IT452 Advanced Web and Internet Systems

Set 5: Perl and Database Connections

Assumptions

• You know Perl
  – We’ll review
  – Can use PHP instead

• (maybe??) You know how to use SQL
  – Help to be provided
Perl Basics

“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" );

Perl Function Calls (“subroutines”)

use CGI qw( :standard );
print header();
print start_html();

# Prints "hello", takes no arguments
sub hello {
  print "\n<br/> Hello.";
}

# Takes two arguments, return their product
sub multiply {
  my($valA, $valB) = @_;  
  return $valA * $valB;
}

my $x = 2;
print "\n<br/> $x * 7 = " . multiply($x,7);
hello();
hello(72145);

print end_html();
Arrays and Parameters

# Defines new global array, @array1
# AND returns a new array with 4 elements.
sub makeArray {
    my($first, @args) = @_; 
    @array1 = ($first, 23, 90); 
    my @array2 = (34, 5.4, 123, 2.01);
    return @args;
}

my @test1 = makeArray(1, 2, 3, 4);

print "\ntest1 is: @test1\n";
print "\narray1 is: @array1\n";
print "\narray2 is: @array2\n";

Database Example
Perl DB Queries

Connect to the DB.
my $databaseHandle = DBI->connect("STUFF");

Create your DB query.
my $query = "SELECT * FROM comments";
my $statementHandle = $databaseHandle->prepare($query);
$statementHandle->execute();

Process the results.
while ( my @row = $statementHandle->fetchrow_array ) {
    print "$row[0] \t $row[1] \t $row[2] \n";
}

Disconnect.
$databaseHandle->disconnect();
$statementHandle->finish();

Perl DB Queries (2)

my $c_id = param("comment_id");
my $databaseHandle = DBI->connect("STUFF");

my $query = "SELECT * FROM comments WHERE id=?";
my $statementHandle = $databaseHandle->prepare($query);
$statementHandle->execute($c_id);

while ( my @row = $statementHandle->fetchrow_array ) {
    print "$row[0] \t $row[1] \t $row[2] \n";
}

$databaseHandle->disconnect();
$statementHandle->finish();
Order of DB Results

my $query = "SELECT * FROM comments";
my $query = "SELECT id, username, comment FROM comments";

Array order.
while (my @row = $statementHandle->fetchrow_array) {
  print "$row[0] \t $row[1] \t $row[2] \n";
}

Hash Table order.
while (my $hashref = $statementHandle->fetchrow_hashref) {
  my %hash = %{$hashref};
  print "$hash{'id'} \t $hash{'user'} \t $hash{'comment'} \n";
}

Output to an HTML Page (ex2)

#!/usr/bin/perl
use strict; use CGI::Carp qw( fatsToBrowser );
use CGI qw(:standard );
use DBI; use DBD::mysql;
print header();  print start_html();
my $c_id = param("comment_id");
my $databaseHandle = DBI->connect("STUFF");
my $query = "SELECT * FROM comments WHERE id=?";
my $statementHandle = $databaseHandle->prepare($query);
$statementHandle->execute($c_id);

# put results in a table
print "<table border='1'> <thead>
  <th> ID </th> <th> User </th> <th> Comment </th></thead><tbody>
while (my @row = $statementHandle->fetchrow_array) {
  print "<tr>
  </tr>
  
}
print "</tbody> </table> <br/> <hr/>";
$databaseHandle->disconnect();
$statementHandle->finish();
print end_html();
Example – Output MATCHING to TEXT (ex3)

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

print( "Content-type: text/plain; charset=UTF-8\n\n" );

my $search = param("search");

my $databaseHandle = DBI->connect( "STUFF" );

my $query = "SELECT * FROM comments WHERE user LIKE '%$search%'";
my $statementHandle = $databaseHandle->prepare($query);
$statementHandle->execute;

# Output plain results. Separate lines with vertical bar
while (my @row = $statementHandle->fetchrow_array) {
    print "$row[0],$row[1],$row[2]|\n";
}

$databaseHandle->disconnect();
$statementHandle->finish();
```

---

Example – Get from DB, output ALL to XML (ex4)

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

print( "Content-type: text/xml; charset=UTF-8\n\n" );

my $databaseHandle = DBI->connect( "STUFF" );

my $query = "SELECT * FROM comments";
my $statementHandle = $databaseHandle->prepare($query);
$statementHandle->execute;

print "<results>\n";
while (my @row = $statementHandle->fetchrow_array) {
    print " <result>\n";
    print "  <id>$row[0]</id>\n";
    print "  <user>$row[1]</user>\n";
    print "  <comment>$row[2]</comment>\n";
    print " </result>\n";
}

print "</results>\n";

$databaseHandle->disconnect();
$statementHandle->finish();
```
**Headers Matter**

- **XHTML**
  ```
  print header();  print start_html();
  ...
  print end_html();
  ```

- **Plain Text**
  ```
  print("Content-type: text/plain; charset=UTF-8\n\n");
  ...
  ```

- **XML**
  ```
  print("Content-type: text/xml; charset=UTF-8\n\n");
  ...
  ```

---

**Example – Simple INSERT (ex5)**

```perl
#!/usr/bin/perl
use strict;
use CGI::Carp qw( fatalsToBrowser );
use CGI qw( :standard );
use DBI; use DBD::mysql;
my $user    = param("user");
my $comment = param("comment");
print header();  print start_html();
my $databaseHandle = DBI->connect( "STUFF");

# Do the SQL insert
my $query = "INSERT INTO comments (user, blurb) VALUES (?,?)";
my $statementHandle = $databaseHandle->prepare($query);
$statementHandle->execute($user, $comment);
print "<h2> SUCCESS -- inserted into the DB! </h2>";

# If the SQL fails, we won't necessarily know. Check here.
# Any errors? Print them here
print "<p>Errors, if any: $DBI::errstr</p>";

# Close up
$databaseHandle->disconnect();
$statementHandle->finish();
print end_html();
```
Perl “strict” mode

• Strict mode forces variables to be declared, and a few other things:
  – my $ii = 0;
  – my @someArray = (1,2,3);

• Required for all IT452 Perl scripts
• Will save you pain!
• Also use “Carp” mode shown in examples
  – Sends errors correctly to the client.

Example from before – what needs to change?

```javascript
// Make synchronous call to server to get data for a new row
function handleQuery() {
  xhr = window.ActiveXObject
  ? new ActiveXObject("Microsoft.XMLHTTP")
  : new XMLHttpRequest()

  // Get data from server
  xhr.open("GET", "dummy_data1.csv", false);
  xhr.send(null); // GET, so no "data" part

  // Deal with results
  if (xhr.status != 200) {
    alert("Error contacting server! Status: "+xhr.status);
  }
  else {
    // Get comma-separated data and make into an array
    var data  = xhr.responseText;
    var elems = data.split(",")

    // Make new row with this data
    insertRow(elems);
  }
  return false;  // false prevents the form from actually submitting
}
```