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 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**
  
  ```
  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?"

Concatenating Strings in PHP

- **Two Text Literals**
  
  - "Hello,"
  - "how are you?"

- **Concatenated**
  
  - "Hello," . "how are you?"
  - Produces:
    
    - "Hello, how are you?"

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

- **Notice . is used to concatenate**
  
  - . joins two (or more) strings into one string
### Concatenating Strings in PHP

#### Using variables
```
$greeting = "Hello,";
$question = "how are you?";
$finalGreeting = $greeting.$question;
```

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

#### Solution 1 (Using variables)
```
$greeting = "Hello,";
$question = "how are you?";
$space = " ";
$finalGreeting = $greeting.$space.$question;
```

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

#### Solution 2 (Using variables)
```
$greeting = "Hello, ";
$question = "how are you?";
$finalGreeting = $greeting.$question;
```

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

### Solution (Using variables)
```
$addressID = $HTTP_POST_VARS["addressID"];
$sql = "SELECT * FROM Address WHERE ";
$sql = $sql . "AddressID='" . $addressID . "' AND ";
$sql = $sql . "Login='" . $_SESSION["Login"] . "'";
```

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

### ASP: Repetition Control Structures

```
while counter = 0
    while counter < 7
        Response.Write("Counter: " & counter)
        Response.Write("<br>")
        counter = counter + 1
    wend
wend
```

**How many times does this loop?**
- 7 times
- 0, 1, 2, 3, 4, 5, 6
- The final value of counter is 7

**This is an infinite loop**
- Variable counter is never incremented
- This loop will never terminate unless the application is terminated
### ASP: Repetition Control Structures

- **RecordSet example**
  ```vbscript
  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 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

### ASP: Repetition Control Structures

- **For... Next**
  ```vbscript
  counter = 0
  For counter <= 9 Step 1
    If counter = 6 Then
      Exit For
    Else
      Response.Write("Counter: " & counter) Response.Write("<br>"
    End If
  Next
  ```

- **For Each... Next**
  ```vbscript
  For Each item in Request.ServerVariables
    Response.Write(item)
  Next
  ```

### PHP: Repetition Control Structures

- **while**
  ```php
  $counter = 0;
  while($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
### PHP: Repetition Control Structures

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

- **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.

### Increment & Decrement Operators

- **preincrement**
  ```php
  ++$a;  
  ```
- **postincrement**
  ```php
  $a++;  
  ```
- **predecrement**
  ```php
  --$b;  
  ```
- **postdecrement**
  ```php
  $b--;  
  ```

### 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"] . "<br>
}
```
foreach Loop (used with arrays)

```php
//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

```asp
<%= Trim(oRS.Fields("LastName")) %>
```

PHP: trim()

- 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    "
**ASP: Trim() continued**

- When comparing:

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

**PHP: trim() continued**

- When comparing:

```
if($login == trim($row["Login"]))
{
    
}
```

**Form**

- In HTML somewhere...

```
<form name="form0" action="getPost.asp" method="post">
    
    <input type="text" name="firstName">
    <input type="text" name="lastName">

</form>
```

**ASP: Form**

- In getPost.asp somewhere...

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

- Try the following sometime:

    Response.Write(Request.Form)
```

**PHP: Form**

- In getPost.php somewhere...

```
$firstName = $HTTP_POST_VARS["firstName"];  
$lastName = $HTTP_POST_VARS["lastName"];  

- Try the following sometime:

    echo $_POST;

- Or perhaps a foreach loop to echo it out

```

**QueryString**

- In HTML somewhere...

```
<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")

PHP: Querystring

- In getGet.php somewhere...
  
  $firstName = $HTTP_GET_VARS["firstName"];
  $lastName = $HTTP_GET_VARS["lastName"];

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

QueryString

- Try this sometime...
  - In HTML somewhere:
    <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.

ASP: On Error Resume Next

- 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.