Oracle® Internet Application Server 8*i*

Overview Guide

June, 2000

Part No. A83707-01

ORACLE®

Oracle® Internet Application Server 8*i* Overview Guide

Part No. A83707-01

Primary Authors: Janice Nygard, Laurel Hale

Contributing Authors: Sanjay Singh, Priya Darshane, Francisco Abedrabbo, Jeremy Lizt, Matthieu Devin, Sheryl Maring

Contributors: Mike Boros, Juliana Button, Steve Button, Florence Chatzigianis, Rakesh Dhoopar, Moe Fardoost, Marie Goodell, Helen Grembowicz, Pat Hinkley, Scott Howley, Pushkar Kapasi, Nick Kritikos, Kai Li, Karen Masterson, Andrew Mendelsohn, Eduardo Mendez, Paul Narth, Gilbert Pascal, Beth Roeser, John Russell, David M. J. Saslav, Mark Scardina, Ingrid Snedecor, Rao Vasireddy, Yaqing Wang, Brian Wright

Contents

Preface xi

1 Introduction

How Oracle Internet Application Server Supports Your Evolution into an E-Business 1-1
The Oracle Solution: Oracle Internet Platform 1-2
Oracle Internet Application Server Overview 1-3
Publishing Content 1-3
Running Transactional Applications 1-4
Oracle Internet Application Server Architecture 1-6
Two and Three-Tier Computing Models 1-6
Two-Tier Computing Model 1-6
Three-Tier Computing Model 1-7
Oracle Internet Application Server 8i 1-8
Available Versions 1-9
Supported Technologies and Programming Languages 1-10

2 Oracle Internet Application Server Services

Communication Services 2-2
Oracle HTTP Server *powered by Apache* 2-3
Oracle HTTP Server Modules (mods) 2-4
mod_ssl 2-4
mod_plsql 2-4
mod_perl 2-4
mod_jserv 2-4

Presentation Services 2-5
Apache JServ 2-6
OracleJSP (JavaServer pages) 2-6
Oracle PL/SQL Server Pages (PSP) 2-7
Perl Interpreter 2-8
Business Logic Services 2-9
Oracle BC4J (Business Components for Java) 2-10
Oracle8*i* JVM 2-10
Oracle8*i* PLSQL 2-11
Oracle Forms Services 2-11
Oracle Reports Services 2-12
Data Management Services 2-13
Oracle8*i* Cache 2-14
Who Should Use Oracle8*i* Cache? 2-14
Performance and Scalability Benefits 2-14
Developer's Kits 2-15
Oracle Database Client Developer's Kit 2-16
Oracle Java Messaging Service (JMS) Toolkit 2-16
Oracle SQLJ Translator 2-17
Oracle Java Database Connectivity (JDBC) Drivers 2-19
Oracle XML Developer's Kit 2-20
Oracle LDAP Developer's Kit 2-21
System Services 2-23
Oracle Enterprise Manager 2-24
Oracle Advanced Security 2-27
Network Security 2-27
Enterprise User Security 2-28

3 Developing Applications for Oracle Internet Application Server

Content Publishing 3-2
Static Content 3-2
Dynamic Content 3-2
Common Gateway Interface (CGI) Applications 3-3
Perl Scripts 3-3
XML 3-4

Servlets 3-5
JavaServer Pages 3-7
PL/SQL Server Pages 3-9
Oracle Reports Services 3-12
Business Logic 3-16
Java Servlets, Applications, and Enterprise JavaBeans 3-16
Oracle Forms Services 3-20

Index

Send Us Your Comments

Oracle® Internet Application Server 8*i* Overview Guide

Part No. A83707-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments to us in the following ways:

- E-mail - **iasdocs@us.oracle.com**
- FAX - 650-654-6206 Attn: Oracle Internet Application Server Documentation
- Postal service:
  Oracle Corporation
  500 Oracle Parkway, M/S 6op4
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, and telephone number below.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Audience

This book is an overview of Oracle Internet Application Server for a general audience.

Assumptions

This book assumes that readers have a general knowledge of application servers, Web servers, and database-driven Web sites. It also assumes that readers are familiar with the technologies and programming languages used in middle-tier environments.

How This Guide Is Organized

This book contains the following chapters:

Preface	Provides you with an introduction to this book and gives you information about typographical conventions and lists sources for additional services and support on this product.
Chapter 1	Provides an introduction to Oracle Internet Application Server, explaining how it supports your e-business. This chapter includes an architectural overview, a description of how you can use this product, and a list of the major technologies and programming languages it supports.
Chapter 2	Gives an overview of each service contained in Oracle Internet Application Server and lists additional information sources for each service.

Chapter 3	Demonstrates how you can use Oracle Internet Application Server to build your applications.

Related Documentation

For more information, see the following documentation in the Release 8*i* Oracle Internet Application Server documentation set:

The following documentation is included on your Documentation Library CD-ROM:

- *Oracle Internet Application Server Quick Tour*
- *Migrating from Oracle Application Server*

The following documentation is included on your product CD-ROM. (The *Oracle Internet Application Server Installation Guide* is also included in printed form with your product.)

- *Oracle Internet Application Server 8i Release Notes*
- *Oracle Internet Application Server Installation Guide*

Other Information Sources

The following sections list resources you can use to increase your knowledge and understanding of Oracle Internet Application Server and how to use it.

Online Documentation

The following documentation for the Oracle services contained in Oracle Internet Application Server are also included on your Documentation Library CD-ROM. Many documents are provided in Adobe Portable Document Format (PDF) and many are also provided in HTML.

Oracle HTTP Server *powered by Apache* Documentation

- *Apache 1.3.9 User's Guide*
- *Apache JServ Documentation*
- *Apache mod_perl Documentation*
- *Using mod_plsql*

Oracle Forms Services Documentation

- *Forms Developer Quick Tour*
- *Deploying Forms Applications to the Web with Internet Application Server*
- *Form Builder Reference Manual*

Oracle Reports Services Documentation

- *Reports Developer Quick Tour*
- *Publishing Reports to the Web with Internet Application Server*
- *Building Reports*
- *Reports Reference Manual*

Common Documentation for Forms Services and Reports Services

- *Guidelines for Building Applications*
- *Graphics Builder Reference Manual*
- *Procedure Builder Reference Manual*
- *Common Built-In Packages Reference Manual*

Oracle8*i* Cache Documentation

- *Quick Tour*
- *Concepts and Administration Guide*

Oracle8*i* JVM Documentation

- *Oracle8i Enterprise JavaBeans and CORBA Developer's Guide*
- *Oracle8i Java Developer's Guide*
- *Oracle8i Java Stored Procedures Developer's Guide*
- *Oracle8i JDBC Developer's Guide and Reference*
- *Oracle8i JPublisher User's Guide*
- *Oracle8i SQLJ Developer's Guide and Reference*

Oracle Enterprise Manager Documentation

- *Quick Tour*
- *Concepts Guide*

- *Configuration Guide*
- *Administrator's Guide*
- *Intelligent Agent User's Guide*
- *Messages Manual*
- *SNMP Support Reference Guide*

Oracle JSP (JavaServer Pages) Documentation

- *Running Oracle JSP on Apache*
- *Developer's Guide*
- *Database Access from JavaServer Pages*
- *JML (Oracle JSP Markup Language) Syntax Card*
- *Developer's Toolkit*

Oracle BC4J (Business Components for Java) Documentation

- *Developing Business Components*
- *Reference API*

Oracle LDAP Developer's Kit Documentation

- *Application Developer's Guide*

Oracle XML Developer's Kit Documentation

- *XML Parser for Java API Reference*
- *XML Transviewer Beans API Reference*
- *XML Class Generator for Java API Reference*

You can order any product documentation not included on your Documentation Library CD-ROM from Oracle Store at:

- **http://store.oracle.com**

For more information about resources for product support and documentation, see "Oracle Services and Support" on page -xv.

Conventions

This manual uses the following typographical conventions:

Convention	Example	Explanation
bold	**tnsnames.ora** **runInstaller** **www.oracle.com**	Identifies file names, utilities, processes, and URLs
italics	*file1*	Identifies a variable in text; replace this place holder with a specific value or string.
angle brackets	`<filename>`	Identifies a variable in code; replace this place holder with a specific value or string.
courier	`owsctl start wrb`	Text to be entered exactly as it appears. Also used for functions.
square brackets	`[-c string]`	Identifies an optional item.
	`[on|off]`	Identifies a choice of optional items, each separated by a vertical bar (|), any one option can be specified.
braces	`{yes|no}`	Identifies a choice of mandatory items, each separated by a vertical bar (|).
ellipses	n,...	Indicates that the preceding item can be repeated any number of times.

The term, Oracle Server, refers to the database server product from Oracle Corporation.

The term, **oracle**, refers to an executable or account by that name.

The term, *oracle*, refers to the owner of the Oracle software.

Oracle Services and Support

A wide range of information about Oracle products and global services is available from:

- **http://www.oracle.com**

The sections below provide URLs for selected services.

Oracle Support Services

Technical Support contact information worldwide is listed at:

- **http://www.oracle.com/support**

Templates are provided to help you prepare information about your problem before you call. You will also need your CSI number (if applicable) or complete contact details, including any special project information.

Product and Documentation

For U.S.A customers, Oracle Store is at:

- **http://store.oracle.com**

Links to Stores in other countries are provided from this site.

Product documentation can be found at:

- **http://docs.oracle.com**

Customer Service

Global Customer Service contacts are listed at:

- **http://www.oracle.com/support**

Education and Training

Training information and worldwide schedules are available from:

- **http://education.oracle.com**

Oracle Technology Network

Register with the Oracle Technology Network (OTN) at:

- **http://technet.oracle.com**

OTN delivers technical papers, code samples, product documentation, self-service developer support, and Oracle key developer products to enable rapid development and deployment of application built on Oracle technology.

1

Introduction

This chapter describes Oracle Internet Application Server, how you can use it, and what technologies it supports.

Contents

- How Oracle Internet Application Server Supports Your Evolution into an E-Business
- Oracle Internet Application Server Overview
- Oracle Internet Application Server Architecture
- Available Versions
- Supported Technologies and Programming Languages

How Oracle Internet Application Server Supports Your Evolution into an E-Business

Becoming an e-business is part of being a business success. When you decide to move into the electronic arena, usually the first step taken is to create and host a company Web site. At the beginning, your Web site is a simple collection of static pages that gives your company market exposure on the Internet. As you become more comfortable with maintaining the Web site and more knowledgeable about the marketing and sales potential offered by an Internet presence, you want to offer more than electronic marketing collateral in the form of static Web pages.

Providing your customers with information that is generated dynamically from inventory or catalog data stores is also essential to growing your business. The next step for your enterprise may be to host an online store or a business-to-business

exchange. As your e-business grows, you want to offer your customers and employees greater access to products and services on a broad level. Self-service applications can help you increase your productivity and efficiency when dealing with customers and when managing your employees.

Each step in your evolution into a full-fledged e-business produces gains that motivate you to continue building your company's Internet presence. However, as you add more functionality to your company Web site, your Internet infrastructure must support that growth. Ideally, you want an infrastructure that can grow with your business. One that can start small and support your growing number of customers and employees. If you start with the right technology in the beginning, then you will not have to change later when you find your company in a success crisis.

The Oracle Solution: Oracle Internet Platform

Oracle Internet Platform provides integrated development, deployment, and management tools to simplify creating and deploying the e-business applications that you need to run your business on the Internet. Using Oracle Internet Platform, which consists of Oracle Internet Application Server and Oracle8*i*, you can:

- Run all of your applications
- Manage all of your data
- Deploy Internet and intranet portals

Run all of your applications Oracle Internet Application Server can support all of your transactional applications and business intelligence applications whether they are newly developed Java-based programs or your legacy applications.

Manage all of your data Oracle Internet Application Server is seamlessly integrated with Oracle8*i* so you can manage all of your data including relational data, business documents, or multimedia files. With Oracle8*i* Cache contained in Oracle Internet Application Server, your users can access your data faster than ever. (For information about Oracle8*i* Cache, see "Data Management Services" on page 2-13.)

Deploy Internet and intranet portals Using Oracle Internet Application Server, you can deploy portals, giving your employees and your customers the clarity of a single view of your applications and your information, and the ease of a single signon. Portals give users a consistent look, feel, and navigation to the interface so they can learn quickly and become more productive. Moreover, using portals you can set up self-service content submission, so your Web site requires no maintenance.

Oracle Internet Application Server Overview

Oracle Internet Application Server is a reliable, scalable, secure, middle-tier application server that is designed to support your evolution into an e-business. Using it, you can deliver Web content, host Web applications, process transactions, and make these services accessible to any client browser. Your users can access information, perform business analysis, and run business applications on the Internet, or on internal and external corporate intranets.

You can use Oracle Internet Application Server for:

- Publishing Content
- Running Transactional Applications

Publishing Content

Oracle Internet Application Server delivers Web content for the Internet and for internal and external intranets in the form of:

- Static Web Pages
- Dynamic Web Pages
- Business Documents
- Business Intelligence

Static Web Pages Using the proven technology of the Apache Web Server, Oracle Internet Application Server delivers static content such as HTML pages, XML data, and text files. Oracle HTTP Server Web listener delivers static content rapidly to a large user base with high performance, reliability, and stability.

Dynamic Web Pages With Oracle HTTP Server modules and the many Apache modules that are included in the HTTP Server component, Oracle Internet Application Server can support all of the current Internet application technologies for delivering dynamic content such as Java servlets, JavaServer Pages, PL/SQL Server Pages, XML, and CGI scripts using Perl.

In addition, Oracle Internet Application Server is tightly integrated with Oracle8*i* Cache and can call PL/SQL stored procedures through the Oracle HTTP Server PL/SQL module (mod_plsql) that integrates seamlessly with Oracle HTTP Server.

Business Documents On internal or external corporate intranets, business documents can easily be published. For example, you can publish word-processed documents, spreadsheets, slide presentations, and various documents in Adobe's Portable Document Format (.pdf).

Business Intelligence Using Reports Services, you can publish reports that you define according to your specifications which pull data directly from Oracle8*i*.

Forms Services also provide presentation services to client browsers via downloaded applets. Users can access Forms-based applications that perform business analysis and deliver the results to the client browser.

Running Transactional Applications

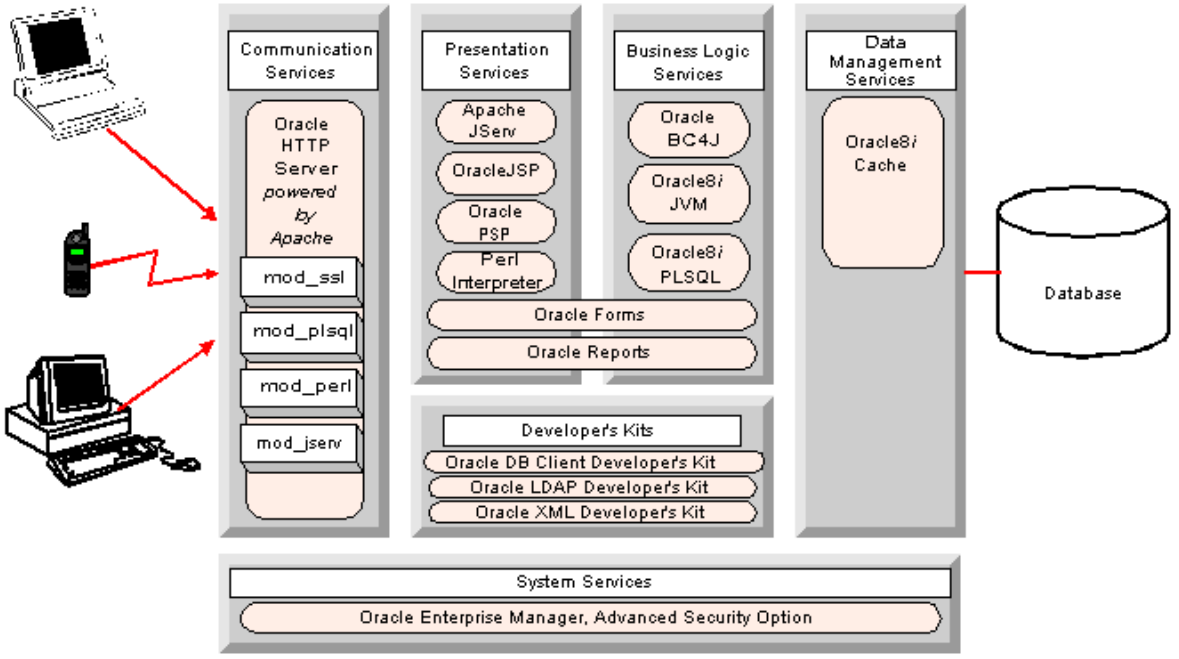
Oracle Internet Application Server supports both stateless and stateful transactional Web applications. A *stateless application* maintains no state information within its environment. However, it may maintain state information in a persistent store such as a database or a browser. For example, a shopping cart program that stores the contents of a user's cart in a cookie in the browser is a stateless application. Every time the user makes a request to the application, the browser sends over both the URL and the cookie. The application itself does not need to keep track of the user's shopping cart between successive client calls. This can be implemented as CGI scripts, stateless Java servlets, or in a variety of other ways.

On the other hand, a *stateful application* does maintain session state information within its runtime environment between successive client calls. If the shopping cart application, which was used in the above example, is stateful, then the application itself will keep track of the contents of the user's shopping cart instead of sending this information to the client's browser every time. Java servlets, for example, maintain session state by attaching state information to an HttpSession object that is specific to each user session. Every time a user issues a new HTTP request to the server, the server associates the request with the HttpSession object for that user.

For information about transactional application development models, see "Business Logic" on page 3-16.

Figure 1–1 lists Oracle Internet Application Server services and elements. For a description of these services and elements, see Chapter 2, "Oracle Internet Application Server Services".

Figure 1–1 Oracle Internet Application Server Services and Major Elements



Oracle Internet Application Server Architecture

Oracle Internet Application Server consists of a set of services that can be implemented in a distributed environment for scalability and reliability. The following sections provide an architectural overview of Oracle Internet Application Server.

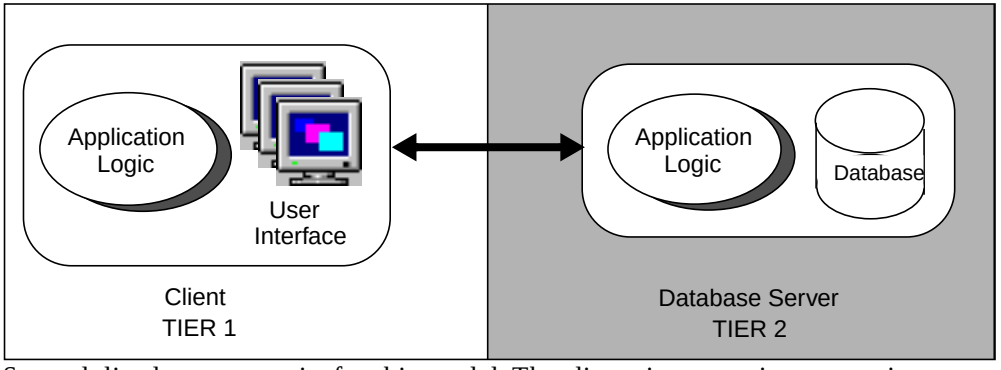
Two and Three-Tier Computing Models

Client/server computing architectures are commonly described as having two or more tiers according to how application logic is distributed between client and server. Minimally, a client/server architecture must have a client tier and a server tier. Oracle's internet computing model is based on a multi-tier computing model in which Oracle Internet Application Server functions as a middle tier, or application server tier.

Two-Tier Computing Model

Traditional database client/server architecture is based on a two-tier computing model. This model consists of a client tier and a database server tier (see Figure 1–2). Processing tasks and application logic are shared between the database server and the client.

Figure 1–2 Two-tier Computing Model



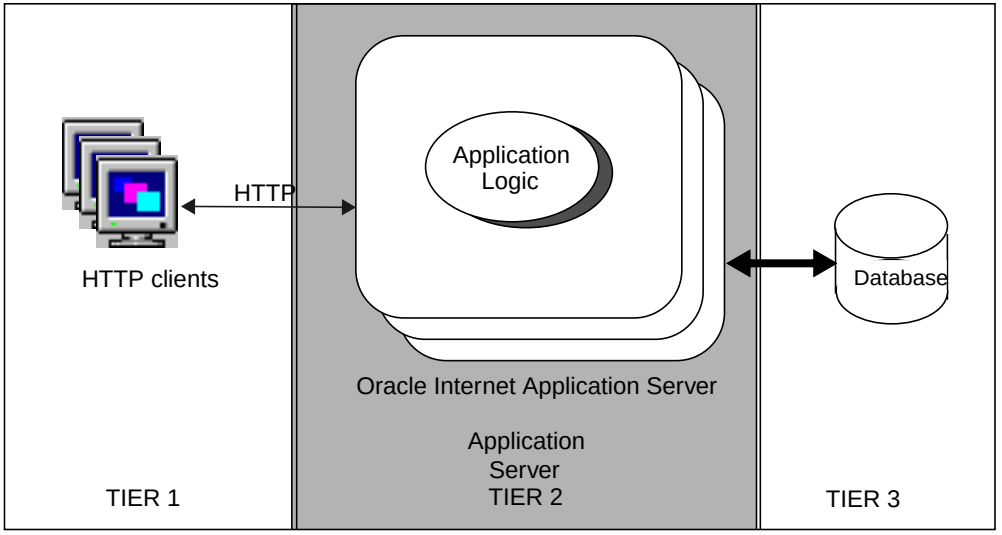
Several disadvantages exist for this model. The clients in a two-tier computing model are fat clients, where much of the processing power and application logic reside. This makes the clients costly to maintain. Furthermore, clients can be operating on different platforms, necessitating the deployment of platform-specific versions of applications.

Three-Tier Computing Model

The three-tier computing model evolved to address the problems of the two-tier model. In a three-tier model, a middle tier exists between clients and the database server. This middle tier consists of an application server that contains the bulk of the application logic. Clients in the model are thin clients. With this architecture, application logic resides in a single tier and can be maintained easily at one point. The architectural design of the middle tier is optimized for server functions including access to a database.

Oracle Internet Application Server serves as the middle tier of the three-tier model as shown in Figure 1–3.

Figure 1–3 Three-tier Oracle Internet Application Server Architecture



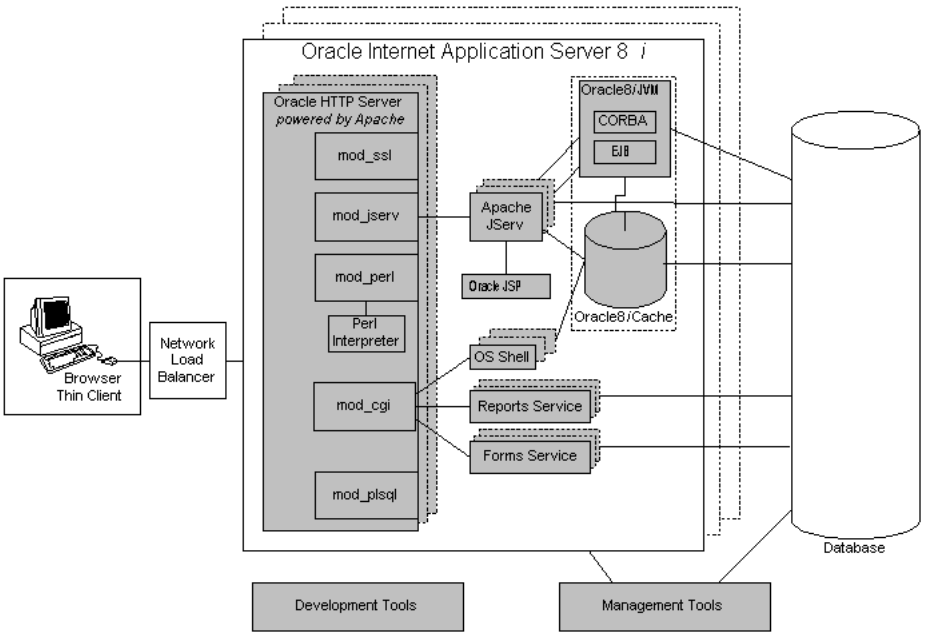
In this thin-client three-tier architecture, client software (the client tier) is lightweight enough to be downloaded on demand, and does little but present the user interface for a server-side application. The bulk of the application logic is implemented either in the middle (application server) tier or is stored in the database, as in the case of PL/SQL , Java stored procedures, or Enterprise JavaBeans.

Oracle Internet Application Server 8*i*

Oracle Internet Application Server enables users to deploy HTTP and HTML based applications within its multi-tiered architecture. The middle-tier server centrally manages application logic sending request responses back to thin clients, typically Web browsers. A third tier houses your origin database, so transaction processing on the database may be optimized. This multi-tiered model offers great savings in administration and maintenance costs when deploying applications.

Oracle Internet Application Server components reside on the middle-tier in a three-tier architecture as shown in Figure 1–4.

Figure 1–4 Oracle Internet Application Server Architecture



Alternatively, you may also run the Oracle Internet Application Server in a multi-tiered architecture of four or more tiers, depending on your application needs. For example, it is possible to divide the Report Services across multiple machines. In a four-tier configuration, you should run the Reports Web CGI or the Reports Servlets on the same machine as the HTTP Server and run the Report Server on a separate machine. In this example, the client browser resides in the first tier and your origin database resides on a fourth-tier.

For more information on this example, see *Publishing Reports to the Web with Internet Application Server* on your Documentation Library CD-ROM.

Available Versions

Currently, Oracle Internet Application Server is available in two versions: the Standard Edition and the Enterprise Edition. The Standard Edition is suitable if you need a lightweight Web server with minimal application support. If you have a medium-sized to large-sized Web site that handles a high volume of transactions, then the Enterprise Edition is recommended. Table 1–1 lists what services each edition contains:

Table 1–1 Available Versions of Oracle Internet Application Server

Services	Standard Edition	Enterprise Edition
Oracle HTTP Server	x	x
mod_plsql	x	x
mod_perl	x	x
mod_ssl	x	x
mod_jserv	x	x
Apache JServ	x	x
OracleJSP (JavaServer pages)	x	x
Oracle PL / SQL Server Pages	x	x
Perl Interpreter	x	x
Oracle BC4J (Business Components for Java)	x	x
Oracle8*i* JVM (Java Virtual Machine)	x	x
Oracle8*i* PLSQL	x	x
Oracle Forms Services		x
Oracle Reports Services		x
Oracle 8*i* Cache		x
Oracle Database Client Developer's Kit	x	x
Oracle XML Developer's Kit	x	x
Oracle LDAP Developer's Kit	x	x
Oracle Enterprise Manager[1]	x	x
Advanced Security Option	x	x

[1] Standard edition contains the Enterprise Manager console only; Enterprise edition contains both Enterprise Manager console and Management Server. See "Oracle Enterprise Manager" on page 2-24.

Supported Technologies and Programming Languages

For publishing content, transaction processing, and program development and deployment, Oracle Internet Application Server supports these technologies and programming languages:

- Oracle BC4J (Business Components for Java Runtime)
- EJB (Enterprise JavaBeans)
- Java 2 (JDK 1.2.2)
- Java Stored Procedures
- JDBC (Java Database Connectivity)
- JMS (Java Messaging Service)
- JNDI (Java Naming and Directory Interface)
- Servlets
- JSP (JavaServer Pages)
- SQLJ
- CORBA (Common Object Resource Broker Architecture)
- PL/SQL
- SQL
- OCI (Oracle Call Interface)
- ODBC (Open Database Connectivity)
- OLE (Object Linking and Embedding)
- IIOP (Internet Inter-ORB Protocol)
- RMI (Remote Method Invocation)/IIOP
- LDAP (Lightweight Directory Access Protocol)
- SSL (Secure Socket Layer)
- HTML (Hyper Text Markup Language)
- XML (eXtensible Markup Language)
- Perl

For a description of each Oracle Internet Application Server service and element, see Chapter 2, "Oracle Internet Application Server Services".

For information about developing and deploying applications in Oracle Internet Application Server, see Chapter 3, "Developing Applications for Oracle Internet Application Server".

2

Oracle Internet Application Server Services

To deliver application hosting, Web content, security, and integration, Oracle Internet Application Server includes services that support the Oracle Internet Platform. This chapter provides an overview of each service in Oracle Internet Application Server.

Contents

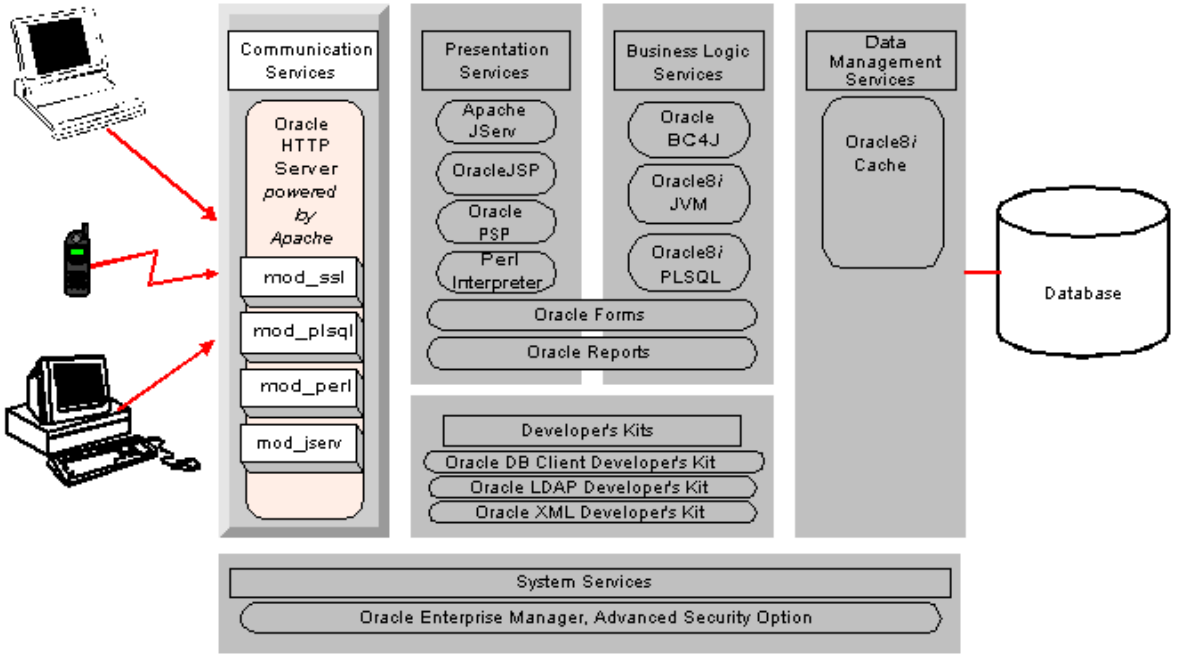
The services described in this chapter are:

- Communication Services
- Presentation Services
- Business Logic Services
- Data Management Services
- Developer's Kits
- System Services

Communication Services

These services handle all incoming requests received by Oracle Internet Application Server. Some of these requests are processed by Oracle HTTP Server and some requests are routed to other areas of Oracle Internet Application Server for processing. The major elements that support these services are described in the following sections and shown in Figure 2–1.

Figure 2–1 Communication Services in Oracle Internet Application Server



Oracle HTTP Server *powered by Apache*

Oracle Internet Application Server uses Oracle HTTP Server, which is powered by Apache Web server technology. Using the Apache server technology offers the following:

- **Scalability**: Because the Apache server is designed to be very scalable, it can be replicated across many middle-tier nodes.
- **Stability**: The Apache server is the defacto standard for Web listeners. Apache, an open source technology, has been continuously improved by a large community of developers since its first release. Consequently, Apache Web server technology is very stable.
- **Speed**: The compact design of the Apache server makes it very fast when it responds to requests, and it is possible to rapidly reconfigure Apache using *graceful restart*, which reloads the configuration files in seconds.
- **Extensibility**: The Apache server delegates the handling of HTTP requests to its modules (mods), which add functionality not included in the server by default. Using the Apache APIs, it is easy to extend Apache functionality. A large number of mods have already been created and are included on your CD-ROM. Although the default Apache HTTP server supports only stateless transactions, you can configure it to support stateful transactions by leveraging the functionality supplied by Apache JServ. (See Apache JServ on page 2-6.)

For detailed information about Oracle HTTP Server, refer to the *Oracle HTTP Server* documentation on your Documentation Library CD-ROM.

Oracle HTTP Server Modules (mods)

In addition to the compiled Apache mods provided with Oracle HTTP server, Oracle has enhanced several of the standard mods and has added Oracle-specific mods, which are described in the following sections.

mod_ssl

This module provides standard HTTPS that is fully supported by Oracle. It enables secure listener connections with an Oracle-provided encryption mechanism via the Secure Sockets Layer (SSL).

mod_plsql

This module routes PL/SQL requests to Oracle8*i* PLSQL service, which, in turn, delegates the servicing of requests to PL/SQL programs.

For detailed information, see *Using mod_plsql* on your Documentation Library CD-ROM.

mod_perl

This module forwards Perl application requests to the Perl Interpreter that is embedded in Oracle HTTP Server. The primary advantages of using mod_perl are power and speed. The embedded Perl Interpreter saves the overhead of starting an external interpreter, and the code caching feature, where modules and scripts are loaded and compiled only once, allows the server to run already-loaded and compiled code.

For detailed information, see the *Apache mod_perl Documentation* on your Documentation Library CD-ROM.

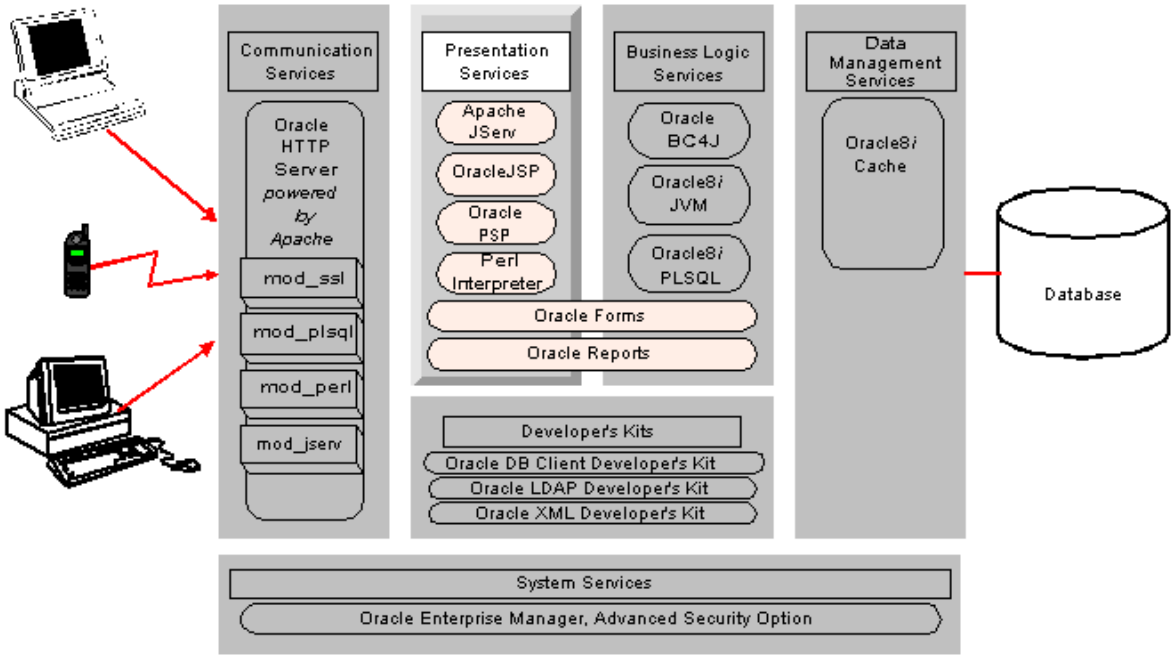
mod_jserv

This module routes all servlet requests to Apache JServ servlet engine that is embedded in Oracle HTTP Server. It can share servlets across multiple zones and ensures that requests get routed to the same servlet engine.

Presentation Services

These services deliver dynamic content to client browsers, supporting servlets, JavaServer Pages, Perl / CGI scripts, PL / SQL Pages, forms, and business intelligence. The major elements that support these services are described in the following sections and shown in Figure 2–2.

Figure 2–2 Presentation Services in Oracle Internet Application Server



Apache JServ

Apache JServ is a 100% pure Java servlet engine fully compliant with the Sun Microsystems Java Servlet APIs 2.0 specification. Apache JServ works on any version 1.1 compliant Java Virtual Machine and executes any Java servlet compliant with version 2.0.

When the HTTP server receives a servlet request, it is routed to mod_jserv, which in turn forwards the request to the Apache JServ servlet engine.

For detailed information about Apache JServ, see the *Apache JServ Documentation* on your Documentation Library CD-ROM.

OracleJSP (JavaServer pages)

As Sun Microsystems explains, JavaServer Pages technology extends Java Servlet technology, and supports the use of Java calls and scriptlets within HTML and XML pages. Using JSP pages, you can combine static template data with dynamic content to create user interfaces. JSP pages support component-based development, separating business logic (usually in JavaBeans) from the presentation, thus allowing developers to focus on their areas of expertise. Consequently, JSP developers (who may not know Java) can focus on presentation logic, while Java developers can focus on business logic.

For general information about JavaServer Pages, refer to the JavaServer Pages Specification (available from http://java.sun.com).

OracleJSP is a complete implementation of JavaServer Pages 1.1 as specified by Sun Microsystems. Moreover, OracleJSP extends the 1.1 specification and provides these benefits:

- **Portability between Servlet Environments**: OracleJSP pages are supported on all Web servers that support Java servlets built to the 2.0 or higher specification. Consequently, you can migrate your existing JSP pages 1.0-compliant code to Oracle Internet Application Server without rewriting it.
- **Support for SQLJ:** SQLJ is a standard syntax for embedding SQL commands directly into Java code. OracleJSP supports SQLJ programming in JSP scriptlets. This includes support for an additional file name extension, `.sqljsp`, which results in OracleJSP translator invoking Oracle SQLJ translator.
- **OracleJSP Markup Language (JML)**: As a sample tag library, Oracle provides the JML tag set which allows developers to use loop, conditional, and other high-level programming logic without having to know Java syntax.

- **Extended National Language Support (NLS)**: OracleJSP provides extended NLS support for servlet environments that cannot encode multibyte request parameters and bean property settings. For such environments, OracleJSP offers the `translate_params` configuration parameter, which can be enabled to direct OracleJSP to override the servlet container and do the encoding itself.
- **Extended Datatypes**: OracleJSP provides the `JmlBoolean`, `JmlNumber`, `JmlFPNumber`, and `JmlString` JavaBean classes in the `oracle.jsp.jml` package to wrap the most common Java datatypes. These extended datatypes provide a way to work around the limitations of Java primitive types and wrapper classes in the standard `java.lang` package.
- **Custom JavaBeans:** OracleJSP includes a set of custom JavaBeans for accessing an Oracle database.

For detailed information about OracleJSP, see the *OracleJSP Developer's Guide* on your Documentation Library CD-ROM.

Oracle PL/SQL Server Pages (PSP)

PL/SQL server pages are analogous to JavaServer Pages, but they use PL/SQL rather than Java for the server-side scripting. Oracle PSP includes the PSP Compiler and the PL/SQL Web Toolkit. Using this service when developing applications, you can separate page format from application logic.

Starting with an existing Web page, or with an existing stored procedure, you can create dynamic Web pages. These dynamic Web pages can perform database operations and display the results as HTML, XML, plain text, or some other document type that your browser has been configured to recognize. Typically, a PL/SQL server page is intended to be displayed in a Web browser. However, it could also be retrieved and interpreted by a program that can make HTTP requests, such as a Java or Perl application.

A PSP file can contain whatever content you like, with text and tags interspersed with PSP directives, declarations, and scriptlets:

- In the simplest case, it is nothing more than an HTML file or an XML file. Compiling it as a PSP produces a stored procedure that outputs the exact same HTML or XML file.
- In the most complex case, it is a PL/SQL procedure that generates all the content of the Web page, including the tags for title, body, and headings.
- In the typical case, it is a mix of HTML or XML (which provide the static parts of the page) and PL/SQL (filling in the dynamic content).

You can author the Web pages in a script-friendly HTML authoring tool, and drop the pieces of PL/SQL code into place. Embedding the PL/SQL code in the HTML page that you create lets you write content quickly and follow a rapid, iterative development process.

For detailed information about Oracle PL/SQL Server Pages, see the *Oracle8i Application Developer's Guide - Fundamentals* in the Oracle8*i* database documentation set.

Perl Interpreter

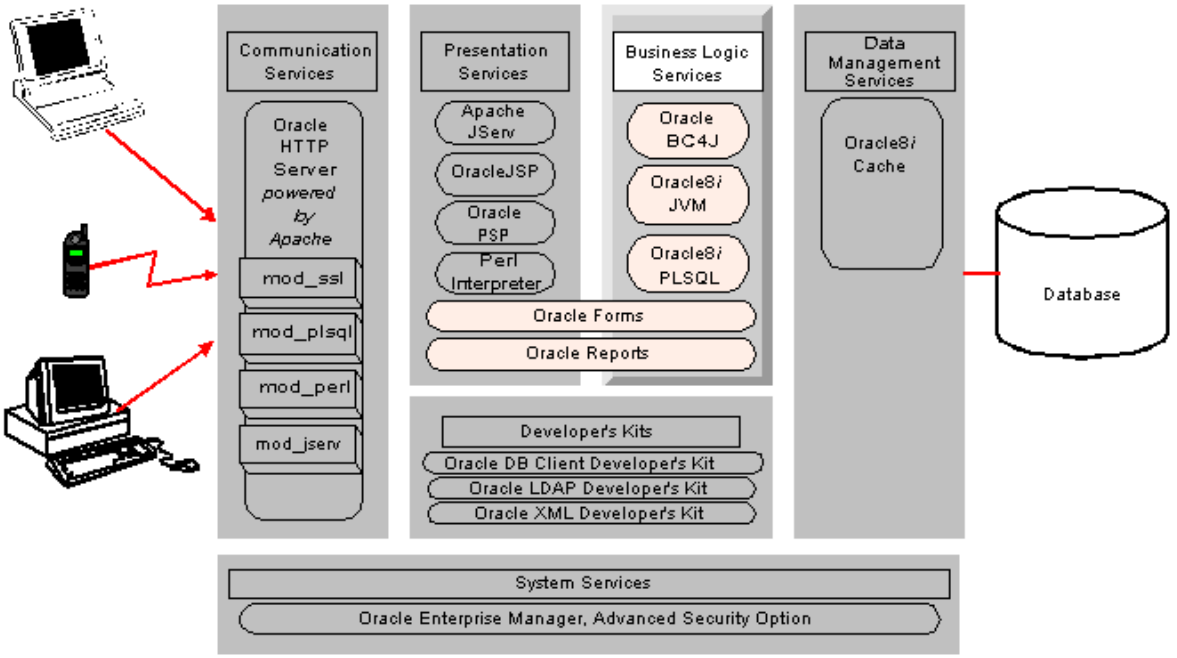
A persistent Perl runtime environment that is embedded in Oracle HTTP Server, thus saving the overhead of starting an external interpreter. When Oracle HTTP Server receives a Perl request, it is routed to mod_perl, which, in turn, routes the request to the Perl Interpreter for processing.

For detailed information about the Perl Interpreter, see the *Apache mod_perl Documentation* on your Documentation Library CD-ROM.

Business Logic Services

These services support your application logic. The following sections describe the major elements that provide business logic services in Oracle Internet Application Server. The major elements that support these services are described in the following sections and are shown in Figure 2–3.

Figure 2–3 Business Logic Services in Oracle Internet Application Server



Oracle BC4J (Business Components for Java)

Oracle BC4J is a 100%-Java, XML-powered framework that enables productive development, portable deployment, and flexible customization of multi-tier, database-enabled applications from reusable business components. Application developers can use this framework to:

- Author and test business logic in components that automatically integrate with databases.
- Reuse business logic through multiple SQL-based views of data that support different application tasks.
- Access and update the views from servlets, JavaServer Pages (JSPs), and Thin-Java Swing clients.
- Customize application functionality in layers without modifying the delivered application.

Once developed, these application services can then be deployed as either EJB Session Beans or CORBA Server Objects on Oracle Internet Application Server.

For detailed information about Oracle BC4J and how to use it, see the *Oracle BC4J* documentation on your Documentation Library CD-ROM.

Oracle8*i* JVM

Designed as a highly scalable, server-side Java platform, Oracle8*i* JVM is an enterprise-class 100% Java-compatible server environment that supports Enterprise JavaBeans, CORBA, and database stored procedures. Oracle8*i* JVM achieves high scalability through its unique architectural design, which minimizes the burden and complexity of memory management when the number of users increases.

Using Oracle8*i* JVM provides a number of advantages:

- Enhanced garbage collection algorithms that facilitate more efficient memory allocation.
- Support for concurrent, stateful, conversational clients.
- All common resources shared across multiple user sessions so only one Java Virtual Machine needs to run to provide user session support.
- No multi-threaded applications are necessary because the Oracle8*i* JVM environment handles load balancing.

Oracle8*i* JVM is the common foundation for running Java and Java services in Oracle Internet Application Server and Oracle8*i*. Consequently, components can be moved across tiers seamlessly without having to change any code for better performance.

For detailed information on Oracle8*i* JVM and its Java environment, see the *Oracle8i JVM* documentation on your Documentation Library CD-ROM.

Oracle8*i* PLSQL

Oracle8*i* PLSQL is a scalable engine for running business logic against data in Oracle8*i* Cache and Oracle8*i* database. It provides an environment that enables users to use their browsers to invoke PL/SQL procedures stored in Oracle databases. The stored procedures can retrieve data from tables in the database and generate HTML pages that include the data to return to the client browser.

For detailed information on Oracle8*i* PLSQL, see *Using mod_plsql* on your Documentation Library CD-ROM.

Oracle Forms Services

Using Oracle Forms Services, you can run applications based on Oracle Forms technology over the Internet or over your corporate intranet. On the application server tier, Oracle Forms Services consist of a listener and a runtime engine, where the application logic is stored. On the client tier, Oracle Forms Services consist of a thin Java-based Forms client (Java applet) that provides the user interface for the runtime engine, and Oracle JInitiator, a Java plug-in that provides the ability to specify the use of a specific Java virtual machine on the client.

In Oracle Internet Application Server, when a user submits a URL to launch an Oracle Forms-based application, the Web listener accepts the request and downloads the Oracle Forms applet to the user's browser. Then the Oracle Forms applet establishes a persistent connection to an Oracle Forms runtime engine. All processing takes place between the Oracle Forms client applet and the Oracle Forms Services runtime engine, seamlessly handling any queries or commits to the database.

For detailed information, see the *Oracle Forms Services* documentation on your Documentation Library CD-ROM.

Oracle Reports Services

Using Oracle Reports Services and its Reports Servlet services, you can run new and existing Oracle Reports Developer reports on an internal company intranet, an external company extranet, or on the Internet. Oracle Reports Services is optimized to deploy Oracle Reports Developer applications (Reports and Graphics) in a multi-tiered environment. It consists of the server component, runtime engines, and the servlet runner.

In Oracle Internet Application Server, when a client submits a request for a report, the Oracle HTTP Server Web listener routes that request to the Oracle Reports Services server component. The server routes the request to the Oracle Reports Services runtime engine, which runs the report. Then the report output is sent back to the client via the Oracle HTTP Server Web listener.

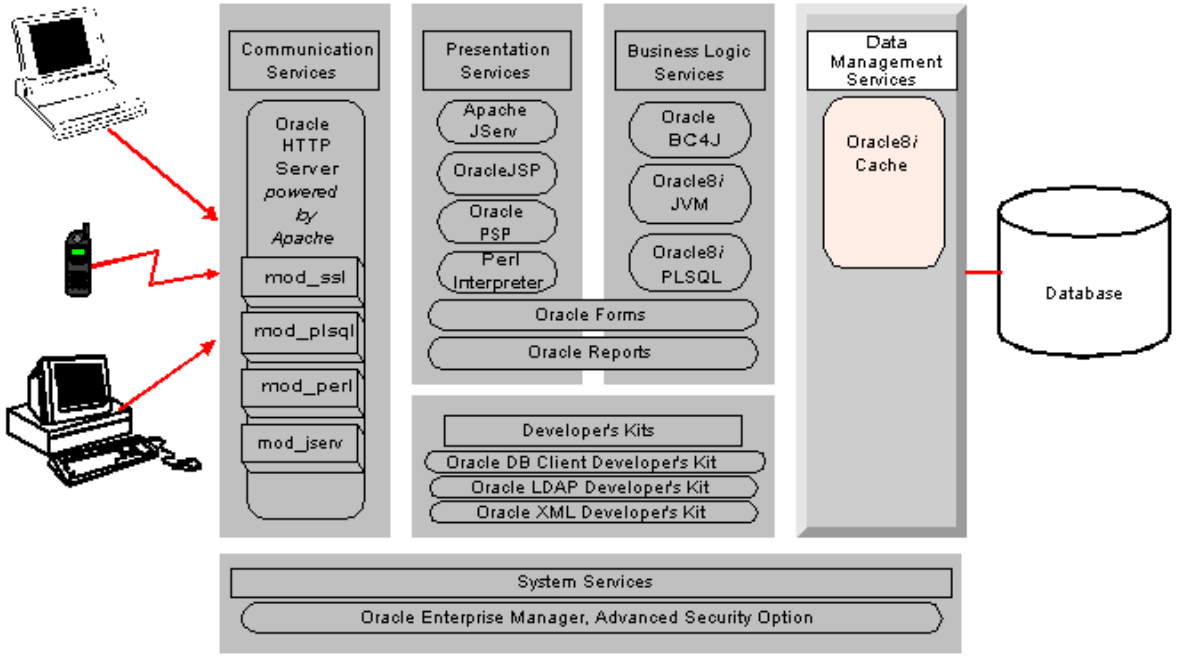
For detailed information, see the *Oracle Reports Services* documentation on your Documentation Library CD-ROM.

Data Management Services

To reduce the load on the back-end database instance, and to avoid network roundtrips for read-only data, Oracle Internet Application Server includes Oracle8*i* Cache. This service runs stateful servlets, JavaServer Pages, Enterprise JavaBeans, and CORBA objects.

Figure 2–4 shows the data management services in Oracle Internet Application Server.

Figure 2–4 Data Management Services in Oracle Internet Application Server



The following section contains information about Oracle8*i* Cache.

Oracle8*i* Cache

Oracle8*i* Cache is a data cache that resides on the middle tier as a component of Oracle Internet Application Server. It improves the performance and scalability of applications that access Oracle databases by caching frequently used data on the middle-tier machine. With Oracle8*i* Cache, your applications can process several times as many requests as their original capacity because round-trips to the database are greatly reduced.

Who Should Use Oracle8*i* Cache?

If your applications meet the following criteria, you can use Oracle8*i* Cache to boost the scalability of your Web sites and the performance of your applications:

- Your applications access Oracle databases.
- You use a multiple-tier environment, where the clients, Oracle Internet Application Server, and Oracle database servers are located on separate systems.
- Your applications communicate with an Oracle database through Oracle Call Interface (OCI), or an access layer built on OCI, like JDBC-OCI, ODBC, or OLE. The applications can be written using scripting or programming languages.
- Web or application users generate mostly read-only queries.
- Your applications can tolerate some synchronization delay with data in the origin database. That is, the cache does not need to be as up-to-date as the data in the origin database. (You decide how often to refresh the data.)

Performance and Scalability Benefits

Using Oracle8*i* Cache in the middle tier provides a number of performance and scalability benefits. The most significant benefits are listed below.

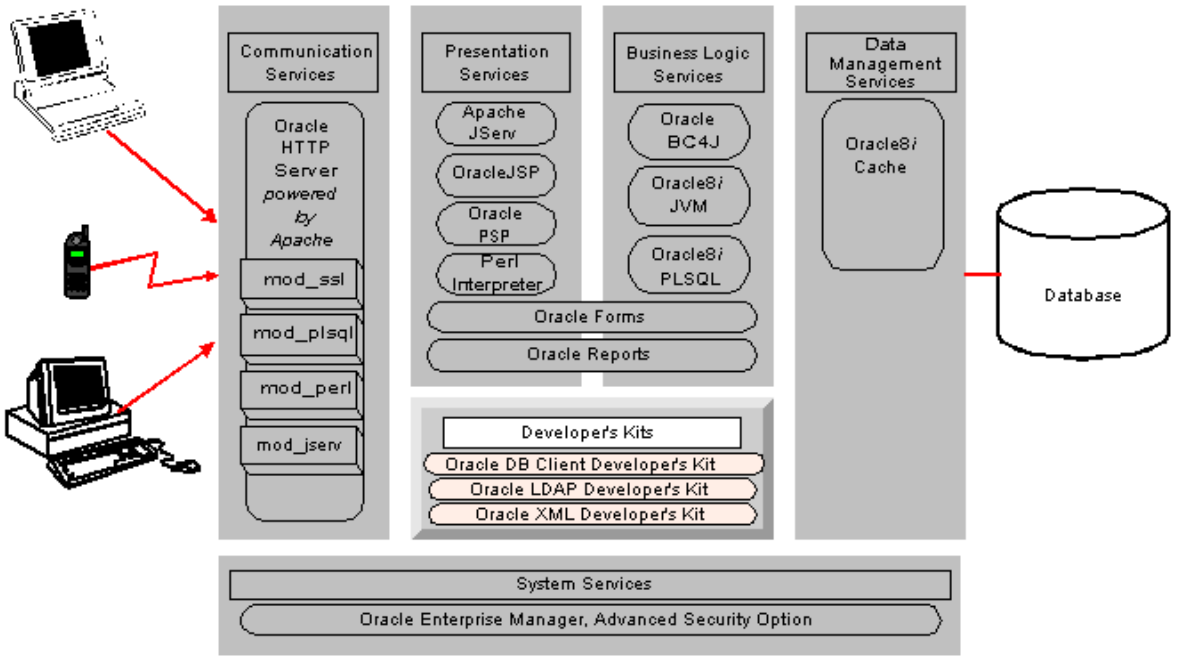
- Processing database queries on the middle tier reduces time spent sending and receiving data over the network so hardware resources are used more efficiently.
- Reducing the load on the database server tier means that existing databases can support more users.

For detailed information about Oracle8*i* Cache, see the *Oracle8i Cache Concepts and Administration Guide* on your Documentation Library CD-ROM.

Developer's Kits

To support application development and deployment, several toolkits containing libraries and tools are included in Oracle Internet Application Server. These services are described in the following sections and are shown in Figure 2–5.

Figure 2–5 Developer's Kits in Oracle Internet Application Server



Oracle Database Client Developer's Kit

Oracle Database Client Developer's Kit contains client libraries for Oracle8*i* and the Java client libraries (JMS, SQLJ, and JDBC) described in the following sections.

Oracle Java Messaging Service (JMS) Toolkit

Oracle JMS extends the standard Sun Microsystems Java Message Service specification, version 1.02 as defined by Sun Microsystems. In addition to the standard JMS features, Oracle JMS provides a Java API for Oracle Advanced Queuing (AQ). This API supports the AQ administrative operations and other AQ features including:

- **An administrative API** to create queue tables, queues, and topics.
- **Point-to-multipoint communication** (extends the standard point-to-point), using recipient lists for topics.
- **Message propagation between destinations**, which allows applications to define remote subscribers.
- **Support for transacted sessions** so you can perform JMS as well as SQL operations in one atomic transaction.
- **Message retention** after messages have been dequeued.
- **Message delay** so you can make messages visible after a specified time.
- **Exception handling** so messages can be moved to exception queues if they cannot be processed successfully.
- **Support for Oracle8*i* `AdtMessages`** message types. (These are stored in the database as Oracle objects. Consequently, the payload of the message can be queried after it is enqueued. Subscriptions can be defined on the contents of these messages in addition to the message properties.)

For detailed information on Oracle Java Messaging Service, see the *Oracle8i Application Developer's Guide - Advanced Queuing* in the Oracle8*i* documentation set.

Oracle SQLJ Translator

Oracle SQLJ is a preprocessor that developers can use to embed *static* SQL operations in Java code. A SQLJ program is a Java program that contains embedded static SQL statements which comply with the ANSI-standard SQLJ Language Reference syntax. Static SQL operations are predefined—the operations themselves do not change in real-time as a user runs the application, although the data values transmitted can change dynamically.

Oracle SQLJ consists of a *translator* and a *runtime component*. The translation process replaces embedded SQL with calls to the SQLJ runtime, which implements the SQL operations. In standard SQLJ, this is typically done through calls to a JDBC driver. In the case of Oracle Internet Application Server, you use an Oracle JDBC driver [see "Oracle Java Database Connectivity (JDBC) Drivers" in the next section]. When end users run SQLJ applications, the runtime is invoked to handle the SQL operations.

Oracle SQLJ Translator is a precompiler, which developers run after creating SQLJ source code. The translator checks the following:

- Syntax of the embedded SQL.
- SQL constructs, against a specified database schema to ensure consistency within a particular set of SQL entities (optional). For example, it verifies table names and column names.
- Datatypes, to ensure that the data exchanged between Java and SQL have compatible types and proper type conversions.

The translator, written in pure Java, supports a programming syntax that allows you to embed SQL operations inside SQLJ executable statements. SQLJ executable statements and SQLJ declarations are preceded by the `#sql` token and can be interspersed with Java statements in a SQLJ source code file.

The translator produces a `.java` file and one or more SQLJ *profiles*, which are serialized Java resources that contain details about the embedded SQL operations in your SQLJ source code. After the translator produces the `.java` file and the profiles, SQLJ automatically invokes a Java compiler to produce `.class` files from the `.java` file.

Oracle SQLJ Runtime is a thin layer of pure Java code that runs above the JDBC driver. When Oracle SQLJ translates your SQLJ source code, embedded SQL commands in your Java application are replaced by calls to the SQLJ runtime. Runtime classes act as wrappers for equivalent JDBC classes, providing special SQLJ functionality. When the end user runs the application, the SQLJ runtime acts as an intermediary, reading information about your SQL operations from the profile and passing instructions along to the JDBC driver.

A SQLJ runtime can be implemented to use any JDBC driver or vendor-proprietary means of accessing the data cache or origin database. Oracle SQLJ runtime requires a JDBC driver but can use any standard JDBC driver. To use Oracle-specific database types and features, however, you must use an Oracle JDBC driver.

For detailed information about Oracle SQLJ, see the *Oracle 8i SQLJ Developer's Guide and Reference* on your Documentation Library CD-ROM.

Oracle Java Database Connectivity (JDBC) Drivers

JDBC is a database access API that enables you to connect to a database and then prepare and execute SQL statements against the database. Core Java class libraries provide the interfaces and Oracle JDBC drivers implement those interfaces to access an Oracle database.

Oracle JDBC drivers are described in Table 2–1:

Table 2–1 Oracle JDBC Drivers

Driver	Description
JDBC Thin Driver	You can use the JDBC thin driver to write 100% pure Java applications that access Oracle SQL data. You can use it for EJBs in the database. The JDBC thin driver is especially well-suited to Web browser-based applications because you can dynamically download it from a Web page.
JDBC Oracle Call Interface Driver	The JDBC Oracle Call Interface (OCI) driver accesses Oracle-specific native code (that is, non-Java) libraries on the client or in the middle tier, providing a richer set of functionality and some performance boost compared to the JDBC thin driver, at the cost of significantly larger size.
JDBC Server-side Internal Driver	Oracle8*i* JVM uses the JDBC server-side internal driver when Java code runs on Oracle Internet Application Server. It allows Java applications running in Oracle8*i* JVM to access locally defined data (that is, on the same machine and in the same process) with JDBC. It provides a further performance boost because of its ability to use underlying Oracle RDBMS libraries directly, without the overhead of an intervening network connection between your Java code and SQL data. This driver can also be used for EJBs in Oracle8*i* JVM.

For detailed information on the Oracle JDBC drivers and Oracle extensions to the standard JDBC API, see the *Oracle 8i JDBC Developer's Guide and Reference* on your Documentation Library CD-ROM.

Oracle XML Developer's Kit

Oracle XML Developer's Kit (XDK) contains XML component libraries and utilities that you can use to XML-enable applications and Web sites. The XDK in the Oracle Internet Application Server contains the components described in Table 2–2.

Table 2–2 XDK Components in Oracle Internet Application Server

XDK Component	Description
XML Parser for Java	Parses XML documents or stand-alone Document Type Definitions (DTDs) so Java applications can process them. This parser uses industry standard Document Object Model (DOM) and Simple API for XML (SAX) interfaces. ■ DOM APIs permit applications to access and manipulate an XML document as a tree structure in memory. This interface is used by such applications as editors. ■ SAX APIs permit an application to process XML documents using an event-driven model.
XML Class Generator for Java	Automatically generates Java class source files from XML DTDs. Using these classes, Java applications can construct, validate, and print XML documents that comply with the input DTD. The class generator works in conjunction with the parser, which parses the DTD and passes the parsed document to the class generator.
XML Transviewer JavaBeans	Displays and transforms XML documents and data via Java components to add graphical or visual interfaces to XML applications. The included beans are: ■ DOM Builder Bean: Encapsulates the Java XML Parser with a bean interface and extends its functionality to permit asynchronous parsing. By registering a listener, Java applications can parse large or successive documents having control return immediately to the caller. ■ TreeViewer Bean: Displays XML formatted files graphically as a tree. ■ SourceViewer Bean: Displays XML files with color syntax highlighting when modifying an XML document with an editing application. ■ XSL Transformer Bean: Transforms an XML document to other text-based formats, including HTML and DDL, by applying an XSL (eXtensible Stylesheet Language) stylesheet.

Table 2–2 XDK Components in Oracle Internet Application Server (Cont.)

XDK Component	Description
XSQL Servlet	Processes SQL queries and outputs the result set as XML. This processor is implemented as a Java servlet and takes as its input an XML file containing embedded SQL queries.

For detailed information about this service, see the *Oracle XML Developer's Kit* documentation on your Documentation Library CD-ROM.

Oracle LDAP Developer's Kit

This service supports client interaction with Oracle Internet Directory, the Oracle LDAP (Lightweight Directory Access Protocol), version 3-compliant directory server product. Oracle Internet Directory combines a native implementation of the Internet Engineering Task Force's (IETF) LDAPv3 standard with an Oracle8*i* back-end data store.

Specifically, you can use Oracle LDAP Developer's Kit to develop and monitor LDAP-enabled applications. It supports client calls to directory services, encrypted connections, and you can use it to manage your directory data. Oracle LDAP Developer's Kit contains the following subcomponents:

Oracle Internet Directory C API An LDAP C API that is based on the Internet Engineering Task Force (IETF) RFC 1823. This component supports client calls to directory services from a standard C programming environment.

JNDI 1.2 (Java Naming and Directory Interface) Sun Microsystems programming interface that enables Java-based programs to communicate directly with LDAP servers such as Oracle Internet Directory. (See http://java.sun.com/products/jndi for documentation about JNDI.)

SSL Toolkit Standards-based extensions to the LDAP API that enable one-way, two-way, and simple encrypted connections to directory services.

Oracle Internet Directory Command Line Tools A set of LDAP-compliant command line tools that you can use for querying and returning results from Oracle Internet Directory.

Oracle Directory Manager An Oracle Java-based application for managing data stored in Oracle Internet Directory.

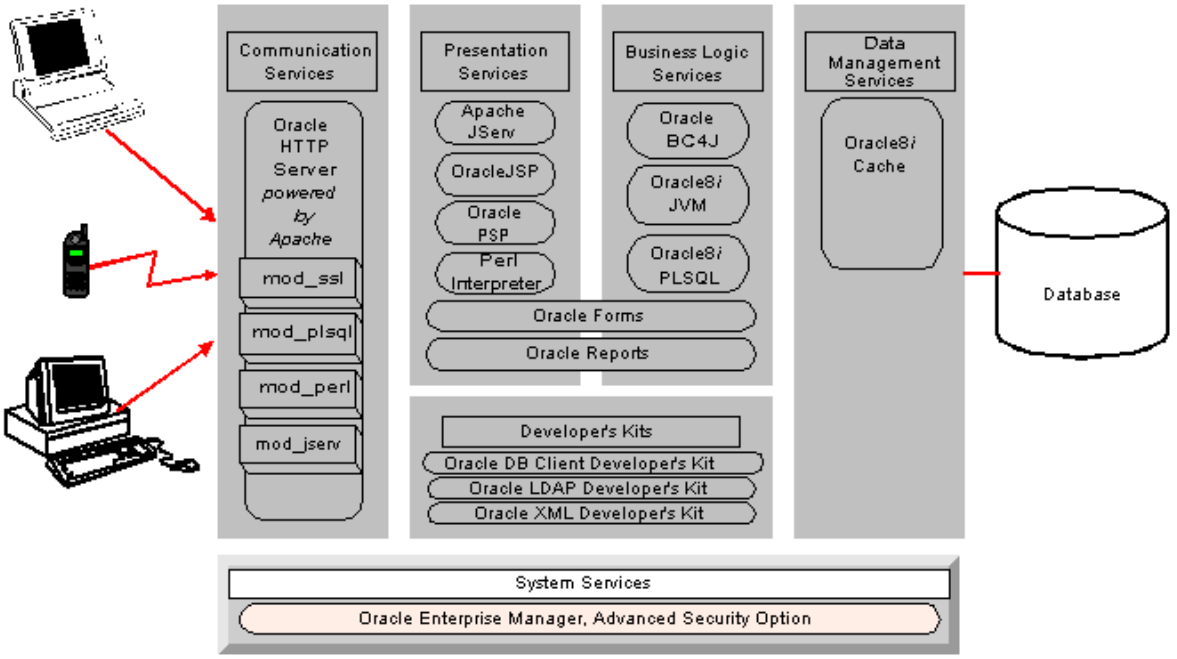
For information about these Oracle LDAP Developer's Kit services, refer to the documentation listed below:

For information about…	Refer to…
Oracle Internet Directory C API SSL Toolkit Oracle Internet Directory Command Line Tools	*Oracle Internet Directory Application Developer's Guide* (located on your Documentation Library CD-ROM)
Oracle Directory Manager	*Oracle Internet Directory Administrator's Guide*

System Services

To provide system management and security services, Oracle Internet Application Server includes Oracle Enterprise Manager and the Advanced Security Option and, in a future release, Advanced Security Option. These system services provide a comprehensive management framework for your entire Oracle environment and network security via SSL (Secure Sockets Layer)-based encryption and authentication facilities. The major elements that support these services are described in the following sections and shown in Figure 2–6.

Figure 2–6 System Services in Oracle Internet Application Server



Oracle Enterprise Manager

Oracle Enterprise Manager is a system management tool that provides an integrated solution for centrally managing your Oracle platform. Combining a graphical console, Oracle Management Servers, Oracle Intelligent Agents, common services, and administrative tools, Oracle Enterprise Manager provides a comprehensive systems management platform for managing your Oracle products.

In Oracle Internet Application Server, you use Oracle Enterprise Manager console, a graphical interface, to manage Oracle8*i* Cache, Oracle Forms Services, and the host operating system.

From the client interface, Oracle Enterprise Manager console, you can perform the following tasks:

- Centrally manage, administer, and diagnose Oracle Internet Application Server Oracle8*i* Cache and Oracle Forms Services
- Effectively monitor and respond to the status of your Oracle family of products and third-party services
- Schedule activities on multiple nodes at varying time intervals
- Monitor networked services for events
- Customize your display by organizing your server components and services into logical administrative groups

Three-Tier Architecture Oracle Enterprise Manager architecture consists of a three-tier framework, which is described in Table 2–3.

Table 2–3 Oracle Enterprise Manager Three-Tier Architecture

Tier	Description
1st-tier Console	Provides a graphical interface for administrators. This tier includes Cache Manager console for managing Oracle8*i* Cache. For information about Cache Manager console, see the *Oracle8i Concepts and Administration Guide* on your Documentation Library CD-ROM.
2nd-tier Management Servers[1]	Provides a scalable middle tier for processing all system management tasks.
3rd-tier Intelligent Agents[2]	Monitors databases and services on each node, and executes tasks received from the Management Server.

[1] Only available with Oracle Internet Application Server Enterprise edition.

[2] Intelligent Agents support Simple Network Management Protocol (SNMP), enabling third-party applications to communicate with the agent and be managed along with Oracle services.

Note: For this release, Oracle8*i* Cache and Oracle Forms are the only services in Oracle Internet Application Server that you can manage using Oracle Enterprise Manager. If you do not plan to use Oracle8*i* Cache or Oracle Forms Services, then you are not required to deploy Oracle Enterprise Manager.

Benefits of Oracle Enterprise Manager This system management tool enables administrators to maintain the highest level of performance and availability while controlling system management costs. The major benefits of Oracle Enterprise Manager are:

- Single point of management
- Multi-administrator system support
- Scalability for growing, distributed environments
- Extensible architecture
- Automated lights-out administration
- Autonomous Intelligent Agents
- Ease of use

For information about…	Refer to…
Oracle Enterprise Manager	Oracle Enterprise Manager documentation on your Documentation Library CD-ROM
Cache Manager	*Oracle8i Cache Concepts and Administration Guide* on your Documentation Library CD-ROM.

Oracle Advanced Security

The Oracle Advanced Security option provides a comprehensive suite of security services for Oracle8*i* Cache, Oracle8*i* JVM, and Oracle8*i* PLSQL. Its functionality is twofold. First, network security features protect enterprise networks and securely extend corporate networks to the Internet. Second, it integrates security and directory services, combining to provide enterprise user management and single signon.

Network Security

- **Data Privacy**

  The Oracle Advanced Security option ensures that data is not disclosed during transmission using the following encryption types:

  – RSA Encryption: Oracle Advanced Security provides RSA RC4 with 128-bit, 56-bit, and 40-bit keys.

  – DES Encryption: Oracle Advanced Security offers a standard, optimized 56-bit key DES (Data Encryption Standard) encryption algorithm and supports DES40 for backwards compatibility.

- **Data Integrity**

  Oracle Advanced Security makes it virtually impossible for an intruder to modify, delete, or replay packets without detection. During transmission, Oracle Advanced Security generates a cryptographically secure message digest through cryptographic checksums using the MD5 algorithm. Then, the secure message digest is included with each packet sent across the network.

- **Authentication**

  Oracle Advanced Security provides strong authentication of Oracle users support for third-party authentication services. The following authentication methods are supported:

  - SSL with X.509v3 certificates
  - Kerberos and CyberSafe
  - Token Cards (SecurID- or RADIUS-compliant)
  - DCE (Distributed Computing Environment)
  - RADIUS
  - Smart Cards (RADIUS-compliant)
  - Biometrics (Identix- or RADIUS-compliant)

- **Single Signon**

  Oracle Advanced Security provides single signon where the user authenticates once, then strong authentication occurs transparently in subsequent connections to other databases or services. Using single signon, users can access multiple accounts and applications with a single password. Oracle Advanced Security supports many forms of single signon, including Kerberos and CyberSafe, as well as SSL-based single signon.

- **Authorization**

  Once users are authenticated, they are then authorized to access only those services permitted by a corporate or business policy as found in a policy repository. Authorizations are provided with some of the third-party authentication solutions, such as DCE (Distributed Computing Environment), as well as with the enterprise user security functionality in Oracle Advanced Security.

Enterprise User Security

The Oracle Advanced Security option integrates with LDAP v3-compliant (Lightweight Directory Access Protocol) directory services, such as Oracle Internet Directory, for enterprise user management, enterprise role management, and single signon.

- **Single Signon**

  Oracle Advanced Security provides SSL-based (Secure Sockets Layer) sing signon for Oracle users by virtue of integration with LDAP v3-compliant directory services. Integrated security and directory services and Oracle PKI (Public Key Infrastructure) implementation in Oracle Advanced Security enable SSL-based single signon to Oracle8*i* databases. Single signon enables users to authenticate once at the initial connection and subsequent connections authenticate the user transparently based on his or her X.509 certificate. This brings ease of use to the users and single station administration for the administrator with centralized management of users and authorizations.

- **Enterprise User Management**

  Oracle Advanced Security provides enterprise user management, allowing administrators to centrally manage users on a central directory service, rather than repeatedly managing the same users on individual databases. Using Oracle Enterprise Security Manager, a tool accessible through Oracle Enterprise Manager, enterprise users and their authorizations are managed in Oracle Internet Directory or other LDAP v3-compliant directory services. Enterprise

users can be assigned enterprise roles that determine their access privileges in a database, and enterprise roles can be granted to one or more enterprise users.

- **Schema-Independent Users**

  Oracle Advanced Security allows the separation of users from schemas so that many enterprise users can access a single, shared application schema. Instead of creating a user account in each database that a user needs to access, administrators only need to create an enterprise user in the directory and point the user at a shared schema, which many other enterprise users can also access. This allows administrators to create an enterprise user once in the directory. Then that enterprise user can access multiple databases using only the privileges he or she needs, thus lowering the overhead of managing users in an enterprise.

- **PKI Credential Management**

  Oracle Wallet Manager provides secure management of PKI user credentials. It issues certificate requests to Certificate Authorities (CA), manages the X.509 certificates and trusted certificates, and creates a private and public key pair for users. In most cases, a user never needs to access a wallet once it has been configured, but can easily access a wallet using Oracle Enterprise Login Assistant, a login tool that hides the complexity of a private key and certificates from users. Users can then connect to multiple services over SSL without providing additional passwords. This provides the benefit of strong, certificate-based authentication as well as single signon.

- **Directory Integration**

  An Oracle Advanced Security license provides the use of Oracle Internet Directory to store and manage users and their authorizations. It supports enterprise user management with Oracle Internet Directory, which is fully integrated with Oracle8*i*. Additionally, Oracle Advanced Security supports other leading LDAP-compliant directories.

For more information, see the *Oracle Advanced Security Administrator's Guide* in the Oracle8*i* documentation set.

For information about using Oracle Internet Application Server services, see Chapter 3, "Developing Applications for Oracle Internet Application Server".

3

Developing Applications for Oracle Internet Application Server

Oracle Internet Application Server provides several options for developing and deploying applications using various programming languages and communication protocols. This chapter demonstrates how you can use Oracle Internet Application Server to build your applications.

Contents

- Content Publishing
- Business Logic

Content Publishing

Static Content

The simplest Web sites consist of static pages where the result of a client request returns one or more HTML pages with content that does not change after its initial display. Typically, a simple Web site identifies an HTTP request, responds by sending the requested content to the client, and returns a resulting response to the Oracle HTTP Server *powered by Apache*.

For information about the Oracle HTTP Server *powered by Apache*, refer to the Apache HTTP Server section of the Oracle Internet Application Server Documentation Library.

Dynamic Content

Usually, the most interesting content requires more complex functionality, which typically consists of dynamic content delivered from the server. Server-side components run on the server and generate output which is then sent to the client browser. Oracle Internet Application Server supports a variety of technologies for developing and deploying dynamic content, including:

- Common Gateway Interface (CGI) Applications
- Perl Scripts
- XML
- Servlets
- JavaServer Pages
- PL/SQL Server Pages
- Oracle Reports Services

Common Gateway Interface (CGI) Applications

Web content developers can write CGI programs that fetch data and produce entire Web pages within the same application. These applications typically contain scripts that mix application logic with presentation logic. Application logic manipulates the data, while the presentation logic formats the content. The separation of HTML content from application logic makes script based applications, such as CGI applications, easier to develop, debug, and maintain.

There are two categories of scripting applications: client-side and server-side scripting. Client-side scripts run in the client's Web browser. Server-side scripts run on the server, fetching and manipulating data. The resulting data is embedded in the HTML page, which is then sent to the client's Web browser.

In Oracle Internet Application Server, the Oracle HTTP Server uses mod_cgi to receive requests for a Common Gateway Interface (CGI) application, the server invokes an operating system shell that runs the application and uses the CGI to deliver any accompanying data to the application. Every time the server receives a request for a CGI application, it starts a new process to run the application.

Oracle Internet Application Server fully supports CGI applications, providing the middle-tier environment to meet the demands for performance and scalability when running Web applications. However, CGI applications are most useful for processing simple forms on a small scale.

Perl Scripts

Perl is an interpreted language with powerful text processing capabilities, which makes it ideal for parsing requests from clients and generating dynamic HTML. Perl scripts contain the logic to produce the dynamic portions of Web pages that run as requested by a client Web browser.

The Perl Interpreter embedded in the Oracle HTTP Server provides a performant, internal interpreter for running Perl scripts. The Perl Interpreter receives and runs Perl scripts via mod_perl, an Oracle HTTP Server module that delegates the handling of HTTP requests to run Perl programs. Using mod_perl, the interpreter links directly with the Oracle HTTP Server demons eliminating outside network communication and reducing access time.

Upon receiving a request, the interpreter loads, compiles, and caches the Perl script. The script remains in the cache as long as the server remains running, so subsequent requests for the same script do not have to recompile. If the script has been modified in any way, the cache purges the original script, then compiles and stores a fresh copy of the modified script.

For general information about Perl and the Perl module in the Oracle HTTP Server *powered by Apache*, refer to the documentation created by the Apache Software Foundation (available at http://www.apache.org/).

For information about running Perl in the Oracle Internet Application Server, refer to *Apache mod_perl Guide* in the Apache HTTP Server section of the Oracle Internet Application Server Documentation Library.

XML

XML (eXtensible Markup Language) is a flexible and standard way to create common document formats for building and deploying Web content. As with CGI scripts, XML data separates the content and structure from the presentation, making it easy to present the data in a variety of layouts and applications.

Oracle Internet Application Server allows you to generate XML on the middle tier, invoke server-side components to access the database, run applications, process the information, and deliver the content to the client Web browser. The Oracle XML Developer's Kit (XDK) provides support for reading, manipulating, transforming, and viewing XML documents for Java applications.

The Oracle XDK includes a set of XML parsers to process XML documents for Java. The parser's job is to verify that the XML is well-formed and to validate the XML against an existing Document Type Definition (DTD). After this verification and validation phase, the parser manipulates the XML documents into a useful format for applications. Oracle also provides several products that support automatic generation of classes from DTD elements and ways to programmatically use class methods to construct XML documents.

XML also gives you a common format for transferring data between databases. You can generate XML from multiple databases to a common, extensible XML document type. You can use XML to give context to words and values, identifying elements as data. For example, you can identify elements for data such as names, addresses, and contact information. When you collect data from several sources using these common elements, you can easily integrate the data into a common format for a specific application.

For information about using XML in the Oracle Internet Application Server, refer to the *XML Developer's Kit* in the Oracle Internet Application Server Documentation Library.

Servlets

Servlets are Java applications that run in a server-side environment and service HTTP requests from client browsers. While servicing client requests, servlets can use common Java technologies to increase their functionality. For example, a servlet using JDBC can connect to a database and execute a query. The servlet can then format the result of the query in an HTML table and return it to the client.

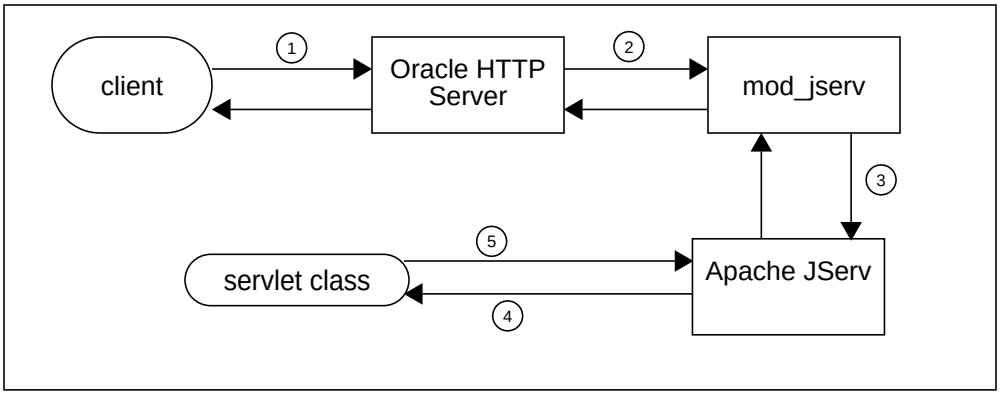
Advantages of Using Servlets The ability to use existing Java code and standard APIs makes servlets a powerful tool for servicing requests. Other advantages of using servlets are:

- Servlets, as a Java-based technology, are platform and Oracle HTTP Server independent. A servlet that is compliant with Sun's servlet specification can run on any Oracle Internet Application Server node without any modifications.
- Servlets often perform better than CGI applications because requests can reuse an existing Java Virtual Machine (JVM). This saves system resources that can, otherwise, be used to start-up and shut-down the JVM for each request.
- Database connections can remain open for a session reducing the overhead of reconnecting to the database with each request.
- Free servlet instances are reused to service new requests instead of creating new instances.

Developing Servlets Developing servlets requires a compiler, a debugger, and access to the standard and servlet Java libraries. These libraries are installed with Oracle Internet Application Server. You can also use integrated development environments, such as Oracle JDeveloper, which provide a set of tools that help in developing and deploying servlets.

Processing Servlet Requests In Oracle Internet Application Server, servlets run in the Oracle HTTP Server component. The mod_jserv module forwards requests from the Oracle HTTP Server to Apache JServ. Apache JServ is the Java servlet engine designed to service servlet requests from the Oracle HTTP Server. The JVM processes the servlet and returns the output to mod_jserv. The mod_jserv module now returns the response to the client. This process is illustrated in Figure 3–1.

Figure 3–1 Control Flow for a Servlet in the Oracle Internet Application Server



1. The Oracle HTTP Server receives a request for a servlet from a client.
2. The Oracle HTTP Server dispatches the request for a servlet class to the mod_jserv module.
3. The mod_jserv module forwards the request to the servlet engine. If the engine is not already running, mod_jserv will spawn one and initialize the servlet.
4. The servlet engine translates the incoming request into a request object. The engine then creates a response object and passes both objects to the servlet class.
5. The servlet class executes and generates response data. This data is passed to the servlet engine in the servlet response object. The response then travels back to the client through the mod_jserv module and Oracle HTTP Server.

For general information about Java servlets, refer to the Java Servlet API Specification 2.0 (available from http://java.sun.com).

For information about running servlets in Oracle Internet Application Server, refer to *Apache JServ Guide* in the Oracle Internet Application Server Documentation Library.

JavaServer Pages

JavaServer Pages (JSP), as specified by Sun Microsystems, offer an easy to use and convenient method for developers to add dynamic content to an HTML-based file with Java code and some special tags. The ability to use existing Java code and standard APIs make JSP a powerful tool for generating dynamic HTML pages. This allows Java developers to simplify their applications because presentation logic can be handled by the JSP. To use a new presentation method, such as a new HTML tag, you only change your JSP. The underlying Java code will not need any modifications since it only delivers the data to the JSP.

Advantages of Using JSP Since JavaServer pages are platform independent, any JSP that is compliant with Sun's JSP specification can run on any Oracle Internet Application Server node without modification.

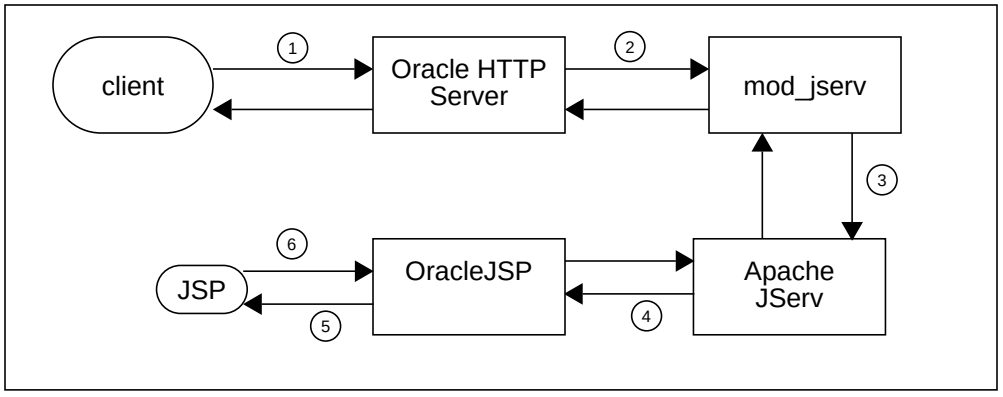
OracleJSP, Oracle's translator and runtime engine for JavaServer Pages, supports the following enhancements available to JSP developers:

- More data types to increase the functionality of embedded logic.
- Two options, primitive types (PT) or java.lang objects. Only objects can be stored in the scope object, therefore, values of PT cannot have a valid scope and cannot be stored in a JSP scope object. Wrapper classes in the java.lang package are objects that cannot be declared in a JSP usebean statement because the wrapper classes do not follow the Javabean model and do not provide a null constructor.
- JSP Markup Language (JML) offers an alternative to the JSP scripting syntax. This supports non-Java developers who want to create dynamic Web pages.
- XML and XSL support developers who want to create XML pages instead of HTML pages.

Deploying JSP Applications Since the servlet engine compiles pages on request, you must test your pages by requesting them through the Oracle HTTP Server in Oracle Internet Application Server. Some integrated development environments, such as Oracle JDeveloper, provide built-in debugging environments for JSP.

Processing JSP Requests In Oracle Internet Application Server, JavaServer Pages run in the Oracle HTTP Server component. The Oracle HTTP Server forwards the request through the servlet environment (mod_jserv and the Apache JServ servlet engine) to OracleJSP. The translator processes the JavaServer Pages and compiles any embedded code. The output returns along the originating path to the client. This process is illustrated in Figure 3–2.

Figure 3–2 Control Flow for a JavaServer Page in the Application Server



1. The Oracle HTTP Server receives a request for a JSP from a client.
2. The Oracle HTTP Server dispatches the request for a servlet class to the mod_jserv module.
3. The mod_jserv module forwards the request to the Apache JServ servlet engine.
4. The servlet engine translates the incoming request into a request object. The servlet engine then creates a response object and passes both objects to OracleJSP.
5. The OracleJSP parses the JSP file and executes the embedded application logic.
6. The result is passed from the OracleJSP to the Apache JServ servlet engine in the servlet response object. The response then travels back to the client through the mod_jserv module and Oracle HTTP Server.

For general information about JavaServer Pages, refer to the JavaServer Pages Specification 1.0 (available from http://java.sun.com).

For information about developing JavaServer Pages in Oracle Internet Application Server, refer to *Oracle Translator for JSP* in the Documentation Library.

PL/SQL Server Pages

Oracle PL/SQL Server Pages (PSP) is Oracle's PL/SQL dynamic server-side scripting solution for Web application development. Oracle PSP includes the PL/SQL Server Pages Compiler and the PL/SQL Web Toolkit. Oracle PSP enables PL/SQL users to develop Web pages with dynamic content by embedding PL/SQL scripts in HTML. PSPs separate application logic (embedded PL/SQL scripts) from the layout logic (HTML) making the development and maintenance of PL/SQL Server Pages easy. Using this method, content developers can design the static portions of Web pages in HTML, then add scripts that generate the dynamic portions of the pages. In addition, PSP uses PL/SQL as the scripting language and tightly integrates with the database so it is easy to retrieve, manipulate, and present data.

Advantages of Using PSP PL/SQL Server Pages provide server-side scripting with traditional database operations and programming logic for developing Web-based applications. The advantages of using PL/SQL Server Pages include the following:

- PSPs support HTML and HTML authoring tools with added dynamic content or applications. You can start with an existing Web page or with an existing stored procedure to create dynamic Web pages that perform database operations and display the results.
- Typically, a PL/SQL Server Pages display in a Web browser. By default, Oracle8*i* PL/SQL transmits files as HTML documents, so that the browser formats them according to the HTML tags. Files can also be retrieved and interpreted by a program that can make HTTP requests, such as a Java or Perl application.
- The PL/SQL Web Toolkit allows you to build PL/SQL code that produces formatted output. You can specify the output format as XML, some other MIME type, or plain text, depending on the technology supported in the user's browser.
- You can run insert, update, and delete operations within PL/SQL Server Pages. As with any program that is expected to return results as HTML, such pages should include some output to confirm that the operation is successful or to show the updated state.
- You can share procedures, constants, and types across different PL/SQL server pages, and compile them into a separate package in the database.
- To handle database errors that occur when the script runs, you can include PL/SQL exception-handling code within a PSP file, which may launch another PSP to handle or to report the error.

Developing PSP Applications Developers can use the Oracle PL/SQL Web Toolkit to develop their PSP applications. The PL/SQL Web Toolkit contains a set of packages you can use in stored procedures to retrieve request information, construct HTML tags, and return header information to the client.

PL/SQL Server Pages compile to PL/SQL stored procedures. Compiling an HTML file as a PL/SQL Server Page produces a stored procedure that outputs the exact same HTML file. The PSP is basically an HTML file mixed with PL/SQL procedures combining all the content and formatting of your Web page. The HTML file contains text and tags interspersed with PSP directives, declarations, and scripts.

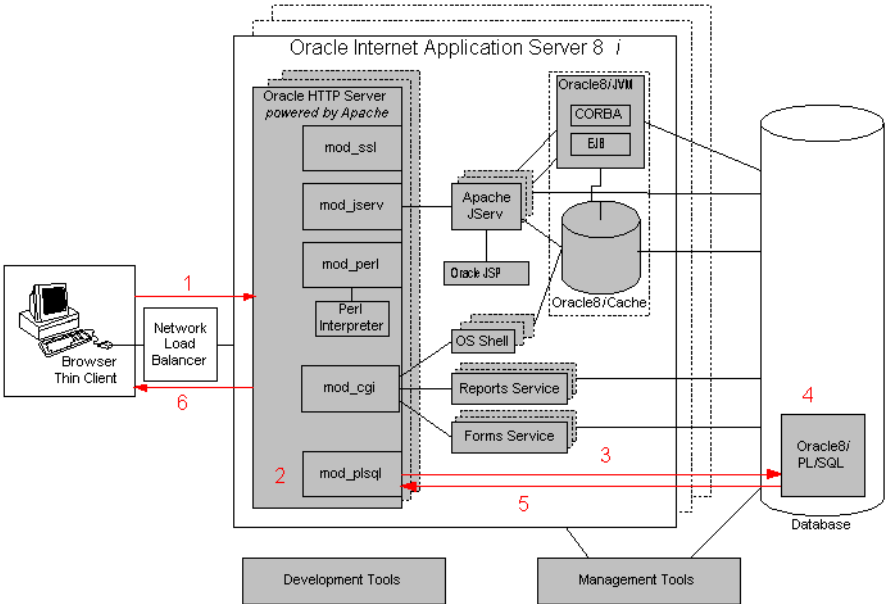
Deploying PSP Applications Oracle8*i* PL/SQL and mod_plsql in Oracle Internet Application Server provides support for deployment and performance of your PL/SQL Server Pages. By deploying PSPs on the middle-tier, the server centrally manages the application logic saving in administration and maintenance costs, and optimizing performance of your PL/SQL applications. Oracle HTTP Server forwards PL/SQL request(s) to the PL/SQL engine with the plug-in mod_plsql. Applications invoke PL/SQL scripts and stored procedures to retrieve data from a database, then generate HTML pages that return the data to the client browser.

Once the PSP has been compiled into a stored procedure, you can run it by retrieving an HTTP URL through a Web browser. The virtual path in the URL depends on the way that mod_plsql is configured.

The POST and GET methods in the HTTP protocol to tell browsers how to pass parameter data to the applications. The POST method passes the parameters directly from an HTML form and are not visible in the URL. The GET method passes the parameters in the query string of the URL. You can use the GET method to call a PSP from an HTML form, or you can use a hardcoded HTML link to call the stored procedure with a given set of parameters.

Processing PSP Requests The process of handling a PL/SQL Server Page is illustrated in Figure 3–3.

Figure 3–3 Control Flow for a PSP in the Oracle Internet Application Server



1. The Oracle HTTP Server receives a PL/SQL Server Page request from a client browser.
2. The Oracle HTTP Server routes the request to mod_plsql.
3. The request is forwarded by mod_plsql to Oracle8*i* PL/SQL. Using the configuration information stored in your Database Access Descriptor (DAD), mod_plsql connects to the database, prepares the call parameters, and invokes the PL/SQL procedure in the database.
4. The PL/SQL procedure generates an HTML page using data and stored procedures accessed from the database.
5. The response is returned to mod_plsql.
6. The Oracle HTTP Server sends the response to the client browser.

For information about deploying PL/SQL Server Pages in the Oracle Internet Application Server, refer to *Using Oracle8i PL/SQL* in the Oracle Modules section of the Oracle Internet Application Server Documentation Library.

Oracle Reports Services

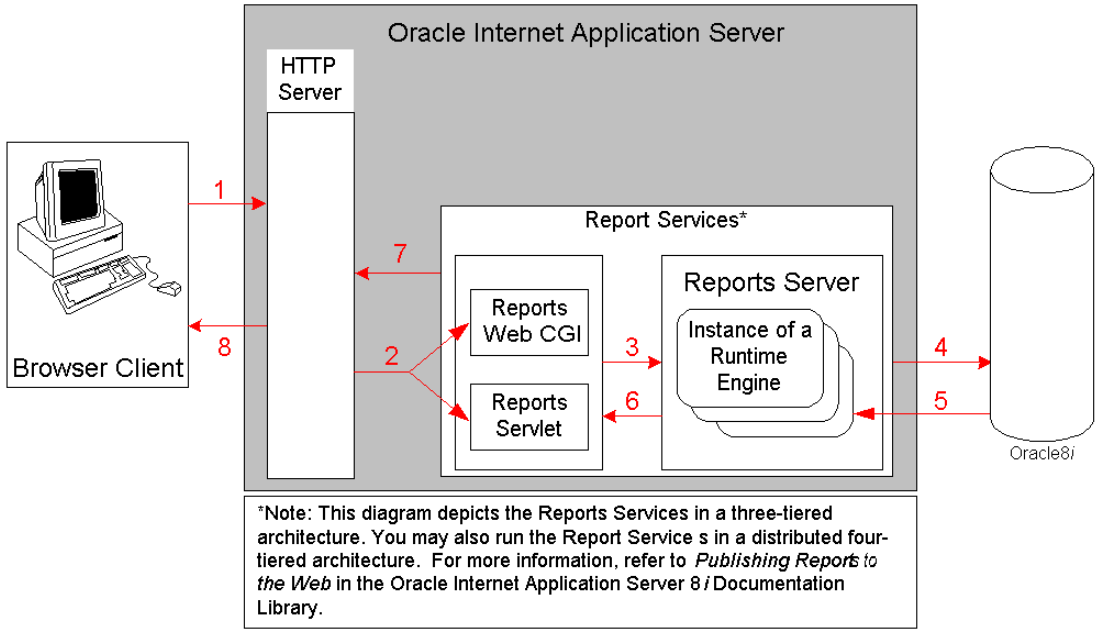
The Oracle Reports Services enable you to deploy new and existing Oracle Reports within the Oracle Internet Application Server multi-tiered architecture. All report processing, administration, and maintenance occurs on the server, which dramatically simplifies the client configuration and reduces the client storage and processing requirements. Oracle Reports Services supports all of the major industry-standard Web output formats, making it easy for users to view reports in their favorite Web browser.

Oracle Reports Services support output in the following industry standard formats:

PDF	Adobe Portable Document Format provides high quality output that users can view on the Web and easily print.
HTML and HTMLCSS	HTML without cascading style sheets provides output that can be viewed in any HTML 3.0 compliant browser. HTML with cascading style sheets provides higher fidelity output, but it requires an HTML 3.0 compliant browser that supports style sheets.
XML	You can use XML (eXtensible Markup Language) to store report definitions. By applying different XML report definitions at runtime, you can customize report formats for different users or purposes.

Processing Report Requests The Reports Services use the Oracle Internet Application Server multi-tiered architecture to publish and run report requests on the Web. Figure 3–4 shows the Reports Services residing on the same machine as the Oracle HTTP Server. Alternatively, you may distribute the Report Services across multiple machines by running the Reports Server on a separate machine. If you choose to use this distributed architecture, the Oracle HTTP Server and the Reports Web CGI or Reports Servlet components must always reside on the same machine.

Figure 3–4 Control Flow for a Report Request in the Oracle Internet Application Server



1. When a client requests a report from their Web browser, the browser passes the request to the Oracle HTTP Server on the Oracle Internet Application Server.
2. The Oracle HTTP Server invokes either the Reports Web CGI or the Reports Servlet, depending on your configuration.
3. The Reports Web CGI or the Reports Servlet parses, handles the login transaction, converts the report to command line format, and submits the request for execution by the Reports Services.

4. The Reports Server checks its output cache for an existing response to the request. If the cache has an acceptable output, the Reports Server immediately returns that output. If the cache does not have the acceptable output, the Reports Server processes the request with the database.
5. The Reports Server receives and queues the job request. When one of the runtime engines becomes available, the Reports Server sends the command line to that runtime engine for execution. The runtime engine runs the report.
6. The Reports Web CGI or Reports Servlet receives the report output from the Reports Server.
7. The Reports Web CGI or Reports Servlet sends the output to the Oracle HTTP Server.
8. The Oracle HTTP Server sends the report output to the client's Web browser.

Deploying Reports When deploying reports in a Web environment, you should consider the following:

- Choose the Report Web CGI or Reports Servlet

  You must install and configure the Reports Web CGI or Reports Servlet to handle the transmission of job requests and output between your Oracle HTTP Server and the Reports Services. Oracle Internet Application Server supports both CGI and Java based applications.

- Choose the location of the Reports Server

  You can deploy the Reports Services in Oracle Internet Application Server using a three-tiered or four-tiered architecture. When using a three-tiered architecture, the Reports Server resides on the same machine as the Oracle HTTP Server. This requires more system resources, since all processing occurs on a single machine. In a four-tiered architecture, the Reports Server resides on a machine separate from your Oracle HTTP Server, spreading the resource requirements across multiple machines. However, if you choose to have the Reports Server and the Oracle HTTP Server on different machines, then the transmissions to the Reports Web CGI and the Reports Servlet must travel across a network resulting in greater network traffic.

- Configure request handling for optimal performance

  The Reports Services handle report requests using dynamic management of runtime engines. You can specify a maximum number of runtime engines to handle your requests with optimal performance for your server. When the Reports Services receive a request to run a report, the server launches a runtime

engine to handle the request. Each runtime engine resides in memory to run additional report requests over a period of time. A runtime engine automatically removes itself from memory when it is either idle for a specified amount of time or to free up system resources. After removal, the Reports Server replaces it with a new engine.

For information about developing and publishing Reports in the Oracle Internet Application Server, refer to *Building Reports* and *Publishing Reports to the Web with Oracle Internet Application Server* in the Forms and Reports Services section of the Oracle Internet Application Server Documentation Library.

Business Logic

The Oracle Internet Application Server provides support for running applications with advanced functionality such as business logic, scalability, and performance.

- Java Servlets, Applications, and Enterprise JavaBeans
- Oracle Forms Services

Java Servlets, Applications, and Enterprise JavaBeans

Java code can be deployed for execution on either of two Java runtime environments in Oracle Internet Application Server:

- Apache JServ running on the JDK JVM
- Oracle8*i* JVM

> **Note:** Java applications running in either of these environments on the Oracle HTTP Server can communicate with Java applications in Oracle8*i* JVM.

Running Java on the JDK JVM Apache JServ supports running Servlets and JSPs. When deploying Java applications on the JDK JVM, you should consider the following:

- Conversational state
- Stateless vs. stateful applications
- JNI access

Conversational State Applications that are stateless or hold on to minimal conversational state will benefit from the responsiveness of the JDK JVM.

Stateless vs. Stateful Applications When weighing the benefits of the JDK JVM vs. Oracle8*i* JVM, we often talk in terms of stateless and stateful applications.

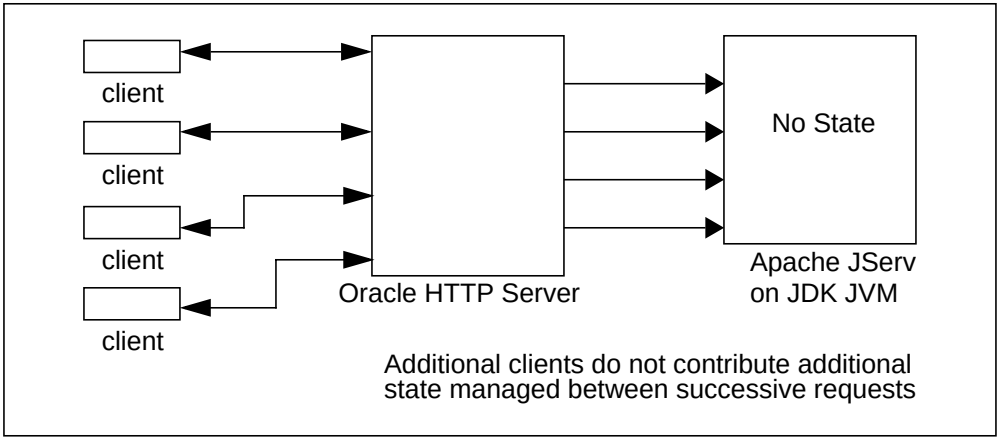
A *stateful* application maintains session state information within its runtime environment between successive client calls.

A *stateless* application maintains no state information within its environment. It may, however, persist state information in a common store such as a database or a browser.

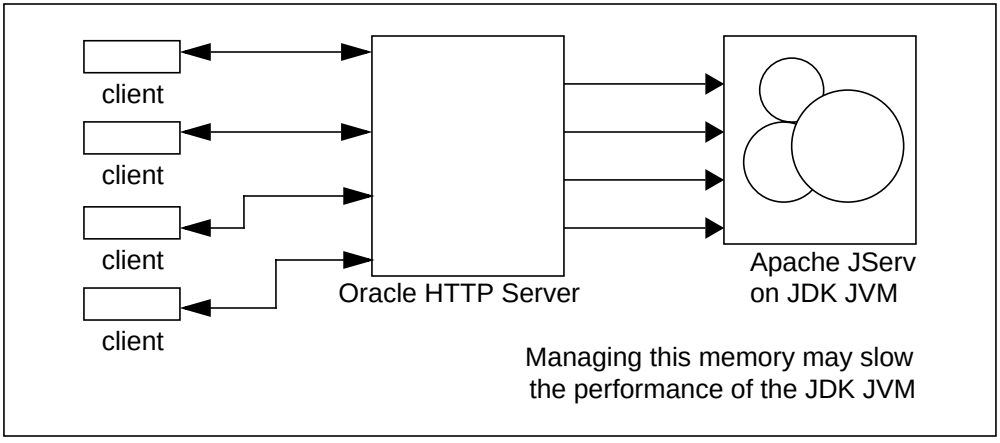
The JDK JVM scales by giving quick performance to many clients, since stateless Java applications do not need to maintain state information. However, stateful

applications force the JDK JVM to perform a lot of concurrent memory management when multiple users access the system. Managing state may inhibit the scalability of the JDK JVM.

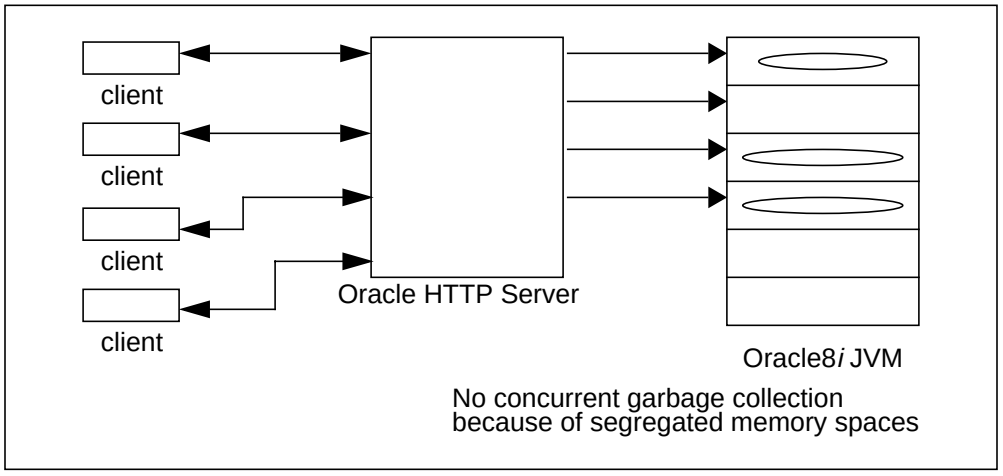
The JDK JVM scales by giving quick performance to many clients. This works well for stateless Java applications because the JVM does not get weighted down by holding onto a lot of state.



Stateful applications force the JVM to perform a lot of concurrent memory management when multiple users access the system. Managing state may inhibit the scalability of the JDK JVM.



Oracle8*i* JVM is a session-based JVM that handles stateful applications with good performance, depending on the hardware capacity. As Oracle8*i* JVM segregates clients' memory spaces, the JVM can garbage collect each user's memory space independently. This architecture avoids concurrent garbage collection, which often constitutes the major scalability bottleneck when running heavily stateful applications on a typical JVM.



JNI Access If your Java application requires access to the Java Native Interface, then you must use the JDK JVM. JNI access is not supported in Oracle8*i* JVM.

Running Java on Oracle8*i* JVM Oracle8*i* JVM support running Enterprise JavaBeans. When deploying Java applications on Oracle8*i* JVM, you should consider the following:

- Conversational state
- Benefit of fast SQL access
- Security, reliability, and availability requirements

Conversational State Applications that hold on to substantial amounts of conversational state will scale better in Oracle8*i* JVM.

Benefit of Fast SQL Access Oracle8*i* JVM runs in the same process space as the Oracle8*i* SQL engine. In the Oracle8*i* database, Oracle8*i* JVM benefits from fast data access when reading and writing to the database. In Oracle Internet Application Server, then Java applications running in Oracle8*i* JVM will benefit from very fast access when reading cached data in Oracle8*i* Cache.

Security, Reliability, and Availability Requirements Oracle8*i* JVM inherits many of the security, reliability, and availability features of the Oracle8*i* database, creating a very stable Java environment. For example, Oracle8*i* JVM isolates client sessions, so that failure in one session is not propagated to other concurrent user sessions. In many JVMs, one user session has the potential to bring down the entire JVM, affecting all concurrent JVM users. The session isolation in Oracle8*i* JVM protects sessions from one another. Even if one user's session goes down in Oracle8*i* JVM, other users' sessions are unaffected.

Oracle Forms Services

Oracle Forms Services deploy Forms applications with database access to Java clients in a Web environment. Together with Oracle Forms Developer, Oracle Forms Services provide:

- application infrastructure and the event model for scalability and performance over a network.
- support for integrating technologies such as PL/SQL, Java Stored Procedures in the Oracle8*i* database, Enterprise Java Beans, XML, and CORBA.
- extensible user interface through native Java with Pluggable Java Components.
- services for building and optimizing Oracle8*i* transactional database applications.

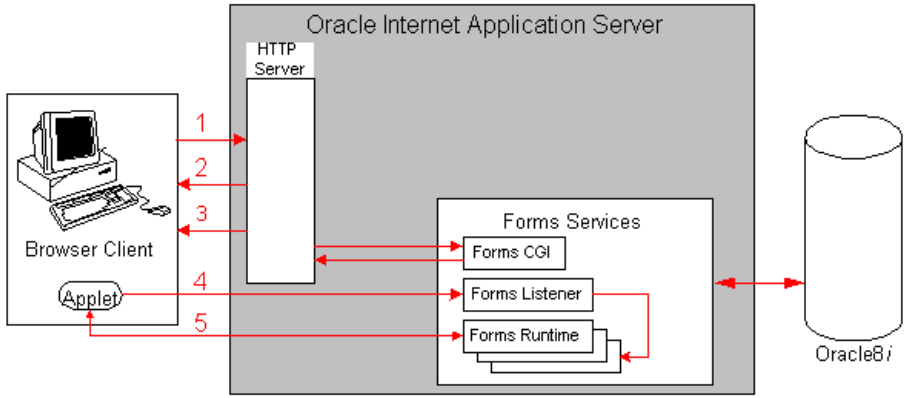
When combined with Oracle Forms Developer and Oracle Designer, you can design, develop, and deploy a complete application framework for Oracle Forms applications on the Internet. Oracle Forms Developer allows you to quickly build complex Java applications for accessing a database, without writing any Java code. The development environment provides a full toolset of wizards, widgets, and utilities, for building custom, extensible applications. Oracle Forms Developer is specifically designed and optimized to build Oracle8*i* transactional database applications, with built-in database connectivity, query, management, and transactional services. Oracle Forms Services and Oracle Forms Developer also integrate with the Oracle Designer modeling tool to provide services for full lifecycle application development and deployment.

With the integration of Forms Developer and Oracle8*i*, your Internet applications can share resources and improve application performance and scalability. Oracle Forms Services supporting this integration include:

- transaction management
- advanced queuing
- record caching
- record locking
- exception handling
- load balancing
- security management features

Processing Forms Requests The Forms Services reside as a component, in the Oracle Internet Application Server, executing forms applications. The process of handling Forms applications is illustrated in Figure 3–5.

Figure 3–5 Control Flow for a Forms Request in Oracle Internet Application Server



1. The Oracle HTTP Server receives an HTTP request from browser and contacts the Forms CGI.
2. The Forms CGI dynamically creates an HTML page containing all information to start the Forms session.
3. The Oracle HTTP Server downloads a generic Java applet to the client. The client caches the applet so the applet does not need to be downloaded again. The applet runs all future Forms applications for as long as it resides in the client's cache.
4. The client applet contacts the Forms listener to start the session. The Forms listener starts an instance of the Forms Runtime on the Forms Server. The Forms Runtime handles the user's context, executing the business logic and conducting necessary transactions with the Oracle8*i* database instance.
5. The client applet and Forms Runtime establish network communication using socket, HTTP, or HTTPS protocols.

The Forms Java client has been optimized for high performance on many platforms allowing efficient display of widgets and minimizing the time and frequency for client page refreshes. Another key element of the optimized Java client is the use of JAR file caching. Oracle Forms Services use Oracle JInitiator to preform persistent client-side caching of the applet after the initial download. Any subsequent access to the application pulls the JAR file directly from the persistent client-side cache, significantly minimizing startup time. JAR file caching is an essential performance feature for any application with remote users who dial in to access the application over a wide area network.

The Forms Services also optimize network traffic by using several methods for communicating to the Java Client including:

- Meta-data messages that reduce the amount of network traffic by using a collection of name-value pairs. These messages tell the Java Client which object to act upon and how,
- Message diffing comparing collections of name-value pairs and sending only the differences between the messages,
- Sending an initial string once and referencing the string in subsequent messages,
- Minimizing the transfer of data types by using the lowest number of bytes required for their value,
- Bundling events triggered between two objects for single packet processing.

Forms Services supports standard protocols to deploy applications including socket, HTTP, and HTTPS.

For information about developing and deploying Forms in the Oracle Internet Application Server, refer to the *Deploying Forms Applications to the Web with Oracle Internet Application Server* in the Forms and Reports Server section of the Oracle Internet Application Server Documentation Library.

Index

A

Apache JServ, 2-6, 3-8
application logic, 1-7
authentication, 2-27

B

business documents, publishing, 1-4
business intelligence, publishing, 1-4

C

Cache Manager console. *See* Oracle Enterprise Manager
cache. *See* Oracle8*i* Cache
clients, thin, 1-7
Common Gateway Interface (CGI), 3-3
content publishing, 1-3

D

database access descriptor (DAD), 3-11
database server, 1-6
Document Object Model (DOM). *See* Oracle XML Developer's Kit
Document Type Definitions (DTDs). *See* Oracle XML Developer's Kit
documentation library CD-ROM, list of contents, xii
dynamic Web pages, 1-3

E

Enterprise JavaBeans (EJBs), 1-7, 3-16

F

Forms, 3-20
Forms Developer, 3-20
Forms Services, 3-20

H

HTML
  Oracle Reports Services, 3-12
HTMLCSS
  Oracle Reports Services, 3-12
HttpSession object, 1-4

I

Internet Platform. *See* Oracle Internet Platform

J

Java
  Java Messaging Service (JMS) client, 2-16
  Java stored procedures, 1-7
  JavaServer Pages (JSP), 3-7
  Oracle SQLJ Translator, 2-19
Java Virtual Machine (JVM), 3-5
JDBC drivers
  JDBC Oracle Call Interface (JDBC-OCI) driver, 2-19
  Oracle JDBC server-side internal driver, 2-19

Oracle JDBC thin driver, 2-19
JInitiator, 3-22
Oracle Forms Services, 3-22
JNDI (Java Naming and Directory Interface). *See* Oracle LDAP Developer's Kit
jserv module, 2-4
JSP Markup Language (JML), 3-7

L

LDAP. *See* Oracle LDAP Develper's Kit

M

mod_jserv, 2-4, 3-6
JavaServer Pages (JSP), 3-8
Oracle HTTP Server powered by Apache, 3-6
servlets, 3-6
mod_perl, 2-4
mod_plsql, 2-4
mod_ssl, 2-4

O

Oracle Designer, 3-20
Oracle Enterprise Manager, 2-24
Oracle Forms Services, 2-11, 3-20
applets, 2-11
Oracle Internet Directory (O*i*D). *See* Oracle LDAP Developer's Kit
Oracle Internet Platform, 1-2
Oracle JDeveloper
Java, 3-5
servlets, 3-5
Oracle LDAP Developer's Kit, 2-21
Oracle Reports Developer applications, 2-12
Oracle Reports Services, 2-12, 3-12
Reports, 3-12
Oracle SQLJ
runtime, 2-18
SQLJ profiles, 2-18
translator, 2-17
Oracle SQLJ Translator, 2-19
Oracle XML Developer's Kit (XDK), 2-20, 3-4
XML Class Generator for Java, 2-20
XML Parser for Java, 2-20
XML Transviewer JavaBeans, 2-20
XSQL Servlet, 2-21
Oracle8*i* Cache, 2-14
application criteria, 2-14
benefits, 2-14
Oracle8*i* PL/SQL, 3-9
Oracle8*i* PLSQL, 2-11
OracleJSP, 3-7, 3-8

P

PDF
Oracle Report Services, 3-12
Perl, 3-3
Perl Interpreter, 2-8, 3-3
Perl module, 2-4
PL/SQL
stored procedures, 1-7
PL/SQL module, 2-4
PL/SQL Server Pages (PSP), 3-9
PL/SQL Web Toolkit, 3-9
PL/SQL Server Pages (PSP), 3-9
portals, 1-2
presentation services, 2-5
programming languages supported, 1-10
publishing content, 1-3
business documents, 1-4
business intelligence, 1-4

R

Reports and Graphics, 2-12
Reports Servlet, 2-12
runtime engines
Apache JServ, 3-8
Oracle8*i* PL/SQL, 3-9

S

security
Oracle Advanced Security, 2-27
servlets, 3-5, 3-16
Apache JServ, 3-8
SQLJ. *See* Oracle SQLJ

SSL module, 2-4
SSL Toolkit. *See* Oracle LDAP Developer's Kit
stateful application, defined, 1-4
stateless application, defined, 1-4
static Web pages, 1-3
stored procedures, 2-11

T

technologies supported, 1-10
thin clients, 1-7
three-tier computing model, 1-7
transactional applications, running, 1-4
two-tier computing model, 1-6

V

versions available
  enterprise edition, 1-9
  standard edition, 1-9

W

Web pages
  dynamic, 1-3
  static, 1-3

X

XDK. *See* Oracle XML Developer's Kit
XML, 3-4, 3-7, 3-12
  Oracle Reports Services, 3-12
  XML Class Generator for Java, 2-20
  XML Parser for Java, 2-20
  XML Transviewer JavaBeans, 2-20
  XSQL Servlet, 2-21