Concise paper

Putting .NET Web Forms in the Frame

John Peppiatt **Mike Lopez**

Manukau Institute of Technology
Manukau, New Zealand
john.peppiatt@manukau.ac.nz

ABSTRACT

The concept of HTML Frames, and their more restricted relation, in-line frames (IFrames), were never designed to assist with web application development. In fact, as many commentators have noted, neither HTML nor the HTTP transport were designed to support state-full web applications at all. None-the-less, the internet, supported by current browser technology, is being used as a major platform for application delivery and frame technology offers a tantalising mix of real-estate management and logic encapsulation.

Historically, HTML Frames were introduced to support the viewing of more than one page in the browser's window. In conjunction with the DHTML object model, Javascript techniques can be used as a client side technology to make frames inter-operate; the most common example being content navigation through one frame and main page display through another. The introduction by Microsoft of .NET web forms provide a rich server centric programming model for web applications but nothing in .NET provides server side support for an application made up of interoperating frames.

This paper looks at some very simple technology that provides a server side paradigm for the development of web applications with Frames. The architecture is described and the basic mechanisms are illustrated.

1. INTRODUCTION

Information Technology rarely seems to evolve to fit a well thought out strategic plan. Innovation comes from many sources and many technologies are created; some (not always the most deserving) gain traction and market influence - thereafter other technology is grafted on top to address requirements that were never a part of any innovator's vision. A prime example of this must be the evolution of internet technologies; these have evolved from a simple method of rendering and linking documents to a rich platform for hosting interactive applications. This paper is about a technique that helps expand the paradigms available for devising web based applications. It is presented in the context of Microsoft's Visual Studio .NET WebForm architecture but the concept can be applied to many web development frameworks.

2. BACKGROUND

The authors observed that many web based applications organize GUI real estate into logical regions where some regions remain relatively static and other regions are used to display responses to requests. Despite this intention, the techniques used to achieve this are visually irritating (because regions are often repainted despite there being no change), and the coding techniques that are used poorly encapsulate the purpose. The other curiosity is the fact that although HTML (see www.w3.org/TR/html401) has evolved to incorporate the ability to organize the browser's display into regions - through the use of HTML FRAMESET and IFRAME (inline frame) tags (see www.w3.org/TR/html401/present/frames.html) - these are rarely exploited to devise interactive applications. In fact the www.w3.org site describes FRAME technology as purely a means of presenting content in regions. Other bodies, such as the Web Design Group http://www.htmlhelp.com/design/frames/usage/, also promote the use of frames purely for content organization and thus emphasize this prevailing paradigm. There is however linguistic support for cross-frame operations in JavaScript and this offers the potential for creating a richer user experience.

3. SOLVING THE PROBLEM

The authors surmised that there must be reasons why FRAME and IFRAME concepts are not frequently used - despite their obvious potential. The authors identified several possible reasons

- The history of HTML technology itself. HTML is essentially a document oriented technology. The programmatic generation of documents through scripting techniques, and the modern document generation techniques like WebForms and JSP (that stemmed originally from CGI, ISAPI, NSAPI concepts) focused on generating single documents.
- FRAMESETs and to a lesser extent IFRAMEs were designed to present static unrelated (or loosely coupled) documents side by side; they were not designed to present an application interface.
- Although client side scripting (e.g. Javascript) provides a mechanism for changing the contents of FRAMEs and IFRAMEs, there is no natural way of addressing this during the server side creation of a document
- Mainstream development tools such as Microsoft's Visual Studio .NET do not present a standard design pattern (project type) for developing solutions this way and developers are inclined to adopt standard patterns offered by tool vendors.

To progress the problem the authors took the view that if a technology was introduced that made the management of FRAMEsets and IFRAMEs simple at the document generation stage a new paradigm could evolve.

4. EXPLORATION OF TECHNOLOGY

The technical problem to solve was to create a mechanism by which content could be created during the generation of an HTML document which, when pushed out to the client, manages the content of a FRAME of IFRAME. Unfortunately, HTML and the client side event model defined in DHTML have lead to the mainstream browsers compartmentalizing the rendering process into regions and limiting the communication between FRAMEs even at the client side. None-the-less, it is possible to affect a pull (navigate) operation in one FRAME by executing JavaScript from a document loaded in another. The approach the authors took was to simulate a server side push by rendering the JavaScript code to affect a reload operation in another FRAME immediately the browser began rendering the returned document.

By way of explanation, consider the diagram below (Figure 1).

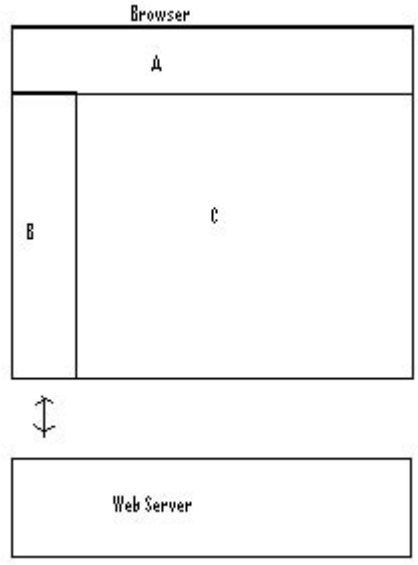

Figure 1

The rectangles A,B,C represent three FRAMEs in an HTML FRAMESET. B contains a document with an HTML form (e.g. .NET WebForm). When the form is transmitted to the server a new document is returned and is rendered in FRAME B. During the rendering process JavaScript code transmitted as a part of the new document is executed by the browser which causes navigate operations to occur to one or both of FRAMEs A and C. The navigate operations can either be to static pages or more typically to other dynamically generated pages (e.g. other WebForms).

Almost by definition, an application constructed this way must be viewed as the aggregation of the documents that are currently, or can potentially be, displayed together with an underlying state that controls the aggregate behavior. Fortunately, any state management approach that works by means of a browser available handle (e.g. cookie that is available across FRAMEs) can be used to identify a server side state store (e.g. WebForm sessions). Failing this, query strings can be used in the urls to navigate to content.

5. IMPLEMENTATION

To prove the operation of this technology and to achieve developer convenience the authors developed a reusable web control, called a FrameController, to extend the developer framework in Visual Studio. The technology was then used in production for several products currently in use at Manukau Institute of Technology.

For those familiar with Microsoft's WebForms, this web control is a component that is visible at design time but has no visual representation at run time. It has one simple method,

AddRequest(framename as string, url as string , optional iframe as boolean=false)

In the context of a page processing cycle for a document from frame B, one or more requests can be sent. E.g. if fmc is an instance of the component then

```
fmc.AddRequest("C","frmContents.aspx")
fmc.AddRequest("A","frmBanner.aspx")
```

causes both frames C and A to be reloaded once the browser begins rendering the document returned as a request from frame B. To manage state, the authors typically use a single WebForm Session object for the complete definition of state; this is passed between all cooperating pages in the application.

6. ISSUES ENCOUNTERED

The technology was initially developed and tested totally within Visual Studio using Internet Explorer and worked exceptionally well. Several production Web sites used mainly in an Intranet context at MIT were developed this way; however as often happens users began experimenting with different browsers (specifically FireFox) and found the technology did not function. Investigation revealed the problem to be due to differences in the dialects of Javascript supported by the different browsers. A form of Javascript was found that was supported by both browsers but it does highlight a weakness generally when using DHTML techniques on web sites.

The second issue that arose when one of the authors had elected to use a query string style URL to pass parameters between frames. In this scenario, the way a browser (or indeed a proxy) is set to cache a page can result in no message being sent to the server (the browser decides it has the data and blocks the request). Although, changing the caching options on the client can fix the problem, the authors found that incorporating the HTML META tags designed to control the caching in the document will generally work – provided the browser is sufficiently capable.

7. COMPARISONS WITH OTHER APPROACHES

To some extent, IFRAMEs evolved to provide the ability to create (or reference) a 'document within a document'. There are some major limitations of working purely with IFRAMEs:-

- It invites a monolithic approach to a complex design problem.
- It fails to offer the inbuilt features of the browser to allow the user to resize FRAMES.
- The container document is also redrawn - which is visually distracting.
- Larger data transfers from the server make the process 'clunky'

The other common techniques include

- The use of server controls to help partition the logic of a complex document. These fail to address the repainting and resizing problems.
- The use of client side plug-ins to download into the client (e.g. Flash or ActiveX) or extensive use of JavaScript. These techniques require the developer to continuously choose where logic is placed – client side or server side – and more often than not a muddled solution results.

The approach described invites the developer to create an application that

- is made of pages where state is shared through a well encapsulated medium,
- the rendering of regions is simple and independent
- the philosophy of WebForms - minimize the need to work with client side logic - is promoted

8. CONCLUSION

This paper is primarily about a paradigm for interactive web page development and secondly about some technology that has been developed to enable the practice. Undoubtedly, new genera-

tions of development tools will emerge that allow developers to produce web applications that:-

- begin to resemble the richness of Win-form type GUI functionality
- promote well layered coding techniques
- speed up the development cycle.

For a while at least, the technique just described is, like many innovations, grafted onto other technology to remedy a weakness - not just of available functionality -but also in underlying design. Such things are stop-gaps, but with the entrenchment of HTML and DHTML, stop-gap technology will continue to play a major role in enabling the web application developer community.

REFERENCES

www.w3.org – World Wide Web Consortium, May 2005

www.w3.org/TR/html401 - HTML 4.01 Specification, World Wide Web Consortium, May 2005

www.w3.org/TR/html401/present/frames.html - Frames in HTML Documents, World Wide Web Consortium, May 2005

www.htmlhelp.com/design/frames/usage/ - Web Design Group, May 2005