Project 1

- Deadline is approaching…
  - Review requirements
Concatenating Strings

Definition of concatenate

- Webster’s
  - To link together; to unite in a series or chain, as things depending on one another.

- Computing Dictionary
  - To join together two or more files or lists to form one big one

- Our Easy to understand definition
  - To join two or more text strings together in order to form one single text string.
Concatenating Strings

- Text Literal
  - “Hello World”
  - “SELECT * FROM Employee”
  - “Welcome Joseph”
Concatenating Strings in ASP

- Two Text Literals
  - “Hello,”
  - “how are you?”
- Concatenated:
  - “Hello,” & “how are you?”
  - Produces:
    - “Hello, how are you?”
Concatenating Strings in ASP

- Need to add a space:
  - “Hello,” & “ ” & “how are you?”
  - Produces:
    - “Hello, how are you?”

- Notice & is used to concatenate
  - & joins two (or more) strings into one string
Concatenating Strings in ASP

- **Using variables**
  
  ```vbscript
  Dim greeting, question, finalGreeting
  greeting = "Hello,"
  question = "how are you?"
  finalGreeting = greeting & question
  ```

- **Produces:**
  - "Hello, how are you?"
Solution 1 (Using variables)
Dim greeting, question, space, finalGreeting

- greeting = "Hello,"
- question = "how are you?"
- space = " "

finalGreeting = greeting & space & question

Produces:
- "Hello, how are you?"
Solution 2 (Using variables)

Dim greeting, question, finalGreeting

greeting = “Hello, ”
question = “how are you?”

finalGreeting = greeting & question

Produces:
- “Hello, how are you?”
Dim SQL, addressID
addressID = Request.Form("addressID")

SQL = "SELECT * FROM Address WHERE"
SQL = SQL & "AddressID='" & addressID
SQL = SQL & "' AND Login='"
SQL = SQL & Session("Login") & "'"

Produces (all on one line):
"SELECT * FROM Address WHERE AddressID='formValue'
AND Login='sessionValue'"
Concatenating Strings in PHP

- **Two Text Literals**
  - “Hello,“
  - “how are you?”

- **Concatenated:**
  - “Hello,“ . “how are you?”
  - **Produces:**
    - “Hello,how are you?”
Concatenating Strings in PHP

- Need to add a space:
  - "Hello," . " " . "how are you?"

- Produces:
  - "Hello, how are you?"

- Notice . is used to concatenate
  - . joins two (or more) strings into one string
Using variables

$greeting = "Hello,";
$question = "how are you?";

$finalGreeting = $greeting.$question;

Produces:
- "Hello, how are you?"

Concatenating Strings in PHP
Solution 1 (Using variables)

```php
$greeting = "Hello,";
$question = "how are you?";
$space    = " ";

$finalGreeting = $greeting.$space.$question;
```

Produce:
- "Hello, how are you?"
Solution 2 (Using variables)
$greeting = "Hello, \"; 
$question = "how are you\"; 

$finalGreeting = $greeting . $question;

Produces:
- "Hello, how are you?"
Concatenating Strings in PHP

```php
$addressID = $HTTP_POST_VARS["addressID"];  
$sql = "SELECT * FROM Address WHERE";  
$sql = $sql . "AddressID='" . $addressID;  
$sql = $sql . " AND Login='";  
$sql = $sql . "$_SESSION["Login"]'";  

Produces (all on one line):

"SELECT * FROM Address WHERE AddressID='formValue' AND Login='sessionValue'"
```
while counter = 0
while counter < 7
    Response.Write(“Counter: ” & counter)
    Response.Write(“<br>”)
    counter = counter + 1
wend

How many times does this loop?
- 7 times
  - 0,1,2,3,4,5,6
  - The final value of counter is 7
while counter = 0
while counter < 7
    Response.Write(“Counter: “ & counter)
    Response.Write(“<br>“)
wend

This is an infinite loop
- Variable counter is never incremented
  - This loop will never terminate unless the application is terminated
RecordSet example

while not oRS.EOF
    Response.Write(Trim(oRS.Fields("Login")))
oRS.MoveNext()
wend

Loops through each record (row) in the recordset
- Prints the Login field of each record (row) in the recordset
- Continues to loop until the last record (row) has been looked at
do while
   counter = 0
   Do
      If counter = 6 Then
         Exit Do
      Else
         Response.Write("Counter: ") & counter
         Response.Write("<br>")
      End If
      counter = counter + 1
   While counter <= 9
Do While *always* executes at least once

- Other than that, it is fundamentally the same as a while loop

Exit Do

- If the counter is equal to 6, the loop is exited without executing any other lines of code within the loop
- Means that the counter will never reach 7, it will always exit when equal to 6
**For... Next**

counter = 0
For counter <= 9 Step 1
    If counter = 6 Then
        Exit For
    Else
        Response.Write("Counter: ");
        Response.Write(counter)
        Response.Write("<br>")
    End If
End If
Next
For Each… Next

For Each item in Request.ServerVariables
    Response.Write(item)
Next

item is a variable
- Value changes each time through the loop
PHP: Repetition Control Structures

- while
  $counter = 0;
  while(counter < 16)
  {
    echo "Counter: " . $counter;
    echo "<br>";
    $counter = $counter + 1;
  }

- How many times does this loop?
  - 16 times
    - 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
    - The final value of counter is 16
while
$c\text{ounter} = 0;
while(counter < 16)
{
    echo "$\text{Counter: }" . $\text{counter};
    echo "<br>";
}

- This is an infinite loop
  - Variable $\text{counter}$ is never incremented
    - This loop will never terminate unless the application is terminated
while
$counter = 0;
while (counter < 16)
{
    echo "Counter: " . $counter;
    echo "<br>";
    $counter++;
}

Notice the use of ++
- This is a post-increment operator
- This is the same as:
  - $counter = $counter + 1;
Increment & Decrement Operators

- **preincrement**
  - $++a$;

- **postincrement**
  - $a++;$

- **predecrement**
  - $--b$;

- **postdecrement**
  - $b--;$

- Increment **a** by 1 then use the new value of **a** in the expression in which **a** resides.

- Use the current value of **a** in the expression in which **a** resides, then increment **a** by 1.

- Decrement **b** by 1 then use the new value of **b** in the expression in which **b** resides.

- Use the current value of **b** in the expression in which **b** resides, then decrement **b** by 1.
Recordset Example

```php
//Create the SQL query
$sqlquery = "SELECT DISTINCT companyName FROM Customers;";

//execute the SQL query and store the result of the execution into $result
$result   = mysql_query($sqlquery);

//Loop through the results
while($row=mysql_fetch_array($result))
{
    //echo out the value of the column (field) pulled from the database
    echo $row["companyName"]."\n";
}
```

Loops through each record (row) in the recordset
- echos the companyName field of each record (row) in the recordset
- Continues to loop until the last record (row) has been looked at
**PHP: Repetition Control Structures**

- **do while**
  ```php
  $num = 100;
  do
      echo "Number: " . $num . "<br/>";
      $num++;
  while($num <= 9)
  ```

- This code will be executed once before the condition of the while loop is checked.
for Loop

```php
for($i=0; $i<5; $i++)
{
    if($i == 4)
        exit;
    else
        echo "i: " . $i . "<br/>";
}
```
foreach Loop (used with arrays)

//initialize array
$products = array('Tires', 'Oil', 'Spark Plugs');

//loop through $products
foreach($products as $current)
{
    echo $current . "<br/>";
}
Permissions

- Example of setting permissions on a folder
- Example of setting permissions on a file
Permissions

- Internet Guest Account
  - IUSR_<machineName>
- ASPNET
  - aspnet
- Everyone
  - Used for FTP & other access
ASP: Trim()

- Use Trim() when pulling information from the database

```<%= Trim(oRS.Fields("LastName")) %>```
Use `trim()` when pulling information from the database

```php
<?php
    echo trim($row["LastName"]);
?>
```
Data pulled from DB

- Data pulled from a DB always has the number of characters that the DB field was created to hold.
  - Our Login field holds 15 characters
    - Store “rjg” into the DB
    - Retrieve login from DB
      - Returns “rjg ”
When comparing:

```
If login = Trim(oRS.Fields("Login")) Then
...
End If
```
When comparing:

```php
if ($login == trim($row["Login"])) {
    ...
}
```
In HTML somewhere...

```html
<form name="form0"
    action="getPost.asp" method="post">

    <input type="text" name="firstName">
    <input type="text" name="lastName">

</form>
```
In `getPost.asp` somewhere...

Dim firstName, lastName
firstName = Request.Form("firstName")
lastName = Request.Form("lastName")

- Try the following sometime:

  Response.Write(Request.Form)
In `getPost.php` somewhere...

```php
$firstName = $_POST['firstName'];
$lastName = $_POST['lastName'];
```

- Try the following sometime:
  ```php
echo $_POST;
```
- Or perhaps a foreach loop to echo it out
In HTML somewhere...

```html
<form name="form0" action="getGet.asp"
    method="get">

    <input type="text" name="firstName">
    <input type="text" name="lastName">

</form>
```
ASP: QueryString

In getGet.asp somewhere...

Dim firstName, lastName
firstName = Request.QueryString("firstName")
lastName = Request.QueryString("lastName")
In `getGet.php` somewhere…

```php
$firstName = $HTTP_GET_VARS["firstName"];  
$lastName = $HTTP_GET_VARS["lastName"];  
```

- Try the following sometime:
  ```php
echo $_GET;
```
- Or perhaps a foreach loop to echo it out
Try this sometime…

- In HTML somewhere:

  ```html
  <a href="getGet.asp?firstName=Ron&lastName=Glotzbach">Testing QueryString</a>
  Or in PHP:
  <a href="getGet.php?addressID=3">Testing QueryString</a>
  ```

Opening possibilities

- This should spark even more ideas in your head for your project.
- These two examples are *static* – think *dynamic*!
ASP: OPTION EXPLICIT

- Means that all variables in the page must be dimensioned.
- The page will crash if you have a variable that is not dimensioned.
- This is not necessary, but recommended.
ASP: Response.Buffer = True

- Means that output from ASP is buffered until the entire page is loaded.
- Only use this on pages that contain HTML (or output something to the browser).
- Buffering the output allows you to flush its contents whenever you choose to:
  - Response.Flush
ASP: Response.Buffer = False

- Use this setting on pass-through pages
- Response.Buffer is not necessary in ASP, but it is recommended.
ASP: On Error Resume Next

- Tricky
  - Why isn’t my page EVER crashing, even when I know it should crash??
    - This is probably why.

- Tells the server not to crash on an error and to resume processing at the next line of code.
ASP: On Error Resume Next

- This refers to runtime errors, NOT syntax errors or compile time errors.
  - For example, divide by zero is a runtime error
  - A spelling error is a syntax error caught at compile time.
Again, this is not necessary in ASP, but recommended where appropriate.

Be careful using it. It may make it much harder for you to debug.

Consider leaving it out of your code until you have finished the page.