1.2.1 J2EE Platform Overview

The J2EE platform is designed to provide server-side and client-side support for developing distributed, multitier applications. Such applications are typically configured as a client tier to provide the user interface, one or more middle-tier modules that provide client services and business logic for an application, and back-end enterprise information systems providing data management. Figure 1.1 illustrates the various components and services that make up a typical J2EE environment.

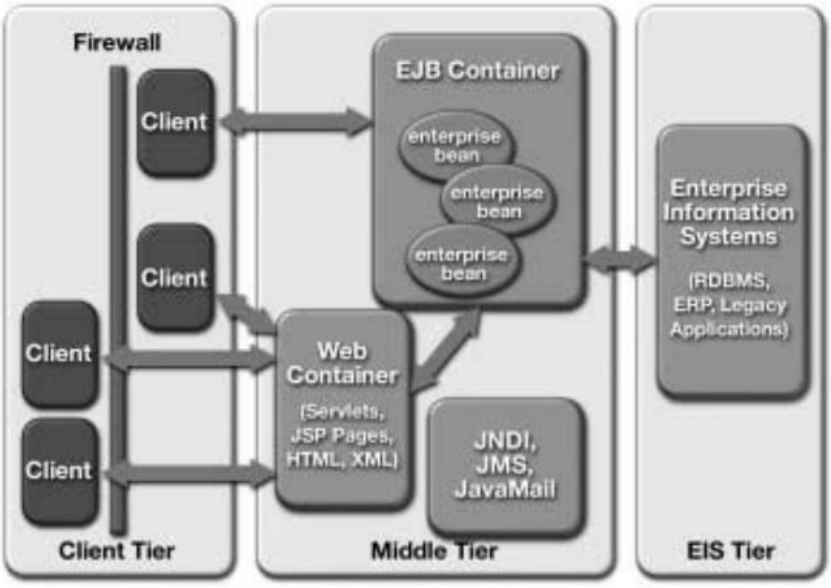

Figure 1.1 J2EE Environment

1.2.1.1 Multitier Model

As illustrated, the J2EE platform provides a multitier distributed application model. This means that the various parts of an application can run on different devices. The

J2EE architecture defines a *client tier*, a *middle tier* (consisting of one or more subtiers), and a *back-end tier*. The client tier supports a variety of client types, both outside and inside of corporate firewalls. The middle tier supports client services through Web containers in the *Web tier* and supports business logic component services through Enterprise JavaBeans™ (EJB™) containers in the *EJB tier*. On the back end, the enterprise information systems in the *EIS tier* are accessible by way of standard APIs.

1.2.1.2 Container-Based Component Management

Central to the J2EE component-based development model is the notion of containers. Containers are standardized runtime environments that provide specific services to components. Components can expect these services to be available on any J2EE platform from any vendor. For example, all J2EE Web containers provide runtime support for responding to client requests, performing request-time processing (such as invoking JSP pages or servlet behavior), and returning results to the client. In addition, they provide APIs to support user session management. All EJB containers provide automated support for transaction and life cycle management of EJB components, as well as bean lookup and other services. Containers also provide standardized access to enterprise information systems; for example, providing access to relational data through the JDBC API.

In addition, containers provide a mechanism for selecting application behaviors at assembly or deployment time. Through the use of deployment descriptors (XML files that specify component and container behavior), components can be configured to a specific container's environment when deployed, rather than in component code. Features that can be configured at deployment time include security checks, transaction control, and other management responsibilities.

While the J2EE specification defines the component containers that a platform implementation must support, it doesn't specify or restrict the containers' configurations. Thus, both container types can run on a single platform, Web containers can live on one platform and EJB containers on another, or a J2EE platform can be made up of multiple containers on multiple platforms.

1.2.1.3 Support for Client Components

The J2EE client tier provides support for a variety of client types, both within the enterprise firewall and outside. Clients can be offered through Web browsers by using plain HTML pages, HTML generated dynamically by JavaServer Pages™

(JSP™) technology, or Java applets. Clients can also be offered as stand-alone Java language applications. J2EE clients are assumed to access the middle tier primarily using Web standards, namely HTTP, HTML, and XML. Because of its flexible programming model, the J2EE platform can support a number of simple application models implemented primarily on the strengths of its Web tier component technologies.

To support more complex user interactions, it may be necessary to provide functionality directly in the client tier. This functionality is typically implemented as JavaBeans™ components that interact with the service in the middle tier via servlets. Client-tier JavaBeans components would typically be provided by the service as an applet that is downloaded automatically into a user's browser. To eliminate problems caused by old or non-standard versions of the Java Virtual Machine in a user's browser, the J2EE application model provides special support for automatically downloading and installing the Java Plug-in. In addition, the J2EE platform is flexible enough to support alternate client models easily, including wireless phones and handheld devices that use programming models provided by the Java 2 Platform, Micro Edition.

Client-tier beans can also be contained in a stand-alone application client written in the Java programming language. In this case, the enterprise typically would make the client available for users to download from a browser using Java Web Start technology. Java Web Start technology makes application deployment portable by providing browser-based download and installation mechanisms. With both application and deployment portability, this ensures that users can always access and work with the latest versions of stand-alone application clients.

If desired, non-Java clients such as Visual Basic programs can present J2EE services to users. Since the service is presented by servlets in the middle tier to first-tier clients using the standard HTTP protocol, it is easy to access it from practically any program running on any operating system.

1.2.1.4 Support for Business Logic Components

While simple J2EE applications may be built largely in the client tier, business logic is often implemented on the J2EE platform in the middle tier as Enterprise JavaBeans components (also known as enterprise beans). Enterprise beans allow the component or application developer to concentrate on the business logic while the complexities of delivering a reliable, scalable service are handled by the EJB container.

In many ways, the J2EE platform and EJB architecture have complementary goals. The EJB component model is the backbone of industrial-strength application architectures in the J2EE programming model. The J2EE platform complements the EJB specification by:

- Fully specifying the APIs that an enterprise bean developer can use to implement enterprise beans
- Defining the larger, distributed programming environment in which enterprise beans are used as business logic components

1.2.1.5 Support for the J2EE Standard

The J2EE standard is defined through a set of related specifications. Key among these are the J2EE specification, the Enterprise JavaBeans specification, the Java Servlet specification, and the JavaServer Pages specification. Together, these specifications define the architecture described in this book. 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. This test suite begins where the basic Java Conformance Kit (JCK) leaves off. The CTS tests conformance to the Java standard extension APIs that are not covered by a JCK. In addition, it tests a J2EE platform's ability to run standard end-to-end applications.

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.

The J2EE SDK, based on the J2EE reference implementation binary, is provided freely to the developer community to help expedite developer adoption of the J2EE standard. Although not a commercial product and not available for commercial use, the J2EE SDK is useful for developing application demos and prototypes, such as the Java Pet Store sample application described in this book. The J2EE SDK also includes application verification and deployment tools to simplify

development, and the J2EE Tutorial, which provides step-by-step examples and information that developers need to begin working with the platform.

Another word on J2EE standards and portability: The J2EE specifications have, by design, set the platform-compatibility bar at a level that's relatively easy to clear. Because the platform specifications are developed collaboratively, platform vendors must have plenty of opportunity to supply J2EE platform implementations. Obvious and unreasonable implementation hurdles were avoided. For example, there are no restrictions on vendors adding value to J2EE products by supporting services not defined in the specifications.

While the J2EE standard is designed to encourage component portability, specific results are primarily a function of how a component uses services provided by its container. Vendor-specific features limit component portability. The J2EE specifications spell out a base set of capabilities that a component can count on, providing components with a level of cross-container portability. Needless to say, an application developer expecting to deploy on a specific vendor implementation of the J2EE platform should be able to do so across a wide range of operating systems and hardware architectures.

1.2.2 J2EE Platform Benefits

With features designed to expedite the process of developing distributed applications, the J2EE platform offers several benefits:

- Simplified architecture and development
- Freedom of choice in servers, tools, and components
- Integration with existing information systems
- Scalability to meet demand variations
- Flexible security model

1.2.2.1 Simplified Architecture and Development

The J2EE platform supports a simplified, component-based development model. Because it is based on the Java programming language and the Java 2 Platform, Standard Edition (J2SE™ platform), this model offers "Write-Once-Run-Anywhere™" portability, supported by any server product that conforms to the J2EE standard.