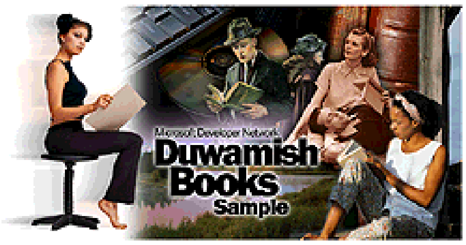
by Kevin Ledley



Duwamish Books, Phase 4: Welcome to the Web

Last Updated: October 8, 1999

> **Editor's Note:** Service Pack 1 for Phase 4 of the Duwamish Books sample application is now available. The purpose of this service pack upgrade was to fix several bugs, some of which were related to running the sample on Windows 2000. See the sidebar for more information on known issues, as well as how to download Service Pack 1 and get the list of bugs the service pack fixes.

Phase 4 of the Duwamish Books sample is now complete -- and ready for you to download. The download includes all the source code as well as a Setup program that allows you to install the database, COM components, and ASP scripts across a number of servers. Duwamish Books, Phase 4 brings our fictitious bookstore onto the Web as a Windows DNA application. Instead of the Win32 clients used by store employees in the earlier phases, our Web store implements three different client applications that allow for multiple browser types: standard HTML 3.2 browsers, Internet Explorer 4, and Internet Explorer 5. From a convenient start page in the application, you can choose to try out any of the three types. Remember, however, that you'll need Internet Explorer 5 installed to be able to run all three.

Architectural Highlights

Since Phase 2 of the sample, we have been gradually replacing, layer by layer, overlapping code from individual client applications with shared code in COM components. We continue

archived columns

News

October 8, 1999

Read about **known issues** with Duwamish Books, Phase 4.

October 1, 1999

Service Pack 1, with **bug fixes** for Duwamish Books, Phase 4 is now available. To download Service Pack 1, go to the Duwamish Books, Phase 4 **download page**.

The complete Duwamish Books documentation as an HTML Help (.chm) file is now available here and in the updated sample download.

this process with Phase 4, breaking out the workflow logic. However, unlike previous phases, where the logic excised from the client was transferred to a single COM component, we have implemented a more sophisticated combination of a COM component and ASP technology running on Internet Information Server (IIS). The COM object uses XML to encapsulate data, providing scalability and client independence.

The Workflow

Figure 1 shows the architecture of the sample, with special attention paid to the Workflow Layer. Here's roughly how it works: The Workflow Layer (WFL) component converts ADO Recordsets it receives from the Business Logic Layer (BLL) component into XML data. In the case of client type 1, the XML is transformed to HTML on the server. In the case of client type 3, the XML is passed to an Internet Explorer 5 client and parsed there. Client type 2 is a little trickier. We use a data source object on the client to get XML from the server and populate a DHTML table. In client types 2 and 3, the XML data sent to the client is the same.

For more details about the client types, read Paul Johns' article "Three Approaches for Three Client Types." Steve Kirk can give you the skinny on all the complexities of the Phase 4 Workflow Layer in "Workflow Design for a Web Application."

As you can see from Figure 1, although we consider Workflow to be an application layer, all the logic is not necessarily implemented on the same tier. In client types 2 and 3, significant workflow logic runs on the presentation, or client, tier.

Client 1 for HTML 3.2 Browsers

Client 2 for Internet Explorer 4.0

Data source object (DSO): data binding and client-side XML data cache

Client 3 for Internet Explorer 5

Formatting: XML+XSL=HTML

ASP Scripting

Stores HTML data in

Stores XML and HTML

Download the Duwamish Books HTML Help file.

August 25, 1999

Want to discuss the Duwamish Books sample with other developers? Stop by the Duwamish Books newsgroup at: **news://msnews.microsoft.com/ microsoft.public.msdn.duwamish** We'll be monitoring the activity (but won't be offering traditional tech support for the sample).

The Phases of Duwamish Books

Phase 1
A classic desktop app

Phase 2
A client/server app

Phase 3
A three-tier app

Phase 3.5
A distributed three-tier app

Phase 4
The Web store

Cache

data in Cache
Stores XML data in
Cache

Formating:
XML+XSL=HTML

Workflow Component

BLL Component

DAL Component

Legend
Presentation Layer
Workflow Layer
Business Logic Layer
Data Layer

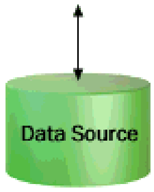


Figure 1. The Phase 4 architecture (click a link for more details)

The Cache

In some ways, the Cache component and caching strategy is a remarkable sample in its own right. This very simple concept gave us a twenty-fold increase in performance against the database. For results of our testing, see Bernadette Bly’s "Performance Testing a Scalable Application."

While this concept is rather simple, we think the design and implementation is unique and very effective. Because it is relatively expensive, performance-wise, to convert the data to XML, we store the most requested pages as HTML or XML (depending on the client type) in the cache on the Web server. So, instead of going all the way back to the database, a good percentage of the requests can be filled from cached pages on the Web server.

Additionally, we save even more with client type 1 by caching HTML, thereby reducing the amount of the very expensive XML-to-HTML conversion that needs to be done. Robert Coleridge, who developed the Cache component in Visual C++, explains the design and implementation in "Creating a Page Cache Object." All the Cache interfaces are documented in the

Cache API reference.

The Business Logic Layer

Of course, this is not the first time the logical layers and physical tiers have not completely meshed. Server-side business logic, as first implemented in Phase 3 and Phase 3.5, is a combination of the BLL component, acting as a client of the Data Access Layer (DAL) component, and SQL stored procedures running on the database server.

In Phase 4, we still use that combination. However, the BLL has been modified to serve the updated Workflow Layer. In order to let clients from previous phases use the same component, we created a strategy for special migration components to act as intermediaries. However, we left the actual coding of these components as an exercise for the reader. Steve Kirk explains the modifications and migration strategy in detail in his article, "Migrating a Business Logic Component to a Web Application."

The Data Access Layer

The DAL component remains unchanged. You can read the articles we've written in the past about its design: "Duwamish Books Data Access Layer, Phase 3" and "Moving a Data Access Layer into Microsoft Transaction Server."

However, the database itself has been souped up-normalized and optimized for search.

Database Optimization and Data Access

We set our new database guru, David Willson, loose on the legacy database from Phase 3.5 and he made some great improvements. First, he normalized it by reducing redundancy, implementing lookup tables, and decomposing problematic tables. David discusses the process, as well as benefits and possible consequences (not all good) of normalizing a database in "Normalizing the Duwamish Books, Phase 4 Database."

David then optimized the database for different search methods. Because the application is search-intensive, our strategy is to provide a progressive choice of searches. We encourage users to try less expensive (more scalable) search methods before resorting to more expensive methods. In "Building a Search Engine for the Duwamish Books, Phase 4 Database," David explains the design of the keyword tables and the English Query functionality. If you just want to see the big picture of the database design, you can go straight to the Phase 4 database schema.

Performance Testing

In order to test performance of an application, you first need to set some goals to direct your testing efforts. In a real e-commerce scenario, a company may want to first establish its broader business goals, such as gross annual profit, market share, or number of concurrent users, and then calculate performance goals for the Web application based on the business goal.

In "Setting Performance Goals for Duwamish Books, Phase 4," David Willson offers some insight into how to determine performance goals for a Web application. David takes a hypothetical business goal of dollars per year and turns it into a performance goal of requests per minute. As the article explains, a good, scalable Web application such as Duwamish Books should be able to meet almost any business goal by scaling to meet the projected performance requirements.

Once you have your performance goals, it’s time to test your application against them. You can read about how we tested Duwamish and see the results in Bernadette Bly’s article "Performance Testing a Scalable Application." In all modesty, I have to say that the Duwamish Books application did very well. On just two PIII-500 machines, Duwamish Books surpassed the performance goals we had set for it. We think that our Mom and Pop who started Duwamish Books in Phase 1 would be proud.

Payments

One of the obvious requirements for an e-commerce application is to receive payment for the product. Because we were focused on architectural issues, we didn’t develop the credit card authorization piece of the application. However, Robert Carter, in "Collecting Payment with an Online Business," discusses a few different collection options, including the use of an independent payment service provider. In fact, we’ve asked a few third-party providers to show how their solution could be implemented with Duwamish Books. Watch this space in the future for information about the progress of these efforts.

Conclusion

Phase 4 marks a huge step forward from the previous phases.

Most obviously, it radically changes the Duwamish Books business model to a Web-based store.

From an architectural standpoint, breaking out the workflow logic increases scalability and (especially through use of the cache) enables the performance necessary to serve the requests and concurrent users that a successful Web-based store requires. To learn more about how we did all this, read the articles, download the sample, and examine the application and the source. We've also created a newsgroup for the sample at news://msnews.microsoft.com/ microsoft.public.msdn.duwamish We'll be monitoring the activity (but won't be offering traditional tech support for the sample).

As usual, feel free to leverage as much of the code as you want in your own projects. One warning though: Before running Setup, be sure to read the Setup instructions carefully. The computers and applications that you install Duwamish Books on (such as SQL Server, IIS, and MTS) require a fair bit of complex configuration.

Back to top

Did you find this material useful? Gripes? Compliments? Suggestions for other articles? Write us!