CHAPTER 1

Getting Started

Apache Geronimo is an open source Java 2 Platform, Enterprise Edition (J2EE) application server from the Apache Software Foundation. It is a framework that integrates a large base of open source projects to provide a J2EE 1.4–compliant, production-quality container for J2EE applications. With Geronimo, you can deploy and run your web, Enterprise JavaBeans (EJB), and enterprise applications. You can create your enterprise applications by using servlets, JavaServer Pages (JSPs), and EJBs; access your database by using Java Database Connectivity (JDBC) connectors; access the directory services through Java Naming and Directory Interface (JNDI); and so on.

At its core, Geronimo is more than a J2EE server; it provides an Inversion of Control (IOC) platform. Many other applications can also make use of this platform to build their services and servers. As a platform, Geronimo provides a modular infrastructure for building any system by implementing the common basic facilities, including service life cycle and service modules (configurations).

Although many open source J2EE servers already exist, Geronimo is unique because of its Apache license. The Apache license is the most liberal of all the open source licenses. You can use Geronimo as it is or even modify and use it in your commercial and noncommercial applications. The only requirement from the user is an acknowledgement of the source.

In Chapter 1, I'll summarize Geronimo's history and present an overview of its features. I'll then explain how to install Geronimo. Last, I'll demonstrate how to start and stop the server.

History

Geronimo started as an Apache Incubator project in the summer of 2003, and by May 2004, it had become a top-level Apache project. In June 2004, the Geronimo Milestone 4 release passed the automated J2EE Technology Compatibility Kit (TCK) 1.4.1a test suite. The Milestone 4 release had many significant advantages over the Milestone 3 release, including full support for EJB, Java Specification Request (JSR) 77, and web services.

In October 2005, Apache released Milestone 5, which is a fully certified J2EE 1.4 application server. This milestone contains a number of enhancements, including a complete Tomcat integration, a more flexible and dynamic service configuration, and a developer preview of a portlets-based management console. In January 2006, Apache announced the Geronimo 1.0 release.

A good number of Geronimo committers were from a company called Gluecode, which built a Geronimo-based portal product, JOE. Recently, IBM took over this company and has since been a significant contributor to the Apache Geronimo project. IBM has also released WebSphere Application Server Community Edition, which is based on the core Geronimo technology.

Features

The goal in creating Geronimo was to provide the best open source, fully certified application server, a J2EE platform for high-end production use, and a high-performance J2EE server. To meet this goal, Geronimo's designers focused on the following aspects:

- Manageability of all services and components
- Flexible configuration capabilities
- Full support for J2EE specifications through integration of open source components
- Custom-developed components that meet Geronimo's stringent performance goals

Geronimo's features match those of any other commercial application server on the market. The sections that follow describe Geronimo's key features.

IOC Container As the Core Geronimo Platform

Geronimo is based on an IOC framework for service components called GBeans. In addition, Geronimo integrates with other popular IOC containers, like Spring.

GBean-Based Modular Services

One or more GBeans implement a specific service, like a web container, an EJB container, or the security infrastructure. Most GBean-based services are either wrappers over an open source component or custom implementations. For instance, the GBeans that provide the web container functionality are wrappers over a Jetty or a Tomcat Hypertext Transfer Protocol (HTTP) server. In short, the Geronimo J2EE server is a set of GBeans deployed and run in the Geronimo platform that implement and provide the J2EE services as required by the J2EE specification. Even J2EE applications are also deployed and run as GBeans.

JSR 77 Support

Geronimo implements JSR 77, which is the J2EE management specification. JSR 77 is based on the Java Management Extension (JMX) specification and defines a standard model for J2EE-managed objects. It abstracts the manageable parts of a J2EE server (including deployed J2EE applications) and defines a standard interface for clients and management consoles to access management information. Geronimo implements the JSR 77 management EJB and exposes all its deployed GBean services to JSR 77 clients. Geronimo's JMX implementation utilizes the open source MX4J product.

JSR 88 Support

Geronimo also supports JSR 88, a standard application programming interface (API) that enables tools to interact with any application server to configure and deploy J2EE applications. Application configuration involves creating the required deployment descriptors, including standard and application server–specific deployment descriptors for J2EE components. Many tools use a graphical user interface (GUI) to collect relevant information from the user, generate required deployment descriptors, and package the component ready for deployment. Application deployment involves application server interactions to deploy, undeploy, start, and stop applications.

J2EE Application Containers

Geronimo implements containers for all the J2EE application module types: web applications, EJB applications, J2EE Application Clients, and Connectors.

Web Applications

Geronimo 1.0 offers one of two open source web containers—Jetty or Apache Tomcat—that can host and run web applications packaged as Web Archive (WAR) files. The web container supports Servlets 2.4 and the JSP 2.0 specification. The Geronimo Milestone 5 release includes both web containers. Jetty is the default web container, but you can manually reconfigure Milestone 5 to use Tomcat.

EJB and Enterprise Applications

Geronimo uses the OpenEJB container to host and run EJB applications. It supports session, entity, and message-driven beans. It also supports advanced J2EE features like timers and web services. Currently, it implements the EJB 2.1 specification.

You can deploy EJBs as a standard EJB Java Archive (JAR) file. You can also package your enterprise application consisting of one or more web applications (packaged as WAR files), one or more EJB applications (packaged as EJB JAR files), and one or more connectors (packaged as Resource Archive [RAR] files) into Enterprise Archive (EAR) files and deploy these elements as a single unit.

J2EE Client Applications

Geronimo has a client container for J2EE application clients. Geronimo uses custom code to implement and provide the client container. Client applications can access and use all server resources available through JNDI.

J2EE Connectors

Geronimo has a J2EE Connector Architecture (J2CA) container that can host and run J2EE connectors. Connectors are J2EE application components that run within the application server environment and provide Enterprise Information System (EIS) connectivity to other J2EE application components, like EJB. Geronimo uses custom code to implement the Java Connector Architecture (JCA) container and supports both inbound and outbound connectors. Inbound connectors pass information from the EIS to the application server, and outbound connectors pass information from the application server to the EIS.

Web Services

Geronimo uses Apache Axis and custom code to support web services. It implements the enterprise web services specification (JSR 109) and can export Plain Old Java Object (POJO)–based and EJB-based service endpoints as web services.

Java Message Service (JMS) and Messaging Services

Geronimo uses the ActiveMQ JMS provider to enable JMS applications. You can deploy JMS destinations (queues and topics) as part of a server configuration or as part of an application configuration. Serverwide JMS destinations are available to all applications on the server, and application-wide JMS destinations are available to a specific application only.

ActiveMQ is an open source messaging server, released under the Apache license, that implements the JMS 1.1, J2EE 1.4, JCA 1.5, and XA standards. It supports many different transport mechanisms, including HTTP, Transmission Control Protocol (TCP), and User Datagram Protocol (UDP).

J2EE Security

The Geronimo security infrastructure is based on the Java Authentication and Authorization Service (JAAS) framework and the Java Authorization Contract for Containers (JACC) specification. JAAS defines a mechanism for supporting pluggable login modules for authentication and a standard mechanism for authorization. The JACC specification defines a standard contract between the security policy provider implementations and the J2EE containers. Geronimo comes with a generic security realm implementation and a variety of login modules, including file-based and database-based implementations.

Transactions

Geronimo uses ObjectWeb's Java Open Transaction Manager (JOTM) for providing its transaction support. JOTM is a transaction manager that allows resource managers to participate in global distributed transactions by coordinating the transactions and implementing the two-phase commit protocol. Geronimo uses High-Speed ObjectWeb Logger (HOWL) for transaction logging and transaction recovery.

JDBC Connection Pools

You can configure a JDBC connection pool by deploying a connector that provides database connectivity and connection pools. Geronimo provides TranQL, the open source product that has a J2EE connector that allows you to configure and deploy a connection pool.

JavaMail

Geronimo provides JavaMail 1.3.3 capability, allowing applications to send and receive e-mails. Geronimo provides GBeans that you can use to configure a JavaMail session and Simple Mail Transport Protocol (SMTP) transport.

Apache Active Directory Support

Geronimo includes Apache Directory, which allows applications to use active directory services. This enables applications to implement a unified security layer for applications.

Specification Support

Geronimo offers all the features of a typical J2EE 1.4 application server, including support for all specifications that J2EE 1.4 requires. These specifications are as follows:

- Servlet 2.4
- JSP 2.0
- EJB 2.1
- JMS 1.1
- JTA 1.0.1B (Java Transaction API)
- JTS 1.0 (Java Transaction Service)
- JMX 1.2
- J2EE Management API 1.0
- J2EE Deployment API 1.1
- JCA 1.5
- JAXR 1.0 (Java API for XML Registries)
- JAX-RPC 1.1 (Java API for XML RPC)
- SAAJ 1.2 (SOAP with Attachment API for Java)
- JACC 1.0
- JavaMail 1.2

Figure 1-1 depicts Geronimo and the components that provide a J2EE 1.4–certified application server.

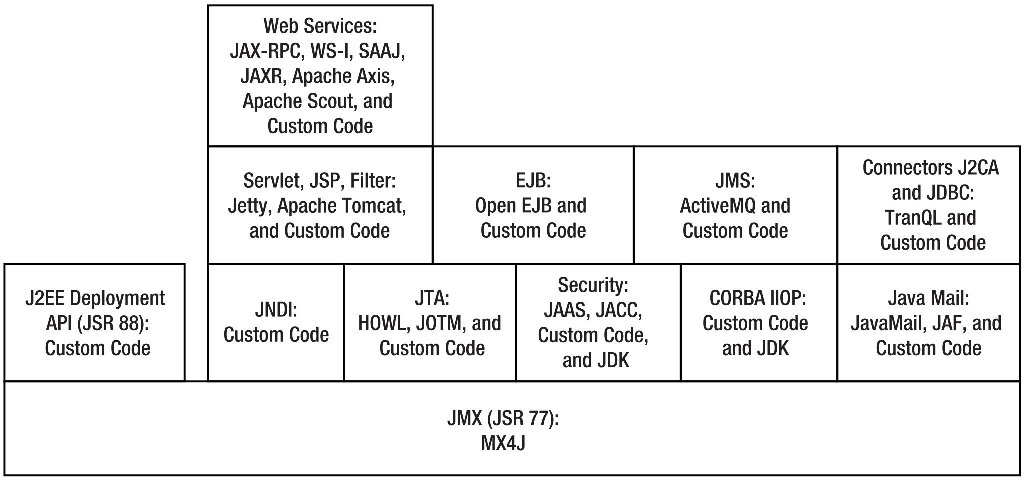

Web Services: JAX-RPC, WS-I, SAAJ, JAXR, Apache Axis, Apache Scout, and Custom Code
Servlet, JSP, Filter: Jetty, Apache Tomcat, and Custom Code
EJB: Open EJB and Custom Code
JMS: ActiveMQ and Custom Code
Connectors J2CA and JDBC: TranQL and Custom Code
J2EE Deployment API (JSR 88): Custom Code
JNDI: Custom Code
JTA: HOWL, JOTM, and Custom Code
Security: JAAS, JACC, Custom Code, and JDK
CORBA IIOP: Custom Code and JDK
Java Mail: JavaMail, JAF, and Custom Code
JMX (JSR 77): MX4J

Figure 1-1. *Geronimo application server components*

Web-Based Administration Console

Geronimo has a web-based administration console application that you can use to administer the Geronimo server. You can also use this application to deploy, undeploy, and manage your own applications.

Other Features

Other useful Geronimo features include integration with the open source ServiceMix Enterprise Service Bus (ESB) and Eclipse tools and plug-ins for integrated development environment (IDE) integration. Geronimo supports the Java Business Integration (JBI) specification by using the ServiceMix ESB provider, and its deployer can deploy service components defined in the META-INF/jbi.xml file of any JAR file.

New Features in Geronimo 1.0

Some of the key features in Geronimo's 1.0 release are as follows:

- Hot deploy and undeploy of applications and services.
- A dynamic deploy directory for applications and services. You can deploy applications by copying their deployment artifacts to the Geronimo root/deploy directory.
- Remote deployment and management capabilities.
- Clustering support with Tomcat.
- Improved web-based administration console.
- Many configuration parameters that can be updated in the var/config/config.xml file.
- Improved documentation with samples.

GERONIMO'S TO-DO LIST

Here are some key features that Geronimo should offer in the next version:

- Provide tooling support for all popular Java IDEs
- Provide full clustering support
- Completely implement the JSR 88 specification
- Include a J2EE application client container that can work from a machine other than the server

Installing Geronimo

Geronimo is a pure Java application server; hence, it can run on any Java Development Kit (JDK) 1.4 platform. However, for all features to run with the 1.0 release, Geronimo requires Sun Java Virtual Machine (JVM) 1.4.2 or higher. This is primarily because Geronimo uses Sun's Common Object Request Broker Architecture (CORBA) implementation classes for its CORBA support—this limitation will likely be rectified with a later release, when Geronimo uses a bundled CORBA implementation.

Obtaining Geronimo

You can download Geronimo from `http://geronimo.apache.org/downloads.html`. There are three types of downloads available for each release: binary, installer, and source code. The binary release is a ZIP file or a TAR.GZ file, and the installer is a JAR file. You can use Subversion to check out Geronimo source code, as shown here:

```
svn co https://svn.apache.org/repos/asf/geronimo/trunk geronimo
```

This will create a directory called geronimo that contains the source code. You need Apache Maven to execute various Maven build commands to download dependencies and build Geronimo. This process is documented in the Geronimo wiki (`http://wiki.apache.org/geronimo/Building`).

Installing Geronimo Quickly

You can use the binary package for a quick installation. As mentioned, the binary package is a ZIP file (or a TAR.GZ file), and you simply unpack the ZIP file into a directory to complete the installation. (There are separate packages available for Tomcat and Jetty web containers, and you can select either one.) With this type of installation, you get Geronimo preconfigured with default settings. If you need to customize the various settings, including ports for services, you need to use the installer-based installation.

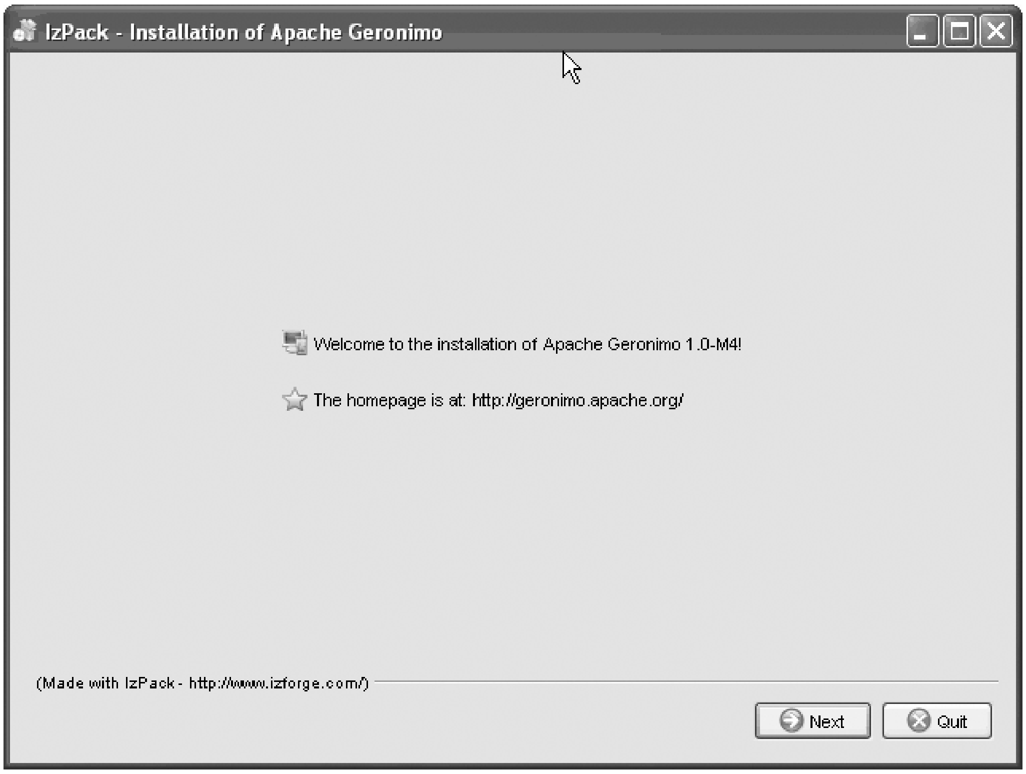
Installing Using the Installer

As of this writing, the Geronimo 1.0 final release does not have an installer download, and hence we will use the Geronimo 1.0 Milestone 4 release installer to explore the installation process. With future releases, the installer options are expected to change; however, the core installation process will remain the same.

The installer application is a JAR file. It guides you through an installation wizard that collects server configuration information and then installs and configures Geronimo for immediate use. To run the installer, use the following command:

```
java -jar geronimo-1.0-M4-installer.jar
```

A welcome screen appears, as shown in Figure 1-2.


IzPack - Installation of Apache Geronimo

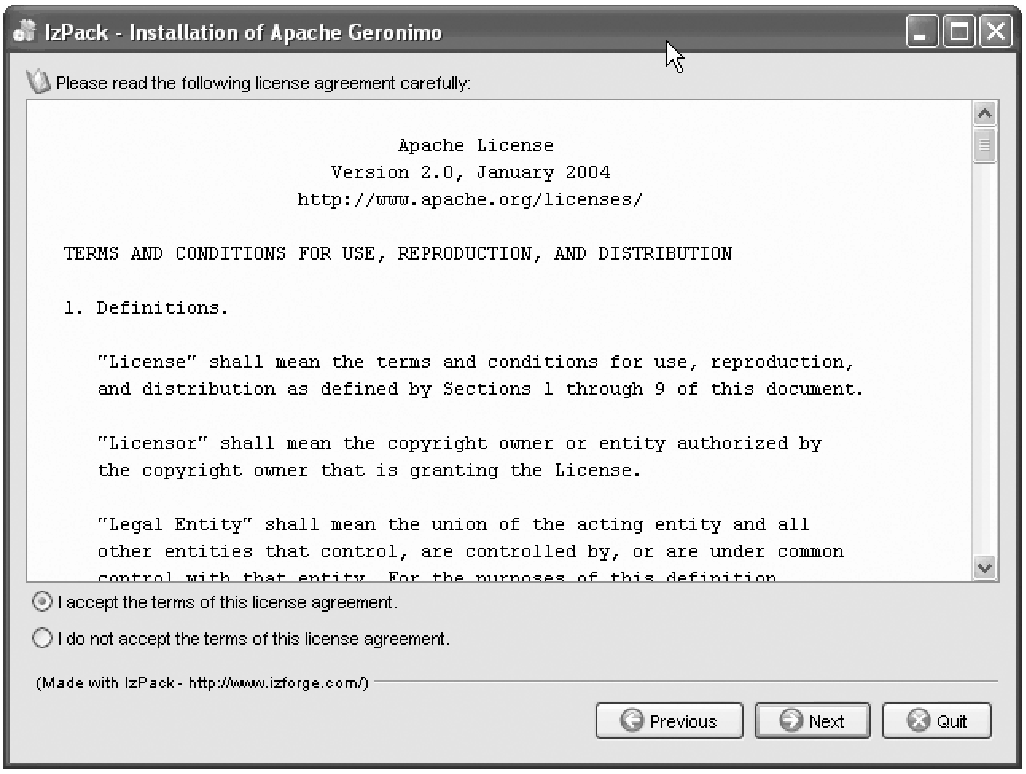
Welcome to the installation of Apache Geronimo 1.0-M4!

The homepage is at: http://geronimo.apache.org/

(Made with IzPack - http://www.izforge.com/)

Figure 1-2. *Welcome screen*

Click the Next button. You will see the license screen, shown in Figure 1-3.

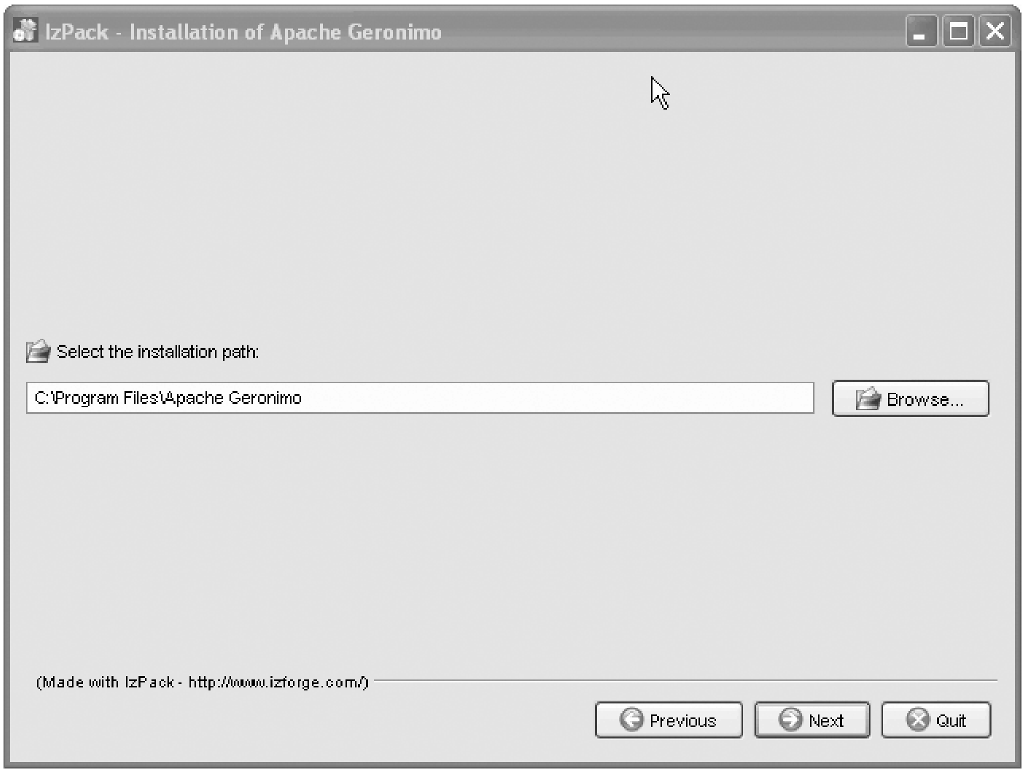

IzPack - Installation of Apache Geronimo
Please read the following license agreement carefully:

```
                          Apache License
                    Version 2.0, January 2004
                 http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction,
   and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by
   the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all
   other entities that control, are controlled by, or are under common
   control with that entity. For the purposes of this definition
```

[✓ I accept the terms of this license agreement.]
I do not accept the terms of this license agreement.

Figure 1-3. *License screen*

Accept the license, and click Next. In the next screen, choose an installation folder in which to install Geronimo, as shown in Figure 1-4. This folder will be the Geronimo root folder.


IzPack - Installation of Apache Geronimo

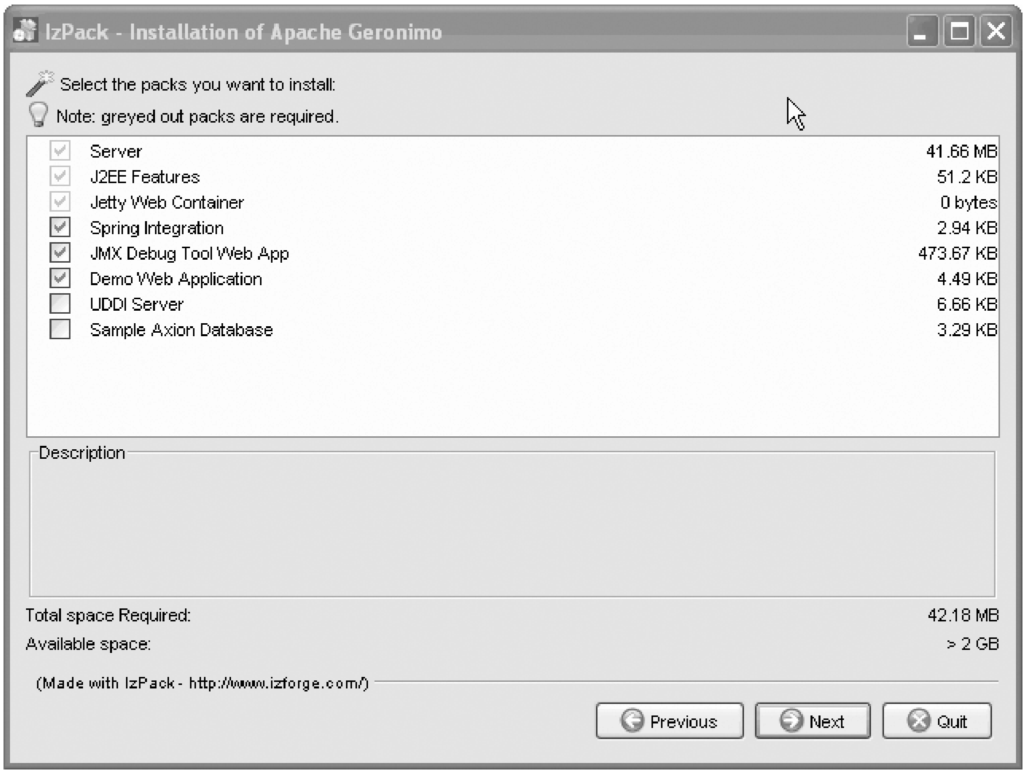
Select the installation path:

C:\Program Files\Apache Geronimo

Browse...
(Made with IzPack - http://www.izforge.com/)
Previous
Next
Quit

Figure 1-4. *Select installation path screen*

If the target directory does not exist, the installer indicates that it will create the directory. Click Next, and the installer displays the feature list, shown in Figure 1-5, from which you can select the optional features that you want to include in the installation. The core server, the J2EE features, and the web container are mandatory features. This feature list is expected to change in future releases.



Select the packs you want to install:

Note: greyed out packs are required.

✓	Server	41.66 MB
✓	J2EE Features	51.2 KB
✓	Jetty Web Container	0 bytes
✓	Spring Integration	2.94 KB
✓	JMX Debug Tool Web App	473.67 KB
✓	Demo Web Application	4.49 KB
☐	UDDI Server	6.66 KB
☐	Sample Axion Database	3.29 KB

Description

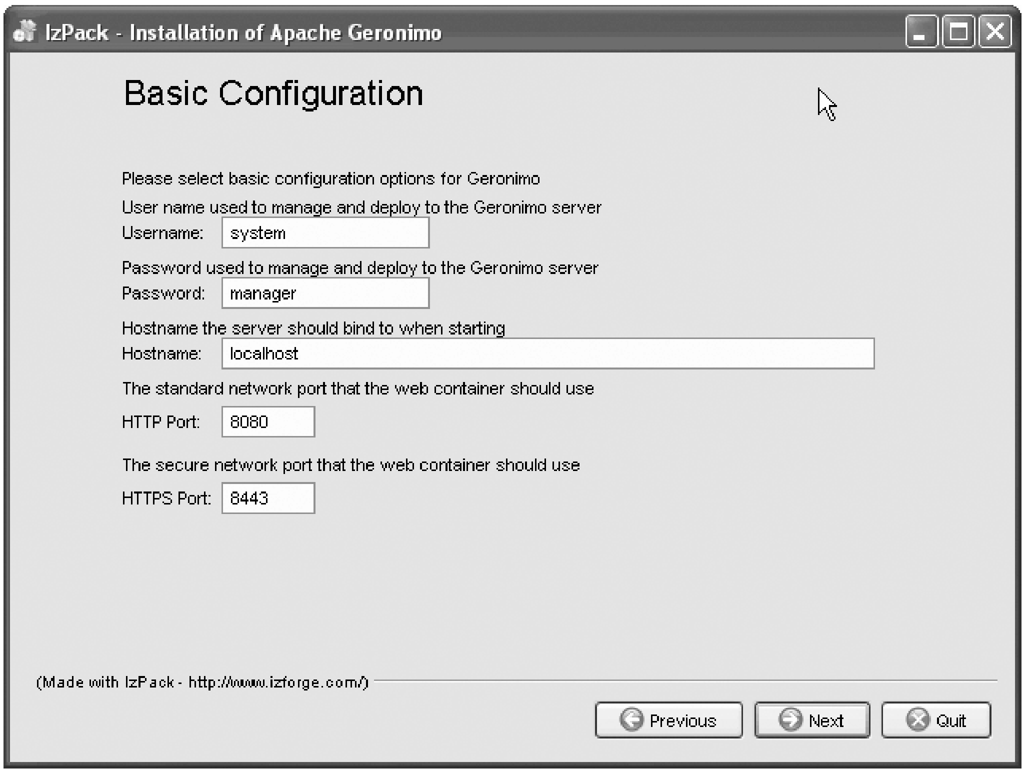
Total space Required: 42.18 MB

Available space: > 2 GB

(Made with IzPack - http://www.izforge.com/)

Figure 1-5. *Feature selection screen*

Click Next, and you will be prompted to enter a system user name and a password, as shown in Figure 1-6. You need this user name and password to use the deployer and the Geronimo administration web tool. You will also be prompted to enter port numbers for various services, like HTTP and JNDI. The web container will use the HTTP port number you specify here to create a listener (connector) for handling HTTP requests, and it will use the HTTPS port number to create another listener for handling secure HTTP (HTTPS) connections. You can choose to use the default values if you do not want to override these values.

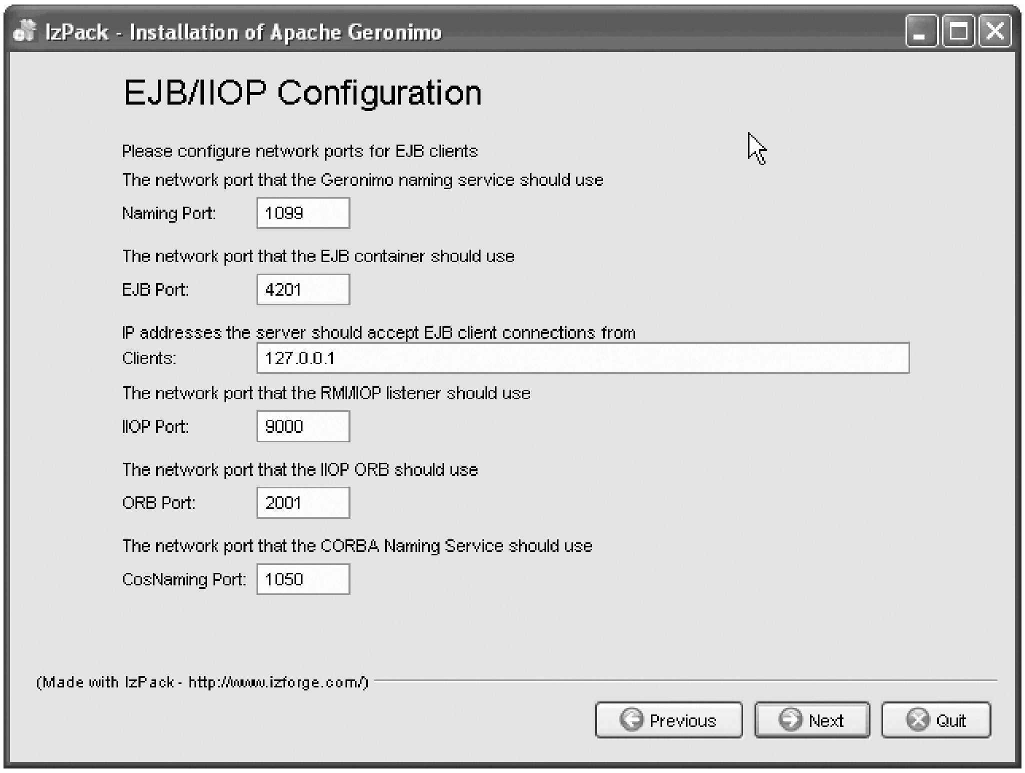


Please select basic configuration options for Geronimo
User name used to manage and deploy to the Geronimo server
Username: system
Password used to manage and deploy to the Geronimo server
Password: manager
Hostname the server should bind to when starting
Hostname: localhost
The standard network port that the web container should use
HTTP Port: 8080
The secure network port that the web container should use
HTTPS Port: 8443

Figure 1-6. *Basic configuration screen*

Click Next to go to the advanced configuration screen, shown in Figure 1-7. Here, you need to enter configuration information for the following items:

- A naming port that Geronimo naming service should use to listen for JNDI connections
- A network port that the EJB container should use
- A list of client Internet Protocol (IP) addresses from which the EJB container should accept EJB client connections
- A network port for Remote Method Invocation over Internet Inter-Orb Protocol (RMI/IIOP) connections
- A network port that the IIOP Object Request Broker (ORB) should use
- A network port that the CORBA naming service should use


IzPack - Installation of Apache Geronimo
(Made with IzPack - http://www.izforge.com/)
Previous
Next
Quit

EJB/IIOP Configuration

Please configure network ports for EJB clients

The network port that the Geronimo naming service should use

Naming Port: 1099

The network port that the EJB container should use

EJB Port: 4201

IP addresses the server should accept EJB client connections from

Clients: 127.0.0.1

The network port that the RMI/IIOP listener should use

IIOP Port: 9000

The network port that the IIOP ORB should use

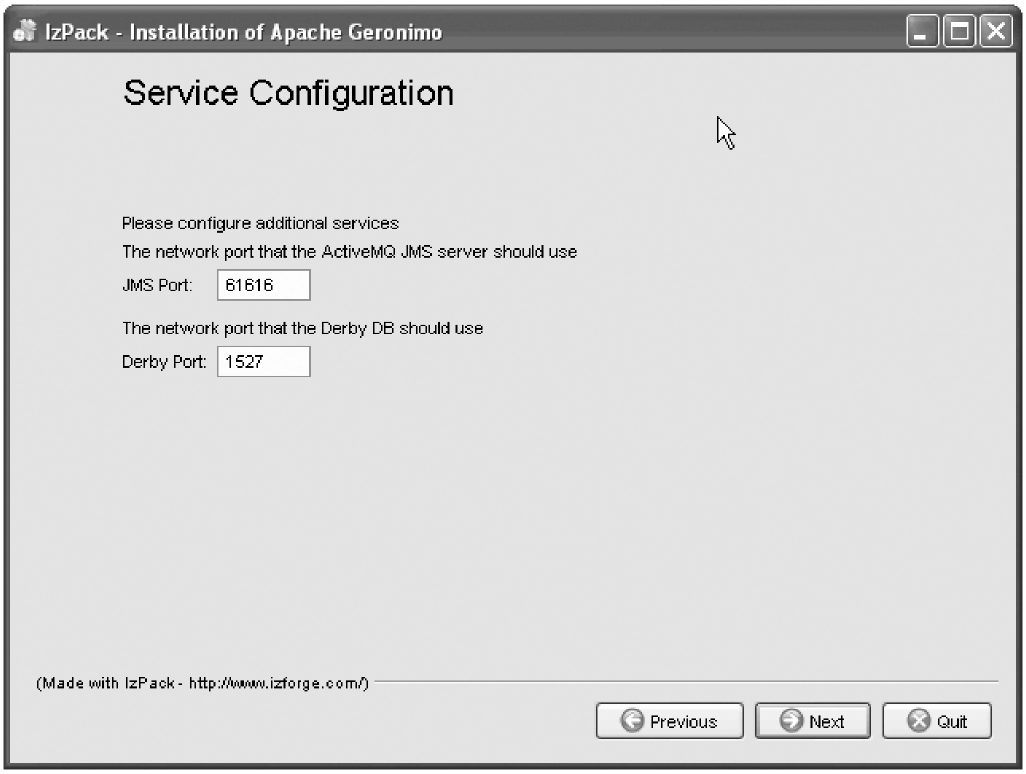
ORB Port: 2001

The network port that the CORBA Naming Service should use

CosNaming Port: 1050

Figure 1-7. *Advanced configuration screen*

Click Next to advance to the final configuration screen, shown in Figure 1-8. Here, you need to provide configuration information for additional services, like JMS and Derby database.


IzPack - Installation of Apache Geronimo
Service Configuration

Please configure additional services

The network port that the ActiveMQ JMS server should use

JMS Port: 61616

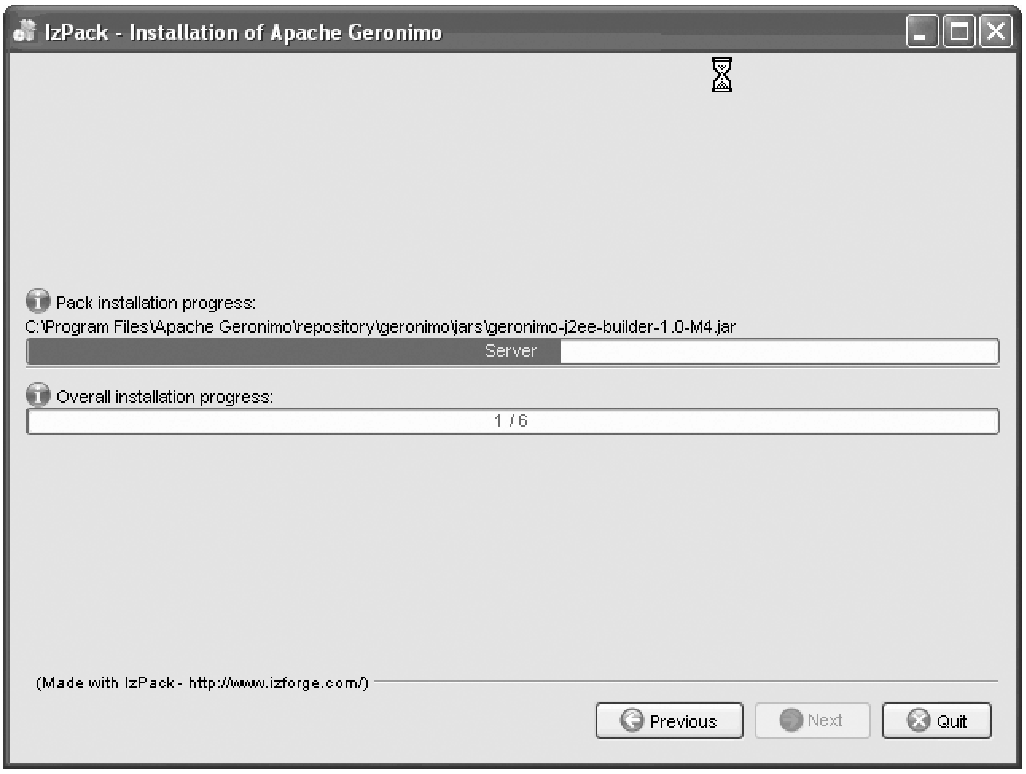
The network port that the Derby DB should use

Derby Port: 1527

(Made with IzPack - http://www.izforge.com/)
Previous
Next
Quit

Figure 1-8. *Additional services configuration screen*

To start the installation, click Next. An installation progress screen, shown in Figure 1-9, appears.


IzPack - Installation of Apache Geronimo

Pack installation progress:
C:\Program Files\Apache Geronimo\repository\geronimo\jars\geronimo-j2ee-builder-1.0-M4.jar

Server

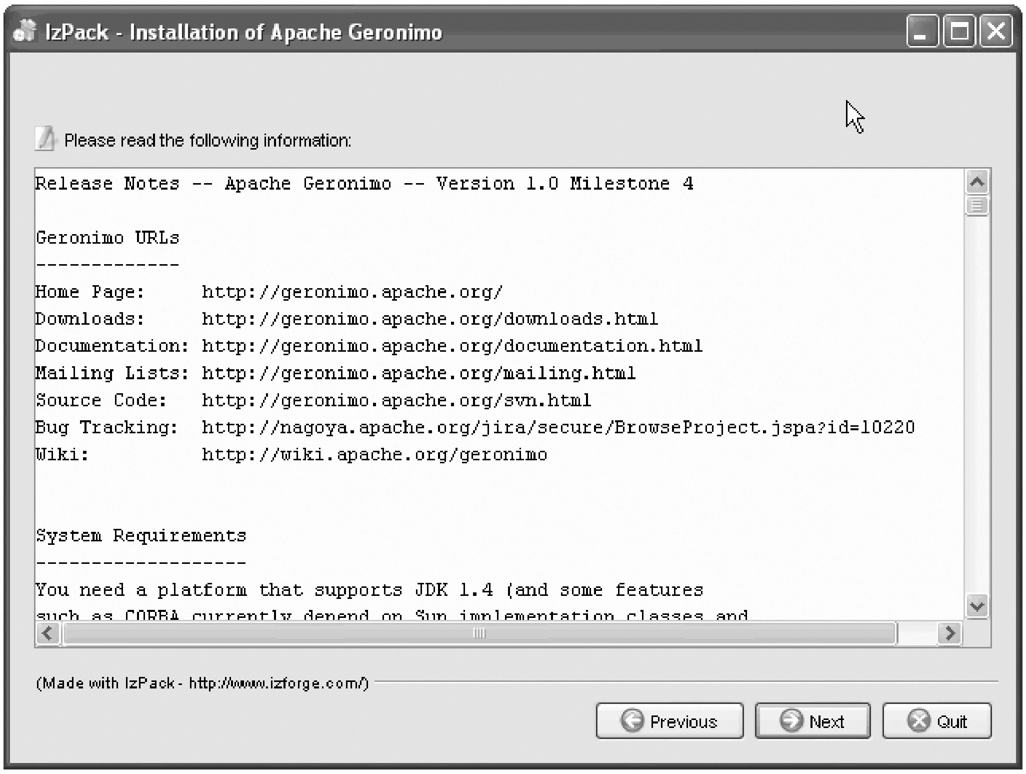
Overall installation progress:

1 / 6

(Made with IzPack - http://www.izforge.com/)

Figure 1-9. *Installation progress screen*

After installation, the installer displays the installation summary, as shown in Figure 1-10.


IzPack - Installation of Apache Geronimo
Please read the following information:
(Made with IzPack - http://www.izforge.com/)
Previous
Next
Quit

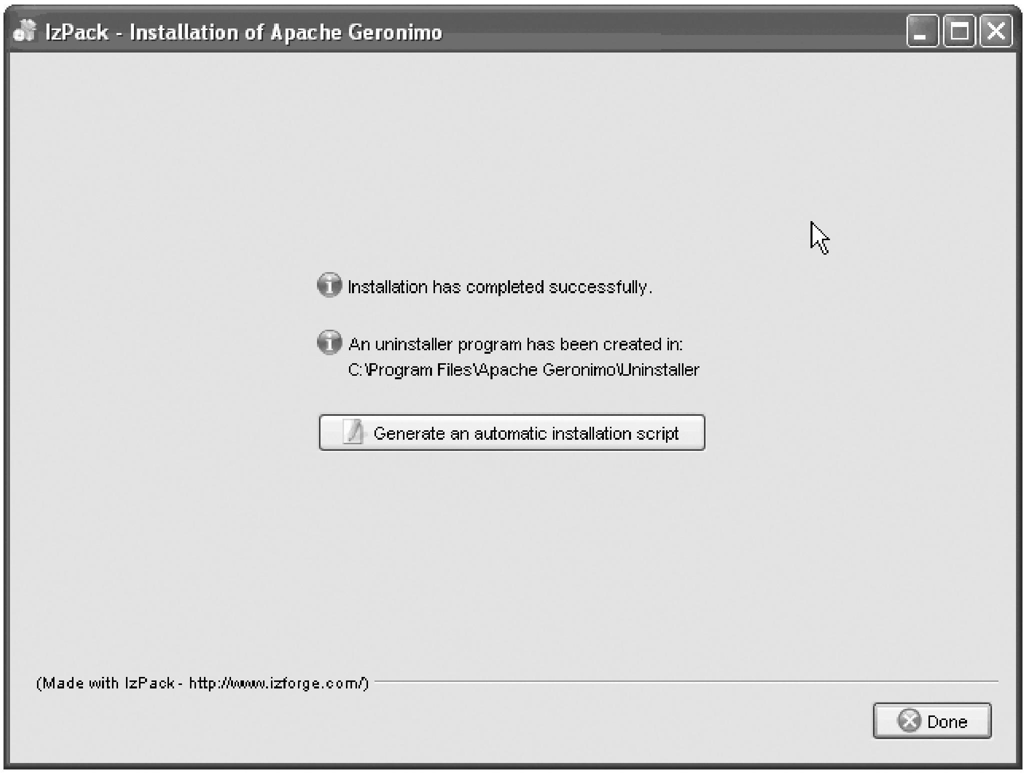
```
Release Notes -- Apache Geronimo -- Version 1.0 Milestone 4

Geronimo URLs
-------------
Home Page:     http://geronimo.apache.org/
Downloads:     http://geronimo.apache.org/downloads.html
Documentation: http://geronimo.apache.org/documentation.html
Mailing Lists: http://geronimo.apache.org/mailing.html
Source Code:   http://geronimo.apache.org/svn.html
Bug Tracking:  http://nagoya.apache.org/jira/secure/BrowseProject.jspa?id=10220
Wiki:          http://wiki.apache.org/geronimo


System Requirements
-------------------
You need a platform that supports JDK 1.4 (and some features
such as CORBA currently depend on Sun implementation classes and
```

Figure 1-10. *Installation summary screen*

Finally, the installer displays the installation success screen, shown in Figure 1-11. If you require an installation script, you can create it and save it as an Extensible Markup Language (XML) file by selecting Generate an automatic installation script.

You can use the following installation script:

```
java -jar geronimo-1.0-M4-installer.jar install-file.xml
```


IzPack - Installation of Apache Geronimo
Generate an automatic installation script
Done

Installation has completed successfully.

An uninstaller program has been created in:
C:\Program Files\Apache Geronimo\Uninstaller

(Made with IzPack - http://www.izforge.com/)

Figure 1-11. *Installation success screen*

Browsing the Geronimo Installation

During installation, the Geronimo installer creates the directory structure shown in Figure 1-12.

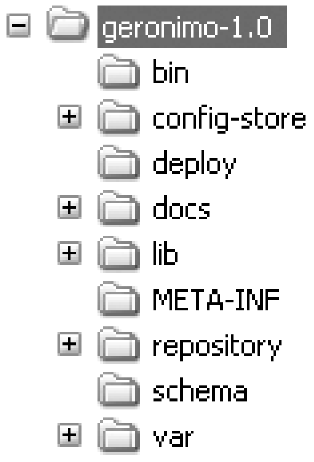


Figure 1-12. *Geronimo directory structure*

The following list describes the directories:

- *bin directory*: Contains JAR files (and scripts) that you can use to start the server and deploy applications to the server in both online and offline modes.
- *config-store directory*: Contains configuration modules in numbered subdirectories. A configuration module can be a user-deployed application or a system service. You can configure the server to start one or more configurations, as required. The default server configuration starts some of these modules. The file index.properties contains a mapping of configuration module names to subdirectory numbers. The Geronimo deploy tool maintains the contents of this directory automatically.
- *deploy directory*: Provides a hot-deployment directory into which you can copy your application artifacts for hot deployment.
- *lib directory*: Holds kernel libraries that are needed to start the bare server, which in turn loads all the required configuration modules.
- *repository directory*: Provides a store for all shared libraries. These libraries are loaded on a need basis. You can add database drivers and other libraries that need to be available to all application modules in this directory.
- *schema directory*: Holds a reference copy of all XML schema definitions.
- *var directory*: Holds server runtime contents like the log files, configuration information, and the security configuration files.

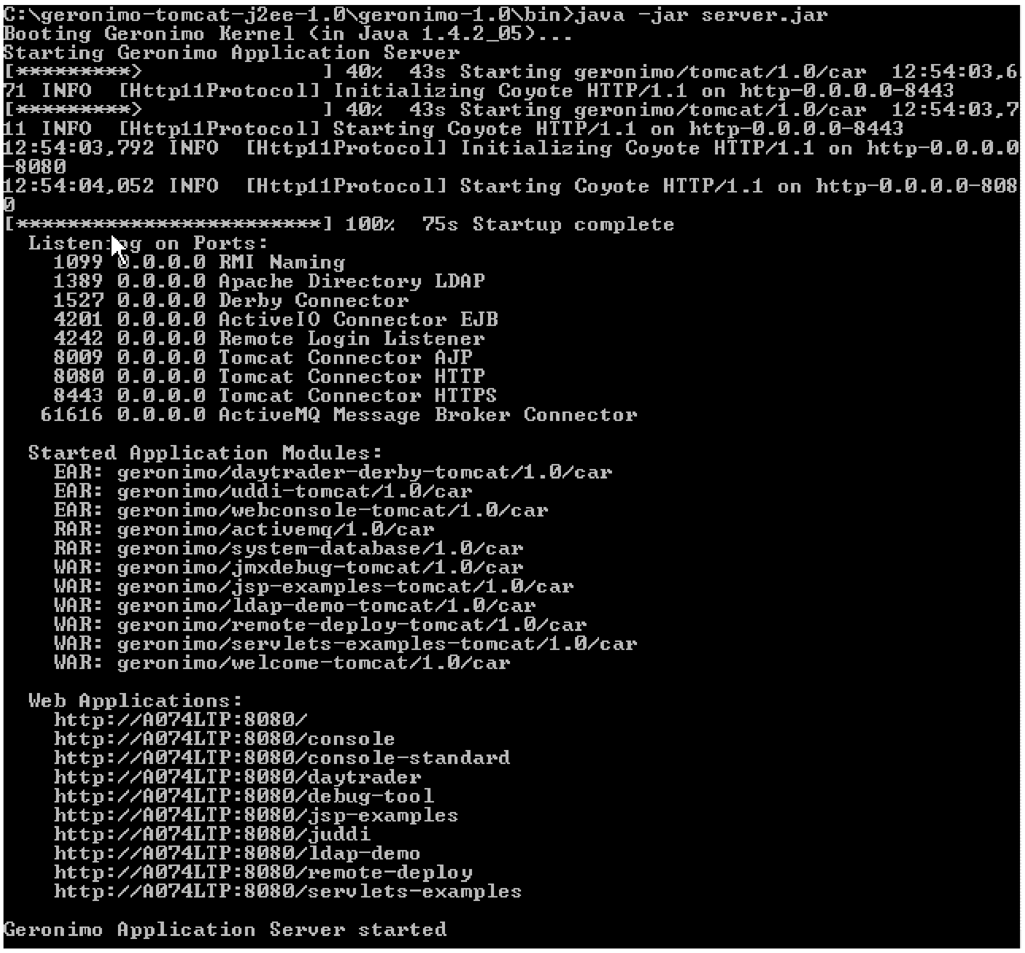
Starting the Server

To start Geronimo, run the following command from the home directory:

```
java -jar bin/server.jar
```

This command first starts the kernel and then loads the required configuration modules, which in turn load the following modules (also shown in Figure 1-13):

- A web container (Tomcat or Jetty on ports 8080 and 8443)
- An EJB container (OpenEJB with naming services on port 1099)
- A JMS broker (ActiveMQ on port 61616)
- An embedded database (Derby on port 1527)
- A log service that writes to var/log/geronimo.log
- A transaction manager
- A security realm based on security configurations in var/security/
- A JMX connector for outside JMX clients to manage and monitor server components



```
C:\geronimo-tomcat-j2ee-1.0\geronimo-1.0\bin>java -jar server.jar
Booting Geronimo Kernel (in Java 1.4.2_05)...
Starting Geronimo Application Server
[*********>              ] 40%  43s Starting geronimo/tomcat/1.0/car  12:54:03,6
71 INFO  [Http11Protocol] Initializing Coyote HTTP/1.1 on http-0.0.0.0-8443
[*********>              ] 40%  43s Starting geronimo/tomcat/1.0/car  12:54:03,7
11 INFO  [Http11Protocol] Starting Coyote HTTP/1.1 on http-0.0.0.0-8443
12:54:03,792 INFO  [Http11Protocol] Initializing Coyote HTTP/1.1 on http-0.0.0.0
-8080
12:54:04,052 INFO  [Http11Protocol] Starting Coyote HTTP/1.1 on http-0.0.0.0-808
0
[************************] 100%  75s Startup complete
  Listening on Ports:
    1099 0.0.0.0 RMI Naming
    1389 0.0.0.0 Apache Directory LDAP
    1527 0.0.0.0 Derby Connector
    4201 0.0.0.0 ActiveIO Connector EJB
    4242 0.0.0.0 Remote Login Listener
    8009 0.0.0.0 Tomcat Connector AJP
    8080 0.0.0.0 Tomcat Connector HTTP
    8443 0.0.0.0 Tomcat Connector HTTPS
   61616 0.0.0.0 ActiveMQ Message Broker Connector

  Started Application Modules:
    EAR: geronimo/daytrader-derby-tomcat/1.0/car
    EAR: geronimo/uddi-tomcat/1.0/car
    EAR: geronimo/webconsole-tomcat/1.0/car
    RAR: geronimo/activemq/1.0/car
    RAR: geronimo/system-database/1.0/car
    WAR: geronimo/jmxdebug-tomcat/1.0/car
    WAR: geronimo/jsp-examples-tomcat/1.0/car
    WAR: geronimo/ldap-demo-tomcat/1.0/car
    WAR: geronimo/remote-deploy-tomcat/1.0/car
    WAR: geronimo/servlets-examples-tomcat/1.0/car
    WAR: geronimo/welcome-tomcat/1.0/car

  Web Applications:
    http://A074LTP:8080/
    http://A074LTP:8080/console
    http://A074LTP:8080/console-standard
    http://A074LTP:8080/daytrader
    http://A074LTP:8080/debug-tool
    http://A074LTP:8080/jsp-examples
    http://A074LTP:8080/juddi
    http://A074LTP:8080/ldap-demo
    http://A074LTP:8080/remote-deploy
    http://A074LTP:8080/servlets-examples

Geronimo Application Server started
```

Figure 1-13. *Geronimo server startup*

You can also start Geronimo by using one of the following two commands: `geronimo.bat start` or `startup.bat`. This starts the server in a new command window.

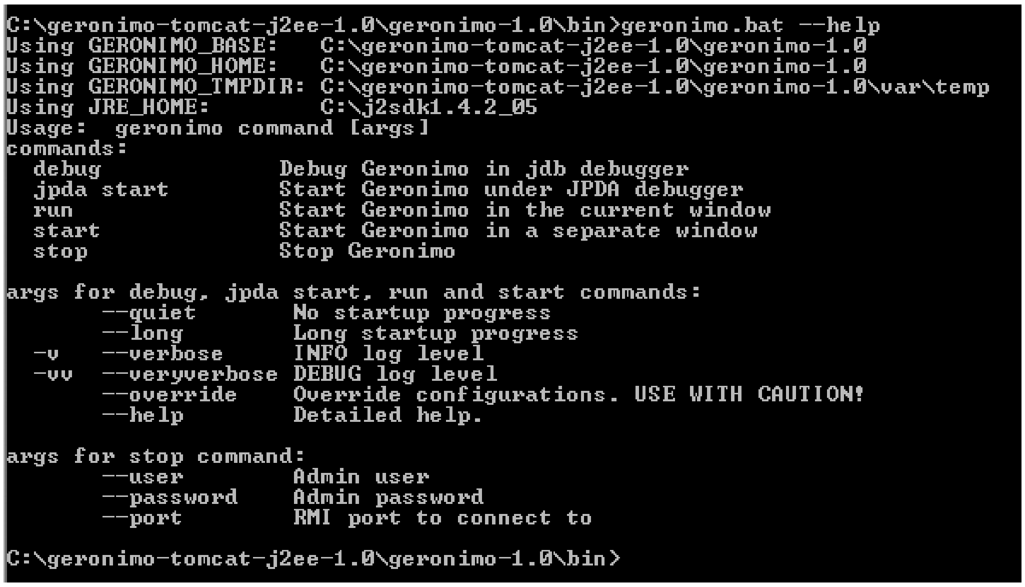
You can get the help listing, shown in Figure 1-14, by issuing this command:

```
geronimo.bat --help
```

You can also give as parameters to the following command a list of configurations that you want to start:

```
geronimo.bat start --override geronimo/j2ee-system/1.0/car
```

This command will load and start only the System configuration. In the default mode, when no configurations are explicitly given as command parameters, Geronimo will load all known configurations. This list of configurations is obtained from the file var/config/config.xml. Geronimo keeps this list up-to-date with all last-known configurations.



```
C:\geronimo-tomcat-j2ee-1.0\geronimo-1.0\bin>geronimo.bat --help
Using GERONIMO_BASE:   C:\geronimo-tomcat-j2ee-1.0\geronimo-1.0
Using GERONIMO_HOME:   C:\geronimo-tomcat-j2ee-1.0\geronimo-1.0
Using GERONIMO_TMPDIR: C:\geronimo-tomcat-j2ee-1.0\geronimo-1.0\var\temp
Using JRE_HOME:        C:\j2sdk1.4.2_05
Usage:  geronimo command [args]
commands:
  debug             Debug Geronimo in jdb debugger
  jpda start        Start Geronimo under JPDA debugger
  run               Start Geronimo in the current window
  start             Start Geronimo in a separate window
  stop              Stop Geronimo

args for debug, jpda start, run and start commands:
       --quiet       No startup progress
       --long        Long startup progress
  -v   --verbose     INFO log level
  -vv  --veryverbose DEBUG log level
       --override    Override configurations. USE WITH CAUTION!
       --help        Detailed help.

args for stop command:
       --user        Admin user
       --password    Admin password
       --port        RMI port to connect to

C:\geronimo-tomcat-j2ee-1.0\geronimo-1.0\bin>
```

Figure 1-14. *Geronimo startup help*

You can also specify verbose options as command parameters when you start the server. To start Geronimo in silent mode, use the following command:

```
geronimo.bat start --quiet
```

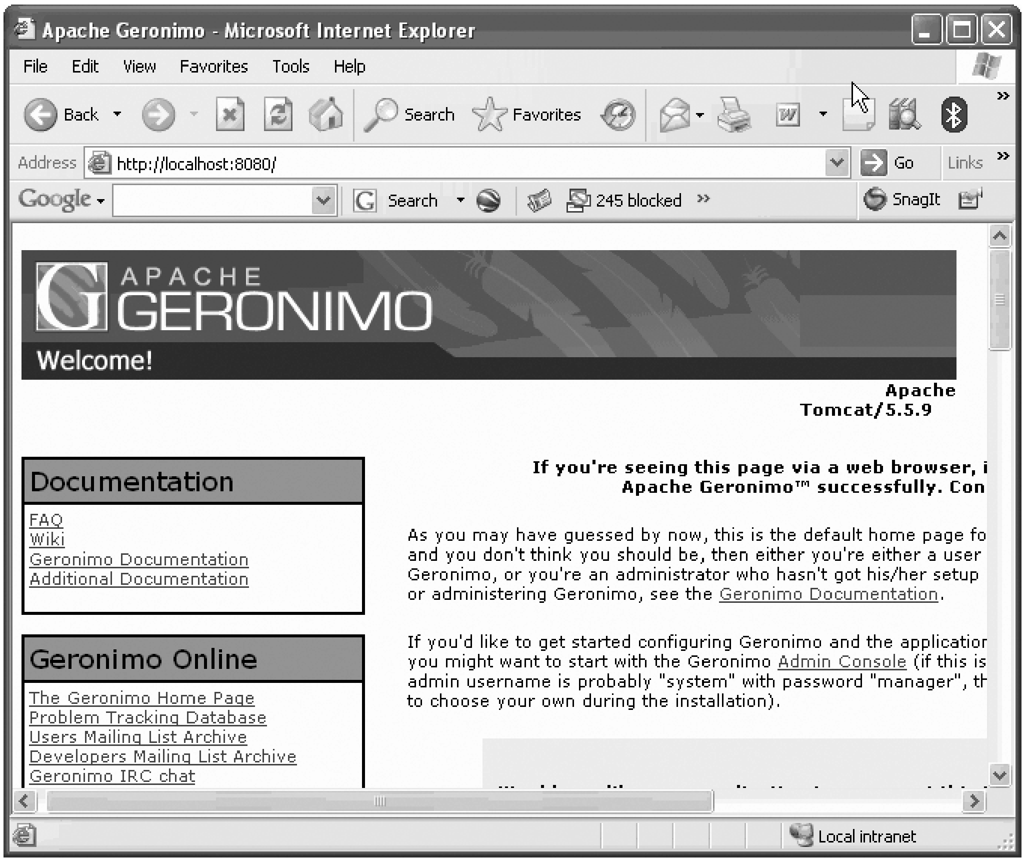
For verbose mode, use the following command:

```
geronimo.bat start -v
```

This option will print all log messages with priority greater than or equal to INFO. To print all log messages (including the DEBUG messages) use the following command:

```
geronimo.bat start -vv
```

Once the Geronimo J2EE server starts, you can test the installation by using a browser to access `http://localhost:8080/`. This should bring up the default home page, as shown in Figure 1-15.


Apache Geronimo - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/
Go Links
Google Search 245 blocked SnagIt
APACHE GERONIMO
Welcome!
Local intranet

Apache
Tomcat/5.5.9

Documentation

FAQ
Wiki
Geronimo Documentation
Additional Documentation

Geronimo Online

The Geronimo Home Page
Problem Tracking Database
Users Mailing List Archive
Developers Mailing List Archive
Geronimo IRC chat

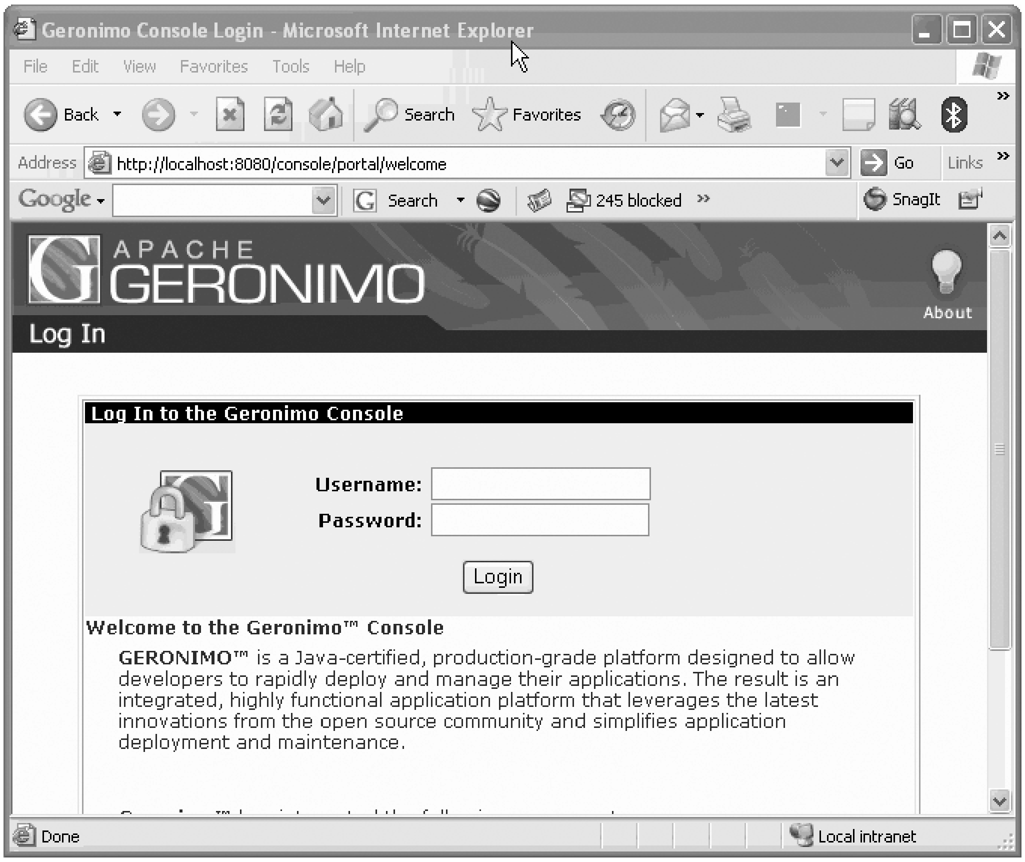
If you're seeing this page via a web browser, i
Apache Geronimo™ successfully. Con

As you may have guessed by now, this is the default home page fo
and you don't think you should be, then either you're either a user
Geronimo, or you're an administrator who hasn't got his/her setup
or administering Geronimo, see the Geronimo Documentation.

If you'd like to get started configuring Geronimo and the applicatior
you might want to start with the Geronimo Admin Console (if this is
admin username is probably "system" with password "manager", th
to choose your own during the installation).

Figure 1-15. *Geronimo default page*

You can access the administration console, shown in Figure 1-16, at `http://host:port/console`. Log in by using the user name "system" and the password "manager."


Geronimo Console Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/console/portal/welcome Go Links
Google Search 245 blocked SnagIt
APACHE GERONIMO
About
Log In

Log In to the Geronimo Console

Username:

Password:

Login

Welcome to the Geronimo™ Console

GERONIMO™ is a Java-certified, production-grade platform designed to allow developers to rapidly deploy and manage their applications. The result is an integrated, highly functional application platform that leverages the latest innovations from the open source community and simplifies application deployment and maintenance.

Figure 1-16. *Geronimo administration console*

Stopping the Server

To stop the server, press Ctrl+C in the console window. You can also use one of the following commands from the command window to stop the server: `geronimo.bat stop` or `shutdown.bat`.

Summary

In this chapter, you learned about the features of the Geronimo application server. You also saw how to install and use Geronimo. In the next chapter, you'll examine Geronimo's architecture.