Introduction to Active Server Pages

INFORMATION RESOURCE CENTER

Introduction to Active Server Pages

Information Resource Center
USCG Training Center Petaluma
599 Tomales Road
Petaluma, CA 94952
Telephone (707) 765-7523

Table of Contents

Section 1

Overview of ASP

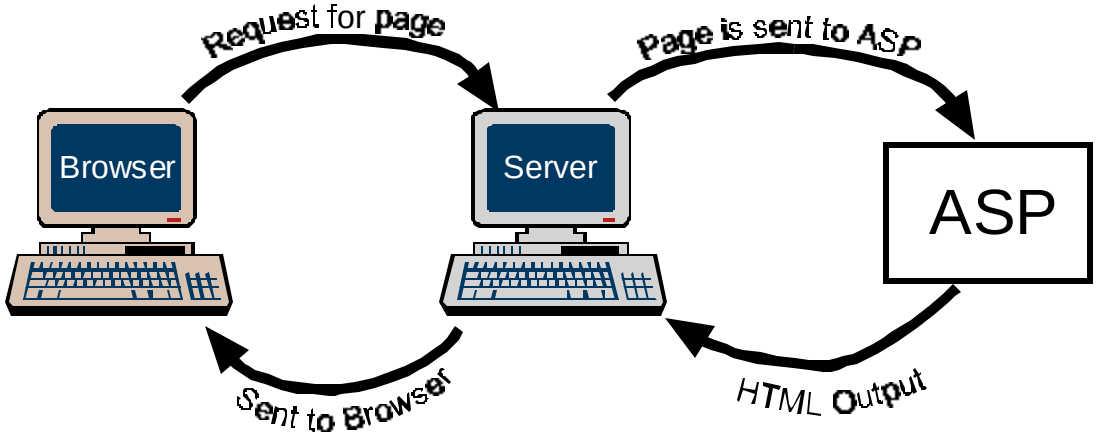
What is ASP?

ASP is a server-side scripting environment that can be used to create dynamic Web pages or web based applications. ASP pages are files that contain HTML tags, text, and server side scripts and end with the .asp extension. Scripts in an ASP page can access built-in ASP objects or external components. The external components come in the form of ActiveX components that perform many tasks such as connecting to a database.

ASP allows you can add interactive content to your Web pages or build entire Web applications that use HTML pages as the user interface.

How ASP Works

ASP technology is entirely server based meaning that the server processes all the script commands. When a user sends a request for a an ASP page, the web server executes the script code prior to sending the page back to the browser. The result is that the browser only gets a normal web page that is a result of the scripts that were executed. The diagram below illustrates how this works.



This illustration is an overly simplified example of how ASP works. In actuality the process is more complicated. Right now it is not necessary to understand all the intricacies of ASP and how

it works with the web server, you will learn more about it as you create more robust web applications. For now just think of ASP as a web server add-on that processes the page before it is sent to the browser.

Internet Information Server

Internet Information Server (IIS) is Microsoft's web server software for Windows NT Servers. ASP is one of the many additions to the Internet Information Server. There are many other server additions that add a variety of functionality to IIS. This tutorial does not discuss other components or how they are integrated into IIS. Rather this tutorial deals only with how to create ASP pages. However, some general information on components and objects will be provided.

Components

Components are separate, small pieces of software that are developed separately, but can be integrated into an 'application'. Think of how a PC is put together- numerous components from numerous manufacturers that are brought together to form a computer. That's generally how IIS is put together. There are many components that are added to IIS to make a robust web server. ASP pages can directly access those components to perform specific functions.

For example there is a component that allows a web page to retrieve, change, or add records in a database. Components are called by scripts in the ASP page. IIS manages the communication between the web browser and the database through the ASP component and the database component.

Objects

Because ASP is a scripting environment it really only knows how to deal with Objects. (Objects are things you can manipulate in a scripting environment.) Using scripting commands you can create objects for each component you wish to access. **Objects** are created instances of the component. Objects are represented by *properties*, *methods*, and *events*. Fortunately creating objects to access components is rather simple. Additionally ASP has six built in objects that can be accessed at anytime. These objects will be discussed later.

Web Applications

ASP pages are commonly referred to as web applications. **Applications** are really just a collection of ASP pages and any components they use that are grouped together to accomplish a specific task. In this class we will not be creating any large scale applications, but the term will be used to refer to the pages created in this tutorial. Just think of applications as groups of ASP pages that accomplish some task.

Built-in Objects

As mentioned earlier, ASP has six built in objects that you can access and use in your pages. These objects allow you to extend the power of scripts and add functionality to your ASP pages. This tutorial will not cover all the objects, but will use the Response and Request objects as examples of

how to access and use objects. A reference of these six objects and their associated properties, events, methods and collection is included in Appendix B. You can also find a reference for the objects at http://cgweb.tcpet.uscg.mil/iishelp/iis/htm/asp/intr1orp.htm.

Request

The Request object contains all the data that is include in an HTTP request for a page. This includes any HTTP headers and form data. The ASP pages you will create in later exercises will use the Request object to access form data.

Response

The Response object contains the data that is sent from the web server to the browser. The Response object is most commonly used to output data to a page before it is sent to the browser.

Server

The Server object is used to access utilities on the server. The most common task is to set the timeout property so that scripts with errors don't get stuck in infinite loops and overload the server.

Application

The Application object is used to store and retrieve data that is shared among the users of an ASP application.

Session

The Session object is used to store and retrieve data pertaining to the current user session. Sessions are activated when a user request a page and persist for a specified amount of time or until the user closes the browser.

ObjectContext

The ObjectContext object is used to control ASP transactions that are managed by the Microsoft Transaction Server (MTS). MTS is a component of IIS much like ASP is a component of IIS.

Getting Help

A Scripter's Reference for VBScript is available online at http://cgweb.tcpet.uscg.mil/iishelp. To locate the reference follow these steps.

1. Go to the web site
2. Once at the web site, on the left side of the page click the plus (+) sign next to **Internet Information Server**. The list will expand to show more options.
3. Click the plus (+) sign next to **Scripter's Reference**. Another list of options will open. You will see the **VBScript Language Reference**.

From there you can click **VBScript Language Reference** or click the plus (+) sign for more options.

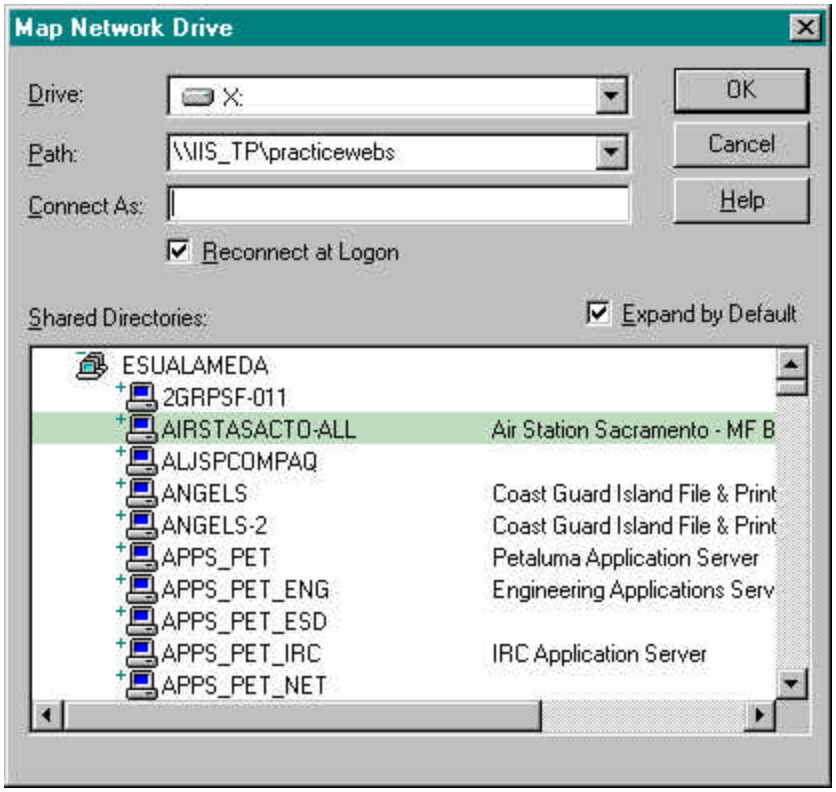
The language reference does not provide any tutorials or instructional content, it simply provides the syntax for various operators, objects, properties and methods.

An online tutorial for ASP is also available on the IISHelp site. To access the tutorial follow steps 1-2 above. Then click the plus (+) sign next to **Web Applications**. You will see the **ASP Tutorial** in the list that appears**.**

Getting Started

In order view the ASP pages correctly the pages must reside on a web server. A folder on the web server has been set up for your use during this class. In order to view your pages you will need to connect to the server and save all your work to the server. Follow these steps to connect to the server:

1. Right Click **My Computer**.
2. From the pop-up menu select **Map Network Drive**. The **Map Network Drive** window will open.
3. In the Drive box select X:\ **NOTE**: You must use X:\
4. In the path box type **\\iis_tp\practicewebs**
5. Click the box next to **Reconnect at Logon**. The box will be checked. Your screen should look **exactly** like the example below.



6. Click **OK**. The drive will be connected. A window for the drive should open.

Viewing Your Pages

Once you have created pages you will view them live from the server using Internet Explorer. The URL for your pages will be:

http://cgweb.tcpet.uscg.mil/practicewebs/*username*/*file.asp*

where *username* is your username and *file.asp* is your filename.

Creating Pages

This workshop will use FrontPage to create most of the ASP pages done during the exercises. Before you begin you will need to create a FrontPage web on the X:\ drive. To do this follow these directions:

1. Open Internet Explorer.
2. Go to http://cgweb.mlcpac.uscg.mil/citrix/citrix.htm.
3. Click the **FrontPage** link. **DO NOT** click FrontPage 2000.
4. A window will open for you to enter your username and password. Enter them and click **OK**. FrontPage will open showing the **Getting Started** window.
5. Click **Create a New FrontPage Web**.
6. Click **OK**. The **New FrontPage Web** window will open.
7. In the list of templates click **Empty Web**.
8. Click the **Change** button.
9. Enter the path to your web. Your web will be saved on your X:\ drive. Use the path given below, but insert your user name in place of *username*. **X:\username**
10. Click **OK**.
11. Enter your username as the title for the web.
12. Click **OK**.
13. You will get a message stating the folder does not exist and would you like to create it. Click **Yes**. Your web will be created.

You are now ready to begin crating ASP pages.

Section 2

Creating ASP Pages

Inserting script tags

ASP uses scripting delimiters <% *script* %> to denote when scripts start and stop. Commands enclosed in the delimiters are primary script commands and are processed using the primary script language. An example of how script delimiters are used is shown below.

```
<%
dim var mytime
mytime = date()
response.write mytime
%>
```

You can also use the standard script tags as shown below.

```
<script language =”VBScript” runat=”server”>
   some script commands
</script>
```

There is a difference in how ASP process the script tag compared to the scripting delimiters. The primary difference is that anything in the standard script tag is execute when the page loads, regardless of where in the web page the script is located. The script tag is generally reserved for functions and subroutines that are called from other scripts. You can also use the script tag to mix scripting languages. For example if your primary scripting language is VBScript, you can use the script tag to insert some JScript code in a page.

Scripting Languages

ASP pages can be written using either JScript or VBScript. JScript is Microsoft's version of Javascript and is essentially the same as Javascript. However, there are a few differences. If you plan to use JScript you should consult Microsoft's scripting web site for more information- http://msdn.microsoft.com/scripting/default.htm?/scripting/jscript/techinfo/jsdocs.htm.

VBScript is much simpler than JScript and is easier to learn. This tutorial will use VBScript as the default for creating ASP Pages. For information on VBScript see the IISHelp web site mentioned in the previous section.

ASP Directives

ASP has two special script delimiters known as *directives*.

The **Output directive** provides an equivalent to the response.write statement (to be discussed later), but is much shorter and easier to use. The output directive is only used when you want to insert something directly into a page. It cannot be used within a script. An example is shown below.

```
<%= time %>
```

Provides the same thing as:

```
<% Response.Write time %>
```

The **Processing directive** is used to set scripting language for a page and when used must be the first line of your ASP page.

```
<%@ language = vbscript %>
<HTML>
<HEAD>
```

You do not have to set the scripting language in this way, but it is a good idea and will likely reduce errors.

Mixing Scripts and HTML

Scripts used in ASP pages can be placed anywhere in a page, and most likely your pages will be littered with pieces of script. In general it is a good idea put any functions or subroutines in the `<Head>` of the page as well as any variable declarations for variables that will be used later in the page.

In most cases you will need to put the script right in the middle of the HTML. When you do this whatever the output of the script or directive is will be displayed whereever the script is located. An examples is shown below.

```
<h1> Hello.  The time is <%= time %></h1>
```

This script inserts the current time directly into the web page.

Exercise – Creating an ASP Page

In this exercise we will create an ASP page that will display a different greeting depending on the time of day.

1. Open a new page in FrontPage.
2. Change to HTML view.
3. In the body of the page enter the following code:
   `<h2><%`

```
dim mytime
mytime = now
if hour(mytime) > 12 then
      Response.Write "Good afternoon"
else
      Response.Write "Good morning"
end if
%>, This is an introduction to ASP</h2>
```

4. Save the page as test.asp
5. Open Internet Explorer and open the page by entering the URL in the address bar. The URL should be http://cgweb.tcpet.uscg.mil/practicewebs/*username*/test.asp where *username* is your username.
   NOTE: You may want to bookmark this page for easy access later on.

When you view the page, you should see the appropriate greeting.

Examining the Code

The script uses standard VBScript commands to display a custom greeting based on the time of day. The script starts with the script delimiter tag <%. A variable was created that contains the current time. An *If…then* statement is used to determine if the time of day is before 12 noon, if the time is before noon "Good morning" is display, otherwise "Good afternoon" is displayed.

The script takes several lines to accomplish the custom greeting, but when the page is displayed, only the text of the message is returned to the browser. So even though the script use several lines in the code, only two words get sent to the browser. This script is an example of an inline script that generates text in the browser window. Next we'll create a script that changes an HTML tag.

Exercise – Using Scripts in Tags

This script will use a *for…next* loop to create several lines in a web page. Scripts used by ASP can be spread through out a page with HTML between the script statements. This allows the script to change or interact with the HTML easier than with client side scripts.

1. Return to FrontPage and add the blank line after the </h2> tag from the previous exercise.
2. Type the script delimiter to start the script and press **Enter** to create a new line.
3. Enter the following text:
   ```
   For i = 1 to 7 step 1
   ```
4. Enter the end script delimiter.
5. Enter the following html tag and script statements:
   ```
   <font size =”<%= i %>”>This is font size <b><%= i %> </b>
   </font> <br>.
   ```
6. Press enter to create a new line and enter the following script:
   ```
   <% next %>
   ```
7. Save the page and preview it in Internet Explorer.

When you are done your code should be similar to this:

```
<% For I = 1 to 7 step 1
%>
<p><font size="<%= i %>">This is font size <b><%= i %></b></font>. <%
next %> <br>
</p>
```

(FrontPage will insert the <P> and </P> tags when the page is saved.)

Examining the Code

This script uses a *For...next* loop to create a new line for each font size. The first statement starts the *For…next* loop. An HTML tag that includes a piece of script code is inserted into the page. We then include some text and the output directive for the variable **i**. We then close the *For…next* loop.

With a *For…next* loop everything will be repeated for as many times as the loop repeats. In our case, the loop repeats seven times. The result is that seven lines are outputted as HTML and the font size is increased in every line.

The point of this exercise is to illustrate how you can spread a script out and include bits and pieces of scripts throughout a web page. Remember that the server processes the script before sending the web page to the browser, so all the pieces of code are treated separately from the HTML. The browser only sees the HTML output of the script code.

Scripting Conventions

White Space

ASP ignores white space in scripts, but preserves white space in quotation marks. ASP also strips white space between two or more output directives. Below are a few examples of how ASP deals with white space. These example refer specifically to scripts written in VBScript.

Example 1- White space in quotes

```
<%  fullname= “G   eo   r    g   e  Wash  in g   ton” %>
```

Result: Spacing is preserved.

Example 2- White space between output directives

```
<% name =”George”
lastname=”Washington” %>
<%= name %>               <%=lastname%>
```

Result: GeorgeWashington

If you want to have space between output directives you will need to include the special character for blank space * *. This is an HTML character and is ignored by ASP. Below is an example.

```
<%=name%> <%=lastname%>
```

Result: George Washington

Example 3- White space in script commands

```
<%       name="George"
last = "Washington"                                    %>
```

Result: All white space is ignored. The return between lines in required to separate statements.

Case Sensitivity

VBScript is not case sensitive so you don't have to worry about what should or should not be capitalized in your statements. JScript, however, is case sensitive. This is another reason why VBScript is easier to use than JScript. This training will not use case sensitive script commands. For a reference on the case of script commands, please see the IIS Help web site at http://cgeweb.tcpet.uscg.mil/iishelp.

Comments

Comments in scripts provide a way for the person writing the script to leave references to others who might be editing the script at a later time. Virtually every scripting or programming language provides a way to include comments and VBScript is not exception.

To include a comment just type a single quotation mark and your comment. Everything after the quotation mark will be a comment. You can include comments on a line by themselves or at the end of a statement. Just remember to put the single quotation mark at the point where you want the comment to start. Below are some example of comments.

```
<%
for i = 1 to count step 1 ' this is a comment
'that line started a for next loop
response.write "Hi"
Next %>
```

In both cases the comments will not cause any scripting errors. However, if you put a comment in an output directive you will get an error. The following script would not work.

```
<%= name 'output the name variable %>
```

Section

3

ASP Objects

Introduction

Up to this point we have only done a couple of scripts that could have been done on the client side and not have used ASP. The real power of ASP comes from the objects that you can manipulate before the page gets sent to the browser. As mentioned earlier, ASP has six native objects available for use. These objects are always available and can be called on at any time. There are many other objects available, but they require additional coding to create an instance of the object. For now we will stick to the native objects. In particular this section will introduce the **Response** and **Request** objects.

The World Wide Web is driven entirely by the request/response paradigm. When you click a hyperlink on a web page a request is sent to the server for that page. The server then sends back a response. Hopefully the response is the page you requested, but occasionally the response may be an error. ASP has objects for the information contained in the request and response for web pages. The Request and Response objects correspond to HTTP request and responses for web pages.

You have already used the Response object in your scripts with the *Response.Write* statement. This statement essentially tells the server to send some information back as part of the response. The exercises in this section will use both the Response and Request objects.

When you are done with this section you will have created a set of ASP pages that use the Response and Request objects to process HTML forms.

Working with Objects

Objects in ASP work much the same as browser objects used by VBScript and JavaScript. Objects are the things that you can manipulate or change and have specified properties, events, and methods. (Properties describe the object and methods are the actions that can be taken on the object.) In addition many ASP objects also have associated **collections**. The collections are like subparts of the object that can be accessed.

ASP objects are referred to like browser objects. The object is stated followed by a period, then the method or property. An example is shown below.

```
Response.Write parameters
```

In this example the object is the **Response** object and the method is the **Write** method. The parameters are what you would want to be written by the response object.

Setting and retrieving properties uses the same format.

Response Object

The Response object corresponds to the response that is sent from the server to the browser after a request is made. Logically, it may more sense to talk about the response after talking about the request, but since we've already be using the Response object we will start there.

Overview of Response Object

The response object has the following methods, properties and collections.

Response Object

Collections:	**Methods:**	**Properties:**
Cookies	AddHeader	Buffer
	AppedToLog	CacheControl
	BinaryWrite	Charset
	Clear	ContentType
	End	Expires
	Flush	ExpiresAsolute
	Redirect	IsClientConnected
	Write	PICS
		Status

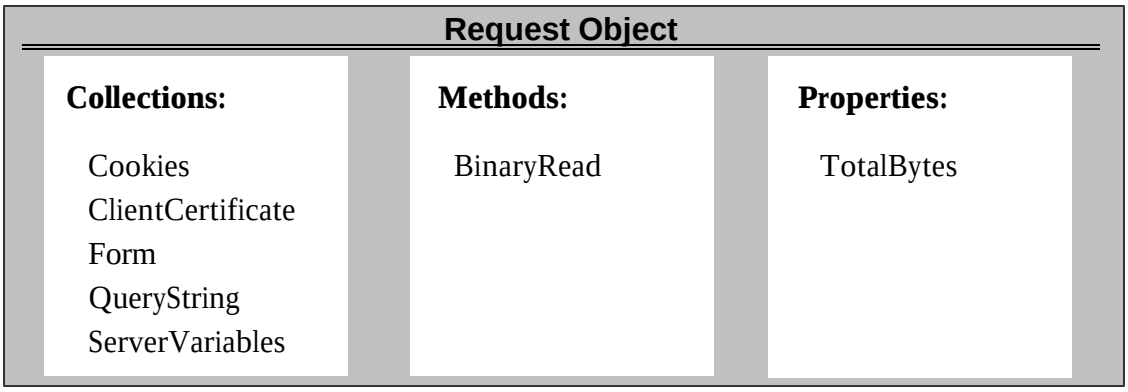
At this time we will not cover every collection, method, or property. Rather, we will use the *Write* method in our exercises to output data as part of the response back to the browser. The *Write* method is used to actually write text to the response that is sent to the browser. The text is written to the web page at the point where the script is inserted.

Request Object

As mentioned earlier the Request object corresponds to the request sent by the browser for a specific page. The Request object is used when you want to do something with the request for a page. This includes processing the contents of a form or analyzing request headers.

Request Object Overview

The Request object has the following collections, methods and properties:



In this tutorial we will further explore the Form collection. The other collections, methods, and properties will not be covered in this tutorial. If you would like more information on them please see http://help.activeserverpages.com/iishelp/iis/htm/asp/intr1orp.htm.

Section

4

Creating a Form Handler

The remainder of this tutorial will walk you through the process of creating a set of ASP pages that process the input from a form. These exercises will use the Response and Request objects. It is important to note that the data from the form will not be stored anywhere once it is processed. That will discussed in a later tutorial on using databases with ASP. These exercises will cover the basics of what you will need to know in order to create more advanced ASP applications.

The concepts that will be covered in these exercises include:

- Creating forms in FrontPage
- Passing form data to an ASP page
- Passing data from one ASP page to another using hidden fields
- Using conditional statements
- Including files in ASP pages
- Outputting data to a web page

Project Overview

A sample of the project you will create can be found at http://cgweb.tcpet.uscg.mil/webstuff/training/asptutorial/project1.asp. The first page of the project contains a form that will start the data capture process. The data will then be passed to a form processor which will display further information based on the initial data. The user is walked through several pages that are used to collect various types of data. This project is based on the idea of collecting information that will be stored in a database. Let's take a look at what the database would actually include.

Database Overview

Since we are using a database as our example let's look at the data we want to capture:

- First name
- Last name
- Unit
- Classification (Enlisted, Officer, Civilian, Other)
- Rank (Enlisted and Officer only)
- Rate (Enlisted only)

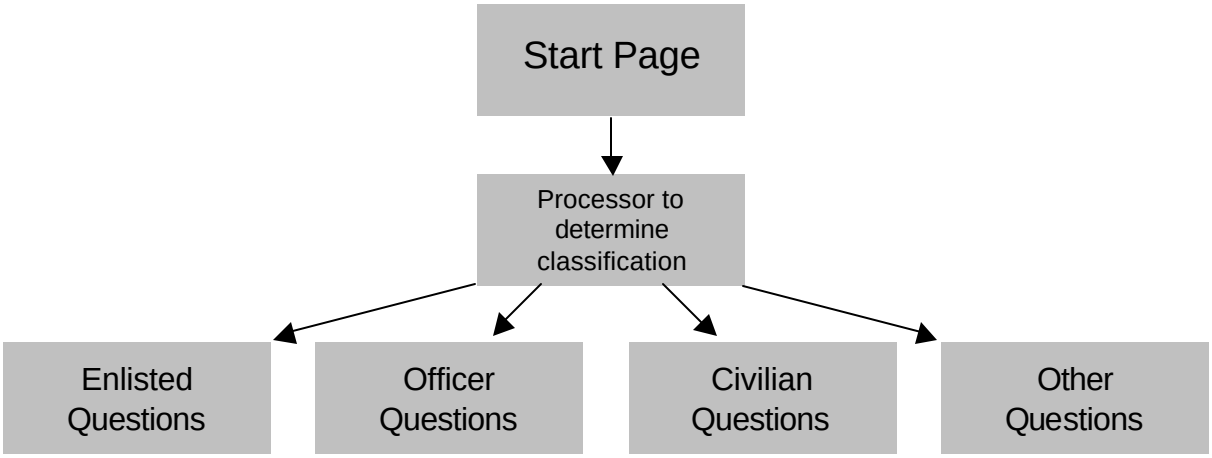
- Did they attend the Academy or OCS? (Officer only)
- Do they have a college degree?
- If so what kind? (Associate, Bachelor's, Master's, Doctoral)
- What school did they receive the degree from?
- How many years have they been in the Coast Guard? (enlisted and officers only)
- How long have they worked in their current job? (civilians and other)

As you can see by this list, several of the data fields depend on the classification of the respondent. As a result we will need to create different pages to collect the necessary data based on the classification.

Outline of ASP Pages

Based on the data we want to collect there will be four sets of pages used to collect data- Enlisted, Officer, Civilian, and Other. However, to make it easier for the respondent we will have ASP choose the correct set of questions to ask. Our pages will need to contain several conditional statements to determine which set of pages a particular respondent should be using.

Additionally, we need to make sure the questions are sequenced properly. For example, we should ask if they have a college degree before asking what kind of degree they have. Below is a flow chart of how the pages will work.



The first place we need to start is with the Start Page. Once that is done, we can work on each of the four branches.

We should note again that we will not be actually writing data to a database in these exercises. Other tutorials will discuss working with database. These exercises are only designed to show you how to use the Response and Request objects.

Starting the Data Collection

Now that you have some idea of what the outcome of the page should be, it is time to start creating the ASP pages that will handle the form data. The first thing we need to do is create a page to capture the initial data that is common among the four classifications. This page will actually be a regular HTML page with a form.

About Forms

Names and Values

Many of the ASP applications you create will use forms of some kind. The keys to creating forms that ASP can use effectively are the **Name** and **Value** attributes. When the form data is passed to an ASP page the data is organized according to the **Name** of the field.

Also, when you refer to form elements in a script you must refer to them by name. VBScript and ASP treat each form field as a separate object that is identified by name. When you use radio buttons and checkboxes you are actually creating a collection of fields with the same name. Only the value of the radio button or checkboxes that are selected is sent as form data.

The bottom line is that you must make sure you give each form element a unique name.

Action and Method

Another Key aspect to forms is the **action**. The **action** is what happens to the form data when the **Submit** button is clicked. The **Submit** button actually sends an HTTP request to the server that includes all the data from the form. All the data is given to whatever resource is identified by the action. In our case we will identify an ASP page as the action. What will actually happen is that the data will get sent to an ASP page the will act on the data.

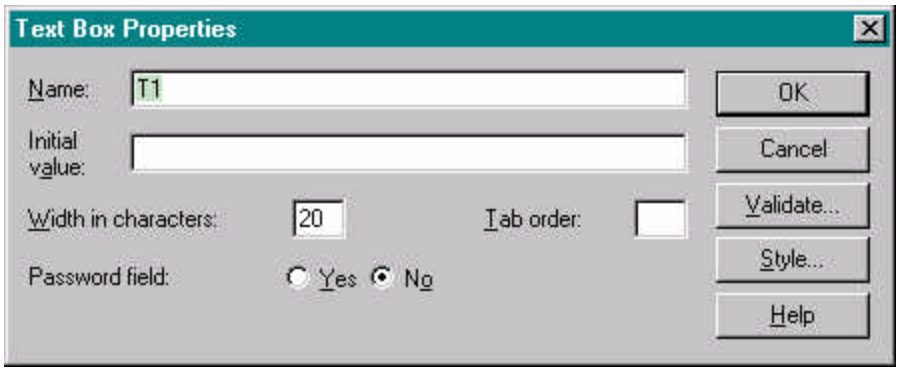
The **method** is how the data is sent. There are only two types of methods- **get** and **post**. Each method sends the form data to the server differently, but the end result is usually about the same. There are some technical differences, but we don't need to discuss at this time. For our purposes the **post** method must be used. In general with ASP you should use the **post** method, unless you have a specific need to use **get**.

Exercise- Creating the Start Page

This exercise will walk you through the process of creating a form in FrontPage and editing the attributes of the form elements. These same techniques will be used in all future exercises.

1. Create a new page in FrontPage Explorer.
2. Name the page `startpage.asp`.
3. Enter the text `Personnel Data Collection` at the top of the page and make it a heading 2.
4. Press **Return** to create another line.

5. From the **View** menu select **Forms Toolbar**. This will display the **Forms Toolbar** as one of your toolbars. This toolbar will be used extensively during the following exercises.
6. In the Forms Toolbar, click the **One-Line Text Box** button. This will start a form by adding a text box, submit button, and reset button to the page.
7. Place the cursor in front of the **Submit** button and press **Return**. This will add a blank line after the text box.
8. Place the cursor in front of the text box and type `First Name:`.
9. Insert a **line break** after the text box.
10. Type `Last Name`:
11. Insert another **One-Line Text Box**.
12. Insert a line break after the text box you just created
13. Type `Unit:` and insert another text box followed by a line break. You should now have three text boxes on the page, each respectively labeled **First Name**, **Last Name**, and **Unit**.
14. Right-click the text box next to **First Name**. A pop-up menu will appear.
15. Select **Form Field Properties**. The **Text Box Properties** window will open as shown below.



16. In the **Name** field type `first`.
17. Click **OK**. The window will close.
18. Repeat steps 13-16 for Last Name and Unit text boxes. Use the following values for the Names:
    Last Name = `last`
    Unit = `unit`
19. In the line below the Unit text box, insert a **Radio Button** using the **Forms Toolbar**.
20. Next to the radio button type `Enlisted` and insert a line break.
21. Right-click the radio button.
22. From the pop-up menu select **Form Field Properties**. The **Form Field Properties** window will open.
23. In the **Group Name** field enter `nclass`.

24. In the **Value** field enter `enlisted`.
25. Set the **Initial State** to **Not selected**.
26. Click **OK** to close the window.
27. Repeat steps 18-24 to create radio buttons with the following **values**:
    `Officer`
    `Civilian`
    `Other`
    NOTE: All radio buttons must have the **Group Name** `nclass`
28. Save the page.

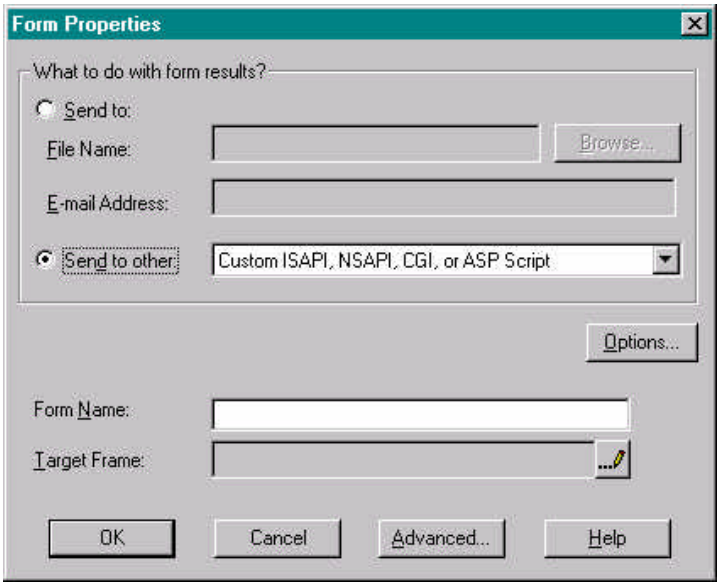
When you are done your HTML source should look similar to this:

```
<html>
<head>
<title>Personnel Data Collection</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
</head>
<body>
<h2>Personnel Data Collection</h2>
<p> </p>
<form method="POST" action="--WEBBOT-SELF--">
  <!--webbot bot="SaveResults" U-File="_private/form_results.txt" S-
Format="TEXT/CSV"
  S-Label-Fields="TRUE" --><p>First Name: <input type="text"
name="first" size="20"><br>
  Last Name: <input type="text" name="last" size="20"><br>
  Unit: <input type="text" name="unit" size="20"><br>
  <input type="radio" value="enlisted" name="nclass">Enlisted<br>
  <input type="radio" name="nclass" value="Officer">Officer<br>
  <input type="radio" name="nclass" value="civilian">Civilian<br>
  <input type="radio" name="nclass" value="other">Other</p>
  <p><input type="submit" value="Submit" name="B1"><input
type="reset" value="Reset"
  name="B2"></p>
</form>
</body>
</html>
```

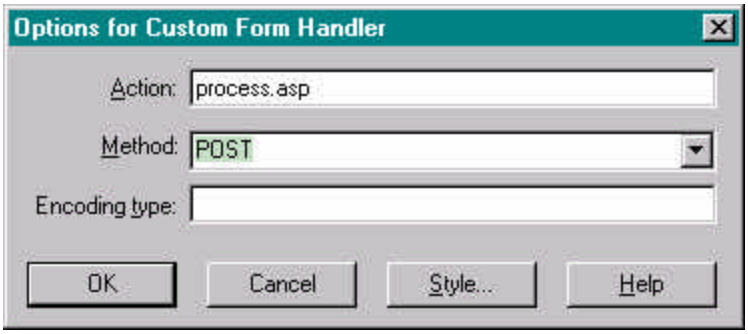
Exercise – Setting the Action

In this exercise we will set the action of the form.

1. Right click anywhere in the form.
2. From the pop-up menu select **Form Properties**. The **Form Properties** window will open as shown below.



3. Select **Send to other**.
4. Click the **Options** button. The **Options for Custom Handler** window will open as shown below.



5. In the **Action** field enter `process.asp`.
6. Click **OK** to close the window.
7. Click **OK** to close the **Form Properties** window.
8. Save the page.

In HTML view the form tag should be the same as below.

```
<form method="POST" action="process.asp">
```

You have now completed the start page for our data collection. The remainder of the page will be ASP pages that process form data. The next step is to create the **process.asp** page.

Creating the Initial Form Handler

Now that we have a form, we need to create the form handler to deal with the data. In the previous exercise we specified the action of the form to be **process.asp** so our handler must be named **process.asp**. This ASP page will look at the data and determine the classification of the respondent. You may recall creating the radio button group named **nclass**. This will be the key field for our data processor. Additionally will convert the other field data into hidden fields so they can be passed to the next page. An easy way to pass data to another page is to include it in the current page.

Including Files

You can tell ASP to include another page within the current page by using the include statement. There are many situations where this is useful. In our case we will create four separate pages, one for each of classification. So when **process.asp** determines the classification we will simply include the correct form in the page.

The include file statement is not actually part of the script and uses the following syntax:

```
<!-- #include file="filename"-->
```

Exercise – Creating process.asp

Follow these steps to create the processor page. The exercise will use the *select case* statement to determine the classification. After the exercise the scripts will be explained.

1. Create a new page in FrontPage Editor.
2. Go to **HTML** view.
3. Just after the <head> tag enter the following script:
```
<%dim first
dim last
dim unit
dim nclass
first = request.form("first")
last = request.form("last")
unit = request.form("unit")
nclass = request.form("nclass")%>
```
4. Change the title to `process.asp`.
5. After the <body> tag enter the following script:
```
<% Select case nclass
case "officer" %>
<!-- #include file="officerform.asp" -->
<% case "other" %>
<!-- #include file="otherform.asp" -->
```

```
<% case "civilian" %>
<!-- #include file="civilianform.asp" -->
 <% case "enlisted" %>
<!-- #include file="enlistedform.asp" -->
 <% end select %>
```

6. Save the page. FrontPage may add line breaks to your scripts, but they will still work.

When you are done your page should look similar to this:

```
<html>
<head><% dim first
   dim last
   dim unit
   dim nclass
   first = request.form("first")
   last = request.form("last")
   unit = request.form("unit")
   nclass = request.form("nclass")%>
<title>process.asp</title>
</head>
<body>
<% Select case nclass
   case "officer"
   %> <!-- #include file="officerform.asp" --> <% case "other"
%> <!-- #include file="otherform.asp" --> <% case "civilian"
%> <!-- #include file="civilianform.asp" --> <% case "enlisted"
%> <!-- #include file="enlistedform.asp" --> <% end select
%> </p>
</body>
</html>
```

About the Code

The one thing you may have noticed about the scripts in this page is that they are spread out with HTML tags between script statements. That is common in ASP pages. Just remember that ASP processes the scripts before sending the page back to the browser.

The scripts start off in a familiar fashion by declaring variables and assigning them values. The values, however, are special in that they use the Request object to determine the actual value. Let's take a look at the actual statement.

```
Request.Form("fieldname")
```

This statement calls on the **Request** object and then **Forms** collection and finally uses the actual form field name. What this statement does is retrieve the value of the specified field from the request that was sent to the server. In our script we took the value and assigned it to a variable. This statement is used often in ASP pages..

The next thing that may have been new was the use of the *select case* conditional statement to determine the classification. *Select Case* is like using multiple *If…Then* statements. The logic is pretty simple. You start by stating criteria for comparison, in our case the variable **nclass**. Then you specify cases that will be executed if the criteria matches. In our script we set four cases, one

for each classification. So if the variable **nclass** equals one of the four *case* statements the corresponding code will be executed.

Let's take a look at the just the first case as an example. First the criteria statement is made:

```
Select case nclass
```

This tells VBScript to use the value of the **nclass** variable as the match criteria. Next we specify the cases. The first case was:

```
Case "officer"
```

This is like saying "*if nclass = 'officer'*". If **nclass** did equal "officer" then whatever statements follow on the next line would be executed. ASP executes any script code and sends back any HTML until it comes across the next *case* statement.

When we are done with all the *case* statements we must end the *select case* as follows:

```
End select
```

Our script told ASP to include a file when the case matched the criteria. Only the statements of the matching case are executed. If none of the cases match the criteria, nothing happens. Additionally, only the HTML of the selected case will be sent to the browser. The HTML statements in the other cases are ignored.

Now that we have told ASP to include a file, we must create the files. If the all files do not exist we will get errors when the scripts execute.

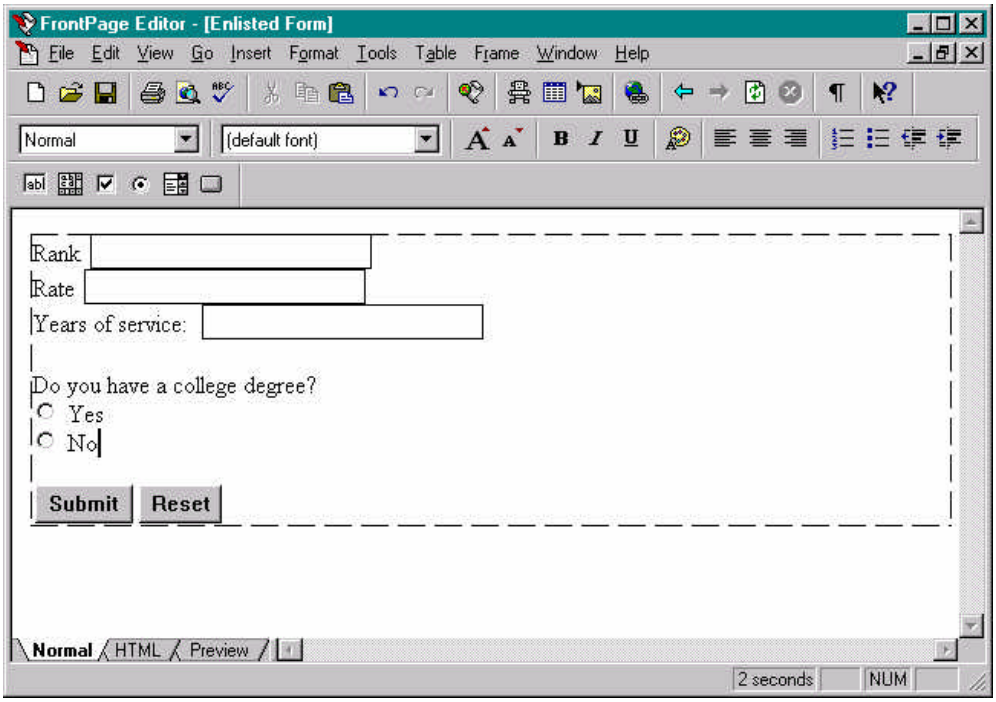
Exercise – Creating the Included Files

For this exercise we will create the **enlistedform.asp** file that will be included in **process.asp**. The page will collect more information on the respondent. It will also use hidden fields to pass form data to the next page. Because this page is another form, we'll start right off creating new form

1. Create a new page in FrontPage editor.
2. Create a form with the following text box fields:
   Rate – name = "rate"
   Rank – name = "rank"
   Years of service – name = "years"
3. Add a blank line below the last text box.
4. Enter the following text:
   `Do you have a college degree?`
5. Add a blank line and create a radio button group named **degree** with the following options:
   Yes – value = "yes" Not selected
   No – value = "no" Not selected

6. Set the form action to `enlistedprocess.asp`.
7. Save the page as `enlistedform.asp`.

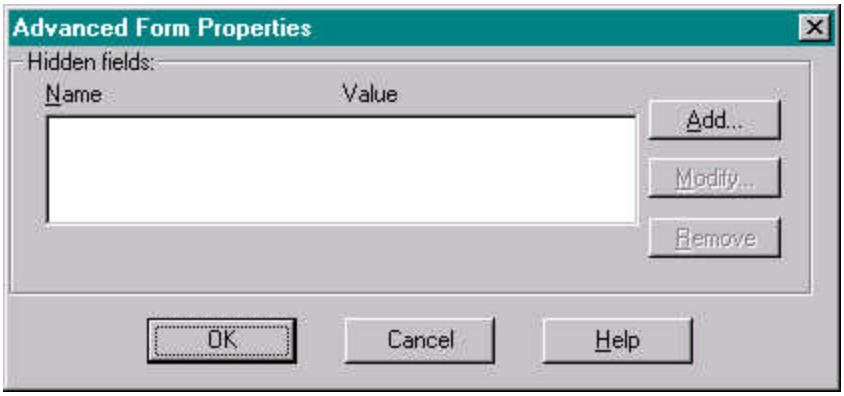
When you are done your page should look similar to the following example in Normal view.



Exercise- Adding Hidden Fields

Follow these steps to add the hidden fields to the form. These fields will be used to pass data to the next ASP page.

1. Right-click the form.
2. From the pop-up menu select **Form Properties**. The **Form Properties** window will open.
3. Click the **Advanced** button. The **Advanced Form Properties** window will open, as shown below.



4. Click the **Add** button. The **Name/Value** pair window will open.
5. In the **Name** field enter `first`.
6. In the **Value** field enter `<%=first %>`.
7. Click **OK**.
8. Repeat steps 4-7 using the following Name/value pairs:
   Name: `last` **Value:** `<%=last%>`
   Name: `nclass` **Value:** `<%=nclass%>`
   Name: `unit` **Value:** `<%=unit%>`
9. Click **OK** to close the **Advanced Form** Properties.
10. Click **OK** to close the **Form Properties** window.
11. Save the page.

For a complete listing of the page source see Appendix A.

Exercise – Copying enlistedform.asp

In order for **process.asp** to work correctly all the files in from all for cases of the select case statement must exist. Follow these steps to create copies of enlistedform.asp.

1. In FrontPage Explorer right click **enlistedform.asp**.
2. In the pop-up menu select **Copy**.
3. Press **Ctrl+V** three times to paste three copy of the page.
4. Rename one copy **officerform.asp**.
5. Rename one copy **civilianform.asp**.
6. Rename one copy **otherform.asp**.

You should now have a form page for each classification, although they are all identical forms.

Practice – officerform.asp

To practice the techniques used to create the enlistedform.asp you will create the officerform.asp page. Fortunately much of the work has already been done since you already copied enlistedform.asp.

1. Open **officerform.asp**.
2. Delete the label and text box for **Rate**.
3. Change the action for the form to **officerprocess.asp**.
4. Change the text above the radio button group to the following:
   `Please select one of the following:`
5. Change the radio button group to include the following options:
   `I attended the Coast Guard Academy.` value = "**academy**"

`I attended Officer Candidate School.` value = "**ocs**"
NOTE: Leave the group name as degree

6. Change the title to **officerform.asp**.
7. Save the page.

When you are done you should have the officerform.asp page completed. For a complete listing of the page see Appendix A.

Testing Pages

So far we have not tested any of our pages. At this point we can't fully test the pages because they will return errors. One thing about ASP applications is that you often have to create numerous pages before testing anything. Often the pages require other pages to work or depend on data passed from one page to the next. At this time all you can test is the start page form. Before we can test our pages we need to create a few more pages.

The next page we will create will take the contents of the enlistedform.asp and display another form based on whether or not the respondent has a degree. We will use an *if…then* statement to determine which form questions to display.

Exercise – Creating enlistedprocess.asp

Follow these steps to create the enlistedprocess.asp page:

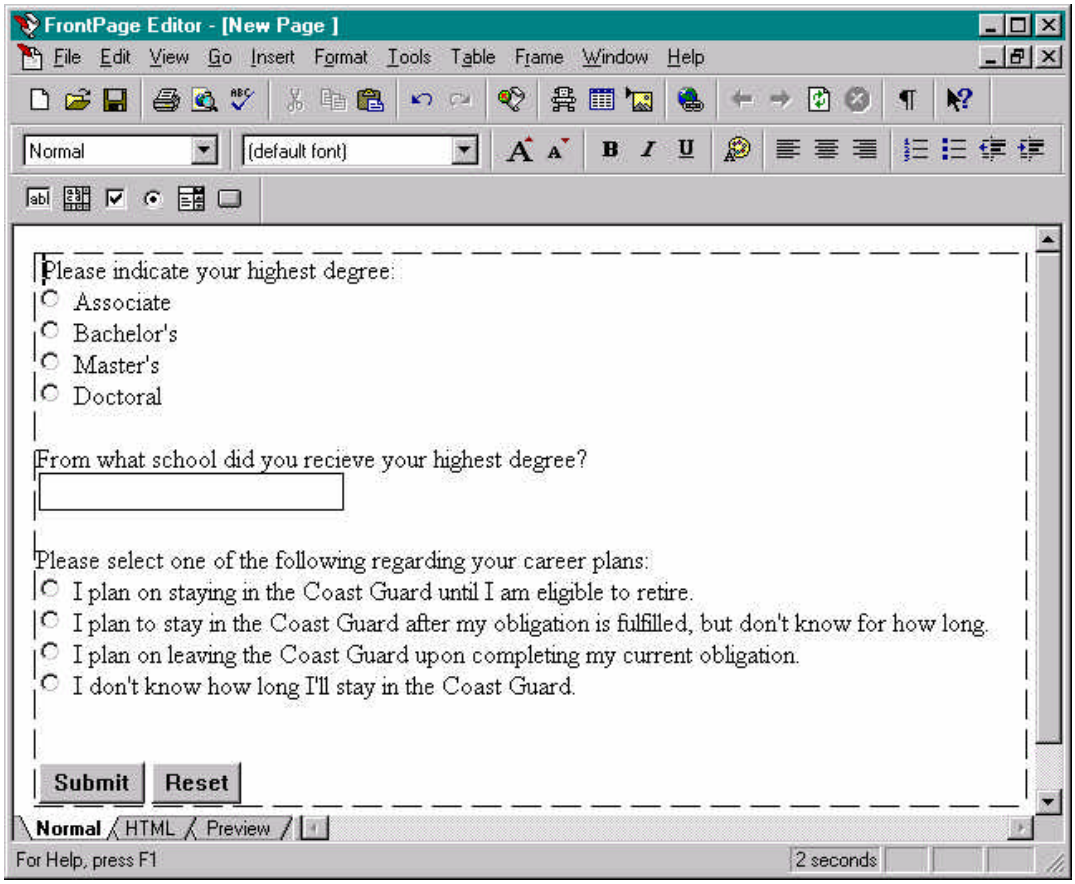
1. In FrontPage Editor create a new page.
2. Switch to HTML view
3. In the <head> of the page declare the following variables in a script and assign their value as the their corresponding form field (an example is provided below):
   degree
   rate
   rank
   first
   last
   nclass
   unit
   years
   Example:
   ```
   dim degree
   degree = request.form("degree")
   ```
4. Switch to Normal view.
5. Add a radio button group named `dgrtype` to the page with the following options:
   `Associate's` – value = "**associate**"
   `Bachelor's` – value = "**bachelor**"
   `Master's` – value = "**master**"
   `Doctoral` – value = "**doctoral**"
6. Place the cursor before the first radio button and insert a line break.

7. In the blank line above the radio buttons type:
   `Please indicate your highest degree:`
8. Add a paragraph break below the last radio button.
9. Add the following text:
   `From what school did you earn your highest degree?`
10. Insert a blank line.
11. Insert a one-line text box with the **name** `college`.
12. Insert a paragraph break after the text box.
13. Add the following text:
    `Please select one of the following regarding your career plans:`
14. Create a radio button group named **careeR** with the following options:
    `I plan on staying in the Coast Guard until I am eligible to retire.` – value = "**retire**"
    `I plan to stay in the Coast Guard after my obligation is fulfilled, but don't know for how long.` – value = "**stay**"
    `I plan on leaving the Coast Guard upon completing my current obligation.` – value = "**leave**"
    `I don't know how long I'll stay in the Coast Guard.` – value = "**unsure**"
15. Set the form action to **enlistedprocess2.asp**.
16. Add the following hidden fields

Name	**Value**
`first`	`<%= first %>`
`last`	`<%= last %>`
`unit`	`<%= unit %>`
`nclass`	`<%= unit %>`
`rank`	`<%= rank %>`
`rate`	`<%= rate %>`
`years`	`<%= years %>`
`degree`	`<%= degree %>`

17. Switch to HTML view.
18. Place the cursor just before the <p> tag before the text **Please indicate your highest degree:**
19. Enter the following script:
```
<% if degree = "yes" then %>
```
20. Place the cursor just after the tag for the text box named **college**.
21. Add the following script:
```
<% else
response.write ("<input type=""hidden"" name=""college"" & _
value=""No Degree"">" & vbNewLine)
response.write ("<input type=""hidden"" name=""dgrtype"" & _
value=""No Degree"">" & vbNewLine)
end if %>
```
22. Save the page as **enlistedprocess.asp**.

When you are done your page should look like the example on the next page.



For a complete list of the code see Appendix A.

Practice

For practice, create officerform.asp and ask the following questions:

- If the respondent attended OCS, what school did they earn their Bachelor's degree from? The field name should be **college**.
- For all respondents, have they attended graduate school? Radio button group name should be **gradschool** and the values should be **Yes** and **No**.
- Set the action to officerprocess2.asp.

Hint: This page is almost identical to the enlistedprocess.asp, so start by copying enlistedprocess.asp. Don't forget to delete any unnecessary hidden fields or variables.

For a complete listing of the code for officerprocess.asp, see Appendix A.

Displaying the Data

The final page in the sequence will take all the data that has been collected and display it in a web page. If we were using a database, this page would write the data to the database. This page

should be relatively simple compared to the previous pages since it does not require any forms or conditional statements. To make things a little easier we will copy **enlistedprocess.asp** to save time writing the scripts.

Exercise – Displaying the data

Follow these steps to create a page to display the data.

1. In FrontPage Explorer right-click **enlistedprocess.asp**.
2. In the pop-up menu select **Copy**.
3. Press **Ctrl+V** to paste the copied page. The copied page will be pasted as **enlistedprocess_copy(1).asp**.
4. Right-click **enlistedprocess_copy(1).asp**.
5. In the pop-up menu choose **Rename**.
6. Rename the page **enlistedprocess2.asp**.
7. Double click **enlistedprocess2.asp** to open it in FrontPage Editor.
8. Switch to HTM view.
9. Delete everything in the body of the page.
10. Add the following variables to the script in the head of the page and assign their values as their corresponding field name
    ```
    dgrtye
    college
    career
    ```
11. Switch to Normal view.
12. Enter the following text:
    ```
    Form Results:
    ```
13. Insert a paragraph break.
14. Insert a table with two columns and eight rows
15. In the first column of the table enter the following text. Each line should be in its own row:
    Name:
    Unit:
    Rank:
    Rate:
    Years of Service:
    Highest Degree:
    College:
    Career Plans:
16. Switch to HTML view.
17. Locate the table cell with the text **Name:**. In the following cell enter the following script:
    ```
    <%=first %>  <%=last %>
    ```
    This will output the first and last name in the cell next to Name:.

18. Using the output directive, enter the scripts to output all the script variables in the cell next to their respective label. The **degree** variable will not be outputted.
19. Save the page.

For a complete listing of the code see Appendix A.

Practice

Create a page to output the Officer data. The file name should be officerprocess.asp. The page will be almost identical to enlistedprocess2.asp.

For a listing of code for this page see Appendix A.

Finishing Up

Your ASP pages should now work. You should be able to go through the Enlisted branch and have your result displayed in a web page.

You have now successfully created a set of ASP pages that can be used to collect data. Additionally, you have successfully used the Response and Request objects to process the data from forms.

Appendix A

enlistedform.asp

```
<head>
<title>Enlisted Form</title>
</head>

<body>

<form method="POST" action="enlistprocess.asp">
  <input type="hidden" name="first" value="<%=first%>"><input
type="hidden" name="last"
  value="<%=last%>"><input type="hidden" name="unit"
value="<%=unit%>"><input type="hidden"
  name="nclass" value="<%=nclass%>"><p>Rank <input type="text"
name="rank" size="20"><br>
  Rate <input type="text" name="rate" size="20"><br>
  Years of service:  <input type="text" name="years" size="20"></p>
  <p>Do you have a college degree?<br>
  <input type="radio" value="Yes" name="degree">Yes<br>
  <input type="radio" name="degree" value="no">No</p>
  <p>Please select one of the following regarding your career plans:</p>
  <p><input type="radio" name="career" value="retire">I plan on staying
in the Coast Guard
  until I am eligible to retire.<br>
  <input type="radio" name="career" value="stay">I plan to stay in the
Coast Guard after my
  obligation is fulfilled, but don't know for how long.<br>
  <input type="radio" name="career" value="leave">I plan on leaving the
Coast Guard upon
  completing my current obligation.<br>
  <input type="radio" name="career" value="unsure">I don't know how long
I'll stay in the
  Coast Guard.</p>
  <p><input type="submit" value="Submit" name="B1"><input type="reset"
value="Reset"
  name="B2"></p>
</form>
</body>
</html>
```

officerform.asp

```
<html>

<head>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
<title>officerform.asp</title>
</head>

<body>

<form method="POST" action="officerprocess.asp">
  <input type="hidden" name="nclass" value="<%=nclass %>"><input type="hidden"
name="first"
  value="<%= first %>"><input type="hidden" name="last" value="<%=last
%>"><input
  type="hidden" name="unit" value="<%= unit %>"><p>Rank: <input type="text"
name="rank"
  size="20"><br>
  Years of Service:  <input type="text" name="years" size="20"></p>
  <p>Please select one of the following:<br>
  <input type="radio" value="academy" name="degree">I attend the Coast Guard
Academy<br>
  <input type="radio" name="degree" value="ocs">I went to Officer Candidate
School</p>
  <p><input type="submit" value="Submit" name="B1"><input type="reset"
value="Reset"
  name="B2"></p>
</form>
</body>
</html>
```

enlistedprocessor.asp

```
<html>
<head>
<title>Enlisted Processor</title>
<% dim degree
       dim rate
       dim rank
       dim years
       dim first
       dim last
       dim nclass
       dim unit
       degree= request.form("degree")
       rate = request.form("rate")
       rank = request.form("rank")
       years = request.form("years")
       first = request.form("first")
       last = request.form("last")
       nclass = request.form("nclass")
       unit = request.form("unit")
%>
</head>
<body>
<form method="POST" action="enlistedprocess2.asp"><input type="hidden"
name="first" value="<%=first%>"><input type="hidden" name="last"
value="<%=last%>"> <input type="hidden" name="unit" value="<%=unit%>"><input
type="hidden"  name="nclass" value="<%=nclass%>"><input type="hidden"
name="rank" value="<%=rank%>"><input  type="hidden" name="rate"
value="<%=rate%>"><input type="hidden" name="years"  value="<%=years%>"><% if
degree = "Yes" then
%>
<p>Please indicate your highest degree: <br>
  <input type="radio" value="Associate" name="dgrtype">Associates' Degree<br>
  <input type="radio" name="dgrtype" value="Bachelor">Bachelors' Degree<br>
  <input type="radio" name="dgrtype" value="Master">Master's Degree<br>
  <input type="radio" name="dgrtype" value="Doctoral">Doctoral Degree</p>
  <p>From what institution did you receive your highest degree?<br>
  <input type="text" name="college" size="40"></p>
<% else %>
<input type="hidden" name="degree" value="<%=degree%>"><input type="hidden"
name="dgrtyp" value="No degree"><input type="hidden" name="college"
  value="No degree">
<%end if %>
  <p>Please select one of the following regarding your career plans:</p>
  <p><input type="radio" name="career" value="retire">I plan on staying in the
Coast Guard until I am eligible to retire.<br>
  <input type="radio" name="career" value="stay">I plan to stay in the Coast
Guard after my obligation is fulfilled, but don't know for how long.<br>
  <input type="radio" name="career" value="leave">I plan on leaving the Coast
Guard upon completing my current obligation.<br>
  <input type="radio" name="career" value="unsure">I don't know how long I'll
stay in the Coast Guard.</p>
  <p><input type="submit" value="Submit" name="B1"><input type="reset"
value="Reset"  name="B2"></p>
</form>
</body>
</html>
```

officerprocess.asp

```
<html>

<head><% dim degree
dim rank
dim first
dim last
dim nclass
dim unit
first = request.form("first")
last = request.form("last")
unit = request.form("unit")
nclass = request.form("nclass")
rank = request.form("rank")
years = request.form("years")
degree = request.form("degree")
%>
<title>officerprocess.asp</title>
</head>
<body>

<form method="POST" action="officerprocess2.asp">
  <input type="hidden" name="nclass" value="<%= nclass %>"><input type="hidden"
name="degree"
  value="<%= degree %>"><input type="hidden" name="first" value="<%= first
%>"><input
  type="hidden" name="last" value="<%= last %>"><input type="hidden" name="rank"
  value="<%= rank %>"><input type="hidden" name="unit" value="<%= unit%>"><input
  type="hidden" name="years" value="<%= years %>"><% if degree = "ocs" then %>
<p> </p>
  <p>From what school did you recieve your degree?<br>
  <input type="text" name="college" size="20"></p>
<% end if %>
  <p>Have you attended graduate school?<br>
  <input type="radio" value="yes" name="gradschool">Yes<br>
  <input type="radio" name="gradschool" value="no">No</p>
  <p><br>
  <input type="submit" value="Submit" name="B1"><input type="reset"
value="Reset" name="B2"></p>
</form>
</body>
</html>
```

enlistedprocess2.asp

```
<html>
<head>
<title>New Page </title>
<% dim degree
dim rate
dim rank
dim first
dim last
dim nclass
dim unit
dim dgrtype
dim college
dim career
first = request.form("first")
last = request.form("last")
unit = request.form("unit")
nclass = request.form("nclass")
rate = request.form("rate")
rank = request.form("rank")
years = request.form("years")
dgrtype = request.form("dgrtype")
college = request.form("college")
career = request.form("career")
%>
</head>
<body>
<p>Form Results:</p>
<table border="1" width="100%">
  <tr>
    <td width="50%">Name:</td>
    <td width="50%"><%= first %>
<p> <%=last%></td>
  </tr>
  <tr>
    <td width="50%">Unit:</td>
    <td width="50%"><%= unit %>
</td>
  </tr>
  <tr>
    <td width="50%">Rank:</td>
    <td width="50%"><%= rank %>
</td>
  </tr>
  <tr>
    <td width="50%">Rate:</td>
    <td width="50%"><%= rate %>
</td>
  </tr>
  <tr>
    <td width="50%">Years of Service:</td>
    <td width="50%"><%= years %>
</td>
  </tr>
  <tr>
    <td width="50%">Highest Degree:</td>
    <td width="50%"><%= dgrtype %>
```

```
</td>
  </tr>
  <tr>
    <td width="50%">College:</td>
    <td width="50%"><%= college %>
</td>
  </tr>
  <tr>
    <td width="50%">Career Plans:</td>
    <td width="50%"><%= career %>
</td>
  </tr>
</table>
</body>
</html>
```

officerprocess2.asp

```
<html>

<head><% dim degree
dim gradschool
dim rank
dim first
dim last
dim nclass
dim unit
dim college
first = request.form("first")
last = request.form("last")
unit = request.form("unit")
nclass = request.form("nclass")
degree = request.form("degree")
rank = request.form("rank")
years = request.form("years")
gradschool = request.form("gradschool")
college = request.form("college")
%>

<title>officerprocess2.asp</title>
</head>

<body>

<p>Form Results:</p>

<table border="1" width="100%">
  <tr>
    <td width="50%">Name:</td>
    <td width="50%"><%= first %>
<p> <%=last%></td>
  </tr>
  <tr>
    <td width="50%">Unit:</td>
    <td width="50%"><%= unit %>
</td>
  </tr>
  <tr>
    <td width="50%">Rank:</td>
    <td width="50%"><%= rank %>
</td>
  </tr>
  <tr>
    <td width="50%">Years of Service:</td>
    <td width="50%"><%= years %>
</td>
  </tr>
  <tr>
    <td width="50%">Attended:</td>
    <td width="50%"><%= degree %>
</td>
  </tr>
  <tr>
    <td width="50%">College:</td>
```

```
    <td width="50%"><%= college %>
</td>
  </tr>
  <tr>
    <td width="50%">Graduate school</td>
    <td width="50%"><%= gradschool %>
</td>
  </tr>
</table>
</body>
</html>
```

Appendix B

ASP Object Reference

Application Object

Collections	Methods	Events
Contents	Lock	Application_OnStart
StaticObjects	Unlock	Application_OnEnd

ObjectContext Object

Methods	Events
SetAbort	OnTransactionAbort
SetComplete	OnTransactionCommit

Request Object

Collections	Properties	Methods
ClientCertificate	TotalBytes	BinaryRead
Cookies		
Form		
QueryString		
ServerVariables		

Response Object

Collections	Properties	Methods
Cookies	Buffer	AddHeader
	CacheControl	AppedToLog
	Charset	BinaryWrite
	ContentType	Clear
	Expires	End
	ExpiresAsolute	Flush
	IsClientConnected	Redirect
	PICS	Write
	Status	

Server Object

Properties	Methods
ScriptTimeout	CreateObject
	HTMLEncode
	MapPath
	URLEncode

Session Object

Collections	Properties	Methods	Events
Contents	CodePage	Abandon	Session_OnStart
StaticObjects	LCID		Session_OnEnd
	SessionID		
	Timeout		