Getting Started Guide

Sun ONE Application Server

Version 7, Update 1

Contents

Preface

This chapter describes the contents of Sun™ Open Net Environment (Sun ONE) Application Server 7 *Getting Started Guide*.

This preface contains the following sections:

- About This Guide
- How This Guide is Organized
- Using the Documentation
- Documentation Conventions
- Product Support

About This Guide

This *Getting Started Guide* is intended for first-time users of the Sun ONE Application Server. It offers a brief, hands-on means of gaining familiarity with basic development and administrative facilities of the product. Following this guide will typically take one to two hours of your time. Prior application server and development experience are not prerequisites for the exercises in this guide.

The guide first introduces you to the application server, then leads you through the configuration of a typical environment. Next, it describes the basic Evaluation Installation process for users who have not yet installed the product. The guide then demonstrates some commonly performed tasks from basic administrative operations, such as starting and stopping application server instances, to building, deploying and modifying an application. Further tasks explore common administrative activities such as changing the server port number, creating virtual servers, creating application server instances, and creating administrative domains.

See the Sun ONE Application Server 7 documentation collection at http://docs.sun.com/ for the latest version of this document.

How This Guide is Organized

The following chapters are included in this guide:

*   Chapter 1, "Introduction to Sun ONE Application Server 7" provides an overview of the application server.

*   Chapter 2, "Evaluation Installation of Sun ONE Application Server" contains the quick installation procedures for both the Windows and UNIX® platforms. This option allows users to rapidly install the Sun ONE Application Server for evaluation purposes.

*   Chapter 3, "Setting Up Your Environment" describes how to add the application server's `bin` directory to your PATH environment variable.

*   Chapter 4, "Starting and Stopping the Application Server" describes how to start the application server environment and verify that it is running properly.

*   Chapter 5, "Becoming Familiar with the Sample Applications" describes how to build and deploy a basic sample application. After successfully verifying that the application works correctly, you will modify the application and redeploy it.

*   Chapter 6, "Setting Up Database Connectivity" describes how to define the necessary JDBC-related settings in the application server environment prior to deploying and exercising the sample.

*   Chapter 7, "Deploying and Running the Sample Application" describes how to use the Ant facility and the sample's `build.xml` file to quickly compile the application source code and reassemble the EAR file from scratch, then deploy this newly built EAR file to the application server and exercise the application.

*   Chapter 8, "Modifying the Sample Application" introduces the dynamic deployment capabilities of the application server and leads you through several practical examples based on the sample application you previously deployed.

*   Chapter 9, "Exploring the Server Further" leads you through several exercises that demonstrate common administrative activities.

*   Chapter 10, "Summary and Next Steps" describes the tasks you've completed in this guide and the next steps to take in working with the product.

Using the Documentation

The Sun ONE Application Server manuals are available as online files in Portable Document Format (PDF) and Hypertext Markup Language (HTML) formats, at:

`http://docs.sun.com/`

The following table lists tasks and concepts described in the Sun ONE Application Server manuals. The left column lists the tasks and concepts, and the right column lists the corresponding manuals.

Sun ONE Application Server Documentation Roadmap

For information about	See the following
Late-breaking information about the software and the documentation	*Release Notes*
Supported platforms and environments	*Platform Summary*
Introduction to the application server, including new features, evaluation installation information, and architectural overview.	*Getting Started Guide*
Installing Sun ONE Application Server and its various components (sample applications, Administration interface, Sun ONE Message Queue).	*Installation Guide*
Creating and implementing J2EE applications that follow the open Java standards model on the Sun ONE Application Server 7. Includes general information about application design, developer tools, security, assembly, deployment, debugging, and creating lifecycle modules.	*Developer's Guide*
Creating and implementing J2EE applications that follow the open Java standards model for web applications on the Sun ONE Application Server 7. Discusses web application programming concepts and tasks, and provides sample code, implementation tips, and reference material.	*Developer's Guide to Web Applications*
Creating and implementing J2EE applications that follow the open Java standards model for enterprise beans on the Sun ONE Application Server 7. Discusses EJB programming concepts and tasks, and provides sample code, implementation tips, and reference material.	*Developer's Guide to Enterprise JavaBeans*
Creating Web Services, RMI-IIOP, or other clients that access J2EE applications on the Sun ONE Application Server 7	*Developer's Guide to Clients*

Sun ONE Application Server Documentation Roadmap

For information about	See the following
J2EE features such as JDBC, JNDI, JTS, JMS, JavaMail, resources, and connectors	*Developer's Guide to J2EE Features and Services*
Creating custom NSAPI plugins	*Developer's Guide to NSAPI*
Performing the following administration tasks:	*Administrator's Guide*
• Using the Administration interface and the command line interface	
• Configuring server preferences	
• Using administrative domains	
• Using server instances	
• Monitoring and logging server activity	
• Configuring the web server plugin	
• Configuring the Java Messaging Service	
• Using J2EE features	
• Configuring support for CORBA-based clients	
• Configuring database connectivity	
• Configuring transaction management	
• Configuring the web container	
• Deploying applications	
• Managing virtual servers	
Editing server configuration files	*Administrator's Configuration File Reference*
Configuring and administering security for the Sun ONE Application Server 7 operational environment. Includes information on general security, certificates, and SSL/TLS encryption. Web-core-based security is also addressed.	*Administrator's Guide to Security*
Configuring and administering service provider implementation for J2EE CA connectors for the Sun ONE Application Server 7. Includes information about the Administration Tool, DTDs and provides sample XML files.	*J2EE CA Service Provider Implementation Administrator's Guide*
Migrating your applications to the new Sun ONE Application Server 7 programming model from the Netscape Application Server version 2.1, including a sample migration of an Online Bank application provided with Sun ONE Application Server	*Migration Guide*

Sun ONE Application Server Documentation Roadmap

For information about	See the following
Using Sun ONE Message Queue.	The Sun ONE Message Queue documentation at: http://docs.sun.com/

Documentation Conventions

This section describes the types of conventions used throughout this guide:

- General Conventions

- Conventions Referring to Directories

General Conventions

The following general conventions are used in this guide:

- **File and directory paths** are given in UNIX® format (with forward slashes separating directory names). For Windows versions, the directory paths are the same, except that backslashes are used to separate directories.

- **URLs** are given in the format:

   `http://`*server.domain/path/file*`.html`

   In these URLs, `server` is the server name where applications are run; `domain` is your Internet domain name; `path` is the server's directory structure; and `file` is an individual filename. Italic items in URLs are placeholders.

- **Font conventions** include:

   ○ The `monospace` font is used for sample code and code listings, API and language elements (such as function names and class names), file names, path names, directory names, and HTML tags.

   ○ *Italic* type is used for book titles, emphasis, variables and placeholders, and words used in the literal sense.

   ○ **Bold** type is used as either a paragraph lead-in or to indicate words used in the literal sense.

- **Installation root directories** for most platforms are indicated by *install_dir* in this document. Exceptions are noted in "Conventions Referring to Directories" on page 12.

  By default, the location of *install_dir* on **most** platforms is:

  ○ Solaris 8 non-package-based Evaluation installations:

    user's_home_directory/sun/appserver7

  ○ Solaris unbundled, non-evaluation installations:

    /opt/SUNWappserver7

  ○ Windows, all installations:

    C:\Sun\AppServer7

  For the platforms listed above, *default_config_dir* and *install_config_dir* are identical to *install_dir*. See "Conventions Referring to Directories" on page 12 for exceptions and additional information.

- **Instance root directories** are indicated by *instance_dir* in this document, which is an abbreviation for the following:

  default_config_dir/domains/*domain*/instance>

- **UNIX-specific descriptions** throughout this manual apply to the Linux operating system as well, except where Linux is specifically mentioned.

Conventions Referring to Directories

By default, when using the Solaris 8 and 9 package-based installation and the Solaris 9 bundled installation, the application server files are spread across several root directories. These directories are described in this section.

- **For Solaris 9 bundled installations**, this guide uses the following document conventions to correspond to the various default installation directories provided:

  ○ *install_dir* refers to /usr/appserver/, which contains the static portion of the installation image. All utilities, executables, and libraries that make up the application server reside in this location.

  ○ *default_config_dir* refers to /var/appserver/domains, which is the default location for any domains that are created.

- *install_config_dir* refers to `/etc/appserver/`, which contains installation-wide configuration information such as licenses and the master list of administrative domains configured for this installation.

- **For Solaris 8 and 9 package-based, non-evaluation, unbundled installations**, this guide uses the following document conventions to correspond to the various default installation directories provided:

    - *install_dir* refers to `/opt/SUNWappserver7`, which contains the static portion of the installation image. All utilities, executables, and libraries that make up the application server reside in this location.

    - *default_config_dir* refers to `/var/opt/SUNWappserver7/domains` which is the default location for any domains that are created.

    - *install_config_dir* refers to `/etc/opt/SUNWappserver7/`, which contains installation-wide configuration information such as licenses and the master list of administrative domains configured for this installation.

Product Support

If you have problems with your system, contact customer support using one of the following mechanisms:

- The online support web site at:

    `http://www.sun.com/supporttraining/`

- The telephone dispatch number associated with your maintenance contract

Please have the following information available prior to contacting support. This helps to ensure that our support staff can best assist you in resolving problems:

- Description of the problem, including the situation where the problem occurs and its impact on your operation

- Machine type, operating system version, and product version, including any patches and other software that might be affecting the problem

- Detailed steps on the methods you have used to reproduce the problem

- Any error logs or core dumps

Product Support

Introduction to Sun ONE Application Server 7

The following topics are included in this chapter:

- About Sun ONE Application Server
- Product Line Overview
- What's New in Sun ONE Application Server
- Major Product Components
- Architectural Overview

About Sun ONE Application Server

Sun ONE Application Server provides the robust J2EE platform for the development, deployment, and management of e-commerce applications and web services to a broad range of servers, clients, and devices. Key features include transaction management, performance, scalability, security, and enterprise application integration. Sun ONE Application Server leverages over six years of Sun expertise in delivering highly scalable application server technology, enabling developers to rapidly build robust applications that are based on JavaServer Pages (JSP™) technology, Java Servlet, and Enterprise JavaBeans™ (EJB™) technology. This technology supports a broad range of business requirements from small departmental applications to enterprise-scale, mission-critical services.

Product Line Overview

Sun ONE Application Server 7 is a J2EE 1.3 specification-compatible application server which also supports emerging Java Web Services standards as well as standard HTTP server programming facilities. Three editions of the application server are offered to suit a variety of needs for both production and development environments:

- Platform Edition

- Standard Edition

- Enterprise Edition

Platform Edition

Platform Edition forms the core of the Sun ONE Application Server 7 product line. This free-for-production-use product offers a high-performance, small-footprint J2EE 1.3 specification-compatible runtime environment that is ideally suited for basic operational deployments, as well as for embedding in third-party applications. Web-services ready, the Platform Edition includes built-in technologies proven by the Sun ONE Web Server and Sun ONE Message Queue products.

Platform Edition deployments are limited to single application server instances (i.e. single virtual machines for the Java platform ("Java virtual machine" or "JVM™")). Multi-tier deployment topologies are supported by the Platform edition, but the web server tier proxy does not perform load balancing. In Platform Edition, administrative utilities are limited to local clients only.

The Platform Edition of Sun ONE Application Server 7 is scheduled to be integrated into Solaris 9 from early 2003 onwards.

Standard Edition

This is the edition that is the focus of this *Getting Started Guide.* The Standard Edition layers enhanced, remote-management capabilities on top of the Platform Edition. Enhanced management capabilities, remote command-line, and web-based administration are all included as part of the Standard Edition. This edition also includes the ability to partition web application traffic through a web server tier proxy. Standard Edition supports configuration of multiple application server instances (JVMs) per machine.

Enterprise Edition

Enterprise Edition enhances the core application server platform with greater high availability, load balancing and clustering capabilities suited for the most demanding J2EE-based application deployments. The management capabilities of the Standard Edition are extended in Enterprise Edition to account for multi-instance and multi-machine deployments.

Clustering support includes easy-to-configure groups of cloned application server instances to which client requests can be load balanced. Both external load balancers and load balancing web tier-based proxies are supported by this edition. HTTP session, stateful session bean instance and Java Message Service (JMS) resource failover are included in the Enterprise Edition. The patented "Always On" highly available database technology forms the basis for the HA persistence store in the Enterprise Edition.

For more product information, see the Sun ONE Application Server page on the Sun Microsystems web site.

What's New in Sun ONE Application Server

The Sun ONE Application Server 7 core introduces a wide variety of new features that enhance both the developer and operational experience.

All of these new features are included in all editions of the product.

* Developer Features

* Operational Features

Developer Features

The Sun ONE Application Server 7 distribution includes the following developer features:

* Fully Java 2 Enterprise Edition 1.3 Compatible

* JavaServer Pages (JSP) 1.2 and Servlet 2.3 Support

* Enterprise JavaBeans (EJB) 2.0 technology

* Message Driven Beans (MDBs)

* Java 2 Platform, Standard Edition (J2SE™ platform) 1.4

- Integrated Java Web Services

- Updated Sun ONE Studio Integration

- Dynamic ("Hot") Deployment and Reloading

- Greatly Enhanced Container Managed Persistence (CMP) Support

- Easy-to-configure, XML-based Server Configuration

- Lifecycle Listener Classes (sophisticated startup and shutdown classes)

- Integrated J2EE Application Verification Utility

- Extensive Sample Applications

- Ant Build Facility Integration

- Easy Installation with Minimal Dependencies (no separate web server required)

Operational Features

The Sun ONE Application Server 7 distribution includes the following operational features:

- Integrated, High-Performance HTTP Server and Web Container

- Integrated, Proven JMS Provider

- Virtual HTTP Server Support for Web Applications

- Multiple Administrative Domains per Install Image (Separate application server configurations. Not available for beta.)

- Web-based Administration

- Full-featured, Remotable Command-line Interface Supporting Remote Monitoring

- Plugable Authentication Based on Java Authentication and Authorization Service (JAAS)

- Improved Logging

Major Product Components

The Sun ONE Application Server 7, Standard Edition distribution includes the following components:

- Application Server Core

- Sun ONE Messages Queue 3.0.1

- Sun ONE Studio 4, Enterprise Edition for Java (trial license)

- PointBase Database Server and Type 4 JDBC Driver

- Java 2 Software Development Kit, Standard Edition 1.4.0_02

Downloadable evaluation distributions are available in two forms: One distribution includes the Sun ONE Studio IDE while the other does not.

The evaluation distribution on CD ROM includes the Sun ONE Studio IDE.

All of the components listed above are included in the evaluation distributions and are installed automatically as part of your installation.

Application Server Core

The Application Server Core is a deployment ready, J2EE 1.3 certified application server and HTTP server.

- Core Runtime Including Runtime Control, Process and Thread Management

- HTTP Server Based on Sun ONE Web Server

- J2EE 1.3 Web and EJB Containers

- Persistence Layer

- JTS-based, All Java Transaction Manager

- Object Request Broker (ORB)

- SNMP Agent (placed after ORB)

- HTTP Proxy Plugin (placed after SNMP Agent)

- J2EE Verifier Utilities

- Apache Ant and Sun Optional Tasks
- Administrative Components
  - ❍ Web-based Administrative Server
  - ❍ Full-featured, Remotable Command-line Interface

Sun ONE Messages Queue 3.0.1

Also available as a separate product, the Sun ONE Message Queue 3.0.1 product (formerly known as iPlanet Message Queue) is an integral part of the of the application server. This all Java component delivers a robust JMS provider for both JMS clients and MDBs.

Sun ONE Studio 4, Enterprise Edition for Java

The Sun ONE Studio 4, Enterprise Edition for Java IDE product including a trial license is available as part of the application server evaluation distribution. This product is the follow on release of the product formerly known as Forte for Java 3, Enterprise Edition for Java.

PointBase Database Server and Type 4 JDBC Driver

PointBase Server 4.2 relational database is included with the application server to support the bundled sample applications and development of JDBC-based applications. PointBase's Type 4 driver supporting the JDBC™ API ("JDBC driver") is preconfigured during installation. Tables are also created and populated for all of the sample applications that depend on JDBC. The bundled PointBase distribution has a 5 Mb total database size limitation.

Java 2 Software Development Kit, Standard Edition 1.4.0_02

Sun ONE Application Server requires J2SE 1.4.x, SDK and leverages the performance and feature improvements that are part of the 1.4 platform.

Architectural Overview

A Sun ONE Application Server 7 deployment consists of a number of application server instances, an administrative server and, optionally, one or more web server tier proxy plugins.

Sun ONE Application Server deployment
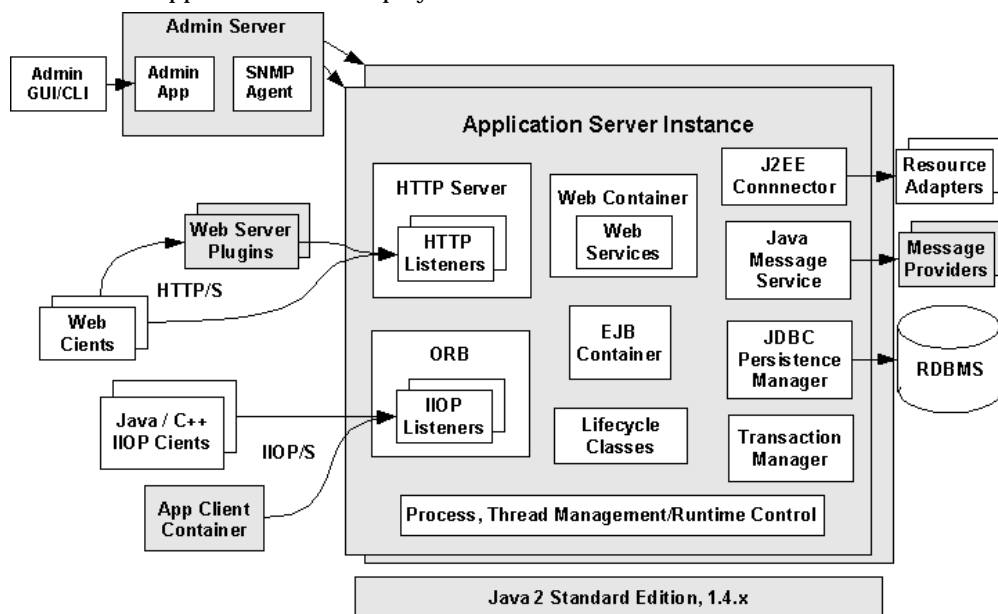
Application Server instances form the basis of an application server deployment. J2EE 1.3 web and EJB containers are included in each application server instance. A proven, high performance HTTP server is positioned in front of the web container while a built in ORB forms the underpinning of the EJB container. In support of access to backend systems, applications can leverage J2EE Connector Architecture support and third party Resource Adapters, JMS and either the built-in JMS provider or third-party providers, and any combination of popular third-party JDBC drivers. Access to backend systems can be managed within the scope of distributed transactions using the built-in, all-Java Transaction Manager.

The Administrative Server houses the core administrative application and an SNMP agent. All remote management activity flows through the administrative server. Both command line and web browser based administrative clients access the administrative server directly either through HTTP or securely through HTTPS.

Web Sever Proxy Plugins enable you to deploy the application server behind one or more web servers housed in a demilitarized zone (DMZ) that is bracketed by one or more layers of firewalls. The plugins provide a means for the front end web server tier to direct incoming HTTP/S traffic received from the Internet to one or more application servers housed in a backend application server tier.

A variety of client applications can access business services deployed to the application server. Web services and browser-based clients can use either HTTP or HTTPS to access the Java Web Services and J2EE web applications. Java application clients can be deployed either in a standalone mode or within a standard Application Client Container and can use Java Remote Method Invocation over Internet Inter-ORB Protocol Technology (Java RMI-IIOP Technology) to access EJBs deployed to the application server. C++ language clients can use Java IDL/IIOP to access EJBs as well.

Development Integration

A key feature of Sun ONE Application Server 7 is its tight integration with the popular Sun ONE Studio 4, Enterprise Edition for Java Integrated Development Environment (IDE). Included with the application server is an integration module that extends key facets of Sun ONE Studio to further enhance developer productivity when working with the Sun ONE Application Server.

Sun ONE Application Server development integration
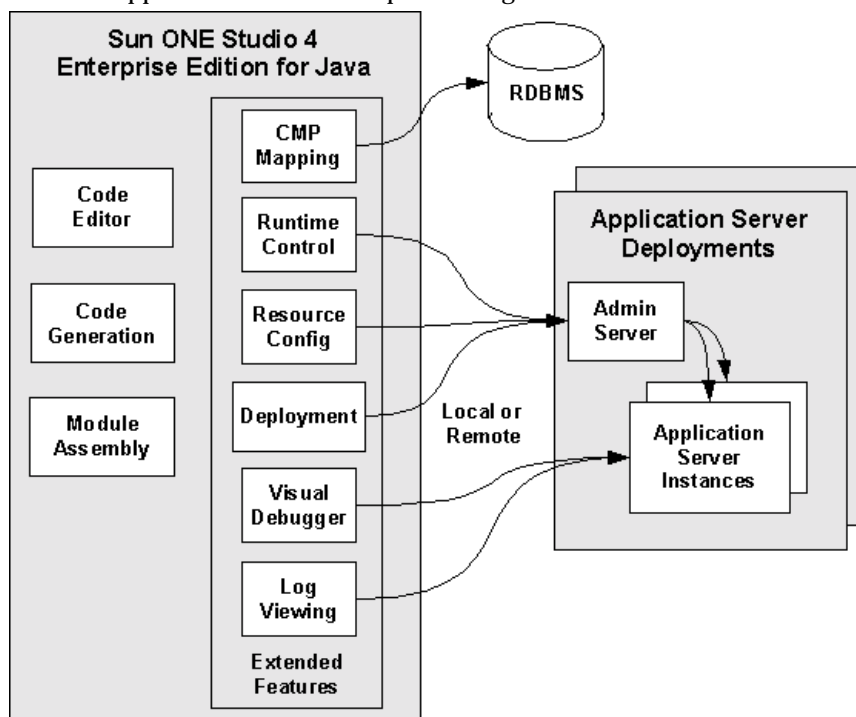
The Sun ONE Application Server integration module enhances key development features of Sun ONE Studio 4, Enterprise Edition for Java as described in the following table.

Key development features

Studio feature	Sun ONE Application Server Extension
CMP mapping	Developers can browse database tables, select related tables and automatically generate Container Managed Persistence (CMP) EJB technology.
Server runtime control	Developers can easily register both local and remote application servers and start and stop application server instances.
Resource configuration	Before deploying applications, developers can register J2EE resources in any of the registered application servers. JDBC resources and connection pools, JMS resources, and a variety of other resources can be configured from within the Studio.

Key development features

Studio feature	Sun ONE Application Server Extension
Application deployment	Developers can select from the list of registered application servers and leverage the dynamic ("hot") deployment and redeployment features supported in Sun ONE Application Server 7.
Debugging and log viewing	Debugging against deployed applications on both local and remote application server instances is simple. No manual configuration of the application servers is necessary.
	While debugging their applications, developers can also view the server event log files from within the Studio.

Deployment Topologies

Sun ONE Application Server 7, Platform and Standard Editions support both single machine and multi-machine, tiered deployments. No application server clustering is supported in these editions. Standard Edition provides enhanced web tier support by allowing you to partition HTTP/S traffic arriving on the same web server instance to multiple application servers in the middle tier. No load balancing from the plugin is supported in these editions.

While Platform Edition is limited to a single application server instance (i.e. a single JVM process) per administrative domain, the Standard Edition can be configured with multiple application server instances per administrative domain.

The Enterprise Edition supports multi-tiered, multi-machine, clustered application server deployments. In Enterprise Edition, the web tier supports load balancing and application traffic partitioning via a web server plugin. Cloning of application server instances as well as cluster oriented administrative operations are supported in Enterprise Edition.

Evaluation Installation of Sun ONE Application Server

If you have already installed the product, proceed to Chapter 3, "Setting Up Your Environment."

If you are using the Solaris 9 Operating Environment and the application server has been installed as part of the Solaris 9 installation, ensure that you follow the Solaris post-installation steps in "Configuring After Solaris 9 Installation" on page 32 to configure your application server environment.

If you are installing the product for the first time, this chapter leads you through the installation of the evaluation product using the interactive methods of installing the product. If you either have the non-evaluation distribution or would like to use the silent installation methods, refer to the Sun ONE Application Server 7 Installation Guide for detailed installation steps.

Sun ONE Application Server 7 allows different types of installation according to user role, as well as different methods of installation, such as command-line, GUI and silent modes. This chapter describes each type of installation and gives instructions for installing as an evaluator in GUI mode. When you are ready to install the product for development or operational use, refer to the *Installation Guide* for complete details.

The following topics are described in this chapter:

*   About Installation
*   Installing for Evaluation
*   Uninstalling Sun ONE Application Server
*   Configuring After Solaris 9 Installation

About Installation

Before you run the setup program as described in Sun ONE Application Server 7 *Installation Guide*, you should review the types and methods of installation described in this chapter. You'll select a type of installation based on your role as a user, then select the method of installation that most suits your installation environment.

Types of Distribution

Sun ONE Application Server is available in both evaluation and non-evaluation distributions. The evaluation distribution's installation experience is streamlined to require a minimum of user input.

The non-evaluation distribution enables you to install the product for the purpose of development or operational deployment, and to select specific components that to installed. This guide leads you through the evaluation installation experience.

The evaluation installation experience automatically installs the following components:

*   Sun ONE Application Server 7 Core Server and Utilities

*   Sun ONE Message Queue 3.0.1

*   *Getting Started Guide*

*   Sample Applications

*   Java 2 Software Development Kit, Standard Edition 1.4_02

*   PointBase Database Server and Type 4 JDBC Driver

*   Sun ONE Studio 4, Enterprise Edition for Java (trial license)

    Sun ONE Studio 4, Enterprise Edition for Java is installed automatically if you obtained the evaluation distribution that includes this component.

Installation Methods

Using the evaluation distribution, you can install the application server using any of three methods:

- Graphical-interface installation method provides a set of graphical dialogs to interact with the user.

- Command line, interactive-based installation methods includes the same steps as found in the graphical interface method of installation, but does not require a graphics-capable display. For example, if you are using telnet to access a remote server, you can use this method to install the product in an interactive fashion. The `-console` option on the installer triggers this method.

- Silent, or parameter-driven, installation enables you to perform scripted installations of the product based on the presence of a parameter file. The product is installed without any interaction by the user.

Although this guide focuses on the graphical interface installation method, you can easily use the command-line, interactive-based installation method as well.

For more information on the silent installation method, consult the *Installation Guide.*

Installing for Evaluation

This section includes the following topics:

- Preparing to Install

- Required Patches on Solaris

- Running the Setup Program

You can opt to download the basic Evaluation version of the Sun ONE Application Server or a version that also includes Sun ONE Studio 4.0.

Preparing to Install

Your system should meet the system requirements described in the following table.

Operating System	Architecture	Minimum Memory	Recommended Memory	Minimum Disk Space	Recommended Disk Space
Sun Solaris 8 for SPARC, Sun Solaris 9 for SPARC	32 and 64 bit	128 MB without Sun ONE Studio 512 MB with Sun ONE Studio	512 MB	250 MB free	500 MB free
Microsoft Windows • 2000 Advanced Server, SP2+ • 2000 Server, SP2+ or above • 2000 Professional, SP 2 or above • Windows XP Professional	Intel 32 bit	128 MB without Sun ONE Studio 256 MB with Sun ONE Studio	256 MB without Sun ONE Studio 512 MB with Sun ONE Studio	250 MB free	500 MB free

Solaris: Required Patches

The following Solaris patches must be installed for Solaris 8 systems:

109326-06 or higher

108827-26 or higher

110934 for packaged-based installation only

These patches can be retrieved from the patch finder page located at the following URL:

http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-access

It is recommended that Solaris 8 users should have the "Sun recommended patch cluster" installed. The cluster includes the three required patches listed above. This patch is available at: http://sunsolve.sun.com/ under "Recommended and Security Patches."

Once you have satisfied the system requirements, you are ready to install the application server.

Running the Setup Program

The following steps enable you to install Sun ONE Application Server 7 using graphical-interface mode. If you choose to use command-line installation instead, the steps involved in that method are identical. In place of GUI-based installation screens, text-based screens are shown during a console-based installation.

The following steps apply to all platforms:

NOTE	You must have administrative privileges on your Windows machine to install Sun ONE Application Server 7. You cannot install more than one Sun ONE Application Server on a single Windows machine.

1. Expand or unzip the installation bundle to a temporary directory.

2. If you are not in the directory already, change to the directory where you expanded the files.

NOTE	UNIX: If you are installing Sun ONE Application Server remotely, you must enable the display configuration on the machine where you are installing the product.

3. Type the following at the command prompt:

   ```
   #./setup
   ```

   To run command-line installation, type

   ```
   #./setup -console
   ```

   On Windows, you can also navigate using Windows Explorer to the directory where you expanded the files and double-click the setup file.

   The installation interface appears.

4. Read the Welcome screen and click Next.

5. Read the License Agreement, select the radio button to agree to the terms of the license, and click Next.

   You must accept the license agreement to continue with the installation.

6. Enter the path to your installation directory or click the ellipsis (...) to browse for a directory.

   If the directory does not already exist, the Create New Directory? dialog box appears. Click Create Directory. You can also click Choose New to select an existing directory.

7. On the Server Configuration Information dialog box, enter the following:

   ❍ **Admin User**: Name of the user who administers the server (for example, admin).

   ❍ **Admin User's Password**: Password to access the administration server. Minimum number of characters is 8. Re-enter the password in the text box below to confirm your choice.

   ❍ **Admin Server Port**: Port number to access the administration server.

     A default port number appears (for example 4848, if that port is not in use on your machine). Change the default number if necessary. The installation program will check port numbers for validity and availability when you click Next.

   ❍ **HTTP Server Port**: Port number to access the default server instance.

     A default port number appears (for example 1024, if that port is not in use on your machine). Change the default number if necessary. The installation program will check port numbers for validity and availability when you click Next.

NOTE	The installation program automatically detects ports in use and suggests currently unused ports for the default settings. By default, if you are running as root on UNIX or are on Windows, the initial default ports are 80 and 4848 for the HTTP server and administrative server respectively. If you are running as non-root on UNIX, the HTTP server port defaults to 1024. If any of these initial default ports are being actively used on your system, the installation program will suggest alternative port numbers.

8. The Checking Disk Space dialog box appears.

9. Click Install Now to complete the installation.

   An Installation Progress indicator bar appears.

   As installation ends, the Installation Complete screen appears. Users should review this screen and click Finish to exit the installer.

Proceed to Chapter 3, "Setting Up Your Environment."

For troubleshooting tips, see the *Installation Guide*.

Uninstalling Sun ONE Application Server

When you uninstall Sun ONE Application Server for Evaluation, all components are automatically selected for uninstallation. You cannot select individual components to uninstall.

The uninstallation program detects any running processes and stops them before continuing with the uninstall process.

Perform the following steps on both UNIX and Windows platforms:

1. Change to your machine's Sun ONE Application Server installation directory.

2. Type one of the following commands at the command prompt:

   ❍ GUI mode:

   ```
   ./uninstall
   ```

   ❍ Command-line mode:

   ```
   ./uninstall -console
   ```

3. Read the Welcome screen and click Next or hit Enter to continue with the uninstall process.

4. On the Ready to Uninstall screen, a list of uninstallable components appears. Review the components and click Next or hit Enter to proceed.

   You cannot select individual components to uninstall using the Evaluation version of Sun ONE Application Server.

5. Click Uninstall Now or hit Enter.

   The uninstallation progress meter appears.

6. The Uninstall Summary screen appears.

   Review the details and click Exit or hit Enter to quit the uninstallation program.

7. Change to your installation directory and manually remove any leftover files and directories.

Configuring After Solaris 9 Installation

The Sun ONE Application Server 7 installation that is installed as part of a Solaris 9 installation contains only the necessary libraries, executables and other files required to support the application server. No application server configuration exists upon installation. To create an initial configuration, you must use the application server's `asadmin` command line interface (CLI). The `create-domain` subcommand of the `asadmin` CLI enables you to create an administrative domain in a location of your choice. Each administrative domain consists of an administrative server and application server instance configurations. When you create an initial administrative domain, you specify the administrative user name, password and port number associated with the administrative server of the domain.

Once you've created an administrative domain, you can create one or more application server instances within the domain. Each application server instance houses an HTTP server, the J2EE[tm] web and EJB containers and other application server facilities.

Administrative domains and application server instances are explained further in the next section of this guide.

Create an Administrative Domain

By default, the `create-domain` subcommand creates the new administrative domain configuration under `/var/appserver/domains/`. If you are logged in as a user that does not have write permissions to this area, you must specify a suitable location when creating the administrative domain. You specify the location in which to create the domain configuration using the `--path` option on the create-domain subcommand.

To create an administrative domain, follow these steps:

1. If you are not using the root user ID, you will need to either request that the systems administrator create an administrative domain on your behalf or request that your user ID be added to a UNIX group that is able to create administrative domains. See the following note for details if you do not have root user access.

NOTE	**User Permissions on UNIX Platforms**: In order for a non-root user to create and delete administrative domains, the user ID must be added to a UNIX group that has write permissions to the domain configuration file. In this case, the following steps must be taken by the systems administrator.

Create a UNIX group that will be the group which is applied to the installation-wide domain configuration files. For example, a UNIX group named `asadmin`.

    a. Set the installation-wide domain config files located under `/etc/appserver` to be owned by the newly created UNIX group. The files are named domains.bin and domains.lck. For example, after changing the group assigned to these files:

```
-rw-r--r-- 1 root asadmin 0 Sep 18 14:34 domains.bin

-rw-r--r-- 1 root asadmin 0 Sep 18 14:34 domains.lck
```

    b. Enable write access to these files by the newly created UNIX group. In this example, the resulting permissions would look like the following:

```
-rw-rw-r-- 1 root asadmin 0 Sep 18 14:34 domains.bin

-rw-rw-r-- 1 root asadmin 0 Sep 18 14:34 domains.lck
```

    c. Add the user ID to the UNIX group.

Alternatively, if a systems administrator does not want to provide non-root users with write access to the installation-wide configuration files, the administrator can use the root user ID to create an administrative domain on behalf of a user. During creation of a new administrative domain, administrators can specify the `--sysuser` and `--path` options to identify the UNIX user ID that will own the domain's directories and files and the location under which the administrative domain will be created.

Once an administrative domain is created under your user ID, you may execute asadmin subcommands to create new application server instances and perform a wide variety of administrative operations against the application server instances without the user ID belonging to the UNIX group that has write privileges for the administrative domain configuration file. Membership in the UNIX group is required only to create and delete administrative domains.

2. Ensure that the `/usr/sbin` directory is included in your path.

3. From the command line, execute the following command to create a new administrative domain, `domain1`:

```
asadmin create-domain --path domain_config_dir --adminport 4848
--adminuser admin --adminpassword password domain1
```

where *domain_config_dir* specifies the location under which the administrative domain configuration will be created.

The `--adminport`, `--adminuser` and `--adminpassword` options specify the initial settings of the new administrative server defined for the domain.

If the root user is executing the `create-domain` subcommand to create an administrative domain on your behalf, the `--sysuser` option should be used to specify the system user ID under which the administrative domain files and directories will be created. For example:

```
asadmin create-domain --sysuser ckamps --path domain_config_dir
--adminport 4848 --adminuser admin --adminpassword password
domain1
```

Upon execution of the `create-domain` subcommand, you should see the following message:

```
Created Domain domain1 successfully
```

If the name `domain1` has already been used, execute the `create-domain` subcommand again with another domain name. You can use periods and other characters in your domain names. You could use your login user name as a qualifier to help ensure that your domain name is unique. For example: `ckamps.domain1.`

When you execute the `create-domain` subcommand and the following error message is encountered, the message is an indication that your user ID does not have permissions to access the domain configuration files:

```
Cannot create domain : domain1 problem locking store
/etc/appserver/domains.lck (Permission denied)
```

See the steps above for either adding your user ID to the appropriate UNIX group or requesting that your systems administrator create the administrative domain on your behalf.

4. Execute the `list-domains` subcommand to display a list of all of the domains configured for the application server installation. (Execution of this read-only command does not require your user ID to be part of the UNIX group that has write privileges to the domain configuration files).

```
asadmin list-domains domain1 [domain_config_dir/domain1]
```

where the value of *domain_config_dir* represents either the default location for newly created administrative domains or the value specified on the `--path` option of the `create-domain` subcommand.

Create an Application Server Instance

Once either the systems administrator has created an administrative domain on your behalf or you created the administrative domain, your next step is to create an application server instance under the newly created administrative domain. Creation of an application server under your own administrative domain does not require that your user ID be part of the UNIX group that has write privileges to the domain configuration files.

To create an application server instance, execute the `create-instance` subcommand to create the application server instance:

```
asadmin create-instance --domain domain1 --instanceport 80 server1
```

where `domain1` is the domain name specified during domain creation, `80` is the HTTP server port number of the application server instance, and `server1` is the name of the instance. Specify appropriate values for these options depending on your specific environment. Since port numbers less than 1024 are not accessible to non-root users, you will need to specify a port number greater than 1024 if you are logged in as a non-root user.

As long as you have only a single administrative domain defined on your system, you do not need to specify the target domain name when creating an instance.

For more information on how to prepare your application server environment after installation of the application server as part of a Solaris 9 installation, consult the Sun ONE Application Server *Administrator's Guide*.

Now that you've created an initial administrative domain and an application server instance for your application server installation, you are ready to proceed to Chapter 4, "Starting and Stopping the Application Server."

Setting Up Your Environment

After installation of the application server, one of the first steps is to configure your environment to include the application server's `bin/` directory.

This section contains the following topics:

- About the PATH Environment Variable

- Setting the PATH Variable on UNIX

- Setting the PATH Variable on Windows

- Troubleshooting Environment Settings

About the PATH Environment Variable

This section addresses how to add the following directory to your PATH environment variable:

install_dir/bin

This is the only environment setting needed to run the `asadmin` command, the application server's administrative command-line utility, and to access the `asant` utility to work with the sample applications.

If you are using the application server installation that was installed as part of a Solaris 9 installation, then you must include the directory `/usr/appserver/bin` in your PATH in order to access the asant utility. Additionally, if your PATH does not include `/usr/sbin`, inclusion of `/usr/appserver/bin` in your PATH ensures that you have access to the `asadmin` command-line interface.

NOTE	On some Windows 2000 systems, the Windows `net` command is not automatically made available to the environment through the system PATH environment variable. You must ensure this Windows utility is available in the environment in order to start and stop the application server.
	To determine whether or not the net command is available in your environment, perform the following steps:
	1. Click the Windows Start button, then choose Run.... to launch a command console window.
	2. In the Open field, type `cmd` and click OK.
	3. As the console starts, type `net` at the command prompt. If the command is not found, you must modify the system PATH environment variable to include your *Windows_install_root*>`\system32` directory. For example:
	`C:\WINNT\system32;`
	If you are unfamiliar with setting the system's PATH environment variable, see the section below for detailed instructions.

If you are familiar with the process of setting environment variables, do so now in your own environment.

After setting your PATH variable, ensure that the `asadmin` command can be found, then proceed to "Starting and Stopping the Application Server" on page 37. If the command is not found, check your PATH setting, refresh your environment settings, and execute `asadmin` again.

Otherwise, select one of the following paths depending on your platform:

• Setting the PATH Variable on UNIX

• Setting the PATH Variable on Windows

Setting the PATH Variable on UNIX

On UNIX systems, it is recommended that you add the application server's `bin` directory to your login profile so it is automatically added to your environment's PATH setting during login.

Once you have set the PATH environment and refreshed your environment, execute the `asadmin` command at the command prompt.

The following result should appear:

```
asadmin
Use "exit" to exit and "help" for online help
        asadmin>_
```

Type `exit` to quit the Administration command-line interface. Leave the terminal window open for the next chapter of this tutorial, "Starting and Stopping the Application Server" on page 43.

If the command is not found, check your PATH setting, refresh your environment settings, and execute `asadmin` again.

NOTE   The `asadmin` command launches the Administration command-line interface of the application server. By executing the `asadmin` command without arguments, you have entered the interactive mode of the command-line interface.

You can type `help` at the `asadmin` command prompt to see the complete list of subcommands supported by the command-line interface.

The `asadmin` command also fully supports the use of prebuilt command files as a means of automating server administration and monitoring. In this guide, since you will be using only a small subset of these commands, you are encouraged to review the command line interface section of the Sun ONE Application Server *Administrator's Guide*.

Setting the PATH Variable on Windows

On Windows, it is recommended that you modify the system PATH environment variable via the Windows control panel as described in the following steps:

1. Click the Windows Start button, choose Settings, then Control Panel.

2. In the Control Panel, double-click System.

3. Click on the Advanced tab, then choose Environment Variables...

The Environment Variables dialog box lists the environment variables that apply to your current user account as well as to the system as a whole.

In this exercise, you will modify the PATH environment variable for your current user account.

4. Select the existing PATH entry and click Edit or click New to create a new PATH environment variable.

5. Add the *install_dir*/bin value to the beginning of the PATH value.

For example, add `C:\Sun\AppServer7\bin;` to the front of the variable value.

6. Click OK to close the Edit User Variable dialog window.

The PATH variable should reflect the directory path that you just entered.

7. Click OK to apply the changes and to close the Environment Variables Window, then click OK to close the System Properties window.

8. Check your work by starting a command window and determining if the application server commands are available from the command line:

   a. Click the Windows Start button, then choose Run.

   b. In the Open field, enter `cmd` and click OK.

   c. When the command window appear, type `asadmin` at the command prompt.

   After execution of the asadmin command, you should see the following result:

   ```
   C:\>asadmin
   Use "exit" to exit and "help" for online help
   asadmin>
   ```

   If you see this result, then you have verified that the application server command line utilities are accessible via your path setting.

NOTE	The `asadmin` command launches the Administration command-line interface of the application server. By executing the `asadmin` command without arguments, you have entered the interactive mode of the command-line interface.
	You can type `help` at the `asadmin` command prompt to see the complete list of subcommands supported by the command-line interface.
	The `asadmin` command also fully supports the use of prebuilt command files as a means of automating server administration and monitoring. In this guide, since you will be using only a small subset of these commands, you are encouraged to review the command line interface section of the Sun ONE Application Server *Administrator's Guide.*

9. Type `exit` to quit the Administration command-line interface.

Leave the command window open and proceed to the next chapter, "Starting and Stopping the Application Server" on page 43. If the command is not found, check your PATH setting, refresh your environment settings, and execute `asadmin` again.

Troubleshooting Environment Settings

If the `asadmin` command is not recognized, go back into the Control Panel as described in Step 2 on page 39, check your PATH setting, and try to execute the `asadmin` command again. Once you have fixed the PATH setting, ensure that you start a new command window to test execution of the `asadmin` command. Only a new command window will pick up the environment variable change. You will encounter the following message when your PATH variable is not set correctly:

```
C:\>asadmin

'asadmin' is not recognized as an internal or external command,
operable program or batch file.

C:\>
```

Proceed to the next chapter, "Starting and Stopping the Application Server."

Starting and Stopping the Application Server

Since the application server is not automatically started during installation, you need to start the application server environment and verify that it is running properly before deploying and exercising the sample application. Before starting the application server, this chapter introduces you to the Administrative Domains feature, then gives a brief overview of the processes that make up an application server environment.

This chapter includes the following topics:

- Administrative Domains

- Application Server Processes

- Installation Directory Structure

- Conventions for Referring to Directories

- Tools for Starting and Stopping the Application Server

Administrative Domains

Sun ONE Application Server 7 introduces a feature named Administrative Domains that enables you to define multiple, completely separate application server runtime configurations that reuse the same installation image. Each administrative domain is represented by an administrative server which in turn controls one or more application server instances. The configuration of an administrative domain can reside anywhere on the machine.

Although developers using their own workstations will likely use a single administrative domain for day-to-day development, both shared development servers and operational environments will greatly benefit from using multiple administrative domains. On shared development servers, creation of an administrative domain for each developer provides a compartmentalized area or "sandbox" for each developer on a shared server machine. In operational environments, administrative domains enable systems administrators to define separate secure runtime configurations without requiring multiple installations of the product.

In the following exercises, you will be working with a single administrative domain that has been either preconfigured during installation of the product or configured as a post installation task after the application server has been installed as part of a Solaris 9 installation.

Application Server Processes

Before starting the application server, take a few moments to familiarize yourself with the processes that will be running behind the scenes in each administrative domain:

- Application Server Daemon

- Application Server Watchdog

- Message Queue Broker

- Message Queue Broker Wrapper

Application Server Daemon

The Application Server Daemon process, `appservd`, forms the core of the application server runtime. It houses the embedded HTTP Server, ORB, J2EE Containers and supporting subsystems such as the Transaction Manager and Persistence Managers. Each `appservd` process houses a Java Virtual Machine (JVM).

In each administrative domain, there is an `appservd` process that houses the administrative server for the domain. This process is referred to as the Administrative Server Instance. The administrative server controls and manages a number of additional `appservd` processes that represent individual application server instances.

NOTE	**appservd Processes on UNIX**: Although you will notice that there is a single `appservd` process started for each application server instance on Windows, there are two `appservd` processes started per application server instance on UNIX systems.
	On UNIX, one `appservd` process is referred to as the "primordial" process while the second `appservd` process is referred to as the "worker" process. The worker process is the process that carries out the actual processing of application requests while the primordial process acts as an overarching controller. In a future release of the application server, you will have the option to define the number of worker processes for each application server instance. In the initial release of the product, only one worker process is supported per application server instance.

Application Server Watchdog

The application server watchdog processes, `appservd-wdog`, are native language processes monitor the application server daemon processes. In case of a daemon failure, the companion watchdog restarts the failed application server daemon process. Each application server daemon has a companion watchdog process.

Message Queue Broker

By default, each application server instance includes a companion Sun ONE Message Queue Message Broker process. The message broker process is the core of the JMS of the application server.

Since there is only one application server instance created during installation, initially there is only a single message broker process started during application server startup.

As you define new application server instances, new companion message broker processes are created automatically. After creating an application server instance, you have the option of disabling the companion message broker process to support cases in which your applications either do not use the JMS facilities or use a message service that is not part of the application server (for example, MQSeries).

Message Queue Broker Wrapper

This process is simply a lightweight wrapper around the message broker process. The wrapper process manages restart of the message broker process.

The relationships among these processes is shown in the following diagram:

Application server processes
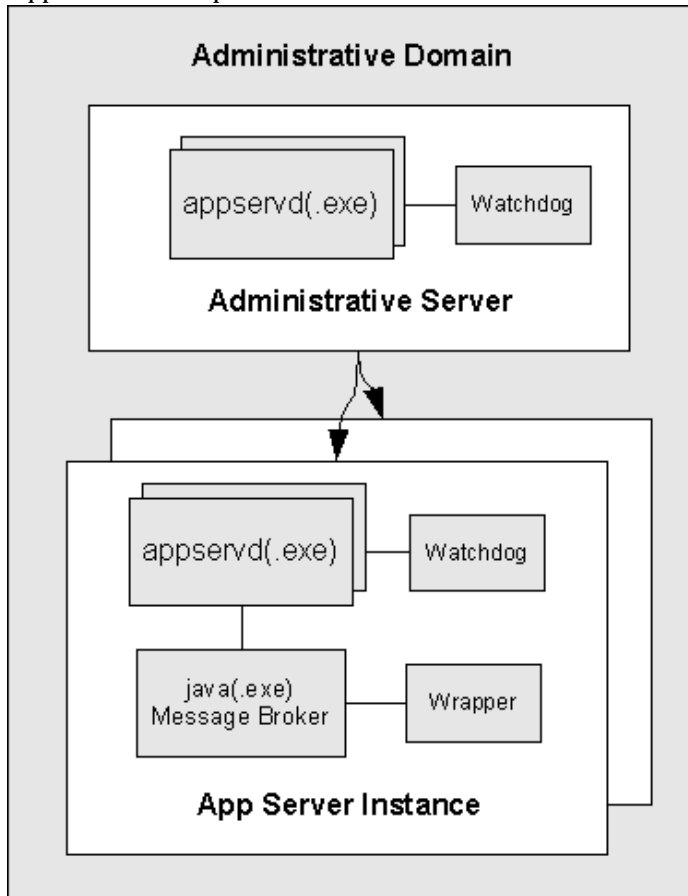
Installation Directory Structure

Depending on the distribution used to install the application server product on your system, your application server installation may either be installed under a single root directory path or be spread across several root directory paths.

*   Bundled Installation on Solaris 9

*   Package-based Installation on Solaris 8 and 9

*   Windows and Evaluation Installations

Bundled Installation on Solaris 9

On Solaris, when using the Solaris package-based installation of the product and when installing the application server as part of a Solaris 9 installation, the application server installation is spread across several root directories.

The application server included in a Solaris 9 installation is installed in the following locations:

*   `/usr/appserver` contains static portion of the installation image. All utilities, executables and libraries that make up the application server reside in this location. Only product patches and upgrades affect this area.

    Subdirectories in this location include:

    ○   `bin/` contains executables and utilities, some of which are symbolically linked from `/usr/bin`

    ○   `include/` contains legacy header files

    ○   `lib/` contains native and Java libraries

*   `/etc/appserver/` contains installation wide configuration information such as licenses and the master list of administrative domains configured for this installation.

*   `/var/appserver/domains` is the default area under which administrative domains are created.

    Since installation of the application server as part of a Solaris 9 installation does not entail initial domain creation, this directory does not exist until the user create an initial domain. As administrative domains are created, they can be placed in any location on the system. This area acts only as the default location in which domains are created.

Package-based Installation on Solaris 8 and 9

By default, when using the Solaris package-based installation of the application server, the installation locations are spread across three directory roots in a similar fashion to the bundled installation case.

- `/opt/SUNWappserver7` contains static portion of the installation image. All utilities, executables and libraries that make up the application server reside in this location. Only product patches and upgrades affect this area.

- `/etc/opt/SUNWappserver7/config` contains installation wide configuration information such as licenses and the master list of administrative domains configured for this installation.

- `/var/opt/SUNWappserver7/domains` is the default area under which administrative domains are created.

  Since installation of the application server as part of a Solaris 9 installation does not entail initial domain creation, this directory does not exist until the user create an initial domain. As administrative domains are created, they can be placed in any location on the system. This area acts only as the default location in which domains are created.

Windows and Evaluation Installations

On Windows, and in the case of installing the evaluation distribution of the application server without Solaris packages, the application server installation is rooted under a single directory path. For example:

- `c:\Sun\AppServer7` for Windows installations.

- *home_dir*`/sun/appserver7` for Evaluation installations on Solaris without the use of Solaris packages.

In these cases, the `config/` and `domains/` directories are positioned under the installation directory root.

Conventions for Referring to Directories

This guide uses the following conventions when referring to key locations within an application server installation:

- *install_dir* refers to the static portion of the install image that contains utilities, executables and libraries that make up the application server.

  This area of the file system may be read-only. Only patches and upgrades to the product affect this area of the installation.

  Examples of this directory include:

  - ❍ `/usr/appserver`
  - ❍ `/opt/SUNWappserver7`
  - ❍ *home_dir*`/sun/appserver7/`
  - ❍ `c:\Sun\AppServer7`

- *install_config_dir* refers to the location in which the license and the master list of administrative domains is managed.

  Examples of this directory include:

  - ❍ `/etc/appserver/config`
  - ❍ `/etc/opt/SUNWappserver7/config`
  - ❍ *home_dir*`/sun/appserver7/config`
  - ❍ `c:\Sun\AppServer7\config`

- *domain_config_dir* refers to the location in which administrative domains are created.

  Examples of this directory include:

  - ❍ `/var/appserver/domains`
  - ❍ `/var/opt/SUNWappserver7/domains`
  - ❍ *home_dir*`/sun/appserver7/domains`
  - ❍ `c:\Sun\AppServer7\domain`

Tools for Starting and Stopping the Application Server

In the following exercises, "starting" the application server entails starting the administrative server and the initially configured application server instance defined in the preconfigured administrative domain.

To start the application server, you can use any of the following facilities:

- Using the Command-line Interface (UNIX and Windows)
- Using the Administrative Console (UNIX and Windows)
- Using the Windows Program Group
- Using Windows Services

Using the Command-line Interface

On all operating systems, you can use the `asadmin` command-line interface to start and stop both the entire application server, a specific administrative domain, and individual application server instances. In terms of start and stop operations, the subcommands of `asadmin` listed in the following table are of interest.

Start and stop commands

Subcommand	Description
start-domain	Starts the administrative server and application server instances of the specified administrative domain.
stop-domain	Stops the administrative server and the application server instances of the specified administrative domain.
start-instance	Starts the specified application server instance. Can be run in either a local or remote mode. In local mode, execution of this subcommand does not require the administrative server to be running.
stop-instance	Stops the specified application server instance. Similar in operation to `start-instance`.

Using start-domain and stop-domain

If the application server is running, use the following command to stop both the administrative server as well as the application server instance of the initially configured domain:

```
asadmin stop-domain --domain domain1
```

where `domain1` is the name of the administrative domain defined during installation of the application server.

As the command completes, you should observe the following results:

```
asadmin stop-domain --domain domain1

Instance domain1:server1 stopped

Domain domain1 Stopped.
```

NOTE	**Windows Platforms**: On Windows platforms, a command window appears on your desktop as you start the domain. This command window is a read only representation of the application server instance's event log file content. It will remain on your desktop as long as the associated application server instance is running. As the initially configured server starts, you will see event messages appearing in the second command window. After a few seconds, you will see a message confirming that the server instance has started successfully.
	Command Window with Startup Message Does Not Appear? Since some Windows 2000 environments do not properly include the Windows net command in the environment, ensure that your system's environment has access to this command. If you cannot run the `net` command at a command prompt, then see the directions in Chapter 3, "Setting Up Your Environment" for details on correcting this problem.

Likewise, you can start the initially configured administrative domain by executing the following command:

```
asadmin start-domain --domain domain1
```

As the command completes, you should observe the following results:

```
asadmin start-domain --domain domain1

Instance domain1:admin-server started

Instance domain1:server1 started

Domain domain1 Started.
```

NOTE	**Local mode of asadmin:** For most subcommands, the `asadmin` utility requires you to specify a target administrative server as well as the appropriate administrative user name and password. However, there are several subcommands that run either in a local mode only or in either a local or remote mode. The `start-domain` subcommand does not rely on an administrative server since it responsible for starting the administrative server and starting all of the application server instances defined for the specified administrative domain. The `start-instance` and `stop-instance` subcommands can run in either local or remote modes. If you do not specify a `--user` and `--password` options for commands that can be used in a remote manner, the subcommand defaults to a local mode of operation.
	You can also explicitly force a given subcommand to run in local mode by specifying either the `--local=true` option or simply `--local`.

Since there is rarely a reason to start and stop the administrative server repeatedly in day-to-day operations of the application server, you are likely to use the instance-level start and stop subcommands more frequently than the overall `start|stop-domain` commands. The next section introduces you to starting and stopping individual application server instances from the command line.

Using start-instance and stop-instance

To stop a specific application server instance without relying on the presence of an administrative server, you can use the following command:

```
asadmin stop-instance server1
```

where `server1` is the name of the application server instance. If your environment contains more than one administrative domain, then you need to specify the administrative domain name when invoking the stop-instance command.

For example:

```
asadmin stop-instance --domain domain1 server1
```

To start a specific application server instance in local mode, you can use the following command:

```
asadmin start-instance server1
```

If you wish to start or stop an instance on a remote system, you can specify the target administrative server and administrative user name and password on the `start-instance` and `stop-instance` commands. Execute either of these subcommands without parameters to see usage information. Alternatively, you can issues the subcommands followed by the `--help` option to obtain complete usage information.

NOTE **Underlying instance-level start and stop scripts:** On both UNIX and Windows platforms, instance-level start and stop scripts exist under the directory *domain_config_dir*/domain1/*instance_name*/bin/. These scripts are named `startserv(.bat)` and `stopserv(.bat)`.

On UNIX platforms, these scripts start the application server processes directly whereas on Windows, these scripts simply start and stop the corresponding Windows services. Since the `asadmin` command is likely already available via your environment's PATH setting, you will most likely find the use of the `asadmin` `start-instance` and `stop-instance` subcommands as a more convenient means of starting and stopping instances than the underlying instance level scripts.

TIP **Relationship between administrative server and application server instances**: Although it may not be obvious, you can start an application server instance without first starting the administrative server. There is no start-up dependency between the administrative server and the associated instances. However, without the presence of the administrative server, you will not be able to perform administrative tasks against the application server instances. Nonetheless, the application server instances will correctly load applications and process requests without the presence of a running administrative server.

After launching the processes, you can perform several other checks to determine whether or not the application server has started successfully. Normally, the successful start-up message displayed in the application server's event log window is an indication of a proper startup. However, it is a useful initial exercise to try out several additional methods of checking the status of an application server startup.

In addition to monitoring the server instance event log on the desktop, you can also use either of the following approaches to determine if the server started properly:

• Accessing the Server Event Log Files

• Accessing HTTP Server of Application Server Instance

Accessing the Server Event Log Files

Although the application server instance event log files are displayed on the desktop by default, the event log of the administrative server is not displayed to the desktop by default. Normally, you won't need to monitor the output of the administrative server. However, if you need to troubleshoot a problem, you can follow these instructions to access the administrative server's event log.

1. Using Windows Explorer, navigate to the area in which the administrative event log files are held:

   domain_config_dir\domain1\admin-server\logs\

   For example:

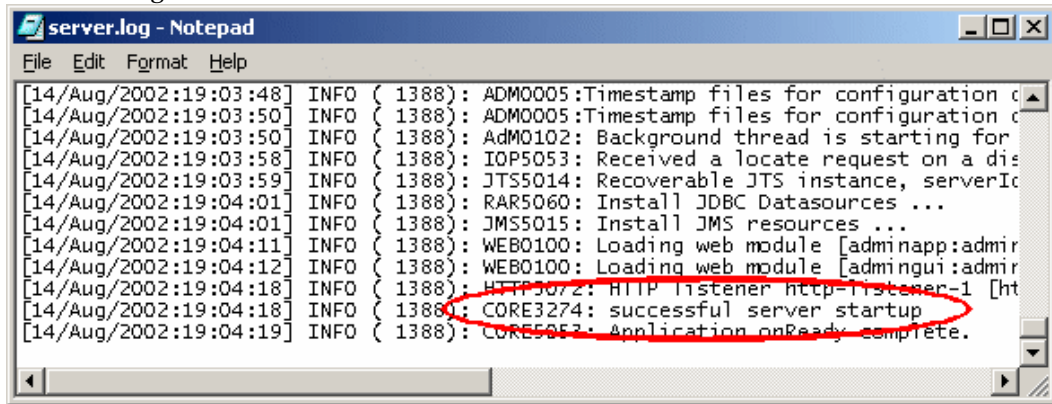   c:\Sun\AppServer7\domains\domain1\admin-server\logs\

Event logs directory
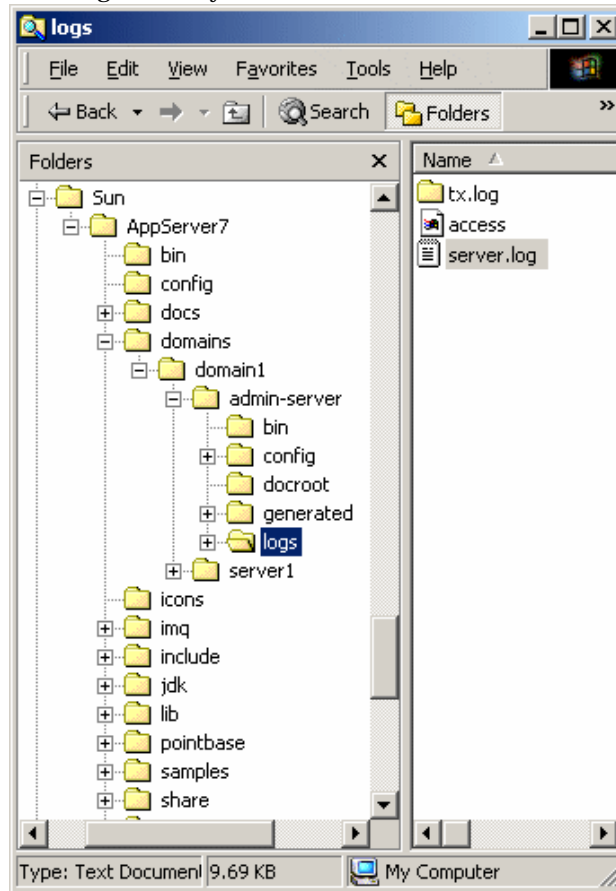
2. Double-click `server.log` to open the file in an editor.

You should find `successful server startup` at the end of the log file.

Log file

```
[14/Aug/2002:19:03:48]  INFO  ( 1388): ADM0005:Timestamp files for configuration
[14/Aug/2002:19:03:50]  INFO  ( 1388): ADM0005:Timestamp files for configuration
[14/Aug/2002:19:03:50]  INFO  ( 1388): AdM0102: Background thread is starting for
[14/Aug/2002:19:03:58]  INFO  ( 1388): IOP5053: Received a locate request on a dis
[14/Aug/2002:19:03:59]  INFO  ( 1388): JTS5014: Recoverable JTS instance, serverId
[14/Aug/2002:19:04:01]  INFO  ( 1388): RAR5060: Install JDBC Datasources ...
[14/Aug/2002:19:04:01]  INFO  ( 1388): JMS5015: Install JMS resources ...
[14/Aug/2002:19:04:11]  INFO  ( 1388): WEB0100: Loading web module [adminapp:admin
[14/Aug/2002:19:04:12]  INFO  ( 1388): WEB0100: Loading web module [admingui:admin
[14/Aug/2002:19:04:18]  INFO  ( 1388): HTTP5072: HTTP listener http-listener-1 [ht
[14/Aug/2002:19:04:18]  INFO  ( 1388): CORE3274: successful server startup
[14/Aug/2002:19:04:19]  INFO  ( 1388): CORE5053: Application onReady complete.
```

If you don't see the successful start-up message, you may have opened the event log file prior to the administrative server completing its startup procedures. Close the log file and open it again to see the very latest event messages.

In day-to-day use of the application server, you will not need to constantly check to see if the administrative server has started successfully. Rather, you will be working primarily with application server instances that are the target of your J2EE development activities.

TIP	**Domain and Instance Directories**: As you navigated to the administrative server log file, you might have noticed the domains/ directory and the associated domains1/ directory. The domains1/ directory houses the administrative domain created during product installation. Later in this guide, as you create a new administrative domain, the new domain configuration will be created by default under the domains/ directory.
	Under the *domain_config_dir*/domain1 directory, there is a directory named server1 at the same level as the admin-server directory. The server1 directory contains the configuration information, deployed applications and log data for the application server instance configured during product installation. Later in this guide, as you create additional application server instances, you will see that new directories parallel to server1 and admin-server will appear under the domains/domain1 directory. You will be able to assign the name of new instances as they are created.

Although the application server instance event log files are displayed on the desktop by default, it is useful for you to know the location of these event log files.

Navigate to the application server instance's log directory and open the server.log file:

domain_config_dir\domain1\server1\logs\server.log

Log file
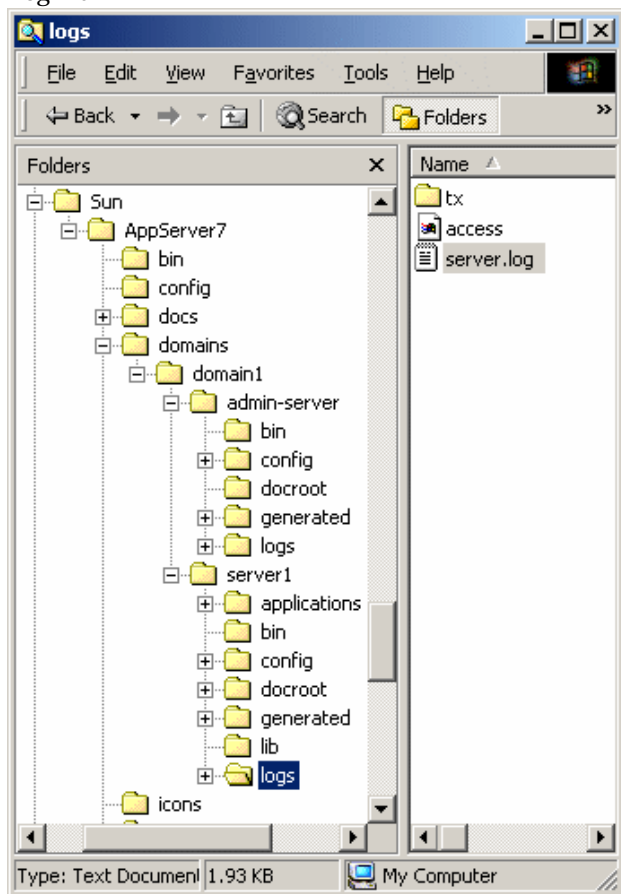
As you open the application server instance's log file, you should see the same messages that appeared the event log window on the desktop.

\

TIP	**Event log attributes**: As you view the event log file content, note the structure of the log records. In each record, you will always see a severity level followed by a unique message ID comprised of an abbreviated subsystem ID and message number. This information will help you lookup more detailed information in the log reference documentation.

TIP	**HTTP server access logs**: By default, the built in HTTP server's access log file is located in the same directory as the server's event log file. You will find the content of this file useful as you troubleshoot HTTP serving problems and/or as you want to trace the activity of HTTP requests entering the application server instances. Although the administrative server also maintains an HTTP access log file, you probably won't need to open it during normal development activities.

Accessing HTTP Server of Application Server Instance

A simple means of determining whether or not an application server instance has started is to access the instance's HTTP server listener through a web browser. After creation of a new application server instance, you will typically use this approach to quickly determine that the application server has started successfully.

Using a browser, access the following location:

`http://`*hostname*`:`*port_number*

where *port_number* is the HTTP server port number specified during installation. The default HTTP server port number is 80, but it may be different based on the ports in use during installation.

| TIP	If you do not remember the HTTP server port number of your server, you can inspect the application server instance's configuration file to determine the HTTP server port number:
	1.Navigate to the *domain_config_dir*`/domain1/server1/config/` directory and open the `server.xml` file in your favorite editor.
	2.Look for the following element:
	`http-listener id="http-listener-1" address="0.0.0.0" port="80"...`
	In this case, port 80 is the HTTP port number in use.

If the application server instance is up and running normally, you should see the following default HTTP server welcome page in your browser:

HTTP server welcome page
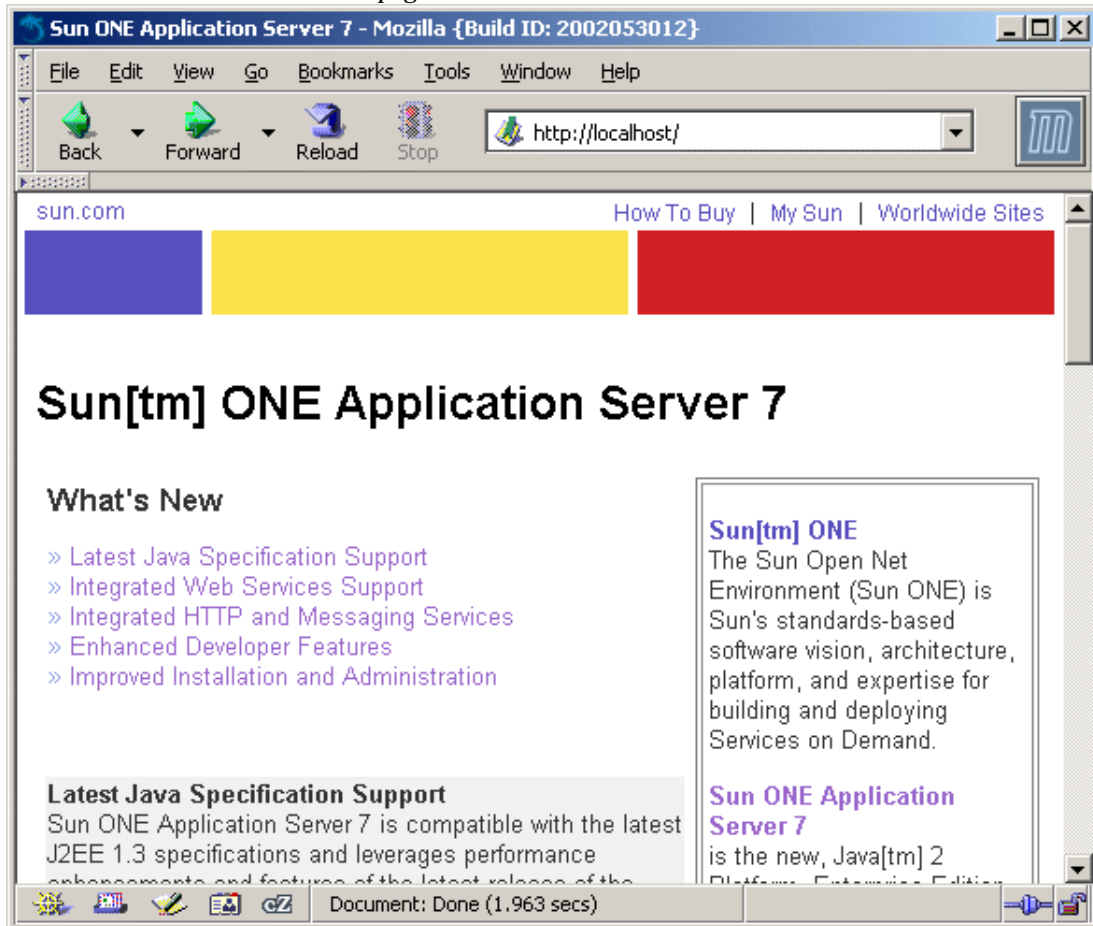
TIP	**HTTP server welcome page:** The HTTP server welcome page is simply an HTML page named `index.html` that resides under the application server instance's default document directory. The application server instance's `server.xml` configuration file contains the setting for the default document root of the instance. After installation, the document root for the instance named `server1` is set to *domain_config_dir*`/domain1/server1/docroot/`. You can find the welcome page at that location.
	Since the application server instance is also a full featured HTTP server, you can configure the server to serve static content and to support dynamic content generated by CGI executables and/or server side include files. Consults the Sun ONE Application Server *Administrator's Guide* for more information.

To completely stop the application server, access the Sun ONE Application Server menu item in the Windows program group as described in "Using the Windows Program Group" on page 66.

In the next section, you will be introduced to the use of Windows services to start the application server.

Using the Administrative Console

As long as the administrative server is running, you can also use the web-based administrative console to start and stop application server instances.

1. Start the Administrative Console.

TIP	**Compatible Browsers:** Netscape 4.79 and Internet Explorer 5.5 or higher are the officially supported browsers to access the administrative console.
	Some installations of Solaris include either Netscape 4.78 or prior versions as the default browser. The `netscape` command is executed to start the Netscape 4 browser on these systems. On Solaris 9, you can start Netscape 6, a compatible browser, by executing the following command:
	`/usr/dt/bin/netscape6`
	Although other popular browsers such as Mozilla work with the administrative console, execution of the administrative console is officially supported only with the browsers listed above.

- On Windows, the easiest means of starting the web-based administrative console is to click the Windows Start button, then select Programs, Sun Microsystems, Sun ONE Application Server 7, Start Admin Console.

  After selecting Start Admin Console, a window of your default browser will be launched with the appropriate location of the administrative server's console as set during installation of the product.

- On UNIX, open a browser window and specify the location of your the administrative server's console application.

  During installation, the default port number for the administrative server is set to 4848. If this port was already in use or you selected another port number, then specify that port number.

  For example:

  `http://localhost:4848`

2. Sign into the administrative console using the administrative user name and password supplied during product installation.

TIP   **Forgot the Admin Server port number?** If you do not remember the HTTP server port number of the administrative server, you can inspect the administrative server's configuration file to determine the HTTP server port number:

1. Navigate to the *domain_config_dir*/domain1/admin-server/config/ directory and open the server.xml file in your favorite editor.

2. Look for the element http-listener id="http-listener-1" address="0.0.0.0" port="4848"...

In this case, port 4848 is the HTTP port number in use.

TIP   **Forgot the user name or password?** If you do not remember the administrative user name that was supplied during installation, try the user name admin. This is the default user name specified in the server configuration dialog during installation.

If you still cannot determine the user name, look in the following file:

domain_config_dir/domain1/admin-server/config/admpw

This file contains the administrator's user name followed by the encrypted form of the administrative user's password.

If you do not remember the administrator's password, then you will need to uninstall and reinstall the application server.

If you do not remember the administrator's password, then you can delete the administrative domain using the delete-domain subcommand of asadmin and create a new domain with a new administrative password.

TIP   **Port not accessible?** If the connection was refused when attempting to contact the administrative server's admin console application, it is likely that the administrative server is not running. Go back to the beginning of this section and check your start-up procedures and the content of the administrative server's log file to determine the reason why the server is not running.

Once you've authenticated successfully, the initial screen of the administrative console appears:

Administrative console
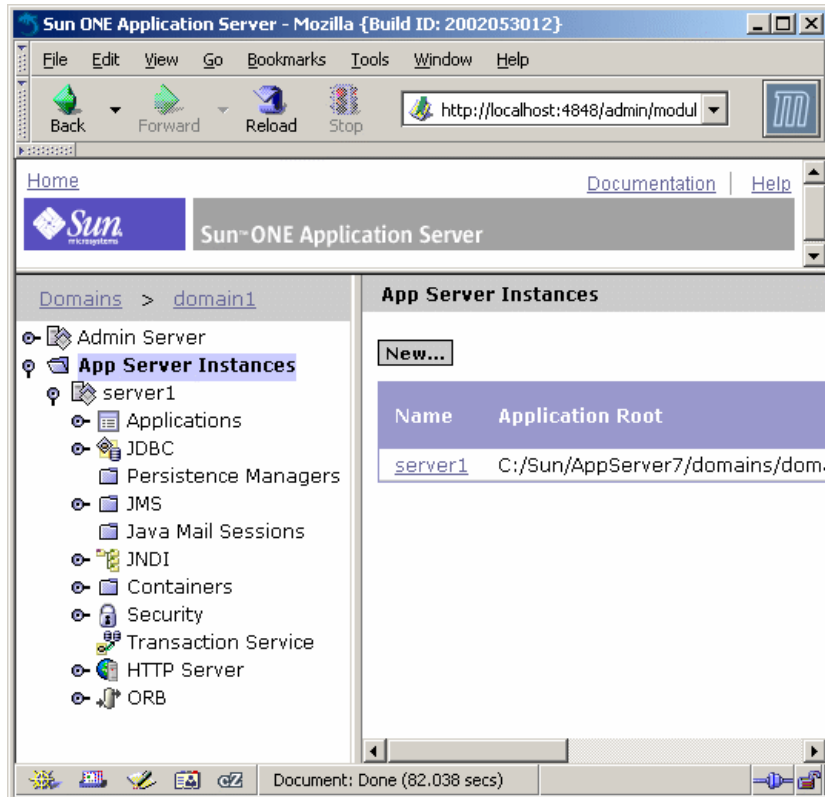
NOTE	**Admin Server and App Server Instance Nodes**: In the administrative console, note the presence of an Admin Server node and an application server instance node named `server1`. These two nodes correspond to the two directories under the directory *domain_config_dir*/`domain1/`. The administrative console's main function is to provide you with an easy-to-use means of managing server configuration information as maintained in each of the instance's `config/server.xml` file.

3. Select the `server1` node to access the start and stop functions.

Server 1 node
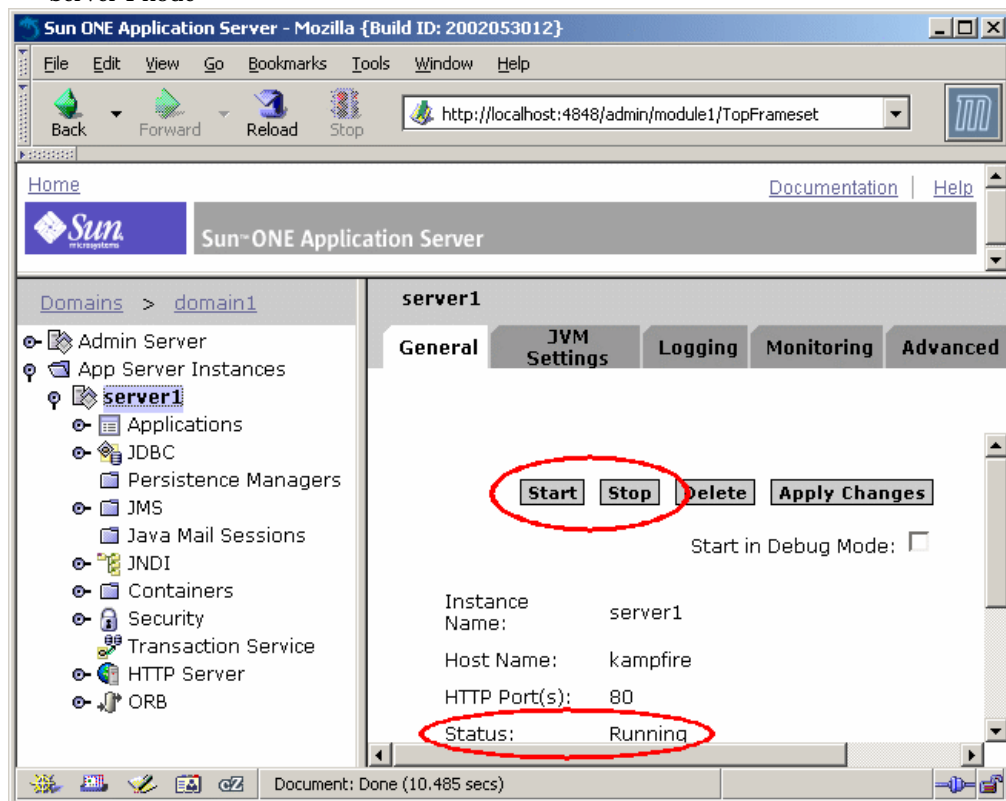
Note the status of the application server instance. It is either in a "running" or "not running" state.

4. Depending on the server instance's state, click either Start or Stop to start or stop the application server instance.

Proceed to Chapter 5, "Becoming Familiar with the Sample Applications" for a brief introduction to the sample application prior to deploying and exercising it.

Using the Windows Program Group

On Windows, the easiest means of starting the entire application server is to access the Windows program group.

1. Click the Windows Start button, then select Programs, Sun Microsystems, Sun ONE Application Server 7, Start Application Server.

    After selecting Start Application Server, you will first see a command window in which the status of the startup process is displayed.

    A second command window also appears on your desktop. This command window is a read only representation of the application server instance's event log file content. It will remain on your desktop as long as the associated application server instance is running. As the initially configured server starts, you will see event messages appearing in the second command window. After a few seconds, you will see a message confirming that the server instance has started successfully:

    ```
    INFO: CORE3274: successful server startup
    ```

NOTE	**Second Window with Startup Message Does Not Appear?** Since some Windows 2000 environments do not properly include the Windows `net` command in the environment, ensure that your system's environment has access to this command. If you cannot run the `net` command at a command prompt, see the directions in "Setting Up Your Environment" on page 37 for details about how to correct this problem.

The application server instance command window displays the startup status as shown in the following graphic. Note the `server1` designation in the title of the window. This is the name of the application server instance.

Application server instance command window

NOTE   **Behind the scenes:** Selecting Start Application Server from the Windows program group results in the execution of the `asadmin` command-line interface with the subcommand `start-appserv`. This subcommand starts both the administrative server and the initially configured application server instance. Later, as you add more application server instances, Start Application Server will also start the newly added instances.

As the initially configured server starts, it displays the event log content to a command window on your desktop. Whether or not an application server instance displays its event log information to the Windows desktop is configurable on an instance-by-instance basis. You will learn how to disable this feature later in this guide.

---

2. As the application server is starting up, start the Windows Task Manager to see the processes as they complete their start-up sequence.

    You do not need to perform this task in the course of your everyday interaction with the application server. This exercise is presented only to help you understand the behind-the-scenes operation of the application server.

To start and view the Windows Task Manager, perform the following steps:

a. Right-click an empty area of the taskbar, then click Task Manager.

You can also start Task Manager by either pressing CTRL+ALT+DELETE and then clicking Task Manager or by pressing CTRL+SHIFT+ESC.

b. Once Task Manager starts, select the Processes tab to view all processes running on the system.

c. Click on the column title Image Name to sort processes in alphabetical order.

In the following window, note the six processes that make up the initially configured application server environment:

Application server processes
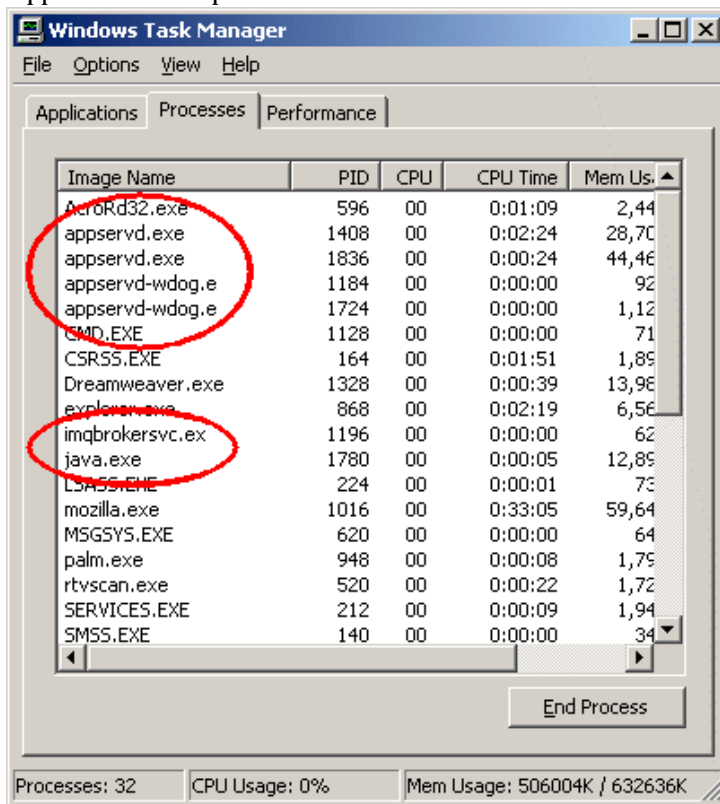
Using Windows Services

After installation of the application server, several Windows services are defined to control the startup and shutdown of the administrative server and the initially defined server1 application server instance. This section describes how to use these services to control the application server processes.
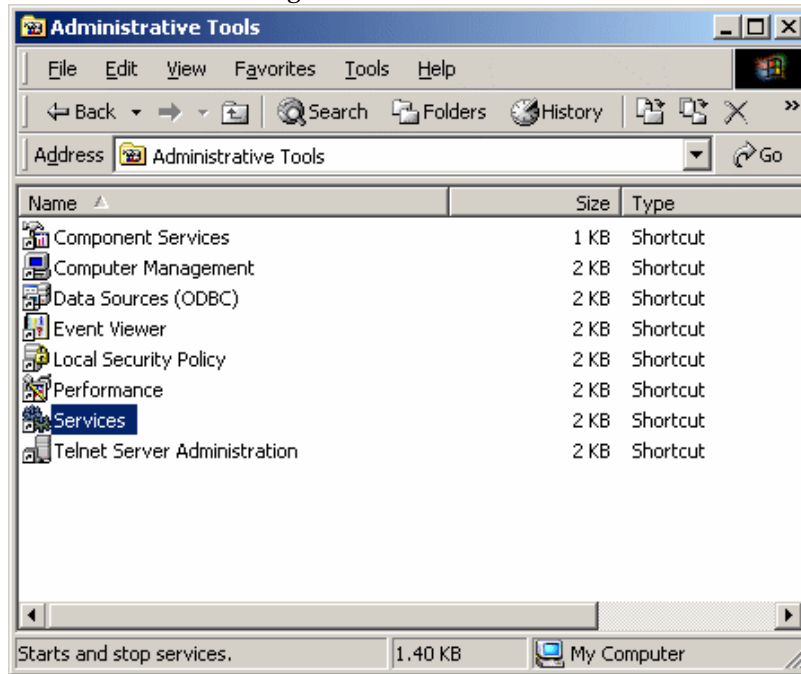
1.   Click the Windows Start button, choose Settings, then Control Panel.

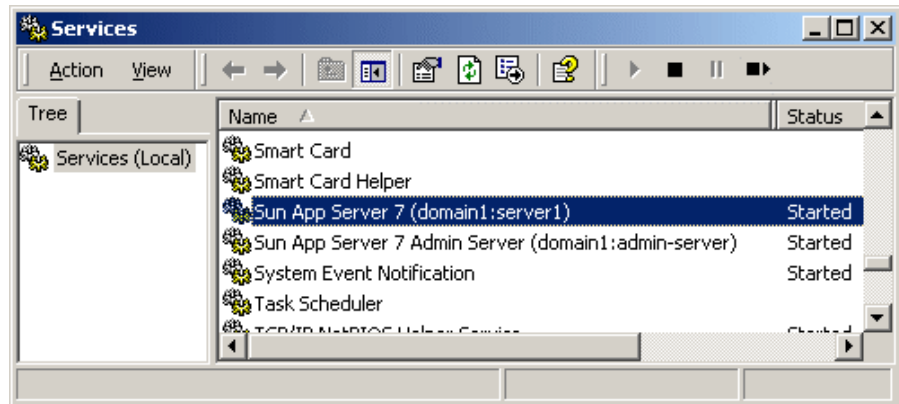2.   As the Control Panel is displayed, double-click Administrative Tools.

Control panel
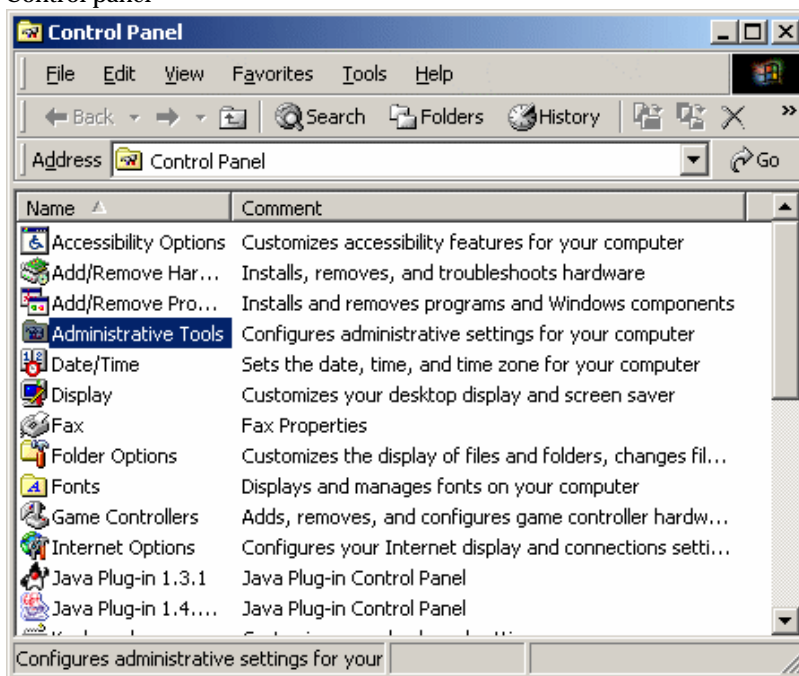
3. Double-click Services to view the services installed on your system.

The complete list of services defined on your system is displayed.

Administrative tools dialog box

4. Scroll down to view the Sun Application Server entries.

Services window

Note that there are separate Windows services for the administrative server and the initially configured application server instance, server1. As long as the administrative server and application server instance are still running from the previous exercises, the Status column of each service entry should be represented as "Started".

5. As an exercise, start the Windows Task Manager as described in Step a on page 68 to view the currently executing processes on your system.

6. Within the Services window, select the Sun Application Server 7 Admin Server entry, right-click it, and select Stop.

Now monitor the Task Manager Windows to see the underlying processes terminate.

7. Follow the same procedure for the application server instance named server1.

As you stop the server1 application server instance, the event log window will disappear from the desktop.

8. Once the administrative server and application server instance have been stopped, start them again by using the Windows services: Instead of using the Stop command when right-clicking a service, use the Start command.

You may find it useful initially to monitor the startup of the underlying processes via the Windows Task Manager.

Note that the event log window for the server1 application server instance appears on the desktop as soon as the corresponding service is started.

TIP	**Creation of Windows services**: Each time you create or delete an application server instance, a corresponding Windows service is also created or deleted. After creation, these Windows services are always set to the Manual Startup Type. This means that after restart of Windows, these services will not be automatically started. Depending on your needs, you may decide to change the Startup Type to Automatic such that the underlying application server processes are started during restart of Windows by performing the following steps:
	1. Select one of the services, right-click it, and select Properties.
	2. Under Startup type, select Automatic, then click Close.

Next you will use the `asadmin` command-line interface to start and stop the application server.

Becoming Familiar with the Sample Applications

The sample application that you will be using during this guide consists of a web application module and an EJB module containing a single stateless session bean. The web and EJB modules are packaged in an Enterprise Application Archive (EAR) file. Although patterned after more basic "Hello World" applications, this sample displays a greeting based on the user's input and the time of day. Since each greeting is recorded to a database table, the user has the option to list all previously generated greetings as well.

The display greeting path through the application is represented by the numbered steps (1, 2, 3, 4, and 5) in the following diagram, while the greeting log display path through the application is represented by the lettered steps (1a, 2a, and 3a) in the diagram.

Sample application
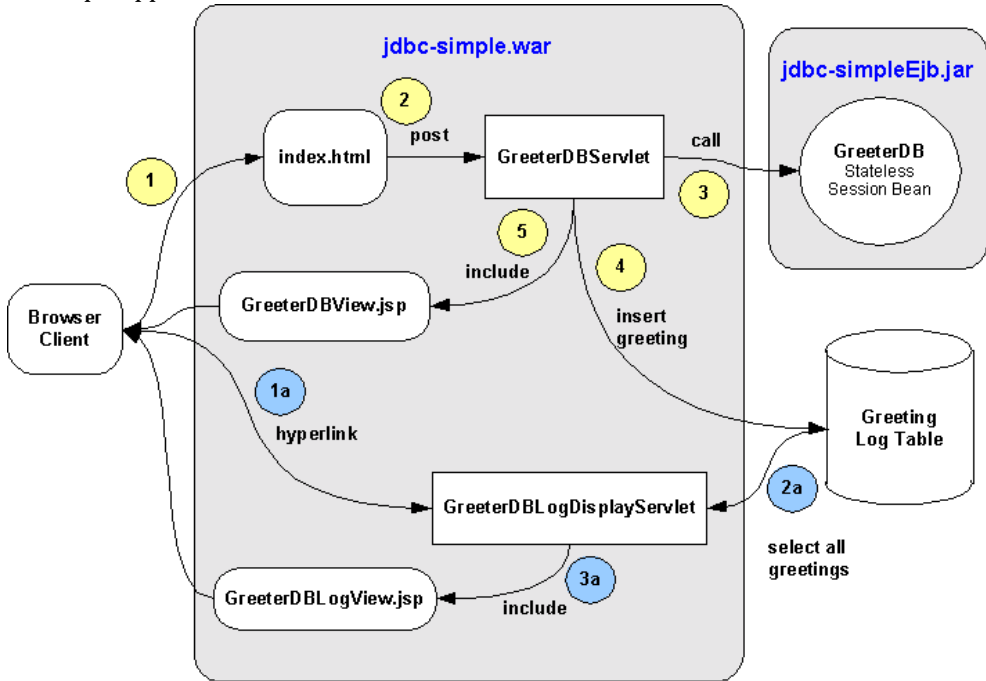
When the user accesses the application, the first web page of the application prompts the user to enter a name that will be displayed in a subsequent greeting page.

Welcome screen
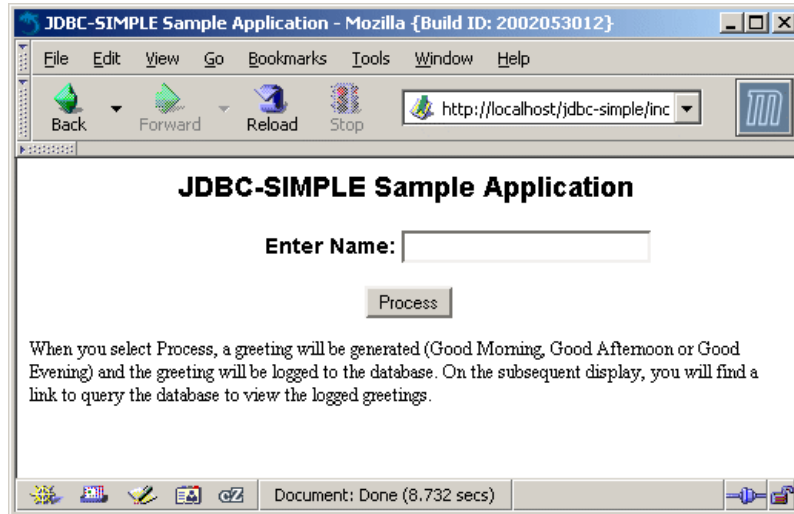
After entering a name and clicking on the Process button, the following page is displayed with the greeting. In the background, the GreeterDBServlet accessed the stateless session bean to determine the appropriate greeting ("morning", "afternoon" or "evening").

The GreeterDBServlet sets the appropriate greeting message in the request object of the servlet request and specifies a JSP to display the result.
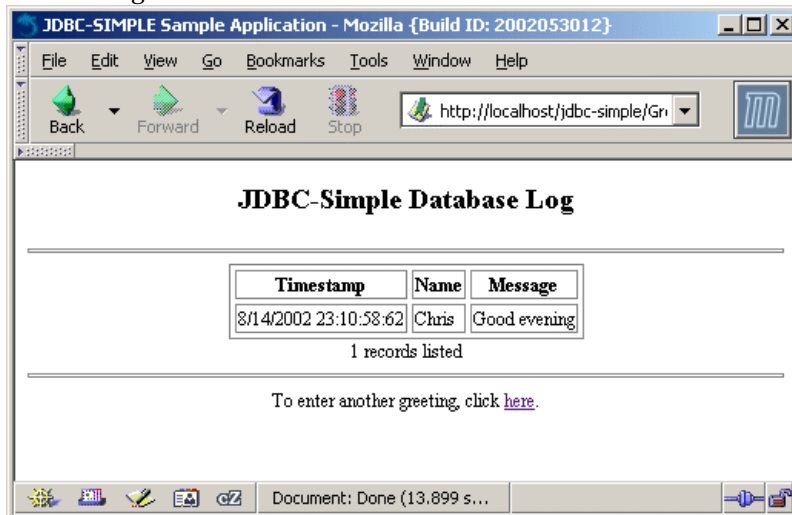
Greeting
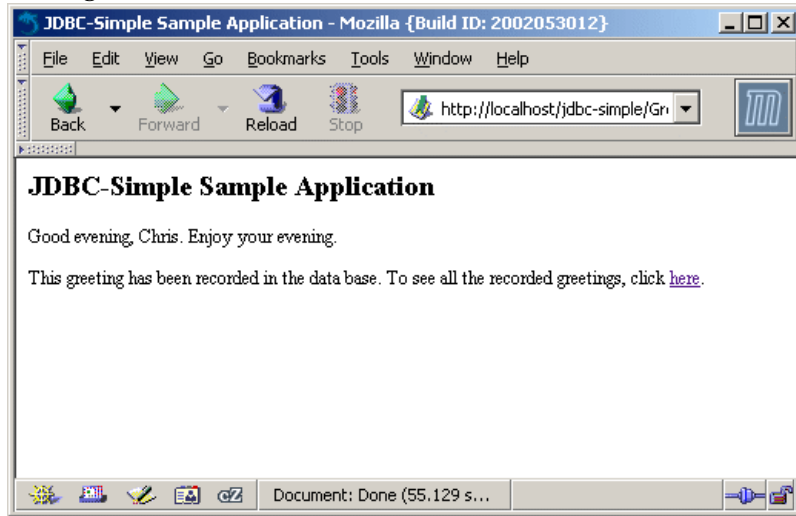
The greeting is displayed.

When the user elects to list all previously generated greetings, the
GreeterDBLogDisplayServlet performs a select all on the database table of
greetings and specifies a JSP to display the result.

Database log

Proceed to Chapter 6, "Setting Up Database Connectivity" to configure the application server's database setup to support deployment and execution of the sample.

Setting Up Database Connectivity

The sample application uses the Java™ Database Connectivity (JDBC™) API to record greetings to a database. In this section you will define the necessary JDBC-related settings in the application server environment prior to deploying and exercising the sample.

JDBC configuration involves setting up a JDBC driver, defining a JDBC connection pool and registering the JDBC resources used by your application. By default, the sample application used in this guide is configured to work with the PointBase Server database product. All distributions of Sun ONE Application Server 7 except for the application server that is installed as part of Solaris™ 9 include the PointBase Server database product. If the directory *appserver_install_dir*/pointbase/ exists in your application server installation, then PointBase has already been installed and you can proceed with the following instructions to define the JDBC connection pool and resource objects.

If the PointBase Server product has not been installed as part of your application server installation, you can follow the steps in the section "Installing and Configuring PointBase Server," on page 97 to set up a PointBase environment before proceeding with the JDBC connection pool and resource set up steps.

The following topics are included in this chapter:

*   Configure the JDBC Driver

*   Define the JDBC Connection Pool

*   Define JDBC Resource

*   Start the PointBase Server Database

Configure the JDBC Driver

A JDBC driver must be configured in the classpath of the application server before you can exercise applications that use JDBC.

Since the PointBase Type 4 JDBC driver should already be configured in your application server instance's environment (based on either the combined installation of the application server with PointBase or through your installation and configuration of PointBase), there is no need for you to perform any additional JDBC driver set up steps.

TIP	**Adding JDBC driver to classpath**: You can easily add a JDBC driver to the application server's classpath through the administrative console. 1. In the console, select the server instance node, the JVM Settings tab, or the Path Settings sub tab. 2. In the Path Settings tab, modify the Classpath Suffix setting to reference the appropriate JDBC driver `.zip` or `.jar` file. If you are using a Type 2 driver, also modify the Native Library Path Suffix setting to reference the appropriate native library. 3. Save your changes and go back to the General tab of the server instance. 4. Click Apply Changes and then restart the application server to pick up the changes. Alternatively, you can simply drop a Type 4 driver's archive in the following directory: *domain_config_dir*`/domain1/server1/lib/` All Java libraries that are present in the instance's `lib` directory are automatically added to the end of the server's classpath during server instance restart.

TIP	**Preconfigured PointBase Database**: To learn more about how the sample applications use PointBase, see the Sample Applications documentation.

Define the JDBC Connection Pool

Before running the sample application, you need to define a suitable JDBC connection pool that maps to the PointBase database server and a JDBC resource that associates the JDBC references made in the sample application to the JDBC connection pool definition.

NOTE	**Server instance configuration**: Resources used by J2EE applications and other server settings are maintained in the `server.xml` file under:
	domain_config_dir/domain1/server1/config/
	You can manually modify this file, but keep in mind that when doing so, other users could be updating the file through the either administrative console or the `asadmin` command-line interface. Manual modifications to the `server.xml` file require a restart of the application server instance in order to make the changes take effect.

1. Through a web browser, access the administrative console. (You might have to start the application server if it is not running).

2. Expand the JDBC node under the application server instance named `server1`. Then click the Connection Pools node and click New to define a new connection pool.
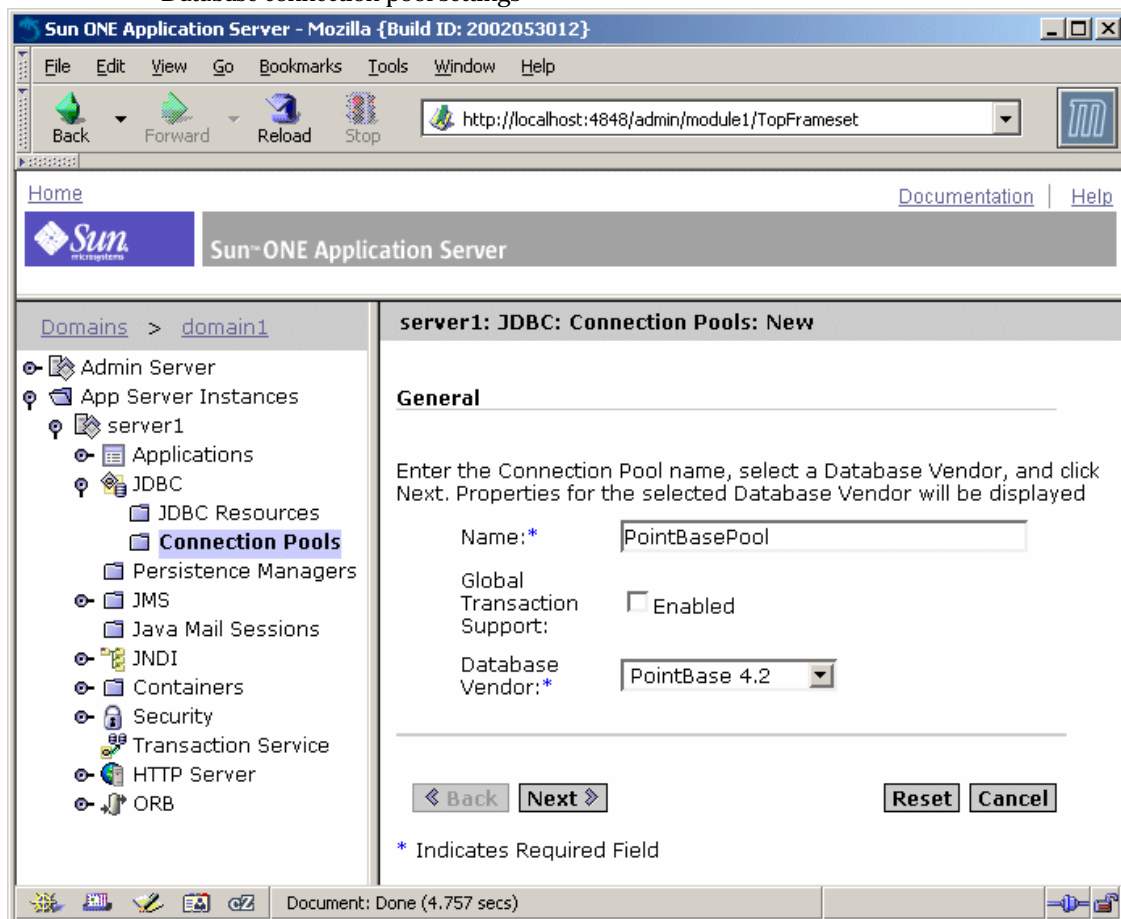
Data sources node
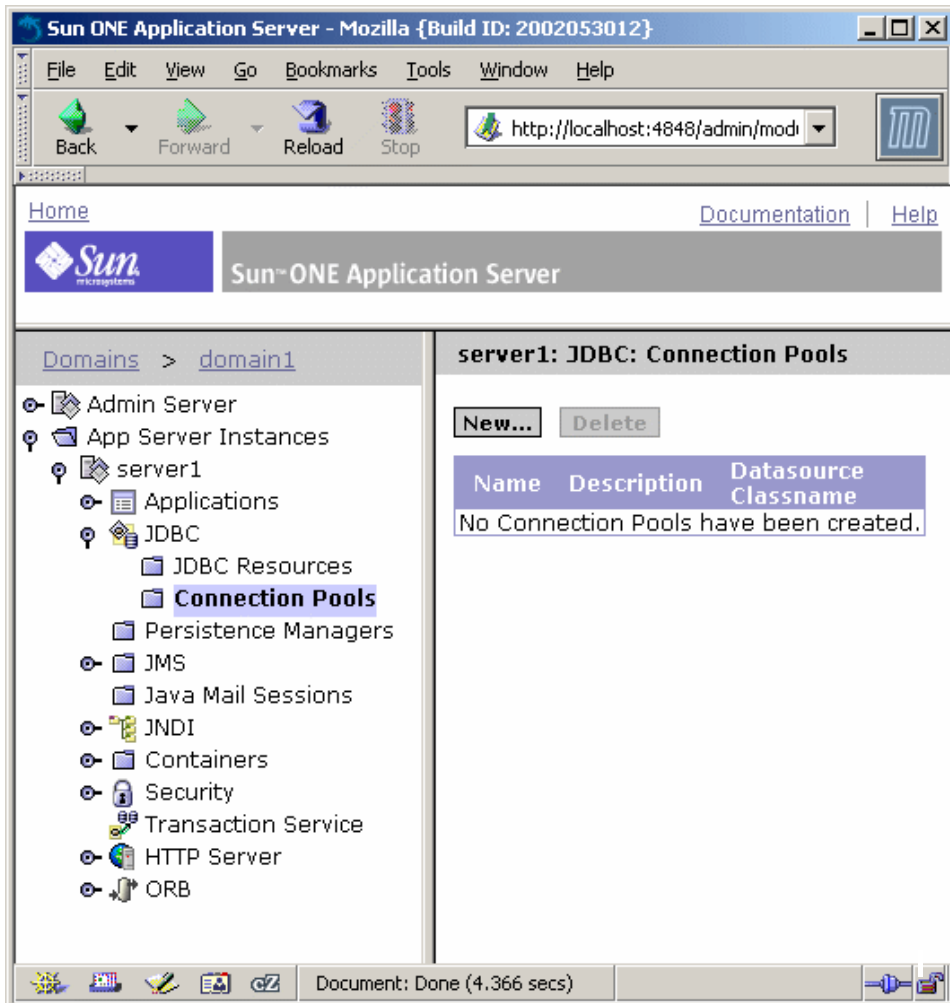
3. Enter the database property settings shown in the following table in the first screen of the create new connection pool wizard and then click the Next button.
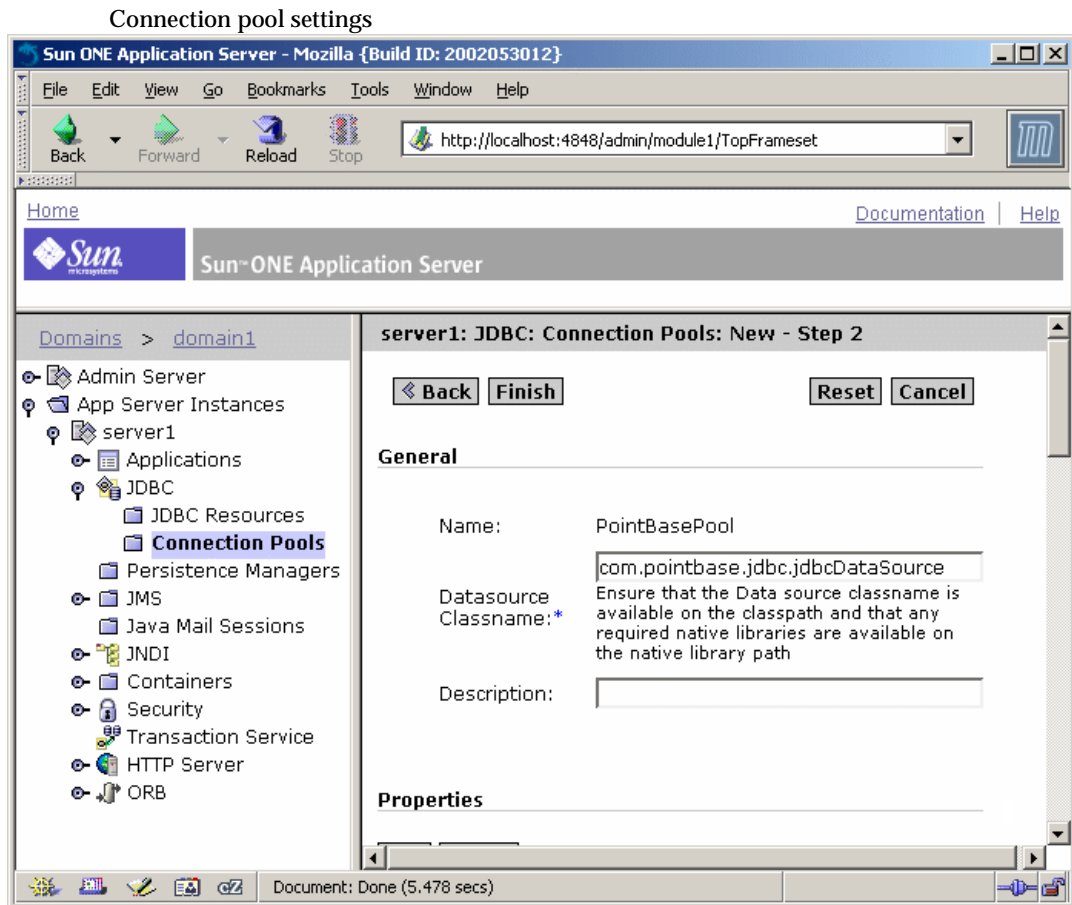
Database property settings

Field	Value
Name	PointBasePool
Database vendor	select: PointBase 4.2

Database connection pool settings

The following screen appears:

Connection pool settings

4. Scroll down to the Properties section and specify the database property settings shown in the following table:

Database property settings

Property	Value
user	`jdbc`
password	`jdbc`
databaseName	`jdbc:pointbase:server://localhost/sun-appserv-samples`

NOTE	**Connection Pool Properties and Using Other Databases**: Although the `databaseName` property is supported by PointBase Server, it is not supported by all other databases. For example, for an Oracle RDBMS, use the `URL` property name to specify the URL to be used by the JDBC driver to connect to the Oracle RDBMS. Consult your JDBC driver's documentation for details on exactly which properties are supported. For an example of using an Oracle JDBC driver with the sample, see the note on "Using an Oracle Type 4 JDBC Driver" in "Define JDBC Resource" for details.

If you see any additional properties in this list, select the checkbox to the left of each of these properties and click the Delete button to remove these unused properties.

Connection pool properties
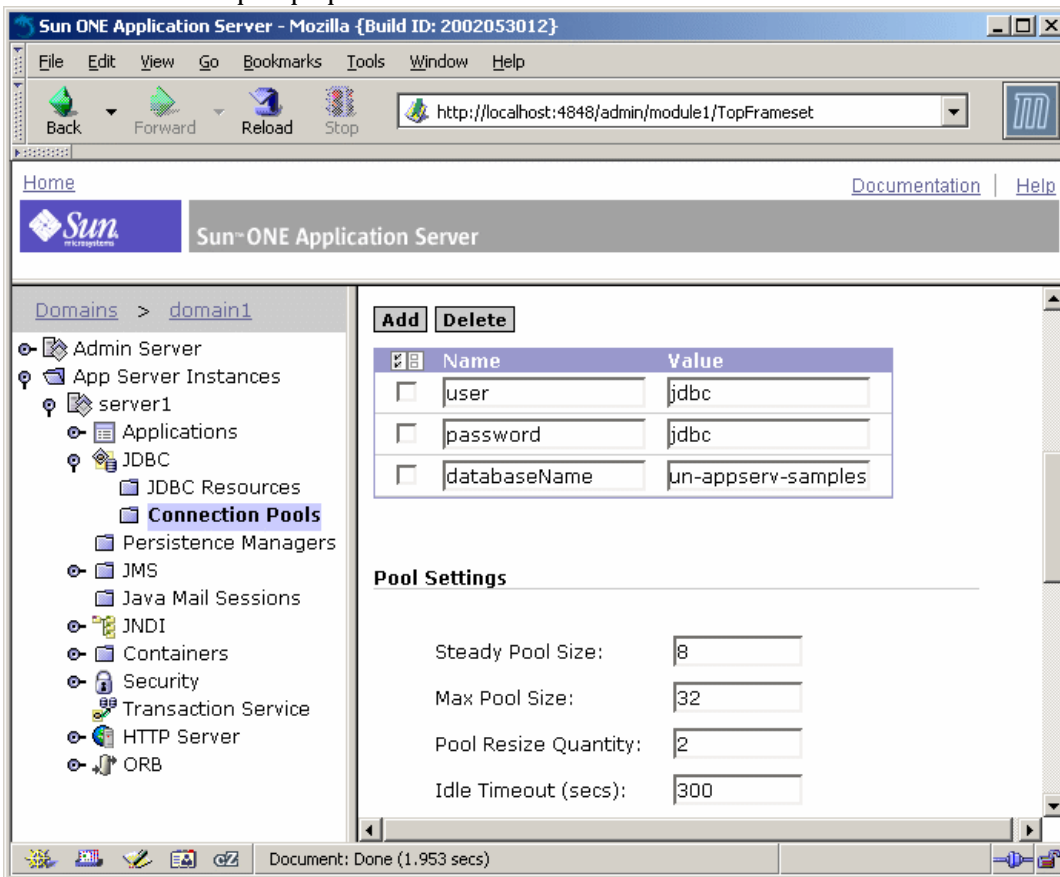
5. Click the Finish button to complete the wizard.

As a result of completing the JDBC connection pool wizard, the administrative server recognizes that although changes have been made to the server's configuration, the changes have not yet been applied to the active configuration. Consequently, several warning icons are displayed in the administrative console to inform you that pending changes need to be applied before they take effect.

Before you apply these changes, proceed to the next step to define the JDBC resource for the sample. After defining the JDBC resource, you will apply both the outstanding connection pool and JDBC resource changes at the same time.
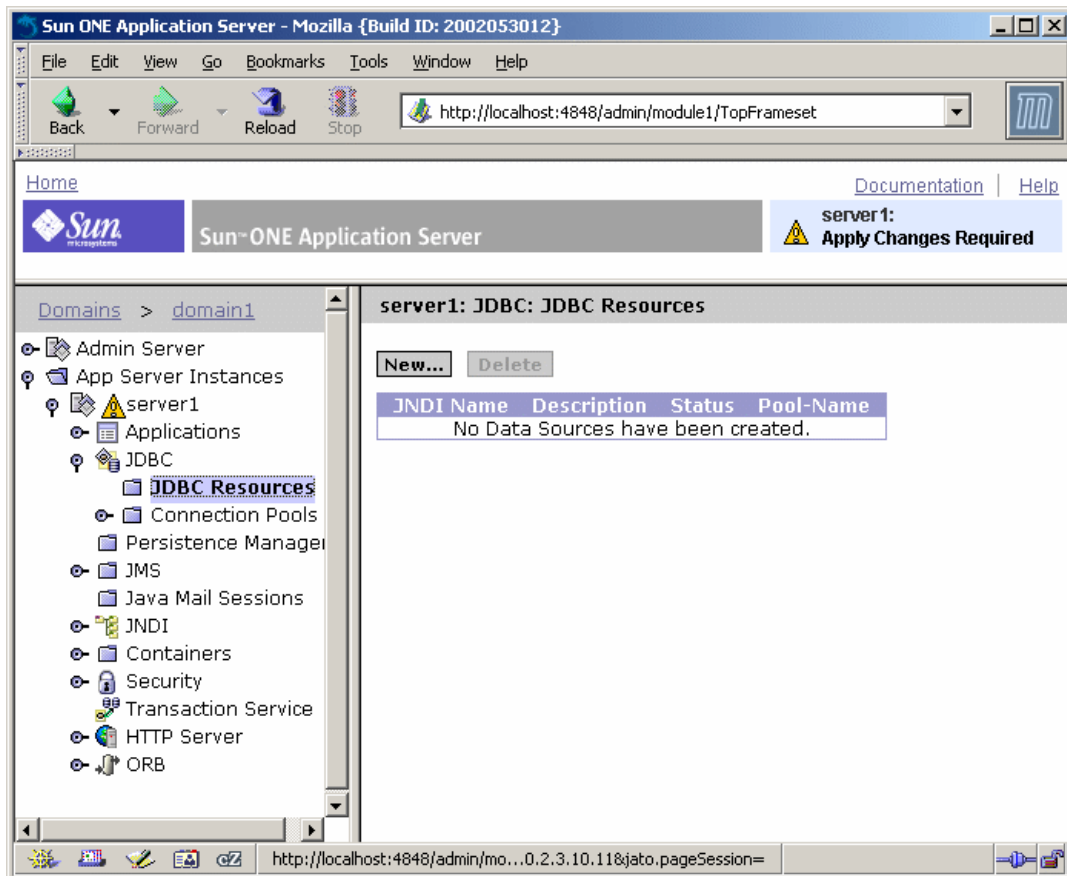
Apply changes
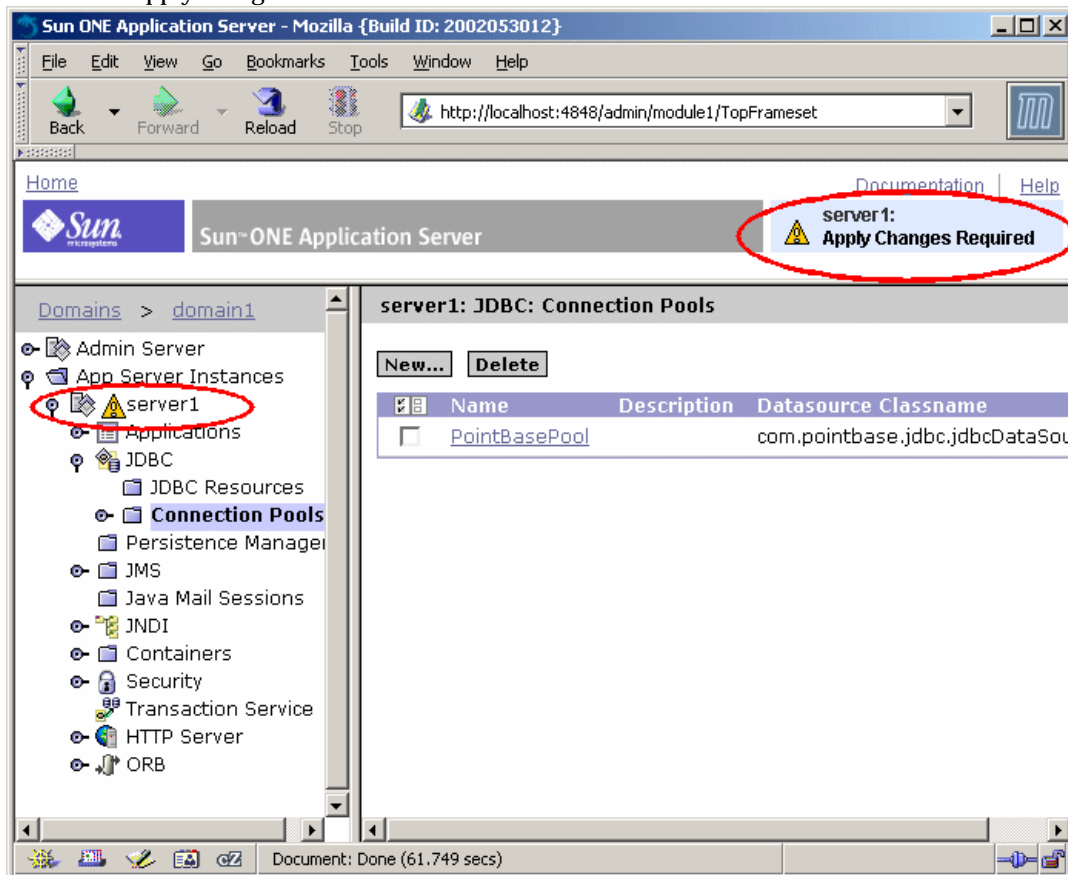
Define JDBC Resource

Now that the JDBC connection pool definition has been created, you are ready to define a JDBC resource and associate it with the connection pool entry.

1. Under the JDBC node, click the JDBC Resources node and click New... to define a new resource entry.

Data sources node

Since the sun-web.xml file of the sample web application refers to the jdbc/jdbc-simple JNDI name and a JDBC resource of that name does not yet exist in the server configuration, you need to define a new JDBC resource.
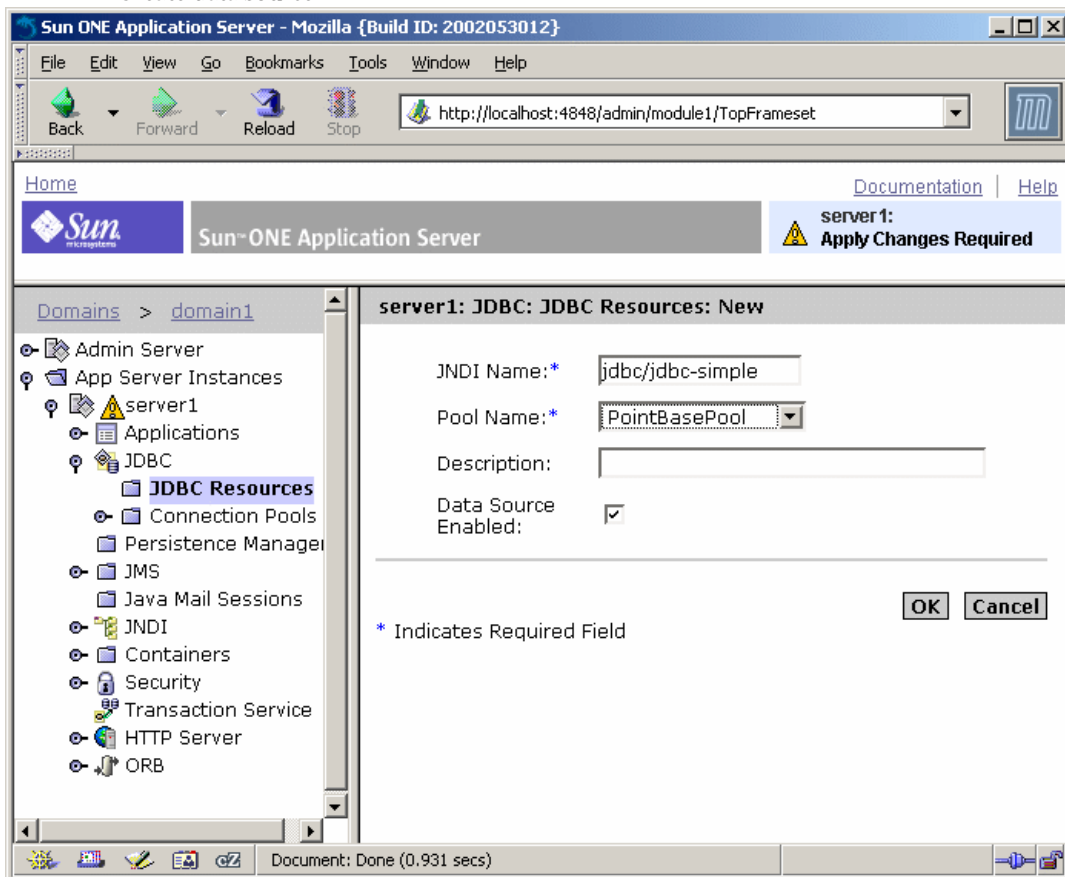
2. Set the fields to the database property settings shown in the following table and click OK to define the resource.

Database property settings

Field	Value
JNDI Name	`jdbc/jdbc-simple`
Pool Name	`select: PointBasePool`

Create data source

3. Now that you have defined both the JDBC connection pool and resource for the sample application, you are ready to apply the changes so to make the application server instance aware of the changes:

   a. Select the server1 node.

   b. Click Apply Changes to apply the outstanding JDBC connection pool and resource changes.

Apply changes
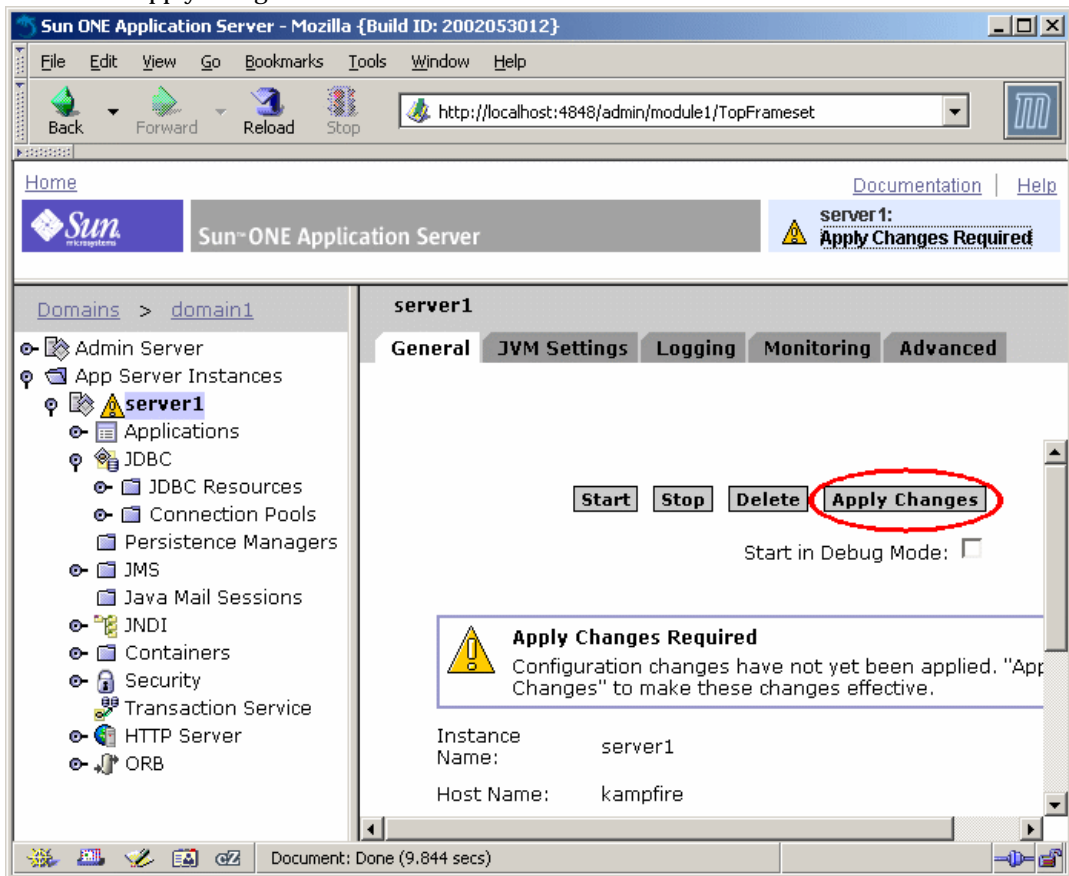
A message is displayed stating that the changes have been applied. Note that a server instance restart is not required in this case.

NOTE	**What Happens During Apply Changes?** When you either define new objects such as JDBC connection pool and resources or modify such objects, the administrative server saves the configuration settings to a server.xml file in a backup directory under the application server instance's configuration area. For example:

domain_config_dir`/domain1/server1/config/backup/server.xml`

As you continue to make and save changes, the changes continue to be made to the server.xml file in the backup area.

In order to make pending changes visible to the application server instance, you must Apply Changes to force the administrative server to copy the server.xml file being maintained in the backup area to the live location under:

domain_config_dir`/domain1/server1/config/server.xml`

Although some changes that are applied to the configuration area are picked up immediately by the application server instance, other changes require a server instance restart in order for the changes to become effective. The administrative console notifies you when such changes require a server restart.

NOTE	**How would one use an Oracle Type 4 JDBC driver with the application?** How would one use an Oracle Type 4 JDBC driver with the application? Using another RDBMS is simple:
	1. Use the supplied SQL file to create the table in an Oracle RDBMS:
	install_dir/samples/jdbc/simple/src/sql/jdbc-simple-ora.sql
	2. Add the Oracle Type 4 driver to the server classpath by either:
	a) simply dropping the driver library into the application server instance's lib/ directory:
	domain_config_dir/domain1/server1/lib/
	All libraries in this directory are automatically added the end of the server's classpath during restart.
	or b) adding the driver library to the Classpath Suffix under the application server instance's JVM Settings->Path Settings area.
	3. Create a JDBC Connection Pool object for the Oracle Type 4 driver:
	Datasource Classname = oracle.jdbc.pool.OracleDataSource
	Connection Pool Properties:
	URL: jdbc:oracle:thin:@$ORACLE_SERVER:1521:$ORACLE_SID
	where $ORACLE_SERVER and $ORACLE_SID must be replaced with the appropriate value for the target database. For example:
	jdbc:oracle:thin:@localhost:1521:orcl
	User: $USER (set as appropriate for your database)
	Password: $PASSWORD (set as appropriate for your database)
	4. Remap the jdbc/jdbc-simple JDBC resource from the PointBasePool connection pool to the newly defined Oracle connection pool.
	5. Apply the changes and restart the application server.
	6. Rerun your application.

Start the PointBase Server Database

Before exercising the sample application, you must start the PointBase Server database.

If you are either sharing your application server installation with other users or your system user ID does not have write permissions to the area in which PointBase is installed (which is typically the case when the application server is installed by the root user on UNIX systems and you are using a non-root user ID), you should make a copy of the pre-built PointBase databases and the PointBase Server startup script.

If you are not sharing the PointBase installation with other users, proceed to "Start the PointBase Server Database," on page 93.

This section contains the following topics:

- Making Your Own Copy of the Database
- Starting the PointBase Server

Making Your Own Copy of the Database

You can create your own copy of the PointBase Server environment by following these steps:

1. Create a directory named `pointbase/` under a directory in which you have write permissions.

2. Copy the prepopulated PointBase database files from the samples installation area to your own PointBase area.

To copy the prepopulated database files to your own PointBase installation, copy the files contained in the following directory:

appserver_install_dir`/samples/pointbase/databases/*`

to:

personal_pointbase_dir`/pointbase/databases/`

This operation will copy the files named `sun-appserv-samples$1.wal` and `sun-appserv-samples.dbn` to the your own PointBase area.

3. Copy the following files from the PointBase installation area to *personal_pointbase_dir*/pointbase/ directory:

File	Comments
StartServer.sh or StartServer.bat	Start up script containing port number on which the PointBase server listens.
pointbase.ini	Initialization file specifying location of database files.

If PointBase was installed as part of the application server installation, these files are located at:

appserver_install_dir/pointbase/server/

If you downloaded an installed the evaluation distribution of PointBase, these files are located at:

pointbase_install_dir/tools/server/

4. Edit the StartServer.sh or StartServer.bat script to specify a port number that does not conflict with other running instances of the PointBase server.

| TIP | **Using a Non-default Port Number**: If you change the port number on which the PointBase Server listens from the default value of `9092`, you must modify the JDBC Connection Pool properties to reflect the non-default port number. Otherwise the application server will not be able to contact your copy of the PointBase Server. |

The `databaseName` property specified for the JDBC Connection Pool must be changed from:

`jdbc:pointbase:server://localhost/sun-appserv-samples`

to:

`jdbc:pointbase:server://localhost:`*port*`/sun-appserv-sample
s`

where *port* is the port number specified in the StartServer script.

To add the port number to the databaseName property, follow these steps:

1.Select the PointBasePool node in the application server administrative console.

2.Scroll down and click the Properties... button to edit the connection pool properties.

3.Modify the `databaseName` property value to reflect the appropriate port number and click OK to save the changes.

4.Apply the changes and restart the application server instance to make the connection pool changes take effect.

Now you are ready to start the PointBase Server from your own directory area.

Starting the PointBase Server

The database server can be easily started by performing one of the following actions:

• On Windows:

PointBase installed with the application server:

Start->Programs->Sun Microsystems->Sun ONE Application Server 7->Start PointBase

PointBase downloaded and installed separately:

Execute: *pointbase_install_dir*/tools/server/startserver.bat

- On UNIX:

  If PointBase was installed with the application server, change to the directory:

  install_dir/pointbase/server

  Execute the script:

  ./StartServer.sh

  If PointBase was downloaded and installed separately:

  pointbase_install_dir/tools/server

  Execute the script:

  ./startserver.sh

Once you execute this script, you will see the following text in either a command window or at the UNIX terminal prompt:

```
Server started, listening on port 9092, display level: 0 ...

>
```

To stop the server, enter quit at the prompt.

TIP   **Don't Need A Console Window?** On UNIX, if you want to run the PointBase server in the background, you can edit the StartServer.sh script and place the option /noconsole at the end of the command line.

```
.../java -classpath ./lib/pbserver42RE.jar
com.pointbase.net.netServer /port:9092 /noconsole
```

Use & at the end of the StartServer.sh command to run the server in the background.

```
./StartServer.sh &

[1] 26418

root@canteloupe-{/opt/appserverBeta/pointbase/server}:
```

See the PointBase server documentation in the following location for more information on starting and stopping the server:

pointbase_install_dir/docs/

Proceed to Chapter 7, "Deploying and Running the Sample Application" to exercise the application.

Installing and Configuring PointBase Server

PointBase is not installed when either the application server is installed as part of a Solaris 9 installation or, during installation of the application server, the PointBase component was deselected as a component to be installed.

To determine if PointBase is installed and configured to work with sample applications, check your application server installation for the following directory:

appserver_install_dir/pointbase/

If this directory exists, then PointBase has already been installed as part of the application server installation. Continue with the database set up steps at the beginning of this document, starting with "Configure the JDBC Driver," on page 80.

To install and configure PointBase for use with the sample applications, follow these steps:

1. Download and Install PointBase Server and Client Products

2. Copy Samples PointBase Database Files

3. Add PointBase Type 4 JDBC Driver to Server Application Server's Classpath

1. Download and Install PointBase Server and Client Products

1. Download the PointBase evaluation software from http://www.pointbase.com.

2. Install at least the PointBase Server and Client products on your system.

   The PointBase Client product includes the PointBase Type 4 JDBC driver.

2. Copy Samples PointBase Database Files

The samples component of Sun ONE Application Server 7 contains a prepopulated PointBase database file for the sample applications. If you do not intend to share your PointBase installation with other users, you must copy the prepopulated database files from the samples area to the PointBase product installation area.

If you intend to share the PointBase installation with other users, then you will learn how to create your own PointBase runtime environment in a subsequent section.

If this is the case, then proceed to the next step, "3. Add PointBase Type 4 JDBC Driver to Server Application Server's Classpath."

To copy the prepopulated database files to your PointBase installation, copy the files contained in the following directory:

appserver_install_dir`/samples/pointbase/databases/*`

to:

pointbase_install_dir`/databases/`

This operation will copy the files named `sun-appserv-samples$1.wal` and `sun-appserv-samples.dbn` to the PointBase installation area.

3. Add PointBase Type 4 JDBC Driver to Server Application Server's Classpath

1. Copy the PointBase Type 4 JDBC driver library from the PointBase installation directory to the `lib/` directory of your application server instance. For example:

    ```
    .../domains/domain1/server1/lib/
    ```

    You can find the JDBC driver under *pointbase_install_dir*`/lib/`. The driver is named `pbclientnn.jar` where `nn` represents the version of PointBase.

2. Restart the application server to make the server aware of the driver.

Alternatively, you can specify the location of the PointBase driver in the Classpath Suffix field in the application server's configuration:

1. Start the administrative console and access the application server instance's `JVM Settings -> Path Settings` area to make this change.

If a PointBase JDBC driver is already configured in the Classpath Suffix field, replace it with the path to the newly installed driver.

2. Once you've changed the Classpath Suffix field, apply the changes and restart the application server instance.

Now that you've installed and configured PointBase, continue with the database set up steps at the beginning of this document, starting with "Configure the JDBC Driver," on page 80.

Deploying and Running the Sample Application

Your next step after having defined the JDBC connection pool and resource entries required by the application is to deploy the application.

Although a prebuilt copy of the sample application's EAR file is included as part of the sample, you will use the Ant facility and the sample's `build.xml` file to quickly compile the application source code and reassemble the EAR file from scratch. You will then deploy this newly built EAR file to the application server and exercise the application.

This section contains the following topics:

- Compile and Reassemble the Application
- Deploy the Sample Application
- Prepare to Monitor the Sample
- Run the Application
- Troubleshooting

Compile and Reassemble the Application

If you are either sharing your application server installation with other users or your system user ID does not have write permissions to the area in which the application server is installed, you should make a copy of the sample applications in your own directory before proceeding with this section. See "Making Your Own Copy of the Samples," on page 102. Otherwise, proceed to "Compiling and Reassembling the Application," on page 102.

This section contains the following topics:

- Making Your Own Copy of the Samples

- Compiling and Reassembling the Application

Making Your Own Copy of the Samples

If you are either sharing your application server installation with other users or your system user ID does not have write permissions to the area in which the application server is installed, copy the following directory to a location in which your user ID has write permissions:

install_dir\samples

To use the Ant build facility with the sample in this guide, you must ensure that the `com.sun.aas.installRoot property` in the following file is set to the installation path of the application server:

personal_samples_dir/samples/common.properties

This property is set automatically when the application server is installed except in the case when the application server is installed as part of a Solaris 9 installation.

In the case of copying the samples from `/usr/appserver/`samples in a Solaris 9 environment, you need to edit the common.properties file to reflect the following:

`com.sun.aas.installRoot=/usr/appserver`

Although customization of this property is sufficient to use Ant to the extent that is is demonstrated in this guide, to work with the full capabilities of Ant and the sample applications, you will need to customize additional properties in the common.properties file. After you complete the *Getting Started Guide*, refer to the "Using Ant with the Samples" section of the sample application documentation for details on customizing the other properties found in the common.properties file.

As you proceed with this guide, replace *install_dir*/samples/ with the location of your own copy of the sample applications.

Compiling and Reassembling the Application

1. Ensure that your environment is configured to include the application server's `bin` directory.

   This step was addressed in the section "Setting Up Your Environment" on page 37.

2. Using the command line, navigate to the source directory of the `jdbc-simple` sample application:

   cd *install_dir*\samples\jdbc\simple\src

3. From the command line, execute the Ant wrapper script `asant(.bat)` without any arguments to compile the Java source files and assemble the J2EE WAR, EJB JAR and EAR files:

   asant

TIP   **What is asant?** The `asant` utility is simply a wrapper script that calls the Apache Ant main class. A wrapper script is used to first set the appropriate classpath settings associated with the application server environment. For example, the appropriate JDK is set in the environment along with the JAR file containing the custom Ant tasks that are bundled with the application server. If you would like to use your own copy of Ant, review the `asant(bat)` file to ensure that the same classpath settings are made in your own Ant environment.

TIP   **What does executing asant do?** The `asant` utility automatically picks up a file name `build.xml` in the same directory in which `asant` is executed, much like the `make` command automatically processes a local Makefile. Like `make`, Ant supports the notion of a default target. In the case of the sample applications included in the application server, a default target named core results in a complete rebuild of a sample application. Other popular targets are: compile and deploy. See the Sample Applications documentation for more details.

---

4. Verify that the EAR file has been created by looking for a file named `jdbc-simple.ear` located in the following directory:

   cd ../assemble/ear

   You should see a file named `jdbc-simple.ear` in this directory.

   Now that you have successfully assembled the application from scratch, you can use the administrative console to deploy the application.

Deploy the Sample Application

1. In the administrative console, select and expand the Applications node under the application server instance `server1`.

2. Select the folder Enterprise Applications.
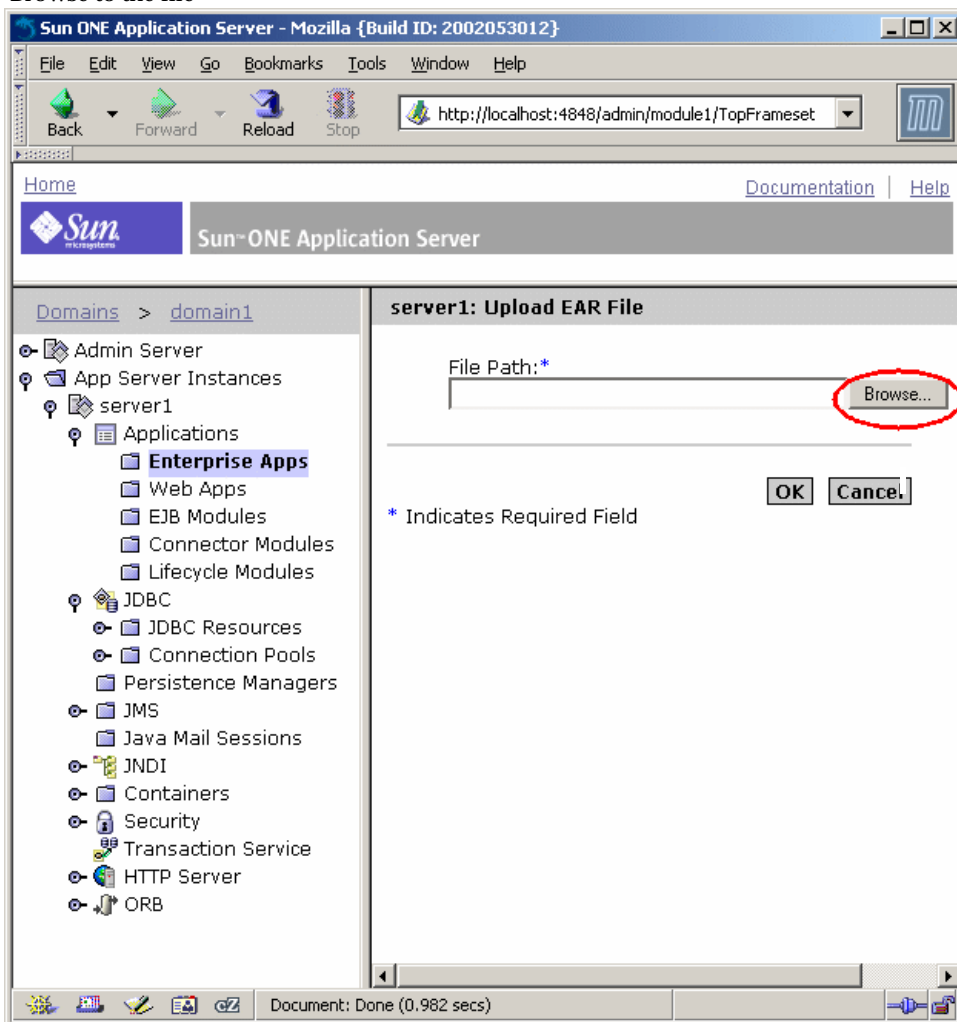
3. Click the Deploy button.

Deploying sample application
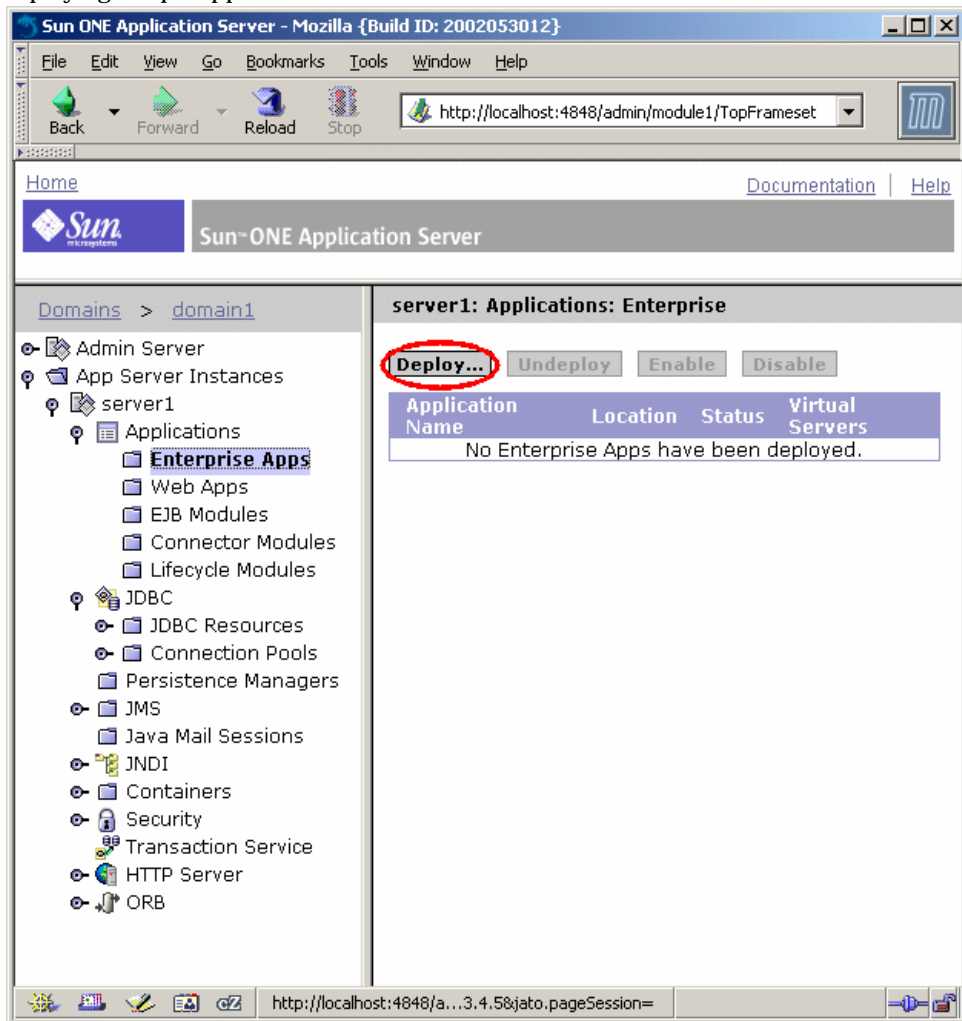
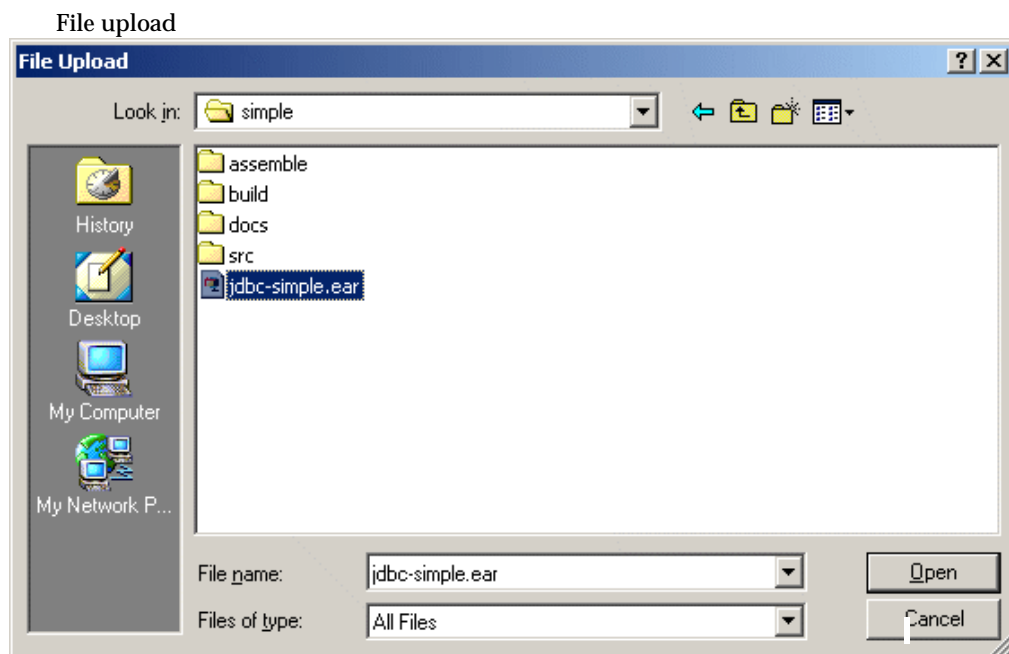**4.** Click the Browse button to bring up the file browser.

Browse to the file

5. Navigate to the following directory and select the `jdbc-simple.ear` file that you just assembled:
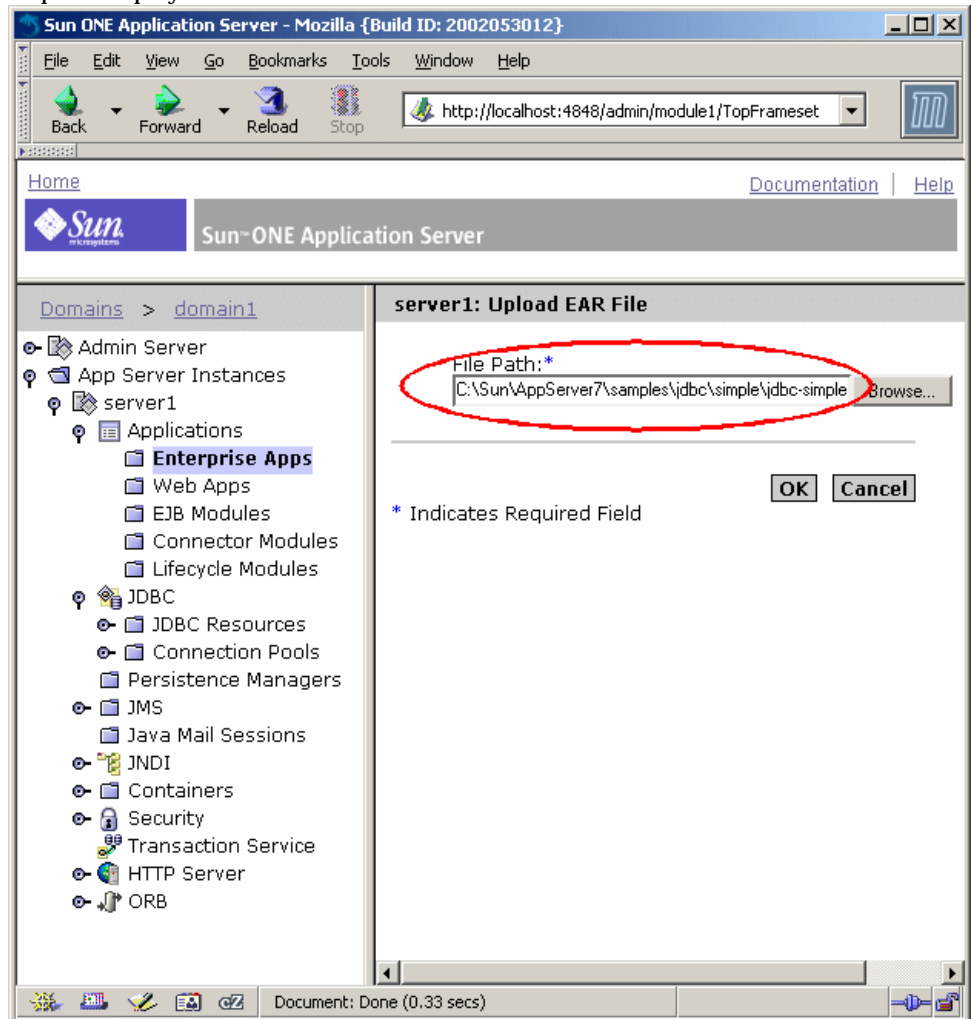
install_dir\samples\jdbc\simple\assemble\ear\

6. Click Open to select the file.

File upload

7. Click OK to proceed to the next step.

Upload deployment file

Before clicking OK to deploy the application, note the Run Verifier and Precompile JSPs check boxes. These options enable you to run a J2EE application verifier as part of the deployment process. (The verifier can also be access through execution of "asant verify" in the sample applications). The Precompile JSP option enables you to have the JSPs files compiled during the deployment process. Although the deployment process will be longer due to the JSP compilation step, the first access to your JSPs will be much faster.

8. Click OK to deploy the application.

jdbc-simple application
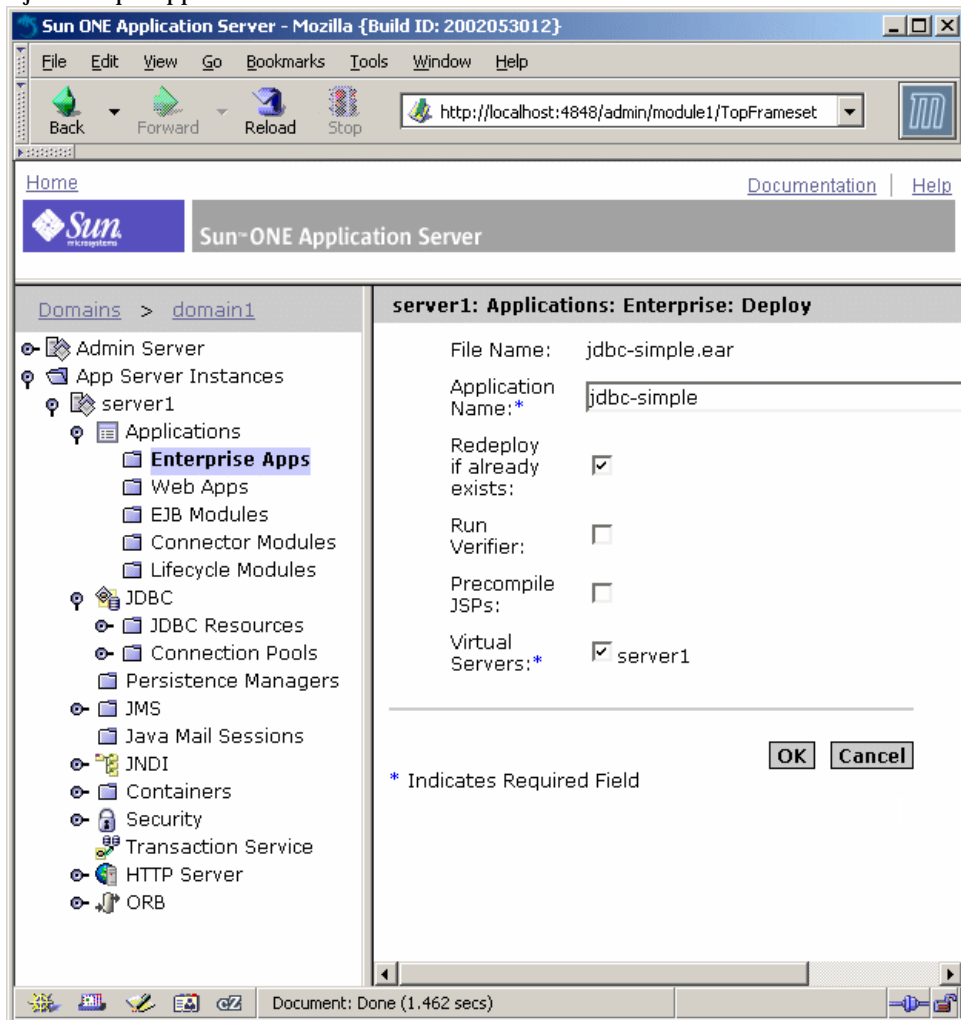
9. Once the deployment process completes, the following page is displayed:

Completed deployment process
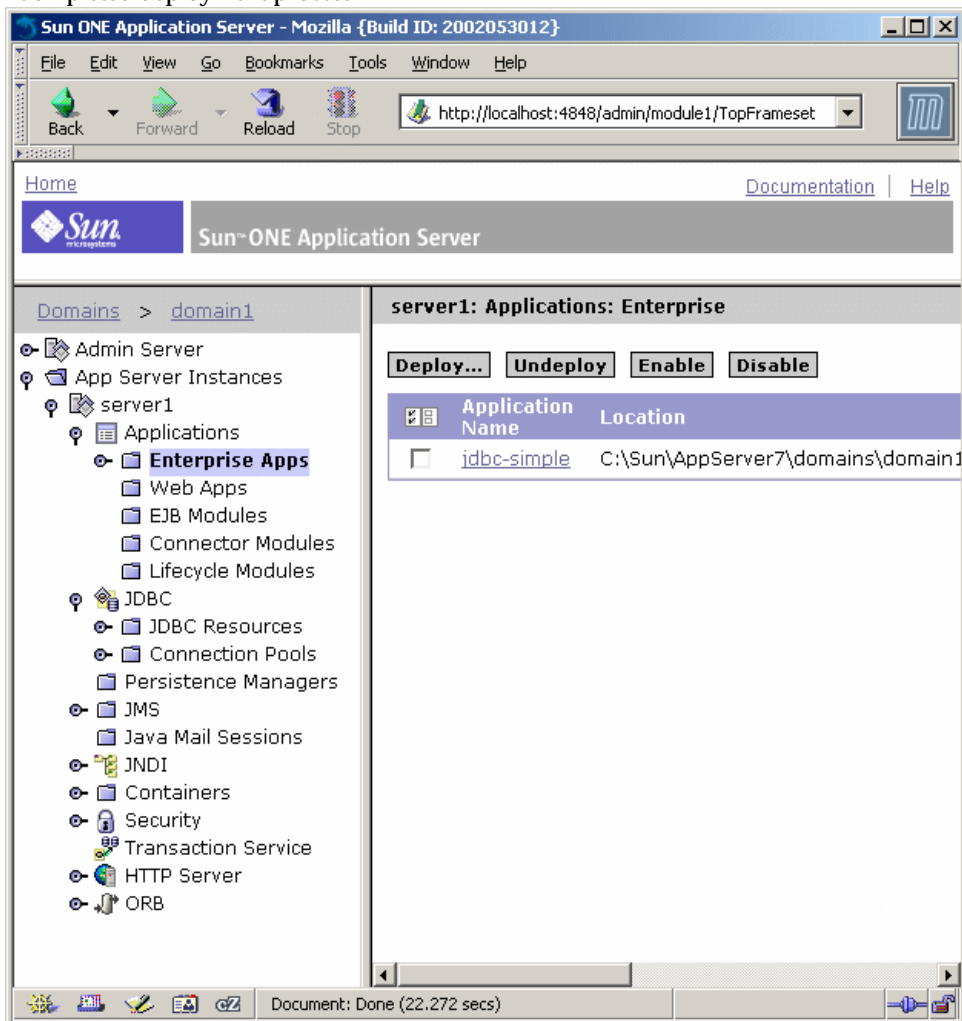
NOTE	**What happens during deployment?** To support formal application deployments, the administrative server must be running. The administrative server receives the deployed application, registers the application in the target server instance's `server.xml` file and installs the application in the target server instance's application deployment area.
	During initial deployment of an application, the administrative server automatically generates stubs and skeletons for all of the EJBs contained in the application (if EJBs exist). Subsequent redeployment of the same application and EJBs are typically much faster than the initial deployment because the deployment infrastructure senses whether or not the EJBs' interfaces have changed. (EJB stub and skeleton generation account for the majority of time spent during the initial deployment). During a redeployment, for those EJBs whose interfaces have not changed, the stub and skeleton generation step is bypassed. This feature is referred to as "smart redeploy".
	After deployment of an EAR, you'll find the application expanded in the following location:
	domain_config_dir/domain1/server1/applications/j2ee-apps/jdbc-simple_1
	The trailing `_1` represents the number of times this application has been deployed. This number is incremented during each redeployment.
	Web and EJB modules that are deployed individually are expanded under the `applications/j2ee-modules` directory.

10. Make the application server instance aware of the newly deployed application by applying the changes:

    a. Select the `server1` node.

    b. Click Apply Changes.

       Note that a server instance restart is not required in this case.

NOTE	**Module deployment**: Sun ONE Application Server 7 supports deployment of individual WAR and EJB JAR modules in addition to EAR file deployments. WAR module deployments are especially useful for web only applications. Under the Applications node in the administrative console, there is a Web Apps folder and an EJB Modules folder. Individually deployed modules reside under these folders. Each module is isolated after deployment in the sense that the application server instance uses a distinct classloader sandbox for each EAR-based application and each individually deployed WAR and EJB JAR module.

TIP	**Deploying from the command line**: In addition to deploying applications via the administrative console, you can also use the asadmin utility to deploy applications from the command line. Execute the following command for more information:

```
asadmin deploy --help
```

Additionally, the application server contains custom Ant tasks that ease the process of deploying applications from Ant `build.xml` files. All of the samples application `build.xml` files support the "deploy" target. See the Sample Applications and Using Ant with the Application Server documentation for more information.

Now that the application is deployed, you will be ready to run it after you first set up your environment to monitor application output in the server log file.

Prepare to Monitor the Sample

Before running the sample, you should prepare to view the output generated by both the sample as well as by the application server runtime. Since by default application output to `stdout` and `stderr` is redirected to the application server's event log, the application server provides you with a single place to monitor both the execution of both your server side application and application server infrastructure.

- Viewing Application Output and Logs on UNIX

- Viewing Application Output and Logs on Windows

- Using the Administrative Console to View Logs

Viewing Application Output and Logs on UNIX

Developers typically use the `tail -f` command to monitor log files on UNIX:

1. Navigate to the `server1` instance's log directory:

   cd *domain_config*`/domain1/server1/logs/`

2. Execute tail on the log file:

   `tail -f server.log`

Proceed to "Using the Administrative Console to View Logs," on page 114.

Viewing Application Output and Logs on Windows

While you are testing applications on an application server instance, you will likely find it useful to monitor server event log information on the Windows desktop. Such information can include your application's output to `stdout` and `stderr`, exceptions and server event messages.

As described in the previous section, display of application server instance event log information on the desktop is enabled by default.

If you would like to disable display of event log data on the desktop, you can use the administrative console to make this change. You can click the Logging tab, then General to access the Create Console setting of an application server instance, which enables you to control whether or not the application server log content for that instance is displayed in a command window on the desktop:
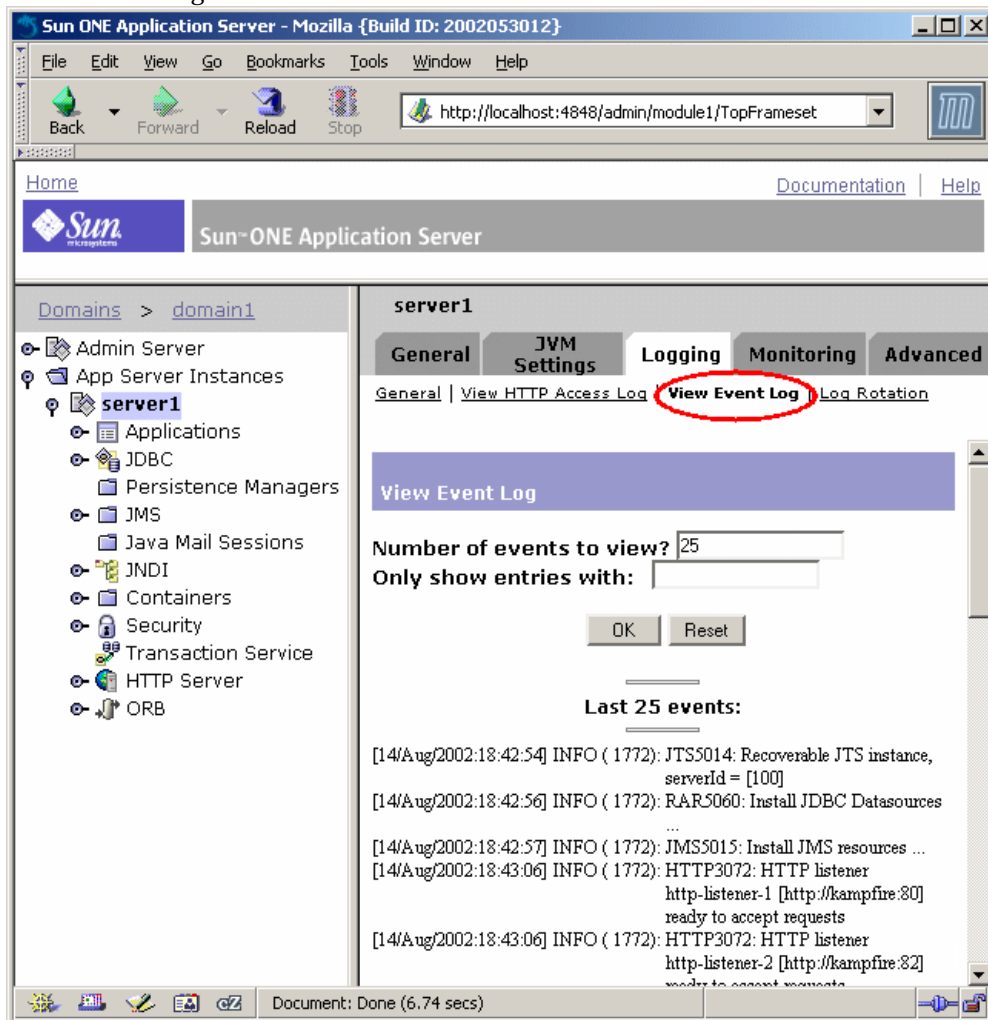
Create console
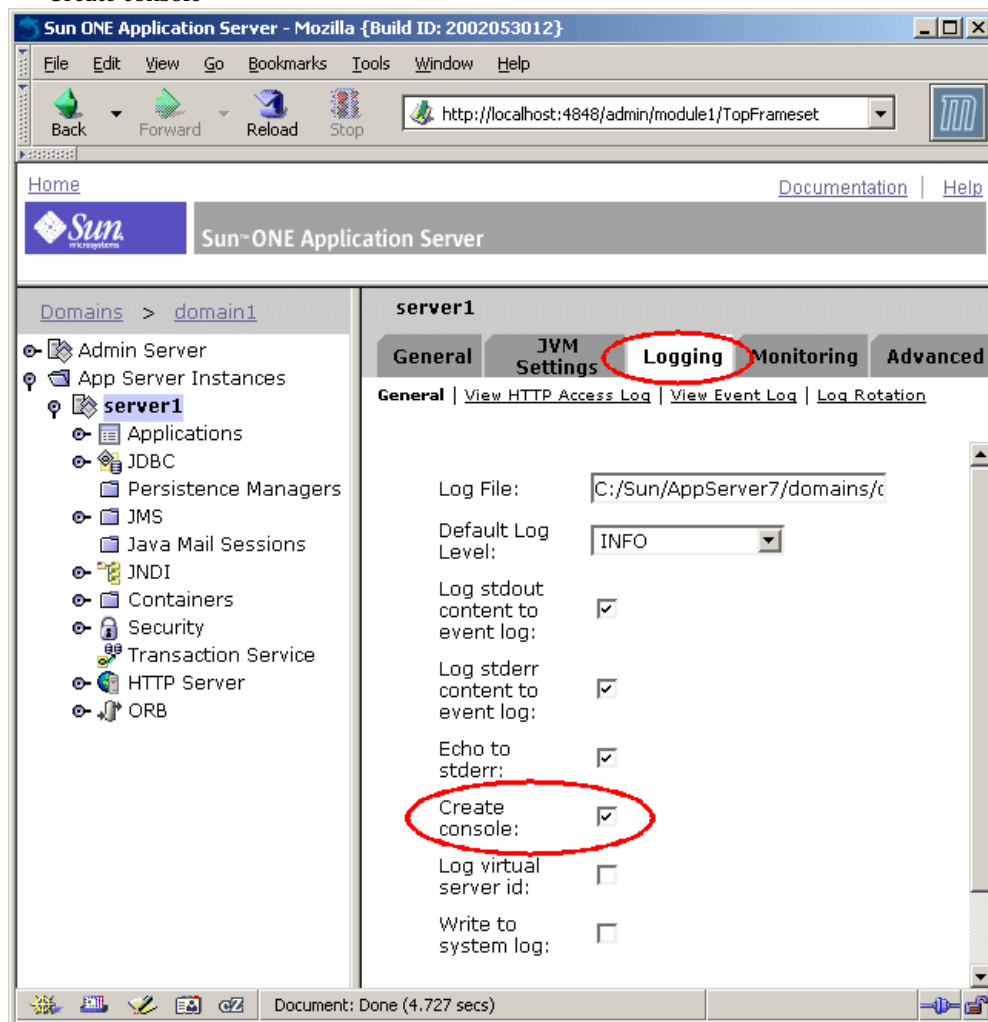
Using the Administrative Console to View Logs

You can also use the administrative console to view server instance log files:

1. Access the administrative console.

2. Select the server1 node.

3. Select the Logging tab followed by selecting the View Event Log link.

View event log

You can refresh the log view by clicking the OK button.

If you would like to see more than 25 log entries, simply enter a larger number in the "Number of errors to view?" area and click OK to refresh the log display. Try a value of 200 or so to see more log entries.

NOTE	**Viewing HTTP access log:** Note the View HTTP Access Log link in the Logging area. Clicking on this link will display the HTTP access log for the server instance. The name of the access log is access. By default, the access log file is located in the same directory as the server event log:
	domain_config_dir/domain1/server1/logs/

Now that you are ready to monitor the server log file, the next step is to start the database prior to running the application.

Run the Application

To run the sample application, perform the following steps:

1. First ensure that you are monitoring the server event log file for application output.

2. From a browser, access the following URL, enter your name, and click Process.

    ```
    http://localhost:port/jdbc-simple
    ```

    As you click on the Process button, note the application output written to the server event log file. Also note that there is a discernible delay while the application server compiles the initial JSP source file for the first time.

NOTE	**Application output in server event log:** When your application writes information to stdout and/or stderr, by default, this information is recorded in the server instance's event log as INFO-level messages with the message ID CORE3282 (stdout) and CORE3283 (stderr). While running the jdbc-simple sample, a number of application generated messages are written the server event log. The following excerpt provides a snapshot of some of these messages:

For example:

```
INFO: CORE3282: stdout: GreeterDBServlet is executing...

INFO: CORE3282: stdout: Retrieving JNDI initial
context...

INFO: CORE3282: stdout: - Retrieved initial context
successfully

INFO: CORE3282: stdout: Looking up dbGreeter bean home
interface...

INFO: CORE3282: stdout: - Looking up:
java:comp/env/ejb/jdbc-simple

INFO: CORE3282: stdout: - Looked up the EJB successfully
```

3. Once the greeting is displayed, click the link provided to view a log of all greetings generated to date.

   Again, there will be a delay as the second JSP source file is compiled.

4. Run the sample again to see how quickly it responds the second time you do so.

   Since the JSP files are already compiled, the second invocation is much faster than the initial run.

Troubleshooting

The most common problems when attempting to run this sample application are described in the following table.

Troubleshooting

Symptom	Probable cause	Solution
Connection failure when attempting to access first page.	Application server not started. Wrong port specified in URL.	Ensure that the application server has been started. Determine the correct HTTP server port number. See "Starting and Stopping the Application Server" on page 43 for more details.
Greeting is returned, but log of previous greetings displays 0 greetings.	Failed JNDI lookup due to `jdbc/jdbc-simple` not being defined. Failed JDBC connection attempt. Database not started.	Ensure that JDBC resource is defined correctly. Ensure that JDBC connection pool properties match those that are specified in the instructions. See "Start the PointBase Server Database" on page 93.
404 error when accessing main page.	Application not deployed.	See "Deploy the Sample Application" on page 104.

When troubleshooting problems, it is very important that you monitor the application server log file. You may also find it useful to review the HTTP access log file to verify that HTTP requests are arriving at the application server as expected.

Proceed to Chapter 8, "Modifying the Sample Application" to learn how the application server supports dynamic redeployment and reloading.

Troubleshooting

Modifying the Sample Application

Sun ONE Application Server 7 provides several dynamic deployment capabilities that can greatly improve developer productivity. This chapter introduces these features and leads you through several practical examples based on the sample application that you exercised in the previous section.

- Dynamic Deployment and Reloading Features
- Modifying Application Components

Dynamic Deployment and Reloading Features

Sun ONE Application Server 7 supports the following features to support fast development cycle times:

- Dynamic Deployment and Redeployment
- Smart Redeployment
- Dynamic Reloading

Dynamic Deployment and Redeployment

Dynamic deployment and redeployment of J2EE applications enables you to perform initial deployments of applications and subsequent redeployment without needing to restart the targeted application server instance. The application server completely reloads your application during redeployment. These features apply to both EAR level deployment as well as to deployments and individual web and EJB JAR module deployments. No server configuration changes are required to enable dynamic deployment and redeployment.

NOTE	**Dynamic Redeployment in Operational Environments** Although dynamic redeployment is positioned primarily as a development time feature, you can make use of this feature in operational environments as long as you take the following steps prior to redeploying an application:
	1. Redirect new requests for the application to another application server instance.
	2. Allow existing sessions to terminate.
	You will have to use external load balancers or other means to redirect incoming requests and can use built in monitoring facilities of the application server to determine when all sessions associated with the application have terminated.

Smart Redeployment

Smart Redeployment ensures that redeployment of EJB-based applications occurs as fast as possible. During redeployment of an application, the stubs and skeleton classes are regenerated only for those EJBs whose interfaces have changed since the last deployment. Since stub and skeleton generation incurs the most overhead of any operation during deployment, smart redeployment is a significant time saver. No server configuration changes are required to activate this feature. Smart redeployment is used during every redeployment operation.

Dynamic Reloading

Dynamic Reloading refers to the application server's ability to reload discrete class files, web content changes and deployment descriptor changes without requiring a complete reassembly and redeployment of the application and without restarting the application server. To support dynamic reloading of class files and deployment descriptor changes, the server instance monitors a special file named `.reload` residing at the top level of the deployment area of the application or individually deployed module. When a developer or make facility modifies or "touches" the `.reload` file, the server instance will completely reload the application or individually deployed module. This feature requires that you enable it explicitly for each application server instance. Since the monitoring incurs additional overhead, this feature is oriented towards development environments only.

Even without the dynamic reloading flag enabled, the web container of the application server automatically monitors web content such as static content and JavaServer Pages (JSP) files for modification. The web container will automatically reload a modified file when it is next accessed.

You may encounter the term "hot deploy" while reading about application servers. In many cases, hot deploy refers to dynamic deployment, dynamic redeployment and dynamic reloading. However, make sure you look closely at the details of the application server documentation to determine if all of these capabilities are supported by a specific product.

Modifying Application Components

In this section, you will modify the following types of content contained in the `jdbc-simple` sample application:

- Web Content Redeployment and Reloading
    - Dynamically Redeploying Servlet Class Files
    - Dynamically Reloading Static Content
    - Dynamically Reloading JSP Files
- EJB Redeployment and Reloading
    - Dynamically Redeploying EJB Implementation Class Files
    - Dynamically Reloading EJB Implementation Class Files

This section contains the following topics:

- Dynamically Redeploying Servlet Class Files
- Dynamically Reloading Static Content
- Dynamically Reloading JSP Files
- Dynamically Redeploying EJB Implementation Class Files
- Dynamically Reloading EJB Implementation Class Files
- Enable Dynamic Reloading
- Modify EJB Implementation Class
- Recompile and Copy to Deployment Area

- Create .reload File:

- Rerun the Application

Dynamically Redeploying Servlet Class Files

This exercise demonstrates the dynamic redeployment feature of the application server by leading you through the steps to modify a servlet class file, reassemble the application and dynamically redeploy the application without restarting the application server.

1. Navigate to the source code directory of the servlets in the example:

   install_dir`/samples/jdbc/simple/src/samples/jdbc/simple/servlet/`

2. Edit the `GreeterDBServlet.java` file: Find the line `System.out.println("\nGreeterDBServlet is executing...");` and modify it by adding "MODIFIED" to the front of the string.

   For example: `System.out.println("\nMODIFIED GreeterDBServlet is executing...");`.

3. Save your changes.

4. Recompile and reassemble the sample:

   a. Navigate back up to the sample's `src` directory.

   b. Execute `asant` without any options to recompile and reassemble the application.

      You should notice that a single source file was recompiled while the rest of the Java source files were not recompiled.

5. Redeploy the sample application using the administrative console.

   a. Ensure that the administrative server is running. If it is not running, you can quickly start it by executing:

      `asadmin start-instance admin-server`

   b. Access the administrative console.

   c. Under the `server1` node, expand the Applications node, select the Enterprise Apps node folder and click Deploy.

   d. Deploy the application in the same manner as you did in the Deploying the
   Application section of "Becoming Familiar with the Sample Applications"
   on page 73.

   Note the speed at which redeployment occurs when no changes have been
   made to the EJB interfaces. This speed is a result of the smart redeployment
   feature of the application server.

6. Rerun the application while monitoring the server event log file.

Note the presence of the word "MODIFIED" at the start of the following
`stdout` message:

```
INFO: CORE3282: stdout: MODIFIED GreeterDBServlet is executing...
```

TIP   **Application reload messages**: Further evidence of the application
            being fully reloaded can be viewed in the server event log file.
            During redeployment of the sample, the following messages are
            displayed to the event log of the targeted server instance:

```
INFO: CORE5022: All ejbs of [jdbc-simple] were unloaded
successfully!

INFO: LOADER5010: All ejbs of [jdbc-simple] loaded
successfully!

INFO: CORE3276: Installing a new configuration

INFO: WEB0100: Loading web module
[jdbc-simple:jdbc-simple.war] in virtual server
[server1]

INFO: WEB0100: Loading web module [default-web-module]
in virtual server [server1]

INFO: CORE3280: A new configuration was successfully
installed

INFO: WEB4004: Closing web application environment for
virtual server [server1]
```

This exercise demonstrated the capability to dynamically redeploy a modified
application in which a servlet class file was changed without requiring a restart of
the application server instance.

Dynamically Reloading Static Content

To dynamically reload static content, perform the following steps:

1. Access the `jdbc-simple` application and note the title string appearing at the top of the application's main page.

2. Navigate the to application deployment area of the application server instance, specifically to the web module's area:

   domain_config_dir/domain1/server1/applications/j2ee-apps/jdbc-simple
   _n/jdbc-simple_war/

3. Open the `index.html` file and modify the following section by adding "MODIFIED" at the beginning of the `<TITLE>` tag:

   ```
   <head>

   <title>MODIFIED JDBC-SIMPLE Sample Application</title>

   </head>
   ```

4. Save your changes.

5. Now access the application again and note the presence of the new title.

   If you do not see the modified title of the web page at the top of the browser window, press SHIFT and click on the browser's Reload button to reload the content from the application server.

Dynamically Reloading JSP Files

To dynamically reload JavaServer Page (JSP) files, perform the following steps:

1. Access the `jdbc-simple` application, enter your name, and click the Process button.

   Note the content of the resulting page. You will be modifying the JSP that displays this page.

2. Navigate the to application deployment area of the application server instance, specifically to the web module's area:

   domain_config_dir/domain1/server1/applications/j2ee-apps/jdbc-simple
   _n/jdbc-simple_war/

3. Open the `GreeterDBView.jsp` file and modify the following section by adding "MODIFIED" at the beginning of the `<TITLE>` tag:

```
<head>

<title>MODIFIED JDBC-SIMPLE Sample Application</title>

</head>
```

4. Save your changes.

5. Now access the application again and note the presence of the new title at the top of your browser window.

   Since the web container must recompile the JSP file after sensing its modification, there will be a noticeable lag when running the application the first time after the modification has been made.

6. Run the application again. You will notice the speed is greatly improved because the JSP has already been compiled.

TIP **Dynamic reloading of web content**: Note that in the static content and JSP examples of dynamic reloading, you did not have to enable the dynamic reloading switch in the application server instance. By default, the web container dynamically reloads web content. Within the `sun-web.xml` file of a web application, you can set a switch that will tell the web container to never reload JSP files. This useful when you want to avoid inadvertent activation of modified and deployed JSP sources files.

Dynamically Redeploying EJB Implementation Class Files

In the same manner that you modified the servlet class file, in this exercise you will modify the EJB implementation class file and leverage dynamic redeployment to quickly retest the application without restarting the application server.

1. Navigate to the source code directory of the EJB in the example:

   install_dir`/samples/jdbc/simple/src/samples/jdbc/simple/ejb/`

2. Edit the `GreeterDBBean.java` file:

   Find the line `System.out.println("GreeterDB EJB is determining message...");` and modify it by adding "MODIFIED" to the front of the string.

   For example: `System.out.println("MODIFIED GreeterDB EJB is determining message...");`.

3. Save your changes.

4. Recompile and reassemble the sample:

   a. Navigate back up to the sample's `src` directory.

   b. Execute `asant` without any options to recompile and reassemble the application. You should notice that a single source file was recompiled while the rest of the Java source files were not recompiled.

5. Redeploy the sample application using the administrative console.

   a. Ensure that the administrative server is running. If it is not running, you can quickly start it by executing:

      `asadmin start-instance admin-server`

   b. Access the administrative console.

   c. Select the Enterprise Apps node folder and click Deploy.

   d. Deploy the application in the same manner as you did in the previous section.

      Again, note the speed at which redeployment occurs when no changes have been made to the EJB interfaces.

6. Rerun the application while monitoring the server event log file.

   Note the presence of the word "MODIFIED" at the start of the following `stdout` message:

   `INFO: CORE3282: stdout: MODIFIED GreeterDB EJB is determining message...`

This exercise demonstrated the capability to dynamically redeploy a modified application in which an EJB implementation class file was changed without requiring a restart of the application server instance.

Dynamically Reloading EJB Implementation Class Files

This exercise demonstrates the dynamic reloading feature. Although the steps required to take advantage of dynamic class reloading appears more complex than that used with dynamic redeployment, the speed enhancements of dynamic reloading surpass that of dynamic redeployment. Using either a make facility or your Java compiler's class file destination setting to automate the process of copying discrete files from your build area to the deployment area will help you take advantage of the benefits of the dynamic reloading feature.

In the first part of this exercise you will make another modification to the GreeterDBBean's implementation class. After recompiling the EJB, you will manually copy it to the application's deployment area. Then you'll create the special .reload file and rerun the sample application. The creation of this special .reload file is your means to inform the server instance that one or more new files have been copied to the application deployment area and that the instance should reload the entire application. During a reload of the complete application, the application server instance flushes all of application components from memory and reloads them again. All user sessions associated with the application are also flushed during an application level reload operation.

After copying the modified class file and creating the .reload file, you can almost immediately test the modified application.

There is no need to reassemble and formally redeploy your applications when leveraging the dynamic reloading facility.

Enable Dynamic Reloading

Before exercising dynamic class reloading, you will need to first enable dynamic reloading.

Since the dynamic reloading feature incurs additional overhead by constantly monitoring the presence and modification of .reload files, this setting should not be enabled in operational environment.

1. Access the administrative console.

2. Select the server1 instance node.

3. Select the Applications folder.

4. Select the Reload Enabled checkbox.

5. Click Save.

6. Select the Server instance `server1`, click Apply Changes, and restart the server instance.

Reload enabled
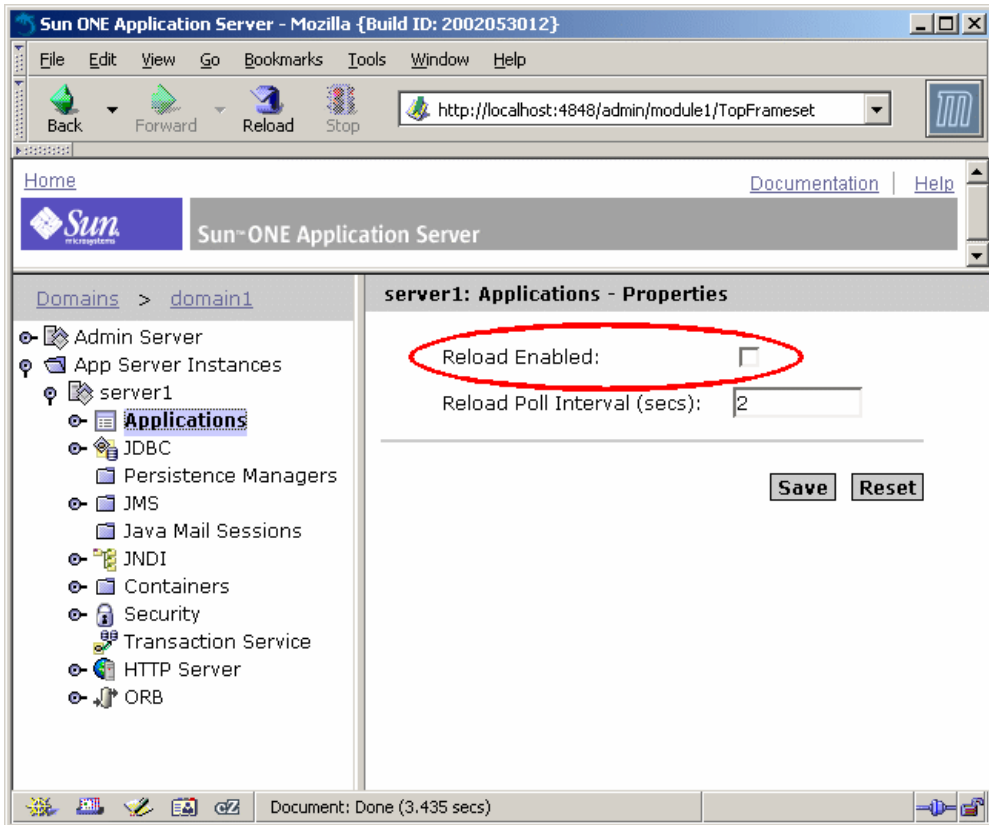
Now modify the EJB implementation class, recompile it and manually copy it to the deployment area.

Modify EJB Implementation Class

To modify the EJB implementation class, perform the following steps:

1. Navigate to the source code directory of the EJB in the example:

   install_dir/samples/jdbc/simple/src/samples/jdbc/simple/ejb/

2. Edit the `GreeterDBBean.java` file again: Find the line
   `System.out.println("MODIFIED GreeterDB EJB is determining message...");` and modify it by adding "MODIFIED AGAIN" to the front of the string.

   For example: `System.out.println("MODIFIED AGAIN GreeterDB EJB is determining message...");`.

3. Save your changes.

Recompile and Copy to Deployment Area

To recompile the application and copy it to the deployment area, perform the following steps:

1. Navigate back up to the sample's `src` directory.

2. Execute `asant` with the compile target name to recompile the modified Java source file. The compile target is used to avoid a complete reassembly of the application.

   ```
   asant compile
   ```

   You should see that one source file was modified and recompiled while the rest of the Java source files were not recompiled.

3. Manually copy the newly generated class from the local `build/classes/` directory to the application deployment area within the target application server instance.

   You will find the newly generated EJB implementation class at the following location:

   install_dir/samples/jdbc/simple/build/classes/samples/jdbc/simple/ejb/GreeterDBBean.class

4. Copy this file to the following location:

   domain_config_dir/domain1/server1/applications/j2ee-apps/jdbc-simple_n/jdbc-simpleEjb_jar/samples/jdbc/simple/ejb/

Create .reload File:

Create an empty file named `.reload` at the root of the application's deployment directory:

- On Windows, use Notepad to create a new text file and save it to the following location:

  domain_config_dir`/domain1/server1/applications/j2ee-apps/jdbc-simple`
  `_n/.reload`

NOTE   **Problems Creating `.reload` File Using Notepad?** As you save the
`.reload` file in Notepad, ensure that you select "Save as file type:"
as All Files to ensure that Notepad does not automatically append a
`.txt` extension to the `.reload` filename.

- On UNIX, simply execute `touch .reload` in the directory above to create an empty file.

Rerun the Application

1. Rerun the application while monitoring the server event log file.

    Note the presence of the words "MODIFIED AGAIN" at the start of the following `stdout` message:

    ```
    INFO: CORE3282: stdout: MODIFIED AGAIN GreeterDB EJB is
    determining messa..."
    ```

2. Modify the EJB implementation class again by changing the `println()` statement. Recompile, copy the EJB implementation class and modify the `.reload` file again to effect a dynamic reload of the application.

TIP	**Manual copying is fast?** In typical command line oriented development scenarios, you'll use either an Ant `build.xml` or a Makefile target named something like `update` to automate the process of copying modified `.class` files from your development workspace area directly to the application's deployment area and "touching" the `.reload` file.
	When using an IDE, you will typically configure the compiler options in the IDE such that `.class` files are created in the deployment area of the application server. Consequently, whenever you recompile a specific Java source file, the resulting `.class` file will automatically be copied to the deployment area. Then all you'll need to do is modify the `.reload` file in another source editing window and the application will be automatically reloaded.
	These approaches yield super fast development edit-compile-test cycles because the process of reassembling and performing a formal redeployment is removed from the overall process. Note that dynamic reloading does not involve the administrative server. In fact, using this method of updating your deployed application, you need not keep your administrative server up and running.

Proceed to Chapter 9, "Exploring the Server Further" to learn about commonly performed tasks.

Exploring the Server Further

Now that you have gained familiarity with basic application deployment, redeployment and reloading capabilities of the application server, you can learn how to carry out common administrative tasks by performing the following brief exercises.

- Changing the HTTP Listener Port Number

- Adding a New HTTP Listener

- Adding a Virtual Server

- Adding a Server Instance

- Creating an Administrative Domain

Changing the HTTP Listener Port Number

When you need to change the HTTP listener port number to avoid conflicts with other servers, you can easily do so via the administrative console:

1. Access the administrative console.

2. Expand the HTTP Server and HTTP Listeners folders under the application server instance.

3. Click the listener named `http-listener-1`.

4. Change the port number to another value.

5. Click Save.

6. Select the instance node named `server1`.

7. Click Apply Changes. (No server restart is required).

8. Using a browser, access the HTTP server on the newly specified port.

Adding a New HTTP Listener

Although the application server is initially configured with a single HTTP listener, it is straightforward to add another listener. In the following exercise you will create a new non-SSL HTTP listener.

To add a new HTTP listener, perform the following steps:

1. Access the administrative console.

2. Expand HTTP Server

3. Click HTTP Listeners.

4. Click New.

5. Enter the information as listed in the following table:

`Name`	new-listener
`IP Address`	0.0.0.0 (listens on any interface)
`Port`	select a suitable port
`Return Server Name`	any valid host/interface name for the machine

6. Click OK.

7. Select the instance node `server1`.

8. Click Apply Changes.

9. Restart the server instance.

10. Using a browser, access the HTTP server on both port numbers.

   During server restart, notice the following messages in the server instance event log:

   ```
   INFO: HTTP3072: HTTP listener http-listener-1
   [http://kampfire:88] ready to accept requests

   INFO: HTTP3072: HTTP listener new-listener [http://kampfire:89]
   ready to accept requests
   ```

Adding a Virtual Server

In operational environments, it is common to configure a single HTTP server to host content and web applications for multiple domains. For example, the domains www.abc.com and www.xyz.com can be easily served by separate virtual servers defined on the same HTTP server. The web traffic served by a virtual server can be distinguished either by the interface or port number over which HTTP traffic is received or by the domain name on the incoming HTTP requests.

In this exercise you will use a newly defined HTTP listener to route traffic to a new virtual server.

1. Access the administrative console.

2. Expand HTTP Server

3. Following the steps above, create a new HTTP Listener that will be associated with the new virtual server.

4. Click Virtual Servers

5. Click New.

6. Enter the information as shown in the following table:

`ID`	new-virtual-server
`Hosts`	host name expected in the incoming URL (not important in this example because a unique port is being associated with this virtual server).
`MIME Types File`	mime1
`HTTP Listeners`	select the appropriate HTTP listener(s)
`Access Log`	*domain_config_dir*/`domain1/server1/logs/`*access_log_file_name*

7. Click OK

8. Select the instance node named `server1`.

9. Click Apply Changes.

10. Restart the server instance.

11. Using a browser, access the HTTP listener port associated with the new virtual server.

Adding a Server Instance

If you are a developer and need to define a separate "sandbox" in which to experiment with either application server configurations, you may need to define a new application server instance. For example, if you need to upgrade to a new Java 2 SDK (JDK) version and would like to experiment with the newer JDK in one environment yet maintain your stable configuration, it might make sense for you to define a new application server instance.

To add a server instance, perform the following steps:

1. Access the administrative console.

2. Select Application Server Instances.

3. Click New.

4. Enter a server instance name (for example, `my-new-server`) and a port number for the initial HTTP Listener.

5. Click OK

6. Select the new server instance in the list of instances.

7. Click Apply Changes.

8. Click Start.

9. Using a browser, access the HTTP listener port associated with the new server instance.

10. Navigate to the `domain1` directory of your installation:

    domain_config_dir/`domain1/`

    Note the presence of a new directory representing your newly created instance.

    In the administrative console, under Application Server Instances, note the presence of a new server instance node. Explore this new node to see the default settings associated with the newly created application server instance.

Creating an Administrative Domain

If you are using the application server that was installed as part of a Solaris 9 bundled installation, you already used the `asadmin` CLI to create a new administrative domain. If this is the case, proceed to the final section, Chapter 10, "Summary and Next Steps."

For all other situations, use the following instructions to create a new domain named `mydomain`, start the domain and access the administrative server that manages this domain.

Since domain creation and deletion is available only through the asadmin CLI, you must use the command line to create the new domain.

1. From the command line, execute the following command to create a new administrative domain named `mydomain`:

```
asadmin create-domain --adminport 9999 --adminuser admin
--adminpassword password mydomain
```

where the port number, user and password fields specify the initial settings of the new administrative server defined for the domain.

Upon execution of this command, you should see the following message:

```
Created Domain mydomain successfully
```

At this stage, you should see a newly defined directory:

domain_config_dir/mydomain/

If domain1 has already been used, execute the create-domain subcommand again with another domain name. You can use periods and other characters in your domain names. You could use your login user name as a qualifier to help ensure that your domain name is unique. For example: ckamps.domain1.

2. Execute the list-domains subcommand to display a list of all of the domains configured for the application server installation.

For example:

```
asadmin list-domains
```

```
domain1 [domain_config_dir/mydomain]
```

where the value of *domain_config_dir* represents either the default location for newly created administrative domains or the value specified on the --path option of the create-domain subcommand.

TIP   **Overriding the Location of the Administrative Domain Configuration**: To override the default location of the newly created administrative domains, use the --path option of the create-domain subcommand. Since the administrative domain configuration will be written to this area, you must have write permissions to the specified location.

```
asadmin create-domain --path domain_config_dir --adminport
4848 --adminuser admin --adminpassword password mydomain
```

In this example, a new directory named mydomain will be created under the specified *domain_config_dir*.

---

3. Now start the new domain by executing the following command:

```
asadmin start-domain --domain mydomain
```

This command simply starts the administrative server of the newly created domain.

4. Once the administrative server of the new domain has started, you should be able to start the administrative console of the admin server by using a browser to access the following URL:

```
http://localhost:9999
```

5. After you authenticate to the administrative server, use the administrative console to create a new application server instance and then access its HTTP port via a web browser.

To stop the domain, you can execute the following command:

```
asadmin stop-domain --domain mydomain
```

Once you have completed these exercises, proceed to the final section Chapter 10, "Summary and Next Steps."

Summary and Next Steps

By following the Getting Started Guide you explored these key aspects of the Sun ONE Application Server:

- Key features and architecture

- Controlling the application server

- How to deploy and exercise a J2EE application

- Using JDBC to access a database

- Basics of using the Ant-based build facility to easily recompile and reassemble J2EE applications

- Basics of the product directory structure

- Dynamic deployment, redeployment and reloading features that help streamline the development cycle.

- Common administrative tasks such as creating new virtual servers and new application server instances.

Review the following next steps to determine the resources that are best suited towards your interests:

Next steps

Interest	Next Steps
Experience J2EE development using a leading Java IDE and the Sun ONE Application Server 7.	See the Sun ONE Studio 4, Enterprise Edition for Java and Application Server Tutorial.
Learn more about J2EE.	See the Sample Applications.

Next steps

Interest	Next Steps
Learn about J2EE design and development best practices.	Consult the Java BluePrints and try out the Java Pet Store and Smart Ticket sample applications that are bundled with the application server.
Learn how specific J2EE features are implemented on Sun ONE Application Server 7.	See the Sample Applications included in the application server installation.
Learn about general development practices using Sun ONE Application Server 7.	Review the Sun ONE Application Server 7 *Developer's Guide* documentation.
Learn about administering Sun ONE Application Server 7.	Review the Sun ONE Application Server 7 *Administrator's Guide* documentation.

Index