CHAPTER SIX

FOCUS ON THE FRAMES INTERFACE

In the ever-expanding, high-paced world of Web development, companies are forced continually to modify and upgrade their Web sites in order to take advantage of every new technological bell and whistle made available by browser manufacturers. The Web has become an MTV medium in which presentation becomes nearly as important as content. A site with yesterday's technology, especially an online store, risks becoming a deleted bookmark. For better or worse, the Web demands perpetual activity on the part of the companies that populate it.

One of the more recent technological advancements to have hit the is that of Frames. Frames allow you to not just utilize the presentation of your information within the browser window, but they also allow you to change the nature of the browser window itself. Frames give you the ability to break the base window into separate cells (or frames) as if each were its own browser. Each frame has its own history list and can be manipulated independently of every other frame, or it can be used to change the contents of other frames.

As with any HTML tag, Frames can be a very useful method of helping your customers get to the information they want as quickly as possible. However, when used poorly, Frames can also be atrocious and can actually make browsing your site awkward. We recommend that before you modify your site to include Frames, you make a careful study of what works and what does not. After all, the more powerful our tools, the easier for us to misuse them. A large list of Web store implementations can be found at the following URL:

```
http://www.eff.org/~erict/Scripts/web_store.html
```

Many of the stores on the list take advantage of Frames. We recommend you browse through the list before you settle upon your own design.

However, once you have decided that Frames will enhance your own catalog, making Frames work with the Web store is a fairly simple process. This chapter will discuss what changes you will need to make to your scripts in order to adopt a Frames interface.

What Are Frames?

Though it is not within the scope of this book to present a detailed discussion of the HTML necessary to build complex Frames-based Web sites, we will take a moment to introduce the main tags which we use in our examples.


NOTE

One good resource on Frames is O'Reilly's, "HTML: The Definitive Guide" written by Musciano and Kennedy. There are also many other HTML Frames references available.

The basic components of a Frames-based site consist of three HTML tags: <FRAMESET>, <FRAME>, and <NOFRAME>.

<FRAMESET> defines the collection of Frames that will describe the browser window. The <FRAMESET> tag takes two attributes, **ROWS** and **COLS,** which define how many rows and columns you will break the base window into and how much of the available screen each frame will take up. Thus, the following tag would create two equal-sized rows of frames, each with three equal-sized frame cells:

```
<FRAMESET ROWS = "50%, 50%"
          COLS = "33%, 33%, 33%">
```

What actually goes within a frame cell is defined by the <FRAME> tag. The <FRAME> tag takes several attributes including **MARGINHEIGHT**, **MARGINWIDTH**, **NAME**, **NORESIZE**, **SCROLLING**, and **SRC**. However, for the purposes of the Frames implementations which we use, the **SRC** and **NAME** attributes will be enough.

The **SRC** attribute defines the location of the object that will be loaded into the frame cell while the **NAME** attribute defines what the frame cell will be referred to within the entire frame document.

For example, consider the following <FRAME> tag

```
<FRAME SRC = "frontpage.html" NAME = "main">
```

In this example, the frame cell will display within it the contents of the document **frontpage.html**. Further, it may be referred to as **main** by other objects in the window. Thus, the following hyperlink when clicked would cause **Vowels.html** to be loaded in the Frame defined above (replacing **frontpage.html**).

```
<A HREF = "Vowels.html" TARGET = "main">
```

Thus, by using the **TARGET** parameter, we are able to affect the contents of any named frame.

Finally, the <NOFRAMES> and its corresponding </NOFRAMES> tags are used to define a default HTML page for browsers that do not support Frames technology. Thus, you might include the following section in any site based on Frames:

```
<NOFRAMES>
We are sorry, but this site takes advantage of Frames.  Why not con-
sider upgrading your browser to Netscape 3.0
</NOFRAMES>
```

Setting up the Web Store to Use Frames

As stated earlier, using a Frames-based Web store is pretty simple. In fact, you need only change a few aspects of the interface. Almost every other setup factor will depend on whether you use a Database or HTML back end. But your choice of back end you use makes no difference in the Frames interface. The Frames interface can handle either choice.

In fact, there are really only two things that you need to do in order to transform your Web store into a Frames-based Web store. First, you must make sure to create all the elements the frames that you wish to display. Second, you must make sure not to fall prey to endlessly recursive front pages.

Let us first deal with preparing the elements of the frames displays. In the sample Frames store we provide in the distribution as **web_store.setup.frames** (notice that we use a Database store back end by default), the interface is divided into three frames. Figure 6.1 shows the results of **frames_frontpage.html** on the Web:

```
<HTML>
<HEAD>
<TITLE>Selena Sol's Meme Mart</TITLE>
</HEAD>
<FRAMESET COLS = "111, 80%">
<FRAMESET ROWS = "90, 60%">
<FRAME SRC = "web_store.cgi?page=../home.html&cart_id="
       SCROLLING = "no">
<FRAME SRC = "web_store.cgi?page=../toc.html&cart_id="
       SCROLLING = "auto">
</FRAMESET>
<FRAME NAME = "main"
       SRC = "web_store.cgi?page=../outlet_frontpage_db.
             html&cart_id=">
</FRAMESET>
```

As you can see, **frames_frontpage.html** defines the frames-based interface as having two navigational frames and one display frame.

The first navigational frame simply loads a static HTML document referencing the front page with the following code:

```
<CENTER>
<A HREF = "web_store.cgi?cart_id=&page=../outlet_frontpage_db.html"
   TARGET = "main">
<IMG SRC = "Html/Images/animated_circuit.gif"
     BORDER = "0"></A>
<BR>Meme Mart Home
</CENTER>
```

Figure 6.1 The Frames-based Web store.

The link, when clicked, instructs the Web browser to use **web_store.cgi** to load **outlet_frontpage.db.html** into the main window.

The second navigational window is the primary navigational frame which provides links for every place in the store with the following code:

```
<CENTER>
<A HREF = "web_store.cgi?page=Letters.html.db&cart_id="
   TARGET = "main">Letters</A>
<P>
<A HREF = "web_store.cgi?product=Numbers&cart_id="
   TARGET = "main">Numbers</A>
```

```
<P>
<A HREF = "web_store.cgi?product=Words&cart_id="
   TARGET = "main">Words</A>
<P>
<A HREF = "web_store.cgi?product=Memes&cart_id="
   TARGET = "main">Memes</A>
<P><HR><P>
<A HREF = "web_store.cgi?modify_cart_button=yes&cart_id="
   TARGET = "main">View Cart</A>
<P>
<A HREF = "web_store.cgi?change_quantity_button=yes&cart_id="
   TARGET = "main">Change Quanity Form</A>
<P>
<A HREF = "web_store.cgi?delete_item_button=yes&cart_id="
   TARGET = "main">Delete Item Form</A>
<P><HR><P>
<A HREF = "web_store.cgi?order_form_button=yes&cart_id="
   TARGET = "main">Order Form</A>
</CENTER>
```

This is basically the familiar list of products page discussed in Chapter 3 and Chapter 4, with the bonus of a set of cart manipulation links. The navigational frame becomes a site map allowing customers to jump anywhere in the site regardless of where they are located.

Finally, the browser includes a "main" frame into which all subsequent store pages are loaded. Initially, **outlet_frontpage_db.html** is loaded into the frame.

One crucial thing to note about the use of frames is that every frame is filtered through **web_store.cgi**. As you will recall from previous chapters, it is essential that no HTML page be introduced to the customer without its first being filtered through the script. As always, every link includes the **cart_id=** flag which the script will recognize and replace with the actual unique cart i.d. of the customer when it loads **frames_frontpage.html**.

The second issue to deal with when creating a Frames interface is the variable **$sc_no_frames_button** defined in the Setup file. **$sc_no_frames_button** must be set to nothing if frames are to work. The reason for this is that frames have the nasty possibility of creating endlessly recursive front pages. That is, consider what would happen if you had the **Return to Frontpage** button in the "main" window. If the customer clicked on it, the **frames_frontpage.html** file would be loaded inside the main frame. Thus, the main frame

would be subdivided into the three basic frames. Figure 6.2 depicts this unfortunate situation.

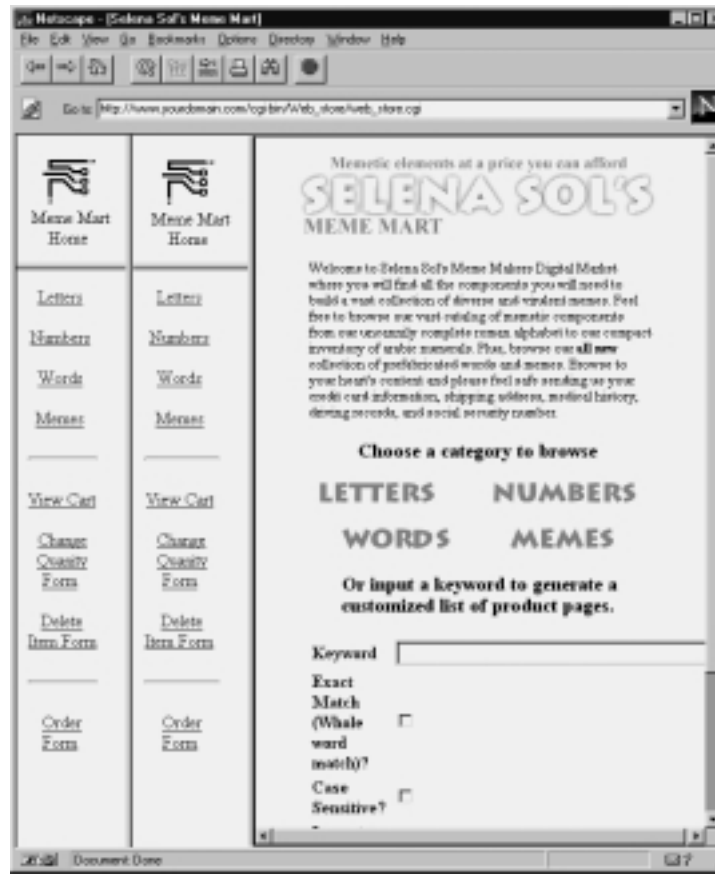

Figure 6.2 Recursive frames.

This could go on indefinitely until the browser fails. Thus, we have isolated out the Return to Frontpage button into a setup variable. If you set this variable equal to nothing, then you will not get a button in your main window and will only be able to reload the front page from the top left frame which will reference **outlet_frontpage.html** instead of **frames_frontpage.html**.

Summary

In summary, to utilize a Frames-based interface, you must do two things. First, you must create the HTML pages that will fill the frames divided within the browser window. Plus, each of these HTML pages must be filtered through **web_store.cgi** before it is displayed in the frame however.

Finally, in the Setup file you must set **$sc_no_frames_button** equal to nothing so that you will not create recursive frames within your browser window.