**HTML Forms and CGI-Scripts**

An HTML form is a web page containing active elements, i.e. elements that the user can interact with such as buttons, input fields, check boxes, drop-down lists etc. In most cases the user choices are submitted to a server-side script or program, using the Common Gateway Interface (CGI).

A form must be enclosed in a <form> tag, which takes two options: a "method", which can be either "get" or "post", and an "action", which lists the script that receives the form input.

Example:

<html>
<h1>A Simple Form</h1>

<p>This is some text explaining the form, but outside the actual form.
</p>

<form method="get" action="http://machine-url/path/executable">

<ul>
 <li>Name: <input type="text" name="name"></input></li>
 <li>Credit: <input type="text" name="credit"></input></li>
</ul>

<input type="submit" name="submit" value="Go">
<input type="reset" name="reset" value="Restart">

</form>

</html>

If the user fills in name and credit card number, the parameters "name=My Name" and "credit=12345", say, will be sent to the program "executable" residing in the CGI script directory of the web server. The params are submitted via a URL of the form:

<http://machine-url/path/executable?name=My+Name&credit=12345>

It is the script’ job to parse the input parameters and act accordingly.

Other form elements you could use include:

<input type="radio" name="sex" value="female" />Female<br />
<input type="radio" name="sex" value="male" checked="true" />Male<br />

<input type="checkbox" name="vehicle" value="Bike" checked="true" /> I have a bike<br />
<input type="checkbox" name="vehicle" value="Car" /> I have a car<br />

<select name="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="fiat" selected="true">Fiat</option>
<option value="audi">Audi</option>
</select>

<textarea rows="10" cols="30"> Default text in text area
</textarea>

The script receiving the form can be written in many languages. Popular are Perl, PHP, Java JSP, Microsoft ASP, or even C, but it could be a generic shell script. In general, shell scripts are not used because of security concerns, but for our now let's use shell scripts anyway (since we just learned how to write them).

**Simple Example**

Our form, saved in **/home/username/httpd/htdocs/myform.html** is:

<h1>My Simple Form</h1>

Check if this user is logged in:

<form method="get" action="http://sciris.shu.edu:7770/cgi-bin/checkin">

 <p><b>Name:</b> <input type="text" name="user"></p>

<input type="submit" value="Check User"> <input type="reset" value="Reset User">

</form>

</html>

The script, as specified in the above form's "action" parameter, must be marked executable and be saved in **/home/username/httpd/cgi-bin/checkin**. It might go as follows:

#!/bin/sh

# Bert Wachsmuth, Feb 2011

# This script does some stuff

# extract the user name from QUERY\_STRING and store it in a variable

USER=`echo "$QUERY\_STRING" | sed 's/user=//'`

# Setting up the response file type as a plain html document

echo "Content-type: text/html"

echo

# Now comes the standard HTML stuff

echo "<html>"

echo "<h3>My script is answering your request</h3>"

# Just to see what's going on, we display the variable names

echo "<p>You entered $QUERY\_STRING as name</p>"

echo "<p>I extracted $USER as your username</p>"

# here is the essence of the script

if [ $USER ]

then

 last | grep $USER | sed –n '1 p'

else

 echo "You need to enter a user name"

fi

echo "</html>"

**Project: Addressbook Simulation**

As a more advanced and complete example of CGI scripting using shell scripts, let's simulate a simple address book lookup program, complete with a menu and everything.

Let's assume our address book is a flat text file with one address per line. Each address has a name, a comma, and a phone number. We want to:

* List all addresses in a reasonably nice way
* Lookup an address by entering a fragment of the address, either case sensitive or not. It should return all addresses matching the search
* Add an address. An address must have a name and phone number, otherwise it cannot be added
* Delete an address identified by line number

We do not want to edit addresses at this time.

**Technical note:**

Using sed to extract a single form parameter (simple minded; fails for invalid input)

USER=`echo "$QUERY\_STRING" | sed 's/user=//g'`

Using sed to extract form parameters properly using regular expressions - don't ask ☺

USER=`echo "$QUERY\_STRING" | sed -n 's/^.\*user=\([^&]\*\).\*$/\1/p'`

* For a nice intro to sed, see <http://www.alexonlinux.com/sed-the-missing-manual>
* For a more complicated use of sed see <http://foxlx.acmesystems.it/?id=165>
* Info on regular expressions <http://www.regular-expressions.info/> or a brief, quick intro at
<http://www.regular-expressions.info/quickstart.html>

**Files**

In class we created several files to get this project started. You should find them as separate links. They should go into the following locations:

* db-menu.html – into /home/username/httpd/htdocs
* db-search.html – into /home/username/httpd/htdocs
* db-list (mark as executable) – into /home/username/httpd/cgi-bin
* db-search (mark as executable) – into /home/username/httpd/cgi-bin
* db.txt – into /home/username/httpd/cgi-bin

**Homework**

As homework, implement the "add" feature to add an address to the 'database'. You should only add a new address if the 'name' and 'phone' field are both not empty. As an extra bonus, you should not allow new entries that are too long to avoid someone posting large amounts of text in your 'database'. You will need to create two files, db-add.html in the 'htdocs' directory and db-add, an executable script, in the 'cgi-bin' directory. To actually append the data, you could use the 'append-redirect' symbol >> as in:

echo "$NAME,$PHONE" >> db.txt

Also, think about how to implement the 'delete' and 'sort' features, but you do not have to actually implement it.