IT350 Web and Internet Programming

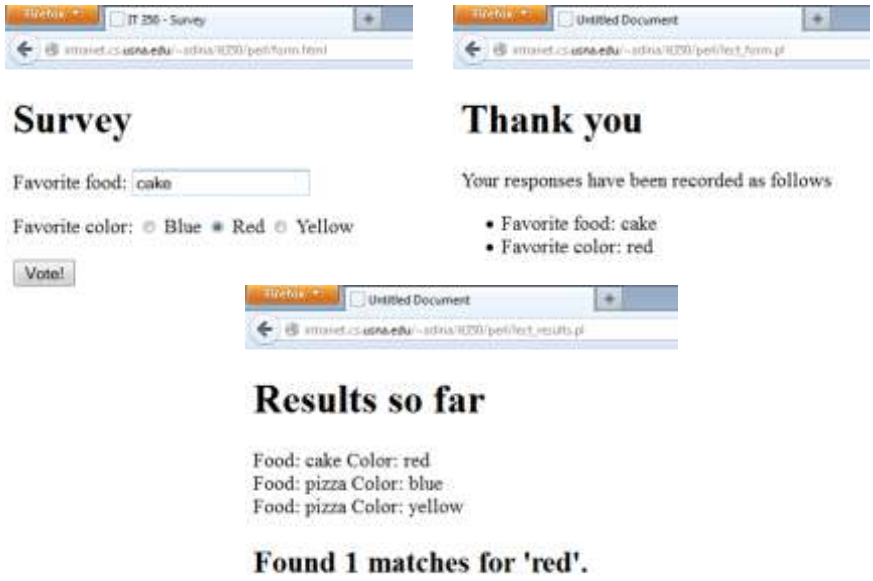
SlideSet #12: CGI and Perl

(see online references)

Things we'll learn and do

- HTML5 – basics, tables, forms
- Cascading Style Sheets
- JavaScript
- Dynamic HTML
- CGI / Perl

CGI – What does it all look like?



CGI Script Basics

- Common Gateway Interface (CGI)
  - "Common": Not specific to any operating system or language

- Output file generated at runtime:
  1. When a program executed as a CGI script, "standard output" is redirected to web server
  2. Web server then redirects output to client's browser

How can CGI get data from user?

Technique #1: Forms

- User enters data via a form, submits
- Web server directs data to a CGI program

- Script receives data in one of two ways:
  1. method = "get"
  2. method = "post"

  Use language-specific method to get these inside CGI program

Technique #2: URL with parameters

```
<a href="http://www.usna.edu/CS/calendar/view.pl?events=seminars">
     Seminars </a>
```

Perl Stuff 1

"Scalar" variables:

```
$x = 3;
$y = "Hello";
```

"Array" variables:

```
@list = (3, 7, "dog", "cat");
@list2 = @list1;      # copies whole array!
```

A single element of an array is a scalar:

```
print "Second item is: $list[1];    # Don't use @
```

Get array length by treating whole array as scalar:

```
$lengthOfList2 = @list2;
```

File operations

```
open ( MYFILE, "input.txt" );
open ( MYFILE, ">output.txt" );
open ( MYFILE, ">>LOG.txt" );
```

form.html

The Big Example Part 1 (the form)

```
(standard header stuff…)
<body>
  <h1> Survey  </h1>

  <form method= "post" action="lect_form.pl">

      <p> Favorite food: <input type="text" name="food" /> </p>
      <p> Favorite color:
          <input type="radio" name="color" value="blue" /> Blue
          <input type="radio" name="color" value="red " /> Red
          <input type="radio" name="color" value="yellow" /> Yellow
      </p>
      <p> <input type="submit" value="Vote!" /></p>

  </form>
</body>
</html>
```

lect_form.pl

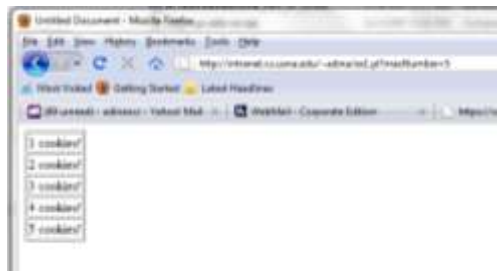
The Big Example Part 2 (CGI to receive)

```
#!/usr/bin/perl
use strict;
use CGI qw( :standard );
use CGI::Carp qw(warningsToBrowser fatalsToBrowser);

# Get inputs from browser/user
my $favFood    = param("food");
my $favColor   = param("color");

print header(); print start_html("Survey Response");

# Save result in file.  Use colon as separator
open ( OUTFILE, ">>favorites.txt" ) or print h1("Could not open file favorites.txt
    for append");
print OUTFILE "$favFood : $favColor" . "\n";
close ( OUTFILE );

# Thank user and display what was received. (provide feedback)
print "<h1> Thank you </h1> \n";
print "<p> Your responses have been recorded as follows</p> \n";
print "<ul> \n";
print li("Favorite food: $favFood");
print li("Favorite color: $favColor");
print "</ul>\n";

print end_html();
```

Exercise #1

- Write Perl code that will, given the URL provided below, generate HTML that looks like the screenshot

  http://zee.cs.usna.edu/~adina/ex1.pl?maxNumber=5



(extra space for Exercise 1)

```
#!/usr/bin/perl
use strict;
use CGI qw( :standard );
```

lect_results.pl

The Big Example Part 3 (CGI to process)

```
#!/usr/bin/perl
use strict;
use CGI qw( :standard );
use CGI::Carp qw(warningsToBrowser fatalsToBrowser);

print header(); print start_html("Survey report");

print h1("Results so far");
my $redCount = 0;

open ( INFILE, "favorites.txt" ) or print h1("Error: could not open
   file favorites.txt for reading");
while (my $aLine= <INFILE>) {
  chomp ($aLine);

  # Split lines wherever we see a colon
  my @myArray = split (/:/, $aLine);

  # Print out the various parts
  print "Food: $myArray[0] Color: $myArray[1] <br/>";

  if ($myArray[1] =~ /red/i) {
    $redCount++;
  }
}
close ( INFILE );

print h2("Found $redCount matches for 'red'.");
print end_html();
```

Perl Stuff 2

- Arithmetic operators and comparisons
  - +, -, *, /, %
  - ==, !=, <, >, <=, >=
- String operators and comparisons
  - .
  - eq, ne, lt, gt, le, ge
- Literal strings and interpolated strings
  - ‘dog’
  - “$dog is a dog”
- Strings and numbers

Perl Basics

```
#!/usr/bin/perl
use strict;
use CGI qw( :standard );
use CGI::Carp qw(warningsToBrowser fatalsToBrowser);
print header(); print start_html('Perl basics'); print ('<p>');
my $x = 2 + 3;
my $y = $x * 4;
if ($x == 5.0) {
  print ("x is five");
}
for (my $i = 0; $i < 3; $i++) {
   my $squared = $i * $i;
   print ("<br/> \$i = $i, squared is $squared");
}
my $pet1 = "dog";
my $pet2 = "ll" . "ama";
# Single quotes vs. double quotes
print ("<br/>I have a $pet1 and a $pet2.");
print ('<br/>I have a $pet1 and a $pet2.');
my $comp1 = ($pet1 eq "dog");
print ("<br/> comp1: $comp1");
print ('</p>'); print end_html();
```

lect_io_array.pl

Exercise #2: What does this code do?

```
#!/usr/bin/perl
use strict; use CGI qw( :standard );
use CGI::Carp qw(warningsToBrowser fatalsToBrowser);

print header(); print start_html('Ex 2');

my $index = 0; my $sum   = 0; my @myArray = ();
my $filename = "numbers.txt";
open ( MYFILE, $filename ) or print h1("Error: could not open file $filename for
    reading");
while (my $aNum = <MYFILE>) {
  chomp $aNum;
  if ($aNum > 0) {
    $myArray[$index] = $aNum;
    $sum             += $aNum;
    $index++;
  }
}
close ( MYFILE );

# Add the sum to the array
$myArray[$index] = $sum;
$index++;

my $size = @myArray;
open ( MYFILE, ">$filename") or print h1("Error: could not open file $filename for
    writing");
for (my $i = 0; $i < $size; $i++) {
  print br() . $myArray[$i];
  print MYFILE $myArray[$i] . "\n";
}
close (MYFILE);
print end_html();
```

Exercise #3: Write Perl code that accepts two numbers from browser user, prints error if num2 is zero, otherwise outputs num1/num2.