WebSphere as an e-business server

by D. F. Ferguson
R. Kerth

In this paper, we provide an overview of the technical functionality of WebSphere™ Application Servers and several related products in the WebSphere product family. The paper specifically addresses the product features that are essential to today's e-businesses. We discuss infrastructure services, business-to-consumer and business-to-business scenarios and detail the existing and future support that the WebSphere product family offers in these areas. We also include an extensive list of references for readers who wish to obtain more detailed information on specific aspects of the WebSphere product family that are beyond the scope of this paper.

Today we are witnessing unprecedented demand for e-business solutions such as portals, e-commerce sites, business-to-business commerce solutions, and electronic marketplaces. The quest for intelligent solutions in this area has become a top priority for both technology customers and technology suppliers. From the perspective of customers, exploring the possibilities of the Internet as a global marketplace has rapidly grown to become an integral part of their business strategy. From the perspective of the suppliers, the requirements of these customers have created a strong demand for new products and services.

In the past, many of the simpler e-business solutions have been built upon Internet relationship management products whose prime specialization has been content management and personalization. This type of product was adequate for early Web sites presenting mainly static content. However, the situation has rapidly developed beyond that, and the focus has shifted to more data-driven applications in which customers of a business can place orders, see inventory, book travel, track packages, and do much more. This shift in focus has led, in an extremely short period of time, to a need to rapidly develop secure, robust, and highly scalable enterprise class applications that access existing corporate data and applications and that provide high-end transactional capabilities.

This evolution entails many new challenges, some of which are:

- Simple HyperText Transfer Protocol (HTTP) requests received over the Internet need to trigger complex business tasks on existing enterprise information systems (EISs).
- Guaranteed, reliable, and asynchronous delivery is rapidly gaining importance as an additional transport mechanism that complements the current "best effort" delivery mechanism of the Internet.
- Confidentiality, auditing, and nonrepudiation of messages become critical characteristics of a request.

IBM's strategic initiative to address this market is the Application Framework for e-business (AFeb).[1] Backed by numerous service partners and IBM's traditionally strong middleware, the AFeb comprises a leadership product set that is well-positioned to

0018-8670/01/$5.00 © 2001 IBM

satisfy the critical requirements of customers operating in this rapidly evolving market.

WebSphere* is a family of Web application servers and the foundation of many development efforts, both inside and outside IBM. Together with MQSeries* and DATABASE 2* (DB2*), it is one of the key technology components of the AFeb. This paper provides an overview of some of the important features that are incorporated into WebSphere to address the requirements of a typical e-business.

In the next section of this paper we introduce some of the generic infrastructure services that the WebSphere Application Server offers to support e-business scenarios. The following section then addresses the particular needs of the business-to-consumer scenarios, and the last section provides details about business-to-business scenarios.

Many of the features described throughout the paper offer a complex functionality that cannot be covered in detail in this paper. For this reason, we include references to additional information where appropriate.

Infrastructure services

The WebSphere Application Servers offer a wide range of features that are designed to support the infrastructure requirements of existing and emerging e-businesses. This section provides a brief description of these enabling concepts and services.

The WebSphere programming model. There are three different editions of WebSphere, the *Standard Edition*, the *Advanced Edition*, and the *Enterprise Edition*. Each of these editions addresses a specific set of requirements of e-business customers as follows:

- The Standard Edition provides support for accessing relational databases and for publishing dynamic Web pages through JavaServer Pages** (JSP**) and servlets.
- The Advanced Edition extends the Standard Edition by introducing support for Enterprise JavaBeans** (EJB). The Advanced Edition also improves scalability by adding support for workload management.
- The Enterprise Edition includes the Advanced Edition and adds support for the Common Object Request Broker Architecture** (CORBA**) as well as for CORBA Common Object Services (CORBA COS). The product supports components written in the C++ or Java** programming language and features a tight integration with legacy systems.

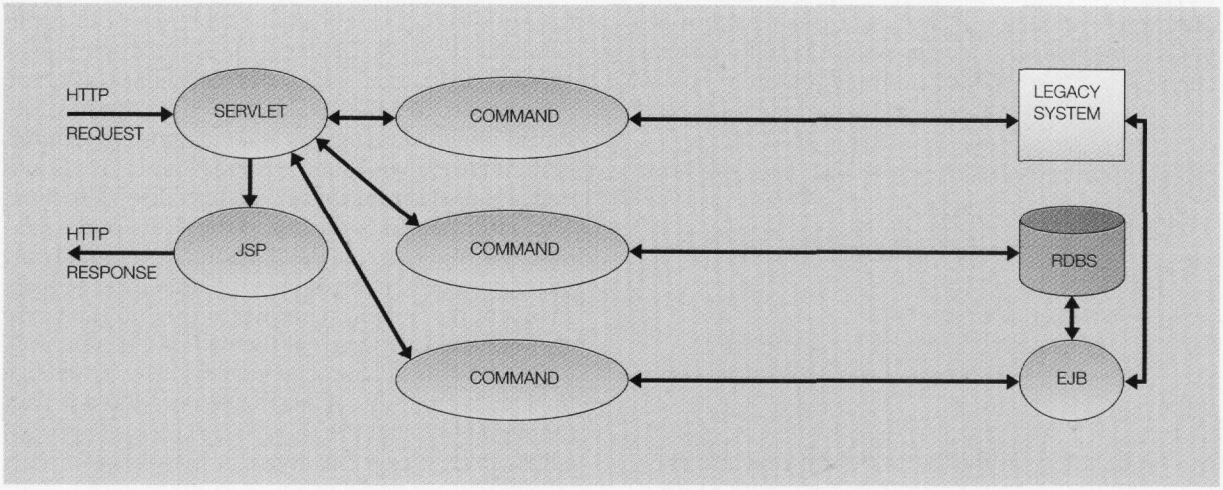
During the last decade, IBM has invested significant efforts in working with different standards bodies to define open standards for software development. The programming model of the WebSphere Application Servers[2,3] is shaped by these efforts across all editions. The WebSphere Standard Edition uses servlets and JSP pages to define the presentation layer and Java Database Connectivity (JDBC**) for database access. The WebSphere Advanced Edition adds support for EJB and will fully support the programming model of the Java 2 Platform Enterprise Edition (J2EE**) in a future release. The WebSphere Enterprise Edition extends J2EE with direct access to advanced CORBA services for greater flexibility and improved interoperability.

The WebSphere programming model incorporates multiple design patterns[4] and best practices for developing scalable, multitiered applications. Some simple examples of such design patterns are depicted in Figure 1. In the WebSphere programming model, a servlet typically functions as a controller that receives requests from the Internet and controls the flow on the server side to coordinate the response to the request. The servlet triggers one or several command objects that access EJB or back-end systems to process the request. When the result is returned, the servlet invokes JSP pages to format the result and to send the response back to the client.

The use of command objects in the above control flow allows the encapsulation of the communication protocol for different back-end systems. This use in turn enables a transparent modification of the protocol or even a switch to a different target system if necessary or desirable. For example, a command accessing a relational database system (RDBS) may use static Structured Query Language (SQL) instead of dynamic SQL as a means to improve performance. It may also switch to using EJB instead of the RDBS to take advantage of the object-oriented infrastructure that is associated with EJB. If the command object is designed correctly, such internal changes do not impact the code that uses the command to retrieve a result from the back-end system.

The use of EJB as an abstraction layer on top of a database or on top of a legacy system is recommended to foster encapsulation and modularized code. Direct access from the presentation layer to

Figure 1 The WebSphere programming model: Interaction flow on an HTTP request



the back-end system is generally discouraged, except in cases where optimum performance is a primary concern. The role of the EJB is to decouple the presentation layer and the database layer by adding a business logic layer in the middle, thus increasing the possibilities of code reuse in all tiers.

Appropriate object-oriented analysis and design methodology[5,6] has to be employed for defining a domain-specific object model of EJB. SessionBeans should represent task-oriented components that drive the interaction with EntityBeans. EntityBeans should represent state-oriented components that drive the interaction with persistent storage. EntityBeans should be accessed through a facade of SessionBeans that define the process context and transaction scope for any access to persistent storage.

Note that the implementation of an EntityBean can take advantage of existing command objects to implement the persistence logic in the case of bean-managed persistence. This allows the EntityBean to reuse existing implementations of persistence protocols. EntityBeans with container-managed persistence do not require explicit persistence logic since the container provides this functionality transparently to them.

JSP pages are used for formatting the layout of the response. Different JSP pages can be invoked by the servlet to communicate different results back to the client. For example, an error would be displayed by a different single JSP than would a successful request. Also, personalized responses can be implemented through appropriate JSP pages, using the mechanisms detailed in the later subsection on personalization.

Figure 1 is obviously an instance of the model-view-controller (MVC) and the command design patterns. The use of the facade design pattern for accessing EJB is recommended, although it is not detailed in Figure 1. In terms of the MVC pattern, the servlet assumes the functionality of the controller, the JSP functions as the view, and various back-end systems, preferably through an abstraction layer of EJB, represent the models. Commands are used for an encapsulation of low-level implementation details. SessionBeans are employed as facades for EntityBeans.

It is important to emphasize that the use of these design patterns is not an abstract exercise to satisfy purely theoretical design requirements. Rather, the patterns structure the code into different functional modules, essential for the maintainability of the code and for the scalability of the system. In fact, the MVC and the command patterns allow independent development of the respective components by teams with appropriate skill sets. This approach speeds up the development of the individual components and allows the rapid incorporation of new technologies when required. Furthermore, the patterns also allow a developer to specialize in one type of component and to follow the technical evolution of this

component very closely. This aspect is crucial for managing the development process and for defining the skill sets that are required in multitiered application development. The facade design pattern enhances scalability by reducing network traffic between client and server. In fact, multiple remote requests

Tools of the WebSphere product family are being structured around a perspective-based development paradigm.

can be bundled on the client before they are submitted to the server. A SessionBean would receive such a bundled request through a task-specific interface. EntityBeans do not (and should not) offer such task-specific interfaces and would require multiple method invocations for transmitting the same data. They should therefore not be invoked directly from the client.

The WebSphere programming model is documented with the different editions of WebSphere and contains many more design patterns and best practices along the general guidelines summarized above.

Tool support. The WebSphere product family supports the programming model described in the previous subsection not only in the server run times but also through the development tools.

WebSphere Studio[7] is a tool aimed at the development of the presentation layer of a multitiered application. It allows the user to generate a set of interrelated servlets, JSP pages, and commands that follow the interaction flow described in Figure 1. If desired, WebSphere Studio can also generate the servlet code with session management logic. Sessions allow a developer to maintain state on the server between HTTP requests, thus adding to the capabilities of the inherently stateless HTTP protocol.

VisualAge* for Java[8] is a tool for the development of different tiers of an application, including parts of the presentation, the business logic, and the data access tier. In this tool, the generation of complex commands for accessing different types of legacy systems is fully supported through SmartGuides (wizards). The generated commands are JavaBeans** and can subsequently be used for a visual composition of servlets. VisualAge also provides support for EJB through a dedicated development environment that incorporates many EJB-specific operations. Additional features that extend and enhance the core EJB standard have been added to VisualAge: the tool can automatically generate and maintain associations between EJB. It also supports inheritance of EJB and performance optimizations such as dirty detection. The latter feature indicates to the WebSphere runtime environment that database updates should only be executed when the data in memory have actually changed; the default behavior of the run time is to always execute updates at the end of a transaction. Note that dirty detection as generated by VisualAge for Java is a purely declarative option that does not require any changes to the EJB code. This approach allows an EJB that is developed in VisualAge for Java to execute unmodified in run-time environments other than WebSphere.

The EJB development environment in VisualAge is completed by an instance of the WebSphere Advanced Edition that executes inside VisualAge. The integration of the application server into the development environment facilitates debugging and unit testing of EJB. In fact, the deployment step in the server run time becomes trivial and can be executed in VisualAge through a simple context menu. A test client to drive requests against the new component is generated automatically. The VisualAge debugger supports stepping through the server-side code, allowing the developer to observe the behavior of the tested component step by step in the server runtime environment. This mode of operation is particularly interesting since the same run time can actually be used in a production environment as a stand-alone version of the WebSphere Application Server. Using the same run time avoids behavioral differences of the component at development time and at production time.

To address the increasing complexity of application development, and consequently of the tools used for this development, IBM has started to structure the different tools of the WebSphere product family around a perspective-based development paradigm. This paradigm allows developers to assume a certain perspective and to customize the tool for use with specific component types. By exposing only features that are essential for the development of a given component type and hiding most of the other features, the developer can focus on learning only the parts

of the tool that are relevant for the component type, enabling him or her to become productive more rapidly.

Pervasive devices. The definition of Wireless Application Protocol (WAP) and the Wireless Markup Language (WML) will lead to widespread use of pervasive devices[9] to access services and information on the Internet. Web servers and application servers need to support these new protocols and formats in order to serve requests from such clients.

Content providers on the Internet often do not wish to develop their content in several formats, with a separate version for each output device. Instead, they prefer to develop the content only once, and then modify it dynamically at run time to match the specific format required by different output devices. The process of translating existing content from one format into another at run time is referred to as *transcoding*.[10] Another paper in this issue discusses transcoding and describes IBM's technology on the subject in greater detail.[11]

The model-view-controller pattern depicted in Figure 1 is one of the key enablers for the pervasive device support provided by WebSphere. The separation of concerns inherent in MVC allows an application developer to address the input/output requirements of the huge variety of pervasive devices relatively easily, and in a very targeted fashion. In fact, since the notion of a view is an integral part of a WebSphere application, it is easy to modify the application to customize the view for a specific target device. Note that the controller performs the view selection dynamically at run time, which allows it to take into account the characteristics of the device that has submitted the request. It is important to mention in this context that views (JSP pages) are not limited to outputting HyperText Markup Language (HTML), even though it is currently the most widely used format; they can also be used to generate other markup languages. In particular, JSP pages can produce generic markup information in Extensible Markup Language (XML), or more specific markup in one of the XML dialects such as WML or VoiceML.

Workload management. On the list of nonfunctional requirements, performance and scalability of a Web site are a primary concern for all customers. The WebSphere Application Servers offer sophisticated, patented workload management (WLM)[12] to address these issues.

The WebSphere Administrative Console supports the notion of a *clone* that can be configured to execute on the same or on a different node on the network. The set of all clones is called a (*server*) *cluster*. All clones in a cluster have the same functionality. They can thus be used interchangeably to service requests. If one of the clones fails, the remaining clones can continue to service requests. In the simplest case, this mechanism does not provide completely transparent failover between clones since the HTTP session state in a clone may be lost if the clone fails. WebSphere allows the session state to be explicitly protected, as described below, to address this issue.

Inside a server cluster, the distribution of requests is policy-driven. Incoming requests are directed to one of the clones, based on the rules specified by the currently active WLM policy (Figure 2).

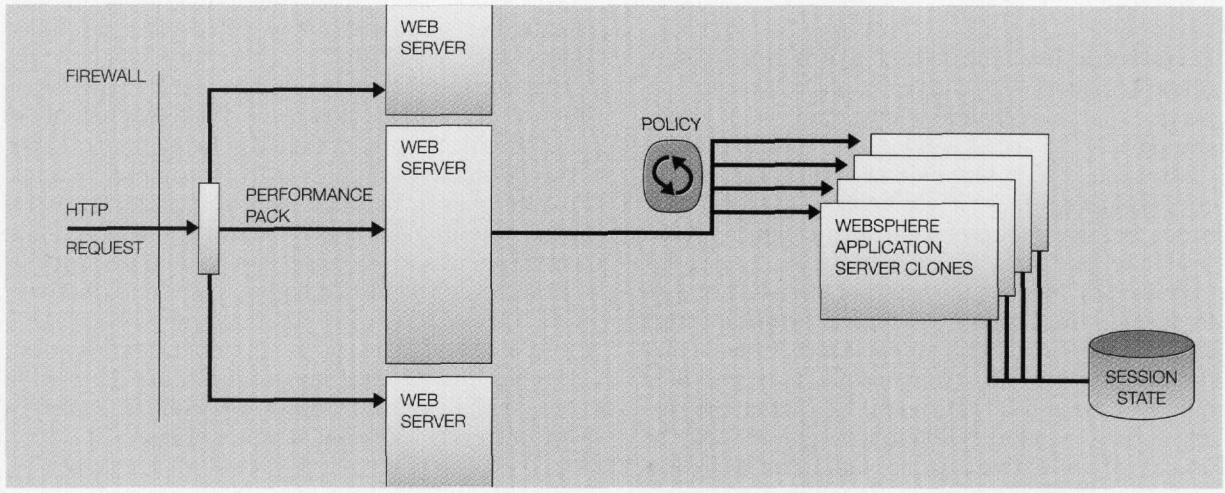
The routing of the requests is subject to general consistency rules. WLM needs to preserve session and transaction affinity, i.e., all requests inside a given session or transaction need to be routed to a clone that has access to the previously defined session or transaction state. Typically, this affinity is achieved by routing all these requests to the same physical clone.

Stateless requests do not need to pay attention to affinity issues; they are directed to an arbitrary clone that is chosen according to the currently active workload policy. Potentially, this clone could be different for each request.

To protect the session state associated with an HTTP request in the case of failures, it can be made to persist in a database. This improves the failover characteristics of the application because the session state can be recovered from the database in case of a clone failure. If an application is configured to take advantage of this possibility, the session state can transparently be shared between the different clones in cluster. Using a database for session persistence is also essential to ensure transactional semantics for updates to the session state. It guarantees that the session state remains valid even if the process using session persistence dies while updating the session state. Other approaches to session state protection in case of a failure, e.g., session state replication in memory, do not offer this quality of service and can lead to session state corruption.

On the OS/390* platform, the WLM facilities for WebSphere Enterprise Edition are implemented us-

Figure 2 Workload management in the WebSphere Application Server



ing the WLM component that is part of the operating system. This implementation provides a tight integration with existing system management tools for this platform.

The WebSphere product family includes the (optional) WebSphere Performance Pack.[13] This product complements the WLM capabilities of the WebSphere Application Servers by providing an Internet Protocol sprayer in front of a cluster of Web servers. It allows HTTP requests to be routed to different Web server instances even before they enter a WebSphere domain and enables extensive horizontal scaling. It is important to point out that the performance pack also supports content-based routing, i.e., it detects the session context of an HTTP request and guarantees session affinity. Other advanced routing options for the performance pack include rule-based routing, weighted routing, and automatic node failure detection.

Caching. Many users are concerned with the overall scalability of their information technology (IT) infrastructure. When investigating the causes for this concern at typical business-to-consumer sites, IBM has found that users are generally satisfied with the scalability of their core business logic executing on large back-end systems. Their concerns are typically focused on the presentation layer because this layer is a fairly new part of their IT infrastructure, it needs to respond to a steadily growing demand, and it has to handle increasingly complex presentation logic. The back-end servers, in contrast, are technically more sophisticated and tend to accommodate growth more easily. Also, traffic on the back-end machines does not increase as sharply as on the front end since not every Web request leads to a business transaction.

WebSphere specifically addresses performance and scalability of the presentation layer by incorporating advanced caching algorithms. The page fragment cache[14] enables caching of partial pages, thus avoiding the overhead of re-executing parts of the presentation logic on subsequent requests. The main advantage of the page fragment cache is to make it possible to selectively cache and refresh fragments of a page rather than entire pages. Entire page caches are widely used in proxy servers and edge servers on the Internet today; however, they fall short of providing the degree of flexibility that is required by numerous Web sites serving dynamic pages.

Caching page fragments is particularly important in the context of a business-to-consumer Web site because it enables flexible integration of personalization into individual pages. Personalization typically leads to a large number of different versions of a page that simply differ in some personalized aspect. Personalization makes it difficult to cache a page as an entire unit because different users have to see different versions of the page. The page fragment cache resolves this issue by allowing the system to cache only those parts of a page that do not change between requests. Thus, even for personalized pages, some degree of caching becomes possible.

Web servers also attempt to minimize load on the presentation layer by collaborating with caching servers across the Internet. These servers are used to deliver static content and are located geographically close to the end user. The WebSphere Application Server cooperates transparently with these servers; this does not require additional functionality on behalf of WebSphere because the caching network uses standard HTML links to redirect client requests to a geographically suitable cache server instance.

Security. A critical part of any system that is connected to an open network like the Internet is a sound security architecture. There are two complementary aspects to this architecture: application-level security and network-level security. In this paper, we focus on the application-level security that is tightly integrated with the WebSphere Application Servers.[15,16] Network-level security can be achieved by careful system administration and should be designed to have a minimal impact on the application itself.

The WebSphere security architecture is based on the following, interdependent notions:

- A *principal* represents an individual user in the system.
- A *resource* is a component that is to be protected through the security architecture. This component can be an HTML file, a single JSP or a servlet, or an EJB. The former three resources are called Web resources; an EJB is often referred to as an object resource in this context.
- An *application* is a set of resources that together deliver a certain functionality.
- A *permission* allows a principal to access a resource inside an application.

WebSphere uses these notions to address the following areas:

- *Authentication* aims at identifying the user through different mechanisms and associating a principal with the user. WebSphere supports a mechanism based on user name and password as well as a credential-based mechanism for authentication; the latter includes digital certificates. Authentication in WebSphere is policy-driven and allows the system administrator to define different qualities of service for authentication. For example, the system administrator may choose to accept user names and passwords only if they are submitted through a secure sockets layer (SSL) channel. The user information accessed during authentication can be extracted from the user repository maintained by the operating system or from a repository accessible through the Lightweight Directory Access Protocol (LDAP).

- *Authorization* checks whether a principal has the permission to invoke a specific method on a resource. Each principal may be granted several permissions. If at least one of these permissions allows the invocation, the invocation succeeds; otherwise it fails with a security exception. By default, such failures are logged in the system log files of WebSphere to help uncover attacks on the Web site.

  Authorization in WebSphere is based on the capability model rather than on access control lists (ACLs). The difference between these two approaches is that the capability model associates permissions with principals, whereas ACLs associate permissions with resources. The capability model is often easier to administer than ACLs because principals change their set of permissions more often than resources do. This kind of modification translates more naturally into the capability model than into an ACL.

- *Delegation* ensures that the appropriate security information is propagated with method calls. The details of propagated security information are determined by a delegation policy; they can be based on the identity of the client, the server, or the system.

The WebSphere security architecture is fully supported through the administration console. It includes a graphical user interface that guides the user through all tasks that are required for security administration.

WebSphere supports single sign-on to the WebSphere domain if configured to run on top of an LDAP repository. The mechanism uses HTTP cookies to communicate the security information between HTTP requests. The cookie contains an encrypted and digitally signed credential that authenticates the associated principal to different servers in the WebSphere domain.

Programmatic access to the security architecture is also possible in WebSphere. It allows the developer to implement a custom log-in mechanism and to query the information that has been obtained from a system-driven log-in.

As indicated above, failed authentication and authorization attempts in WebSphere are subject to auditing by default. Additional auditing can be configured to establish accountability for individual actions on the site. If performance is a primary concern, it is also possible to disable the auditing functionality.

In future releases of the WebSphere Application Servers, IBM will integrate the Tivoli SecureWay* product family with the WebSphere security architecture.

Connectors. Connectors are the approach taken by WebSphere for connecting to existing enterprise information systems (EISs). This WebSphere facility is particularly important in that it provides the capability to integrate existing IT assets of an enterprise (programs and data) into WebSphere applications. The integration not only covers simple data access but also allows the WebSphere servers to control various aspects of the EIS infrastructure services.

IBM's Common Connector Framework (CCF)[8,17] defines an architecture for the interfaces that WebSphere programs can use for such an integration. Support for the Java 2 connector architecture will also be integrated into WebSphere in a future release. As some details of the Java 2 connectors are still under review, we will focus on CCF in this paper. In CCF, two key contracts between the client, WebSphere, and the connector guarantee the behavior of a program as follows:

- The *CCF client contract* defines a programming interface between the client and the connector. This interface enables a client to talk to different EISs by selecting an appropriate connector, without introducing major changes to the client code. Transformations between EIS data formats and client data formats are supported in a number of different styles.
- The *CCF infrastructure contract* defines a programming interface between the server run time and the connector. This interface allows the server run time to manage various infrastructure services for a CCF connector, e.g., life-cycle management, security, transactions, sessions, and state management. Note that CCF allows different "quality of service" parameters to be communicated to the connector through the infrastructure interfaces. Changes to these parameters do not impact the application code. Examples of such parameters are the transaction and the security context.

Figure 3 illustrates the various elements that combine to make up a connector solution. Some of these elements can be implemented by a third party and are collectively referred to as a CCF connector. Most importantly, the CCF connector contains the implementation of the client contract. Other parts in Figure 3 belong to the WebSphere run time, e.g., the server-side implementations of the infrastructure contract. These parts are implemented by IBM as part of WebSphere.

Some context properties of a connection, e.g., the transaction context, are established when acquiring a connection. A thread context is then employed to propagate this information throughout the WebSphere server run time until the connection is executed, i.e., until it is used to transmit some data to the EIS system. Therefore, a connection must not be shared between different threads; it must remain associated with only one thread during its lifetime to preserve the correct run-time context.

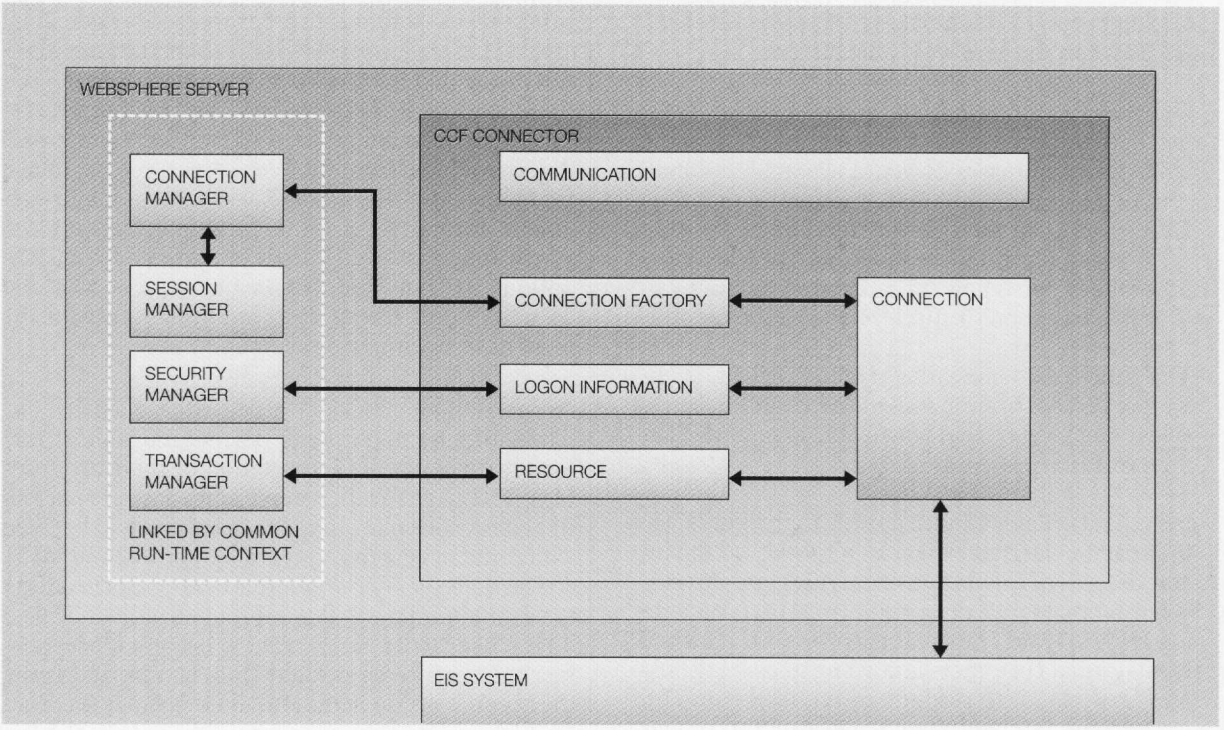
The CCF architecture is fully supported in VisualAge for Java to ease the development of applications using CCF.

Personalization. As Web sites become more sophisticated, personalization[18] plays an increasingly important role in the development of the presentation logic of the site. WebSphere includes flexible and extensible support for personalization.

On an abstract level, there are three ingredients to personalization: a *user model*, a *content model*, and a *matching technology*. The user model captures the properties of a user of the site and makes them available to the run-time environment. The content model captures the information about different types of content. The matching technology associates users with content; it employs algorithms that are based on the attributes defined in the user model and the content model.

The user model in the WebSphere Application Server is accessible through a UserManager class and is represented by instances of a UserProfile class. A default UserProfile class is delivered with WebSphere. It offers a standard set of properties, e.g., name, address, and telephone, and is implemented as an EJB. It is automatically persisted in a database using a default database schema. WebSphere allows the developer to replace or extend the default UserProfile in different ways. Simple extensions would just add new properties to the existing UserProfile

Figure 3 The Common Connector Framework in the WebSphere Application Server



and maintain the existing persistence mechanism. More sophisticated extensions can be used to retrieve the UserProfile information from user repositories on legacy systems and expose them inside the WebSphere Application Servers.

The user model is complemented by a content model. For WebSphere, content can be stored in and accessed from a variety of data sources, e.g., file systems, relational databases, or LDAP directories. The main requirement on data sources is for them to be searchable. The content residing in these data sources is represented to the personalization engine as a set of attributes that contain meta information about the content. Typical attributes are a categorization of the content, keywords, validity dates, and storage location. The meta information is used by the personalization engine to search and retrieve the content at run time.

WebSphere personalization uses a common resource engine to define and manage the user model and content model. The central component of the resource engine is the Hierarchical Resource Framework (HRF) that allows resources to be organized in hierarchies. Nonleaf nodes inside an HRF hierarchy can be thought of as groups of resources; they carry properties that apply to all of their (leaf or nonleaf) children. Leaf nodes represent individual resources. By introducing a hierarchical structure for resources, the HRF enables resource management by groups rather than by instances. The hierarchies also support dynamic registration and lookup of resources.

For example, a Web page designer can insert a group into a page and reserve some layout space for content from that group. At run time, the group is searched for suitable content, taking into account additional parameters from the current request, and the result of the search is displayed in the corresponding layout space.

The matching technology in the WebSphere personalization engine can use filters or rules as follows:

- The filter-based approach allows matching to occur based on simple selection criteria that are obtained directly from the user. Typically, the user

explicitly chooses specific contents that he or she is interested in, e.g., a stock price or a newsletter on a certain topic. The content matching these selection criteria is then inserted into personalized pages when the user visits the site.

- Collaborative filtering is a special kind of filtering. In this approach, general usage patterns in the behavior of all users of a site are analyzed, and some typical users are distinguished as mentors. Other users can then be classified in groups that are associated with these mentors. This classification allows personalization recommendations to be made based on the preferences of the mentor.

- The rule-based matching technology in WebSphere personalization uses the accessible business rules (ABR) engine that is available as part of WebSphere Enterprise Edition.

  In general, rules are more flexible than filters since they can be defined explicitly, allowing for the incorporation of existing business rules into the personalization logic. For example, rule-based matching allows "gold" customers to be distinguished from normal customers according to existing business criteria, or to respond to requests based on the current date and time. Rules can also trigger actions that go beyond simple content selection, e.g., sorting results or sending e-mail.

  ABR permits the high-level formulation of rules, enabling a business analyst without programming skills to define and modify the behavior of rules. ABR will be made fully accessible to users of WebSphere Advanced and Enterprise Editions in a future release. Full accessibility will allow users to extend the use of rules beyond just personalization. At this time, rule editing for ABR will be supported by two versions of a rule editor, one integrated in WebSphere Studio and another one that does not depend on WebSphere Studio. The latter version is provided for the benefit of users who do not have access to an installation of WebSphere Studio.

The HRF-ABR approach offers a flexible programming style; in particular, it is not necessary to modify the page template to achieve personalization. Because the HRF approach allows dynamic registration of resources, new content and new users can transparently be incorporated into an existing template. The different matching technologies in WebSphere personalization are all wrapped into standard JavaBeans to offer a uniform access mechanism. These beans are typically instantiated in servlets or JSP pages and receive some parameters from the UserProfile and from the HTTP request to customize the matching logic. The servlets or JSP pages then extract the result of the matching process from the bean and position it correctly in the overall page layout. The result could be as simple as a reference to a resource that is available from the Web server, e.g., an image file or an audio clip. It could also be a complete document, formatted in the appropriate markup language, that is inserted into the main document. Both options are transparent to the end user, i.e., the end user has no means of detecting which parts of the page have been personalized.

As an example, consider a page that needs to display appropriate sports equipment to customers, taking into account the information of their user profiles. The page would reserve some layout space to this effect, specifying that this space needs to be filled with content from the group called sports equipment. At run time, the user's preferences are extracted from the user hierarchy. Preferences of an individual user do not need to be hard-coded in his or her user profile; they can be associated with, or inferred from, the broader group in the HRF that includes the user. Once the relevant preferences of the user have been determined, the sports equipment group is searched for matching content. The result of the search will reflect the current status of the group and could potentially return content that was not available when the page was built.

Site analysis. Understanding the behavior of a visitor to the site is the key factor for improving usability, acceptance, and responsiveness. In some cases, it may also impact the profitability of the site; for example, sites with an advertisement-based business model strive to maximize their revenues by identifying and classifying their users and offering a targeted advertising campaign to interested parties.

The WebSphere Site Analyzer[19] can be used for extracting the necessary information from the server log files and analyzing it. Site analysis offers three different modules:

- Content analysis helps the system administrator to understand the interdependencies between the individual components of the Web site. It visualizes the entire site and identifies broken links. It also signals excessively large page and HTML syn-

tax errors. Content analysis is implemented using a Web crawler that examines the Web site.
- User analysis aims at classifying the behavior of visitors to the Web site. The available functionality in this module covers statistics for individual pages, including dynamically generated pages, parameter tracing for dynamic pages, (Web) origin determination, and site entry and exit analysis. The latter data allow individual pages to be classified in the context of their execution.
- The reporting module allows the system administrator to extract data from the other two modules that respond to common questions. Reports can be scheduled for certain times and are then executed automatically. The output can be formatted in HTML and XML and may include a visual presentation of the data using various kinds of charts.

Business-to-consumer scenarios

In a business-to-consumer scenario, the typical business transaction is an interaction with a consumer across the Internet. There are two broad types of business-to-consumer interactions: portals and on-line shopping sites. A portal provides personalized services and information to customers and has an advertisement-based revenue model. A shopping site offers goods in a browseable catalog and sells those goods on line; its revenue stems mainly from sales conducted on the site.

Note that some Web sites actually fit into both categories. Shopping sites that include advertisements targeted for certain user groups move their business model into the portal space. They can base their advertising on their knowledge of a user's preferences. Portals, in contrast, often rate shopping sites as part of their core services to their users. Including strongly integrated advertisement offerings as part of the rating can actually lead to an on-line shopping functionality as part of the portal. Nevertheless, it is helpful to distinguish between the two types of business-to-consumer sites, since they have distinct business models that lead to specific technical requirements for different functions.

Business-to-consumer sites have obviously the same fundamental requirements as many other Web sites in terms of scalability, security, personalization, etc. The features that the WebSphere product family offers in these areas were described in the previous sections. In this section, we focus on additional features that WebSphere offers on top of these infrastructure services.

On-line shopping. The WebSphere product family covers trading-oriented functionality with the WebSphere Commerce Suite (WCS).[20] It addresses the needs of a typical business-to-consumer retail site and runs on top of the Advanced Edition of the WebSphere Application Server. The Commerce Suite is also integrated with WebSphere tooling: Specific extensions to the tools allow the user to easily develop applications for the Commerce Suite.

The WebSphere Commerce Suite supports servlets and JSP pages as the fundamental building blocks for the Web site. The overall interaction flow in the Commerce Suite follows the pattern described in Figure 1, i.e., servlets act as a controller, JSP pages define the view, and commands are used to drive the interaction with persistent storage. The resulting structure of a Commerce Suite application supports rapid modifications of the presentation layer by exchanging the views, thus allowing promotions and other daily changes to be introduced relatively easily.

The WebSphere Commerce Suite introduces a comprehensive set of high-level concepts to model an on-line store and to support the different operations that a user can perform in the store. It features support for cross- and up-selling of products (i.e., the ability to link to related products or to better versions of a product on a page displaying a base product), accessory selling, and product substitutions within predefined categories. Personalized views of the store can be defined. The Commerce Suite also includes support for auctions that allows a developer of an on-line store to offer a dynamic pricing mechanism to visitors of the site. Finally, the WebSphere Payment Server, bundled with the Commerce Suite, offers a complete solution for the payment processing requirements of a retail Web site. The Payment Server supports the Secure Electronic Transaction (SET) protocol for credit card processing and allows support for other protocols to be implemented as pluggable *cassettes*. Cassettes are components that encapsulate the details of a payment protocol while taking advantage of the general services of the payment server.

Advanced dynamic trading mechanisms are supported by the Marketplace Edition of the WebSphere Commerce Suite.[21] It supports catalog aggregation and different types of (forward and reverse) auctions. The Marketplace Edition goes beyond simple auctions by introducing matchmaking algorithms for more complex constellations of products, quan-

tity, and price that can be part of an exchange. It also includes the infrastructure to send e-mail notifications to different participants in the electronic marketplace at certain stages of an exchange. Therefore, it starts to address some of the needs of the typical business-to-business scenario as described in the next section.

Portals. A *portal* represents an evolution of the traditional functionality of the Internet: information distribution. However, some Web sites offer increasingly sophisticated services, which has solicited the introduction of a dedicated name for such sites. Portals typically address some or all of the following issues:

- Data aggregation from different sources
- Content management
- Intelligent search and data mining
- Access to Web applications
- Personalization
- Membership administration with anonymous user support
- Access control with single sign-on
- Transcoding support
- Configurable notification mechanisms (agents)
- Integrated desktops with support for instant messaging

Different kinds of portals have developed over time. *Personal portals* provide individuals with general information and productivity tools. *Community portals* offer information to a particular group of people. *Corporate portals* are basically a community portal for the employees of a company. *Application service provider portals* concentrate on access to hosted applications, augmented by some value-added services. Finally, *interenterprise portals* give other companies access to the goods and services offered by the company that runs the portal.

The common service that all these portals provide to their users is an integrated, personalized view of the information available in a specific domain. This view is usually accessible over the Internet from anywhere in the world.

Note that a portal application requires support in all tiers: In the data management tier, advanced search and categorization are required to provide an integrated view over a wide range of data types, typically available from a variety of sources. In the middle tier, customizable business logic must implement support for personalization, security, notification, and other portal services. Finally, in the presentation tier, different navigational styles as well as several output devices need to be supported, depending on the preferences of the user. Thus, support for portals cannot be limited to one tier. It must span the entire software stack involved in the implementation.

IBM addresses the requirements for portals in all tiers:

- The Enterprise Information Portal (EIP)[22] initiative is IBM's product offering for the data management tier. EIP offers support for structured and unstructured data types and allows searches across federated back ends. Categorization of search results is possible through automated meta-data extraction.

- The WebSphere Portal Server[23] provides the required support for portals in the business logic and in the presentation layer. It includes a set of EJB and commands to manage users, resources, and events, as well as a set of servlets or JSP pages to customize the presentation layer.

  The Portal Server is tightly integrated with existing members of the product family, from a tool perspective as well as from a run-time perspective. Specifically, the portal architecture exploits the personalization, transcoding, and content management functionality of WebSphere as described in other sections of this paper. Tooling is provided by wizard-based extensions to WebSphere Studio.

A portal solution with a very specific target audience is the Raven knowledge portal from Lotus.[24] This solution addresses the collaboration requirements of large enterprises through a tight integration with Lotus Domino**. Its collaborative design is built on the concepts of the theme of "People, Places, and Things" that allow users to effectively share knowledge across the network.

Content management. Content management (CM) addresses the need to manage the content of a business-to-consumer site through different stages of the publishing process: preparation, validation, authorization, publication, intelligent search, and rating of search results. The management chain can be fairly complex and often requires specialized solutions to address the needs of a particular business-to-consumer site. For example, news portals need to interface to networks that continuously publish infor-

mation; they need to select the information that is relevant to them and integrate the selection into the existing Web site. Updates occur continuously, but the volume of data that is to be handled after content selection is relatively low. On the other extreme, an on-line shopping site needs to extract the required information from pre-existing catalogs, possibly available in different formats, and present it in a standard template to the user of the Web site. Updates of the core catalog data do not occur frequently, but the volume of product data that is to be managed can be extremely high.

Business-to-consumer sites have a strong need for CM since it represents the core of their value proposition to the customer. In fact, business-to-consumer sites typically do not provide content that is unique to the site but rather content that is publicly available on the Internet or from product manufacturers. Therefore, these sites need to distinguish themselves from their competition by offering a well-structured view of that content.

IBM's role in this area is mainly to be an infrastructure provider. Many of the customized tools used to address specific business-to-consumer scenarios are developed by IBM business partners that build their products on top of this infrastructure. Some of IBM's product offerings in this area are part of the DB2 product family and are beyond the scope of this paper. Additional information on these products is available from Reference 25.

Other product offerings in this area are part of the WebSphere product family. For example, the WebSphere Catalog Architect[26] supports the creation and administration of product catalogs for shopping sites that run on top of WebSphere Commerce Suite. The Catalog Architect offers a tight integration with the run-time structure of the commerce suite and focuses on those parts of CM that are directly related to the server run time, namely content modeling, publishing, and search. The other areas of CM are addressed through offerings from IBM business partners that can integrate with the Catalog Architect to access these run-time services.

Because of the tight integration of the Catalog Architect and Commerce Suite, the user of the Catalog Architect can define advanced catalog features such as personalization, intelligent catalog search, and merchandising. Catalog entries can be defined in an object-oriented view of the data, enabling attribute inheritance for similar catalog items and selective overwriting of attributes. The Catalog Architect verifies the catalog information and helps the user to catch errors during the data entry process. It then inserts the gathered information into the database that is used by the Commerce Suite. Thus, it shields the user from most of the low-level details of data entry.

In addition to the Catalog Architect, the WebSphere product family provides additional, more generic functionality in the CM space. IBM offers a distributed authoring environment called *WebSphere content management* that enables distributed CM over the Web. The environment consists of the following elements:

- A *presentation layer* that contains a set of servlets and JSP pages, executing in WebSphere, that send and receive requests through the Web-Based Distributed Authoring and Versioning (WebDaV) Protocol
- A *CM server* that executes the business logic for common tasks such as content contribution, version management, preview, and content publishing
- An *access control server* that is responsible for authenticating and authorizing the user to execute certain tasks in the system
- A *repository* that persists different types of content, referred to as assets in this context. Certain assets are predefined, e.g., HTML or XML documents, style sheets, and images. However, it is also possible to extend the repository with new asset types. The functionality of the repository is based on existing CM products from the DB2 product family.
- A *workflow engine* that ensures that the overall CM process is correctly executed. The CM process is user-configurable and uses the organizational hierarchy of the user's company to define the rules for content routing. This workflow functionality is provided by Domino Workflow.

The WebSphere content management environment is designed to support different content languages. Furthermore, it integrates with the personalization framework of WebSphere described previously, i.e., it allows the user to publish content into certain groups of the HRF.

Business-to-business scenarios

Business-to-business (B2B) scenarios describe the interaction between two or more companies when conducting business transactions over the Internet. In

general, some parts of this interaction can be automated, whereas other parts may require human intervention. However, the B2B interaction does not involve a consumer. This characteristic is the main difference from the business-to-consumer scenarios described in the previous section.

The most significant new challenge for B2B scenarios is the tight integration between Web technology, workflow, and messaging. This integration enables complex, distributed, and asynchronous tasks to be launched and completed with a minimum of human intervention.

Another feature of B2B scenarios, above and beyond many business-to-consumer scenarios, is the use of XML as the *lingua franca* of the information exchange. Since XML is versatile enough to handle many different kinds of information, the use of XML alone is not sufficient to communicate with other systems. Standardized formats, or some support for format translation, are required when exchanging messages with a remote system.

The rest of this section outlines how these additional challenges are addressed by the specific B2B features and functions of the WebSphere product family and other IBM product families. The infrastructure topics already covered in the previous sections represent a very important foundation for these extended features and functions and indeed for the overall successful development of any B2B solution.

Enterprise application integration. Enterprise application integration (EAI) is the key element for all B2B scenarios since existing applications need to cooperate across enterprise boundaries. It is generally not an option for a company to replace the existing infrastructure just to satisfy emerging interoperability requirements with other companies. Rather, the existing infrastructure needs to be carefully extended or aggregated to accommodate more flexible input and output channels.

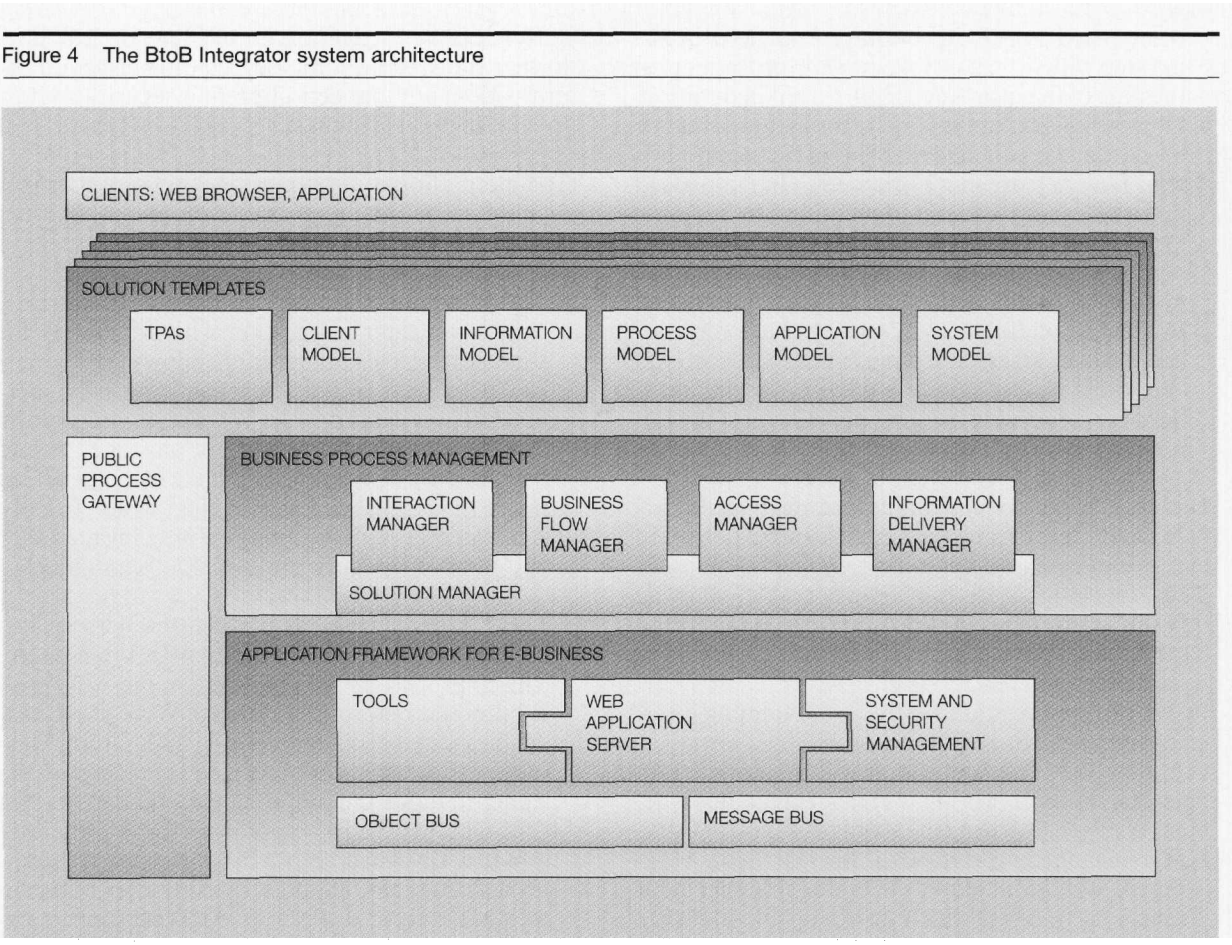
The architecture of the WebSphere BtoB Integrator[27,28] provides extensive support for this kind of scenario. The BtoB Integrator is based on a comprehensive set of IBM products that includes MQSeries, MQSeries Workflow, MQSeries Integrator, SecureWay Policy Director, and the WebSphere Application Server. It offers an integrated view of the feature set of the underlying technology.

The BtoB Integrator addresses two main functional requirements: *collaboration* and *integration*. The collaboration functionality enables a workflow to span multiple applications as well as enterprises to execute a given business task. The integration functionality gives access to data in different back-end systems.

The architecture of the BtoB Integrator introduces several building blocks to address these functional requirements, as shown in Figure 4.

- The business flow manager executes business processes as a multilevel business transaction, i.e., collaborative, state-driven workflow uses short transactions to update the persistent state of an application. Compensation transactions can be employed to undo such updates if the business transaction needs to be rolled back at a later stage. The business flow manager is implemented using capabilities of both the MQ Workflow and WebSphere Application Server. The business flow manager features specialized EJB, the so-called adaptive documents (ADOCs), that provide a means to aggregate content from multiple applications and data sources to execute a given task or activity in a business process. Task and document controllers associated with an ADOC define its multilevel transaction behavior.
- The interaction manager renders role-based and business-process-sensitive executable content and provides an integrated user experience in cooperation with the access manager and the business flow manager. The interaction manager maintains the shared session context in the case of user access through the browser and provides a mechanism to integrate presentation layers from various applications that follow the WebSphere programming model.
- The access manager defines and executes the organization and trust model. It is implemented using Java Naming and Directory Interface** (JNDI) to access the Tivoli Policy Director and LDAP directory. It can support both the capability and the access-control-based authorization mechanisms, and in cooperation with components in the WebSphere Performance Pack (notably the Web Traffic Express) can support single sign-on for multiple back-end systems.
- For a tight integration with existing back-end systems, the BtoB Integrator architecture introduces the information delivery manager that is responsible for sending and receiving requests to and from external systems. This component is realized as a distributed architecture, with adapters bridging the disparate application programming interfaces

Figure 4 The BtoB Integrator system architecture



(APIs) of legacy applications. The information delivery manager uses the Business Object Documents of the Open Application Group[29] as the messaging standard between participating applications, thus defining a common data exchange format. MQSeries is employed in the transport layer of the information delivery manager, and the MQSeries Integrator can (optionally) be used for message routing and transformation.

- The public process gateway, also referred to as the BtoB Integrator gateway, provides a mechanism to interchange messages between enterprises using multiple business, delivery, and transport protocols.
- Finally, the solution manager provides application-level logging, generic reliability and serviceability functionality, and support for exception handling. It also manages a system registry consisting of relevant configuration parameters that are stored in LDAP.

Workflow. The formalized process modeling functions of sophisticated workflow products offer an efficient and robust mechanism for specifying business transactions. Once modeled as workflows, they depend on workflow engines to control the overall execution of the transaction. Human-controlled action requests may have to be launched at particular points in a process, and processing must be resumed or rerouted depending on the result of the users' actions.

Domino Workflow and MQ Workflow are IBM's product offerings in this area. These two products have the following different and complementary strengths in their implementations of workflow management:

- MQ Workflow is a transactional workflow engine, focused on high-volume, repetitive processing tasks. In MQ Workflow the activities of each workflow step are typically initiated by the user, but managed by the workflow engine. Each task is ac-

complished by the execution of a single processing step with little control or *ad hoc* processing on the part of the user, and is short-lived in duration.
- Domino Workflow is a nontransactional engine based on the concept that the activities at each workflow step are more oriented to human-based processing and data manipulation. In Domino Workflow, the user typically modifies document contents and workflow data with processes that are not directly under control of the workflow system. Processing of each activity may involve many user actions over an extended period of time. Domino Workflow is also closely linked to the organizational and document routing capabilities of the Lotus Domino application server.

Both systems offer capabilities that allow workflows to be initiated by messages that can be delivered across enterprise and workflow engine boundaries. Thus, the systems are fully interoperable, allowing different parts of business processes to be supported by the most appropriate engine.

IBM is working on unifying the programming interfaces to these two engines by providing a library of workflow EJB. The unification will be modeled on open standards for workflow management. IBM is actively involved in driving the standardization of workflow management interfaces at the Workflow Management Coalition and at the Object Management Group (OMG). Along with other companies, IBM has recently submitted the Workflow Management Facility Specification to the OMG.[30] IBM provides and will continue to provide standards-based access interfaces to the workflow engines in its product lines.

Messaging. Due to the distributed nature of the Internet, some communications need to be performed asynchronously because a reasonable response time cannot be guaranteed. Message brokers have perfected this kind of service over the last few years but are not widely used for sending messages across the Internet. A tight integration between message brokers and object brokers such as the WebSphere Application Servers adds significant capabilities to both technologies.

IBM addresses this field by integrating the MQSeries product family[31] with the WebSphere Application Servers. The integration of messaging with objects is a major focus of IBM's current development activities. Note that the integration of messaging into the managed environment of an object-based application server is technically challenging since it involves receiver activation on message arrival, data format translation, some degree of transaction coordination, and the definition of a security context for the incoming request.

The integration of IBM's message and object broker technology occurs on several levels and at different degrees of sophistication:

- The WebSphere Enterprise Edition has already featured a tight integration with MQSeries for some time. This integration allows MQSeries-backed objects to participate in externally coordinated transactions under WebSphere Enterprise. The developer can put and get messages to and from queues by creating an object in a specialized home that is associated with the queue. The objects can represent incoming or outgoing messages and behave similarly to normal CORBA objects.

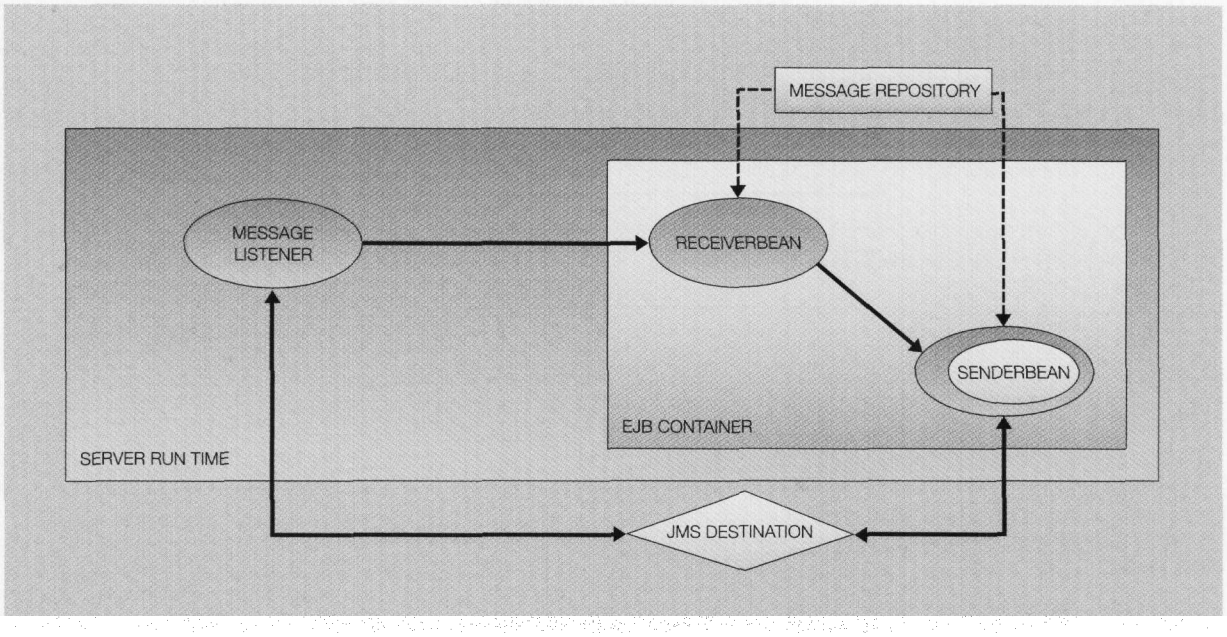
- The CCF architecture, as described earlier, can also be used to integrate MQSeries with WebSphere. This approach can be employed in the Advanced Edition and Enterprise Edition of WebSphere. CCF enables a simple object/message integration. Messages can be sent and received by means of the standard input and output objects that are part of CCF. The MQSeries CCF connector is fully integrated with VisualAge for Java. However, the approach does not support receiver activation on message arrival, and it does not (yet) support external commit coordination across multiple transactional resources.

- The third solution for integrating MQSeries with WebSphere is the use of a Java Message Service (JMS) interface to communicate with MQSeries. A JMS library for accessing MQSeries is available. It includes support for publish and subscribe along with the basic support for queues.

  Version 2.0 of the EJB standard provides the architecture for an integration between EJBs and JMS-based messaging. When this standard is finalized, IBM will incorporate support for it into the WebSphere product family.

Building on top of the third approach described above, IBM supports container-managed messaging (CMM) in WebSphere Enterprise Edition. The goal of CMM is to make message sending and receiving as easy as invoking a method on an object. If compared to the three approaches above, the main difference is that the details of message administration

Figure 5 Container-managed messaging



are encapsulated in helper objects that are generated by the WebSphere tools. Therefore, CMM enables a higher-level, more productive use of the messaging technology.

CMM is based on the concept of a MessageBean which can either be a ReceiverBean or a SenderBean (Figure 5). ReceiverBeans are currently implemented as stateless SessionBeans but will be based on the EJB 2.0 concept of a MessageDrivenBean in a future release. A SenderBean is a standard JavaBean that can be used to transparently send messages. It is typically used by an EJB.

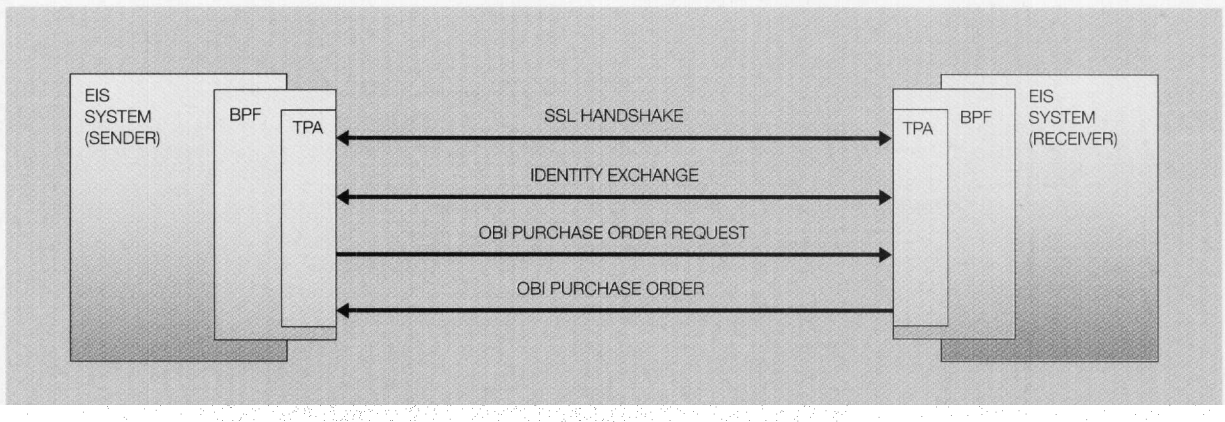
In CMM, a MessageRepository defines a data-mapping layer specifying the details of the format that is used for sending and receiving messages. A MessageBean can receive and send such messages by invoking methods on, and receiving method invocations from, several generated helper objects as follows:

- The MessageListener is responsible for listening for JMS messages from the JMS destination on behalf of a ReceiverBean. When a message arrives, it invokes a ReceiverBean.
- The ReceiverBean is a stateless SessionBean that processes the message. It can apply some data format conversion if required and forward the message to another EJB. A ReceiverBean can be generated by tools, or it can be developed by a user of CMM. In the latter case it can contain business logic.
- The SenderBean is a custom JavaBean that allows transparent message sending, i.e., the message is sent by simply invoking a method. The SenderBean is typically instantiated and used inside an EJB. Again, SenderBeans can be generated by tools or developed manually by a user of CCM.

Transaction and security management are handled declaratively in CMM, allowing some degree of flexibility when tying message sending and receiving into the existing transaction and security layer in the EJB container.

In addition to object integration, another essential property of messaging is message transformation and routing. IBM addresses this area with the MQSeries Integrator.[32] The product supports rule-based routing and formatting of messages in a network of message brokers. The message flow through the network can be defined through a graphical tool that allows

Figure 6 Trading partner agreements in a dynamic exchange (OBI messages for purchase order request and purchase order)



wiring of related messages. Version 2.0 supports the following:

- Message enrichment by merging relational database content into an in-flight message
- Routing decisions based on message headers or content
- Transformation of message content (using a built-in SQL engine)
- Message transmission via MQSeries queues
- Data capture or warehousing from in-flight message to relational database
- Dynamic routing of message using matching table evaluation of registered subscriptions (topic-based and content-based publish or subscribe)

XML support. Many data exchanges across the Internet use XML as a data format. Its flexibility and internal structure position it as an ideal instrument for that purpose. IBM has already enabled most of its products for XML; the next step is to define higher-level business services in XML.

When exchanging XML messages over the Internet, two business partners need to agree on a common data format and a high-level protocol for successfully completing the transaction. For this reason, negotiated trading partner agreements play an important role in the communication flow. These protocols may change from time to time as the business relationship evolves. Thus, it is important to encapsulate the details of the protocol to enable a centralized administration.

IBM has developed a framework that defines the technical infrastructure for this area, the Business-to-Business Protocol Framework (BPF).[33] It provides a generic execution framework for sending and receiving XML messages across the Internet. Another paper in this issue describes in more detail the concepts of BPF and includes information on the status of the current implementations of BPF in IBM products.[34]

BPF uses trading partner agreements (TPAs)[33,35] to capture the details of a contract that connects two business partners on line (Figure 6). TPAs are XML documents that define a wide range of parameters for an exchange over the Internet. These parameters include:

- *Invocation independent properties* specify a name, a period of validity of the TPA, etc.
- *Identification* defines contact information for the participants. Optionally, this section includes information on a third-party arbitrator to settle disputes.
- *Communication* describes the type of communication to be used for that TPA together with appropriate attributes. Examples are HTTP and SMTP, with a uniform resource locator (URL) and an e-mail address attribute, respectively.
- *Security* captures authentication or nonrepudiation protocols as well as required attributes, e.g., certificates or public keys.
- *Data definition* describes the format of the data to be exchanged.
- *Role definition* describes one or several groups of potential participants in this TPA. Roles can be used

to define reusable TPAs, i.e., TPAs that can be applied to any organization that assumes the responsibilities of that role.
- *Action list* contains allowed verbs of an interaction according to the TPA, defining the high-level protocol flow.
- *Sequencing rules* define constraints on the order of the actions.
- *Error-handling* specifies details of conditions that should be treated as an error, e.g., time-outs.

A TPA is defined using the TPA Markup Language (tpaML). The resulting XML file is used to configure the BPF at run time. IBM currently provides an implementation of the Open Buying on the Internet Protocol[36] as a TPA. Other protocols will be made available as well. The tpaML proposal has been submitted to the Organization for the Advancement of Structured Information Standards[37] for standardization.

Together with others in the industry, IBM is currently working on a specification of an XML-based Web services architecture that allows e-businesses to publish and dynamically discover services on the Internet. Central elements of this architecture are the Simple Object Access Protocol (SOAP),[38] the Universal Description, Discovery, and Integration Protocol[39,40] as well as the Universal Definition Language.[41] Recent development work has integrated support for SOAP and other enabling technologies into WebSphere.[42]

As this area is evolving very rapidly, a more detailed description of the technology is likely to be either obvious or obsolete by the time this paper is published. Therefore, we have to refer the reader to one of the references above or to Reference 43 for up-to-date information on IBM's activities in this area.

Conclusion

Today, the WebSphere product family satisfies many of the requirements of a typical e-business Web site. It has grown to become a solid foundation of many development initiatives inside and outside IBM, and its functionality will continue to evolve to satisfy new requirements as they arise.

The WebSphere Application Servers come with a fairly complete portfolio of complementing products that address many aspects of application development and run-time operations. Some of these features are described in this paper. Others, such as a complete set of predefined business components, performance tuning tools, system management, high-performance Java virtual machines, and machine translation were omitted in order not to make the paper unreasonably long. This broad portfolio allows customers to use IBM as a "one-stop-shopping" supplier for the entire software stack that is required to run an e-business.

Together with MQSeries and DB2, WebSphere is one of the strategic server products of IBM. IBM expects the WebSphere Application Servers to become the platform of choice for many e-business initiatives in the industry and positions and funds the product accordingly. Future development efforts in the WebSphere product family will focus on the following two aspects: (1) the evolution of the platform to support new standards as they become available and (2) a tighter integration of WebSphere with IBM and non-IBM products.

Acknowledgments

The authors would like to thank the referees as well as Kumar Bhaskaran, Jim Frank, Vernon Green, Susan Hanis, Brian Martin, Nataraj Nagaratnam, Tony Storey, Robert Will, and Douglas Wilson for their valuable comments.

Cited references

1. The Application Framework for e-Business (AFeb), IBM Corporation, http://www.ibm.com/software/ebusiness/. Also see: G. Flurry and W. Vicknair, "The IBM Application Framework for e-Business," *IBM Systems Journal* **40**, No. 1, 8–24 (this issue, 2001).
2. WebSphere Advanced Documentation Center, IBM Corporation, http://www.ibm.com/software/webservers/appserv/library.html.
3. WebSphere Enterprise Programming Guides, IBM Corporation, http://www.ibm.com/software/webservers/appserv/doc/v30ee/cbpdf/.
4. E. Gamma, J. Vlissides, R. Johnson, and R. Helm, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley Longman, Inc., Reading, MA (1994).
5. G. Booch, *Object Oriented Analysis and Design with Applications, Second Edition*, Benjamin/Cummings Publishing Co., Redwood City, CA (1994).
6. I. Jacobsen, *Object-Oriented Software Engineering—A Use Case Driven Approach*, Addison Wesley Longman, Inc., Wokingham, England (1992).
7. WebSphere Studio, IBM Corporation, http://www.ibm.com/software/webservers/studio/library.html.

8. VisualAge for Java, IBM Corporation, http://www.ibm.com/software/ad/vajava/.
9. Pervasive Computing, IBM Corporation, http://www.ibm.com/pvc/tech/whitepapers.shtml.
10. Transcoding, IBM Corporation, http://www.ibm.com/software/webservers/transcoding/.
11. K. Britton, R. Case, A. Citron, R. Floyd, Y. Li, C. Seekamp, B. Topol, and K. Tracey, "Transcoding: Extending e-business to New Environments," *IBM Systems Journal* **40**, No. 1, 153–178 (2001, this issue).
12. K. Ueno, T. Alcott, J. Blight, J. Dekelver, D. Julin, C. Pfannkach, and T. Shich, "WebSphere Scalability—WLM and Clustering Using WebSphere Application Server Advanced Edition," WebSphere Workload Management, Redbook SG24-6153-00, IBM Corporation, http://www.redbooks.ibm.com.
13. WebSphere Performance Pack, IBM Corporation, http://www.ibm.com/software/webservers/perfpack/.
14. Page Fragment Cache, to be announced, refer to the main WebSphere site for further information, IBM Corporation, http://www.ibm.com/software/webservers/.
15. WebSphere Standard/Advanced 3.02 Security Overview, IBM Corporation, http://www.ibm.com/software/webservers/appserv/library.html.
16. WebSphere Enterprise Security, in the Advanced Programming Guide, IBM Corporation, http://www.ibm.com/software/webservers/appserv/doc/v30ee/cbpdf/.
17. G. Bist and J. Green, "Using Connectors with VisualAge for Java," CCF Connectors, available from the VisualAge Developer Domain at http://www.ibm.com/software/vadd, under Library—Technical Articles—Components.
18. WebSphere Personalization, IBM Corporation, http://www.ibm.com/software/webservers/personalization/.
19. WebSphere Site Analyzer Guide, IBM Corporation, http://www.ibm.com/software/webservers/appserv/library.html.
20. WebSphere Commerce Suite, IBM Corporation, http://www.ibm.com/software/webservers/commerce/servers/.
21. WebSphere Commerce Suite, Marketplace Edition, IBM Corporation, http://www.ibm.com/software/webservers/commerce/wcs_me/.
22. Enterprise Information Portals, IBM Corporation, http://www.ibm.com/software/data/eip/.
23. WebSphere Portal Server, IBM Corporation, http://www.ibm.com/software/webservers/portal/.
24. Knowledge Management, Lotus Development Corporation, http://www.lotus.com/lotus/km.nsf.
25. Data and Data Management, IBM Corporation, http://www.ibm.com/software/data.
26. WebSphere Catalog Architect, IBM Corporation, http://www.ibm.com/software/webservers/commerce/servers/catalogarchitect.html.
27. WebSphere BtoB Integrator, IBM Corporation, http://www.ibm.com/software/webservers/btobintegrator/.
28. J. Frank, "Business-to-Business Electronic Commerce—A Technical White Paper," BtoB Integrator, to be published at http://www.ibm.com/software/webservers/btobintegrator/.
29. Open Application Group, http://www.openapplications.org/.
30. Workflow Management Facility Specification, Version 1.2, Organization Management Group, ftp://ftp.omg.org/pub/docs/formal/00-05-02.pdf.
31. MQSeries, IBM Corporation, http://www.ibm.com/software/ts/mqseries.
32. MQSeries Integrator v2.0 Technical White Paper, available from http://www.ibm.com/software/ts/mqseries/library/ (in the white papers section).
33. L. Ennser, P. Leo, T. Meszaros, and E. Valade, *The XML Files: Using XML for Business-to-Business and Business-to-Consumer Applications*, Business-to-Business Protocol Framework, Redbook SG24-6104-00, IBM Corporation, http://www.redbooks.ibm.com.
34. A. Dan, D. Dias, R. Kearney, T. Lau, T. Nguyen, F. Parr, M. Sachs, and H. Shaikh, "Business-to-Business Integration with tpaML and a Business-to-Business Protocol Framework," *IBM Systems Journal* **40**, No. 1, 68–90 (2001, this issue).
35. Trading Partner Agreements, IBM's submission to OASIS, http://www.xml.org/xmlorg_resources/index.shtml.
36. Open Buying on the Internet, OBI Consortium, http://www.openbuy.org.
37. Organization for the Advancement of Structured Information Standards (OASIS), http://www.xml.org.
38. Simple Object Access Protocol (SOAP) Specification, Version 1.1., IBM Corporation, http://www.ibm.com/software/developer/library/soap/soapv11.html.
39. Universal Description, Discovery, and Integration (UDDI), http://www.uddi.org.
40. Universal Description, Discovery, and Integration (UDDI), IBM site, http://www.ibm.com/services/uddi.
41. Universal Definition Language (UDL), IBM Corporation, http://www.ibm.com/services/udl.
42. Web Services Toolkit, IBM Corporation, http://www.alphaworks.ibm.com/tech/webservicestoolkit.
43. XMLZone, IBM Corporation, http://www.ibm.com/developer/xml/.

Accepted for publication October 18, 2000.

Donald F. Ferguson *IBM Software Group, Route 100, Somers, New York 10589 (electronic mail: dff@us.ibm.com).* Dr. Ferguson received a Ph.D. degree in computer science from Columbia University in 1989. His Ph.D. thesis applied concepts and algorithms from economics to resource management problems in distributed systems and computer networks. This included work on system load balancing, data replication and file placement, and network flow control. He joined IBM in 1987 and has led research and development efforts in the areas of operating system performance, database tuning, scalable transaction processing, object transaction monitors, and Web application servers. Since 1993, he has been the chief technical lead for IBM's WebSphere family of products that provides an infrastructure and programming model for integrating new and pre-existing applications with the Internet, providing business-to-person and business-to-business solutions. Dr. Ferguson holds approximately a dozen granted or pending U.S. patents and has authored more than 24 technical publications. In 1998, he was named a Distinguished Engineer. He is also a member of the IBM Academy of Technology.

Rainer Kerth *IBM Software Group, Route 100, Somers, New York 10589 (electronic mail: rkerth@us.ibm.com).* Dr. Kerth joined the WebSphere Architecture Team in Somers in July 1999. His areas of expertise are object-oriented application development, based on CORBA or EJB/J2EE, and Web application development. He is currently responsible for the documentation of and extensions to the WebSphere programming model. In this role, he has worked on extended transaction models and participates in IBM's review of different standards initiatives, mainly the EJB and the J2EE specifications. Prior to joining the WebSphere team, he worked for several years in IBM's Object Technology Practice in Germany, specializing in application development in customer engagements. Dr. Kerth earned a Ph.D. degree in math-

ematics and foundations of computer science at the University of Paris in 1995. His thesis addressed several fundamental questions relating to the denotational semantics of lambda calculus.