Performance Test: 4D Web Server and Static Documents

By Tom Fitch, Technical Services Team Member, 4D Inc.

Technical Note 09-25

Table of Contents

Table of Contents 2
Abstract 3
Introduction 3
Using the 4D Web Server 3
Web Server Test Cases 5
  Using the HTML Root Folder 5
  Programmatically Managing Content 5
Using and Exploring the Sample Database 6
  Handling Web Requests in On Web Connection 7
  SEND HTML BLOB 7
  SEND HTML FILE 8
  SEND HTTP RAW DATA 9
Data: Results and Analysis 9
  Text Only HTML Tests 10
  Large Complex File Tests 10
  Small Complex File Tests 11
Conclusion 11

Abstract

This Technical Note analyzes the 4D Web Server's performance when using different methods to serve static documents. The sample database included uses 4D's Web Server to serve content without code and uses various tactics within the On Web Connection database method to serve the same pages via 4D code. The performance testing is done using third party software and all results are included in the Technical Note. The goal of this document is not to say which method of serving content is functionally best, as there are more considerations at play than just performance when serving static content. Rather, the goal is to give 4D developers information on how to move their databases to the web and achieve the fastest performance.

Introduction

There are a few different ways the 4D Web Server can be used to serve static content. One large distinction is the use of the On Web Connection database method. 4D can serve pages at the HTML Root automatically, completely bypassing the On Web Connection method. If the On Web Connection method is used then the developer can programmatically manage what pages are displayed. This Technical Note analyzes the performance difference between those two methods with some different ways of approaching the On Web Connection database method. The Technical Note describes the test cases used (included in the sample database provided) and then gives the data found and provides analysis of the results.

Using the 4D Web Server

You can turn on the 4D Web Server either via the Run menu in Design Mode and "Start Web Server" or by choosing to have it automatically start when your database starts, via the database Web Preferences. You can also programmatically start it with the START WEB SERVER command. Once the web server is running, 4D handles web requests in a few different ways; for this Technical Note we look at whether the 4D Web Server uses the On Web Connection database method or not.

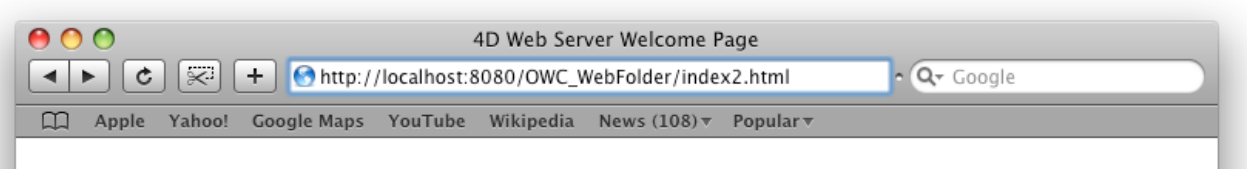
When a web request is made, 4D checks three things:

- Is the 4D Web Server in Contextual Mode?
- Does the URL begin with "/4DCGI"?
- Is the URL invalid? (Is the document referenced missing from the HTML Root folder?)

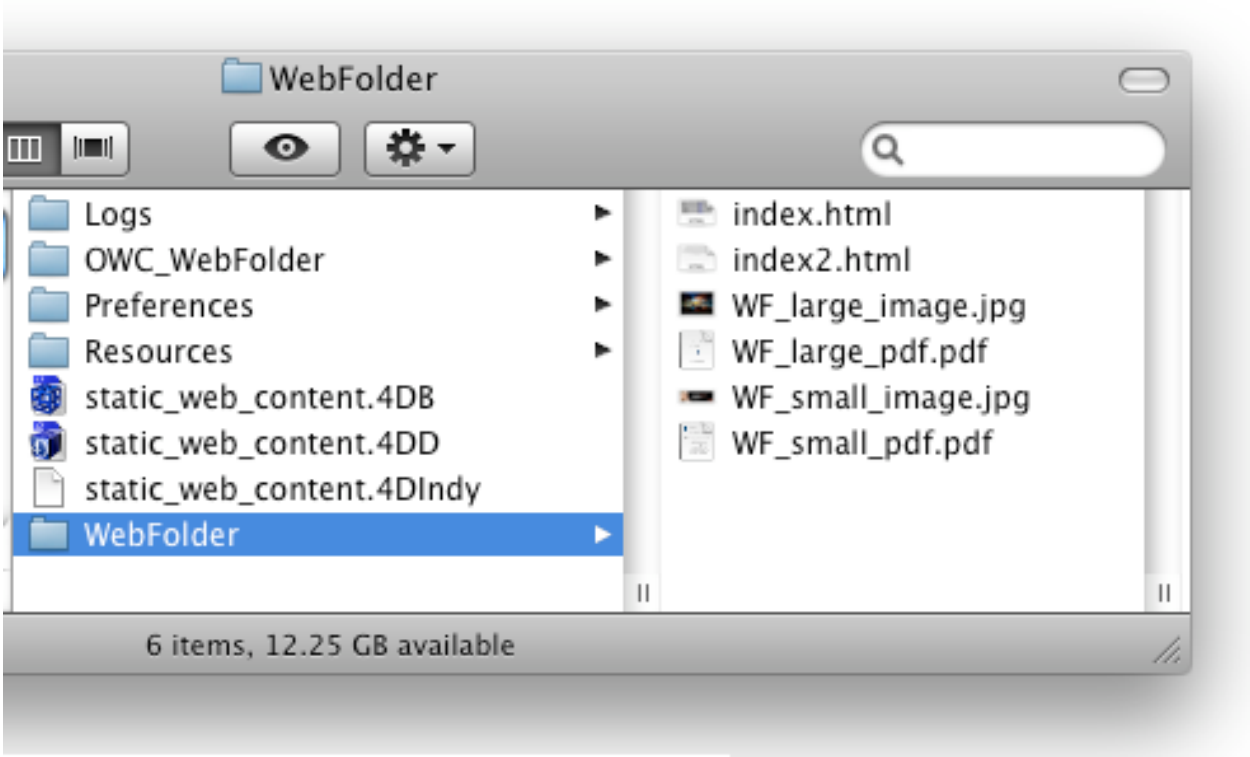
If any of these are true then the 4D Web Server calls the On Web Connection database method to handle the connection. Otherwise the 4D Web Server serves the document stored at the HTML Root and bypasses the On Web Connection folder. For the purposes of this Technical Note when we want the On Web

Connection database method to be called we force it by attempting to access an invalid URL.

Here is an example, from the sample database, of how the On Web Connection database method is called. In the sample database, we use the default HTML Root of "WebFolder". In this case the URL "OWC_WebFolder/index2.html" is put into Safari's address bar:



As you can see from the following screenshot, the document "OWC_WebFolder/index2.html" does not exist at the HTML Root for this database:



In this case, due to the invalid URL, the On Web Connection database method is called. Had the URL instead been: http://localhost:8080/index2.html, the file would have been served by the 4D Web Server without accessing the On Web Connection database method. This is because "index2.html" exists in the HTML Root folder ("WebFolder") but "/OWC_Webfolder/index2.html" does not.

Web Server Test Cases

As mentioned above, there are two main categories of test cases that are used for this study:

- Serving static content stored at the HTML Root without database interaction.
- Serving static content via the On Web Connection database method by referencing documents that do not exist at the HTML Root.

Using the HTML Root Folder

The HTML Root is selected from the 4D Web Preferences. By default it is named "WebFolder" and located at the database level next to the structure. The test cases using the HTML Root Folder are quite simple. Simply place the desired test document in the folder and create a link to it on the homepage. The cases that are tested are:

- Plain text document (an HTML page)
- Small JPEG image
- Large JPEG image
- Small PDF file
- Large PDF file

In this case the small image is 39kb and the large image is 4MB. The small PDF is 177kb and the large PDF is 8.3MB. The same documents are used later for different tests.

Programmatically Managing Content

In this test, all calls to the On Web Connection database method are prompted by references to documents that do not exist at the sample database's HTML Root. Once the On Web Connection database method has been invoked, three different ways of handling web requests are used:

- SEND HTML BLOB
- SEND HTML FILE
- SEND HTTP RAW DATA

The SEND HTML BLOB technique is used with all the types of files discussed in the "Using the HTML Root Folder" section:

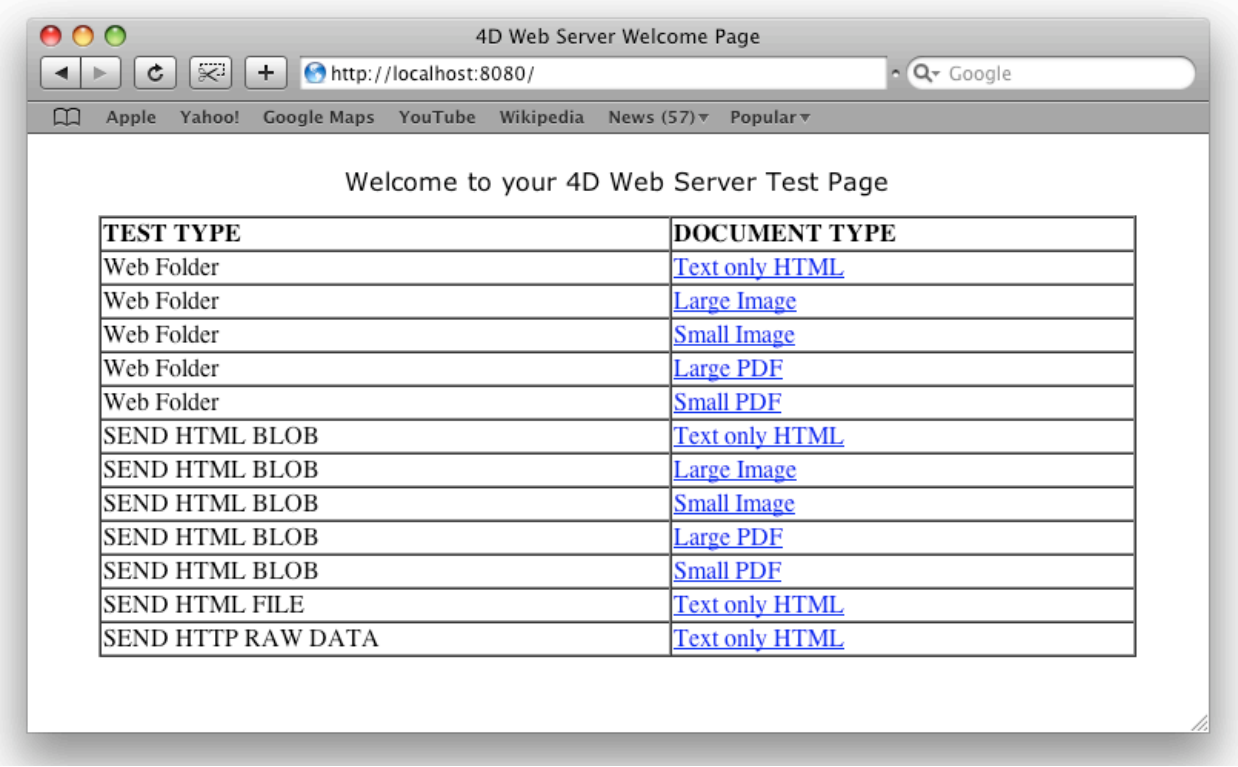
- Plain text document (an HTML page)
- Small JPEG image
- Large JPEG image
- Small PDF file
- Large PDF file

The SEND HTML FILE and SEND HTTP RAW DATA methods are only used with plain text document test cases.

Using and Exploring the Sample Database

Because the sample database is only designed to be used as a web server, all you need to do to use it is open it with 4D v11 SQL and connect to the local host via your browser. The web server uses port 8080 by default, though that can be changed via the database Preferences.

The homepage appears as shown here:



Each type of test case discussed above in the test cases section can be accessed via this homepage. To return to the homepage from a test case simply use the back button in your browser.

The links for the non "Web Folder" (the default HTML Root) test cases are to nonexistent documents. These links invoke the On Web Connection database method of the sample database. Documents are served either from the data in the database or from another folder (the OWC_WebFolder) at the database level in the case of the SEND HTML FILE test.

When exploring the sample database the method to look at is the On Web Connection database method. It shows three different ways to serve documents, as discussed above:

- SEND HTML BLOB
- SEND HTML FILE
- SEND HTTP RAW DATA

Before looking at these different techniques of responding to web requests, we need to look into how to handle and parse web requests.

Handling Web Requests in On Web Connection

The first step to understanding how the sample database works is to understand what we are doing with web requests once they have been passed to the On Web Connection database method. A Case statement is used therein as the decision tree, based on the $url process variable. That variable holds the $1 parameter of the On Web Connection database method, which is the URL that leads to the method being called. Based on what URL was passed, we should use one of the above listed methods of sending a document to the 4D Web Server.

The case statement follows:

```
Case of
    : ($url="@OWC_blob@")
          ` Query the database for a record
          ` Once you find the record send its BLOB to the Web Server
          ` In this case use SEND HTML BLOB
    : ($url="/OWC_Web@")
          ` Fix the path so that it leads to the OWC_WebFolder
          ` Send that file to the Web Server with SEND HTML FILE
    : ($url="@OWC_raw@")
          ` Query the database for a record
          ` Once you find the record use BLOB TO TEXT
          ` Send that BLOB to the Web Server with SEND HTTP RAW DATA
End case
```

A sample URL used in this Technical Note would be:

"http://localhost:8080/OWC_blob_large_image.jpg"

That would then be transferred to the On Web Connection database method's $1 parameter as "/OWC_blob_large_image.jpg" which would then be handled by our Case statement under the ": ($url="@OWC_blob@")" case. And then the SEND HTML BLOB command would be used. This is just one example of how this works in the sample database.

SEND HTML BLOB

This command is used to send data stored as a BLOB to the 4D Web Server. In this database it is used to send a BLOB stored in the database as the static content. This is simply an optimization; another way to look at this situation would be to dynamically load the document to be sent to the 4D Web Server from disk into a BLOB and then send it. The developer could use DOCUMENT TO

BLOB with the path to the desired document, loading it into a BLOB, and then use SEND HTML BLOB to serve the document.

In the sample database, first a query is run to find the correct record in the database; that part is shown below:

```
QUERY([files];[files]ID=$url)
```

Each document is saved as a BLOB in the sample database, as the "file" field in the "files" table, with a text field, "ID", used as its unique ID. Each ID is actually the URL that is passed as $1 of the On Web Connection database method. Therefore that parameter ($url in the above example) can be used to query the "file" table. Once that is done the BLOB can be passed to the 4D Web Server. The code used in the sample database to actually send the HTML BLOB follows:

```
If (Records in selection([files])=1)
   $my_blob:=[files]file
   SEND HTML BLOB($my_blob;$tail;False)
   CLEAR VARIABLE($my_blob)
   ` handle errors if necessary
End if
```

The important code is here in the SEND HTML BLOB command; it is simply used to send the blob that was found via QUERY to the Web Server. The data type is parsed from the file requested and passed to the command in the $tail variable. The "False" boolean value at the end of the command simply keeps the Web Server in its current mode.

This same code is used to serve the PDF's, JPEG's, and plain text HTML files in the test cases.

SEND HTML FILE

This command is only used to serve an HTML file. The file is stored in the OWC_WebFolder at the database level. It is served using the following code:

```
$path:=Get 4D folder(Database Folder )+$url
$path:=Fix_Path($path)
If ((Test path name($path)=1))
   SEND HTML FILE($path)
   ` handle errors if necessary
End if
```

Two things to note from this code are the $url variable and the Fix_Path method. The $url variable is assigned the value of $1 (in the case of the sample database it is "/OWC_WebFolder/index2.html") from the On Web Connection database method. This means that it holds the URL that is being requested by the browser. We simply add that URL on to the end of the database folder to find the HTML file for SEND HTML FILE in the OWC_WebFolder. The Fix_Path project method takes the HTML style URL from $url and parses it so that the path is appropriate for Mac or Windows, depending on which OS you are running the database. As we saw above the HTML style would be:

/OWC_WebFolder/index2.html

The style needed for Windows is:

\OWC_WebFolder\index2.html

And for Mac the path style would be:

:OWC_WebFolder:index2.html

SEND HTTP RAW DATA

This command is also used only to serve plain text HTML. The HTML is passed as raw data to the web server and stored in the database. The database is queried for the BLOB holding the data and then it is passed to the SEND HTTP RAW DATA command. The query works in exactly the same manner as described above under the SEND HTML BLOB heading.

In general the BLOB is sent in a very similar way to how the SEND HTML BLOB command works, but instead of sending a full document, the SEND HTTP RAW DATA command can be used to send chunks of text. In this case it is used to send the text of the "index2.html" document at the HTML root. The file's text has been saved in a BLOB in the database. The following code shows how it is done in the sample database's On Web Connection database method:

```
If (Records in selection([files])=1)
   $my_blob:=[files]file
   SEND HTTP RAW DATA($my_blob)
   CLEAR VARIABLE($my_blob)
   ` handle errors if necessary
End if
```

In the same manner as the SEND HTML BLOB case, the QUERY to find the given record of the "files" table is left out of this sample code.

Data: Results and Analysis

The test cases were set up to use ten virtual users accessing the web server repeatedly over a two minute period. All "think times" for users were set to a default of 10ms. The following data is the result of those tests.

Note: *All data for these test cases was obtained using NeoLoad from NeoTys. For more information on this 3rd party tool, visit www.neotys.com.*

Let's look at the results on a case-by-case basis to more easily analyze the differences:

Text Only HTML Tests

Test Type	Doc. Type	Page Response Data			Request Response Data		
		Min Time (s)	Avg. Time (s)	Max Time (s)	Min Time (s)	Avg. Time (s)	Max Time (s)
Web Folder	Text	0.033	0.106	0.115	0.098	0.108	0.692
SEND HTML BLOB	Text	0.045	0.108	0.113	0.103	0.109	0.416
SEND HTML FILE	Text	0.051	0.109	0.120	0.103	0.110	0.469
SEND HTTP RAW DATA	Text	0.092	0.107	0.116	0.092	0.108	0.644

Based on the Text Only test cases above, it is apparent that the method of sending data does not matter if it is a simple small plain text HTML file. The speed of all methods is similar in that case. It ranges from an average of .106 to .110 seconds in all cases.

Large Complex File Tests

Test Type	Doc. Type	Page Response Data			Request Response Data		
		Min Time (s)	Avg. Time (s)	Max Time (s)	Min Time (s)	Avg. Time (s)	Max Time (s)
Web Folder	Large Image	0.189	0.199	0.296	0.189	0.199	0.324
SEND HTML BLOB	Large Image	1.195	2.840	6.191	1.195	2.926	6.457
Web Folder	Large PDF	0.300	0.329	0.401	0.300	0.329	0.391
SEND HTML BLOB	Large PDF	4.924	8.226	17.253	2.166	4.221	28.185

The more interesting situations are with larger files. Looking at the above data, it is apparent that for larger files there is a noticeable difference between allowing the 4D Web Server to handle serving content and managing it programmatically within the On Web Connection database method. The large PDF and large image files show considerably slower response times with the SEND HTML BLOB command than they do when the file is stored at the HTML Root and the 4D Web Server handles loading and serving the image. For example the Large Image takes over ten times as long

to on average with SEND HTML BLOB and the Large PDF takes over twenty times as long. Interestingly, as noted at the beginning of this document, the Large PDF file is 8.3MB and the Large Image is 4MB. One corollary that can be drawn here is that the larger a file, the more benefit you get from using the HTML Root as and letting 4D Web Server handle the documents. We can look at the small complex file tests next to see if they agree with that statement.

Small Complex File Tests

Test Type	Doc. Type	Page Response Data			Request Response Data		
		Min Time (s)	Avg. Time (s)	Max Time (s)	Min Time (s)	Avg. Time (s)	Max Time (s)
Web Folder	Small Image	0.050	0.107	0.116	0.096	0.108	0.475
SEND HTML BLOB	Small Image	0.087	0.137	0.143	0.130	0.137	0.150
Web Folder	Small PDF	0.616	0.782	0.837	0.726	0.783	0.911
SEND HTML BLOB	Small PDF	0.523	0.558	0.603	0.525	0.559	0.668

With smaller complex data such as the small PDF and small image files the differences are minimal. In this case it becomes apparent that the larger the document you are serving, the more important it is to allow 4D to handle the Web Server functionality.

Conclusion

The main conclusion that can be drawn from this is that it does matter how you serve your documents to the web. The 4D Web Server sometimes functions faster and better than managing content via the On Web Connection database method. Of course there are trade-offs in any situation as the developer has more control when using the On Web Connection database method to manage content. Use this data to help make the decision about when to manage the content yourself and when to let 4D handle it.