

Contents

Foreword	xv
Acknowledgments	xvi
Introduction	xvii
Chapter 1: Introduction to Enterprise Java Computing	1
Key Enterprise Technologies	2
Using Java for Enterprise Application Development	5
Summary	6
Chapter 2: Java Database Connectivity	7
JDBC Architecture	8
JDBC Driver Types	9
Type 1: JDBC-ODBC bridge plus ODBC driver	9
Type 2: Native-API partly Java driver	10
Type 3: JDBC-Net pure Java driver	10
Type 4: Native-protocol pure Java driver	10
Selecting a JDBC Driver	10
JDBC URLs	13
Java/SQL Type Mappings	13
Using the JDBC API	14
Loading a driver	15
Connecting to a database	15
Executing SQL statements	17
Optimizing Queries with Prepared Statements	22

viii **Enterprise Java Computing**

Calling Stored Procedures	24
SQL Escape Syntax	25
Metadata	26
Transaction Management	30
Transaction Isolation Levels	31
Implementing Connection Pools	32
A Three-Tier Solution: Integrating RMI with JDBC	34
JDBC 2.0	43
Implementing scrollable cursors	43
Updating result sets	45
Making batch updates	49
Handling SQL3 datatypes	50
JDBC 2.0 standard extensions	51
Summary	51
 Chapter 3: Deploying Java Servlets	 53
Why Servlets Are Better than CGI Scripts	54
The Java Servlet Development Kit	55
Using servletrunner	56
Understanding the Servlet Life Cycle	57
Demonstrating Session-Tracking: The Shopping Cart Servlet	62
HttpSessionBindingListener Interface	72
Using Cookies for Client-Side Persistency	72
The SingleThreadModel Interface	74
Integrating Servlets with JDBC	75
A three-tier example	75
Tips for servlet/JDBC integration	82
Integrating Servlets with RMI	84
Servlets as RMI clients	84
RMILEts—servlets as remote objects	90
Applet-Servlet Communication Using Object Serialization	95
Summary	101

Chapter 4: Melding Java with Legacy Systems Using JNI 103

 Introducing Native Methods 104

 Why do we need native methods? 104

 Disadvantages of native methods 105

 Current support for native methods 105

 JNI Design Goals 106

 What Can You Do with JNI? 106

 Understanding the JNI Life Cycle 107

 1. Write the Java code 108

 2. Compile the Java source 108

 3. Generate the JNI-style header file 108

 4. Implement the native method 110

 5. Create the shared library or DLL 110

 6. Run the Java program 111

 Mapping Between Java and C Data Types 111

 Accessing Java objects from Native Methods 112

 Accessing instance data 112

 Accessing static data 114

 Obtaining JVM Type Signatures 116

 Accessing Java Arrays 118

 Accessing Java Methods from Native Methods 121

 Accessing instance methods 121

 Accessing static methods 125

 JNI Exception Handling 126

 Multithreaded Programming with JNI 128

 Using the Invocation API 133

 Summary 136

Chapter 5: Object Serialization 137

 Fundamentals of Object Serialization 138

 The Serializable Interface 139

 Handling object references 142

 Protecting sensitive information 143

x **Enterprise Java Computing**

Implementing Custom Serialization	144
Transmitting Objects over Sockets	147
A client/server example	147
Transmitting objects over compressed sockets	150
Advanced Serialization	154
The externalizable interface	154
Authenticating serialized objects using message digests	155
Encrypting Serialized Objects via Sealed Objects	157
Understanding Object Versioning	162
Versioning serialized classes	162
Compatible and incompatible changes	162
Using serialver	163
Serialization—Advantages and Disadvantages	164
Summary	165
 Chapter 6: Remote Method Invocation	 167
Developing an RMI System: A Stepwise Approach	168
1. Define the remote interface	169
2. Develop the remote object by implementing the remote interface	171
3. Develop the client program	174
4. Compile the Java source files	175
5. Generate the client stubs and server skeletons	175
6. Start the RMI registry	176
7. Start the remote server objects	178
8. Run the client	179
Understanding Remote Polymorphism	180
Implementing RMI Callbacks	181
Developing an RMI system with callbacks	182
Distributed Garbage Collection	191
The unreferenced interface	191
The leaseValue property	191
Working with Firewalls	192
Remote Object Activation	194
Implementing Custom Sockets	199
The rot13 Cipher	200

	Contents	xi
1. Create the custom socket implementing rot13 encryption	202	
2. Create a custom client-side socket factory by implementing RMIClientSocketFactory.	206	
3. Create a custom server-side socket factory that implements RMIServerSocketFactory	207	
4. Enable the RMI server with the custom client and server socket factories	208	
5. Select your custom socket type as the transport protocol	209	
6. Run the application	210	
Jini Technology	210	
How Jini works	211	
The runtime services	213	
The echo service	217	
The distributed programming model	224	
Comparing Jini with RMI	226	
Applying Jini Technology	227	
Future Trends	227	
Summary	228	
Chapter 7: Java IDL: Java Meets CORBA	229	
The CORBA Distributed Computing Model	229	
The Object Management Architecture	231	
The CORBA 2.0 Architecture	232	
The Object Request Broker (ORB)	232	
Object Adapters	233	
The Implementation Repository	234	
The Dynamic Skeleton Interface (DSI)	234	
Interface Repository	235	
The Dynamic Invocation Interface (DII)	235	
The Interface Definition Language (IDL)	235	
IDL to Java Mapping	235	
Identifiers, naming, and scope	236	
Generated classes	236	
Holder classes	236	
Helper classes	237	
Mapping for basic types	237	

xii **Enterprise Java Computing**

Mapping for enum	237
Mapping for struct	238
Mapping for union	238
Mapping for ordered collections	238
Mapping for attributes and operations	239
Parameter-passing modes	239
Mapping for user-defined exceptions	240
Mapping for system exceptions	240
Developing CORBA Servers and Clients	240
The TimeServer example	242
Defining the IDL	243
Develop the client application	244
Develop the server application	246
Start the name service	249
Start the CORBA TimeServer	249
Execute the TimeServer client	249
Stringified Object References	250
The InterestRates Server Example	250
Define the IDL	251
Code the server application	251
Code the client	253
Start the CORBA InterestRates server	255
Execute the InterestRates client	255
Common Object Services—The Naming Service	255
Common Object Services—The Event Service	256
Proxy consumers and suppliers	256
Communication models	256
Push model	257
Pull model	257
Putting It All Together—The Bank Example	257
Define the IDL	257
User-defined exceptions	257
The generated code	258
Develop the client	259
Code the AccountImpl servant	260
Code the CheckingAccountImpl servant	261

	Contents	xiii
Develop the server class	263	
Run the Naming Service	264	
Run the CheckingAccount server	264	
Run the client	265	
Comparing CORBA and RMI	265	
Application sample: the StockMarket server and client	265	
Summary	272	
Chapter 8: Enterprise JavaBeans	275	
Background—The Relevant Technologies	276	
What are transactions?	276	
The ACID properties	276	
The Java Transaction Service	277	
Two-phase commits	278	
What is a naming service?	278	
The Java Naming and Directory Interface	278	
JavaBeans—the generic component model	279	
Enterprise JavaBeans—Components at the Server	281	
Why use EJB?	281	
Differences between JavaBeans and Enterprise JavaBeans	283	
The EJB Model	284	
EJB server	285	
EJB containers	285	
The home interface and home object	285	
The remote interface and EJBObject	285	
The Enterprise JavaBean	286	
The EJB clients	286	
The EJB Lifecycle	286	
EJB server provider	287	
EJB container provider	287	
EJB developer	287	
EJB deployer	288	
Application developer	288	
EJB Components	288	
The home interface and the home object	288	
The remote interface	290	

xiv	Enterprise Java Computing	
	The EJBObject	291
	Entity and session beans	291
	Developing an N-Tier EJB Application	294
	Developing entity beans	296
	Developing session beans	306
	Modeling Using Session and Entity Beans	324
	EJB Container	325
	Obtaining services from the container	325
	Relationship between EJB container and the server	327
	Building customized EJB containers	327
	Contract between the client and the EJB container	328
	Contract between a container and an EJB	330
	EJB Servers	331
	Server infrastructure	333
	Deploying EJBs	338
	Future Trends	339
	Summary	339
	Index	341