Web – From Nothing and Beyond

Presented by: **Dani Beaubien**

INTRODUCTION

As you know, 4D is a relational database that is highly coupled with an extremely effective RAD tool for developing desktop applications. This can also true when it comes to developing web applications. The gotcha, just like with the desktop side of 4D, is there is a steep learning curve that must be traversed before powerful web applications can be developed. One of the difficulties is that the basic terminology of a traditional 4D desktop application is fundamentally different than a 4D web application.

This article and the accompanying summit session will help you "quicken" your climb up the learning curve by giving you a solid working foundation and a real working web shell that I have built and refined over the last few years. Hot topics like session management, logins, searching, and how to organize your web pages are all covered. Even for the seasoned pro, there are items and approaches that I have taken that are worth considering.

The advantages of using 4D to drive a dynamic web site are numerous. The ability to fully utilize our hard-earned 4D skills in this new environment is the challenge.

THE STATELESS REQUEST RESPONSE LOOP

The foundation of anything published on the web is infamous "request – response" loop. The client makes a request and the server responds. The client makes another request and the server makes another response. Everything that the server needs to respond is included within the client request either directly in the HTML or in the HTTP header. The client is aware of "its state" and passes that on to the web server. This is fine for a simple web site, but when it comes to a dynamic web site, the server needs to start tracking information.

When we are using 4D to build a desktop application, we are in full control of the 4D processes. We know what the process variables will be defined every time the user does some action. We don't have to worry about which user is doing what and on what data.

When it comes to 4D on the web. Every time the user makes are request to your 4D Web Server you will have to "re-create" the state of that user. You have to recreate what data they are working on, what rights they have, what their user record is. All this has to be done "every time" the user makes a request. You don't even have any control as to which process the request comes in on. This is a huge paradigm shift, and a very challenging one to overcome. In some ways, you have to throw out what you know about 4D when it comes to the 4D web.

WHAT YOU SHOULD KNOW (OR LEARNING FIRST)

The following is a list of topics that I have found very helpful and have enhanced my ability to create and manage web applications using 4D. As always, the more knowledge that we have the better and more complete our solutions will be. Since a 4D Web solution is really a traditional 4D solution coupled with a traditional Web solution, this means that we have two environments that we need to understand and build one solution that handles both.

4D

This might seem obvious, but what is not obvious is that if you are writing a 4D based Web application, then you are writing a multi-user application where multiple users are all being served from a common set of variables and processes. A good understanding of multi-users issues like record locking and multiple processes are valuable. Process variables and their limitations are also quite relevant.

HTML

This might seem obvious as well. You need to understand HTML to the point where you can read it and understand what you are reading without using a fancy WYSIWYG editor. You can get by without this working knowledge, but it will severely limit your debugging abilities and the HTML you will be able to produce.

CSS

This kind of fits in with HTML but is worth mentioning. Having a comfortable feel with CSS can really help simplify the HTML that you have to create. Leave all the "pretty" stuff in the style sheet and your HTML will be a fraction of the size if you did not use one or more style sheets. There are many web sites that give tutorials and examples of CSS. Along with those are style sheets that you can use in your own applications. Using a pre-existing style sheet as a starting point can be quite beneficial because you do not need as much understanding of CSS to use a style sheet.

HTTP, URLs, TCP/IP

Some might not feel that this is required, but I feel that you need to be exposed to this and have, at the very least, a basic understanding of what these items are. There are some useful items that are part of HTTP that you can use to communicate in special and unusual ways with the web clients. Your web client might not be a web browser and in that case, understanding these items are even more important.

JavaScript

This one is more optional but there are certain GUI elements that can only be done with JavaScript. If you don't want to get to deep into JavaScript, then do what I have done, find a few good web sites that have sample scripts that you can download and use whenever you need.

ObjectTools

ObjectTools is a fantastic plug-in and it a very useful tool for state/session management of the connected users. There will be more on this later. If you are going to use ObjectTools then you will need to invest a bit of development time in having a good object viewer so that you can view (and potentially) modify objects that are in memory.

OTHER THINGS THAT CAN HELP YOU

What follows is a list of items that I have found to be useful in the process of developing 4D web based applications.

David Adam's "The 4D Web Companion"

This book will give you a lot of background information. Probably more than you need, but it is worth the effort. It covers all aspects of using 4D to publish information on the Web.

Good Text Editor

The chances are extremely good that you will be hand editing HTML files. If you do, then you will want a good text editor. Particularly one that has a good search and replace functionality. You will need to do this across multiple files and perhaps even multiple directories. Apple's Xcode does this well. BBEdit works well also. Syntax coloring can be very helpful, but can be a bit of a pain when it comes to 4D web tags.

Reference Books

A quick trip to you local computer bookstore can get you a good CSS reference book and a good HTML reference book. You want books that you can use as a quick reference. There are also some good resources for on-line help.

SLL

Your solution might want to use SSL, if it does, then you will need to know how to do this (or who to pay to get it configured).

Apache/4D WebSTAR

Using a third party web server for serving the static parts of your web solution is an easy and obvious way to increase performance and potentially make your workload easier. This also applies to SSL as well. Using Apache or 4D WebSTAR to handle all the SSL and let your 4D Web application sit behind them is a good utilization of resources and can make your web application simpler to implement and configure.

Caching versus Dynamic Pages

An understanding of using cached pages versus generating all the pages on demand is useful. There are positive and negative tradeoffs to having a fully dynamic web site.

MAINTAINABLE CODE IS MY GOSPEL

All the studies that I have seen say the same thing ... Creating code is, for the most part, bug free. Modifying existing code, whether it is to fix a bug or enhance it in some way, is generally not bug free. That is when the bulk of errors are introduced, when modifying existing code. A lot of the newly introduced bugs are hard to find. That code could be 5 years old or 3 days old. The challenge, when modifying code, is to be in the same headspace as the author at the time that they wrote it.

There are many web sites and books that are devoted to this topic. Personally I think that Steve O'Connell's book *Code Complete* is a fantastic and it covers a lot of topics that are worth considering. How you organize your code, how you name your variables, how you well you understand the problem you are solving, how you comment, how experienced you are, how many coders there are, if you have coding standards, all these affect maintainability.

Writing maintainable code is like being spiritual. If you want to be spiritual, then you are all the time. It is a way of life and a way of thought. Everything that you do and think is geared towards being maintainable. You understand and accept that there are always ways to improve and you continually strive to be better. It is a way of life. It is a personal concept. The benefits are massive. Less bugs. More productive work. More code re-use. Better understand of what the customer wants (since you have less bugs, you have more time to focus on customer needs). More enjoyment in what you do since you are creating more rather than fixing.

Wrappers are not maintainable code. Wrappers are wrappers. Maintainable code is code that can be changed and if an error is introduced, it is discovered quickly. It must be changeable by someone other than the original person that wrote it.

Optimized code is often not easily maintained. Only optimize if it is needed since you will make it more difficult to make changes in the future. Use performance measuring tools to verify where optimizations should occur and what effect the optimizations have had.

HTML IS CODE AS WELL, NOT JUST 4D

This is an important concept. When you are using 4D to create a web application. Not only are you writing 4D code, but you are writing HTML with embedded 4D HTML tags. 4D executes both. Since both are "code", that means that there is an opportunity for code re-use and good design to enhance maintainability.

Your design of the HTML and the relevant 4D code can have a huge impact on how easy it is to maintain and how prone to the introduction of errors. One of the hardest design decisions is to figure out where the dynamic HTML code will live. Will it be part of the HTML pages, via the 4D HTML tags or Active 4D, or inside 4D directly? This is a very hard decision to make. It depends on how comfortable you are with HTML and with 4D. The choice that you make will affect how maintainable your application is. Ideally you want to utilize code re-use as much as possible.

A typical web page can be broken into a few separate areas. Using this web page as an example. There are 7 distinct components that this page is comprised of:

- Header
- Tabs
- Navigation column on the left
- Footer (just beneath the navigation)
- Center body column
- Team Roster on the right
- Regular stats on the right
- Personal stats on the right

Each one of these components is it's own HTML file. The HTML file that this page is from includes those other components. The benefit of this that if I decide that I want to change the tabs, I just change the tab HTML file and the entire site is then using the new tabs. I only make the change in once place and everywhere it is used, the change is made. This is a very powerful way to design your applications. It greatly reduces the complexity of the HTML, since each file is smaller and easier to understand and maintain. There is a trade off, that trade off is that there are many more files and they are dependent on each other. Another trade off is that since these HTML components are not

complete HTML pages, the usual WYSIWYG editors will not work which is where a comfortable working knowledge of HTML is helpful.

Using this technique, we are able to easily re-use HTML code. This is especially helpful if you are building more than one web application. You can build a library of useful HTML components that you can then re-use many times. Another benefit is that it makes it much easier to make sweeping changes to your application.

MY APPROACH, SOME BACKGROUND

4D has two different modes: contextual and non-contextual. Back in the v6.x days of 4D contextual was all that was available and now that 4D offers non-contextual serving of web pages, I have never looked back. All the information that I am presenting is based on the non-contextual mode of 4D.

4D is able to serve HTML pages directly. It does this when the requested file exists in it's web folder. Generally, I don't do that. I have a public (the 4D web folder) that contains style sheets, image files, JavaScript files. Just items that do not require dynamic content are placed in this folder.

I have another folder that contains all my HTML source files. When 4D receives a URL for a file that doesn't exist in it's web folder then it considers that as a "Custom URL" and it is then handled entirely by your 4D code. I prefer this method since that means that nothing will be served to the user without my code telling 4D which files to serve. In my opinion, this is more secure and allows me to hide the internal structure of my application files.

TYPICAL PROGRAM FLOW

When the built-in 4D Web Server receives a custom URL request, the two built-in method "On Web Authentication" and "On Web Connection" will run. See David Adam's book for full details on when these two methods will and will not be executed. This can vary depending on the URL and what content is being asked for.

"`On Web Authentication`" can be used to force the web client to ask for a user id and password. For the most part, I do not use this method since I have my own user/password system.

"`On Web Connection`" is where all the action happens. If a custom URL has come in, then your code must determine which files are then sent out using the built-in 4D commands "`SEND HTML BLOB`", "`SEND HTML TEXT`" or "`SEND HTML FILE`". When this method is executed, you **MUST** assume that it is in a brand new process without any variables defined. In fact, the previous request, by some other user, that 4D just handled might have been served from this same process, which means that the process variables will still have those values in them. Each time a request comes in; you will need to mange any previous values in the process variables.

Something that is handy is that any 4D HTML tags that you have imbedded inside your html page; 4D will handle those tags in the same process with the same process variables and the same record selections. This is a good thing to understand so that you can take advantage of that.

A big question that often arises is when do the embedded HTML tags get executed. This is best illustrated with an example. Lets say we have this simple page:

```
<html>
      <head><title>Welcome to Summit 2005</title></head>
      <body>
            <!--#4Dvar includeFile("/includes/header.html")-->
            <h1>This is my page</h1>
            <p>The time is <!--#4Dvar current time-->.</p>
            <p>The last time you were here was: <!--#4Dvar
theLastTimeIWasHere4DMethod--></p>
            <!--#4Dvar includeFile("/includes/footer.html")-->
      </body>
</html>
```

The user requests this page using a custom URL. `On Web Authentication` executes. `On Web Connection` executes. The code that you have determines that this file needs to be sent and executes a `SEND HTML BLOB` with the HTML above in the blob. The `On Web Connection` method finishes. Now, 4D takes the HTML that you want to send back and parses it for 4D Tags. Then, starting at the top, tag by tag, interprets, executes

and replaces the tag with any results. If those 4D tags call a 4D method, it will be in the same process, same process variables, same record selections, etc. If a method makes a change to the 4D environment, then that change will still be in effect for the next embedded 4D HTML tags that are further down the HTML document. In effect, the HTML document becomes a method that 4D goes through and interprets from top to bottom. Once it gets to the bottom, then it completed and the results are served to the requestor.

TO 4D OR NOT TO 4D – WHERE TO PUT THE CODE

When you are building your HTML code, should you try to put as much of the 4D code into the HTML file itself or call a routine in 4D and have it return the HTML? This is a big question and one that is dependent on many issues. Check out the following two very simple pages and the 4D method.

```
<html>
      <head><title>Page 1 – Code is in the HTML</title></head>
      <body>
            <!--#4Dvar includeFile("/includes/header.html")-->
            <h1>Product List</h1>
            <div>
                  <!--#4Dloop [Products]-->
                        <!--#4Dvar [Products]Name--><br>
                  <!--#4Dendloop-->
            </div>
            <!--#4Dvar includeFile("/includes/footer.html")-->
      </body>
</html>
```

This approach has the advantage that there is more code in the HTML files which means that you can make changes directly in the HTML without changing the 4D application. There are advantages to this for quick turn around of bug fixing. Potentially, you can fix a bug on a deployed 4D Web application by editing one or two HTML files rather than having to rebuild and redeploy the entire 4D application. You get to keep your 4D application smaller since there isn't extra code to handle some specifics for the web.

```
<html>
      <head><title>Page 2 – Code is in 4D</title></head>
      <body>
            <!--#4Dvar includeFile("/includes/header.html")-->
            <h1>Product List</h1>
            <!--#4Dvar Product_ShowList-->
            <!--#4Dvar includeFile("/includes/footer.html")-->
      </body>
</html>
```

```
` 4D METHOD: Product_ShowList
C_TEXT($0)
$0:=Char(1)
For ($i;1;Records in Selection([Products]))
      $0:=$0+[Products]Name+"<br>"
end for
```

This approach has the advantage that there is more code in 4D. Assuming that the 4D application is compiled, then there is a performance improvement since all the HTML code in interpreted. Any field or table name changes are automatically updated in all the 4D methods, this doesn't happen in the 4D HTML tags. Another advantage is that if you have complex business rules that need to be respected, sometimes it is just easier to do it in 4D directly.

These are both very valid approaches to building a page that presents a list of products based on the current selection. Each approach has advantages and disadvantages. Personally, I utilize both approaches depending on what I need out of that page. The more code that is in the HTML files, then the harder it is to debug. This can be a big issue if you have a non-trivial HTML page.

One approach that I have taken that I like is to build a set of utility routines that I then employ in the HTML pages. Routines let me perform a query, query in selection, and sort. These have been very helpful. This helps me to have

some code in 4D but leave some of the code in the HTML pages. If I have a situation where performance is important, then I put more code into 4D.

MAXIMIZING PERFORMANCE

Depending on your needs for, you can utilize one or more of these ideas for improving performance of your 4D Web application.

Compile your application

This will always have an impact on performance. The more 4D code your web application is dependent on, then the greater the impact that compiling the application will have.

Apache/4D WebSTAR

An easy way to increase performance is to offload the serving of graphics, CSS style sheets and JavaScript files to a different web server. Let 4D do what it is good at and let a dedicated web server do what it is good at.

Another easy way to increase performance is to offload the SSL processing to a web server that sits in front of your web application. Apache has the ability to act as a proxy server for another server that sits behind it. I think that 4D WebSTAR is capable of this but I do not have direct experience.

These two items are particularly effective when the web server is on different hard ware as well.

Format your HTML – follow indenting rules

Formatting your html file (i.e. indenting properly) increases your performance since you will be able to read the files easier and make fewer mistakes. The chances are quite good that you will spend most of your time debugging and making changes. Anything that you can do to make that process easier is worth it, within reason of course.

Design Based on Performance Expectations

Design your site based on the expected load. A simple site that no one will see should be designed differently than a site that will have thousands of transactions per hour.

Design Based on Data Requirements

Design your site based on the amount of data that you will need to process to present the web pages. The more data that must be processed, then more thought is required to get decent performance.

Normalizing your Database

De-Normalizing your database can have a huge impact on performance for a web application, negative and positive. Give this type of decision some thought and be aware of you data and performance requirements.

Custom URL Caching

If you are using custom URLs then that bypasses any caching that 4D has built-in. Consider implementing your own caching. It is not too difficult to do and can have a significant improvement on your performance. There is a cost, that is more RAM is consumed and there is the issue of when do you refresh the cache.

Pre-Build pages

Some pages, or components, can be pre-built and saved as a file. Then served as needed without having to generate them. An example would be a list of active products, or the “hot deals” for a web store. That will not change often and there is, potentially, a big processing cost to produce. It will add complexity since you will need to deal with the issue of when to refresh the page.

SESSION MANAGMENT

Session management can be handled in one of two ways; through records in the database or by variables in memory. Each has it pros and cons. If you manage your sessions via records in the database, then you could have multiple 4D Clients acting as web servers and be able to have your users use any client and be able keep their session accurate.

Personally, I like to use ObjectTools to create a session object that then contains any persistent data that I want to keep for the user's session. As the user makes request, I identify which session object is theirs and process the user's request. If I need to, I can store the session objects as BLOBs into records. I can even have it so that when the 4D web server is quit, I can write those session objects to disk and restore then when the 4D web server is launched. This means that I can quit 4D, apply an update and restart it and the outside users will not know unless they requested a page during that 1-2 min down time.

If you decide to utilize ObjectTools then I strongly suggest that you invest some time in incorporating an ObjectTools object viewer/editor. You will need to be able to look at your objects and see what is inside them. The better and more sophisticated your view/editor is, the easier it will be to debug your application.

SUPPLIED WEB SHELL CODE

I have included, on the 4D Summit CD, the web shell that I use as a starting point for the web applications that I develop. I have implemented a number of the performance enhancements that I have talked about here in the shell.

Originally, when I started researching 4D's web serving capabilities, I did a survey of all the available tech sheets, tips and sample code that I could find on the web. Most of the examples where too basic to be of any value. One example that was very useful was 4D's WebMail v1.0 source code. I am very happy that 4D decided to release this to the 4D developer community. It was a complete solution that accomplished something of value. It was amazing and really helped me get over the initial hump of using 4D's non-contextual web pages. That was a number of years ago and I have done some heavy modifications to it and this is what I am sharing with you now. I hope that you are able to get some value from what I have gone through and potentially take it further than I have.

Why am I releasing so much code? It is not hard to build high-quality 4D web applications, but it does require more knowledge. This shell will give you a solid foundation to start from. You won't need to understand the entire shell to use it, you just need to understand how to add to it. This is good for 4D and good for us. We all want to have successful solutions. It helps us all.

APPENDIX – DANI'S BIO – A BIT OF SHAMELESS SELF PROMOTION ;-)

Dani is a lead developer and development supervisor for Jonoke's development department as well as providing continued expert consulting/contracting to other 4D development organizations. He has over 15 years experience building specialized commercial database applications for vertical markets using 4D and 4D on the web. Specializing in enhancing products through improved usability, Dani fully understanding how the users view software and what their priorities are. He is always focused on the "big picture" and delivers ever improving products. He has developed many innovative solutions such as custom XML SAX parser, 4D based web shell, auto updating of 4D applications, data synchronization using TCP/XML, on-line help using SOAP, drug/allergy interaction checker using SOAP, integrating independent EHR using TCP/HTML/HTTP/XML/HL7 and many others.

His broad technical foundation started with two Bachelor of Science degrees from the University of Alberta, one in Mathematics and the other in Computing Science. He has worked every aspect of the software development process -- from requirements gathering, design, development, mentoring, training, direct customer support, to project and staff management. He prides himself on his creativity and ability to diagnose and problem solve. Through out his entire career, he has worked closely with the end user and is able to communicate clearly in a way that is atypical of software developers.