

Java™ 2 Platform, Enterprise Edition Technical Overview

Please send technical comments to: j2ee-spec-technical@eng.sun.com
Please send business comments to: j2ee-spec-business@eng.sun.com

Please send comments on this draft by July 15, 1999.

Alpha Draft

Public Draft



Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94043 USA
650 960-1300 fax 650 969-9131

Sun Microsystems, Inc. - Confidential

Copyright Information

© 1999, Sun Microsystems, Inc. All rights reserved.
901 San Antonio Rd., Palo Alto, California 94303 U.S.A.

This document is protected by copyright. No part of this document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

The information described in this document may be protected by one or more U.S. patents, foreign patents, or pending applications.

TRADEMARKS

Sun, Sun Microsystems, the Sun Logo, Java, JDBC, JavaBeans, Enterprise JavaBeans, JavaMail, JavaServer Pages, Java Naming and Directory Interface, and Write Once, Run Anywhere are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE DOCUMENT. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.



Please
Recycle



Adobe PostScript

Contents

Introduction 1-1

What Is the Java 2 Platform, Enterprise Edition? 1-1

J2EE Application Model 1-2

 The Java Technology Foundation 1-3

 Security 1-3

 The Middle Tier 1-4

 The Client Tier 1-4

 HTML Page Based Clients 1-5

 HTTP Content Based Clients 1-5

 Intranet Clients 1-6

 Other Client Types 1-6

 The Enterprise Information Systems 1-7

 J2EE Declarations 1-7

J2EE Platform 1-7

 J2EE Application Assembly and Deployment 1-8

 Java Technology Standards for the J2EE Platform 1-9

 IETF Standards for the J2EE Platform 1-9

 CORBA Technology Standards for the J2EE Platform 1-9

J2EE Compatibility Test Suite 1-9

J2EE Reference Implementation 1-10

Example: A Web Store 2-1

Introduction

This document provides an overview of Java 2 platform, Enterprise Edition.

What Is the Java™ 2 Platform, Enterprise Edition?

Enterprises today need to extend their reach, reduce their costs, and lower their response times by providing easy-to-access services to their customers, partners, employees, and suppliers.

Typically, applications that provide these services must combine existing Enterprise Information Systems (EISs) with new business functions that deliver services to a broad range of users. These services need to be:

- *Highly available*, to meet the needs of today's global business environment.
- *Secure*, to protect the privacy of users and the integrity of enterprise data.
- *Reliable and scalable*, to insure that business transactions are accurately and promptly processed.

In most cases, these services are architected as multi-tier applications. The middle-tier of these applications is where the majority of the application development work is done. The middle-tier implements the new services that integrate existing EISs with the business functions and data of the new service. The middle-tier shields first-tier clients from the complexity of the enterprise and takes advantage of rapidly maturing Internet technologies to minimize user administration and training.

Java 2 platform, Enterprise Edition reduces the cost and complexity of developing these multi-tier services, resulting in services that can be rapidly deployed and easily enhanced as the enterprise responds to competitive pressures.

The Java 2 platform, Enterprise Edition (J2EE) achieves these benefits by defining a standard architecture that is delivered as the following elements:

- **J2EE Application Programming Model** - A standard programming model for developing multi-tier, thin-client applications.
- **J2EE Platform** - A standard platform for hosting J2EE applications, specified as a set of required APIs and policies.
- **J2EE Compatibility Test Suite** - A suite of compatibility tests for verifying that a J2EE platform product is compatible with the J2EE platform standard.
- **J2EE Reference Implementation** - A reference implementation for demonstrating the capabilities of J2EE and for providing an operational definition of the J2EE platform.

J2EE Application Model

J2EE applications implement enterprise services for customers, employees, suppliers, partners, and other interests that make demands on or contributions to the enterprise.

To better control and manage these applications, their business functions are conducted in the middle-tier. The middle-tier refers to an environment that is closely controlled by an enterprise's IR department. The middle-tier is typically run on dedicated server hardware and has access to the full services of the enterprise.

J2EE applications often rely on the EIS-Tier to store the enterprise's business-critical data. This data and the systems that manage it are at the inner-core of the enterprise.

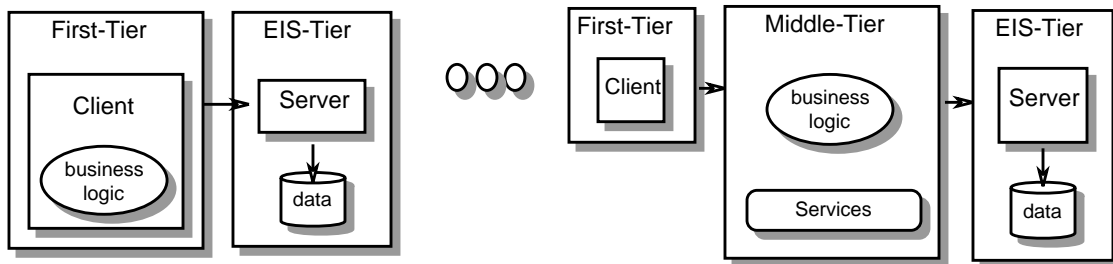


FIGURE 1 Two-Tier vs. Multi-Tier Application Models

Originally, the two-tier, client-server application model promised improved scalability and functionality. Unfortunately, the complexity of delivering EIS services directly to every user and the administrative problems caused by installing and maintaining business logic on every user machine were major limitations.

These two-tier limitations are avoided by implementing enterprise services as multi-tier applications. These systems provide the increased accessibility that is now demanded by all elements of an enterprise. This shift is driving major investments in the development of middle-tier software.

Developing multi-tier services has been complicated by the need to develop both the service's business function and much of its basic systems function. Because each multi-tier server product had its own application model, it was difficult to hire and train a development staff. In addition, as service volume increased it was often necessary to change the whole multi-tier infrastructure, resulting in major porting costs and delays.

The J2EE application model defines an architecture for implementing services as multi-tier applications that avoid these problems and deliver the scalability, accessibility, and manageability that is needed.

The J2EE application model partitions the work needed to implement a multi-tier service into two parts: the business and presentation logic to be implemented by the developer, and the standard system services provided by the J2EE platform. The developer can rely on the platform to provide the solutions for the hard systems-level problems of developing a middle-tier service.

The J2EE application model provides the benefits of Write Once, Run Anywhere™ portability and scalability for multi-tier applications. This standard model minimizes the cost of developer training while providing the enterprise with a broad choice of J2EE servers and development tools.

The J2EE application model is a major step forward in minimizing the cost and complexity of building multi-tier applications.

The Java Technology Foundation

The J2EE application model begins with the Java programming language and the Java virtual machine. The proven portability, security, and developer productivity they provide forms the basis of the application model.

The application model also includes the JavaBeans™ component model. JavaBeans components make it easy to componentize the Java technology-based code for common functions, then customize and combine these components visually with JavaBeans development tools.

Security

By shielding applications from the complexity of implementing security, applications are portable to a wide variety of security implementations.

J2EE defines standard declarative access control rules for applications that are defined by the application programmer/assembler and interpreted when the application is deployed on the enterprise platform. J2EE also requires platform vendors to supply standard login mechanisms so applications do not have to incorporate these mechanisms into their logic. The same program works in a variety of different security environments without change.

The Middle Tier

The focus of the J2EE application model is the middle-tier. Middle-tier business functions are implemented as Enterprise JavaBean™ (EJB) components, as shown in FIGURE 2. EJB components allow service developers to concentrate on the business logic and let the EJB server handle the complexities of delivering a reliable, scalable service.

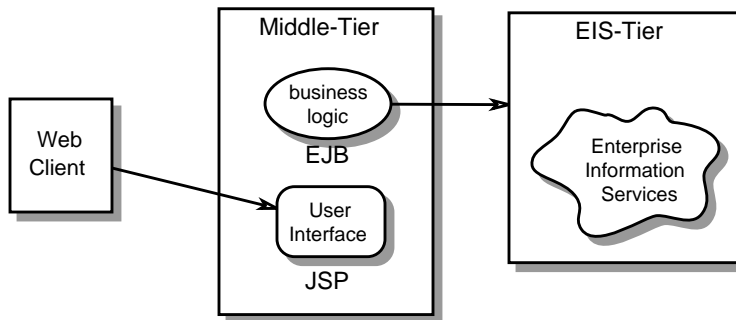


FIGURE 2 EJB Components Implement Business Logic in the Middle-Tier

JavaServer Pages™ technology and servlets present middle-tier functions to first-tier clients as simple-to-access Internet-style services. JavaServer Pages (JSP) software makes it easy for Internet user interface developers to present dynamically generated pages to anyone with a browser. Servlets give more sophisticated developers of Java technology-based applications the freedom to implement dynamic presentations completely in the Java programming language.

The Client Tier

J2EE supports several types of first-tier clients.

Many J2EE services will be designed to support web browser clients. These services interact with their clients via dynamically generated HTML pages and forms.

More sophisticated services will interact with their first-tier clients by directly exchanging business data. Here, JSPs and Servlets are used to format this business data in a way that is easy for J2EE clients to work with. These clients can be both Java applets running in a web browser and Java technology-based programs.

It is important to note that security is a key part of all multi-tier services. In J2EE, security is handled almost entirely by the platform and its administrators. In most cases, neither the service nor its clients require developer-written security logic.

HTML Page Based Clients

A service can be presented directly to a user's web browser as dynamically generated HTML pages. JavaServer Pages technology is an easy way to dynamically compose these pages using a familiar scripting paradigm that combines HTML and Java technology-based code, as shown in FIGURE 3. In some cases, a service may require some fairly complex code. This can be handled by placing code in a JavaBeans component and calling it from a JSP. A service can also be directly programmed in the Java programming language using a servlet.

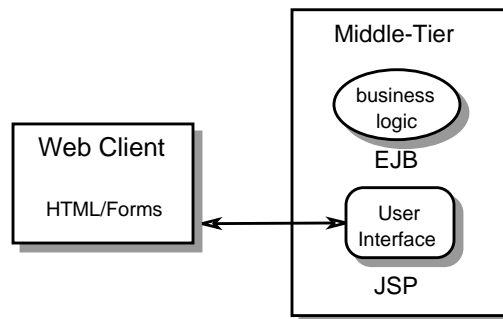


FIGURE 3 Presenting Services Directly to a Browser

HTTP Content Based Clients

It is often useful to provide functionality directly at the client that helps a user organize and interact with the service's information. In this case, the service exchanges raw content with the client instead of HTML pages. This content is typically in the form of XML documents that are exchanged between the client and the service using the HTTP protocol.

Typically this XML content is handled in the first-tier by JavaBeans components that are provided by the service in an applet that is automatically downloaded into a user's browser, as shown in FIGURE 4. To avoid problems caused by old or non-standard versions of the Java

runtime environment in a user's browser, the J2EE application model provides special support for automatically downloading and installing the Java Plug-in. This content can also be handled by a Java technology-based program acting as a J2EE client.

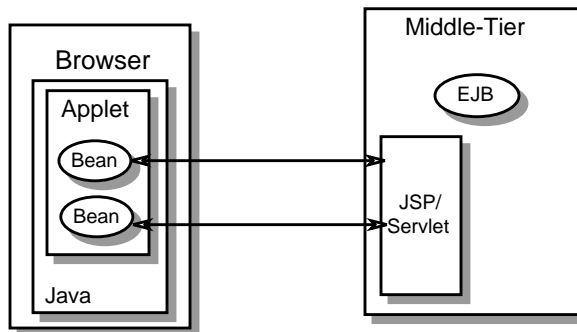


FIGURE 4 Providing JavaBeans components to a Browser

Intranet Clients

Both HTML page based services and HTTP content based services can be effectively used on an enterprise's intranet as well as the Internet.

In addition, the intranet provides the extra infrastructure that allows Java programs to directly access EJBs within the intranet domain.

Other Client Types

J2EE services presented via standard HTTP, HTML and XML are easily accessible to all clients including Microsoft clients such as Visual Basic and Office 2000.

The stated goal of EJB is to define CORBA standard RMI-IIOP as the required interoperability mechanism. This will make any a J2EE service available to any CORBA client. While most elements of this standard are complete there are a few items that are still in progress. After the final work is complete, J2EE will add this interoperability requirement. In the interim, many J2EE vendors will support the parts of the standard that are available.

In conjunction with J2EE, Sun will provide whitepapers and technology demonstrations that illustrate techniques for integrating Microsoft COM objects with EJBs using RMI-IIOP. These will cover how to access EJBs from first-tier clients such as Visual Basic and Windows 2000 via COM, as well as using EJBs in combination with middle-tier functions implemented in Microsoft Transaction Server.

The Enterprise Information Systems

A service's middle-tier business functions must access and update the information in the EIS-tier.

The following standard Java service APIs provide basic access to these systems:

- **JDBC™** - the standard API for accessing relational data from Java.
- **Java Naming and Directory Interface™ (JNDI)** - the standard API for accessing information in enterprise name and directory services.
- **Java™ Message Service (JMS)¹** - the standard API for sending and receiving messages via enterprise messaging systems like IBM MQ Series and TIBCO Rendezvous.
- **JavaMail™** - the standard API for sending E-mail.
- **JavaIDL** - the standard API for calling CORBA services.

J2EE Declarations

An important goal of the J2EE application model is to minimize application programming.

One of the ways that this is accomplished is to shift the burden of implementing common tasks to the J2EE platform. These common tasks include enforcing an application's security roles, implementing its transaction semantics, and linking its components to the resources and other components they require.

J2EE provides a simple, declarative way to specify these behaviors. These declarations are separated from component code and stored in a *deployment descriptor* that is packaged with the application. This allows these behaviors to be modified without having to modify the components themselves.

J2EE Platform

The J2EE platform is the standard environment for running J2EE applications. The J2EE platform is composed of the following elements:

- **J2EE deployment specification** - a standard that defines a common way of packaging applications for deployment on any J2EE compatible platform.
- **Java technology standards for the J2EE platform** - a set of standards that all J2EE platform products must support.

1. JMS is not required for J2EE 1.0; however, it will be made mandatory in a later release.

- **IETF standards for the J2EE platform** - a set of Internet standards that all J2EE platform products must support.
- **CORBA standards for the J2EE platform** - a set of CORBA standards upon which the J2EE platform bases its middle-tier interoperability.

The J2EE platform defines the rich set of facilities that are needed to implement enterprise-class, multi-tier services. The J2EE platform is based on proven, open standards to deliver the broadest adoption and highest level of portability.

J2EE Application Assembly and Deployment

A J2EE application is packaged into one or more standard units for deployment to any J2EE platform-compliant system.

Each unit contains a standard deployment descriptor that describes its content and also includes the J2EE declarations which have been specified by the application developer and assembler.

In addition to its deployment descriptor, a J2EE unit can contain a full mix of the elements that make up a J2EE application.

Once a J2EE unit has been produced, it is ready to be deployed to a J2EE platform, as shown in FIGURE 5.

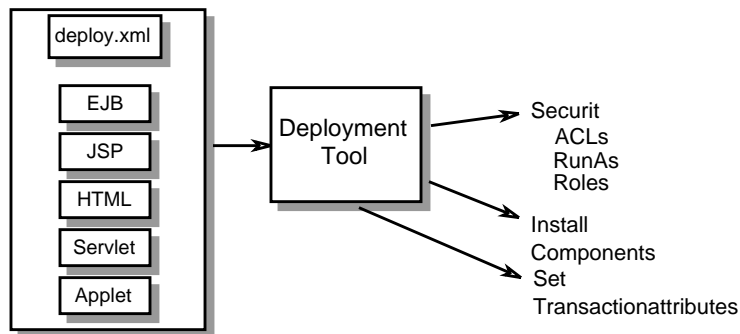


FIGURE 5 Deploying J2EE Applications

Deployment typically involves using a platform's deployment tool to specify location-specific information, such as a list of local users that can access it and the name of the local database. Once deployed on the local platform, the application is ready to run.

Java Technology Standards for the J2EE Platform

The primary element of the J2EE platform is the list of Java technology standards that all J2EE products are required to support.

Since the J2EE platform is focused on the end-to-end development of enterprise services, it goes beyond simply requiring that each Java API be supported. It requires that each API be fully integrated with the platform. This insures that the platform delivers a consistent end-to-end environment for the deployment of J2EE applications.

IETF Standards for the J2EE Platform

The emergence of the Internet has had a major impact on the way enterprise applications are developed. The foundation of this revolution is the IETF standards for HTML, HTTP, and now XML.

The Java programming language, having grown up with these standards, has become the preferred way of writing applications for them. The J2EE application model and the J2EE platform continue this trend.

CORBA Technology Standards for the J2EE Platform

The Object Management Group in conjunction with Sun has produced the RMI/IIOP specification. This standard defines how the CORBA IIOP protocol is used by the Java Remote Method Invocation facility.

The EJB specification uses the application mapping for RMI/IIOP as its standard for calling EJBs. J2EE strongly supports the use of RMI/IIOP and it will become the EJB interoperability standard after OMG resolves a few outstanding details having to do with security interoperability.

J2EE Compatibility Test Suite

J2EE platform vendors will need to verify that their implementations conform to the J2EE platform specification. Toward that end, Sun Microsystems, Inc. will license to platform vendors the J2EE Compatibility Test Suite (CTS). J2EE compatibility will be a self-certification branding program.

Licensees will deploy, configure, and run this test suite (via its GUI framework) on their platform implementations. The suite will include tests for ensuring that the J2EE APIs are implemented. The tests will verify that the J2EE component technologies are available and working together properly. It will include full J2EE applications and verify that all platforms are capable of deploying and running them.

Upon successful completion of the J2EE Compatibility Test Suite, vendors gain the brand for their implementation of Java™ 2, Enterprise Edition.

J2EE Reference Implementation

The J2EE reference implementation fulfills several roles.

Its primary role is as an operational definition of the J2EE platform. In this role, it is used by vendors as the J2EE platform's "gold standard" to determine what their implementation must do under a particular set of application circumstances. It is also used by developers to verify the portability of an application. Most importantly, it is used as the standard platform for running the J2EE Compatibility Test Suite.

A secondary, but more visible, role for the reference implementation is as a freely available platform for popularizing Java 2 platform, Enterprise Edition. Although it is not a commercial product and its licensing terms will prohibit its commercial use, it will be freely available in binary form for demonstrations, prototyping and education.

The reference implementation will also be made available in source form. The specifics of how this will be handled will be determined later.

Example: A Web Store

The Java™ 2 Platform, Enterprise Edition can be used to build a wide range of services. This chapter describes how the J2EE was used to create a web store—CHEAPBOOKS.COM. As the example shows, the J2EE middle-tier provides the complete foundation upon which this service's presentation, business logic, and EIS access is built. FIGURE 6 illustrates the J2EE components that were used to create the Web store sample application.

The most significant thing about this web store example is what is *not* illustrated. The normal complexity you would expect to see is not here because it did not have to be built by the developers of the store. Instead, the developers used the J2EE platform's built-in support for middle-tier development.

The J2EE application model guided the developers of the web store through the development process. It simplified their work by naturally separating the components used for presenting the store (JSPs) from those that implement the store's business processes (EJBs). In both areas, the J2EE platform allowed the developers to dedicate their time to the store-specific tasks – the J2EE platform handled the store's complex system demands.

Java Server Pages simplified the task of customizing the store's presentation to individual customers. JSPs provide an easy-to-use combination of HTML and Java that efficiently generates a customer-specific view of the store. Since JSP also provides a built-in facility for calling Enterprise JavaBeans™, this allowed the store's presentation components to simply and directly access its business functions.

Enterprise JavaBeans significantly reduced the effort needed to build the store's business functions. Instead of having to write complex state management code, the developer built the shopping cart EJB and let the J2EE platform automatically manage its state. Instead of spending time designing and building a database connection manager, the developer of the catalog EJB let J2EE handle connection management. Since EJB makes it easy to combine the multiple actions required to process an order into a single transaction, the problems caused by partial failure of a purchase are avoided.

Since J2EE provides back-office access APIs that are integrated as a standard part of the platform, the store's business functions were developed and integrated with the company's existing systems, all within one consistent environment.

The productivity of the Java™ platform plus the sophisticated middle-tier facilities provided by J2EE significantly reduced the time required to get the store operational. Because J2EE is an open standard, the store developers were able to select the vendor and computing hardware that best fit their needs. In addition, the wide selection of development and content authoring tools that can be used with J2EE gave the developers the flexibility they needed to build a highly competitive store.

The middle-tier architecture needed to implement the web store is very similar to the architecture needed to implement a wide range of services that directly reach the important wide range of customers, employees, partners and suppliers that an enterprise in today's world must directly and efficiently interact with. Typical examples of these services are customer management, supply chain management, and employee expense accounting.

FIGURE 6 illustrates how the CHEAPBOOKS.COM web store was implemented as a J2EE application.

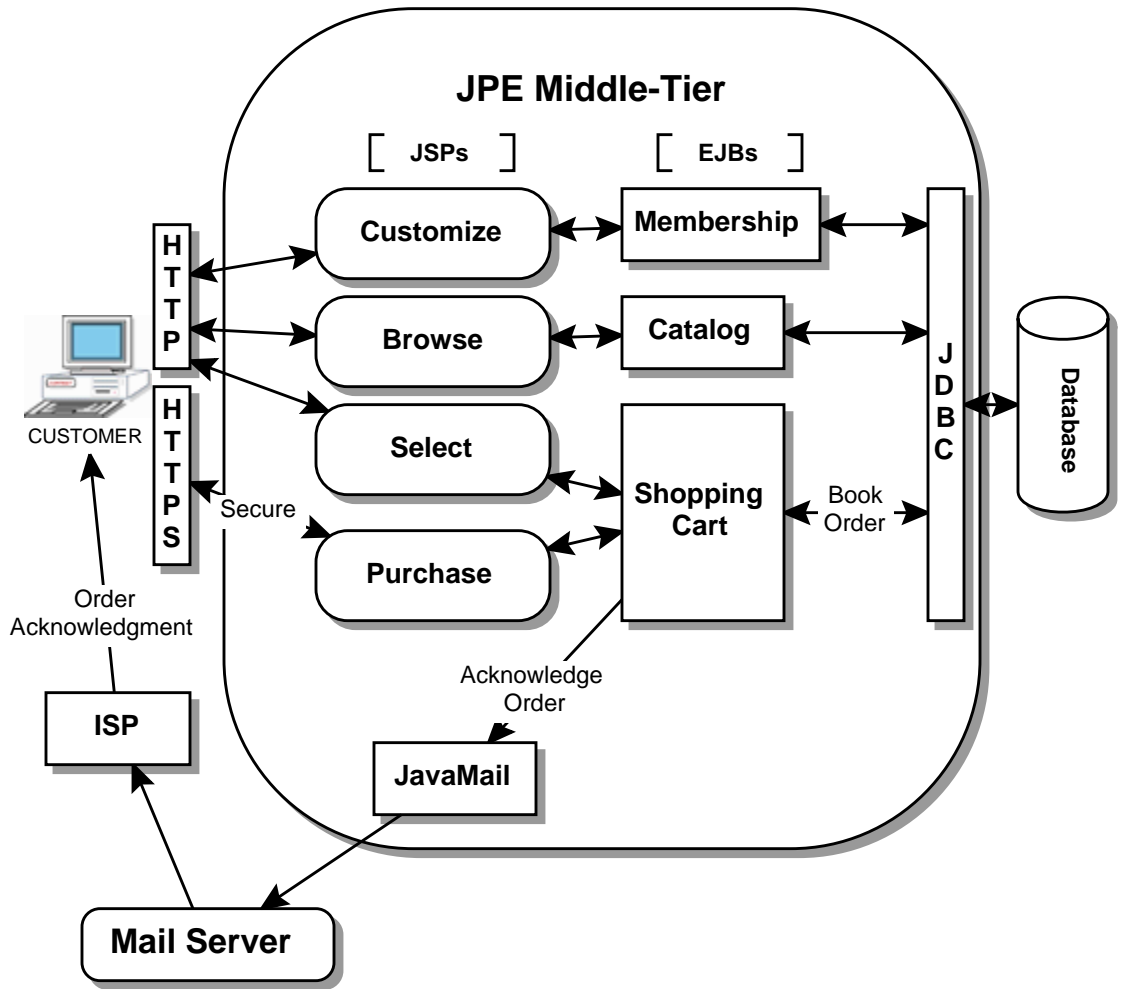


FIGURE 6 CHEAPBOOKS.COM, A Web Store Example

