Part VIII
Java EE Supporting Technologies

Part VIII explores several technologies that support the Java EE platform.

This part contains the following chapters:

- Chapter 42, Introduction to Java EE Supporting Technologies
- Chapter 43, Transactions
- Chapter 44, Resource Connections
- Chapter 45, Java Message Service Concepts
- Chapter 46, Java Message Service Examples
- Chapter 47, Advanced Bean Validation Concepts and Examples
- Chapter 48, Using Java EE Interceptors

Introduction to Java EE Supporting Technologies

The Java EE platform includes several technologies and APIs that extend its functionality.

These technologies allow applications to access a wide range of services in a uniform manner.

These technologies are explained in greater detail in Chapter 43, Transactions and Chapter 44, Resource Connections, as well as Chapter 45, Java Message Service Concepts, Chapter 46, Java Message Service Examples, and Chapter 47, Advanced Bean Validation Concepts and Examples.

The following topics are addressed here:

- Transactions
- Resources
- Java Message Service

Transactions

In a Java EE application, a transaction is a series of actions that must all complete successfully, or else all the changes in each action are backed out.

Transactions end in either a commit or a rollback.

The Java Transaction API (JTA) allows applications to access transactions in a manner that is independent of specific implementations.

JTA specifies standard Java interfaces between a transaction manager and the parties involved in a distributed transaction system: the transactional application, the Java EE server, and the manager that controls access to the shared resources affected by the transactions.

The JTA defines the `UserTransaction` interface that applications use to start, commit, or abort transactions.

Application components get a `UserTransaction` object through a JNDI lookup by using the name `java:comp/UserTransaction` or by requesting injection of a `UserTransaction` object.

An application server uses a number of JTA-defined interfaces to communicate with a transaction manager; a transaction manager uses JTA-defined interfaces to interact with a resource manager.

See Chapter 43, Transactions for a more detailed explanation.

The JTA 1.1 specification is available at http://www.oracle.com/technetwork/java/javaee/tech/jta-138684.html.

Resources

A resource is a program object that provides connections to such systems as database servers and messaging systems.

The Java EE Connector Architecture and Resource Adapters

The Java EE Connector Architecture enables Java EE components to interact with enterprise information systems (EISs) and EISs to interact with Java EE components.

EIS software includes such kinds of systems as enterprise resource planning (ERP), mainframe transaction processing, and nonrelational databases.

Connector architecture simplifies the integration of diverse EISs.

Each EIS requires only one implementation of the Connector architecture.

Because it adheres to the Connector specification, an implementation is portable across all compliant Java EE servers.

The specification defines the contracts for an application server as well as for resource adapters, which are system-level software drivers for specific EIS resources.

These standard contracts provide pluggability between application servers and EISs.

The Java EE Connector Architecture 1.6 specification defines new system contracts such as Generic Work Context and Security Inflow.

The Java EE Connector Architecture 1.6 specification is available at http://jcp.org/en/jsr/detail?id=322.

A resource adapter is a Java EE component that implements the Connector architecture for a specific EIS.

A resource adapter can choose to support the following levels of transactions:

- `NoTransaction`: No transaction support is provided.

- `LocalTransaction`: Resource manager local transactions are supported.

- `XATransaction`: The resource adapter supports the XA distributed transaction processing model and the JTA `XATransaction` interface.

See Chapter 44, Resource Connections for a more detailed explanation of resource adapters.

Java Database Connectivity Software

To store, organize, and retrieve data, most applications use relational databases.

Java EE applications access relational databases through the JDBC API.

A JDBC resource, or data source, provides applications with a means of connecting to a database.

Typically, a JDBC resource is created for each database accessed by the applications deployed in a domain.

Transactional access to JDBC resources is available from servlets, JavaServer Faces pages, and enterprise beans.

The connection pooling and distributed transaction features are intended for use by JDBC drivers to coordinate with an application server.

For more information, see `DataSource` Objects and Connection Pools.

Java Message Service

Messaging is a method of communication between software components or applications.

A messaging system is a peer-to-peer facility: A messaging client can send messages to, and receive messages from, any other client.

Each client connects to a messaging agent that provides facilities for creating, sending, receiving, and reading messages.

The Java Message Service (JMS) API allows applications to create, send, receive, and read messages.

It defines a common set of interfaces and associated semantics that allow programs written in the Java programming language to communicate with other messaging implementations.

The JMS API minimizes the set of concepts a programmer must learn in order to use messaging products but provides enough features to support sophisticated messaging applications.

It also strives to maximize the portability of JMS applications across JMS providers in the same messaging domain.