National Center for Supercomputing Applications

HTML Forms

This is a primer for producing HTML Forms. The URL for this file is

```
  http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/
fill-out-forms/overview.html.
```

Mosaic for X version 2.0 Fill-Out Form Support

Here are details about what we have implemented for fill-out forms in Mosaic for X 2.0.

The FORM Tag

The FORM tag specifies a fill-out form within an HTML document. More than one fill-out form can be in a single document, but forms cannot be nested.

```
<FORM ACTION="url"> ... </FORM>
```

The attributes are as follows:

• ACTION is the URL of the query server to which the form contents will be submitted; if this attribute is absent, then the current document URL will be used.
• METHOD is the HTTP/1.0 method used to submit the fill-out form to a query server. Which method you use depends on how your particular server works; we strongly recommend use of (or near-term migration to) POST. The valid choices are:
• GET -- this is the default method and causes the fill-out form contents to be appended to the URL as if they were a normal query.
• POST -- this method causes the fill-out form contents to be sent to the server in a data body rather than as part of the URL.
• ENCTYPE specifies the encoding for the fill-out form contents. This attribute only applies if METHOD is set to POST -- and even then, there is only one possible value (the default, application/x-www-form-urlencoded) so far.

NOTE: If you want to use the METHOD of type POST with the NCSA httpd you will need to get version 1.0a5 or later.

Inside a FORM you can have anything except another FORM. Specifically, INPUT, SELECT, and TEXTAREA tags are used to specify interface elements within the form.

Forms are not automatically visually differentiated from the rest of a document. We recommend using the HR (horizontal rule) tag before and after a form to cleanly differentiate it from surrounding text and/or other forms.

The INPUT Tag

The INPUT tag is used to specify a simple input element inside a FORM. It is a standalone tag; it does not surround anything and there is no terminating tag -- i.e., it is used in much the same way as IMG.

In Mosaic for X, various types of INPUT tags are instantiated as Motif widgets (text entry fields, toggle buttons, pushbuttons, etc.).

The attributes to INPUT are as follows:

• TYPE must be one of:
• "text" (text entry field; this is the default)
• "password" (text entry field; entered characters are represented as asterisks)
• "checkbox" (a single toggle button; on or off)
• "radio" (a single toggle button; on or off; other toggles with the same NAME are grouped into "one of many" behavior)
• "submit" (a pushbutton that causes the current form to be packaged up into a query URL and sent to a remote server)
• "reset" (a pushbutton that causes the various input elements in the form to be reset to their default values)
• NAME is the symbolic name (not a displayed name -- normal HTML within the form is used for that) for this input field. This must be present for all types but "submit" and "reset" , as it is used when putting together the query string that gets sent to the remote server when the filled-out form is submitted.
• VALUE, for a text or password entry field, can be used to specify the default contents of the field. For a checkbox or a radio button, VALUE specifies the value of the button when it is checked (unchecked checkboxes are disregarded when submitting queries); the default value for a checkbox or radio button is "on". For types "submit" and "reset", VALUE can be used to specify the label for the pushbutton.
• CHECKED (no value needed) specifies that this checkbox or radio button is checked by default; this is only appropriate for checkboxes and radio buttons .
• SIZE is the physical size of the input field in characters; this is only appropriate for text entry fields and password entry fields . If this is not present, the default is 20. Multiline text entry fields can be specified as SIZE=width,height; e.g. SIZE=60,12.
• MAXLENGTH is the maximum number of characters that are accepted as input; this is only appropriate for text entry fields and password entry fields (and only for single-line text entry fields at that). If this is not present, the default will be unlimited. The text entry field is assumed to scroll appropriately if MAXLENGTH is greater than SIZE.

The SELECT Tag

Inside <FORM> ... </FORM>, any number of SELECT tags are allowed, freely intermixed with other HTML elements (including INPUT and TEXTAREA elements) and text (but not additional forms). In Mosaic for X, SELECT tags are instantiated as Motif option menus and scrolled lists.

Unlike INPUT, SELECT has both opening and closing tags. Inside SELECT, only a sequence of OPTION tags -- each followed by an arbitrary amount of plain text (no HTML markup) -- is allowed; for example:

```
<SELECT NAME="a-menu">
<OPTION> First option.
<OPTION> Second option.
```

```
    </SELECT>
```

The attributes to SELECT are as follows:

• NAME is the symbolic name for this SELECT element. This must be present, as it is used when putting together the query string for the submitted form.
• SIZE: if SIZE is 1 or if the SIZE attribute is missing, by default the SELECT will be represented as a Motif option menu. If SIZE is 2 or more, the SELECT will be represented as a Motif scrolled list; the value of SIZE then determines how many items will be visible.
• MULTIPLE, if present (no value), specifies that the SELECT should allow multiple selections (n of many behavior). The presence of MULTIPLE forces the SELECT to be represented as a Motif scrolled list, regardless of the value of SIZE.
The attributes to OPTION are as follows:

• SELECTED specifies that this option is selected by default. If the SELECT allows multiple selections (via the MULTIPLE attribute), multiple options can be specified as SELECTED.

The TEXTAREA Tag
The TEXTAREA tag can be used to place a multiline text entry field with optional default contents in a fill-out form. The attributes to TEXTAREA are as follows:

• NAME is the symbolic name of the text entry field.
• ROWS is the number of rows (vertical height in characters) of the text entry field.
• COLS is the number of columns (horizontal width in characters) of the text entry field.
TEXTAREA fields automatically have scrollbars; any amount of text can be entered in them.

The TEXTAREA element requires both an opening and a closing tag. A TEXTAREA with no default contents looks like this:

```
    <TEXTAREA NAME="foo" ROWS=4 COLS=40></TEXTAREA>
```

A TEXTAREA with default contents looks like this:

```
    <TEXTAREA NAME="foo" ROWS=4 COLS=40>
    Default contents go here.
    </TEXTAREA>
```

The default contents must be straight ASCII text. Newlines are respected (so in the above example there will be a newline both before and after "Default contents go here.").

Form Submission
For Method = GET
When the submit button is pressed, the contents of the form will be assembled into a query URL that looks like this:

```
action?name=value&name=value&name=value
```

("action" here is the URL specified by the ACTION attribute to the FORM tag, or the current document URL if no ACTION attribute was specified.)

Strange characters in any of the "name" or "value" instances will be escaped as usual; this includes "=" and "&". Note: This means that instances of "=" that separate names and values, and instances of "&" that separate name/value pairs, are not escaped.

For text and password entry fields, whatever the user typed in will be the value; if the user didn't type anything, the value will be empty but the "name=" part of the query string will still be present .

For checkboxes and radio buttons, the VALUE attribute specifies the value of a checkbox or radio button when it is checked. An unchecked checkbox is disregarded completely when assembling the query string. Multiple checkboxes can have the same NAME (and different VALUEs), if desired. Multiple radio buttons intended to have "one of many" behavior should have the same NAME and different VALUEs.

For Method = POST
The contents of the form are encoded exactly as with the GET method (above), but rather than appending them to the URL specified by the form's ACTION attribute as a query, the contents are sent in a data block as part of the POST operation. The ACTION attribute (if any) is the URL to which the data block is POSTed.

Test Servers
If you wish to write a prototype fill-out form and test it against a query server that simply shows you what you submitted (with name/value pairs decoded and itemized), you can use the following:

• For METHOD="POST", use ACTION="http://hoohoo.ncsa.uiuc.edu/htbin-post/post-query"
• For METHOD="GET", use ACTION="http://hoohoo.ncsa.uiuc.edu/htbin/query"

The fill-out form examples listed below use these query servers as their back ends, so you can see e fill-out form examples listed below use these query servers as their back ends, so you can see them in action and know what to expect with your own forms.

Important note: If you use the GET method in your form, you will notice that the example GET server will choke if too much data (more than a couple hundred bytes) is submitted at a time -- the server is passing the data to the form-processing module via a shell command line, and the maximum shell command line length is being exceeded. This problem does not exist with the POST method and server.

Things To Note
• Some things in the HTML+ spec aren't yet supported. In particular, mailto URLs aren't

supported as actions yet, and also the Mosaic 2.0 support only includes a subset of the allowable types of input fields (obviously "text" can also serve as "url", "int", "float", and "date" but without intrinsic range/error checking).
• A "submit" element is necessary in all forms except those containing only a single INPUT element of type TEXT (in which case Return in the text entry area submits the form) or at least one INPUT element of type IMAGE (in which case a click in the image submits the form).
• Mosaic has a stealth feature: if a single text field is in a form and the NAME of the text field is "isindex", then the submitted query will be just like you are used to getting with ISINDEX -- "url?querystring" (i.e., not "url?isindex=querystring"). This allows you to use simple fill-out forms in your documents to point to existing query servers (including Gopher and WAIS servers!). This of course assumes that the POST method is not specified for such forms.
• There seems to be a bug under IRIX 5.1.0.1 and 5.1.1 (at least) that causes text entry fields in forms to only be half as (physically) big as they should be. This is not a bug in Mosaic -- the fields are the correct size on all other X servers we have seen, and this general geometry problem afflicts other programs running under IRIX 5.1.0.1/5.1.1 also (e.g. Marc's Epoch windows come up half as big as they should, etc.). There's nothing we can do until SGI fixes the bug in their X server.

ISINDEX Handling

Now that we have this fancy fill-out form support, we're supporting the ISINDEX tag differently. Instead of having a browser-interface method (dialog box or text field) for entering a query, an instance of ISINDEX is instantiated as a preloaded fill-out form suitable for entering a query; the form is inlined at the location of the ISINDEX tag itself in the document.

Justifications for change:

• Consistent approach to query mechanisms inside Mosaic.
• Allow code and interface complexity to be reduced.
• A few nasty problems were hitting the "popup text entry field" approach attempted for prerelease 2; notably, lots of unavoidable flashing when field popped up and down and intermittent but crippling Motif geometry management problem. This change removes those problems.
• Everyone wants the ability to enter a query (and the corresponding widgets) to be present only when the document is actually an ISINDEX document; this provides that very cleanly.
• This solves existing problem of when to retain contents of search field -- since search field now exists on a per-document basis, this is very clean.
• Encourage experimentation with fill-out forms by presenting an example for existing query engines.

Examples

Twelve examples are available. All of the examples now use METHOD=POST.

- Example 1 -- a ludicrously simple fill-out form.
- Example 2 -- three text entry fields.
- Example 3 -- text entry fields and checkboxes.
- Example 4 -- changing the default values of text entry fields and checkboxes.
- Example 5 -- changing various attributes of text entry fields.
- Example 6 -- multiple, independent forms in a single document.
- Example 7 -- radio buttons, "one of many" behavior.
- Example 8 -- password entry fields.
- Example 9 -- option menus.
- Example 10 -- scrolled lists with single and multiple selections.
- Example 11 -- multiline text entry areas.
- Example 12 -- image maps in forms.

Writing Your Own Fill-Out Form Query Engine

If you want to set up one or more query engines corresponding to fill-out forms that you create for database accesses or other purposes, you may want to take a look at NCSA httpd 1.0. One of the example programs included with NCSA httpd is a sample fill-out form server (the same server used in the examples above, actually) and will provide a good model for writing your own servers.

We strongly , strongly recommend use of the POST method with fill-out forms. One reason: with the GET method, given the way many servers (e.g. NCSA httpd) pass query strings from URLs to query server scripts, you run an excellent chance of having the forms contents truncated by hardcoded shell command argument lengths. With POST (again, e.g., with NCSA httpd) you should be able to do a total end run around such problems.

NCSA httpd 1.0 (the official release) should be out shortly and will contain an example POST fill-out form server.

Fill-Out Forms in the Real World

Here are some fill-out forms in use in the Real World(tm). For up-to-date examples, search on "form" in What's New.

- Registration form at CEA.
- Cardiff's Movie Database Browser
- ArchiePlex forms interface
- Free For All
- A weather map "order form" is available from MSU.
- Document Center is a hard copy document delivery service specializing in government and industry specifications and standards. Document Center has the complete Department of Defense (DoD) Index of Specifications and Standards collection available, as well as the complete collection of the American Society for Testing Materials (ASTM). The Center also has additional industry (ANSI, SAE, IPC, IEEE, EIA) and govenment (NASA, DLA, FAA) documentation at its Silicon Valley location.
- Another Archie gateway at Oldenburg.

Comments?
Please send mail to mosaic-x@ncsa.uiuc.edu.