Getting Started Guide

Netscape Application Server

Version 4.0 SP2

Recycled and Recyclable Paper

Version 4.0 SP2

Copyright ©1999 Netscape Communications Corp. All rights reserved.

Printed in the United States of America. 00 99 5 4 3 2 1

Netscape Communications Corporation, 501 East Middlefield Road, Mountain View, CA 94043

Contents

Preface

The *Getting Started Guide* provides an overview of the Netscape Application Server (NAS) and information about how to get started creating applications on your own. In addition, this guide describes how to use the quick installation script to install NAS.

This preface contains the following topics:

- How This Guide Is Organized
- Using the Documentation
- Documentation Conventions

How This Guide Is Organized

The following chapter are included in this guide:

- Chapter 1, “Product Overview” provides background information about the product components, features, and system architecture of Netscape Application Server (NAS).
- Chapter 2, “Installing NAS on Solaris 8” describes how to quickly install NAS 4.0 SP2.
- Chapter 3, “Using the Sample Applications” describes the suite of sample applications that were installed with NAS and tells you how to run them.
- Chapter 4, “Developing Applications” describes the relationships between files used in creating applications to run on the Netscape Application Server and provides an in-depth review of the online bookstore sample application.
- Chapter 5, “Deploying Applications” describes how to deploy applications from one machine to another using the Deployment Manager.

Using the Documentation

The following table lists the tasks and concepts that are described in the Netscape Application Server (NAS) printed manuals and online read-me file in addition to this *Getting Started Guide*. If you are trying to accomplish a specific task or learn more about a specific concept, refer to the appropriate manual.

Note that the printed manuals are also available as online files in PDF and HTML format.

For information about	See the following	Shipped with
Late-breaking information about the software and the documentation	`readme.htm`	NAS 4.0 Developer Edition (Solaris), NAS 4.0, NAB 4.0
Installing Netscape Application Server and its various components (Web Connector plug-in, Netscape Application Server Administrator), and configuring the sample applications	*Installation Guide*	NAS 4.0 Developer Edition (Solaris), NAS 4.0
Installing Netscape Application Builder.	`install.htm`	NAB 4.0
Basic features of NAS, such as its software components, general capabilities, and system architecture.	*Overview*	NAS 4.0 Developer Edition (Solaris), NAS 4.0, NAB 4.0

For information about	See the following	Shipped with
Deploying Netscape Application Server at your site, by performing the following tasks: • Planning your Netscape Application Server environment • Integrating the product within your existing enterprise and network topology • Developing server capacity and performance goals • Running stress tests to measure server performance • Fine-tuning the server to improve performance	*Deployment Guide*	NAS 4.0
Administering one or more application servers using the Netscape Application Server Administrator tool to perform the following tasks: • Deploying applications with the Deployment Manager tool • Monitoring and logging server activity • Setting up users and groups • Administering database connectivity • Administering transactions • Load balancing servers • Managing distributed data synchronization	*Administration Guide*	NAS 4.0
Migrating your applications to the new Netscape Application Server 4.0 programming model from version 2.1, including a sample migration of an Online Bank application provided with Netscape Application Server	*Migration Guide*	NAS 4.0 Developer Edition (Solaris), NAS 4.0, NAB 4.0

For information about	See the following	Shipped with
Creating NAS 4.0 applications within an integrated development environment by performing the following tasks: • Creating and managing projects • Using wizards • Creating data-access logic • Creating presentation logic and layout • Creating business logic • Compiling, testing, and debugging applications • Deploying and downloading applications • Working with source control • Using third-party tools	*User's Guide*	NAB 4.0
Creating NAS 4.0 applications that follow the new open Java standards model (Servlets, EJBs, JSPs, and JDBC), by performing the following tasks: • Creating the presentation and execution layers of an application • Placing discrete pieces of business logic and entities into Enterprise Java Bean (EJB) components • Using JDBC to communicate with databases • Using iterative testing, debugging, and application fine-tuning procedures to generate applications that execute correctly and quickly	*Programmer's Guide (Java)*	NAS 4.0 Developer Edition (Solaris), NAB 4.0

For information about	See the following	Shipped with
Using the public classes and interfaces, and their methods in the Netscape Application Server class library to write Java applications	*Server Foundation Class Reference (Java)*	NAS 4.0 Developer Edition (Solaris), NAB 4.0
Creating NAS C++ applications using the NAS class library by performing the following tasks: • Designing applications • Writing AppLogics • Creating HTML templates • Creating queries • Running and debugging applications	*Programmer's Guide (C++)*	Order separately
Using the public classes and interfaces, and their methods in the Netscape Application Server class library to write C++ applications	*Server Foundation Class Reference (C++)*	Order separately

Documentation Conventions

File and directory paths are given in Unix format with slashes separating directory names.

This guide uses URLs of the form:

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

In these URLs, *server* is the name of server on which you run your application; *domain* is your Internet domain name; *path* is the directory structure on the server; and *file* is an individual filename. Italic items in URLs are placeholders.

This guide uses the following font conventions:

- 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.

Chapter

1

Product Overview

This chapter explains the role of Netscape Application Server (NAS) within the multitier enterprise environment. In addition, the chapter summarizes the main product components that come with NAS.

This chapter contains the following sections:

- The Multi-tiered Environment
- Product Components

The Multi-tiered Environment

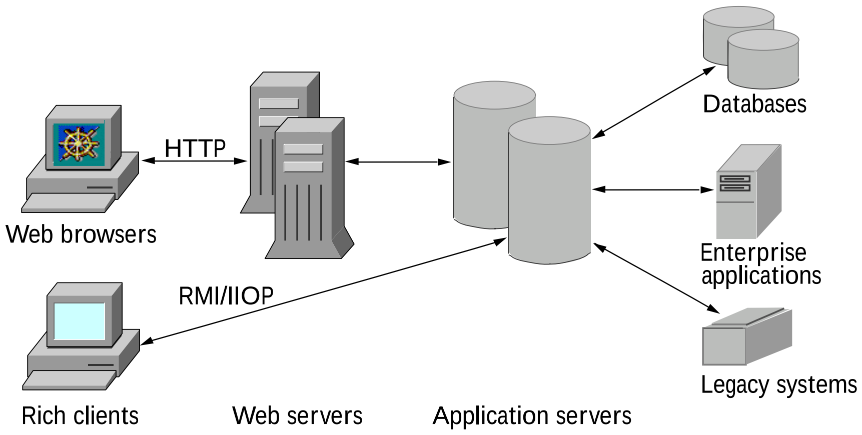
Netscape Application Server is the middleware between enterprise data sources and the clients that access those data sources. Business code is stored and processed on Netscape Application Server (NAS) rather than on clients. An application is deployed and managed in a single location, and the application is accessible to large numbers of heterogeneous clients.

NAS applications run in a distributed, multi-tiered environment. This means that an enterprise system might consist of several application servers (computers running the Netscape Application Server software), along with multiple database servers and web servers. Your application code can be distributed among the application servers.

Overall, the machines and software involved are divided into three tiers:

- a client tier, represented by web browsers or rich clients (such as a Java application).
- a server tier, represented by a web server (such as iPlanet Web Server) and an application server (such as Netscape Application Server).
- a data tier, represented by relational databases or other back-end data sources.

The following figure shows a multi-tiered environment:



End users interact with client software, typically a web browser, to use the application.

- When a request originates from a web browser, it is sent to the web server. Assuming the request requires application processing or data access, the web server forwards the request to Netscape Application Server (NAS).
- When a request originates from a CORBA client, it is sent to NAS by way of an RMI/IIOP link. (An RMI/IIOP-based product will be shipped in the next generation of iAS 6.0.)

NAS handles requests by running the appropriate application code (and accessing data sources if needed). NAS returns the results to the web server, which in turn forwards the reply back to the client.

For more information about the components and interactions in the multitier environment, see the *Programmer's Guide*.

Product Components

This section describes the software and product components of NAS. This section consists of the following topics:

- Programming APIs
- System Services and Application Services
- Sample Applications
- NAS Administrator
- NAS Deployment Manager
- Netscape Directory Server

Programming APIs

NAS supports several industry-standard Java APIs. In particular, NAS supports the APIs and technologies as defined by the following specifications:

- *Java Servlet 2.1 API Specification*
- *Enterprise JavaBeans 1.0 Specification*
- *JavaServer Pages 0.92 Specification*
- *JDBC 2.0 Core API Specification*
- *JDBC 2.0 Standard Extensions Specification*

Note that NAS 4.0 provides full support for JDBC 1.0 and partial support for JDBC 2.0. The supported JDBC 2.0 functionality is described in the *Programmer's Guide (Java)*.

For building application components written in C++, NAS provides the Foundation Class Library. Java application developers can also use the NAS Foundation Class Library to leverage additional capabilities of NAS that are not supported through standard APIs.

For more information about the NAS Foundation Class Library, see the *Programmer's Guide* or the *Netscape Application Server Foundation Class Reference*. Both of these documents are available in Java or C++ versions.

For more information about the industry-standard Java APIs that are supported in NAS 4.0, see the appropriate API specification. All specifications are accessible from `installdir/nas/docs/index.htm`, where *`installdir`* is the location in which you installed NAS.

System Services and Application Services

System and application services provide a variety of application-level capabilities and system-level capabilities. These services enable the development, deployment, and management of complex business-logic and transaction-based applications.

For more information about these services, see the *Overview Guide.*

Sample Applications

NAS includes sample web-based applications, enabling you to more quickly learn techniques for developing and deploying applications in a NAS environment.

One sample presents a bookstore application that simulates browsing, searching, and ordering books online. This Java application demonstrates the NAS application model that uses industry-standard components such as servlets, JavaServer Pages, Enterprise JavaBeans, and data access with JDBC. For information about installing or using the online bookstore application, see the *Installation Guide* or the *Programmer's Guide (Java)*.

Another sample presents a banking application that simulates a user session with an online account. This sample demonstrates techniques for migrating existing applications to comply with the industry-standard Java application model. For information about installing the bank application, see the *Installation Guide*. For details about the application code, see the *Migration Guide.*

NAS Administrator

NAS Administrator is a GUI tool that contains several smaller tools for managing one or more NAS machines or applications. For more information, see the "Management Capabilies" section in this guide. For detailed information about using NAS Administrator, see the *Administration Guide*.

NAS Deployment Manager

An application must be deployed before it can be used, and NAS Deployment Manager is a GUI tool that makes application deployment easier. You access this tool either from NAS Administrator or from Netscape Application Builder.

When you deploy an application, NAS Deployment Manager installs all the application's files and registers all of its components on the destination server (a server on which NAS has been installed).

For information about using NAS Deployment Manager to deploy applications, see the *Administration Guide*.

Netscape Directory Server

Netscape Directory Server provides a comprehensive, enterprise-wide directory service for managing information about users, groups, and access control lists. NAS 4.0 includes Netscape Directory Server 4.0, which supports versions 2 and 3 of the Lightweight Directory Access Protocol (LDAP).

For more information, see the NAS *Deployment Guide*. In addition, consult the *Installation Guide* for Netscape Directory Server 4.0.

Features

This section describes the key features of Netscape Application Server.

This section contains the following topics:

- Rapid Application Development

- High Scalability
- High Performance
- High Availability
- Session and State Management
- Security
- Management Capabilities

Rapid Application Development

NAS enables rapid development of enterprise applications through the following features, which are described in this section:

- a multi-tier application model.
- core services for building high-performance, scalable, transactional applications.
- support for a variety of application development tools.

Application Model

An application model is the conceptual division of a software application into functional components. The NAS application model promotes code reusability and faster application deployment.

The NAS application model divides an application into multiple layers: presentation, business logic, and data access. Presentation is further separated to distinguish page layout and presentation logic. Data access refers to both databases and other data sources.

The NAS application model is component-oriented. For Java applications, the application model has been enhanced. The model now incorporates standard components based on Java technologies.

The following table lists the main components that make up the functions of an application in the NAS environment:

Functionality in Application Model	Application Components	
	Java Applications	C++ Applications
Presentation logic	servlets	AppLogics
Page layout	JavaServer Pages	HTML templates
Business logic	Enterprise JavaBeans	AppLogics
Database access	Enterprise JavaBeans using JDBC; query files	AppLogics using the NAS API; query files
Access to other data sources	Extensions	Extensions

These application component fall into two categories:

- Industry-Standard Components
- Other Components

Industry-Standard Components

For developing Java applications, we recommend that you use standard components whenever possible. These standards-based components include servlets, JavaServer Pages (JSPs), and Enterprise JavaBeans (EJBs), described as follows:

- Servlets are Java classes that define page logic and page navigation. Servlets also support the creation or invocation of business components.
- JSPs are web browser pages written in a combination of HTML, JSP tags, and Java.
- EJBs encapsulate an application's business rules and business entities.

In addition, application components can invoke JDBC calls. JDBC is a standard API for database connectivity.

For more information about using these standard components in NAS applications, see the *Programmer's Guide (Java)*.

Other Components

Other application components include AppLogics, HTML templates, query files, and extensions. An application must use these components if it is written in C++ or if it will run in the NAS 2.x environment. And if your application must access propriety enterprise data sources, you may need to use prebuilt extensions or create your own.

- An AppLogic is a set of programming instructions, written in C++ or Java, that perform a well-defined, modular task within an application.
- HTML templates are text files that merge HTML with dynamic data to produce formatted output. Templates use special markup tags called GX tags.
- Query files are text files that contain SQL commands, such as for querying or updating a database.

- Extensions enable business logic components to connect to custom or proprietary enterprise resources. Extensions are persistent modules, written in C++ or Java, that are dynamically loaded into NAS and are accessed by multiple AppLogics or EJBs over the life of the extension.
  - — Ready-to-use extensions are provided as part of NAS Integration Solutions. NAS Integration Solutions are available for MQSeries, Tuxedo, CICS, IMS, and R/3 applications. These applications can be easily integrated with applications deployed on NAS.
  - — Developers can create their own custom extensions using Netscape Extension Builder, a separately packaged, GUI-based tool that integrates with NAS.

For more information about AppLogics, HTML templates, and query files, see the *Programmer's Guide (C++)* or the *Migration Guide*. For more information about extensions, consult your Netscape Extension Builder documentation.

Core Services

NAS provides a set of application services and system services for building, deploying, and managing high-performance, scalable, transactional applications. These services include built-in state and session management, request and transaction management, results caching, connection caching, and so on.

For more information about these services, see the *Overview Guide.*

A Choice of Tools

Netscape Application Server supports applications written in either Java or C++. However, Java applications are easier to develop and maintain, because they can take advantage of the enhanced, standards-based application model.

Application developers can choose from among various tools to build applications. These tools can range from simple text editors, to visual Java editors and visual HTML editors, to integrated development environments (IDEs).

Netscape's tools include Netscape Application Builder and Netscape Extension Builder. These tools are tightly integrated with NAS but are packaged separately.

- Netscape Application Builder is used for building applications and deploying them on NAS. Netscape Application Builder also interoperates with third-party tools such as Symantec Visual Cafe.
- Netscape Extension Builder is used for designing and building custom extensions.

High Scalability

Netscape Application Server has a scalable architecture. This means that applications built to meet the needs of initial deployment can adapt as these business needs grow.

Application scaling is accomplished in two main ways: either by adding more servers to a cluster of servers, or by adding more CPUs to a multi-CPU system. Application logic can then be deployed to the new servers. Developers do not need to change any application logic as the user base grows.

In addition, application tasks can be assigned to the server best able to process the request efficiently. This is accomplished either through application partitioning or through dynamic load balancing.

Application Partitioning

The Netscape Application Server architecture supports application partitioning, which allows logic to be distributed across servers as an application scales to accommodate heavier loads. Using Netscape Application Server Administrator, system administrators can partition an application into functional areas.

For example, in an online catalog application, the application logic for order processing, inventory management, and checkout processing can reside on different servers. Regardless of how applications are partitioned and distributed, the application functions as a single, cohesive unit.

Application logic can also be grouped where each group consists of related operations. For example, a group might contain all logic associated with order processing. Each application component can belong to one or more groups. Applications can also share application logic.

System administrators can deploy these groups of application logic objects locally or globally across application servers in the following ways:

- Portions of an application might uniquely reside on different Netscape Application Servers, yet still run as a single application. In this way, application logic can be stored on the server that can run it most efficiently. For example, data-intensive application logic can be run on the server that is closest to the data source to avoid latencies associated with accessing remotely located data.
- For load-balanced applications, the same group of application logic objects can be stored on multiple servers. This allows an application to run the application logic more efficiently on the server with the most available resources.
- Applications might dynamically share certain application logic objects. For example, all applications in a network might share the same application logic for user login and authentication, or for credit card authorization.

Application partitioning gives system administrators tremendous flexibility to scale and tune the performance of applications. In addition, having application components stored on multiple servers helps ensure high application availability in the event of a server shutdown.

Dynamic Load Balancing

Netscape Application Server supports dynamic load balancing to provide optimal performance levels under heavy loads. With load balancing enabled, the Netscape Application Server can direct certain requests to be run on an available server instead of waiting for a busy server to become available. If one server is overburdened, another can take its place to process the request. The Load Monitor tabulates information on server resource availability.

To use load balancing, application logic must be partitioned across all Netscape Application Servers that might process the application logic at run time. System administrators must determine if an entire application or specific parts of an application should be load balanced.

For optimal performance and resource utilization, system administrators can deploy application logic on servers that are best configured to handle execution of that logic. Each Netscape Application Server has its own load balancing module which determines the optimal server to process an incoming request.

Partitioning characteristics are dynamically known to all servers in the cluster. Netscape Application Servers regularly update their load statistics and broadcast them to other Netscape Application Servers in the cluster. Based on load balancing factors, requests are dynamically routed to servers. Load balancing factors are configured using Netscape Application Server Administrator.

High Performance

Netscape Application Server is a high performance, multi-threaded, and multi-processing application server. NAS can handle a high number of concurrent requests, database connections, and sessions, and provides high performance even under heavy loads.

NAS delivers high performance between web servers, other NAS machines, and heterogeneous back-end data sources through the following features:

- Database Connection Caching
- Result Caching
- Data Streaming
- Multi-threaded Capabilities
- Optimized Communication with Web Servers

Aside from the application server, other factors affecting application performance include network topology, network and server hardware, database architecture, and application programming.

Database Connection Caching

To improve performance, the Netscape Application Server caches database connections so that commonly used, existing connections are re-used rather than re-established each time. Connection caching avoids the overhead involved in creating a new database connection for each request.

Application developers can enable database connection caching in their application logic. At run time, when a request creates a new connection to a database, Netscape Application Server stores the connection in the cache. When the request has completed using the connection, the connection is marked as free in the cache. If a new request is made to the same database by the same user, Netscape Application Server can first check the cache and use an existing free connection rather than creating a new one. If the freed connection remains unused after a specified time-out period, it is released by the server.

System administrators can use the Netscape Application Server Administrator to specify server-wide settings for database connection caching, such as the initial number of slots in the cache and the time-out limit for free connections, and so on.

Using Netscape Application Server Administrator, system administrators can monitor server performance and tune the number of available connections in the cache. This ensures an optimal ratio of cached connections to system resources.

Result Caching

Netscape Application Server improves application performance by caching the results of application logic execution. Developers can optionally enable this feature in their applications.

If caching is enabled, Netscape Application Server saves the application logic's input parameters and results in the cache. The next time the Netscape Application Server executes the same request, the server can first check the cache to determine whether the input parameters match the cached input parameters. If they match, the server retrieves the results from the cache instead of executing the request again. Result caching is especially effective for large data requests that involve lengthy processing time and for frequently accessed application logic.

Data Streaming

Netscape Application Server provides data streaming facilities. Streaming improves performance by allowing users to begin viewing results of requests sooner, rather than waiting until the complete operation has been processed. Application developers can explicitly control what data is streamed, or allow the system to provide automatic streaming.

Streaming is especially useful for large data sets involving lengthy queries. For example, suppose a user requests a price list containing 10,000 items. The application can process the query and display items to the user as they become available, for example, 40 at a time (or one full page view), rather than wait until all 10,000 items have been retrieved from the database.

Multi-threaded Capabilities

Netscape Application Server supports the multi-threading capabilities of the host operating system. An application can optimize performance by processing requests on multiple threads, which maximizes CPU resource utilization.

Application developers automatically take advantage of multi-threading in their applications. In addition, developers can run database operations such as queries, inserts, updates, deletes, and so on, asynchronously. Asynchronous operations allow an application to do other work while a time-consuming operation, such as a large query, runs in the background.

System administrators can use Netscape Application Server Administrator tool to specify settings for multi-threading, such as the following:

- minimum and maximum number of threads to handle all requests
- minimum and maximum number of threads to handle asynchronous database requests.

Typically, administrators will monitor server performance and tune the number of available threads to achieve an optimal ratio of threads to system resources.

Optimized Communication with Web Servers

Netscape Application Server optimizes application performance through tighter integration with web servers. This integration occurs using Web Connector Plug-ins and corresponding Listeners. Netscape Application Server supports NSAPI, ISAPI, and optimized CGI for Netscape, Microsoft, and CGI-compatible Web servers, respectively.

High Availability

Many enterprise applications must be accessible (available) to users 24 hours a day, 7 days a week. Netscape Application Server provides a highly available and reliable solution through the use of load balancing and dynamic failover (also called failure recovery).

NAS enables you to distribute all or part of an application across multiple servers. As a result, if one server goes down, the other servers can continue to handle requests.

NAS minimizes downtime by providing automatic application restarting. In addition, NAS maintains and replicates distributed user-session information and distributed application-state information. Information is maintained as long as more than one NAS installation is running in a cluster with the server that crashed.

Developers need not be concerned with building recovery and scalability features into their application. The application inherits these features simply by being hosted on the runtime environment.

For more information about load balancing and failure recovery, see the *Overview Guide*, the *Administration Guide* or the *Deployment Guide*.

Session and State Management

Netscape Application Server supports state and session management capabilities required for web-based applications. NAS provides a number of classes and interfaces that application developers can use to maintain state and user session information.

State and session information is stored on each server in a distributed environment. For example, an application can display a login screen, prompt users to enter their user name and password, then save this information in a session object. Thereafter, the application uses this same information to log in to multiple databases without prompting the user to type it in again.

Similarly, in an online shopping application, a session object can store a list of products selected for purchase (such as quantity, price, and so on) and persistent variables (such as running order totals).

State and session management is especially important for applications that have complex, multi-step operations. In an environment where application logic is partitioned across different servers, system administrators can use the Netscape Application Server Administrator to optionally designate a single server to act as the central repository for all state information.

For more information about session and state management, see the *Programmer's Guide.*

Security

Netscape Directory Server provides NAS applications with data security by using access control lists, Secure Sockets Layer (SSL), and password policies.

In addition, NAS provides secure web server communication and supports SSL, HTTPS, and HTTP challenge-response authentication. To bridge the security gap between browsers and data sources, NAS supports user authentication, cookies, and database access controls for the secure handling of transactional operations. Event logging and tracking enables detection of, and protection against, unauthorized access.

User Authentication

NAS Administrator and Netscape Directory Server provide facilities to enable user authentication to ensure that only authorized users can access applications, databases, and directories.

Server administrators can use the security tool of the Administrator to create users, groups, and roles to manage access to specified resources.

Access Controls to Data Sources

Netscape Application Server works within the framework of existing access controls for relational database management systems. A user or application must log into the database before gaining access to the data.

Developers can write applications so that users enter login information only once and the application saves the information in a session object. Thereafter, the application uses the initial login information to log into different databases, as needed, in the background without requiring additional user input.

Netscape Application Server shields back-end data by acting as a secure gatekeeper between the web server and the relational database system.

Management Capabilities

Netscape Application Server eases application management by providing integrated management facilities. These facilities include the following:

- Netscape Application Server Administrator
- Netscape Directory Server
- Support for Third-Party Management Tools

Netscape Application Server Administrator

Netscape Application Server (NAS) Administrator is a Java application with a graphical user interface. NAS Administrator enables the following capabilities:

- remote management of multiple servers and distributed applications.
- dynamic deployment and scaling of applications.
- performance tuning and optimization of the server environment.

Management and tuning involves tasks such as adjusting database connection threads, adjusting load-balancing parameters, configuring web servers, and managing access control lists (ACLs).

Additional features of NAS Administrator include:

- Dynamic Application Management
- Event Logging and Failure Analysis

Dynamic Application Management

Netscape Application Server's architecture allows partitioned applications to run even if one or more servers fail. In a load-balanced server configuration, application logic can be replicated on multiple servers. If a server fails, the load balancing module dynamically directs requests to other available servers, thus preventing application-wide failure.

Because the NAS architecture promotes high availability of applications, server administrators can use NAS Administrator to perform a variety of tasks in real time, without interrupting an application's operation. These tasks include:

- monitoring, reconfiguring, or replacing servers.
- swapping out or updating application components.

Event Logging and Failure Analysis

Netscape Application Server provides facilities for logging requests from Web servers and logging system-level and application-level events on Netscape Application Servers. For deployed applications, system administrators can use contemporaneous logs to assist with failure analysis and to detect attempted security breaches.

Event logging occurs in multiple ways on the Netscape Application Server:

- Application developers can enable logging in their application logic to assist with failure analysis. For example, an application can log messages like “Transaction succeeded” or “Transaction failed” depending on conditions and events at run-time.
- System administrators can enable automatic event logging to record the messages generated by dynamically loadable modules (DLMs) and application execution.
- System administrators can enable HTTP request logging to record and monitor the requests received by a Web server. Administrators can specify brief, normal, or detailed logging. If logging is enabled, Netscape Application Server logs information about the HTTP requests to a specified target database. Administrators can then analyze the logs, generate custom reports, and so on. HTTP request logging requires NSAPI or ISAPI Web Connectors.

Netscape Directory Server

Netscape Directory Server, which is packaged with NAS, is Netscape’s implementation of the Lightweight Directory Access Protocol (LDAP). NAS uses Netscape Directory Server not only to store NAS configuration data but also as a central repository for user and group information. A single Netscape Directory Server can support multiple instances of NAS—up to five clusters, in fact. This means that administrative data for all NAS installations can be centralized in one place.

The NAS Administrator acts as an LDAP client and can access information about users and groups. As a result of this integration with LDAP, NAS provides unified management of users, groups, and ACLs across the enterprise.

Support for Third-Party Management Tools

NAS provides the ability to be monitored and managed via SNMP agents. SNMP is a protocol used to exchange data about network activity.

NAS stores variables pertaining to network management in a tree-like hierarchy known as the server's management information base (MIB). Through this MIB, NAS exposes key management information to third-party tools that run SNMP. As a result, NAS can integrate with an enterprise's server management tools, thereby allowing other solutions for remote administration.

Chapter

2

Installing NAS on Solaris 8

This chapter explains how to install Netscape Application Server (NAS) on the Unix platform using the quick installation script. After installation you can start developing applications on your own. This chapter includes the following topics:

- About Installing NAS
- Checking Software Requirements
- Running the Quick Installation Script
- Verifying Installation

Note If you wish to use the standard product installation script as detailed in the *Installation Guide*, please contact your sales representative to obtain a product key or call our sales office at 888.786.8111. You will not be able to proceed with standard installation without a key.

About Installing NAS

If you are new to the Netscape Application Server, this evaluation version includes a quick installation program to get you up and running on NAS quickly and easily. In addition, a number of sample applications have been provided to assist you in your exploration of the server. These applications will help you to learn about the various components available to you when developing a new application.

For those already familiar with the Netscape Application Server, the standard installation as defined in the *Installation Guide* can be used in place of the quick installation.

Checking Software Requirements

Before running the quick installation script, verify the following:

- iPlanet Web Server 4.0 is installed on the system. See the *iPlanet Web Server Installation Guide*.
- Oracle 8.0.5 client library is installed on the system and Oracle database environment variables are set. Though the Oracle 8.0.5 client libraries are required, the Oracle 8.0.5 server or later can be used for the Oracle server itself. This is necessary for the operation of the Online Bookstore and online bank sample applications.

  You should verify that you can connect to the Oracle server from the application server machine by using an Oracle utility such as `sqlplus.` This will ensure that the basic Oracle networking environment is functioning properly.

Note If you do not have Oracle installed on the system, you can still run the quick installation script and perform a basic evaluation of the Netscape Application Server. The “Fortune” sample application as described in Chapter 3, “Using the Sample Applications” can be used to verify the installation.

Running the Quick Installation Script

Log on to the system as the same user (or as a member of the same group) that installed the iPlanet Web Server with which your Netscape Application Server will interface. If you install as a regular user, and elect to configure the Netscape Application Server for automatic startup, you will have to log on again as the root user after you install to enable automatic startup.

Note iPlanet Web Server 4.0, iPlanet Directory Server 4.0, and NAS 4.0 SP2 will all be installed on the same machine. Futhermore, NAS will install and use its own copy of the directory server. It will not use existing directory servers, if any.

To install Netscape Application Server 4.0 SP2

The following steps can be performed as root user. Before proceeding with this installation, you should uninstall any previous installations of NAS.

1. Ensure that the ORACLE_HOME environment variable is set appropriately. If it is not set to the appropriate directory then the installation will appear successful but NAS will be unable to access the database while running the sample application.

2. Insert the Solaris 8 CD-ROM into the CD-ROM drive and mount the CD-ROM on, for example, `/cdrom/cdrom0`.

3. At the shell prompt, run the following command:

   ```
   cd /cdrom/cdrom0/NAS/NAS4.0
   ```

4. Execute the installation script by typing the following:

   ```
   ./quick_install
   ```

   You are then prompted to specify the following information.

5. Specify a target installation directory as the base directory within which all NAS components are installed (includes both NAS and the directory server). Do not include spaces in the path name. The default location is:

   ```
   "/usr/netscape/server4"
   ```

6. Specify the full path for the Web server instance. No default value is set. You will not be able to continue until a path is specified.

   The full path for the Web server instance is the root directory where iPlanet Web Server (iWS) is installed plus the https subdirectory for your server name. For example, your iWS installation directory is `/usr/netscape/server4` and your Web server instance name is *`myiws`*. The full path for your Web server instance is then `/usr/netscape/server4/https-myiws.<`*`your domain name`*`>.com`

7. Specify an Administrator's password. Your user name / password combination will be used to administer NAS via the Netscape Application Server Administration (NASA) console. The default administrator's name is set to "admin." The default password is "changeit".

Verify Web Server Configuration

Once you've finished the installation you'll need to ensure that the web server process is running with the same Solaris user ID as NAS. For Example, if NAS was installed with the root user ID, the web server process must also use the root user ID. To verify that the web server is using the same ID, type:

```
ps -ef | grep httpd
```

then look for your web server instance. If it is not using the same user ID as the application server, you can change it by using the web server administration console.

If you need to change the Web Server user ID:

1. Access the web server administration console via the appropriate URL.
2. Select "View Server Settings"
3. Edit the "User" entry located under "Technical Settings"

Next, you'll need to update and restart your webserver. During the installion of NAS, changes were made to the web server's configuration file. So, you'll need to apply these changes thru the web server administration console and then restart the web server.

Verifying Installation

You can use the "Java Fortune" sample application installed with NAS to verify your installation. This servlet based application will return a unique fortune to the user each time he/she clicks the reload button on the browser. This servlet does not rely on a back-end database.

To verify installation:

1. Open Netscape Navigator, enter the following URL, and press Enter:

   `http://`*`yourwebserver:portnumber`*`/GXApp/index.html`

2. Click the link for the Java Fortune sample application.

   The sample application returns a fortune. Click the browser's reload button to see a new fortune.

Note The Netscape Application Server Administrator (NASA) console can be started and used to manage your NAS runtime environment. As noted in the quick installation procedure, your user name is set to "admin" by default and your password is set to "changeit" by default, unless you specified otherwise during your installation. For more information on this tool and its features, see the *Administration Guide*.

Chapter

3

Using the Sample Applications

A number of sample applications have been provided to assist you in your exploration of the server. These applications will help you learn about the various components available to you when developing a new application. Once you have installed Netscape Application Server and your database servers and clients, you can use the sample applications to ensure that the server is working properly and as a guide for developing your own applications. This chapter covers the Online Bookstore application. For information about the other sample applications installed with NAS, see the *Installation Guide* and *Programmer's Guide*.

The Online Bookstore application discussed in this section demonstrates the J2EE oriented programming model and its implementation of the following technologies: EJBs, servlets, JSP, JDBC, and transaction manager. Note that this sample application requires the Oracle 8.0.5 client library to run as described on page 34.

It is recommended that you first run the Fortune application to test that NAS has been installed properly (see "Verify Web Server Configuration" on page 36). Then, after verifying that the server is installed properly, run the Online Bookstore application to understand the NAS 4.0 programming model and its implementation of servlets, JSPs, EJBs, JDBC, and transaction manager.

In the instructions that follow, *`nas install directory`* refers to the directory where you have installed NAS.

Online Bookstore Sample Application

This section explains how to configure the Online Bookstore sample application.

Note The quick installation described in Chapter 2, “Installing NAS on Solaris 8” configures NAS for Oracle 8.0.5 client libraries. However, DB2 and Sysbase client libraries are also supported by NAS. For more information regarding the use of these client libraries, refer to the *Installation Guide* and the *Programmer’s Guide.*

Configuring the Oracle Database

You should ensure that you have an Oracle user defined for the sample application. You can use either an existing Oracle user or define a new Oracle user for the sample application.

To define a new Oracle user:

1. Use SQL Plus to define an Oracle user name (e.g. “kdemo”).

   ```
   SQL > create user kdemo identified by kdemo;
   ```

   Where the first “kdemo” is the user name and the second “kdemo” is the password.

2. Grant connect, resource, and dba privledges to the “kdemo” Oracle user.

   ```
   SQL> grant connect, resource, dba to kdemo identified
     by kdemo;
   ```

To create and populate the sample database tables:

A shell script named `setup_ora.sh` configures the database tables, populates the tables with sample data and updates the NAS registry with database information to support the Online Bookstore sample application.

The setup_ora.sh script performs the following steps:

1. Registers the datasource in the NAS registry.
2. Uses SQL Plus to test access to the Oracle database.
3. Creates and populates the sample tables, if the access test is successful.

To run the script, follow these steps:

1. Ensure that the ORACLE_HOME environment variable is set appropriately.
2. From the `<nas install directory>/APPS/nsOnlineBookstore/database directory`, run `setup_ora.sh` with the following arguments:

   ```
   setup_ora.sh <DataSource> <Database> <DBuser>
     <DBpassword> <ResourceManagerName>
   ```

   For example:

   ```
   setup_ora.sh ksample ORCL kdemo kdemo rm1
   ```

   Where:

   `DataSource` is the Oracle service or TNS name which maps to the appropriate Oracle database instance. In the example above, “ksample” is the Oracle service or TNS name. (This assumes that the Oracle client has been configured to recognize “ksample” as a valid Oracles service/TNS name.)

   `Database` is the Oracle SID of the target database.

   `DBUser` and `DBpassword` are the Oracle user name and password under which the tables will be created and populated. These are also the values that will be used by the sample application to access the Oracle database.

   `ResourceManagerName` is a dummy value that must be supplied, but is not currently used the by the sample application (e.g. `rm1`)

Cleanup Resource Manager Entry in NAS Registry

Due to a bug in the setup of the sample application, a distributed transaction resource manager entry is added to the NAS registry. Before running the Online Bookstore application, you must remove this entry from the NAS registry.

To cleanup resource manager entry in NAS registry:

1. Run *$(NAS_INSTALL)/nas/bin/kregedit*
2. Locate the registry key *\\SOFTWARE\Netscape\Application Server\4.0\DataSource\nsOnlineBookstoreDS\Resource Manager=rm1*

   Where `rm1` is the ResourceManager name you specified when you created the database tables.
3. Select the ResourceManager registry key and choose Edit > Delete from the pull-down menu at the top banner.
4. Exit from kregedit.
5. Restart NAS.
   1. Use the following command to stop NAS:

      $(NAS_INSTALL)/nas/bin/KIVAes.sh stop

   2. Use the following command to start NAS.

      KIVAes.sh start

Creating a Group and a User in the Directory Server

The sample application has two primary features: the ability to purchase books from the bookstore and the ability to manage the bookstore from the manager's office. To manage the bookstore, you must create a user with special privileges. Because the sample application takes advantage of LDAP integration, any user you create is verified by the Directory Server configured with NAS. Therefore you must create this user on Directory Server.

1. Start the Netscape Console.

   ```
   /usr/netscape/server4/startconsole
   ```
2. Enter the Console Administrator user ID and password. As noted in the quick installation procedure, the default user name and password are "admin" and "changeit".

3. Click the “Users and Groups” tab.
4. Choose New User in the drop-down list on the lower right portion of the panel and click Create.
5. In the "Select Organizational Unit" dialog, select People and click OK.
6. Enter the name, user ID, and password for the user you want to create and click OK. (For example, you could choose to set your user name and password to be “bookmgr” and “bookmgr”).
7. Click the "Users and Groups" tab.
8. Choose New Group in the drop-down list on the lower right portion of the panel and click Create.
9. In the "Select Organizational Unit" dialog, select Groups and click OK.
10. Under Create Group, enter the name of the Group you want to create, for example: BookAdmin. If the group already exists, go to Step 11.
11. Click the Members tab and then click Add.
12. Click Users and then click Search.
13. Choose the name of the user you created in Step 6 and click OK.
14. Exit from the console.
15. If you created a group other than the one named “BookAdmin” in Step 10, do the following:
    1. Open the file `*nas install directory*/APPS/GXApp/nsOnlineBookstore/ldap/ldapInfo.properties`
    2. Add the following line in the file: `ADMIN_GROUP_DN = cn=*name of group created in Step 10*, ou=Group, o=mcom.com`
    3. Save the file.

The user you created is stored on the Directory Server with which this NAS installation is configured. Whenever you administer the application, this user is verified by the Directory Server.

Running the Sample Application

To run the sample applications on a NAS installation that includes the Web Connector plug-in, run the `sample.sh` script. Then click the link for the sample application.

To run the sample applications on a NAS installation that does not include the Web Connector plug-in, do the following:

1. Ensure that NAS is running.

   If you're not sure if NAS is running, perform the following steps:

   1. Stop NAS:

   `$(NAS_INSTALL)/nas/bin/KIVAes.sh` stop

   2. Start NAS:

   `KIVAes.sh` start

2. Open Netscape Navigator, enter the following URL, and press Enter:

   `http://`*`yourwebserver:portnumber/`*`GXApp/index.html`

3. Click the link for Java Online Bookstore

Once the Online Bookstore application is up and running you can enter the bookstore in one of two ways -- as a customer or as a manager.

Entering the Bookstore as a Customer

To enter the bookstore as a customer, click the Store Entrance image on the left portion of the page. There you will find the shopping area of the application. You will have the option of placing an order as a new customer or as a customer who has visited the site before.

If you choose to place an order as a new customer, simply go through the process of buying a book. When you are ready to check out, you will be prompted to register as a new customer.

If instead, you would like to go through the shopping experience as a pre-existing customer, you can use one of the pre-defined user names and passwords provided with this installation.

To purchase a book as a pre-existing customer do the following:

1. When prompted for the use name type:

   `user<#>@mycorp.com`

   where <#> is any number from 1 to 10.

2. When prompted for the password type:

   `user`

Entering the Bookstore as a Manager

To enter the bookstore as a manager, click the Manager’s Office image on the right portion of the page. There you will find the administration area of the application. Use the user name and password that you added to the directory server earlier (e.g. “bookmgr”, “bookmgr”) to authenticate yourself.

As the manager, you will see a list of customers and the orders they have placed. However, if you have not walked through the shopping experience as a customer, you will not see any book orders when you enter the Manager’s Office, since no orders have been placed.

For more detailed information regarding the Online Bookstore sample application, refer to the NAS *Programmer’s Guide*.

Updating the Sample Application

If you change any of the source files of the Online Bookstore application after installing and configuring it, you must invoke the `defaults.mak` file, located in <*`nas install directory`*>`/APPS/GXApp/nsOnlineBookstore/src`, to rebuild the sample application.

To invoke the makefile, which includes `defaults.mak`, type the following command at the prompt:

```
make -f makefile
```

If you want to update source files in a particular directory, run the makefile located in that directory.

Chapter

4

Developing Applications

This chapter describes the relationships between files used in creating applications to run on the Netscape Application Server. In addition, this chapter describes in detail the Online Bookstore sample application.

The following topics are described in this chapter section:

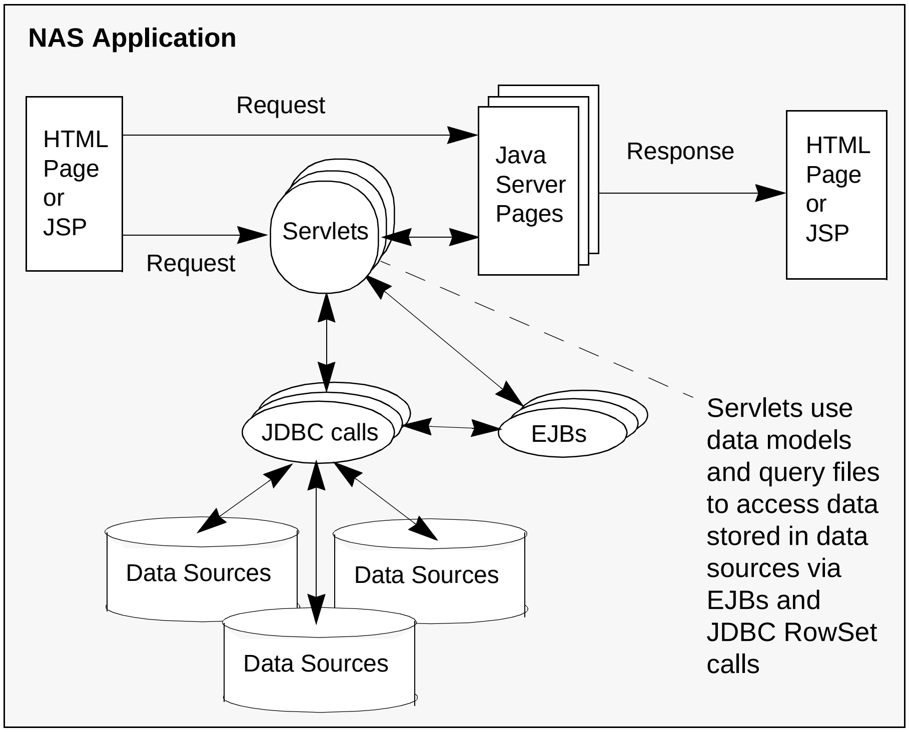
- Relationships between files
  - — Parts of an Application
  - — Assembling the Pieces
  - — Getting Started
- Online bookstore sample application description
  - — Inside the Online Bookstore Sample Application
  - — Presentation Flow
  - — Directory Structure
  - — Servlets
  - — EJB Functionality

Parts of an Application

In the Netscape Application Server environment, an application can contain three types of programming layers. As the developer, you use or create objects to deliver each specific layer, as listed in the following table:

Application Layer	Objects Used
Data access layer	Query files, data models, and JDBC RowSet classes
Presentation layer	Java servlets for the presentation logic; JavaServer Pages and HTML files for the presentation layout.
Business layer	Enterprise Java Beans

The relationships among the objects which represent the various application layers are shown in the following figure and are explained in the three sections that follow.



Data Access Layer

Data access logic determines what types of back-end data sources (if any) the application accesses, such as a relational database or a legacy system. In this guide, it is assumed that a data source already exists for use with your application, and that there is someone at your site, such as a database administrator, who manages it. Netscape Application Builder enables you to access your data sources via data models and data access query files.

To legacy systems, your code may need to call into an extension. Netscape Application Server includes prebuilt extensions. These extensions allow an application to communicate with transaction-processing systems such as CICS/IMS, IBM MQ Series, and BEA Tuxedo. Additional prebuilt extensions allow communication with ERP systems such as SAP R/3. If you need to create a custom extension, use Netscape Extension Builder.

Relational Data Sources

If the application accesses a relational database, you use queries to access the data. To use a query, create two types of files:

- a **data model**, which specifies the database tables and relationships used by the application.
- a **query file**, which contains one or more queries, each of which defines a single interaction with a relational database.

You cannot create a query without a data model. Every query file depends on a data model, but not necessarily the same data model. A query can be loaded into a `NASRowSet` object.

Nonrelational Data Sources

If the application is required to access a nonrelational data source, you must write Java code to communicate with the data source. In some cases, you can call into an extension that is prebuilt into Netscape Application Server. In other cases, you may want to build a custom extension using Netscape Extension Builder, another product in the NAS product line.

Presentation Layer

The presentation layer handles the presentation logic and layout of the application. The following sections introduce these concepts.

About Presentation Logic

Presentation logic controls and processes the data generated from application users by invoking Enterprise JavaBeans to perform business logic functions, and then generating a dynamic page that sets up the next user interaction. Presentation logic is processed by servlets on the Netscape Application Server. Servlets handle such tasks as page-to-page navigation, session management, simple input validation, and the tying together of business logic.

About Presentation Layout

Presentation layout determines how users experience the application. Typically they use a web browser and navigate from page to page (HTML pages or JavaServer Pages). For example, suppose a user is viewing the left web page in the previous figure. As a result of a user action, such as pressing a Submit button, the web browser displays the page on the right. The intervening processing remains transparent to the user.

Business Logic Layer

Business logic maintains the application-specific processing and business rules of an application.

The application components designed for your business logic connect the components designed for the presentation layer (servlets, HTML pages, and JavaServer Pages) to the components designed to implement the data access layer (data models, queries, and JDBC RowSet calls).

Enterprise JavaBeans

In the Netscape application model, you design session and entity EJBs to implement the various types of business logic, such as transactions, security, and remote access.

Assembling the Pieces

To create an application, you assemble files that execute the presentation logic, business logic, and data access logic that the application requires. Assembling files means either creating new files or locating existing files to add to a project, which is a container for your files during development. A project also lists the files you will deploy as an application, after development is completed.

Types of Files in a Project

In developing an application, you assemble the following types of files:

File Description	File Suffix
HTML pages and templates	`.html`, `.htm`
JavaServer Pages	`.jsp`
JavaScript	`.js`
Image files	`.gif`, `.jpg`, `.jpeg`
Servlets, EJBs, and other Java code	`.java, .ebx, .class`
Query files	`.gxq`
Data models	`.kdm`
Connection files	`.props`
NTV files	`.ntv`

How Files Relate to Each Other

The files in a project are related as follows:

- HTML pages and JavaServer Pages (JSPs) contain the web content. In addition to entering formatted text, you can insert links to image files. You also design HTML pages and JSPs that call JavaScript, Enterprise JavaBeans, and servlets.

- Servlets contain the application's business logic written in Java. For example, a servlet can call another servlet or return data via a RowSet object. The servlet handles data access, which may involve calling a query file to execute a query against a relational database. And when a servlet returns data via a RowSet object, it merges this output with a JSP.
- If the application must query a relational database, you create one or more data models, and one or more query files based on a data model.

Although you can use Netscape Application Builder to manually define the relationships between files, one of the benefits of Netscape Application Builder is its automated code generation wizards. Wizards guide you in creating sets of related files to perform commonly needed actions.

For example, many applications require login functionality, which derives from the interaction of several files. Using the Login wizard, you can quickly create the needed files, with the links between them automatically defined.

Getting Started

Before you launch Netscape Application Builder, the following conditions must be in place:

- Understand the Netscape Application Server Environment
- Define the Application Environment
- Define the Application Requirements
- Define the User Interface
- Set Up the Data Sources

Understand the Netscape Application Server Environment

Because you will be deploying applications to the Netscape Application Server, it is important that you understand the role of these applications within a three-tiered environment. Before using Netscape Application Builder, it is recommended that you become familiar with the concepts presented in the *Programmer's Guide*.

Define the Application Environment

What type of application do you want to design? Possible variations include: Internet, intranet, and extranet.

Some examples are shown in the following table:

Application Type	Examples
Internet	E-Commerce, Financial, Security, Portal, Internet Service Provider (ISP).
Intranet	Human Resources self-service, Sales force automation, internal help desk
Extranet	Supply chain management, insurance quotes, car dealership management
Workgroup	Accounting reports, RFE management

Define the Application Requirements

Once you've defined the application type, the next step is to gather the requirements of that application. Requirements are often determined by answering questions about the application's purpose or functionality. For example:

- Will users be anonymous, or will they be tracked by a username and password?

- Will the application support different user interfaces, such as multiple languages or different versions of a browser?
- What kinds of data can users query or update, and how is it accessed?
- Additional issues include: performance, throughput, capacity, scaling, reliability, and security.

The previous list is only a sample of questions to consider. For more information on gathering application requirements, see the *Programmer's Guide.*

Define the User Interface

Defining the user interface determines the page flow, which affects how an end-user navigates through the application. You define page flow by deciding what files will call other files. The general principle of page flow is shown in the previous figure, on .

The following questions can guide your decisions about user interface design:

- What is the page layout and flow for the application?
- What commands and buttons are available on each page?
- Will the pages use frames or not?
- Does the application require user validation?

  If so, you may decide to create a login page through which users access the application. Note that Netscape Application Builder provides a Login wizard to guide you in creating login functionality.
- How will data be entered and displayed? In particular, the following questions pertain to this issue:
  - How many HTML pages or JavaServer Pages require input forms?
  - What kinds of form elements are required for the HTML pages or JSPs?
  - Will database results be displayed together on a single page or displayed across multiple linked pages?

- Are there any corporate standards that determine headers and footers, logos, menu bars, or banner ads?
- Are there any international issues? For example, are the icons or cartoon images meaningful to all users? Does translation pose a formatting problem?
- Are the commonly used features easy to find?

Netscape Application Builder provides several wizards that make it easy to create some of the more common page designs. In addition, NAB includes a “project map” feature to verify page flow. In HTML Flow view, the project map displays web page relationships the way the end user will experience them. In File Dependencies view, the project map displays the relationships of all files, not just those seen by the end user.

For additional information on application design, see “Guidelines for Effective Development” in Chapter 2, “Designing NAS Applications,” in the *Programmer’s Guide*.

Set Up the Data Sources

A data source can be any of the following examples:

- relational database
- directory server
- legacy system (for example, mainframe files)
- client/server application (for example, Enterprise Resource Planning, or a spreadsheet)

Most web applications access a data source in some way; for example, when they display query results or when they allow users to update account information.

In the case of a relational database, a database administrator or data architect sets up and manages access at your site. Similarly for non-relational data sources, it is assumed that these systems are available and that they provide access to clients.

Writing Secure Applications

This section describes how to write secure application components that perform authentication to establish and maintain a user's identity.

The security model described in this section was developed for EJBs, and is part of the standard for EJBs. As of the 2.1 servlet specification, there is no standard security model for servlets. NAS has created a model for servlets in order to provide security throughout the application, with some collaboration with the developers of Java standards, so that the model presented here for servlets is based heavily on an emerging standard.

Inside the Online Bookstore Sample Application

The Online Bookstore sample application, which is shipped with NAS, simulates an Internet e-commerce site where customers can search for books, show details, add books to their shopping cart, place orders, and register for billing. Furthermore, bookstore managers can access information about customer usage and order status.

The user interface is a web browser that displays HTML pages and JavaServer pages. The application was designed to run on Solaris. In addition, the application can access an Oracle8 database and an LDAP server such as Netscape Directory Server.

Application Model

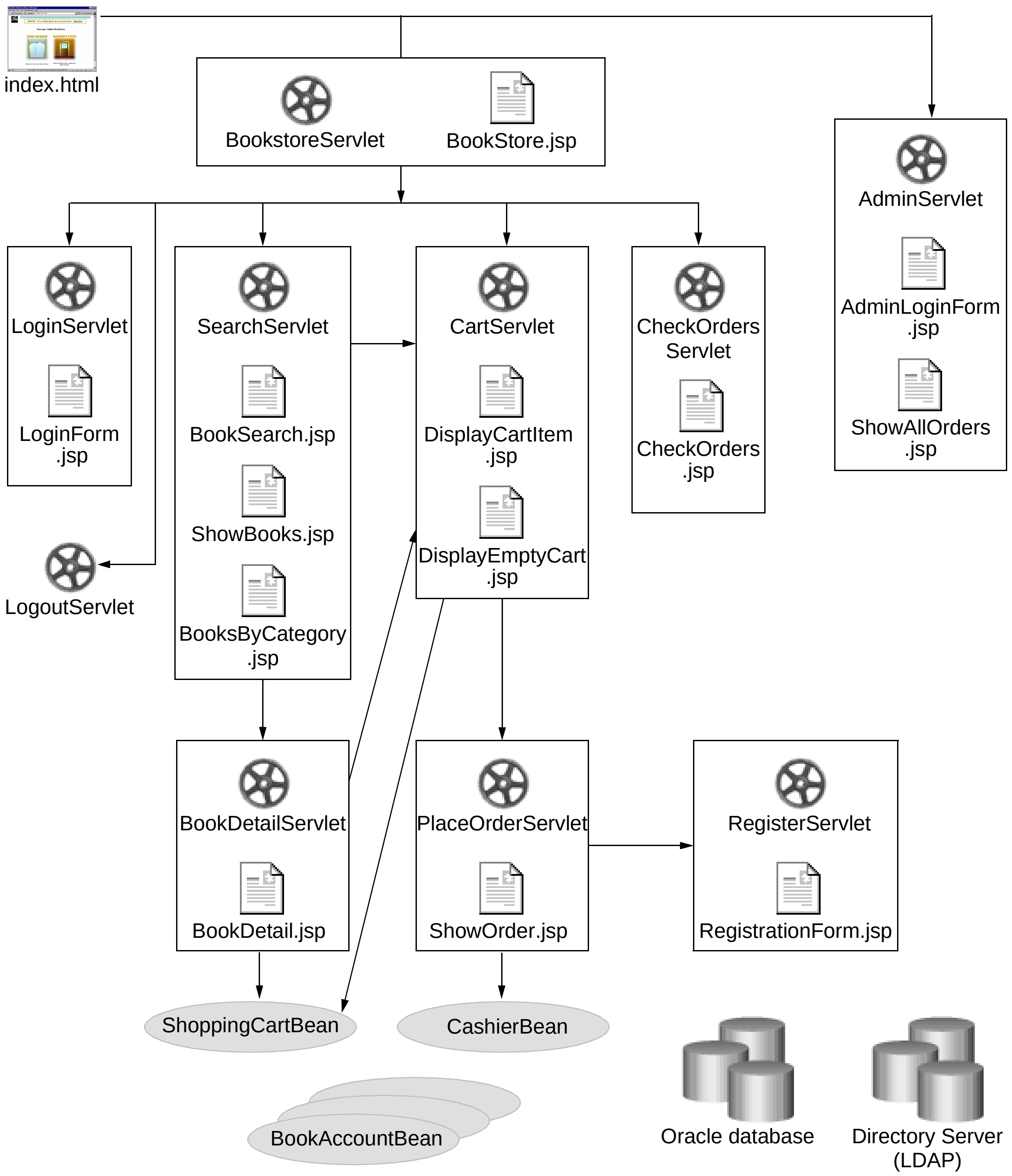
The Online Bookstore demonstrates the NAS application model by incorporating the following features:

- HTML
- JSPs
- servlets
- EJBs (stateful session bean, stateless session bean, and entity bean)
- database access through helper classes and JDBC APIs
- transaction processing, both local and global

- integration with an LDAP server

Application Flow

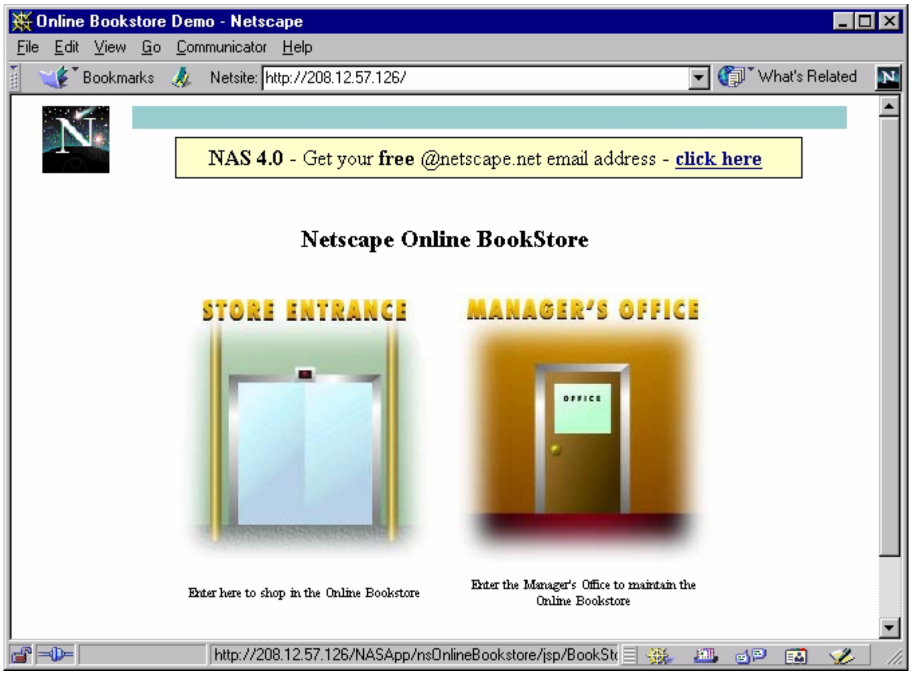
The flowchart on the following page summarizes the design of Online Bookstore. You can think of this application as a set of functional modules, each controlled by a servlet. This functional grouping is conceptual, not structural—a convenience for better understanding the application flow. This modular approach encourages parallel code development, through which different teams handle each design module at the same time, with minimal overlap.


index.html
BookstoreServlet
BookStore.jsp
AdminServlet
AdminLoginForm .jsp
ShowAllOrders .jsp
LoginServlet
LoginForm .jsp
LogoutServlet
SearchServlet
BookSearch.jsp
ShowBooks.jsp
BooksByCategory .jsp
CartServlet
DisplayCartItem .jsp
DisplayEmptyCart .jsp
CheckOrders Servlet
CheckOrders .jsp
BookDetailServlet
BookDetail.jsp
PlaceOrderServlet
ShowOrder.jsp
RegisterServlet
RegistrationForm.jsp
ShoppingCartBean
CashierBean
BookAccountBean
Oracle database
Directory Server (LDAP)

Presentation Flow

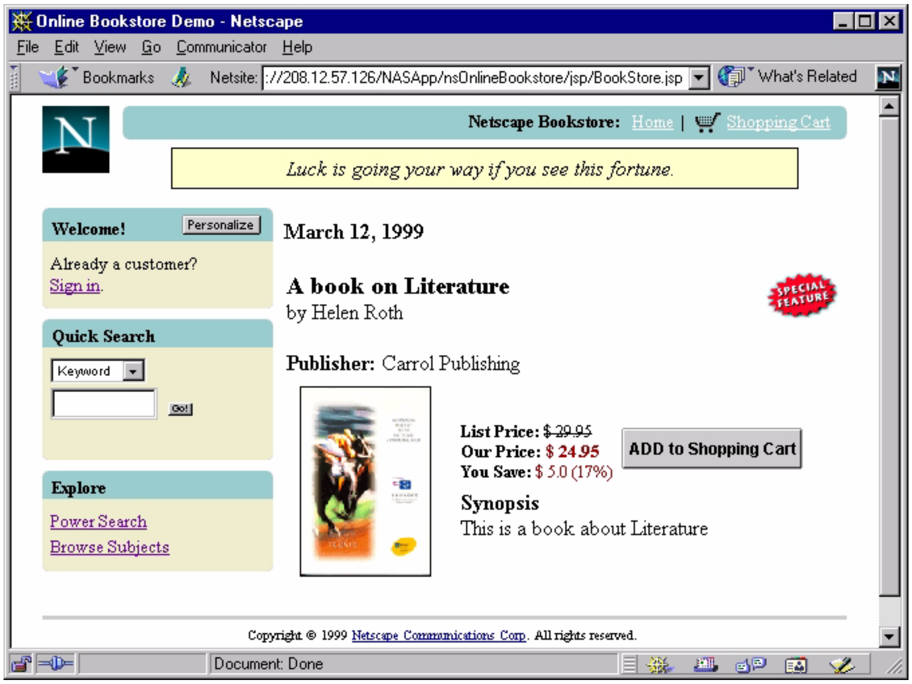
The following sequence describes a typical user session with the Online Bookstore:

1. From a web browser, the user navigates the Internet to reach the first page of the application, index.html.


Online Bookstore Demo - Netscape
File Edit View Go Communicator Help
Bookmarks
Netsite: http://208.12.57.126/
What's Related
NAS 4.0 - Get your free @netscape.net email address - click here
Netscape Online BookStore
STORE ENTRANCE
MANAGER'S OFFICE
OFFICE
Enter here to shop in the Online Bookstore
Enter the Manager's Office to maintain the Online Bookstore
http://208.12.57.126/NASApp/nsOnlineBookstore/jsp/BookSt

2. From index.html, users can follow two main branches: customers can click the Store Entrance image to display BookStore.jsp, and application administrators can click the Manager's Office image to invoke AdminServlet.

3. As a typical user, a customer clicks the Store Entrance image. The application then displays the main customer page, BookStore.jsp.


Online Bookstore Demo - Netscape
File Edit View Go Communicator Help
Bookmarks
Netsite: ://208.12.57.126/NASApp/nsOnlineBookstore/jsp/BookStore.jsp
What's Related
Netscape Bookstore: Home | Shopping Cart
Personalize
Keyword
Go!
ADD to Shopping Cart
SPECIAL FEATURE
Document: Done

Luck is going your way if you see this fortune.

Welcome!

Already a customer?
Sign in.

Quick Search

Explore

Power Search
Browse Subjects

March 12, 1999

A book on Literature
by Helen Roth

Publisher: Carrol Publishing

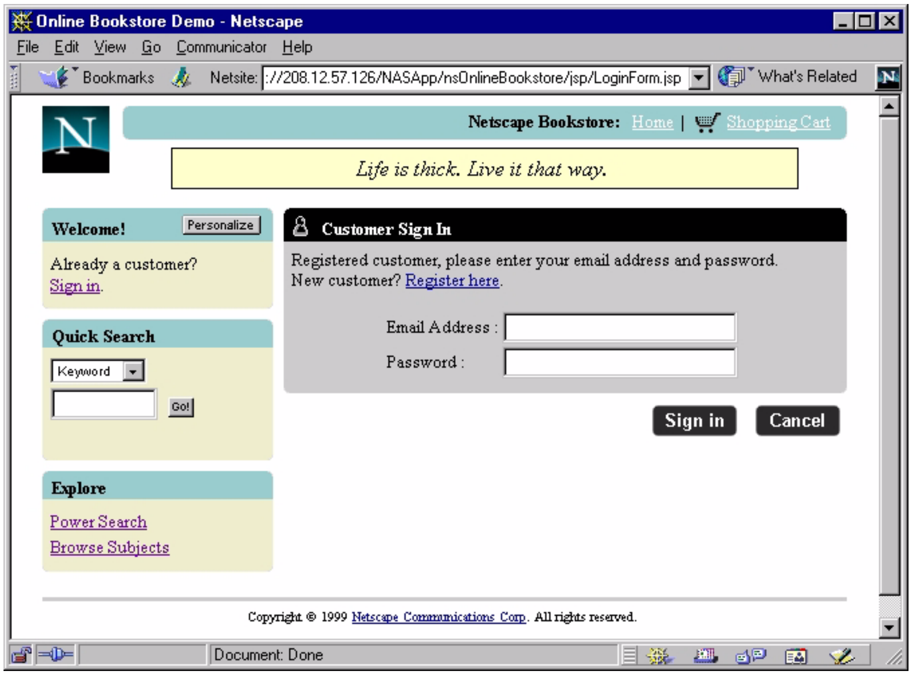
List Price: ~~$ 29.95~~
Our Price: $ 24.95
You Save: $ 5.0 (17%)

Synopsis
This is a book about Literature

Copyright © 1999 Netscape Communications Corp. All rights reserved.

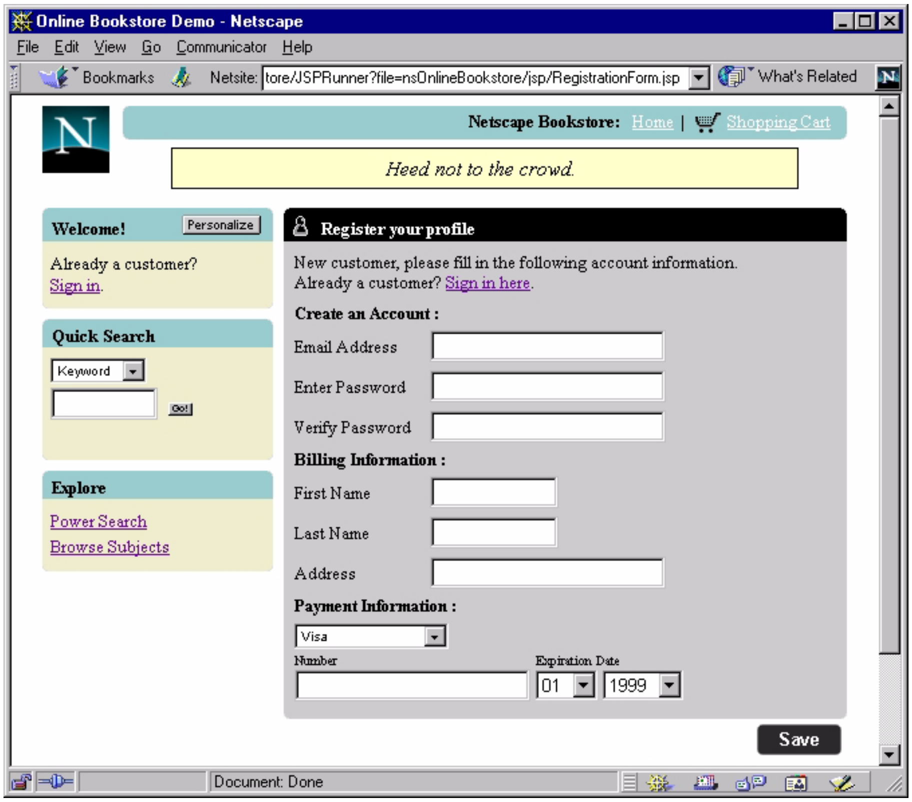
4. From the BookStore.jsp page, a customer can browse books by subject, search books by keyword, or add books to a shopping cart. Any of these tasks can be performed as an anonymous user. However, before placing an order or checking previous orders, a customer must sign in.

5. Clicking the **Sign In** link invokes the Login servlet. This servlet causes the right side of the page to display the Customer Sign In form, a JSP named LoginForm.jsp.


Online Bookstore Demo - Netscape
File Edit View Go Communicator Help
Bookmarks
Netsite: ://208.12.57.126/NASApp/nsOnlineBookstore/jsp/LoginForm.jsp
What's Related
Netscape Bookstore: Home | Shopping Cart
Life is thick. Live it that way.
Welcome!
Personalize
Already a customer?
Sign in.
Quick Search
Keyword
Go!
Explore
Power Search
Browse Subjects
Customer Sign In
Registered customer, please enter your email address and password.
New customer? Register here.
Email Address :
Password :
Sign in
Cancel
Copyright © 1999 Netscape Communications Corp. All rights reserved.
Document: Done

6. From the Customer Sign In form, customers can take either of two actions. Customers who previously registered can enter their email address and password, whereas new customers must click the **Register here** link.

7. Clicking **Register here** invokes the Register servlet. This servlet causes the right side of the page to display the Registration form, a JSP named RegistrationForm.jsp.


Online Bookstore Demo - Netscape

File Edit View Go Communicator Help

Bookmarks Netsite: tore/JSPRunner?file=nsOnlineBookstore/jsp/RegistrationForm.jsp What's Related

Netscape Bookstore: Home | Shopping Cart

Heed not to the crowd.

Welcome! Personalize

Already a customer?
Sign in.

Quick Search

Keyword

Go!

Explore

Power Search
Browse Subjects

Register your profile

New customer, please fill in the following account information.
Already a customer? Sign in here.

Create an Account :

Email Address

Enter Password

Verify Password

Billing Information :

First Name

Last Name

Address

Payment Information :

Visa

Number

Expiration Date 01 1999

Save

Document: Done

8. Regardless of whether a user is registered, the user can perform different types of searches from the left side of the page, using either the Quick Search table or the Explore table. These page elements invoke the SearchServlet.

9. When a results page appears, users can click a book title hyperlink. Clicking a book title invokes the BookDetail servlet, which in turn displays a JSP of item details.

10. If the book is of interest, the user presses a button to add the item to the shopping cart. This action invokes the CartServlet, which in turn displays a JSP of selected books. The displayed prices are calculated with the help of ShoppingCartBean.

11. When the user is ready to buy the items in the shopping cart, the user presses the Place Order button. This action invokes the PlaceOrder servlet. This servlet uses the CashierBean to calculate the charges, then displays the charges using the ShowOrder JSP.

12. The user can continue to navigate the application, searching for more books or checking previous orders, for example.

13. Eventually, the user clicks the Logout hyperlink to invoke the Logout servlet.

Directory Structure

The sample application includes component files organized into the following directories.

Directory	Contents
account	The BookAccount bean and related files to support accounting and inventory.
cart	The Cart bean and related files to support the selection of books.
cashier	The Cashier bean and related files to support the buying of books.
custom	Files involved with customizing the look and feel of the user interface. This functionality is invoked using the Personalize button on the Bookstore JSP.
database	Helper classes and other code for connecting to the database.
images	GIF and JPG files used in the web pages.
jsp	JavaServer Pages
ldap	A java file and property file to control LDAP access.
ntv	The .gxr file (used in deployment) and various configuration files.
util	Java files helpful to application developers.

In addition, many servlet files reside at the same level as the previous directories. These servlets are described in the next section.

Servlets

This section describes the servlets used in the Online Bookstore. The servlets are listed in the approximate order they would be encountered during a typical session.

BookstoreServlet, called from index.html, displays the Bookstore JSP. This JSP is the main customer page. Several of the application's servlets are invoked from the Bookstore JSP.

LoginServlet is invoked from the Bookstore JSP. This servlet launches LoginForm.jsp, which in turn displays the Customer Sign In form. LoginServlet verifies a customer's email ID and password against records in the database, then saves this information into an HTTP session.

SearchServlet is invoked from the Bookstore JSP. This servlet implements logic for basic keyword searches, for detailed, form-based searches, and for browsing by categories. SearchServlet also launches the JSPs necessary to input the search criteria or to display the results.

BookDetailServlet is called from any book title link, as might appear in ShowBooks.jsp and BooksByCategory.jsp. The servlet validates the bookID parameter and launches BookDetails.jsp to display more information about the item.

CartServlet manages the addition, deletion, and modification of shopping cart items. This servlet is called by the "Shopping Cart" link on the Bookstore JSP, as well as by buttons labeled "Add to Shopping Cart" or "Delete from Shopping Cart." CartServlet launches either of two JSPs: DisplayCartItem.jsp or DisplayEmptyCart.jsp. The ShoppingCartBean EJB determines the items and prices that appear.

PlaceOrderServlet is invoked from the "Place Order" button on DisplayCartItem.jsp. The servlet checks the customer object saved in the session. If the customer is registered, then control passes to the CashierBean EJB, the order is processed, and the results appear by way of ShowOrder.jsp. If the customer is not registered, then control passes to RegisterServlet.

RegisterServlet launches RegistrationForm.jsp, in which customers enter personal information. When users submit their data, RegisterServlet connects to a database, inserts the data, and saves the information into an HTTP session.

RegisterServlet is invoked under two circumstances: when the user clicks "Register Here" in the LoginForm JSP, or when the PlaceOrder servlet detects an unregistered user.

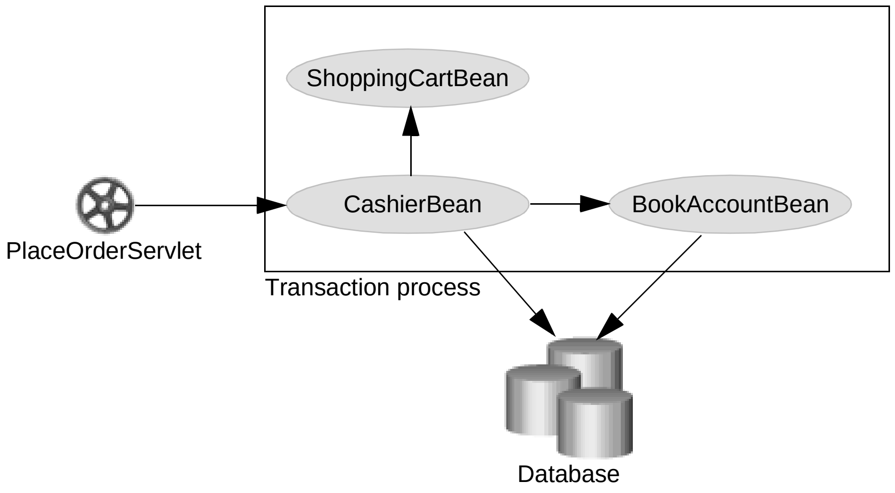
CheckOrdersServlet is invoked from the Bookstore JSP. When invoked by registered users, the servlet displays CheckOrders.jsp, which shows the user's previous orders, if any. When invoked by an administrator, the servlet displays ShowAllOrders.jsp, which shows orders from all customers.

AdminServlet, called from index.html, displays AdminLoginForm.jsp. This JSP form accepts a bookstore administrator's ID and password. AdminServlet passes this input to an LDAP server for authentication. If authentication succeeds, the servlet allows viewing of customer information on another JSP, ShowAllOrders.jsp.

LogoutServlet, invoked from the Bookstore JSP, implements the sign-out logic. LogoutServlet invalidates the customer's HTTP session and creates a new one.

EJB Functionality

The sample application contains three EJBs, and their database interactions are shown in the following figure.



ShoppingCartBean is a stateful session bean. It holds the book items chosen during the shopping session. The bean is stateful because the data it holds is unique to each client and must not be shared. CartServlet relies on ShoppingCartBean to display book information to the appropriate JSP. The CashierBean EJB calls ShoppingCartBean to obtain data needed when an order is placed.

BookAccountBean is an entity bean that controls the inventory of each book. Unlike the other beans, BookAccountBean is involved in transactions only with the database. In other words, this bean does not calculate data for display in JSPs. CashierBean passes to BookAccountBean a set of book IDs and the quantities sold for each. BookAccountBean then connects to the database and performs a global transaction, thereby reducing the book counts as needed.

CashierBean is a stateless session bean. It coordinates with ShoppingCartBean and BookAccountBean to process the customer's order. CashierBean encapsulates business logic, such as applying the necessary shipping fees and taxes. BookAccountBean then connects to the database and updates it using a global transaction.

Chapter

5

Deploying Applications

This chapter describes how to deploy applications on a Netscape Application Server machine.

The following topics are included in this chapter:

- Deploying an Application Using Deployment Manager
- Deploying Application Files Manually

Deploying an Application Using Deployment Manager

You can deploy an application using the Deployment Manager, a separate tool accessible from Netscape Application Server (NAS) Administrator or from Netscape Application Builder (NAB). When you deploy an application, the Deployment Manager installs all the application's files and registers all its components on the destination server, a NAS machine. An application must be deployed before it can be used.

Generally, a developer creates an application on a development machine using tools such as NAB, then deploys that application from the development machine to an application server using the Deployment Manager. NAS administrators, on the other hand, might use the Deployment Manager to both download an application from a NAS machine and subsequently redeploy that application to one or more applications servers.

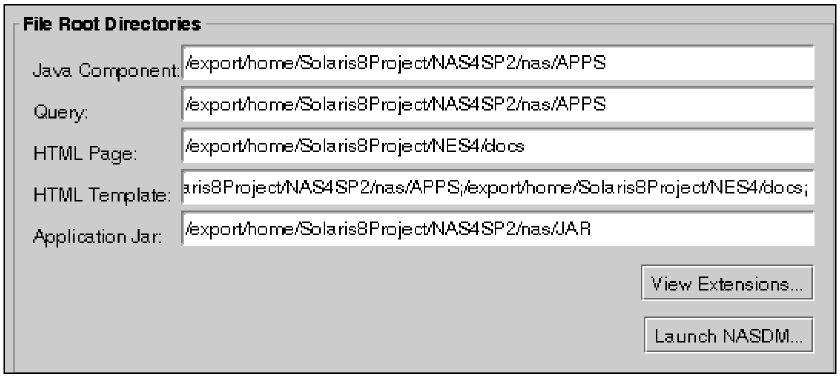
Specifying Application Directories

Before you deploy an application, you can change the application root directories that specify where the Java Server or C++ Server processes should look for application component files such as query files and template files. By referencing application root directories, you can move these components around without having to rewrite application code. If you do not specify particular root directories, application files are deployed to default directories.

The Java Server and C++ Server processes use root directories to find application components when those component are needed. For example, after a result set is returned from the database, the application most likely uses a template to format the data. The process, whether Java Server or C++ Server, scans the template root directory or directories to find the specified template file referenced by the application or query file.

To specify application root directories, perform the following steps:

1. From the NAS Administrator toolbar, click the Application button to open the Application window.
2. In the left pane of the Application window, select the NAS machine whose application root directories you want to change.
3. In the right pane of the Application window, use the text boxes to modify root directories for the specified application components as shown in the following figure:


File Root Directories

Java Component:	/export/home/Solaris8Project/NAS4SP2/nas/APPS
Query:	/export/home/Solaris8Project/NAS4SP2/nas/APPS
HTML Page:	/export/home/Solaris8Project/NES4/docs
HTML Template:	aris8Project/NAS4SP2/nas/APPS;/export/home/Solaris8Project/NES4/docs;
Application Jar:	/export/home/Solaris8Project/NAS4SP2/nas/JAR

Use a semicolon-delimited list when specifying more than one directory for an application component.

4. Click Apply Changes to save your changes to your NAS machine.

Packaging Application Files for Deployment

Before you deploy an application, you must bundle its files into a JAR file or “package.” A package contains information about the application files, such as their type and destination directory. You can select specific application files to include in the package or bundle an entire application into a package ready for deployment.

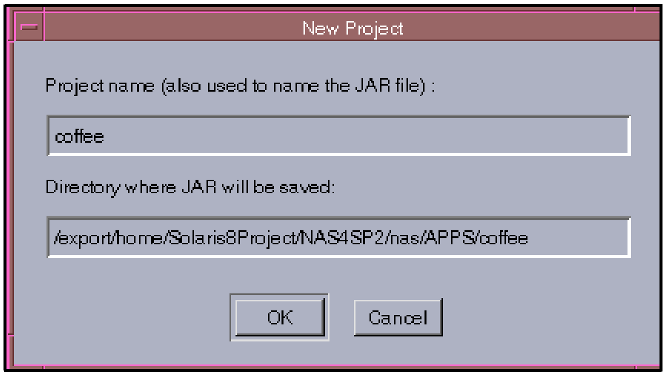
When you later deploy the package, the bundled application files are automatically distributed to their appropriate directories on one or more servers. For example, Java files and query files are automatically sent to the applications directory of your application server. At the same time, web server files, such as HTML templates, are automatically deployed to a user-defined directory on the web server. In addition, the HTML template files that reside on the application server, rather than on the web server, are deployed to their proper directory.

Creating a Package

To create a package, perform the following steps:

1. Open the Deployment Manager in one of two ways:
   - From the NAS Administrator Application window, click the Launch NASDM button.
   - From the *$NASROOT/nas/bin*, execute deployGUI
2. From the Deployment Manager's File menu, choose New Project.

   The following dialog box appears:



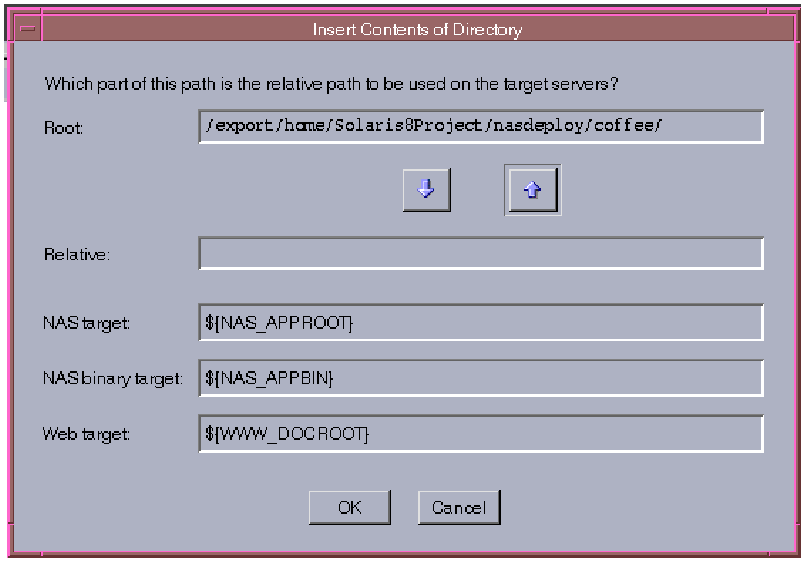
Project name (also used to name the JAR file) :
coffee
Directory where JAR will be saved:
/export/home/Solaris8Project/NAS4SP2/nas/APPS/coffee

3. Type a name for your project.

   This name is also used to identify your JAR file or package. The project name and directory name must be the same.
4. If necessary, edit the path to the directory where your JAR file is stored.
5. Click OK.
6. From the Edit menu, choose Insert.

   You can insert files one at a time, insert all files in a directory, or insert all files from the subtree of a selected directory.
7. Navigate to the directory where application files are stored and select the files to include in the package.

The following dialog box appears:



Which part of this path is the relative path to be used on the target servers?

Root: /export/home/Solaris8Project/nasdeploy/coffee/

Relative:

NAS target: ${NAS_APPROOT}

NAS binary target: ${NAS_APPBIN}

Web target: ${WWW_DOCROOT}

8. If necessary, edit the application and web file root directories for your application server using the arrow buttons.

On a Unix machine, the default root directory for application files is similar to the following:

```
/export/home/Solaris8Project/NAS4SP2/nas/APPS
```

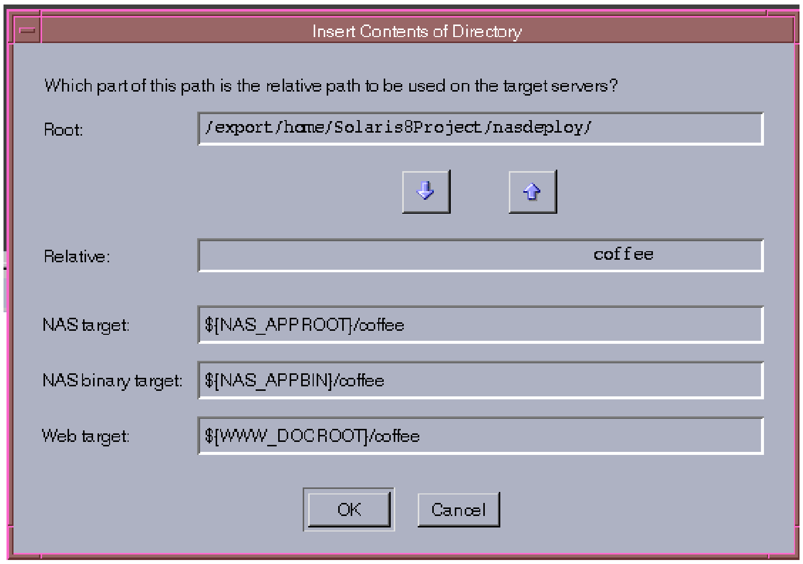
When you deploy an application using the Deployment Manager, the destination server organizes the application files relative to the root directory on your development machine. For instance, application files stored on your development machine in the location

```
/export/home/Solaris8Project/nasdeploy
```

are organized on the destination server in the following location:

destination server application root directory/myapps/

In the previous figure, pressing the down arrow button would add the `coffee` portion of the path to the target directories as shown in the following figure:



Which part of this path is the relative path to be used on the target servers?

Root: /export/home/Solaris8Project/nasdeploy/

Relative: coffee

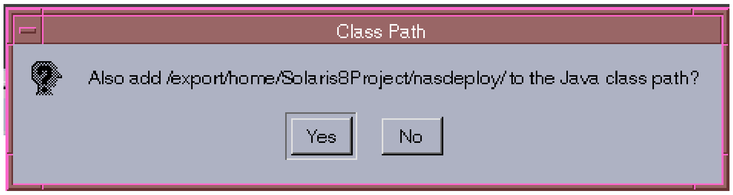
NAS target: ${NAS_APPROOT}/coffee

NAS binary target: ${NAS_APPBIN}/coffee

Web target: ${WWW_DOCROOT}/coffee

9. Click OK.

The following dialog box appears:



Also add /export/home/Solaris8Project/nasdeploy/ to the Java class path?

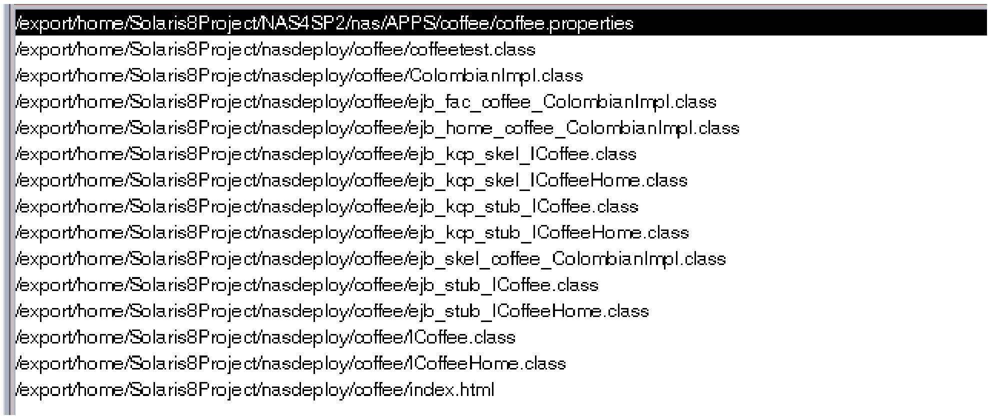
10. Click Yes if the directory contains `.class` files that will be used in the application.

If the directory does not contain `.class` files that will be used in the application, there is no need to add to the class path. If the directory does contain `.class` files and you click No, the Deployment Manager will not be able to load these `.class` files or deploy them correctly.

The Deployment Manager must have a class path in order to load `.class` files. Class files must be loaded to edit an EJB's deployment descriptor or to determine if a `.class` file is a servlet.

11. Once the application files appear in the JAR window as shown in the following figure, create meta-info files such as:

- EJB descriptors: see "Preparing an Enterprise Java Bean for Deployment" on page 75.
- `servletInfo` files: see "Editing a Servlet" on page 78.

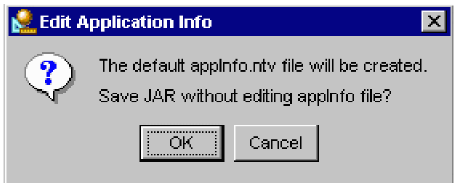


```
/export/home/Solaris8Project/NAS4SP2/nas/APPS/coffee/coffee.properties
/export/home/Solaris8Project/nasdeploy/coffee/coffeetest.class
/export/home/Solaris8Project/nasdeploy/coffee/ColombianImpl.class
/export/home/Solaris8Project/nasdeploy/coffee/ejb_fac_coffee_ColombianImpl.class
/export/home/Solaris8Project/nasdeploy/coffee/ejb_home_coffee_ColombianImpl.class
/export/home/Solaris8Project/nasdeploy/coffee/ejb_kcp_skel_ICoffee.class
/export/home/Solaris8Project/nasdeploy/coffee/ejb_kcp_skel_ICoffeeHome.class
/export/home/Solaris8Project/nasdeploy/coffee/ejb_kcp_stub_ICoffee.class
/export/home/Solaris8Project/nasdeploy/coffee/ejb_kcp_stub_ICoffeeHome.class
/export/home/Solaris8Project/nasdeploy/coffee/ejb_skel_coffee_ColombianImpl.class
/export/home/Solaris8Project/nasdeploy/coffee/ejb_stub_ICoffee.class
/export/home/Solaris8Project/nasdeploy/coffee/ejb_stub_ICoffeeHome.class
/export/home/Solaris8Project/nasdeploy/coffee/ICoffee.class
/export/home/Solaris8Project/nasdeploy/coffee/ICoffeeHome.class
/export/home/Solaris8Project/nasdeploy/coffee/index.html
```

12. From the File menu, choose Save JAR when all files you want to include in the package appear in the list.

If the list of files in your project shows files marked in red with an asterisk, these files require additional action. Skip to step 13.

If your project files require no further action, the following dialog box appears:



The default appInfo.ntv file will be created.
Save JAR without editing appInfo file?

A Name Type Value (NTV) file called `appinfo.ntv` is created by default if such a file doesn't already exist. This file contains the name of the JAR file, associated session information, and a list of servlets included in the JAR file.

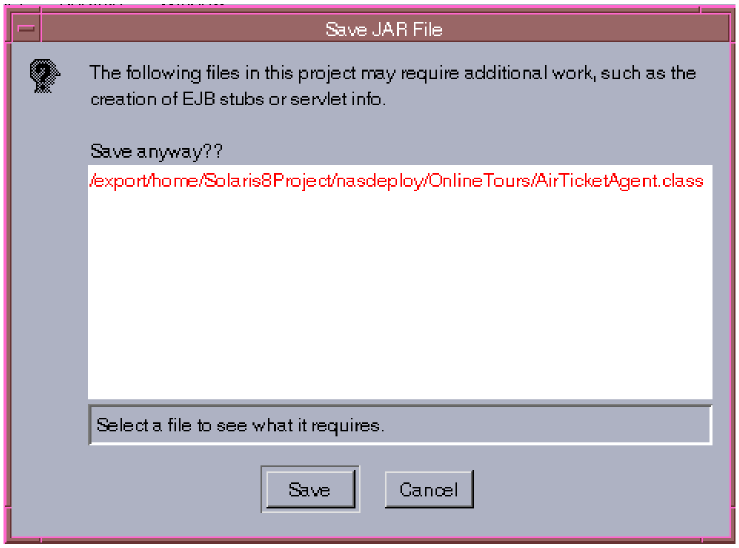
13. Choose one of the following options:

- Click Cancel to open the appInfo editor.

If you choose to edit the appInfo file, see "Editing a Servlet" on page 78 for more information about this editor.

- Click OK to save the JAR file without editing the appInfo file.

14. If your project files require additional action (indicated by red type and an asterisk next to the files in the JAR window), the following dialog box appears:



Save JAR File

The following files in this project may require additional work, such as the creation of EJB stubs or servlet info.

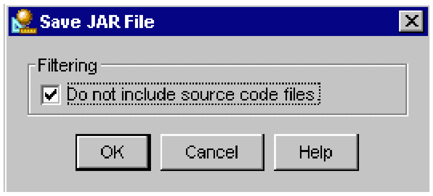
Save anyway??

/export/home/Solaris8Project/nasdeploy/OnlineTours/AirTicketAgent.class

Select a file to see what it requires.

You can click a file to see what action is required. For instance, an EJB may be missing stubs. Click Cancel to dismiss the dialog box and correct these problems or click Save to save the JAR as-is.

15. When saving a JAR file, you can filter `.java` files out of the package by selecting the corresponding checkbox in the Save JAR File dialog box as shown in the following figure:



Do not include source code files [✓]

16. Click OK.

    If you want to save an "in-progress" list of files without completing the package, you can save the list of files as a `.dxm` file by choosing Save List from the File menu. You can later open the file and modify it.

    The package is ready for deployment.

Preparing an Enterprise Java Bean for Deployment

You can prepare an Enterprise Java Bean (EJB) to include in your JAR file using the packaging tool. To prepare an EJB, perform the following steps:

1. From the list of files in the JAR window, select the following three types of classes:

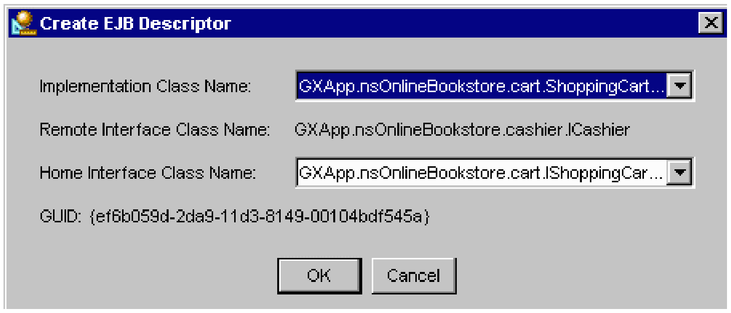
    - implementation class
    - remote interface class
    - home interface class

    These three classes are required to create an EJB. You can add other classes in addition to these three.

2. From the EJB menu, choose Create Descriptor.

    A dialog appears showing which classes are used in which roles.

    If you selected only one `.class` file, that file appears next to the appropriate class type (Remote Interface, for example). You can then select the remaining two required files from drop-down boxes that appear next to the appropriate class type as shown in the following figure:

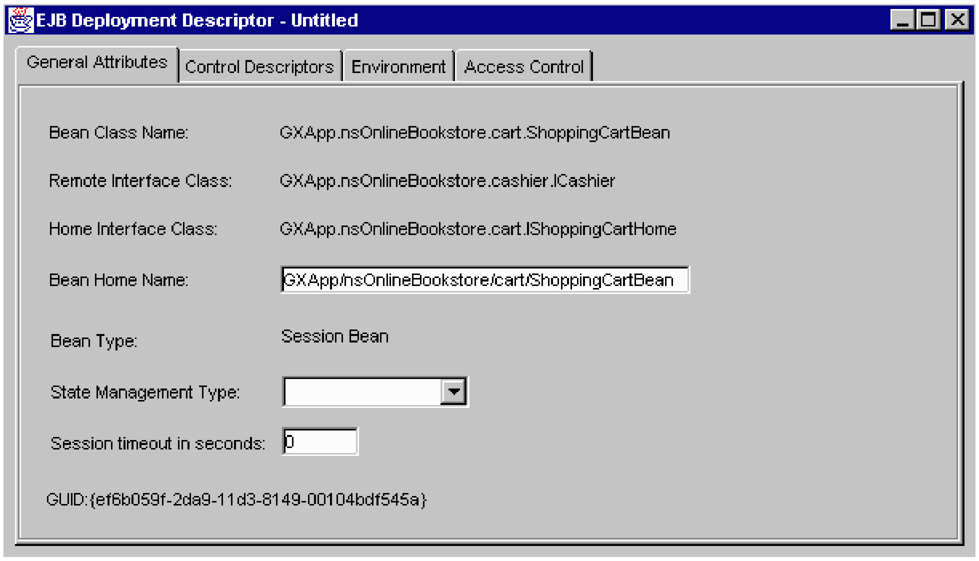


Implementation Class Name: GXApp.nsOnlineBookstore.cart.ShoppingCart...
Remote Interface Class Name: GXApp.nsOnlineBookstore.cashier.ICashier
Home Interface Class Name: GXApp.nsOnlineBookstore.cart.IShoppingCar...
GUID: {ef6b059d-2da9-11d3-8149-00104bdf545a}

The drop-down boxes list only those `.class` files that satisfy the requirements of the EJB.

3. Click OK when each field is satisfied.

   The EJB deployment descriptor editor appears as a separate tool in the Deployment Manager window:



Bean Class Name: GXApp.nsOnlineBookstore.cart.ShoppingCartBean
Remote Interface Class: GXApp.nsOnlineBookstore.cashier.ICashier
Home Interface Class: GXApp.nsOnlineBookstore.cart.IShoppingCartHome
Bean Home Name: GXApp/nsOnlineBookstore/cart/ShoppingCartBean
Bean Type: Session Bean
State Management Type:
Session timeout in seconds: 0
GUID:{ef6b059f-2da9-11d3-8149-00104bdf545a}

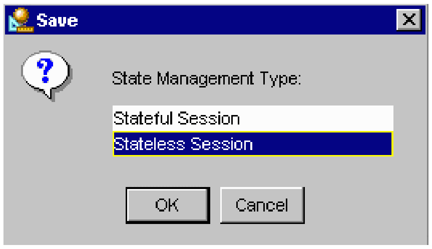
4. Edit EJB deployment information if necessary using this editor.

   See “Editing an EJB’s Deployment Descriptor” on page 79.

5. From the Deployment Manager’s EJB menu, choose Save Descriptor.

The file is saved as a `.properties` file.

If your EJB is a session bean rather than an entity bean, and you have neglected to specify the state management type in the bean's deployment descriptor, the following dialog box appears:

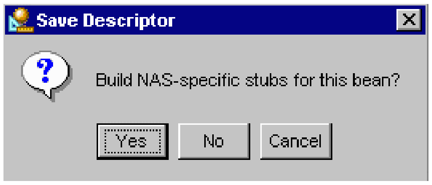


6. Select the type of your EJB: Stateful Session or Stateless Session.

   A stateless session bean is completely transient and encapsulates a temporary piece of business logic needed by a specific client for a limited time span.

   A stateful session bean is also transient, and uses a "conversational state" to preserve information about its contents and values between client calls. The conversational state enables the container to maintain information about the state of the session bean and to recreate that state at a later point in program execution when needed.

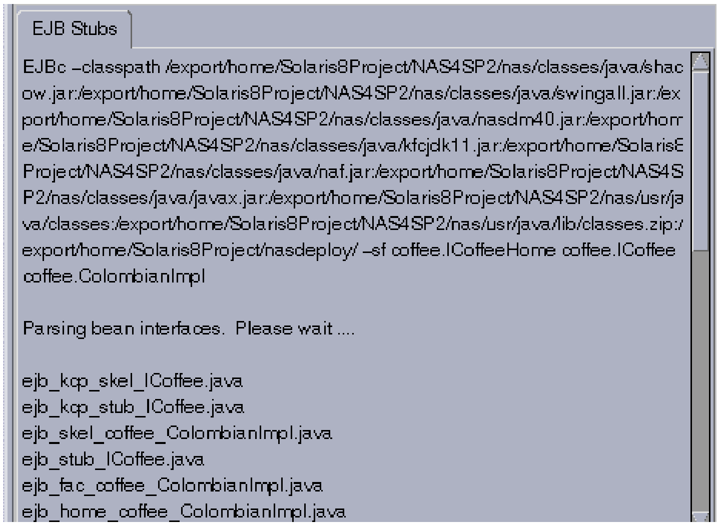
   The following dialog box appears:


Build NAS-specific stubs for this bean?

7. Decide whether to build stubs for the EJB or not.

   Stubs and skeletons are required by the EJB container and must be deployed with the application files. These stubs and skeletons enable remote communication and allow the container to intercept all bean requests. When you create stubs and skeletons, the Deployment Manager automatically adds them to the list of application files.

After the Deployment Manager builds the stubs, the following status window appears (this may take a moment or two):



```
EJBc –classpath /export/home/Solaris8Project/NAS4SP2/nas/classes/java/shac
ow.jar:/export/home/Solaris8Project/NAS4SP2/nas/classes/java/swingall.jar:/ex
port/home/Solaris8Project/NAS4SP2/nas/classes/java/nasdm40.jar:/export/hom
e/Solaris8Project/NAS4SP2/nas/classes/java/kfcjdk11.jar:/export/home/Solaris8
Project/NAS4SP2/nas/classes/java/naf.jar:/export/home/Solaris8Project/NAS4S
P2/nas/classes/java/javax.jar:/export/home/Solaris8Project/NAS4SP2/nas/usr/ja
va/classes:/export/home/Solaris8Project/NAS4SP2/nas/usr/java/lib/classes.zip:/
export/home/Solaris8Project/nasdeploy/ –sf coffee.ICoffeeHome coffee.ICoffee
coffee.ColombianImpl

Parsing bean interfaces.  Please wait ....

ejb_kcp_skel_ICoffee.java
ejb_kcp_stub_ICoffee.java
ejb_skel_coffee_ColombianImpl.java
ejb_stub_ICoffee.java
ejb_fac_coffee_ColombianImpl.java
ejb_home_coffee_ColombianImpl.java
```

If you choose to build stubs later, you can select Build Stubs from the EJB menu while the EJB deployment descriptor editor is open and displaying that EJB. You can choose Build All Stubs at anytime to build stubs for all EJBs whose `.properties` files are in the list of files appearing in the JAR window.

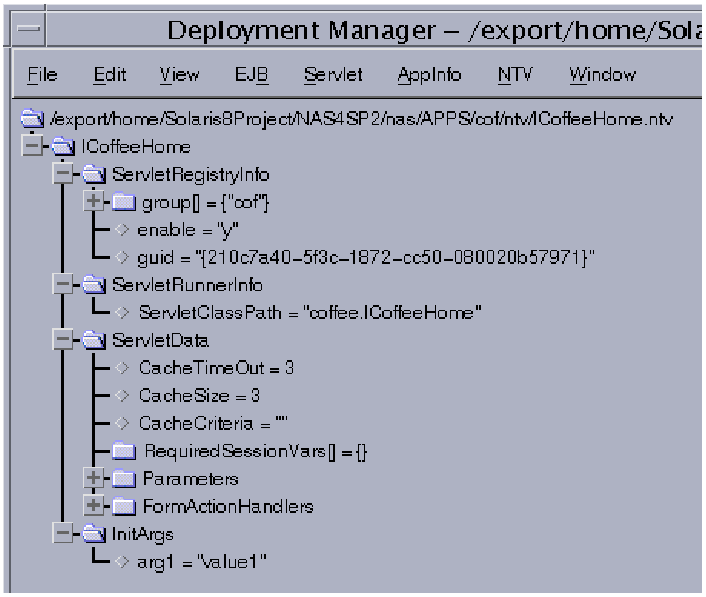
Editing a Servlet

You can edit any class file in a package as a servlet. The file is then marked as a servlet and an `.ntv` file is created for it by default.

To edit a servlet, perform the following steps:

1. From the Servlet menu, choose Edit Info.

   The NTV editor appears.



```
/export/home/Solaris8Project/NAS4SP2/nas/APPS/cof/ntv/ICoffeeHome.ntv
ICoffeeHome
  ServletRegistryInfo
    group[] = {"cof"}
    enable = "y"
    guid = "{210c7a40-5f3c-1872-cc50-080020b57971}"
  ServletRunnerInfo
    ServletClassPath = "coffee.ICoffeeHome"
  ServletData
    CacheTimeOut = 3
    CacheSize = 3
    CacheCriteria = ""
    RequiredSessionVars[] = {}
    Parameters
    FormActionHandlers
  InitArgs
    arg1 = "value1"
```

2. Right-click a branch in the tree to edit that branch or remove it.

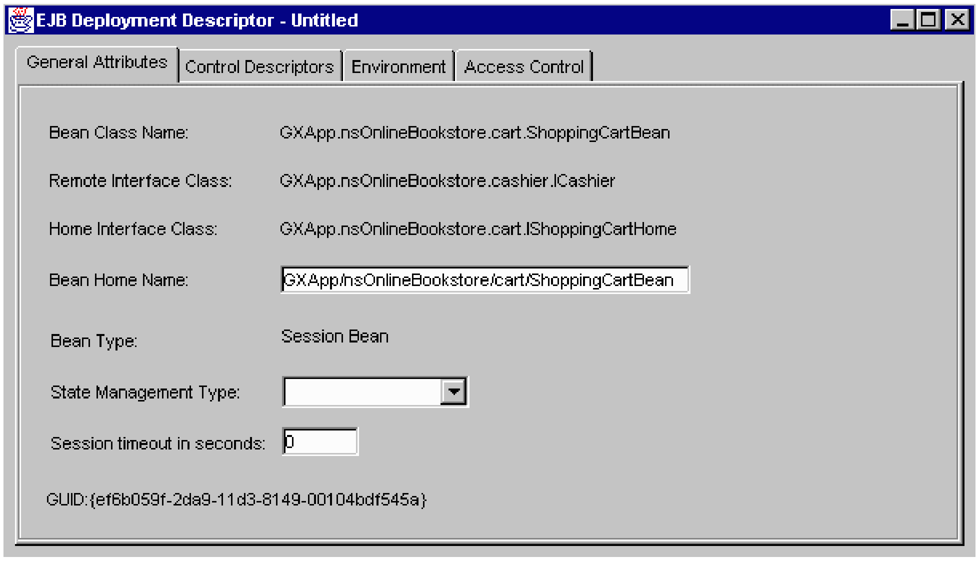
See “Creating a Package” on page 70 for more information about `.ntv` files.

Editing an EJB’s Deployment Descriptor

Deployment descriptors include the declarative attributes associated with an EJB (or bean). These attributes tell an EJB’s container how to manage the bean. A container is where an EJB “lives” from its creation to its destruction. The container manages the EJB’s life cycle and support services while providing services that allow clients to look up the interfaces of installed EJB classes.

A developer might edit an EJB’s deployment descriptor as part of a cycle of developing and testing an application. The editor allows either administrators or developers to easily modify the attributes of EJBs to better work within an application.

You can edit an EJB’s deployment descriptor using the editor shown in the following figure:



Bean Class Name: GXApp.nsOnlineBookstore.cart.ShoppingCartBean

Remote Interface Class: GXApp.nsOnlineBookstore.cashier.ICashier

Home Interface Class: GXApp.nsOnlineBookstore.cart.IShoppingCartHome

Bean Home Name: GXApp/nsOnlineBookstore/cart/ShoppingCartBean

Bean Type: Session Bean

State Management Type:

Session timeout in seconds: 0

GUID:{ef6b059f-2da9-11d3-8149-00104bdf545a}

Editing General Attributes

To edit the General Attributes of an EJB, perform the following steps:

1. If the editor is not already open, you can open it by right-clicking on a `.properties` file displayed in the JAR window and choosing Edit Descriptor.

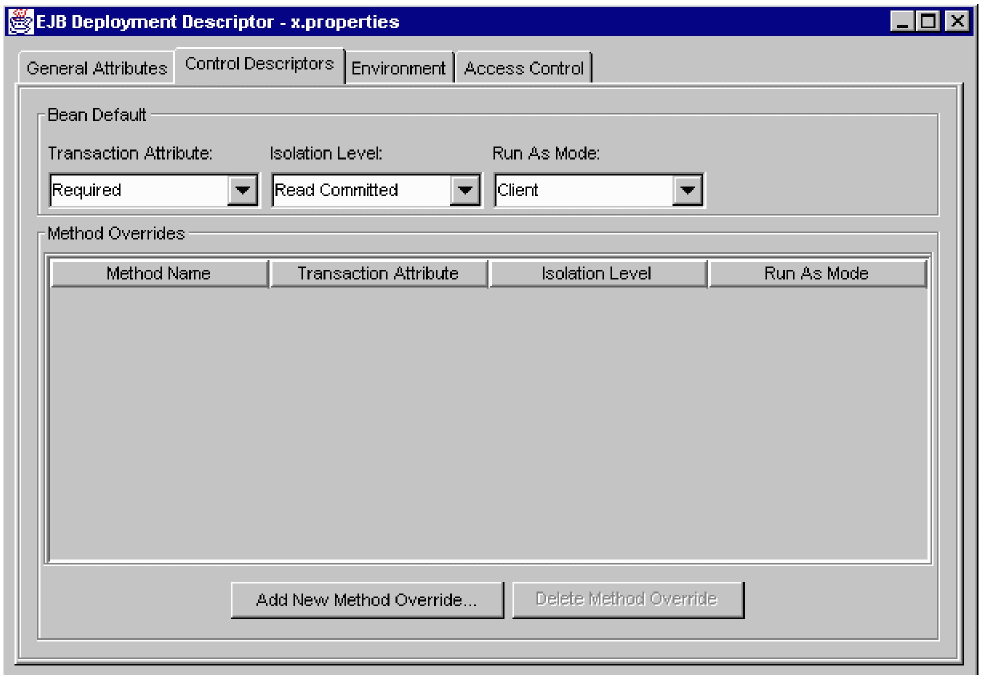
   You can edit the Bean Home Name as necessary, change the State Management Type of the bean, or edit the bean's session timeout value.

   See the *Programmer's Guide* for more information about these values.

2. From the File menu, choose Save Descriptor to save your changes.

Editing Control Descriptors

The Control Descriptor tab allows you to configure meta-data for an EJB at deployment time. You can edit transaction isolation levels, transactional attributes, and the mode entry for the bean.



Transaction Attribute: Required
Isolation Level: Read Committed
Run As Mode: Client

To edit the Control Descriptors for an EJB, perform the following steps:

1. Click the Control Descriptors tab to display the window as shown in the previous figure.

The EJB's meta-data appears in the Bean Default area of the window. You can edit this information for the bean as a whole or choose a particular method within the bean and change that method's meta-data in the Method Overrides area of the window.

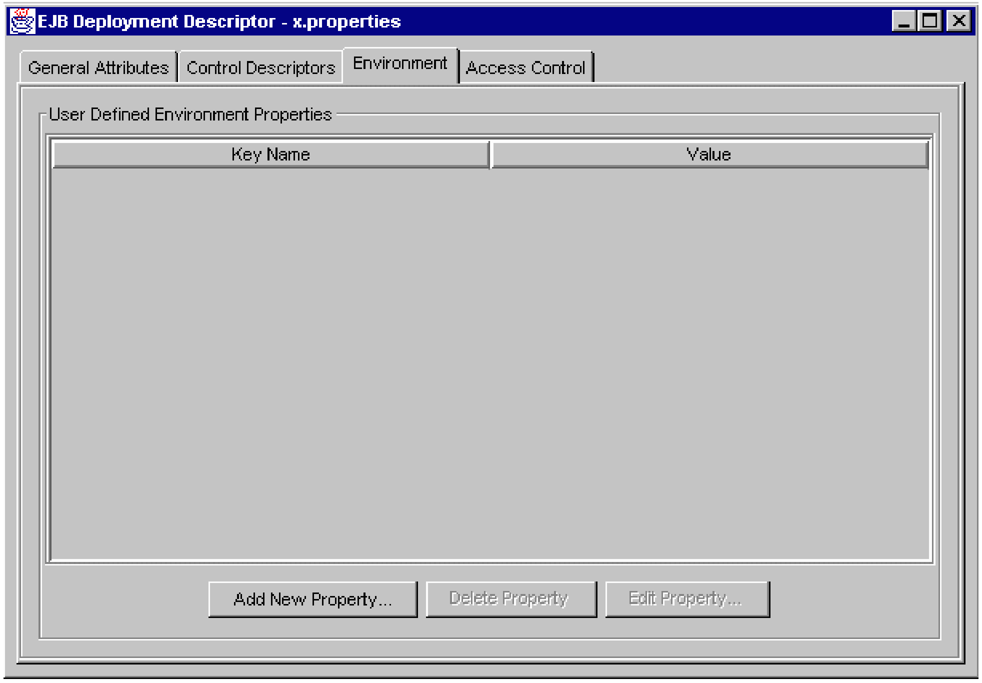
The Run As Mode entry defines the identity a bean uses during execution and how a bean identifies its user to other beans or resources. Most commonly, this value is Client, which means that a bean or its methods are executed using the client's identity. See the *Programmer's Guide* for more information.

Transaction Attributes specify when a bean needs to begin a transaction. See he *Programmer's Guide* for more information.

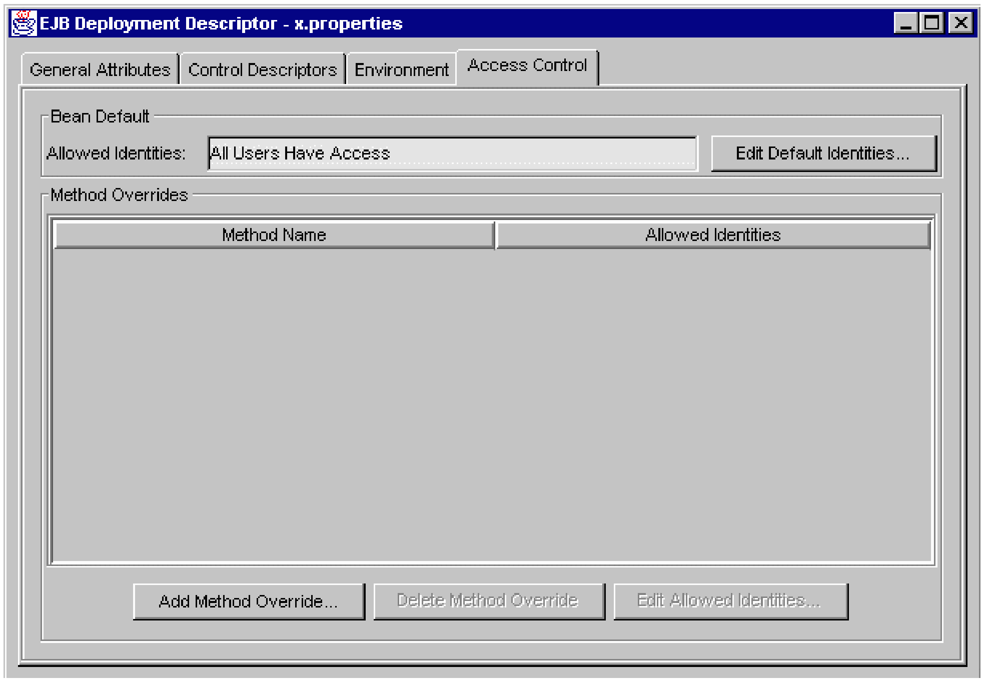
An isolation level specifies how much or how little a transaction can see of other, simultaneous interactions with a database. For more information, see the *Programmer's Guide*.

Editing Environment Properties

.



Editing Access Control



Deploying an Application

After you have created a package using the packaging tool, you use the Deployment Manager to send the package to a NAS machine.

If there is more than one JAR file in a single directory, you can deploy multiple JAR files at the same time from that directory.

To deploy a JAR using the Deployment Manager, perform the following steps:

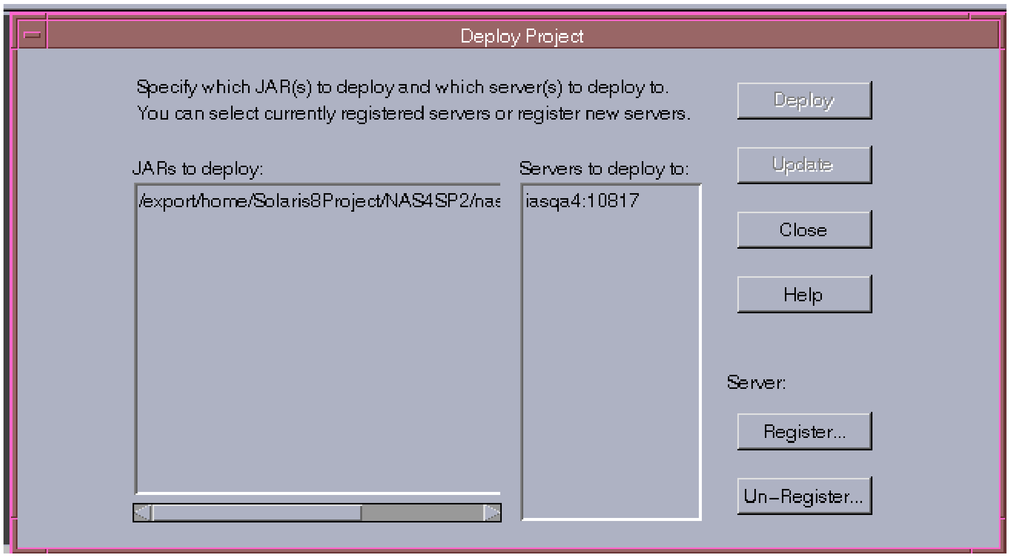
1. From the File menu, choose Deploy.

   You can deploy the currently open package, or you can choose a directory and deploy one or more packages stored therein.

You are prompted to save the JAR file as well as the `appInfo` file if either has changed since you last saved it.

2. From the list of registered servers, choose a NAS machine to deploy to.

   To register additional servers, click Register and enter the necessary server information.



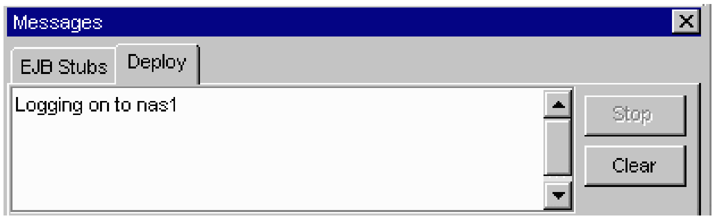
Specify which JAR(s) to deploy and which server(s) to deploy to.
You can select currently registered servers or register new servers.

JARs to deploy: /export/home/Solaris8Project/NAS4SP2/nas

Servers to deploy to: iasqa4:10817

3. Click Deploy.

   The status of the deployment process appears in a separate window.



```
Logging on to nas1
```

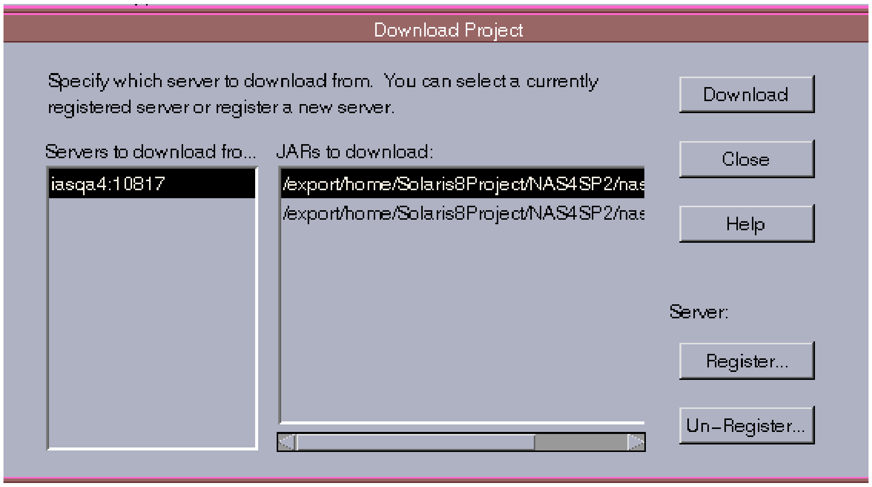
Downloading a Package

Once an application has been installed on an application server, you can download that application to your machine using the Deployment Manager. When you download an application, you save that application's JAR file to your local machine or another machine on your network.

To download an application, perform the following steps:

1. From the File menu of the Deployment Manager, choose Download.

   The following dialog box appears:



Download Project

Specify which server to download from. You can select a currently registered server or register a new server.

Servers to download fro...

iasqa4:10817

JARs to download:

/export/home/Solaris8Project/NAS4SP2/nas
/export/home/Solaris8Project/NAS4SP2/nas

2. Select a server from which to download an application.

   You can register additional servers by clicking the Register button. See "Deploying an Application" on page 83 for details.

3. From the list of JAR files found on the selected server, select a JAR file to download.

4. Click Download.

5. Choose the directory where you will save the JAR file on your local machine or the network.

   The message window displays the status of the file you are downloading.

Deploying Application Files Manually

The Deployment Manager does not allow you to deploy individual application components that are not part of an application. Instead, you can deploy individual application files manual.

This section contains separate procedures for deploying Enterprise Java Beans, servlets and JavaServer Pages (JSPs), and data sources:

- Manually Deploying EJBs
- Manually Deploying Servlets and JSPs
- Manually Deploying Data Sources

Manually Deploying EJBs

To deploy an EJB manually, perform the following steps:

1. Compile the EJB.
2. Generate stubs and skeletons.
3. Create a deployment descriptor.
4. Copy files to NAS.
5. Register the EJB.

Compile the EJB

Using `javac`, compile all EJB files including the bean, home interface, and remote interface.

The following example shows what you might type at the command line on Solaris operating systems:

```
% $GX_ROOTDIR/usr/java/bin/javac ShoppingCartBean.java
IShoppingCart.java IShoppingCartHome.java
```

In this examples, `ShoppingCartBean.java` is the bean implementation file, `IShoppingCart.java` is the remote interface, and `IShoppingCartHome.java` is the home interface.

Generate Stubs and Skeletons

Stubs and skeletons are required by the EJB container and must be deployed with the application files. These stubs and skeletons enable remote communication and allow the container to intercept all bean requests.

Using `ejbc`, generate stubs and skeletons for your EJB following these example:

```
$GX_ROOTDIR/bin/ejbc -sf cart.ShoppingCartBean cart.IShoppingCart
cart.IShoppingCartHome
```

`-sf` is used for stateful session beans.

Create a Deployment Descriptor

Deployment descriptors include the declarative attributes associated with an EJB (or bean). These attributes tell an EJB's container how to manage the bean. A container is where an EJB "lives" from its creation to its destruction. The container manages the EJB's life cycle and support services while providing services that allow clients to look up the interfaces of installed EJB classes.

Copy the Files to NAS

Copy the class files from your development machine to the destination server's `APPS` directory. You must preserve the directory structure used on the development machine.

Register the EJB

Once you have copied the files onto the destination machine, you must register the EJB using `beanreg` on the NAS host machine. BeanReg registers a bean locally using the `.properties` file that describes the bean. Registering your bean requires the deployment descriptor file you created in "Create a Deployment Descriptor."

For example, on Solaris machines, type the following at the command line;

```
% $GX_ROOTDIR/bin/beanreg filename.properties
```

Manually Deploying Servlets and JSPs

To deploy a servlet or JSP manually, perform the following steps:

1. Create servlet configuration files.
2. Copy files to NAS.
3. Register the servlet.

Create Servlet Configuration Files

You must create a configuration file for each servlet, and one for the application as a whole. These configuration files must be referenced in an application-wide configuration file called `appInfo.ntv`. Servlets that are not part of a specific application are part of the "generic" application, which also must have a configuration file `appInfo.ntv`.

Servlet configuration files can be named anything as long as the name contains the `.ntv` suffix and the filename is referenced in `appInfo.ntv`. They must be placed in a certain directory, which is specified in the next section, "Copy the Files to NAS."

The following is an example of an `appInfo.ntv` file:

```
NTV-ASCII
{
   “SessionInfo”     NTV     {
     “timeout”     Int     “400”,
     “flags”          StrArr     [“SESSION DISTRIB”],
     “sessionManagement”     Int     “0”
   },
    “ServletFiles”     StrArr     [“servInfo”],
    “AppName”     Str     “nsOnlineBookstore”,
}
```

`AppName` refers to the name of the application, which, in this case, is nsOnlineBookstore. `servInfo` refers to the servletInfo file, `servInfo.ntv`. You can find this file in the following location:

`NAS install directory/``APPS/NSOnlineBookstore/ntv`

Copy the Files to NAS

Servlets and JSPs can be part of an application or exist outside of an application, depending upon how the web client invokes the servlet.

To copy these files to NAS, first follow the directions that apply to your application component:

- Copy Servlets as Part of an Application
- Copy JSPs as Part of an Application
- Copy Servlets Not Part of an Application
- Copy JSPs Not Part of an Application

Next, copy the class files into the NAS class path, which is usually `$GX_ROOTDIR/APPS`. You must preserve the directory structure when copying the class files. See the examples in “Sample Directory Structures for Servlets and JSPs Not Part of an Application” on page 91 and “Sample Directory Structures for Servlets and JSPs as Part of an Application” on page 91 for details.

Finally, copy static content like HTML and GIF files to the web server documentation root.

Copy Servlets as Part of an Application

For a servlet that’s part of an application, the following URL would appear in the web client:

`http://`*`$HOSTNAME:$PORT/NASApp/$AppName/$ServletName`*

In this URL, *`$HOSTNAME`* is the DNS name of the host machine, *`$PORT`* is the TCP/IP port, NASApp is a key defined the registry to signal to NAS that the URL references a servlet, *`$AppName`* is the AppName variable in the `appInfo.ntv` files, and *`ServletName`* is the name of the servlet.

To mirror this URL on your own machine, copy the application's NTV files to the `$GX_ROOTDIR/APPS/$`*`AppName/`*`ntv` directory.

Note The `appInfo.ntv` file must point to all relevant `servletInfo.ntv` files. If you change the names of the `servletInfo.ntv` files, you must reflect those changes in the `appInfo.ntv` file.

Copy JSPs as Part of an Application

For a JSP that's part of an application, copy the JSP into the `$GX_ROOTDIR/ APPS/`*`$AppName`* directory, then refer to them using *`$AppName/ jspnam`*`e.jsp`.

Copy Servlets Not Part of an Application

For a servlet that's not part of an application, the following URL would appear in the web client:

`http:`*`//$HOSTNAME:$PORT/`*`servlets/`*`$ServletName`*

In this URL, *`$HOSTNAME`* is the DNS name of the host machine, *`$PORT`* is the TCP/IP port, `servlets` is a non-changable string, and *`ServletName`* is the name of the servlet.

To mirror this URL on your own machine, copy the `servletInfo.ntv` files to the `$GX_ROOTDIR/APPS/ntv` directory. You must update the `appInfo.ntv` file with the new `servletInfo.ntv` files.

Copy JSPs Not Part of an Application

For a JSP that's not part of an application, copy the JSP into the `$GX_ROOTDIR/APPS/` directory, then refer to them using *`jspnam`*`e.jsp`.

Sample Directory Structures for Servlets and JSPs Not Part of an Application

For servlets and JSPs not in an application, use the following directory structures as examples:

NAS install directory (`APPS` is in the NAS class path):

`/disk2/ns-home/nas/APPS/`

Servlet with class name `javaSpec`:

`/disk2/ns-home/nas/APPS/javaSpec.jsp`

JSP referenced by `javaSpec.jsp`:

`/disk2/ns-home/nas/APPS/javaSpec.jsp`

Default NTV directory for all servlets and JSPs not in any application:

`/disk2/ns-home/nas/APPS/ntv`

Default `appInfo.ntv` file referencing all servlets not in an application:

`/disk2/ns-home/nas/APPS/ntv/appInfo.ntv`

Sample Directory Structures for Servlets and JSPs as Part of an Application

For servlets and JSPs in an application, use the following directory structures as examples. Here, the application name is Project1.

NAS install directory (`APPS` is in the NAS class path):

`/disk2/ns-home/nas/APPS/Project1/`

Location of all NTV files for Project1:

`/disk2/ns-home/nas/APPS/Project1/ntv/`

Location of `appInfo.ntv` for Project1:

`/disk2/ns-home/nas/APPS/Project1/ntv/appInfo.ntv`

`servletInfo.ntv` file for `Project1.webapp.NASservlet` servlet which is referenced in the `appInfo.ntv` file for this project:

`/disk2/ns-home/nas/APPS/Project1/ntv/NASservlet.ntv`

`NASservlet` servlet with a full class name of `Project1.webapp.NASservlet:`

```
/disk2/ns-home/nas/APPS/Project1/webapp/NASservlet.class
```

`NASservlet` JSP file referencable with the name `Project1/webapp/NASservlet.jsp`:

```
/disk2/ns-home/nas/APPS/Poject1/webapp/NASservlet.jsp
```

URL to access `NASservlet`:

```
http://warp/NASApps/Project1/NASservlet
```

Register the Servlet

On the NAS host machine, run one of the following scripts on the `appInfo.ntv` file to register all servlets in the `appInfo.ntv` file:

On Solaris: `servletReg.sh`

For example, on Solaris, you might type the following at the command line:

```
%$GX_ROOTDIR/bin/servletReg.sh -i appInfo.ntv
```

Manually Deploying Data Sources

To deploy a data source manually, perform the following steps:

1. Create the data source file.
2. Register the data source.

Create the Data Source File

The data source file contains information necessary for accessing the database such as a user name and password. Using a text editor, create a data source file (`*filename*.props`) using the following example as a guide:

```
DataBase=ksample
DataSource=ksample
UserName=kdemo
PassWord=kdemo
ResourceMgr=rm_orcl       #Optional
```

Data source files are not referenced at run time, so you need not place them in a specific directory on your NAS machine. However, it is good convention to place them in the `APPS/`*`$AppName`* directory.

For more information, see the *Programmer's Guide*.

Register the Data Source

After you have created the data source file, you must run `dsreg` on the NAS host machine.

For example, on Solaris machines, you might type the following at the command prompt:

```
$GX_ROOTDIR/bin/dsreg “jdbc/BookstoreDS” BookstoreDS.props
```

where `jdbc/BookstoreDS` is the data source name.

Index