10 Understanding Java-Related Platforms and Integration Technologies

CERTIFICATION OBJECTIVES

- Understanding Java Platforms
- Working with the Java Remote Method Invocation API
- Working with Database Technologies
- Working with Additional Java Integration APIs

✓ Two-Minute Drill

Q&A Self Test

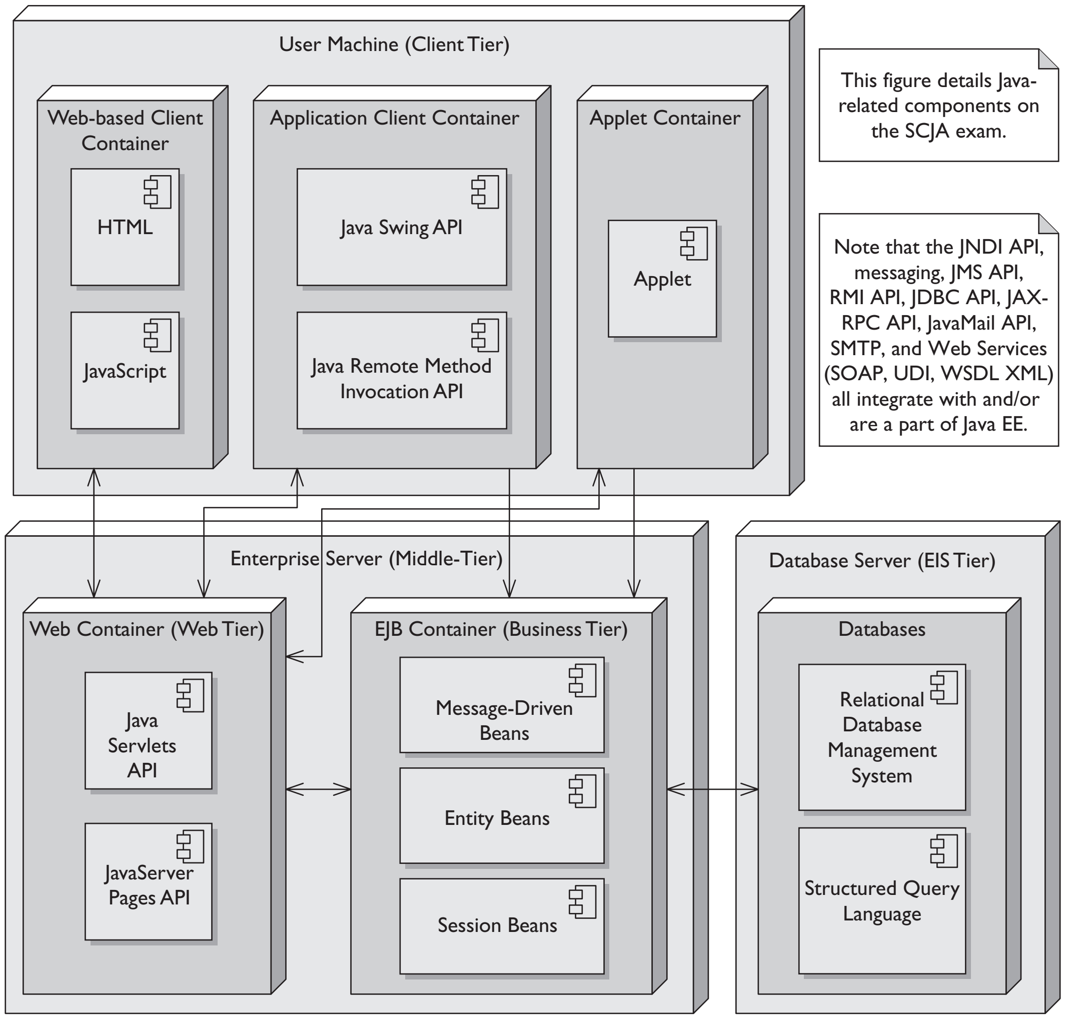
Java-based technologies provide the components and means necessary for creating client, client-server, enterprise, and mobile applications. Figure 10-1 represents a good portion of these technologies. The figure also includes other important technologies that are not Java-based (for example, SQL). Review this figure because all of the technologies represented are on the exam. These technologies are also covered in Chapters 11 and 12, as well as this chapter. The technologies are needed to be understood only from a high level. For example, you'll need to know what JavaServer Pages is, why it's used, and its key benefits. You will not be asked to create a JavaServer Pages web page or describe its low-level details, however. When you are ready to dive in deeper, you can find lower-level types of questions in Sun's specialty exams, such as the Sun Certified Web Component Developer. High-level knowledge, as obtained from this study guide, will help you lead yourself—and/or your team—in selecting the most appropriate and optimal Java-based solution, as well as pass the SCJA exam.

This chapter will provide you with the high-level platform information needed for the exam, discussing the general layout of Java SE, Java ME, and Java EE. The RMI, JDBC, and JNDI integration technologies will also be discussed as needed for the exam. A few thoughts may cross your mind when reading through these sections such as, "Specifically, how do I develop with these technologies? Why isn't this book showing me any sample code for the APIs?" For the SCJA exam, Coding against the technology APIs, Java ME, and Java EE platforms is beyond the scope of the SCJA exam. You simply need to know what the technologies are, their benefits, and when to use them.

exam watch

You may see the original shorthand for Java platforms in the exam referenced as J2SE, J2ME, and J2EE. Currently, the platforms are known as Java SE, Java ME, and Java EE, respectively.

FIGURE 10-1 Java technologies


User Machine (Client Tier)
Web-based Client Container
HTML
JavaScript
Application Client Container
Java Swing API
Java Remote Method Invocation API
Applet Container
Applet
This figure details Java-related components on the SCJA exam.
Note that the JNDI API, messaging, JMS API, RMI API, JDBC API, JAX-RPC API, JavaMail API, SMTP, and Web Services (SOAP, UDI, WSDL XML) all integrate with and/or are a part of Java EE.
Enterprise Server (Middle-Tier)
Web Container (Web Tier)
Java Servlets API
JavaServer Pages API
EJB Container (Business Tier)
Message-Driven Beans
Entity Beans
Session Beans
Database Server (EIS Tier)
Databases
Relational Database Management System
Structured Query Language

CERTIFICATION OBJECTIVE

Understanding Java Platforms

Exam Objective 6.1 Distinguish the basic characteristics of the three Java platforms: J2SE, J2ME, and J2EE, and given a high-level architectural goal, select the appropriate Java platform or platforms.

The Java SE platform is designed primarily for client-side solutions; the Java ME platform is designed for embedded solutions; and the Java EE platform is designed for enterprise solutions. To pass the SCJA exam, you'll need to understand what each platform offers and when you will need to select a particular platform to help you accomplish a specific goal.

Let's do a brief case study. You've spent several years working with teams of people on several projects. However, you've recently started a new job being the IT director for a small firm. You show up on the first day to find out that not only are you the IT director but you will be playing the role of senior architect, designer, and developer over a small group of junior programmers. The owner of the firm comes to you and says, "There are a few improvements that I've wanted to add to our Java-based security system for a while now. There have been a lot of break-ins in the area in the last couple of months so I need a system that is more informative. So here are the requirements. See if you guys can have the improved system up and running in 30 days." He hands you a small piece of paper with the following information:

- Requirements for alarm system improvements:
  - Must be able to record all security events into a database.
  - Must be able to administer the system with a local application.
  - Must automatically receive alarm notification via e-mail.
  - Must be able to disable or enable the security system from my cell phone.
  - Must be able to view the audit log from my home computer.
  - Must be able to authenticate remote logins from a preexisting naming and directory service.

Being new to the computer and maybe even Java technologies, you may look at this list and think there is no way this can be done in such a small amount of time. However, if you were Java-technology savvy, you would be able to quickly associate a technology with each requirement. You would then be able to work up a quick architecture and task out the pieces of the assignment to the different task members. Let's take a look at breaking things up:

- Alarm system improvements, forward plan:
  - Task programmer A to build a SQL database and interface code with the JNDI API of the Java SE platform.
  - Task programmer B to build a client application administration tool using the Swing API of the Java SE platform.
  - Task programmer C to write a notification module using the JavaMail API of the Java EE platform.
  - Task programmer D to build a cell phone application to enable or disable the alarm system using the Java ME platform.
  - Task programmer E to build a web-based application to log in and view the audit log using JSP and servlets of the Java EE platform.
  - Task programmer F to authenticate remote logins from a preexisting naming and directory service using the JNDI API of the Java SE platform.
  - Task yourself to integrate all of the components together as the developers return them to you.

So now that you know the concept of selecting Java-based technologies for real-world solutions, what you need to know is exactly what the technologies are and when and where you would use them. This section details the platforms that house the technologies; the following sections and chapters detail the technologies themselves:

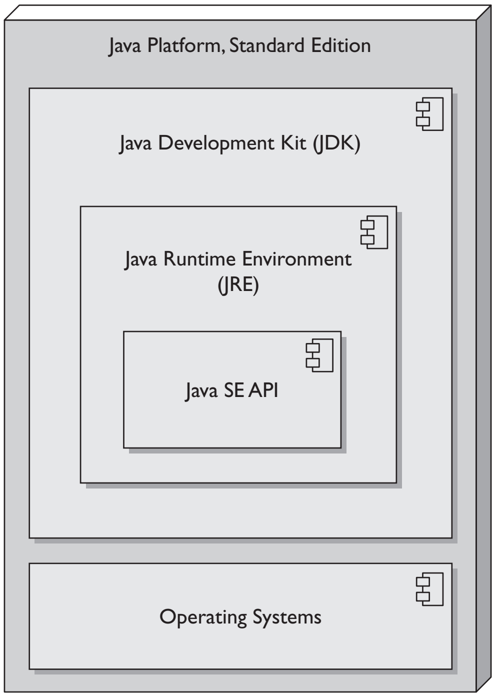
- Java Platform, Standard Edition
- Java Platform, Micro Edition
- Java Platform, Enterprise Edition

Java Platform, Standard Edition

The Java Platform, Standard Edition is composed of the rich Java SE API, the Java Runtime Environment, the Java Development Kit, and the underlying operating system, as shown in Figure 10-2. Each of these elements serves its own specific purpose. Collectively, these components provide the means to develop, deploy, and run client and/or client-server based systems.

FIGURE 10-2

The Java SE platform


Java Platform, Standard Edition
Java Development Kit (JDK)
Java Runtime Environment (JRE)
Java SE API
Operating Systems

The Java SE API

The Java SE API is a collection of software packages. Each package houses a related set of classes and interfaces. All the packages you will need to have general knowledge of are represented in Table 10-1. Many of these packages have subpackages that are not listed since the subpackages won't likely be on the exam.

The Java Runtime Environment

The Java Runtime Environment (JRE) is the set of software that allows Java applications to run. The JRE includes the following items:

- Java Virtual Machines (JVMs)
  - Java Hotspot Client Virtual Machine
  - Java Hotspot Server Virtual Machine

TABLE 10-1 Java Packages Covered in the SCJA Exam

Name	Description	Package Name
Java Abstract Window Toolkit API	Provides native GUI functionality and an event handling model	`java.awt`
Java Basic I/O API	Provides general input/output functionality	`java.io`
Java Database Connectivity API	Provides universal data access	`java.sql`, `javax.sql`
Java Core Language	Provides core Java language classes and interfaces	`java.lang`
Java Naming and Directory Interface API	Provides naming and directories services access	`javax.naming`
Java Networking API	Provides general networking functionality	`java.net`, `javax.net`
Java RMI API	Provides Remote Method Invocation functionality	`java.rmi`
Java Swing API	Provides GUI building functionality	`javax.swing`
Java Utilities API	Provides general utilities including the collections framework, event models, and time facilities	`java.util`

- Deployment Technologies
  - Java Plug-in
  - Java Web Start Technology
  - Java Control Panel
  - Java Update Mechanism

Java is compiled into byte code, and each operating system has its own Java Virtual Machine that will run that byte code. This is what gives Java the "WORA" capability. WORA is the acronym for Sun Microsystems' slogan, "Write once, run anywhere."

The Java Development Kit

The Java Development Kit is, in essence, a developer's toolbox. Not only does it contain the JRE and the Java SE API, but it also contains all of the utilities you will need to compile, test, and debug your applications. Table 10-2 lists the common

TABLE 10-2

Common Development Tools Used as Part of the JDK

JDK Tool	Description
jar	The Java archiving tool
java	The Java interpreter
javac	The Java compiler
javadoc	The API documentation generator
javap	The class file disassemble
jconsole	The monitoring and management utility released with Java 5.0
jdb	The Java debugger
JPDA	The Java Platform Debugger Architecture
rmic	The RMI stub and skeleton generator

tools in the JDK that you will be using as a developer. For more information on the Java compiler and interpreter, see Chapter 1.

When you install a version of the JDK to your computer, you will want to take note of its location. This is important since you may need to specify its location in your system's path or point to it with your IDE.

Supported Operating Systems

Sun directly supports the Solaris, Linux, and Microsoft Windows operating systems with fully compliant JVMs, JDKs, and JREs. You can get the latest JRE and JDK here: http://java.sun.com/javase/downloads/. Legacy versions are kept in Sun's

SCENARIO & SOLUTION

You wish to compile Java source code. Exactly where will you find the compiler?	The Java compiler (`javac`) resides in the `bin` folder of the Java Development Kit.
You wish to interpret Java bytecode. Exactly where will you find the interpreter?	The Java interpreter (`java`) resides in the `bin` folders of both the Java Runtime Environment and the Java Development Kit.
You wish to find various Java demonstration applications provided by Sun. Where can you find them?	The Java Development Kit has a `demo` folder that contains various Java demonstration applications. The corresponding source code is also included.

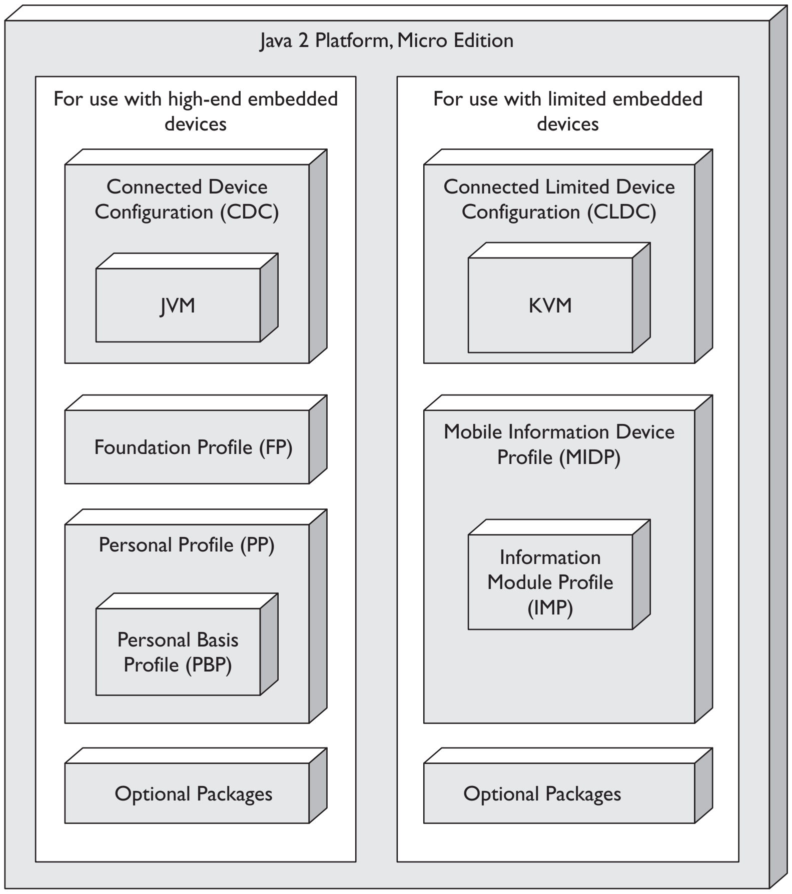
archives here: http://java.sun.com/products/archive/. Third-party sources also make JREs and JDKs available for other operating systems. Many of these third-party solutions are listed and linked here: http://java-virtual-machine.net/other.html.

Java 2 Platform, Micro Edition

The Java 2 Platform, Micro Edition is designed for embedded devices such as high-end PDAs and mobile phones. Java ME's architecture is based on configurations, profiles, and optional packages, as shown in Figure 10-3. For additional detailed information, outside of the following sections, on Java ME and MIDlets see the "Java ME MIDlets" section in Chapter 11.

FIGURE 10-3

The J2ME 1.4 platform


Java 2 Platform, Micro Edition
For use with high-end embedded devices
Connected Device Configuration (CDC)
JVM
Foundation Profile (FP)
Personal Profile (PP)
Personal Basis Profile (PBP)
Optional Packages
For use with limited embedded devices
Connected Limited Device Configuration (CLDC)
KVM
Mobile Information Device Profile (MIDP)
Information Module Profile (IMP)
Optional Packages

Configurations

Java ME has two configurations: the Connected Device Configuration (CDC) and the Connected Limited Device Configuration (CLDC). These configurations contain a virtual machine containing a small but focused set of libraries making up the runtime environment. CDC is supplied with the standard Java Virtual Machine and is utilized for devices that do not have extreme constraints of resources. CLDC has a small compact virtual machine known as Sun's K Virtual machine (KVM) and a reduced set of class libraries.

Profiles

Profiles are necessary to work in conjunction with configurations as part of the necessary runtime environment. Profiles are APIs that define the application's life-cycle model, user interface, and device properties access. CDC contains the Foundation Profile (FP), the Personal Profile (PP), and the Personal Basis Profile (PBP). PBP is a subset of PP. The CLDC contains the Mobile Information Device Profile (MIDP) and the Information Module Profile (IMP). IMP is a subset of MIDP.

Optional Packages

As discussed in detail in Chapter 1, packages are collections of related classes and functionality. Additional packages can be added as needed to expand on Java ME functionality. These packages that can be optionally used are initially excluded (not included by default) in order to keep the Java ME footprint as small as possible.

on the job

When designing a system, it can be beneficial to think outside of the box. For example, consider using peripheral technologies. Unfortunately, many business systems are developed without even considering technologies such as mobile solutions. Can you think of any handheld device or mobile phone application that could be integrated into the architecture of a system you are currently working on?

Squawk

Squawk is a Java-compliant and CLDC-compatible virtual machine implementation, making it a piece of the Java ME architecture. Where most JVMs are written in C and C++, the majority of the Squawk JVM is pure Java. Squawk was designed to be as light as possible and is used with Sun's wireless Small Object Programmable Technology kits (Sun SPOTs). These "hobby" kits which include a 3D accelerometer, temperature and light sensors, LCDs, and push buttons are designed to encourage research and development of mobile technologies. The Squawk JVM is not on the exam, but you can find out more about the Sun SPOT project at www.sunspotworld.com.

Java Platform, Enterprise Edition

The Java Platform, Enterprise Edition provides a means to create true enterprise systems that are flexible, scalable, and secure. A major benefit of enterprise systems is the separation of software components. Java EE follows the Model-View-Controller (MVC) architecture where servlets work as the controller, JavaServer Pages handle the view or presentation logic, and the business logic is represented as the model, typically the Enterprise JavaBeans (EJBs). Servlets, JSPs, and EJBs are covered in Chapter 12. The Java Enterprise Edition requires a collection of optional packages that support each of these areas, as well as complementary technologies. The packages are actually implementations of specifications. Table 10-3 depicts the specifications of the Java EE 5 platform.

A Java Specification Request (JSR) is the description of Java platform–related specifications—proposed and final. For more information on JSRs, visit the Java Community Process (JCP) home page: http://jcp.org/en/home/index. The JCP maintains the JSRs.

Remember that the SCJA exam is currently geared towards the J2EE 1.4 specification. Therefore, J2EE 1.4 APIs are described in this and the following two chapters, which include:

- Enterprise Java Beans 2.1
- Servlet 2.4
- JavaServer Pages 2.0
- Java Message Service 1.1
- JavaMail 1.3
- Web Services 1.1
- JAX RPC 1.1

When developing Java EE systems, you will always need a Java Development Kit. Since the JDK is the main piece of the Java SE platform, you could essentially say that Java SE is part of Java EE. You will often have the option of using newer

TABLE 10-3 Java EE 5 Technology JSRs

JSR	Specification Name	Abbreviation	Version
Web Services Technologies			
JSR-67	Java APIs for XML Messaging 1.0	SAAJ	1.3
JSR-101	Java API for XML-Based RPC	JAX-RPC	1.1
JSR-109	Implementing Enterprise Web Services	N/A	1.2
JSR-173	Streaming API for XML	StAX	1.0
JSR-181	Web Service Metadata for the Java Platform	N/A	2.0
JSR-222	Java Architecture for XML Binding (JAXB) 2.0	JAXB	2.0
JSR-224	Java API for XML-Based Web Services (JAX-WS) 2.0	JAX-WS	2.0
Web Application Technologies			
JSR-52	A Standard Tag Library for JavaServer Pages	JSTL	1.2
JSR-154	Java Servlet 2.4 Specification	Servlets	2.5
JSR-252	JavaServer Faces 1.2	JSF	1.2
JSR-245	JavaServer Pages 2.1	JSP	2.1
Enterprise Application Technologies			
JSR-112	J2EE Connector Architecture 1.5	N/A	1.5
JSR-220	Java Persistence API	N/A	1.0
JSR-220	Enterprise JavaBeans 3.0	EJB	3.0
JSR-250	Common Annotations for the Java Platform	N/A	1.0
JSR-907	Java Transaction API (JTA)	JTA	1.0
JSR-914	Java Message Service (JMS) API	JMS	1.1
JSR-919	JavaMail	N/A	1.4.1
JSR-925	JavaBeans Activation Framework 1.1	JAF	1.1
Management and Security Technologies			
JSR-77	J2EE Management	N/A	1.0
JSR-88	Java EE Application Deployment	N/A	1.2
JSR-115	Java Authorization Contract for Containers	JACC	1.1

versions of the JDK with the Java EE implementation of your choice. Be aware that there is no direct correlation between the Java SE and Java EE version numbers. So you must check the documentation to see which versions of the JDK will work with your Java EE implementation. For a specific example of a case that would work, you could use Sun's JDK 1.5 with Oracle's Application Server 10gR3, which is a J2EE 1.4 implementation.

exam watch

The exam will give you scenarios where you will need to determine which Java platforms are necessary to build a specific application. Understanding which APIs are included in each platform is critical in answering these questions.

The following Scenario & Solution section exercises your knowledge of which platforms contain specific APIs. However, we do not explicitly supply the names of the APIs used—you'll have to figure that out yourself. Review this Scenario & Solution section again after you have completed the chapter.

SCENARIO & SOLUTION

You wish to automate a business process using online forms and a relational database. Which Java editions would you use?	You would need to use the Java Platform, Standard Edition and the Java Platform, Enterprise Edition.
You wish to develop an application to convert database records into XML files. Which Java edition would you use?	You would need to use the Java Platform, Standard Edition.
You wish to develop a simple client-side text editor. Which Java edition would you use?	You would need to use the Java Platform, Standard Edition.
You wish to develop a prize-fighting boxing game for use on a cell phone. Which Java edition would you use?	You would need to use the Java Platform, Micro Edition.
You wish to develop a web-accessible application that also accesses a naming and directory service. Which Java editions would you use?	You would need to use the Java Platform, Standard Edition and the Java Platform, Enterprise Edition.

Getting and staying involved in the Java community is an excellent way to keep up to date with the latest Java Technologies. Consider joining a Java User Group (JUG) to share people's experiences and expertise. You can find out more about JUGs here: http://java.sun.com/community/usergroups/. Online technology forums also offer a rich opportunity for acquiring Java knowledge. Consider frequenting Sun Java forums or the Java Ranch.

EXERCISE 10-1

Embracing Java Technology Forums as Valuable Information Resources

Many individuals are familiar with forums, especially technology forums. We are not going to make the assumption that you have this familiarity, since many entry-level programmers will be studying for this exam who may have never visited a technology forum. Forums provide an excellent place to ask questions and find answers about technologies, hard-to-solve problems, bugs, certifications, and the like. Newbies to technology forums need to understand that a certain etiquette is followed when posting messages. Always search the forum for your question before asking it, since it may have already been asked and even answered. Don't ask the same question across multiple forums of the same web site and make sure your questions are in the right place. Let's take a look at using a very popular Java forum, the JavaRanch.

1. Sit back and think of a Java-related question you may have, preferably Java technology–related since we are in the Java technologies chapter.
2. Head off to the JavaRanch—www.javaranch.com—to start the process of getting your answer.
3. Click the link to the Big Moose Saloon or the animated image of the one-eyed moose. This will bring you to the JavaRanch's message forum. You will see several forums, each specializing in a specific area such as the Java Micro Edition, JDBC, and the SCJA exam forums.
4. At the top of the web page, find the Search link and click it. You'll be brought to the search form. Remember that question you came up with? Derive from that question some keywords, place them into the search form, and click the Search button. You will be presented with a list of links showing the topics of related questions posted to the forum.
5. Browse through the list, clicking the topics you feel may have your answer.

6. If you can't find your answer, consider registering (for posting rights) in the forum (it's free). Then, post your question. Since you were to come up with a specific Java technology–based question, make sure you place it in the appropriate forum. Remember the etiquette: If it is a real basic question, you may wish to post it to the "Java in General (beginner)" forum.
7. Check back to the forum regularly to see if an answer to your question has been posted. Note that this and many other forums can send you e-mail notifications when your questions have been responded to.

CERTIFICATION OBJECTIVE

Working with the Java Remote Method Invocation API

Exam Objective 6.2 Describe at a high level the benefits and basic characteristics of RMI.

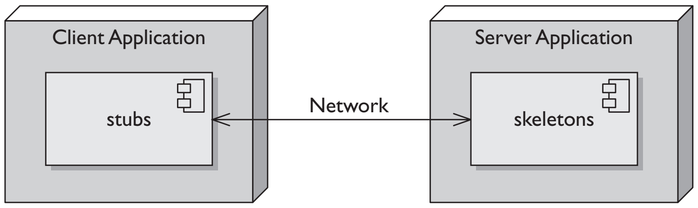
Wouldn't it be great if you could invoke a method from an object on another machine? Remote Method Invocation (RMI) technology does just that since it is Java's basic remote procedure call mechanism. This section discusses RMI's main goal, which is to allow for the sharing of Java objects between Java Virtual Machines.

The Java Remote Method Invocation API

The Java Remote Method Invocation API resides in the `java.rmi` package. This API provides the means in which Java applications perform distributed computing. In a classic client-server scenario, the RMI server maintains the remote objects, while the RMI client maintains the client objects that invoke the remote methods. This is done by implementing a remote interface through a client stub. You must understand that all of this work is not performed auto-*magic*-ally. The developer needs to solicit the creation of the stubs in order for the distributed communications to occur, as shown in Figure 10-4.

FIGURE 10-4

Remote method invocation



Stubs (client stubs) and skeletons (remote stubs) are created with the command-line utility '`rmic`'. Stubs and skeletons have their own key points:

- Client stubs
  - The '`rmic`'-created stub for the client side is called a "stub."
  - When using JDK 1.4 and earlier versions (prior to JDK 5.0), you are required to use the '`rmic`' utility to generate stub classes.
  - In JDK 5.0 and later, the RMI runtime generates the stub classes automatically (by constructing a proxy).
  - The client stub's marshal method calls the name and arguments and sends them to the skeleton on the server side. After they receive the results from the sender, they return the result back to the method call invoker.
- Server stubs (skeletons)
  - The '`rmic`'-created stub for the server side is called a "skeleton."
  - As of JDK 1.2 and later, skeletons are no longer needed.
  - The server stub's receive marshal method calls the name and arguments from the client stubs, performs the necessary operations, and returns the result back to the client stub.

The underlying protocol for the Java-only implementation of RMI is Java Remote Method Protocol (JRMP). RMI is not the only Java technology available for remote procedure calls. Java Remote Method Invocation over Internet Inter-Orb Protocol (RMI-IIOP) is used for Java to non-Java distributed computing solutions. Java-IDL is used by Common Object Request Broker Architecture (CORBA) developers to provide a CORBA to Java distributed computing solutions. You should not see RMI-IIOP or Java-IDL on the exam.

exam watch

The term "marshal" is the technique of passing method parameters and results for a remote procedure call. "Demarshal" is the reverse technique. The action form of these terms is marshalling and demarshalling.

A few benefits of RMI technologies include the heavy usage with Enterprise Java Beans; RMI is lightweight, allowing for servers to be initialized with minimum effort; and RMI hides the fine details of network communication and object serialization. Serialization is the process of flattening and restoring objects, thereby allowing the application to save the object's state as a series of bytes. This flattened object can therefore be saved to disk and/or sent across a network, where it is then reconstituted on the other side, as in the case of RMI.

exam watch

RMI's competing technology is simple sockets. Sockets are not on the exam per se—that is, you may see socket-related information referenced in answer choices.

INSIDE THE EXAM

Refining Your RMI Skills Through the SCJD Assignment

You only need to know RMI from a very high level for the SCJA exam. However, you will have the opportunity to master RMI through Sun's certification path if you decide to take the Sun Certified Java Developer (SCJD) assignment/exam. The various assignments that are part of the SCJD certification process require that a choice be made between implementation of an RMI or a "serialized objects over simple socket connection" solution. If you choose RMI for your given assignment, which many candidates do, you'll have the full opportunity to work with RMI. Be aware, though, that you must pass the SCJP exam before signing up for the SCJD.

When you finally get to the SCJD exam, you'll want to consider reviewing the following book: *SCJD Exam with J2SE 5, Second Edition* by Andrew Monkhouse and Terry Camerlengo (Apress, December 2005).

CERTIFICATION OBJECTIVE

Working with Database Technologies

Exam Objective 6.3 Describe at a high level the benefits and basic characteristics of JDBC, SQL, and RDBMS technologies.

Database Management Systems (DBMS) are designed to organize and maintain valuable information. Relational Database Management Systems (RDBMS) provide advanced flexible features that are directly related to their table-based design. The Structured Query Language (SQL) is an industry standard programming language used to work with these relational databases. SQL is supported by the Java Database Connectivity API. The following subsections will acquaint you with everything you'll need to know about database technologies as represented on the exam.

Relational Database Management Systems

A Relational Database Management System (RDBMS) is a type of database management system that organizes its data in the form of interrelated tables. Benefits of an RDMBS include the following:

- Provides a persistent data store
- Processes SQL queries
- Manages users
- Performs backups and restores

A list of popular RDBMS programs is shown in Table 10-4. All of these databases can be interfaced via the Structured Query Language.

TABLE 10-4 RDBMS Programs

RDBMS Program	Web Site Link
Java DB	http://developers.sun.com/javadb/
MySQL Enterprise Server	www.mysql.com/products/enterprise/server.html
Oracle Database	www.oracle.com/database/
PostgreSQL	www.postgresql.org/
Microsoft SQL Server	www.microsoft.com/sqlserver/2008/en/us/
IBM DB2	www-01.ibm.com/software/data/db2/9/

Structured Query Language

The Structured Query Language (SQL), pronounced "ess-kew-ell," is a software language designed for retrieval and management of information in RDBMS systems. More specifically, SQL provides the following features:

- Operations to support querying/retrieval of information from relational databases.
- Operations to support modification—for example, insertion, updating, and deletion of information in relational databases.
- Operations to support the management of relational databases.
- Operations to support execution plans. An execution plan is generated by the RDBMS to specify how it will execute a piece of application code.
- Operations to support stored procedures. A stored procedure is a piece of application code that is stored and executed within the database. You will get an execution plan as a result of compiling a stored procedure.

SQL is an ANSI and ISO standard and has several implementations with comprehensive extensions to the language.

The Java Database Connectivity API

The Java Database Connectivity (JDBC) API provides for a Java application to connect to an RDBMS server and take advantage of SQL. JDBC allows for the invocation of SQL commands and stored procedures, as well as the processing of such queries. In order to use JDBC, you'll need to import the necessary JDBC packages from the Java SE platform, as well as implement the service provider interface and make use of the RDMBS-specific JDBC driver.

on the job

Sun maintains a resource of over 200 JDBC drivers via their JDBC Data Access API, allowing you to get JDBC drivers from various vendors. You can access the API at http://developers.sun.com/product/jdbc/drivers.

exam watch

For the novice, JDBC and JNDI technologies may be easily confused since both are integration technologies that perform connection and access capabilities. Remember that the JDBC API is used to connect to and interface with databases, while the JNDI API is used to connect to and interface with naming and directory services. You will probably see these technologies in the same group of answers on the exam.

CERTIFICATION OBJECTIVE

Working with Additional Java Integration APIs

Exam Objective 6.4 Describe at a high level the benefits and basic characteristics of JNDI, messaging, and JMS technologies.

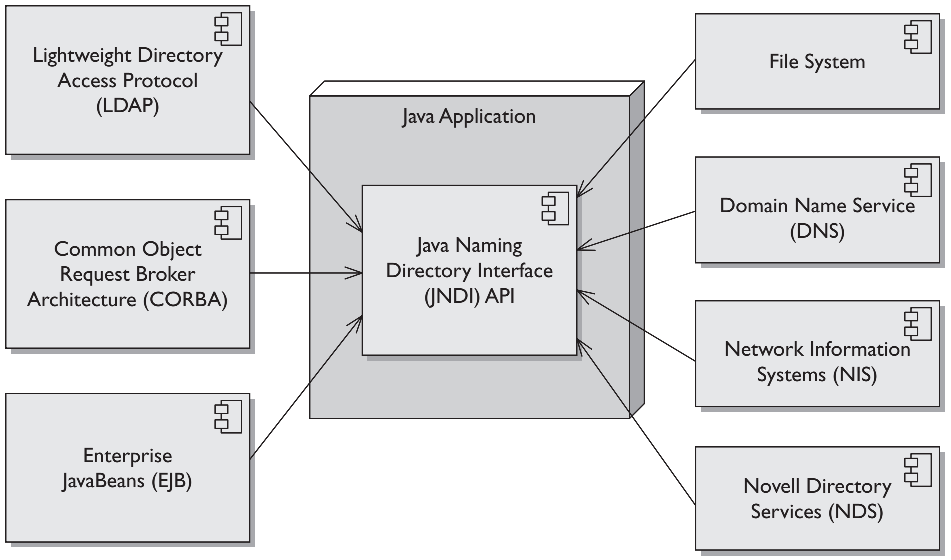
Several Java integration APIs exist. We already discussed RMI and JDBC. Let's now take a look at JNDI and JMS. The JNDI API allows for integration with naming and directory services. The JMS API allows for access to point-to-point and publish/subscribe messaging systems. JNDI and JMS objectives are grouped together under the presumed reason that JMS, as well as other APIs, uses JNDI to interface with necessary resources. These topics will be covered in the following subsections.

The Java Naming and Directory Interface API

The Java Naming and Directory Interface (JNDI) API provides general client-side querying features against directory and naming services by both attributes and a hierarchy of names. Examples of supported directory and naming services include the Lightweight Directory Access Protocol (LDAP), the Novell Directory Services (NDS), the Domain Name Service (DNS), the Network Information Systems (NIS), File Systems, EJB naming services, and Common Object Request Broker Architecture (CORBA), as shown in Figure 10-5. Naming and directory services are designed as database structures that are typically laid out as hierarchal directories, not relational table-based databases.

FIGURE 10-5

Integrating with naming and directory services



Lightweight Directory Access Protocol (LDAP)
Common Object Request Broker Architecture (CORBA)
Enterprise JavaBeans (EJB)
Java Application
Java Naming Directory Interface (JNDI) API
File System
Domain Name Service (DNS)
Network Information Systems (NIS)
Novell Directory Services (NDS)

JNDI methods include connection capabilities, as well as searching, retrieval, and storage of naming and directory service objects. JNDI operations support the association of attributes with objects to assist in the management and retrieval of information related to those objects. JNDI provides an event interface that allows the client application to be notified when directory information has been changed.

JNDI lays out its elements as a tree structure of directories and objects. When creating a connection, you need to establish an initial base context. Think of the context as the directory starting point in the tree. You may change the context whenever you like. However, when the context is set, you can only traverse the downward portion of the tree when performing querying operations.

on the job

Several LDAP browsers are freely available, such as JXplorer and the Softerra LDAP Browser. These utilities let the user remotely access LDAP servers. Their viewing and filtering capabilities are extremely beneficial in understanding the full layout of an LDAP server. These structural details are important to have before you start developing your client code. Many LDAP browser vendors have a full LDAP administrator version available that is often provided in a commercial fashion.

The Java Message Service API

The Java Message Service (JMS) API resides in the `javax.jms` package. This API is used to access the common features of enterprise messaging systems. The JMS API allows for the creation, sending, receiving, and reading of messages with Java EE application components. All of this distributed-computing communications of separate applications is done asynchronously; this is essentially the definition of a messaging system.

Several types of messaging systems can be utilized. JMS supports two common models: the publish/subscribe and point-to-point messaging models. Look for this on your exam.

The Publish/Subscribe Messaging Model

Publish/subscribe messaging is based on events. Consumers subscribe to events of interest by specifying a topic that is part of a set of messages. The producers of these messages will route these messages to the consumers who register for them. The consumers will then consume the events when they arrive.

The Point-to-Point Messaging Model

Point-to-point (PTP) messaging involves applications routing messages to consumers while using a shared queue. The consumer maintains this queue of awaiting messages, and the messaging application sends messages to that queue.

on the job

Apache ActiveMQ is an open-source message broker that fully supports JMS 1.1 as part of the J2EE 1.4 specification. Consider researching ActiveMQ for a more practical understanding of the Java Message Service API; http://activemq.apache.org/.

exam watch

As silly as it sounds, we can't stress enough that you must know the *exact* ***full names of the acronyms related to the exam. Reviewing the glossary of this book a few times will help familiarize you. Meaning, don't be surprised if you see a question on the exam similar to, "What does JSP stand for?" followed by four very similar answers.***

CERTIFICATION SUMMARY

This chapter discussed the differences between the Java platforms, as well as their practical applicability. All three platforms were covered in detail: Java SE, Java ME, and Java EE. SCJA-related Java integration technologies, including the Java RMI API, JDBC API, JNDI API, and the JMS API, were also explored. Let's take a minute to summarize the high-level points of what we've learned.

The Java Platform, Standard Edition comprises the JDK, JRE, and Java SE APIs. The Java Development Kit includes all of the tools necessary to develop, debug, and test Java applications. The JDK includes the JRE. The Java Runtime Environment contains the deployment technologies and Java Virtual Machines necessary to execute bytecode. The JRE includes the Java SE API, which in turn contains software packages encompassing related classes and interfaces.

The Java Platform, Micro Edition is a configurations-based architecture designed for embedded devices such as high-end PDAs and mobile phones.

The Java Platform, Enterprise Edition is used to build flexible, scalable, and secure enterprise systems. In Java enterprise systems, there is a clear separation between business and presentation logic.

The Remote Method Invocation API allows for distributed computing through remote procedure calls. Client-side stubs and server-side stubs (skeletons) provide the stubs necessary for marshaling system references and values.

The Java Database Connectivity API provides database support, allowing you to execute SQL queries and process the results of the queries.

The Java Naming and Directory Interface API lets you access naming and directory services. JNDI methods include the searching, retrieval, and storage capabilities of naming and directory service objects.

The Java Message Service API is used to access the features of messaging systems, specifically systems that use the publish/subscribe and point-to-point messaging models.

You will not need to retain all of the finer details discussed in this chapter in regards to the core topics of platforms and integration technologies. However, the more information you can retain, the easier the high-level questions will be for you come exam time.

✓ TWO-MINUTE DRILL

Understanding Java Platforms

- ❑ The Java SE platform is used to build client and/or client-server systems.
- ❑ The Java SE platform contains the Java SE API, which includes JDBC, JNDI, RMI, AWT, I/O, Swing, networking, language, and utilities APIs.
- ❑ The Java SE platform contains the Java Runtime Environment, which houses the Java SE API, JVMs, and deployment APIs.
- ❑ The Java SE platform contains the Java Development Kit, which includes all necessary compilation and debugging tools.
- ❑ The Java SE platform is considered to contain the underlying operating system—for example, Solaris, Linux, or Windows.
- ❑ The Java ME platform is used to build applications for embedded and mobile devices.
- ❑ Java ME includes the CDC and CLDC configurations.
- ❑ Java ME includes the CDC Personal Profile (PP), Personal Basis Profile (PBP), and Foundation Profile (FP).
- ❑ Java ME includes the CLDC Mobile Information Device Profile (MIDP) and the Information Module Profile (IMP).
- ❑ The Java EE platform is used to build enterprise systems.
- ❑ The Java EE platform includes an MVC design pattern that cleanly separates the controlling tier, presentation tier, and business tier, providing a scalable, flexible, and secure development and deployment environment.
- ❑ Java EE includes the EJB, JMS, Web Services, JAX-RPC, Servlets, JSP, and JavaMail APIs.

Working with the Java Remote Method Invocation API

- ❑ The Remote Method Invocation API provides a means for Java applications to perform distributed computing.
- ❑ RMI technologies are heavily used with Enterprise JavaBeans.
- ❑ RMI allows for servers to be initialized with minimum effort.

- ❑ Java Remote Method Protocol (JRMP) is the Java-only implementation of RMI.
- ❑ The command-line tool 'rmic' is used to create RMI stubs and skeletons. This tool is not required for J2SE 5.0 and later.

Working with Database Technologies

- ❑ An RDBMS is a type of database management system that organizes its data in the form of interrelated tables.
- ❑ SQL is a software language designed for retrieval and management of information in RDBMS systems.
- ❑ A group of SQL statements is considered a stored procedure. A stored procedure is compiled into a single execution plan and is executed within the database.
- ❑ The JDBC API establishes connections with relational databases.
- ❑ The JDBC API sends SQL queries to relational databases.
- ❑ The JDBC API receives and processes the results of SQL queries from relational databases.

Working with Additional Java Integration APIs

- ❑ JNDI allows for interaction with naming and directory services.
- ❑ JNDI methods include connection capabilities of naming and directory service objects.
- ❑ JNDI methods include searching, retrieval, and storage capabilities of naming and directory service objects.
- ❑ JNDI operations support the association of attributes with objects to assist in the management and retrieval of information related to those objects.
- ❑ JNDI provides an event interface that allows the client application to be notified when directory information has been changed.
- ❑ JNDI essentially lays things out in a tree structure of directories and objects.
- ❑ JMS messages include asynchronous requests, reports, and events.
- ❑ JMS supports the publish/subscribe messaging system model.
- ❑ JMS supports the point-to-point messaging system model.

SELF TEST

Understanding Java Platforms

1. Which Java platform contains the specifications for servlets, JavaServer Pages, and JavaServer Faces?

A. Java ME

B. Java SE

C. Java EE

D. Java EA

2. Which two Java ME profile statements are correct?

A. The Personal Profile (PP) is a CDC profile.

B. The Personal Basis Profile (PBP) is a CLDC profile.

C. The Foundation Profile (FP) is a CLDC profile.

D. The Mobile Information Device Profile (MIDP) is a CLDC profile.

3. Which Java platform contains the JNDI and JDBC integration APIs?

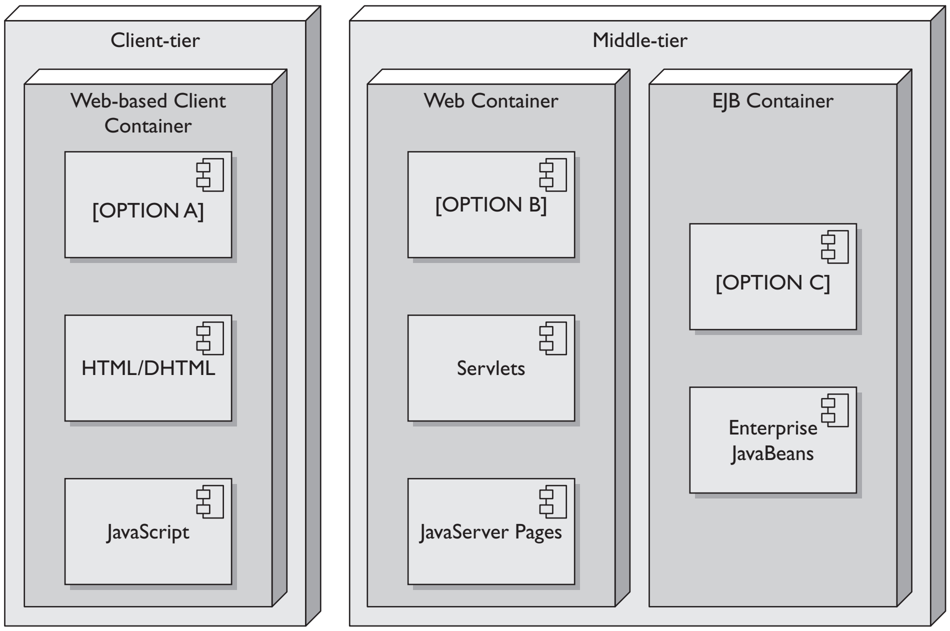
A. Java ME

B. Java SE

C. Java EE

D. Java SE and Java EE

4. Consider the following illustration. Cascading Style Sheets (CSS) is a stylesheet language used to support the rendering of presentation logic. Which container does CSS belong in?



A. Option A—the web-based client container

B. Option B—the web container

C. Option C—the EJB container

D. None of the above.

Working with the Java Remote Method Invocation API

5. Which is not an advantage of RMI?

A. Creating an RMI solution would involve creating a custom protocol.

B. RMI technologies are heavily used with Enterprise JavaBeans.

C. RMI allows for servers to be initialized with minimum effort.

D. RMI hides the fine details of network communication and object serialization.

6. What is the underlying protocol for the Java-only implementation of RMI?
   - A. Jini
   - B. Java Remote Method Protocol (JRMP)
   - C. RMI-IIOP
   - D. CORBA
7. What is the command-line tool used to create RMI stubs?
   - A. `rmi_create`
   - B. `stub_create`
   - C. `rmi -c`
   - D. `rmic`

Working with Database Technologies

8. What is a group of SQL statements called that is compiled into a single execution plan?
   - A. RMI
   - B. JNDI
   - C. A stored procedure
   - D. JDBC
9. What are the three main capabilities of the JDBC API?
   - A. Sending of SQL queries
   - B. Establishment of a connection with a database
   - C. Processing the results from SQL queries
   - D. Establishment of a connection with a naming server
10. Do you need to have a JDBC technology–enabled driver for a given database, in addition to using the JDBC API?
    - A. Yes
    - B. No

Working with Additional Java Integration APIs

11. JNDI allows for interaction with which of the following naming and directory services?

A. LDAP, NDS, DNS, NIS(YP)

B. LADP, NDS, DNS, NIS(YP)

C. LDAP, SDN, DNS, NIS(YP)

D. LDAP, NDS, DNS, NIIS(XP)

12. JMS messages consumed by enterprise applications include which asynchronous items?

A. requests

B. listeners

C. reports

D. events

13. Which model does JMS support?

A. Publish/subscribe

B. Transmit/receive

C. Dictate/annotate

D. Publish/render/subscribe

SELF TEST ANSWERS

Understanding Java Platforms

1. Which Java platform contains the specifications for servlets, JavaServer Pages, and JavaServer Faces?

A. Java ME

B. Java SE

C. Java EE

D. Java EA

Answer:

☑ **C.** The Java EE platform contains the specifications related to dynamic web content solutions, including servlets, JavaServer Pages, and JavaServer Faces.

☒ **A, B,** and **D** are incorrect. **A** is incorrect because Java ME is the Java Micro Edition used for embedded solutions. **B** is incorrect because Java SE is the Java 2 Standard Edition used for basic application development. **D** is incorrect because Java EA is fictitious.

2. Which two Java ME profile statements are correct?

A. The Personal Profile (PP) is a CDC profile.

B. The Personal Basis Profile (PBP) is a CLDC profile.

C. The Foundation Profile (FP) is a CLDC profile.

D. The Mobile Information Device Profile (MIDP) is a CLDC profile.

Answer:

☑ **A** and **D. A** is correct because the Personal Profile (PP) is a CDC profile. **D** is correct because the Mobile Information Device Profile (MIDP) is a CLDC profile.

☒ **B** and **C** are incorrect because the Personal Basis Profile (PBP) and Foundation Profile (FP) are actually CDC profiles.

3. Which Java platform contains the JNDI and JDBC integration APIs?

A. Java ME

B. Java SE

C. Java EE

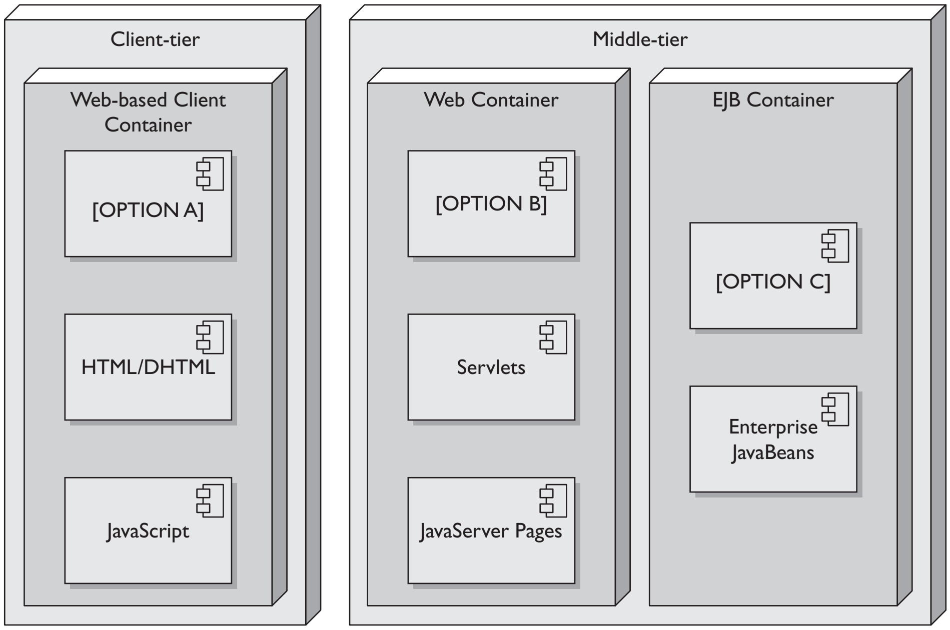
D. Java SE and Java EE

Answer:

☑ **B.** The Java Platform, Standard Edition houses the implementations of the APIs for JNDI and JDBC.

☒ **A, C,** and **D** are incorrect because Java ME and Java EE do not house the APIs.

4. Consider the following illustration. Cascading Style Sheets (CSS) is a stylesheet language used to support the rendering of presentation logic. Which container does CSS belong in?



A. Option A—the web-based client container

B. Option B—the web container

C. Option C—the EJB container

D. None of the above.

Answer:

☑ **A.** Cascading Style Sheets (CSS) are used within web-based client containers such as web browsers. CSS is commonly used to style web pages written in HTML. Note that CSS is not on the exam, this was a simply a critical-thinking question in relationship to the Java EE platform.

☒ **B, C,** and **D** are incorrect. JSPs and servlets can produce inline CSS code in relationship to HTML content, but the code is ultimately sent to and used by web browsers. Therefore, **B** is incorrect because CSS does not directly relate to the web container. **C** is incorrect because CSS has no relationship to the EJB container.

Working with the Java Remote Method Invocation API

5. Which is not an advantage of RMI?

A. Creating an RMI solution would involve creating a custom protocol.

B. RMI technologies are heavily used with Enterprise JavaBeans.

C. RMI allows for servers to be initialized with minimum effort.

D. RMI hides the fine details of network communication and object serialization.

Answer:

☑ **A.** Creating an RMI solution does not involve creating a custom protocol.

☒ **B, C,** and **D** are incorrect. These answers represent advantages of RMI. **B** is incorrect because RMI is utilized with EJBs. **C** is incorrect because servers that initialize using RMI do so with minimum effort. **D** is incorrect because network communications and object serialization is handled behind the scenes with RMI.

6. What is the underlying protocol for the Java-only implementation of RMI?

A. Jini

B. Java Remote Method Protocol (JRMP)

C. RMI-IIOP

D. CORBA

Answer:

☑ **B.** JRMP is the underlying protocol for Java-based implementations of RMI.

☒ **A, C,** and **D** are incorrect. These answers are not underlying Java-based implementations of RMI. Jini is a dynamic networking architecture. Java Remote Method Invocation over Internet Inter-Orb Protocol (RMI-IIOP) provides Common Object Request Broker Architecture (CORBA) distributed computing support.

7. What is the command-line tool used to create RMI stubs?

A. `rmi_create`

B. `stub_create`

C. `rmi -c`

D. `rmic`

Answer:

☑ **D.** The `rmic` command is used to create RMI client-side stubs.

☒ **A, B,** and **C** are incorrect. These answers do not represent valid tools because the commands `rmi_create`, `stub_create`, and `rmi` do not exist.

Working with Database Technologies

8. What is a group of SQL statements called that is compiled into a single execution plan?

A. RMI

B. JNDI

C. A stored procedure

D. JDBC

Answer:

☑ **C** is the correct answer because a set of SQL statements is known as a stored procedure.

☒ **A, B,** and **D** are incorrect answers. RMI, JNDI, and JDBC are all integration technologies and are not directly related to a single execution plan.

9. What are the three main capabilities of the JDBC API?

A. Sending of SQL queries

B. Establishment of a connection with a database

C. Processing the results from SQL queries

D. Establishment of a connection with a naming server

Answer:

☑ **A, B,** and **C.** The JDBC API allows for the establishment of database connections, the sending of SQL queries, and the processing of results from those queries.

☒ **D** is incorrect. The best way to connect to a naming server is through the JNDI API.

10. Do you need to have a JDBC technology–enabled driver for a given database, in addition to using the JDBC API?

A. Yes

B. No

Answer:

☑ **A.** Yes, the JDBC API is designed to use different drivers, but you must obtain the driver you will need to use.

Working with Additional Java Integration APIs

11. JNDI allows for interaction with which of the following naming and directory services?

A. LDAP, NDS, DNS, NIS(YP)

B. LADP, NDS, DNS, NIS(YP)

C. LDAP, SDN, DNS, NIS(YP)

D. LDAP, NDS, DNS, NIIS(XP)

Answer:

☑ **A.** The list represents existing naming and directory services supported by JNDI. The services listed are more formally known as the Lightweight Directory Access Protocol (LDAP), the Novell Directory Services (NDS), the Domain Name Service (DNS), and the Network Information Systems (NIS). In the case of NIS, YP represents the original name of Yellow Pages that had to be changed due to trademark issues.

☒ **B, C,** and **D** are incorrect. LADP, SDN, and NIIS (XP) are not naming and directory interfaces that actually exist.

12. JMS messages consumed by enterprise applications include which asynchronous items?

A. requests

B. listeners

C. reports

D. events

Answer:

☑ **A, C,** and **D.** Asynchronous requests, reports, and events are consumed by enterprise applications.

☒ **B** is incorrect. "Listeners" makes no logical sense to this question.

13. Which model does JMS support?

A. Publish/subscribe

B. Transmit/receive

C. Dictate/annotate

D. Publish/render/subscribe

Answer:

☑ **A.** JMS supports the publish/subscribe model also known as an asynchronous messaging paradigm.

☒ **B, C,** and **D** are incorrect.