CHAPTER
1

Installing WebLogic Server and Using the Management Tools

T he introduction to this book provided a quick outline of the Java Enterprise Edition (Java EE) and the nature of web applications for which you use WebLogic Server. Since the primary goal of this book is for you to understand how to administer WebLogic Server, you will begin by learning key administration topics such as installing and upgrading the Oracle WebLogic Server and becoming familiar with the administration tools you use day in and day out to manage WebLogic Server. There are three major administrative tools that are going to be your day-to-day companions when managing the Oracle WebLogic Server: the Administration Console, the Node Manager utility, and the WebLogic Scripting Tool (WLST). This chapter briefly introduces these tools, and you'll learn how to use all three of these tools, as well as other WebLogic Server management commands, in later chapters. In this and the next chapter, I make extensive use of the sample Oracle WebLogic Server applications that you can install to learn various administrative and deployment-related concepts. This chapter introduces the WebLogic Server sample domains that host the sample applications. This and other chapters use the sample domains to explain various WebLogic Server management concepts. Before we start reviewing the installation, upgrading, and management of Oracle WebLogic Server, however, let's review the Oracle WebLogic Server product set as well as key terminology and important architectural concepts that illustrate how Oracle WebLogic Server functions.

Oracle WebLogic Server: An Overview

Before you learn how to install, upgrade, and manage Oracle WebLogic Server, let's do a quick review of the various Oracle WebLogic Server products. In addition, a brief summary of key terminology will help you understand the components of a WebLogic Server *domain*, a collection of WebLogic Server instances and related resources and services that are managed together as a single unit.

Oracle WebLogic Server Product Set

Oracle WebLogic Server 11*g* is a component of Oracle Fusion Middleware 11*g*, which consists of several Oracle products that span business intelligence, collaboration tools, content management, and integration services. The underlying application server supporting these middleware applications is Oracle WebLogic Server 11*g*. Products such as Oracle SOA Suite and Oracle Fusion applications rely on Oracle WebLogic Server 11*g* to run their code.

Oracle offers three distinct products as part of the Oracle WebLogic Server 11*g* application family:

- Oracle WebLogic Server Standard Edition (SE)
- Oracle WebLogic Server Enterprise Edition (EE)
- Oracle WebLogic Suite

Oracle WebLogic Server Standard Edition

The WebLogic Server Standard Edition (SE) is a full-featured server, but is mainly intended for developers to develop enterprise applications quickly. WebLogic Server SE implements all the Java EE standards and offers management capabilities through the Administration Console.

Oracle WebLogic Server Enterprise Edition

Oracle WebLogic Server EE is designed for mission-critical applications that require high availability and advanced diagnostic capabilities. The EE version contains all the features of the SE version, of course, but in addition supports clustering of servers for high availability and the ability to manage multiple domains, plus various diagnostic tools.

Oracle WebLogic Suite

Oracle WebLogic Suite offers support for dynamic scale-out applications with features such as in-memory data grid technology and comprehensive management capabilities. It consists of the following components:

- Oracle WebLogic Server EE
- Oracle Coherence (provides in-memory caching)
- Oracle Top Link (provides persistence functionality)
- Oracle JRockit (for low-latency, high-throughput transactions)

This book deals exclusively with the Oracle WebLogic Server EE 11*g* (10.3.4.0) product. (I refer to it simply as WebLogic Server in the rest of the book.) You manage WebLogic Server essentially the same way regardless of the operating system it is running on. This book uses examples run on a Windows installation of WebLogic Server; however, where necessary or relevant, certain tasks or commands are also shown for UNIX/Linux-based systems.

NOTE
Oracle WebLogic Server uses a configured pool of JDBC connections to interact with databases. You can use any RDBMS that supports a JDBC 2.0–compliant driver. This includes Oracle, IBM DB2, Microsoft SQL Server, MySQL, and other databases.. The WebLogic Server installation includes an embedded database called Apache Derby. (Previously, Oracle shipped a different database by the name of PointBase.)

Terminology

Before we delve into the administration of WebLogic Server, I want to make sure you clearly understand the key terminology you're going to encounter throughout the book. Some of the WebLogic Server terms and definitions are obvious, but others aren't, such as the concept of a *machine*, for example.

WebLogic Server Instance

A WebLogic Server instance is a Java Virtual Machine (JVM) process that runs the Java code. The instance is the actively working component, receiving client requests and sending them on to the appropriate components, and sending the processed requests back to the originating clients. The server instance manages the resources necessary for applications, such as the JTA and JDBC services, to function. In each domain, one instance serves as the Administration Server, which is your primary means of managing the domain. The rest of the WebLogic Server instances are called Managed Servers. If you have a domain with just one WebLogic Server instance, as is the case in a development environment, the single server instance functions as both the

Administration Server and the Managed Server. Note that the terms WebLogic Server and WebLogic instance are often used interchangeably.

WebLogic Server Domain

A domain is a set of WebLogic Server instances that you manage with the Administration Server, which itself is nothing but another WebLogic Server instance, albeit a special one. Any configuration changes you make to a domain will apply to all members of that domain. Domains offer you ease of administration—for example, you can apply configuration changes on a domain-wide basis that apply to all the management servers that belong to that domain. Every domain has exactly one Administration Server, which is used to configure and manage that domain. In addition to the WebLogic Server instances, a domain also includes the application components that you deploy, as well as all the services required by the server instances of that domain. The Administration Server is usually referred to as the Admin Server for short.

The concept of a domain offers you the administrative ease you need to manage your WebLogic environment. A domain encompasses the Admin Server, Managed Servers (including those configured into WebLogic clusters), machines (servers), and all the services necessary to run your applications. The fact that a domain includes all the configuration data for the servers, deployments, and physical network makes it very easy to configure and manage complex, geographically dispersed WebLogic Server deployments. A domain lets you easily deploy applications across multiple WebLogic Server instances located on heterogeneous servers and multiple networks, with varying physical and network descriptions. Administering a domain makes it easy for you to configure high availability with the help of multiple WebLogic Server instances and administer various services spread across heterogeneous host servers.

The first step in using Oracle WebLogic Server to deploy applications is to create a domain. A domain can consist of just a single Admin Server, with no Managed Servers at all, as is common in a development environment. A production cluster ranges over several physical machines to provide high availability and failover protection, but you can also configure a cluster on a single server for testing and development purposes. WebLogic Server stores the configuration information for a domain in the *config.xml* file, which is stored on the machine where the Admin Server runs and serves as the domain's configuration file. The domain also contains security settings, log files, and startup scripts for the Admin and Managed Servers that belong to that domain. The WebLogic Configuration Wizard and the WebLogic Domain Wizard offer you extremely easy ways to create domains, as well as the servers and clusters that belong to that domain.

> **NOTE**
> *Each Managed Server contains a local copy of its domain configuration. Upon startup, it synchronizes its configuration with the Admin Server. Similarly, when you make domain configuration changes on the Admin Server, those changes are propagated to the Managed Server's configuration.*

Administration Server

A server is an instance of WebLogic Server that runs in its own JVM, and the Admin Server is a special instance of WebLogic Server designed for managing the domain rather than running applications. There is a one-to-one relationship between domains and the Admin Server—an Admin Server belonging to Domain A can't manage Domain B.

You can deploy applications on the Admin Server, but unless you're operating in a purely developmental environment, use the Admin Server strictly for performing management tasks, not for deploying any applications. Although you can deploy applications on the Admin Server in a development environment, it's a best practice not to do so in a production environment. For one thing, you don't want application work to compete with administrative work in a production environment. You also want to firewall the Admin Server separately so external clients can't access it.

The Admin Server is critical to the functioning of a WebLogic Server domain, since it manages the domain configuration, including the servers that are part of the domain, as well as all the applications and services you deploy to the various servers. Apart from this management of the domain configuration information, the Admin Server has all of the functionality of a Managed Server; in fact, an Admin Server runs the same code and is managed internally the same way as a Managed Server. The Admin Server hosts the Administration Console, which is a web application front end used for configuring, monitoring, and managing a domain. You can access the Administration Console with any supported browser that can access the Admin Server. All WebLogic system administration tools and APIs interact with the Admin Server. If you install the optional Node Manager service, the Admin Server communicates with the Node Manager service on each machine to talk to the Managed Servers running on that machine.

Managed Server

Managed servers are the workhorses of WebLogic Server. Any additional servers you create after the creation of the default Admin Server are Managed Servers. The Managed Server contacts the Admin Server, only when you start it up, to get the configuration and deployment settings. For this reason, you should always start up the Admin Server before you start a Managed Server. Once a Managed Server starts running, it operates completely independent of the Admin Server.

Although you can deploy an application to the Admin Server itself, the recommended approach is to deploy applications to the Managed Servers. In a production environment, it's common to run multiple Managed Servers as part of a cluster. A Managed Server hosts your Java EE applications, as well as all related resources and services such as Java Database Connectivity (JDBC) connection pools and data sources, Java Transaction API (JTA) transaction services, and Java Messaging Service (JMS) connection factories that are necessary to support application deployments. As mentioned earlier, upon its startup, a Managed Server will contact the Admin Server to retrieve any configuration changes since the Managed Server was last shut down. However, a Managed Server can continue to run, and it's even possible to start it up, in the absence of an Admin Server. Chapter 2 shows how you can start a Managed Server without a running Admin Server, in a special mode of operation called the Managed Server Independence (MSI) mode. The MSI mode is enabled by default, and it allows the Managed Server to start up using its locally cached configuration.

WebLogic Server Cluster

A WebLogic Server cluster is a group of WebLogic Server instances consisting of multiple Managed Servers that run simultaneously. The multiple Managed Servers work together to provide replication services for one another, and the Admin Server is not generally a part of any cluster. Most production deployments use clusters to increase reliability and scalability through load distribution and high availability. To achieve the high availability capability, you deploy resources and services in a homogeneous fashion on each of the Managed Servers that are part of a cluster. Clusters host applications that respond to HTTP requests that are routed to the cluster through a hardware load balancer. You can also set up load balancing on a WebLogic Server

instance or a third-party Web Server with the help of plug-ins supplied by WebLogic Server. The load balancer handles the HTTP requests after the requests pass through a firewall. Cluster members pass replicated copies of objects such as HTTP sessions among themselves to provide the failover capability for the cluster.

NOTE
The simplest domain will consist of just one server—the Admin Server. In a development environment, you can sometimes get by with such a simple setup and host all applications directly on the Admin Server without using a Managed Server or a cluster comprising several Managed Servers.

A WebLogic Server domain can consist of multiple Managed Servers that aren't part of a cluster, or that are part of a cluster or it can consist of multiple clusters—just remember that even if you have multiple Managed Servers in a domain, you can avail yourself of WebLogic Server's high availability and load-balancing features only by deploying a cluster of servers. High availability lets you continue serving clients even when you experience a failure such as a machine or WebLogic Server failure. WebLogic Server offers you many powerful features such as replication, failover, and the ability to migrate services so you have high availability for your system. Clusters provide load-balancing capabilities by letting you spread requests across the cluster members. Clusters also offer scalability by letting you easily add on additional servers to the cluster to accommodate increased demand for WebLogic Server services. The important thing to understand here is that the cluster automatically provides these capabilities without the user experiencing any disruption.

Each domain may consist of multiple Java EE resources such as JDBC connection pools and JMS servers, which the domain makes available to all the applications it hosts. Note that domain resources, such as a JDBC connection pool, are not shared across domains—each WebLogic domain must create its own set of resources. This applies when dealing with clusters as well, which are treated as a domain resource. A cluster's Managed Servers thus can't overlap domains and belong to more than one domain at the same time. Therefore, whenever you perform a failover within a cluster, you can fail over from one Managed Server to another Managed Server within the same domain but not to a Managed Server that belongs to a different domain.

How does one design a domain? Other than the simple requirement that you must install the same version of WebLogic Server for all the Managed Server instances in a cluster, it's easy to design a cluster. Although a WebLogic Server cluster can run entirely on a single machine, to take advantage of the high availability features, Managed Servers are typically installed on two or more physical machines. To increase a cluster's capacity, you can either add more Managed Server instances to the existing cluster architecture, or you can add more physical machines to the cluster, with the additional machines hosting new Managed Server instances, of course. Managed Servers can serve as backups for services such as JTA and JMS that another Managed Server in the same cluster hosts.

There's really no hard and fast rule for organizing your domains; one way to organize domains is to create separate domains to handle different types of work. For example, you can have one domain dedicated to online shopping and another to support your internal e-business operations. In general, you design domains based on your service needs, security requirements, and management considerations You can also create separate domains for physically separate business locations.

It's sometimes easy to get confused as to how a cluster relates to a domain. Just remember that a domain is simply a set of WebLogic Server instances, some of which may be clustered and some not, and that a domain can contain multiple clusters.

Machine

A machine in the WebLogic Server context is the logical representation of the computer that hosts one or more WebLogic Server instances (servers). The Admin Server uses the machine definitions that you create to start remote servers through the Node Managers that run on those servers. A machine could be a physical or virtual server that hosts an Admin or Managed Server that belongs to a domain. You'll see later in the book that you must define a machine first if you want the Admin Server to use the Node Manager service to monitor, start, and stop the Managed Servers running on a server. In a sense, a machine in a WebLogic Server environment is more or less equivalent to an instance of a Node Manager and this is essentially the concept that a machine represents. WebLogic clusters make use of the machines you define in order to decide the optimal way to replicate session data on a different server that is part of a cluster.

Network Channels

Network channels are an optional feature that allows you to separate different classes of network traffic. You can make use of separate network channels to separate server and client traffic and direct it to different listening ports or addresses. If you need to allow both secure and nonsecure traffic on the same server, you can create multiple channels to support the diverse traffic with different security protocols. You can also use network channels to manage quality of service by using weighted, value-based priorities for different channels. This enables you to assign high weighted values to faster channels that use faster network interface cards and dedicate them to the types of traffic that require faster throughput, for example. Network channels control all communication-related aspects such as listen addresses, protocols, and port numbers throughout the domain.

Node Manager

The Node Manager is an optional process that runs on a machine and manages the availability of all servers that run on that machine. Node Managers help you remotely start, stop, suspend, and restart Managed Servers. The Node Manager works with the Admin Server using a secure channel and lets you manage the availability, as well as monitor the health, of all Managed Servers in a single domain. The Managed Servers that the Node Manager controls can be independent servers or they can be members of a cluster. The Node Manager monitors remote Managed Servers and is capable of automatically restarting them when they fail. It also kills Managed Servers that exhibit unstable behavior. It is recommended that you install a Node Manager service on each machine that hosts a Managed Server. Managing the servers with Node Managers is actually a key requirement for configuring automatic server migration in a cluster following a server failure, as explained in Chapter 7. In Chapter 2, I explain how you can use Node Manager and the WebLogic Scripting Tool (WLST) together to perform various administrative tasks.

Virtual Hosts

Virtual hosts rely on the Domain Name System (DNS) to map hostnames to the IP address of a single server or a cluster of servers. By doing so, multiple domain names can be hosted on your server wherein different web applications can be assigned to different virtual hosts, effectively sharing all resources and being differentiated only by their hostnames.

Work Managers

Work Managers help you manage the WebLogic Server instance workload, specifically by letting you prioritize work execution, which you do by defining request classes and constraints. You can configure Work Managers at the domain level (using global Work Managers) or at the application or module level.

Services

Following are some of the main services used in a WebLogic environment:

■ JDBC (Java Database Connectivity) enables Java programs to handle database connections established through connection pools.

■ JMS (Java Messaging Service) is a standard API that enables applications to communicate through enterprise messaging systems.

■ JTA (Java Transaction API) specifies standard Java interfaces between transaction managers and the parties in a distributed transaction system.

TIP
You can create Jolt Connection Pools to enable your applications to connect to Oracle Tuxedo domains. Jolt clients will then manage requests from your applications to the Oracle Tuxedo Services.

Deployment

When you want to make a Java EE application or a stand-alone application module available to users, you must first install those applications and modules in a WebLogic domain. Once you install the applications and modules, you must start those so the applications can begin processing user requests. Deployment is the process of installing the applications or modules and starting them so they are available to clients. Developers package applications for delivery to administrators, who then deploy the applications to WebLogic Server instances or clusters. Chapter 7 shows the various ways in which you can deploy applications to development and production environments, by using the Administration Console, WLST and the *weblogic.Deployer* utility.

NOTE
*Oracle WebLogic Server 11*g *fully supports Oracle Real Application Clusters (RAC). Chapter 4 shows you how to configure data sources to connect to Oracle RAC database services.*

Security Realm

You use security realms to protect the WebLogic Server resources. A security realm is simply a logical container for your user, group, roles, security policies, and security providers. It's the security realm that authenticates users and determines which resources they can access. WebLogic Server uses a default security realm by the name *myrealm*. In the default security realm, the Admin Server stores the domain security data in an LDAP server, but you can also choose an RDBMS store for this instead. The Managed Servers replicate this LDAP server, and when the Admin Server fails it can use their copy of the LDAP server for providing security services to the deployed applications.

When you create a domain, the username/password credentials you provide are used by the Configuration Wizard to seed the security realm myrealm. The username you provide will be the initial administrative user in myrealm. When you start up the server, it uses the default security realm to authenticate usernames. You can configure the server to use other security realms, but you must always specify one of them as the default security realm.

If these simple definitions of the key WebLogic Server terminology don't satisfy your curiosity, not to worry—subsequent chapters discuss all these entitities in great detail!

Important WebLogic Server Concepts

In order to fully comprehend how WebLogic Server works and to get the best performance out of it, it's important to understand several concepts. The most important concepts are discussed in the following section.

Execute Threads and Queues

This section briefly describes the internal architecture of Oracle WebLogic Server in the sense of how the server performs its work of satisfying user requests. When a client sends a request to the WebLogic Server, the actual work to satisfy that request is performed by a Java thread called an execute thread. A user can submit work to WebLogic Server using an HTTP-based request to the Servlet engine or a request for Remote Method Invocation (RMI) access to objects such as Enterprise JavaBeans (EJBs). When a server process starts up, it binds itself to a port and assigns a *listen thread* to the port to listen for incoming requests. Once the request makes a connection, the server passes the control of that connection to the *socket muxer*. The socket muxer reads requests off the socket and places the work requests into the self-tuning *execute queue* as they arrive. An idle execute thread will pick up a request from the execute queue and may in turn hand off the job of responding to those requests to special threads. The execute thread executes the requests and returns the responses.

Oracle WebLogic Server uses socket muxers, which are software modules, to read incoming requests on the server. Muxers read messages from the network, bundle them into a package of work, and queue them to the Work Manager, which then finds a thread on which to execute the work and makes sure the response gets back to the same socket from which the request came. There are two types of muxers—a Java muxer and a native muxer. A Java muxer uses pure Java to read data from the sockets, whereas the native muxers use platform-specific native binaries. By default, Oracle WebLogic Server uses the native muxer that is, the *Enable Native IOP* parameter for the server is set to the value *SELECTED*. Note that with a native muxer, the server creates a fixed number of threads to read incoming requests, whereas with a Java muxer you can configure the number of threads in the Administration Console by modifying the *Percent Socket Readers* parameter. The native muxer allocates a certain percentage of the server threads to act as socket reader threads, which perform the pooling function, while the rest of the server threads are busy processing client requests. In general, you need to be careful about changing the number of socket reader threads. In many cases, the best optimization is to set it to 1.

You can tell if you're using a native muxer or a Java muxer by looking at the messages that involve the execute thread. If you're using the native muxer, the server error messages will refer to it as *weblogic.socket.EPollSocketMuxer*, whereas if you're using the Java muxer you'll see *weblogic.socket.SocketMuxer* instead. Note that the *EPollSockerMuxer* is associated only with a JRockit JVM operating on a Linux server. You'll see the word *poll* in the case of a native muxer

because it uses a polling mechanism to query a socket for data. Native muxers are seen as providing superior performance, especially when scaling to large user bases, because they implement a nonblocking thread model. When administering WebLogic Server instances, you'll frequently encounter the so-called "stuck thread" situation, which occurs when a thread isn't returned to the thread pool within the time you set for it (the default is 10 minutes). Resolution of stuck threads is a key component of WebLogic Server troubleshooting, and you'll notice that Oracle Support often asks for several server thread dumps when you open a service request. Chapter 6 shows you how to take a thread dump and analyze many performance issues on your own by identifying the resource contention leading to the stuck thread situation.

Implementing the JMX API and MBeans

WebLogic Server implements the system administration infrastructure with Sun's Java Management Extensions (JMX). Implementing the JMX API involves using Java MBeans (managed beans) to model system administration tasks. If you understand MBeans and the JMX API, you can use them to create your own custom management tools. However, all administrative tools, such as the Administration Console, use the same MBeans and JMX API, so you don't have to reinvent the wheel by creating custom management tools. While a WebLogic Server administrator doesn't need to know how to program the JMX API, it helps to understand the different types of MBeans and how the JMX API interacts with them.

WebLogic Server uses two basic types of MBeans—*configuration MBeans* and *runtime MBeans*—to configure, monitor, and manage the server and its resources.

■ *Configuration MBeans* contain the configuration information for servers and resources that is stored in the domain's configuration files such as the *config.xml* file and other XML files. These are persistent MBeans, and the domain's configuration file, *config.xml*, stores the attribute values for these MBeans. Whenever you change a configuration attribute using a system administration tool such as the Admin Server, those changes persist in the *config.xml* file. Configuration values can also be set by modifying the startup scripts and adding additional arguments via the *-D* option in the Java startup command. The *config.xml* file automatically gets updated when you change any configuration settings. When a Managed Server starts up, it contacts the Admin Server and gets a copy of the configuration information, which it stores in memory as configuration MBeans. Thus, all server instances in a domain have the same in-memory representation of the domain's configuration. Note that any attributes you change when starting a Managed Server won't affect the *config.xml* file, which is modified only if you change an attribute value on the Admin Server. When you shut down a server instance, all configuration MBeans hosted by that server are destroyed.

■ *Runtime MBeans* help you monitor running server instances and contain attributes that hold run-time information for server instances and applications. Each of the server's resources updates the relevant run-time MBean following a change in its state. For example, the *ServerRuntimeMBean* is instantiated by the server when it starts up and contains the run-time data for the server. Runtime MBeans consist only of run-time data and nothing else, and when you shut down the server, the run-time statistics in the *ServerRuntimeMBean* are destroyed, as is the case with all the other runtime MBeans.

MBean Servers act as containers for the various MBeans, and the servers create and provide access to the MBeans. Oracle provides three types of MBean Servers. The Admin Server hosts an

instance of the Domain Runtime MBean Server, which manages the MBeans for domain-wide services. Both Managed Servers and the Admin Server host an instance of the Runtime MBean Server, which lets you configure server instances. The Admin Server also hosts the Edit MBean Server, which manages pending configuration changes. The Admin and the Managed Servers also can optionally host the JVM's Platform MBean Server, which controls MBeans that contain monitoring information for the JDK.

You can change most domain configuration attributes dynamically while the server instances are running. In cases where dynamic configuration of an attribute isn't possible, you must restart the server instance. Run-time values of the attributes you configure will reflect your changes immediately, and the values are persisted in the *config.xml* file.

Development and Production Mode

By default, WebLogic Server domains run in the development mode using the Sun Java Development Kit (JDK). In this mode, auto-deployment of applications is enabled and the Admin Server creates a *boot.properties* file automatically when you start it up. You can also use the demo certificates for Secure Sockets Layer (SSL) without any warnings from WebLogic Server. The development mode is provided to get developers up and running quickly without having to worry about advanced deployment, configuration, or security issues.

In the production mode, WebLogic Server defaults to using JRockit as the default JDK. In addition, you can't use the auto-deployment feature in production, and WebLogic Server issues warnings if you use the demo certificates for SSL. In the production mode, you're also prompted for usernames and password when you start up the instances.

It's easy to toggle between the development and production modes, and you learn how to do so in Chapter 2.

Listen Ports and Listen Threads

Listen ports listen for connection requests, and as connections come in the listen thread assigned by the server to the listen port accepts the connection requests, establishes connections, and hands the requests over to the socket muxer.

By default, Oracle WebLogic Server uses two listen ports to listen to incoming requests for connections. The first listen port, which I'll call a normal listen port, will accept any type of request—administrative as well as user requests. The normal listen port accepts connections from various protocols such as HTTP, t3, IIOP, COM, LDAP, and SNMP. When you start up a WebLogic Server instance, it starts listening on two different ports. The first one is a normal plaintext port, and the second is a SSL listen port that also accepts requests for connections from clients over protocols such as HTTPS, t3s, IIOPS, COMS and LDAPS.

The second listen port is called an administration port. When you configure an administration port, the requests must use SSL, at which point you won't be able to direct any administrative requests to the normal port. Here's an informational message from the server when it starts up that shows the two default listen ports in action:

```
<Jan 30, 2011 12:12:01 PM EST> <Notice> <Server> <BEA-002613> <Channel
"Default[7]" is now listening on fe80:0:0:0:e066:c24c:22cc:3d30:7001
for protocols iiop,t3, ldap, snmp, http.>
<Jan 30, 2011 12:12:01 PM EST> <Notice> <Server>  <Channel "DefaultSecure[3]"
is now listening on 192.168.1.2:7002 for protocols iiops, t3s, ldaps, https.>
```

While using the administration port is optional, note that you can start a server in the standby mode only if you use an administration port. In standby mode the normal port will be unavailable, so you must use the administration port to manage the server. In addition, having two separate ports—one for administrative operations and the other for the application traffic—prevents a conflict between these two types of network traffic. In a production environment, you can thus ensure that critical administrative operations, such as starting and stopping servers or deploying applications, don't compete with the application traffic. The administration port accepts only secure SSL traffic, so all connections through this port will have to be authenticated. Note that only administrative users can authenticate on the administration port, and no administrative traffic is rejected on nonadmin ports when you enable the administration port.

Choosing a JVM

In order to run Oracle WebLogic Server, you'll need a Java Virtual Machine (JVM). Oracle offers two types of JVMs for you when you install Oracle WebLogic Server—the Sun Hotspot JVM and the Oracle JRockit JVM. Oracle recommends that you use the JRockit JVM for production installations because of the many benefits offered by it, including higher performance, increased scalability, and better manageability when compared to the Sun JVM.

You configure the default JVM for a domain when creating a domain with the Configuration Wizard or with the WebLogic Scripting Tool (WLST). If you choose Production Mode on the Configure Server Start Mode and JDK page during the Configuration Wizard's domain creation process, the choice of the JVM will default to the JRockit SDK. If you select Development Mode, on the other hand, your domain will be configured to use the Sun SDK.

It's easy to change the JDK after you create the domain. Just set the *JAVA_VENDOR* environment variable in the *startWebLogic.cmd* script (or the *startWebLogic.sh* script in UNIX), as shown here:

```
$ set JAVA_VENDOR=BEA          /* For JRockit JVM
$ set JAVA_VENDOR=sun          /* for Sun JVM
```

In the latest release of WebLogic Server, you can also set the value of the *JAVA_VENDOR* variable to Oracle in order to specify the JRockit JVM. You can confirm the JVM version the server is using by viewing the command window output after you start a WebLogic Server instance. Be sure to check the JRockit documentation for vendor-specific options if you're new to this JVM. You can use JRockit to run any applications that were created with the Sun JDK.

Using Web Server Plug-Ins

While WebLogic Server comes with a built-in web server, you can also use a third-party web server, such as the Apache HTTP Server, for example. Web servers can be used to field requests for simple, static HTML content; but dynamic content, such as that delivered by Java web applications developed as JSPs or servlets, are hosted on the WebLogic Server and the web server routes requests for the dynamic content to WebLogic Server. The web server can use a WebLogic proxy plug-in or the WebLogic Server–provided servlet named *HTTPClusterServlet* to direct servlet and JSP requests to the cluster. You must configure *HTTPClusterServlet* as the default web application on the proxy server machine if you want to use this instead of a proxy plug-in.

You can install a WebLogic plug-in on the web server, allowing it to talk to the applications running on WebLogic Server. Your WebLogic Server installation comes with plug-ins for the following web servers:

■ Apache HTTP Server

■ Microsoft Internet Information Server

■ Sun Java System Web Server

You can use a proxy plug-in to proxy requests from the web server to the clustered WebLogic Server instances to provide load-balancing and failover capabilities for those requests. You can configure the Secure Sockets Layer (SSL) protocol to secure data exchanged between the Apache HTTP Server Plug-In and WebLogic Server. Please refer to Oracle WebLogic Server documentation on WebLogic Server plug-ins for more details about the various available plug-ins.

Although you use WebLogic Server for its capabilities in hosting dynamic enterprise-level applications, you can also use it as a full-fledged web server capable of hosting high-volume web sites and server-static HTML files, servlets, and JSPs.

Management APIs

All the WebLogic Server administration tools and utilities you'll use to manage WebLogic Server call on various WebLogic application programming interfaces (APIs) to perform their tasks. Instead of relying exclusively on the management tools, you can also make use of the rich offering of WebLogic APIs to create your own custom management utilities. Here are brief descriptions of the key management APIs:

■ **WebLogic Diagnostic Service APIs**   These APIs support monitoring of the servers and the access and control of diagnostic data.

■ **Java Management Extensions (JMX)**   JMX is a public standard that you can use for monitoring and managing applications, devices, system objects and service-oriented networks. WebLogic Server uses JMX-based services to manage its resources.

■ **Deployment API**   The deployment API enables the configuration and deployment of applications.

■ **Logging APIs**   Logging APIs help you write messages to log files and distribute those messages. WebLogic Server offers both the standard JDK logging APIs as well as the Jakarta-Log4J Project APIs, as explained in Chapter 4.

■ **Java EE Management API**   This API enables you to create tools to discover resources such as connection pools and deployed applications.

Here are some things to note about the various management APIs:

■ They implement and usually extend the relevant Java specification. For example, the deployment API implements the JSR-88 deployment specification.

■ They enable you to integrate management tasks with other tools that comply with the same specification.

■ The WebLogic Server administration tools, such as the Administration Console, use these APIs to perform various management tasks

Installing Oracle WebLogic Server 11*g*

This section shows you how to install the latest release of WebLogic Server. As you'll see, the installation is remarkably easy. Of course, once you create your WebLogic Server domain, you'll need to spend time configuring it, and this could take a significant amount of time. Chapter 3 explains WebLogic Server domain configuration.

Although the installation steps and screenshots pertain to a Windows installation, they're very similar to the installation steps you need to follow for a UNIX or Linux installation, except for a few operating system differences. All the scripts provided for starting and stopping the servers, for example, come in two versions—a Windows and a UNIX version. For example, the counterpart in UNIX for the Java Windows command script for starting a Managed Server, *startManagedWebLogic.cmd*, is the *startManagedWebLogic.sh* script.

You can download a full-fledged version of WebLogic Server from the Oracle web site. Navigate to www.oracle.com from your browser, click Downloads, then WebLogic Server. After accepting the license agreement, simply download the appropriate Oracle WebLogic Server packages for your operating system. You can also access the WebLogic Server installation software from the Oracle Technology Network (OTN) web site or the Oracle E-Delivery web site (http://edelivery.oracle.com).

You'll notice that you have multiple options to select from with regard to the installation packages. If you wish, you can use the Oracle WebLogic Server installer to install Oracle Coherence and the Oracle Enterprise Pack for Eclipse, along with Oracle WebLogic Server. If you already have Java installed on your server, you can choose the Generic Package Installer. Several operating system package installers are available for specific platforms. My Oracle Support now offers an upgrade installer that lets you upgrade your current installation to the latest patch release. If your current 10.3.*x* installation includes Workshop for WebLogic, you must first uninstall the Workshop before you perform an upgrade with the Upgrade Installer.

Since our main goal is to learn about Oracle WebLogic Server, you can download just the basic Oracle WebLogic Server 11.1 zip files, but these are meant mainly for developing applications and don't have the sample applications (and the Derby database that goes with it) or the WebLogic plug-ins. Therefore, it's a smart idea to go ahead and download the more complete installer, named Oracle WebLogic Server 11*g*R1 (10.3.4) + Coherence + OEPE - Package Installer. This way, if you want to work with Oracle Coherence or the Oracle Enterprise Pack for Eclipse, you'll have access to those products.

Development-Only Installation

On most operating systems, you can use a development-mode installation that comes in the form of a zip file. This installation is purely for development purposes. Although it has a smaller footprint than the full-deployment version, note that the development-only installation doesn't come with the web server plug-ins, the Sun or JRockit JDK, the sample applications, or the Derby database.

Let's turn to how you install Oracle WebLogic Server on a Windows system.

Installation Prerequisites

The installation procedures explained here are for the Windows platform, and they're mainly designed to get a working installation of WebLogic Server up and ready so you can play with it. For example, the prerequisites for a basic installation require just 1GB of RAM and a 1 GHz processor. As for disk space for the installation, it takes about 1.6GB for the entire installation (including Oracle Coherence and Oracle Enterprise Pack for Eclipse). For actual production implementations, you must refer to the appropriate requirements by going to the following link: www.oracle.com/technology/software/products/ias/files/fusion_certification.html.

A key requirement is that you must have a JDK (Java Development Kit) installed prior to the installation, but for Windows 32-bit and Linux platforms, the installer comes with a JDK distribution (Sun and JRockit), so you're fine with the JDK requirement.

The following example shows how to install WebLogic Server through the executable that's included with the installer. However, you may also install from the command line, which lets you denote a specific temporary directory instead of the default TEMP directory that the installer uses:

```
$ wls1034_oepe111161_win32.exe -mode=console -Djava.io.tmpdir=C:\Temp1
```

Note that the following procedures install Oracle WebLogic Server Release 11.1—specifically the 11.1.1.4 release.

Installation Modes

There's more than one way to install WebLogic Server. The first and easiest method is to use the graphical mode, which is an interactive mode. The console mode is also an interactive mode, but is run from the command line. The silent mode is a noninteractive mode of installation, where you can use a script or a text file when you need to install WebLogic Server on many hosts. The example that follows uses the graphical mode to install WebLogic Server.

Oracle offers two types of product installers for WebLogic Server:

- Oracle WebLogic Server and Oracle Coherence
- Oracle WebLogic Server, Oracle Coherence, and Oracle Enterprise Pack for Enterprise

In most operating systems, the installer will also automatically install the Java run-time JDKs. The two types of JDKs available are the Sun JDK and the Oracle JRockit JDK. In production mode, Oracle recommends that you use the JRockit JDK.

Installation Procedures

Follow these steps to install WebLogic Server using the graphical mode:

1. Double-click the executable you downloaded from the Oracle web site (*wls1034_oepe111161_win32.exe*) and click Run in the Open File box. The installer starts preparing itself for the installation, a process that takes a couple of minutes.
2. Click Next in the Welcome screen.

3. In the Choose Middleware Directory, specify the directory where you want the installer to place the WebLogic Server files. Your domains will also, by default, use this directory. In our case, the directory is C:\MyOra\Middleware. Of course, if you've installed Oracle Middleware products before already, you can choose to install WebLogic Server in that directory as well. Click Next.

    Oracle recommends that if you already have an Oracle Middleware installation, you should point to it instead of choosing a brand new directory.

4. In the Register for Security Updates screen, you have the option of specifying your My Oracle Support e-mail address if you choose to receive security updates by e-mail. Since we're merely using this installation for testing purposes, you can just click Next. Note that this step also lets you use the Configuration Manager, if you wish.

5. In the Choose Install Type window, shown here, select the Custom option.
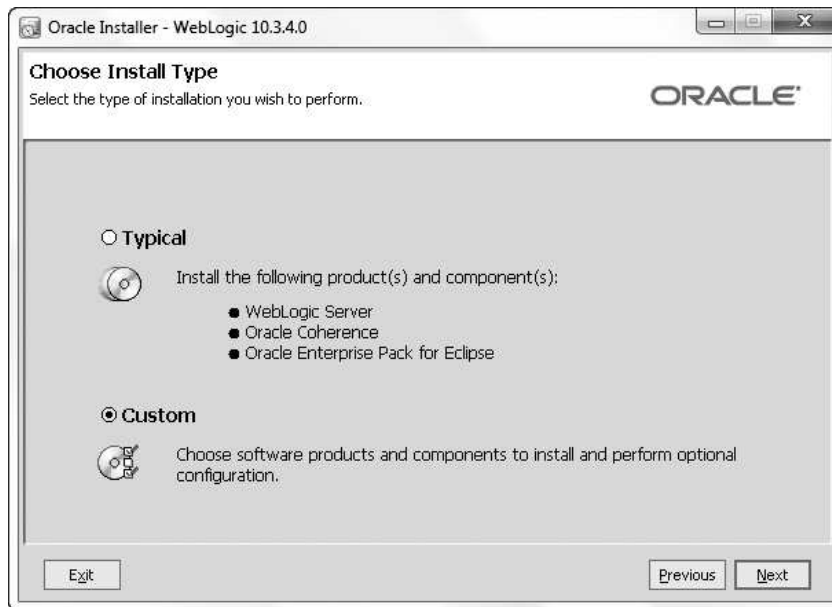
    The Custom option lets you choose the products and components you want to install, as well as perform optional configuration. Click Next after making your selection.

6. In the Choose Products and Components window, Server Examples is unchecked in the list of products by default. Server Examples include some applications that you can use to learn about WebLogic Server, and it comes with its own sample database server named Derby Server. This book makes heavy use of the example domains and applications provided by WebLogic Server. You must check the Server Examples box in the product list to install these applications. Click Next after making your selection.

7. In the JDK Selection page, the installer selects by default both the JDKs it offers—Sun SDK and the JRockit SDK—but you can also browse through your server directories to select another JDK if you wish. Click Next.

8. In the Choose Product Installation Directories window, shown next, the installer shows the directory you chose in step 3 as the home for the Oracle Middleware products. You can't change this without going back. The installer also shows three other directories, all under the Oracle Middleware Home directory (C:\ora\Middleware). These three directories are for Oracle WebLogic Server (C:\ora\Middleware\wlserver_10.3), Oracle Coherence, and Oracle Enterprise Pack for Eclipse. You can change these directories if you wish.
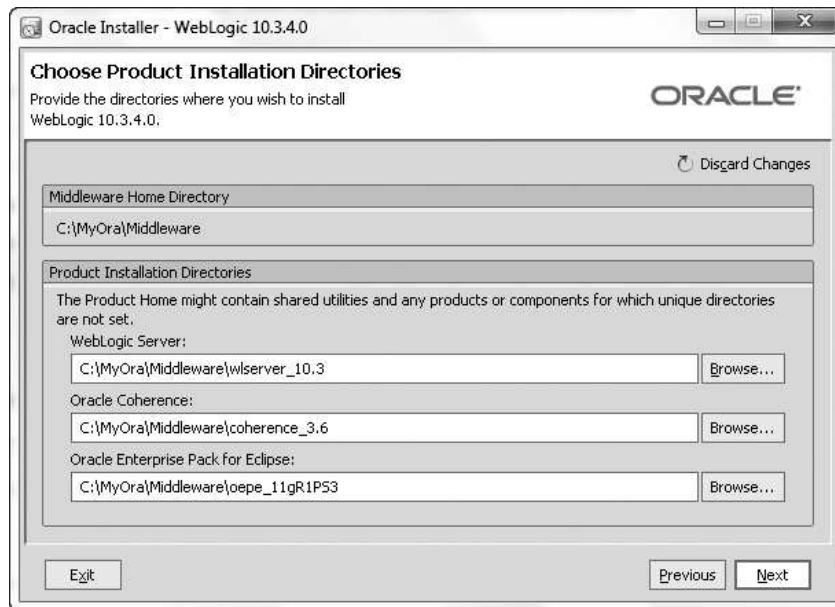
The default values are generally acceptable, so click Next to continue.

9. The Install Windows Service window offers you a chance to install the Node Manager, which you use as a Windows service to control the Managed Servers. Select No for now, since Chapter 4 will show you how to do this in detail. Click Next to continue.

10. In the Choose Shortcut Location, select the recommended "All Users" Start Menu Folder and click Next.

11. Review the Installation Summary window and click Next once you're satisfied with the products you chose to install.

12. Once the installation completes, you'll see the following Installation Complete window.
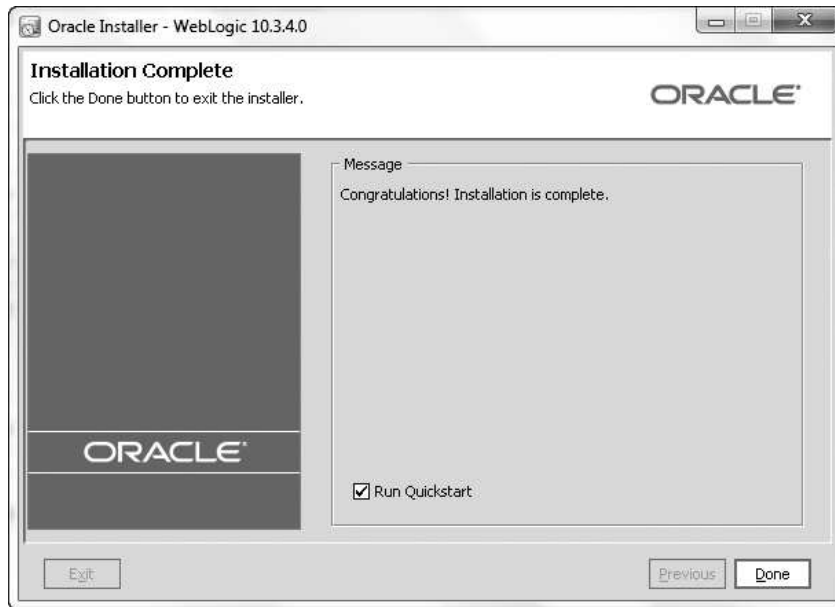
13. You'll notice that the Run Quickstart check box on the Installation Complete page is checked by default. If you click Done, that will let you configure your first domain. However, let's wait until Chapter 4 to learn all about creating and configuring domains. Uncheck the Run Quickstart box and click Done.

The installation of Oracle WebLogic Server 10.3.4 is complete at this point. It was easy, wasn't it?

Checking the Installed Features

Once the installation is completed, the installer places the WebLogic Server icons in the Windows Start program. Go to Start | All Programs | Oracle WebLogic, where you'll find the new WebLogic Server program components under Oracle WebLogic. Click Oracle WebLogic to explore the installed products, which are summarized next.

Online Documentation

This is a link to the Oracle WebLogic Server 11*g* documentation, so you have the relevant Oracle manuals at your fingertips.

Oracle Enterprise Pack for Eclipse

This is a link to Oracle Enterprise Pack for Eclipse (OEPE), which appears only if you chose to install this along with WebLogic Server.

Smart Update

Smart Update is a new feature that lets you download patches and maintenance packs from Oracle Support, provided you have a valid Oracle Customer Support Identifier.

QuickStart

When you install Oracle WebLogic Server using the GUI mode, the installer will automatically launch the QuickStart application once your WebLogic Server installation completes, if you select the Run Quickstart check box. You can launch QuickStart from here at any time.

   The purpose of QuickStart is to provide new users an easy way to get started. WebLogic Server QuickStart lets you do the following, without any prior experience with WebLogic Server:

- Create a new domain by invoking the Configuration Wizard
- Upgrade an existing WebLogic domain to the latest version
- Start the WebLogic Server samples domain

On a UNIX or Linux platform, you access QuickStart as follows:

1. Go to the /common/bin directory of the WebLogic Server installation:

   ```
   $ $MW_HOME/wlserver_10.3/common/bin
   ```

2. Run the shell program that starts the QuickStart application:

   ```
   $ ./quickstart.sh
   ```

Uninstall Oracle WebLogic

The Uninstall Oracle WebLogic button lets you access the Oracle Uninstaller to easily remove an existing WebLogic Server installation. The Uninstaller removes the entire WebLogic Server Platform installation with just a single click. On a Windows system, for example, the Uninstaller removes all files, shortcuts, Windows registry keys, and registry entries related to WebLogic Server.

WebLogic Server 11*g*R1

This button offers you the following shortcuts.

Online Documentation   This is one more way to access the Oracle WebLogic Server 11*g* Release (11.1.1.4) documentation.

WebLogic Server   You can click this button to start the Admin Server for the *wl_server* domain that's automatically configured during the installation process.

Examples   When you choose to install the samples during installation, the installer creates three sample domains for you—*wl_server*, *medrec*, and *medrec-spring*. (Note that the Examples Server you see under the Oracle WebLogic programs under All Programs is the Admin Server for the *wl_server* domain.) These sample domains are discussed later in this chapter.Here, you'll find the buttons to stop and start the Admin Servers for the three sample domains: Examples Server, MedRecServer, and MedRecServer (Spring Version).

Tools   Under Tools, you'll find several important wizards—the Configuration Wizard helps you create a domain or modify and extend an existing domain. The Domain Upgrade Wizard provides an automated way to upgrade WebLogic Server all the way from the 6.1 release to the current 10.3.4 release. The Domain Template Builder helps you create domain templates that you can use with the Configuration Wizard to easily create new domains. A domain template provides preconfigured settings that include database components, services and security, and other environmental options. The Domain Template Builder also helps you create extension templates that you can use with the Configuration Wizard to update WebLogic domains. Finally, you can click the Node Manager and the WebLogic Scripting Tool shortcuts to access these key administrative tools. I introduce both tools toward the end of this chapter.

Reinstalling WebLogic Server

If you need to reinstall an identical version of WebLogic Server in the same location as a previously existing installation for any reason, first remove the previous installation by clicking the Uninstall Oracle WebLogic shortcut under Start | All Programs | Oracle WebLogic. This will invoke the Oracle Uninstaller wizard, which leads you through the simple steps necessary to remove the previous installation. Make sure you stop all running WebLogic Server instances before you start uninstalling—that way, the Uninstaller will automatically remove all the files and directories under WebLogic Server Home.

You can add new products to an existing installation, but you can't reinstall the same WebLogic Server release over an existing WebLogic Server installation of the same release.

Exploring the Installation Directories

Now that you've seen how easy it is to install WebLogic Server, let's explore the installation directories a bit. As I mentioned during the installation steps, you have two major home directories—Oracle Middleware Home and Oracle WebLogic Server Home. The Oracle Middleware Home directory is where all the WebLogic Server and other middleware product files are located—it's the top-level directory for all Oracle Fusion Middleware products, including the Oracle WebLogic Server. In our example here, there's only a single middleware product, which of course is the Oracle WebLogic Server. During the installation, I chose C:\MyOra\Middleware as the Oracle Middleware Home directory. Under the C:\MyOra\Middleware directory, there's a wlserver_10.3 directory that serves as the Oracle WebLogic Server Home directory, denoted by WL_HOME. Remember, however, that by default the Oracle Installer installs WebLogic Server under the Middleware Home directory, but there's no strict requirement that this needs to be so always—you can choose to create the WebLogic Server Home in any directory you choose, including a brand new directory for which you need only provide the name. The installer will automatically create that directory for you. If you've installed and removed WebLogic Server earlier, Oracle recommends that you reuse the same directory for your new installation.

Table 1-1 shows the main directories under the Oracle Middleware Home directory. Note that the last directory in the table is your WebLogic Server Home directory and that it's usually denoted by the environment variable *WL_HOME*.

Let's review what I've accomplished thus far: I've successfully installed the Oracle WebLogic Server software, located in its home directory, C:\MyOra\Middleware\wlserver_10.3, but I have not yet created a custom domain. There's not a whole lot you can do with this installation, until you create a WebLogic Server domain. When you create a domain, you'll automatically have one Admin Server and can also create multiple Managed Servers or clusters to host your web

Directory	Contents
/coherence_3.6	Home Directory for Oracle Coherence
/jdk160_21	Directory for the default Sun JDK
/jrockit_160_22_D1.1.1-3	Directory for the JRockit JDK
/logs	Location for various log files, including those for WLST
/modules	Location for modules such as Apache Ant
/oepe_11gR1PS3	Files related to Oracle Enterprise Package for Eclipse
/user_projects	Standard location for domains
/utils	Various utilities such as cloning scripts and QuickStart
/wl_server_10.3	WebLogic Server Home directory

TABLE 1-1. *The Oracle Middleware Home Directory*

applications. Chapter 3 is devoted to managing and configuring domains. In that chapter, you'll learn how to create domains and configure servers so you can get ready to deploy and run web applications through Oracle WebLogic Server.

WebLogic Server Home

The WebLogic Server home is simply the directory where I installed WebLogic Server, and by default it's located in the MW_HOME\wlserver_10.3 directory. You refer to this home as the WL_HOME directory, distinguished from the Oracle Middleware Home directory, which in my example is C:\MyOra\Middleware. Thus, the complete path of the WebLogic Server home is C:\MyOra\Middleware\wlserver_10.3.

Under the WebLogic Server Home (WL_HOME), you'll find the following directory structure:

- /bugsfixed
- /common
- /inventory
- /samples
- /server
- /sip
- /uninstall

The bin directory under the WL_HOME server directory contains the *startNodeManager* script to start up the Node Manager. Under the /samples directory you'll find the default domains installed by the Oracle Installer with the help of the WebLogic Server Configuration Wizard when you chose to install the examples during the installation process. There are three such domains, all located under the /samples directory. The domains are actually in the WL_HOME/samples/ domains directory. When you open this directory, you'll find three domain directories, one for

each of the domains the installer creates for you. These are the *medrec*, *medrec-spring*, and *wl_ server* domains. The next section explores the contents of these domain directories, all of which have the same structure.

WebLogic Server Domain Directory

Each domain that you create will have the following directory structure:

- ■ /bin
- ■ /config
- ■ /console-ext
- ■ /init-info
- ■ /lib
- ■ /nodemanager
- ■ /security
- ■ /servers

Under the /bin directory of each domain is where you'll find the various scripts to start and stop the Admin and Managed Servers, such as *startWebLogic.cmd* and *stopWebLogic.cmd*. Note that UNIX versions of these scripts are also located in this directory.The all-important domain configuration file, *config.xml*, is stored in the /config directory.

For now, it's enough for us to be aware of the basic structure of a WebLogic domain. Chapter 3 details how to configure a domain, and it's better to postpone the detailed examination of a domain directory until that point.

The WebLogic Server Sample Applications

In order to demonstrate the basic features of the Administration Console, let's use one of the three sample domains created during the installation when you chose to install the samples using the Custom installation option. The code examples provided by Oracle are located in various domains, all under the Samples directory. I understand that some readers don't need to use the sample applications at all because they already have a working knowledge of WebLogic Server. However, for those new to Oracle WebLogic Server, the sample applications provided by Oracle help you understand web applications, and the sample domains help you learn how to administer and manage WebLogic Server.

NOTE
All the sample domains the Configuration Wizard creates for you during the WebLogic Server installation (if you choose to install the samples) are located by default in the WL_HOME/samples/domains directory. However, as you'll see in Chapter 3, when you create your own domains with the Configuration Wizard, by default the wizard places all the domains under the MW_HOME/user_projects/domains directory. (In our example, MW_HOME is C:\MyOra\Middleware.) You can specify alternative locations for the domain directories.

The WebLogic Server samples contain two different types of applications to help you get familiar with enterprise Java EE applications and understand how Oracle WebLogic Server works. The first set of applications are part of athe domain called *wl_server*, and the domain's Admin Server is named Examples Server. The *wl_server* domain contains Oracle WebLogic Server API examples designed to show you how to implement Java EE APIs and related Oracle WebLogic Server features. Oracle also provides a web application called examplesWebApp, which includes several of these examples. You'll also find detailed instructions on using the various examples. In addition, there's a full-blown sample Java EE web application by the name of Avitek Medical Records as part of the domain named *medrec*. When you choose to install the examples, two versions of the Avitek Medical Records application are installed for you. The first one is the MedRec application designed to show various features of the Java EE platform. The second application, called MedRec-Spring, is the same as the MedRec application, but is created using the Spring Framework, and is part of the *medrec-spring* domain.

Oracle recommends that you start working with the *wl_server* domain to understand the basics of Java EE programming and Oracle WebLogic Server. If you're already familiar with both of these, it's a good idea to check out the Avitek Medical Records and the Avitek Medical Records (Spring) sample applications. Both of these present you a realistic example of how to develop and deploy full-blown Java EE applications. The two applications also serve as great learning tools for Java EE developers as well as for WebLogic Server administrators who wish to understand application deployment concepts.

Starting the Examples Server

The Examples Server is the Admin Server for the *wl_server* domain and contains basic web application examples. To launch the Examples Server, go to Start | Oracle WebLogic | WebLogic Server 11gR1 | WebLogic Server. Once you click the WebLogic Server button, the Admin Server for the sample domain is started.

TIP
You don't need to install the WebLogic Server Examples on your production servers because they're purely for learning purposes. Leaving them on a production server introduces vulnerabilities that can be exploited by hackers.

The Admin Server will start booting and you can follow the boot sequence in the command window that pops up. You'll also see a separate command window that shows the starting up of the Derby database for the Examples Server. Once the Admin Server boots up, you'll see the following in the command window:

```
..
JAVA Memory arguments: -Xms512m -Xmx512m
.WLS Start Mode=Development
.CLASSPATH=C:\MyOra\Middleware\wlserver_10.3\samples\server\examples\build\
**************************************************
*  To start WebLogic Server, use a username and   *
*  password assigned to an admin-level user.  For *
*  server administration, use the WebLogic Server *
*  console at http:\\hostname:port\console        *
**************************************************
```

```
starting weblogic with Java version:
Oracle JRockit(R) (build R28.0.0-679-130297-1.6.0_17-20100312--windows-ia32,
Starting WLS with line:
C:\MyOra\MIDDLE~1\JROCKI~1.0-6\bin\java -jrockit   -Xms512m -Xmx512m -Dwebl
<Jan 16, 2011 6:44:23 PM EST> <Notice> <WebLogicServer> <BEA-000365> <Server
state changed to RUNNING>
<Jan 16, 2011 6:44:23 PM EST> <Notice> <WebLogicServer> <BEA-000360> <Server
started in RUNNING mode>
```

In addition to the main Windows command console (don't close it or else your server instance will promptly die!) that displays the server life cycle messages throughout the server's life, you'll also see another window that shows that the default Derby database server is also up and ready to receive requests. (You can change the database server to a different server, say Oracle Database 11*g*, later on.) Here are the Derby Server window contents when it starts up:

```
2011-02-03 01:08:31.980 GMT : Security manager installed using the Basic
server security policy.
2011-02-03 01:08:33.384 GMT : Apache Derby Network Server - 10.6.1.0 -
(938214) started and ready to accept connections on port 1527
```

Once you see the WebLogic Server started in RUNNING mode, the Examples Server is ready for your use. Your browser will automatically launch at this point and show the Oracle WebLogic Server Samples Introduction Page, which is the gateway to the sample applications. If, for some reason, the browser doesn't automatically launch, you can go to the following URL to see the page:

```
http://localhost:7001/examplesWebApp/index.jsp
```

Note that port 7001 must be available for you to access the Administration Console for this domain.

To launch one of the other sample applications, for example, the Avitek Medical Records Sample Application, run the following command, which starts the Admin Server for the medrec domain:

```
C:\> C:\MyOra\Middleware\wlserver_10.3\samples\domains\medrec\startWebLogic.cmd
```

This command will start the application and display the startup page. You can click the Start Using MedRec button to start the application. You can also start the Administration Console to manage the MedRec domain by clicking the Start the Administration Console button.

Stopping the Server

You can stop a running server by closing the command window or by pressing CTRL-C in the command window. However, in production environments you use a shutdown script to stop the servers. You can use the following command script to shut down the Admin Server for the medrec domain, for example:

```
C:\> C:\MyOra\Middleware\wlserver_10.3\samples\domains\medrec\bin\stopWebLogic.cmd
```

On a Windows server, it's very easy to start up and stop any of the sample servers. Go to Start | All Programs | Oracle WebLogic | WebLogic Server11gR1. There, you'll find shortcuts to start and stop the Medrec Medical Records Server (including the Spring version) and the Examples Server. Note that you need to point your browser toward a different port number to access the

Administration Console for each of these servers. By default, the example domains run in the development mode. You can configure all servers in a domain to run in production mode by clicking Domain in the Administration Console home page and checking the Production Mode box. You must first click the Lock & Edit button in the Change Center and activate the change. You must also restart the server so it can start in production mode. All servers in this domain will now run in production mode. Note that you can't toggle back to a development mode once you enable the production mode—you can disable the production mode only at the Admin Server startup command line by specifying the *-Dweblogic.ProductionModeEnabled=false* option, as explained in Chapter 2.

Upgrading Oracle WebLogic Server

The latest version of Oracle WebLogic Server, as of the writing of this book, is the 10.3.4 release. You can upgrade to this release from earlier versions of WebLogic Server. When you upgrade Oracle WebLogic Server, not only must you install the new software, of course, you have to upgrade the existing domains, the Node Manager, and the security providers. If you're upgrading from WebLogic Server 9.*x*, or 10.0 to the 10.3.4 release, upgrading the domain is optional because the domains from these releases will run without modification in 10.3.4. If you're upgrading to the 10.3.4 release from earlier versions, however, you must follow specific upgrade procedures following the installation of the new release. The following sections provide a summary of the upgrade procedures.

Upgrade Tools

You can use the WebLogic Upgrade Wizard to facilitate server upgrades. You can run the wizard in interactive or silent mode. You don't have to use the wizard, but it'll certainly make life easier for you during an upgrade. You can, for example, manually upgrade a domain by installing the software for the current release, updating the domain script files to point to the new installation, and updating the CLASSPATH to remove outdated information. As you can see, the Upgrade Wizard can prevent many headaches with its automated approach to the upgrade.

NOTE
If you're upgrading from a pre-9.2 release, your administrative scripts may not work due to changes in the MBean hierarchy. You are better off updating your scripts to incorporate the current capabilities of tools such as WLST, Node Manager, and wlconfig.

Upgrade Procedures

When you upgrade to a newer release of WebLogic Server, in most cases you don't have to upgrade the web applications you deploy. The latest release of WebLogic Server, 10.3.4, will work with all applications you created on WebLogic Server 7.0, 8.1, 9.*x*, 10.0, or any 10.3 release servers. However, you have to upgrade several components of the server:

- Domains
- Custom security provider
- Node Manager

In addition, you need to ensure that any external resources WebLogic Server connects to, such as an Oracle database, for example, are compatible with the new release. The following sections briefly describe the upgrade procedures. Before you start the actual upgrade process, however, do the usual due diligence effort, such as verifying the supported configurations and the compatibility of the various software applications, as well as doing an inventory of your current WebLogic Server environment. As with any upgrade of a server, back up your applications, shut down all running servers, and start by installing the new release of the Oracle WebLogic Server software.

As mentioned earlier, you may not have to do much to make your current applications run on the latest release of WebLogic Server. However, you must upgrade the security providers, the domain, the remote Managed Servers, and the Node Manager. The following sections briefly explain each.

Upgrading the Security Provider

During the upgrade of the security providers, the Upgrade Wizard upgrades the JAR files for security providers so the providers can work under a 10.3.4 environment. If you're using a custom security provider in the 7.0 or 8.0 release, the Upgrade Wizard can upgrade those providers to run in a 10.3.4 environment as well.

Upgrading the Node Manager

Upgrade the Node Manager (on all machines where they currently run) only if you intend to use any customized versions in the new environment. Otherwise, there's nothing for you to do here during an upgrade. Once the upgrade is completed, you must enroll the Node Manager with all machines, and Chapter 2 shows you how to do this.

Upgrading Existing Domains

You must use the Upgrade Wizard to upgrade domains created in the WebLogic Server 7.0 or 8.1 release. You can, but don't have to, upgrade a domain created in the 9.*x* or 10.*x* release because these domains run under WebLogic Server 10.3.4 without any changes. Before upgrading any remote Managed Servers, upgrade the domain on the machine where the Admin Server resides. If you have any Managed Servers on the same server as the Admin Server, you don't have to upgrade them. Upgrading the domains updates the *config.xml* file—the key domain configuration file—and also updates persistent data in the JMS file stores.

Upgrading the Remote Managed Servers

During an upgrade, you need to upgrade just those Managed Servers that reside on remote servers. Before upgrading, you must first copy two important files (*config.xml* and *SerializedSystemIni.dat*) from the root directory of the original domain directory of the Admin Server to the root directory of the remote Managed Server domain.

Performing a Rolling Upgrade

In these days of hard-to-find downtime windows and 24×7 uptime, it's not always easy to perform production WebLogic Server upgrades without adding some risk to the operation of your enterprise applications. If you're using a WebLogic cluster, you can perform many types of minor release upgrades and patching without having to shut down the domain. Even if you aren't using a cluster but are just running multiple Managed Servers in your domain, you can still perform a rolling upgrade of all the servers without first having to shut down the entire domain. Rolling

upgrades keep you from having to shut down your domain during an upgrade, by simultaneously allowing you to utilize an upgraded domain in parallel with the domain from the earlier release.

You must be using at least the WebLogic Server 9.2 release to avail yourself of the rolling upgrade capability. Also, it's important to understand that you can perform a rolling upgrade only within a product family, say from 10.0 to 10.4, but not from the 9.0 release to the 10.0 release, for example. Rolling upgrades are easy to perform, with the only requirement being that you maintain two separate product directories for the two versions of the software during the upgrade process—you can't overlay the new version on the older version during the upgrade.

You can perform a rolling upgrade manually with operating system commands or with the Smart Upgrade tool. You can also do this via the silent installation method if you want to upgrade several servers at once. The following sections offer a brief summary of the rolling upgrade procedures for patches, maintenance packs, and minor versions.

TIP
Both the patch and the Upgrade Installer install the latest version of JRockit, but WebLogic Server will be unaware of this version until you set the value in the setDomainEnv.cmd *file, as shown here:*

```
set BEA_JAVA_HOME=C:\MyOra\Middleware\jrockit_160_22_D1.1.1-3
```

Rolling Upgrades for Patches and Maintenance Packs
Here are the steps to install a single patch or a maintenance pack consisting of a set of fixes to an existing installation:

1. Shut down the Admin Server. (Remember that once the Managed Servers are up and running, they can continue to service application requests even if the Admin Server is brought down.)

2. Patch the installation files on the server where the Admin Server was running earlier.

3. Restart the Admin Server.

4. For the Managed Servers, you follow a sequential patching approach: shut down each Managed Server in turn, patch the server, and restart it. When you finish patching and restarting the last Managed Server on your list, you're done with the entire patch process.

NOTE
Before shutting down any Managed Servers to perform an upgrade, ensure that the web servers or load balancers are stopped from sending requests to the particular Managed Server.

Rolling Upgrade Process for a Minor Version
The procedure for upgrading to a new minor release is similar to the rolling patch process, with some additional steps, as shown here:

1. Shut down the Admin Server.

2. In a completely new directory, install new WebLogic Server installation files on the server running the Admin Server.

3. Update the start and stop scripts (*setDomainEnv.cmd*, *startWebLogic.cmd*, and *stopWebLogic.cmd*) by pointing the environment variables such as *MW_HOME*, *JAVA_HOME*, and *WLS_HOME* to the new installation.

4. Restart the Admin Server.

5. For the Managed Servers, follow a sequential rolling upgrade process by stopping each Managed Server, installing the new software, and updating all start scripts and environmental variables.

Be careful not to make any configuration changes until every Managed Server in the domain has been upgraded. There is nothing to upgrade in a cluster itself because it's just a set of Managed Servers.

Thus far, you've learned how to install and upgrade WebLogic Server. Before we discuss the management of servers (see Chapter 2) and the creation and configuration of WebLogic Server domains, I'd like you to get acquainted with the key administration tools you must be familiar with in order to do all the fun stuff with WebLogic Server. The following sections introduce you to the three key management tools—the Administration Console, the Node Manager, and the WebLogic Scripting Tool (WLST).

Using the Administration Console

WebLogic Server offers a browser-based Administration Console to help manage a domain The Admin Server hosts the Administration Console application, and you can access the Console from any browser which has network access to the Admin Server.

The Administration Console lets you administer your entire domain—the server instances, web applications, modules, and all the resources that the applications and modules need to use. Not only can you configure and monitor the servers, you can create new server instances with the console. The console also helps you tune your applications. The console makes it easy for you perform various configuration and management tasks without having to learn how to use the underlying JMX API, which is what you need to manually configure the domains. In Chapter 2, you'll learn the various ways in which you can start and stop WebLogic Server instances. The easiest as well as the recommended way to manage your servers is through the Administration Console. You can even edit and save changes to the domain configuration file, *config.xml*, through the console. Throughout this book, you'll learn how to configure various aspects of WebLogic Server through the Administration Console.

TIP
If you're new to the Administration Console, it pays to check out the excellently detailed help material you can access from there by clicking the How Do I link, where you'll find crystal clear steps for performing any task within the Administration Console.

Since the Administration Console is linked to a domain, there's no Administration Console until you create a domain. When you create a domain, by default a single Admin Server is created for you. It's the Admin Server that runs the web-based Administration Console that

enables you to manage the entire domain. Thus, you must first start the Admin Server before you can access the Administration Console. Once you create a domain (see Chapter 3), you can access the Administration Console at the default port 7001, but you can assign the console any other free port number if you wish.

NOTE
You can disable the Administration Console by clearing the Console Enabled box in the Administration Console's configuration page for the Admin Server. If you do this, you can then manage the domain only with management APIs.

Any configuration changes you make through the Administration Console will update the *config.xml*, which is the domain configuration file.

Logging into the Administration Console

Once you've created a domain, start up the Admin Server first. Once the Admin Server is in running mode, you can access the Administration Console and manage the domain. Invoke the Administration Console by using the following URL:

```
http://localhost:7001/console
```

Note that 7001 is the default port that WebLogic Server uses. You can set the port to any valid port number you choose. In the following example, as explained in the preceding section, let's use the Examples Server (Admin Server for the wl_server domain) provided by the installer. Start up the Examples Server by going to Start | Oracle WebLogic | WebLogic Server 11gR1 | WebLogic Server. Since every domain will host its own Admin Server, if you are running multiple domains on your machine, each Admin Server will have to bind to a different port. For example, you can access the Administration Console for the *wl_server* domain at:

```
http://localhost:7001/console
```

Meanwhile, the Administration Console for the *medrec-spring* domain can be accessed at:

```
http://localhost:7011/console
```

If you're using Secure Sockets Layer (SSL) to start your Admin Server, use https instead of http, as shown here, and note that you use a different port number from that of the non-SSL port:

```
https://localhost:7002/console
```

TIP
If you've configured a proxy server, configure your browser so it doesn't direct the Admin Server requests to the proxy. If you're running both the Admin Server and your web browser on the same server, make sure that the requests are sent to local host (or IP 127.0.0.1) and not to the proxy server.

FIGURE 1-1. *The Oracle WebLogic Server Administration Console login page*

The default administrative username for the Admin Server is weblogic, and the default password is welcome1. You may also log in later on by choosing a username that you granted a default global security role. If you grant the default global security role Admin to a user, for example, that user can perform any task using the Administration Console. If you gave another user a more limited security role such as Deployer, Monitor, or Operator, the user won't be able to edit the configuration data—such users can only view but not modify the configuration data.

In the Administration Console login page, shown in Figure 1-1, enter the default username and password (**weblogic** and **welcome1**, respectively), or for a custom domain, use the username and password combination you chose during domain creation. You can log out of the console by clicking the Logout button at the top of the right-hand pane of the console.

Once you successfully log in to the Administration Console, you'll see the initial Home page, as shown in Figure 1-2. You'll notice that the Home page of the Administration Console contains two panels. The left panel is where your resources and servers are listed. At the top of the right panel are the Log Out and Preferences buttons. When you click a server under a domain, the relevant configuration items will show up on the right-hand side screen, from where you can check or modify the server's configuration.
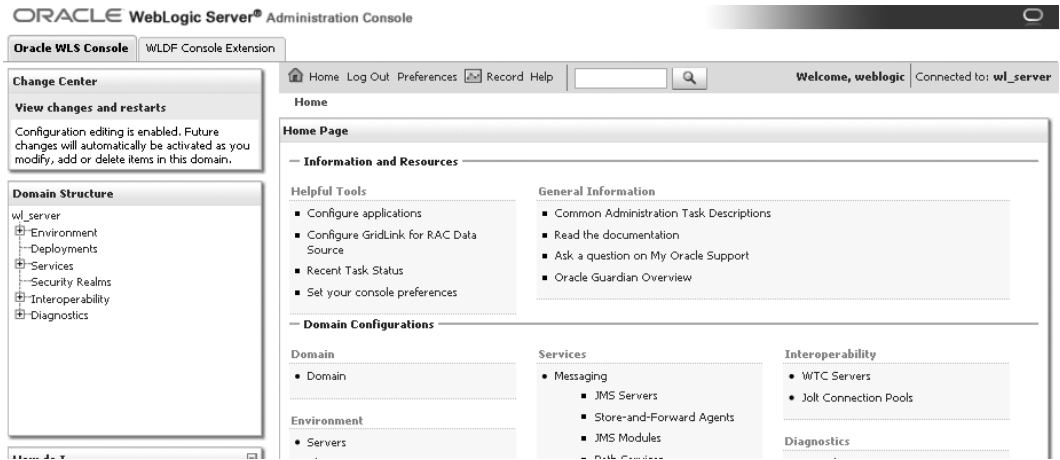
FIGURE 1-2. *The Administration Console Home page*

Navigating the Administration Console

The tree menu on the left pane of the Home page provides quick access to functionality that allows you to manage not only the servers and clusters but also the configuration of resources such as JMS servers and data sources. For example, by expanding wl_server | Services | Data Sources, you will see a complete list of data sources configured on this domain. Under Domain Structure, you'll notice several key items, which I explain here.

Environment

You'll find the following under Environment:

- Servers
- Clusters
- Virtual Hosts
- Migratable Targets
- Coherence Servers
- Coherence Clusters
- Machines
- Work Managers
- Startup and Shutdown Classes

For example, if you click Servers, you'll then see in the right-hand side pane the configuration page for the lone server in the domain, *examplesServer*, which is the Admin Server for this domain (*wl_server*), as shown in Figure 1-3.

FIGURE 1-3. *The Configuration page for the Administration Server*

Deployments
This group takes you to the deployments page in the right-hand pane, from where you can manage the enterprise applications or web modules you've deployed. You can start, stop, redeploy, or remove an application or module from this page.

Services
Important resources you can manage include messaging, consisting of Java Messaging Service (JMS) servers and JMS modules, Java Database Connectivity (JDBC) data sources, and Java Transaction APIs (JTA).

Security Realms
This group contains all security realms you have configured for this domain. Select a realm from under Security Realms in the left pane. When you do this, you'll find all the subnodes for all the security providers in a realm, providing you access to a realm's users, groups, and roles. The Administration Console lets you configure any aspect of a security realm.

Interoperability
This group contains features to allow your applications to operate with Tuxedo Services, such as the WebLogic Tuxedo Connector and Jolt, a Java-based client that manages requests made to Oracle Tuxedo Services.

Diagnostics

This group contains diagnostic modules and diagnostic images (snapshots) to help manage the WebLogic Diagnostic Framework (WLDF). You can also configure Simple Network Management Protocol (SNMP) agents from here.

Using the Change Center

The Administration Console offers the Change Center, where you can lock a domain's configuration while you're making changes to any configuration attributes. By default, the Change Center is always enabled when you run a server in production mode, and it's disabled in development mode. In order to make lasting configuration changes from the Administration Console, you must first obtain a lock, make your changes, and activate them. By doing so, other accounts are prevented from making changes during your edit session, thus preventing conflicting or overlapping configurations from being made.

   Instead of making configuration changes piecemeal, you can make multiple changes and activate them all at once. You can click the View Changes and Restarts button to view all the pending changes that you have applied but not activated yet. The Change List page shows you all changes that are saved but not yet activated. By clicking the Restart Checklist tab, you can view the changes that have been activated but are waiting for a server restart before they become effective.

   In development mode, the domain configuration locking feature is disabled by default. That is, the Automatically Acquire Lock and Activate Changes property is enabled. Automatic locking means you don't explicitly have to acquire a lock on the domain configuration before making any changes to it. When you look at the top left-hand corner of the Administration Console, you'll find the following note: "Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain." This means that when you make a configuration change and save the change, it's automatically activated. This is fine for a development environment, but for a production environment you should always enable the locking feature. In fact, when you run the server in production mode, you don't have the option to set up automatic acquisition of configuration locks and activation of changes. The Automatically Acquire Lock and Activate Changes property is exclusive to servers running in development mode.

   You can enable domain configuration locking by going to the right-hand pane of the Administration Console and clicking Preferences. At the bottom of the page, you'll see the box Automatically Acquire and Activate Changes. You can leave this box checked for a development domain so you can make quick configuration changes on the fly, but it should always be unchecked for a production domain. Clear this option and click Save. Once you click the Release Configuration button in the Change Center, the Lock & Edit button appears, as shown in Figure 1-4.

   Once you enable domain configuration locking, you must use the Lock & Edit button to make any configuration changes, including editing, adding, or deleting any type of configuration attributes. The main purpose behind all this is to ensure that other sessions don't make configuration changes while you're trying to make changes. If you don't click the Lock & Edit button in the Change Center, the server won't even let you start the configuration process—the check boxes for selecting the server or a subcomponent you want to configure will be grayed out. Once you complete any configuration changes and save them, you must click the Activate Changes button to make those changes effective. As you'll see later, some configuration changes require you to restart the server.
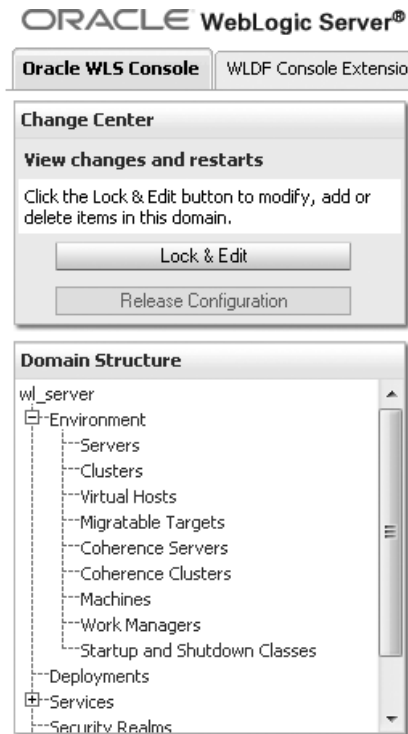
FIGURE 1-4.   *The Change Center in the Administration Console*

To illustrate how to use the Lock & Edit feature, the following example shows you how to disable the Administration Console (something you may want to do to prevent access to the Console in a production environment), which is a configuration change you can make from the Console:

1. Click Lock & Edit, as shown in Figure 1-4. This locks the configuration MBean hierarchy so you can make changes to it.

2. In the Domain Structure pane on the left, click the name of your domain—in the case of the Examples Server, this would be *wl_server*.

3. In the Configuration tab in the right-hand pane of the console, click the General tab and then click Advanced at the bottom of the page. Uncheck the Console Enabled option and click Save. When you click Save, you'll see the following message (in green) on the top of the page where you made the change, confirming that the change was successful:

   ```
   Settings Updated Successfully
   ```

4. Finally, click Activate Changes in the Change Center.

Once you successfully save the configuration changes, you'll see the Activate Changes button in the Change Center, as shown in Figure 1-5. Click the Activate Changes button to make your changes effective.

After you click the Activate Changes button in the Change Center, you'll see the following message (in green) at the top of the right-hand pane:

```
All changes have been activated. However 1 items must be restarted for the
changes to take effect.
```

The reason the message states that "1 items must be restarted" is because disabling the console is a non-dynamic change that requires a server restart.

NOTE
Some configuration changes are dynamic and therefore will go into effect right away; other changes will require a server restart.
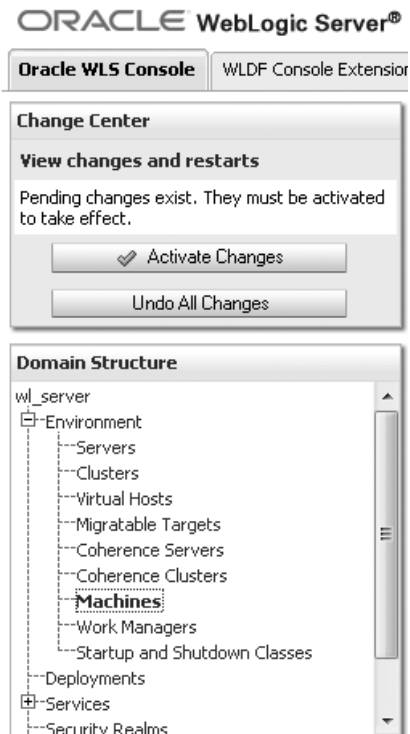

FIGURE 1-5.   *The Activate Changes button in the Change Center*

You'll also see the following in the command console, following the change you just made:

```
<Feb 1, 2011 1:15:16 PM EST> <Warning> <Management> <BEA-141239> <The non-
dynamic attribute ConsoleEnabled on
weblogic.management.configuration.DomainMBeanImpl@
d5bf4c12([wl_server]) has been changed. This may require redeploying or
rebooting configured entities>
<Feb 1, 2011 1:15:16 PM EST> <Warning> <Management> <BEA-141238> <A non-
dynamic change has been made which affects the server examplesServer. This
Server must be rebooted in order to consume this change.>
```

Once you disable the Administration Console, you can reenable it only through the WebLogic Scripting Tool (WLST). Once the Admin Server is started up, invoke WLST by navigating to Start | Program Files | Oracle WebLogic | Tools | WebLogic Scripting Tools and issue the following commands:

```
wls:/offline> connect("weblogic", "welcome1")
wls:/wl_server/serverConfig> edit()
wls:/wl_server/edit !> startEdit()
wls:/wl_server/edit !> cmo.setConsoleEnabled(true)
wls:/wl_server/edit !> save()
wls:/wl_server/edit !> activate()
```

Working with the Administration Console

You already know how to log into the Administration Console. The following sections show how to log out of the console and to set console preferences.

Logging Out of the Console

To log out of the Administration Console, click the Log Out button at the top of the right-hand pane. Logging out of the Administration Console doesn't affect the Admin Server. To log back in, use the URL for the console—http://*<hostname>*:port/console. When you shut down the Admin Server from the Administration Console, the console immediately shuts down and won't be available until you manually restart the Admin Server. Once you restart the Admin Server, you can log back in to the console by using our familiar URL:

```
http://127.0.0.1:7001/console
```

Setting Console Preferences

You can set console preferences by clicking the Preferences button at the top of the right-hand pane. You can select several configuration-related properties from the Preferences page, including whether the server asks for confirmation of operations. You can also choose your preference for whether the server issues a warning message when a user logs out with an active domain configuration lock for a resource in place. Note that when this happens, other users won't be able to lock that resource for making their own configuration changes. The lock holder must either release the configuration changes or activate them first.

Changing the Console URL

You can change the console URL (by default, http://localhost:7001/console) to a different URL. To change the console URL, in the configuration page for the domain, click General and then Advanced at the bottom of the page. Enter the context path in the Console Context Path box. If you specify a new context path named *newconsole*, you can then use the following URL to access the console: http://localhost:7001/newconsole.

Changing the Listen Port and Listen Address

In order to change the listen port or listen address that you use to access the Administration Console, you must change those settings for the domain's Admin Server. You can change the following network-related configuration attributes from the Administration Console. Go to the Admin Server's configuration page and click General. From this page, you can configure the following network-related settings:

- **Listen Address**   IP address or DNS name for the server
- **Listen Port**   Default TCP/IP port for listening for non-SSL connection requests
- **Listen Port Enabled**   Lets you enable or disable the default non-SSL listen port
- **SSL Listen Port**   TCP/IP port on which to listen for secure, SSL connection requests
- **SSL Listen Port Enabled**   If you haven't enabled the optional administration port, both application traffic and administrative traffic will go through the normal Listen Port and the SSL Listen Port. If you've enabled the administration port, then the administrative traffic will go through just the administrative port.

The preceding is a very brief summary of what you can do with the Administration Console. Throughout this book, you'll have plenty of chances to review the many capabilities of the Administration Console, as we discuss topics such as deployment, security, configuration management, and diagnostics.

Node Manager

The Node Manager, as mentioned earlier in this chapter, is a purely optional process (or daemon) that lets you remotely manage both the Admin Server and all Managed Servers within that domain. If you're in a production environment with high availability requirements, Oracle recommends that you use the Node Manager to manage the servers running on different machines. In Chapter 2, you'll find a detailed explanation of how to configure the Node Manager and how to manage servers using WLST and Node Manager together.

Unlike the Admin Server, of which there's only a single instance running per domain, you must run the Node Manager on each of the servers (machines) on which you plan to run the Admin Server or one of the Managed Servers. You don't have to install the Node Manager separately—each installation of WebLogic Server comes with the Node Manager. You just need to start the Node Manager service or process on each of the machines running WebLogic Server instances. Thus, if you have WebLogic Server instances running on five different servers, you must have five Node Manager processes running, one per machine.

Oracle WebLogic Server offers you two types, or versions, rather, of the Node Manager, one a Java-based and the other a script-based version. Although both versions offer the same functions, you need to configure them differently. Also, different security considerations apply to the two

versions, with the Java-based version offering you more security features than the script-based version. You can configure the Java-based Node Manager with the *nodemanager.properties* file, as shown in Chapter 2. The Java-based version allows you to use SSL, and the script-based version offers you the capability to manage servers over an SSH-enabled (or RSH-enabled) network once you copy the scripts to the remote servers.

You can run the Node Manager as a Windows service or an OS daemon so it automatically starts up when you reboot the server. Chapter 2 shows you how to run the Node Manager as an operating system service, post installation. The Configuration Wizard gives you the option to install the Node Manager as an operating system service, which Oracle recommends you do. When you install the WebLogic Server software, choose the Java-based Node Manager if you're working on a Windows or a UNIX platform and wish to run the Node Manager as an operating system process.

NOTE
The Node Manager is not supported on all platforms, so check the Oracle documentation to ensure it's supported on your platform.

The script-based Node Manager uses UNIX-style shell scripts, so you can run it only on UNIX and Linux systems. Oracle recommends that you run this version as an operating system service to enable automatic restarts.

The choice between the Java-based and script-based versions of the Node Manager isn't really hard. Only the Java-based version works on a Windows system, so your choice on that platform is already made for you! Throughout this book, I use a Java-based Node Manager, as the examples are from a Windows environment. As for UNIX/Linux systems, you can use either version, with the script-based version being easier to configure from the security point of view. Other than this, the way the Node Manager interacts with server instances is essentially the same under the two versions.

Surprisingly, as critical as the Node Manager is for managing WebLogic Server, you really don't access the process directly. You access the Node Manager through either the Admin Server or the WLST scripting tool—both act as Node Manager clients. When you use the Admin Server as the client, you do so through the Administration Console. When you are using the Node Manager from the command line, you do so by first invoking WLST and using it as the interface to run Node Manager commands. For the script-based Node Manager, you can use an SSH client to connect to the Node Manager remotely.

You can perform the following functions by connecting with the Node Manager process through WLST:

- You can control the Admin Server by starting, stopping, and restarting the server with the Node Manager.

- The Node Manager can stop and start as well as suspend any Managed Server. When you start or stop a Managed Server through the Administration Console, the Admin Server first accesses the Node Manager, which in turn performs the actual task.

- The Node Manager also monitors the Managed Servers and tries to restart a failed Managed Server.

This chapter provides a very simple introduction to Node Manager and its capabilities. Chapter 2 shows how to work with Node Manager to perform various administrative tasks.

The WebLogic Scripting Tool (WLST)

Most application servers provide you with a good scripting tool. For example, IBM's WebSphere has a scripting tool called *wsadmin* that is based on Jython, and JBoss has a similar scripting tool. Oracle WebLogic Server offers you a wonderful scripting tool called WebLogic Scripting Tool (WLST). WLST is a very powerful tool, capable of performing several different types of administrative tasks for you, including configuration, management, and monitoring of tasks. As you'll see in Chapter 2, you can connect to Node Manager through the WLST interface to manage the server instances. For ease of use, you can use simple Python scripts as wrappers for WLST commands.

You can use WLST in interactive mode by invoking it at the command line. You can also use it in batch mode by putting WLST commands in scripts, and you can embed WLST commands in Java code by importing *weblogic.management.scripting.utils.WLSTinterpreter* into your Java class.

Offline and Online WLST

You can use WLST in online mode by connecting to an active Admin or Managed Server. When connected to the Admin Server, you can use WLST to configure a domain. As is the case with the Administration Console, WLST in online mode acts as a Java Management Extensions (JMX) client that manages the domain's resources by modifying the server's Configuration MBeans. Thus, WLST offers you all the domain management configuration capabilities as the Administration Console.

In offline mode, WLST helps you extend a domain, create domain templates, and create domains with those templates. Since you aren't connected to an active Admin Server, you won't be able to modify domain configuration in offline mode. In offline mode, WLST acts as an interface to the Node Manager and you can issue WLST commands to start and stop Managed Server instances without connecting to the Admin Server. Note that you can't start and stop Managed Servers through WLST without the Node Manager, as explained in Chapter 2.

Invoking WLST

In a Windows environment, you can invoke WLST through the Windows Start program and from the command line. The following sections show how to invoke WLST.

Starting WLST from the Start Program

You can invoke WLST in a Windows environment by simply going to Start | All Programs | Oracle WebLogic | WebLogic Server 11gR1 | Tools | WebLogic Scripting Tool.

Invoking WLST from the Command Line

You can invoke WLST from the command line by using either the java *weblogic.WLST* command, or through the command script *wlst.cmd*. The following shows how to invoke WLST with the Java command *weblogic.WLST*.

```
C:\MyOra\Middleware\wlserver_10.3\samples\domains\medrec\bin>setDomainEnv.cmd
C:\MyOra\Middleware\wlserver_10.3\samples\domains\medrec>java weblogic.WLST
Initializing WebLogic Scripting Tool (WLST) ...
Welcome to WebLogic Server Administration Scripting Shell
Type help() for help on available commands
wls:/offline>
```

Note that when you invoke WLST, by default you're in offline mode. You can only issue certain commands in offline mode, such as those that create a new domain or domain template, for example. In offline mode, you can't view performance data pertaining to any domain resource or add and remove users. To issue any online commands, you must first connect to the Admin Server using the *connect* command. Once you use WLST to connect to an Admin Server, you can manage the configuration of the domain and view performance data. While you can connect to a Managed Server through WLST, you can't modify the configuration for a Managed Server.

You can also invoke WLST by issuing the script *wlst.cmd* (*wlst.sh* in UNIX), as shown here:

```
C:\NewOra\Middleware\wlserver_10.3\common\bin>wlst.cmd
Your environment has been set.
Initializing WebLogic Scripting Tool (WLST) ...
Welcome to WebLogic Server Administration Scripting Shell
Type help() for help on available commands
wls:/offline>
```

Using WLST in Script Mode

While you can use WLST in interactive mode to quickly make configuration changes in a development environment, that mode offers only limited scripting language features and is cumbersome to use in a real-life WebLogic environment. You can use WLST scripts to automate configuration of the servers and deployment of applications. A WLST script is a text file with the *.py* extension, and includes WLST commands. WebLogic Server provides online and offline sample WLST scripts. For example, the sample online script *cluster_creation.py* lets you create an entire cluster with ten Managed Servers. Similarly, the basic *WLSdomain.py* script lets you create a simple WebLogic domain for development purposes. You'll find the online scripts in the WL_HOME\samples\server\examples\src\examples\wlst\online directory. The offline scripts are in the WL_HOME\common\templates\scripts\wlst directory.

You can invoke a WLST script (*.py*) by providing the name of the script as an argument to the java *weblogic.WLST* command, as shown here:

```
C:\MyOra\Middleware\wlserver_10.3\samples\domains\medrec> java weblogic.WLST
C:\MyOra\Middleware\wlserver_10.3\samples\domains\medrec\shutdown.py
```

Here are the contents of the *shutdown.py* script:

```
import os
if os.environ.has_key('wlsUserID'):
    wlsUserID = os.environ['wlsUserID']
if os.environ.has_key('wlsPassword'):
    wlsPassword = os.environ['wlsPassword']
connect(  url='t3://MIROPC61:7011', adminServerName='MedRecServer')
shutdown('MedRecServer','Server', ignoreSessions='true')
exit()
```

Alternatively, you can first invoke WLST and use the *execfile* command to execute the *shutdown.py* script.

```
wls:offline>
    execfile('C:\MyOra\Middleware\wlserver_10.3\samples\domains\medrec\shutdown.py')
```

If you're embedding WLST commands in a shell script or a Windows command script, invoke WLST with the *wlst.cmd* script (WL_HOME\common\bin\wlst.cmd). This will ensure that all the environment variables are set correctly. WebLogic Server also allows you to write all the WLST commands you enter during an interactive session to a file that you can later run as a WLST script. Simply issue the *startRecording* command to record all your interactive commands and issue the *stopRecording* command to stop the capturing of the commands, as shown here:

```
wls:/mydomain/serverConfig> startRecording('C:/MyOra/test.py')
Started recording to C:\MyOra\test.py
```

Issue the WLST commands you want to capture in the test.py. Once you're done, stop the recording of the commands by issuing the *stopRecording* command:

```
wls:/mydomain/serverConfig> stopRecording()
Stopped recording to C:/MyOra/test.py
wls:/mydomain/serverConfig>
```

You can edit the test.py file and execute it as a WLST script.

Connecting to a WebLogic Server Instance

In the offline mode, you aren't connected to a running server. Use the *connect* command to connect to the Admin Server, as shown here:

```
wls:/offline> connect()
Please enter your username :weblogic
Please enter your password :
Please enter your server URL [t3://localhost:7001] :
Connecting to t3://localhost:7001 with userid weblogic...
Successfully connected to Admin Server 'examplesServer' that belongs to
domain 'wl_server'.
Warning: An insecure protocol was used to connect to the
server. To ensure on-the-wire security, the SSL port or
Admin port should be used instead.
wls:/wl_server/serverConfig>
```

In the example, you'll notice a warning because I am not using a secure port such as the administration port or an SSL port. Oracle recommends that you use either SSL or the administration port in a production system. You can ignore this warning in a development environment.

TIP
To view the help topics, type **help** *at the WLST command line. You must specify an argument for the* help *command—for example,* help(connect) *will show information about using the* connect *command.*

You can also directly specify the administrator's credentials at the command line, as shown here:

```
wls:/offline> connect('weblogic','welcome1','t3://localhost:7001')
Connecting to WebLogic Server instance running at t3://localhost:7001 as
```

```
username weblogic...
Successfully connected to Admin Server 'ExamplesServer' that belongs to
domain 'examples'.
wls:/mydomain/serverConfig>
```

As you can see, I had to supply the user credentials (the same ones you use for the Administration Console) to connect to the Admin Server. Oracle recommends that you do this only when using WLST in the interactive mode. The default behavior is for WLST to see if you have created a "user configuration file" to store the encrypted credentials and a "key file" with which the server can unencrypt the credentials. If you start WLST from the domain directory from which you started the Admin Server, it can use the boot.*properties* file to get the encrypted credentials. (The *boot. properties* file is discussed in Chapter 2.)

When you use WLST in scripts, it's safer not to use the clear text credentials in the script. You can use the *storeUserConfig* command to store the credentials in an encrypted form, following which you can specify the name of the user configuration file instead of the credentials. Here's how to do this:

```
wls:/offline> connect(userConfigFile='C:\MyOra\myuserconfigfile.secure',
userKeyFile='C:\MyOra\myuserkeyfile.secure')
Connecting to t3://localhost:7001 with userid username ...
Successfully connected to Admin Server 'AdminServer' that belongs to domain
'examples'.
wls:/examples/serverConfig>
```

In order to use the *userConfigFile* option, you must first use the *storeUserConfig* command to create a user configuration file and its key file. The configuration file will contain the encrypted credentials and the key file contains the key the server uses for encryption and decryption of the credentials. Here's an example that shows how to do this:

```
wls:/testDomain/serverConfig>
storeUserConfig('C:\MyOra\myuserconfigfile.secure',
'C:\MyOra\myuserkeyfile.secure')
Creating the key file can reduce the security of your system if it is not
kept in a secured location after it is created. Do you want to create the
key file? y or n   y
Please confirm user config key creation: y or n   y
The username and password that were used for this current WLS connection are
stored in C:/MyOra/mysuserconfigfile.secure and
C:/MyOra/myuserkeyfile.secure
wls:/testDomain/serverConfrg>
```

Once you generate the user configuration file and the key file, you can supply the names of these two files instead of entering administrator credentials on the command line.

Disconnecting from the Server

You disconnect from a server by issuing the *disconnect* command, as shown here.

```
wls:/wl_server/serverConfig> disconnect()
Disconnected from WebLogic Server: examplesServer
wls:/offline>
```

To exit from WLST, use the *exit* command:

```
wls:/offline> exit()
Exiting WebLogic Scripting Tool.
C:\MyOra\Middleware\wlserver_10.3\samples\domains\medrec >
```

By default, the server outputs all WLST messages or output to standard output, that is, to the screen. You can redirect all the messages to any file you wish by using the *redirect* command:

```
wls:/wl_server/serverConfig> redirect ('C:\MyOra\wlst.log')
```

Using the Help Command

WLST has numerous commands that you can use in your daily work. You can check out these commands and their syntax using the help facility. Here's a listing of all the help facility commands.

```
wls:/wl_server/serverConfig> help()
WLST is a command line scripting tool to configure and administer WebLogic Server. Try:
     help('all')              List all WLST commands available.
     help('browse')           List commands for browsing the hierarchy.
     help('common')           List the most commonly used commands.
     help('control')          List commands for controlling the domain/server.
     help('deployment')       List commands for deploying applications.
     help('diagnostics')      List commands for performing diagnostics.
     help('editing')          List commands for editing the configuration.
     help('information')      List commands for displaying information.
     help('lifecycle')        List commands for managing life cycle.
     help('nodemanager')      List commands for using Node Manager.
     help('offline')          List all offline commands available.
     help('online')           List all online commands available.
     help('storeadmin')       List all store admin commands.
     help('trees')            List commands use to navigate MBean hierarchy.
     help('variables')        List all global variables available.
```

Key WLST Command Groups

As I mentioned earlier, WLST offers a large number of commands to help perform various management and programming tasks. Here's a brief description of the key WLST command types. Note that you can execute some commands only in offline mode and others in online mode.

Life Cycle Commands

You can use the life cycle commands to manage the life cycle of both the Admin and the Managed Servers. WLST offers the *start*, *startServer*, *suspend*, *resume*, and *migrate* commands to control a server life cycle. Here are a couple of examples showing how to suspend and resume the Admin Server instance:

```
wls:/wl_server/serverConfig> suspend('examplesServer')
..Server examplesServer suspended successfully.
wls:/wl_server/serverConfig> resume('examplesServer')
Server examplesServer resumed successfully.
wls:/wl_server/serverConfig>
```

Node Manager Commands

You can use the Node Manager commands to start, stop, and monitor server instances. Before you can use Node Manager to manage server instances, you must connect WLST to the Node Manager using the *nmConnect* command. The *nmStart* command lets you start a server instance with the help of the Node Manager. Here's how you use the *nmConnect* command to connect to the Node Manager from WLST. First, make sure that the Node Manager is running—if not, you can start it from the Windows Start command.

```
wls:/myserver/serverConfig> nmConnect('weblogic', 'welcome1', 'localhost',
'7011', 'medrec',
'C:\MyOra\Middleware\user_projects\domains\medrec','ssl')
Connecting to Node Manager Server ...
Successfully connected to Node Manager.
```

Chapter 2 explains other important Node Manager–related commands such as *nmDisconnect*, *nmEntroll*, and *nmkill*.

Deployment Commands

Deployment commands such as *deploy*, *undeploy*, *startApplication*, and *stopApplication* enable you to deploy, undeploy and redeploy applications, update deployment plans, as well as start and stop applications. Here's how you execute the *deploy* command:

```
wls:/testdomain/serverConfig/Servers> deploy('myApp',
'C:\MyOra\myApps\demos\app\myApp.ear',targets='ManagedServer1',
planPath='C:\MyOra\myApps\demos\app\plan\plan.xml',timeout=120000)'
```

In this example, the myApp application is packaged in the form of a Java EAR file, *myApp.ear*. The server targets this application to the Managed Server named ManagedServer1 using the deployment file C:\MyOra\myApps\demos\app\plan\plan.xml. The server will wait for 120,000 ms for the deployment to finish.

Editing Commands

You can use commands such as *get*, *set*, *edit*, *startEdit*, *stopEdit*, *save*, and *activate* to view and edit the MBean domain configuration hierarchy. You can edit and modify a domain's configuration in both offline and online modes. Oracle recommends that you change only the Admin Server's domain configuration MBeans, and not those of the Managed Servers, to avoid ending up with an inconsistent configuration. As you may recall, domain configuration changes are synchronized between the Admin Server and the Managed Server. You can, however, view the hierarchy for the Managed Server MBeans. Note that you must connect to the Admin Server before editing any of the configuration beans. Here's a simple example that shows how to use the *startEdit*, *stopEdit*, and *activate* commands.

```
wls:/wl_server/edit> startEdit(30000, 60000)
Starting an edit session ...
Started edit session, please be sure to save and activate your changes once
you are done.
wls:/wl_server/edit !> stopEdit()
Sure you would like to stop your edit session? (y/n)
y
```

```
Edit session has been stopped successfully.
wls:/wl_server/edit !> activate(200000, block='true')
Activating all your changes, this may take a while ...
the edit lock associated with this edit session is released once the
activation is completed.
Action completed.
wls:/wl_server/edit>
```

Diagnostic Commands

Diagnostic commands such as *exportDiagnosticData* and *getAvailableCapturedImages* help you work with diagnostic data stored in the WebLogic Diagnostic Framework (WLDF) data stores. Chapter 6 shows how to use key WLST diagnostic commands.

Summary

This chapter introduced you to key WebLogic Server concepts and terminology. You learned how to install WebLogic Server, as well as how to upgrade it. The chapter also introduced you to the key WebLogic Server administrative tools such as the Administration Console, Node Manager and WLST. WLST is a very powerful tool, capable of assisting with a wide variety of administrative tasks. I've attempted merely to introduce you to the WLST interface in this chapter. Chapter 2 shows you how to use WLST to manage a server's life cycle. Similarly, other chapters show how you can effectively use the many powerful, yet easy-to-use WLST commands to perform other types of management tasks.

Now that you've a basic understanding of WebLogic Server, let's learn how to use WLST and Node Manager commands together to manage servers in the next chapter. The chapter also explains the various server run states and how to manage them.