CHAPTER 1

Introduction

THE realm of Web services—software components that are programmatically accessible over standard Internet protocols—is expanding rapidly due to the growing need for application-to-application communication and interoperability. Web services expose a standard interface that is platform and technology independent. By conforming to accepted industry-wide standards, Web services provide a means of communication among software applications running on different platforms and written in different application development languages and that present dynamic context-driven information to the user.

Since its successful introduction in 1999, the J2EE platform has become an integrated standard for implementing and deploying portable and distributed enterprise applications. One significant factor contributing to this success is that the J2EE platform has been designed through an open process, the Java Community Process (JCP). This open process has engaged a range of enterprise computing vendors to ensure that the platform meets the widest possible spectrum of enterprise application requirements. As a result, the J2EE platform addresses the core issues that impede organizations' efforts to maintain a competitive pace in the information economy. Organizations have recognized this and quickly adopted the J2EE platform standard.

The J2EE platform has evolved further to accommodate the growing popularity and use of Web services. The platform has added support for Web service-specific components and technologies. The Web services standards ensure interoperability across platforms and programming languages. Also portability—the ability to migrate applications from one platform to another—is just as important. Significantly, the J2EE platform adds portability to Web services. By combining portability and interoperability, the J2EE platform is a compelling platform

for developing distributed systems. These new J2EE components and technologies meet the requirements of most enterprises: These technologies allow enterprises to expose their existing applications as Web services and to develop and implement new Web services. This Web-service evolution of the J2EE platform has been achieved with the same open process as before involving leading enterprise computing vendors. It has ensured that the J2EE platform is the platform of choice for developing and deploying portable and interoperable Web services and enterprise applications.

Let's look at some of the reasons why this is so. The J2EE platform has added its Web services features to a platform that has already standardized development and deployment of portable enterprise applications. With these new features, the J2EE platform offers enterprises the following benefits:

- Enables existing enterprise applications to be extended as Web services in a manner that is standard, easy, portable, and interoperable
- Helps extend the reach of existing and new enterprise applications to a new set of clients beyond the already wide variety of clients supported by the earlier J2EE platforms
- Enables using commercially proven, robust technologies (such as enterprise bean components, servlets, and so forth) as the backbone for developing new services
- Helps existing enterprise information systems (EISs) to be integrated to end users in a portable way

This book is about developing robust, effective Web services on the Java™ 2 Platform, Enterprise Edition (J2EE™), version 1.4 or later. The book not only describes the technologies used for Web services, but also presents guidelines for using these technologies to design and develop Web services. Many of these guidelines have been formulated from our own Web service development experience, and, as such, we have illustrated the guidelines with real examples when possible.

This first chapter sets the groundwork for the book—it gives you a look at some of the key concepts underlying the architectural details of Web services and it provides motivations for developing Web services-based applications. It highlights the benefits of Web services and explains why it makes sense for distributed applications to use Web services as their underlying architecture.

1.1 What Are Web Services?

At this point, you might be wondering exactly what Web services are and, more importantly, why your enterprise should consider making some of its own functionality available as a Web service. There are numerous definitions given for Web services, ranging from the highly technical ones to simplistic. For example, the World Wide Web Consortium (W3C) organization, which establishes the standards for Web services, defines them as follows: "A Web service is a software system identified by a URI whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML-based messages conveyed by Internet protocols." A simpler definition, and perhaps more useful, might be: "a Web service is a software application, accessible on the Web (or an enterprise's intranet) through a URL, that is accessed by clients using XML-based protocols, such as Simple Object Access Protocol (SOAP) sent over accepted Internet protocols, such as HTTP. Clients access a Web service application through its interfaces and bindings, which are defined using XML artifacts, such as a Web Services Definition Language (WSDL) file."

Web services are a result of the natural evolution of the Web. Initially, the Web consisted of sites that were plain HTML pages. Later, Web applications dynamically generated these same HTML pages. For example, a map Web site initially provided only static links to maps of various cities and locales. Later, this same map Web site became a map Web application that provided driving directions, customized maps, and so forth. Despite their expanded capabilities, Web applications are still limited to the restricted GUI capabilities of their HTML pages—a Web application is usable only through the limited GUI bound to the HTML pages. Web services go beyond this limitation, since they separate the Web site or application (the service) from its HTML GUI. Instead, the service is represented in XML and available via the Web as XML. As a result, the same map Web site can extend its functionality to provide a Web service that other enterprises can use to provide directions to their own office locations, integrate with global position systems, and so forth.

Web services, or simply services, build on knowledge gained from more mature distributed computing environments (such as CORBA and Java Remote Method Invocation) to enable application-to-application communication and interoperability. Web services provide a standardized way for applications to expose their functionality over the Web or communicate with other applications

over a network, regardless of the application's implementation, programming language, or computer platform.

Why should an enterprise consider implementing Web services-based applications? Web services can provide a means for an enterprise to expand its business offerings, increase the efficiency of its business processing, and to improve its customer experience. By including with its own services the offerings of multiple partners, both the enterprise and the partners expand their capabilities and their business base. Not only can Web services help automate business processing, it can streamline interactions with outside services, such as credit card and shipping services. As a result, customers are offered an enriched experience: more options and greater choices, along with more flexibility.

Like any application, Web services-based applications can perform a range of functions. Some may handle only simple requests for information, while others may implement complex business processes and interactions. Whereas browser-based applications are concerned with the representation of data to end users, Web services let clients programmatically not only use the Web to obtain information but also to access these service components and their functionality. Furthermore, applications can incorporate Web service functionality for their own use.

- Perhaps the most important reason for the increased use of Web services—the main force for their widespread adoption—is that Web services promote interoperability across different platforms, systems, and languages.
- Use of Web services is also increasing because it reduces operational costs by enabling organizations to extend and reuse their existing system functionality.

A Web service enables a *service-oriented architecture*, which is an architectural style that promotes software reusability by creating reusable services. Traditional object-oriented architectures promote reusability by reusing classes or objects. However, objects are often too fine grained for effective reuse. Hence, component-oriented architectures emerged that use software components as reusable entities. These components consist of a set of related classes, their resources, and configuration information. Component-oriented architectures remain a powerful way to design software systems; however, they do not address the additional issues arising from current day enterprise environments. Today, enterprise environments are quite complex due to the use of a variety of software and hardware platforms, Internet-based distributed communication, enterprise application integration, and so on. The service-oriented architectures address these issues by

using a service as a reusable entity. The services are typically coarser grained than components, and they focus on the functionality provided by their interfaces. These services communicate with each other and with end-user clients through well-defined and well-known interfaces. The communication can range from a simple passing of messages between the services to a complex scenario where a set of services together coordinate with each other to achieve a common goal. These architectures allow clients, over the network, to invoke a Web service's functionality through the service's exposed interfaces.

In a service-oriented architecture, you have the following:

- A service that implements the business logic and exposes this business logic through well-defined interfaces.
- A registry where the service publishes its interfaces to enable clients to discover the service.
- Clients (including clients that may be services themselves) who discover the service using the registries and access the service directly through the exposed interfaces.

An important advantage of a service-oriented architecture is that it enables development of loosely-coupled applications that can be distributed and are accessible across a network. To enable this architecture, you need the following:

- A mechanism that enables clients to access a service and registry.
- A mechanism to enable different services to register their existence with a registry and a way for clients to look up the registry of available services. Web services are based on an architecture in which the service can be located over the network and its location is transparent, which means that clients may dynamically discover a particular service they wish to use.
- A mechanism for services to expose well-defined interfaces and a way for clients to access those interfaces.

An important point to note is that Web services are strongly backed by virtually all key players in the Web arena. This backing virtually ensures that Web services technology is here to stay, and its acceptance will become more widespread.

Unlike traditional distributed environments, Web services emphasize interoperability. Web services are independent of a particular programming language,

whereas traditional environments tend to be bound to one language or another. Similarly, since they can be easily bound to different transport mechanisms, Web services offer more flexibility in the choice of these mechanisms. Furthermore, unlike traditional environments, Web services are often not bound to particular client or server frameworks. Overall, Web services are better suited to a loosely coupled, coarse-grained set of relationships. Relying on XML gives Web services an additional advantage, since XML makes it possible to use documents across heterogeneous environments.

Web services, by building on existing Web standards, can be used without requiring changes to the Web infrastructure. However, while they may be more firewall friendly than traditional computing environments, Web services tend not to be as efficient in terms of space and time processing.

1.2 Benefits of Web Services

Web services are gaining in popularity because of the benefits they provide. Listed here are some of the key benefits:

- **Interoperability in a heterogeneous environment**—Probably the key benefit of the Web service model is that it permits different distributed services to run on a variety of software platforms and architectures, and allows them to be written in different programming languages. As enterprises develop over time, they add systems and solutions that often require different platforms and frequently don't communicate with each other. Later, perhaps due to a consolidation or the addition of another application, it becomes necessary to tie together this disparate functionality. The greatest strength of Web services is their ability to enable interoperability in a heterogeneous environment. As long as the various systems are enabled for Web services, they can use the services to easily interoperate with each other.
- **Business services through the Web**—An enterprise can use Web services to leverage the advantages of the World Wide Web for its operations. For example, an enterprise might make its product catalog and inventory available to its vendors through a Web service to achieve better supply chain management.
- **Integration with existing systems**—Most enterprises have an enormous amount of data stored in existing enterprise information systems, and the cost to replace these systems is such that discarding these legacy systems may not

be an option. Web services let enterprise application developers reuse and even commoditize these existing information assets. Web services provide developers with standard ways to access middle-tier and back-end services, such as database management systems and transaction monitors, and integrate them with other applications. In addition, because these services are provided consistently, developers do not need to learn new programming models or styles as integration needs expand.

- **Freedom of choice**—Web service standards have opened a large marketplace for tools, products, and technologies. This gives organizations a wide variety of choices, and they can select configurations that best meet their application requirements. Developers can enhance their productivity because, rather than having to develop their own solutions, they can choose from a ready market of off-the-shelf application components. Tools, furthermore, provide the ability to move quickly and easily from one configuration to another as required. Web services also ensure the standardization of tools, so that development tools can adopt new tools, whether from server vendors or third-party tool developers, as needs arise.
- **Support more client types**—Since a main objective of Web services is improving interoperability, exposing existing applications or services as Web services increases their reach to different client types. This occurs regardless of the platform on which the client is based: it doesn't matter if the client is based on the Java or Microsoft platforms or even if it is based on a wireless platform. In short, a Web service can help you extend your applications and services to a rich set of client types.
- **Programming productivity**—To be productive in the information economy requires the ability to develop and deploy applications in a timely fashion. Applications must go quickly from prototype to production and must continue to evolve even after they are placed into production. Productivity is enhanced when application development teams have a standard means to access the services required by multitier applications and standard ways to support a variety of clients. Web services, by creating a common programming standard, help to enhance programming productivity. Prior to the advent of Web services, developers programming in the distributed computing environment have relied on a diverse set of not-always-compatible technologies. Developers have attempted to tie together various diverse back-end systems, such as both custom and standard database management systems and transaction processors, with traditional Web technologies, but have had to deal with a multitude of programming

models. Because Web services introduce a common standard across the Web, vendors, in the interest of staying competitive, are more likely to develop better tools and technologies. These tools and technologies will attract developers because they emphasize increased programming productivity. As a result, the entire industry benefits.

Application development has also been complicated by the requirement that a particular application support a specific type of client or several types simultaneously. Web services, because they promote interoperability, simplify this facet of the application development process.

1.3 Challenges of Web Service Development

There are numerous challenges faced by Web service developers. Web services are a new paradigm, and, as such, they are far from mature. Because they are in their initial stages, Web services technologies are evolving rapidly and sometimes in unexpected directions. No matter what platform a developer chooses to use, there are some specific challenges to developing Web services. For one, technologies and standards are evolving. As a result, developers can expect that these technologies may change as they are extended to provide enhanced Web services support. While such changes may cause some difficulties early on, as time goes on developers should find that using technologies such as those offered with the J2EE 1.4 platform will actually lessen the challenges to Web services development.

It is important to keep these factors in mind when developing Web services.

1.3.1 Evolving Technologies and Products

A key challenge in creating Web services-based solutions today is that Web services are still in their infancy. Web services, because of their emphasis on interoperability across platforms, operating systems, and programming languages, rely on a collection of technologies and various standards and specifications, many of which are still being defined and refined. Additional new standards are currently being defined to enable Web services to realize their full potential.

The emerging Web services technologies and products have a lot of promise and, as standards become formalized and additional vendors sign on, their promise is being fulfilled. More and more products and technologies implementing the Web service specifications have begun to appear, and they are meeting with more success.

Enterprises often use Web services as a means to distribute data or information. In addition, many businesses use services to conduct business transactions. Such business transactions may require a service to access other services; in a sense, to perform a global transaction. Although in the process of being defined, currently there are no universally accepted standards for such global transactions.

Interoperability is a continuous challenge. Web services have already achieved a significant degree of interoperability, but further standards are needed for the sort of widespread interoperability that fulfills the vision of Web services.

An additional challenge with Web services is the coordination of multiple services for processing business logic. Often, what appears to the end user as a single business process is really implemented as a series of stages in a workflow, and each stage of the workflow might be implemented as a separate service. In such cases, all the services must coordinate with each other during the various steps of the business logic processing to achieve the desired goal. Standards are necessary for coordination among services. Such standards are in the process of being defined, but none has yet been universally accepted. Similar to those previously mentioned, various other standards for other areas, such as security and so forth, are currently being defined.

1.3.2 Security

It should come as no surprise to anyone, but security is just as important for Web services as it is for other enterprise applications. In fact, now that applications on the Web open an enterprise's business processes and data to distributed clients, security becomes an even more important factor.

Information systems security, while of utmost importance to IT managers and system architects, must also be handled with the proper balance. It is not only important to protect information assets to maximize their value, but care must be taken to not overprotect these assets to a degree that jeopardizes that very value. Traditionally, IT departments have maintained a relatively high level of control, if not complete control, over the environment of both servers and clients. When information assets are exposed in less-protected environments, such as through Web services, it becomes increasingly important to maintain tight security over the most sensitive assets, while allowing relatively easier access to other assets.

One of the difficulties in handling disparate systems is providing an integrated security model. Security needs to be compatible with existing mechanisms. In cases where customers need to access secure information, the mechanisms need to

maintain high security (and user confidence) while remaining as unobtrusive and transparent as possible.

The key security issues for Web services concern authentication, authorization, and ensuring confidentiality. Web services standards for security are a high priority area for the community, but, as they are still evolving, they remain a work in progress. Chapter 7 examines these issues and provides some interim solutions.

1.3.3 Reliability, Availability, and Scalability

Web services are often about large-scale distributed applications. With these types of applications, the reliability, availability, and scalability of services of the underlying platform and its technologies are important considerations.

Reliability is the aspect of a Web service representing how well it maintains its service and service quality. Often, reliability is measured by the number of failures that occur in a given time period. For Web services, reliability may be more difficult to achieve because of the unreliable nature of the underlying transport, HTTP; HTTP provides only best-effort delivery and does not guarantee packet or in-order delivery. A Web service is considered more reliable the more easily and automatically it can handle changes in use patterns and system configurations.

Availability concerns whether the Web service is present or ready for immediate use. Does a client have to wait for the Web service, or is the client's request immediately handled? In other words, availability represents the probability that a service is available.

Web services that scale effectively can easily handle a large number of client interactions. For such Web services, the platform and technologies must efficiently manage system resources and services (such as database connections and transactions). The platform must also be able to handle XML parsing, which is done to validate a document's contents. XML parsing is a compute-intensive process that significantly affects performance. XML by itself entails a very verbose format that increases payload size significantly when compared to an equivalent binary format.

To achieve reliability, availability, and scalability, Web services should not only be flexible enough to run on any server configuration appropriate to anticipated client volumes, they should also be capable of easily switching configurations when necessary. A platform's support for clustered application deployment environments contributes to achieving many of these goals. It is also helpful when the platform can handle such changes without requiring the applications to be redesigned, recoded, or redeployed.

1.4 Typical Web Service Scenarios

Enterprise applications cover a wide spectrum of scenarios, such as interactions between business partners, supply chain management, inventory management, and even simple services (specialized converters, calculators, and so forth). Web services, when used strategically, can augment an application's functionality, but keep in mind that Web services are not an appropriate solution at every point in an application. While they do provide functional richness and interoperability, Web services may come at the price of increased performance overhead. Choosing when to implement a Web services-based solution comes down to addressing such issues as:

- Interoperability requirements for the enterprise application in a heterogeneous enterprise environment
- Integration requirements for those whose environments contain various enterprise information systems (EISs)
- Types of clients expected to be supported, such as J2EE applications, wireless devices, PDAs, and so forth
- Availability of tools to implement the solution
- Level of sacrifice, in terms of complexity and performance, that can be tolerated to achieve the advantages (interoperability, diverse client reach, and so forth) of Web services

This section looks at a typical enterprise scenario and identifies where a Web services solution might be appropriate. By no means does this section attempt to cover all scenarios suited for Web services. Rather, it tries to give the reader a flavor of the typical scenarios that might benefit from choosing Web services. Rather than continuing in the abstract, let's examine a sample enterprise application and see where in such an application it is beneficial to use Web services. That is, let's see where the use of Web services improves the application's functionality, its ease of use, or its ability to reach all types of clients.

Our sample scenario, called the Java™ Adventure Builder Reference application (referred to as adventure builder), is a fictitious enterprise that sells adventure packages for vacationers. This enterprise is considering going online. To do so, the enterprise needs to create a Web site, and it is considering having that site take advantage of the benefits of Web services. The enterprise is especially interested in using Web services to integrate with its numerous partners.

This enterprise

- Provides customers with a catalog of adventure packages
- Lets the customer build his or her particular trip experience, choosing accommodations, transportation, and scheduling various adventure activities, and finally booking the trip
- Provides customers with additional services, such as enabling them to track the status of their orders.

For example, a customer might choose an adventure on Mt. Kilimanjaro. In addition to selecting accommodations, mode of transport, and preferred dates, the customer also can select from an assortment of adventures, such as mountain climbing, guided hiking tours of the volcanic peaks, a safari, and so forth. To achieve this, the enterprise might have in place non-Web-based applications or operations that do the following:

- Interact with clients who want to book adventures. There might be toll-free telephone numbers available to clients.
- Receive orders from clients and process the orders. Orders pass through a workflow, such as a workflow engine, during their various stages for completion.
- Keep track of client preferences, plus update clients regarding the status of an order. Customer relations management (CRM) is one such means for tracking clients.
- Verify and obtain approval for client payment.
- Interact with business partners—airlines, hotels, adventure or activity providers—so that the enterprise can provide a complete adventure package to the client

When the developers of the adventure builder enterprise take it online, they implement individual modules for each function. (See Figure 1.1.) Let us see where Web services might fit in and where they might provide the enterprise with advantages in terms of integration and interoperability.

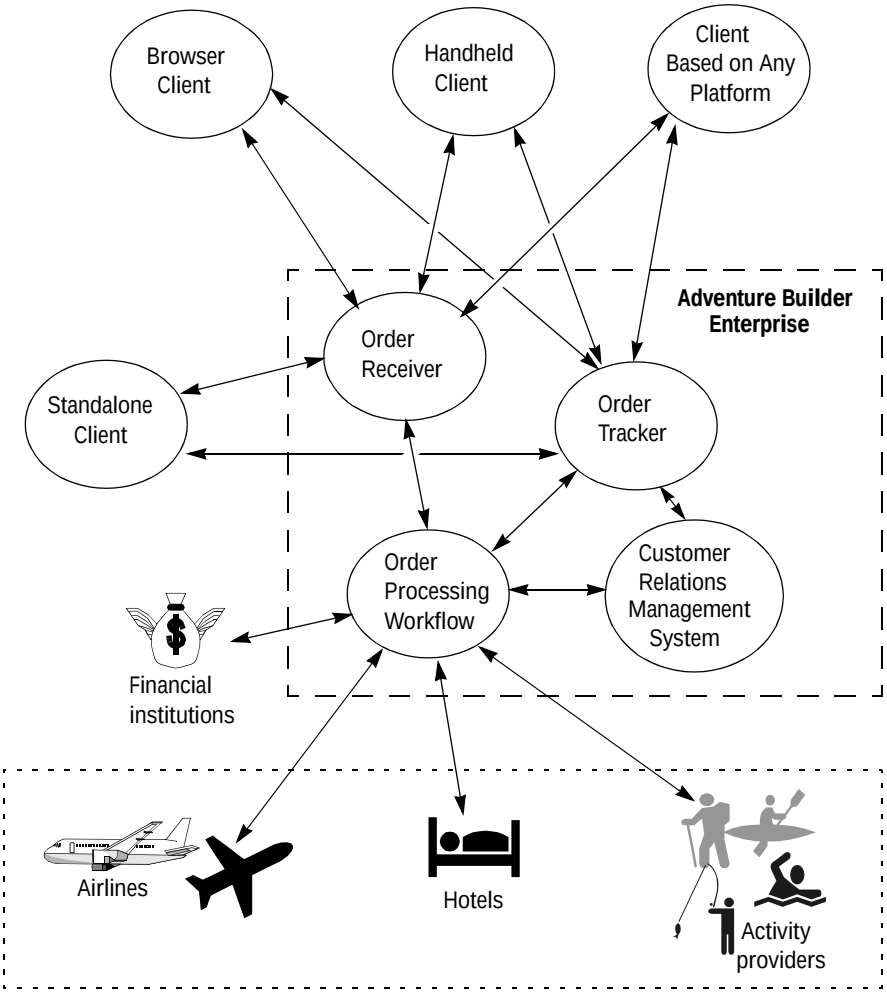

Figure 1.1 Adventure Builder Enterprise Modules

❒ Before proceeding, keep in mind that Web services basically help you to expose an interface to a software application or component that can be used by the client of that application or component, regardless of the client's platform.

Note: This sample scenario is used in the remainder of this book to illustrate specific design issues and recommendations.

1.4.1 Interacting with Business Partners

The adventure builder enterprise works with a number of partners to help it provide complete adventure packages to its customers. These partners range from small businesses, such as “mom and pop”-type shops that might arrange a fun activity in a locale, to large enterprises, such as major airlines and hotel chains. In effect, these partners fulfill various components of the travel packages ordered by customers.

Additionally, there is a two-way communication between adventure builder and its partners. The adventure builder enterprise communicates various orders to different partners. The partners, in turn, each bill the adventure builder enterprise for their costs. It is essential that the enterprise resolves the handling of this inter-partner communication when it goes online.

Web services can be an ideal way to integrate the enterprise with multiple partners for several reasons. Web services are more cost effective than electronic data interchange (EDI), currently the most common solution available for business partner interaction. EDI requires a significant up-front investment by all parties. While its large business partners may have an extensive EDI infrastructure in place, the adventure builder enterprise cannot expect its many small business partners to expend the resources to implement such an EDI infrastructure. Web services-based interaction is a good solution for small businesses that have no investment in an EDI infrastructure and more cost effective for larger enterprises with existing EDI infrastructures.

In addition, Web services can be used on a variety of hardware and software platforms. Existing and potential partners need not purchase or lease a particular hardware or software platform. Most likely, the partners can use their existing systems as long as these systems support Web services. Since many of the industry leaders cooperate on Web service standards and specifications, it is expected that the majority of platforms will support Web services.

Web services-based interaction gives the partners, especially the larger partners with significant technology infrastructure investments, a further advantage: They can use Web services to integrate their existing enterprise systems. Businesses that have integrated their existing systems with Web services offer an enhanced, more automated interaction experience for partners such as adventure builder. The result is an improved customer experience, cost saving, and increased interoperability.

1.4.2 Integrating with Existing Enterprise Information Systems

The adventure builder enterprise may have invested much time and resources in their existing enterprise information systems. The customer relations management system alone is a costly system that could have taken a significant amount of time and resources to implement and fine-tune to the enterprise's unique needs. The other enterprise information systems, too, may have been customized and tweaked so that they work efficiently within the unique combination of requirements and constraints of the adventure builder enterprise. Getting all these systems to work right is an evolving and expensive process. Although the adventure builder enterprise management is tempted by the advantages of some new technologies, they have invested too much in their existing systems to completely replace them with new systems.

Similarly, most enterprises have made extensive investments over the course of many years in information technology (IT) processing, data, and system resources. Often, most of this investment is in enterprise information systems (EISs), or what are sometimes referred to as legacy systems. Discarding existing systems is never easy. It means not only writing off money spent to install, use, and customize these systems, but also spending equivalent sums or more to install and customize new systems. There is also the disruption to the business itself, such as the lost productivity during the transition to the new system, and the potential changes in business processes that a new system may impose. Most businesses consider it more cost effective to evolve and enhance their existing information systems. As a result, enterprises are looking for solutions that let them not only stay current with the times but also evolve their infrastructure and integrate their existing systems with new systems.

Web services can efficiently meet this need to integrate existing technologies with new technologies. Since they are based on universally accepted, platform-independent standards, Web services make a natural integration layer. It is now possible to provide software to expose an existing EIS as a Web service. This makes it feasible for users who require access to that EIS to do so through its Web service.

1.4.3 Reaching Diverse Clients

Web services make it possible for an enterprise to make its functionality available to various types of clients. The adventure builder enterprise, to enhance the customer experience, can provide services such as customer order tracking and weather information at travel destination points, and the enterprise can ensure that these services are accessible by any type of client. Customers can get the information they want

from anywhere, using almost any device. Some customers may use a Web browser to access these services, while others may prefer to use a more full-featured interface such as that of a rich client written with J2SE technology or Microsoft technologies. Other customers may use mobile devices, such as cell phones or PDAs, running on the Java™ 2 Platform Micro Edition (J2ME) environment. The customer experience is enhanced when end users have such a range of choices for accessing a service.

Web services are an ideal way to open these enterprise services to all types of clients. Since they expose a standard interface that is platform and technology independent, Web service-based features make it easy to develop clients on any platform that supports Web services.

1.4.4 Aggregation of Partner Data

Much of the adventure builder enterprise's interactions with its various business partners centers on obtaining and aggregating information from partners and other disparate sources. To provide a complete adventure package to its customers, adventure builder must obtain from its partners such information as the details of modes of travel, accommodations, and activities or adventures. This information is dynamic in nature, and the adventure builder enterprise needs these updates. For example, adventure builder needs to know which seats for a particular airline are available or which hotels have rooms available, and at what price. This information changes frequently, and it is important for the adventure builder enterprise to keep its data current.

The dynamic nature of the adventure builder enterprise's information extends beyond current availability of services. It is entirely possible that one or more of its partners may change its services, or even that a business partner ceases to exist. To maintain the best possible customer experience, the adventure builder enterprise needs to make available the most current package details possible. To do so, the adventure builder enterprise needs to maintain a dynamic catalog of its offerings. That is, adventure builder not only builds its catalog of adventure packages based on the services available from its various partners, but it may also periodically update its catalog of adventures, particularly when changes occur with its partners.

Web services enable a truly dynamic way to build and maintain this catalog of information. Furthermore, by incorporating its Web services with registry services, adventure builder can expand its network of suppliers and allow any number of suppliers or partners to participate at their own choosing. Once suppli-

ers signal their participation through the registry, adventure builder can dynamically build its own catalog from the suppliers' offerings. Keep in mind, however, that a truly dynamic arrangement requires a set of legal and financial agreements to be in place beforehand.

1.5 J2EE 1.4: The Platform for Web Services

This section briefly describes the entire J2EE platform, providing an overview of the technologies that are part of the platform and the platform's benefits. The J2EE 1.4 platform, while it introduces new technologies for Web services, also builds upon its earlier technologies for its Web services support.

For more complete coverage of the earlier platform technologies, you should refer to the previous book in the Java BluePrint's series, *Designing Enterprise Applications with the J2EE Platform, Second Edition* (Addison Wesley) and the Java BluePrints Web site: `http://java.sun.com/blueprints/`. (See "References and Resources" on page xx.)

1.5.1 J2EE Platform Overview

The J2EE platform is designed to provide server-side and client-side support for developing Web services and distributed, multi-tier enterprise applications. Let's briefly examine the main concepts underlying the architecture of the J2EE platform. (Readers familiar with the J2EE platform may want to skip this section.)

- **Multi-tier model—**The J2EE platform is architected to facilitate the deployment of multi-tier distributed applications and Web services. The platform defines different tiers, including a client tier, one or more middle tiers, and a back-end tier. It also defines a standard way for these tiers to communicate with each other. Because applications can communicate in a standard way, it is possible for an application's logic to reside in different tiers. Thus, applications can be deployed across different tiers in a distributed manner. Furthermore, the platform's multi-tier model enables various parts of an application to run on different systems. Each tier may also be assigned different and distinct responsibilities. This multi-tier development model also enables the platform to provide rich support to a wide variety of clients. The platform supports

browser clients, wireless clients, rich Java-based GUI client, even non-Java clients.

- **Component-based development**—The J2EE platform supports component-based development of applications and services. The portions of application logic that reside on separate tiers can make use of the different components provided by the platform for each tier. Such component-based development encourages reusability, since components are discrete modules that can be reused by virtually any other component when needed. The platform can support a range of components for representing application business logic. This type of component-based development model also enables apportioning development responsibilities by skill set; that is, assigning different and distinct responsibilities to various developers based on their skill set. Such division by skill set helps to efficiently use developer skills.
- **Container-based component management—**To ensure that the components interact in a standard way, the platform introduces the concept of container-based component management. Components run within containers, which are standardized runtime environments that provide specific services to components and thus ensure application portability across platform implementations. In such a managed environment, containers interpose on all method calls and apply their services, which include standard session management, automated and declarable support for database transactions, security, standardized access to EISs, standardized deployment of applications (including the means to modify application behavior at deployment), and so forth. Components can expect these services to be available on any J2EE platform from any vendor. Along with this emphasis on standardization, the J2EE platform still remains open and flexible so that vendors can provide their own value-added support without compromising the portability of the application.
- **Support for the J2EE standard to promote portability**—The J2EE standard, which is defined through a set of related specifications—the Enterprise JavaBeans specification, the Java Servlet specification, the JavaServer Pages specification, Java API for XML Based RPC specification, among many others—define and ensure a platform architecture that enables development of distributed, portable, and interoperable enterprise applications and Web services. In addition to the specifications, several other technology deliverables support the J2EE standard, including the J2EE Compatibility Test Suite, the J2EE reference implementation, and the J2EE SDK. The J2EE Compatibility Test Suite (CTS) helps maximize the portability of applications by validating the

specification compliance of a J2EE platform product. The J2EE reference implementation, a complete implementation of the J2EE standard provided by Sun Microsystems, represents an operational definition of the J2EE platform. It is used by licensees as the "gold standard" to determine what their product must do under a particular set of application circumstances. It is the standard platform for running the J2EE Compatibility Test Suite, and it can be used by developers to verify the portability of an application. The J2EE reference implementation is available in both binary and source code form.

- **Support for the WS-I standard for interoperability**—The Web Services Interoperability Organization (WS-I) promotes Web service interoperability across development platforms, operating systems, and programming languages, especially with the WS-I Basic Profile, which the J2EE 1.4 platform supports. Ensuring interoperability and portability starts with the specifications themselves. The J2EE 1.4 platform includes specifications and technologies that support Simple Object Access Protocol (SOAP), Web Services Definition Language (WSDL), and Universal Discovery, Description, and Integration (UDDI) specification. The platform includes technologies that enable standardized use of Web service specifications within the Java platform: Java™ API for XML-Based RPC (JAX-RPC), Web Services for J2EE (JSR-109), SOAP with Attachments API for Java (SAAJ), Java API for XML Registries (JAXR), and Java API for XML Processing (JAXP).

1.5.2 J2EE Platform Benefits

The J2EE platform, with features designed to expedite developing distributed and interoperable applications, offers Web services application developers these benefits:

- **Simplifies architecture and development—**The component-based J2EE development model enhances application development productivity because the components map well to the desired functionality of the application and service to be developed. Using tools, developers can configure application behavior at assembly or deployment, as required, knowing they can rely on the standardized interaction between components and the standard services provided by the container for the component.
- **Ensures support for emerging Web service standards**—The J2EE platform continues to evolve its support for Web service standards and the WS-I Basic

Profile. This ensures that application logic, whether new logic or enhancements to existing logic, developed using the standardized J2EE components can easily be exposed as Web services.

- **Ensures the development of portable and interoperable services**—The J2EE platform, since it supports both common J2EE and WS-I standards, ensures not only that applications are portable across J2EE implementations, but also that services are interoperable with any Web service implemented on any other platform that conforms to WS-I standards.
- **Allows for integration with existing information systems—**The J2EE platform provides industry-standard APIs, such as the J2EE Connector architecture, the JDBC API, Java Message Service, among others, for accessing EISs. Coupling these APIs with its support for Web services, the J2EE platform provides an excellent way to integrate existing EISs and make their data available to clients on heterogeneous platform environments.
- **Is scalable to meet demand variations—**J2EE containers provide a mechanism that supports simplified scaling of distributed applications with limited application development effort. J2EE containers provide transaction support, database connections, life cycle management, and other services that are both scalable and require no code from application developers.
- **Provides a flexible security model—**The J2EE security model is designed to be flexible, allowing component developers to declaratively specify component security requirements. Both Enterprise JavaBeans technology and Java Servlet APIs also provide programmatic security control.

In short, the J2EE standards and the J2EE brand have created a huge marketplace for servers, tools, and components. The J2EE brand on a server product ensures the consistent level of service fundamental to the goals of the J2EE platform. The J2EE standards encourage a lively marketplace for tools and components. Based on past experience coupled with the industry momentum behind Web services, all leading enterprise software vendors are expected to participate in the marketplace for J2EE 1.4 products.

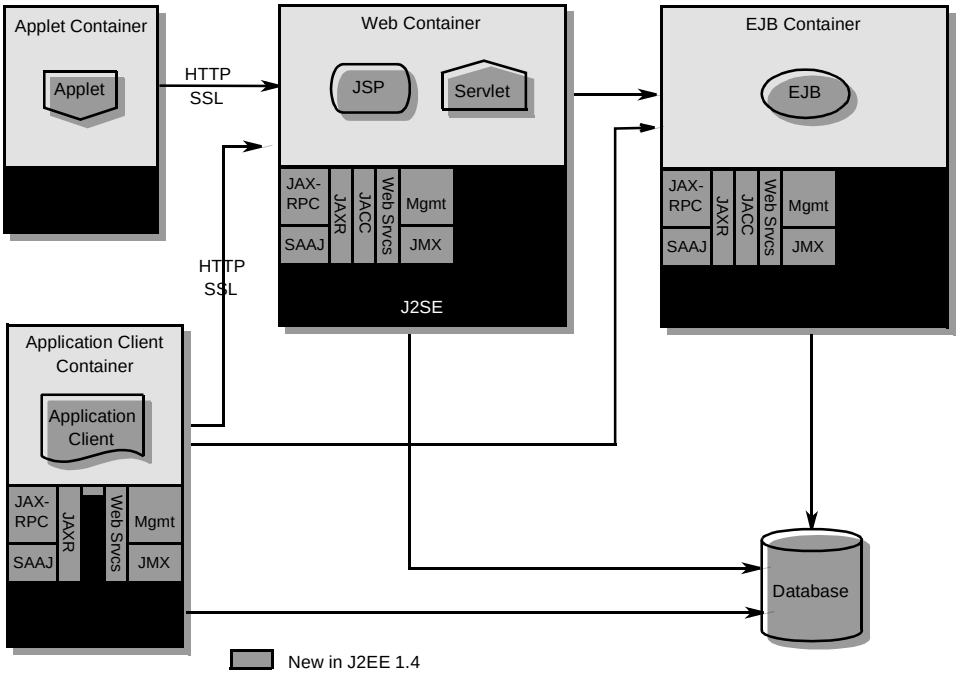

Figure 1.2 J2EE 1.4 Platform Architecture

1.5.3 J2EE Platform Technologies

The J2EE platform consists of technologies that support the development of distributed enterprise applications and services. These technologies fall into three broad categories: components, service, and communication. Figure 1.2 shows how these technologies interrelate. This section provides a brief overview of the J2EE 1.4 platform technologies. Refer to Chapter 2 for a more detailed discussion of these technologies.

1.5.3.1 Component Technologies

Developers use the platform's component technologies to create the essential parts of an enterprise application and a Web service, such as the business logic of the service. The approach of using component technologies to develop the essential parts

of a service keeps development modular and promotes the reuse of modules. The various component technologies of the J2EE platform are as follows:

- **Client component—**The platform provides support for different types of clients to interact with components on the server side. *Applet clients* are Java-based clients that usually run from within a Web browser and have full access to the features of the J2SE platform. *Application clients* (often referred to as stand-alone clients) execute in their own containers and have full access to J2EE platform services such as JNDI lookups, asynchronous messaging, and so forth. These clients can directly interact with the Web and EJB components on the server-side of the application.
- **Web component—**Web components provide a response to a request received via HTTP. The J2EE platform defines two distinct Web component types. *Servlet components* extend the functionality of a Web server in a portable and efficient manner. With servlets, developers can map a set of URLs to a set of servlets. As a result of such mapping, an HTTP request to one of the URLs invokes the mapped servlet, which in turn processes the request and returns a response. Servlet components can also be exposed as Web services. *JSP components*, as well as servlet components, enable the generation of dynamic content.
- **Enterprise JavaBeans component—**Enterprise JavaBeans (EJB) components are designed specifically with business logic in mind. EJB components are scalable, transactional, and secure. *Session bean components* usually provide services to a single client, and their state cannot be recovered after a server crash. Furthermore, stateless session bean components can be exposed as Web services. *Entity bean components* are the object representation of data maintained in a data store. These components manage persistent data, either managing the persistence on their own or depending on the container to manage their persistence. *Message-driven bean components* enable clients to access business logic contained within enterprise bean components in an asynchronous manner. A *timer service* enables the implementation of timed operations.

Using these component technologies for implementing an application or service ensures the standardization of the application or service. Such standardization furthermore enables the reusability and portability of the application and service. Because they address different aspects of an enterprise application or

service, using these components helps to divide development into different skill sets and make efficient use of different skills.

1.5.3.2 Platform and Container Services

The J2EE platform component technologies—the client, Web, and EJB components—depend on the support of the J2EE container to function properly. The J2EE platform standard, to ensure that components are portable, requires a conforming platform provider to make certain services available. Enterprise applications and Web services built with portable components and standard services are themselves assured of portability. Among the platform's required services are:

- **Naming service**—A naming service allows symbolic access to EIS resources and components within a naming environment. These components can be customized when assembled or deployed without requiring changes to the look-up source code.
- **Deployment service**—A deployment service allows changes to component behavior (such as transaction requirements and security requirements) at deployment without the need to change a component's source code.
- **Transaction service**—A transaction service frees the component developer from having to include code to handle such transactional issues as multi-user access and failure/recovery. A transaction service allows the transaction requirements for components to be specified when they are assembled.
- **Security service**—A security service ensures that components and resources are accessed by only those authorized for access. In addition, a security service provides authentication and confidentiality, among other services, for users.

1.5.3.3 Communication

The J2EE platform, in addition to specifying component technologies and platform services, also requires a set of standard communication technologies. These communication technologies bring the platform's components and services together, making the J2EE platform an integrated, standard platform for developing portable, interoperable enterprise applications and Web services.

- **Internet protocols**—The J2EE platform supports such standard, common Internet protocols as TCP/IP, HTTP, SSL, and so forth. These Internet protocols

enable communication between components and between components and their clients.

- **Remote Method Invocation (RMI) protocols**—The J2EE platform supports the Java RMI. Java RMI relies on the Remote Method Invocation APIs, which use Java language interfaces to define remote interface objects. The platform uses IIOP to turn local method invocations into remote method invocations.
- **Messaging technologies**—In addition to its support for these Internet and RMI synchronous protocols, the J2EE platform supports technologies that enable asynchronous communication. Examples of these technologies are the Java Message Service API and the JavaMail API.
- **Web service technologies**—The J2EE platform also supports Web service-specific technologies and protocols that, along with the already mentioned technologies and protocols, standardize communication between J2EE components and J2EE clients. With the advent of Web services, which improve interoperability with non-Java clients, the J2EE platform supports Web service standards such as SOAP and UDDI using technologies such as Java API for XML-based RPC, Java API for XML Registries, and so forth.

1.6 Conclusion

The challenge to IT and computing professionals today is to efficiently develop and deploy distributed applications and Web services for use on both corporate intranets and over the Internet. Companies that can do this effectively are sure to gain strategic advantage in the information economy.

The Java 2 Platform, Enterprise Edition is a standard set of Java technologies that streamline the development, deployment, and management of Web services and enterprise applications. The J2EE platform provides a functionally complete environment, one in which it is possible to develop a large class of Web services and enterprise applications using J2EE technologies. Furthermore, developers can be assured that applications written for the J2EE platform will run on any J2EE-compatible server. Overall, the J2EE platform provides numerous benefits for organizations developing these Web service applications, including a simplified development model; industrial-strength scalability; support for existing information systems; choices in servers, tools, and components; and a simple, flexible security model.

By providing the ability to deploy component-oriented Web service endpoint and client applications in a platform-neutral manner, the J2EE platform gives fast-moving enterprises a significant and measurable competitive edge.