



**ASPIRE EXECUTIVE
ARCHITECTURE AND HIGH LEVEL FUNCTIONAL
DESCRIPTION**

OVERVIEW / WHITE PAPER

INTRODUCTION

This white paper outlines a new approach to introducing .NET development into a company. The use of Aspire Executive, a .NET based web infrastructure tool, offers companies the ability to rapidly and cost efficiently produce .NET web applications. This technology can be utilised in both out-sourced and in-house development teams.

ACTIVE SERVER PAGES (ASP) AND COMPONENT OBJECT MODEL (COM) DEVELOPMENT

Prior to the release of .NET, the mainstay of web application development on the Microsoft platform was ASP combined with the use of COM or COM Plus (COM+) components. These components could be written in Visual Basic (VB) or Visual C++ (VC) using the Active Template Library (ATL).

ASP was a hybrid of server-side and client-side technologies and a single ASP page could have a mixture of Hyper Text Mark-Up Language (HTML), ASP code and Javascript code. This provided the ability for lower-skilled developers to produce complex web sites at a lower cost. It did not however always lead to maintainable sites nor to secure or reliable web-sites.

To address the maintainability, security and reliability issues some companies adopted a design pattern of calling COM/COM+ components from ASP pages which solved many of these problems, but this raised the skill level required and the resulting costs.

ASP.NET DEVELOPMENT

On release of .NET, ASP.NET became available to the Microsoft development community and offered the ASP developer all the functionality previously only available to the COM/COM+ component developer. Whilst some ASP developers have moved smoothly to the new environment, many have struggled and often systems have been re-written with no additional functionality merely to ensure that the ASP developer can up-skill to .NET. The language independence of .NET has introduced the ASP community to C#, and where the type-less VB style of code was used before, the more rigid strongly typed C/Java style of coding has often been chosen over the more natural migration to VB.NET, simply for the reason that the developers felt that C# gave more 'credibility' to their CV.

For the last year in the USA, C# developers could earn up to 20% more than their VB.NET counter-parts, even though the latter are able to write systems which are every bit as complex or wide ranging as those in C#. At several conferences in the USA and UK, speakers asked, 'which is better C# or VB.NET?' Many answered C#, when pressed as to why, many admitted that the only reason was because as a C# developer you could earn more.

Ultimately the cost and expense of the move to .NET has been borne by companies, many of them unknowingly. Whilst undoubtedly the .NET development environment does provide the opportunity for quicker and cheaper development, many times it is the approach of the development team that renders this benefit null and void.

CHANGING THE WAY WE WORK

Whilst the development environment has moved on, often the approach to development has not. In order to cut down on development costs significantly the following needs to be achieved:

- The number of lines of code written needs to be reduced; and
- The time span for testing of the application needs to be reduced.

Due to the pressure of work, many companies simply do not have the time or money to invest in producing reusable code. It is more expensive to produce and must be inherently more reliable in order for the development teams to trust using it. However, in order to compete against off-shore competition, UK software companies must begin to adopt a Code Only If Necessary (COIN) approach. By adopting this approach, a lower skill set is required which in turn reduces overall project costs and increases the maintainability of the development due to the system being inherently simpler.

WEB APPLICATIONS

In essence all web applications are the same. The life cycle of a web request at the server follows the following work flow:

- Authenticate the Request
- Gather the Request Information
- Validate Input (Optional)
- Execute a Task or Tasks (Optional)
- Construct the Page to Render
- Issue the Response

Figure 1 shows this diagrammatically:

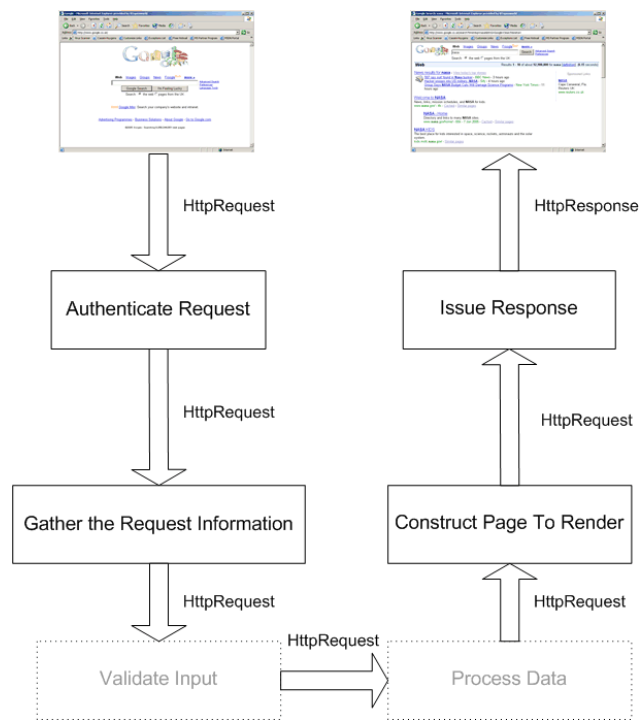


Figure 1 – Life Cycle of a Web Request

The workflow is the same be it Google, Amazon or NHBC. The major difference in the systems is what occurs in step 4 and in how the other steps are implemented, but all web applications have these steps. Whilst it has not been absolutely measured it is not unreasonable to assume that a project which does not utilise code reuse spends between 60%-80% of budget producing application infrastructure, from such basic elements as exception handling and data access to the more complex issues of validation, transactional control of tasks and implementation of authentication protocols. Even in development companies that do practice code reuse, often the usage is not as great as expected. A report recently indicated that code re-use industry wide could be as little as 4%.

The fact remains however that there are not that many different ways of completing these procedures and it seems inefficient to continue to keep re-writing these same infrastructures over and over again.

RIDGIAN WEB APPLICATIONS

For the past 4 years Ridgian has developed its web applications on the basis of a Canvas/Snap-In model where the Canvas is a container for other 'web-parts' called Snap-Ins. This design although similar to Portal design goes beyond that and allows for any sort of Web Application to be built in a modular basis.

Figure 2 shows the Canvas/Snap-In Model:

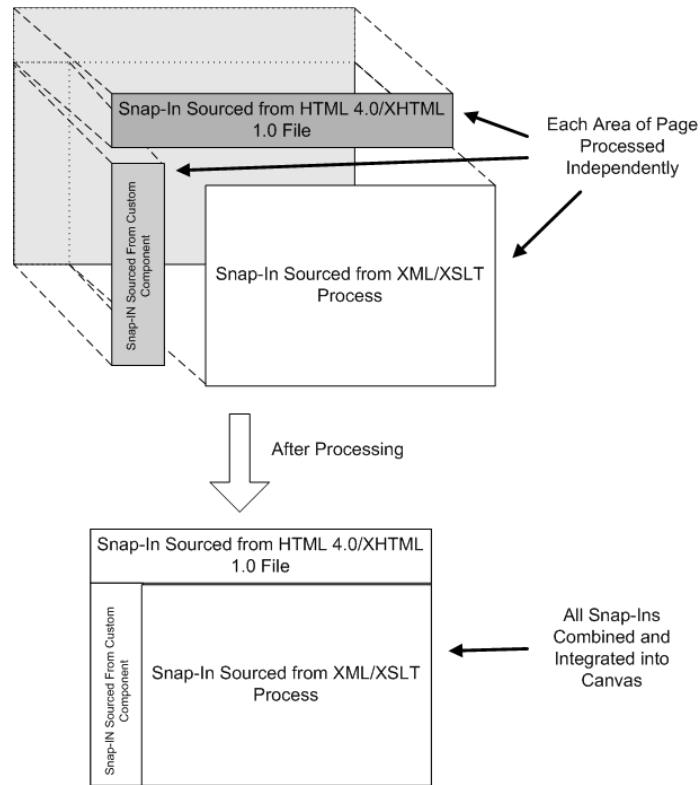


Figure 2 – Canvas/Snap-In Model

The Canvas dictates what Snap-Ins are present and the Snap-Ins can be sourced from static HTML files, XML/XSLT or Components written in COM or .NET. All the elements are merged into a single HTML document, but elements within the page can interact with the server independently via Web Service Calls, or the whole page can be refreshed using standard HTTP Postback.

ASPIRE EXECUTIVE PRINCIPLES

Aspire Executive has been written based on the following observations:

- All web-sites operate according to the same workflow pattern.
- The infrastructure processing of a Web Application can be fully configured.
- The majority of business processing of a Web Application can be configured.
- A minority of business processing of a Web Application cannot be configured and must be coded.

These observations have lead to the following design principles in Aspire Executive:

- All Aspire Executive Powered Web Applications can be based on a standard set of components.
- The infrastructure of Aspire Executive can be controlled through .config file settings and database stored configuration settings.
- The business logic of a Web Application based on Aspire Executive can often be executed in database stored procedures.
- Business logic which can either not be written in database stored procedures or is more easily written in code can be placed in custom components which interact with Aspire Executive via Interfaces.

CURRENT CAPABILITIES

As of release 1.2 Aspire Executive can act as the engine for an ASP.NET Web Application and provide the following configurable services:

- Authentication
 - Anonymous
 - Basic
 - Windows Integrated
 - Forms

- Authorisation
 - Bit- Mask Based Security

- HTML Parsing:
 - HTML 4.0 files
 - XHTML 1.0 files
 - XML/XSLT processing
 - Code Generated HTML

- Look-Up Data:
 - Hard-Coded
 - Static
 - Data driven

- Application Level Variable Persistence
 - Page Level State Persistence

- Partial Page Refresh
 - Update browser information withoutPostBack
 - Optimises bandwidth usage

- Codeless UI Driven Data Capture
 - Capture data in a database simply by adding standard input tags onto a HTML page.
 - Allows front end development to proceed independent of database development

- Data Validation
 - Simple syntax in HTML page enables complex server-side validation.

- Data Processing
 - Declarative tasks allows complex processing between pages
 - Multiple Task Chaining Available
 - Multiple Tasks Chaining System.EnterpriseServices Transactions Available

- Interface Based Extensibility
 - Extend data processing and page processing through Plug-In Architecture.

- Debugging
 - Switchable debug outputs available
 - Custom exceptions allow simple tracking of failures

- Tracing
 - Switchable performance tracing outputs available.

CORE COMPONENTS

An Aspire Executive powered Web Application is made of components which fall into a number of categories:

- Web Infrastructure
- Core Infrastructure
- Extensibility
- Utility
- Database
- Aspire Productivity Library (APL)

We will look at each of the categories in turn:

WEB INFRASTRUCTURE

The web infrastructure components are those components that interact directly with the Http Request that is sent to the Web Application. These components control the authentication and information collation from the Web Request. These components operate through a highly configurable design pattern and allow Aspire Executive to deal with the majority of different Web Application requirements.

Aspire Executive comes with a number of configuration files (.config), the majority of which are discussed later as they are component specific. However, a Web Application powered by Aspire Executive also requires a configuration file called 'AspireExecutive.config' which contains start-up information. Aspire Executive runs under ASP.NET and the IIS application under which it runs has a 'web.config' file which must also be configured to enable Aspire Executive.

CORE INFRASTRUCTURE

The Core Infrastructure components are those components which deal with navigation, validation, data processing control and construction of the response. Like the Web Infrastructure components these components operate through the same highly configurable design pattern.

EXTENSIBILITY

The extensibility components allow developers to add additional custom components or to add to existing components where configuration of the infrastructure components fails to meet the Web Application requirements.

UTILITY

The utility components are those components that provide common support to the infrastructure and extensibility components. These components provide interface definitions, configuration, data access and function libraries.

DATABASE

An Aspire Executive installation requires a single SQL Server 2000 or MSDE 2000 Server installation to operate. The SQL Server must contain a minimum of 2 databases. One must be named AspireExecutive and has a predetermined, static schema. The second and subsequent databases have a minimum schema requirement, which can be added to if necessary. The AspireExecutive database controls the Aspire Executive installations for a specified machine or farm. The second and additional databases are application specific and contain the configuration settings for each Web Application hosted.

ASPIRE PRODUCTIVITY LIBRARY (APL)

The Aspire Productivity Library is a set of components which developers can use to shorten the time taken to develop custom components for Aspire Executive or any non Aspire Executive components. They provide libraries of Data Access routines, Exception Handling routines, Logging, Tracing, Configuration, Caching, Encryption and Web Service Base Functionality.

Figure 3 shows that the Aspire Executive is built upon the APL which is in turn built upon the .NET Framework.

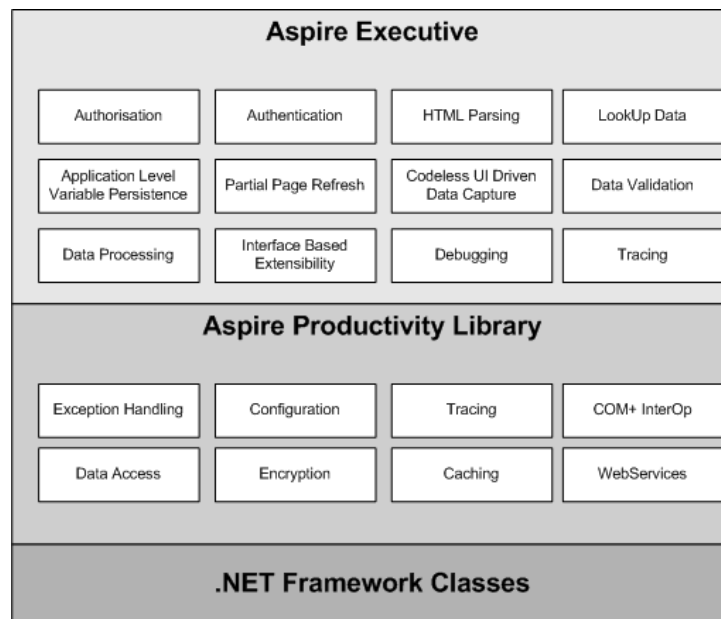


Figure 3 – Layered Construction of Aspire Executive

FUNCTIONAL DESCRIPTION

Aspire Executive is a highly componentised application. **Figure 4** shows the high level organisation of components along with information flow.

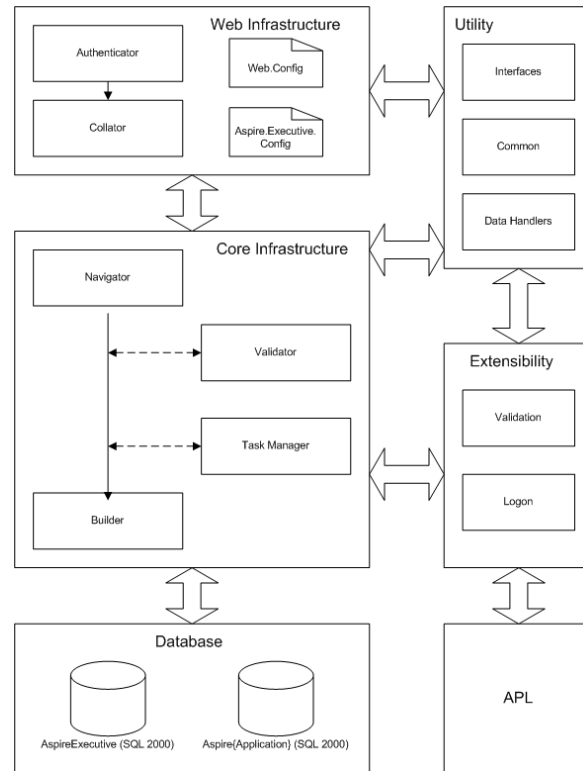


Figure 4 – High Level Organisation of Aspire Executive

The web request is re-directed by IIS to the Aspire Executive Authenticator as configured in the web.config which analyses the request and using any one of Anonymous Authentication, Basic Authentication/SSL or Windows Integrated Security determines if the request can be accepted. If accepted it appends additional information to the request and passes it to the Collator Component which loads all of the request data into a custom data container which is then passed into the Navigator.

The Navigator component then initiates any Validation which has been requested from the page and subsequently initiates any Processing required prior to the next page being displayed. The Navigator then decides based upon the results returned from the validation and processing what is the next page (canvas/snap-in combination) to display. The Builder component is responsible for constructing the canvas/snap-ins combination which constitutes the next page.

Once the page has been constructed it is sent back to the Authenticator which sends the stream back to the browser.

The output of Aspire Executive is not limited to HTML or XHTML. XML can also be generated to produce an RSS type of feed.

DEVELOPMENT BENEFITS OF USING ASPIRE EXECUTIVE

The proven benefits of using Aspire Executive to develop web applications are:

BUSINESS LOGIC FOCUSED

A team can quickly establish a web-site and almost immediately focus in on the areas where custom business logic is required. Since the presentation of web pages, security, drop down construction and validation is essentially taken care of by configuring Aspire Executive, the development team is immediately working with maximum efficiency.

BACK OFFICE INDEPENDENT

The Codeless UI Driven Data Capture allows development teams not to have to wait for back end stores to become available before work commences. Front End Development can immediately begin which means that resources are not left unutilised, and also project management and the client can immediately see progress from day one.

FAST APPLICATION DEVELOPMENT

The previous two benefits lead to a real overall reduction in delivery times for applications over standard development approaches.

EASY MAINTENANCE

The architecture of Aspire Executive Systems is 80%-90% identical between applications, and for this reason developers can quickly move from one application to another to help maintain, debug and trace application issues more effectively and efficiently.

SUMMARY

Aspire Executive has provided Ridgian with a very powerful tool which greatly speeds up the development of applications by providing a ready-made Web Application Infrastructure. Applications can be extended into the three key areas of Validation, Data Processing and Page Rendering. By utilising Aspire Executive Ridgian has been able to meet the stringent deadlines demanded by clients and provide functionally rich applications at the same time.