Advanced HTML
February 4 & 5, 2002

LAB EXERCISE:
Using SSI for global change & CGI for form data

A. Establishing a Path

A simple way to organize a lot of content like this is to establish a path of folders in your public_html, and to do that we're going to have to open FTP to Dante. When in the computer labs the direction is:

> **Start > Programs > FTP Clients > SSH Secure FTP > Secure FTP Dante**
> Enter your **"public_html"** or **"students_html"** folder by double clicking on it.
> When you are inside your public_html folder **Right Click and select New Folder**
> Name this folder **portfolio**

You have now established a path in your student page. The URL will read:
http://students.washington.edu/UWNetID/portfolio/

Now that we have our space established we're going to begin writing our SHTML file. **Leave Dante open** but reduce the window. In order to truly see the power of SSI we have to upload our files to the Dante server so that they can be compiled.

B. Setting up an SHTML File

Just as we did in the previous workshops we're going to be working in Notepad.

1. To open Notepad from the desktop, select:
   Start > Programs > Accessories > Notepad

2. Prior to typing anything in the blank document, begin first by saving the file:
   - Select: **File > Save As**
   - Save in: **A Drive (if you've brought a disk) or C Drive and create a personal folder with your name**
   - File name: **index.shtml**
   - Save as type: **all files**

*The **shtml** extension instructs the server to compile SSI files referenced in this html document. All pages on your site that use SSI need to have an shtml extension.*
This index file will not replace your previous files, because the file will be in a new location.
This file will be the index for your new portfolio folder that we just established.

C. Starting Ground

Once a file has been saved, it is now possible to begin writing the document. Previously, we used CSS to change the style of our web pages. This exercise will build from that workshop. To begin let's all copy the code from the CSS compliant page on my website so that we're all on the same page.

Open the **Internet Explorer** and go to:
http://students.washington.edu/eriley/advanced/before.html
View > Source to see the notepad with the code.
Highlight the code and copy it to your **index.shtml** file
Save your file and drop the window down. We will work on this file in just a minute

You'll need to copy the CSS files into new blank text files

Open a new Notepad file
Save your file as **style1.txt**
Go to **http://students.washington.edu/eriley/advanced/style1.txt**
Copy the text from the screen and paste into your blank text file
Save your file as stylesheet

Repeat for **http://students.washington.edu/eriley/advanced/style2.txt**

Now you have your basic index page and two different style sheets. This will be the starting ground for our new portfolio website. Go ahead and **Close the stylesheets.**

D. Creating an SSI File

SSI stands for Server Side Includes; this basically means that the SHTML document that is being presented contains additional information that exists on the server. This can be anything from information about time and date that the page was accessed to entire chunks of code. We will be using SSI to create a template for our navigation bar.

1. Startup

The first thing we need to do is to create an SSI text file:

Open Notepad
Select: **File > Save As**
File name: **navbar.txt**
Save as type: **all files**

SSI files can be saved as either text or html. It's better to save your SSI files as text, because sometimes there can be conflicts if it's saved as html.

2. Selecting Your Include

The next thing that you need to decide is how much information you want to include in the SSI file. For this class we're going to include all the head information, including our CSS reference, and all the information from our website that will remain static across all pages, i.e. the navigational information.

Open the **index.shtml** file
Select all text from the **<head>** tag to **<td class="shorty">**
Cut this portion of text from the index page and
Paste it into the **navbar.txt** file
Save changes to both files

3. The <!--#include --> tag

The include tag is the cue for the server to pull and load your SSI file. It may look a little confusing, because it looks like a comment tag. The difference between the include and the comment tag is that the include tag has a pound sign preceding the code.

Open your **index.shtml** page
Insert the tag **<!--#include file="navbar.txt" -->**
Save your file

Note: there should always be one space between the final quotation mark and the end dashes. Some servers won't load the file properly without the space.

4. Viewing your work

One of the only drawbacks about SSI is that it requires being loaded onto an actual server to be viewed. The page won't compile from your hard disk or your floppy. So, we have to upload all our pages into our new folder.

Open your Secure FTP to Dante
Open the **portfolio** folder
Upload: style1.txt, style2.txt, navbar.txt, and **index.shtml**
View your page at **students.washington.edu/UWnetID/portfolio/**

WARNING: **Always verify you are in the correct folder before uploading**; otherwise you may erase other material in your HTML account. This is especially hazardous for files named "index.shtml." Dante provides you with a visual display on the left hand side of the screen showing where you are in the folder structure. Make sure your **portfolio** folder is open.

You may be asking yourself at this point, what the big deal is. The website looks the same as it did before, right? Well let's have a little experiment.

5. Adding multiple pages

Using your index.shtml file on your disk, make copies to fill all of the needed page slots.

Open your **index.shtml** file in notepad
File > Save As: teaching.shtml
Repeat for scholarship, leadership, technology, and service
Upload all five new pages.

Your links should now be active to all pages.

6. Using UNIX commands and PICO

Up till now we have used notepad to do all of our text editing. But you can also edit your webpages directly in the Dante shell using a UNIX command line editor called PICO. We're going to use it here to illustrate the global change possibilities with SSI.

Open your Internet Explorer and look at your new website.

This is how you're currently configured. Let's do a quick change in PICO.

Open your **SSH to Dante**
Select "S" to enter UNIX commands to the Shell

• *The Working Directory*
UNIX refers to the folders in your Dante account as directories. The working directory shows you which folder you are working from. We need to see where we are so that we can access our navbar.txt file.

Type **pwd** to see which directory you are currently logged into

• *Changing Directories*
In order to move around in the shell we have to use the **cd** command to open a new folder. **cd** will only work if you follow it up with the folder you need to open

Type **cd public_html** to Change Directories to your public_html folder

• *The List Command*
If you are unsure what files and directories are in your folder you can see a copy of them using the list command.

Type **ls** to list which files are in this folder if you are unsure about where you're going.
Type **cd portfolio**
Type **ls**

• *The PICO text editor*
Similar to the cd command you have to combine the pico command with the file you want to open**.** Pico allows you to edit your html or shtml files directly on in the shell. You should be really careful when editing here though, because when you change it in pico, it's changed for good.

Type **pico navbar.txt**

This will load the navbar file in the PICO editor on your SSH window. It should look just like your normal text file.

Change your style sheet from **style1.txt** to **style2.txt**
Ctrl+X (written ^X) will exit from the file, and you will be prompted to
Save Modified buffer
Return to save over the file.
Type **exit** to get out of the shell

If you refresh your portfolio now, you can see that every page that has the navbar SSI has completely changed in look. By separating out redundant bits of information with CSS and SSI you can change the look and feel of your website much faster, because you only have to change one page instead of 2-1000.

E. SSI Warning

Always edit your SHTML files in one of two ways:
1) Edit them with PICO
2) Download the file from Dante to your hard drive and edit it in Notepad

Don't ever pull your source code from the internet browser using View > Source. The source code has been compiled by the server, and you will lose your normal navbar.txt include.

F. Creating a Form

Forms can be one of the best ways to transmit standardized data on the internet. Forms are completely customizable, and they can give you a good amount of control over what people can send to your email. To accomplish this we're going to create two small shtml pages and we're going to use a premade CGI file to transmit our data.

1. Starting a file

1. To open Notepad from the desktop, select:
   Start > Programs > Accessories > Notepad

2. Prior to typing anything save the file:
   - Select: **File > Save As**
   - Save in: **A Drive (if you've brought a disk) or C Drive**
   - File name: **form.shtml**
   - Save as type: **all files**

2. Defining HTML

The first action needed is to convey to the computer that the document being created is to be in HTML. To do that, the following tags should be inserted:

```
<html>
</html>
```

These tags will go at the top and bottom of the HTML document. All of the content will fall in between these tags.

3. Adding our Navigation Bar

Because we've established an SSI file for a standardized look and feel on our page we're going to insert our include.

```
<html>
<!--#include file="navbar.txt" -->
```

4. Heading Information

This tag formats our interior heading to keep the look and feel the same on every page. The <hr> tag instructs the browser to insert a horizontal line.

```
<h2 class="sub">Email Comments Form</h2>
<hr>
```

While we're at it, let's just add in a little piece of content as well.

**<p class="content">How's my Site? Why not drop me a line to let me know.
**

*Note: The
 tag is a newer HTML tag format. The /> indicates that there is no end tag following this break.*

Save your work.

5. The <form> tag

The form tag and all of its associated tags is fairly complex and contains a substantial number of attributes. We will go through each of the attributes one by one, like we did in the basic class for the <body> tag.

<form>

This tells the browser that we're starting to create a form

- *Method Attribute*

  <form method=post>

  This is the HTTP method of passing data to the web server. There are two main methods Get and Post. Post must be used if there are more than 100 characters in the form data

- *Action Attribute*

  <form method=post action="form2mail.cgi">

  This tells the browser to access and execute this program to transmit the data, and the CGI file (which we will download later) formats and transmits the data to your email.

  Note: I don't know why some of these attributes require quotes and others don't. We should just accept it on a leap of faith.

Save your work.

6. The <input> tag

Input is just what it says it is. It serves as an indicator that there is data that will be gathered immediately following this tag. Like the form tag it also is fairly complex, so we will approach this in stages as well.

- *The type attribute*

  <input type="hidden">

  There are many different types of input. We'll encounter them as we move along. This hidden input tag means that this information is not really meant to be encountered by the user. Instead, we will input commands that the CGI file will read to format our email response.

- *The name attribute*

  <input type="hidden" name="_order">

  The name attribute defines a specific bit of data by giving it a unique name. The name we've give here of _order is required in our CGI program to label the data from the form with the following values.

• *The value attribute*

<input type="hidden" name="_order"
value="name,email,iam,response,topic,comments">
**
**
We are going to create a form with six different input areas, where the name associated with the data is recognized as name, email, iam, response, topic, and comments. All of the values have to be run together, because it is a requirement of this CGI program. Also, in order to make sense of the results, all subsequent input tags must use these names in this order.

Note: The beauty of this particular CGI file is that you can have as many values as you need, and you can name them whatever you like.

7. Input Type="TEXT"

Perhaps the most common type of form interface is the single line text box. Type attribute **Text** produces that little box.

Your name:
**<input type="text" name="name" size=20>
**
**
**
Because this is where we want people to enter their name we've labeled the attribute to respond as "Name." Size indicates the length of the box in column width. This is not a limitation on how many characters can be entered into the box. To limit the length of response you would use the attribute **maxlength="##"** where the numbers are how many characters are allowed in the field.

Let's add another.

Your email:
**<input type="text" name="email" size=20>
**
**
**

Save your work.

8. The Dropbox

Another great form option is the menu list or dropbox. This type of interface doesn't use the <input> tag, but operates with similar attributes.

• *The <select> tag*

I am:
<select name="iam">
This instructs the computer that there we will present a list of options to select from.

• *The <option> tag*

The option tag is used to separate individual items in the dropbox (similar to the <li> tag in ordered and unordered lists). You can only choose one item in the dropbox. Values have to be added to options, because the CGI will return blank information without it.

<option value="A Prospective Employer">A Prospective Employer
<option value="A Professor">A Professor
<option value="A Friend">A Friend
<option value="A Total Dork">A Total Dork

```
<option value="Other">Other
</select><br />
<br />
```

Save your work.

9. Radio Buttons

Radio buttons are those loveable little bubbles, similar to those on standardized tests.

```
Would you like a response?
   Yes <input type="radio" name="response" value="Yes">
   No <input type="radio" name="response" value="No">
<br />
<br />
```

Note: by naming this input area "response" on both buttons the user must choose between the two. If they had separate names they would function similar to checkboxes.

One more note: We encountered the "&----;" tag format in the Intermediate lab. The tag above is the non-breaking space character entity.

Now let's add a topic line to this message

```
Topic:
<input type="text" name="topic" value=" " size=40><br />
<br />
```

Note Madness: The value attribute is optional. You can insert some generic information to alert your users as to how they should enter their info, or you can leave it blank as it is here, or you can leave it out. Anything written by the user in the text line overrides preset values

Save your work.

10. Text Area

This section will provide your users with a little empty block where they can write a little comment to you.

```
Comments:<br />
<textarea name="comments" rows=7 cols=40 wrap=on></textarea>
<br />
<br />
```

Similar to input type text you can specify the size, but since this is a square, not a line, you must give the dimensions in rows and columns. The wrap attribute turns on and off word wrapping.

11. Input Type=Submit

This input type is actually a command to send the form according to the instructions that we outlined in the <form> tag at the beginning. The value entered here can be whatever you want the button to read. I personally happen to like "SEND IT!"

```
<input type="submit" value="    Send It!    ">
</form>
```

Note: Unlike other areas in HTML, values can have spaces that are read normally.

Let's finish our page with our standardized ending.

```
<p class="credits">&copy; Eric Riley, 2002</p>
</td>
</tr>
</table>
<br />
<br />
</body>
</html>
```

Save your work.

G. Finishing Up

At this point you should have the following text in your form.shtml file.

```
<html>
<!--#include file="navbar.txt" -->
<h2 class="sub">Email Comments Form</h2>
<hr>
<p class="content">How's my Site?
Why not drop me a line to let me know.<br />
<form method=post action="form2mail.cgi">
<input type="hidden" name="_order" value="name,email,iam,response,topic,comments">
<br />
Your name:
<input type=text name="name" size=20><br />
<br />
Your email:
<input type=text name="email" size=20><br />
<br />
I am:
<select name="iam">
<option value="A Prospective Employer">A Prospective Employer
<option value="A Professor">A Professor
<option value="A Friend">A Friend
<option value="A Total Dork">A Total Dork
<option value="Other">Other
</select><br />
<br />
Would you like a response?
   Yes <input type=radio name="response" value="Yes">
   No <input type="radio" name="response" value="No">
<br />
<br />
Topic:
```

```
<input type="text" name="topic" value=" " size=40><br />
<br />
Comments:<br />
<textarea name="comments" rows=7 cols=40 wrap=on></textarea>
<br />
<br />
<input type="submit" value="     Send It!     ">
</form>
<p class="credits">&copy; Eric Riley, 2002</p>
</td>
</tr>
</table>
<br />
<br />
</body>
</html>
```

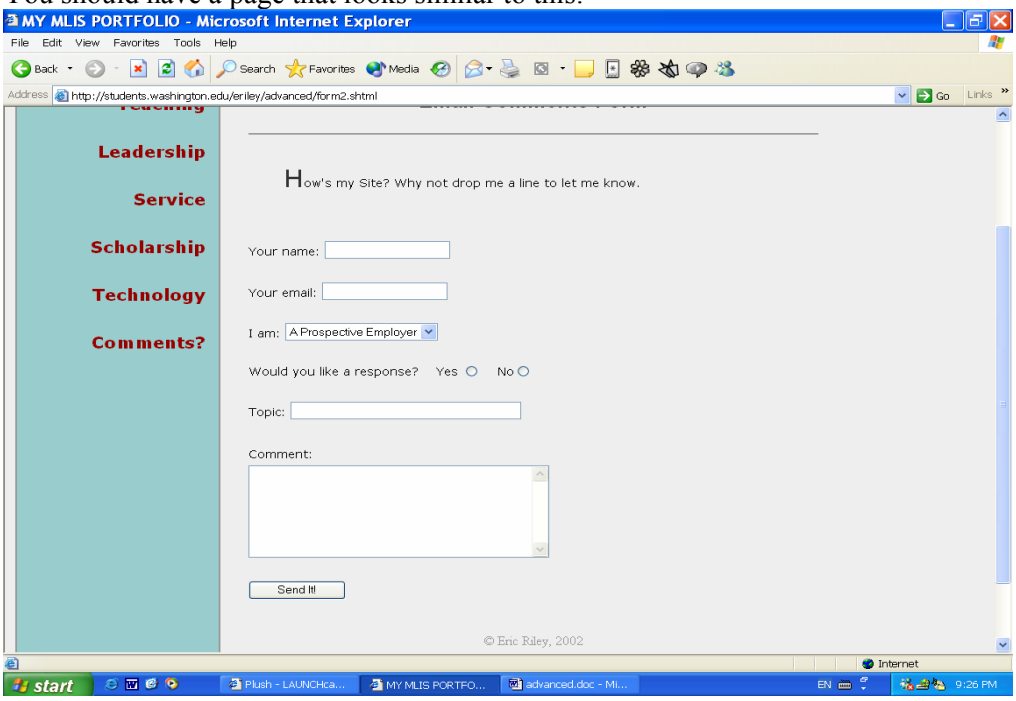
H. Uploading your Form

Now that we've done all this work, let's see what it actually looks like.

Open your Secure FTP to Dante
Open the **portfolio** folder
Upload: form.shtml
View your page at **students.washington.edu/UWnetID/portfolio/form.shtml**

I. Example

You should have a page that looks similar to this:



J. Hot CGI Action

CGI stands for Common Gateway Interface, and it basically just means that you have a program somewhere in the background used to perform these user interface functions. Many CGI programs are written in PERL, but some are in JavaScript and others, God help us, in C or C++. It all depends on how you need information transmitted from your forms, and where they will be going. For our purposes we are going to be using a little PERL email program. We've actually referred to it in composing our form, and those references are mentioned above. Part of the joy of CGI is that many little CGI programs are already made for you. We'll be using a premade PERL program.

K. form2mail.cgi

The first thing we're going to do is copy the text of this cgi file from my website into a new notepad file.

1. To open Notepad from the desktop, select:
   Start > Programs > Accessories > Notepad

2. Prior to typing anything in the blank document, begin first by saving the file:
   - Select: **File > Save As**
   - Save in: **A Drive or C Drive as before**
   - File name: **form2mail.cgi**
   - Save as type: **all files**

Now use the Internet Explorer to retrieve the following page

> http://students.washington.edu/eriley/advanced/form2mail.txt

Just highlight and copy all the text directly from the page.

Note: You must save the file with a .cgi extension for the server to recognize it as a program.

L. PERLS of wisdom

Now we're going where our little web class has never tread before, a real, albeit comprehensible, programming language. here are some things you need to know about PERL

- *Pound Signs*

Whenever your see # that indicates that the following words are just comments and that they aren't meant to be executed with the rest of the program. All the business at the beginning of this program is just notes.

- *Dollar Signs*

$ stands for a scalar or singular variable. Directly beneath the credits you'll see:

$recipient = "eriley\@u.washington.edu";

This is a single variable that gets called upon later in the program

- *@ Signs*

@ is an arrayed variable. Whenever you have something that is composed of many parts a way to save some code is to use an array. A real version of an arrayed variable is in this:

@order = split(/,/,$data->para('_order'));
This command refers to our hidden input area with the name "_order" and the variables are represented by the / marks. The problem that you run into here is when you want to insert an email address into your code (like in our recipient area).

• *Forward Slash*
\ is a command to ignore the next immediate symbol, because it's being used in a non-PERL way.
See: **$recipient = "eriley\@u.washington.edu";**
We have to use the slash here to send this email.

1. Important bits in this program
There are a couple of things in this program that are important to notice and/or change.

a) You are the recipient of this email.
Change the **$recipient** line to your email address

b) The subject line of this email will read however you want it to
Change the **$subject = "Comments on your portfolio"**
This will remind you that it's coming from the form, and identifies it specifically in case you have more than one form

c) The response to submitting the form directs the user to a webpage in this current directory labeled **"submitted.shtml".** We still need to create this file.

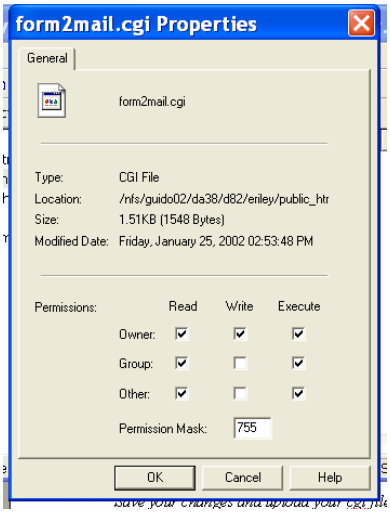
Save your changes and upload your cgi file.

M. UNIX File Permissions

Now that we've uploaded our cgi file we need to check the permissions. Since this is a program file we have to set some different permissions in order to get it to work. Up to now we've given everyone in the world "Read" permissions and "Write" permissions to yourself so that only you can change the file: Permission Mask 644. For CGI files we have, which need to run, we need to set execute permissions for everyone: Permission mask 755.

Open Secure FTP
Navigate to your portfolio folder to find **form2mail.cgi**
Right Click on the cgi file
Select Properties
Activate Execute Permissions

It should look like this:



To see how to change your file permissions in the shell, see the Advanced tags handout, page 2, using **chmod**

N. Submission Response Page

The great thing about a response page is that they are totally simple, because all you have to do is say thanks. According to our PERL program we have to call it submitted.shtml

1. Open a new notepad file

- **Open Notepad**
- Select: **File > Save As**
- Save in: **A Drive or C Drive as before**
- File name: **submitted.shtml**
- Save as type: **all files**

2. The Code
Heck, I'll just give it to you.

```
<html>
<!--#include file="navbar.txt" -->
<p class="content">Thank you for your comments!</p>
<p class="content">If you have opted for a response I will respond as soon as possible.</p>
<br />
<br />
<p class="credits">&copy;Eric Riley, 2002</p>
</td>
</tr>
</table>
<br />
<br />
</body>
</html>
```

Save and upload your work to the portfolio folder.

Take a moment now to view this page to see that it's written properly.

O. The Moment of Truth

Now that we've written, updated and installed all the necessary pages and scripts, we need to test our form to make sure that everything is in properly and that we haven't totally ruined our life. There are four things that need to happen:

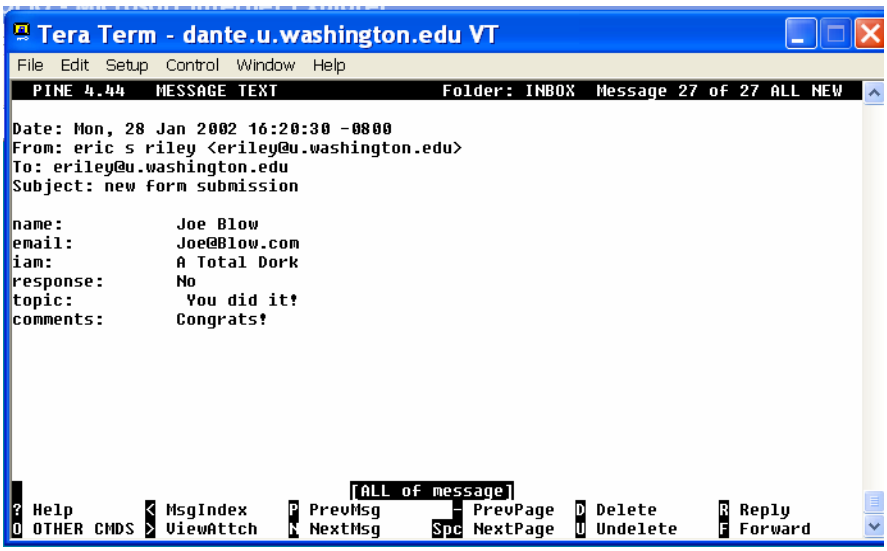
1) All the pieces of the form should work (We've tested this earlier)
2) The CGI file needs to execute
3) Our submitted.shtml page needs to be returned upon completion and submission of the form.
4) We need to receive an email containing all of our submitted information

1. The Test
Navigate to your form **> students.washington.edu/UWnetID/portfolio/form.shtml**

Enter data in every field.

Name: Joe Blow
Email: Joe@Blow.com
I Am: A Total Dork
Response: No
Topic: You did it!
Comment: Congrats!

Just hit the "Send It!" button and you should receive an email like this:



Note: You should notice that the email you received looks like you sent it to yourself. Don't reply to this message! Reply to the email address listed inside the data you received. Different CGI programs have different ways to format the data received.

P. A Final Challenge

Now add the link to your form page into your navbar.txt file using PICO.
Then make a link to your Portfolio folder from your top level index.html page in your public_html.
Then check your links in your IE window.

Congratulations!
You've completed the Advanced HTML workshop