Web Technologies and Applications

Winter 2001

CMPUT 499: Perl, Cookies and other

Dr. Osmar R. Zaïane

University of Alberta

What is Perl & What is it for?

- Perl is an acronym for Practical Extraction and Report language.
- It is currently the most used language for CGI-based Web applications.
- Perl is powerful and flexible like a high-level programming language. Perl 5 is object-oriented.
- Perl is scripting language that combines features from awk and sed, yet as powerful as C.

Course Content

• Introduction	• Databases & WWW
• Internet and WWW	• SGML / XML
• Protocols	• Managing servers
• HTML and beyond	• Search Engines
• Animation & WWW	• Web Mining
• Java Script	• CORBA
• Dynamic Pages	• Security Issues
• **Perl Intro.**	• Selected Topics
• Java Applets	• Projects

Objectives of Lecture 8

Perl, Cookies and Other

- Introduce Perl language for CGI development **(This is not a Perl course)**
- Learn about magic cookies and what we can use them for.
- See some examples with cookies in Perl and Javascript.

Outline of Lecture 8

- What is Perl?
- Variables and Expressions
- Control Structures
- File Input and Output
- Pattern matching
- A CGI example with Perl
- Cookies and example with Perl
- Cookie example with JavaScript

Perl the scripting Language

- Perl is commonly used to write scripts.
- It is interpreted and is available on most platforms (free software foundation and GNU)
- It suffices to write a script in a text file, make it executable and run it.
- Initially designed to monitor software projects and generate reports, Perl gained popularity thanks to the Internet & WWW.
- Example

```
#!/usr/local/bin/perl
$input=<STDIN>;
print "$input";
```


Perl Capabilities

- Perl has rich and easy-to-use built-in text processing capabilities.
- Perl is very flexible when it comes to file text processing and pattern matching.
- There are many intrinsic functions for manipulating strings
- Expressions are simple and concise
- There are many ways to write the same thing in Perl.

Programming and Debugging

- Perl programs are written in any text editor
- Start the first line with #!/usr/local/bin/perl
- You can also run a perl expression in a command line with "perl …."
- Debugging is tricky. There are debuggers but they are not as good as C/C++ and Java debuggers
- Use "perl –w" or try with "perl –c"
- Use print statements in your script to trace.

Outline of Lecture 8

- What is Perl?
- Variables and Expressions
- Control Structures
- File Input and Output
- Pattern matching
- A CGI example with Perl
- Cookies and example with Perl
- Cookie example with JavaScript

Scalar Variables

- Scalar variables start with $. Ex. $myVariable
- A scalar variable can contain:
  - a string $myVariable="this is a string";
  - a integer number $myVariable=42;
  - a floating-point number $myVariable=49.33;
- A string with "string" is evaluated while a string with 'string' is not evaluated:
  - $myString = "this is your total: $total";

Arrays and Hash

- Arrays in Perl are indexed from 0 to n-1
- An array is prefixed with @:
  - @myArray = ('bill', 'john', 'sue', 'Amelia');
- An element of an array is prefixed with $
  - $myArray[3] = 'alpha'; $myVar=$myArray[0];
- $#myArray is the last index of @myArray
- Hash or associative array uses keys to reference elements %myHash
  - $myHash{'billy'} = 42; $myHash{$foo}="abc";

Outline of Lecture 8

- What is Perl?
- Variables and Expressions
- Control Structures
- File Input and Output
- Pattern matching
- A CGI example with Perl
- Cookies and example with Perl
- Cookie example with JavaScript

Conditionals

- if (condition) {statements}
  - if ($number) {print “the number is not zero!”;}
- if (condition) {statements} else {statements}
- if (condition) {statements }
  elsif (condition) {statements}
  else {statements}
- Variable = (condition)? expression1 : expression2;

Loops

- while (condition) {statements}
- until (condition) {statements}
- do {statements} while (condition);
- do {statements} until (condition);
- for (init;condition;increment) {statements}
- foreach (list) {statement}
  - foreach $line (@myArray) {print “$line\n”;}
- next, last, redo

Outline of Lecture 8

- What is Perl?
- Variables and Expressions
- Control Structures
- File Input and Output
- Pattern matching
- A CGI example with Perl
- Cookies and example with Perl
- Cookie example with JavaScript

Opening a File

- open(variable, filename)
  - open(MYFILE, “/usr/me/myfile.txt”);
- Open for writing
  - open(MYFILE, “>/usr/me/myfile.txt”);
- Open for reading
  - open(MYFILE, “</usr/me/myfile.txt”);
- Open for appending
  - open(MYFILE, “>>/usr/me/myfile.txt”);
- Closing a file with close(MYFILE)

Reading/Writing to/from a File

- $line=<MYFILE>
- @lines=<MYFILE>
- while ($line=<MYFILE>) {
  chop $line;
  print “line $i:[$line]\n”;
  }
- print MYFILE “this will go in the file as a line\n”;

Determining the status of a file

- if (-e “file”) # checks if file exists
- if (-d “file”) # checks if file is directory
- if (-f “file”) #checks if file is ordinary file
- if (-l “file”) #checks if file is symbolic link
- if (-r “file”) #checks if file is readable
- if (-w “file”) #checks if file is writable
- if (-x “file”) # checks if file is executable
- …

Outline of Lecture 8

- What is Perl?
- Variables and Expressions
- Control Structures
- File Input and Output
- Pattern matching
- A CGI example with Perl
- Cookies and example with Perl
- Cookie example with JavaScript

The power of Perl

- One of the most powerful and attractive capabilitites of Perl is the use of regular expressions for pattern matching.
- Allows large amount of text to be searched with relatively simple expressions.
- if ($myString eq “hello”) # equality operator
- if ($myString =~ /hello/) # matching operator

Pattern Matching

- if ($myString =~ /^hello/) # at the beginning
- if ($myString =~ /hello$/) # at the end
- if ($myString !~ /hello/) # no match
- if ($myString =~ / \b (\w+ ing) \b/x)
  - $1 would be mached with a word ending with “ing”
  - /x ignores spaces /i ignores case
- if ($myString =~ / [^a-z0-9]/)
  - matches any character not in a to z and 0 to 9

Substitutions

- Substitution is replacing substrings with other substrings.
- $myString =~ s/\t/\s/g;

- Spliting @myarray = split(/*/,$myString)

Outline of Lecture 8

- What is Perl?
- Variables and Expressions
- Control Structures
- File Input and Output
- Pattern matching
- A CGI example with Perl
- Cookies and example with Perl
- Cookie example with JavaScript

Parsing CGI input

```
#!/usr/local/bin/perl
#------------------------------------------------------------------------
#######       Getting the input from STDIN or command line
#------------------------------------------------------------------------
$my_input =   ($ENV{REQUEST_METHOD} eq "POST") ?
        <STDIN> : $ENV{QUERY_STRING};
 #------------------------------------------------------------------------
#######       Splitting input by parameter and value
#------------------------------------------------------------------------
@my_QUERY_LIST = split( /&/, $my_input);          # Splitting all pairs
foreach $item (@my_QUERY_LIST) {
     ($my_param, $my_value) = split( /=/, $item);      # Splitting variables and values
     $my_value =~ s/\+/ /g;                            # Change +'s to spaces
     $my_value =~ s/\s*$//;                            # eliminate spaces at the end
     $my_value =~ s/\%0D\%0A/\n/g;
     $my_value =~ s/%(..)/pack('C',hex($1))/ge;
     if ($my_in{$my_param}) {
          $my_in{$my_param} .= ' ';
          $my_in{$my_param} .= $my_value;
      } else {
           $my_in{$my_param} = $my_value;
      }
}
```

Snippet of CGI with Perl

```
$name = $my_in{‘variable1’};
$habit = $my-in{‘variable2’};

…

print “Content-type: text/html\n\n”;          #printing the header
# printing the page
print “<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\">”;
print “<HTML><HEAD><TITLE>Testing Cookies</TITLE></HEAD>\n”;
print “<BODY><h1>My page title</h1>\n”;

…

print “</BODY></HTML>\n”;
```


Outline of Lecture 8

- What is Perl?
- Variables and Expressions
- Control Structures
- File Input and Output
- Pattern matching
- A CGI example with Perl
- Cookies and example with Perl
- Cookie example with JavaScript

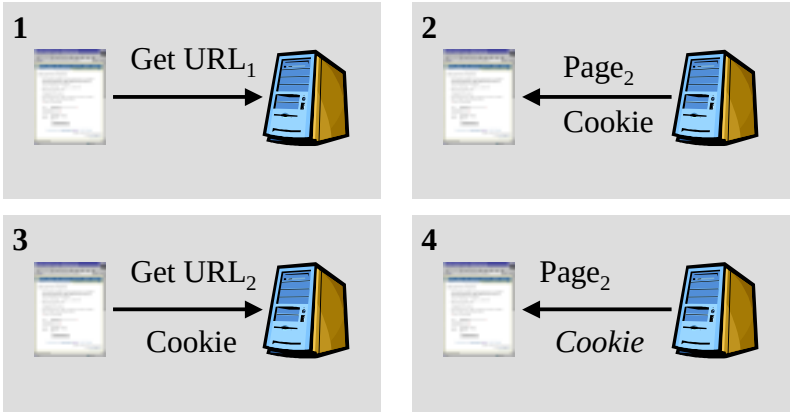
Hidden Fields

- You have used hidden fields in your assignment 3 to maintain some data between different forms in the same session.
- This data is lost in later sessions.
- Noting is actually stored on the client.
- How do we store data on the client to maintain persistent data between sessions?

Magic Cookies

- A cookie is some text like a text file that is passed transparently between the client (browser) and the server to maintain state information.
- The server creates the cookie and sends it to the browser which will send it back each time it connects to the server again.

Cookie Life Cycle



The browser always sends back the cookie to the server

The server may send back the cookie with changes

Cookies are not Forever

- A cookie has an expiry date attached to it
- After the expiry date, the browser automatically deletes the cookie
- If no expiry date is specified, the cookie last only for the current session.
- The expiry date is of the form:

  WeekDay DD-Month-YYYY HH:MM:SS GMT

Passing Cookies

- The client receives a cookie in the HTTP response header with ***Set-Cookie***
- Set-cookie: name1=value1; name2=value2;…
- The CGI running on the server receives the cookie in an HTTP variable HTTP-COOKIE
- When constructing the HTTP response header, the CGI adds a cookie with *set-cookie*.

Restrictions on Cookies

- Cookies are sent back only to the server that generated them or to the Internet domain specified by the creator of the cookie.
- Cookies can be restricted within a site (web server) using a specified path.
- The size of a cookie is limited to 4 Kb.
- No more than 20 cookies can be created at a time.

Cookie Attributes

- **Name** cookie name
- **Value** value attached to cookie
- **Domain** domain that can receive cookie
- **Path** restricts the cookie in the site
- **Expires** expiry date
- **Secure** specifies that the cookie in encrypted (not very secure)

Syntax for Cookies

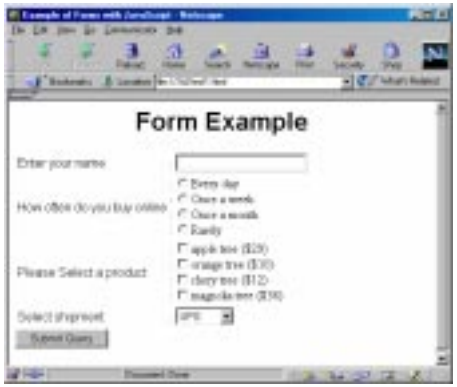
- A cookies with all its attributes is defines in one line with attributes separated by ";"

  Set-Cookie: name=test; value=20; expires=Tuesday, 13 Feb 2001 16:00:00 GMT; domain=.ualberta.ca
- The value can not have spaces and semi-column, etc. ➔ use escaped characters.
- If more than one cookie are needed by the application then it is necessary to have each cookie with one set-cookie line.

What are Cookies Used For?

- Cookies are used whenever a Web Application needs to store information on the client-side for persistent state
- Used as a cart in e-commerce applications
- Used to personalize web pages (user preferences)
- Used to identify users and sessions
- Used to track users on a site for user behaviour analysis
- Etc.

Example with Cookies



- We want to create a cookie to store in the browser information about the customer, the preferences, as well as the cart.
- We will create different cookies.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html> <HEAD> <title>Example of Forms with JavaScript</title></HEAD>
<BODY bgcolor="#ffffff">
<center> <h1> Form Example</h1> </center>

<FORM name="myform" action="cgi-bin/cookies.pl" >
Enter your name: <input type="text" name= "variable1" size=20 >  <br>
How often do you buy online
: <input type="radio" name="variable2" Value="every day">Every day<br>
: <input type="radio" name="variable2" Value="once a week">Once a week<br>
: <input type="radio" name="variable2" Value="once a month">Once a month<br>
: <input type="radio" name="variable2" Value="rarely">Rarely<br>
Please Select a product
:  <input type="checkbox" name="var1">apple tree ($29) <br>
:  <input type="checkbox" name="var2">orange tree ($10) <br>
:  <input type="checkbox" name="var3">cherry tree ($12) <br>
:  <input type="checkbox" name="var4">magnolia tree ($36) <br>
Select shipment:  <select name="variable3"> <option> FedEx <option selected> UPS
  <option> Surface  <option> Air   <option> Urgent  </select> <input type=submit>
</form> </body> </html>
```


Parsing CGI input

```
#--------------------------------------------------------------------
#######       Getting the input from STDIN or command line
#--------------------------------------------------------------------
$my_input =   ($ENV{REQUEST_METHOD} eq "POST") ?
        <STDIN> : $ENV{QUERY_STRING};
 #--------------------------------------------------------------------
#######       Splitting input by parameter and value
#--------------------------------------------------------------------
@my_QUERY_LIST = split( /&/, $my_input);          # Splitting all pairs
foreach $item (@my_QUERY_LIST) {
      ($my_param, $my_value) = split( /=/, $item);      # Splitting variables and values
      $my_value =~ s/\+/ /g;                            # Change +'s to spaces
      $my_value =~ s/\s*$//;                            # eliminate spaces at the end
      $my_value =~ s/\%0D\%0A/\n/g;
      $my_value =~ s/%(..)/pack('C',hex($1))/ge;
      if ($my_in{$my_param}) {
            $my_in{$my_param} .= ' ';
            $my_in{$my_param} .= $my_value;
       } else {
             $my_in{$my_param} = $my_value;
      }
}
```


Generating Cookies in Perl

```
$name = $my_in{'variable1'};
$habit = $my-in{'variable2'};
$expirydate="Monday, 31-Dec-2001 23:59:00 GMT";
$servers="129.128.0"
print "Set-Cookie: Customer=$name; expires=$expirydate; domain=$servers\n";
print "Set-Cookie: Preference=$habit; expires=$expirydate\n";

…

print "Content-type: text/html\n\n";
print "<HTML><HEAD><TITLE>Testing Cookies</TITLE></HEAD>\n";
print "<BODY><h1>Testing Cookies</h1>\n";
print "The cookies have been set.";
print "</BODY></HTML>\n";
```


Reading Cookies in Perl

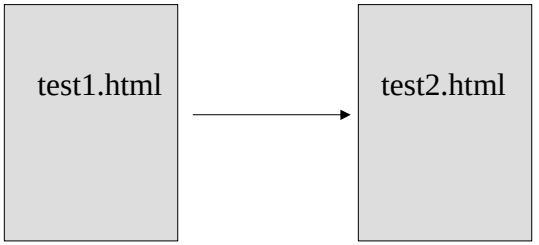
```
sub readCookies {
        @rawCookies = split (/; /,$ENV{'HTTP_COOKIE'});
        foreach (@cookies) {
                ($cookieName, $cookieValue) = split (/=/, $_);
                $Cookies{$cookieName}=$cookieValue;
        }
        return %Cookies;
}
```

Outline of Lecture 8

- What is Perl?
- Variables and Expressions
- Control Structures
- File Input and Output
- Pattern matching
- A CGI example with Perl
- Cookies and example with Perl
- Cookie example with JavaScript

Simple Example

- See test1.html and test2.html



HTML Form
Generates cookies

Displays cookie content


```
function SetCookie (name, value) {
   var argv = SetCookie.arguments;
   var argc = SetCookie.arguments.length;
   var expires = (argc &gt; 2) ? argv[2] : null;
   var path = (argc &gt; 3) ? argv[3] : null;
   var domain = (argc &gt; 4) ? argv[4] : null;
   var secure = (argc &gt; 5) ? argv[5] : false;
   document.cookie = name + "=" + escape (value) +
            ((expires == null) ? "" : ("; expires=" + expires.toGMTString())) +
            ((path == null) ? "" : ("; path=" + path)) +
            ((domain == null) ? "" : ("; domain=" + domain)) +
            ((secure == true) ? "; secure" : "");
}
function setC(form) {
   var expdate = new Date ();
   expdate.setTime (expdate.getTime() + (24 * 60 * 60 * 1000 * 31));
   SetCookie (form.name, form.value, expdate);
}
```


Call setC() in the form for each possible input with onChange, onBlur, onClick, etc.
For input fields with different values, call a function when form is submitted.

```
<form …>
  Enter your name:
   <input type=text size=30 name=“myname" onChange="setC(this)">
</form>
```

```
<FORM name="myform" action="JavaScript:shopper()" >
```

```
<script language="JavaScript“>
function getCookieVal(offset) {   // Get Cookie Value function
   var endstr = document.cookie.indexOf (";", offset);
   if (endstr == -1) endstr = document.cookie.length;
   return unescape (document.cookie.substring(offset, endstr));
}
function GetCookie(name) {      // Get Cookie function
   var arg = name+"=";
   var alen = arg.length;
   var clen = document.cookie.length;
   var i = 0;
   while (i < clen) {
      var j = i + alen;
      if (document.cookie.substring(i, j) == arg) return getCookieVal(j);
      i = document.cookie.indexOf(" ", i) + 1;
      if (i == 0) break;
   }
   return null;
}
</script>
```