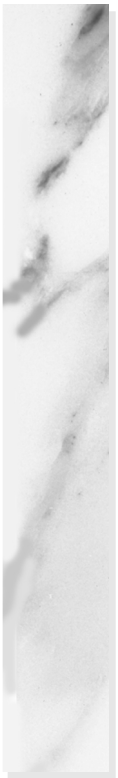
ENTERPRISE JAVABEANS COMPONENT ARCHITECTURE

Designing and Coding Enterprise Applications

INTRODUCTION



Topics in This Chapter

- The Enterprise JavaBeans™ Architecture
- How This Book Is Organized
- Our Vision
- Reader Audience
- About the Examples
- Source Code Online

Chapter 1

Distributed computing has always had tough problems to solve: security, concurrency, database transactions, data integrity, and performance requirements, to name a few. How does the Java 2 Platform, Enterprise Edition (J2EE) address these issues and what is the role of the Enterprise JavaBeans architecture?

As we describe the Enterprise JavaBeans component architecture, let's define where we hope to go, how we hope to get there, and who we hope to take with us. The software industry has exciting offerings right now and in the near future. The architecture of loosely coupled components promises to offer flexibility, scalability, and portability. There is no better time than now to delve into the world of distributed component computing!

1.1 What Is the Enterprise JavaBeans Architecture?

The Enterprise JavaBeans (EJB) architecture is a specification developed at Sun Microsystems. It describes a component-based architecture that provides for the development and deployment of distributed applications. The specification details the services and requirements of an application server which manages EJB components. It also describes coding requirements that bean developers must follow to create portable applications. The lofty and worthwhile goal is

for bean developers to write EJB components once and deploy them on any application server that is compliant with Enterprise JavaBeans technology. Furthermore, the EJB architecture makes enterprise applications scalable, secure, and transactional.

Enterprise JavaBeans are components that execute within an "EJB container," under the supervision of an application server. There are three main EJB types: session, entity, and message driven. Furthermore, session beans come in two flavors: stateless and stateful. The application server and EJB container provide system services for EJBs, such as data persistence, transactions, security, and resource management. The EJB container maintains pools of database connections, as well as pools of EJB instances that can be assigned to clients as needed.

The Java 2 Platform, Enterprise Edition (J2EE) is an industry-standard suite of Java APIs for enterprise computing from Sun Microsystems. It includes the Enterprise JavaBeans architecture and a set of related packages that make everything work together. For example, a Java client may use the Java Naming and Directory Interface (JNDI) to look up the location of an EJB component. The application server, which provides the system services that make EJBs work, uses Java Remote Method Invocation (RMI) and RMI-IIOP to implement remote calls across a network. Message-driven beans, a new type of EJB, use the Java Message Service (JMS) to provide a bean capable of responding to messages. So, while Enterprise JavaBeans technology provides specific services in the realm of enterprise computing, it is part of a larger picture. This picture is inscribed by J2EE and its many independent packages that provide specific services.

Why should developers care about J2EE and EJB? Because the application server manages an EJB and provides the system services we mentioned, bean developers can concentrate on designing enterprise applications that adhere to specific business requirements. Instead of writing transactional database code, bean developers can pay attention to business rules, business processes, and how to best keep track of business data. Furthermore, as J2EE technology matures, commercial application servers that support this specification will become more numerous. Better still, as these J2EE application server products improve, the same enterprise application written today will perform better tomorrow—unchanged—because the application server will provide a better implementation.

1.2 How This Book Is Organized

The chapters follow a gradual progression from the simplest type of EJB (stateless session with no database access) to a complete enterprise application with

eight EJBs, one web component, and multiple application clients. It's best to read the chapters in this order.

Chapter 2 begins with an overview of the Enterprise JavaBeans architecture. This chapter introduces the J2EE components and explains the role of the application server and container in managing EJBs and the system services they use. You should read this material first if you are new to enterprise beans, but even experienced developers may find it necessary to return to this chapter from time to time. By doing this, you discover how everything fits into the big picture.

Chapter 3 introduces the simplest EJB component, the stateless session bean. Our example, Loan EJB, is a component that produces amortization tables and monthly payment amounts for long term fixed-rate loans. We introduce the Value Object Pattern. After giving you an overview of Java Server Pages (JSP), we also present our first web component client.

Chapter 4 presents Java Database Connectivity (JDBC) with EJBs. In this example, we use a stateless session bean as a database reader. After introducing the basics of JDBC, we show you how to use an EJB for read-only database access. We implement the Data Access Object (DAO) Pattern and explain how to customize an application with a deployment descriptor. The chapter also presents a Swing application client (complete source listing found in Appendix A) and a JSP web component client.

Chapter 5 presents stateful session beans and contrasts them with stateless session beans. This is the beginning of our online shopping application example. Our stateful session bean, MusicCart EJB, is a virtual shopping cart that holds items selected by customers running a JSP web component client. We also introduce the Value List Iterator Pattern and explain EJB local interfaces.

Chapter 6 introduces entity beans. Although our example uses Bean-Managed Persistence (BMP), the reader should be familiar with the material in this chapter because much of it applies to entity beans in general. We present entity bean finder and home methods. Our example is Customer EJB, an isolated "customer" entity bean with BMP. We use the DAO pattern for the persistence implementation. We introduce the very important Session Facade Pattern and local interfaces with entity beans. We enhance our JSP web component client to perform customer lookup and verification against the persistent datastore.

Chapter 7 continues with entity beans. We now explore Container-Managed Persistence (CMP) and Container-Managed Relationships (CMR). We introduce EJB QL, the J2EE query language required to specify custom finder semantics and select methods. Our example includes three related entity beans: Customer EJB, Order EJB, and LineItem EJB. We describe the expanded role of local interfaces and revisit the Session Facade Pattern. The JSP web component client creates customer orders using the data collected in the stateful session MusicCart EJB presented in Chapter 5. The full application now contains seven EJBs, one web component, and an administrative client to inspect the database.

Chapter 8 introduces message-driven beans, the newest EJB component. We begin with an overview of the Java Message Service (JMS) and explain the Publish-Subscribe and Point-to-Point messaging models. Messaging provides a loosely coupled architecture that can enhance the performance of enterprise applications. Our first example is a Message-Driven Bean (MDB) that provides a response mechanism to a client. Our final example is a Java client which sends a message to a ShipOrder MDB. The ShipOrder MDB interfaces with the previously written Session Facade to request the shipment of certain customer orders.

1.3 Our Vision

Enterprise JavaBeans technology is not an easy topic. Yet we believe EJB to be a significant offering in the enterprise computing arena. For the first time, there exists a specification that allows bean developers to write transactional, multiuser, scalable enterprise applications without being experts in transactions, multithreaded programming, security, or database programming. Pulling the enterprise system services out of the components and standardizing them within an application server has tremendous benefits. The most obvious benefit is that we leave the implementation of these system services to developers who *are* experts in enterprise issues. As the technology matures, the application servers will get better, and our enterprise applications will in turn run better.

Does this make EJB authoring simplistic? Not hardly. But it makes EJB authoring accessible and portable across multiple platforms. Our vision is to create a text with examples that teach, not just how to create an EJB, but how to design components that work best within the framework of a particular application. We strive to create examples that show you when to use an entity bean, not as an isolated entity bean example, but as a component within a whole, real-world application.

The industry is learning as pockets of developers here and there build on experiences. We learn what works and what doesn't. And from these battles with reality emerge design patterns that address common problems. So, along with our examples, we've also attempted to attack some of these problems. We show you how to apply commonly accepted design patterns to your enterprise designs (and why). Our hope is that you can take our examples and build your own solutions more quickly with an understanding of the design trade-offs that you'll make.

As you wind your way through these examples, we hope your EJB journey will be both exciting and rewarding.

1.4 Reader Audience

We are writing to an audience of Java programmers. If you're reading this book, then you're interested in enterprise computing. A complete enterprise application entails many parts: one or more clients, an application server to manage the EJBs you write and deploy, and a Java Virtual Machine (JVM) to execute Java bytecode. Clients come in many flavors: a command-line Java program, a Java Server Pages (JSP) web component, or even a client that uses Java's Swing GUI are all common types of clients. Some familiarity with database software is helpful. Java Database Connectivity (JDBC) is the Java API for portable database operations. While we don't assume you know JSP or JDBC, we do provide only an overview of these subjects—enough so that you can follow and understand our examples.

We take a similar approach with our overviews of the Java Message Service (JMS) and the EJB Query Language (EJB QL). JMS underlies the implementation of message-driven beans. Bean developers use EJB QL to compose custom queries for CMP entity beans.

1.5 About the Examples

Enterprise JavaBeans Component Architecture is an example-driven book. Our goal is to show you how to design effective EJB components through our examples. Each example teaches you some aspect of EJB design. For instance, our first example (Loan EJB) illustrates the design and use of a stateless session bean (see "The Loan Enterprise Bean" on page 36).

The enterprise computing community has developed a rich set of design patterns. Where possible, we apply accepted design patterns within our examples, explaining the benefit that each pattern brings, as well as presenting its complete implementation. See, for example, the Session Facade Pattern implemented in several examples ("Session Facade Pattern" on page 250 as well as "Session Facade Pattern" on page 329).

We've attempted to bring an element of real-world problem solving to our program examples. At the same time, we avoid solutions that entail layers and layers of abstraction. We hope the developer can recognize the types of enterprise applications our examples represent and use our code as a starting point for further development. For example, we've written a JSP web component client that allows a user to log into a "system" based on a user name and a password. An underlying database stores "customer data," and several enterprise Java beans provide the database lookup and verification steps. There's hardly a customer-based web site that doesn't need a similar capability (see "Web Component Client" on page 262).

Note that besides presenting code for all our EJB examples, we also present complete code for the clients. It may be surprising to see JSP programs in a book about EJB, but our belief is that the examples become more useful when you see how they work within a complete application. Also, an EJB is a component—and components can have all sorts of clients. Thus, we show you a JSP client and a stand-alone Java client with the same EJB.

We've developed and deployed all our examples using the Sun Microsystems Reference Implementation. We run the J2EE server on a Solaris Machine, but have tested our clients on several platforms including Windows and Red Hat Linux.

Table 1.1 describes the different examples. Some chapters include a single application (one or more clients with one or more EJBs). Other chapters include multiple examples built on earlier ones. This table will be helpful as you install and run the examples on your own system.

Table 1.1 Examples by Chapter

Chapter and Topics	*Example Components*	*Source Directory*
Chapter 3: Stateless Session Beans		`loanSession`
Stateless Session Bean	Loan EJB	
Application Exceptions	LoanObjectException	
Value Object Pattern	LoanVO	
Stand-alone Java client	AmortClient	
JSP web component client	paymentGet	
	paymentPost	
Chapter 4: Session Beans with JDBC		`musicSession`
Session Beans with JDBC	Music EJB	
Database Reader		
Application Exceptions	NoTrackListException	
Value Object Pattern	RecordingVO	
	TrackVO	
Java Swing client	MusicApp	(see Appendix A)
JSP web component client	musicGet	
	musicPost	

Table 1.1 Examples by Chapter *(continued)*

Chapter and Topics	*Example Components*	*Source Directory*
Chapter 4: Session Beans with JDBC Data Access Object Pattern		`musicDAO`
DAO Pattern	Music EJB with DAO	
Factory Pattern	MusicDAOFactory	
System Exceptions	MusicDAOSysException	
Naming Environment Entry		
Chapter 5: Stateful Session Beans		`musicVL`
Stateful Session Bean	MusicCart EJB	
Value Object Pattern	CustomerVO	
Value List Iterator Pattern	MusicIterator EJB	
	MusicPage EJB	
JSP web component client	login	
	loginPost	
	musicCart	
	shoppingPost	
Chapter 5: Stateful Session Beans Local Interfaces		`musicLocal`
Local interfaces	MusicPage EJB	
Local vs. Remote interfaces	MusicIterator EJB	
Stateful vs. Stateless		
Chapter 6: Entity Beans with BMP		`customerDAO`
Bean-managed persistence	Customer EJB	
Finder/Home methods		
Transactions		
Local Interfaces		
DAO Pattern	CustomerDAO	
Session Facade Pattern	CustomerSession EJB	
Value Object Pattern	CustomerVO	
Test Client	CustomerTestClient	

Table 1.1 Examples by Chapter *(continued)*

Chapter and Topics	*Example Components*	*Source Directory*
JSP web component client	signUp	
	signUpPost	
	loginPost	
Chapter 7: Entity Beans with CMP		`ordersCMP`
Container-managed persistence	Customer EJB	
Relationship fields	Order EJB	
EJB Query Language	LineItem EJB	
Finder/Select Methods		
Session Facade Pattern	CustomerSession EJB	
Value Object Pattern	OrderVO	
	LoanVO	
JSP web component client	processOrder	
	submitOrder	
Administrative client	AdminClient	
Chapter 8: Message-Driven Beans		
JMS message queues	PingServer	
Publish/Subscribe Pattern	SchoolApp client	`School`
	Student MDB	
Point-to-Point Pattern	OrderApp client	`ShipOrders`
EJB integration	ShipOrder MDB	

1.6 Source Code Online

We maintain all the source code in this book at an FTP site. You can reach this site through our web site at **`http://www.asgteach.com`**.