CLASSIC
SYSTEM SOLUTIONS, INC.

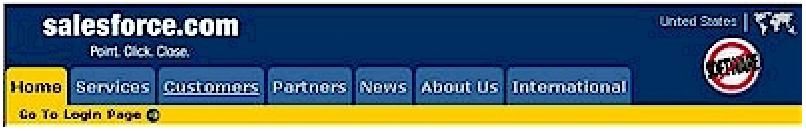
Using Tab Patterns with Web Applications

By James Hobart, President, and Andrew Harden, Dir. of Professional Services, Classic System Solutions

January 2003

Christopher Alexander, the famous architect and author of "A Timeless Way of Building" provides us with the inspiration behind patterns. "A pattern describes both a problem, which occurs over and over again in our environment, and then describes the core of the solution to that problem in such a way that you can use this solution a million times over, without ever doing it the same way twice."

Our continuous work on enterprise application design projects has allowed us to encounter, ponder, and solve many user interface problems. One of them is selecting a single item from a list of categorized items using a file folder metaphor, in other words a tab pattern.



An example of the use of tabs.

Issues Solved by the Tab Pattern

The purpose of the tab pattern is to allow the user to easily see all of the available options and select one. When implemented correctly, the user can easily relate the tab metaphor to the tabs on a traditional set of paper files. Easy recognition of the selected tab and a basic understanding of how to make a selection should come naturally to the user. The tab pattern can also be used to show a complex object as whole, with each of its peer components displayed as a separate tab.

How the Tab Pattern Works

In general, there should be a set of six to eight (6 - 8) tabs from which the user can select. Using more than eight (8) tabs can significantly reduce usability, as you are asking the user to select a single item from a domain set that is too large.

Typically, the information to be shown on the tabs should be categorized in a way that matches the user's conceptual model of the data. The goal of tabs is to instantly show the user the available information and make it reachable with one click. The information on each tab is typically unrelated to the information on the other tabs, and, ideally, the information is already known to the user or is easy to categorize. While tabs can be categories, they can also be points on a discrete scale; for example, the alphabet in an alphabetical index.

Tabs should be presented like tabular sheets. This relationship can be enforced by showing the tabular sheets visually, i.e., connecting the information area to the current tab.

The tabs should also be presented horizontally. This places an important limitation on the number of tabs that fit in the layout. This number is usually smaller than ten (10), unless an alphabetical index is being used.

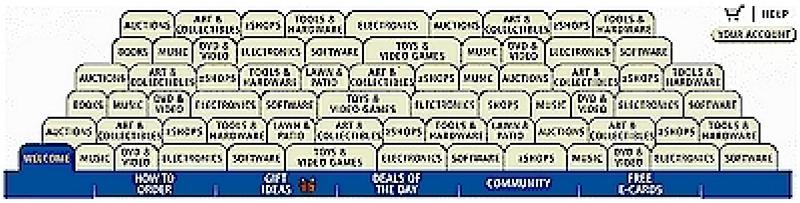
In summary,

- Divide the information to be shown into tabs and present one tab at a time.
- Show all of the tabs and allow the user to easily switch between them.
- Show the tabs and labels horizontally.
- Place the information for the current tab immediately underneath it.
- Visually highlight the currently selected tab.
- Always present a default selected tab.

When to Use a Tab Pattern

You should use a tab pattern when the categorized information is static and mutually exclusive, and when a folder metaphor will match the user's mental model of viewing the data. You should avoid using tabs for dynamic information.

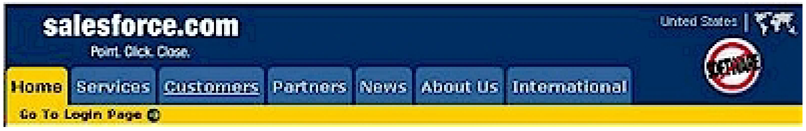
To increase the utility of the tab metaphor, some designers use multiple rows of tabs. Unfortunately, this increase in utility comes at great sacrifice to usability. Multiple rows of tabs are confusing and intimidating, particularly to new users who find it difficult to locate the desired tab. In addition, screen real estate is compromised, and the repositioning of tabs to the front row causes tremendous confusion and complexity. For instance, Amazon got into trouble when they decided to use a tab for each store they opened. Soon, a single row of tabs evolved into a multi-row nightmare, which they eventually abandoned.



Too many tabs. Avoid using dynamic data on your tabs.

Examples

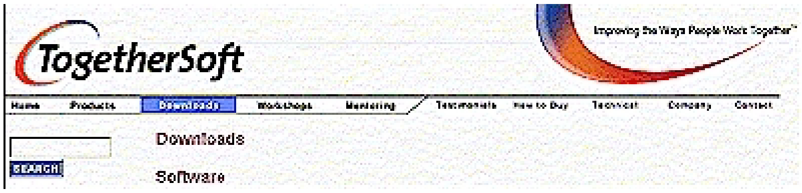
Good Example (HTML) - The example below illustrates the use of tabs in a web application. HTML tables and in-line images are combined to create the illusion of a tab control. The user sees multiple tabs on a single web page, when, in reality, selecting a tab loads a new HTML page from the server (often referred to as round-trip HTML).



Good example of tabs in HTML.

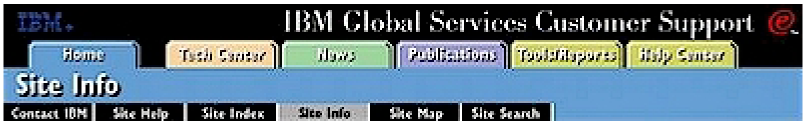
Note from the field: For high volume transactional systems displaying complex data spread across a set of tabs, you can open multiple threads to a server and asynchronously load and cache data from multiple tabs, thus reducing trips to the server and significantly improving user interface response time as the user moves between them.

Bad Example (HTML) - The Downloads tab in the example below is currently selected, but it seems to be part of one of two much larger tabs each having five menu items. The diagonal line between "monitoring" and "testimonials" confuses the eye by suggesting that the right hand tab is layered in front. Tabs should be recognizable as tabs; text contained in a rectangular box with square corners does not look like a tab. Too often, web page designers confuse navigation bars with tabs.



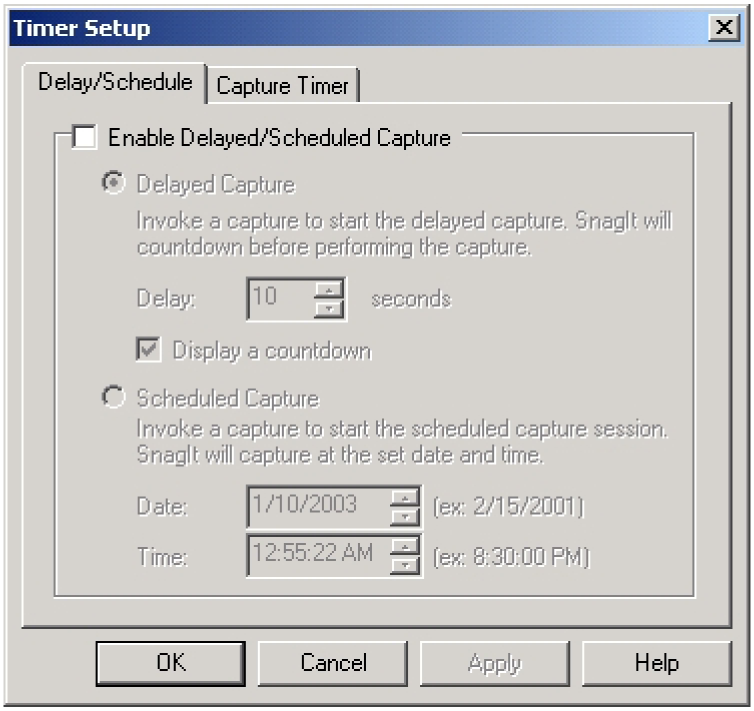
Bad example of tabs in HTML. The appearance is confusing.

Good and Bad Example (HTML) - Notice how the Home tab in the example below is separate from the remaining tabs. This approach can be useful if there is a logical distinction between the tab categories. This is also an example of how NOT to use colors and font styles. The white text looks washed out over the pastel colors, and the shadowed font is very hard to read. Black or a dark font color would have been more effective.



Good and bad example. Good for the separation, but bad in color.

Good Example (VB, C++) - This example is typical of a property sheet in a Windows application. The two distinct tabs can easily be selected and combined under a general feature labeled "Timer Setup". This usage avoids the need to create two separate dialogs and associated menu items, thus flattening out the navigation model.



□ Enable Delayed/Scheduled Capture

● Delayed Capture

Invoke a capture to start the delayed capture. SnagIt will countdown before performing the capture.

Delay: 10 seconds

☑ Display a countdown

○ Scheduled Capture

Invoke a capture to start the scheduled capture session. SnagIt will capture at the set date and time.

Date: 1/10/2003 (ex: 2/15/2001)

Time: 12:55:22 AM (ex: 8:30:00 PM)

Good example in VB or C++.

Free Sample Code Templates

HTML code, PhotoShop slices, and style sheets are available to users of GUIguide, our enterprise GUI design repository. Or, you can contact us for **free** sample tab navigation templates!
This pattern and many others will be discussed in detail at our upcoming Advanced Web Design seminars.

Further Reading

The Humane Interface – Jeff Raskin
Software for Use - Constantine and Lockwood

About the author:

James Hobart is an internationally recognized user interface design consultant based in California, USA. He specializes in the design and development of large-scale, high-volume client/server and web applications. He is an expert in GUI design for transaction processing systems and strategies for migration to thin-client graphical user interfaces. He can be reached at jimh@classicsys.com

Find more articles on Usability at www.classicsys.com