

1

Introduction to WebSphere

We are in the early part of the 21st century, where our software challenges have grown and continue to grow faster than our means of dealing with them. Web application servers and the capabilities that they possess represent the component servers being used to solve today's complex business problems. Web application servers combine the best of object technology for clients and presentation (servlets, JavaServer Pages) with the latest technology for representing and implementing the business model in software (web services, Enterprise JavaBeans).

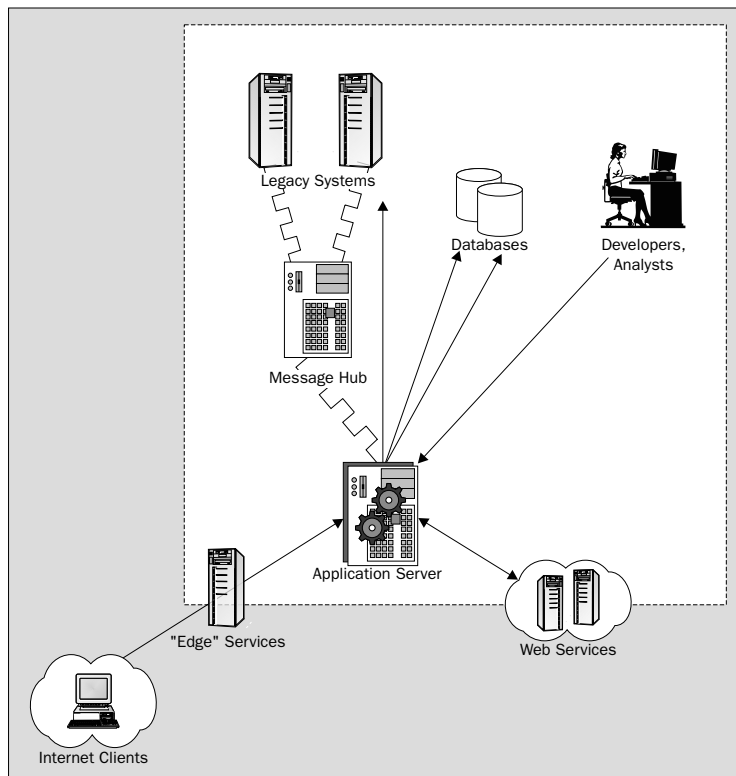
Application servers are also utilizing the latest integration software using connectors and asynchronous communication (Java Message Service) to tie together old and new systems into robust clusters of computing power that meet the ever changing demands of modern e-business and business in general. Many of these application servers rely on Java for implementation technology and basic services.

We are in the application server generation. This phase of the evolution of computing technology is as significant as the introduction of computing models such as the relational database and the transaction monitor. Each of these generations of software has leveraged the newest in hardware technology and in fact has driven hardware and the base operating systems in particular directions. The same applies to the application server, which in many ways is driving our industry forward. Middleware, as a class of software has been traditionally dominated by products based on technologies such as DCE (Distributed Computing Environment) and OMG's CORBA. Middleware now includes the application server as a full member.

What is WebSphere?

The WebSphere brand represents a platform for today's e-business applications. The WebSphere vision and direction represent an ongoing strategy and vision about how software will evolve and meet future application needs.

WebSphere is an application server that runs business applications and supports the J2EE and web services standards. It is a place to host business and presentation logic, which can integrate into the existing computing infrastructure of small as well as large organizations. The following figure provides an overview of where an application server fits into a typical computing environment:



As you can see from the diagram, the application server is the hub of the e-business computing infrastructure. Application servers provide a place to execute policies, to enforce terms and conditions, and to apply business rules. Global clients commonly access these engines using browsers and pervasive devices that operate on both static and dynamic content. The web server sits between the browser and the application server in the most common topologies. Static content is normally served directly from the web server while dynamic content requests are passed on to the application server. Content can be cached at the edge of the network to drastically reduce both network traffic and system load. "Edge Services" refers to the WebSphere capabilities that work in conjunction with the web server to provide this caching at the edge of the network.

Of course, the application servers are not islands, as they typically need to access persistent information stored in one or more databases. In addition, existing systems and applications often need to be leveraged as part of an e-business solution; messaging technology and message brokering are frequently used to drive such legacy system interactions. We complete the picture by showing that developers need an easy way to create and update the applications that are hosted by the application server. Similarly, business analysts need a way to model, monitor, and maintain the various business processes and policies that run there.

How does the application server perform these tasks? The application server, through the services that it provides, handles the presentation and business logic. As the J2EE standards have matured and web services standards have quickly emerged, more services are inherently part of the application server.

Application servers also contain messaging capabilities via the Java Messaging Service (JMS). JMS and the additional messaging features of WebSphere 5.0 provide further evidence for the fact that the asynchronous and synchronous programming models are both required to build next generation applications. The application server provides these features and functions for environments where existing infrastructure is not in place, or where the features need to be more tightly integrated and managed by the application server.

Interoperability, co-existence, and plugability are important themes in the WebSphere Application Server. A first glance at how these themes are leveraged is provided in the next section, where details on the various application server offerings are described.

The WebSphere Application Server

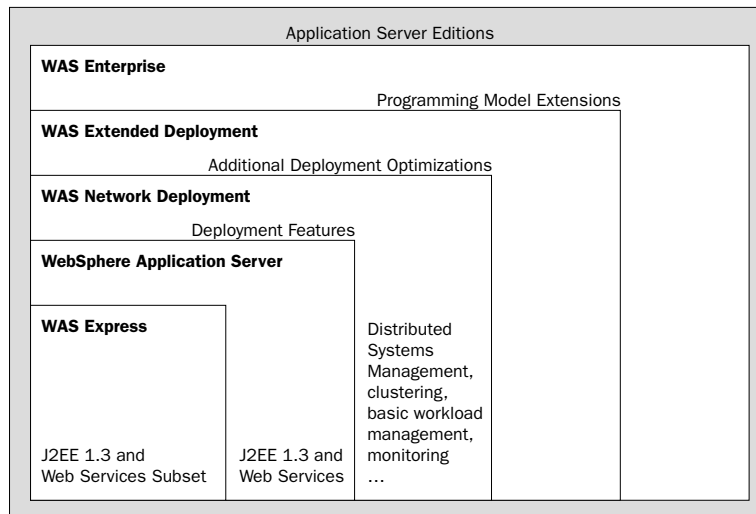
WebSphere Application Server (WAS) represents a set of application server offerings each having its own specific capabilities and functions. This allows WebSphere Application Server to address a broad spectrum of solutions, ranging from the most rudimentary web application to transactional and scaleable e-business applications.

In Version 5.0, WebSphere Application Server has a variety of packages:

- ❑ **WebSphere Application Server – Express**
This is a new entry-level offering, which will support servlets and JSP pages. It is targeted to rapid application development and building applications based on associated application-development tooling.
- ❑ **WebSphere Application Server**
The generic name, WebSphere Application Server, applies to the J2EE 1.3 certified version of WebSphere that is configured and managed on a single machine. This is similar to the version 4.0 package known as the WebSphere Application Server – Single-Server edition.
- ❑ **WebSphere Application Server Network Deployment**
This package is similar to the version 4.0 package, known as WebSphere Advanced or WebSphere Application Server, Advanced Edition. It adds the ability to manage multiple application servers and handle clustered environments. This package comes with a basic WebSphere Application Server.
- ❑ **WebSphere Application Server Extended Deployment**
The edition of WebSphere that extends the WebSphere Application Server Network Deployment, with additional features for scalability and manageability. This is a new package for version 5.0.

- ❑ **WebSphere Application Server Enterprise**
The high-end package for WebSphere Application Server. It introduces additional programming interfaces and supporting run-time capabilities referred to as the programming model extensions. All of the possible capabilities and configurations for WebSphere Application Server are enabled by this package.
- ❑ **WebSphere Application Server for z/OS**
The application server package delivered for the z/OS environment. The packages described so far apply to all operating system environments except for the z/OS. WebSphere Application Server for z/OS is a special packaging optimized for the z/OS environment that encompasses basically all of the functions described by those packages listed previously that are appropriate for the z/OS environment.

The following diagram positions the various editions of the application server:



Now let's look at the features provided by these various editions on more detail.

WebSphere Application Server – Express

WebSphere Application Server Express provides a J2EE and web services subset application server. This is the extension of the Standard Edition concept that was part of WebSphere Application Server 3.5 package. It does not support EJBs; however, there is application development tooling provided as part of the package. WebSphere Studio Site Developer code is provided with the runtime in a single integrated offering. The emphasis is on integrated tooling, with a full Java, servlet/JSP page building, and support for JavaScript and tag libraries. You do not need to be a Java programmer. The secondary focus will be on JSP pages and servlets (but not EJBs). Ease of use, small footprint, and pre-canned applications are all included here.

WebSphere Studio Site Developer is available separately as a tool-only environment. However, even in this configuration, there is an instance of WebSphere Application Server – Express provided as the unit test environment.

WebSphere Application Server (WAS)

The WebSphere Application Server provides a run-time and management environment that can host applications that fully leverage the J2EE 1.3 and web services programming models. This environment is managed through a built-in management capability driven by a web browser. WebSphere Application Server is a single machine execution environment. WebSphere Application Server 4.0 supported J2EE 1.2. The major additions in J2EE 1.3, and thus WebSphere Application Server 5.0, include support for EJB 2.0 and Servlet 2.3. Additional details on each of these are provided in upcoming chapters.

In WebSphere Application Server version 4.0, the environment similar to the WebSphere Application Server (WAS) was known as the WAS Single Server (meant for single machine or single server production applications). The same server is in the test environment of the application development tools offering named WebSphere Studio Application Developer.

WebSphere Application Server Network Deployment (WAS-ND)

This package of WebSphere Application Server is focused on providing a deployment environment for multi-node and multi-server environments. It provides the same programming model support as the base WebSphere Application Server, but adds support for a variety of topologies and architectures consisting of multiple machines and multiple application servers managed under a single umbrella.

WAS-ND provides deployment optimizations mainly in the form of clustering, workload management, and administration. These features allow larger scale deployment than is possible in the base WAS configuration. A deployment manager is introduced as key new concept in an environment that is capable of managing a set of applications from a single console that run across multiple server and multiple machines. More details on this are in upcoming chapters.

WebSphere Studio Application Developer is the associated application development tooling used to construct applications for this execution environment.

WebSphere Application Server Extended Deployment (WAS-XD)

WebSphere Application Server Extended Deployment provides additional optimizations, extending the base and objectives established by WAS-ND. It does this by adding capabilities such as cross-domain failure bypass and dynamic load balancing, in support of enterprise class deployments. Optimizations in WebSphere Application Server Extended Deployment are actually broken into three categories, as follows:

Performance Optimization

This enables the same number of machines to do more work by using a series of workload balancing features. Application server instances will be able to detect variable run-time conditions, and then redirect work dynamically to the machines that are the least busy. These and other examples of performance optimizations will be explained in later chapters.

Availability Optimization

Highly available systems often need to have at least two instances of key run-time components, such as configuration repositories and workload controllers. This allows work to continue in the event of component failure. The more failure bypass that a system offers, including for the failure of internal components, the less that an end user will be disrupted when something goes wrong. Additional features relating to workload management and systems management extend those available in the Network Deployment configuration.

Scalability

Scalability means many things in the context of application serving and WAS-XD addresses all of them. Most traditionally, it means that more work can be handled easily. Effective scaling ensures that all customers receive the service that they request. Effective scaling lets your business expand beyond its traditional boundaries to embrace new partners and suppliers. Features and functions, which support these capabilities, are described in later chapters.

Scaling also means that you can deploy new applications into large environments easily and automatically. This includes applications that extend out to the edge of the network. You can effectively manage those environments, and through plugability, you can leverage the investments that you have made in existing management and security software, as well as operational skills. There is obviously more to come on these topics as well.

Applications that run in a WAS-XD environment will be created with the WSAD tooling. Additional configuration and deployment activity, specific to the XD-enabled runtime will be performed at deployment and installation time.

To summarize, it is expected that WAS-XD configurations, when compared to WAS-ND configurations, will:

- ❑ Run applications faster based on additional run-time optimizations
- ❑ Support high availability configurations more directly
- ❑ Readily handle more machines and more servers in a distributed topology

WebSphere Application Server Enterprise (WAS-EE)

The WebSphere Application Server Enterprise is the next generation of the WebSphere Application Server Enterprise Edition of Version 4.0. The Enterprise product offers some new programming model extensions. A programming model extension is an application programming interface and the association run-time and management features. The application programming interfaces are used by application developers, and these interfaces generally complement the standard interfaces and are used in conjunction with them.

With reference to the packaging described earlier, WAS-EE contains the combined content of WAS, WAS-ND, and WAS-XD, in addition to the programming model extensions. This makes it the most comprehensive WebSphere Application Server offering available. WAS-EE also includes a WebSphere MQ license, enabling construction of applications that leverage WebSphere MQ and WebSphere Application Server.

WebSphere MQ is IBM's flagship messaging product, providing middleware to enable asynchronous communication between various environments, including application servers.

The first set of capabilities provided by the programming model extensions, center on extending and leveraging the EJB 2.0 component model that is part of the J2EE 1.3 specification. The Dynamic Query Service extends EJBQL, enabling query statements to be processed at runtime. This provides a great deal of flexibility and lets applications start dynamic e-business.

Complementing the EJB 2.0 component model is the support for Access Intent via Application Profiles. An example of access intent would be concurrency. The concurrency access intent, for example, can be set to optimistic or pessimistic, depending on the desired behavior of a transaction. Once again, this will make higher performing applications and enable more advanced reuse of components. As this function has both a programming implication and a deployment aspect, portions of this may appear in the Extended Deployment package of the server as well.

WebSphere Enterprise provides strong support for accessing and leveraging existing assets. The WebSphere Studio Application Developer Integration Edition development environment and the WebSphere Application Server Enterprise run-time, work together to provide a service-oriented architecture from which advanced solutions can be constructed. These solutions use services, which represent and encapsulate a variety of resources both within and outside the organization, as the building blocks for complex compositions and choreographies.

The concept of business process choreography is introduced in the enterprise server. This provides for a variety of workflow patterns, including interruptible flows, compensated flows, and micro-flows all to be created with development tools and executed in the runtime. These capabilities are supported by the tool using a Service-Oriented Architecture (SOA). Workflows can drive any service, including intra-enterprise calls to components and connectors and external calls to web services.

For example, you could create basic service definitions through "adapter tooling" that visually connects your Java applications to Enterprise Information Systems. You could then choreograph these basic services into "composed services" that perform higher-level business activities. Wiring these interactions together in a visual fashion makes it easier for developers to create applications, and to preserve the flow structure of the application when underlying service implementations change over time.

Other productivity gains come from the close integration of components and messaging systems. This includes the automated transformation and mappings required between message flows and components to satisfy application needs.

The ideas of messaging and JMS are built upon through the introduction of Extended Messaging Support. These tool-supported APIs are used to program various patterns that are common when dealing with messaging-based interactions to WebSphere and non-WebSphere systems.

Transactions provide a unit of work scoping mechanism for short duration activities, and the compensation support that complements workflow provides a long running unit of work concept. The Activity Session Service and the Last Participant Support service both provide intermediate unit of work and resource management options. The combination of these features offers a comprehensive set of capabilities for properly scoping business activities and handling exceptional conditions that may arise.

Services such as WorkArea and Internationalization allow more flexible and adaptive applications to be constructed. There is a CORBA C++ SDK, to enable C++ clients to EJBs and to provide a basic C++ CORBA Server capability. There is a Scheduler service and Asynchronous Beans delivered as part of WebSphere Application Server Enterprise. These capabilities to allow parallel and asynchronous work to be spawned and managed in the system under prescribed environments.

WebSphere Application Server for z/OS

The z/OS edition of WebSphere is specially optimized to provide scalability and reliability, consistent with that of the z/OS environment. Some of the core application server features have optimized implementations on the z/OS platform, taking advantage of the rich and mature systems services, which are available. Special affinity is provided for the resource managers such as IMS, DB/2, and CICS that reside on the z/OS platform.

WebSphere Application Server applications are portable from a source code perspective and can be easily moved from one of the other application server editions onto the z/OS environment. Additional information on the vision for WAS 5.0 and beyond can be acquired at <ftp://ftp.software.ibm.com/software/websphere/partners/TheVisionForWASV5AndBeyond.pdf>.

WebSphere is a Platform

WebSphere Application Server is actually just one set of the many offerings that are associated with the WebSphere name. In fact, the WebSphere brand includes a number of products. IBM WebSphere Portal is a label for a set of offerings in the portal market. The WebSphere brand also includes the IBM WebSphere Commerce set of products, IBM WebSphere Host On Demand and Host Publisher IBM WebSphere Translation Server, the IBM WebSphere Voice products, IBM WebSphere EveryPlace products, IBM Transcoding Publisher, and so on. These are complemented by the IBM WebSphere MQ run-time products and the IBM WebSphere Studio set of application development tools.

WebSphere as a platform is essentially about providing three things for users and solution providers who choose this platform:

- ❑ **Reach and User Experience**
Personalized and streamlined access to content and collaborative services on any variety of devices (including pervasive devices). Note that this also includes the ability to conduct electronic commerce.
- ❑ **Business Integration**
Integration services both within and between enterprises to promote business agility and to strongly support business-to-business initiatives.
- ❑ **Foundation and Tools**
An infrastructural underpinning for a whole range of e-business solutions. Application serving and integrated development environments are some of the key elements here.

The WebSphere Application Server provides a fundamental role in the platform. All of the run-time products that are in the platform depend upon the WebSphere Application Server to provide basic services. The Portal server, for example, depends upon servlets and business rules which come from various applications server packages. The Commerce products make heavy use of the J2EE programming model and of many of the extensions introduced in the application server. The consistent usage of the application server by the platform products provides a more manageable solution and a better overall user experience for customers.

WebSphere Product Objectives

WebSphere Application Server has evolved to where it is today because of a few basic product objectives and goals:

- ❑ Provide a platform for enterprise computing
- ❑ Provide a platform for innovation
- ❑ Enable application developers to focus on building applications, not infrastructure
- ❑ Establish and maintain standards leadership
- ❑ Provide a flexible set of product configuration options

The first WebSphere product objective is to provide the platform from which we can really observe and realize the fusion of the web computing world with that of the core enterprise computing world. Properly combining these competencies can provide significant competitive advantage in the marketplace. This is a foundational role, if ever there were one, in the world of middleware and software in general. This objective helps to establish WebSphere as the platform for enterprise computing.

While being a stable and reliable platform from which a company can run the business is important and primary, WebSphere also serves as a platform for innovation. WebSphere is a modern software engineering platform. WebSphere got here by introducing new technologies in ways that were usable, consumable, and palatable to large and small organizations trying to solve business problems. For example, WebSphere has recently begun introducing web services to a heavily J2EE-based environment. The presentation of this implementation has been done in a way that web services are easily adopted and leveraged. There is no leap of faith or step increase in skill requirements to adopt and leverage the new things in a given version of WebSphere.

Innovation is also demonstrated by the Enterprise Edition of the WebSphere Application Server. This package contains a couple of key categories of function. First, there are those functions that extend the programming model available in J2EE and web services. These new interfaces are intended to enable developers to easily solve the more complex problems that are bubbling to the top of lists across the world. These new interfaces are previews of the interfaces that will emerge in future versions of J2EE. Activity Service (JSR-95), WorkArea (JSR-149), and Internationalization Service (JSR-150) are Enterprise Edition programming extensions that really do provide tomorrow's standards today.

This innovation comes in a production-ready platform, so not only are the new capabilities available, but they are ready for production use. A second set of capabilities in the Extended Deployment offering, introduced in WebSphere 5.0, also represent innovation. This set, often referred to as "qualities of service", is provided to deliver WebSphere applications into complex and dynamic environments. These innovations and features do not affect the application programming interfaces, but focus on ensuring that large-scale deployments of applications can be successful in a variety of complex environments.

The third objective of WebSphere that lives in the hearts and minds of the engineering team revolves around the goals of middleware. The objective of WebSphere and perhaps the ongoing quest is to let application developers get back to building applications instead of middleware. The types of applications being built today are increasingly more functional, and thus more complex. It is the job of middleware to keep providing services and capabilities that let the developers build applications, rather than generic middleware that applications are constructed on.

As an example, there has been a perceived need for additional synergy between the synchronous invocation model and the asynchronous model of computing. It is increasingly the case that applications need a combination of both of these programming styles in a single solution. WebSphere Application Server adds some new capabilities in this area in version 5.0 to further encapsulate the differences in the models as they are presented in the application server, while still making these architectural patterns available in the application server. There is also an evolving component model in the form of EJBs that again provides additional abstractions to the application builder and implies the need for more run-time capabilities.

Establishing a platform certainly includes focus on the objectives already described. However, establishing a platform also means establishing and maintaining leadership in standards. This is the fourth product objective for WebSphere Application Server. The J2EE standard, the evolving web services standards, the CORBA standards, and many others are of ongoing interest to WebSphere as a product and to WebSphere as a platform. Not only will WebSphere continue to introduce standards and contribute to the ongoing standards definition activity, but the goal is now to be able to deliver early implementations of these standards, and most importantly be persistent in delivering compliant, robust, scalable, and reliable implementations of those standards.

J2EE 1.3 is a perfect example of the kind of leadership that is important from a WebSphere perspective. The contribution of the IBM team during the formation of the J2EE 1.3 components is significant. WebSphere architects were in on the ground floor of J2EE 1.3 highlights, such as EJB 2.0 CMP support and EJB 2.0 message-driven beans support. Through the timely introduction of WebSphere Technology for Developers version 5.0, WebSphere demonstrated early implementations of the standards by becoming the first major run-time vendor to be certified. Through the introduction of WebSphere Application Server version 5.0 offerings, the J2EE 1.3 loop is being "closed" by delivering run-time implementations of the standard across a variety of platforms that can enable large-scale production usage of these standards.

A final product objective focuses on the product packaging and organization of capabilities into the various WebSphere Application Server editions. While the various configurations of WebSphere have different purposes and function, consistency and structure allows customers to easily upgrade from one edition to another.

This theme is often referred to by the engineering team as the "pay as you go" principle of WebSphere. This principle states that any additional development and management complexity, targeted at complex and large-scale environments, must not be observable until the capability is required. This means that WebSphere does not overwhelm developers starting out with simple applications on simple topologies. This also, however, does mean that WebSphere is customized and specifically architected for a large variety of complex environments.

WebSphere Principles

Beyond a set of product objectives and goals lie a set of values and principles that internally drive the WebSphere Platform. The basic principles, which drive the engineering activities around WebSphere, include:

- ❑ Treating platform as a development principle
- ❑ Leveraging core competencies
- ❑ Robustness
- ❑ Using what we sell

The engineers that work on WebSphere take the platform concept and interpret it with their own set of perspectives and contexts. To the WebSphere engineers, a platform means that they have something that is consistent, works as a single unit, and provides a user experience that is complete, rich, and robust. This is the first and most broad reaching of the WebSphere principles.

We have a platform, what does this mean? To some, it means that WebSphere Application Server is like an operating system. The WebSphere engineering team runs on and abides by many of the principles of operating system development. These are engrained in the team. Many members of the WebSphere engineering team work in development labs in places where operating systems such as OS/400, AIX, OS/2, and OS/390 were invented, delivered, and supported. Today the WebSphere engineers work in buildings alongside many of the teams that continue to be involved in operating systems development. The synergies are too many to describe.

WebSphere is effectively a layer over the top of the operating system that provides all of the programming abstractions (at least in combination with Java) that are needed to build next generation e-business applications. J2EE and application servers are effectively distributed operating systems from an API perspective. WebSphere is a distributed operating system from the perspective of performance, reliability, availability, recoverability, usability, and serviceability. This is a fundamental tenet of WebSphere.

WebSphere is based on a philosophy and value set that runs deep into the history of computing and into the history of IBM. Operating systems were the extent of software that was provided by the computer manufacturers in the beginning, or at least shortly after the beginning. IBM pioneered such concepts as transaction monitors and databases. These provided a layer that shielded, simplified, and expedited solution development. This layering idea has grown to now encompass the rich programming model that is contained within WebSphere. Within WebSphere, this platform-oriented thinking lives on and contributes in many ways to the WebSphere Application Server that exists today.

A second principle revolves around leveraging core competencies. IBM has a rich and varied set of engineering talents, spread across the globe. When specific skills are needed, the team within IBM that has those skills is found and commissioned to become contributors to the application server.

For example, when JMS became part of the J2EE, the WebSphere team went to the messaging team that provides WebSphereMQ and acquired the necessary JMS components of the application server. When object-oriented query entered the J2EE specification, the team that constructed the IBM Component Broker query service was once again called upon to deliver. In general, WebSphere takes a pragmatic approach to constructing the WebSphere Application Server, soliciting and in fact requiring contributions from the IBM Software Group at large.

Robustness is a third key principle. Being robust, WebSphere isolates the execution of customer code from the execution of system code. This is generally not the case in a world where we have no "kernel mode" to rely on for separating and isolating customer-written applications from the system. Through a series of internal components and a set of powerful reliability, availability, and serviceability capabilities, WebSphere does provide the environment of choice for application serving.

For example, our reliability, availability, and serviceability capabilities, which have been improved again in version 5.0, clearly demonstrate a commitment to robustness and reliability. In many cases, the WebSphere runtime will report a problem, point to the actual line of code (customer code and system code alike) that causes an event or failure to occur and suggest a solution. Version 5.0 actually adds an additional First Failure Data Capture (FFDC) capability that allows problems to be diagnosed without collecting additional trace data, or re-running the application to gather logs.

A knowledge base is built into the product. If a failure occurs a second time, the system will remember this and immediately gather the additional data necessary for diagnosis and providing a solution. Keeping the WebSphere JVM alive is a key part of the robustness story. Through careful programming and isolation techniques, WebSphere does everything possible to prevent a rogue piece of application code bring down the entire server.

The fourth and final principle that applies to the application server and the entire platform is that of "Use What We Sell". This principle states that the technology that WebSphere provides is the technology that we use to build some of the components of WebSphere. A good example of this in the WebSphere Application Server is the administration support provided by version 5.0. This is a J2EE application that makes heavy use of servlets and JSP pages. In WebSphere Application Server Enterprise, a number of components, including Business Rule Beans and Workflow, leverage J2EE APIs such as entity EJBs in their implementation.

A Vision for the Future

What is next for WebSphere? Where does the platform go from here? WebSphere Application Server, version 5.0 is one of the strongest and richest modern application server available today. Today is the keyword, as there is constantly a change in technology.

In the months following the general availability of WebSphere Application Server version 5.0, we can expect mostly the usual and maybe a bit of the unusual. There will be service releases and an appropriate introduction of the rest of IBM's WebSphere branded products that leverage and run on WebSphere Application Server version 5.0. With these additional IBM products in place, partners will then complement these offerings with additional run-time and tooling-based offerings. Customers will enjoy a complete platform offering and proceed to deploy large-scale production applications.

In addition to service releases, it is likely that incremental functional capabilities will rollout and the J2EE 1.3 base will be enriched with more capabilities. The openness and flexibility of the WebSphere Application Server in version 5.0 suggests that it will be a production platform, used widely and over a long period by many large customers.

Meanwhile, it is also expected that there will be tactical activity in the standard areas. J2EE 1.4 is already something being defined and WebSphere architects are working on the steps necessary to provide support for this next standard. The version 5.0 architecture has already planned for and has even implemented some of the future J2EE standards. JMX support, for example, is already built into the administration model of WebSphere version 5.0. Another example is in the area of web services. JSR-109 will add web services directly into J2EE. WebSphere Application Server will have early support for this in version 5.0 and will formalize this support as the standards become final. The intention will be to deliver production-ready products supporting these features and others, in a fashion similar to that which was used for the version 4.0 and 5.0 deliveries.

Strategically, the WebSphere Application Server will continue to evolve with the industry, to lead the way as the platform for building that next application which delivers business value. Some of the version 5.0 features and functions clearly represent an initial offering of function in a specific area. This is especially evident in the programming model extensions. Many of these could be enhanced and upgraded as initial usage yields additional requirements and new application types are invented which require additional middleware support.

Internal to IBM, the role of WebSphere Application Server will also grow. The evolving portfolio of WebSphere branded products will come to depend on the application server for a growing number of services and capabilities. These could be called base or basic services, but in actuality, they represent capabilities that are functionally beyond what is evident today in WebSphere Application Server version 5.0. Internal to IBM, the strategy is to continue to achieve engineering efficiencies in middleware product development, by focusing efforts on building high-level business value that extends and leverages the functionality of the underlying application server.

Summary

You should now have an initial understanding of what WebSphere Application Server is. We have defined the server configurations:

- WebSphere Application Server Express
- WebSphere Application Server
- WebSphere Application Server Network Deployment
- WebSphere Application Server Extended Deployment
- WebSphere Application Server Enterprise
- WebSphere Application Server for z/OS

We have also looked at the platform concept and the important role of the application server as part of the platform. A quick visit to the past and some projections for the future has also been provided.

Some key points from Chapter 1 include:

- Application Servers are central to solving today's and tomorrow's business problems
- Standards are important to the application server industry and will continue to be so in the future
- Application Servers represent the results of years of product development and standards evolution
- Application Servers provide a distributed operating system concept that encapsulates and abstracts details of individual platforms

With this introduction to WebSphere now in place, let's move ahead to the details.