



e-Business Architectures

The Application Servers role in the Information Technology Revolution

Introduction

'e-business', the electronic delivery of products and services, to customers, suppliers, and employees, is now a mainstream business activity. Indeed, some commentators have proffered the view that 'e-' is dead, and that the e-business phenomenon should be viewed as just another business channel. This does not however take due account of the practical situation facing many companies, which are either struggling to get started, or have embarked upon an e-business venture only to find that assimilating the project with the rest of the organisation is anything but a straightforward task.

Comparing the technology models used to deploy business applications today versus the norms employed only five years ago highlights a revolution in architecture, standards and capability. There is no doubt that the Internet was the catalyst for this revolution. It changed the way software developers thought about designing and building solutions and was the stage for the creation of a number of new technologies that have brought huge benefits to organisations looking to IT for competitive advantage.

The late 1980's and early 90's was a time of huge competition between large technology vendors who fought to assert that their own unique way of employing technology was the "one and only" true path for their customers. These 'standard wars' raged in almost every arena of software and systems development but were particularly unhelpful in the area of interoperability: TCP/IP vs. OSI, DCE vs. ONC, etc. All this activity did little to further benefits to the customer and was more to do with the vendors' battle for market supremacy and their search for technology barriers that could be used to fend off competition.

During the 90's the rise of the Internet and the rapid adoption of a range of simple applications provided a common set of standards that created the potential for inter-operability across the entire range of IT markets. These quickly evolved into common models for building applications with an emphasis on simplicity and development speed.

The advantages delivered by this revolution can be distilled into the following key areas:

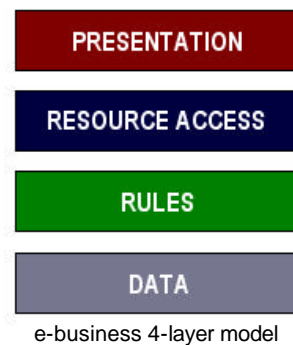
- Clarity of Standards
- Tiered Architectural Models
- Lightweight Transactional Design
- Mainstream Adoption of Object Orientation
- Universal Presentation Layer

This documents looks at e-business architecture and analyses Xology's Concerto product and the benefits it brings to end-users and software developers in the key areas of productivity, ease-of-use, and accessibility. It is the aim of Xology to allow organisations to do "more with less" and to allow them to maximise their existing knowledge capital in building their new e-business solutions.

E-Business Architecture

The flexibility of interoperability offered by clarity of standards has driven an evolution in the way systems are inter-connected in order to build solutions. The e-business architecture has evolved from the traditional three-tier model to a flexible multi-tier model; where each tier of system contributes an aspect to the overall solution.

As a minimum there are usually four tiers or layers: the presentation device (usually a PC running a browser); a resource access point (usually a web server running HTTPD); the rules layer (where the application intelligence resides) and a data layer (some sort of RDBMS or other data filing system).



Three of the layers are well known and well defined: presentation, web server and database. The other, the rules layer, is a more recent addition to the architectural landscape. In early three-tier architectures, the rules layer was indistinct and often spread across the presentation and data layers. These so-called client/server systems made the application complex and hard to maintain and update. Multi-tiered systems evolved a separate rules layer that was custom-built for every e-business application. Soon it was recognised that many aspects of the customer solutions were common and could be packaged as a reusable framework – the concept of the application server emerged.

The role of the application server is to act as the nexus for the dynamic aspects of an application. Much in the same ways as the web server acts as the resource manager for static text and images, the application server acts as the resource manager, controlling access to business rules, application state, and application data. The application server is as fundamental to modern e-business solutions as the web server is to the Internet.

The Application Server

In essence, an application server provides the foundation for an e-business solution and should conform to the standard models and patterns that have made the e-business architecture so successful. In a phrase, an application server should provide “an object-oriented, database neutral, transaction environment” on top of which organisations can rapidly deploy e-business solutions.

The basic objectives of an application server are:

- To provide an application framework based on accepted industry standards to allow interoperability with both existing business applications and additional third-party products.
- To provide an object-oriented technology foundation that provides for component development, integration and deployment.
- To provide reliable, scalable performance of the deployed application.
- To provide an application management and monitoring capabilities to analyse the solutions performance and use of system resources and allow configuration changes and tailoring.

In addition to these, Xology would add the following:

- To support the global nature of the Internet and e-business applications by providing support for multi-language application delivery.

- To reduce the total-cost of ownership of the application solution through the 3 key phases: application development, deployment, and evolution.
- To maximise the value of existing knowledge capital within an organisation that can be employed when building and delivering new e-business solutions.

How Xology Concerto Measures Up

Xology Concerto is a package of integrated components that combine to provide a cost-effective tool-set for deploying e-business solutions. The key components are summarised below and discussed in more detail.

- Java Web Server
- Servlet Container
- Java Application Server
- Rules Engine
- Development Server
- Java Application Components
- Integrated Development Environment
- Administration Server and Console

Java Web Server

The outer layer of the Concerto environment is a Java Web Server supporting the latest web access (HTTPD) protocols. The web server provides a stand-a-lone capability for Concerto allowing it to be installed and used in isolation from any other third-party product. However, Xology realise the importance of interoperability and allow the other Concerto components to work with other web server technologies via: ISAPI, ApacheMod, CGI, etc.

The real advantage for the Concerto web server is packaging. The embedded web-server allows a 'one-click' install and configuration on any Java-enabled platform. This is useful for developers who can install and run Concerto in a small footprint on their desktop system, without the need for other product integration. It is also useful for application builders who want to provide the same levels of integration and ease-of-deployment to their end-users.

Servlet Container

The next layer in the Concerto model is the Servlet container. Servlets is a widely adopted standard that allows server-side software written in Java to be packaged and deployed in different runtime environments, called a Servlet container. Any software developed to this standard can be placed inside the Xology Servlet container. This allows users to integrate third party Java components into the Concerto environment allowing them to benefit from its runtime capabilities and management framework.

Java Application Server

At the heart of Concerto is an application server written in 100% Java and therefore capable of running on any Java enabled platform. The essence of the java application server (JAS) is to provide the execution environment for the ebusiness solution. The JAS controls access to system resources (memory, files, network) as well as providing scheduling and management for individual user transactions using a multi-threaded model.

The main objective of the JAS is to match incoming job requests to a set of known execution rules. The JAS then manages a pool of worker threads based on criteria set up by the system administrator. The request will either be handled, blocked or rejected depending upon system load and the settings defined by the administrator.

A job request can be as simple as matching a browser request via a URL to a set of rules to extract data from storage and format it as a dynamic web page.

However the JAS also has the ability to select the most appropriate execution rules based on the international language context (locale) of the request. The world of e-business is a global one and the need to support multiple languages is key. Xology have integrated knowledge of global application delivery into the heart of the JAS. The same URL accessed by a user in Japan can be configured to map to a different set of execution rules than the same request coming from a user in the US.

The JAS is content-neutral and can produce results in a variety of forms suitable for standard browsers, mobile phones, etc, and is extensible to support the range of new devices that are currently being developed.

Rules Engine

In addition to the basic Java environment Xology provides a business rules layer that allows developers to integrate Java components and server-side JavaScript to produce solutions more rapidly. The rules engine provides a JavaScript “ease-of-use” layer allowing application to be blended from a combination of server-side JavaScript, JavaBeans and other Java class files.

The mission of Xology is to make the creation of e-business solutions more efficient and effective. The use of Java is a key tool that is employed to make this happen. However, Java is still a 3GL language that uses a technically rich (complicated) framework in order to support the breadth of functions necessary to compete across the range of problems into which it is deployed. This rich Java framework can hinder the developer when used to describe business rules. Xology needed to find an alternative, simpler approach and chose JavaScript, rather than building a proprietary language or code generator (a path chosen by other vendors)¹.

The Xology JavaScript Rules Engine provides a perfect environment in which to build business level functionality. As discussed later, Xology provides a range of server-side Java plug-in components that provide high-level functionality, which also removes technical complexity from the business layer of the application.

The JavaScript engine also allows seamless integration with Java objects and is capable of dynamically loading custom Java code and, using introspection, maps these objects into the application environment.

The combination of Java and JavaScript mirrors the development roles often found in larger software foundries. A collection of technical developers, using Java, will create components that provide low-level systems functionality: access to legacy systems, financial calculators, data processors, etc. A second group of developers, sometimes referred to as business analysts, will assemble these low level components and map them with the business rules needed by the organisation. Xology provides each group with the right tools for the job.

Xology Concerto Rules Engine provides for both the Java component developers and the business analysts and allows them to work collaboratively in a seamless environment. For the organisation, Concerto allows existing knowledge about business process to be included in the development of an e-business application without the technicalities of understanding the Java 3GL becoming an issue.

Development Server

Sitting along the JAS and Rules Engine is the Development Server responsible for providing network services to the client-side, developer tools provided as part of the Concerto package. Today the Development Server provides two features: file access and debugger support.

A unique feature of the Concerto environment is its ability to provide network-based debugging facilities in a multi-user/team development environment. Each developer may work within the same server environment and have separate and concurrent debug sessions.

¹ See a discussion of Java and JavaScript in the appendix to this document.

Xology's debug capability extends Internet-wide and supports its own file services for use when standard platform mechanism may not be available. This optional file service is secured via an ACL (access-control list) at a system network address level. It only provides read-only access to application source (if available) for remote debugging purposes.

The debugging service provides transaction-level (per user/per thread) debugging of the applications inside the JAS. The one-to-one mapping between debugger functions – break, step in, resume, pause, etc. – and the source-code written by the developer mean that what the debugger presents is as relevant and useful to the developer as possible – maximising efficiency when diagnosing problems.

The debugger allows the application-under-test to be animated normally using a client browser, until the test condition occurs. The debugger then takes control of the application, pauses it, and sends the internal state of the application server thread to the debug client. The state information contains all application level objects, the execution-frame context, and results of any specific evaluations requested by the debug client.

Crucially, the debugger will only affect the thread under analysis; all other application users or developers will continue normally. Using these tools the developer gets unprecedented visibility of the internals of the application, allowing faster bug diagnostic and resolution.

Java Application Components

Inside the Concerto environment is a collection of powerful Java objects that can be re-used to enable the creation of e-business solutions more quickly. They allow the developer to concentrate on the business functionality rather than grapple with the technology implementation:

XML Processor: provides both document object model (DOM) and event-based processing of XML documents.

XML Generator: provides transformation between Java object models and XML structures to ensure well formed, syntactically correct documents are produced.

Database (JDBC): provides SQL access to transactional database systems. Supports configurable connection pooling per database, per user level.

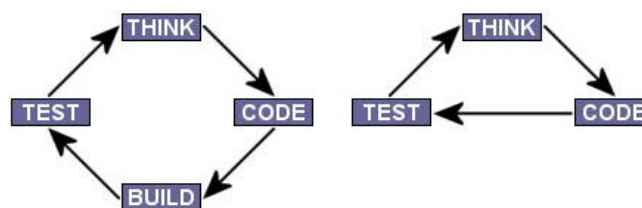
Session Manager: provides session support across multiple web transactions allowing developers to track individual users, persist session data, etc.

Resource Manager: provides access to application components via resource names. Allows a developer to abstract application configuration out of code and into separate resource management layer. Allows management of: database connections, property files, data files, JavaBean, etc.

Other components provide additional functionality: ObjectFile for object persistence, International Tokens for internationalisation (i18n) of e-business applications, Email integration, File access, Fragments, etc.

Integrated Development Environment

Xology developers benefit from the integrated development tool-set called Coda. This provides an integrated editing, debugging, and test environment that focuses on speeding the primary developer cycle.



JavaScript speeds the development cycle.

Outside of the design phase, software developers spend the majority of their development time in this primary developer cycle. Xology's first job is to make the cycle as rapid and productive for the developer as possible. Unusually, we start right at the point the developer puts finger to keyboard with our advanced editing capabilities.

Every time the developer types the keystroke is analysed in context to the files type, the position in the file and its current contents. In fact, the Coda Editor uses a technique called 'full block syntax parsing' (FBSP) to quickly analyse the structure of the file and its current state. This information is then relayed back to the developer using a colour coding system and structure view.

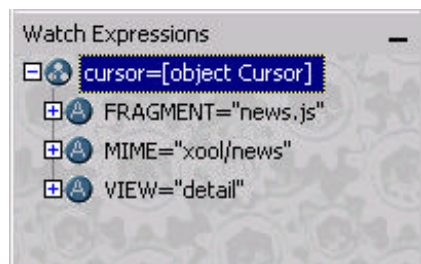
Unlike other colour keyword highlighting systems, Coda understands the grammar of individual languages and can even cope with single files that have multiple languages embedded within them – a good example of this would be a Dynamic HTML file with client-side JavaScript. Coda will identify the HTML and JavaScript separately and parse each block according to its own rules.

The aim with the FBSP system is to inform the developer of coding errors as soon as possible in the developer cycle. A problem found early saves wasted time. Developers using Coda find their typing error rates fall and productivity increases as a result of using the FBSP system.

Once a piece of application code has been developed it needs to be tested. Coda provides the developer a way of running the code with a single push of a button without leaving their development environment. Integrating with Internet Explorer 5.0 and above, Coda can be used to browse applications under development as a user would.

To support this capability, Coda manages any number of sessions to a combination of application servers, web servers or simple directory folders. When a file is ' browsed ' Coda knows how to execute the file in context of the session it belongs to. If the file is part of a simple folder, then it just renders the same file in a browser view. If the file is part of a web server session then it requests the file from relevant URL to ensure any server-side processing is completed. The same is true with application server sessions, including those to the Concerto JAS.

Unfortunately, when developing software the initial results are not always what the developer predicted. For this phase of the developer cycle, Coda provides an integrated debugging capability by coupling with Concerto's debug server. This allows developers to see "inside" their applications as they execute enabling problem resolution to be as efficient as possible.



Examining the structure of a RDBMS result-set and current state using the Coda Debugger

Administration Server and Console

Once the application has been built it needs to be managed. Each installation of the Concerto application server has a shadow administration server – at least one administration server is needed per physical system on which Concerto is installed. However, a single administration server can police multiple Concerto servers in a cluster environment.

The separate administration server is critical for 24x7 support, allowing dynamic reconfiguration and management of the application server.

The Administration Console is the main public interface into the administration server functions. It allows an administrator access to all the normal configuration and management features. In addition, the administration console provides detail runtime analysis of system resource utilisation and can display charts and statistics of resource access, thread utilisation, down to individual thread level.

The administration console also contains a wealth of online documentation, tutorials, and self-help guides to allow individuals to quickly master the Concerto environment.

Did we measure up? – A summary

Standards Compliance

Xoology supports Java and JavaScript – both well defined industry standards – we also support a wide range of de-facto and industry compliant interfaces:

- Web access protocols: HTTP, HTTPD, ISAPI, ApacheMod, CGI and Java Web Server
- Component/Container interfaces: Servlets, JavaBean, Java class, JNI
- Content standards: XML, XPATH, HTML, WML, XHTML, (any)
- Platforms: Windows NT/2000, LINUX, Solaris, AIX, HP/UX, OpenVMS.
- Enterprise Application Integration Interfaces: PROIV, Reality/X, PICK, Glovia ERP, etc.
- An assortment of mail, directory and other application services.

Java Integration

Xoology Concerto is a 100% Java development environment allowing our developers access to the wealth of features within the Java 2 world:

- Database access via JDBC
- Name system and directory access via JNDI
- Remote object invocation via RMI
- Dynamic component development using JavaBean, etc.

Database Neutrality

Access to database is provided by JDBC providing connectivity to all major relational database systems, flat filing systems, and open-source products. Access to additional third-party middleware greatly extends the range of data access options available.

- Qualified examples: Oracle, SQL*Server, UDB/DB2, MySQL, MS-Access, Sybase, Informix, JBase, RMS, IMS

Component-based Development

Access to Java component and container technologies (JavaBean, Servlets) as well as dynamic loading of Java class files means that Xoology Concerto will provide a superb component framework.

Team-based Development

Xoology Concerto is the only JAS environment that provides a multi-user, team-based debugging environment. Application developers can develop and debug on a central server at the same time without affecting each other.

Transaction Orientation

Concerto provides transaction control within JDBC interfaces and session management across multiple transactions.

Reliable, Scalable Performance

Concerto is a proven application foundation with a lightweight, multi-threaded engine capable of scaling across multiple processors and multiple instances with software clustering.

High-level Functionality

Concerto includes numerous high-level application components that simplify the development of XML-based, business applications.

Management, Testing, Monitoring Tools

The integrated Coda development tools and Administration features that Xology Concerto provides a well-rounded, feature-rich, offering that maximises organisations return on ebusiness development investment.

Appendix A – Java and JavaScript

Why Java?

Even if application servers are the accepted way of supporting e-business solutions, and have proved that they increase productivity; why a Java Application Server?

The reasons are that Java is a relatively new technology that was designed to be part of the “online” world and in particular the Internet. Because of this it has a number of characteristics that make it highly relevant to this space and very powerful:

- It's a better 3GL – Java is a “new” third-generation language that fixes a number of the issues found with earlier predecessors such as C++. Java language removes some of the common “trip-wires” for developers that used to cause so many bugs in the days of C and C++; e.g. memory allocation is controlled by Java meaning no allocation problems, no referencing errors and no memory leaks.
- Its object oriented – Java supports a much more productive development model, O-O. This style of development allows software to be built in a way that encourages re-use through the development process. Well-designed O-O systems will be much faster to develop, involve much less code, and be easier to maintain.
- It supports component-based development – Java comes with a wide-range of useful classes that include rich functionality into the basic environment. It also comes with support for component interfaces that allow Java objects from different vendors to be combined to create solutions.
- On the server, Java is mature – the multi-threaded nature of the Java environment has made it well suited for server-side application development. The industry has collectively invested many \$millions in making Java the premier server development platform.
- It is robust – the Java Virtual Machine (JVM) is a standard part of the Java environment available on all platforms. This is the single execution “manager” for all Java applications – it is proven and robust.
- Its ubiquitous – Java is very popular and runs on a wide range of operating platforms.
- It is platform independent – a Java application written on NT will execute on OpenVMS. This provides the customer of a Java solution a high degree of platform independence and the software developer with access to the widest possible market.
- Its fast – You do pay a performance penalty with Java above native languages such as C/C++. However with processor performance becoming so cheap and the differential between Java and Native language so small, that today this is a non-issue.

Why JavaScript?

Netscape originally developed JavaScript as a more productive development language for building Internet Applications. The most popular early implementations were inside web browsers such as Netscape Communicator and Internet Explorer. However the “language of the Internet” was developed for much wider deployment and is capable of much more.

Today JavaScript is a standard language that is managed by the European Computer Manufacturers Association or ECMA. This body ensures that JavaScript is a standard language, supported by clear definition and requirement documentation.

The key benefits of JavaScript are:

- It is simpler – Low-level languages such as Java require a relatively complex framework – and therefore a high level of knowledge – to use. JavaScript was derived from a combination of Java, BASIC, COBOL, etc. and has a simpler framework that is accessible to anyone who has experience in any programming environment.
 - It’s quicker to learn – allowing existing IT staff – and therefore existing IT knowledge – to be used in the creation of new IT solutions.
 - It’s faster to develop in – having a simpler, more flexible framework means that building application is far quicker in JavaScript. The language deals with issues such as data type conversion automatically, allowing the developer to concentrate on the substance of the application being built.
- Retains powerful O-O features – JavaScript supports some of the key capabilities of the Java environment such as reuse, inheritance, encapsulation and data hiding.
- Its dynamic – the interpreted nature of JavaScript allows powerful runtime capabilities: Loading of JavaScript object files, blending with JavaBeans or Java classes at runtime, etc.
- It’s even more popular than Java – the population of JavaScript developers is about two-four times bigger than the next closest language. This testifies to the accessibility of JavaScript and means that organisation will find it easier to resource their projects with qualified individuals.

All comments on/questions about this article should be directed to Xology Software at the address below.

Xology Software
Northgate Building
Boundary Way
Hemel Hempstead
Hertfordshire
HP2 7HU
Tel: +44 (0) 1442 272658
Fax: +44 (0) 1442 231382

E-Mail: marketing@xology.com