# J2EE Platform Technologies

*by Vijay Ramachandran*

THE J2EE platform specifies technologies to support multitier enterprise applications. These technologies fall into three categories: component, service, and communication.

The component technologies are those used by developers to create the essential parts of the enterprise application, namely the user interface and the business logic. The component technologies allow the development of modules that can be reused by multiple enterprise applications. The component technologies are supported by J2EE platform's system-level services. These system-level services simplify application programming and allow components to be customized to use resources available in the environment in which they are deployed.

Since most enterprise applications require access to existing enterprise information systems, the J2EE platform supports APIs that provide access to databases, enterprise information systems such as SAP and CICS, and services such as transaction, naming and directory, and asynchronous communication. Finally, the J2EE platform provides technologies that enable communication between clients and servers and between collaborating objects hosted by different servers.

This chapter will provide an overview of the J2EE platform technologies.

## 2.1 Component Technologies

A *component* is an application-level software unit. In addition to JavaBeans™ components, which are part of the J2SE™ platform, the J2EE platform supports the following types of components: applets, application clients, Enterprise JavaBeans™ (EJB™)components, Web components, and resource adapter components.

Applets and application clients run on a client platform, while EJB, Web, and resource adapter components run on a server platform.

Except for resource adapters, application architects and developers typically design and develop the components of a J2EE application. EIS and tool vendors design, develop, and provide resource adapter components, which are then deployed on the server and used by other components of the platform to access data in an EIS.

All J2EE components depend on the runtime support of a system-level entity called a *container*. Containers provide components with services such as lifecycle management, security, deployment, and threading. Because containers manage these services, many component behaviors can be declaratively customized when the component is deployed in the container. For example, an application component provider can specify an abstract name for a database that an Enterprise JavaBeans component needs to access, and a deployer will link that name with the information (such as a user name and password) needed to access the database in a given environment.

The following sections provide overviews of the different types of J2EE components and containers.

### 2.1.1 Types of J2EE Clients

The J2EE platform allows different types of clients to interact with server-side components.

- *Applets* are Java-based client components that usually execute within a Web browser, and that have access to all features of the Java programming language. J2EE applications can use applets for a more powerful user interface. Browser-based applet clients communicate over HTTP.
- An *application client* executes in its own client container. (The client container is a set of libraries and APIs that support the client code.) Application clients are user interface programs that can directly interact with the EJB tier of a J2EE platform-based application using RMI-IIOP. These clients have full access to J2EE platform services such as JNDI lookups, asynchronous messaging, and the JDBC™ API. An application client's container provides access to these J2EE services and handles RMI-IIOP communication.
- A *Java Web Start-enabled rich client* is a stand-alone client based on JFC/Swing APIs and enabled for the J2EE platform through the Java Web Start

technology. A rich client has increased user interface features available to it, such as a better interactive environment and richer graphic capabilities, along with the J2EE platform features and services. Java Web Start technology enables application deployment through a single-step download-and-launch process performed by means of a Web browser. Rich clients communicate with the server using the J2SE environment to execute XML over HTTP(S). As Web service technologies gain ground in the future, these rich clients are well-positioned to efficiently use open communication standards such as JAX-RPC technology.

- A *wireless client* is based on Mobile Information Device Profile (MIDP) technology. MIDP is a set of Java APIs which, along with Connected Limited Device Configuration (CLDC), provides a complete J2ME environment for wireless devices.

### 2.1.2 Web Components

A *Web component* is a software entity that provides a response to a request. A Web component typically generates the user interface for a Web-based application. The J2EE platform specifies two types of Web components: servlets and JavaServer Pages™ (JSP™) pages. The following sections give an overview of Web components, which are discussed in detail in Chapter 4.

#### 2.1.2.1 Servlets

A *servlet* is a component that extends the functionality of a Web server in a portable and efficient manner. A Web server hosts Java servlet classes that execute within a servlet container. The Web server maps a set of URLs to a servlet so that HTTP requests to these URLs invoke the mapped servlet. When a servlet receives a request from a client, it generates a response, possibly by invoking business logic in enterprise beans or by querying a database directly. It then sends the response—as an HTML or XML document—to the requestor.

A servlet developer uses the servlet API to:

- Initialize and finalize a servlet
- Access a servlet's environment
- Receive/forward requests and send responses
- Maintain session information on behalf of a client