Aspose.Words

Paperless Proof-of-Delivery for a Manufacturing Company

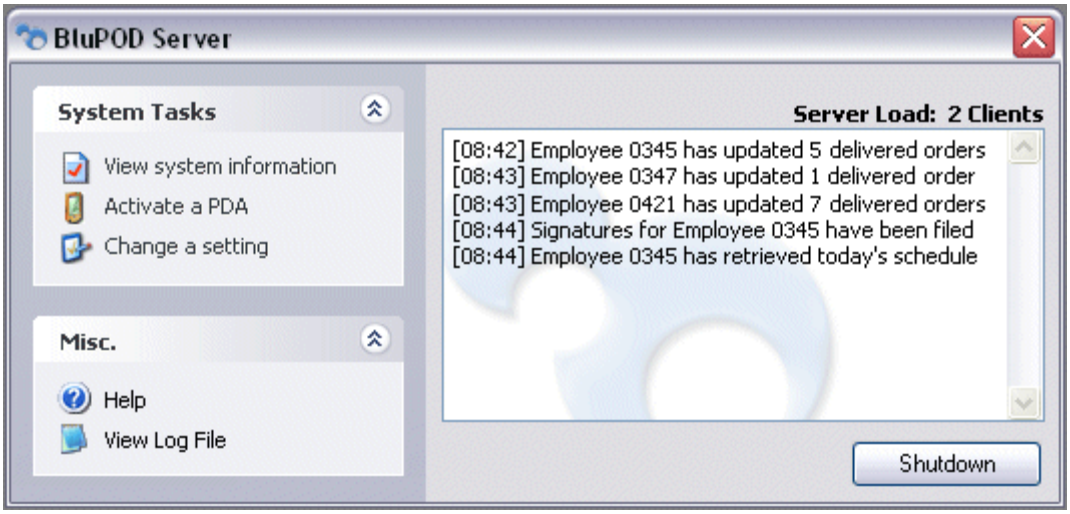
29th November 2005, By Graham PLUMB

Background

I am a software developer currently working for a company in the Midlands (UK) that specializes in Manufacturing Software Solutions. Our flagship product is a fully traceable MRP system which can be applied to almost any business situation and tailored (both by the user as well as by us) to fit any level of a company's day to day operations. Whether it is entering new customer details or tracing where a batch of material originated on the shop floor, our software provides complete transparency and traceability during the manufacturing process.

Scenario

One of our clients had a requirement for a proof-of-delivery system to tie in with our core product. The process was simple enough, use a PDA to collect a user's signature at the time of delivery (in a similar fashion to the bigger distribution companies) and then merge them into a Word Document template for subsequent storage. In effect, the solution was to harness a paperless method of storing delivery acknowledgements.



Solution

The solution was to be coded using version 1.1 of the .NET framework and C#. To actually perform the merge of the signature bitmap with a template Word Document, there was a choice. Use the Microsoft Office Primary Interop Assemblies (PIA) or use Aspose.Words.

Pros and Cons

The PIA initially seemed like a good approach, it is free, made by Microsoft and is almost guaranteed to work as they wrote the software which actually creates .DOC files. How could an outside company like Aspose (which charges for its components) contend with that?

Well, as it turned out, quite easily!

One of the biggest advantages is that the Aspose.Words component is entirely managed code (written in C#) from the ground up. Unlike the Office PIA (which is just a generated wrapper), it

comes with complete documentation and has all the benefits of garbage collection, removing the complexities of pointers, memory allocation and correctly casting objects (unlike the PIA!). For ease of use alone, this component is streets ahead of the Office PIAs.

However, the crucial factor for any component is performance. In the given scenario, several users may be synchronizing data concurrently and so the merge had to be performed quickly so that users were not waiting around before obtaining their delivery schedules.

To test this, I wrote 2 programs (one using the PIAs and one using Aspose.Words) that performed the merge based on a particular bookmark being present in the document template. The resulting document was then saved to a new location. The programs were run 50 times each and the average times for the merge were taken.

Bookmark Find and Image Insertion Test Results

Microsoft Office Primary Interop Assemblies	5755.2 ms (nearly 6 secs)
Aspose.Words Component	709.4 ms

This works out to be just over 8 times faster! The resulting documents were opened in Microsoft Office (2003) and looked identical visually, but there was one difference. The document produced by Aspose.Words was actually smaller than the original template file (created in Word 2003)!

Comparison of the Generated .DOC File Sizes

Original File Size (created in Word 2003)	23.5 k
Microsoft Office Primary Interop Assemblies	34 k (over 30% bigger)
Aspose.Words Component	20.5 k

What's even more impressive is that the code only took 10 lines of code to create using Aspose.Words as opposed to 17 using the Office PIAs.

Since the Aspose.Words component is written from scratch for .NET, it can bring memory savings as the garbage collector can transparently amalgamate related structures in memory, thus using less of it. This is more secure than the Microsoft approach (which actually loads an entire instance of Word in the background when the constructor is called) and more suitable for web servers. The security features of the .NET runtime also help to ensure that the Aspose.Words component is a more secure approach to use in both desktop and server environments.

Another advantage is that the component is entirely independent of the Microsoft Office suite. This means that documents can be created without having Microsoft Word (or any of the Office components) installed. This is a bigger coo than it first seems as corporate licensing for Microsoft products can be very expensive – especially if it is just to run a small program on a computer. By removing this dependency, the license cost can be circumvented for systems that just need to produce .DOC files without any direct user interaction.

Conclusion

In the original scenario, there were potentially lots of drivers requiring a responsive mechanism of sending signatures to be merged into a word document.

This would simply not have been possible using the Office PIAs as each user would have to either synchronize their data sequentially or the company would have had to buy a supercomputer's amount of memory to contain multiple instances of Microsoft Word (not to mention the licensing cost of Word itself!).

Because of the Aspose.Words component, we were able to offer our client a license-free solution that operated in a fast and effective manner. The end result was that the end user only has to wait seconds for a days' worth of deliveries to be synchronized with our MRP system. This happens concurrently, reducing the amount of time drivers have to wait before making deliveries. This in itself has lead to further savings as the drivers are now freed from the red-tape that used to plague their delivery schedules and can just focus on getting goods to their destination on time.

Aspose.Words has enabled the paperless office to become a reality for our client. This product comes highly recommended and is a must have for any developer's toolbox. There simply is no real alternative.