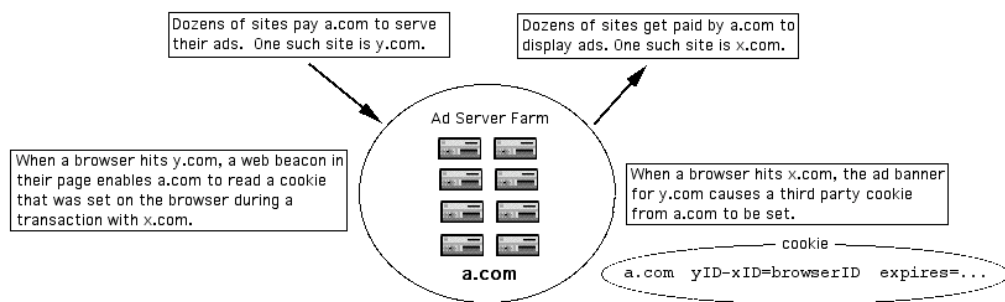
- An **impression**: When y.com's ad server gets a request for xID.gif, for example, they can assume that someone has seen their ad at x.com. It serves the image (which may be the same it serves in response to yID.gif) and sets a cookie.
- A **click-through**: When someone visits y.com's main site the cookie is sent back, so y.com knows to which ad (xID or yID) the user responded

state maintenance (based on notes on the <http://www.cknuckles.com/webapps>)

69

But it gets better!

- Online mass marketers (mediaplex.com, hitbox.com) have come into existence just to run advertising for many other sites.
- By embedding an ID for both the advertiser and the advertisee in the request for the third party graphic, a.com can monitor impressions and click-throughs for all of its customers.
- If they also include some random ID to mark your browser, they can actually track you among their client's sites!
- a.com can compile quite elaborate demographic statistics for their customers in this way



70

And even better!!

- A Web beacon is typically a 1x1 pixel graphic which is the same color as the background of the page. It is invisible and its purpose is to set third party cookies and to cause them to be returned.
- When every site affiliated with msn, for example, puts a Web beacon in EVERY one of their pages, the implications are staggering.
- When you first hit a page in their network, a Web beacon marks your browser with a cookie containing an ID of some sort.
- The network can then track you as you surf to any other pages in their network because those pages have beacons as well.
- If you have an account with one of the sites in the network, they could actually track you personally as you surf.
- Do a search for "Web beacon privacy" and you will see disclaimers from more major commercial sites than you care to look at. They readily admit they are tracking you, but claim the demographics they are compiling are completely impersonal.