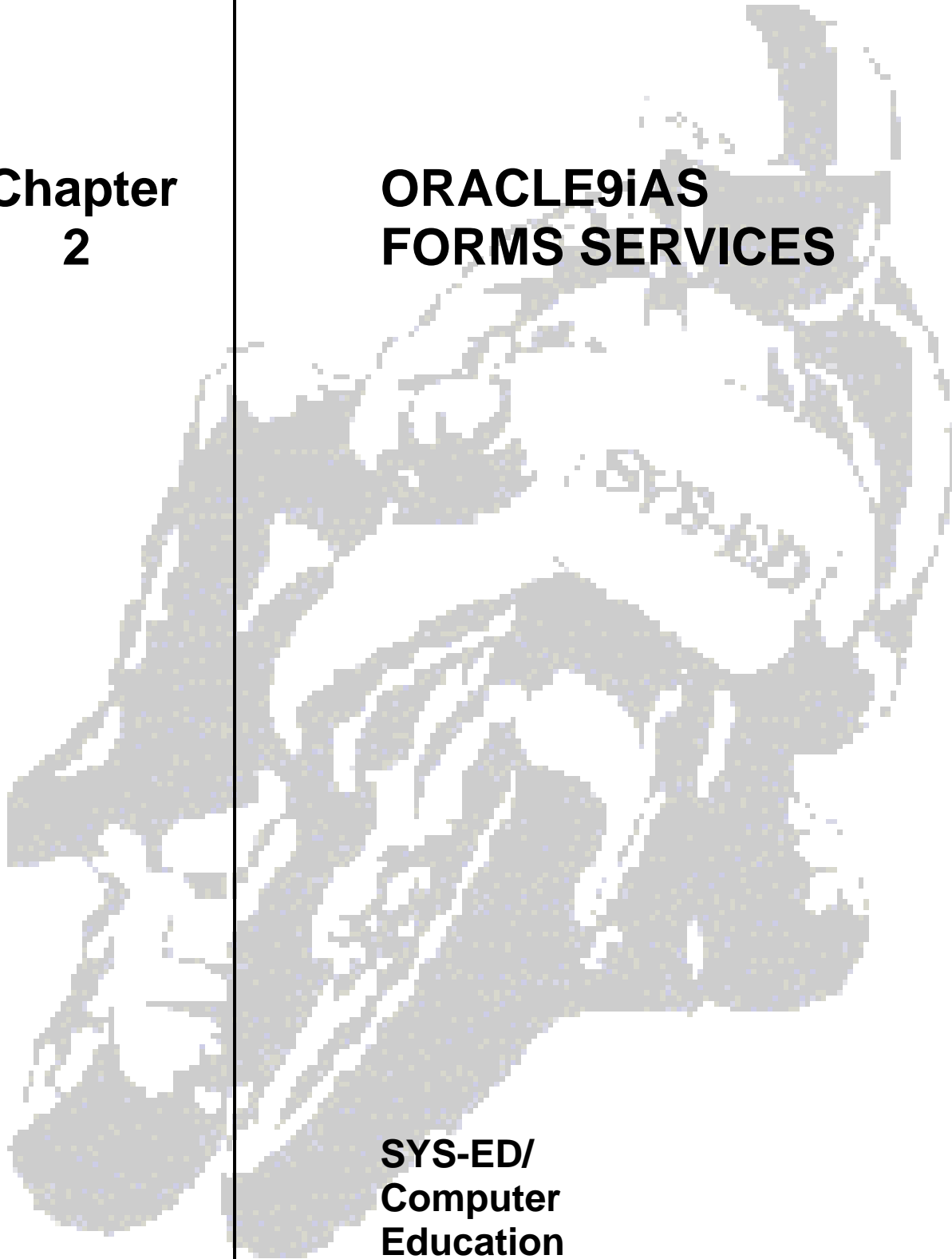


**Chapter
2**

**ORACLE9iAS
FORMS SERVICES**



**SYS-ED/
Computer
Education
Techniques, Inc.**

Objectives

You will learn:

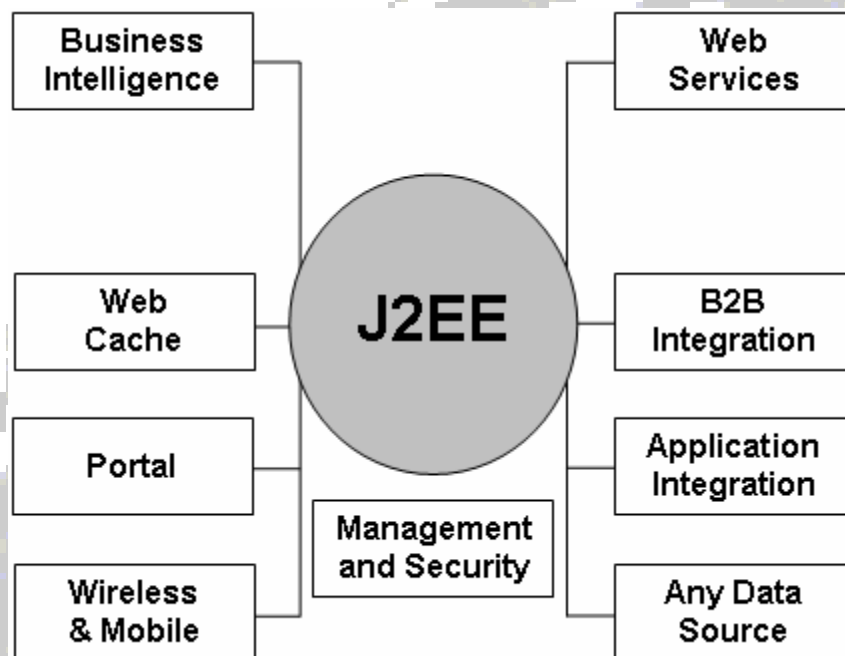
- Oracle9i Application Server.
- J2EE support.
- Web services and integration.
- Performance and scalability.
- Coding standards.
- Naming convention.
- Organizing codes.
- Coding style.
- User interface standards.

1 Oracle9i Application Server

Oracle9i Application Server, which is also known as Oracle9iAS, is an integrated J2EE-application server.

Oracle9i Application Server can be used for:

- Deploying e-business portals and transactional applications.
- Deploying Java applications in conjunction with security and reliability.
- Turbo-charging web sites and saving money on web site infrastructure with integrated web caching.
- Taking advantage of XML and web services standards for integrating a business.



1.1 J2EE Support

Oracle9i Application Server supports the J2EE standards:

- Development and deployment support for J2EE applications and web services.
- Java code execution.
- Development environment which is supported by all major Java IDEs including Oracle9i JDeveloper.

1.2 Web Services and Integration

Oracle9iAS integrates web services, B2B: business-to-business and EAI: enterprise application integration technologies.

- Web services support is merged directly into a light J2EE implementation. Applications can be deployed as web services.
- Integration links component-based Java/J2EE applications with packaged, custom and mainframe applications.
- Business processes are defined graphically and used to automate and accelerate operations. Processes are managed, monitored and optimized for continuous operational improvement.

1.3 Portal

This portal acts as the single source of interaction with corporate information and the focal point for conducting day-to-day business.

- Wizard-driven page design and development features give administrators, page designers, and users an environment to create portal pages. Pre-built "portlets" accelerate development.
- Information can be accessed from any application or data source.
- The following standards are supported: HTTP, HTML, XML, SOAP, and J2EE. Integrated support for the WebDAV standard allows business users to publish content directly from their desktop tools.

1.4 Business Intelligence

Oracle provides:

- Ad-hoc query functionality enabling end users with fast, self-service data access and enhanced analytic capabilities via client/server or over the web.
- Enterprise reporting capabilities which support data from any source.
- Clickstream intelligence for measuring web traffic and improving site effectiveness by transforming web server log data into actionable information.
- Data mining technology to enable 1:1 marketing by dynamically serving real-time, personalized recommendations to both registered customers and anonymous visitors.
- Business intelligence beans for extending and customizing applications with OLAP services.

1.5 Wireless

The wireless features of Oracle9iAS offer a platform for the development, deployment, and management of mobile applications.

- Multi-channel delivery features of Oracle9iAS enable a new class of web applications that are device independent.
- Alert mobile users of important events.
- Mobile services make wireless and voice applications easier to use.

1.6 SSO: Single Sign-On

Oracle9iAS supports single sign-on and centralized user provisioning while adhering to the Java standards.

- Single sign-on across applications, services and web sites which increases security.
- Centralized user provisioning ensures a single definition of users, roles, groups and access rights.

1.7 Reliability

Oracle9i Application Server offers advanced clustering and high availability features:

- Zero Unplanned Downtime with end-to-end clustering, transparent application failover, and the Fast Start Fault Recovery Architecture.
- Deploy applications without shutting down and re-starting the application server with hot deployment and rolling upgrade support.
- Oracle9iAS applications that access the Oracle database can be made more available by taking advantage of Oracle9i Real Application Clusters, resulting in end-to-end availability across the application server and database tiers.

1.8 Performance and Scalability

Oracle9i Application Server optimizes web site performance and e-business applications while scaling to meet increased demand.

- Performance and throughput of J2EE applications can be improved with the light and efficient J2EE server.
- By off-loading frequent web page requests to the cache, money can be saved on hardware and software infrastructure while better managing fluctuations in web site load.
- Incremental scaling can be used to support thousands of users while benefiting from the highest degree of reliability and availability with end-to-end cluster support.
- Low-cost commodity hardware can be added to clusters.

1.9 Management

Oracle9i Application Server includes system management products:

- There is a centralized web-based management framework for streamlining the management of database, application servers and deployed applications.
- Administration requires less time and fewer staff members because Oracle servers can be managed from a single location.

2 Standards

Forms Developer is a RAD tool. It can be used for creating screens which start out as prototypes and which then evolve into production systems.

Structure and standards need to be imposed on the application from the start, without which the end result of the development will be inefficient, difficult to maintain and bug ridden.

Standards can be divided into three major areas:

• Coding	• UI	• Deployment
----------	------	--------------

2.1 Coding Standards

Standards will provide the following benefits:

- Code that is simpler to maintain.
- Code that contains fewer bugs.
- Good standards can prevent many common coding bugs.
- Efficient code, both in performance and storage.
- Reusable code.

2.2 Naming Convention

PL/SQL provides flexibility in terms of naming variables and program units. Objects can be created with the same name as a different object existing at a different scope

Example:

```
DECLARE
Width NUMBER;
BEGIN
Width := to_number(
GET_WINDOW_PROPERTY( 'WINDOW1' ,WIDTH)
);
END;
```

The code compiles without error, but when run it gives the error

Argument 2 to Built-in cannot be null”.

The local declaration of the variable "width" has overridden the declaration of the Forms constant "WIDTH". These kind of errors can be very difficult to debug.

The most important factor in preventing this situation is to ensure that name collisions do not take place by enforcing naming standards.

Hungarian notation can be adapted for PL/SQL:

Datatype	Prefix (lowercase)
VARCHAR2	Vc
CHAR	C
PLS_INTEGER / BINARY_INTEGER	I
NUMBER	N
BOOLEAN	B
DATE	D
ROWID	Rw
Handle type - e.g. WINDOW, ITEM	H
Java Object (ORA_JAVA.OBJECT)	J

The variables themselves are then names with this prefix and an init-capped descriptive name, without any underscores.

3 Organizing Codes

The location of code needs to be evaluated on a system-by-system basis. A decision needs to be made whether the code should be located on the server or in the application.

3.1 Libraries

PL/SQL libraries provide a way to organize the majority of PL/SQL coding within the application.

One approach is to include no code within FMB files, except stubs to call to the real code that is all held in libraries. This will lead to module proliferation and makes maintenance harder. It will also increase memory usage when multiple users run on the same application server due to differences in the way that memory is shared between PLLs/PLXs and FMXs.

An acceptable approach would be to keep the generic codes in libraries, and code unique to each form in the respective forms.

Putting code into libraries helps in re-using the codes.

3.2 Packages

The majority of the code should reside in PL/SQL packages; the package can be in a Form or a Library.

The important advantages associated with packages are:

- They can contain persistent variables.
- Packages can be used to group areas of functionality, simplifying maintenance.
- The naming notation used when executing package functions and accessing package variables lessens the chance of name collision.
- In order to perform recursion - Program unit A calls B calls A - then this will only be possible when there are forward declarations for the program units; using a package is the way to achieve this.

3.3 Sharing Data Between Modules

An important area to concentrate in application design is how data is shared between modules at runtime. Traditionally, Forms programmers have used global variables, but this is typically expensive in terms of memory. There are also sizing and datatype translation issues.

Modern applications will use package variables in shared PL/SQL libraries, or global record groups as ways to propagate global data throughout the application.

4 Coding Style

The following areas need to be standardized:

Layout and formatting	Well formatted and indented code is simpler to read and debug. Capitalization can help with keyword emphasis.
Labeling of Loops	Explicitly labeling loops using the <<name>> notation makes both programmatic control simpler and code reading easier.
Labeling of Program	Units END statements should state the name of the Program Unit they are ending. This is important for packaged code.
Explicit handling of datatype translation	Make intentions clear by explicitly stating the implicit.

4.1 String Values

To facilitate implementation of multiple languages, and user customizable error messages by the customer, the hard coded strings should be separated within the PL/SQL code from the programming logic itself.

There are two approaches to solving this problem:

- Storing strings in database tables.
- Using the TOOL_RES package and compiled resource files (.RES) that can be produced using the RESPARSE utility

RESPARSE ships with Forms Developer. Access to strings in these files is very fast and does not incur network overhead.

4.2 Coding for Portability

In order to make the application portable to a different operating system, or to build on one operating system for deployment on another, the following issues need to be considered/addressed:

- Module Naming Windows is not case-sensitive; UNIX is. All the OPEN_FORM calls could fail if the case is wrong.
- Operating system paths on different operating systems have different formats for their Paths.
- Calls to operating system programs or services are rarely going to be portable. It is a good practice to put an abstraction layer in place.

5 User Interface Standards

The most important step in deciding upon a set of UI standards is acknowledging the platforms on which it will run.

The runtime platform influences the following issues:

Screen Size	How much area of the screen will there be and what the resolution of the clients will be running at.
Font	Different fonts are available from platform-to-platform.
Color	Will all the displays support the same color resolution? When running applications over the web, the programmer has less control over what browser the user has, or what their current screen resolution is.
Fonts	Choose common fonts.
Color palette	Choose a simple color palette of web-safe common colors.
Coordinate System	The three "Real" coordinate systems of Points, Centimeters and Inches are similar. Pixels should be avoided. Character cells should be avoided. All Forms built-ins that deal with object sizing and positioning will operate within the context of the current coordinate system. The coordinate system should be selected early.