1.3 J2EE Application Scenarios

The following sections present a number of application scenarios, setting the stage for a detailed discussion of the sample application. The J2EE specifications encourage architectural diversity. The J2EE specifications and technologies make few assumptions about the details of API implementations. The application-level decisions and choices are ultimately a trade-off between functional richness and complexity.

The J2EE programming model is flexible enough for applications that support a variety of client types, with both the Web container and EJB container as optional. Figure 1.2 reflects a range of possible application configurations, including cases where clients interact solely with the Web container, where clients interact directly with the EJB container, and full-blown multitier applications with stand-alone clients, Web-tier components, middle-tier EJB components, and EIS-tier access to resources and data. While the J2EE platform has no implicit bias

favoring one application scenario over another, a J2EE product should be able to support any and all of these scenarios.

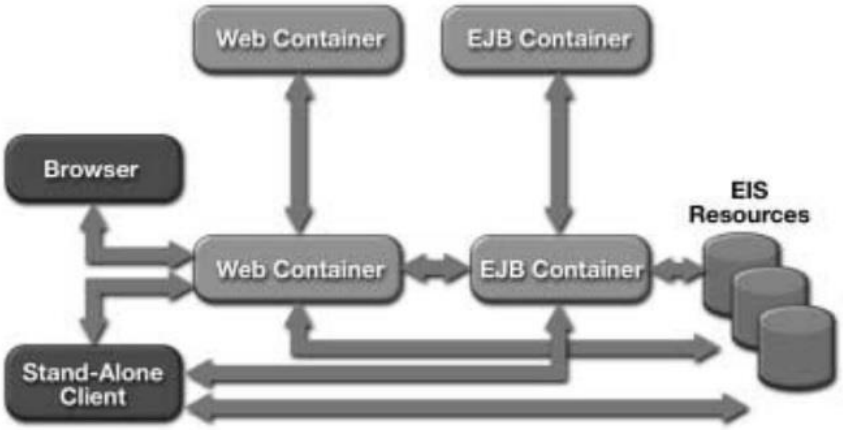

Figure 1.2 J2EE Application Scenarios

The sample application is a multitier application that uses both a Web container and an EJB container. The following enterprise requirements heavily influenced the choices made in developing the sample application:

- The need to make rapid and frequent changes to the "look" of the application
- The need to partition the application along the lines of presentation and business logic so as to increase modularity
- The need to simplify the process of assigning suitably trained human resources to accomplish the development task such that work can proceed along relatively independent but cooperating tracks
- The need to have developers familiar with back-office applications unburdened from GUI and graphic design work, for which they may not be ideally qualified
- The need to have the necessary vocabulary to communicate the business logic to teams concerned with human factors and the aesthetics of the application

- The ability to assemble back-office applications using components from a variety of sources, including off-the-shelf business logic components
- The ability to deploy transactional components across multiple hardware and software platforms independently of the underlying database technology
- The ability to externalize internal data without having to make many assumptions about the consumer of the data and to accomplish this in a loosely coupled manner

Clearly, relaxing any or all of these requirements would influence some of the application-level decisions and choices that a designer would make. Although it is reasonable to speak of "throw-away" presentation logic (that is, applications with a look and feel that ages rapidly), there is still significant inertia associated with business logic. This is even more true in the case of database schemas and data in general. It is fair to say that as one moves further away from EIS resources, the volatility of the application code increases dramatically; that is, the code's "shelf-life" drops significantly.

In summary, the J2EE programming model promotes a model that anticipates growth, encourages component-oriented code reusability, and leverages the strengths of inter-tier communication. It is the tier integration that lies at the heart of the J2EE programming model.

1.3.1 Multitier Application Scenario

Figure 1.3 illustrates an application scenario in which the Web container hosts Web components that are almost exclusively dedicated to handling a given application's presentation logic. JSP pages, supported by servlets, generate dynamic Web content for delivery to the client. The EJB container hosts application components that use EIS resources to service requests from Web-tier components. This architecture decouples data access from the application's user interface. The architecture is also

implicitly scalable. Application back-office functionality is relatively isolated from the end-user look and feel.

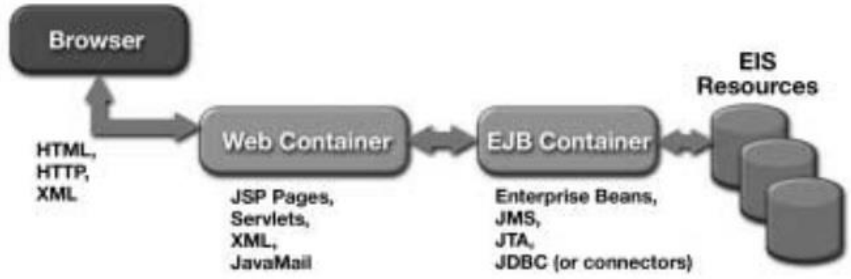

Figure 1.3 Multitier Application

It is worth noting that XML plays an integral role in this scenario. The ability to both produce and consume XML data messages in the Web container is an extremely flexible way to embrace a diverse set of client types. These platforms range from general purpose XML-enabled browsers to specialized XML rendering engines targeting vertical solutions. XML data messages typically use HTTP as their transport protocol. Java and XML are complementary technologies: The Java language offers portable code, XML provides portable data.

In the Web tier, the question of whether to use JSP pages or servlets comes up repeatedly. JSP technology is intended for application user interface components, while Java Servlets are preferred for request processing and application control logic. Servlets and JSP pages work together to provide dynamic content from the Web tier.

1.3.2 Stand-Alone Client Scenario

Figure 1.4 illustrates a stand-alone client scenario.

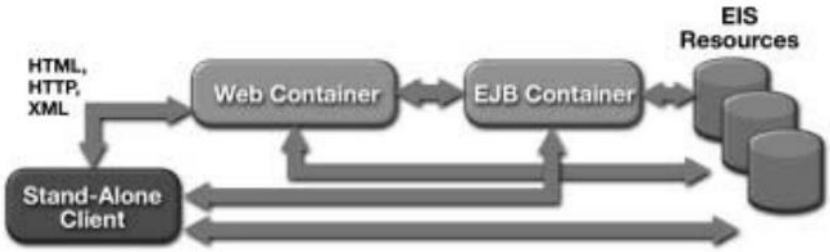

Figure 1.4 Stand-Alone Clients

The stand-alone client may be one of three types:

- EJB clients interacting directly with enterprise beans hosted in an EJB container within an EJB server, as shown in Figure 1.5. This scenario uses RMI-IIOP, and the EJB server accesses EIS resources using JDBC and the J2EE Connector architecture.

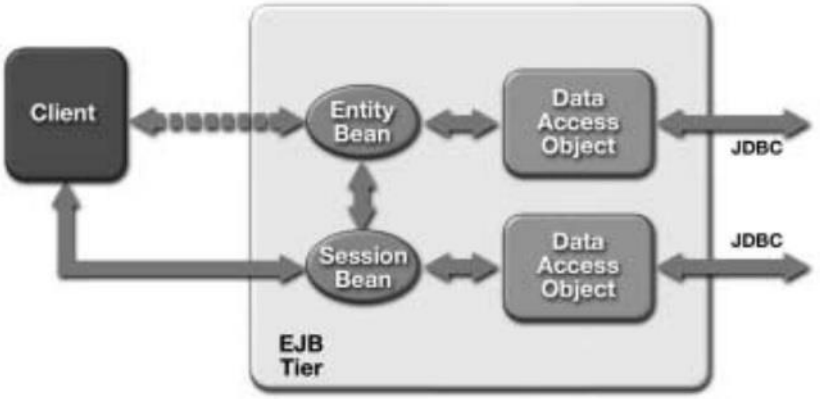

Figure 1.5 EJB-Centric Java Client