Oracle Forms Developer 10*g*: Build Internet Applications

Instructor Guide • Volume 1

D17251GC10
Edition 1.0
June 2004
D39560

ORACLE®

Author

Pam Gamer

Technical Contributors and Reviewers

Alena Bugarova
Purjanti Chang
Laurent Dereac
Punita Handa
Mark Pare
Jasmin Robayo
Bryan Roberts
Divya Sandeep
Raza Siddiqui
John Soltani
Lex van der Werff

Editors

Nishima Sachdeva
Elizabeth Treacy

Publisher

Giri Venugopal

Contents

Preface

Profile

Before you begin this course

Before you begin this course, you should be able to:

- Create SQL statements.
- Create PL/SQL constructs, including conditional statements, loops, procedures and functions.
- Create PL/SQL stored (server) procedures, functions, and packages.
- Use a graphical user interface (GUI).
- Use a Web browser.

Prerequisites

Either

- Oracle Database 10*g*: SQL Fundamentals I
- or the following CBT Library:
  - Oracle SQL: Basic SELECT statements
  - Oracle SQL: Data Retrieval Techniques
  - Oracle SQL: DML and DDL
- or Introduction to Oracle Database 10*g* for Experienced SQL Users (InClass)
- or Oracle Database 10*g*: Introduction to SQL (InClass)

And either

- Oracle Database 10*g*: Program with PL/SQL (InClass)
- or the following CBT Library:
  - PL/SQL: Basics
  - PL/SQL: Procedures, Functions, and Packages
  - PL/SQL: Database Programming
- Or both:
  - Oracle Database 10*g*: PL/SQL Fundamentals (InClass)
  - Oracle Database 10*g*: Develop PL/SQL Program Units (InClass)

Suggested prerequisites

- Oracle Database 10*g*: SQL Fundamentals II (InClass) (if you attended the Oracle Database 10*g*: SQL Fundamentals I (InClass))
- Oracle Database 10*g*: Advanced PL/SQL (InClass)
- Oracle Forms Developer 10*g*: Move to the Web (eStudy)

How this course is organized

*Oracle Forms Developer 10*g*: Build Internet Applications* is an instructor-led course featuring lecture and hands-on exercises. Online demonstrations and written practice sessions reinforce the concepts and skills introduced.

Related Publications

Oracle publications

Title	Part Number
*Oracle Forms Developer, Release 6*i*: Getting Started (Windows 95/NT)*	A73154-01
*Oracle Forms Developer and Reports Developer, Release 6*i*: Guidelines for Building Applications*	A73073-02
Oracle Application Server Forms Services Deployment Guide 10g (9.0.4)	B10470-01

Additional publications

Release notes: `<ORACLE_HOME\doc\welcome\release_notes\chap_forms.htm`

Typographic Conventions

Typographic conventions in text

Convention	Element	Example
Bold italic	Glossary term (if there is a glossary)	The ***algorithm*** inserts the new key.
Caps and lowercase	Buttons, check boxes, triggers, windows	Click the Executable button. Select the Can't Delete Card check box. Assign a When-Validate-Item trigger to the ORDERS block. Open the Master Schedule window.
Courier new, case sensitive (default is lowercase)	Code output, directory names, filenames, passwords, pathnames, URLs, user input, usernames	Code output: `debug.set (‘I”, 300);` Directory: `bin` (DOS), `$FMHOME` (UNIX) Filename: Locate the `init.ora` file. Password: User `tiger` as your password. Pathname: Open `c:\my_docs\projects` URL: Go to `http://www.oracle.com` User input: Enter `300` Username: Log on as `scott`
Initial cap	Graphics labels (unless the term is a proper noun)	Customer address (*but* Oracle Payables)
Italic	Emphasized words and phrases, titles of books and courses, variables	Do *not* save changes to the database. For further information, see *Oracle7 Server SQL Language Reference Manual.* Enter `user_id@us.oracle.com`, where *user_id* is the name of the user.
Quotation marks	Interface elements with long names that have only initial caps; lesson and chapter titles in cross-references	Select “Include a reusable module component” and click Finish. This subject is covered in Unit II, Lesson 3, “Working with Objects.”
Uppercase	SQL column names, commands, functions, schemas, table names	Use the SELECT command to view information stored in the LAST_NAME column of the EMP table.

Typographic Conventions (continued)

Typographic conventions in text (continued)

Convention	Element	Example
Right arrow	Menu paths	Select File > Save.
Brackets	Key names	Press [Enter].
Commas	Key sequences	Press and release keys one at a time: [Alternate], [F], [D]
Plus signs	Key combinations	Press and hold these keys simultaneously: [Ctrl]+[Alt]+[Del]

Typographic conventions in code

Convention	Element	Example
Caps and lowercase	Oracle Forms triggers	`When-Validate-Item`

Lowercase	Column names, table names	`SELECT last_name`
		`FROM s_emp;`

	Passwords	`DROP USER scott`
		`IDENTIFIED BY tiger;`

	PL/SQL objects	`OG_ACTIVATE_LAYER`
		`  (OG_GET_LAYER (‘prod_pie_layer’))`

Lowercase italic	Syntax variables	`CREATE ROLE` *`role`*

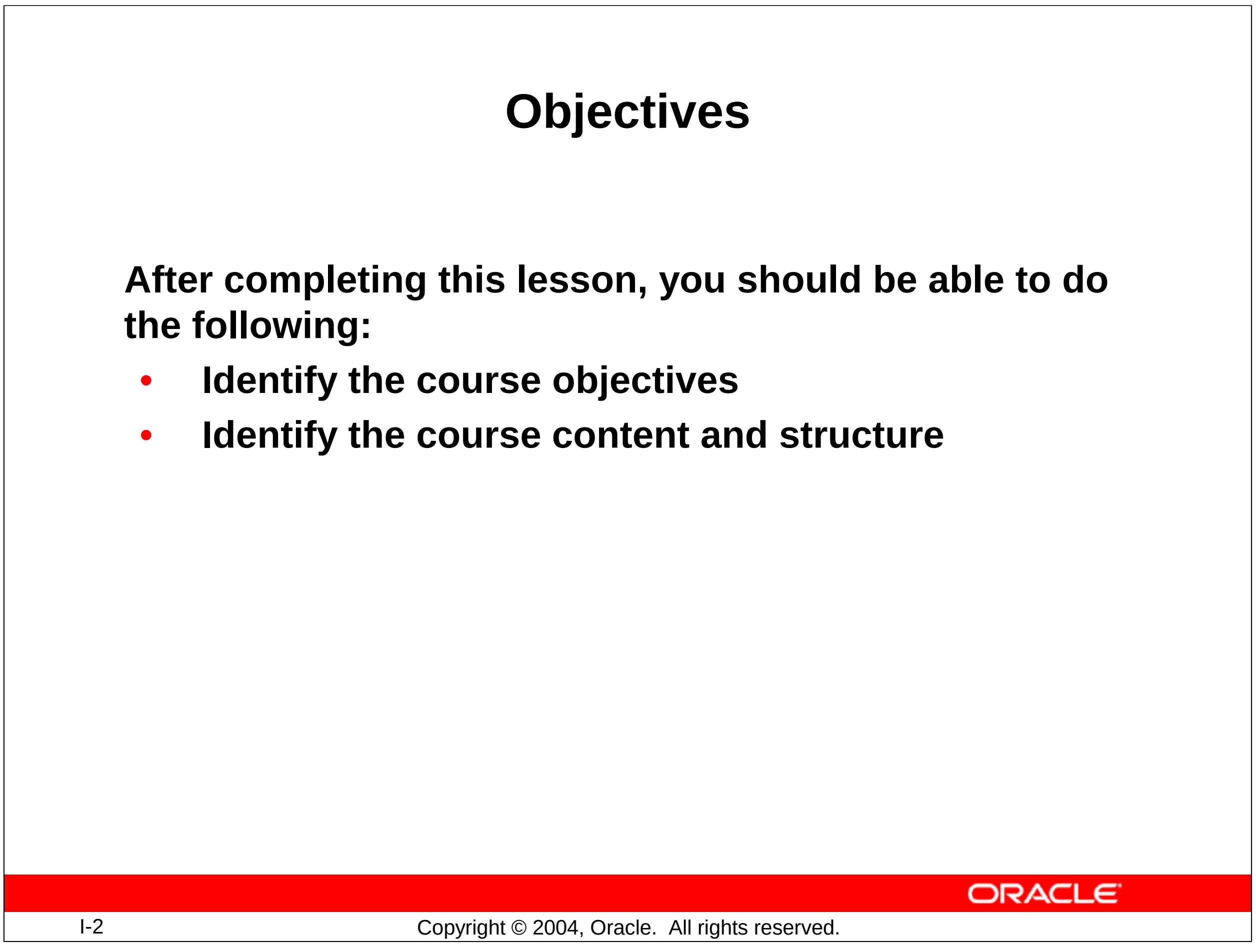
Uppercase	SQL commands and functions	`SELECT userid`
		`FROM emp;`

I

Introduction

ORACLE

Schedule:	Timing	Topic
	15 minutes	Lecture
	15 minutes	Total



Objectives

After completing this lesson, you should be able to do the following:

- **Identify the course objectives**
- **Identify the course content and structure**

ORACLE

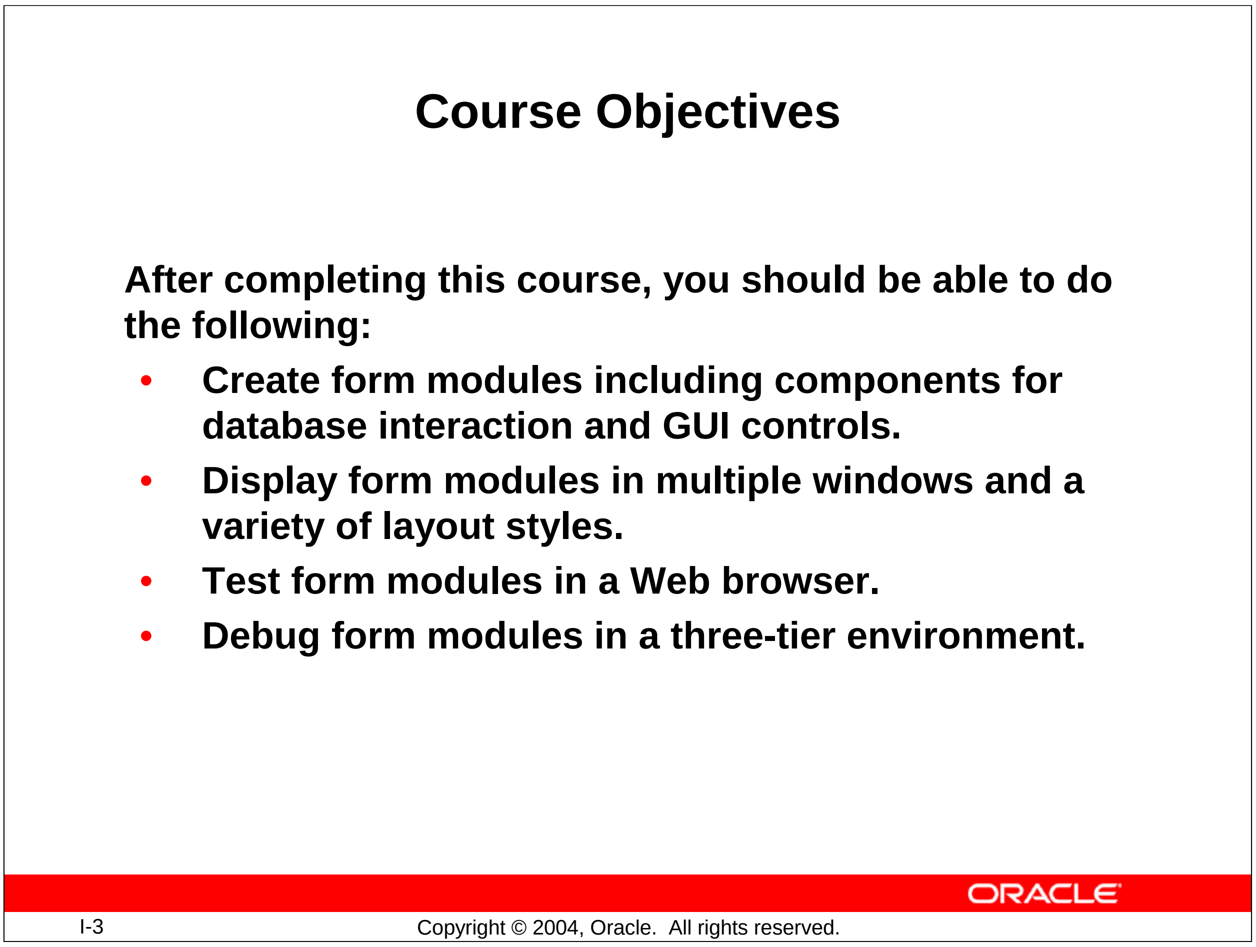
I-2

Introduction

Overview

This lesson introduces you to the *Oracle Forms Developer 10*g*: Build Internet Applications* course:

- The objectives that the course intends to meet
- The topics that it covers
- How the topics are structured over the duration of the course



Course Objectives

After completing this course, you should be able to do the following:

- Create form modules including components for database interaction and GUI controls.
- Display form modules in multiple windows and a variety of layout styles.
- Test form modules in a Web browser.
- Debug form modules in a three-tier environment.

I-3

Course Objectives

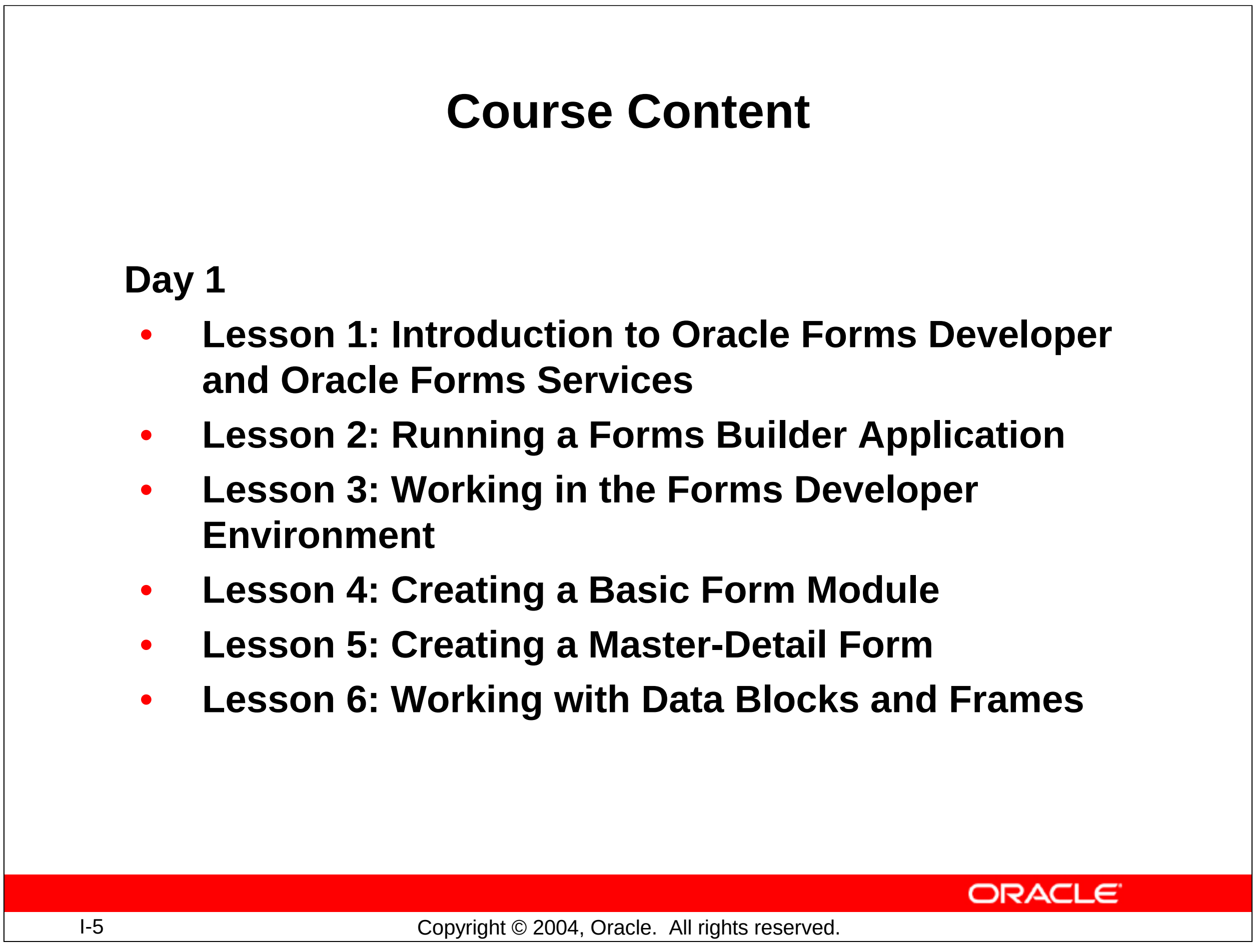
Course Description

In this course, you will learn to build, test, and deploy interactive Internet applications. Working in a graphical user interface (GUI) environment, you will learn how to create and customize forms with user input items such as check boxes, list items, and radio groups. You will also learn how to modify data access by creating event-related triggers, and you will display Forms elements and data in multiple canvases and windows.

Course Objectives

- Implement triggers to:
  - Enhance functionality
  - Communicate with users
  - Supplement validation
  - Control navigation
  - Modify default transaction processing
  - Control user interaction
- Reuse objects and code
- Link one form module to another

ORACLE



Course Content

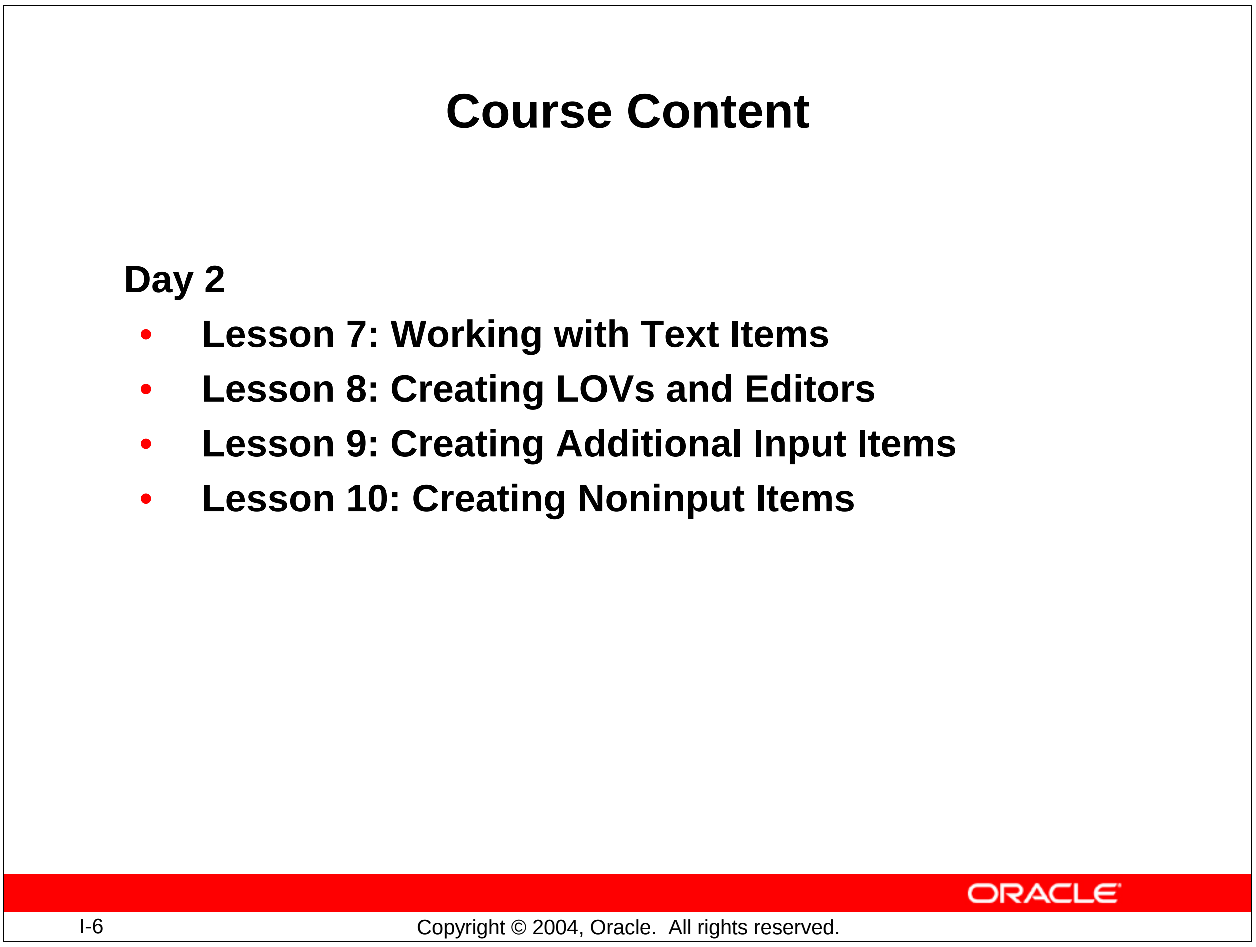
Day 1

- Lesson 1: Introduction to Oracle Forms Developer and Oracle Forms Services
- Lesson 2: Running a Forms Builder Application
- Lesson 3: Working in the Forms Developer Environment
- Lesson 4: Creating a Basic Form Module
- Lesson 5: Creating a Master-Detail Form
- Lesson 6: Working with Data Blocks and Frames

I-5

Course Content

The lesson titles show the topics that is covered in this course, and the usual sequence of lessons. However, the daily schedule is an estimate, and may vary for each class.



Course Content

Day 2

- **Lesson 7: Working with Text Items**
- **Lesson 8: Creating LOVs and Editors**
- **Lesson 9: Creating Additional Input Items**
- **Lesson 10: Creating Noninput Items**

I-6

Course Content

Day 3

- **Lesson 11: Creating Windows and Content Canvases**
- **Lesson 12: Working with Other Canvas Types**
- **Lesson 13: Introduction to Triggers**
- **Lesson 14: Producing Triggers**
- **Lesson 15: Debugging Triggers**

Course Content

Day 4

- **Lesson 16: Adding Functionality to Items**
- **Lesson 17: Run-time Messages and Alerts**
- **Lesson 18: Query Triggers**
- **Lesson 19: Validation**
- **Lesson 20: Navigation**

Course Content

Day 5

- Lesson 21: Transaction Processing
- Lesson 22: Writing Flexible Code
- Lesson 23: Sharing Objects and Code
- Lesson 24: Using WebUtil to Interact with the Client
- Lesson 25: Introducing Multiple Form Applications

I-9

1

Introduction to Oracle Forms Developer and Oracle Forms Services

ORACLE

Schedule:	**Timing**	**Topic**
	30 minutes	Lecture
	30 minutes	Total

Objectives

After completing this lesson, you should be able to do the following:

- **Define grid computing**
- **Explain how Oracle 10*g* products implement grid computing**
- **Describe the components of Oracle Application Server 10*g* and Oracle Developer Suite 10*g***
- **Describe the features and benefits of Oracle Forms Services and Oracle Forms Developer**
- **Describe the architecture of Oracle Forms Services**
- **Describe the course application**

ORACLE

1-2

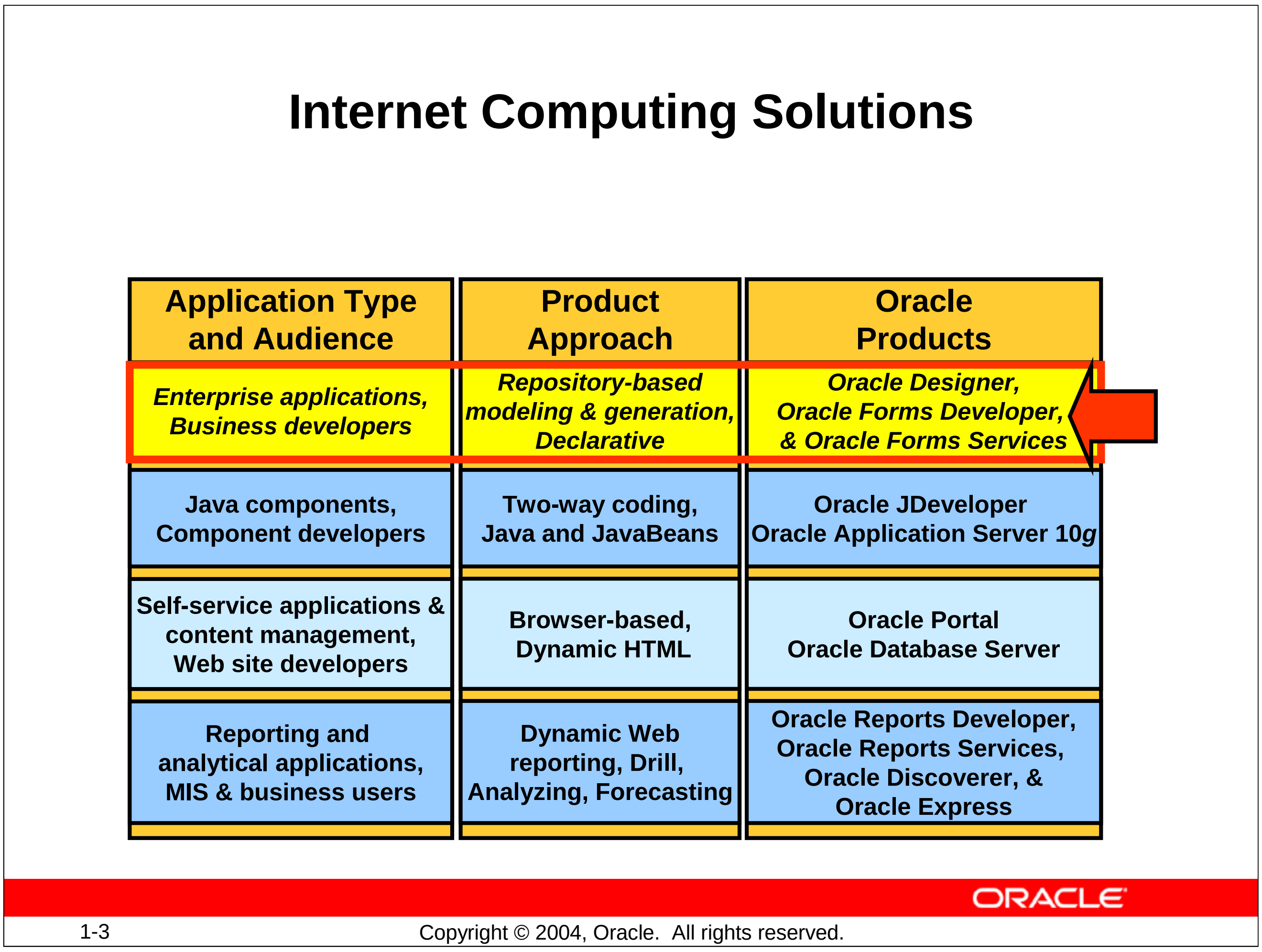
Introduction

Overview

This course teaches you how to build effective and professional form applications using Oracle Forms Developer.

This lesson identifies the key features of Oracle Application Server 10*g*, Oracle Developer Suite 10*g*, Oracle Forms Services, Oracle Forms Developer, and the course application model and contents.

There are many terms used in this course that may be unfamiliar to you. For a glossary containing definitions of many of these terms, see http://www.oracle.com/glossary.



Internet Computing Solutions

Application Type and Audience	Product Approach	Oracle Products
Enterprise applications, Business developers	*Repository-based modeling & generation, Declarative*	*Oracle Designer, Oracle Forms Developer, & Oracle Forms Services*
Java components, Component developers	Two-way coding, Java and JavaBeans	Oracle JDeveloper Oracle Application Server 10*g*
Self-service applications & content management, Web site developers	Browser-based, Dynamic HTML	Oracle Portal Oracle Database Server
Reporting and analytical applications, MIS & business users	Dynamic Web reporting, Drill, Analyzing, Forecasting	Oracle Reports Developer, Oracle Reports Services, Oracle Discoverer, & Oracle Express

ORACLE

1-3

Internet Computing Solutions

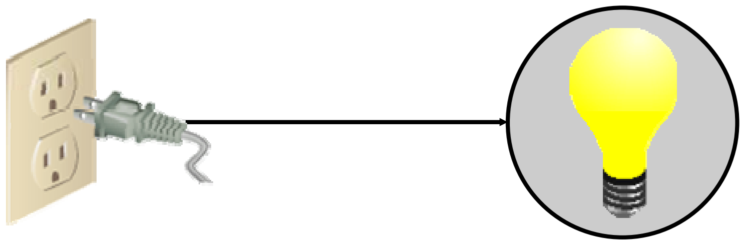
Oracle offers a range of tools and deployment options for Internet computing. Different types of developers and applications require different toolsets.

- Enterprise application developers need a declarative model-based approach. Oracle Designer and Oracle Forms Developer provide this solution. This course focuses on how you can use Oracle Forms Developer to rapidly build scalable, high-performance applications for the Internet and then deploy the applications with Oracle Forms Services.
- Component developers need different tools and methods. For these developers, Java is the language of choice. Oracle's solution is JDeveloper.
- For Web site developers and content publishers who want to build self-service dynamic Hypertext Markup Language (HTML) applications for Web sites, Oracle Portal provides an easy-to-use development environment that resides entirely inside an Oracle database. Portal provides a browser-based environment from development through deployment of an application.
- For Management Information System (MIS) developers and end users, there is the Oracle Business Intelligence toolset. Oracle Reports Developer, Oracle Reports Services, Oracle Discoverer, and Oracle Express provide the whole range for reporting, analysis, and trending facilities.

Plugging into the Grid

Grid computing is:

- Software infrastructure that uses low-cost servers and modular storage to:
  - Balance workloads
  - Provide capacity on demand
- Made possible by innovations in hardware
- Powered by software



Plugging into the Grid

When you plug in a lamp, you have no knowledge of the source of the electricity that powers the lamp. You don't know, or even care, where the generator is or how the electric grid is wired. You only know that the bulb lights up on demand.

Grid computing is a concept similar to that of electricity. Its aim is to make computing power as reliable, pervasive, and transparent as a utility.

Enterprise grid computing builds a critical software infrastructure that can run on large numbers of small networked computers at a lower cost than running on large servers. Costs of management and administration are also reduced. The workload is balanced among the machines, and capacity can be added on demand, thus providing for efficient resource sharing.

This is made possible by recent hardware innovations, such as low-cost powerful processors, blade servers, networked storage, and network interconnect technologies. But software powers the grid by creating a single logical entity, such as a database or an application server, from a cluster of machines, and by managing and administering groups of such systems as one.

Oracle Enterprise Grid Computing

Oracle's grid infrastructure products:

- **Oracle Database 10*g***
- **Oracle Application Server 10*g***
- **Oracle Enterprise Manager 10*g* Grid Control**

Oracle Enterprise Grid Computing

In recognition of the significant new capabilities required to power grid computing, Oracle has named its new technology products Oracle 10*g*, the first major name change since adding Internet capabilities to Oracle8*i*. Oracle 10*g* provides the first complete, integrated software infrastructure to power grid computing through every element of the grid—storage, databases, application servers, and applications:

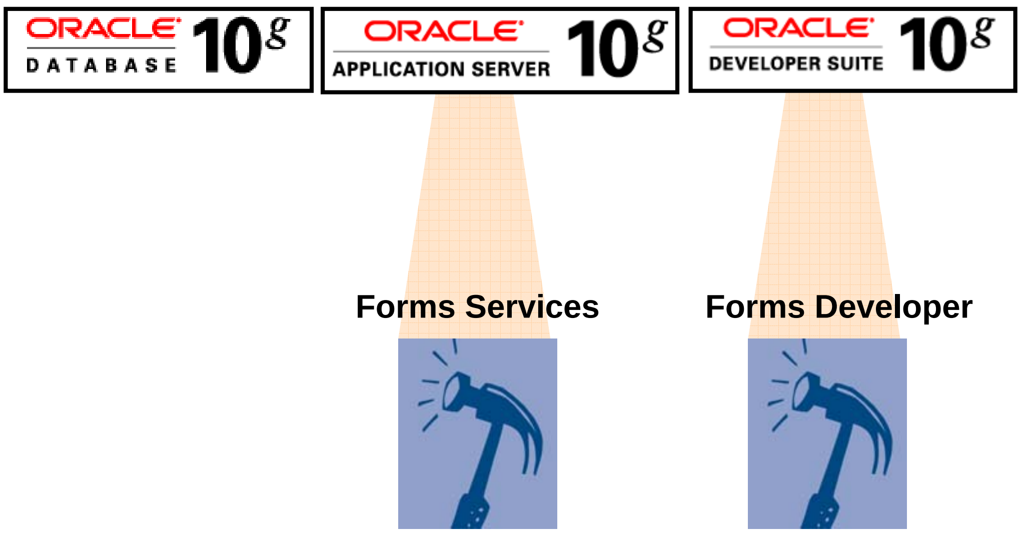
- Oracle Database 10*g* provides:
  - Real Application Clusters (RAC), which enables a single database to run across multiple clustered nodes in a grid and features Cluster Workload Management to quickly respond to fluctuations in grid workloads
  - Automatic Storage Management (ASM), which abstracts the details of managing storage to provide sophisticated data provisioning and enables DBAs to manage disk groups rather than many database files
  - Information provisioning, providing access to information when and where it is needed
  - A self-managing database, thus reducing the maintenance and tuning tasks of DBAs

Oracle Enterprise Grid Computing (continued)

- Oracle Application Server 10*g* provides:
  - Application Server Clusters that can pool and virtualize run-time services; all services can be distributed across multiple machines in a grid, and new application server instances can be automatically added and started to deliver capacity on demand
  - Interaction with Oracle RAC to improve application reliability
  - Identity management features that provide centralized user administration, which is even more important in a grid environment
  - Ease of application deployment: Enterprise applications do not need to be redesigned, because when they are deployed on the Oracle Application Server 10*g* in a grid, the applications benefit immediately from the transparent workload distribution, load balancing, and scheduling necessary to coordinate work across multiple servers. JDeveloper 10*g*, available in Oracle Developer Suite 10*g*, enables applications to expose their behavior to other applications and to management tools through standardized interfaces, so that they can communicate with other applications and heterogeneous resources across a grid.
- Oracle Enterprise Manager 10*g* Grid Control is the complete, integrated, central management console and underlying framework that automates administrative tasks across sets of systems in a grid environment. It enables:
  - Grouping of multiple hardware nodes, databases, application servers, and other targets into single logical entities
  - Software provisioning that automates installation, configuration, and cloning of Oracle Application Server 10*g* and Oracle Database 10*g* across multiple nodes, making it it possible to add capacity as needed or to easily patch and upgrade existing systems
  - Ease of application deployment: Applications can be deployed once to a single application server instance, registered with the central repository, then automatically deployed to all relevant nodes on the grid, with nodes being synchronized as changes are made.
  - Application Service Level Monitoring by viewing the availability and performance of the grid infrastructure as a unified whole so that performance or availability issues can be traced throughout the entire application, and the root cause can be determined by drilling down into the infrastructure

Note: For additional information about Oracle’s enterprise grid computing solution, see the Grid Technology Center on OTN: http://otn.oracle.com/tech/grid/index.html.

Oracle 10*g* Products and Forms Development



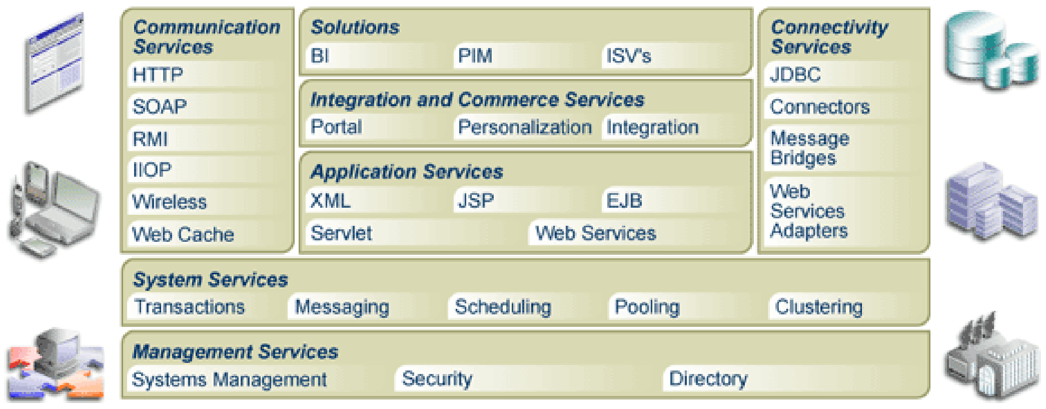
Oracle 10*g* Products and Forms Development

Oracle Database: Manages all of your information, such as Word documents, Excel spreadsheets, XML, and images. Oracle tools such as Forms can automatically reuse the database structure and its integrity constraints, which reduces the amount of manual coding.

Oracle Application Server: Runs all of your applications, including Java, wireless, portals, and business intelligence. Using Oracle Application Server, you can deploy and manage in a single application server all applications developed with Oracle Developer Suite. The Oracle Application Server contains Oracle Forms Services, which you use to deploy your Forms applications.

Oracle Developer Suite: Leverages the infrastructure offered by Oracle Application Server and Oracle Database, enabling developers to quickly and easily build scalable, secure, and reliable e-business applications. The suite provides a complete and highly productive development environment for building applications. Oracle Forms Developer, which you use to build Forms applications, is part of Oracle Developer Suite.

Oracle Application Server 10*g* Architecture



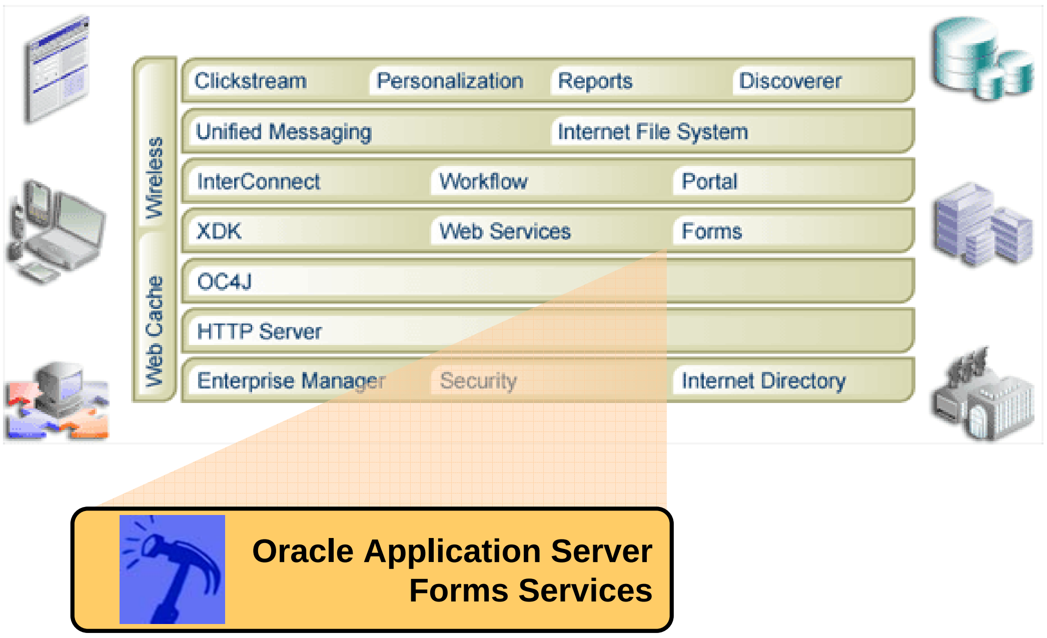
ORACLE

Oracle Application Server 10*g* Architecture

Oracle Application Server 10*g* has a layered architecture including the following services:

- **Communication Services:** Communication management for a variety of protocols
- **Application Runtime Services:** J2EE Container that provides a common runtime environment for Applications developed as JSPs, Servlets,EJBs, and Web Services
- **System Services:** A common set of runtime services that are necessary for J2EE Applications and Web Services, such as request dispatch and scheduling, resource management, resource pooling,clustering, fault monitoring, transaction management, and messaging
- **Management Services:** A common set of systems management services to monitor the status, performance and faults of the system; to monitor resource consumption and usage; to manage a single instance or cluster of instances; to centrally administer security for users and applications; and to provide a comprehensive directory service framework to manage users
- **Connectivity Services:** Provide connectivity to a variety of systems
- **Solutions:** A comprehensive set of solutions all built on the infrastructure described above including Enterprise Portals, Enterprise Integration, Business Intelligence, Wireless, and ISV Solutions

Oracle Application Server 10*g* Components



Oracle Application Server Components

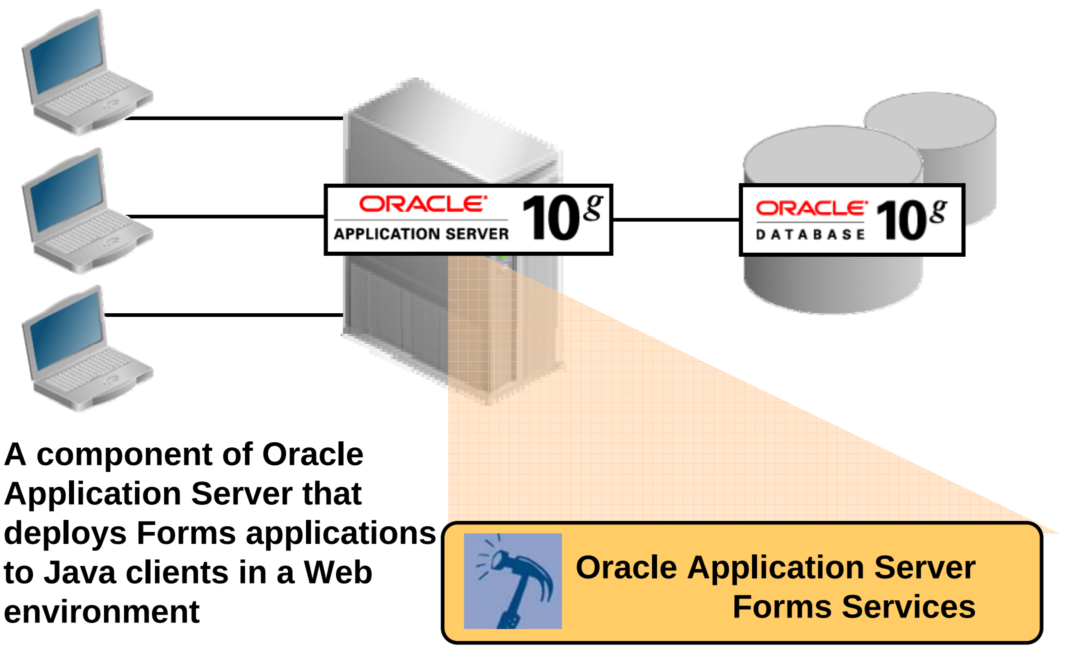
With the components of Oracle Application Server, you can:

- **Extract and analyze business intelligence:** Clickstream, Personalization, Reports Services, Discoverer
- **Integrate your business:** InterConnect, Workflow, Unified Messaging, Internet File System
- **Create personalized portals:** Oracle Portal
- **Deploy dynamic Web applications:** XDK, Web Services, **Forms Services**, OC4J, HTTP Server
- **Manage and secure your Web infrastructure:** Enterprise Manager, Security, Internet Directory.

Instructor Note

The overviews of Oracle Application Server and Oracle Developer Suite should not be presented in detail, but merely to show where Forms Services and Forms Developer fit into the product set. For further information on Oracle Application Server architecture and components, see the Oracle Application Server products page on OTN: http://otn.oracle.com/products/ias/index.html.

Oracle Forms Services Overview


ORACLE APPLICATION SERVER 10g
ORACLE DATABASE 10g
Oracle Application Server Forms Services

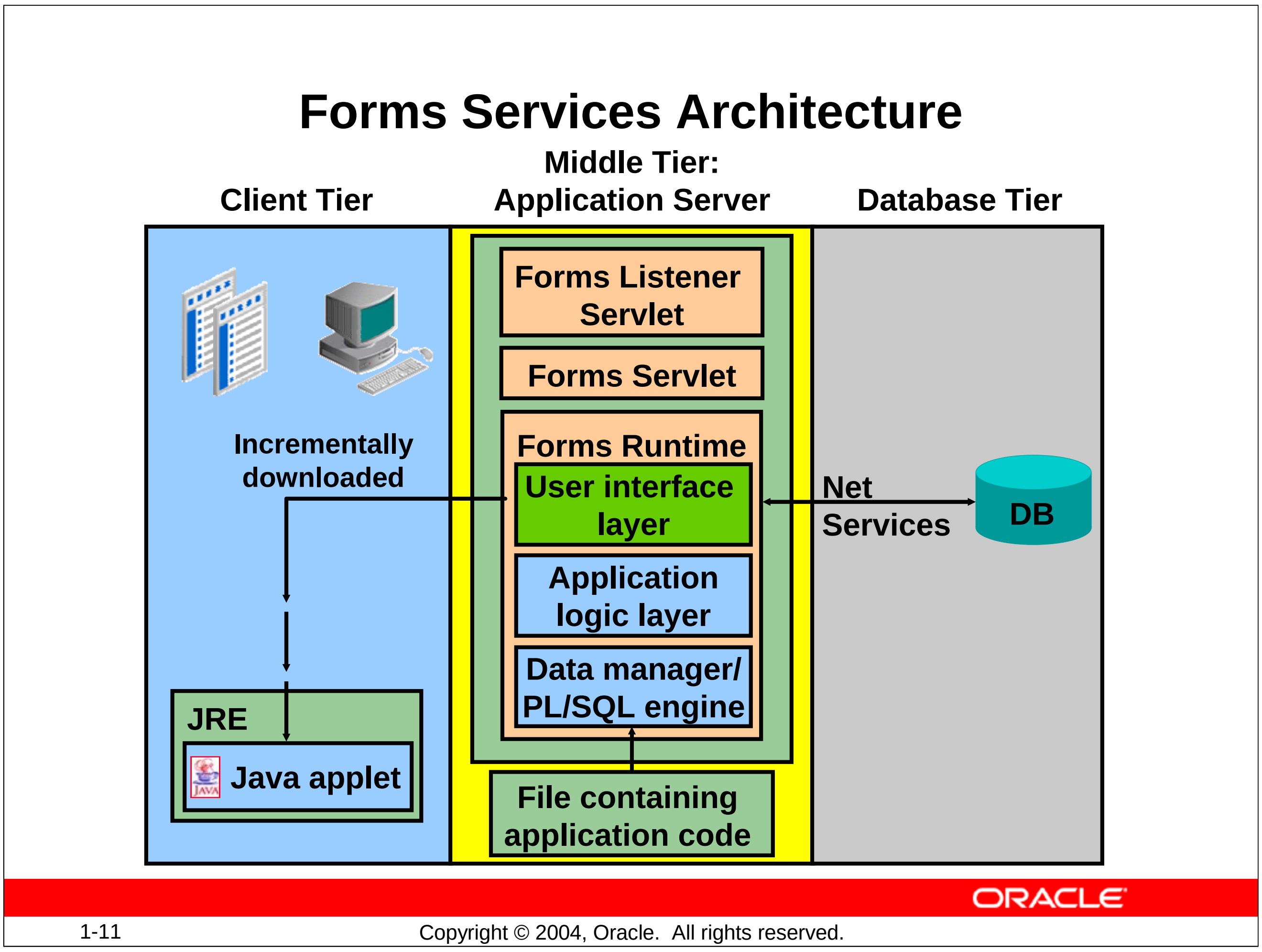
A component of Oracle Application Server that deploys Forms applications to Java clients in a Web environment

What Is Oracle Forms Services?

Oracle Forms Services is a component of Oracle Application Server for delivering Oracle Forms Developer applications to the Internet. Oracle Forms Services automatically provides the infrastructure that is needed to successfully deliver applications on the Internet through built-in services and optimizations.

Oracle Forms Services uses a three-tier architecture to deploy database applications:

- The client tier contains the Web browser, where the application is displayed and used.
- The middle tier is the application server, where the application logic and server software reside.
- The database tier is the database server, where enterprise data is stored.



Forms Services Architecture

Forms Services Architecture

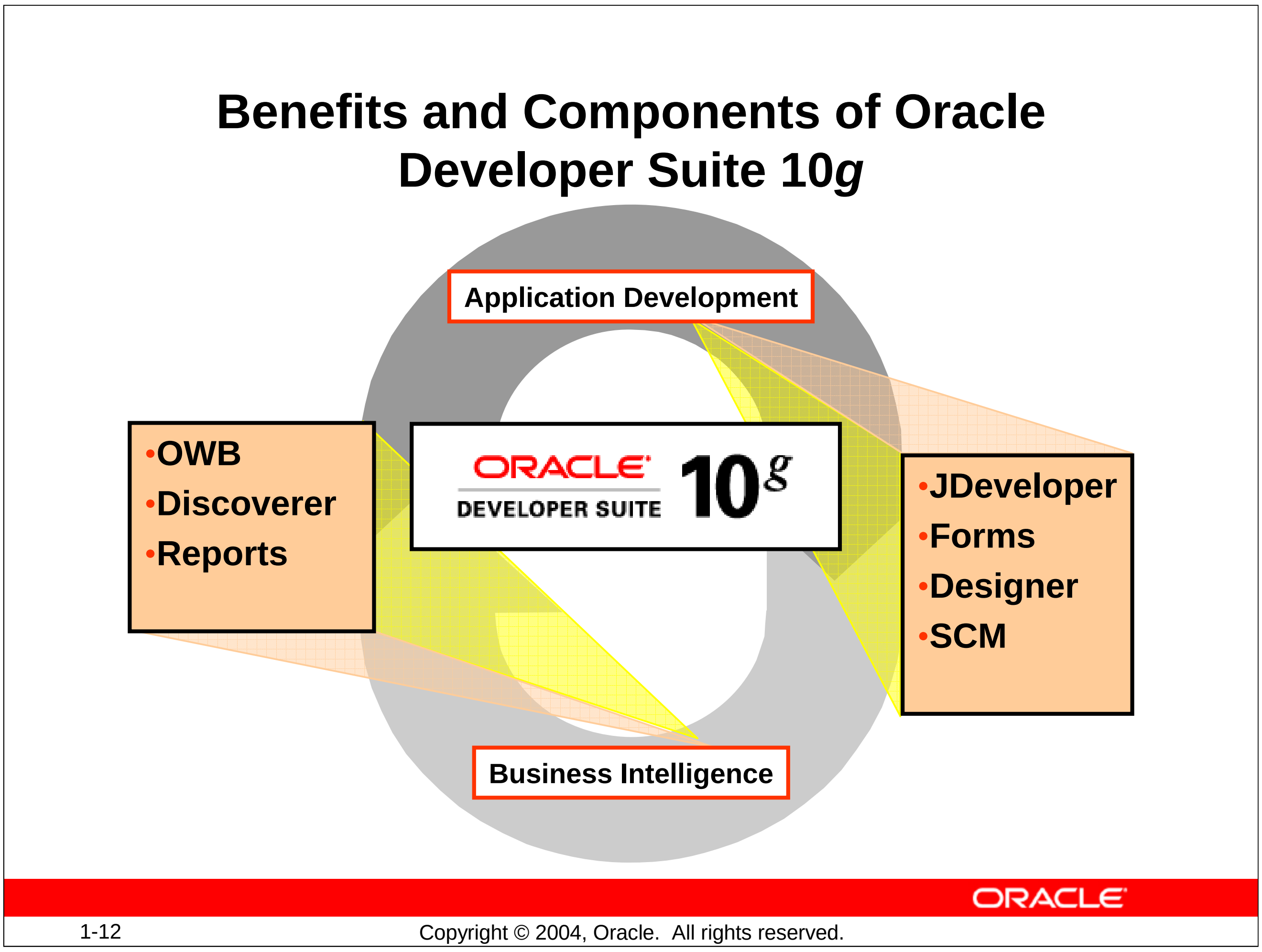
Forms Services consists of four major components: the Java client (Forms Client), the Forms Listener Servlet, the Forms Servlet, and the Forms Runtime Engine. You learn about these components in Lesson 2, Running a Forms Developer Application.

When a user runs a forms session over the Web, a thin, Java-based Forms applet is dynamically downloaded from the application server and automatically cached on the Java client machine. The same Java applet code can be used for any form, regardless of size and complexity.

Although Forms Services uses a Java applet for displaying the form on the client browser, the developer does not need to know Java in order to develop and deploy a Forms application.

Instructor Note

See *Oracle Application Server Forms 10g (9.0.4) Technical Overview* for additional information on Forms Services.

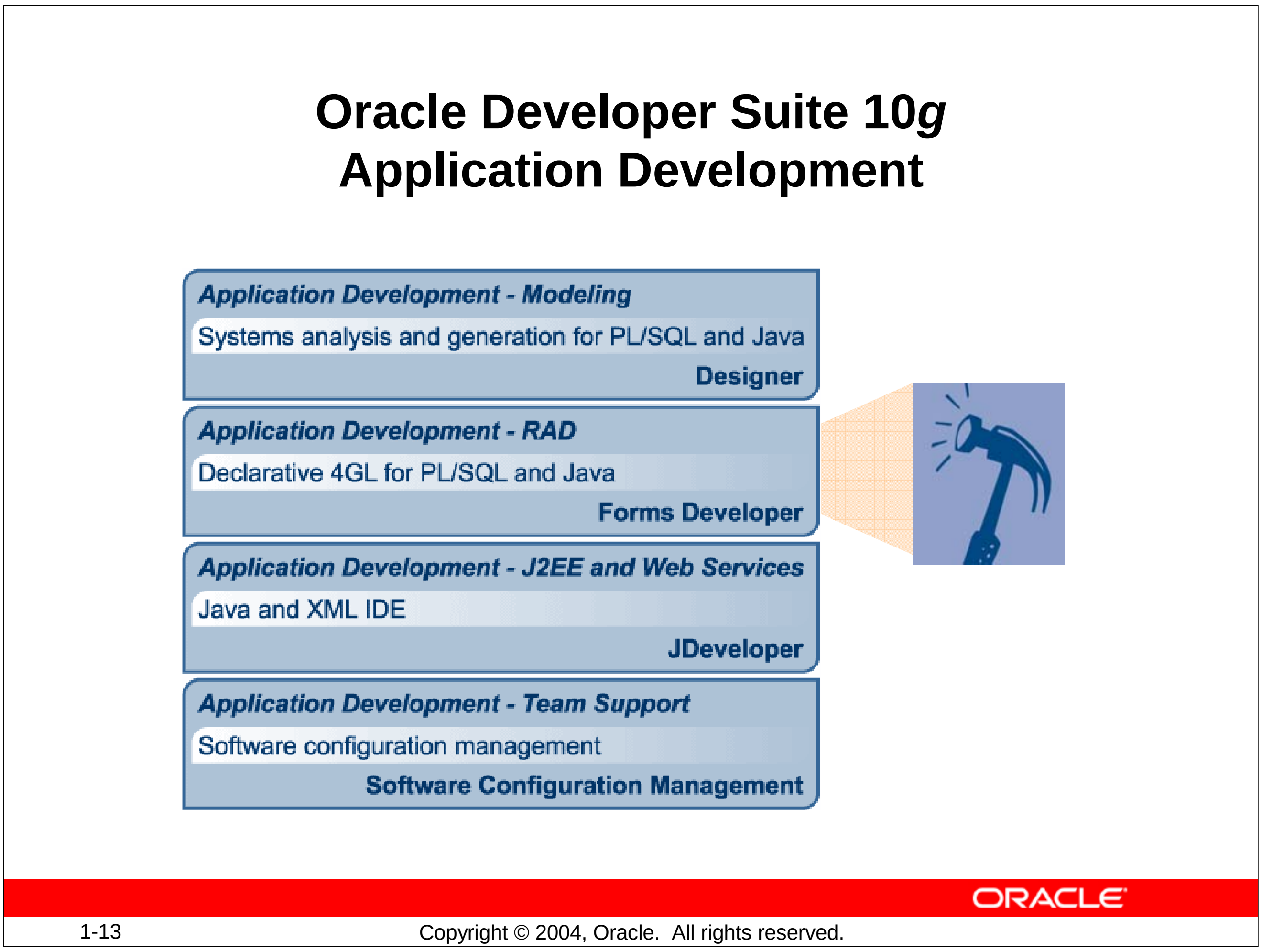


Benefits and Components of Oracle Developer Suite 10*g*

Benefits of Oracle Developer Suite 10*g*

The Oracle Developer Suite:

- Combines the power of Oracle application development tools and Oracle business intelligence tools.
- Provides a standards-based, Java and XML integrated development environment and supports the full application development life-cycle.
- Provides flexible and scalable solutions for data warehousing and business Intelligence.
- Optimized for Oracle Database and Oracle Application Server 10*g*



Oracle Developer Suite 10*g* Application Development

Application Development - Modeling
Systems analysis and generation for PL/SQL and Java
Designer

Application Development - RAD
Declarative 4GL for PL/SQL and Java
Forms Developer

Application Development - J2EE and Web Services
Java and XML IDE
JDeveloper

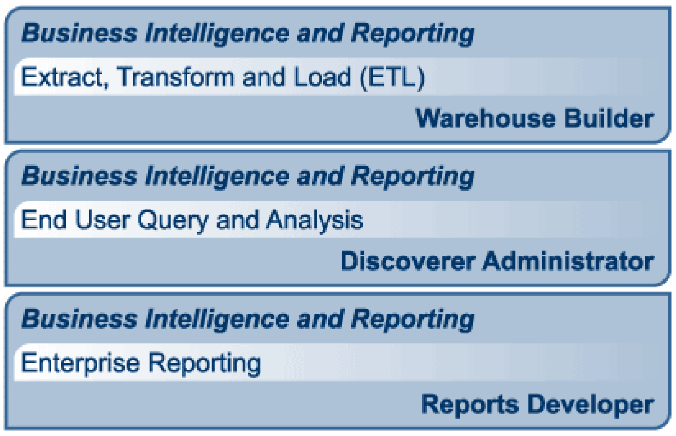
Application Development - Team Support
Software configuration management
Software Configuration Management

Application Development Features of Oracle Developer Suite 10*g*

Oracle Developer Suite provides the following features for application development:

- **Modeling:** Oracle Designer provides visual modeling, reverse engineering, and code generation tools. Oracle Developer Suite also supports UML (Unified Modeling Language) by utilizing visual tools for Activity and Class modeling within the JDeveloper component.
- **Rapid Application Development (RAD):** RAD capabilities in Oracle Developer Suite feature integrated builders, re-entrant wizards, live previewers, and property inspectors. The JDeveloper component provides additional productivity through Business Components for Java (BC4J), a built-in J2EE framework.
- **J2EE and Web Services:** Oracle Developer Suite supports the latest J2EE 1.2 APIs, including Enterprise JavaBeans (EJB), Java Server Pages (JSP), and Servlets. Web services support SOAP (Simple Object Access Protocol), WSDL (Web Service Definition Language), and UDDI (Universal Description, Discovery, and Integration).
- **Team Support:** Oracle Software Configuration Management provides versioning, dependency management, and impact analysis for all objects and file types.

Oracle Developer Suite 10*g* Business Intelligence


Business Intelligence and Reporting
Extract, Transform and Load (ETL)
Warehouse Builder
Business Intelligence and Reporting
End User Query and Analysis
Discoverer Administrator
Business Intelligence and Reporting
Enterprise Reporting
Reports Developer

Business Intelligence Features of Oracle Developer Suite 10*g*

Oracle Developer Suite 10*g* provides the following Business Intelligence features:

- **Extract, transformation, and load (ETL):** Oracle Warehouse Builder provides a graphical interface for mapping and transformation. It also provides an extensible framework for integrating a diverse set of data sources and integration with Business Intelligence Tools.
- **End user query and analysis:** With Oracle Discoverer Administrator, you can create and maintain a business-oriented view of the data that supports the Discoverer client tools: Discoverer Plus and Discoverer Viewer (in Oracle Application Server), and Discoverer Desktop (in Oracle Developer Suite).
- **Enterprise Reporting:** Oracle Reports Developer enables the developer to access any data, to publish it in any format, and to send it anywhere. Supported formats include HTML with CSS, PDF, RTF, Postscript, and XML.

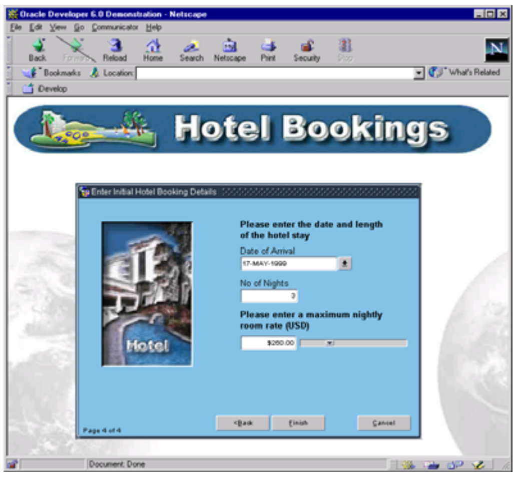
Instructor Note

For further information on Oracle Developer Suite and components, see the Oracle Developer Suite products page on OTN: http://otn.oracle.com/products/ids/index.html.

Oracle Forms Developer Overview

Oracle Forms Developer:

- Is a productive development environment for Internet business applications
- Provides for:
  – Data entry
  – Queries


Oracle Developer 6.0 Demonstration - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: What's Related
Develop
Hotel Bookings
Enter Initial Hotel Booking Details
Hotel
<Back Finish Cancel
Page 4 of 4
Document: Done

Please enter the date and length of the hotel stay

Date of Arrival: 17-MAY-1999

No of Nights: 3

Please enter a maximum nightly room rate (USD): $260.00

What Is Oracle Forms Developer?

Oracle Forms Developer is a productive development environment for building enterprise-class, scalable database applications for the Internet. Oracle Forms Developer provides a set of tools that enable business developers to easily and quickly construct sophisticated database forms and business logic with a minimum of effort.

Oracle Forms Developer uses powerful declarative capabilities to rapidly create applications from database definitions that leverage the tight integration with the Oracle database. The toolset leverages Java technology, promotes reuse, and is designed to allow developers to declaratively build rich user interfaces. Developer productivity is further increased through a single integrated development environment that enables distributed debugging across all tiers, utilizing the same PL/SQL language for both server and client.

Oracle Forms Developer's tight integration with Oracle Designer enables you to use a productive model-driven development approach. Oracle Forms Developer applications can be automatically generated from business requirements designed in the Oracle Designer modeling environment. These models are stored in the Oracle Repository. Code-level changes made within the Oracle Forms Developer environment can be automatically reverse engineered back into the models, preserving the integrity between the models and the application.

Oracle Forms Developer: Key Features

- Tools for rapid application development
- Application partitioning
- Flexible source control
- Extended scalability
- Object reuse

Oracle Forms Developer 10*g*: Key Features

Tools for Rapid Application Development: You can create and modify applications with little or no code. Productivity is enhanced with wizard-based rapid application development and built-in commands that perform common functions.

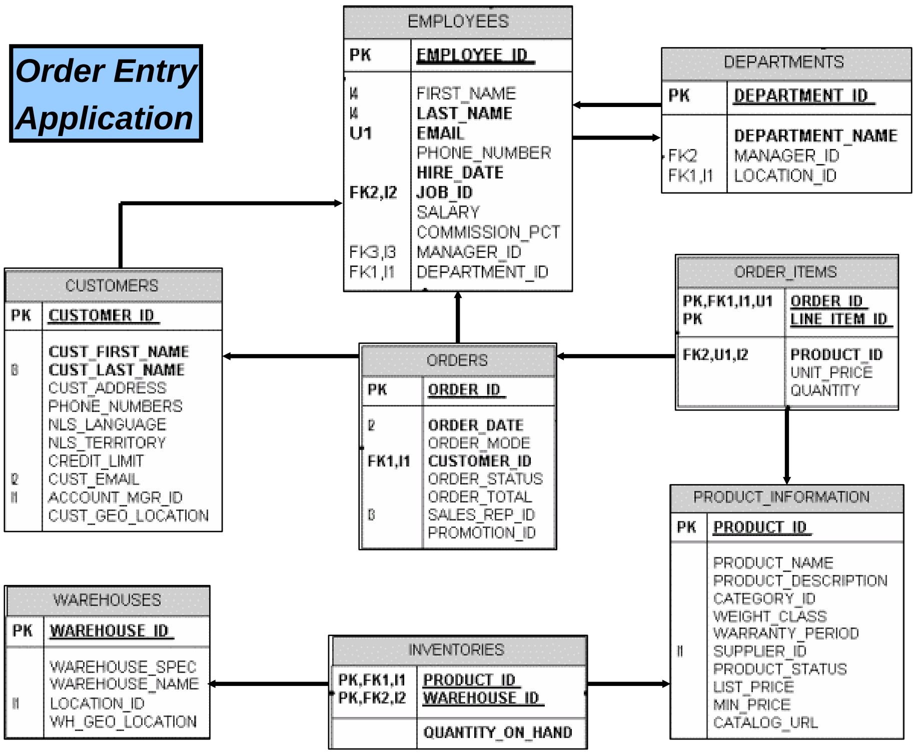
Application Partitioning: You can place individual PL/SQL program units on the database server or in the application, whichever is most suitable. You can drag-and-drop objects between modules and the database server.

Flexible Source Control: Oracle Software Configuration Manager (SCM) is integrated directly in Forms Developer to provide source control options, such as checkin and checkout capability, versioning, diff and merge utilities, and impact analysis.

Extended Scalability: The multi-tiered architecture enables you to scale applications from a single user to tens of thousands of users, with no changes to the application. You can use server functionality, such as array DML, database cursors, or bind variables, to improve scalability.

Object Reuse: Oracle Forms Developer offers an inheritance model that facilitates the inheritance of attributes and code from one object to another and from one application to another, through subclassing and object libraries.

Summit Office Supply Schema



ORACLE

1-17

Introducing the Course Application

The Summit Office Supply Schema

The simplified table diagram shows the tables that are used throughout the course to build the Forms application. These same tables are used in other Oracle courses as well.

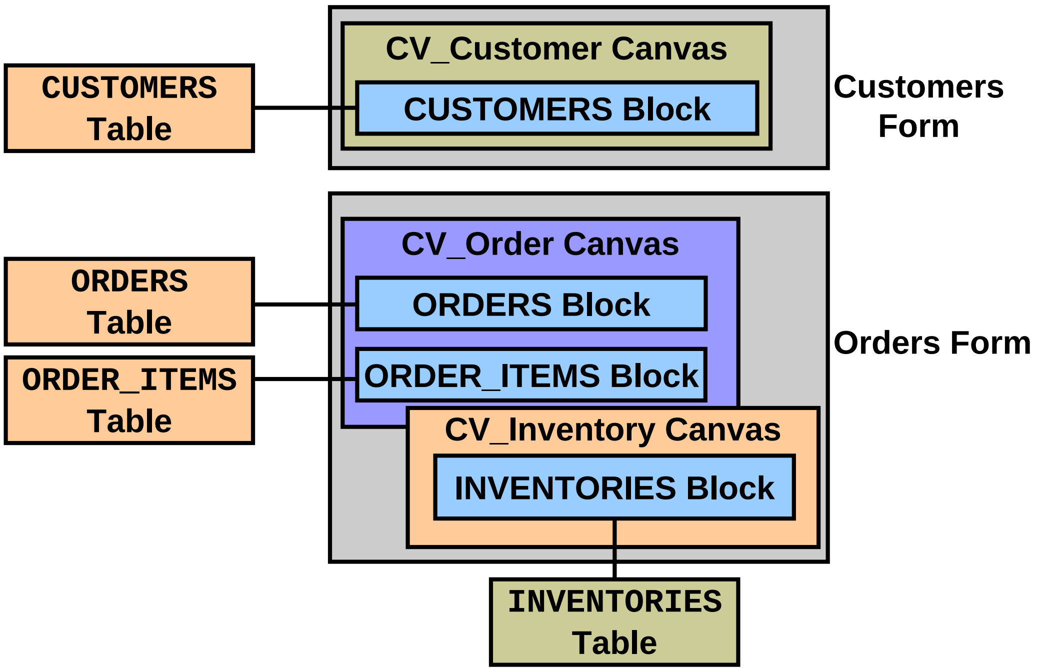
Summit Office Supply is a company that sells office products to customers. Summit has a number of employees in several departments. Some employees are sales representatives who have a relationship with specific customers.

Customers place orders. Each order consists of one or more line items. Each line item represents a product.

Many products have an associated image, in the form of an image file.

The company products are stored in a number of warehouses. The contents of the warehouses are managed in the inventory.

Summit Application



The Summit Office Supply Application

The following example of a Forms Builder application will familiarize you with the main run-time facilities of the product. You will also build your own version of this application during the practices in this course.

The Summit company produces a range of office supplies that they sell to businesses and individuals (their customers). The Summit application is an order-entry system that maintains customer details, their orders, and the available stock (inventory).

The application consists of two main forms:

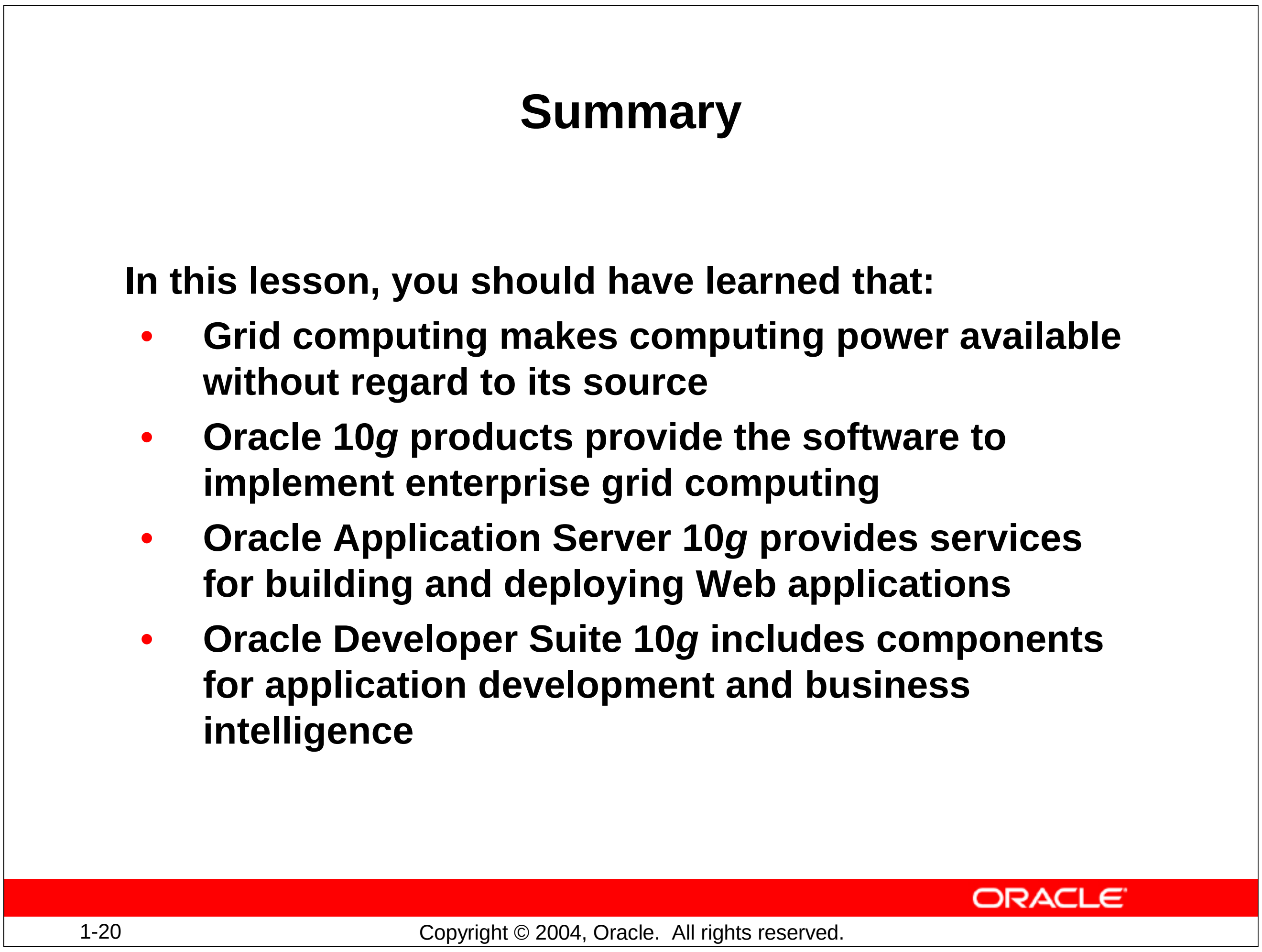
- **Customers form:** The Customers facilitates queries on existing customers and the insertion, update, or deletion of customer records. When a customer is selected, the user can open the Orders form to enter or view orders for that customer. The form consists of a single block, the Customers block, a single record block, whose base table is Customers.

The Summit Office Supply Application (continued)

- **Orders form:** Opened from the Customers form, the Orders form displays orders for a customer and the line items that belong to each order. Orders may also be created, modified, or deleted in this form. You can also display the stock available on the ordered products. The form consists of three blocks:
  - Orders block: The Orders block is a single record master block for the form The base table is Orders, but the block also displays associated information from other tables, such as the name of the customer.
  - Order_Items block: The Order_Items block is the related detail block for an order, showing its line items and the products ordered. This is a multirecord block whose items are on the same canvas as those in the Orders block. The base table of the Order_Items block is Order_Items, but the block displays information from other tables, such as the product description.
  - Inventories block: The Inventories block is a multirecord block showing warehouse stock for a product. Its items are on a separate canvas, which is assigned to its own window. This block is linked to the current product in the Order_Items block, but the two blocks can operate independently.

Instructor Note

Explain to students that the concepts of blocks, windows, canvases, and so on are presented in later lessons. The descriptions above are just to give an overview of the application. Students may want to refer back to this overview later when the concepts are presented.



Summary

In this lesson, you should have learned that:

- Grid computing makes computing power available without regard to its source
- Oracle 10*g* products provide the software to implement enterprise grid computing
- Oracle Application Server 10*g* provides services for building and deploying Web applications
- Oracle Developer Suite 10*g* includes components for application development and business intelligence

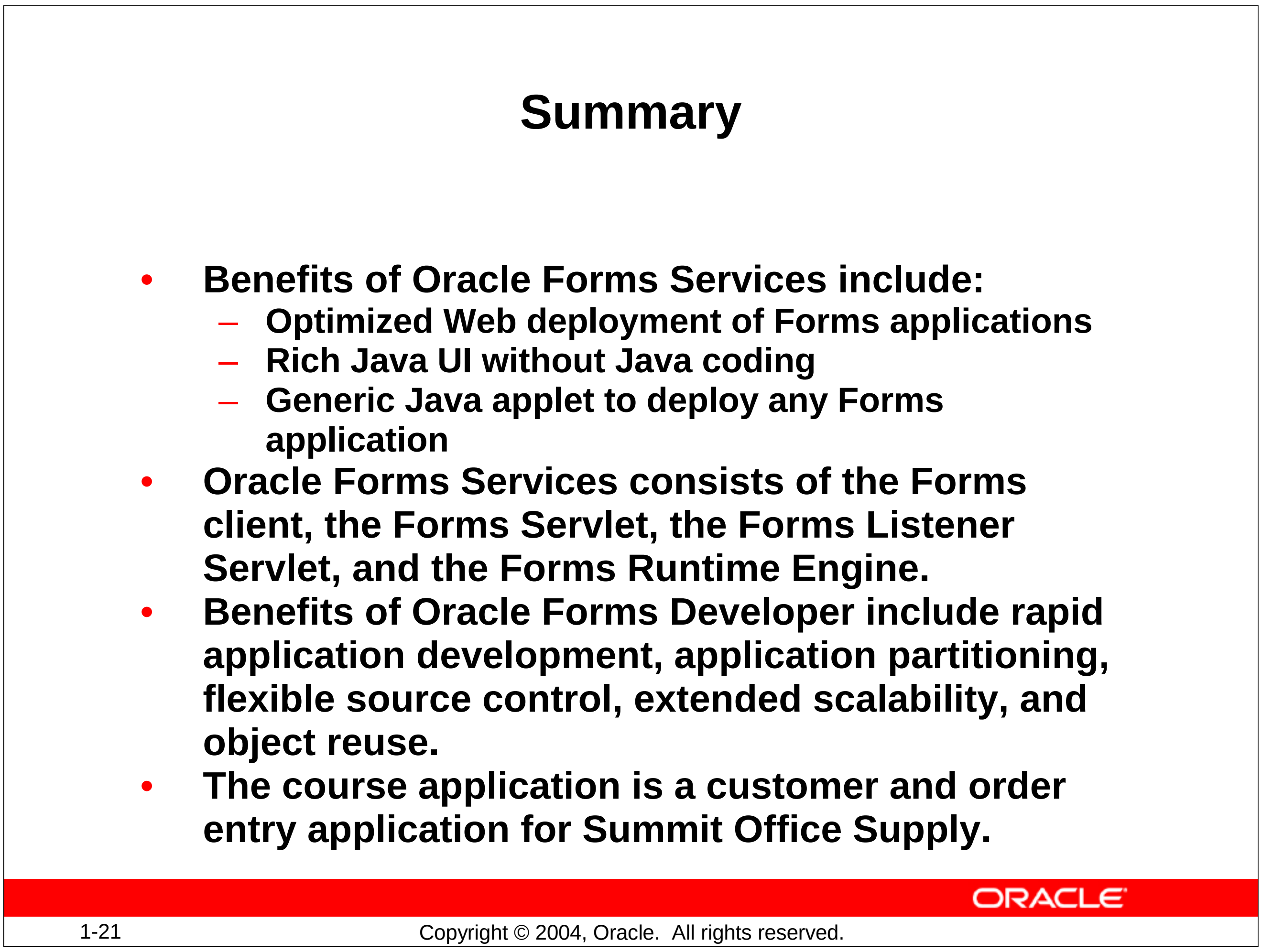
1-20

Summary

Grid computing seeks to make computing power available on demand, without regard to where the data or application resides or which computer processes the request. Oracle 10*g* products (Database, Application Server, and Enterprise Manager Grid Control) provide the software infrastructure to power enterprise grid computing.

Oracle Application Server provides a variety of services for building and deploying Web applications, including the Oracle HTTP Server (OHS), Oracle Containers for J2EE (OC4J), Reports Services, and Forms Services.

Oracle Developer Suite includes components for application development (JDeveloper, Designer, Software Configuration Manager, and Forms Developer) and for business intelligence (Warehouse Builder, Discoverer, and Reports).



Summary

- **Benefits of Oracle Forms Services include:**
  - **Optimized Web deployment of Forms applications**
  - **Rich Java UI without Java coding**
  - **Generic Java applet to deploy any Forms application**
- **Oracle Forms Services consists of the Forms client, the Forms Servlet, the Forms Listener Servlet, and the Forms Runtime Engine.**
- **Benefits of Oracle Forms Developer include rapid application development, application partitioning, flexible source control, extended scalability, and object reuse.**
- **The course application is a customer and order entry application for Summit Office Supply.**

1-21

Summary (continued)

Oracle Forms Services, a component of Oracle Application Server 10*g*, provides for the Web deployment of Forms applications with a rich Java user interface. It uses the same generic applet for any form.

The components of Oracle Forms Services all play a role in running an application. These components are the Forms client (Java applet), the Forms Servlet, the Forms Listener Servlet, and the Forms Runtime Engine.

Oracle Forms Developer is the component of Oracle Developer Suite 10*g* that enables you to develop Forms applications. Benefits of Oracle Forms Developer include:

- **Rapid application development:** Create and modify applications with little or no code
- **Application partitioning:** Drag objects between modules and the database server
- **Flexible source control:** Integration with Software Configuration Manager (SCM)
- **Extended scalability:** Use of server functionality such as array DML, database cursors, or bind variables
- **Object reuse:** Subclassing, object libraries

Running a Forms Developer Application

Schedule:	Timing	Topic
	50 minutes	Lecture
	25 minutes	Practice
	75 minutes	Total

Objectives

After completing this lesson, you should be able to do the following:

- Start OC4J
- Describe the run-time environment
- Describe the elements in a running form
- Navigate a Forms application
- Describe the two main modes of operation
- Run a form in a Web browser
  - Retrieve both restricted and unrestricted data
  - Insert, update, and delete records
  - Display database errors

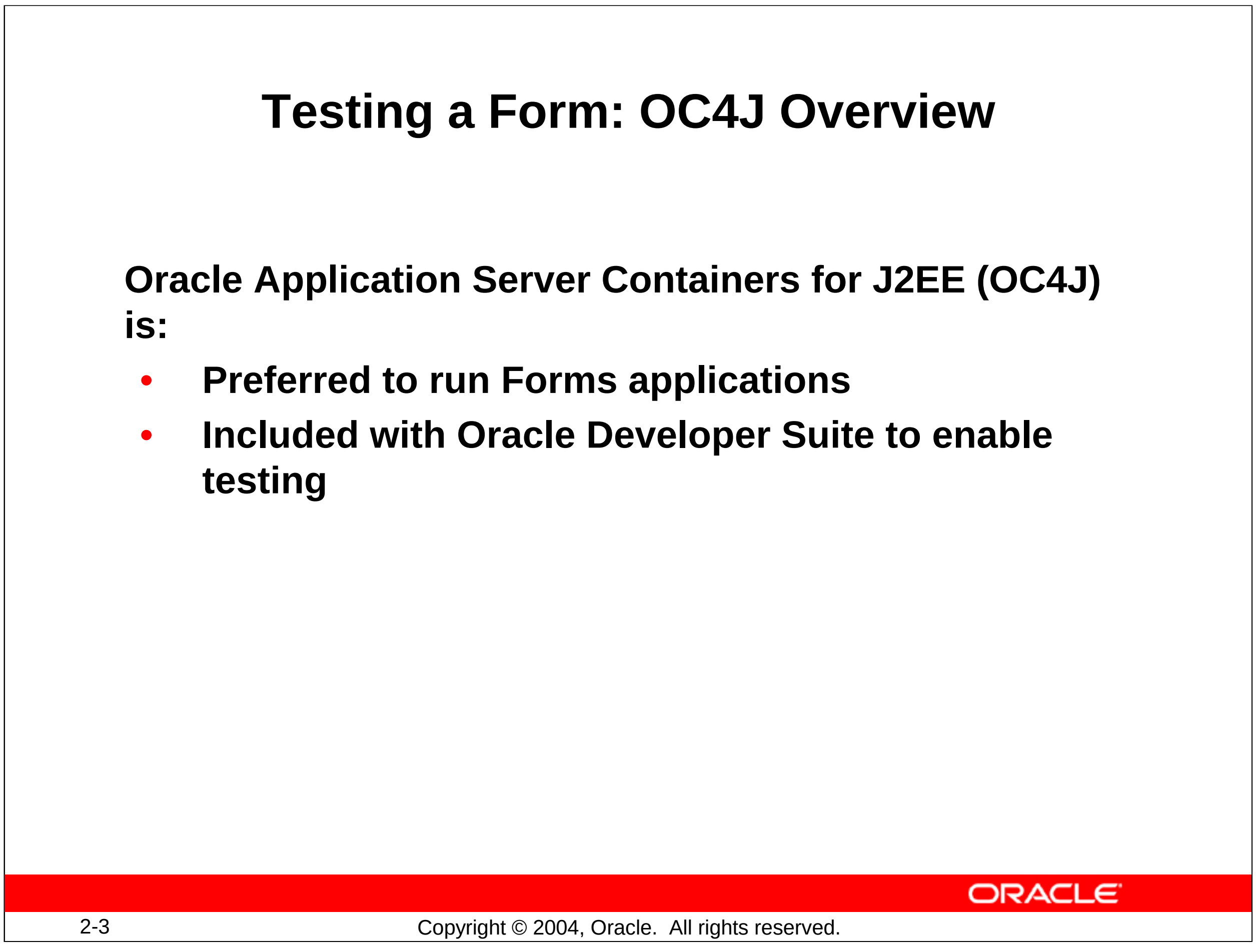
ORACLE

2-2

Introduction

Overview

It is essential that you understand the environment of the form operator before designing and building your own applications. In this lesson you run an existing application on the Web in order to familiarize yourself with the run-time interface of the Oracle Forms Developer.



Testing a Form: OC4J Overview

Oracle Application Server Containers for J2EE (OC4J) is:

- **Preferred to run Forms applications**
- **Included with Oracle Developer Suite to enable testing**

Testing a Form with OC4J

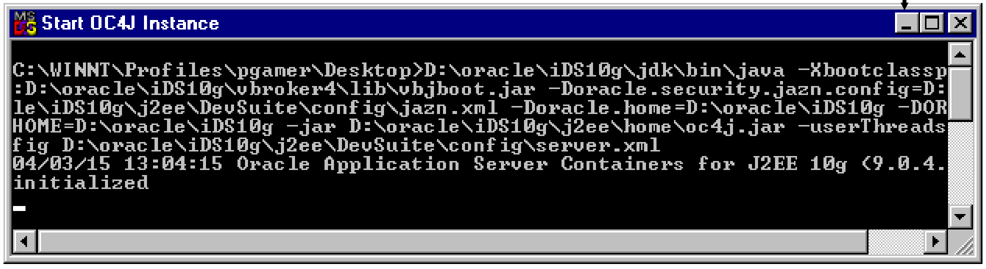
What Is OC4J?

Oracle Application Server Containers for J2EE (OC4J) is Oracle's Java 2 Enterprise Edition (J2EE) container that executes on any Java Virtual Machine (JVM), which is the Java interpreter that is provided on each operating system and hardware platform. It is implemented completely in Java, making it lightweight and easy to install. At the same time, it provides complete support for J2EE applications, including servlets, Enterprise JavaBeans, JavaServer Pages, and so on.

OC4J is ideally suited to run Forms applications. It is included in Oracle Developer Suite to enable you to test your applications, if desired, on the same machine where you are running Forms Builder. In other words, you do not need to install Oracle Application Server to test your applications.

Testing a Form: Starting OC4J

- On NT, run batch file to start OC4J: `startinst.bat`.
- OC4J starts in DOS window:
  - Minimize window
  - Closing window aborts OC4J
- Run batch file to stop OC4J: `stopinst.bat`.



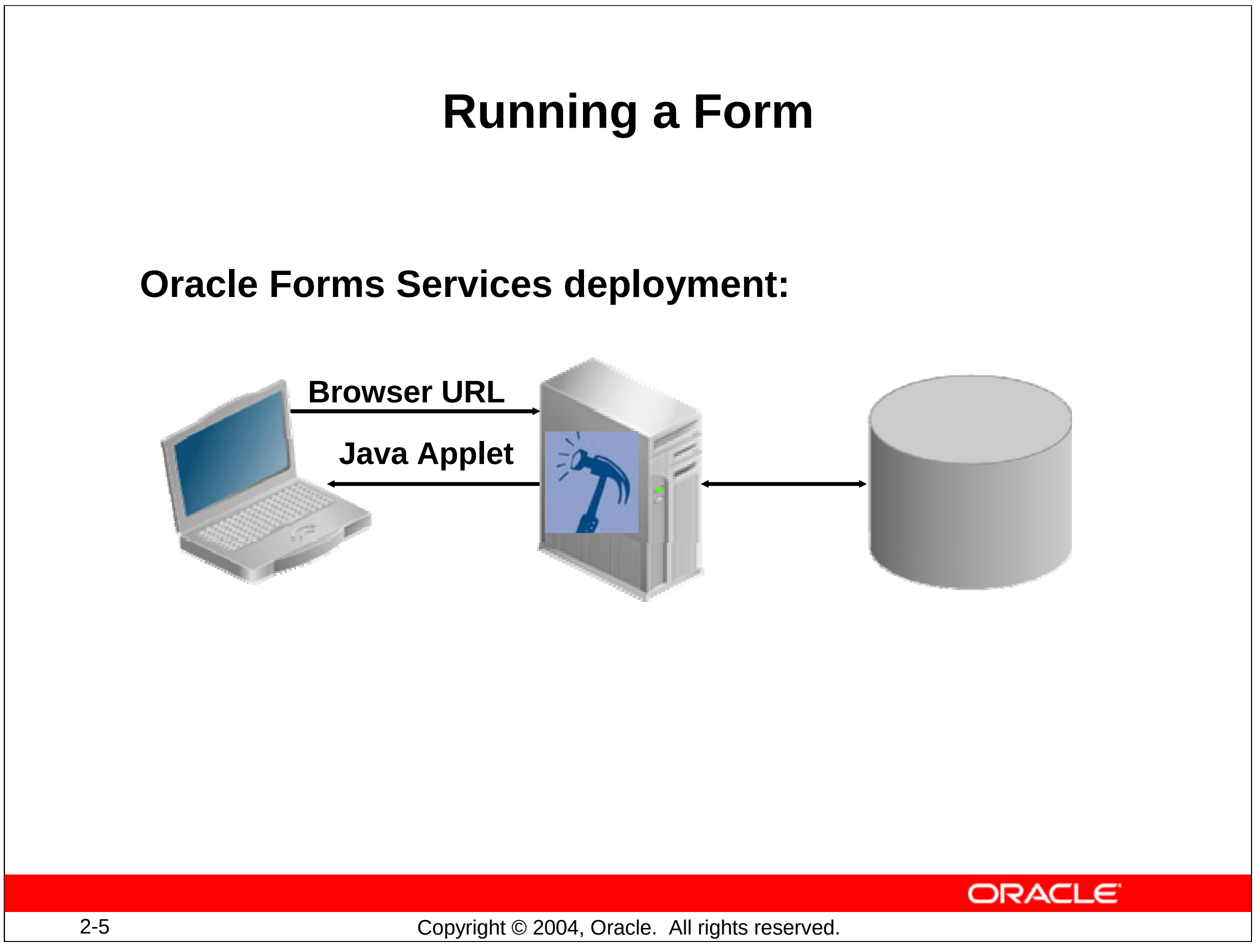
```
C:\WINNT\Profiles\pgamer\Desktop>D:\oracle\iDS10g\jdk\bin\java -Xbootclassp
:D:\oracle\iDS10g\vbroker4\lib\vbjboot.jar -Doracle.security.jazn.config=D:
le\iDS10g\j2ee\DevSuite\config\jazn.xml -Doracle.home=D:\oracle\iDS10g -DOR
HOME=D:\oracle\iDS10g -jar D:\oracle\iDS10g\j2ee\home\oc4j.jar -userThreads
fig D:\oracle\iDS10g\j2ee\DevSuite\config\server.xml
04/03/15 13:04:15 Oracle Application Server Containers for J2EE 10g (9.0.4.
initialized
```

Testing a Form: Starting OC4J

To use OC4J on Windows, start it by executing the batch file provided, called `startinst.bat`. This file is located in the `j2ee\DevSuite\` subdirectory of the Developer Suite home directory. If you will be testing your applications on your client machine, it is a good idea to set up a shortcut to this batch file, and also to the batch file called `stopinst.bat`, which stops the OC4J instance. Alternatively, you can call these batch files from the Windows Start menu: Programs > Oracle Developer Suite – iDS10*g* > Forms Developer > Start [Shutdown] OC4J Instance.

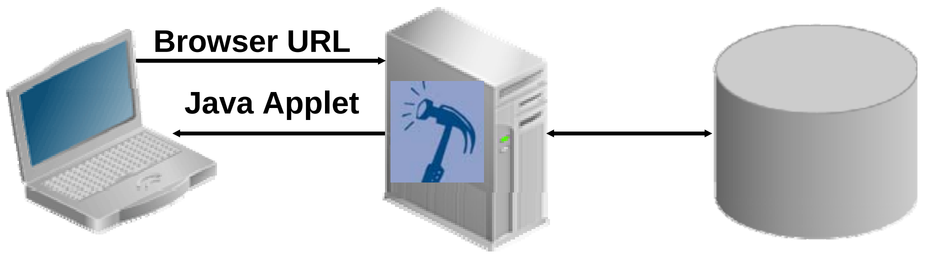
The batch file executes in a separate window; you can minimize this window if desired. Do not close this window, or you will abort the OC4J instance.

When you no longer need to run OC4J you can execute the batch file called `stopinst.bat` to stop the OC4J instance.



Running a Form

Oracle Forms Services deployment:

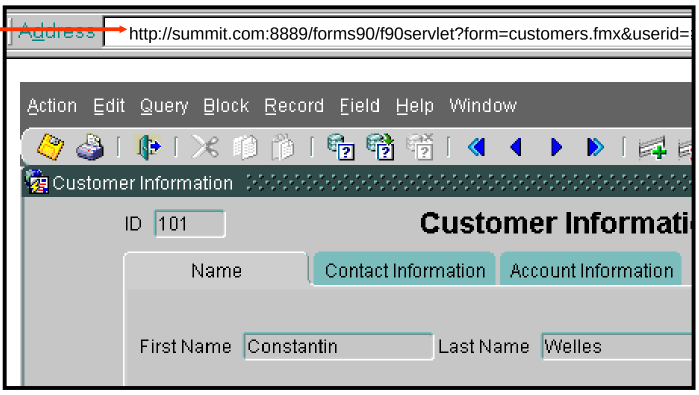


Running a Form

Deploying form applications to the Web is implemented by the three-tier architecture of Oracle Application Server. Application logic and the Forms Services Runtime Engine reside on the middle-tier application server. All trigger processing occurs on database and application servers, while user interface processing occurs on the Forms client. End users can run Forms Developer applications in a Web browser.

Users request an application in their Web browsers by entering a URL that points to the application. Forms Services then generates an HTML file that downloads a Java applet to the client machine. This small applet is capable of displaying the user interface of any form, while the application logic is executed on the middle tier.

Running a Form: Browser



```
http://summit.com:8889/forms90/f90servlet
?form=customers.fmx&userid=username/password@database
&buffer_records=NO&debug_messages=NO&array=YES
&query_only=NO
```

ORACLE

Running a Form: Browser

The URL to invoke an application must have the following format: http://host[:port]/forms servlet or html file[parameters] (optional portions of URL enclosed in brackets)

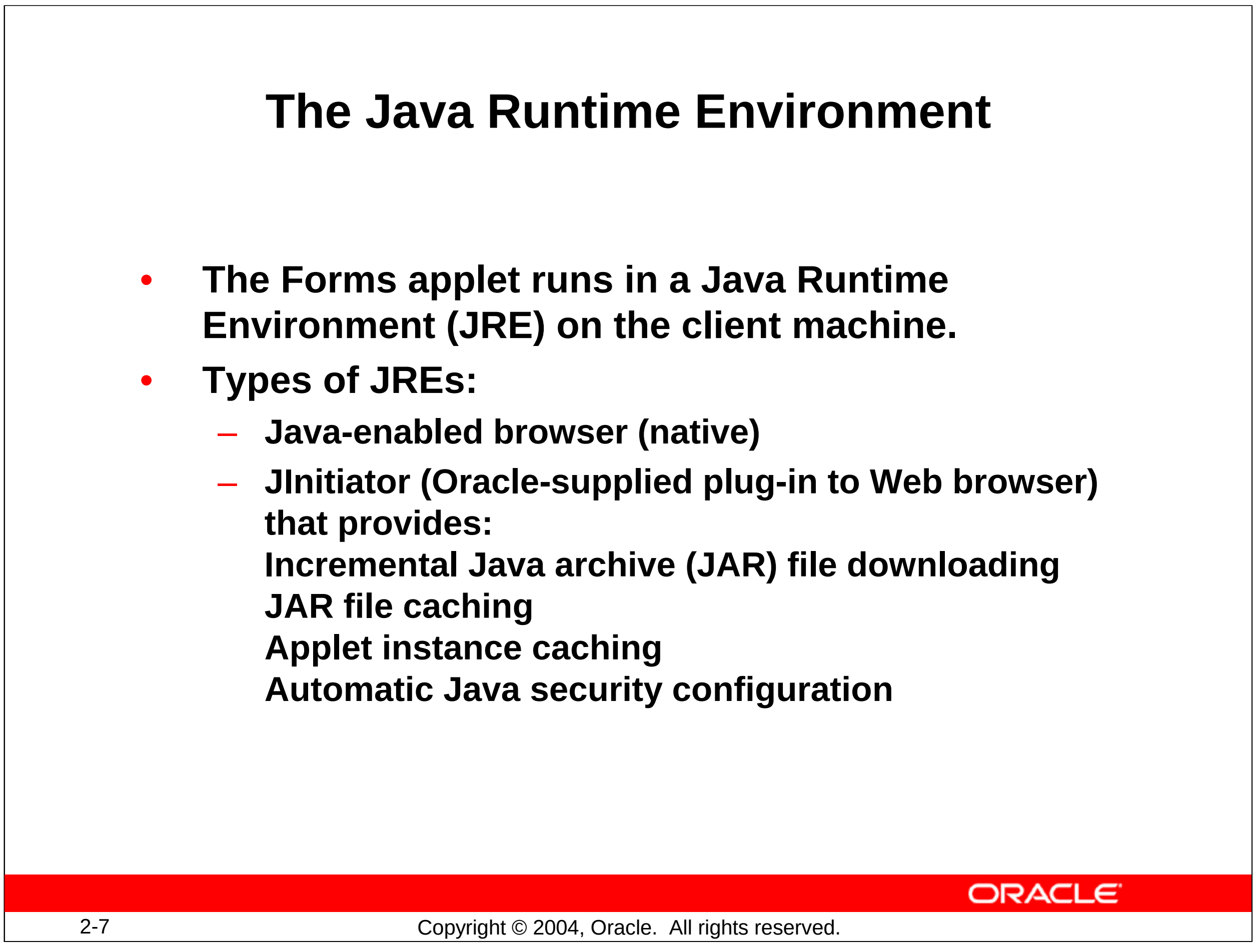
Summit's URL consists of the following components:

Protocol	`http`
Domain	`summit.com`
Port for HTTP Server or OC4J	`xxxx` default for HTTP Server `8889` default for OC4J
Forms Servlet Alias or static html file	`/forms90/f90servlet`

Parameters: this section begins with "?"; parameters separated by "&" (can be specified in the URL or taken from configuration file)	`form=customers.fmx`
	`userid=`*`username/password@database`*
	`buffer_records=NO`
	`debug_messages=NO`
	`....`

Instructor Note

Mention to students that it is common to provide an HTML front end to allow the user to choose the application to start, then construct the URL based on the user's choice.



The Java Runtime Environment

- The Forms applet runs in a Java Runtime Environment (JRE) on the client machine.
- Types of JREs:
  - Java-enabled browser (native)
  - JInitiator (Oracle-supplied plug-in to Web browser) that provides:
    Incremental Java archive (JAR) file downloading
    JAR file caching
    Applet instance caching
    Automatic Java security configuration

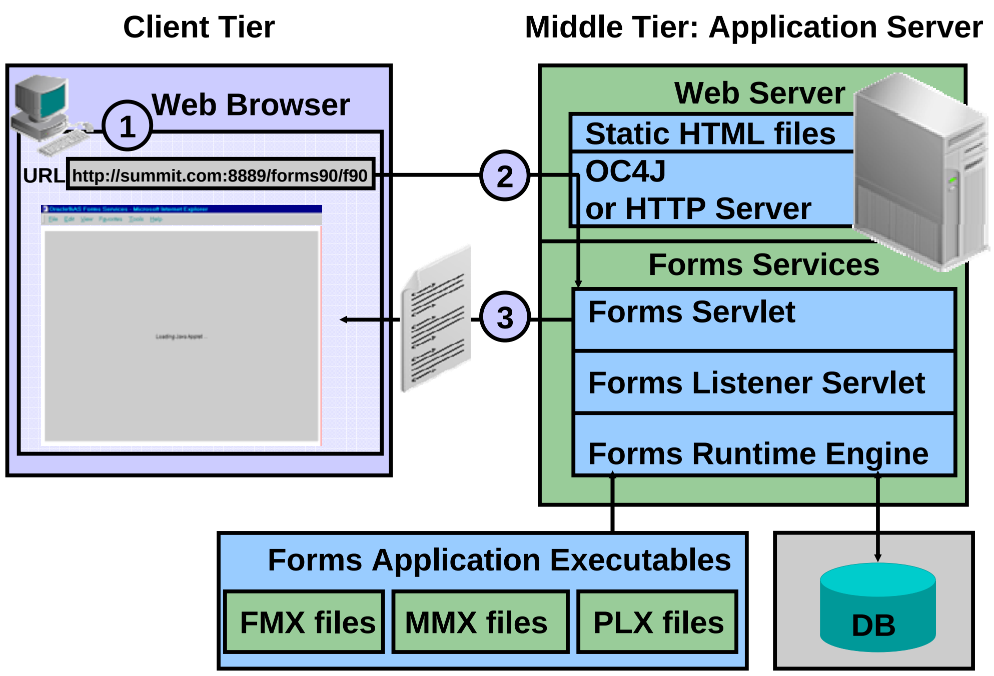
The Java Runtime Environment

The Web browser can run a Java applet because it provides a Java Runtime Environment (JRE). However, not all Web browsers are able to natively run the Forms client. On Windows platforms, Oracle provides a plug-in called JInitiator that provides an alternative JRE capable of running the Forms applet.

JInitiator provides several benefits:

- It is able to incrementally download the Java ARchive files (JAR files) needed for the Forms client, providing faster application startup.
- It caches the JAR files locally, so that they do not need to be downloaded again.
- It improves application performance within a browser session by applet instance caching. When a user navigates from the current page in the browser, the running Forms application is cached. When the user comes back to the page containing the applet, the applet that was running is automatically fully restored, including all of the data entered in the application.
- It is automatically configured to run the Forms application in trusted mode. This enables the application to have access to resources that the Java sandbox model normally prohibits it from using, such as print services.

Starting a Run-Time Session

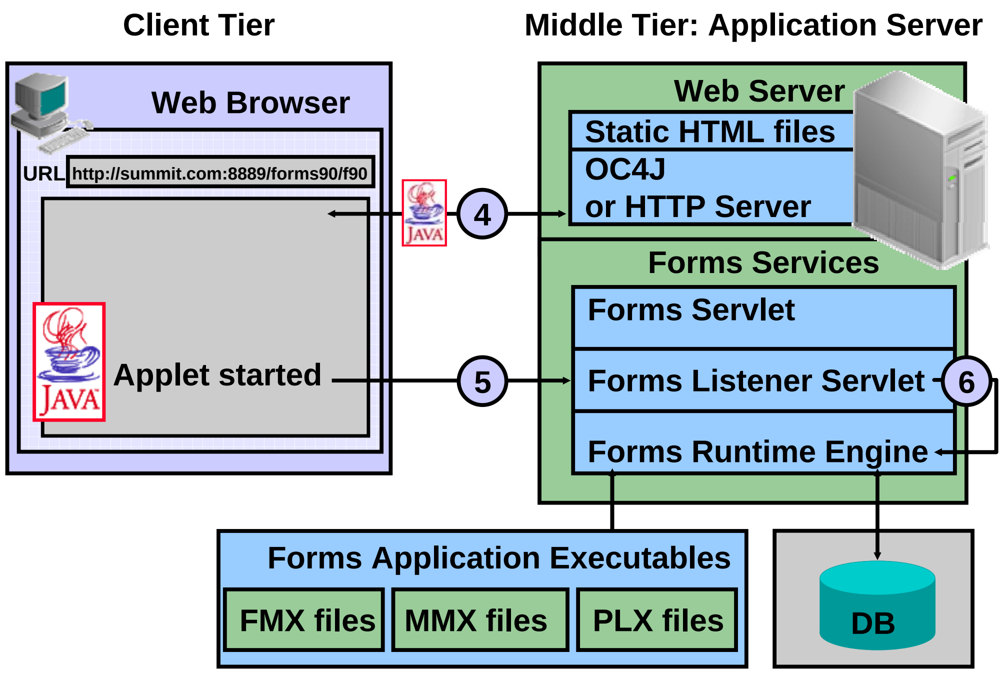


Starting a Run-Time Session

Starting a Run-time session involves the following steps:

1. The user accesses the URL that indicates that a Forms application should be run.
2. The Oracle HTTP Server or OC4J receives an HTTP request from the browser client and contacts the Forms Servlet.
3. The Forms Servlet dynamically creates an HTML page containing all the information to start the Forms session.

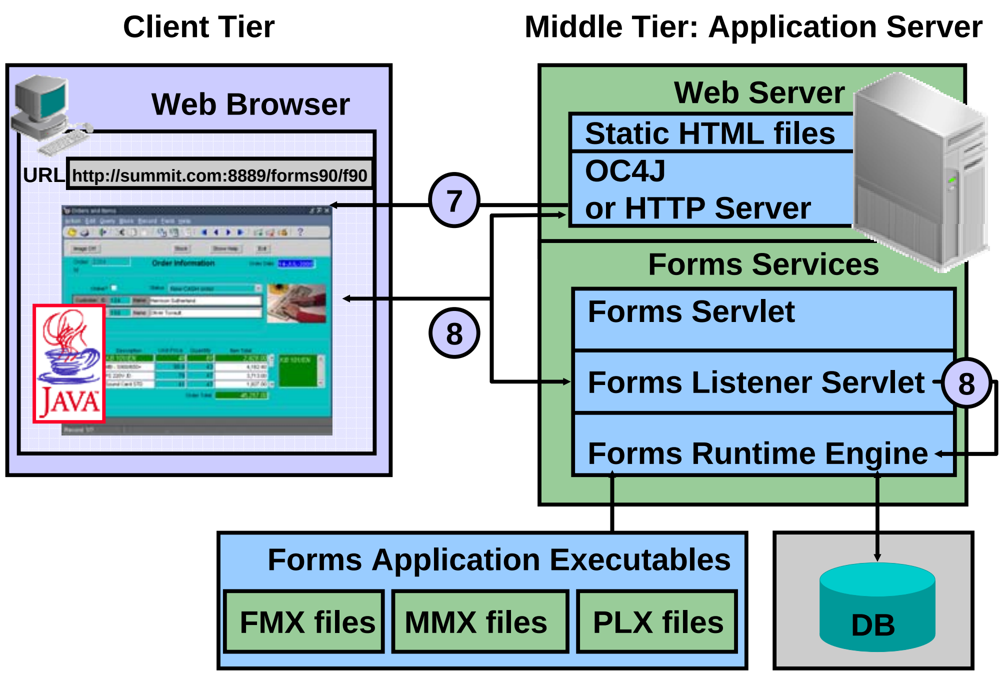
Starting a Run-Time Session



Starting a Run-Time Session (continued)

4. The Oracle HTTP Server or OC4J downloads a generic applet to the client after checking that it has not already been downloaded. The client caches the applet so that it can run future Forms applications without downloading it again.
5. The client applet contacts the Forms Listener Servlet to start the session. The Forms Listener Servlet starts an instance of the Forms Runtime Engine on the Forms Server (middle tier). If included in the HTML file, Forms Runtime command-line parameters (such as form name, user ID and password, database SID, and so on) and any user-defined Forms Builder parameters are passed to the process by the Forms Listener Servlet.
6. The Forms Listener Servlet establishes a connection with the Runtime Engine, which connects to the database if needed and loads application executable files.

Starting a Run-Time Session



Starting a Run-Time Session (continued)

7. The Forms applet displays the user interface of the application in the main window of the user's Web browser.
8. The Forms Listener Servlet, working through OC4J or the HTTP Server, manages communication between the Forms applet and the Runtime Engine.

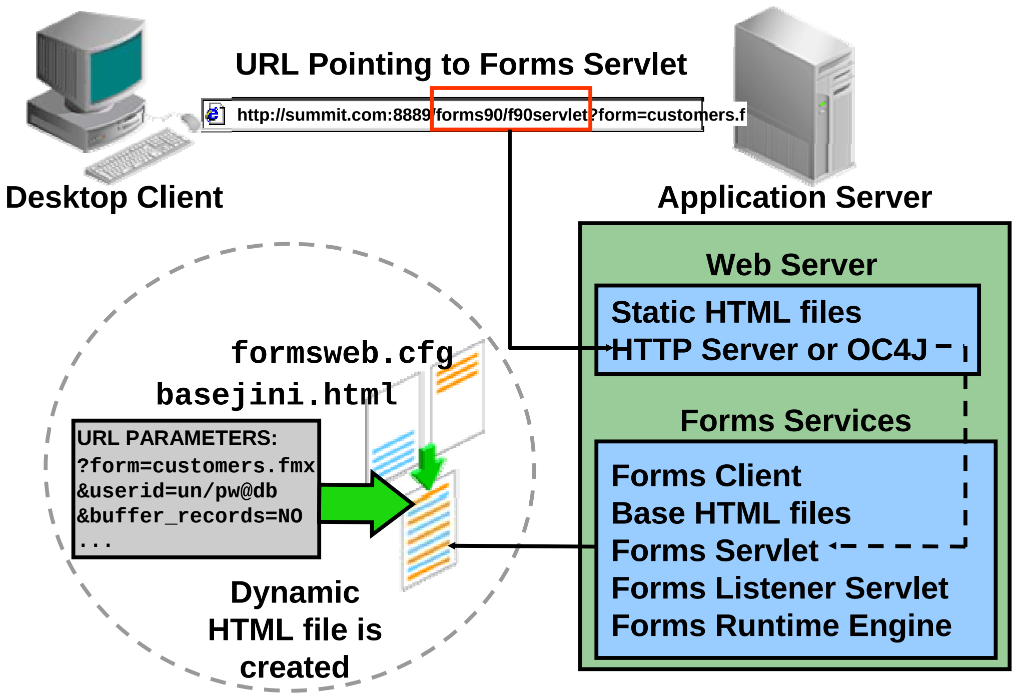
Technical Note

More information about the Oracle Forms Listener Servlet is available in the Oracle white paper: *Oracle Application Server Forms 10*g *(9.0.4) Technical Overview*.

Instructor Note

Explain the process of using FTP to move a form from NT to the middle-tier server, compiling the form, and then running it from a browser. As you present the next few slides, explain to the students that the Forms Services run on the middle tier, and that this is where the run time environment variables are set.

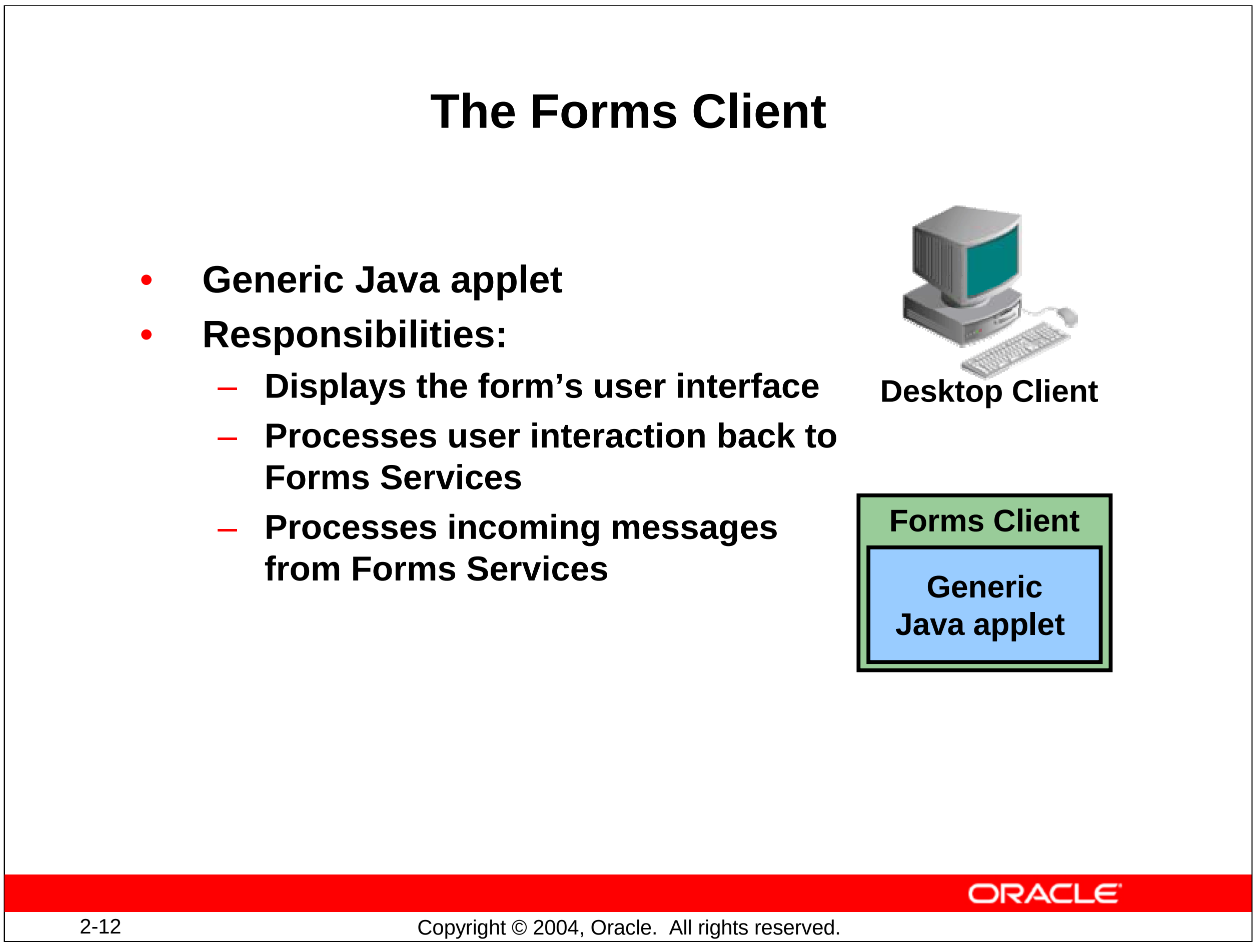
The Forms Servlet



The Forms Servlet

The Forms Servlet is a Java servlet that creates a dynamic HTML file by merging information from the following sources:

- The Forms Web configuration file
- The Forms base HTML file
- The application’s URL parameters



The Forms Client

- Generic Java applet
- Responsibilities:
  - Displays the form’s user interface
  - Processes user interaction back to Forms Services
  - Processes incoming messages from Forms Services

Desktop Client

Forms Client

Generic Java applet

ORACLE

2-12

The Forms Client?

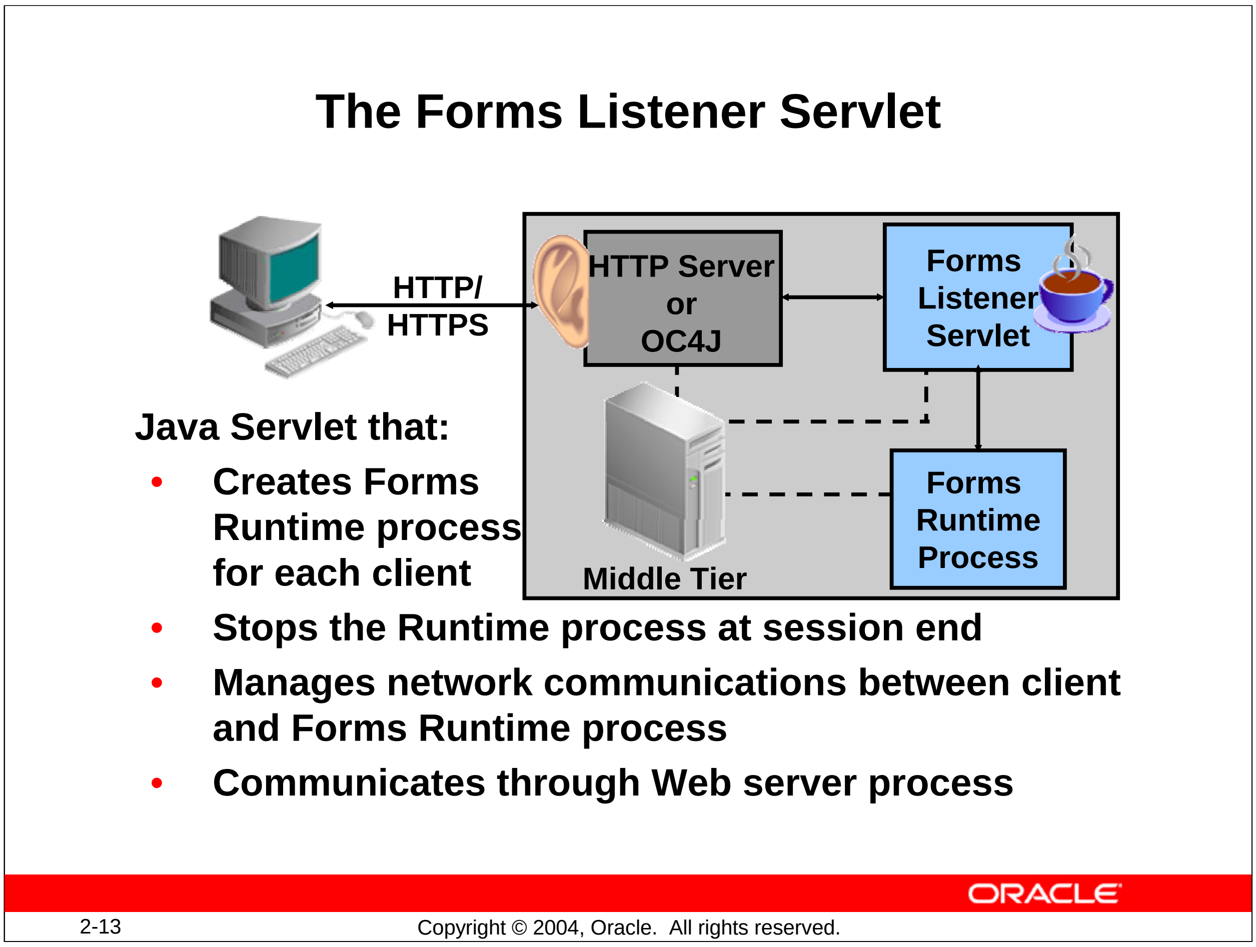
The Forms Client is a generic Java applet. Forms Services dynamically downloads this applet and automatically caches it on the client machine. The Forms Client consists of a set of Java classes. At startup, only those Java classes that are necessary to initialize the application are downloaded. Additional class files are downloaded dynamically, as needed, to support additional user interface activity.

You do not have to deploy a separate Java applet for each application. The same generic applet is used to run any Forms Services application, regardless of its size and complexity.

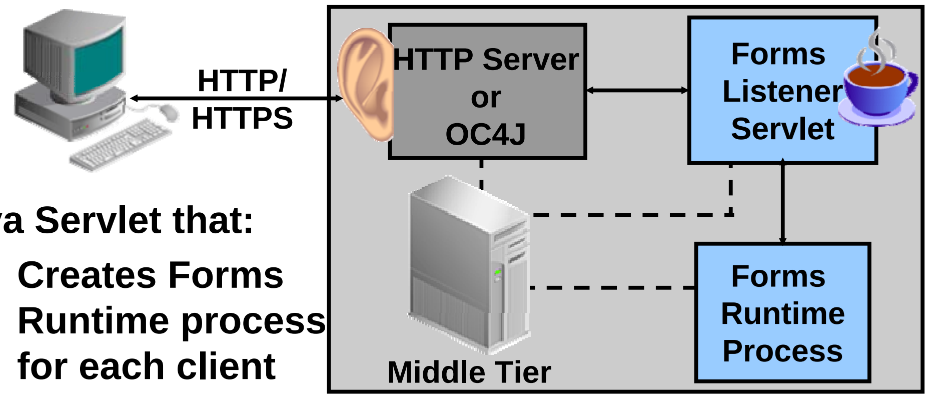
Responsibilities of the Forms Client

The Forms Client represents the user interface layer and has three primary functions:

- To render the Forms Services application display for the user
- To efficiently process user interaction back to Forms Services
- To process incoming messages from Forms Services and translate them into interface objects for the end user efficiently



The Forms Listener Servlet



Java Servlet that:

- **Creates Forms Runtime process for each client**
- **Stops the Runtime process at session end**
- **Manages network communications between client and Forms Runtime process**
- **Communicates through Web server process**

2-13 Copyright © 2004, Oracle. All rights reserved.

The Forms Listener Servlet?

The Forms Listener Servlet is a Java servlet that runs in a Web server equipped with a servlet engine, such as OC4J. The Web server directs HTTP requests for the Forms Listener Servlet directly to the servlet instances.

The Forms Listener Servlet is in charge of:

- Managing the creation of the Forms Runtime process for each client
- Managing the network communications that occur between the client and its associated Forms Runtime process, through the Web server

This means that the client sends HTTP requests and receives HTTP responses from the Web server process itself. Since the Web server acts as the network endpoint for the client, there is no need to expose additional server machines and ports at the firewall.

The Runtime Engine

The Forms Runtime Engine:

- Is a process (`ifweb90`) that runs on the Application Server
- Manages application logic and processing
- Communicates with the client browser and the database

ORACLE

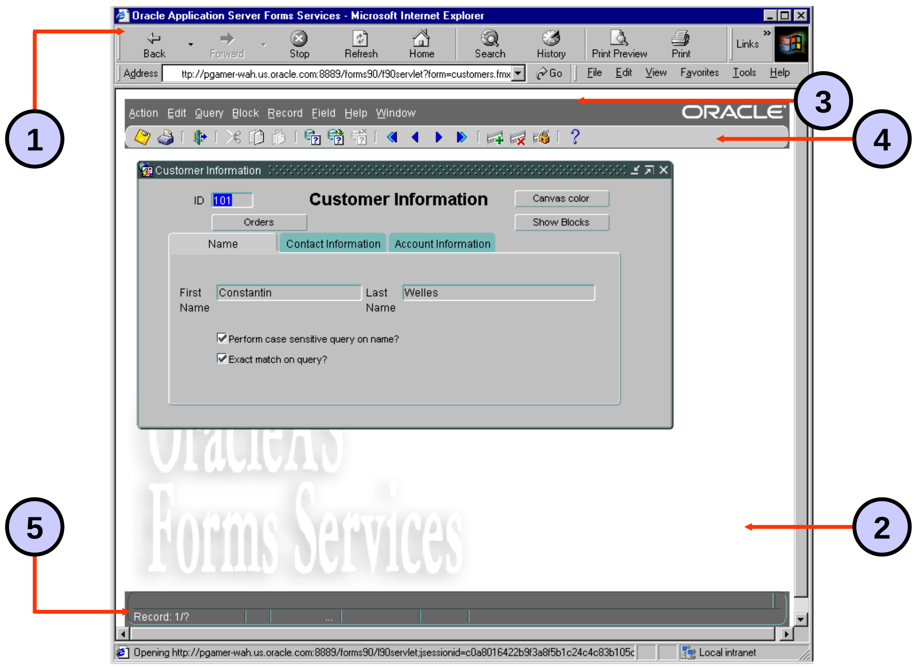
2-14 Copyright © 2004, Oracle. All rights reserved.

The Forms Runtime Engine?

The Forms Runtime Engine is a process on the Application Server that is started by the Forms Listener Servlet. You cannot start the runtime engine directly.

The Forms Runtime Engine handles all the application logic and Forms functionality and executes the code written into the application. It manages requests from the Forms Client and sends metadata to the client to describe the user interface. It connects to and communicates with the Oracle database via Oracle Net Services, the replacement for Net8 and SQL*Net.

What You See at Run Time



ttp://pgamer-wah.us.oracle.com:8889/forms90/f90servlet?form=customers.fmx

ID 101

First Name Constantin

Last Name Welles

✓ Perform case sensitive query on name?

✓ Exact match on query?

Record: 1/?

Opening http://pgamer-wah.us.oracle.com:8889/forms90/f90servlet;jsessionid=c0a8016422b9f3a8f5b1c24c4c83b105c

What You See at Run Time

At run time, you will see the following components:

1. Browser window
2. Java applet (contained within browser window)
3. Default menu (contained within applet)
4. Menu toolbar (contained within applet)
5. Console (contained within applet)

What is the Default Menu?

The *Default menu*, which is part of all Oracle Forms Developer applications, is an alternative to keystroke operations. You can replace or customize the Default menu to introduce your own functionality into a form module.

What is the Menu Toolbar?

The *Menu toolbar* contains buttons corresponding to menu items. At run time, it appears above any user-defined toolbars. It executes the same code as menu items, and it is a shortcut to menu commands that does not duplicate code or effort.

What You See at Run-Time (continued)

What is the Console?

The *console* is the generic name for the standard features that provide information at run time. The console is displayed at the bottom of the window and consists of:

- The message line that displays both Forms and application-specific messages.
- The status line that displays a variety of indicators to reflect the current state of the form module.

Indicator	**Description**
Record: *n/m*	The *n*th record retrieved and displayed so far, out of *m* number of total records that can be retrieved by the query. Until the last record is fetched, *m* displays as “?”; after that, it displays the number corresponding to the last record.
Enter-Query	The current block is in Enter Query mode and no records have been retrieved
List of Values	A list of values (LOV) is associated with the current item.

Instructor Note

Demonstration

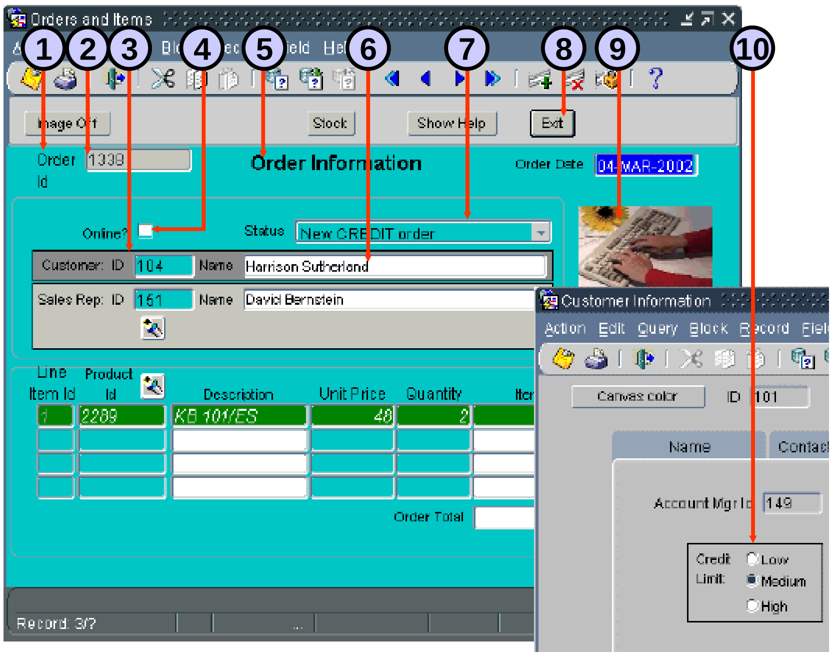
Start an OC4J instance. In Forms Builder, open the form module Customers and run it with the Run Form button. Point out the Browser window, Java applet, Default menu, Menu toolbar, and Console components. Describe the elements of the URL in the browser.

Select Help > Keys from the menu. Execute an unrestricted query. Execute a restricted query to retrieve information about the customer with the ID of 212.

Leaving this browser session open, start another browser session; copy the URL from the first session into the second. Add the following to the end of URL: `&lookAndFeel=generic` (the default Look and Feel is “oracle”). Point out the different appearance of the user interface, especially the rounded appearance of the buttons and the new look of the Menu toolbar.

Terminate these run-time sessions by exiting the forms, and then closing the browser windows. Be sure to advise students to always exit the form to terminate the applet before closing the browser window.

Identifying the Data Elements



Identifying the Data Elements

A Forms application may contain many different kinds of data elements:

1. Prompts
2. Text Items
3. Boilerplate graphics
4. Check boxes
5. Boilerplate text
6. Display items
7. List items
8. Push buttons
9. Image items
10. Radio groups

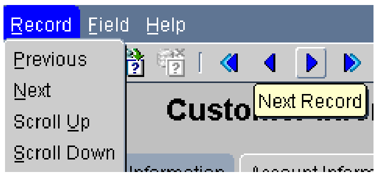
Not Pictured:

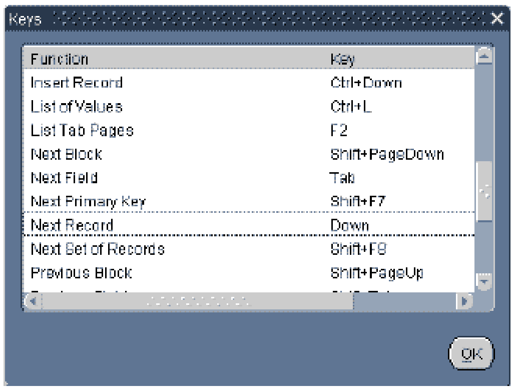
- Hierarchical tree items
- Chart items
- Custom items

Navigating a Forms Developer Application

Methods of Navigation:

- **Default menu**
- **Menu toolbar**
- **Mouse**
- **Buttons**
- **Function keys**





Function	Key
Insert Record	Ctrl+Down
List of Values	Ctrl+L
List Tab Pages	F2
Next Block	Shift+PageDown
Next Field	Tab
Next Primary Key	Shift+F7
Next Record	Down
Next Set of Records	Shift+F8
Previous Block	Shift+PageUp

Navigating a Forms Application

The Default Menu

The Default menu is automatically available in a form, unless it is disabled or replaced with a customized menu. Select options from the menu by using the mouse or function keys. At run time, use the menu to perform the following tasks:

- Move the cursor and navigate between data blocks, records, and items.
- Save or clear all changes.
- Execute queries.
- Insert new records or delete existing records.
- Invoke Help.

The Menu Toolbar

You can use the Default menu toolbar buttons to perform the following operations also available through the Default menu:

- Save all changes.
- Exit the form.
- Execute queries.
- Navigate between data blocks or records.

Navigating a Forms Application (continued)

The Menu Toolbar (continued)

- Insert new records or delete existing records.
- Invoke Help to see properties of an item.

The Mouse

You can use the mouse to navigate and to perform many user operations in a bitmapped environment without needing to learn the function keys. Use the mouse to perform the following actions:

- Move the cursor.
- Select from a menu.
- Select from an LOV.
- Select or clear a check box.
- Select a button, including a radio button.
- Switch to an open window.
- Respond to an alert.
- Scroll records or lines by using a data block or item scroll bar.
- Manipulate a custom item.

Buttons

Web applications often use buttons as a means of navigation. You can click buttons with the mouse. Use buttons to perform the following tasks:

- Move input focus.
- Display a list of values.
- Invoke an editor.
- Invoke another window.
- Commit data.
- Issue a query.
- Perform calculations.
- Exit the form.

Function Keys

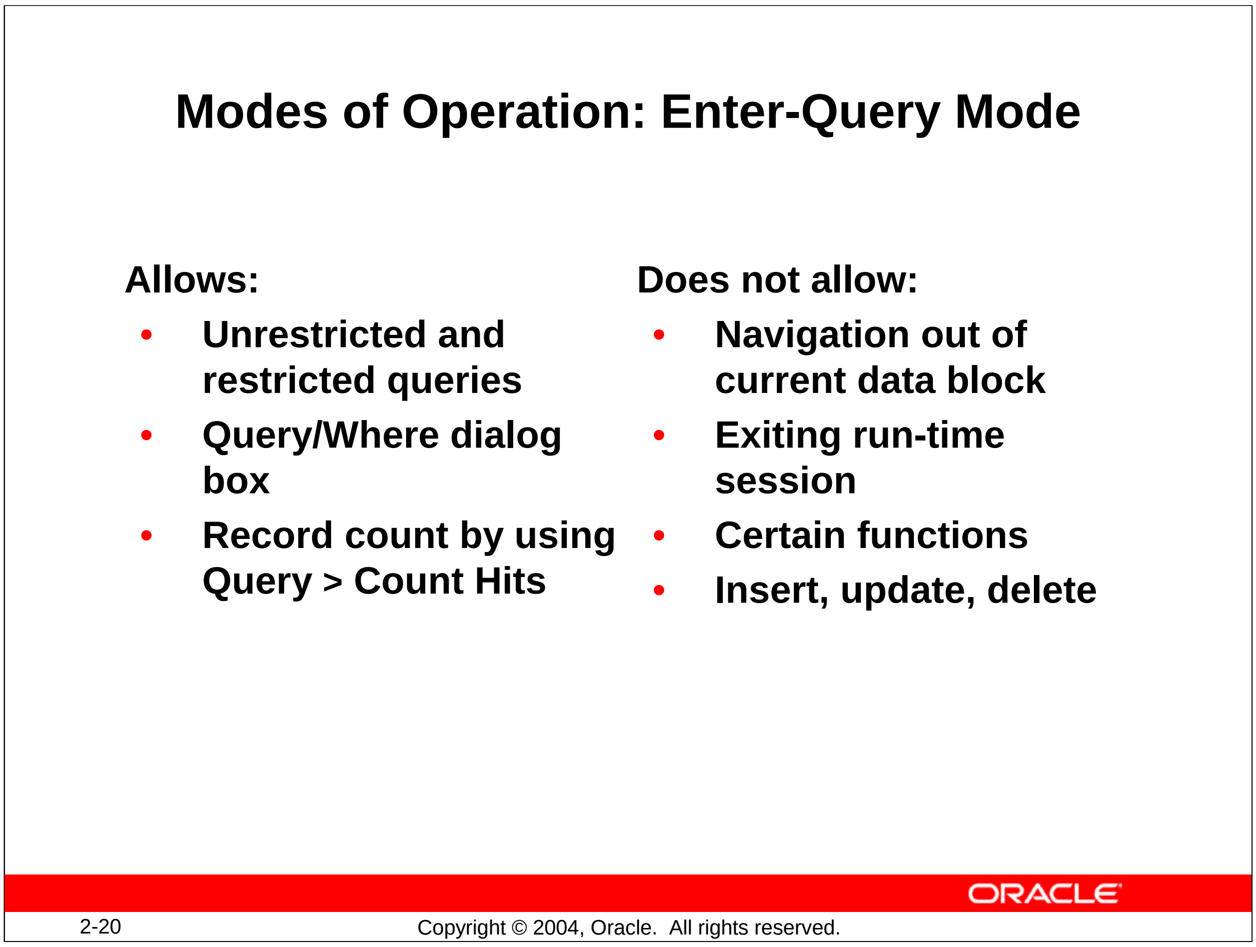
In addition to navigating with the mouse, you can move from item to item in sequence with function keys. Use function keys to perform the following tasks:

- Navigate between data blocks, records, and items.
- Execute queries.
- Insert new records or delete existing ones.
- Invoke Help.

To view a list of keys and the functions they perform, select Help > Keys, or press `[Ctrl]+K`.

Instructor Note

Demonstration: From Forms Builder, run the ORDERS form. Point out the main data elements. Show the different means of navigation.



Modes of Operation: Enter-Query Mode

Allows:
- Unrestricted and restricted queries
- Query/Where dialog box
- Record count by using Query > Count Hits

Does not allow:
- Navigation out of current data block
- Exiting run-time session
- Certain functions
- Insert, update, delete

2-20 Copyright © 2004, Oracle. All rights reserved.

Modes of Operation

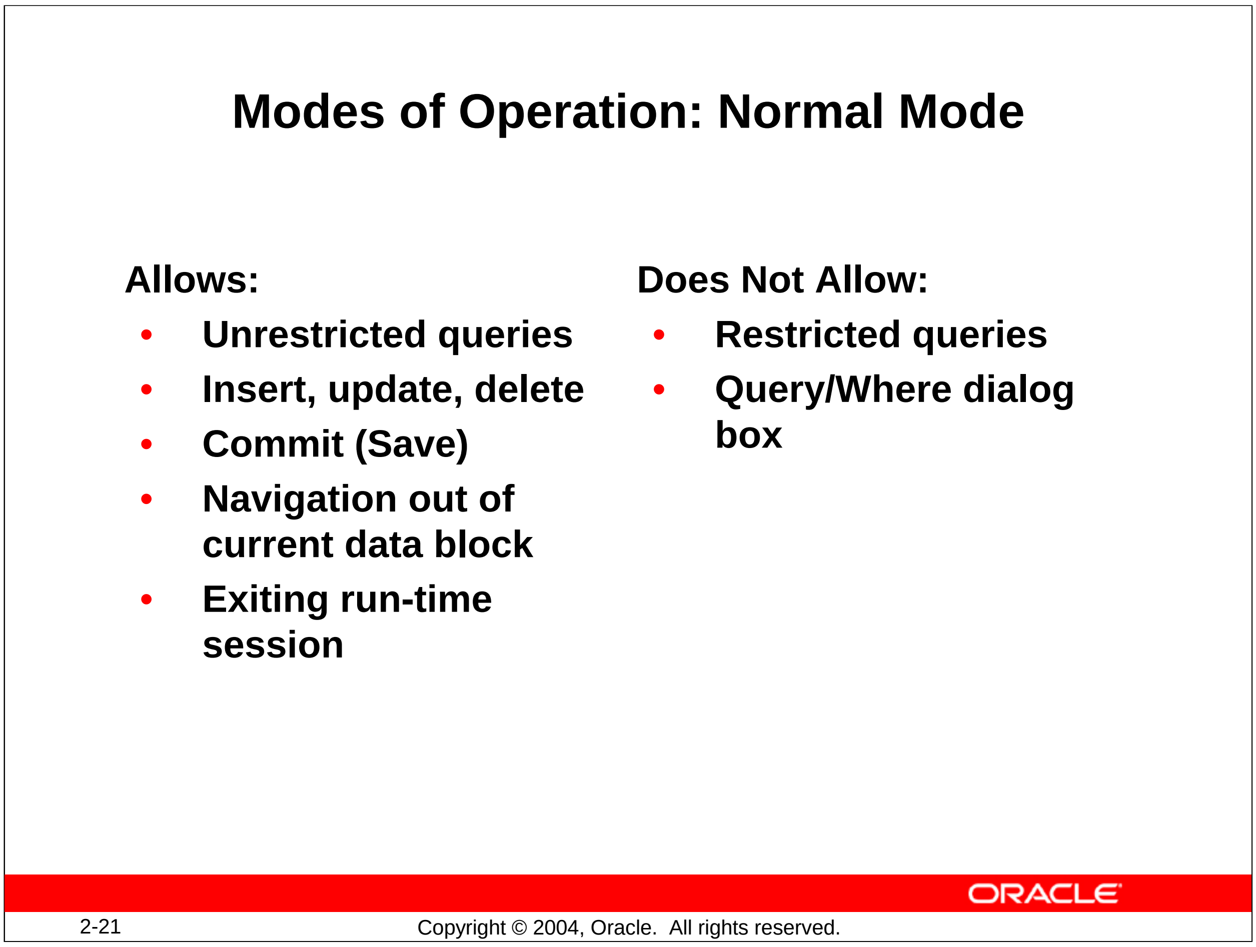
Forms Builder has two main modes of operation: Enter-Query mode and Normal mode. The third mode of operation, Query mode, occurs while Forms is processing a query; the user cannot interact with the form while processing is taking place.

Enter-Query Mode

Use Enter-Query mode to enter search criteria for a database query. In Enter-Query mode, your keystrokes are interpreted as search criteria for retrieving restricted data.

What you can do in Enter-Query mode
- Retrieve all records
- Retrieve records by using selection criteria
- Retrieve records by using the Query/Where dialog box
- Obtain the number of records that will be retrieved before fetching them from the database by using Query > Count Hits

What you cannot do in Enter-Query mode
- Navigate out of the current block
- Exit from the run-time session
- Use certain functions, such as Next Record
- Insert new records
- Update existing records
- Delete records



Modes of Operation: Normal Mode

Allows:

- **Unrestricted queries**
- **Insert, update, delete**
- **Commit (Save)**
- **Navigation out of current data block**
- **Exiting run-time session**

Does Not Allow:

- **Restricted queries**
- **Query/Where dialog box**

ORACLE

2-21

Normal Mode

Use Normal mode to insert and alter records in the database. In Normal mode, your keystrokes are interpreted as either the entering of new records or the altering of existing ones.

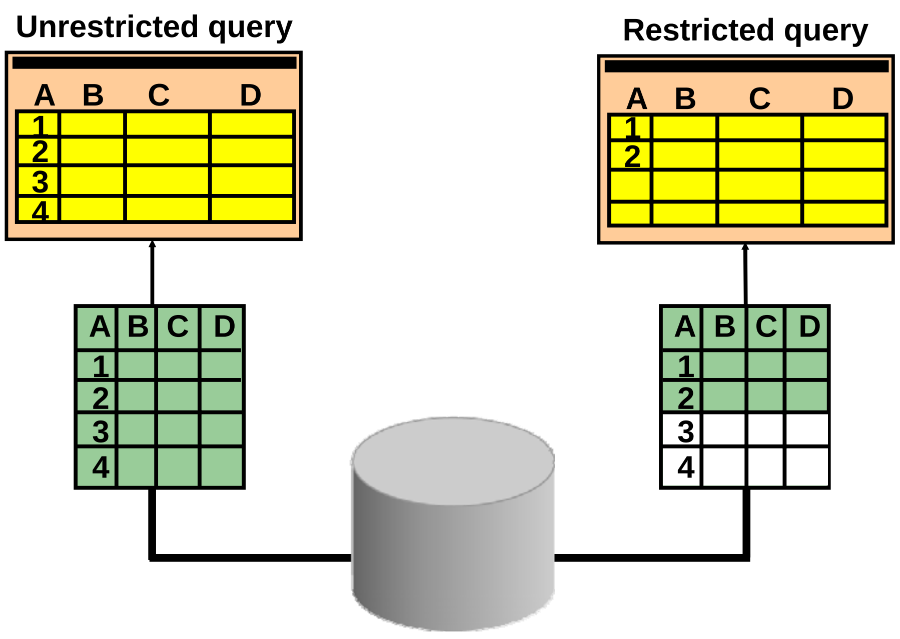
What you can do in Normal Mode

- Retrieve all records.
- Insert new records.
- Update records.
- Delete records.
- Commit (Save) records.
- Rollback (Clear) records.
- Navigate outside of the current data block.
- Exit the run-time session.

What you cannot do in Normal Mode

- Retrieve a restricted set of records.
- Invoke the Query/Where dialog box.

Retrieving Data



Retrieving Data

You can use a form module to retrieve information from the database without knowing any SQL syntax. However, if you are an experienced SQL user, you may want to supplement Oracle Forms Developer default processing with your own SQL predicates. There are two general types of queries:

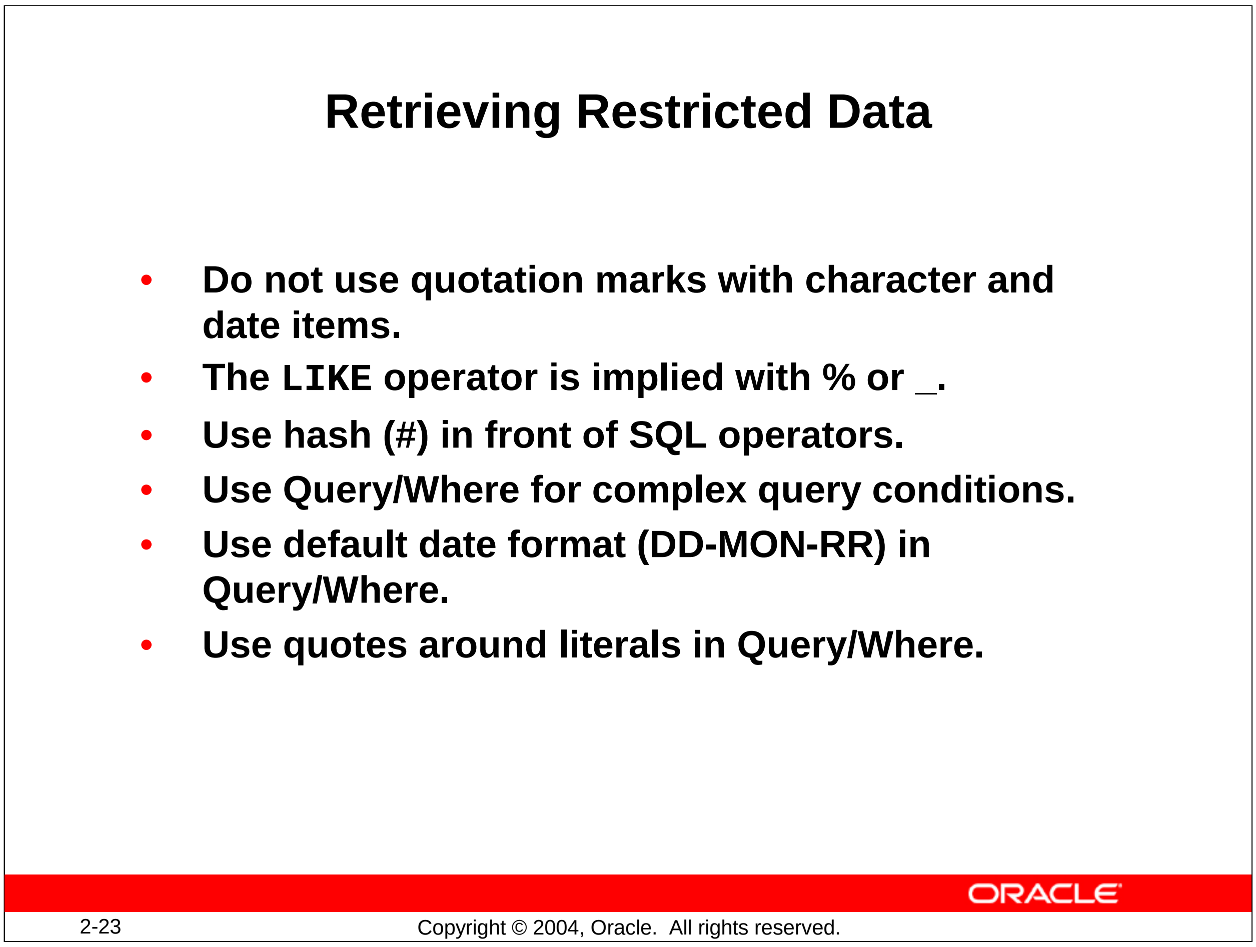
Query Type	**Description**
Unrestricted (global query)	The equivalent of selecting all the rows for all the represented columns from the base table for the queried data block
Restricted	The equivalent of selecting a restricted set of rows for all the represented columns from the base table for the queried data block

Performing an Unrestricted Query

You can retrieve unrestricted data by performing one of the following actions:

- Select Query > Execute.
- Press the appropriate function key.
- Click the Execute Query button.

Note: You cannot perform a query while you have unsaved updates, inserts, or deletes. Either save or undo the changes before you continue with the query.



Retrieving Restricted Data

- Do not use quotation marks with character and date items.
- The `LIKE` operator is implied with % or _.
- Use hash (#) in front of SQL operators.
- Use Query/Where for complex query conditions.
- Use default date format (DD-MON-RR) in Query/Where.
- Use quotes around literals in Query/Where.

2-23

Performing a Restricted Query

You can use any one of the following methods to perform a restricted query:

- Matching values
- Matching patterns (wildcards)
- A Query/Where dialog box for user entry of SQL predicates

Valid Search Criteria

Item	**Criterion**	**Uses**
Order ID	`2356`	Exact match
Customer ID	`%04`	Implied `LIKE` operator
Order ID	`#BETWEEN 2350 AND 2360`	`BETWEEN` operator
Order Status	“CREDIT order paid” selected from pop-up list	Exact match on item value (10)
Sales Rep ID	`:S`	Query Where dialog

Performing a Restricted Query (continued)

How to Perform a Restricted Query

You can perform a restricted query with the following steps:

1. Perform one of the following:
   - Select Query > Enter.
   - Click the Enter Query button.
   - Press the appropriate function key.
2. Enter-Query displays on the status line.
3. Enter search criteria into appropriate items.
4. Perform one of the following:
   - Select Query > Execute.
   - Click the Execute Query button.
   - Press the appropriate function key.

Note: Forms Builder constructs a select statement by using the AND operator for all specified conditions.

Instructor Note

The default date format is dependent on the language/territory. In Oracle8*i* Release 2 (8.1.6), the default date formats were modified, replacing ’YY’ with ’RR’ and ’YYYY’ with ’RRRR’. For example, if NLS_LANG is set to Finland, then the default date format will be DD.MM.RRRR.

Query/Where Dialog Box

- **Invoke by:**
  - **Entering :`variable_name`**
  - **Executing query**
- **Used to write:**
  - **Complex search conditions**
  - **Queries with `OR` predicates**
  - **`ORDER BY` clause**

2-25

Using the Query/Where Dialog Box

In the Query Where dialog box you can enter complex search criteria by using raw SQL. To use the Query/Where dialog box effectively requires knowledge of SQL. Use Query/Where to perform the following tasks:

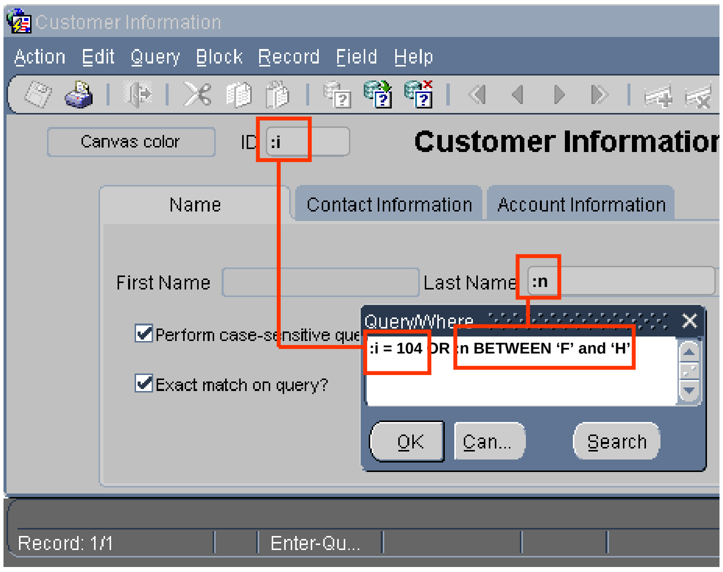
- Write complex search conditions.
- Write queries with OR predicates.
- Order the result of a query.

Note: Forms Builder logically uses the `AND` operator to append the Query/Where conditions to any other search criteria (including those imposed by the form designer) and constructs a `SELECT` statement. The Query/Where dialog box does not work in an item whose name differs from the name of its corresponding database column.

Example

1. To restrict the query to orders with a Sales Rep ID (:S) of 11 *or* an Order ID (:O) between 100 and 200, enter the following in the Query/Where dialog box:
   ```
   :S = 11 OR :O between 100 and 200
   ```
2. To sort the data by Sales rep ID (:S), enter the following in the Query/Where dialog box:
   ```
   ORDER BY :S
   ```

Query/Where Dialog Box



:i = 104 OR :n BETWEEN 'F' and 'H'

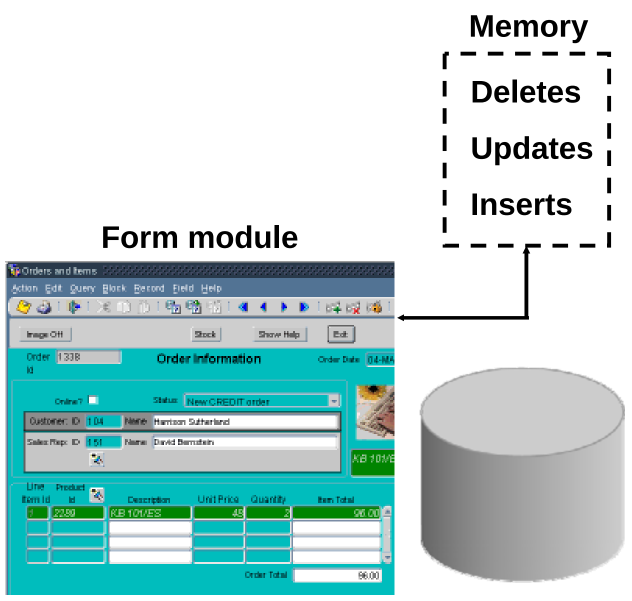
How to Use the Query/Where Dialog Box

To use the query/where box to enter query criteria:

1. Perform one of the following:
   - Select Query > Enter.
   - Click Enter Query.
   - Press the appropriate function key.
2. Enter a colon (:) followed by a unique character variable name in one or more items.
3. Perform one of the following: Select Query > Execute, click Execute Query, or press the appropriate function key.
   Note: You can select Query > Count Hits if you simply want to know how many records will match your criteria.
4. The Query/Where dialog box is displayed.
5. Enter the search criteria by using variables, SQL, and logical operators. Click OK.
   Note: To perform a query without any variables, type only the colon (:) and execute the query. Doing so also displays the Query/Where dialog box.

If you enter an `ORDER BY` at run time, it overrides any ordering defined by the designer.

Inserting, Updating, and Deleting



Inserting, Updating, and Deleting Records

Upon entering a typical form module you are in Normal mode. This means that Forms Builder regards anything that you type into a blank record as an insert and anything that you type over an existing record as an update.

How to Insert a Record

To insert a record, perform the following steps:

1. Ensure that you have the cursor positioned on a blank record by performing one of the following steps:
   - Scroll down until you find a blank record (always the last in the block).
   - Select Record > Insert.
   - Click Insert Record (green +).
   - Press the appropriate function key.
2. Enter the data into the relevant items.

Inserting, Updating, and Deleting Records (continued)

How to Update a Record

To update a record, perform the following steps:

1. Select Query > Enter.
2. Enter the search criteria to retrieve the appropriate record.
3. Select Query > Execute to retrieve all records that satisfy your specific search criteria.
4. Scroll through the records, stopping at the record to be updated.
5. Update the record.

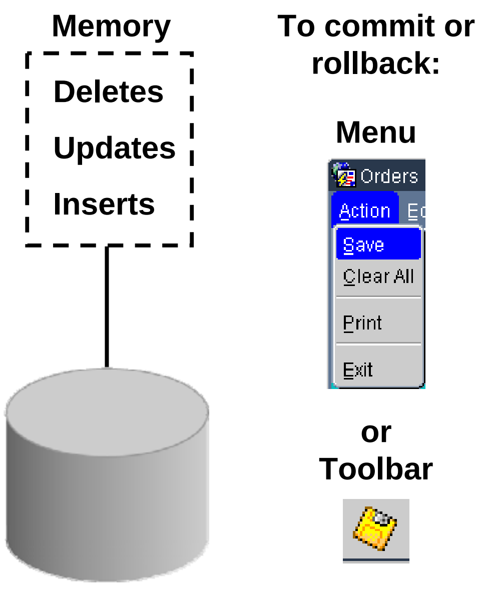
How to Delete a Record

To delete a record, perform the following steps:

1. Select Query > Enter.
2. Enter the search criteria to retrieve the appropriate record.
3. Select Query > Execute to retrieve all records that satisfy your specific search criteria.
4. Scroll through the records, stopping at the record to be deleted. Delete the record by taking one of the following actions:
   - Select Record > Remove to clear the record and mark it for deletion.
   - Click Remove Record (red X) to clear the record and mark it for deletion.
   - Press the appropriate function keys.

Making Changes Permanent

- Select Action > Save to make changes permanent.
- Select Action > Clear All to discard changes.



Making Inserts, Updates, and Deletes Permanent

To make any inserts, updates, or deletes permanent you must save (commit) them to the database. To do this, take one of the following actions:

- Select Action > Save.
- Click Save in the menu toolbar.

Discarding Inserts, Updates, and Deletes

To discard any inserts, updates, or deletes, you must clear the records (rollback) instead of saving. Perform a rollback by selecting Action > Clear All.

Exiting a Run-Time Session

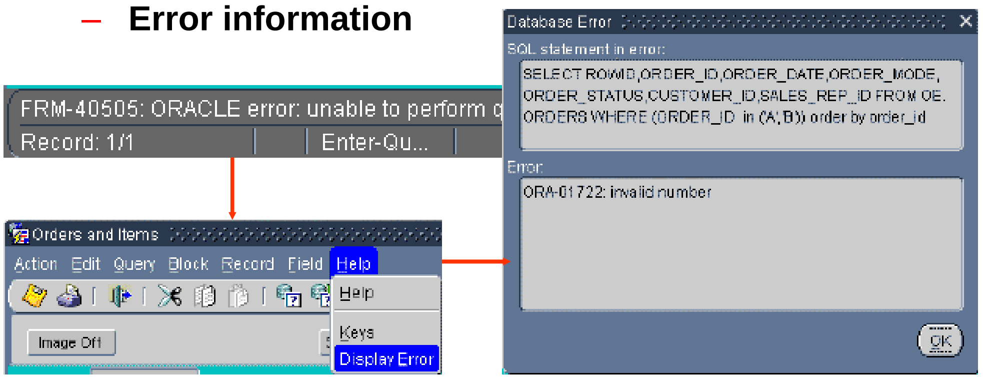
You exit the run-time session by taking one of the following actions:

- Select Action > Exit.
- Click Exit.
- Press the appropriate function keys.

Note: By default, you cannot exit the form while you have unsaved updates, inserts, or deletes. You need to either save or undo the changes before you can exit.

Displaying Errors

- Use to view Oracle errors
- Select Help > Display Error
- Shows Database Error window:
  - SQL statement
  - Error information



FRM-40505: ORACLE error: unable to perform q

Record: 1/1 | | Enter-Qu...

SELECT ROWID,ORDER_ID,ORDER_DATE,ORDER_MODE, ORDER_STATUS,CUSTOMER_ID,SALES_REP_ID FROM OE. ORDERS WHERE (ORDER_ID in ('A','B')) order by order_id

ORA-01722: invalid number

Displaying Errors

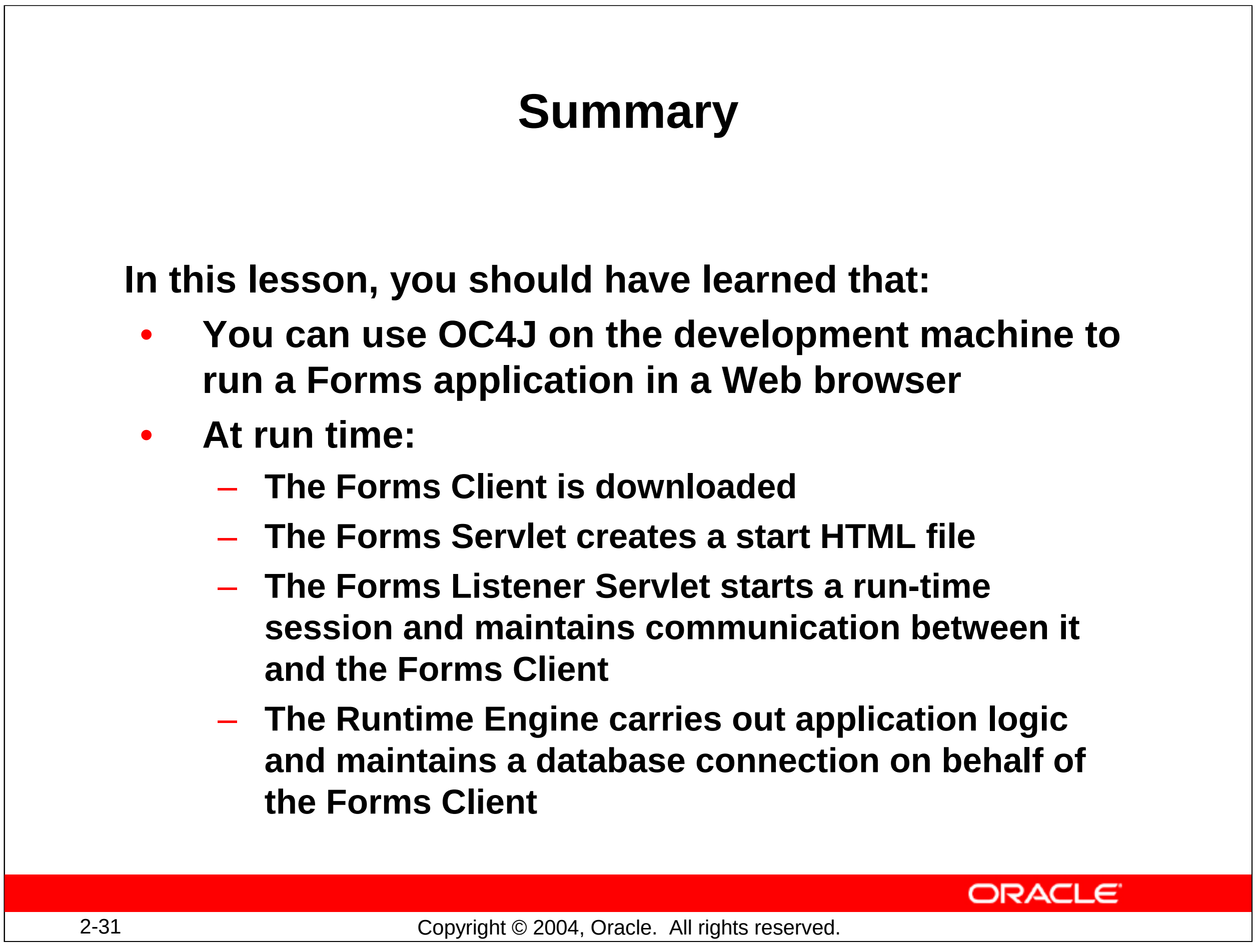
If an Oracle error is displayed on the message line while you are operating a form application, you can view the underlying SQL code by selecting Help > Display Error.

Example

Here is the SQL statement in error and its corresponding error:

```
SELECT order_id, order_date, order_mode, order_status,
       customer_id, sales_rep_id
FROM orders
WHERE (order_id in ('a','b'))
ORA-01722: invalid number
```

Note: Selecting Help > Display Error displays only those errors where the error on the message line is preceded by ORACLE error.



Summary

In this lesson, you should have learned that:

- **You can use OC4J on the development machine to run a Forms application in a Web browser**
- **At run time:**
  - **The Forms Client is downloaded**
  - **The Forms Servlet creates a start HTML file**
  - **The Forms Listener Servlet starts a run-time session and maintains communication between it and the Forms Client**
  - **The Runtime Engine carries out application logic and maintains a database connection on behalf of the Forms Client**

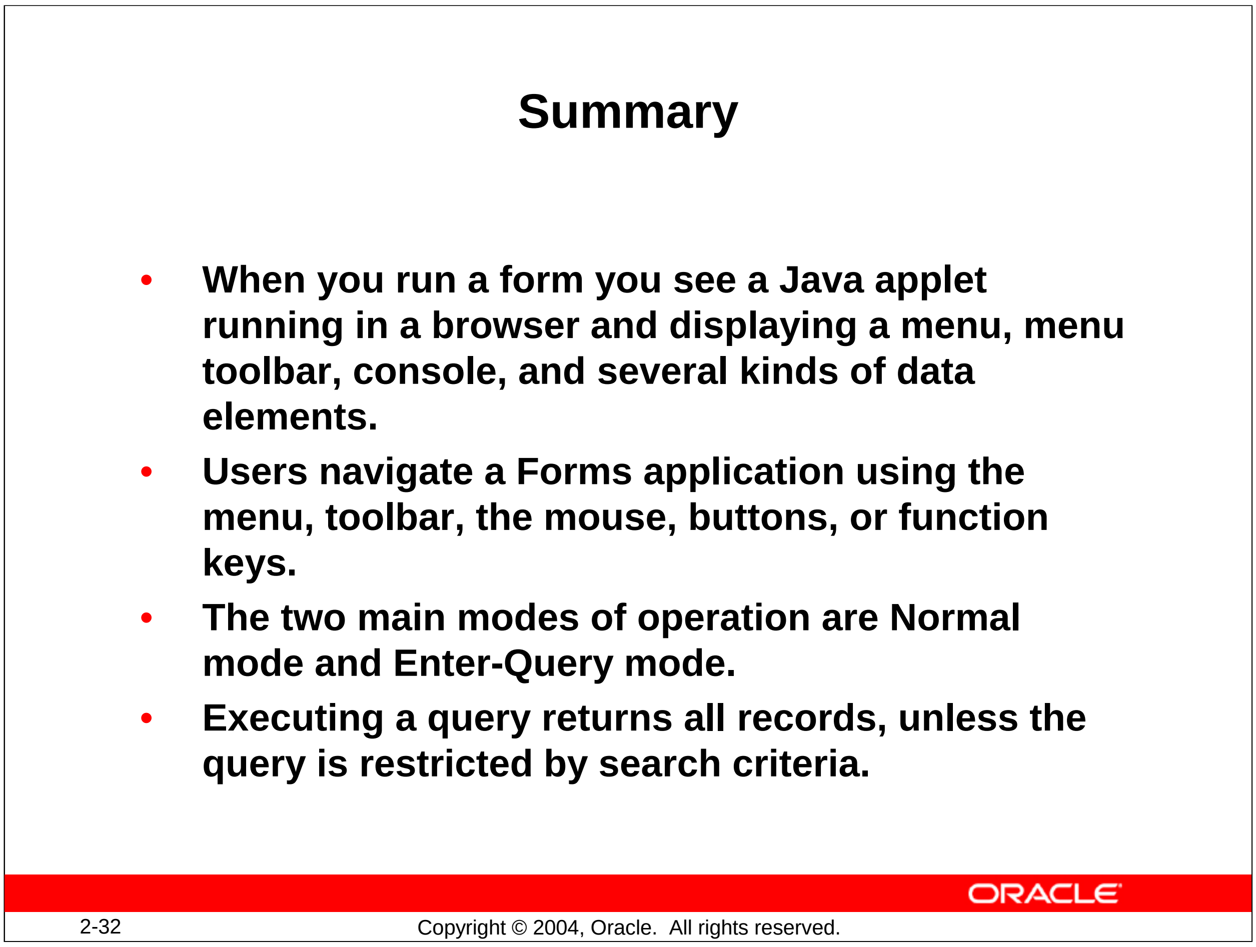
ORACLE

2-31

Summary

This lesson introduced the operator interface of Forms Builder. The following concepts were covered in this lesson:

- Starting OC4J on the development machine
- The run-time environment for Forms:
  - Running a form in a Browser
  - Starting a run-time session: The Forms Servlet, the Forms Client, and the Forms Listener Servlet
- The data elements of a form



Summary

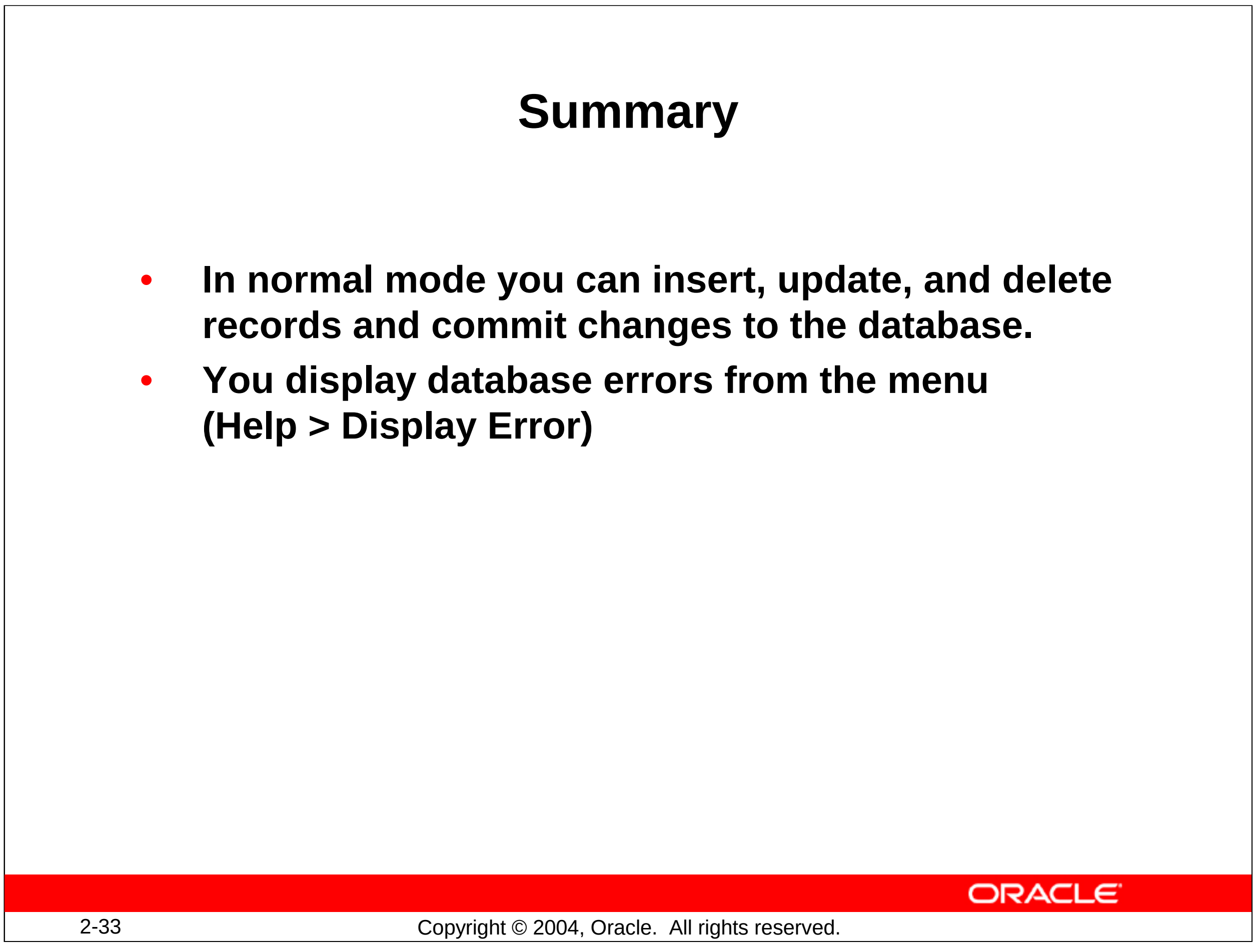
- When you run a form you see a Java applet running in a browser and displaying a menu, menu toolbar, console, and several kinds of data elements.
- Users navigate a Forms application using the menu, toolbar, the mouse, buttons, or function keys.
- The two main modes of operation are Normal mode and Enter-Query mode.
- Executing a query returns all records, unless the query is restricted by search criteria.

2-32

Summary (continued)

- Elements of a running form
- Navigation methods
- Modes of operation:
  - Normal mode
  - Enter-Query mode

    (There is also a Query mode that occurs when the form is fetching records; the operator cannot interact with the form in Query mode.)
- Retrieving data by performing:
  - Restricted queries—you supply search criteria
  - Unrestricted queries—you supply no search criteria



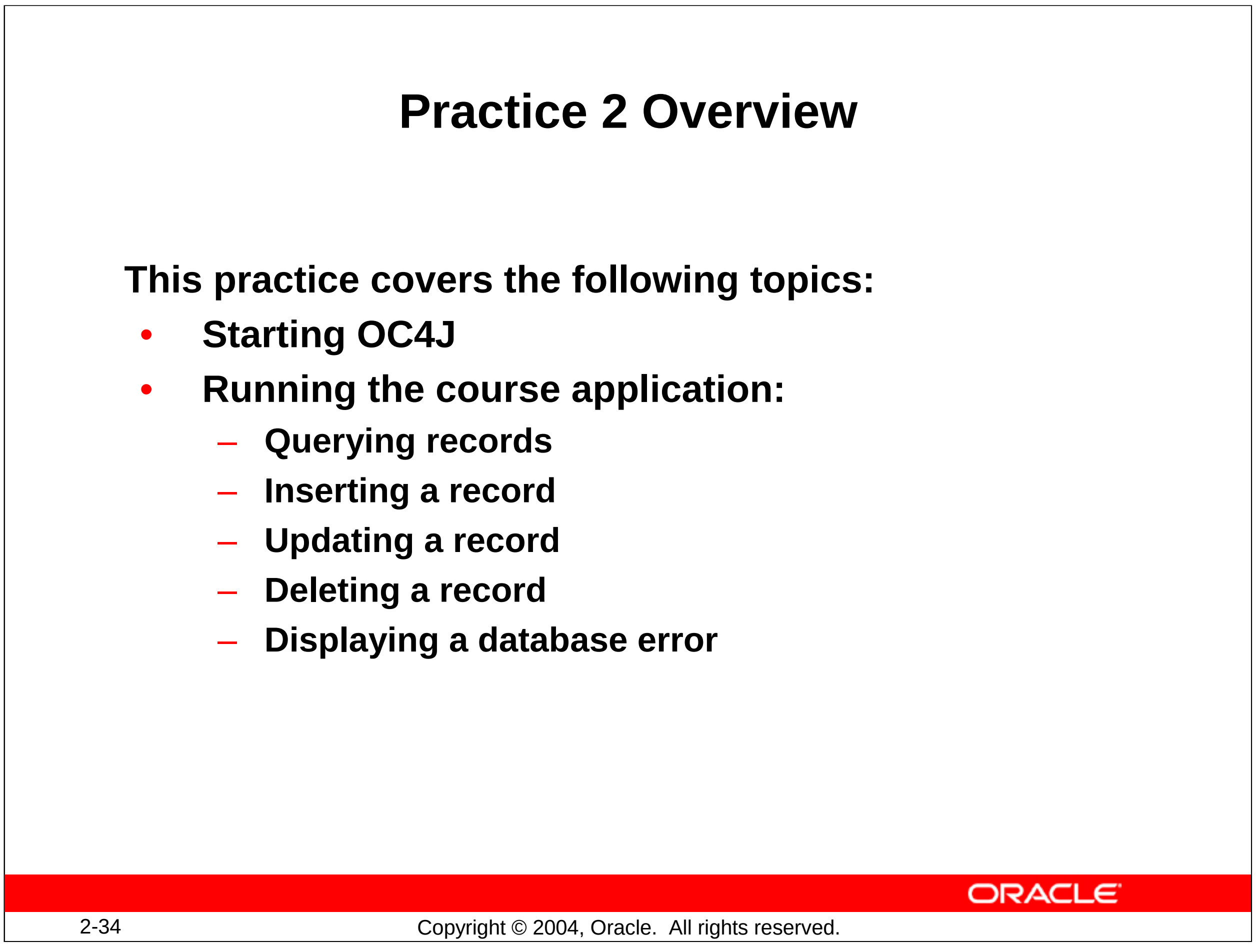
Summary

- In normal mode you can insert, update, and delete records and commit changes to the database.
- You display database errors from the menu (Help > Display Error)

2-33

Summary (continued)

- Inserting, updating, and deleting records
- Saving or clearing changes
- Displaying database errors



Practice 2 Overview

This practice covers the following topics:

- Starting OC4J
- Running the course application:
  - Querying records
  - Inserting a record
  - Updating a record
  - Deleting a record
  - Displaying a database error

2-34

Practice 2 Overview

In this practice session, you start OC4J to function as a Web server on your local machine. You run the course application in a browser and execute both unrestricted and restricted queries. You navigate through the application and use it to insert, update, and delete records.

Note: For solutions to this practice, see Practice 2 in Appendix A, “Practice Solutions.”

Instructor Note

Stress to students that they would normally be running an application that is deployed to a middle-tier application server. However, for this course the Oracle Application Server is not installed, so they will use OC4J on their local machines to run the application.

Practice 2

1. Start an instance of OC4J.
2. Invoke Internet Explorer and enter the following URL:
   http://<machine>:<port>/forms90/f90servlet?form=customers.fmx
   Your instructor will tell you the machine name and port number to use.
3. Select Help > Keys from the menu.
4. Click OK to close the Keys window. Browse through the records that were returned by the unrestricted query that executed automatically when the form started.
5. Execute a restricted query to retrieve information about the customer with the ID of 212.
6. Execute a restricted query to retrieve the customer whose first name is “Meenakshi”.
7. Try each of these restricted queries:
   a. Retrieve all cities starting with San.
   b. Retrieve all customers based in the USA with a low credit limit.
8. Display the customer details for Harrison Sutherland and click Orders to invoke the Orders form module.
9. Click Image Off and notice that the image item is no longer displayed. Click Image On and notice that the image item is displayed.
10. Query only those orders that were submitted online.
11. Move to the fourth record (Product ID 2322) in the Item block of Order 2355 and click Stock.
    The Inventory block is displayed in a separate window with stock information for that item.
12. Close the Stock Levels window. For the customer Harrison Sutherland, insert a new record in the ORDER block, as detailed below.
    Notice that some items are already populated with default values. Enter the following:

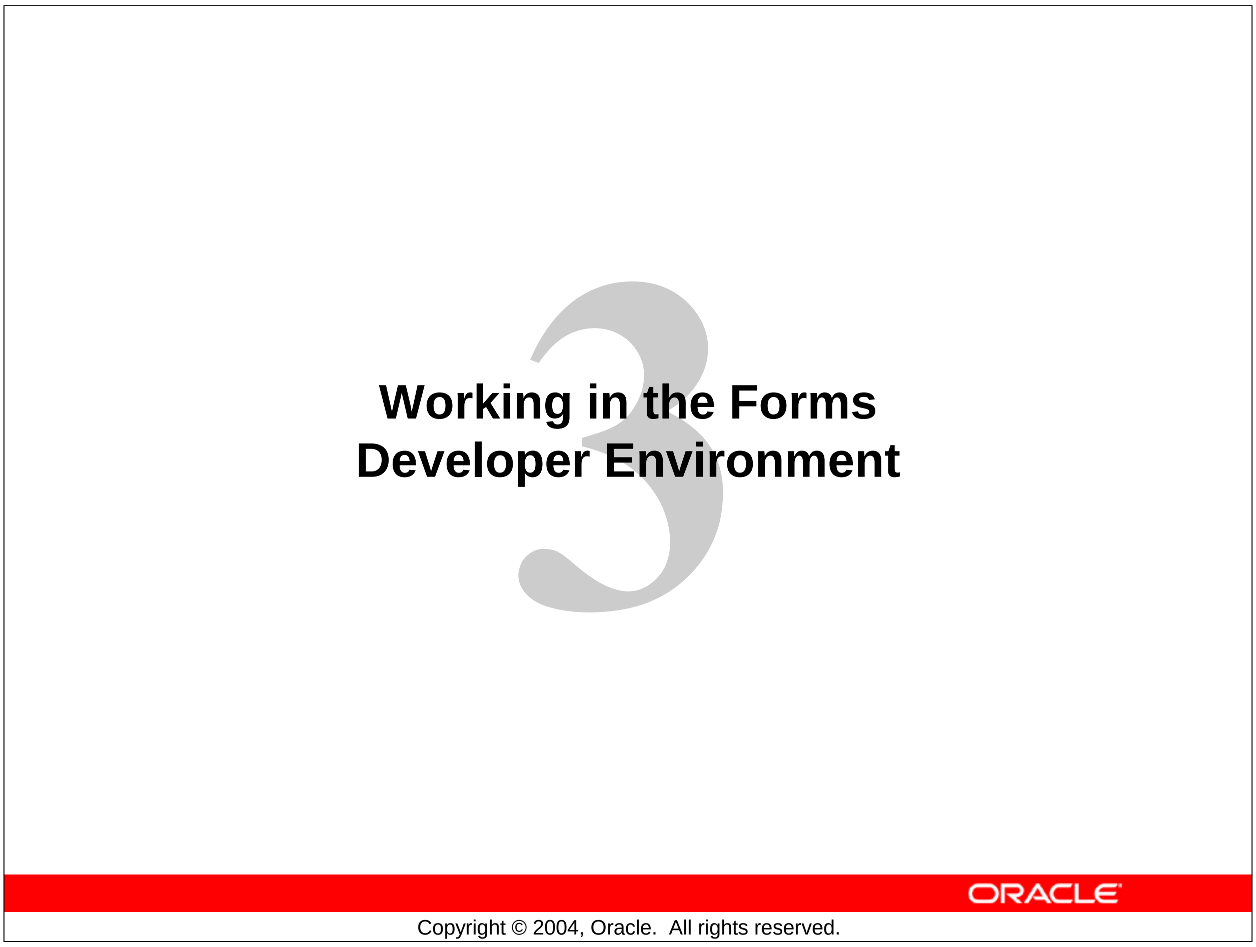
Item	**Value**
Online	Unchecked
Status	New Credit Order (poplist)
Sales Rep ID	151 (Enter the ID, or click the List button and select David Bernstein)

13. Insert a new record in the ITEM block with the following values:

Item	**Value**
Product ID	2289 (Enter the ID, or click the List button and select KB 101/ES.
Quantity	2

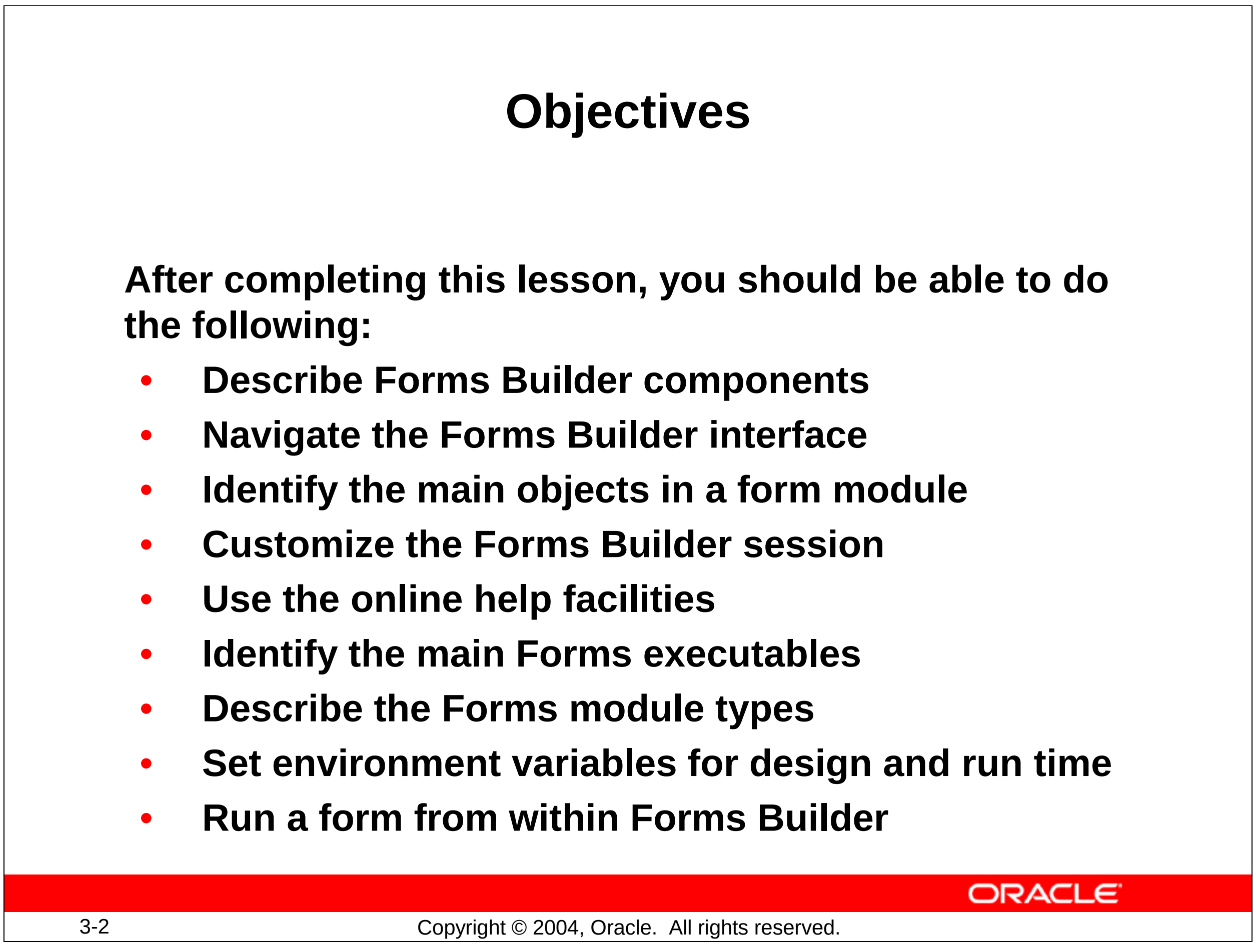
Practice 2 (continued)

14. Save the new records.
15. Update the order that you have just placed and save the change.
    Note: You may receive a message indicating that there are no changes to save. This message is generated by the Customers form, because both forms are saved at the same time. Changes to the Orders form should be saved successfully, so you can acknowledge the message and then ignore it.
16. Attempt to delete the order that you have just placed. What happens?
17. Delete the line item for your order and save the change.
18. Now attempt to delete your order and save the change.
19. Exit the run-time session and close the browser window.



Working in the Forms Developer Environment

Schedule:	Timing	Topic
	45 minutes	Lecture
	40 minutes	Practice
	85 minutes	Total



Objectives

After completing this lesson, you should be able to do the following:

- Describe Forms Builder components
- Navigate the Forms Builder interface
- Identify the main objects in a form module
- Customize the Forms Builder session
- Use the online help facilities
- Identify the main Forms executables
- Describe the Forms module types
- Set environment variables for design and run time
- Run a form from within Forms Builder

3-2

Introduction

Overview

This lesson provides you with an overview of Forms Builder, including a high-level description of its components and object hierarchy. Using this knowledge, you can plan and implement the structure of your form applications.

Forms Builder Key Features

With Forms Builder you can:

- **Provide an interface for users to insert, update, delete, and query data**
- **Present data as text, image, and custom controls**
- **Control forms across several windows and database transactions**
- **Use integrated menus**
- **Send data to Oracle Reports**

Forms Builder Key Features

Forms Builder is a major component of Oracle Forms Developer. You can use Forms Builder to quickly develop form-based applications for presenting and manipulating data in a variety of ways.

Users of Forms Builder applications can:

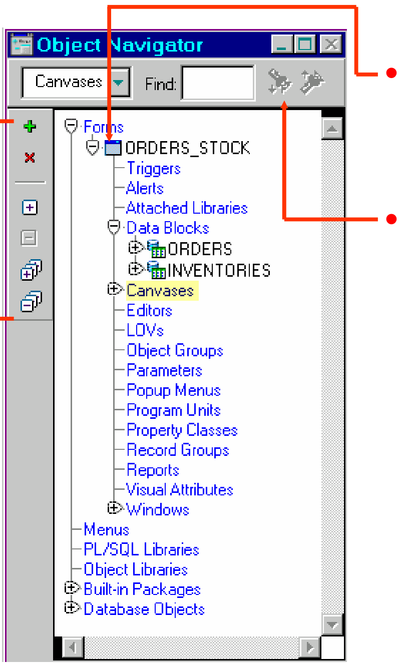
- Insert, update, delete, and query data by using a variety of interface items
- Present data by using text, image, and custom controls, including JavaBeans and Pluggable Java Components
- Control forms across several windows and database transactions
- Access comprehensive facilities by using integrated menus
- Send data directly to Oracle Reports

As the designer of Forms Builder applications, you can:

- Design forms that use a number of data sources, including Oracle databases
- Build applications quickly and easily by using powerful GUI development tools
- Design applications for Internet deployment
- Copy and move objects and their properties easily between applications
- Use design features such as wizards, the Layout Editor, Object Navigator, and PL/SQL Editor

Forms Builder Components: Object Navigator

- Client-side and server-side objects displayed hierarchically
- Toolbar to create, delete or unload, expand or contract



- Icons to represent objects
- Fast search feature

Forms Builder Components: Object Navigator

The interface components of the Forms Builder tool help to provide the flexibility and productivity of the Oracle Forms Developer development environment.

The Object Navigator is a hierarchical browsing and editing interface. You can use the Object Navigator to locate and manipulate application objects quickly and easily. Features include:

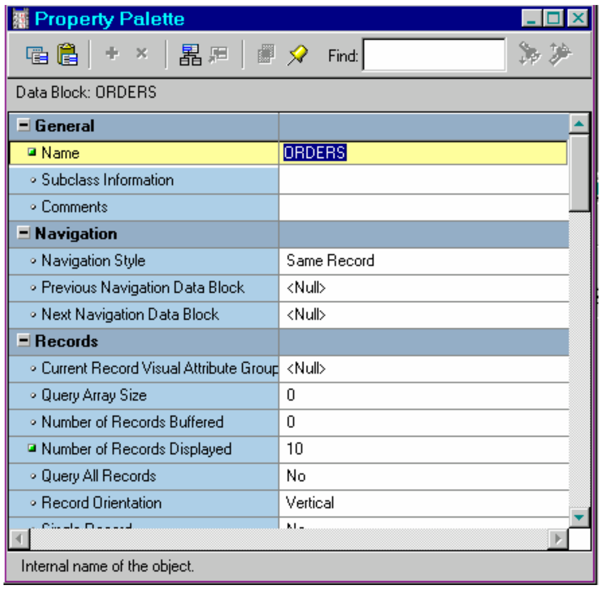
- A hierarchy represented by indentation and expandable nodes
  (Top-level nodes show module types, database objects, and built-in packages. All other nodes and the objects they contain are indented to indicate that they belong to these higher-level nodes.)
- Find field and icons, enabling forward and backward searches for any level of node or for an individual item in a node
- Icons in the vertical toolbar replicating common Edit and View menu functions
- An icon next to each object to indicate the object type

Instructor Note

Mention these briefly. You can start Forms Builder to demonstrate each component as you discuss it. Open an existing form, and briefly explain the features of the Object Navigator.

Forms Builder Components: Property Palette

- Copy and paste properties
- Fast search feature



Property Palette

Find:

Data Block: ORDERS

General	
Name	ORDERS
Subclass Information	
Comments	
Navigation	
Navigation Style	Same Record
Previous Navigation Data Block	<Null>
Next Navigation Data Block	<Null>
Records	
Current Record Visual Attribute Group	<Null>
Query Array Size	0
Number of Records Buffered	0
Number of Records Displayed	10
Query All Records	No
Record Orientation	Vertical

Internal name of the object.

ORACLE

Forms Builder Components: Property Palette

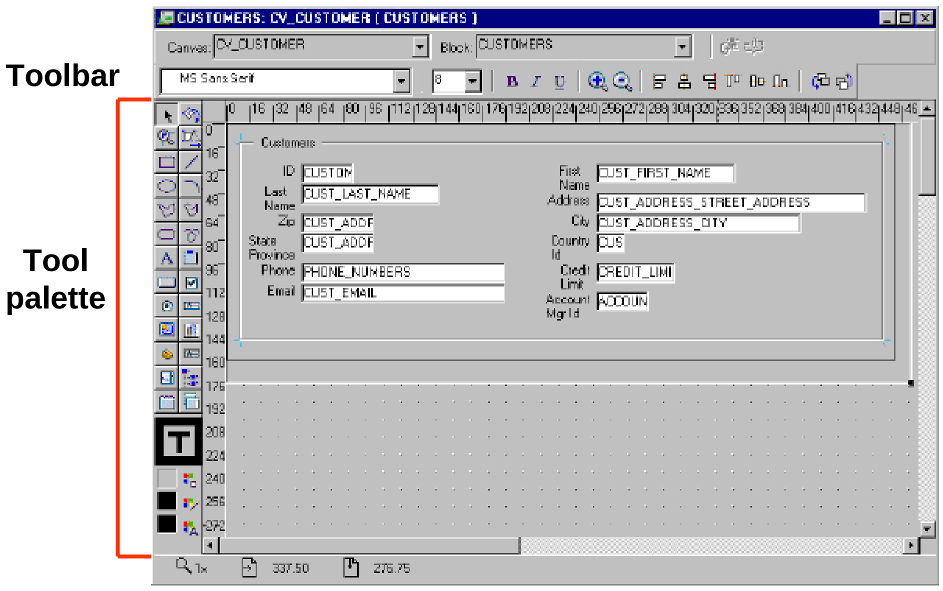
All objects in a module, including the module itself, have properties that you can see and modify in the Property Palette. Features include:

- Copy and reuse properties from another object
- Find field and icons, similar to Object Navigator

Instructor Note

Demonstration: Open the Property Palette for one of the objects in the form and point out its features. Click a different object to show that the Property Palette changes to the properties of the new object.

Forms Builder Components: Layout Editor



Forms Builder Components: Layout Editor

The Layout Editor (or Layout Model) is a graphical design facility for creating and arranging interface items and graphical objects in your application. You can use the Tool palette and the toolbar available in the Layout Editor to design the style, color, size, and arrangement of visual objects in the application. The layout can include graphical objects and images.

Instructor Note

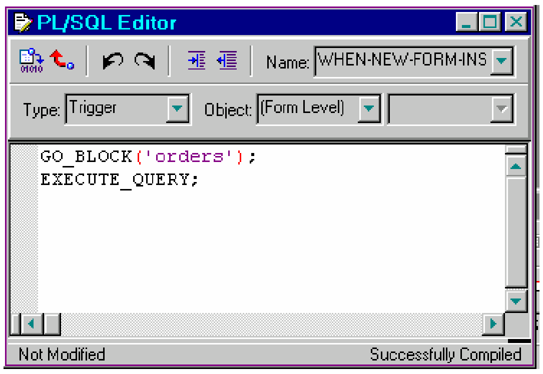
Demonstration

- Go to the Layout Editor. Point out the major tools in the toolbar. Demonstrate some of the major tools in the Tool palette.
- Draw a rectangle and an ellipse, and also show how to draw a square and a circle by holding down the [Shift] key when using the rectangle and ellipse tools.
- Demonstrate selecting multiple objects and show how to group various objects.
- Demonstrate resizing, moving, aligning, coloring, and deleting objects.
- Demonstrate using the Text tool and changing the font style and size.
- Demonstrate how to pin a tool by double-clicking a tool.

Forms Builder Components: PL/SQL Editor

With the PL/SQL Editor you can:

- **Use PL/SQL in Forms.**
- **Enter and compile code.**

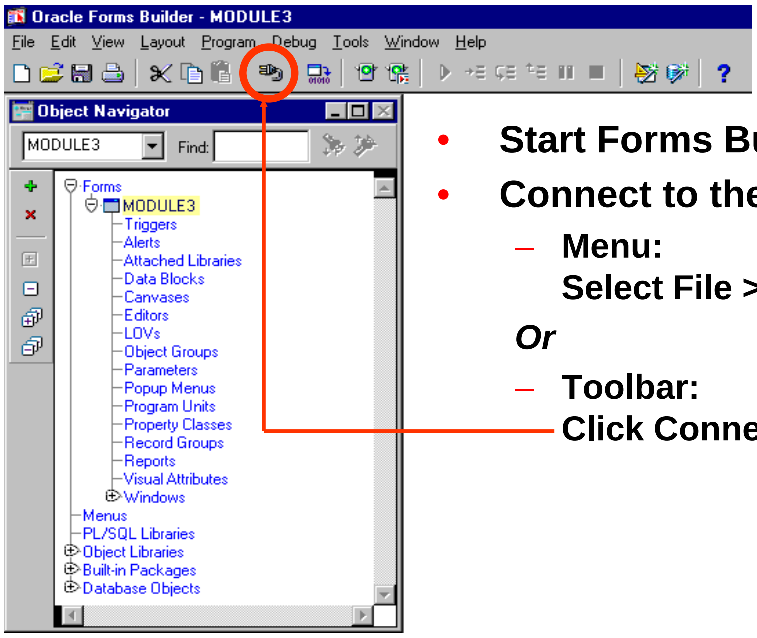


```
GO_BLOCK('orders');
EXECUTE_QUERY;
```

Forms Builder Components: PL/SQL Editor

The PL/SQL Editor enables you to incorporate PL/SQL code objects into your form. Code objects in Forms Developer include event triggers, subprograms (functions and procedures), menu item commands, menu startup code, and packages. You enter and compile code in the PL/SQL Editor. You will learn more about the PL/SQL Editor in later lessons when you use it to code triggers in Forms Builder.

Getting Started in the Forms Builder Interface



- **Start Forms Builder**
- **Connect to the database:**
  - **Menu:**
    Select File > Connect

  Or

  - **Toolbar:**
    Click Connect

Getting Started in the Forms Builder Interface

Starting Forms Builder

To start Forms Builder, invoke it from the Windows Start menu: Programs > Oracle Developer Suite – iDS10*g* > Forms Developer > Forms Builder.

When you invoke Forms Builder, you first see a Welcome dialog box. If you click Cancel to dismiss the dialog box, you see the Object Navigator and an empty new module.

If you build applications that access database objects, you must connect to a database account from the Forms Builder. Connect to a database if you need to:

- Compile code that contains SQL
- Access database objects in the Object Navigator
- Create Oracle Forms Developer objects that are based on database objects

Getting Started in the Forms Builder Interface (continued)

How to Connect to Oracle

1. Select File > Connect from the menu, or click the Connect icon in the toolbar.
2. Enter the database user and password in the Connect dialog box. If not connecting to the default database, also provide the necessary connect string or database alias.
   Note: Oracle Forms Developer automatically displays the Connect dialog box if you try to perform a task that requires connection.

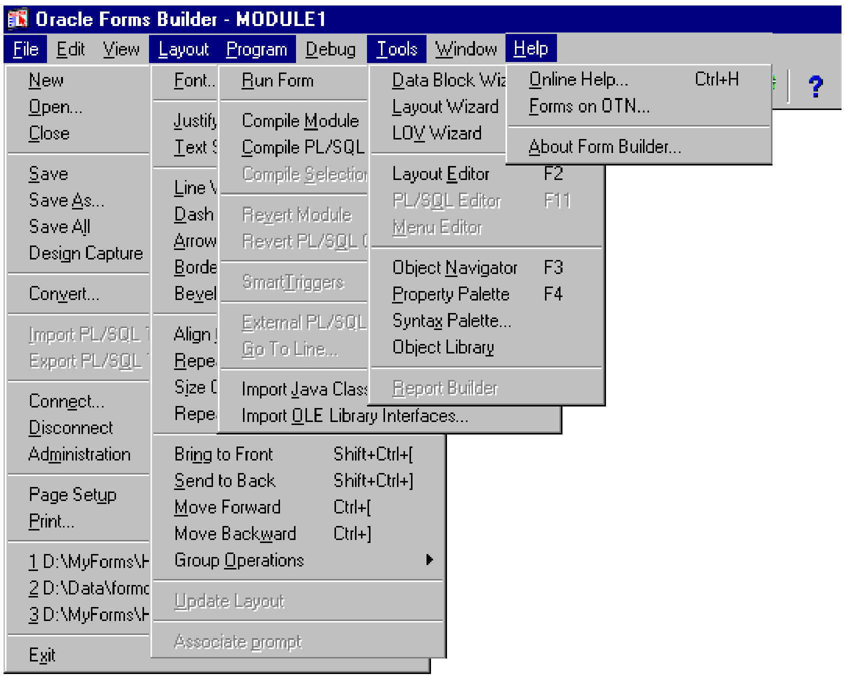
Instructor Note

- Invoke Forms Builder.
- Open an existing form. Use one of the forms from the existing prebuilt application.

In this lesson, an existing form is used to familiarize students with the Oracle Forms Developer interface. Later students create new forms by using the wizards.

Forms Builder will probably prompt you to connect to the database when it tries to run the form. Also show students the File > Connect menu option.

Forms Builder: Menu Structure



Navigating the Forms Builder Main Menu

The Forms Builder main menu contains options to enable you to create, modify, and manage your form modules.

Common Menu Features

The following table describes some common features in GUI menus:

Feature	Description
Underline	Shortcut key: [Alt] + letter
Ellipsis (...)	Additional input, usually by using a dialog box
▸	Menu option has a submenu
Windows menu	List of open windows; select any window to make it active
Help	List of help facilities: Online Help, Forms on OTN, About box

Native GUI Interface

The menu shown in the slide depicts the Windows NT environment. However, menus appear with the same look and feel of your native GUI interface.

For example, in Motif, the Windows Print Dialog options appear as submenus of the Font menu.

Navigating the Forms Builder Main Menu (continued)

Forms Builder Main Menu

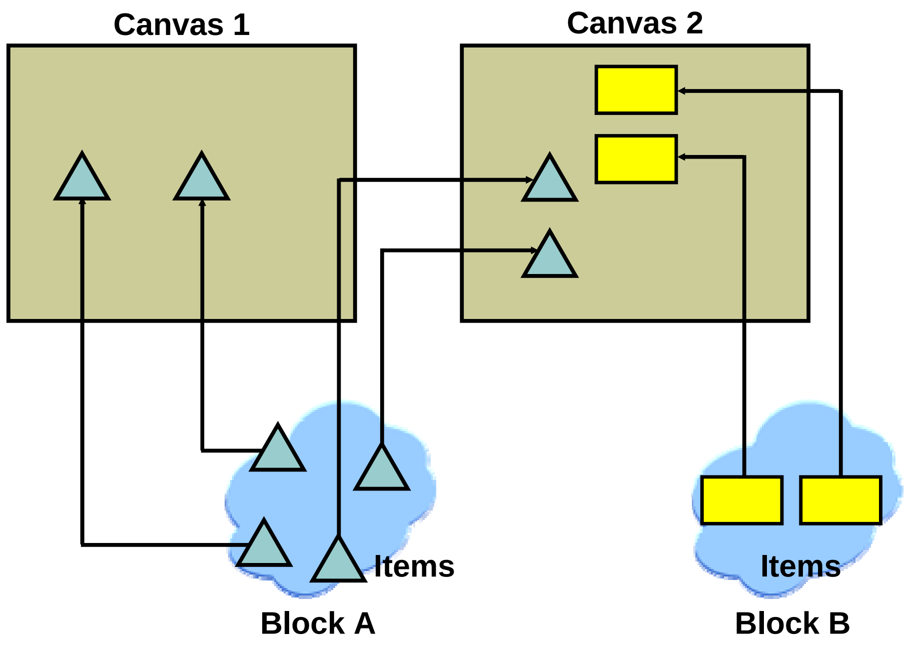
Menu Item	Description
File	Common file utilities, such as open, save, connect, administration
Edit	Cut, copy, paste, create, preferences, and so on
View	Switch view in current window; options vary depending on context
Layout	Common commands for use in Layout Editor
Program	Includes compilation and commands related to code
Debug	Invokes debugger functionality
Tools	Access to wizards and other Forms Builder components

Instructor Note

Show the Main menu options, and how they differ in context. For example, show how different Layout menu options are enabled in the Layout Editor when objects are selected, and how these menu options are not enabled in the Object Navigator.

Do not describe Help in detail; there are more details for these options later in this lesson.

Blocks, Items, and Canvases



Forms Builder Components

Form modules make up the main "body" of an Oracle Forms Developer application. They can consist of many types of objects, some of which are visible to the user at run time.

The three major objects in a form are:

- **Items:** These are interface objects that present data values to the user or enable the user to interact with the form, depending upon the item type. There are several different types of items. Items are logically grouped into blocks and visibly arranged on canvases.
- **Blocks:** A block is the intermediate building unit for forms. Each form consists of one or more blocks. A block is the logical owner of items, and each item in a form belongs to a block. Items in one block are logically related; for example, they may correspond to columns in the same database table or may need to be part of the same navigation cycle.
  Blocks therefore provide a mechanism for grouping related items into a functional unit for storing, displaying, and manipulating records.

Forms Builder Components (continued)

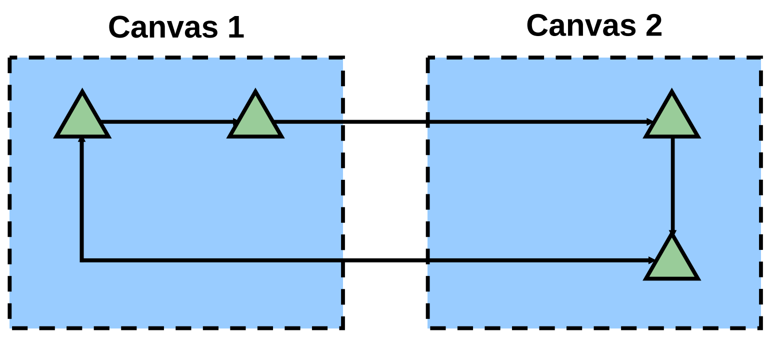
- **Canvases:** A canvas is a “surface” where visual objects, such as graphics and items, are arranged. A form module can have several canvases (like the pages of a paper form). A canvas can display items from one or more blocks. To see a canvas and its items, you must display the canvas in a window. By default, all canvases in a form appear in the same window (which could mean you see only one canvas at a time), but you can assign separate windows for each canvas so that several canvases can be viewed at once.

Note: Items in one block do not need to be physically grouped. They can span many canvases (and windows).

Instructor Note

A canvas is like a picture portrait, and a window is like a picture frame. Just as you need a picture frame to display a picture portrait, you need a window to display a canvas and its contents.

Navigation in a Block



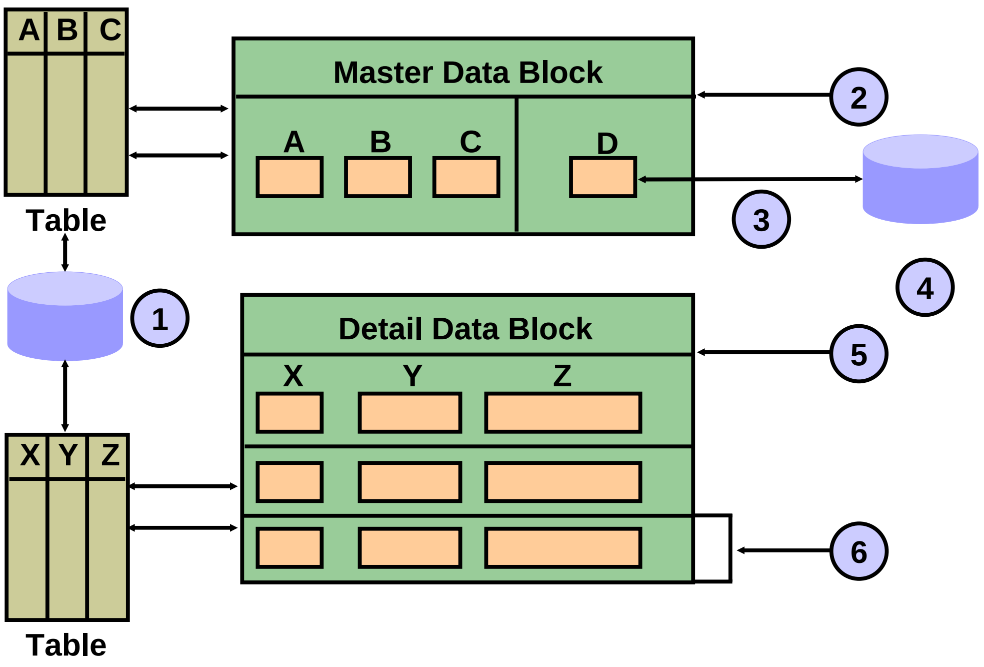
Navigation in a Form Module

When you run a form, you principally navigate by way of items and blocks, not by canvases. Each item has a sequenced position within its block, and each block has a sequenced position in the form.

When a user requests to move to the next item in a block, focus will be set on the next item in sequence, wherever that may be. If the next item is on a different canvas, Oracle Forms Developer displays that canvas automatically. Similarly, users can request to move to the next block (or previous block). If the first item in this block resides on another canvas, then that canvas is displayed automatically.

Of course, if you can already see the item that you want to move to, then you may click on it directly with the mouse. You can also program mechanisms into the application to enable navigation in other ways.

Data Blocks



Types of Blocks

In Forms Builder there are two main types of blocks: data blocks and control blocks.

Data Blocks

When you build database applications with Forms Builder, many of the blocks will be data blocks. A data block is associated with a specific database table (or view), a stored procedure, a `FROM` clause query, or transactional triggers.

If it is based on a table (or view), the data block can be based on only one base table, even though the data block can be programmed to access data from more than one table and data sources. By default, the association between a data block and the database enables the user to automatically access and manipulate data in the database. However, to access data from other tables (nonbase tables), you need to write triggers.

1	Base table source
2	Single-record data block
3	Trigger access
4	Nonbase table source
5	Multirecord data block
6	Record

Types of Blocks (continued)

Data Blocks (continued)

For a base table, Forms Builder can automatically perform the following actions:

- Creates items in the data block to correspond to columns in the table (These items are data items or base table items.)
- Produces code in the form to employ the rules of the table's constraints
- Generates SQL at run time (implicit SQL) to insert, update, delete, and query rows in the base table, based upon the user's actions

At run time, you can use standard function keys, buttons, menu options, or standard toolbar options to initiate query, insert, update, or delete operations on base tables, and the subsequent commit of the transaction.

Control Blocks

A control block is not associated with a database, and its items do not relate to any columns within any database table. Its items are called control items. For example, you can create many buttons in your module to initiate certain actions and to logically group these buttons in a control block.

Master Versus Detail Blocks

To support the relationship between data blocks and their underlying base tables, you can define one data block as the detail (child) of a master (parent) data block. This links primary key and foreign key values across data blocks, and synchronizes the data that these data blocks display.

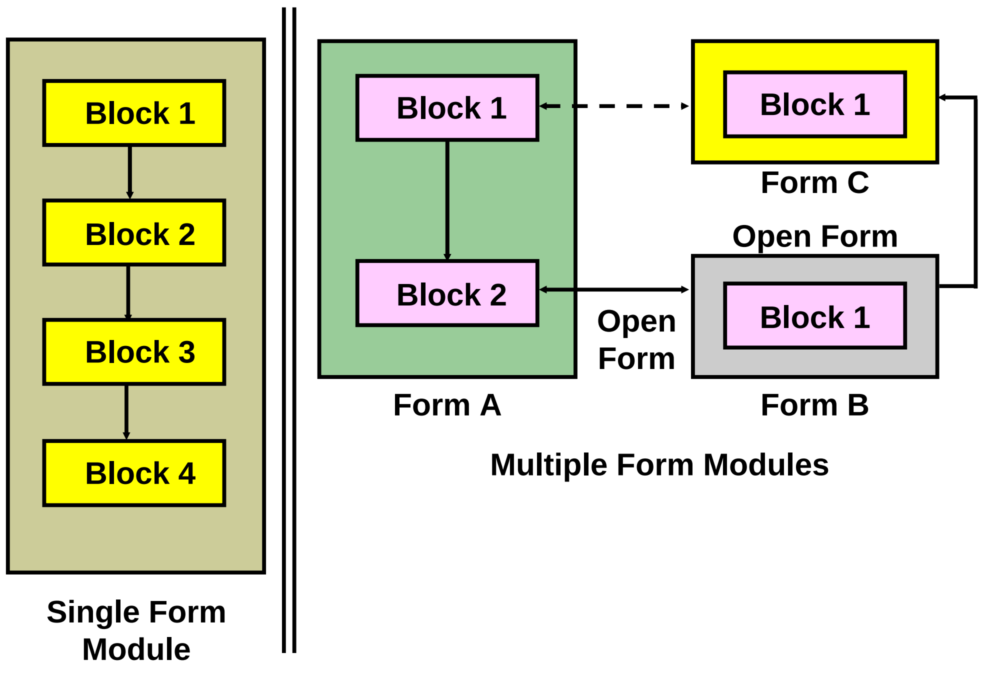
Forms Builder automatically generates the objects and code needed to support master-detail relationships. As the designer, you need only request it.

Note: If your application requires it, you can also create independent data blocks in which there is no relationship between the two data blocks.

Single-Record Versus Multirecord Blocks

You can design a data block to show one record at a time (single-record block) or several records at once (multirecord block). Usually, you create a single-record data block to show master block data and a multirecord data block to show detail block data. In either case, records in a data block that are currently not visible on the screen are stored in a block buffer.

Forms and Data Blocks



Multi-Block and Multi-Form Applications

Typically, a Forms Builder application consists of more than one data block. With more than one data block, you can do the following:

- Separate the navigation cycle of one group of items from another
- Map each data block to a different database table (You can have one base table per data block.)
- Produce a master-detail form, with a master data block and corresponding detail data blocks that are related to the master

You can create a large form module with many data blocks. Alternatively, you can create several smaller form modules with fewer data blocks in each.

Generally, a modular application with several smaller form modules has the following characteristics:

- Modules are loaded only when their components are required, thus conserving memory.
- Maintenance can occur on one module without regenerating or loading the others.
- Forms can call upon one another, as required.

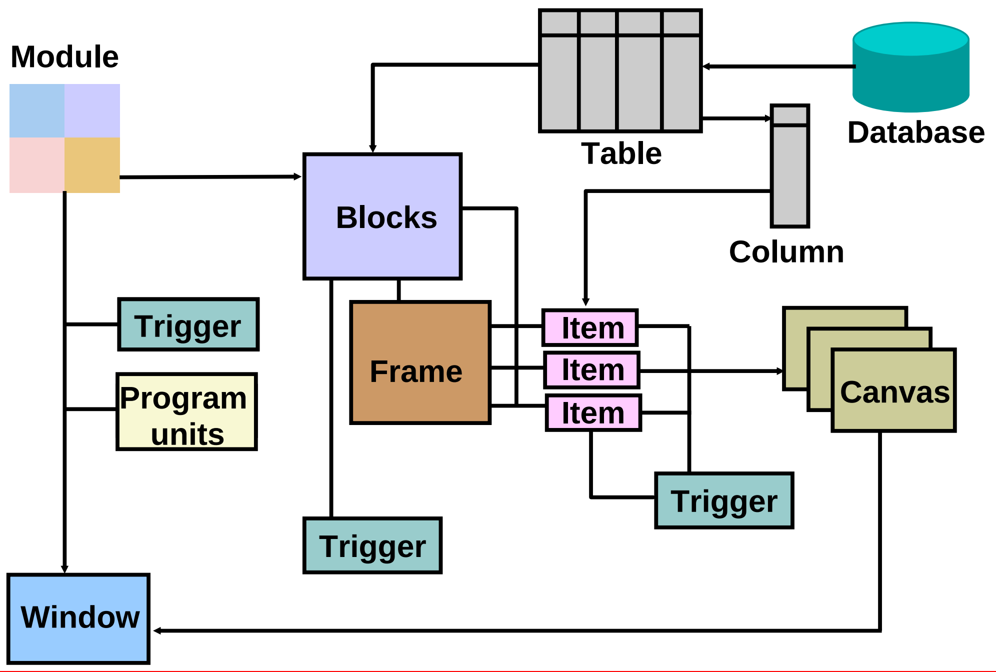
Multi-Block and Multi-Form Applications (continued)

- Parallel development can be carried out by different team members on different components.

Here are some points to consider when grouping data blocks in the application:

Data Blocks in the Same Form Module	**Data Blocks in Different Form Modules**
The data blocks can be directly linked in master-detail relationships.	The data blocks cannot be linked by the standard interblock relations.
Navigation between data blocks is handled by default functionality.	Navigation between data blocks of different forms is programmed by the designer (although mouse navigation to visible items can be automatic).

Form Module Hierarchy



Technical Note

A form module is made up of one or more blocks. A data block is based on a database object, such as a table or a view. A data block can contain both data items and control items. A frame can be created to arrange data block items. Each item in a block must appear on a canvas, and each canvas must appear in a window. A form module can have one or more canvases and windows.

Using triggers, you can add functionality to your form. Triggers can be written at different levels in a form module. User-named program units enable you to write additional PL/SQL code through procedures, functions, and packages.

The Object Hierarchy

You can create many types of objects in a form module. They are discussed in more detail in later lessons.

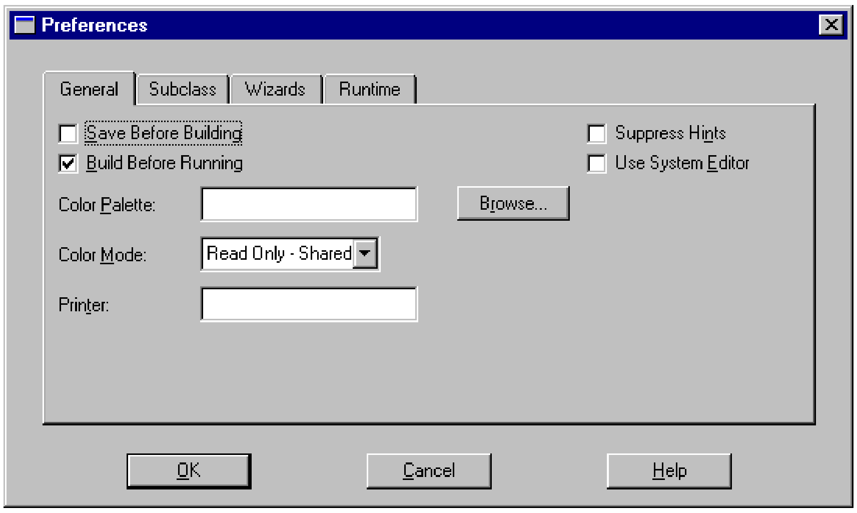
In the following table, note that some objects are associated, even though one might not be “owned” by the other.

Object	**Description**
Block	Logical section of a form; owned by the form module
Item	Member of a data block (items are functionally grouped into records)
Canvas	The surface where visual objects are arranged; owned by the form module; can contain text and graphics—static information that the user cannot interact with.
Window	Produced to contain the views of canvases; owned by the form module
Frame	A graphic object that appears on a canvas, is owned by that canvas, and is used to arrange the items within a data block
User-named program unit	Named procedure, function or package owned by the form module
Trigger	PL/SQL block executed on an event; depending on scope, can be owned by the form module, a data block, or an item
Other objects	Mainly owned by the form module itself; include alerts, parameters, and record groups

Instructor Note

Point out the above objects to the students, using the Object Navigator.

Customizing Your Forms Builder Session



☐ Save Before Building
☑ Build Before Running
☐ Suppress Hints
☐ Use System Editor

Color Palette:	
Color Mode:	Read Only - Shared
Printer:	

Customizing Your Forms Builder Session

What Are Oracle Forms Developer Preferences?

You can use preferences to customize some aspects of your Forms Builder session.

Forms Builder Preferences

There are four tabs in the Preferences dialog box.

To see a description of each preference, click Help in the Preferences dialog or press the Help key (`[F1]` for Windows NT/95).

In addition to session preferences, you can also set run-time settings that apply to running your form from within the builder.

To modify preferences, perform the following steps:

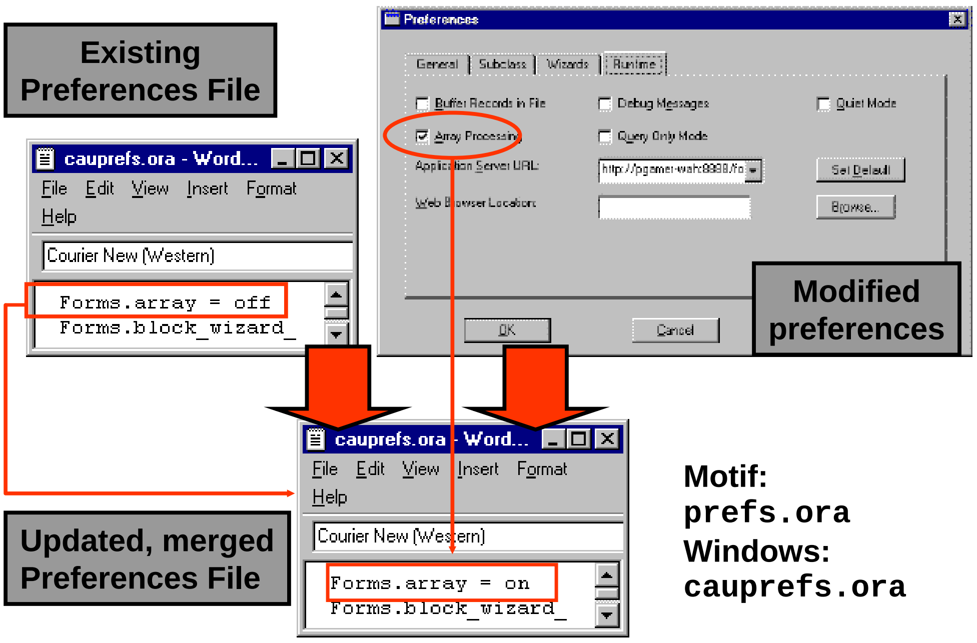
1. Select Edit > Preferences.
2. Specify any options that you require.
3. Click OK to save changes, or Cancel to cancel changes.

Customizing Your Forms Builder Session (continued)

You can specify several related preferences on each of the Preference tabs. The table below describes one preference on each of the tabs:

Tab	Preference Name	Description
General	Build Before Running	Determines whether Forms Builder automatically compiles the active module when you run a form. This option enables you to avoid issuing separate Compile and Run commands each time you modify and run a form.
Subclass	Subclassing Path	Options for keeping or removing the subclassing path
Wizards	Welcome Dialog	Check box to suppress or display the first Welcome dialog box. There are several similar check boxes.
Runtime	Array Processing	Determines whether Forms Builder processes groups of records at a time, reducing network traffic and increasing performance.

Saving Preferences



Existing file:

```
Forms.array = off
Forms.block_wizard_
```

Modified preferences: [✓ Array Processing]

Updated, merged file:

```
Forms.array = on
Forms.block_wizard_
```

Motif:
`prefs.ora`
Windows:
`cauprefs.ora`

Saving Preferences

When you click OK in the Preferences dialog box, Oracle Forms Developer updates your current session with the changes.

When you exit the builder (File > Exit), Oracle Forms Developer writes the changes to a preference file for future sessions. The name of the preference file varies on different platforms.

Oracle Forms Developer and Oracle Reports Developer share the same preference file. If the preference file already exists, Oracle Forms Developer merges its changes with the existing file. This does not affect preferences for Reports.

Each option in the preference file is prefixed by the tool name to which it belongs.

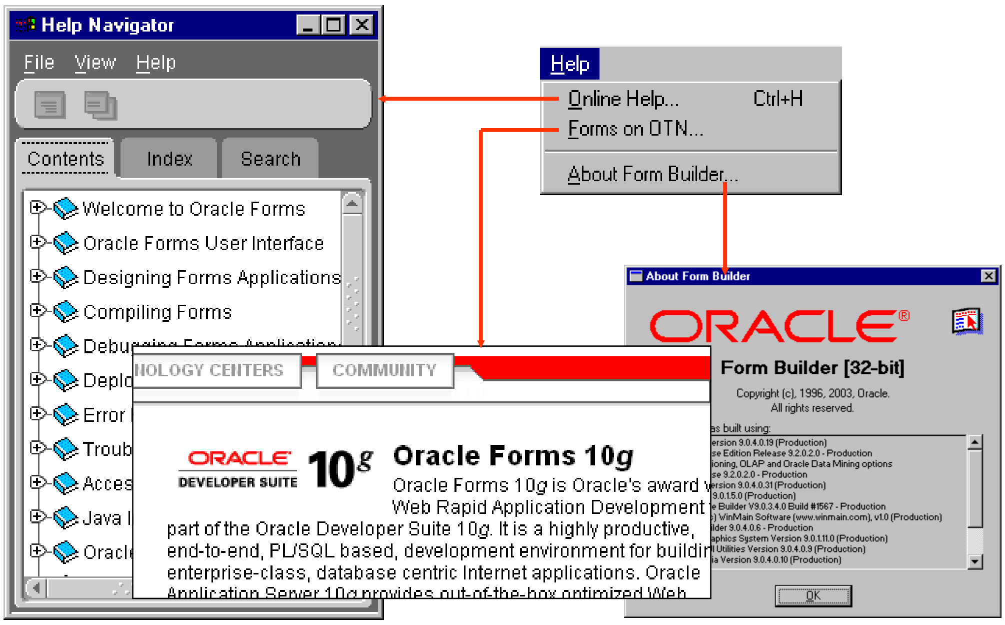
Example:

```
Forms.build_before_run = on
Forms.welcome_dialog = on
```

Oracle Forms Developer reads the preference file whenever you invoke Forms Builder. Oracle Reports Developer reads the preference file whenever you invoke Report Builder.

Note: The preferences file is an editable text file. If possible, however, you should alter the options in the Preferences dialog box.

Using the Online Help System



Oracle Forms 10*g* is Oracle's award winning Web Rapid Application Development part of the Oracle Developer Suite 10*g*. It is a highly productive, end-to-end, PL/SQL based, development environment for building enterprise-class, database centric Internet applications. Oracle Application Server 10*g* provides out-of-the-box optimized Web

Invoking Online Help Facilities

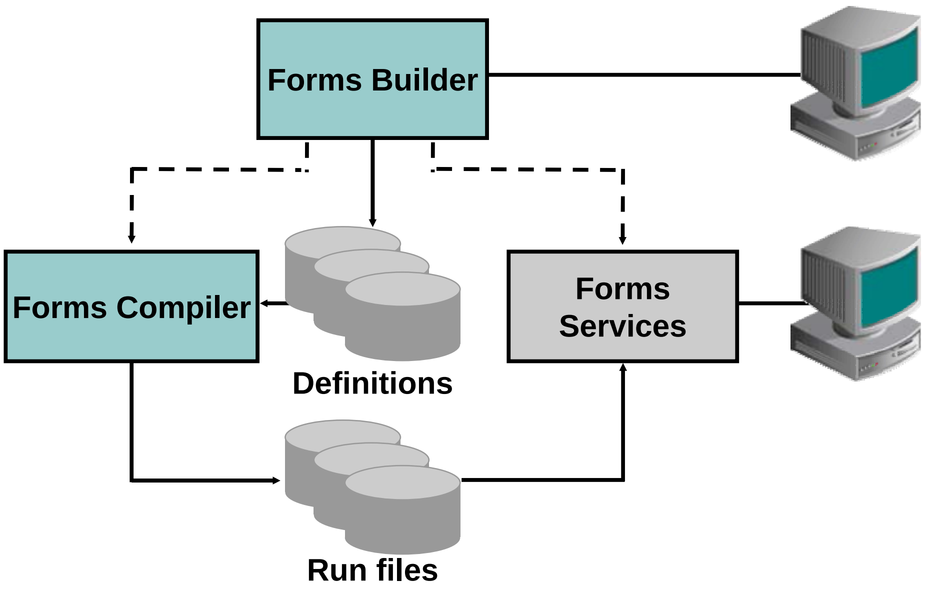
Oracle Forms Developer Help Options

The table below describes the Help menu options in Forms Builder:

Help Menu Option	Description
Online Help	Comprehensive online help window with 3 tabs. The Contents tab provides access to a variety of manuals and references. There are also Index and Search tabs. The Help key ([F1] for Windows NT/95) displays context-sensitive online help at any place in the Builder.
Forms on OTN	The latest product information on the Oracle Technology Network
About Form Builder	This is a separate window that shows product components and their version numbers. When you are connected to a database server, it also displays similar information for server-side product components.

You can also invoke context-sensitive online help from Forms Builder by pressing `[Help]` (`[F1]` on Windows).

Forms Developer Executables



Forms Developer Executables

Forms Builder includes two executables (components) that you can access as the designer of applications.

Forms Builder

This is the application-building component of Oracle Forms Developer. You can use Forms Builder to design and store the definitions of form, menu, and library documents. While in the Forms Builder, you can invoke the other component, Forms Compiler. You must run the Forms Builder component in a GUI environment in order to use its graphical design facilities.

Forms Compiler

Once your form is built, use the Forms Compiler. This reads the definition of your module and creates an executable run file.

Invoking Forms Builder Executables

In a GUI environment, you usually store commands to invoke Forms Builder components in menus and window icons for convenient access. You can also enter these commands on the command line.

Forms Developer Executables (continued)

For example:

```
        IFBLD90 [my_form] [scott/tiger@my_database]
```

Note: Commands for invoking the product components vary according to platform.

Forms Services

Because Forms applications are Web based, it is not possible to run them directly from the command line. Instead, they are invoked by typing a URL into a browser or applet viewer command, directed to Forms Services.

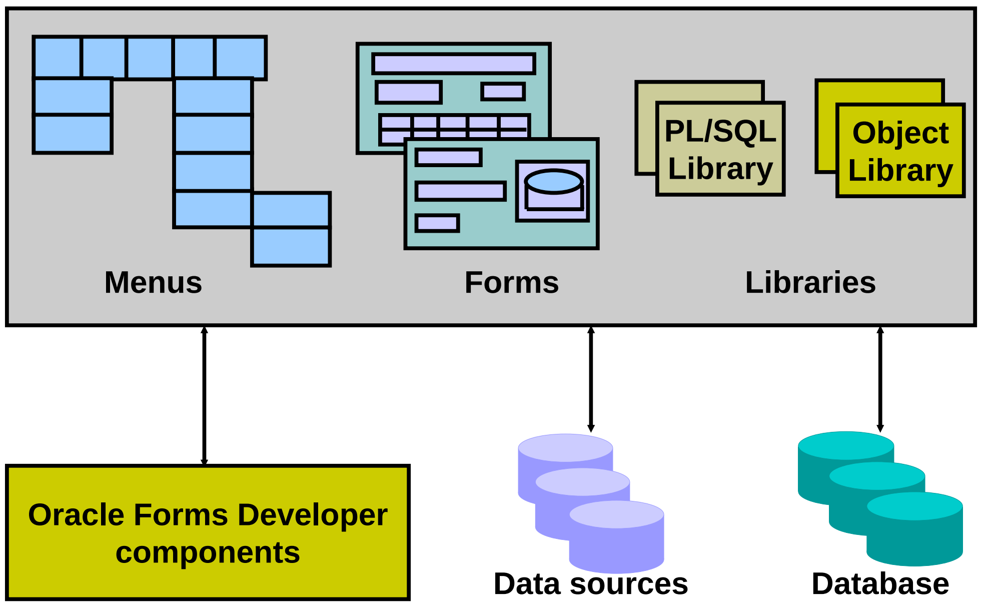
The files used at run time must already have been compiled by the Forms Compiler component. These files must reside on the middle tier machine in a directory accessible to the Forms Runtime Engine (in `FORMS90_PATH`).

To test your applications, you also can access Forms Services directly from Forms Builder by setting certain preferences, as described later in this lesson.

Instructor Note

Explain the components of the product and their relationships.

Forms Developer Module Types



Forms Developer Module Types

A Forms application can consist of many modules—that is, files. A module is a major component of your application and is the basis for storage and ownership. A module owns the rest of the objects in the system.

A Forms Developer module can be of the following types:

- **Form:** As the main component of an application, the form module presents the objects and data that users can see or interact with. Data items in a form are arranged into records.
- **Menu:** A menu module can consist of a hierarchy of menus, each with selectable items.
- **PL/SQL Library:** A PL/SQL Library is a collection of PL/SQL program units whose code can be referenced and called from other modules.
- **Object Library:** An Object Library is a collection of form objects that you can use in other modules. You can create it to store, maintain, and distribute standard objects that can be reused across the entire development organization.

Forms Developer Module Types (continued)

Forms Builder Module Types

Forms Builder provides the default menu for every form. The default menu includes commands for all basic database operations, such as insert, delete, query, and so on. If your application has specific requirements that are not met by the default menu, you can create a custom menu module. Menu modules are not functional by themselves, but when attached to form modules they can provide a service to the facilities offered by a form, as well as options to invoke facilities elsewhere.

PL/SQL Library documents can contain program units that can be used by other form and menu modules.

You can build an application from multiple form modules, menu modules, and library documents as needed.

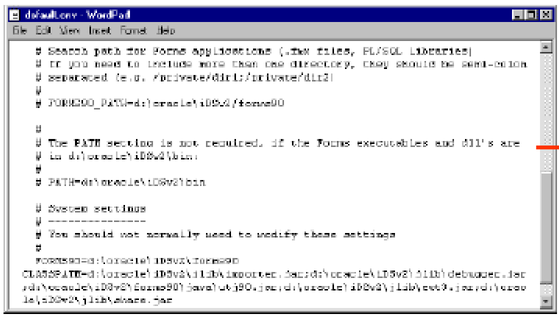
Instructor Note

Explain the relationship among multiple forms, menus, and libraries.

Defining Forms Environment Variables for Run Time

Set on middle-tier machine (used at run time):

- `FORMS90_PATH`
- `ORACLE_PATH`
- `CLASSPATH`



```
    # Search path for Forms applications (.fmx files, PL/SQL libraries)
    # If you need to include more than one directory, they should be semi-colon
    # separated (e.g. /private/dir1:/private/dir2)
    #
    # FORMS90_PATH=d:\oracle\iDSv2/forms90

    #
    # The PATH setting is not required. if the Forms executables and dll's are
    # in d:\oracle\iDSv2\bin:
    #
    # PATH=d:\oracle\iDSv2\bin

    # System settings
    # ----------------
    # You should not normally need to modify these settings
    #
    FORMS90=d:\oracle\iDSv2\forms90
CLASSPATH=d:\oracle\iDSv2\jlib\importer.jar;d:\oracle\iDSv2\jlib\debugger.jar
;d:\oracle\iDSv2\forms90\java\ut j90.jar;d:\oracle\iDSv2\jlib\ewt3.jar;d:\orac
le\iDSv2\jlib\share.jar
```

For Forms deployment, the settings in the environment control file override system settings.

Oracle Developer Environment Variables

Introduction

Oracle Forms Developer uses many environment variables. These have default values, all of which you can modify in your own environment for different applications.

Setting Search Paths for Run Time

Forms uses some environment variables set on the middle-tier machine to search at run time for files such as forms, menus, and libraries. This enables you to build applications that are portable across platforms and directory structures by avoiding hard-coded paths in file references.

Forms searches the following paths in order until the required file is found:

- The current working directory
- Directories in `FORMS90_PATH`
- Directories in `ORACLE_PATH`

Although you could set these variables at the machine level, such as in the Windows Registry, it is preferable to set them in the Forms environment file. Settings in this file override system settings for running a Forms application.

Defining Forms Environment Variables for Design Time

Set on Developer Suite machine (used by Forms Builder):

- `FORMS90_BUILDER_CLASSPATH`
- `UI_ICON`
- `UI_ICON_EXTENSION`
- `FORMS90_HIDE_OBR_PARAMS`

Windows: Modify in Registry (`REGEDIT.EXE` or `REGEDT32.EXE`)

Oracle Developer Environment Variables (continued)

Defining the Design Time Environment

Design time environment variables help govern the behavior of Forms Builder. These environment variables must be set on the machine where Oracle Developer Suite is installed. For example, on Windows platforms you set them in the Windows Registry.

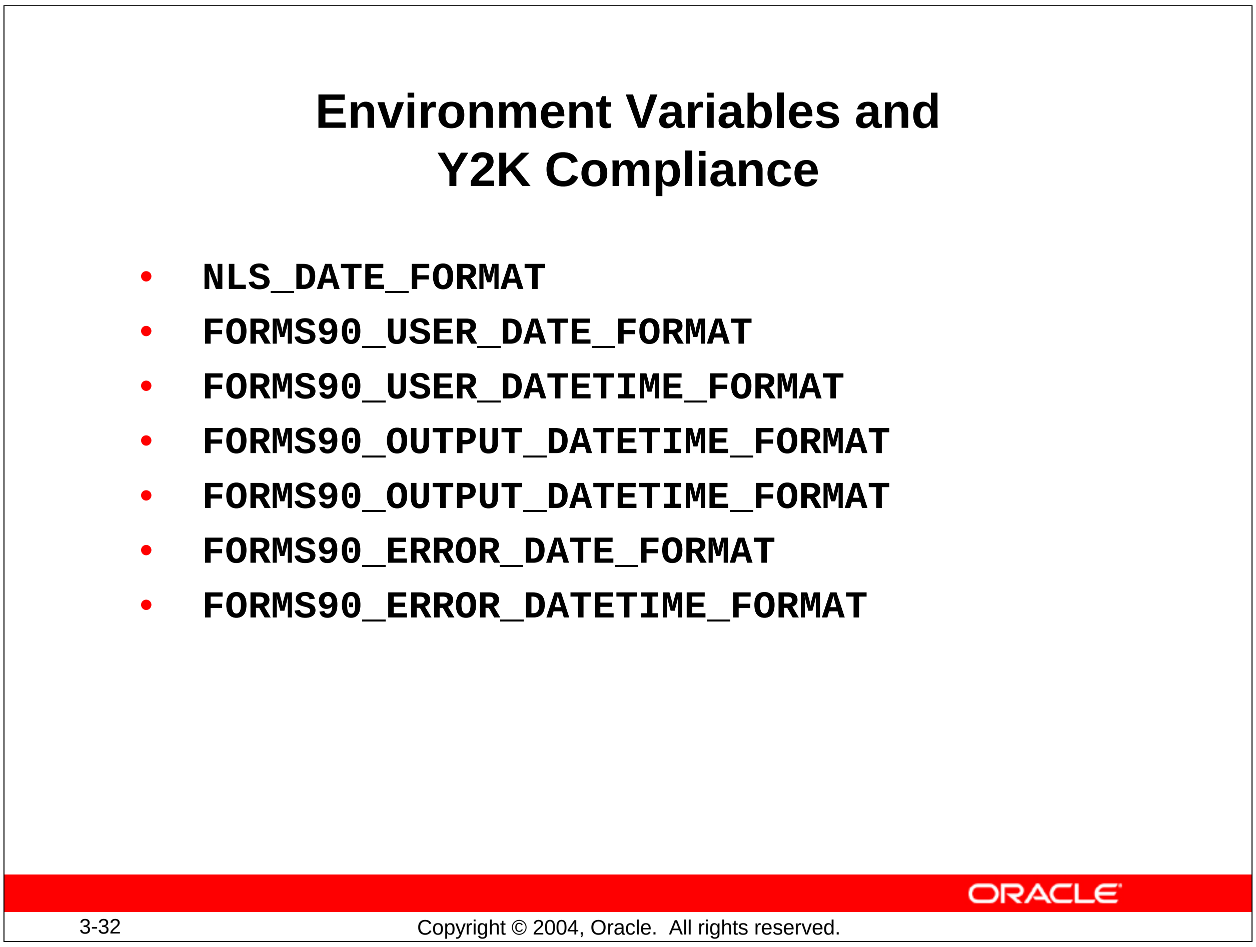
Java class files: Forms Builder needs access to certain Java classes for some of its features, such as Help, the debugger, and the Java importer. You set `FORMS90_BUILDER_CLASSPATH` so that Forms Builder can find the Java classes it needs during development and testing of an application.

Icon files: Forms Builder enables you to create buttons with iconic images. You can use `.ico`, `.gif`, or `.jpg` images. You set `UI_ICON` to the directory path where these images are located. You set the environment variable `UI_ICON_EXTENSION` to tell Forms Builder which type of image to display on buttons. Valid values are `ico` (the default), `gif`, or `jpg`.

Although you can use `.ico` images to display on iconic buttons within Forms Builder, such images will not be displayed when running the form — use `.gif` or `.jpg` files for deployed icons.

Oracle Developer Environment Variables (continued)

URL parameters: By default, when you run a form from Forms Builder, the parameters that are passed in the URL are hidden. For testing purposes, if you want to be able to see these parameters, such as the form name, you can set `FORMS90_HIDE_OBR_PARAMS` to `FALSE` (the default is `TRUE`) . OBR stands for one button run, a term used for running a form from within Forms Builder by clicking the Run Form button.



Environment Variables and Y2K Compliance

- `NLS_DATE_FORMAT`
- `FORMS90_USER_DATE_FORMAT`
- `FORMS90_USER_DATETIME_FORMAT`
- `FORMS90_OUTPUT_DATETIME_FORMAT`
- `FORMS90_OUTPUT_DATETIME_FORMAT`
- `FORMS90_ERROR_DATE_FORMAT`
- `FORMS90_ERROR_DATETIME_FORMAT`

3-32

Oracle Developer Environment Variables (continued)

Dates in Oracle Forms Developer

Dates in Oracle Forms Developer applications can come from several sources:

- Fetched from the database
- Entered by the end user
- Defined in the application itself

Date Format Masks

In a later lesson, you will learn how to specify a format mask for a date item in your form. In addition to the format masks a developer might explicitly define, Forms Builder uses several of its own internal masks. The values for these internal masks can be specified with property values and environment variables.

To eliminate potential errors caused by the year 2000 (Y2K), you can set environment variables that Forms Builder uses to format date items:

- **Database date format mask:** Each database session within a Forms application has a single database date format mask. A default value for this mask is established by the Oracle server's initialization parameter. You can override this value in each new database session for a particular client by setting the client's `NLS_DATE_FORMAT` environment variable.

Oracle Developer Environment Variables (continued)

Date Format Masks (continued)

- **Input date format masks:** This mask (potentially, a set of masks) is used to convert a user-entered string into a native format date value. You can set the environment variables `FORMS90_USER_DATE_FORMAT` or `FORMS90_USER_DATETIME_FORMAT` to specify these format masks.
  For example, you could set `FORMS90_USER_DATE_FORMAT` to the value `FXFMDD-MM-RRRR`. This would force the user to enter values into date items (with no format mask) in the format exemplified by 30-06-97. The `RRRR` token enables years between 1950 and 2049 to be entered with the century omitted.
- **Output date format masks:** Converts dates displayed in output such as lists of values: `FORMS90_OUTPUT_DATE_FORMAT` or `FORMS90_OUTPUT_DATETIME_FORMAT`
- **Error date format masks:** Converts dates displayed in error messages: `FORMS90_ERROR_DATE_FORMAT` or `FORMS90_ERROR_DATETIME_FORMAT`

Forms Files to Define Run-Time Environment Variables

Environment control file:

- `\forms90\server\default.env` *or*
- Other file specified in Forms configuration file

Forms configuration file:

- `\forms90\server\formsweb.cfg` or other
- Used to specify:
  - System parameters, such as `envFile` and `workingDirectory`
  - User parameters, such as form and user ID
  - Settings for the Java client
  - Other settings

ORACLE

3-34

Oracle Developer Environment Variables (continued)

Modifying Run-Time Environment Variables

In a Windows 32-bit environment, use the Windows Registry to modify these paths, except for `CLASSPATH`, which is set in the System settings of the Control Panel. You can also override these settings at run time in the file that controls the Forms run-time environment, which is the `default.env` file unless a different file is specified.

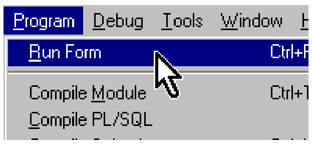
Using the Forms environment file makes it easier to deploy the application on any platform. You can specify which environment file to use in a special Forms configuration file called by default `formsweb.cfg`. In this configuration file, you can set system parameters, such as the name of the environment control file. You also can set parameters to control which form to run, the user ID, aspects of the Java client and the HTML file that contains the Java applet, and many other settings.

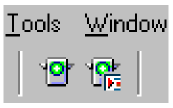
Instructor Note

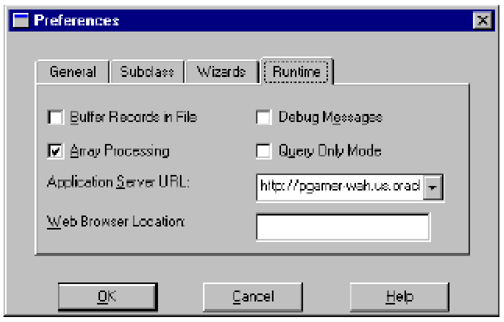
Warn students to take care when modifying environment variables, because they immediately affect the environment and mistakes can prevent applications from running. Whenever possible later in the course, point out when these variables are used, and remind students to avoid hard coding path names in their applications.

Testing a Form: The Run Form Button

- With the Run Form menu command or button, you can:
  - Run a form from Forms Builder
  - Test the form in a three-tier environment
- The Run Form command takes its settings from Preferences:
  - Edit > Preferences
  - Runtime tab
  - Set Web Browser Location if desired
  - Set Application Server URL to point to Forms Servlet:
    `http://127.0.0.1:8889/forms90/f90servlet`







☐ Buffer Records in File
☐ Debug Messages
☑ Array Processing
☐ Query Only Mode
Application Server URL: http://pgamer-wah.us.oracl
Web Browser Location:

Testing a Form with the Run Form Button

The Run Form menu command or button enables you to run a form module in a Web browser from within Forms Builder. This makes it easy to test your application in a three-tier environment, with all components appearing and behaving as they would for a user of the application.

You must define some basic information to enable this three-tier testing environment. You set this information in the Preferences dialog that you access from Edit > Preferences. Once the Preferences dialog is open, perform the following steps:

1. Click the Runtime tab.
2. Set the Application Server URL: This must be a URL pointing to the Forms Servlet on the middle tier. Note that it is typically on the same machine where you are running Forms Builder. You can also use the `config` parameter to specify a named configuration in the Forms Web configuration file (`formsweb.cfg` by default). Example for the same machine with OC4J running on the default port of 8889: http://127.0.0.1:8889/forms90/f90servlet?config=myapp
3. Set the Web Browser Location (only needed if you want to run in a different browser than the default for your machine).

Testing a Form with the Run Form Button (continued)

Note: The correct way to exit a Forms session is to exit the form (File > Exit, or click Exit), then close the browser. If you close the browser without first exiting the form, your session may hang.

You will notice this because you may not be able to recompile the same form, but will receive the error: `FRM-30087: Unable to create form file`. If this happens, you will need to open Task Manager and end the `ifweb90` process manually.

Instructor Note

Please emphasize to students the correct way to exit a Forms session.

Whether or not a developer is able to recompile a form when that form is already running depends on the parameter `OBR`, which stands for one button run. In Forms 10*g* (9.0.4), this parameter defaults to `YES`, which turns off memory mapping so that the form stays open only while it is being loaded into process memory. When memory-mapped file I/O is in effect, the `.fmx` file stays open until the form is closed, so you cannot recompile. For a description of the `OBR` parameter, see Bug 2145774.

Summary

In this lesson, you should have learned that:

- **Forms Builder includes the Object Navigator, the Property Palette, the Layout Editor, and the PL/SQL Editor**
- **You can use the Object Navigator or the menu and its associated toolbar icons to navigate around the Forms Builder interface**
- **The main objects in a form module are blocks, items, and canvases**
- **The Edit > Preferences dialog box enables you to customize the Forms Builder session**

ORACLE

3-37

Summary

- With Forms Builder, an Oracle Forms Developer component, you can develop form-based applications for presenting and manipulating data in a variety of ways. Forms Builder enables screen-based queries, inserts, updates, and deletes of data.
- Forms Builder provides powerful GUI and integration features.
- Applications consist of form modules, menu modules, and library documents.
- Form modules consist of logical data blocks. A data block is the logical owner of items. Items in one data block do not need to be physically grouped. Items in one data block can span several canvases.
- Environment variables govern the behavior of:
  - Running forms (set in the Forms environment file)
  - Forms Builder (set on the development machine, such as in the Windows Registry for Windows platforms)
- You can run a form application from within Forms Builder in order to test it in a browser. You specify the URL to use in the Runtime tab of the Preferences dialog box.

Summary

- The Help menu enables you to use the online help facilities to look up topics, or you can invoke context-sensitive help
- The Forms Developer executables are the Forms Builder and the Forms Compiler
- The Forms Developer module types are forms, menus, and libraries
- You can set environment variables in the Forms environment file (for run time) or on the development machine (for design time).
- You can use the Run Form button to run a form from within Forms Builder

Practice 3 Overview

This practice covers the following topics:

- Becoming familiar with the Object Navigator
- Setting Forms Builder preferences
- Using the Layout Editor to modify the appearance of a form
- Setting run-time preferences to use OC4J to test applications
- Running a form application from within Forms Builder
- Setting environment variables so the Layout Editor in Forms Builder displays `.gif` images on iconic buttons

Practice 3 Overview

In this practice you become familiar with Oracle Forms Developer by performing the following tasks:

- Examining the Object Navigator in Forms Builder
- Setting Forms Builder preferences
- Using the Layout Editor in Forms Builder to modify the appearance of a form.

In addition, you set run-time preferences to use OC4J to test your application on your local machine. You also set environment variables so that images display on iconic buttons in the Layout Editor of Forms Builder.

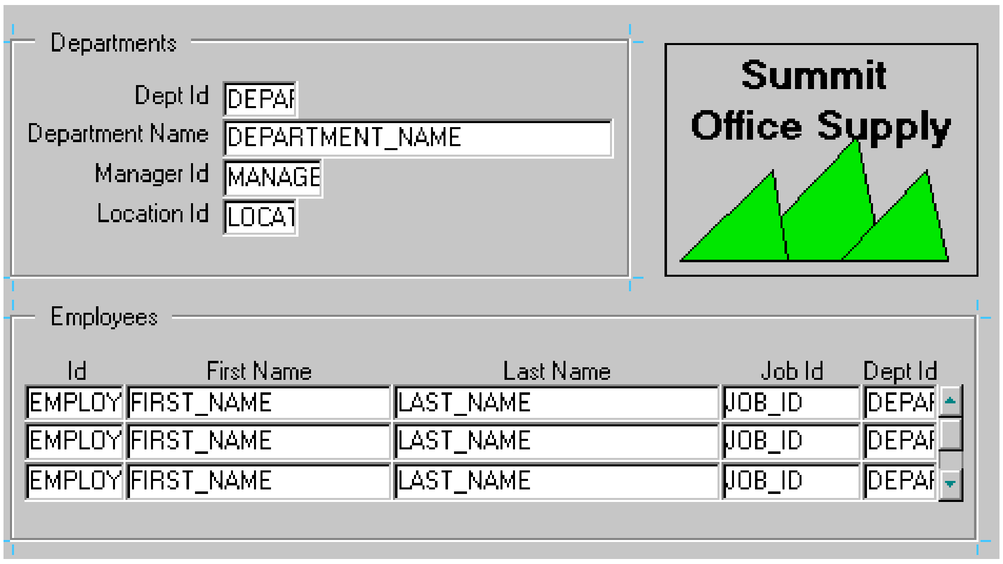
Note: For solutions to this practice, see Practice 3 in Appendix A, “Practice Solutions.”

Instructor Note

This practice is intended to make students familiar and at ease with the interface. For additional help with the Layout Editor, refer them to Appendix F, “Using the Layout Editor.”

Practice 3

1. Invoke Forms Builder. If the Welcome page is displayed, select “Open an existing form”. If the Welcome page is not displayed, select File > Open.
2. Open the `Orders.fmb` form module from the Open Dialog window.
3. Set your preferences so that Welcome dialogs display when you first open Forms Builder and when you use any of the wizards.
4. Close the Orders form.
5. Open the `Summit.fmb` form module.
6. Expand the Data Blocks node.
7. Expand the Database Objects node. If you cannot expand the node, connect to the database and try again. What do you see below this node?
8. Collapse the Data Blocks node.
9. Change the layout of the `Summit.fmb` form module to match the following screenshot. At the end, save your changes.



   a. Invoke the Layout Editor.
   b. Move the three summit shapes to the top-right corner of the layout. Align the objects along the bottom edge.
   c. Select the summit shape in the middle and place it behind the other two shapes.
   d. Draw a box with no fill around the summit shapes.
   e. Add the text Summit Office Supply in the box. If necessary, enlarge the box.
   f. Move the Manager_Id and Location_Id items to match the screenshot.
   g. Move the First_Name item up to align it at the same level as the Last_Name item.
   h. Resize the scroll bar to make it the same height as the three records in the Employees block.
   i. Save the form module.

Practice 3 (continued)

10. Set the run-time preferences for Forms Builder to use OC4J to test your applications. Set the Application Server URL by pressing Reset to Default, which will enter the following settings:

URL Component	Value
Machine name	`127.0.0.1` (or your local machine name)
Port	`8889` (for OC4J)(or the OC4J port on your local machine)
Pointer to Forms Servlet	`forms90/f90servlet`

11. In Forms Builder, open and run the Customers form located in your local directory (you must have OC4J running first).
    Note: Run-time fonts may look different than the fonts used in Forms Builder because Forms Builder uses operating system–specific fonts, but at run time only Java fonts are used. In addition, the appearance is different from the Layout Editor because the Oracle (rather than the generic) look and feel is used by default.
12. Click the Account Information tab. You should be able to see the image of a flashlight on the List button. Exit the run-time session and close the browser window.
13. In Forms Builder, open the Layout Editor for the CV_Customer canvas by expanding the Canvases node in the Object Navigator and double-clicking the CV_Customer canvas icon. In the Layout Editor, click the Account Information tab. What do you observe about the List button?
14. From the Windows Start menu, choose Run, type `regedit`, and click OK. Expand the registry nodes `HKEY_LOCAL_MACHINE` > `SOFTWARE` > `ORACLE`. Click into the `ORACLE` node, or into one of the `HOME` nodes beneath it; your instructor will tell you which node to open. Ensure that you have opened the correct node by verifying that the key `FORMS90` exists in that node.
15. Set the path for Forms Builder to locate icons:
    a) Double-click the `UI_ICON` key to open it for editing.
    b) For the value data, append the path to the `.gif` file that you will use for the button icon, which is the `\icons` subdirectory of your lab directory. Separate this path from the remainder of the string with a semicolon; for example: `;e:\labs\lab\icons`, then click OK.
16. In a similar fashion, set the value for Forms Builder to use for the icon extension, then close the registry editor.
17. Close and reopen Forms Builder. Open the Customers form and verify that the flashlight icon now displays in the Layout Editor.

4

Creating a Basic Form Module

Schedule:	Timing	Topic
	40 minutes	Lecture
	30 minutes	Practice
	70 minutes	Total

Objectives

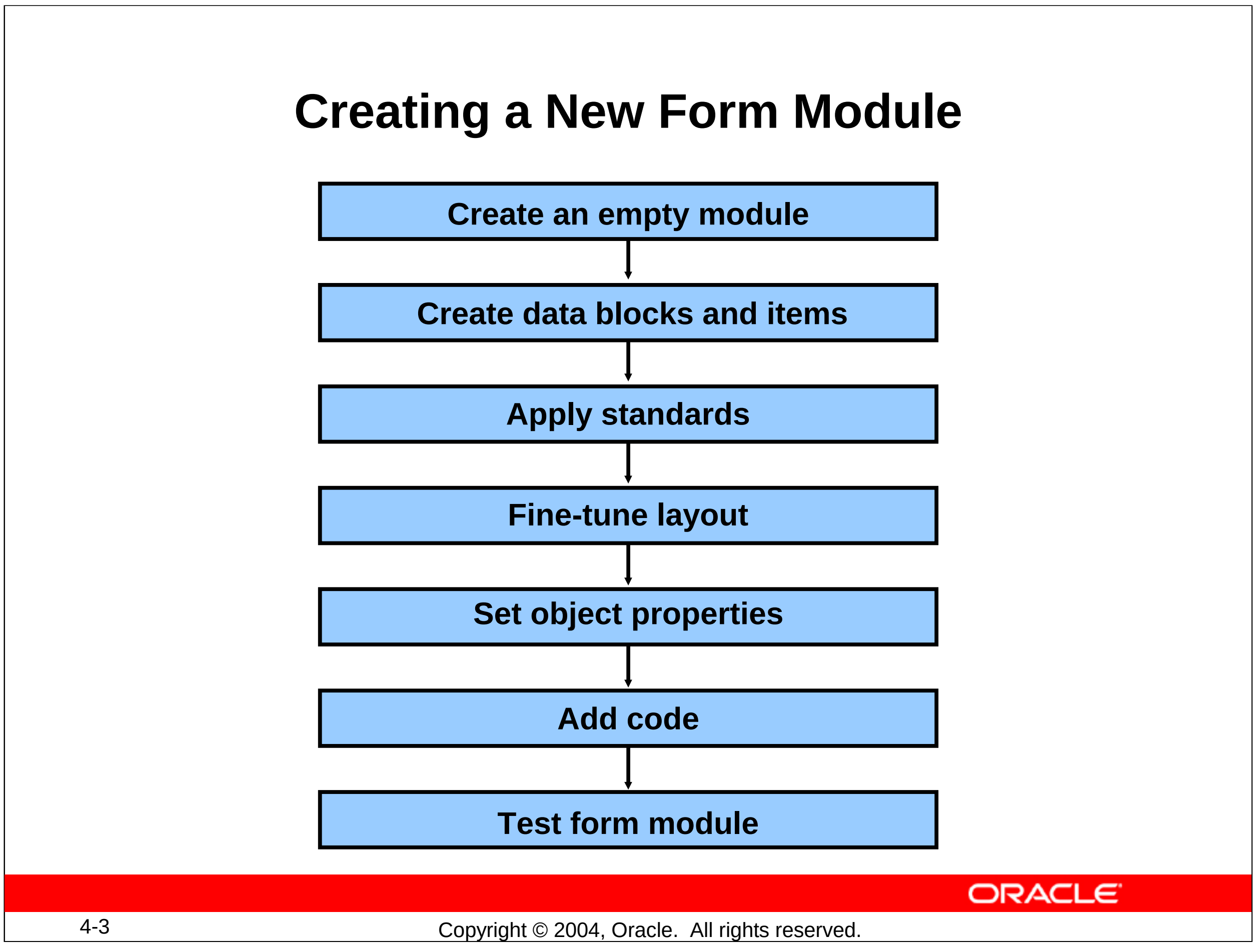
After completing this lesson, you should be able to do the following:

- Create a form module
- Create a data block
- Save and compile a form module
- Identify Forms file formats and their characteristics
- Describe how to deploy a form module
- Explain how to create documentation for a Forms application

Introduction

Overview

Oracle Forms Developer applications usually consist of one or more form modules. Each form module comprises data blocks that are built using table specifications from the database. This lesson shows you how to create a basic form module and its data blocks. You also learn to deploy a form module.



Creating a New Form Module

Create an empty module → Create data blocks and items → Apply standards → Fine-tune layout → Set object properties → Add code → Test form module

4-3

Creating a New Form Module

This lesson covers the basic process for creating a new form module and data blocks within it.

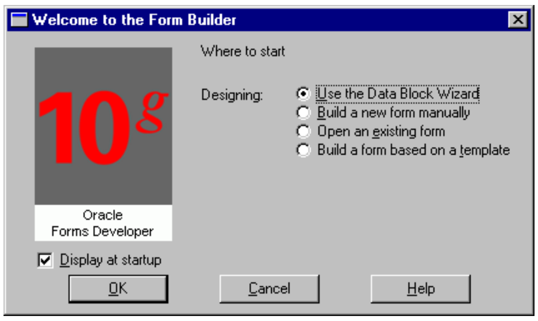
How to Create a New Form Module

Step	Action	Forms Builder Tool
1.	Create an empty module	Object Navigator
2.	Create data blocks and items	Data Block Wizard
3.	Apply user interface standards to objects	Object Library
4.	Fine-tune the layout	Layout Wizard or Layout Editor
5.	Set object properties	Property Palette
6.	Add code	PL/SQL Editor
7.	Test the form module	Run Form button

Creating a New Form Module

Choose one of the following methods:

- **Use wizards:**
  - **Data Block Wizard**
  - **Layout Wizard**
- **Build module manually**
- **Use template form**


Welcome to the Form Builder

Where to start

Designing: [✓ Use the Data Block Wizard] Build a new form manually / Open an existing form / Build a form based on a template

[✓] Display at startup

Creating a New Form Module

There are several ways to create a new form module.

- Invoke the Forms Builder component. This takes you to the Forms Builder Welcome page, unless you have changed the Preferences to not display it. Now do one of the following:
  - Select the “Use the Data Block Wizard” option, then follow the required data block creation steps. Then follow the Layout Wizard steps.
  - Select the “Build a new form manually” option. This takes you into the Forms Builder Object Navigator (automatically creating an empty form module).
  - Select the “Build a form based on a template” option and use a template form.
- If you are already in the Forms Builder component, you can create a new form module by doing one of the following:
  - Double-click the Forms node in the Object Navigator (only when no other form modules are available).
  - Select File > New > Form.
  - Select the Object Navigator node for Forms, and then click the Create icon.

Creating a New Form Module (continued)

Changing the Form Module Name

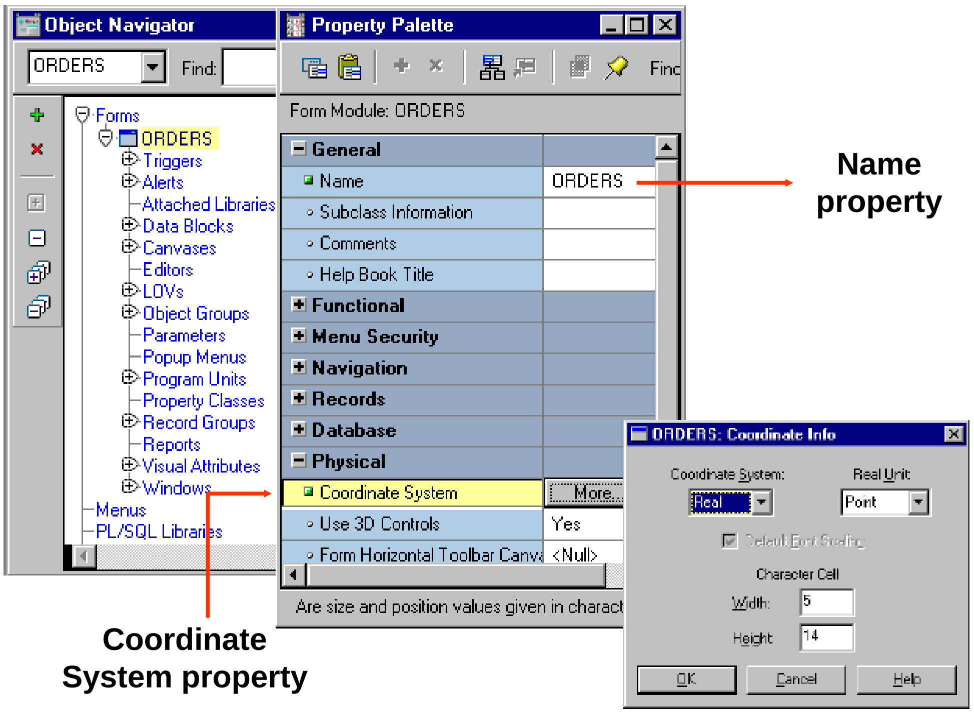
- When you first build a form module, Forms Builder assigns the name MODULEXX to the new form module, where XX is the next number available for module names. This name is displayed in the Object Navigator and in the Property Palette. You should change the default name to a meaningful name in either of the following places:
- In the Object Navigator:
  - Click the form module name.
  - Change the default name as desired and press [Enter].
- In the Property Palette (shown on the next page)

Note: Follow Oracle naming rules. Do not give two objects of the same type the same name. The name cannot include Oracle or Forms Builder reserved words.

Instructor Note

Demonstration: Create a new, empty form module and change its name to DEPT_EMP.

Form Module Properties



Setting Form Module Properties

Each form module consists of several objects. Objects within a form, and the form module itself, have properties that define their behavior. You can see the properties of an object and their values in its Property Palette.

To open the Property Palette of an object, do one of the following:

- Double-click the object’s icon in the Object Navigator.
- Select the object in the Object Navigator and select Tools > Property Palette.

To obtain online help for any of the properties, click the property and use the Help key, `[F1]`, to bring up a description of that property.

Define the properties of the form module when you first create it. The properties affect the general behavior of the form and the objects within it. Properties for a form module include the following:

Property	**Description**
Name	Specifies the internal name of the form module, as it appears in the Object Navigator
Coordinate System	Defines the units used to measure objects in the form and their positions (see next section)

Setting Form Module Properties (continued)

Choosing a Unit for the Coordinate System

When you click More in the Property Palette window with the Coordinate System property selected, the Coordinate Info window opens.

The Coordinate System unit for a form can be one of the following:

- **Real:**
  - Unit can be pixel, centimeter, inch, point, or decipoint.
  - Real units are suitable for GUI applications and enable flexibility and fine alignment when adjusting object positions and sizes.
- **Character:** Units are character cells (default size taken from the default font settings).

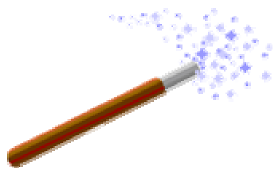
The default unit is point (Real). This means that object positions and sizes within the form are measured by this unit. Points provide fine alignment and consistency across different platforms and video devices.

Instructor Note

Open the Property Palette of the new form module. Concentrate only on the Name and Coordinate System properties. Point out that other properties will be discussed in later lessons. Open the Coordinate Info window and explain the implications of choosing the units. (72 points = 1 inch). Points allow for flexibility when designing cross platform applications.

Creating a New Data Block

- Use Forms Builder Wizards:
  - Data Block Wizard: Create a data block with associated data source quickly and easily
  - Layout Wizard: Lay out data block contents for visual presentation
- Create manually



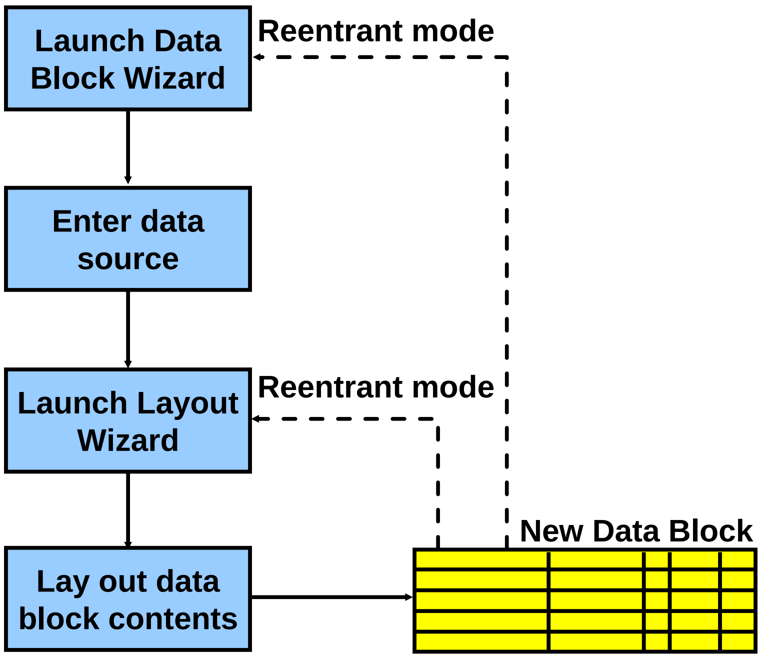
Creating a New Data Block

A form module consists of one or more data blocks and control blocks. Now that you know how to create a new form module, you need to create new data blocks within it.

Block creation involves creating the data block and then laying out its contents for visual presentation. You can create a data block manually or by using the Forms Builder wizards. In this lesson you learn how to create a new data block based on a database table, using the Data Block Wizard and the Layout Wizard.

Note: Recall that a data block can be based on a table or view, a stored procedure, a `FROM` clause query, or a transactional trigger. In this course, you use database tables as the source.

Creating a New Data Block



4-9

Creating a New Data Block (continued)

Data Block Wizard

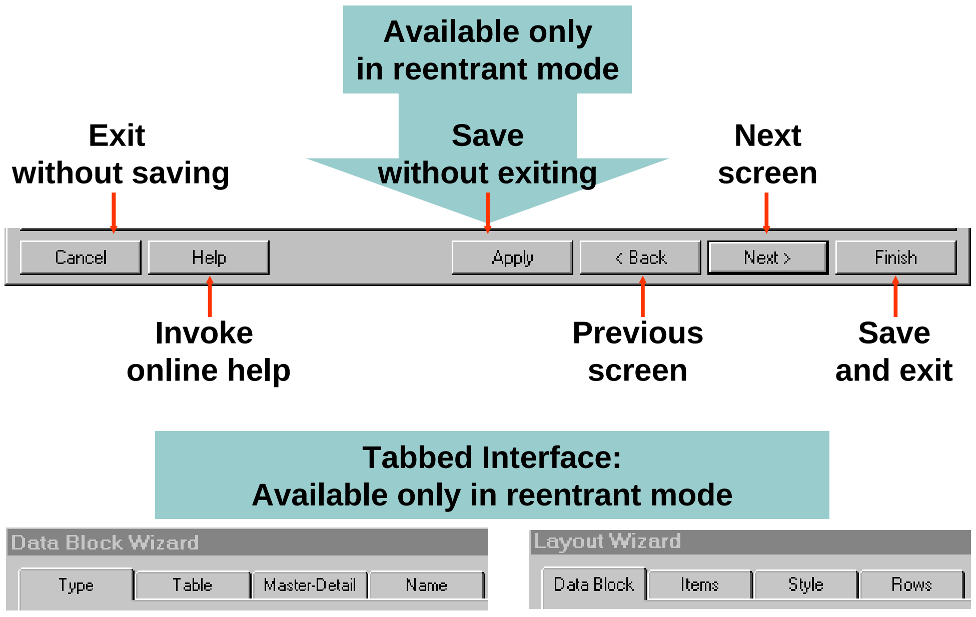
The Data Block Wizard enables you to create (or modify) data blocks quickly and easily for use in your application. The wizard can automatically generate code to enforce integrity constraints in the database.

Layout Wizard

Although the Data Block Wizard allows you to create a new data block easily with its associated data sources, it does not deal with the visual presentation of objects included in the data block. Once you create the data block, you need to lay out its contents for user interaction. To accomplish this task quickly and easily, use the Layout Wizard.

Note: The wizards are not the only way to perform a task such as building a data block, but they are usually the simplest. You can build a block manually instead of using the wizards.

Navigating the Wizards



Navigating the Wizards

The Data Block Wizard and the Layout Wizard provide several buttons to help you navigate.

Button	Description
Cancel	Cancels any changes and exits the wizard
Help	Displays online help text for the current wizard page
Back	Navigates to the previous page in the wizard
Next	Navigates to the next page in the wizard
Apply	Applies your changes without exiting the wizard (available only in reentrant mode)
Finish	Saves any changes and exits the wizard

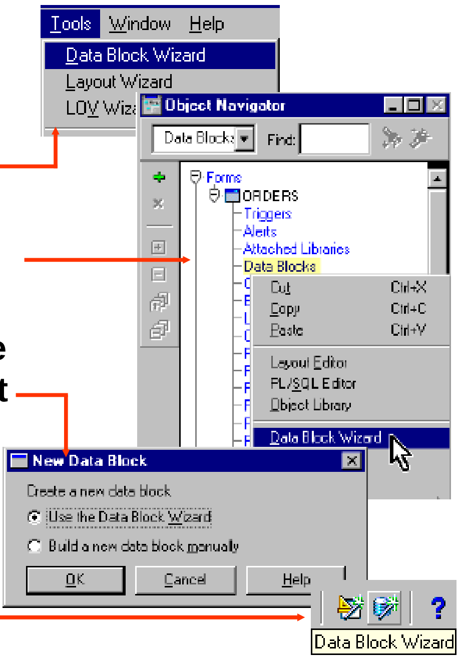
If you click Next or Back before entering all necessary information for a particular wizard page, the wizard prevents you from navigating to another page. Similarly, if you have not entered all necessary information into the wizard when you click Apply or Finish, the wizard automatically takes you to the page where you can finish entering the required information.

In reentrant mode, which you learn about later in this lesson, the wizards have a tabbed interface that enables you to quickly navigate to the section you want to modify.

Launching the Data Block Wizard

In Forms Builder, do one of the following:

- Select Tools > Data Block Wizard.
- Right-click and select Data Block Wizard.
- Select the Data Blocks node and click Create icon; select Use the Data Block Wizard option.
- Use the Data Block Wizard button on the toolbar in the Layout Editor.



ORACLE

Launching the Data Block Wizard

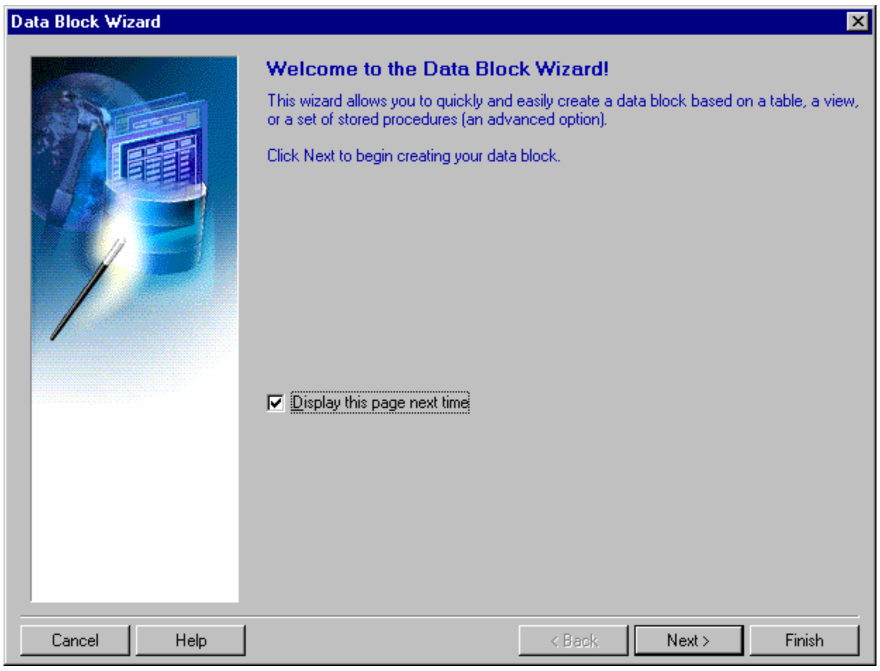
To launch the Data Block Wizard, perform one of the following:

- In the Forms Builder, do one of the following:
  - Select Tools > Data Block Wizard from the Forms Builder default menu system.
  - Right-click and select the Data Block Wizard option.
  - In the Object Navigator, select the Data Blocks node, then click the Create icon. In the New Data Block dialog box, select the Use the Data Block Wizard option.
  - In the Layout Editor, click Data Block Wizard on the toolbar.
- If you are not already in Forms Builder, launch Forms Builder and select the Use the Data Block Wizard option in the Forms Builder Welcome page.

Instructor Note

Demonstration: Create a new data block by using the Forms Builder wizards. Base the data block on the `DEPARTMENTS` table. Use the Form (not tabular) layout.

Data Block Wizard: Type Page


Data Block Wizard

Welcome to the Data Block Wizard!

This wizard allows you to quickly and easily create a data block based on a table, a view, or a set of stored procedures (an advanced option).

Click Next to begin creating your data block.

☑ Display this page next time

Creating a New Data Block with the Data Block Wizard

Use the Data Block Wizard to create a new data block with its associated data sources. The Data Block Wizard consists of several pages. To create a new data block, you must interact with each page.

Welcome Page

Click Next to continue.

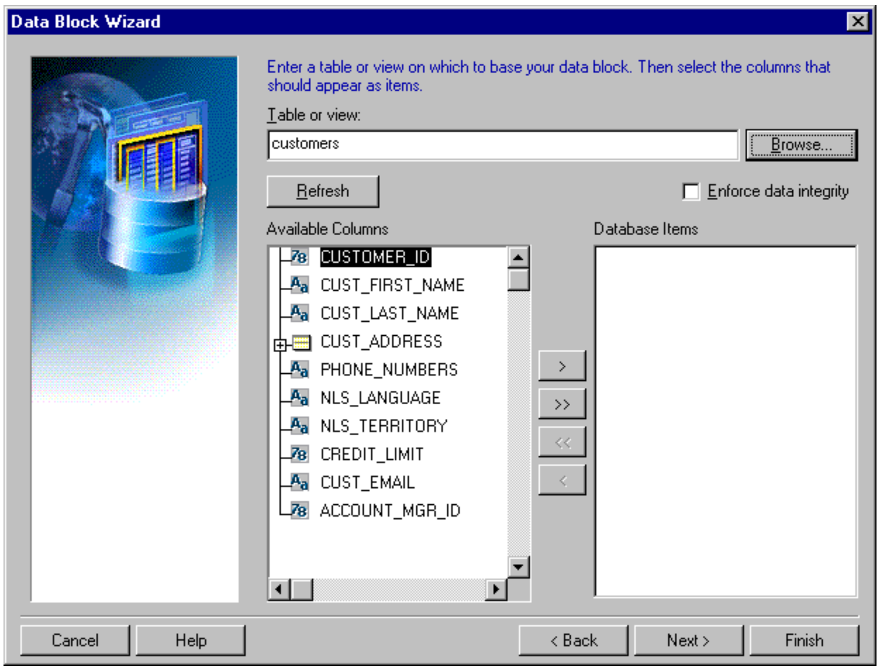
Type Page

Choose between one of two data source types:

- Table or View
- Stored Procedure

Select the Table or View (default) option.

Data Block Wizard: Table Page


Data Block Wizard

Enter a table or view on which to base your data block. Then select the columns that should appear as items.

Table or view: customers

CUSTOMER_ID
CUST_FIRST_NAME
CUST_LAST_NAME
CUST_ADDRESS
PHONE_NUMBERS
NLS_LANGUAGE
NLS_TERRITORY
CREDIT_LIMIT
CUST_EMAIL
ACCOUNT_MGR_ID

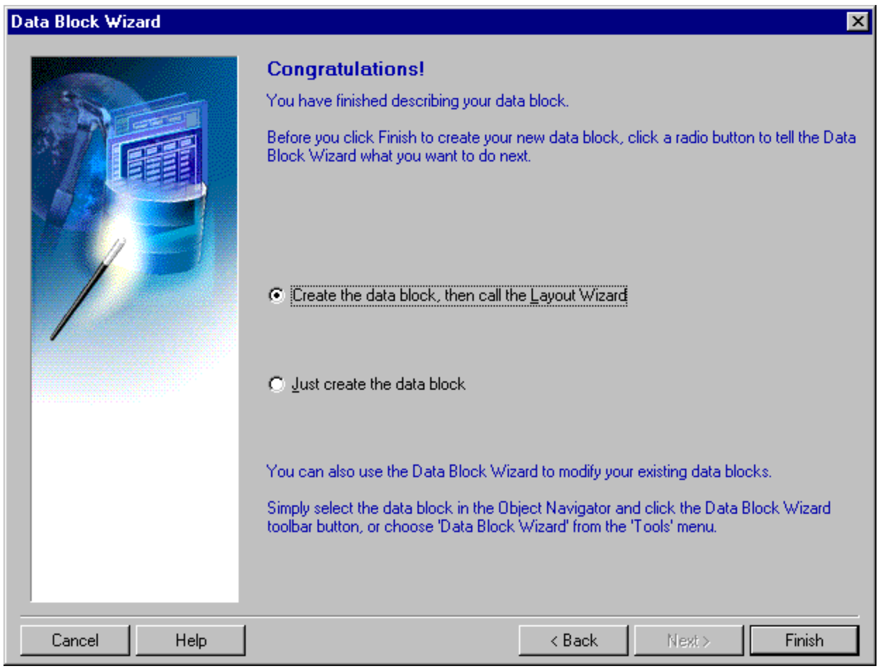
Creating a New Data Block with the Data Block Wizard (continued)

Table Page

1. Enter the table or view name for the data source name, or click Browse and select a name from a dialog box.
2. Click Refresh to display a list of columns in the selected table or view. If you are not connected to the database, the Connect box is displayed.
3. Select the columns you want to include in the data block. (Use `[Control]`- click to select more than one column.)
4. Click the double right arrow or the double left arrow to include or exclude all columns, or click the right arrow or the left arrow to include or exclude selected columns only. You can also drag selected columns from one list to another.
5. Select the "Enforce data integrity" check box if you want the wizard to enforce the database integrity constraints.

Note: If there is at least one other existing block in the current module, the next page that displays is the Master-Detail page, where you can associate the new data block with other master data blocks. This page is discussed later in the lesson.

Data Block Wizard: Finish Page



Congratulations!

You have finished describing your data block.

Before you click Finish to create your new data block, click a radio button to tell the Data Block Wizard what you want to do next.

[✓ Create the data block, then call the Layout Wizard]

Just create the data block

You can also use the Data Block Wizard to modify your existing data blocks.

Simply select the data block in the Object Navigator and click the Data Block Wizard toolbar button, or choose 'Data Block Wizard' from the 'Tools' menu.

Creating a New Data Block with the Data Block Wizard (continued)

Finish Page

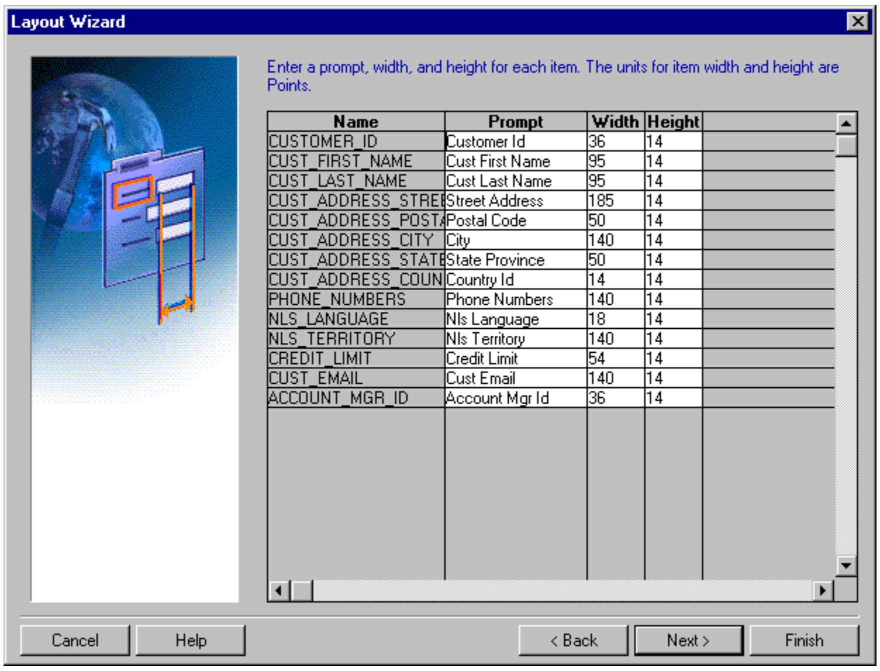
Select the “Create the data block, then call the Layout Wizard” option. Select Finish to create the new data block and immediately invoke the Layout Wizard.

Note: You have the option of exiting the Data Block Wizard at this stage, without immediately invoking the Layout Wizard. If you do so, you can either lay out the data block manually or invoke the Layout Wizard at a later time to lay out the items of a data block. To invoke the Layout Wizard at a later time, select the data block in the Object Navigator, and choose Tools > Layout Wizard.

Instructor Note

You can also select objects as object data sources in the Layout Wizard. Appendix E of this course describes the basics of Oracle objects.

Layout Wizard: Items Page



Enter a prompt, width, and height for each item. The units for item width and height are Points.

Name	Prompt	Width	Height
CUSTOMER_ID	Customer Id	36	14
CUST_FIRST_NAME	Cust First Name	95	14
CUST_LAST_NAME	Cust Last Name	95	14
CUST_ADDRESS_STREE	Street Address	185	14
CUST_ADDRESS_POSTA	Postal Code	50	14
CUST_ADDRESS_CITY	City	140	14
CUST_ADDRESS_STATE	State Province	50	14
CUST_ADDRESS_COUN	Country Id	14	14
PHONE_NUMBERS	Phone Numbers	140	14
NLS_LANGUAGE	Nls Language	18	14
NLS_TERRITORY	Nls Territory	140	14
CREDIT_LIMIT	Credit Limit	54	14
CUST_EMAIL	Cust Email	140	14
ACCOUNT_MGR_ID	Account Mgr Id	36	14

Laying Out a New Data Block with the Layout Wizard

Use the Layout Wizard to lay out the data block items for visual presentation quickly and easily. The Layout Wizard consists of several pages. You must interact with each page.

Canvas Page

1. Select New Canvas from the Canvas pop-up list to get a new canvas on which to display the data block items.
2. Select Content as the canvas type in the Type pop-up list.

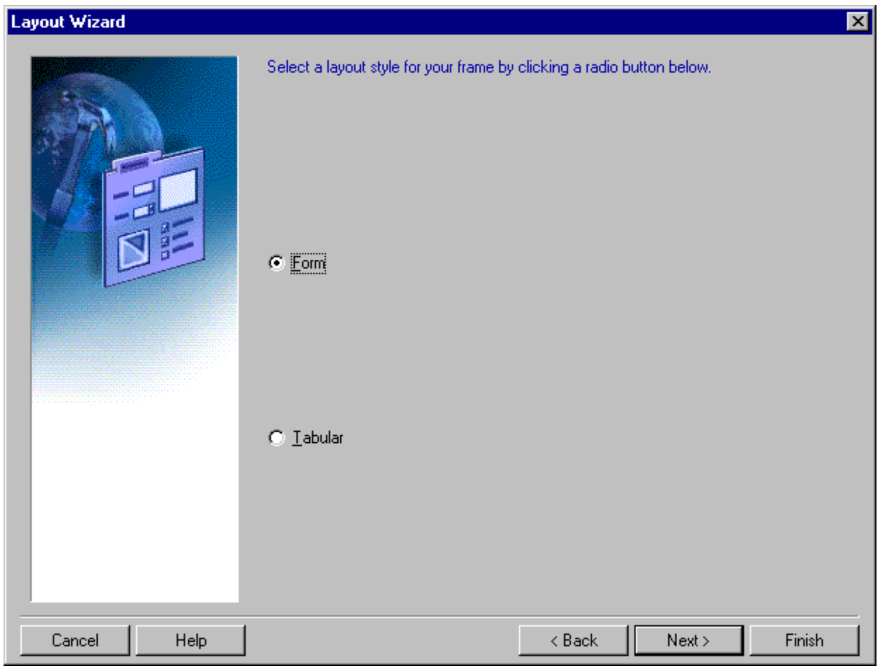
Data Block Page

1. Select the items that you want to display in the data block frame. (Use [Ctrl]-click to select more than one column.)
2. Click the double right arrow or double left-arrow to include or exclude all items, or click the right-arrow or the left-arrow to include or exclude selected items only. You can also drag selected items from one list to another.
   Note: To lay out the items in a particular sequence, drag items into that sequence.
3. You can use the Item Type pop-up list to select a type for each item. The default type is Text for each item.
   Note: An item type can also be changed later to something else, such as pop-up list or radio group.

Items Page

Specify prompt text and display width and height for each display item.

Layout Wizard: Style Page



Select a layout style for your frame by clicking a radio button below.

[● Form] ○ Tabular

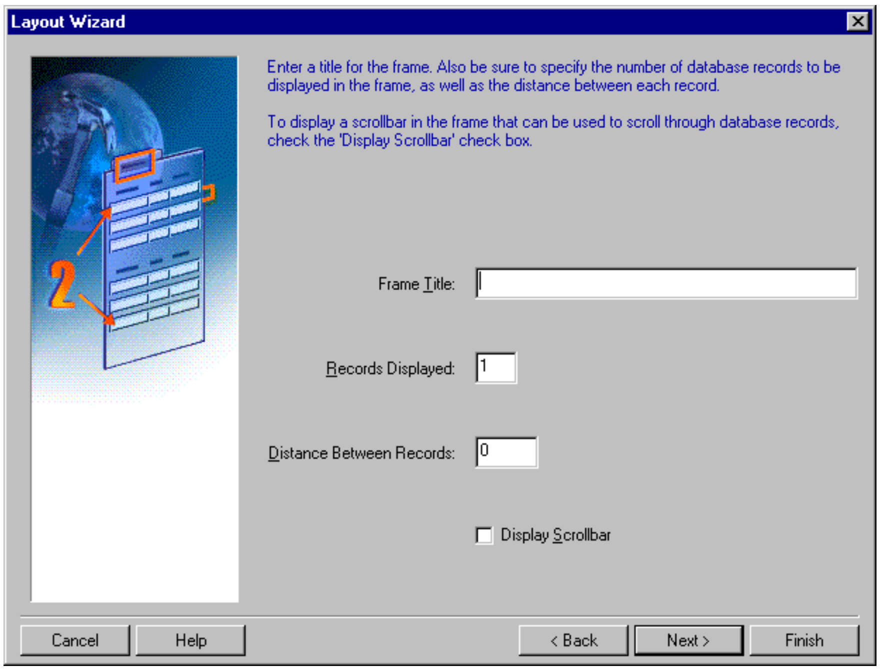
Laying Out a New Data Block with the Layout Wizard (continued)

Style Page

Select a layout style for your frame. Your options are:

- Form (usually used to create single-record data blocks)
- Tabular (usually used to create multirecord data blocks)

Layout Wizard: Rows Page



Layout Wizard

Enter a title for the frame. Also be sure to specify the number of database records to be displayed in the frame, as well as the distance between each record.

To display a scrollbar in the frame that can be used to scroll through database records, check the 'Display Scrollbar' check box.

Frame Title:

Records Displayed: 1

Distance Between Records: 0

☐ Display Scrollbar

Laying Out a New Data Block with the Layout Wizard (continued)

Rows Page

1. Enter a title in the Frame Title field.
2. Enter the number of records that you want to display at run time in the Records Displayed field.
3. Enter the physical distance (in the coordinate system unit of the form) between records if you are displaying more than one record at a time.
4. You can select the Display Scrollbar check box to display a scroll bar next to the frame (common for multirecord data blocks).

Finish Page

Click Finish to create a new frame and lay out the selected items for the new data block. The Layout Wizard steps are complete.

Note: Once you complete the Layout Wizard steps, you can view the layout in the Layout Editor, where you can customize or modify the layout if necessary.

Data Block Functionality

Once you create a data block with the wizards, Forms Builder automatically creates:

- **A form module with database functionality including query, insert, update, delete**
- **A frame object**
- **Items in the data block**
- **A prompt for each item**
- **Triggers needed to enforce database constraints if “Enforce data integrity” is checked**

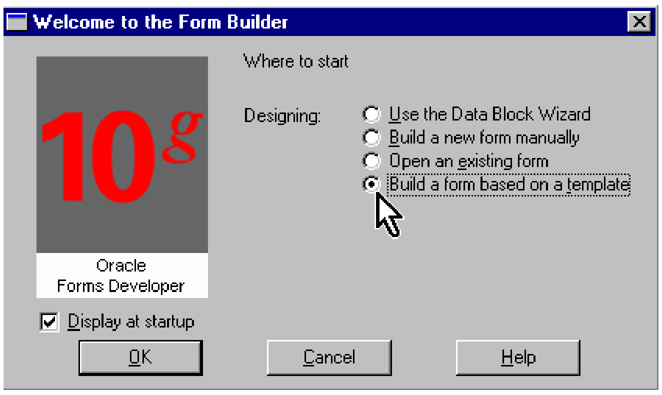
Data Block Functionality

Once you create a new data block by using the wizards, Forms Builder automatically creates the following objects for you:

- A new form module with a default menu (Basic database functionality such as querying, inserting, updating, and deleting is automatically available on the items in the base table block when you run the new form.)
- The new data block is created with default property values. These values can be modified to change the behavior of the form
- A frame object to arrange the items within the new data block
- An item for each database table column included in the data block (Each item is assigned default property values to match the underlying column specifications.)
- A prompt for each item in the data block (The default prompt is the name of the column.)

In addition, Forms Builder may create triggers to validate user input if you check “Enforce data integrity” in the table page of the Data Block Wizard.

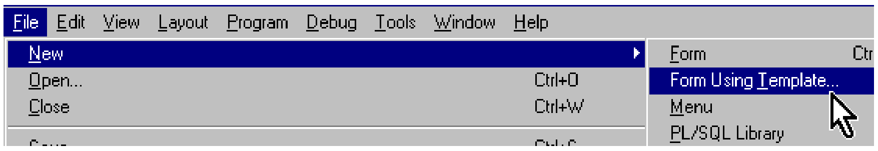
Template Forms


Welcome to the Form Builder

Where to start

Designing: ○ Use the Data Block Wizard ○ Build a new form manually ○ Open an existing form [✓ Build a form based on a template]

[✓ Display at startup]



Template Forms

You can create a new form based on standard template forms, so that you can provide other team members with a default starting point. Templates typically include generic objects, such as graphics, toolbars, and program units. You can define standard window layouts, standard toolbars, and other common objects that you want to include in new forms.

Creating a Form Based on a Template

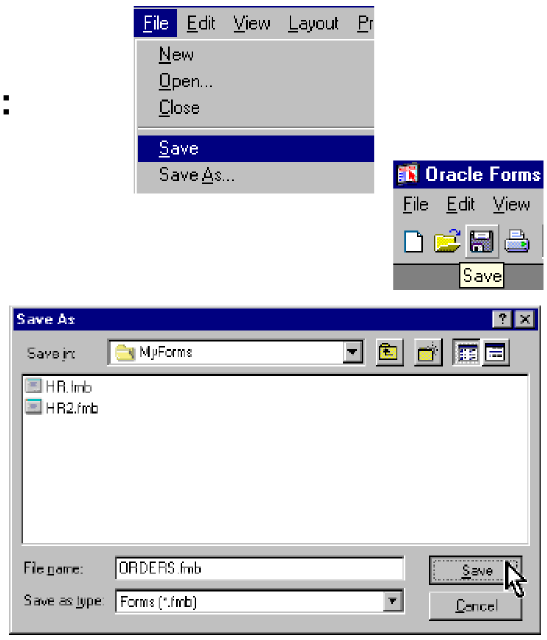
To create a form based on a template, do one of the following:

- Start Forms Builder. In the Welcome to the Forms Builder dialog box, select the "Build a form based on a template" option, and then click OK.
- Select File > New > Form Using Template from the menu.

Saving a Form Module

To save the form module:

- **Select File > Save**
  OR
  Click the Save icon
- **Enter a filename**
- **Navigate to desired location**
- **Click Save**



Saving a Form Module

To save the form module definition, perform one of the following:

- Select File > Save
- Select the Save icon

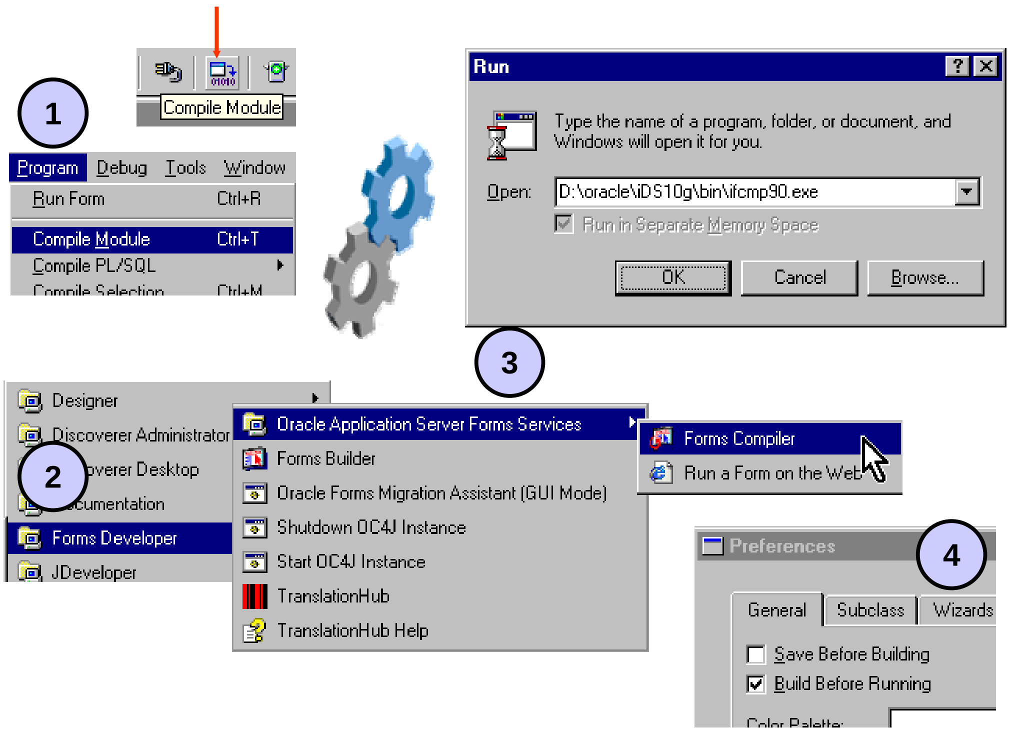
Both of these options display the Save As dialog box for the initial save. In the dialog box, do the following:

1. Enter a file name.
2. Navigate to the directory where you wish to save the file.
3. Click Save.

Note

- When you save a form, a `.fmb` file is produced. This saved definition of a form in the file system is not executable, and can be opened only by Forms Builder.
- When you work with more than one module at a time, Forms Builder separately keeps track of the changes that you make to each module. When you execute a Save command, only the current module is saved.

Compiling a Form Module


Run

Type the name of a program, folder, or document, and Windows will open it for you.

Open: D:\oracle\iDS10g\bin\ifcmp90.exe

☑ Run in Separate Memory Space

Preferences

☐ Save Before Building

☑ Build Before Running

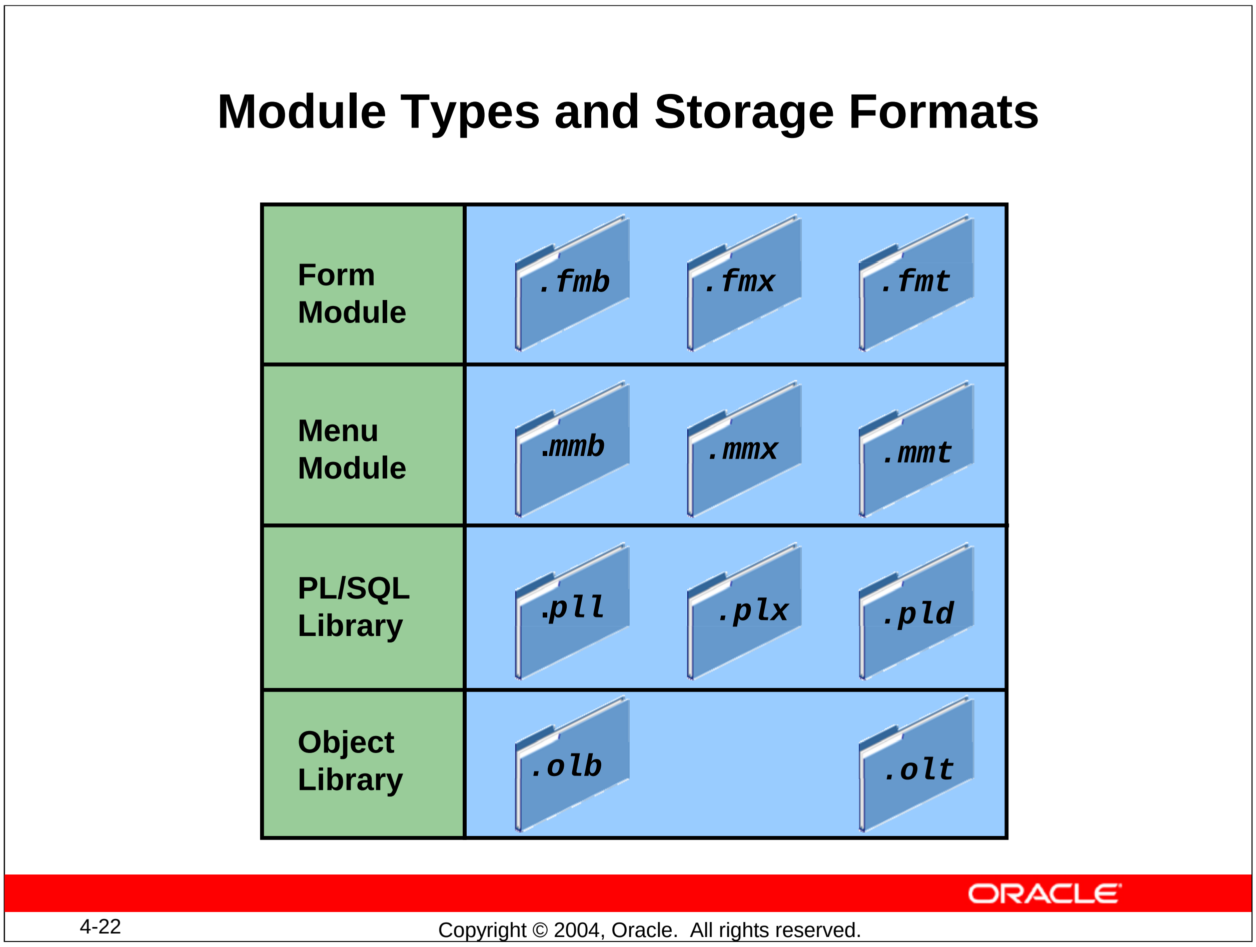
4-21

Compiling a Form Module

Before you can run a form, you must compile an executable (`.fmx`) file from the design (`.fmb`) file that you created in the Forms Builder. Compiling a form (or menu) module creates the needed executable file.

	Action	Type of Compilation
1.	With module open in Form Builder, select Program > Compile Module (or click the Compile Module icon)	Explicit
2.	Launch the Form Compiler component from Windows Start menu	Explicit
3.	Launch the Form Compiler component from the command line (`ifcmp90.exe`)	Explicit
4.	Set Build Before Running preference	Implicit

Note: Compiling and saving are two independent tasks. Performing one does not automatically accomplish the other. Both tasks must occur separately.



Module Types and Storage Formats

Form Module	.fmb	.fmx	.fmt
Menu Module	.mmb	.mmx	.mmt
PL/SQL Library	.pll	.plx	.pld
Object Library	.olb		.olt

ORACLE

4-22

Module Types and Storage Formats

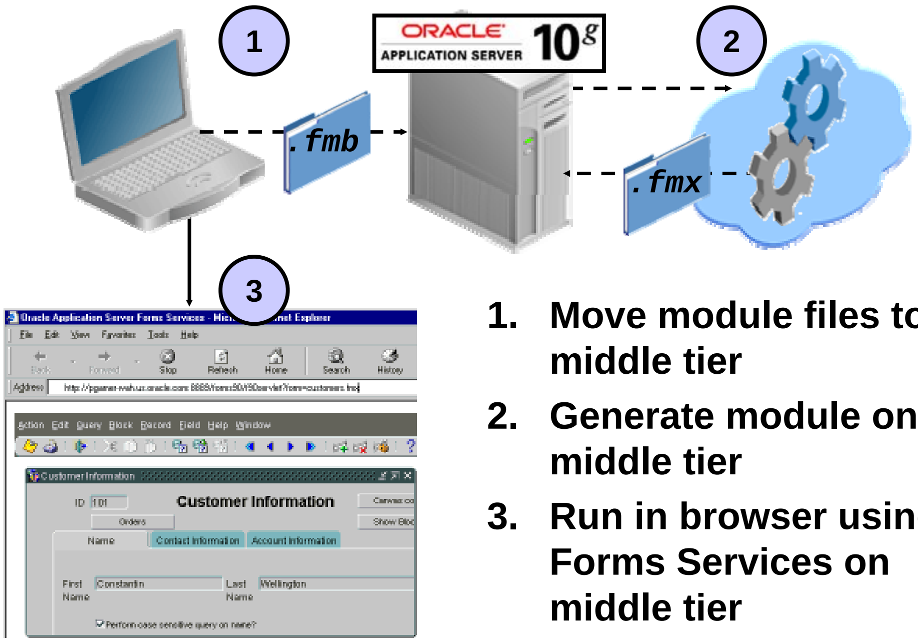
When you create form modules, menu modules, and library documents in the Forms Builder, they are stored in source files (`.fmb`, `.mmb`, and `.pll`) that have a binary format and are portable across platforms. The executable application files (`.fmx`, `.mmx`, and `.plx`) are also in a binary format; however, they are not portable across platforms.

Module Type and Storage Formats (continued)

Module/ Document	Extension	Storage Format	Portable
Form	`.fmb`	Form module binary	Yes
	`.fmx`	Form module executable; executable	No
	`.fmt`	Form module text	Yes
Menu	`.mmb`	Menu module binary	Yes
	`.mmx`	Menu module executable; executable	No
	`.mmt`	Menu module text	Yes
PL/SQL Library	`.pll`	PL/SQL Library document binary	Yes
	`.plx`	PL/SQL Library document executable (no source)	No
	`.pld`	PL/SQL Library document text	Yes
Object Library	`.olb`	Object Library module binary	Yes
	`.olt`	Object Library module text	Yes

Note: `.pll` is portable but requires recompilation, because it contains both source and compiled pcode.

Deploying a Form Module



1. **Move module files to middle tier**
2. **Generate module on middle tier**
3. **Run in browser using Forms Services on middle tier**

Deploying a Form Module

Although you may test a form on your client machine, for production applications you usually deploy the module to a middle tier machine. This machine may be running on a different platform; if so, you need to recompile the module once you transfer it to the middle tier.

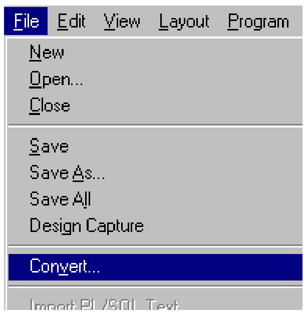
You can use an ftp utility to move the `.fmb` and other needed files to the middle tier machine, into a directory specified in `FORMS90_PATH`. If the platform is the same as your development platform, you can move the `.fmx` and other executable files there as well. If it is a different platform, you can invoke the Forms Compiler on the middle tier to recompile the module files.

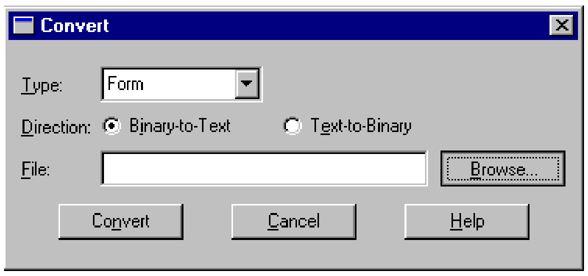
Once the executables have been placed on the middle tier, you can invoke the application in a browser, using a URL that points to the Forms Servlet on the middle tier Web server.

Instructor Note

The standard classroom setup for this course does not include Oracle Application Server. You will perform demonstrations and students will complete practices by running forms on local machines using OC4J, installed as part of Oracle Developer Suite on each student and instructor machine.

Text Files and Documentation




Convert

Type: Form

Direction: [✓ Binary-to-Text] Text-to-Binary

File:

- **Convert a binary file to a text file.**
- **Create an ASCII file for a form module.**

Producing Text Files and Documentation

The files normally produced by saving and generating modules are in binary format. To convert a binary file to text, perform the following:

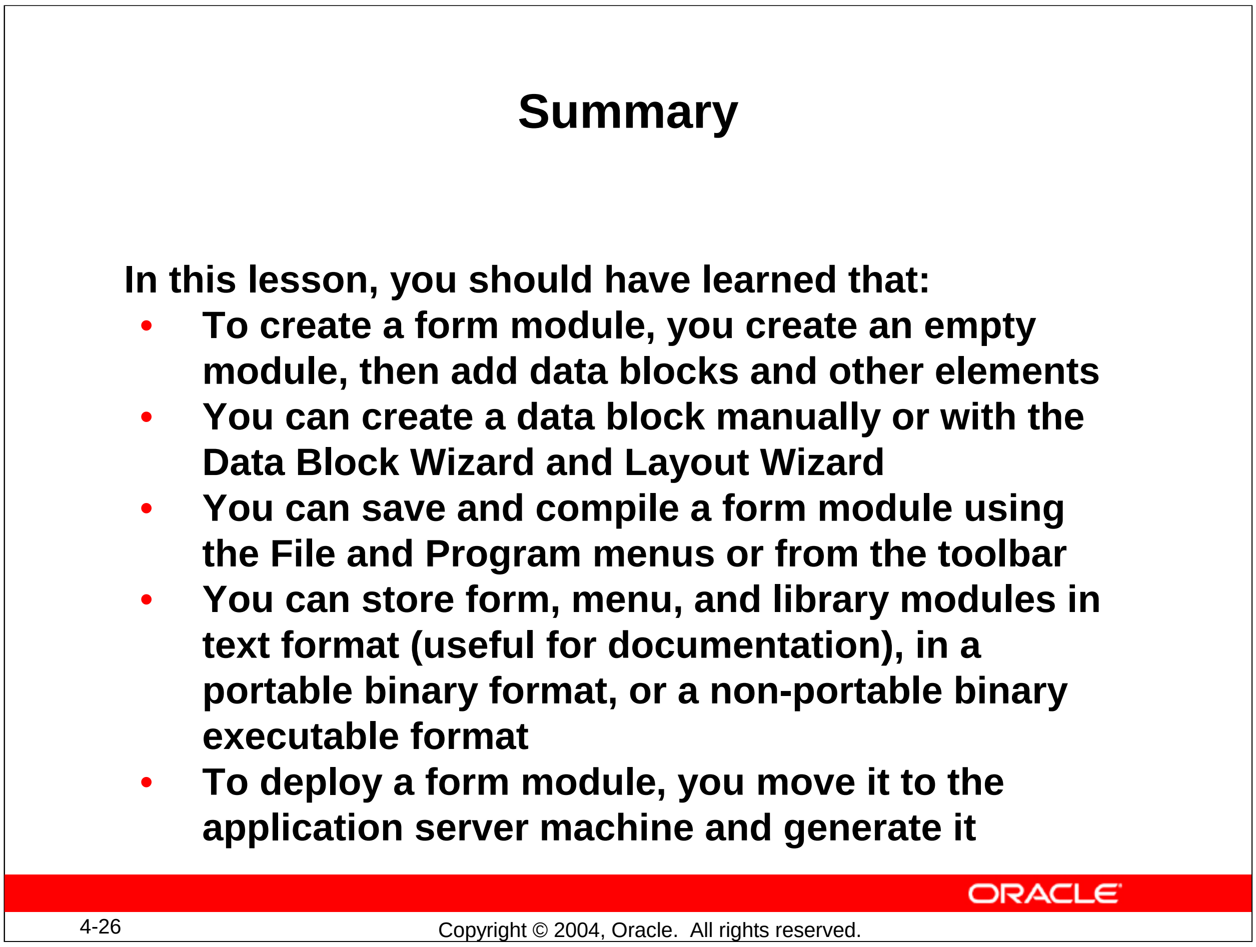
1. Select File > Convert.
   This opens the Convert dialog box.
2. Select the type of module (Form, Menu, PL/SQL Libraries, Object Libraries), the file to convert, and the direction (Binary-to-Text).
3. Select Convert. This produces a text file for the module with the name `<module>.fmt`.

To produce documentation for your module, perform the following:

1. Select the module to be documented in the Object Navigator.
2. Select File > Administration > Object List Report from the menu. This produces an ASCII file with the name `<module>.txt`.

You can also produce documentation in other ways not covered in this course:

- Use Forms API to produce custom documentation of the module.
- Convert the module to a `.xml` file with a separate utility included with Oracle Developer Suite.



Summary

In this lesson, you should have learned that:

- **To create a form module, you create an empty module, then add data blocks and other elements**
- **You can create a data block manually or with the Data Block Wizard and Layout Wizard**
- **You can save and compile a form module using the File and Program menus or from the toolbar**
- **You can store form, menu, and library modules in text format (useful for documentation), in a portable binary format, or a non-portable binary executable format**
- **To deploy a form module, you move it to the application server machine and generate it**

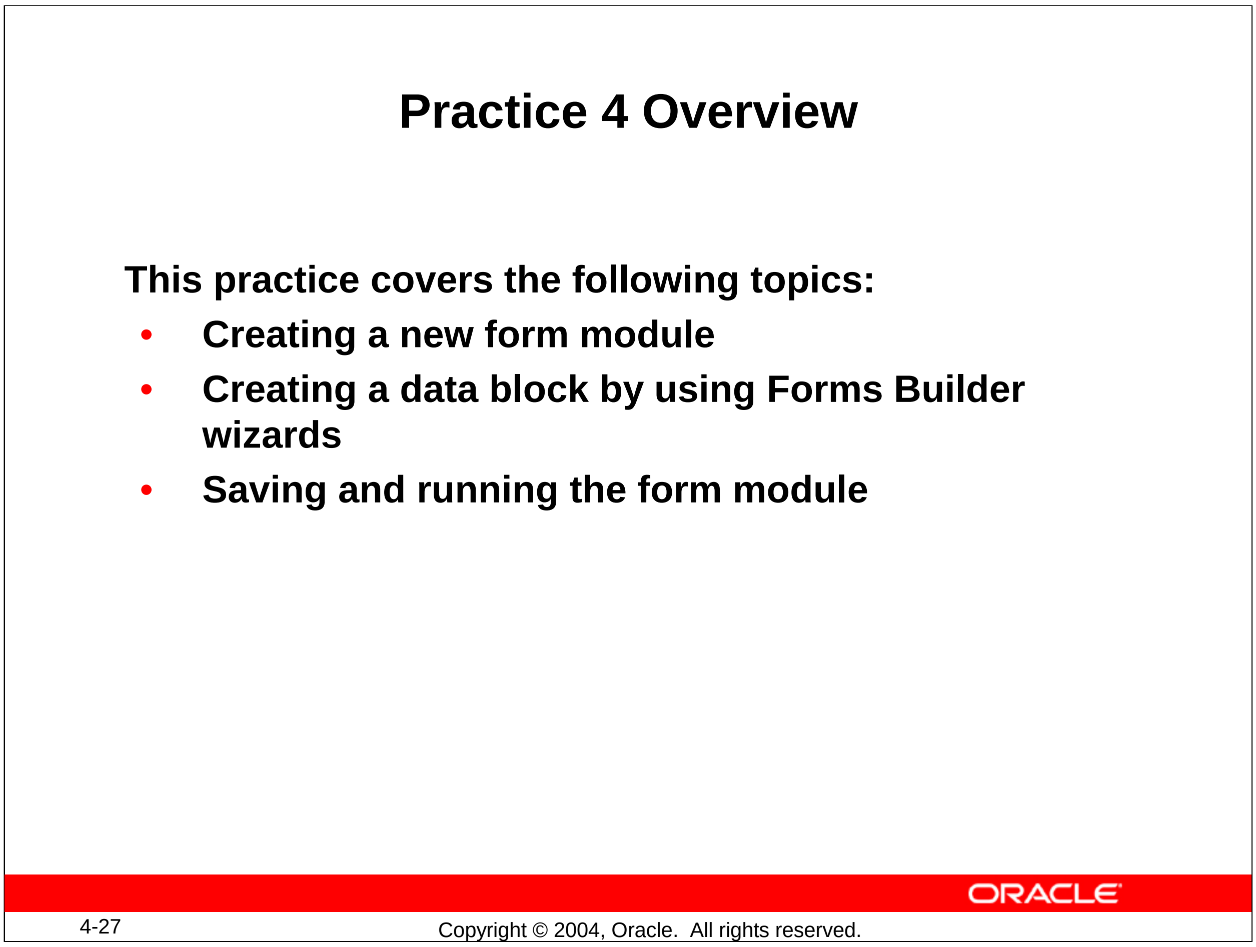
ORACLE

4-26

Summary

This lesson presented information about:

- Building a new form module by using the following methods:
  - Forms Builder wizards
  - Manually
  - Template form
- Using the Data Block Wizard to create a new data block with its associated data sources quickly and easily
- Using the Layout Wizard to quickly lay out the new data block contents for user interaction
- Saving the form module to preserve its definition; compiling it to get an executable file; running the form module to test it
- Using several module types and storage formats that are available for form modules, menu modules, PL/SQL Library documents, and Object Library modules, including a text format that can be used for documentation



Practice 4 Overview

This practice covers the following topics:

- Creating a new form module
- Creating a data block by using Forms Builder wizards
- Saving and running the form module

4-27

Practice 4 Overview

In this practice you create one new form module. You create a single-block form that displays a single record.

- Create a new form module called CUSTOMERS. Create a new data block in this form by using the Forms Builder wizards, and base it on the `CUSTOMERS` table. Using the Layout Editor, reposition the items in this block to match the screenshot provided.
- Save and run the new form module on the Web.

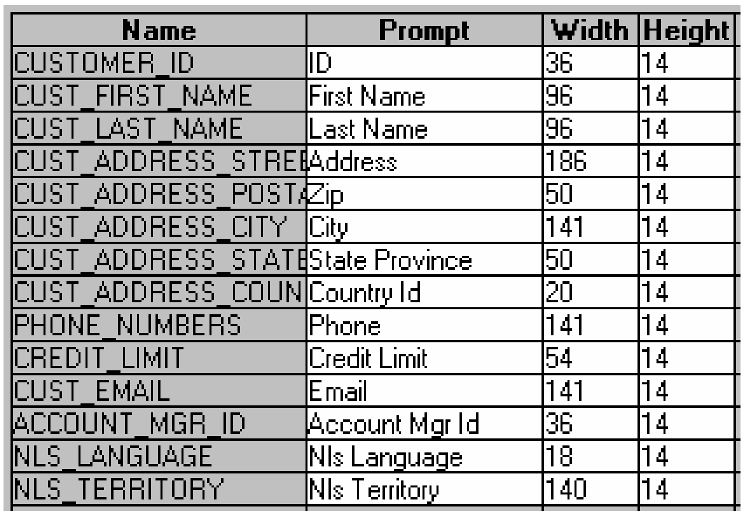
Note: For solutions to this practice, see Practice 4, in Appendix A, "Practice Solutions."

Instructor Note

Advise the students not to spend too much time trying to get the layout of the CUSTOMERS block to exactly match the screenshot provided in the practice, because the layout for the items in this block will be modified in a later lesson.

Practice 4

1. Create a new form module.
   Create a new single block by using the Data Block Wizard.
   Base it on the `CUSTOMERS` table and include all columns.
   Display the `CUSTOMERS` block on a new content canvas called `CV_CUSTOMER` and show just one record at a time. Set the frame title to Customers. Set column names and widths as shown in the following table:

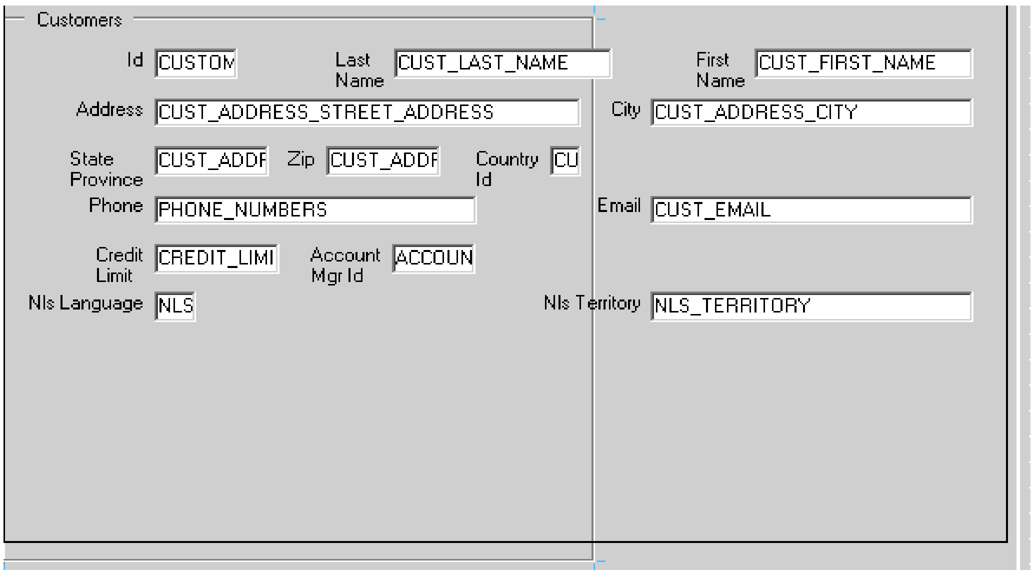


Name	Prompt	Width	Height
CUSTOMER_ID	ID	36	14
CUST_FIRST_NAME	First Name	96	14
CUST_LAST_NAME	Last Name	96	14
CUST_ADDRESS_STREE	Address	186	14
CUST_ADDRESS_POSTA	Zip	50	14
CUST_ADDRESS_CITY	City	141	14
CUST_ADDRESS_STATE	State Province	50	14
CUST_ADDRESS_COUN	Country Id	20	14
PHONE_NUMBERS	Phone	141	14
CREDIT_LIMIT	Credit Limit	54	14
CUST_EMAIL	Email	141	14
ACCOUNT_MGR_ID	Account Mgr Id	36	14
NLS_LANGUAGE	Nls Language	18	14
NLS_TERRITORY	Nls Territory	140	14

2. Save the new module to a file called `CUSTG`*`XX`*, where *XX* is the group number that your instructor has assigned to you.
3. Run your form module and execute a query.
   Navigate through the fields. Exit the run-time session and return to Forms Builder.
4. Change the form module name in the Object Navigator to CUSTOMERS.

Practice 4 (continued)

5. In the Layout Editor, reposition the items and edit item prompts so that the canvas resembles the following:
   Hint: First resize the canvas. Do not attempt to resize the frame, or the items will revert to their original positions.



Customers

Id	CUSTOM	Last Name	CUST_LAST_NAME	First Name	CUST_FIRST_NAME
Address	CUST_ADDRESS_STREET_ADDRESS			City	CUST_ADDRESS_CITY
State Province	CUST_ADDF	Zip	CUST_ADDF	Country Id	CU
Phone	PHONE_NUMBERS			Email	CUST_EMAIL
Credit Limit	CREDIT_LIMI	Account Mgr Id	ACCOUN		
Nls Language	NLS			Nls Territory	NLS_TERRITORY

6. Save and compile the form. Click Run Form to run the form. Execute a query.
7. Exit the run-time session and close the browser window.
   No formal solution.

Creating a Master-Detail Form

Schedule:	Timing	Topic
	40 minutes	Lecture
	30 minutes	Guided Practice
	70 minutes	Total

Objectives

After completing this lesson, you should be able to do the following:

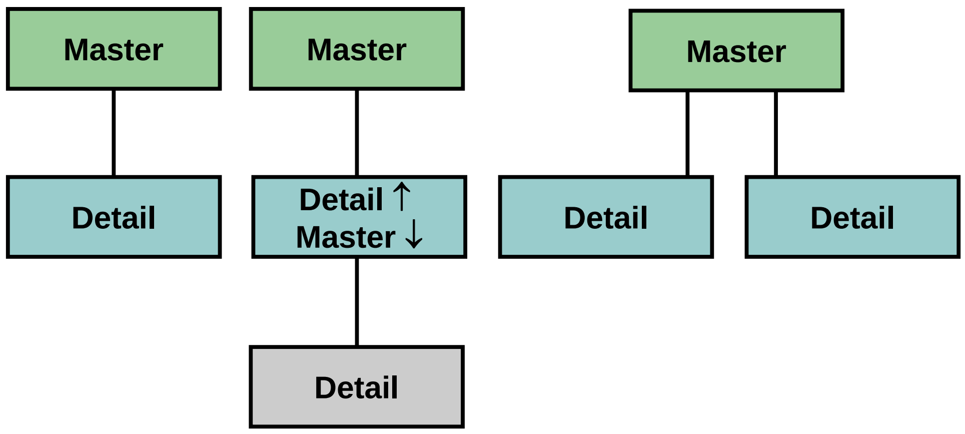
- Create data blocks with relationships
- Modify a data block
- Modify the layout of a data block
- Run a master-detail form

Introduction

Overview

A very common type of application is to show a record or records from one table along with associated records from another table. Forms Developer gives you the ability to quickly define additional data blocks and to define relations between the new blocks and any existing blocks in the form. This lesson shows you how to create a master-detail form and how to modify a data block and its layout.

Form Block Relationships



Creating Data Blocks with Relationships

A form module can contain one or more data blocks. Each data block can stand alone or be related to another data block.

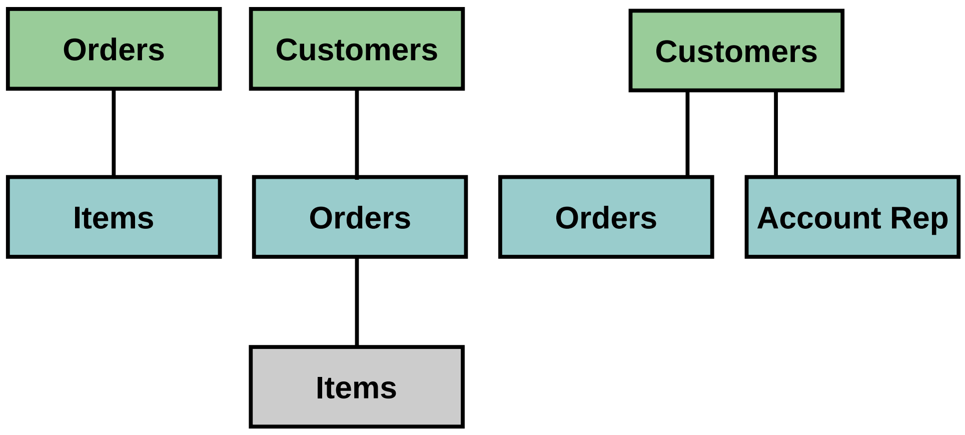
Master-Detail Relationship

A master-detail relationship is an association between two data blocks that reflects a primary-foreign key relationship between the database tables on which the two data blocks are based. The master data block is based on the table with the primary key, and the detail data block is based on the table with the foreign key. A master-detail relationship equates to the one-to-many relationship in the entity relationship diagram.

A Detail Block Can Be a Master

You can create block relationships in which the detail of one master-detail link is the master for another link.

Form Block Relationships



Creating Data Blocks with Relationships (continued)

A Master Block can have more Details

You can create more than one detail block for a master block.

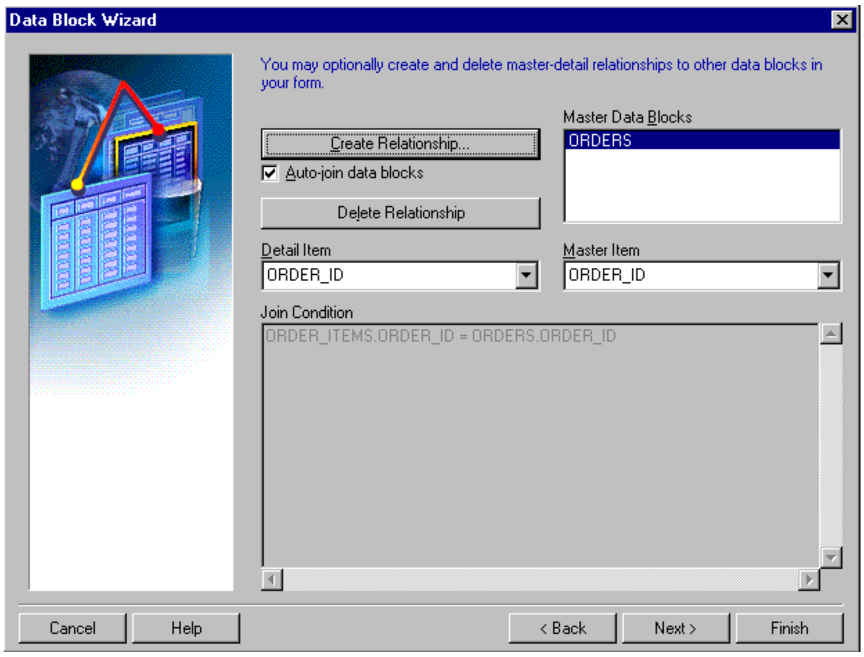
Note: The following are examples of the master-detail structure:

- **Master-detail:** Order-items
- **Master-detail-detail:** Customer-order-items
- **Master-2*detail:** Customer-order and customer-account reps

Instructor Note

The Data Block Wizard also enables you to select an Oracle object when creating relationships.

Data Block Wizard: Master-Detail Page


Data Block Wizard

You may optionally create and delete master-detail relationships to other data blocks in your form.

Master Data Blocks: ORDERS

Create Relationship...

[✓] Auto-join data blocks

Delete Relationship

Detail Item: ORDER_ID

Master Item: ORDER_ID

Join Condition:

```
ORDER_ITEMS.ORDER_ID = ORDERS.ORDER_ID
```

Creating a Master-Detail Form Module with the Data Block Wizard

You can build a master-detail form module either by creating a relation between a master and detail block explicitly, or implicitly by using the Data Block Wizard.

1. Create the master block as described earlier in this lesson in the topic Creating a New Data Block.
2. Invoke the Data Block Wizard in the Object Navigator.
3. Follow the same steps as before to create a new data block in the Data Block Wizard until you come to the Master-Detail page. On this page, select the “Auto-join data blocks” check box and click Create Relationship.
   Note: If the “Auto-join data blocks” check box is clear, the Data Block dialog is displayed with a list of all data blocks in the form without any foreign key constraint names.

Creating a Master-Detail Form Module with the Data Block Wizard (continued)

4. Select a master data block in the Data Block dialog and click OK. The wizard automatically creates the join condition between the detail and master data blocks in the Join Condition field and displays the name of the master data block in the Master Data Blocks field.
   Note: If the "Auto-join data blocks" check box is clear, the wizard does not automatically create the join condition between the detail and master data blocks. You must use the Detail Item and Master Item pop-up lists to create a join condition manually.
5. Click Next; then complete the Data Block Wizard steps. Go through the Layout Wizard steps as described earlier in the previous lesson to finish creating and laying out the detail data block.
   Note: The master data block must exist in the form module before you create the detail block.

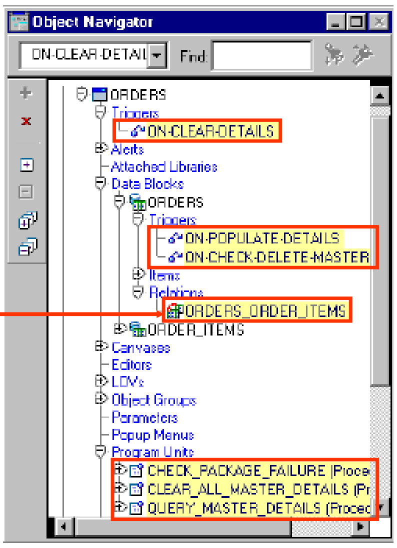
You can also create a relation by invoking the Data Block Wizard in reentrant mode.

Instructor Note

Demonstration: In the form you created for the earlier demo, create a second data block, using the Forms Builder wizards. Base the block on the `EMPLOYEES` table. In the Data Block Wizard, create a master-detail relationship between the new data block and the `DEPARTMENTS` data block.

Relation Object

- New relation object created in Object Navigator under master data block node
- Default name assigned: `MasterDataBlock_DetailDataBlock`
- Triggers and program units generated automatically

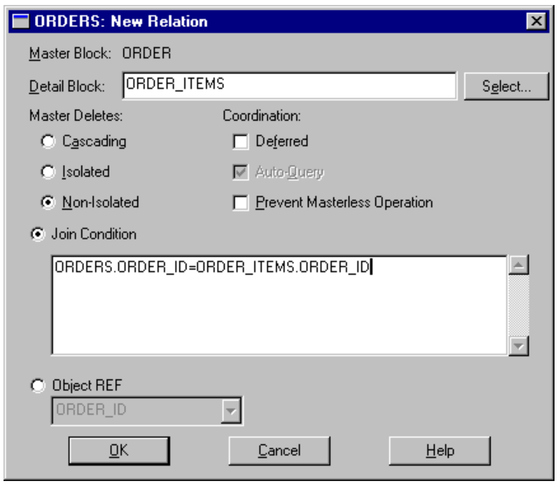


New Relation

Once you create a master-detail form module, the Data Block Wizard automatically creates a form object that handles the relationship between two associated data blocks. This object is called a *relation*. The following tasks occur automatically:

- The new relation object is created under the master data block node in the Object Navigator with default properties.
- The relation is given the following default name: MasterDataBlock_DetailDataBlock, for example `CUSTOMERS_ORDERS`.
- Triggers and program units are generated to maintain coordination between the two data blocks.

Creating a Relation Manually


ORDERS: New Relation

Master Block: ORDER

Detail Block: ORDER_ITEMS

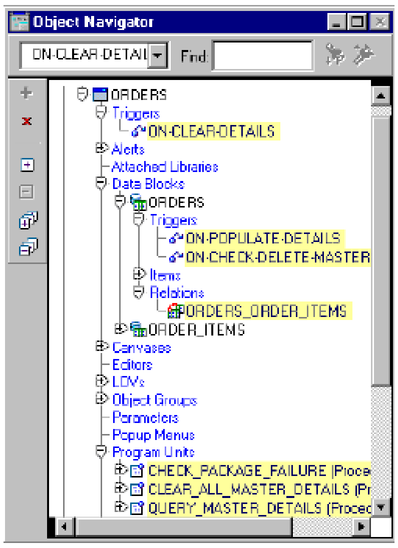
Master Deletes:	Coordination:
○ Cascading	☐ Deferred
○ Isolated	☑ Auto-Query
● Non-Isolated	☐ Prevent Masterless Operation

● Join Condition

```
ORDERS.ORDER_ID=ORDER_ITEMS.ORDER_ID
```

○ Object REF

ORDER_ID


Object Navigator

```
ORDERS
  Triggers
    ON-CLEAR-DETAILS
  Alerts
  Attached Libraries
  Data Blocks
    ORDERS
      Triggers
        ON-POPULATE-DETAILS
        ON-CHECK-DELETE-MASTER
      Items
      Relations
        ORDERS_ORDER_ITEMS
    ORDER_ITEMS
  Canvases
  Editors
  LOVs
  Object Groups
  Parameters
  Popup Menus
  Program Units
    CHECK_PACKAGE_FAILURE (Proce
    CLEAR_ALL_MASTER_DETAILS (Pr
    QUERY_MASTER_DETAILS (Proced
```

Creating a Relation Manually

You can create a relation either implicitly with the Data Block Wizard, or explicitly in the Object Navigator.

Explicit Relations

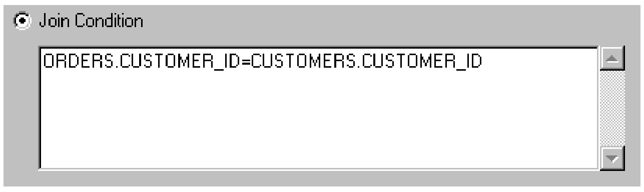
If a relation is not established when default blocks are created, you can create your own. To explicitly create a relation, perform the following steps:

1. Select the Relations node under the master block entry in the Object Navigator.
2. Click the Create icon. The New Relation window is displayed.
3. Specify the name of the detail block.
4. Choose your master delete property.
5. Choose your coordination property.
6. Specify the join condition.
7. Click OK.

The new relation, new triggers, and new program units are highlighted in the Object Navigator. Like implicitly created relations, the relation is given the default name of MasterDataBlock_DetailDataBlock, for example `CUSTOMERS_ORDERS`. The same PL/SQL program units and triggers are created automatically when you explicitly create a relation as when the relation is created implicitly.

Join Condition

- The join condition creates primary-foreign key link between blocks
- Define a join condition using:
  - Block and item names (not table and column names)
  - Do not precede names with colon
  - SQL equijoin syntax



ORDERS.CUSTOMER_ID=CUSTOMERS.CUSTOMER_ID

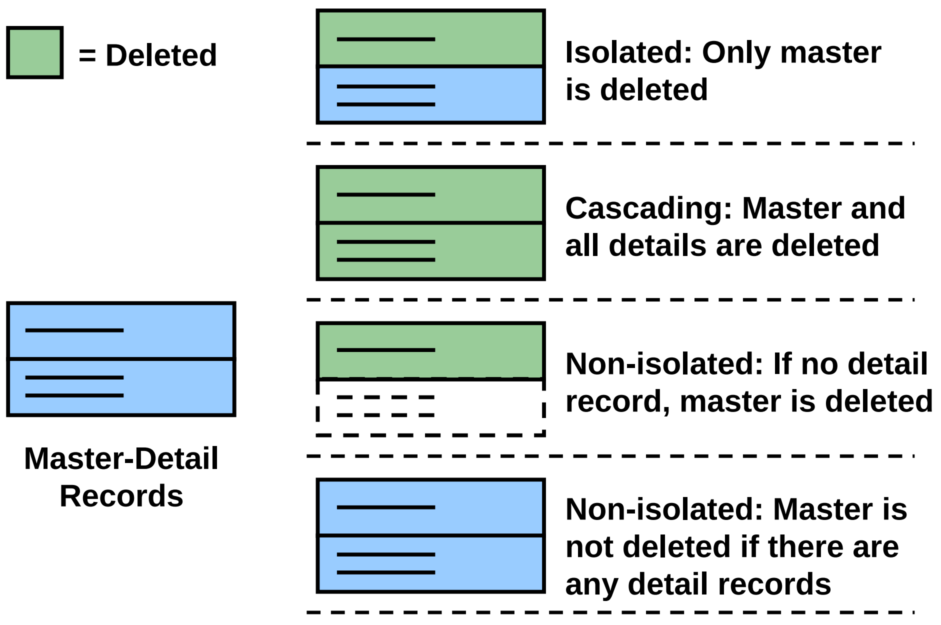
Join Condition

- **Use a join condition to:**
  - Create links between blocks using SQL
  - Alter links between blocks using SQL
- **Define a join condition using:**
  - Usual SQL equijoin condition syntax (necessary because Forms copies the value from the master block to the related item in the detail block)
  - Block names instead of the base table names (do not precede with colon)
  - Item names that exist in the form module instead of base table column names

Instructor Note

Demonstration: In your DEPT_EMP demo form, delete the relation you created using the Data Block Wizard. Explicitly add a relation between the Department and Employee blocks. Set the deletion property to Cascading and the coordination property to Deferred with Auto-Query.

Deletion Properties


= Deleted
Isolated: Only master is deleted
Cascading: Master and all details are deleted
Non-isolated: If no detail record, master is deleted
Non-isolated: Master is not deleted if there are any detail records
Master-Detail Records

Deletion Properties

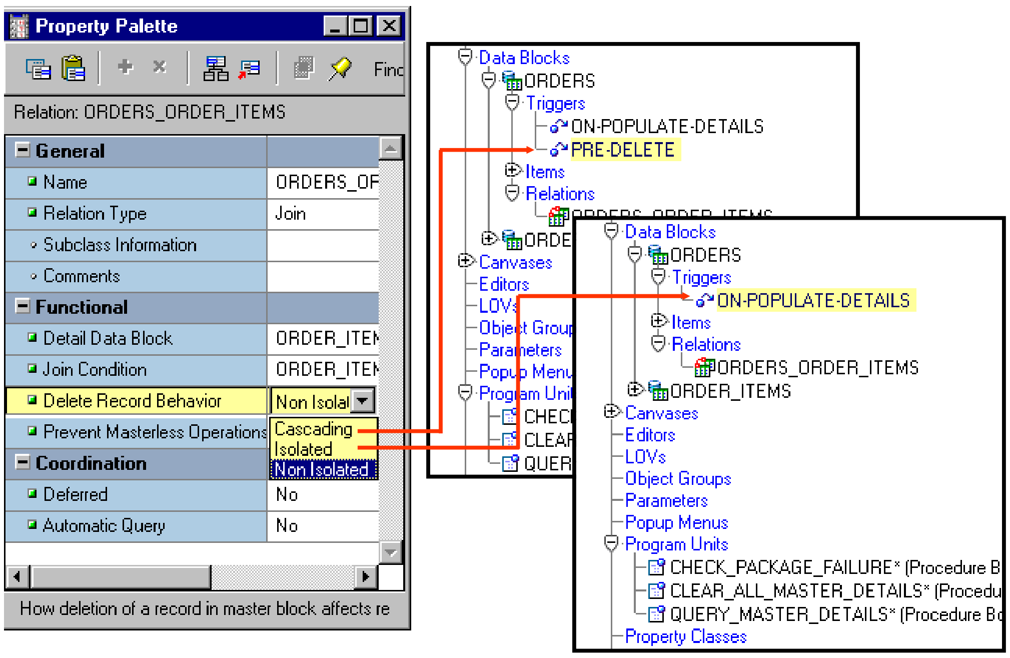
Master Deletions

You can prevent, propagate, or isolate deletion of a record in a master block when corresponding records exist in the detail block by setting the Delete Record Behavior property. For example, you can delete all corresponding line items when an order is deleted.

Property	Use
Non-Isolated	Prevents the deletion of the master record when detail records exist; the master record is deleted only if no detail records exist
Cascading	Deletes the detail records when a master record is deleted
Isolated	Deletes only the master record

Note: Although deleting with the cascading property may remove many detail records, the commit message shows only the number of records deleted from the master block.

Modifying a Relation



Property Palette

Relation: ORDERS_ORDER_ITEMS

General	
Name	ORDERS_OF
Relation Type	Join
Subclass Information	
Comments	
Functional	
Detail Data Block	ORDER_ITEM
Join Condition	ORDER_ITEM
Delete Record Behavior	Non Isolat
Prevent Masterless Operations	Cascading / Isolated / Non Isolated
Coordination	
Deferred	No
Automatic Query	No

How deletion of a record in master block affects re

```
Data Blocks
  ORDERS
    Triggers
      ON-POPULATE-DETAILS
      PRE-DELETE
    Items
    Relations
      ORDERS_ORDER_ITEMS
  ORDE
Canvases
Editors
LOVs
Object Group
Parameters
Popup Menu
Program Uni
  CHECI
  CLEAF
  QUER
```

```
Data Blocks
  ORDERS
    Triggers
      ON-POPULATE-DETAILS
    Items
    Relations
      ORDERS_ORDER_ITEMS
  ORDER_ITEMS
Canvases
Editors
LOVs
Object Groups
Parameters
Popup Menus
Program Units
  CHECK_PACKAGE_FAILURE* (Procedure B
  CLEAR_ALL_MASTER_DETAILS* (Procedu
  QUERY_MASTER_DETAILS* (Procedure Bo
Property Classes
```

ORACLE

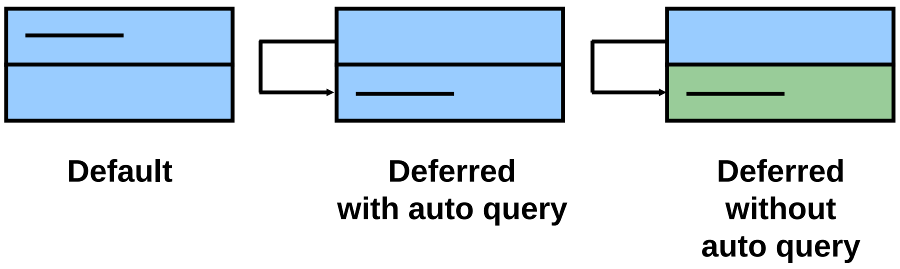
Modifying a Relation

You can alter the relation properties to affect the way deletes and block coordination are handled.

What Happens When you Modify a Relation?

- If you change the Delete Record Behavior property from the default of Non-Isolated to Cascading, the On-Check-Delete-Master trigger is replaced with the Pre-Delete trigger.
- If you change the Delete Record Behavior property from the default of Non-Isolated to Isolated, the On-Check-Delete-Master trigger is removed.

Coordination Properties



Coordination

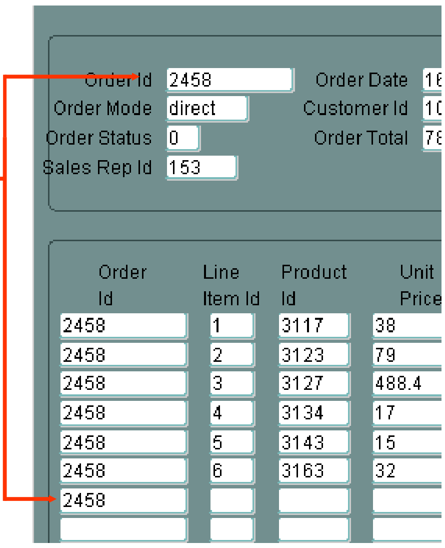
Setting the coordination property controls how the detail records are displayed when a master block is queried. For example, you can defer querying the line items for an order until the operator navigates to the item block.

Coordination Property	Use
Default	Forces coordination of blocks to occur whenever the master record is changed by a user or a trigger
Deferred with Automatic Query	Postpones potentially expensive detail query processing until the cursor visits the related blocks
Deferred without Automatic Query	Allows entry of additional query criteria in the detail block prior to querying
Prevent Masterless Operations	Ensures that the detail block cannot be queried or used to insert records when a master record is not displayed

Note: In the New Relation dialog, setting the Deferred property to Yes enables the Auto Query check box.

Running a Master-Detail Form Module

- Automatic block linking for:
  - Querying
  - Inserting
- Default deletion rules: Cannot delete master record if detail records exist



Order Id	2458	Order Date	1
Order Mode	direct	Customer Id	1
Order Status	0	Order Total	7
Sales Rep Id	153		

Order Id	Line Item Id	Product Id	Unit Price
2458	1	3117	38
2458	2	3123	79
2458	3	3127	488.4
2458	4	3134	17
2458	5	3143	15
2458	6	3163	32
2458			

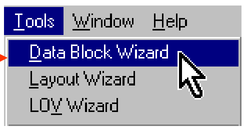
Running a Master-Detail Form Module

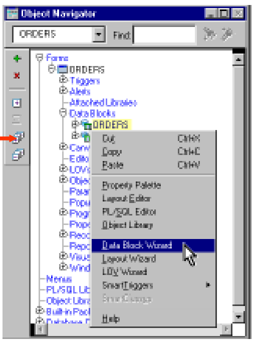
When you run your master-detail form module you will find that:

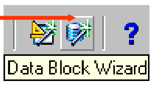
- Querying the master data block immediately retrieves corresponding detail records.
- Deleting a master record is prevented if detail records exist.
  Note: You can change the above behavior by modifying the relation object properties.
- Inserting a detail record automatically associates it with the currently displayed master.

Modifying the Structure of a Data Block

- **Reentrant Data Block Wizard:**
  1. **Select frame or object in Layout Editor, or data block or frame in Object Navigator**
  2. **Select Tools > Data Block Wizard OR Right-click and select Data Block Wizard OR Click Data Block Wizard**
- **Object Navigator:**
  – **Create or delete items**
  – **Change item properties**
- **Block Property Palette: Change property values**







Modifying the Structure of a Data Block

Once you create a data block, you may want to customize or modify it by performing one of the following:

- Reenter the Data Block Wizard, and use it to make the changes.
- Make manual changes, such as adding or deleting items, in Object Navigator.
- Change the property values of the block by using the Property Palette.

Invoking the Data Block Wizard in Reentrant Mode

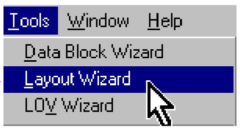
A very powerful feature of the Data Block Wizard is its ability to operate in reentrant mode. Use the reentrant mode to modify the data block, even if the block was not originally created with the Data Block Wizard.

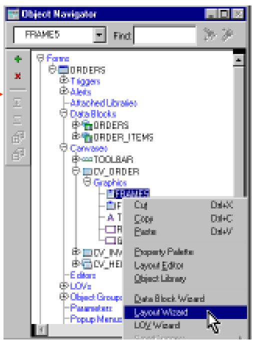
To invoke the Data Block Wizard in reentrant mode:

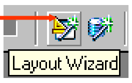
1. Select the frame or a component of the block in either the Object Navigator or the Layout Editor.
2. Invoke the Data Block Wizard by performing one of the following:
   - Select Tools > Data Block Wizard from the menu
   - Right-click and select Data Block Wizard from the pop-up menu
   - Click Data Block Wizard

Modifying the Layout of a Data Block

- Reentrant Layout Wizard:
  - Select frame in Object Navigator or Layout Editor
  - Select Tools > Layout Wizard
  - OR
  - Right-click and select Layout Wizard OR
  - Click Layout Wizard
- Layout Editor:
  - Select Tools > Layout Editor
  - Make changes manually
- Frame Property Palette: Change property values







Modifying the Layout of a Data Block

You may want to customize or modify the layout of the data block items on the canvas. You can do this by doing one of the following:

- Reenter the Layout Wizard (see the next section), and use it to make the changes.
- Select Tools > Layout Editor to invoke the Layout Editor and make changes manually in the editor.
- Change the property values of the frame in its Property Palette.

Invoking the Layout Wizard in Reentrant Mode

A very powerful feature of the Layout Wizard is its ability to operate in reentrant mode. Use the reentrant mode to modify the layout of items in an existing frame, even if the frame was not originally created with the Layout Wizard.

Modifying the Layout of a Data Block (continued)

Invoking the Layout Wizard in Reentrant Mode (continued)

You can invoke the Layout Wizard in reentrant mode from the Object Navigator or the Layout Editor:

- **From the Object Navigator:**
  - Select the appropriate frame (under the Canvases node).
  - Select Tools > Layout Wizard.

  or

  - Right-click and select the Layout Wizard option.
- **In the Layout Editor:**
  - Select the appropriate frame.
  - Click Layout Wizard.

Note: Before you reenter the Layout Wizard, it is important to select the correct frame in the Object Navigator or the Layout Editor. If you overlook this when you reenter the Layout Wizard, you may create an additional frame instead of modifying the current frame.

Either method takes you to the Data Block page in the Layout Wizard. Use Next and Back as you do when not in reentrant mode, or go directly to a certain page by clicking its tab.

Instructor Note

Demonstration: Invoke the Layout Wizard in reentrant mode to show how to modify the layout of the DEPARTMENTS data block created in the previous demonstration. Make sure you select the correct frame before invoking the Layout Wizard.

Summary

In this lesson, you should have learned that:

- You can create data blocks with relationships by using the Data Block Wizard or by manually creating a Relation object
- When you run a master-detail form, block coordination is automatic depending on properties of the Relation object
- You can modify a data block manually or with the Data Block Wizard in reentrant mode
- You can modify the layout manually or with the Layout Wizard in reentrant mode

5-17

Summary

This lesson presented information about:

- Creating data blocks with a master-detail relationship
- Modifying the data block layout:
  - Using reentrant wizards
  - Changing frame properties

Practice 5 Overview

This practice covers the following topics:

- Creating a master-detail form module
- Modifying data block layout by using the Layout Wizard in reentrant mode
- Saving and running the form module

Practice 5 Overview

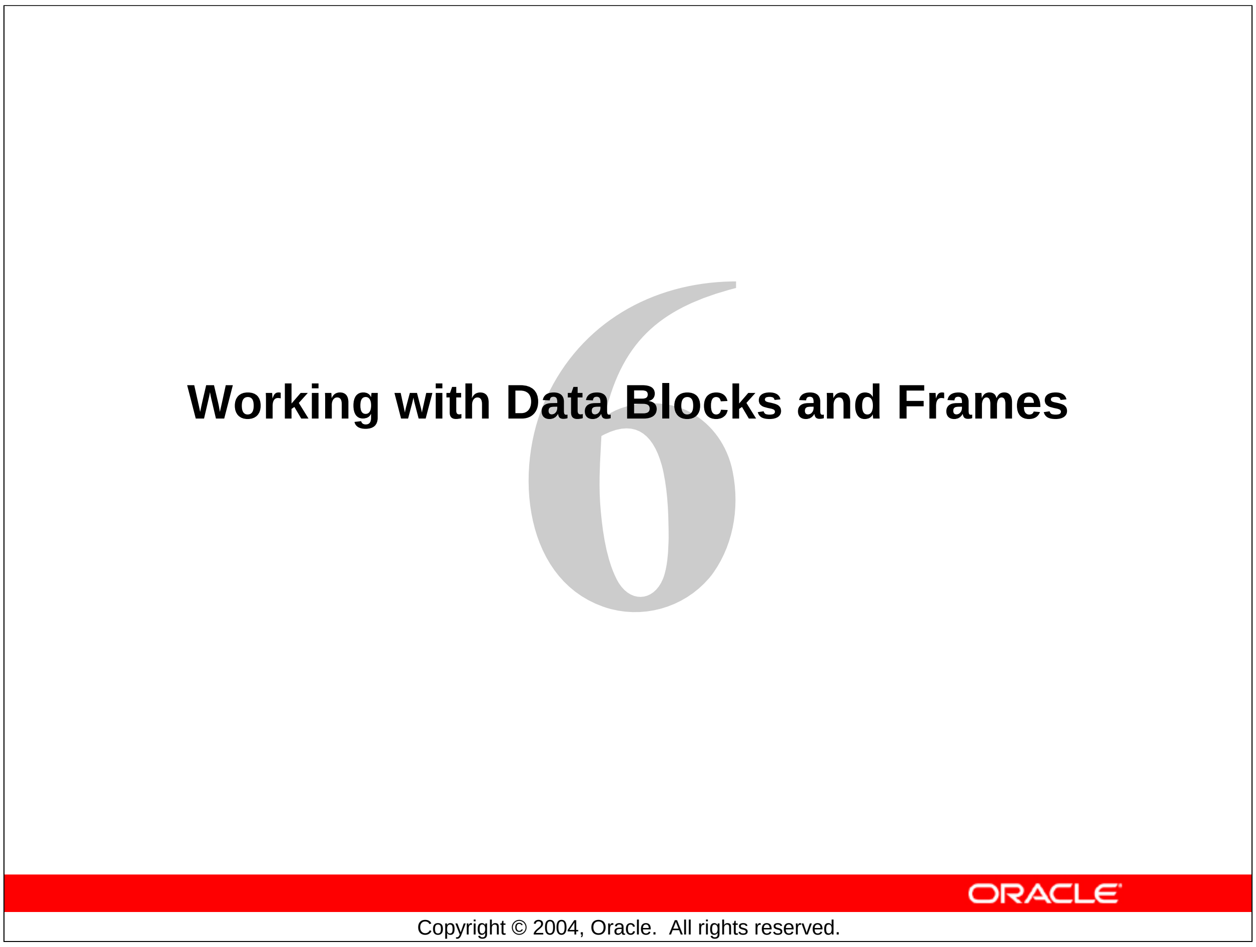
In this practice you will create a new form module that displays master-detail information.

- Create a master-detail form module called `ORDERS`. Create a master block based on the `ORDERS` table and a detail block based on the `ITEMS` table. Create a third data block that is not related to any other block in the form module. Base this block on the `INVENTORIES` table, and manually create a relation with the block based on the item table. Use the Forms Builder wizards to create all three data blocks.
- Invoke the Layout Wizard in reentrant mode, and change the layout of the `ITEMS` and `INVENTORIES` data blocks.
- Save and run the new form module on the Web.

Note: For solutions to this practice, see Practice 5 in Appendix A, “Practice Solutions.”

Practice 5

1. Create a new form module.
   Create a new block by using the Data Block Wizard.
   Base it on the ORDERS table and include all columns except ORDER_TOTAL and PROMOTION_ID.
   Display the ORDERS block on a new content canvas called CV_ORDER and show just one record at a time. Use a form style layout. Set the frame title to Orders.
2. Create a new block by using the Data Block Wizard.
   Base the block on the ORDER_ITEMS table and include all columns.
   Create a relationship and select the master block as ORDERS.
   Display all items except ORDER_ID on the CV_ORDER canvas.
   Display six records in this detail block on the same canvas as the master block.
   Use a tabular style layout and include a scroll bar.
   Change the order of the blocks in the Object Navigator, moving the ORDER_ITEMS block after the ORDERS block. Set the frame title to Items.
3. Save the new module to a file called ORDG*XX*, where *XX* is the group number that your instructor has assigned to you.
4. Create a new block based on INVENTORIES (do not create any relationships with other blocks at this time) to display on a different canvas.
   Base it on the INVENTORIES table.
   Display four records in this block and ensure that they are displayed on a new content canvas called CV_INVENTORY.
   Use a tabular style layout, and include a scroll bar.
   In the Object Navigator, move the INVENTORIES block after the ORDER_ITEMS block. Set the frame title to Stock.
   Do not create any relationships between blocks at this stage.
5. Explicitly create a relation called Order_Items_Inventories between the ORDER_ITEMS and INVENTORIES blocks.
   Ensure that line item records can be deleted independently of any related inventory.
   Set the coordination so that the Inventories block is not queried until you explicitly execute a query.
6. On the ORDER_ITEMS block, change the prompt for the Line Item ID item to Item# by using the reentrant Layout Wizard. First select the relevant frame in the Layout Editor, and then use the Layout Wizard.
7. In the INVENTORIES data block, change the prompt for Quantity on Hand to In Stock by using the Layout Wizard.
8. Save and compile your form module.
   Click Run Form to run your form.
   Execute a query.
   Navigate through the blocks so that you see the INVENTORIES block.
   Exit the run-time session, close the browser, and return to Forms Builder.
9. Change the form module name in the Object Navigator to ORDERS and save.



6

Working with Data Blocks and Frames

Schedule:	Timing	Topic
	45 minutes	Lecture
	40 minutes	Practice
	85 minutes	Total

Objectives

After completing this lesson, you should be able to do the following:

- Identify the components of the Property Palette
- Manage object properties
- Create and use Visual Attributes
- Control the behavior and appearance of data blocks
- Control frame properties
- Create blocks that do not directly correspond to database tables
- Delete data blocks and their components

ORACLE

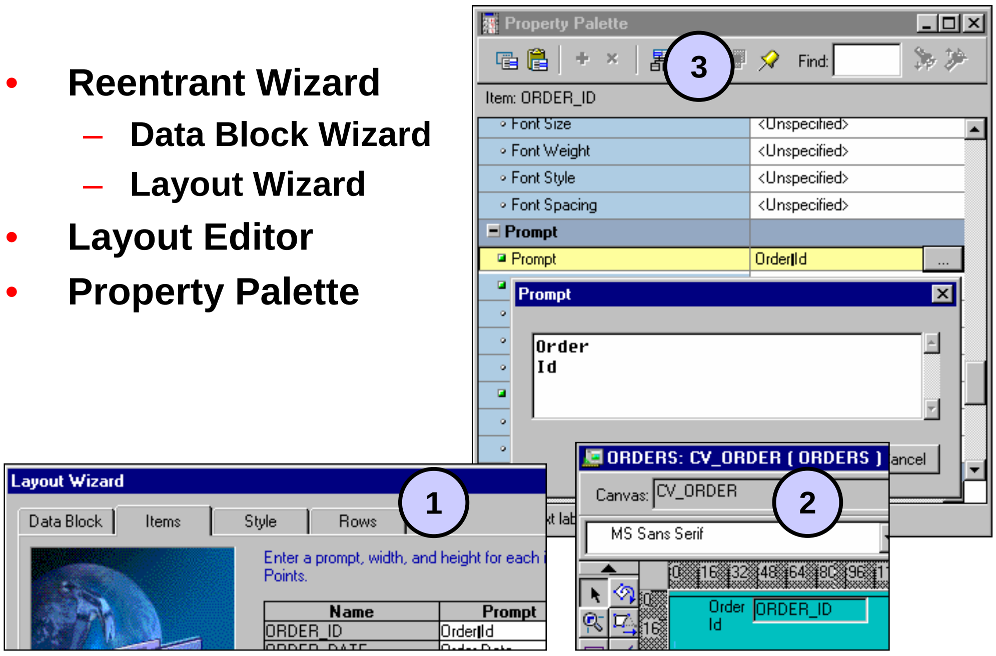
Introduction

Overview

In this lesson you will learn how to customize existing data blocks and modify frames. You will also learn how to include blocks that are not associated with the database.

Managing Object Properties

- Reentrant Wizard
  - Data Block Wizard
  - Layout Wizard
- Layout Editor
- Property Palette



Managing Object Properties

There are three ways to modify properties of Forms Builder objects:

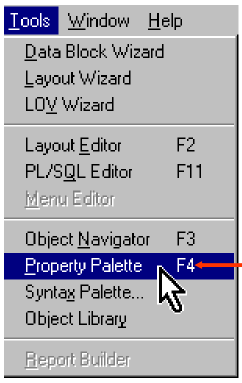
- **Reentrant Wizards:** You can modify data block and layout properties through the reentrant wizards, as explained in the previous lesson.
- **Layout Editor:** If the object appears on a canvas, you can modify properties by using the graphical Layout Editor.
- **Property Palette:** You can set individual properties for each Forms Builder object in its Property Palette.

The wizards, the Layout Editor, and the Property Palette all depict object properties. Changes made in one tools are reflected in the others. In the example above, the prompt for `ORDER_ID` is shown identically in:

1. Reentrant Layout Wizard
2. Layout Editor
3. Property Palette

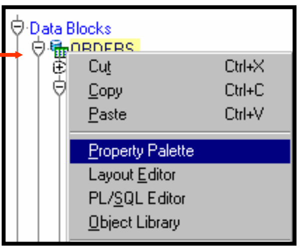
You can use the Property Palette to control the behavior and appearance of any Forms Builder object with a greater degree of granularity. In the Property Palette you can fine-tune objects that you have initially created in the wizards or the Layout Editor.

Displaying the Property Palette



To display the Property Palette, use one of the following methods:

- Select Tools > Property Palette (or use the shortcut key).
- Double-click the object icon in the Object Navigator.
- Double-click the object in the Layout Editor.
- Right-click the object icon in the Object Navigator.
- Right-click the object in the Layout Editor.



Property Palette

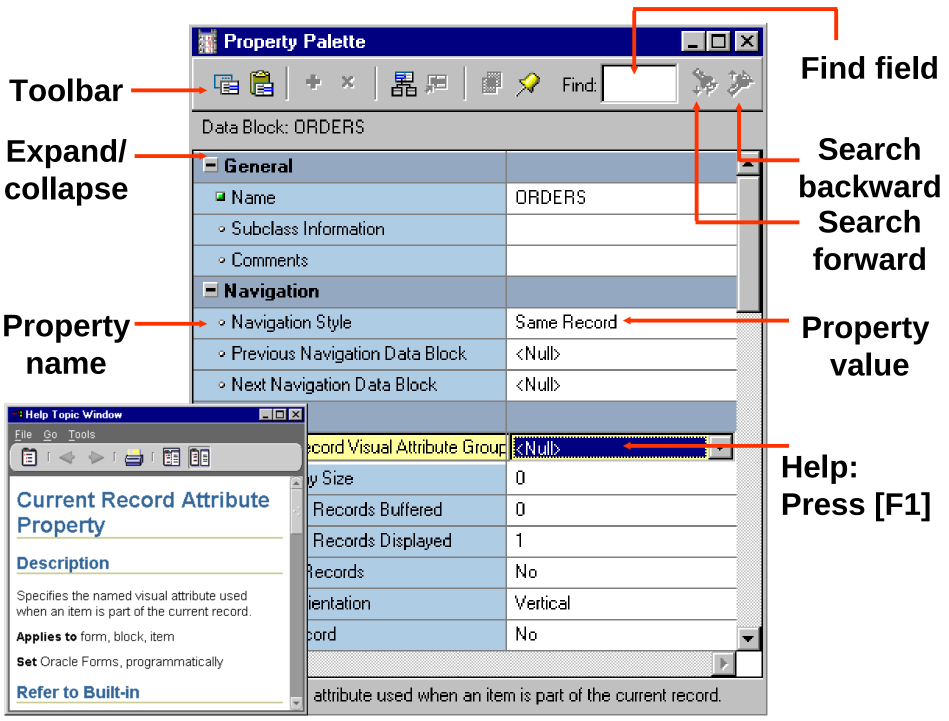
Every object in a form module, as well as the form module itself, has properties that dictate the object's behavior. When an object is first created, it is automatically assigned several property values by default. You can change these property values in the Property Palette.

Displaying the Property Palette

To display the Property Palette of an object, use one of the following methods:

- Select the object in the Object Navigator and then select Tools > Property Palette from the menu.
- Double-click the object icon for the object in the Object Navigator (except for code objects and canvases).
- Double-click an item in the Layout Editor.
- Right-click the object in the Layout Editor or the object icon in Object Navigator. From the pop-up menu, select the Property Palette option.

Property Palette: Features



Specifies the named visual attribute used when an item is part of the current record.

Applies to form, block, item

Set Oracle Forms, programmatically

6-5 Copyright © 2004, Oracle. All rights reserved.

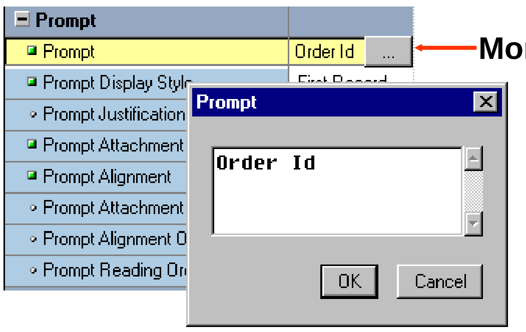
Property Palette: Features

Features of the Property Palette include the following:

Feature	Description
Property list	Displays a two-column list of names and values of properties that are valid for a particular object. Properties are grouped under functional headings or nodes. You can expand or collapse a node by using the plus and minus icons beside the node name.
Find field	Enables you to quickly locate a particular property. The Search Forward and Search Backward buttons enhance your search.
Toolbar	Consists of a series of buttons that provide quick access to commands.
Help	Obtain description, usage, and other information about any property by pressing `[F1]` with the property selected.

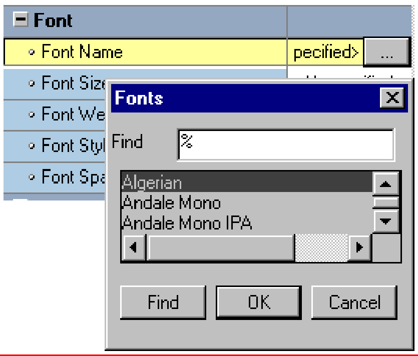
Property Controls

Text field

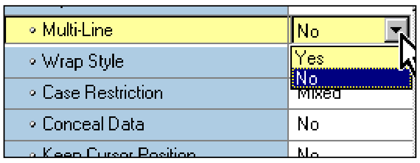


More button

LOV window



Pop-up list

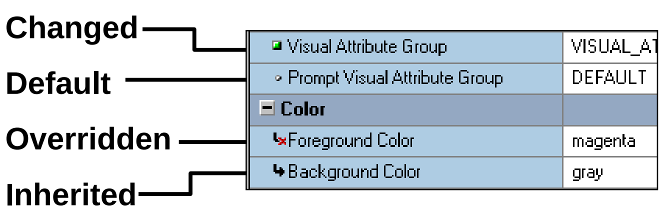


Using the Property Palette

Each form object has various types of properties. Properties are manipulated differently, depending on the property type. The following is a summary of the controls that are used in the Property Palette:

Property Control	Description
Text field	Displayed when the current property can be set by entering a text value. For longer text values an iconic button also appears, enabling you to open a text editor.
Pop-up list	Occurs where a fixed set of values, such as Yes or No, is allowed for the property. Click the down arrow to open the list and select a value.
LOV window	LOVs occur where a potentially large list of possible values is available. Click the iconic button in the property value column to invoke an LOV.
More button	Use this when more complex settings are needed. Click the More button to open the extra dialog.

Property Controls



Visual Attribute Group	VISUAL_AT
Prompt Visual Attribute Group	DEFAULT
Color	
Foreground Color	magenta
Background Color	gray

Property Palette Icons

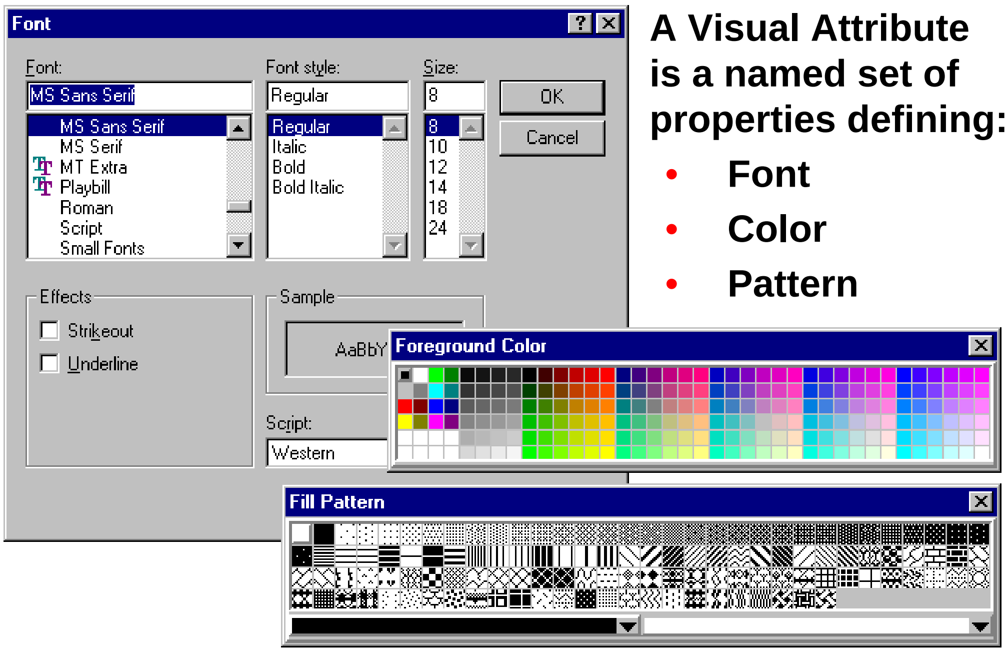
Each property in a Property Palette has an icon to its left. The following is a summary of these icons and their description:

Icon	**Description**
● Circle	Specifies that the property value is the default value
🟩 Square	Specifies that the property value has been changed from the default
↳ Arrow	Specifies that the property value is inherited
↳✗ Arrow with a cross	Specifies that the property value was inherited but has been overridden

Note: Once you activate the Property Palette, its window remains open until you close it. The window automatically displays the properties of each object you visit in the Layout Editor or Object Navigator. This is because, by default, the list of properties in the Property Palette is synchronized whenever you select an object.

You can turn the synchronization on or off for a specific palette by clicking Pin/Unpin in the Property Palette toolbar.

Visual Attributes



A Visual Attribute is a named set of properties defining:

- Font
- Color
- Pattern

Visual Attributes

Visual attributes are the font, color, and pattern properties that you set for form and menu objects.

A visual attribute is another object that you can create in the Object Navigator with properties such as font, color, and pattern combinations. Set the Visual Attribute Type property of the Visual Attribute to Title if you plan to apply it to objects such as frame titles, or to Prompt if it will be used for prompts. Otherwise, set Visual Attribute Type to Common, which is the default.

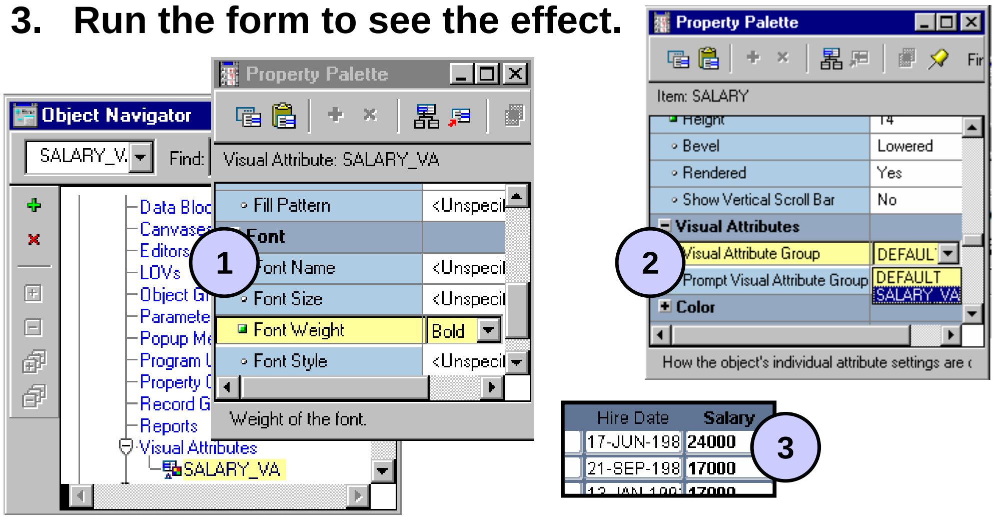
Every interface object in a forms application has a property called Visual Attribute Group, which determines how the individual Visual Attribute settings of an object are derived. The Visual Attribute Group property can be set to Default, NULL, or the name of a Visual Attribute object. Blocks have a Current Record Visual Attribute Group property that defines the Visual Attribute to be used for the current record in the block.

Partial Visual Attributes

You can define a Visual Attribute by setting only the properties that you want to be inherited by the objects that use them. This means that you can apply a Visual Attribute that changes the font color without having to set the font name.

How to Use Visual Attributes

1. **Create a Visual Attribute.**
2. **Set the Visual Attribute–related property of an object to the desired Visual Attribute.**
3. **Run the form to see the effect.**



Visual Attribute: SALARY_VA

Fill Pattern	<Unspecif
Font	
Font Name	<Unspecif
Font Size	<Unspecif
Font Weight	Bold
Font Style	<Unspecif

Weight of the font.

Item: SALARY

Height	14
Bevel	Lowered
Rendered	Yes
Show Vertical Scroll Bar	No
Visual Attributes	
Visual Attribute Group	DEFAUL
Prompt Visual Attribute Group	DEFAULT / SALARY_VA
Color	

How the object's individual attribute settings are

Hire Date	Salary
17-JUN-198	24000
21-SEP-198	17000
13-JAN-199	17000

ORACLE

Using Visual Attributes

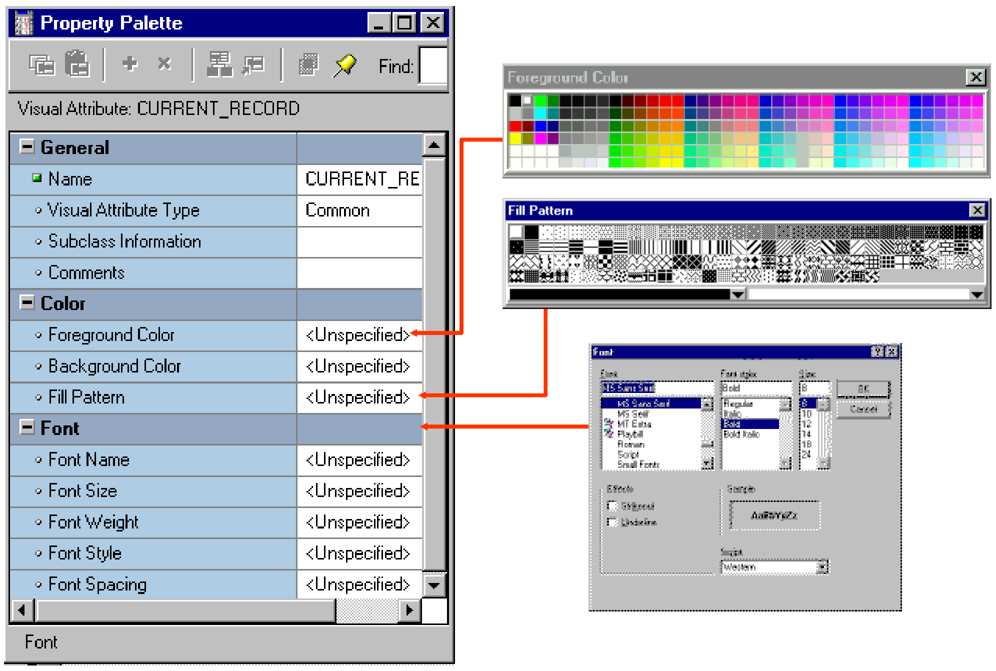
To use a Visual Attribute, perform the following steps:

1. Create the Visual Attribute:
   - Select the Visual Attributes node in the Object Navigator
   - Click Create
   - Invoke the Property Palette for the Visual Attribute and set the desired font, color, and pattern properties
2. Set the object properties to use the new Visual Attribute:
   - For items and canvases, set the Visual Attribute Group property
   - For blocks, set the Current Record Visual Attribute Group property
3. Run the form to see the changes.

Instructor Note

Demonstration: Create a Visual Attribute to be used later in this lesson. Use the Color, Pattern, and Font pickers. Use the `ordwk04.fmb` file.

Font, Pattern, and Color Pickers



Property Palette

Visual Attribute: CURRENT_RECORD

General	
Name	CURRENT_RE
Visual Attribute Type	Common
Subclass Information	
Comments	
Color	
Foreground Color	<Unspecified>
Background Color	<Unspecified>
Fill Pattern	<Unspecified>
Font	
Font Name	<Unspecified>
Font Size	<Unspecified>
Font Weight	<Unspecified>
Font Style	<Unspecified>
Font Spacing	<Unspecified>

Font

Font, Pattern, and Color Picker

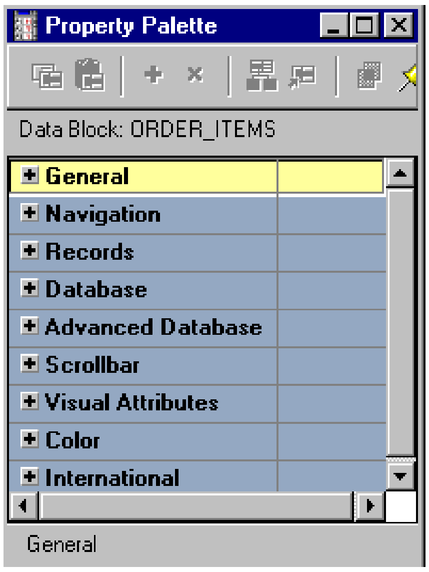
When you create Visual Attributes, you can use the Font, Pattern, and Color Pickers to select the font, pattern, and color.

When changing a font from the Property Palette, you can click the Font group itself to invoke the Font Picker. This enables you to select all font properties from one window; otherwise you can select each font property and invoke individual windows or pop-up lists that are specific to the property.

Controlling Data Block Behavior and Appearance

Data Block Property Groups:

- General
- Navigation
- Records
- Database
- Advanced Database
- Scrollbar
- Visual Attributes
- Color
- International



Data Block: ORDER_ITEMS

- General
- Navigation
- Records
- Database
- Advanced Database
- Scrollbar
- Visual Attributes
- Color
- International

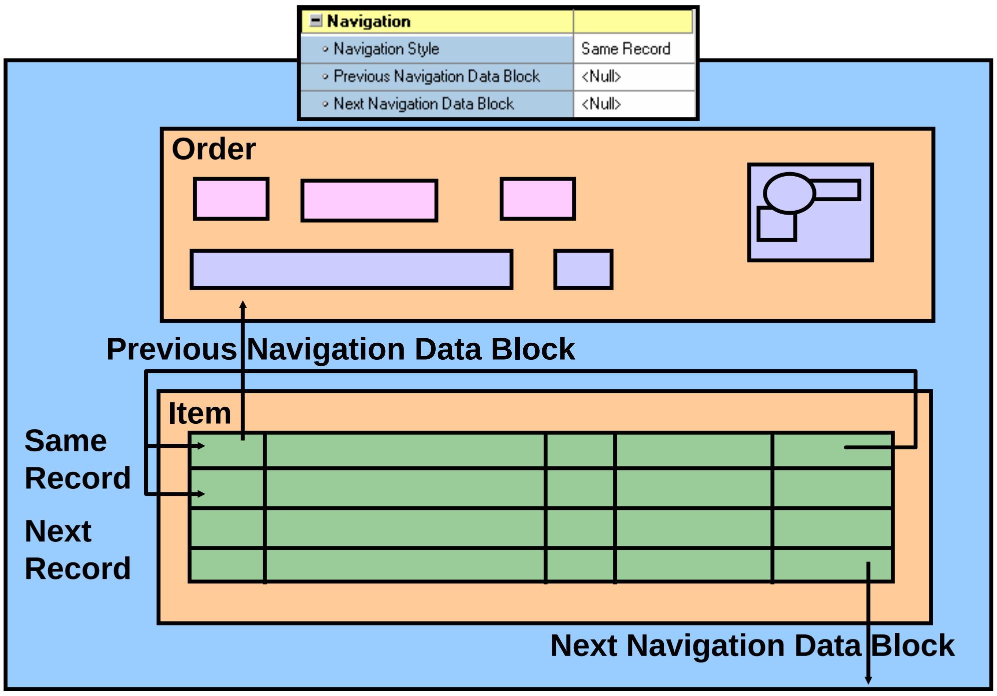
General

Data Block Properties

Each data block has several properties. These properties are divided into groups as shown in the slide.

You can modify the properties of the data block to control both the appearance and the behavior of the block.

Navigation Properties



Navigation	
Navigation Style	Same Record
Previous Navigation Data Block	<Null>
Next Navigation Data Block	<Null>

Controlling the Behavior of Data Blocks: Setting Navigation Properties

Navigation Style

Normally, when you navigate beyond the last item in a record, Forms returns you to the beginning of the same record. With this property you can change this behavior to navigate to the next record or data block instead. The valid settings are: Same Record (default), Change Record, or Change Data Block.

Note: If you want the cursor to move to the next record when you reach the end of the current record, set the Navigation Style property for the block to Change Record.

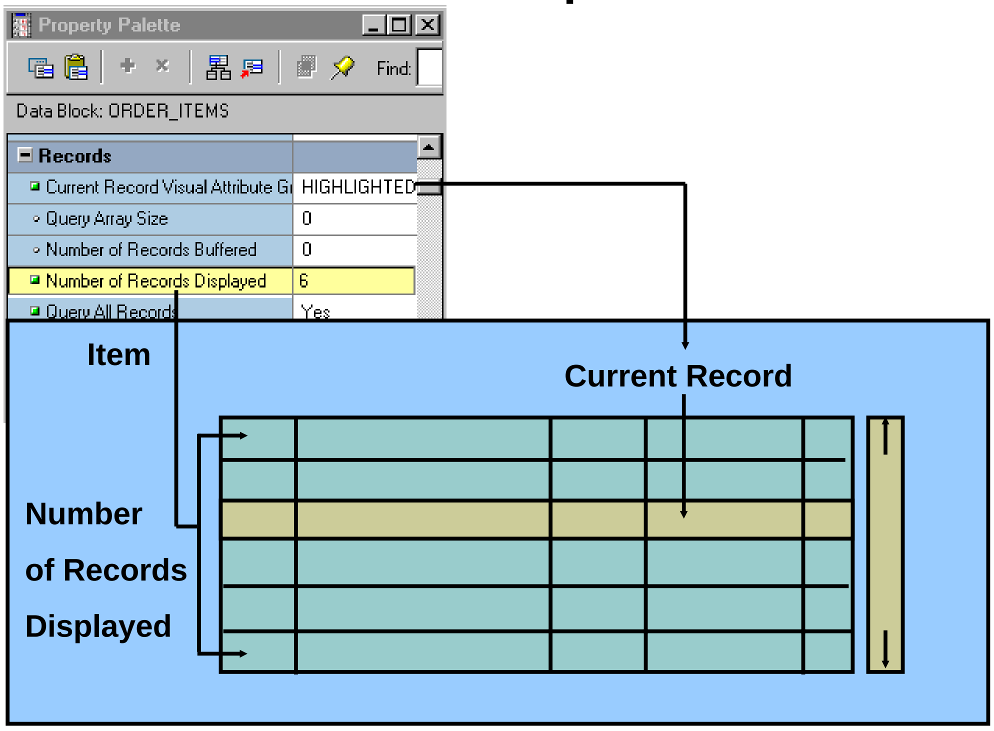
Previous/Next Navigation Data Block

Normally, when you perform an operation to move to the previous or next data block at run time, Forms moves control to the previous or next adjacent data block in sequence. These properties enable you to name the previous or next data block.

Instructor Note

Demonstration: Use the `ordwk04.fmb` file to show the run-time effect of changing the Navigation Style property on the ITEMS block.

Records Properties



Data Block: ORDER_ITEMS

Records	
Current Record Visual Attribute Gr	HIGHLIGHTED
Query Array Size	0
Number of Records Buffered	0
Number of Records Displayed	6
Query All Records	Yes

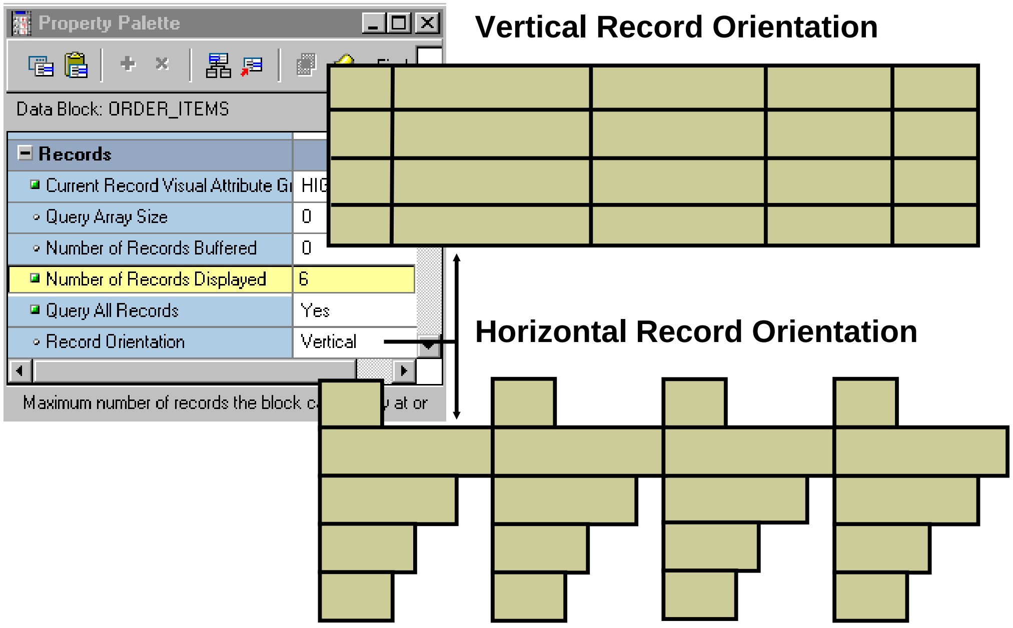
Controlling the Behavior of Data Blocks: Setting Record Properties

- **Current Record Visual Attribute Group:** The Visual Attribute that will be used to highlight the current record in the data block.
- **Query Array Size:** Specifies the maximum number of records that Forms should fetch from the database at one time. A lower value in this property value means faster response time; however, a larger value means fewer calls to the database for records, thereby resulting in reduced overall processing time. When set to 0 this property defaults to the Number of Records Displayed.
- **Number of Records Buffered:** Is the minimum amount of buffer space retained for holding queried records in the data block. The minimum setting allowed is the value of the Number of Records Displayed property plus 3. Forms buffers any additional records to a temporary disk file. A higher value improves processing speed, but uses more memory.

Instructor Note

Demonstration: Using the `ordwk04.fmb`, assign the Visual Attribute created earlier to the ITEMS block property Current Record Visual Attribute Group. Show the run time effect of doing this. Also show how to increase or decrease the number of records displayed in a data block.

Records Properties



Property Palette

Data Block: ORDER_ITEMS

Records	
Current Record Visual Attribute Gr	HIG
Query Array Size	0
Number of Records Buffered	0
Number of Records Displayed	6
Query All Records	Yes
Record Orientation	Vertical

Maximum number of records the block c... at or

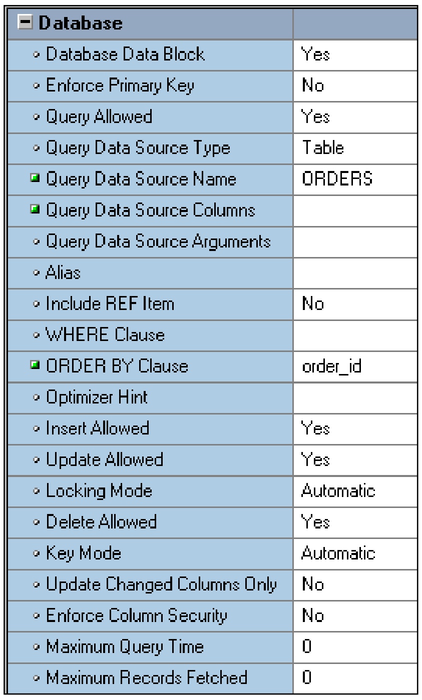
Controlling the Behavior of Data Blocks: Setting Record Properties (continued)

- **Number of Records Displayed:** This property specifies the maximum number of records the data block can display on the canvas at one time and how many records you can see at once. If you change this value, make sure there is enough room on the canvas layout for the number of records, or objects may overlap.
- **Query All Records:** This property specifies whether all the records matching the query criteria should be fetched when a query is executed. (This property is necessary to support the Calculated Field feature.)
- **Record Orientation:** This property determines the orientation of records in the data block—horizontal or vertical. When you set this property, Forms Builder adjusts the display position of items in the data block accordingly.
- **Single Record:** This property specifies that the control block should always contain one record. Set this property to Yes for a control block that contains a summary calculated item. You cannot set this property to Yes for a data block.

Database Properties

Use properties in the Database group to control:

- Type of block—data or control block
- Query, insert, update, and delete operations on the data block
- Data block's data source
- Query search criteria and default sort order
- Maximum query time
- Maximum number of records fetched



Database	
Database Data Block	Yes
Enforce Primary Key	No
Query Allowed	Yes
Query Data Source Type	Table
Query Data Source Name	ORDERS
Query Data Source Columns	
Query Data Source Arguments	
Alias	
Include REF Item	No
WHERE Clause	
ORDER BY Clause	order_id
Optimizer Hint	
Insert Allowed	Yes
Update Allowed	Yes
Locking Mode	Automatic
Delete Allowed	Yes
Key Mode	Automatic
Update Changed Columns Only	No
Enforce Column Security	No
Maximum Query Time	0
Maximum Records Fetched	0

ORACLE

6-15

Controlling the Behavior of Data Blocks: Setting Database Properties

In the Database group of the Property Palette, you can set numerous properties to control interaction with the database server. Some of these properties are:

- **Database Data Block:** Set to Yes if the data block is based on a database object and No if it is a control block.
- **Enforce Primary Key:** Controls whether Forms checks for uniqueness before inserting or updating records in the base table, in order to avoid committing duplicate rows in the database. A value of Yes means that the form checks that inserted or updated records are unique before an attempt is made to commit possible duplicate rows.
- **Query/Insert/Update/Delete Allowed:** Control whether the associated operations can be performed on the data block records.

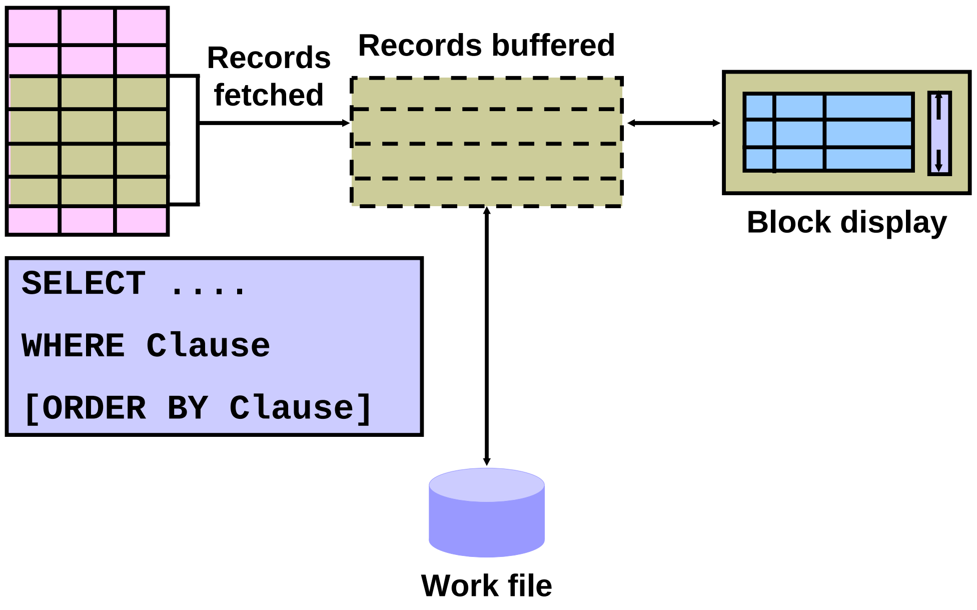
Instructor Note

Due to time constraints, discuss only the properties depicted on the slides. Remind students that they can select any property and press [F1] to get context-sensitive help about it.

Controlling the Behavior of Data Blocks: Setting Database Properties (continued)

- **Query Data Source Type:** Specifies the type of the query data source for the data block. Possible values for this property are None, Table, Procedure, Transactional Triggers, or `FROM` clause query.
- **Query Data Source Name:** Specifies the name of the query data source for the data block. This property is used only if the type of the query data source is Table, `FROM` clause query, or Procedure.
- **Query Data Source Columns:** Specifies, in a dialog box, the name and data type of the columns associated with the query data source. This property is used only if the type of the query data source is Table, `FROM` clause query, or Procedure.
- **Query Data Source Arguments:** Specifies, in a dialog box, the names, datatypes, and values of the arguments that are to be passed to the procedure for querying data. This property is valid only when the Query Data Source Type property is set to Procedure.
- **`WHERE` Clause:** Specifies a SQL condition that is attached to every default `SELECT` statement associated with the data block through implicit SQL; use to define general restrictions on the rows this data block may fetch. This clause is automatically appended (ANDed) with any conditions supplied by the operator in Enter Query mode.
- **`ORDER BY` Clause:** Defines a default order for records displayed from a query. The operator can alter this order by using the Query Where dialog box at run time.
- **Optimizer Hint:** Specifies a hint string that Forms passes to the Optimizer when constructing implicit SQL on the data block. The Optimizer can improve the performance of database transactions.
- **Locking Mode/Key Mode:** Controls how Forms handles records and transactions when the data block is primarily associated with non-Oracle data sources. The default settings are usually appropriate for data blocks connected with an Oracle database.
- **Update Changed Columns Only:** When this property is set to Yes, only those items updated by the operator are written to their corresponding database columns. If the operator commonly updates or inserts records with only one or two columns, this can save network traffic. By default, this property value is set to No, so that all columns are included in the default `UPDATE` statement.

Database Properties



```
SELECT ....
WHERE Clause
[ORDER BY Clause]
```

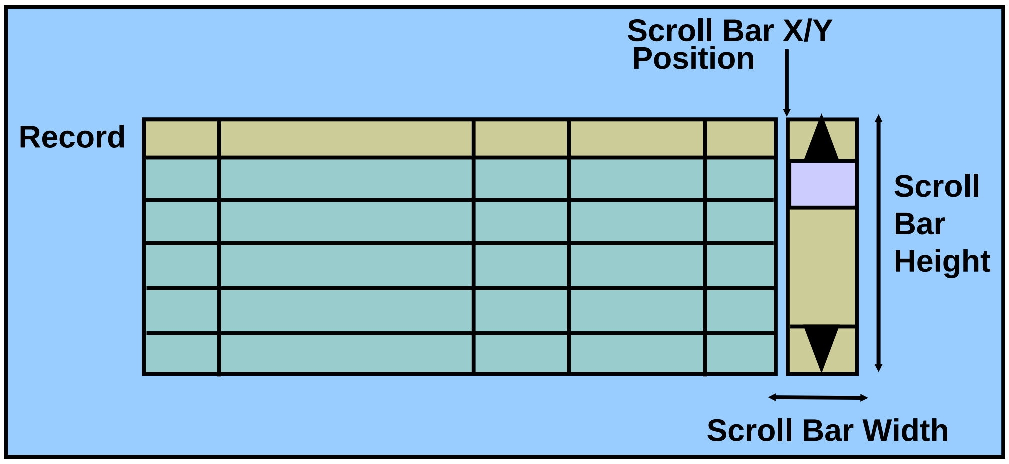
Controlling the Behavior of Data Blocks: Setting Database Properties (continued)

- **Enforce Column Security:** When this property is set to Yes, items in the data block can be updated only if the current user has privileges to update the corresponding database columns.
- **Maximum Query Time:** Provides the option to abort a query when the elapsed time of the query exceeds the value of this property; useful when the Query All Records property is set to Yes.
- **Maximum Records Fetched:** Provides the option to abort a query when the number of records fetched exceeds the value of this property; useful when the Query All Records property is set to Yes.

Instructor Note

When Update Changed Columns Only is set to No, Forms Builder can reuse the same SQL statement for multiple updates without the database having to reparse each time in its system global area (SGA). Changing this property value to Yes can degrade performance because the database must reparse the `UPDATE` statement each time.

Scroll Bar Properties

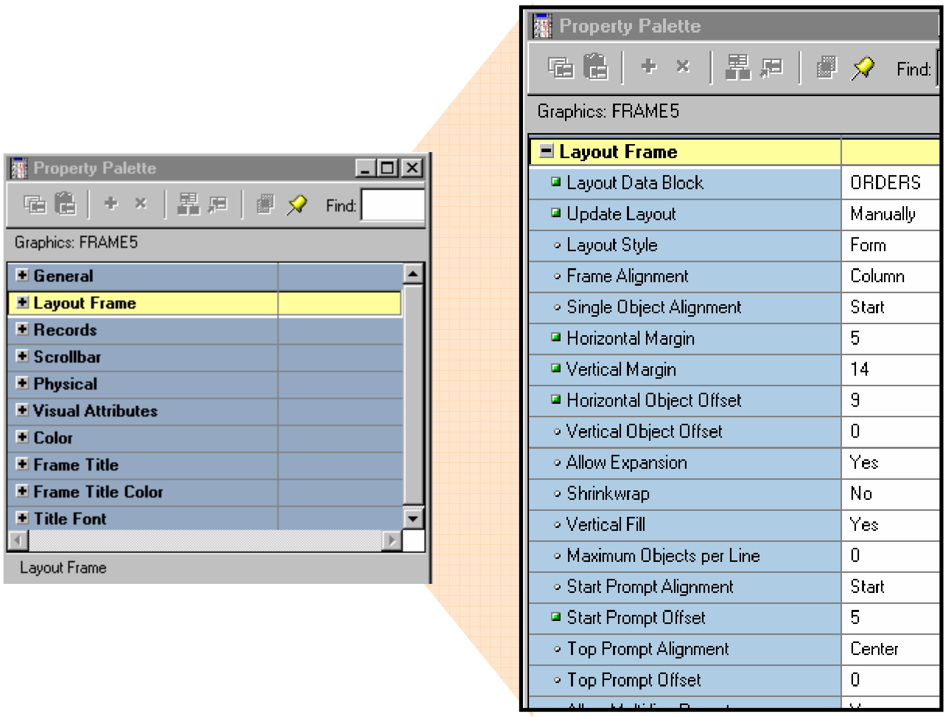


Controlling the Behavior of Data Blocks: Setting Scroll Bar Properties

In the Scrollbar group of the Property Palette, you can set numerous properties to the appearance and function of the data block's scroll bar. Some of these properties are:

- **Show Scroll Bar:** Specifies whether Forms Builder should create a scroll bar for the data block. To delete an existing scroll bar, set this property to No.
- **Scroll Bar Canvas:** Specifies the canvas on which the data block scroll bar will be displayed. The specified canvas must exist in the form.
- **Scroll Bar Orientation:** Specifies whether the scroll bar should be displayed horizontally or vertically.
- **Scroll Bar X/Y Position:** Specifies the x and y coordinates (measured in the coordination system units of the form) where the scroll bar will display on the canvas. The default value for both coordinates is 0.
- **Scroll Bar Width/Height:** Specifies the width and height of the scroll bar.

Controlling Frame Properties


Property Palette

Graphics: FRAME5

General	
Layout Frame	
Records	
Scrollbar	
Physical	
Visual Attributes	
Color	
Frame Title	
Frame Title Color	
Title Font	

Layout Frame

Property Palette

Graphics: FRAME5

Layout Frame	
Layout Data Block	ORDERS
Update Layout	Manually
Layout Style	Form
Frame Alignment	Column
Single Object Alignment	Start
Horizontal Margin	5
Vertical Margin	14
Horizontal Object Offset	9
Vertical Object Offset	0
Allow Expansion	Yes
Shrinkwrap	No
Vertical Fill	Yes
Maximum Objects per Line	0
Start Prompt Alignment	Start
Start Prompt Offset	5
Top Prompt Alignment	Center
Top Prompt Offset	0

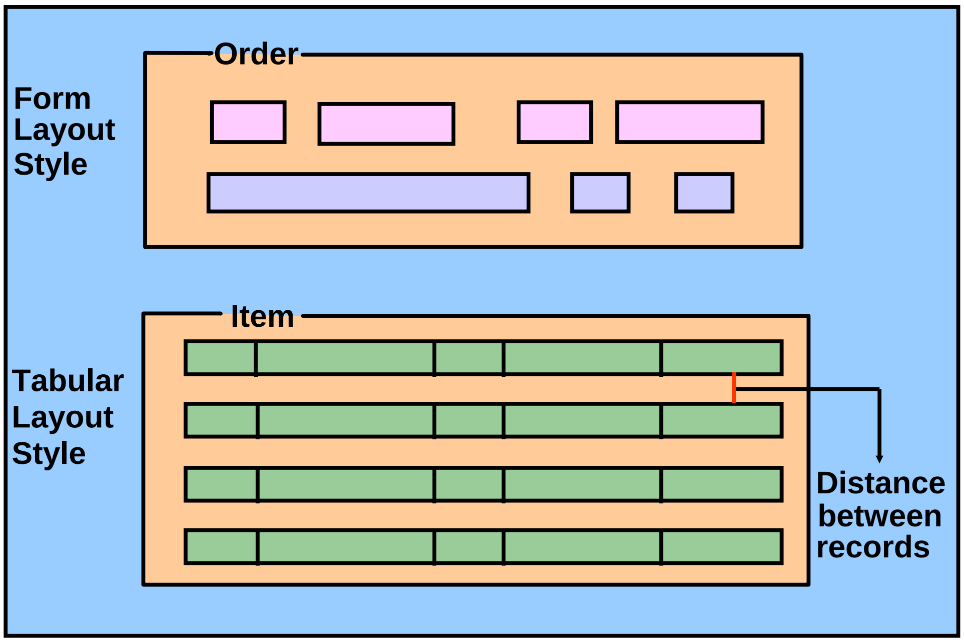
6-19

Controlling Frame Properties

The selections that you make in the Layout Wizard when creating a data block are recorded as properties of the resulting layout frame object. You can change frame properties to modify the arrangements of items within a data block. The main frame properties are as follows:

- **Layout Data Block:** Specifies the name of the data block with which the frame is associated. The items within this data block are arranged within the frame.
  Note: A data block can be associated with only one frame. You cannot arrange a block item within multiple frames.
- **Update Layout:** Specifies when the frame layout is updated. Valid settings are:
  - Automatically: The layout is updated whenever you move or resize the frame, or modify any frame layout property.
  - Manually: The layout is updated whenever you use the Layout Wizard to modify the frame, or in the Layout Editor, when you click Update Layout or select the Layout > Update Layout menu option.
  - Locked: The layout is locked and cannot be updated.

Controlling Frame Properties



Controlling Frame Properties (continued)

- **Layout Style:** Specifies the layout style for the items within the frame. Choose between Form and Tabular styles.
- **Distance Between Records:** Specifies the physical distance (measured in the form's coordination system units) with which to separate records displayed in the frame.
- **X/Y Position:** Specifies the x and y coordinates (measured in the form's coordination system units) of the frame's position on the canvas.
- **Width/Height:** Specifies the width and height of the frame (measured in the form's coordination system units).

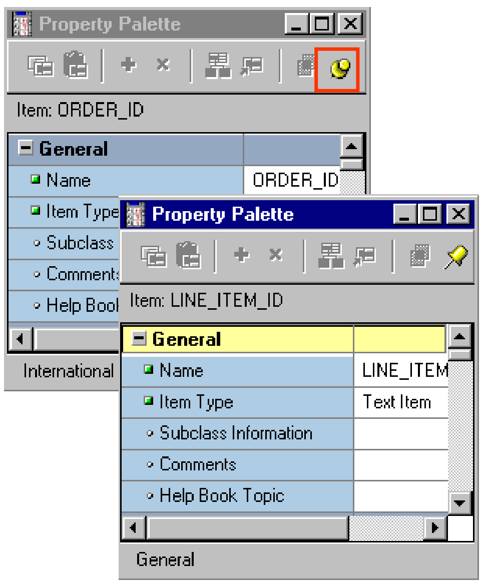
Note: You can arrange a frame as well as the objects within it manually in the Layout Editor.

Instructor Note

Stress the importance of the Update Layout property.

Displaying Multiple Property Palettes

Two Palettes for Two Items:



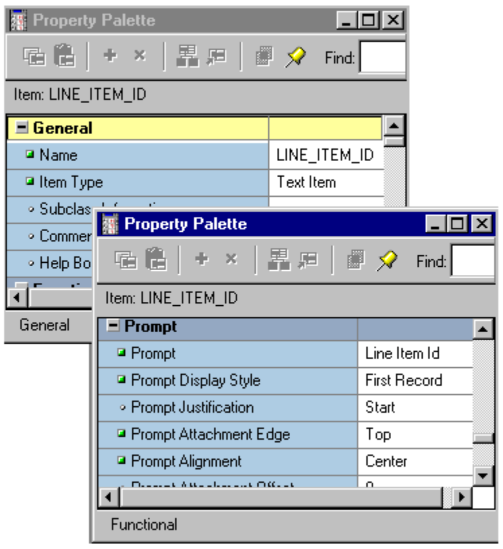
Item: ORDER_ID

General	
Name	ORDER_ID
Item Type	
Subclass	
Comments	
Help Bool	

Item: LINE_ITEM_ID

General	
Name	LINE_ITEM
Item Type	Text Item
Subclass Information	
Comments	
Help Book Topic	

Two Palettes for One Item:



Item: LINE_ITEM_ID

General	
Name	LINE_ITEM_ID
Item Type	Text Item
Subclas	
Commer	
Help Bo	

Item: LINE_ITEM_ID

Prompt	
Prompt	Line Item Id
Prompt Display Style	First Record
Prompt Justification	Start
Prompt Attachment Edge	Top
Prompt Alignment	Center

ORACLE

6-21

Displaying Multiple Property Palettes

More Than One Property Palette for One Object

You may want to see more properties for an object than there is room for in a single Property Palette. To display the properties of an object in multiple Property Palettes, perform the following steps:

1. Open a Property Palette for the object.
2. Hold down the [Shift] key and double-click the object icon for the object in the Object Navigator.

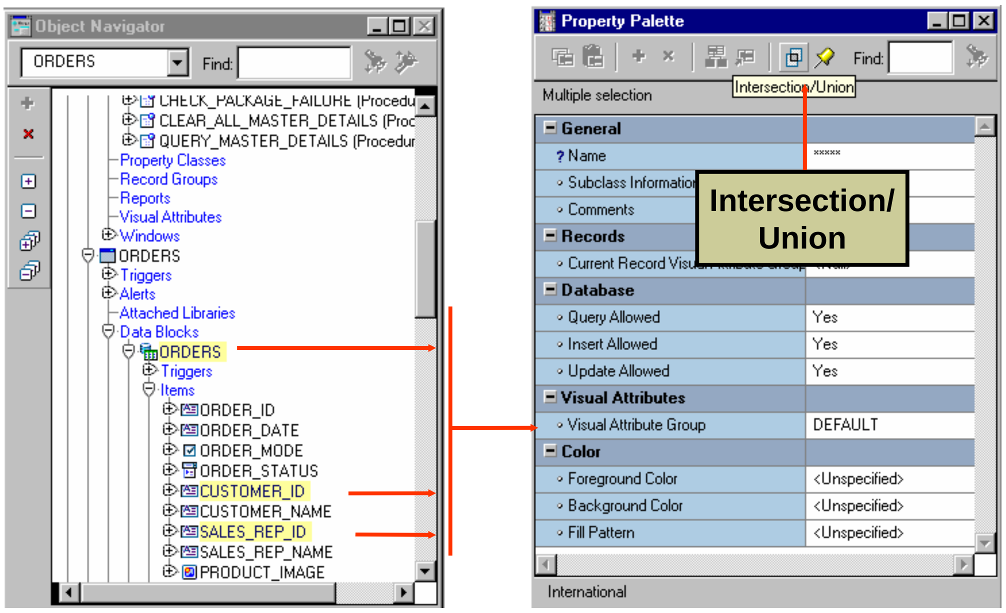
More Than One Property Palette for Multiple Objects

You may want to display properties for multiple objects simultaneously. To display the Property Palettes for multiple objects at the same time, perform the following steps:

1. Open the Property Palette of the first object.
2. Click Pin/Unpin on the toolbar to “freeze” this palette.
3. Invoke the Property Palette for another object. This Property Palette appears in a separate window.

If the second window is on top of the first one, drag it so that both windows are visible.

Setting Properties on Multiple Objects



General	
Name	*****
Subclass Information	
Comments	
Records	
Current Record Visu	[illegible]
Database	
Query Allowed	Yes
Insert Allowed	Yes
Update Allowed	Yes
Visual Attributes	
Visual Attribute Group	DEFAULT
Color	
Foreground Color	<Unspecified>
Background Color	<Unspecified>
Fill Pattern	<Unspecified>

Setting Properties on Multiple Objects

You can view and set the properties of several objects simultaneously, whether they are the same or different object types. You can select the objects in the Object Navigator and display a combination of the properties in the Property Palette. The combination may be:

- **Intersection:** A subset in which you display only the common properties of the selected objects (This is the default set operator.)
- **Union:** A superset in which you display both the common properties and the unique properties of the selected objects

Where there are differing values for a property across the selected objects, you will see ***** in the property value. This changes to a definitive value once you enter a new value in the Property Palette. This new value then applies to each of the selected objects to which the property is relevant.

Setting Properties on Multiple Objects (continued)

How to Set Properties on Multiple Objects

To set properties on multiple objects at one time, perform the following steps:

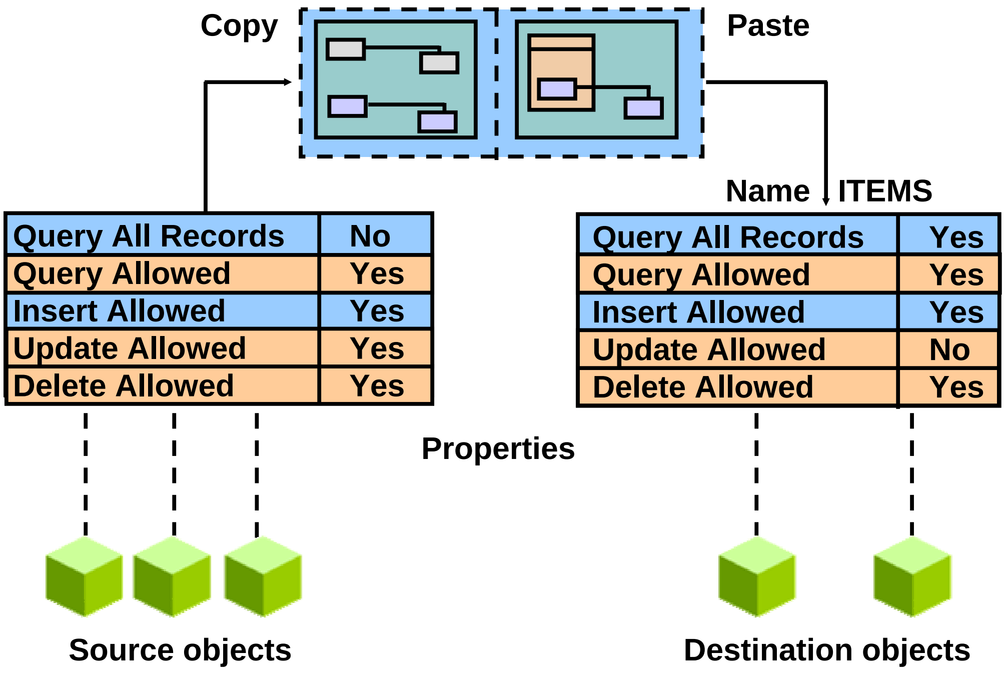
1. Open the Property Palette for one of the objects.
2. Hold down the [Ctrl] key and click each object in the Object Navigator or the editors whose properties are to be viewed or changed in combination. The selected objects are highlighted.
3. Set the Intersection/Union button from the toolbar in the Property Palette to the desired operation. This button toggles between the two options.
4. Change the displayed properties, as required. Your changes are applied to all selected objects with these properties.

Note: With a union, you may see some properties that are not relevant to all of the selected objects. Changes to a property are applied only to objects that have the property.

Instructor Note

Show how to select multiple objects in the Object Navigator, and how their combined properties can be represented in the Property Palette. Switch between Intersection and Union, pointing out the difference between the properties.

Copying Properties



Query All Records	No
Query Allowed	Yes
Insert Allowed	Yes
Update Allowed	Yes
Delete Allowed	Yes

Query All Records	Yes
Query Allowed	Yes
Insert Allowed	Yes
Update Allowed	No
Delete Allowed	Yes

Copying Properties

You can copy the properties and values from the Property Palette to a buffer, so that they can be applied (pasted) to other objects in your design session. To copy properties, perform the following steps:

1. In the Property Palette, display and set the properties that are to be copied. This may be from one object or a combination of objects.
   - To copy all property settings from the Property Palette, select Edit > Select All.
   - To copy the selected property settings only, press and hold [Ctrl] while you click each property individually.
2. Click Copy Properties on the toolbar of the Property Palette.
3. From the Object Navigator select the object into which the properties are to be copied.
4. In the Property Palette, click Paste Properties. The selected object receives values from all copied properties that are relevant to their object types.

Note: It is possible to copy the property settings of an object to objects of different types. In this case, properties that do not apply to the target object are ignored.

Copying Properties (continued)

Property Classes

When you display a list of properties (from either one object or a combination of objects) in the Property Palette, the list of property names and associated values can be saved for future application to other objects. This is known as a property class, which is a Forms Builder object in its own right.

Objects can inherit some of their properties from a linked property class, so their properties will automatically change if the associated properties are changed in the property class.

Property classes are discussed in more detail in a later lesson.

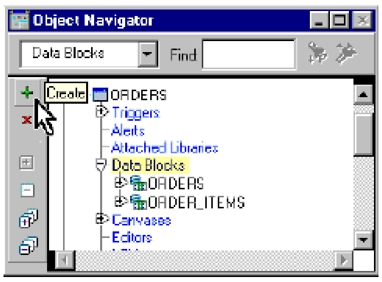
Instructor Note

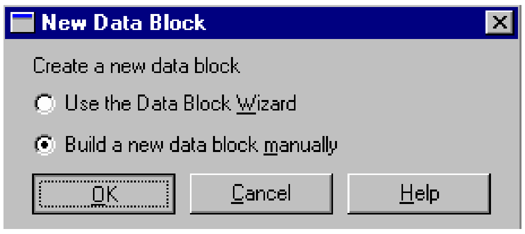
Show how to copy the properties of one object to another. Do not save the effects of this demonstration.

Note: Property inheritance and variance are covered in a later lesson.

Creating a Control Block

- Click the Data Blocks node
- Click the Create icon
  OR
  Select Edit > Create.
- Select the "Build a new data block manually" option in the New Data Block dialog box.




New Data Block

Create a new data block

( ) Use the Data Block Wizard

(●) Build a new data block manually

Creating Control Blocks

A control block is a block that is not associated with any database, and its items do not relate to any columns within any database table.

This means that Forms does not perform an automatic query when the operator issues an Enter Query or Execute Query command, nor does it issue an automatic Insert, Update, or Delete for the block when the operator saves changes to the database.

How to Create a Control Block

1. Click the Data Blocks node in the Object Navigator.
2. Click the Create icon on the toolbar, or Select Edit > Create from the menu.
3. In the New Data Block dialog box, select the "Build a new data block manually" option.
4. Open the Property Palette of the new data block and change its name.

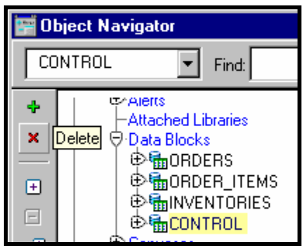
Note: Because there are no database columns on which to base control block items, a control block has no items until you manually add them later.

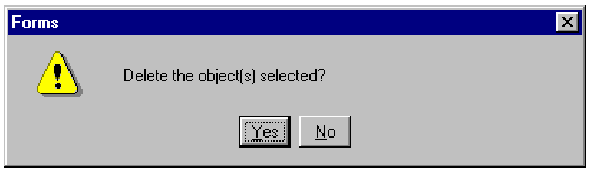
Instructor Note

Demonstration: Create a control block in the form module, using the `ordwk04.fmb` file.

Deleting a Data Block

- Select a data block for deletion
- Click the Delete icon
  OR
  Press [Delete]
- Click Yes in the alert box.





Delete the object(s) selected?

ORACLE

Deleting Data Blocks

To delete a data block:

1. Select the data block to be deleted in the Object Navigator.
2. Click the Delete icon on the toolbar.
   or
   Press [Delete].
3. An alert is displayed for delete confirmation. Click Yes to delete the data block.

Note: Deleting a data block also deletes its subordinate objects (items and triggers). If the data block was a master or detail block in a relation, the relation is also deleted. However, the frame border and its title will remain. Delete the frame manually in the Layout Editor.

Instructor Note

Demonstration: Delete the ITEMS data block in the `ordwk04.fmb` file. Show the effect of doing this in the Layout Editor. Do not save the effects of this demonstration.

Summary

In this lesson, you should have learned that:

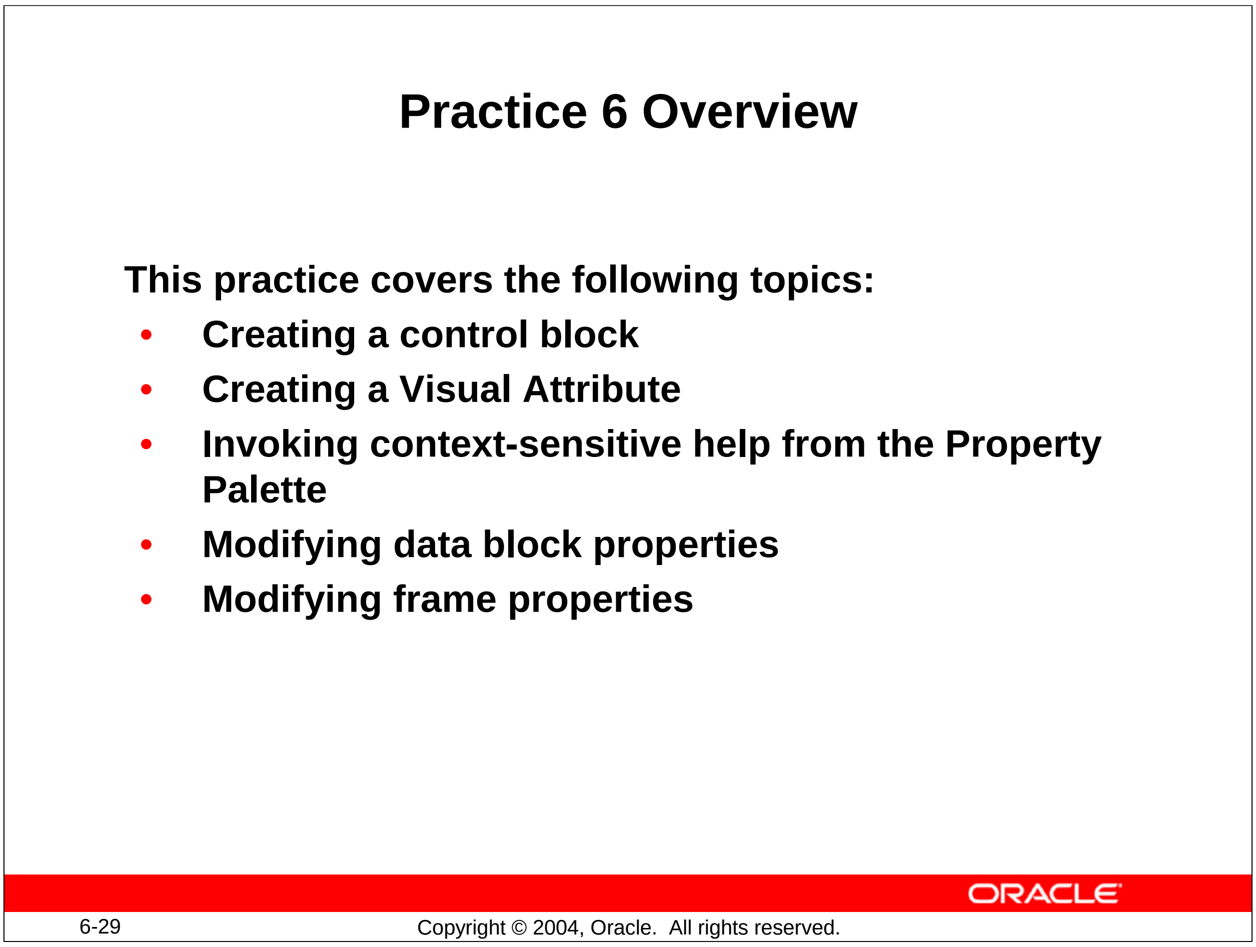
- The Property Palette:
  - Contains property names and values that enable you to modify Forms objects
  - Has tools to search for properties, inherit properties, expand or collapse property categories, and pop up lists and dialog boxes for various properties
  - Shows different icons for default, changed, inherited, and overridden properties
- Block properties control the behavior and appearance of data blocks
- Frame properties control how block items are arranged
- You can create blocks that do not directly correspond to database tables by choosing to create the block manually rather than using the Data Block Wizard
- Deleting a data block deletes all of its components

ORACLE

6-28

Summary

- Modify the data block properties in its Property Palette to change its behavior at run time.
- Data blocks have Navigation, Database, Records, Scrollbar, and other properties.
- Database properties include `WHERE` Clause, Query Data Source Type, and Maximum Records Fetched.
- You can change frame properties to modify the arrangements of items within a data block.
- You can copy properties between data blocks and other objects.
- You can view and change the properties of several objects together. You can use Intersection or Union settings to connect their properties in the Property Palette.



Practice 6 Overview

This practice covers the following topics:

- **Creating a control block**
- **Creating a Visual Attribute**
- **Invoking context-sensitive help from the Property Palette**
- **Modifying data block properties**
- **Modifying frame properties**

6-29

Practice 6 Overview

In this practice, you will perform the following tasks:

- Create a control block in the CUSTOMERS form.
- Explore property definitions using online Help from the Property Palette.
- Change properties in the CUSTOMERS data block and frame to change run-time appearance and behavior and design time behavior of the frame.
- Create a control block in the ORDERS form.
- Create a Visual Attribute in the ORDERS form and use it to highlight the current record in the ITEMS and INVENTORIES data blocks at run time. Use the multiple selection feature in the Property Palette.
- Change properties in the ITEMS and INVENTORIES data blocks to change their run-time appearance and behavior. Change the frame properties of all the data blocks in the ORDERS form to change their run-time appearance and to keep any layout changes you make manually in the Layout Editor.
- Save and run the forms after the changes are applied.

Note: For solutions to this practice, see Practice 6 in Appendix A, “Practice Solutions.”

Practice 6

CUSTG*XX* Form

1. Create a control block in the CUSTG*XX* form.
   Create a new block manually, and rename this block `CONTROL`.
   Set the Database Data Block, Query Allowed, Insert Allowed, Update Allowed, and Delete Allowed Database properties to No. Set the Query Data Source Type property to None. Set the Single Record property to Yes. Leave other properties as default.
   Move the `CONTROL` block after the `CUSTOMERS` block.
2. Ensure that the records retrieved in the `CUSTOMERS` block are sorted by the customer's ID.
3. Set the frame properties for the `CUSTOMERS` block as follows:
   Remove the frame title, and set the Update Layout property to Manually. Once you have done this, you may resize the frame if desired without having the items revert to their original positions.
4. Save and run the CUSTG*XX* form.
   Test the effects of the properties that you have set.
   Note: The Compilation Errors window displays a warning that advises you that the CONTROL block has no items. This is expected (until you add some items to the CONTROL block in a later lesson).

ORDG*XX* Form

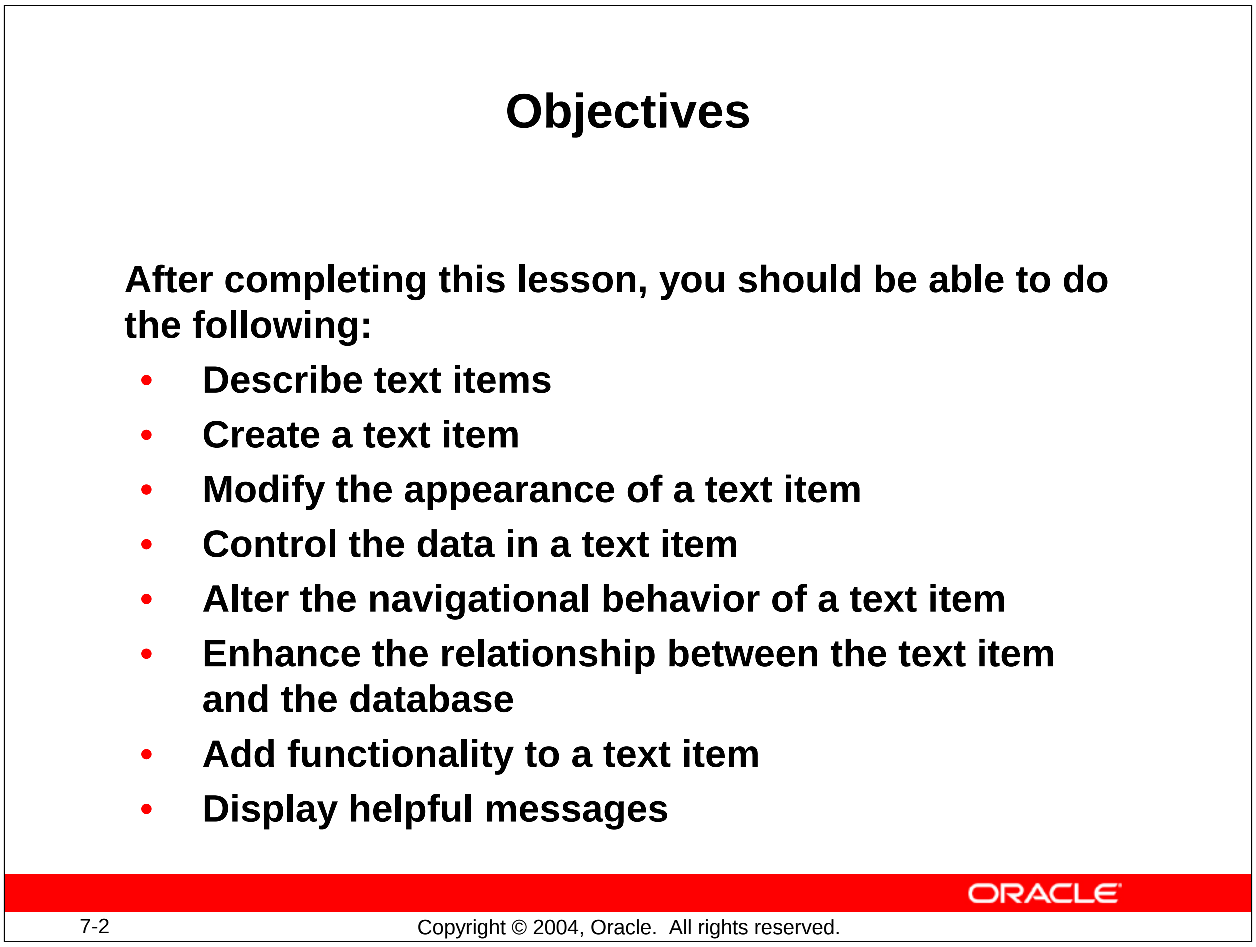
5. Create a `CONTROL` block in the ORDG*XX* form.
   Create a new block manually, and rename this block `CONTROL`.
   Set the Database Data Block, Query Allowed, Insert Allowed, Update Allowed, and Delete Allowed database properties to No. Set the Query Data Source Type property to None. Set the Single Record property to Yes.Leave other properties as default.
   Position the `CONTROL` block after the `INVENTORIES` block in the Object Navigator.
6. Ensure that the records retrieved in the `ORDERS` block are sorted by the `ORDER_ID`.
7. Ensure that the current record is displayed differently from the others in both the `ORDER_ITEMS` and `INVENTORIES` blocks.
   Create a Visual Attribute called Current_Record.
   Using the Color Picker, set the foreground color to white and the background color to gray. Using the Pattern Picker, choose any fill pattern. Using the Font Picker, set the font to MS Serif italic 10 point. (If that font is not available on your window manager, use any available font.)
   Use the multiple selection feature on both data blocks to set the relevant block property to use this Visual Attribute.

Practice 6 (continued)

8. For the `ORDER_ITEMS` block, change the number of records displayed to 4 and resize the scroll bar accordingly.
9. Ensure that the records retrieved in the `ORDER_ITEMS` block are sorted by the `LINE_ITEM_ID`.
10. Set the `ORDER_ITEMS` block to automatically navigate to the next record when the user presses [Next Item] while the cursor is in the last item of a record.
11. Set the frame properties for all blocks as follows:
    Remove the frame title and set the Update Layout property to Manually.
12. Save and compile the ORDG*XX* form.
    Click Run Form to run your form.
    Test the effects of the properties that you have set.
    Note: The Compilation Errors window displays a warning that advises you that the `CONTROL` block has no items. This is expected (until you add some items to the `CONTROL` block in a later lesson).

7 Working with Text Items

Schedule:	Timing	Topic
	45 minutes	Lecture
	40 minutes	Practice
	85 minutes	Total



Objectives

After completing this lesson, you should be able to do the following:

- **Describe text items**
- **Create a text item**
- **Modify the appearance of a text item**
- **Control the data in a text item**
- **Alter the navigational behavior of a text item**
- **Enhance the relationship between the text item and the database**
- **Add functionality to a text item**
- **Display helpful messages**

7-2

Introduction

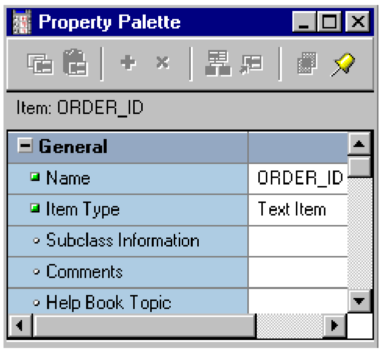
Overview

The default item type in an Oracle Forms Developer application is the text item or field. You have seen how creating a new data block based on a table creates text items for each selected column from that table. This lesson shows you how to customize text items to change their appearance and behavior.

Text Item Overview

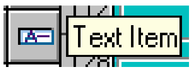
What is a text item?

- Default item type
- Interface object for:
  – Querying
  – Inserting
  – Updating
  – Deleting
- Behavior defined in the Property Palette



Item: ORDER_ID

General	
Name	ORDER_ID
Item Type	Text Item
Subclass Information	
Comments	
Help Book Topic	



Text Items

A text item is an interface object through which you can query, insert, update, and delete data. A text item usually corresponds to a column in the database table. When an item is first created, its default type is text.

The item type determines the properties available in the Property Palette. In this lesson you look at the properties of a text item. Remaining item types are covered in subsequent lessons.

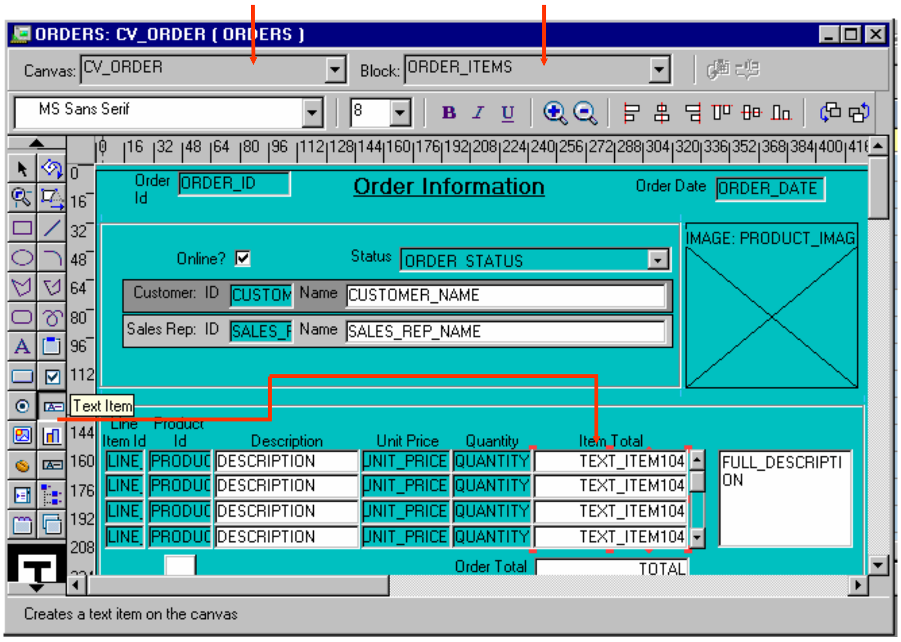
Use the Property Palette to define, alter, or examine the characteristics of items.

Instructor Note

- Although the technical term *item* is used in design terminology, end users may think of these objects as *fields*.
- Ensure that the class does not get confused between *item* as a Forms Builder term for the objects in a block, and *item* as the application term for line item records in the Order Entry system.

Creating a Text Item

Canvas selection   Block selection



Creating a Text Item

You can create a text item by doing one of the following:

- Converting an existing item into a text item
- Using the Text Item tool in the Layout Editor
- Using the Create icon in the Object Navigator
- Using the wizards

How to Create a Text Item in the Layout Editor

1. Invoke the Layout Editor.
   It is important to point to the correct data block where you want to create the text item. In the Layout Editor, select the data block from the Block pop-up list.
2. Click the Text Item tool.
3. Click the canvas.
   The text item appears.
4. Double-click the text item.
   The text item Property Palette appears.
5. Set the item properties as required.

Creating a Text Item (continued)

How to Create a Text Item in the Object Navigator

1. Locate the block in which you want to create the item.
2. Select the Items node.
3. Click the Create icon.
   A new item entry is displayed in the Object Navigator.
4. Double-click the icon to the left of the new item entry.
   The Property Palette appears.
5. Set all item properties as required, keeping the Type property set to Text Item.

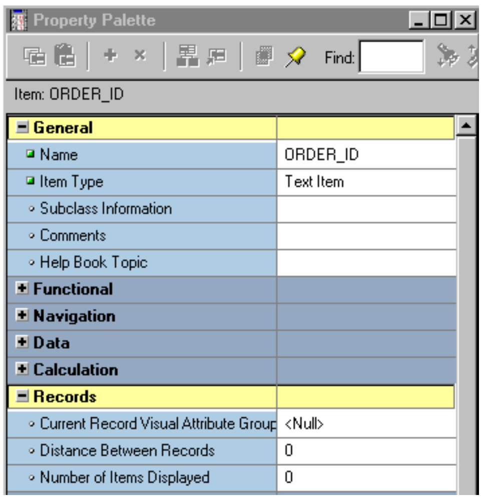
Note: To display an item at run time, you must assign the item to a canvas. Do this in the Property Palette of the text item by setting the Canvas property to the desired canvas.

Instructor Note

Demonstration

- In the first demonstration, show how to create a text item in the Layout Editor.
- In the second demonstration, create a text item in the Object Navigator and do the following:
  - Go to the Layout Editor. The new text item is not seen. Point out to the students that this is a Null canvas item, because it is not assigned to any canvas yet.
  - Open the text item property sheet, and assign the Canvas property to the desired canvas.
  - Show the text item in the Layout Editor.
- Point out the importance of creating a new text item in the correct data block and canvas. Also point out the Canvas and Block pop-up lists in the Layout Editor.

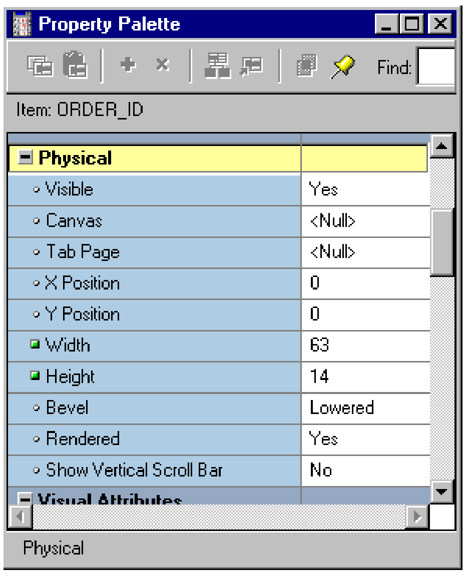
Modifying the Appearance of a Text Item: General and Physical Properties


Property Palette

Find:

Item: ORDER_ID

General	
Name	ORDER_ID
Item Type	Text Item
Subclass Information	
Comments	
Help Book Topic	
Functional	
Navigation	
Data	
Calculation	
Records	
Current Record Visual Attribute Group	<Null>
Distance Between Records	0
Number of Items Displayed	0


Property Palette

Find:

Item: ORDER_ID

Physical	
Visible	Yes
Canvas	<Null>
Tab Page	<Null>
X Position	0
Y Position	0
Width	63
Height	14
Bevel	Lowered
Rendered	Yes
Show Vertical Scroll Bar	No
Visual Attributes	

Physical

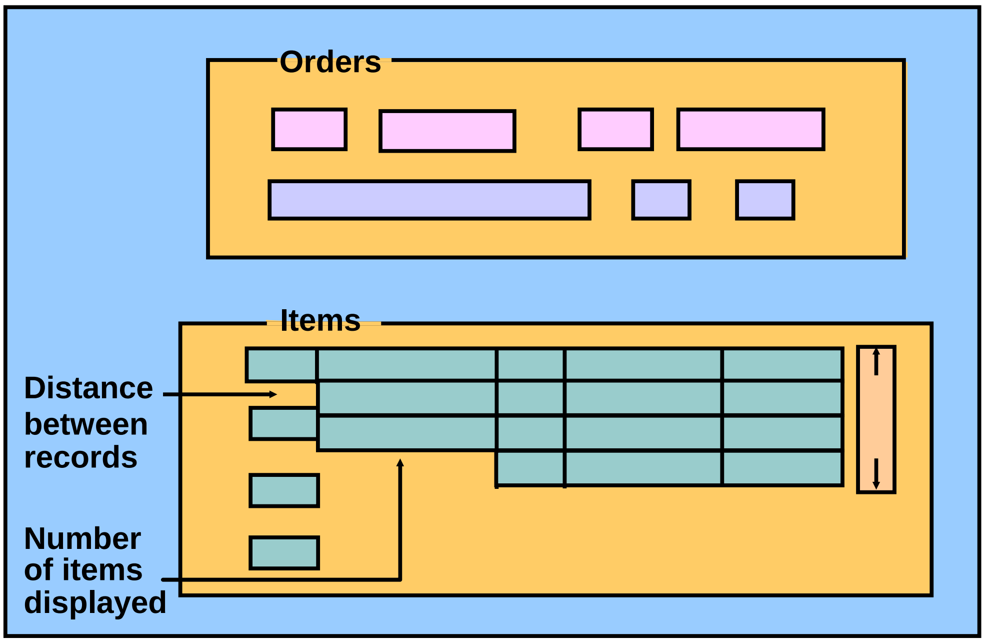
Modifying the Appearance of a Text Item

The properties of an item are divided into several groups.

You can affect the way the text item is displayed by altering its General, Physical, Records, Font, and Color group properties. To view descriptions of any of these properties, click the property in the Property Palette and press `[F1]`. Some of these properties are:

- **General:**
  - Item type: Selects the type of item you want to create (pop-up list)
- **Physical:**
  - Visible: Determines whether the item is displayed
  - Canvas: Determines the canvas on which the item is displayed. If left unspecified, the item is said to be a *Null canvas* item, and will not display at run time or in the Layout Editor.
  - X and Y Position: Sets the X and Y coordinates of the item relative to the canvas
  - Width and Height: Sets the width and height of the item in the current form coordinate units
  - Bevel: Controls appearance of bevel around the item; can also be set to Plain (flat) or None

Modifying the Appearance of a Text Item: Records Properties



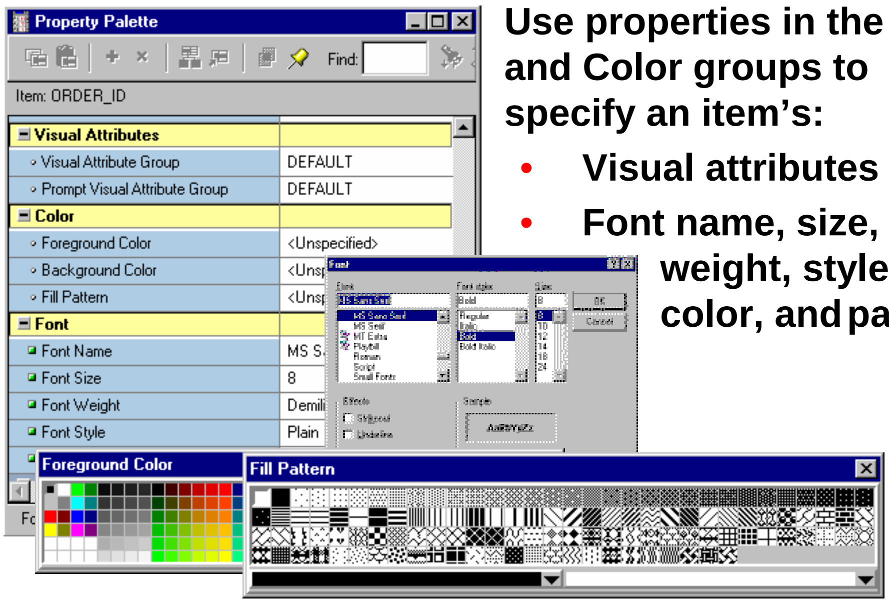
Modifying the Appearance of a Text Item (continued)

- **Records:**
  - Current Record Visual Attribute Group: Specifies the name of the visual attribute to use when the item is part of the current record
  - Distance between records: Specifies the amount of space between instances of the item in a multirecord block
  - Number of Items Displayed: Specifies the number of item instances that are displayed for the item when the item is in a multirecord block

Instructor Note

Demonstration: Using the `ordwk05.fmb` file, show how to alter the width and height of a text item. Show how to change the number of items displayed at run time.

Modifying the Appearance of a Text Item: Font and Color Properties



Item: ORDER_ID

Visual Attributes	
Visual Attribute Group	DEFAULT
Prompt Visual Attribute Group	DEFAULT
Color	
Foreground Color	<Unspecified>
Background Color	<Unsp
Fill Pattern	<Unsp
Font	
Font Name	MS S
Font Size	8
Font Weight	Demili
Font Style	Plain

Use properties in the Font and Color groups to specify an item's:

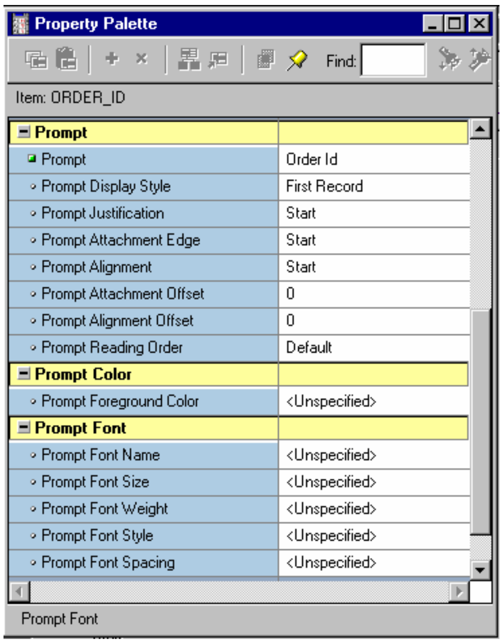
- Visual attributes
- Font name, size, weight, style, color, and pattern

Modifying the Appearance of a Text Item (continued)

- **Visual Attributes:** You set the Visual Attribute Group for an item or its prompt to specify how the visual attributes are derived (select DEFAULT or a named Visual Attribute).
- **Color:** The properties in this section specify the foreground color, background color, and fill pattern for the item. You select these from color and pattern pickers.
- **Font:** The properties in this section determine the font used for the item, along with its size, weight, style, and spacing. You can double-click the Font group to display a Font dialog enabling you to set all these properties at once, or you can click the individual properties to select each from an appropriate control, such as a pop-up list or LOV.

Note: When the form module does not contain any named visual attribute objects, the pop-up list for the Visual Attribute Group property shows only Default or Unspecified. An item that has the Visual Attribute Group property set to default, or that has individual attribute settings left unspecified, inherits those settings from the canvas to which it is assigned.

Modifying the Appearance of a Text Item: Prompts


Property Palette

Item: ORDER_ID

Prompt	
Prompt	Order Id
Prompt Display Style	First Record
Prompt Justification	Start
Prompt Attachment Edge	Start
Prompt Alignment	Start
Prompt Attachment Offset	0
Prompt Alignment Offset	0
Prompt Reading Order	Default
Prompt Color	
Prompt Foreground Color	<Unspecified>
Prompt Font	
Prompt Font Name	<Unspecified>
Prompt Font Size	<Unspecified>
Prompt Font Weight	<Unspecified>
Prompt Font Style	<Unspecified>
Prompt Font Spacing	<Unspecified>

Prompt Font

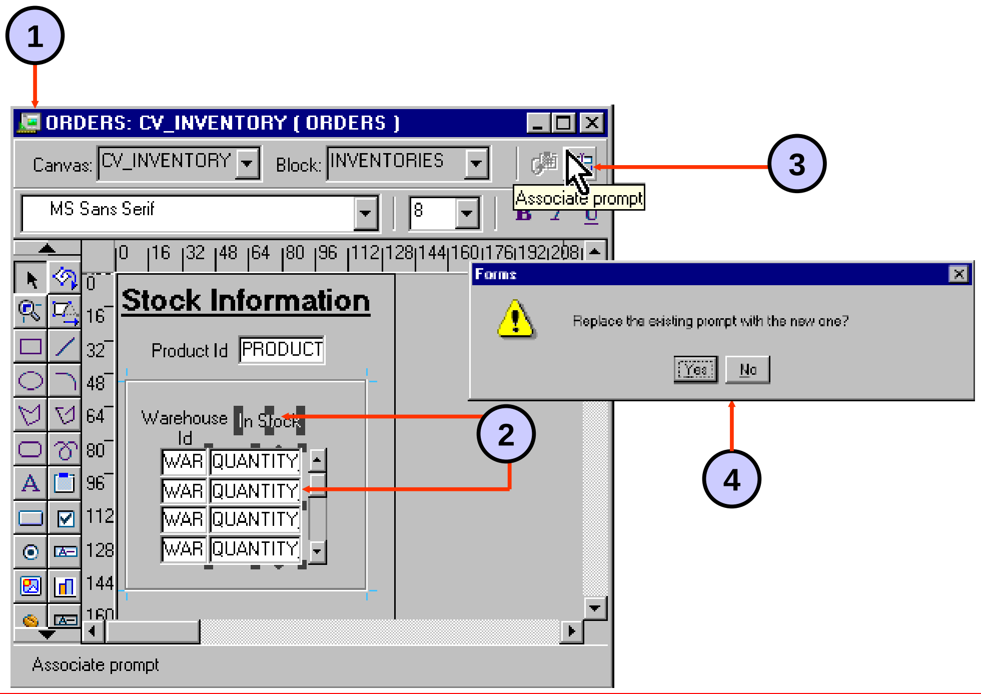
- **A prompt specifies the text label that is associated with an item.**
- **Several properties are available to arrange and manage prompts.**
- **Use prompt properties to change the appearance of an item prompt.**

Modifying the Appearance of a Text Item (continued)

You can control the appearance of the prompt, or label, of a text item using properties in the following groups:

- **Prompt:** You can set the prompt text and other properties, such as:
  - Display Style: Pop-up list with choices of First Record, Hidden, and All Records
  - Attachment Edge: Specifies the item edge to which the prompt is attached
  - Attachment Offset: Specifies the distance between the item and its prompt
- **Prompt Color:** The property in this section specifies the foreground color for the item prompt. You can select this from a color picker.
- **Prompt Font:** The properties in this section determine the font that is used for the item prompt, along with its size, weight, style, and spacing. You can double-click the Prompt Font group to display a Font dialog enabling you to set all these properties at once, or you can click the individual properties to select each from an appropriate control, such as a pop-up list or LOV.

Associating Text with an Item Prompt



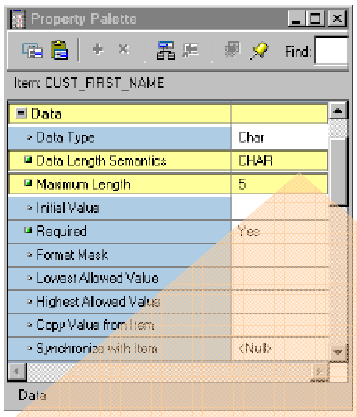
Replace the existing prompt with the new one?

Associating Text with an Item Prompt

The Forms Builder Layout Editor has a tool called Associate Prompt which enables you to create a prompt for an item using any boilerplate text displayed in the editor. To create a prompt-item association using the Associate Prompt tool, perform the following steps:

1. Open the Layout Editor window.
2. Select the item and boilerplate text you want as the item's prompt in the editor.
3. Click the Associate Prompt tool.
4. If you are replacing an existing prompt, answer Yes to the dialog box.

Controlling the Data of a Text Item

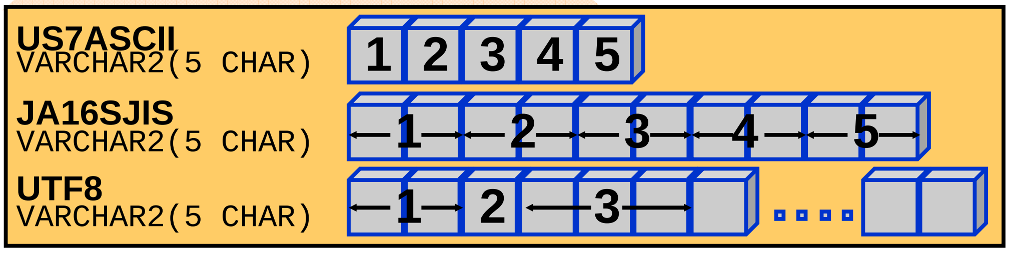

Property Palette

Item: CUST_FIRST_NAME

Data	
Data Type	Char
Data Length Semantics	CHAR
Maximum Length	5
Initial Value	
Required	Yes
Format Mask	
Lowest Allowed Value	
Highest Allowed Value	
Copy Value from Item	
Synchronize with Item	<Null>

Use properties in Data group to control the data:

- Type
- Length
- Format
- Value



Controlling the Data of a Text Item

The properties in the Data group of the text item are used to control the way data is displayed and entered. You can see descriptions of any of these properties by clicking the property in the Property Palette, then pressing `[F1]`. Some of these properties are:

- **Data Type:** Enables you to choose CHAR, DATE, DATETIME, and NUMBER; the others listed are for backward compatibility.
- **Data Length Semantics:** You can set to Null, BYTE, or CHAR to be compatible with multiple character sets. If Data Length Semantics is CHAR, the correct amount of storage will be automatically allocated as required for the Maximum Length with either a single-byte or multi-byte character set.
- **Maximum Length:** Specifies the maximum length of the data value that can be stored in an item. If the Maximum Length exceeds the display width of the item, Forms automatically enables the end user to scroll horizontally.
  Note: In the example on the slide, whether the form operator is using a single-, double-, or variable-byte character set, the right amount of storage is allocated. To hold the same value if the Data Length Semantics had been set to BYTE, the Maximum Length would have needed to be 5 for single-byte, 10 for double-byte, and an unknown value for a variable-byte character set.

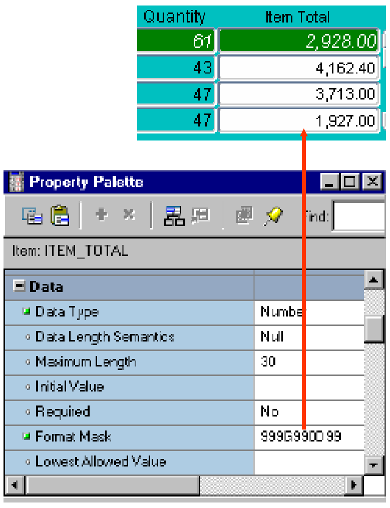
Controlling the Data of a Text Item: Format

Format masks:

- **Standard SQL formats**
  - **Dates** **FXDD-MON-YY**
  - **Numbers** **L099G990D99**
- **Nonstandard formats**

  Use double quotes for embedded characters “(“099”)”099“-”0999

Note: Allow for format mask’s embedded characters when defining Width property.



Quantity	Item Total
61	2,928.00
43	4,162.40
47	3,713.00
47	1,927.00

Property Palette

Item: ITEM_TOTAL

Data	
Data Type	Numbe
Data Length Semantics	Null
Maximum Length	30
Initial Value	
Required	No
Format Mask	999G990D99
Lowest Allowed Value	

Controlling the Data of a Text Item (continued)

- **Format Mask:** You can specify any format mask that is valid for the data type. Use the Format Mask property to specify the format in which the user sees the item value.
  - Use standard SQL formatting syntax for dates and numbers; for example, DD/MM/YY and $99,999.99.
  - Enclose non-SQL standard embedded characters in double quotes; for example, hyphen (-) and comma (,).

  Note: It is recommended that you avoid creating individual masks if the general purpose masks (see Lesson 1) will suffice.

  FX Format Mask: The FX format mask in a date value ensures that the date is entered exactly as defined in the mask. Element D is for decimal and G is a group (thousands) separator.

  Example: With a date format of DD/MM/YY, valid entries are: 10/12/00, 10 12 00, 10-DEC-00, or 101200. You can enter any character to represent the (/) in the value. Allow for the embedded characters of the format mask when defining the Width property. The embedded characters are used only for display purposes and are not stored in the database.

Controlling the Data of a Text Item: Values

Initial Values:

- **Are used for every new record**
- **Can be overwritten**
- **Must be compatible with item’s data type**
- **Use:**
  - **Raw value**
  - **System variable**
  - **Global variable**
  - **Form parameter**
  - **Form item**
  - **Sequence**

Controlling the Data of a Text Item (continued)

- **Required:** Specifies whether Forms will allow the item to have a null value. When you create a data block, Forms derives this value from the existence of a NOT NULL constraint on the database column, but you can change the value.
- **Lowest/Highest Allowed Value:** Specifies range of accepted values
- **Initial Value:** Specifies default value assigned to an item whenever a record is created; can be set to select from a sequence. Must be compatible with the item data type. If the Lowest/Highest Allowed values are specified, the initial value cannot be outside the range.

Controlling the Data of a Text Item (continued)

Creating an Initial Value

You can use any one of the following values to issue an initial item value whenever a new record is created:

- Raw value
  Example: 340, RICHMOND
- System variable
  - Variables giving current application server *operating system* date/time:

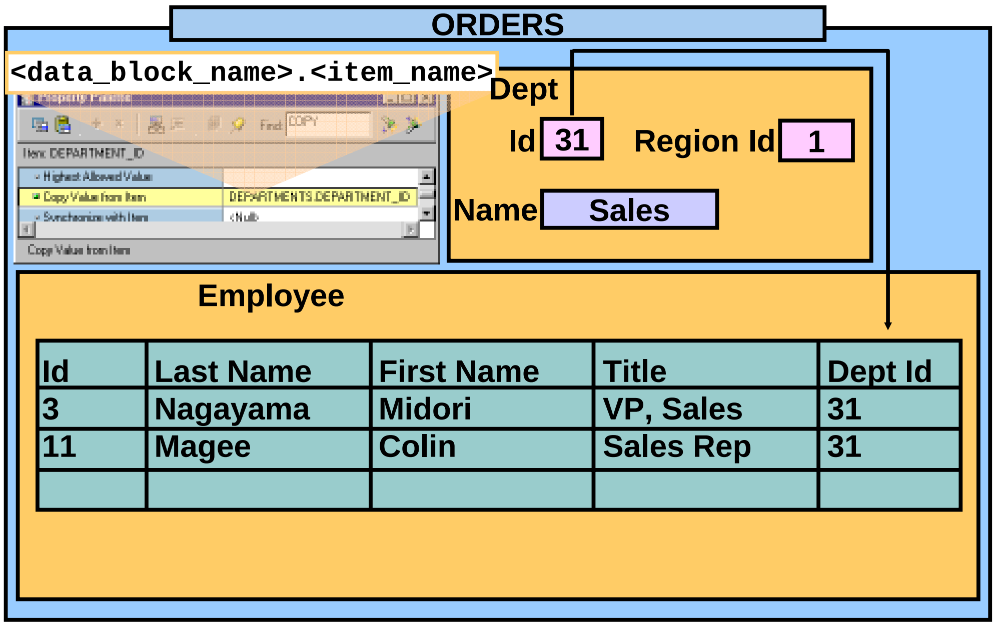
Variable	**Format**
`$$DATE$$`	`DD-MON-YY`
`$$DATETIME$$`	`DD-MON-YYYY hh:mi[:ss]`
`$$TIME$$`	`hh:mi[:ss]`

  - Variables giving current *database* date/time:

Variable	**Format**
`$$DBDATE$$`	`DD-MON-YY`
`$$DBDATETIME$$`	`DD-MON-YYYY hh:mi[:ss]`
`$$DBTIME$$`	`hh:mi[:ss]`

- Global variable
  Example: `:GLOBAL.CUSTOMER_ID`
- Form parameter
  Example: `:PARAMETER.SALES_REP_ID`
- Form item
  Example: `:ORDERS.ORDER_ID`
- Sequence
  The initial value can reference a sequence in the database. Forms automatically writes generated sequence numbers into the text item. To use a sequence, enter: `:sequence.<sequence name>.nextval`.
  Example: `:SEQUENCE.ORDERS_SEQ.NEXTVAL`

Controlling the Data of a Text Item: Copy Value from Item



Item: DEPARTMENT_ID

Highest Allowed Value	
Copy Value from Item	DEPARTMENTS.DEPARTMENT_ID
Synchronize with Item	<Null>

Copy Value from Item

Id	Last Name	First Name	Title	Dept Id
3	Nagayama	Midori	VP, Sales	31
11	Magee	Colin	Sales Rep	31

7-15

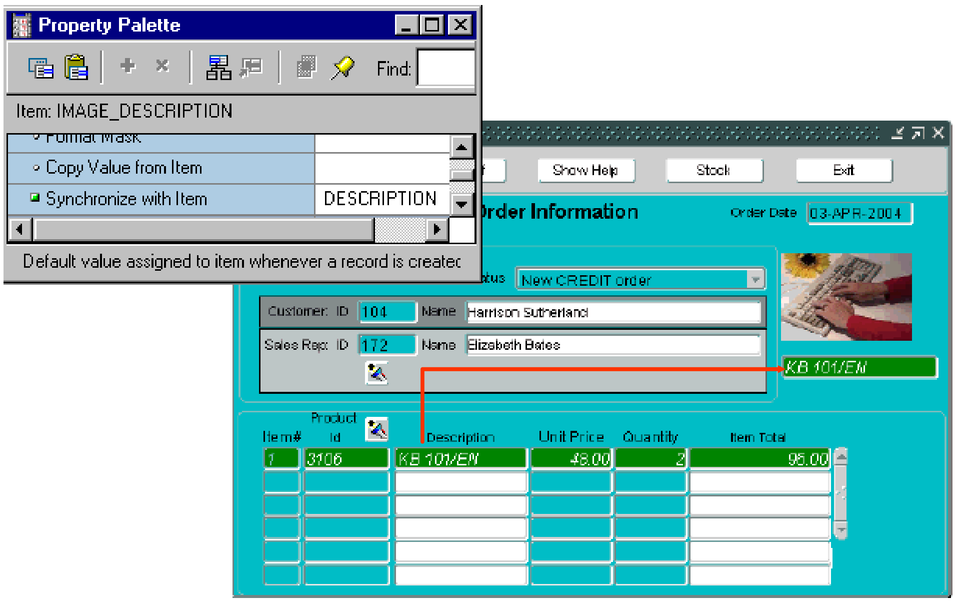
Controlling the Data of a Text Item (continued)

- **Copy Value from Item:** Specifies the source of the value that Forms uses to populate the item. When you define a master-detail relation, Forms Builder sets this property automatically on the foreign key item(s) in the detail block. In such cases, the Copy Value from Item property names the primary key item in the master block whose value gets copied to the foreign key item in the detail block whenever a detail record is created or queried.
  Note: The text item should disable input; otherwise, the user could violate the foreign-key relationship. To prevent this, set the Enabled property to No for the foreign-key item, or do not display it at all.

Instructor Note

When data blocks are related through a compound join, the Copy Value from Item property is set on two or more foreign-key items in the detail data block.

Controlling the Data of a Text Item: Synchronize with Item



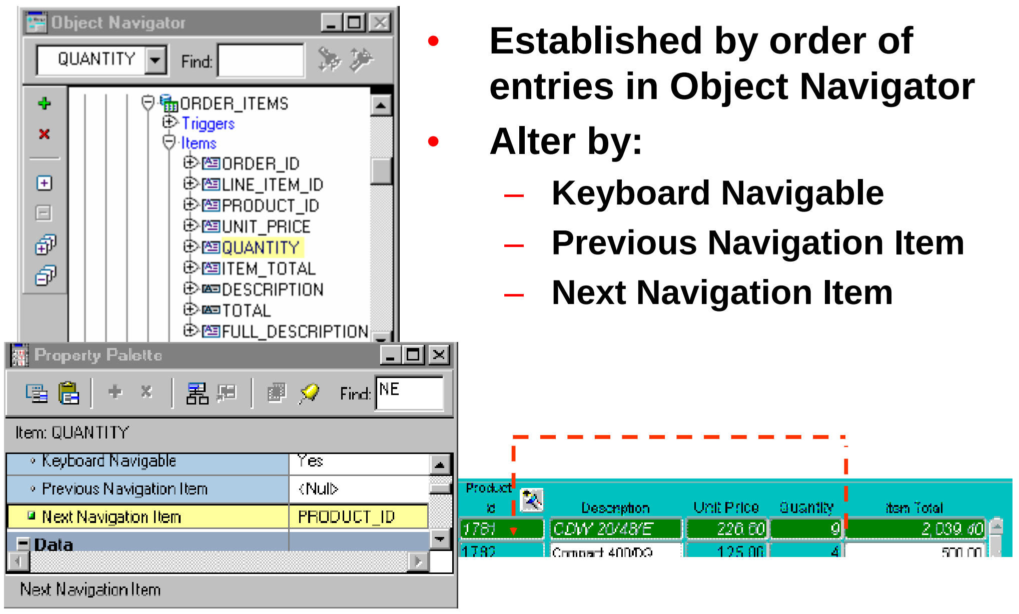
Controlling the Data of a Text Item (continued)

- **Synchronize with Item:** Specifies the name of the item from which the current item should derive its value and synchronizes the values of the two items, so that they effectively mirror each other. When the end user or the application changes the value of either item, the value of the other item changes also.

Instructor Note

You can use the Synchronize with Item property to create more than one item in a single data block that displays the same database column values.

Altering Navigational Behavior of Text Items



- Established by order of entries in Object Navigator
- Alter by:
  - Keyboard Navigable
  - Previous Navigation Item
  - Next Navigation Item

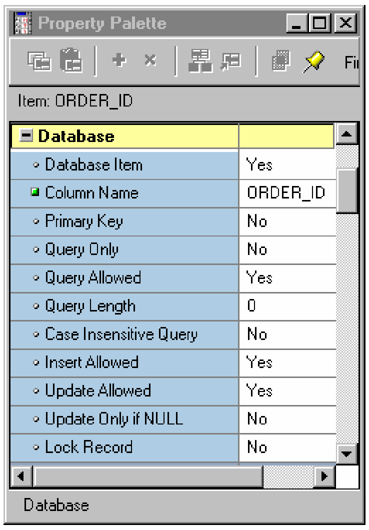
Controlling the Navigational Behavior of Text Items

You can see the default navigational sequence of items in the Object Navigator, as the item entries are displayed in the navigational order. However, you can also use the Navigation group properties to control the navigational behavior of a text item.

Navigation Property	Function
Keyboard Navigable	Determines whether you can navigate to an item during default navigation with the function keys or menu items and place focus on it. When this property is set to No, Forms skips over the item and enters the next navigable item in the default navigation sequence.
Previous Navigation Item	Determines the previous item to be visited when you navigate out of the current item
Next Navigation Item	Determines the next item to be visited when you navigate out of the current item

Note: The next or previous navigation item must be in the same data block as the current item.

Enhancing the Relationship Between Text Item and Database


Property Palette

Item: ORDER_ID

Database	
Database Item	Yes
Column Name	ORDER_ID
Primary Key	No
Query Only	No
Query Allowed	Yes
Query Length	0
Case Insensitive Query	No
Insert Allowed	Yes
Update Allowed	Yes
Update Only if NULL	No
Lock Record	No

Database

Use properties in the Database group to control:

- **Item's data source—base table item or control item**
- **Query, insert, and update operations on an item**
- **Maximum query length**
- **Query case**

Enhancing the Relationship Between Text Item and Database

You can alter or enhance the way in which a text item interacts with its corresponding database column by setting the Database group properties. Some of these are:

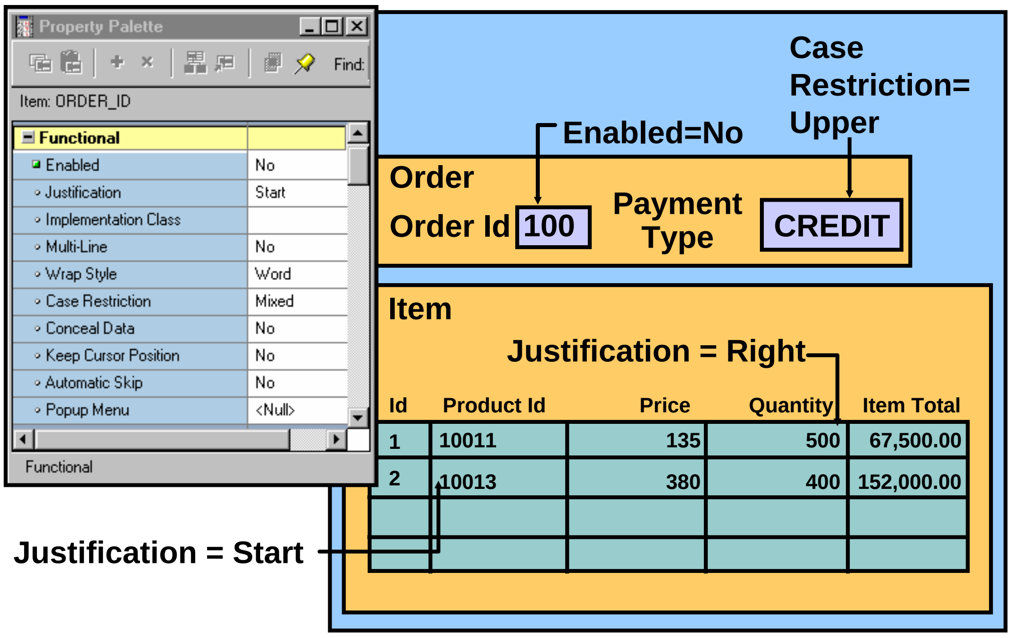
- **Database Item:** Indicates whether the item is a database column
- **Query/Insert/Update Allowed:** Controls whether DML operations are allowed
- **Query Length:** Specifies the maximum length of query criterion in Enter Query mode
- **Case Insensitive Query:** Controls whether case is recognized in query processing

Note: When you create an item in a data block, Forms Builder assumes that the item is a data item, sets its Database Item property to Yes, and automatically includes it in any `SELECT`, `UPDATE`, and `INSERT` statements issued to the database. If an item that you are creating is a control item, you must explicitly set its Database Item property to No.

Instructor Note

- Setting Case Insensitive Query to Yes may take queries longer to execute.
- Update Allowed must be set to No before setting Update Only if NULL to Yes.
- You can use Update Only if NULL for a value that should be set only once, for example, a ship date. Once an order is shipped the ship date cannot be modified.

Adding Functionality to a Text Item



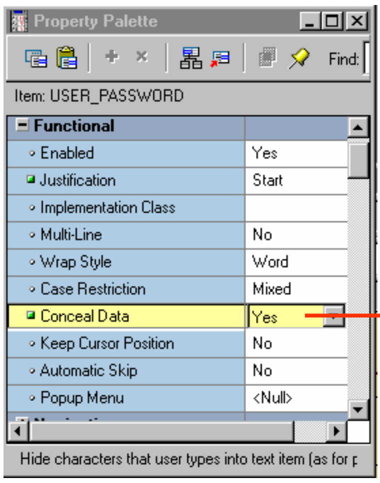
Adding Functionality to a Text Item

Augment the default functionality of a text item by introducing some of the additional features you can set in the Functional group of the Property Palette. Some of these are depicted above or will be discussed in the next few pages. For descriptions of other properties in the Functional group, select the property in the Property Palette and press `[F1]`.

Note

- The Enabled property set to No grays out the item. If you want the item to appear normally but do not want the users to change it, do the following:
  - Set Insert Allowed to No.
  - Set Update Allowed to No.
  - Set Enabled to Yes.
- A pop-up menu is a context-sensitive menu that enables users to access common functions and commands quickly. It is a top-level object in the Object Navigator and belongs to a form module (as opposed to a form menu, which belongs to a separate menu module).

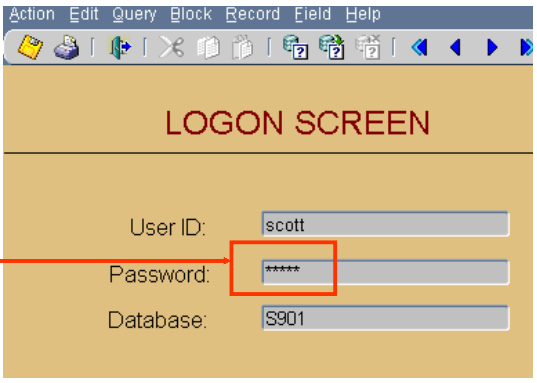
Adding Functionality to a Text Item: Conceal Data Property



Item: USER_PASSWORD

Functional	
Enabled	Yes
Justification	Start
Implementation Class	
Multi-Line	No
Wrap Style	Word
Case Restriction	Mixed
Conceal Data	Yes
Keep Cursor Position	No
Automatic Skip	No
Popup Menu	<Null>

Hide characters that user types into text item (as for p



User ID: scott
Password: *****
Database: S901

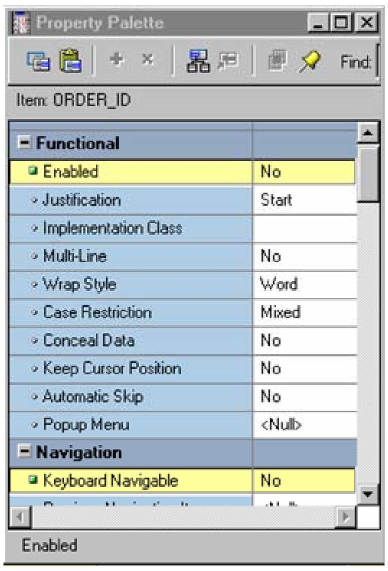
Adding Functionality to a Text Item (continued)

Conceal Data: Hides characters that the operator types into the text item. This setting is typically used for password protection. Choose Yes to disable the echoing back of data entered by the operator; with this setting, the entered value displays as an asterisk for each character entered.
Note: Conceal Data set to Yes is valid only for single-line text items.

Instructor Note

Demonstration: Using the `ordwk05.fmb` file, show the run-time effects of changing the Enabled, Justification, Multi-Line, Case Restriction, Conceal Data, and Automatic Skip properties.

Adding Functionality to a Text Item: Keyboard Navigable and Enabled



Property Palette

Item: ORDER_ID

Functional	
Enabled	No
Justification	Start
Implementation Class	
Multi-Line	No
Wrap Style	Word
Case Restriction	Mixed
Conceal Data	No
Keep Cursor Position	No
Automatic Skip	No
Popup Menu	<Null>
Navigation	
Keyboard Navigable	No

Enabled

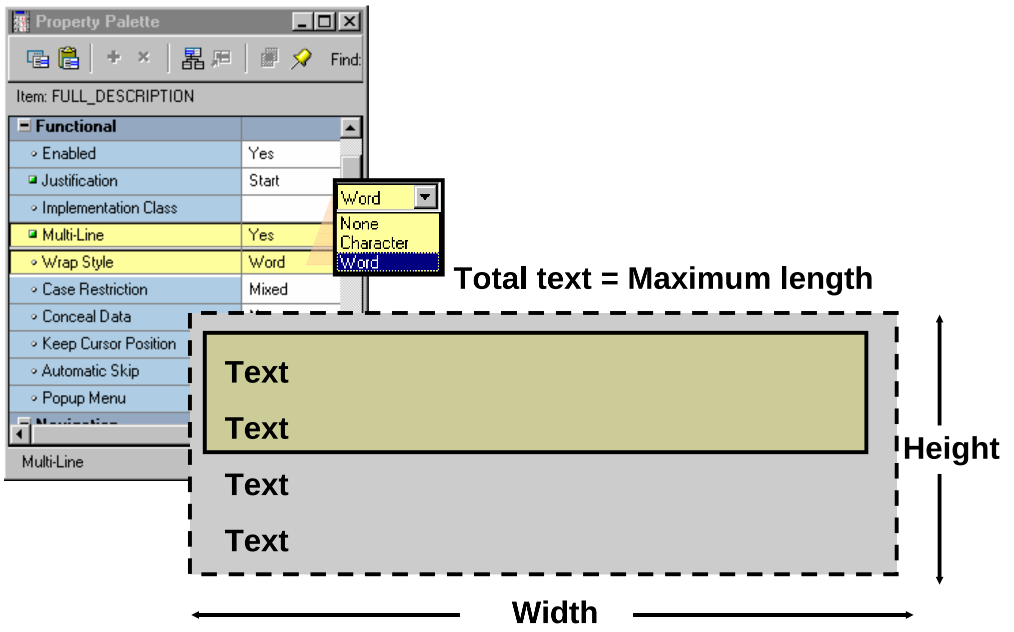
- Set both properties to allow or disallow navigation and interaction with text item.
- When Enabled is set to Yes, Keyboard Navigable can be set to Yes or No.
- When Enabled is set to No, the item is always nonnavigable.

Setting Keyboard Navigable and Enabled Properties

You can set the Keyboard Navigable and Enabled properties for items to specify whether operators can navigate to and interact with them. The Enabled property determines whether end users can use the mouse to manipulate an item. The following table describes the behavior of combinations of these settings:

Enabled	Keyboard Navigable	Navigation Behavior
Yes	Yes	Item is included during default navigation. The item can be navigated to and manipulated with the mouse.
Yes	No	Item is excluded during default navigation. The item can be navigated to and manipulated with the mouse.
No	No	Item is excluded during default navigation. The item cannot be navigated to and manipulated with the mouse.
No	Yes	Item is excluded during default navigation. The item cannot be navigated to and manipulated with the mouse. The Keyboard Navigable property is also effectively set to No.

Adding Functionality to a Text Item: Multi-line Text Items



Item: FULL_DESCRIPTION

Functional	
Enabled	Yes
Justification	Start
Implementation Class	
Multi-Line	Yes
Wrap Style	Word
Case Restriction	Mixed
Conceal Data	
Keep Cursor Position	
Automatic Skip	
Popup Menu	

Word
None
Character
Word

Multi-Line

7-22

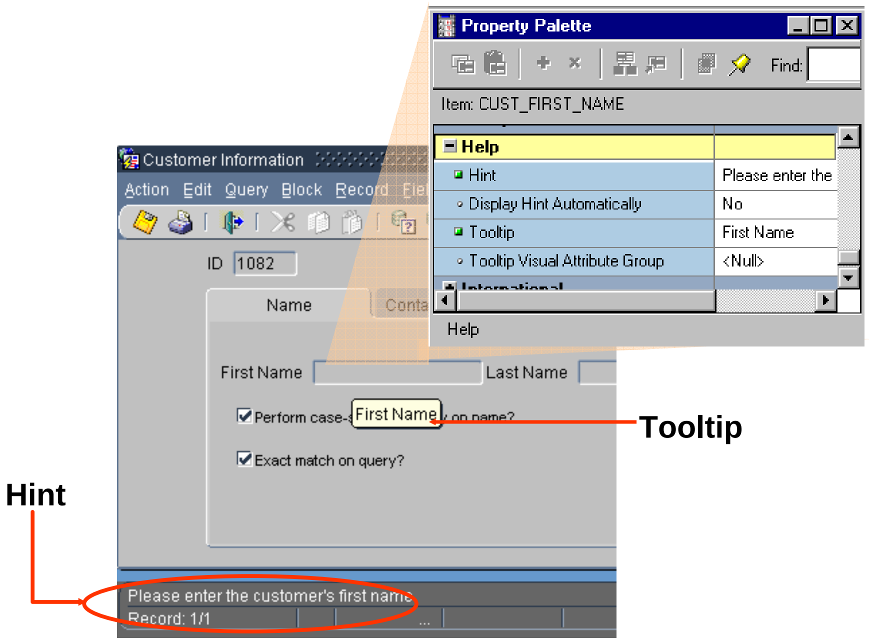
Adding Functionality to a Text Item (continued)

Multi-Line: Determines whether the text item displays in a single-line or multi-line region. Use multi-line text items to display and/or edit such items as addresses, comments, or descriptions. The data in a multi-line text item must be of Char, Alpha, or Long datatype, not numeric or date.

Setting the Multi-Line property to Yes enables a text item to store multiple lines of text, but it does not automatically make the item large enough to display multiple lines. It is up to you to set the Width, Height, Font Size, and Maximum Length properties to ensure that the desired number of lines and characters are displayed.
Note: Setting right or center justification for scrollable text items may result in values being hidden from the user.

Wrap Style: For multi-line text items, specifies how text is displayed when a line of text exceeds the width of a text item or editor window.

Displaying Helpful Messages: Help Properties



Displaying Helpful Messages

You can use the Help group properties to provide context-sensitive help to users:

Help Property	**Function**
Hint	Writes item-specific Help text that is displayed on the message line at run time. The Help text is available when input focus is on the item.
Display Hint Automatically	Determines whether the hint for the item is displayed automatically. If set to No, the hint displays only when the operator presses `[Help]` or selects the Help command on the default menu.
Tooltip	Help text that should appear in a small box beneath the item when the mouse enters the item. The item does not need to have input focus for the tooltip to appear.
Tooltip Visual Attribute Group	Specifies Visual Attribute to use for the tooltip

Instructor Note

Tool tips are used most often for push buttons, which are discussed in Lesson 9.

Summary

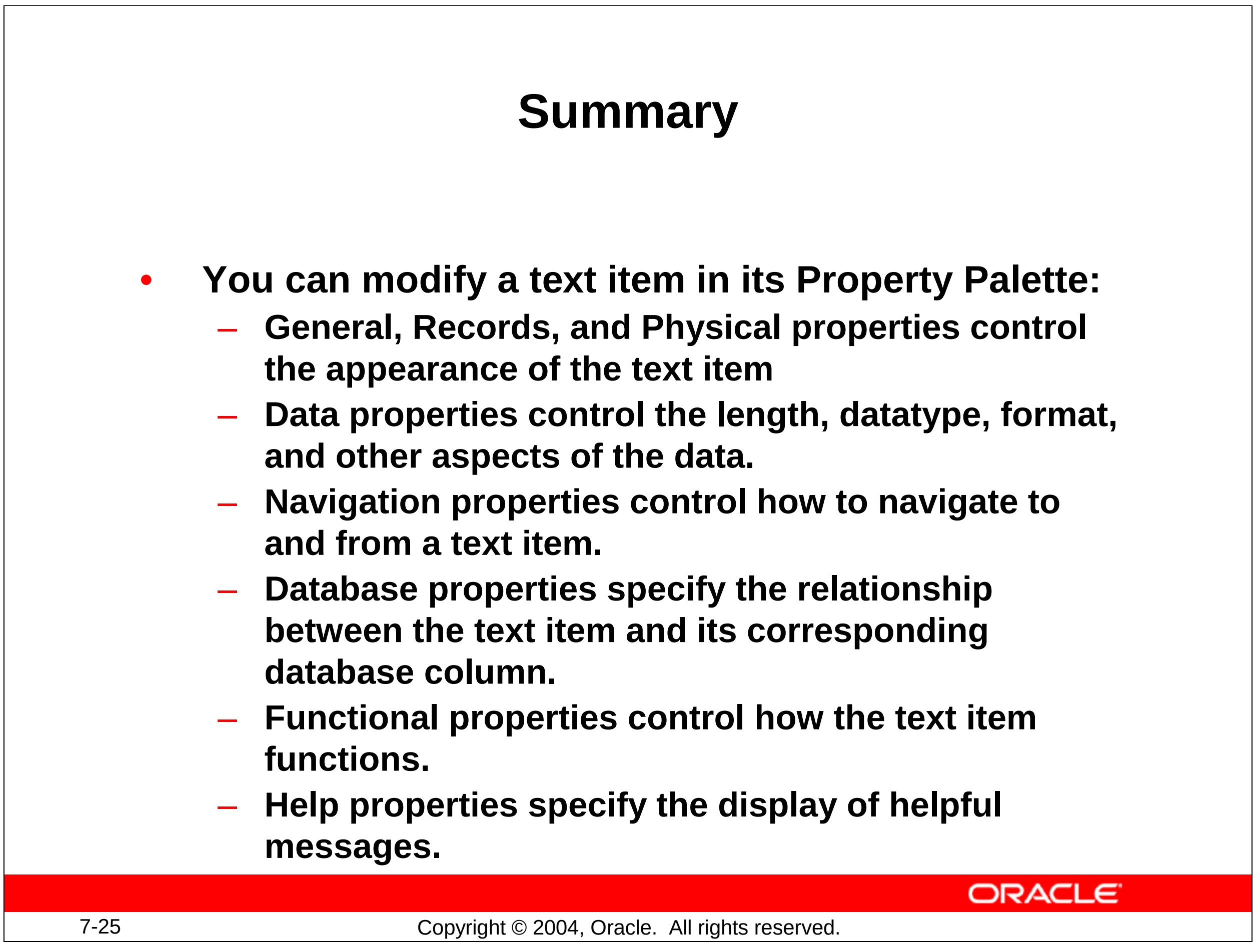
In this lesson, you should have learned that:

- Text items are interface objects that usually correspond to database columns
- You can create a text item with:
  – The Text Item tool in the Layout Editor
  – The Create icon in the Object Navigator
  – The Data Block Wizard

ORACLE

Summary

This lesson showed you how to create and modify a text item that Forms Builder creates for each column flagged for inclusion in a data block.



Summary

- You can modify a text item in its Property Palette:
  - General, Records, and Physical properties control the appearance of the text item
  - Data properties control the length, datatype, format, and other aspects of the data.
  - Navigation properties control how to navigate to and from a text item.
  - Database properties specify the relationship between the text item and its corresponding database column.
  - Functional properties control how the text item functions.
  - Help properties specify the display of helpful messages.

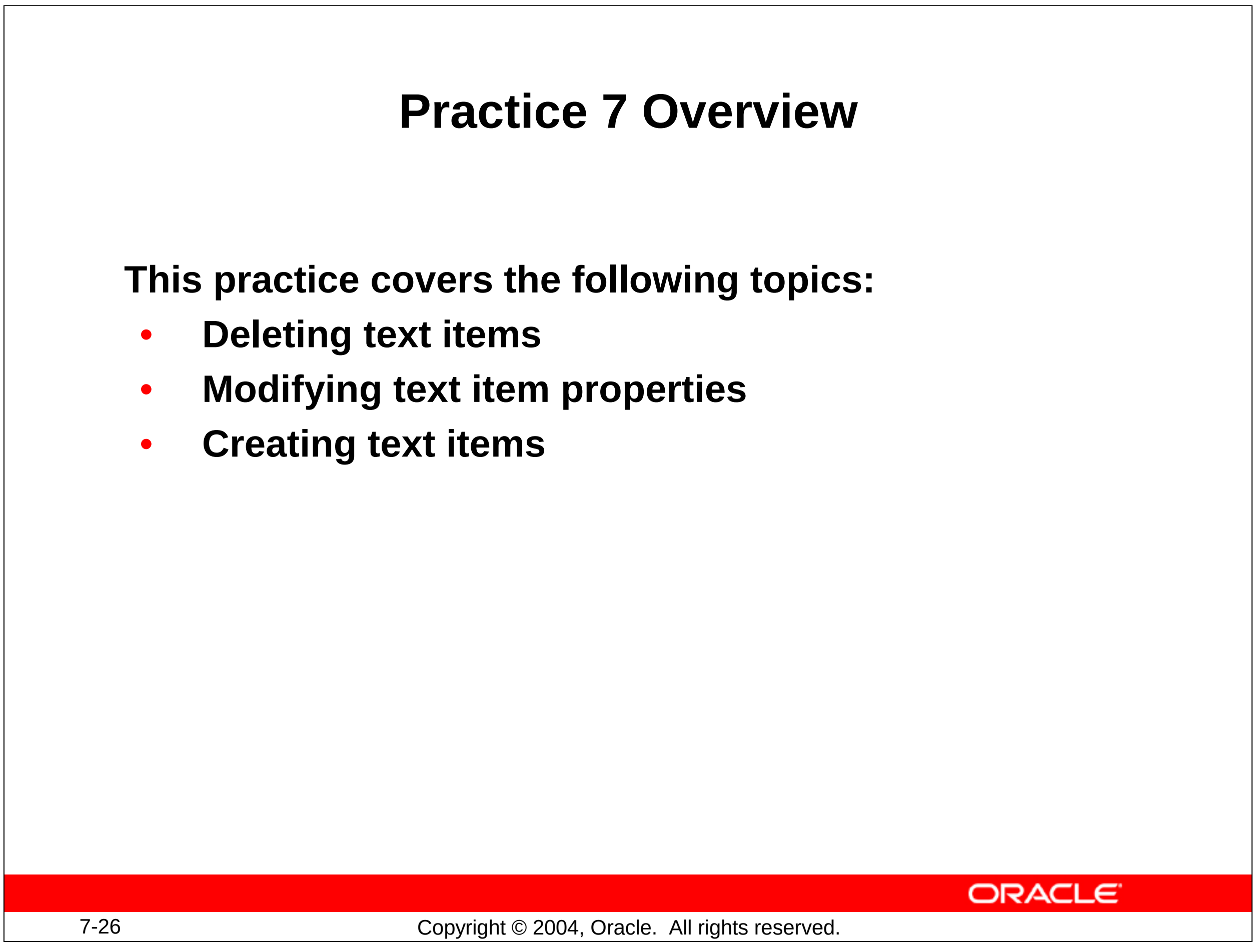
ORACLE

7-25

Summary (continued)

In particular, text items have properties that enable you to do the following:

- Modify their appearance
- Control the data stored in the item
- Alter navigational behavior
- Enhance the relationship with the database
- Add functionality
- Include Help information



Practice 7 Overview

This practice covers the following topics:
- Deleting text items
- Modifying text item properties
- Creating text items

7-26 Copyright © 2004, Oracle. All rights reserved.

Practice 7 Overview

In this practice session you will create text items, alter the behavior and the appearance of text items, and delete text items.

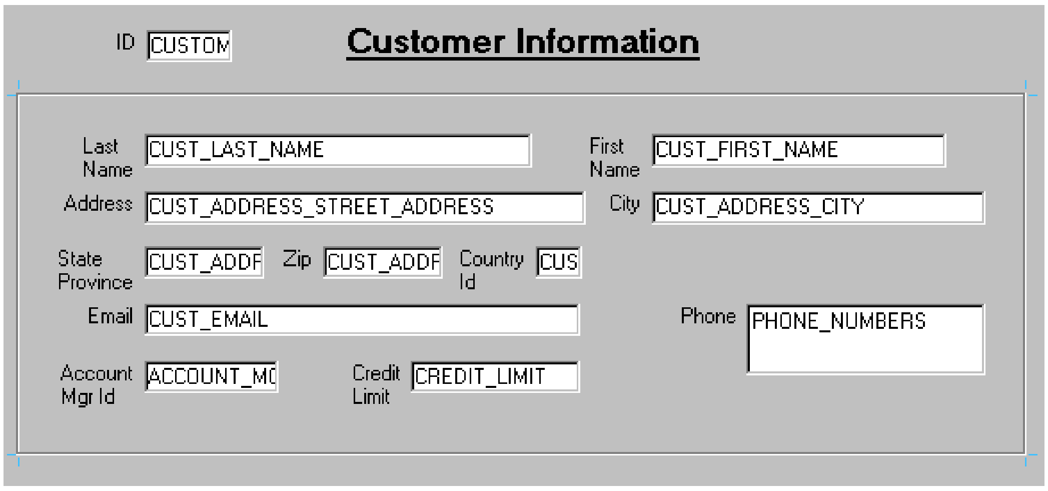
- Delete the region ID item in the CUSTOMERS form.
- Using the Property Palette, change the properties of several text items in the `CUSTOMERS` data block to change their run-time appearance. Save and run the form after the changes are applied.
- In the ORDERS form, create new text items to hold the customer name and sales rep name values in the `ORDERS` block, and set the suggested properties. Change additional text item properties in the `ORDERS`, `ORDER_ITEMS`, and `INVENTORIES` data blocks to change their run-time appearance and behavior. Save and run the form after the changes are applied.

Note: For solutions to this practice, see Practice 7 in Appendix A, “Practice Solutions.”

Practice 7

CUSTG*XX* Form

1. Remove the NLS_Language and NLS_Territory items.
2. Make sure that the Phone_Numbers item accepts multi-line text to display. The database column is long enough to accept two phone numbers if the second one is entered without “+1” in front of the number.
3. Automatically display a unique, new customer number for each new record and ensure that it cannot be changed.
   Use the `CUSTOMERS_SEQ` sequence.
4. In the CUSTG*XX* form, resize and reposition the items. Add the boilerplate text Customer Information. Reorder the items in the Object Navigator. Use the screenshot as a guide.



ID CUSTOM

Customer Information

Last Name CUST_LAST_NAME
First Name CUST_FIRST_NAME
Address CUST_ADDRESS_STREET_ADDRESS
City CUST_ADDRESS_CITY
State Province CUST_ADDF
Zip CUST_ADDF
Country Id CUS
Email CUST_EMAIL
Phone PHONE_NUMBERS
Account Mgr Id ACCOUNT_MC
Credit Limit CREDIT_LIMIT

5. Save and compile your form.
   Click Run Form to run your form and test the changes.
   Note: The entire form may not be visible at this time. This will be addressed in a later lesson.

Instructor Note

Customer_Name and Sales_Rep_Name are created as text items in this course. They could also be created as display items. Display items use less memory than text items.

Practice 7 (continued)

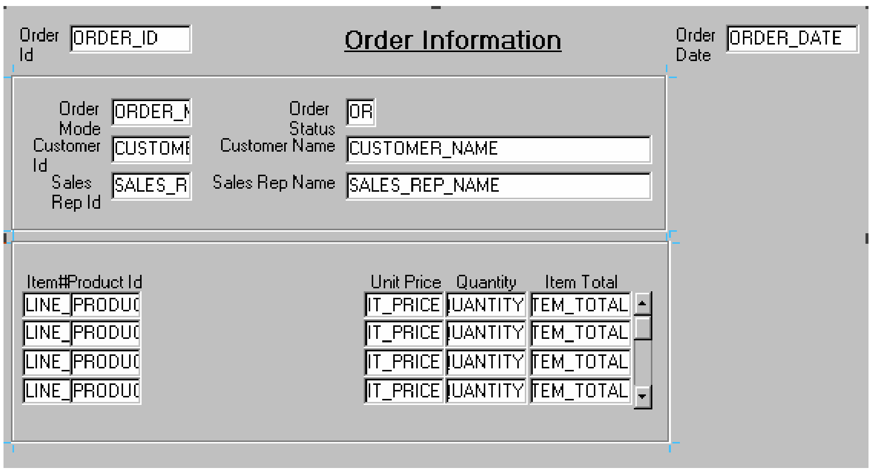
ORDG*XX* Form

6. In the `ORDERS` block, create a new text item called Customer_Name.
   Ensure that Customer_Name is not associated with the `ORDERS` table.
   Do not allow insert, update, or query operations on this item, and make sure that navigation is possible only by means of the mouse. Set the Prompt text to Customer Name. Display this item on `CV_ORDER` canvas.
7. In the `ORDERS` block, create a new text item called Sales_Rep_Name.
   Ensure that Sales_Rep_Name is not associated with the `ORDERS` table.
   Do not allow insert, update, or query operations on this item and make sure that navigation is possible only by means of the mouse. Set the Prompt text to Sales Rep Name. Display this item on the `CV_ORDER` canvas.
8. Set the relevant property for Order_Date, so that it displays the current date whenever a new record is entered.
9. In the `ORDER_ITEMS` block, create a new text item called Item_Total.
   Ensure that Item_Total is not associated with the `ORDER_ITEMS` table.
   Do not allow insert, update, or query operations on this item and make sure that navigation is possible only by means of the mouse.
   Allow numeric data only and display it by using a format of 999G990D99.
   Set the Prompt text to Item Total. Display this item on the `CV_ORDER` canvas.
10. Justify the values of Unit_Price, Quantity, and Item_Total to the right.
11. Alter the Unit_Price item, so that navigation is possible only by means of the mouse, and updates are not allowed. Set its format mask to be the same as that used for Item_Total.
12. In the ORDG*XX* form, resize and reposition the items according to the screenshot and the following table.

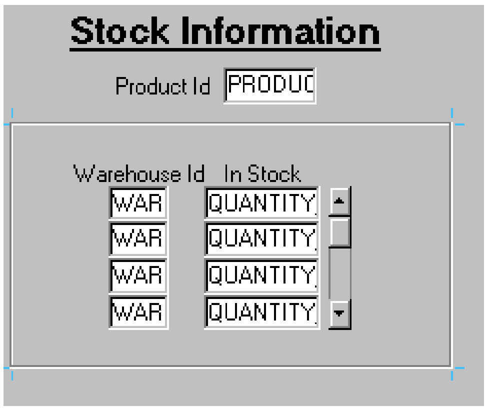
ORDERS Block Items	**Suggested Width**
Order_Id	60
Order_Date	65
Order_Mode	40
Customer_ID	40
Customer_Name	150
Order_Status	15
Sales_Rep_ID	40
Sales_Rep_Name	150

ORDER_ITEMS Block Items	**Suggested Width**
Line_Item_Id	25
Product_ID	35
Unit_Price	40
Quantity	40
Item_Total	50

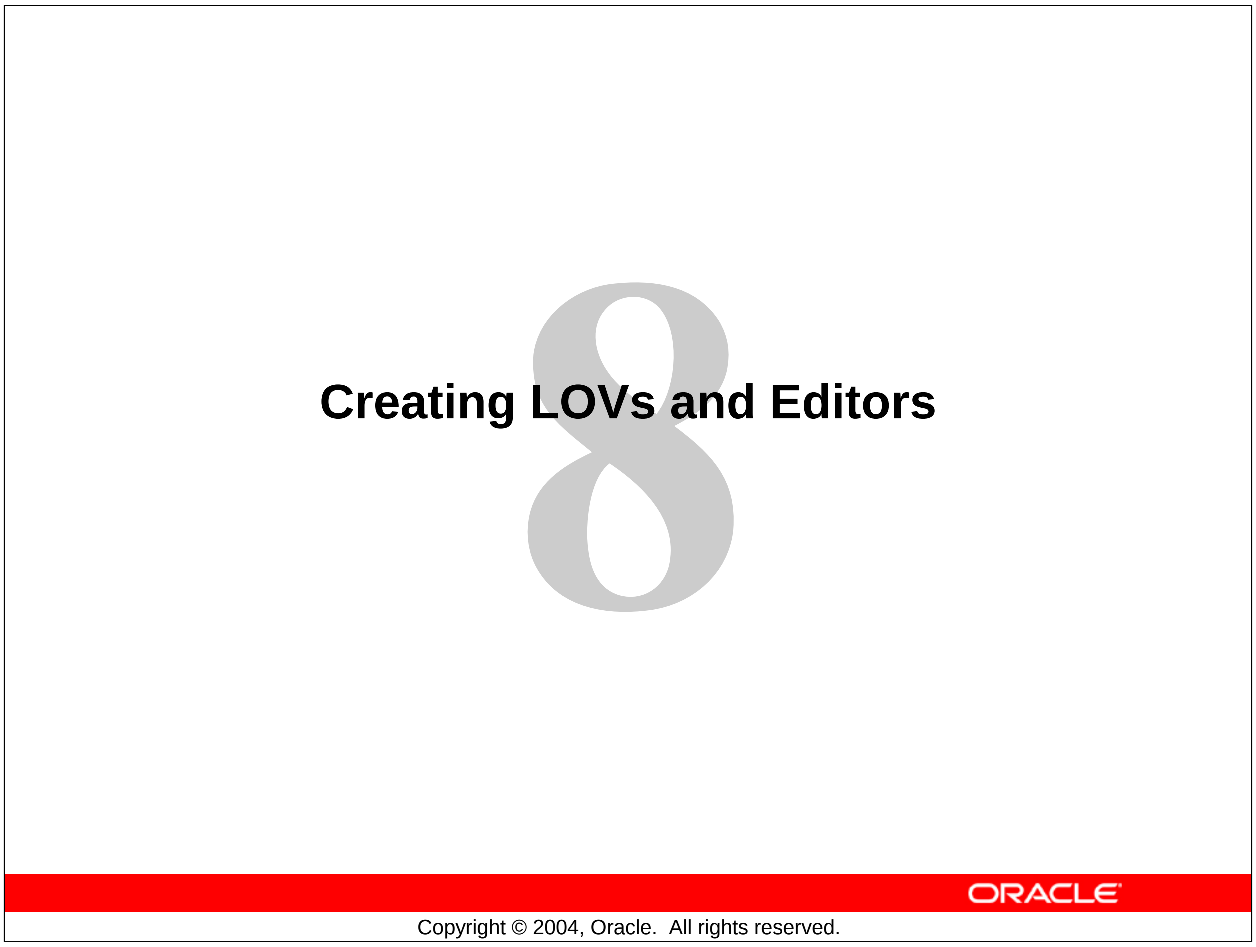
Practice 7 (continued)



13. In the `INVENTORIES` block, alter the number of instances of the Product_ID, so that it is displayed just once. Make its prompt display to the left of the item.
14. Arrange the items and boilerplate on `CV_INVENTORY`, so that it resembles the screenshot.
    Hint: Set the Update Layout property for the frame to Manually.



15. Save, compile, and run the form to test the changes.



8 Creating LOVs and Editors

Schedule:	Timing	Topic
	30 minutes	Lecture
	30 minutes	Guided Practice
	60 minutes	Total

Objectives

After completing this lesson, you should be able to do the following:

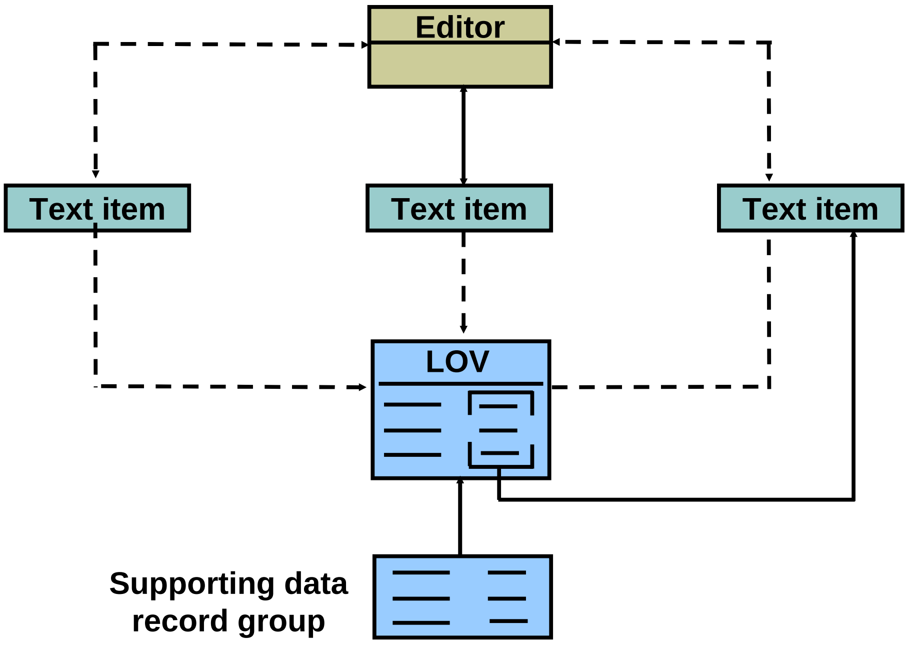
- Describe LOVs and editors
- Design, create, and associate LOVs with text items in a form module
- Create editors and associate them with text items in a form module

Introduction

Overview

With Oracle Forms Developer you can enhance your application with lists of available values and text editors to supplement the text item object. In this lesson you will learn how to create lists of values (LOVs) and text editors, and to associate them with items in your application.

Overview of LOVs and Editors



What Are LOVs and Editors?

Lists of values (LOV) and editors are objects in a form module that each open their own window when activated at run time. They are defined at the form level, which means you can use them to support text items in any block of the form module.

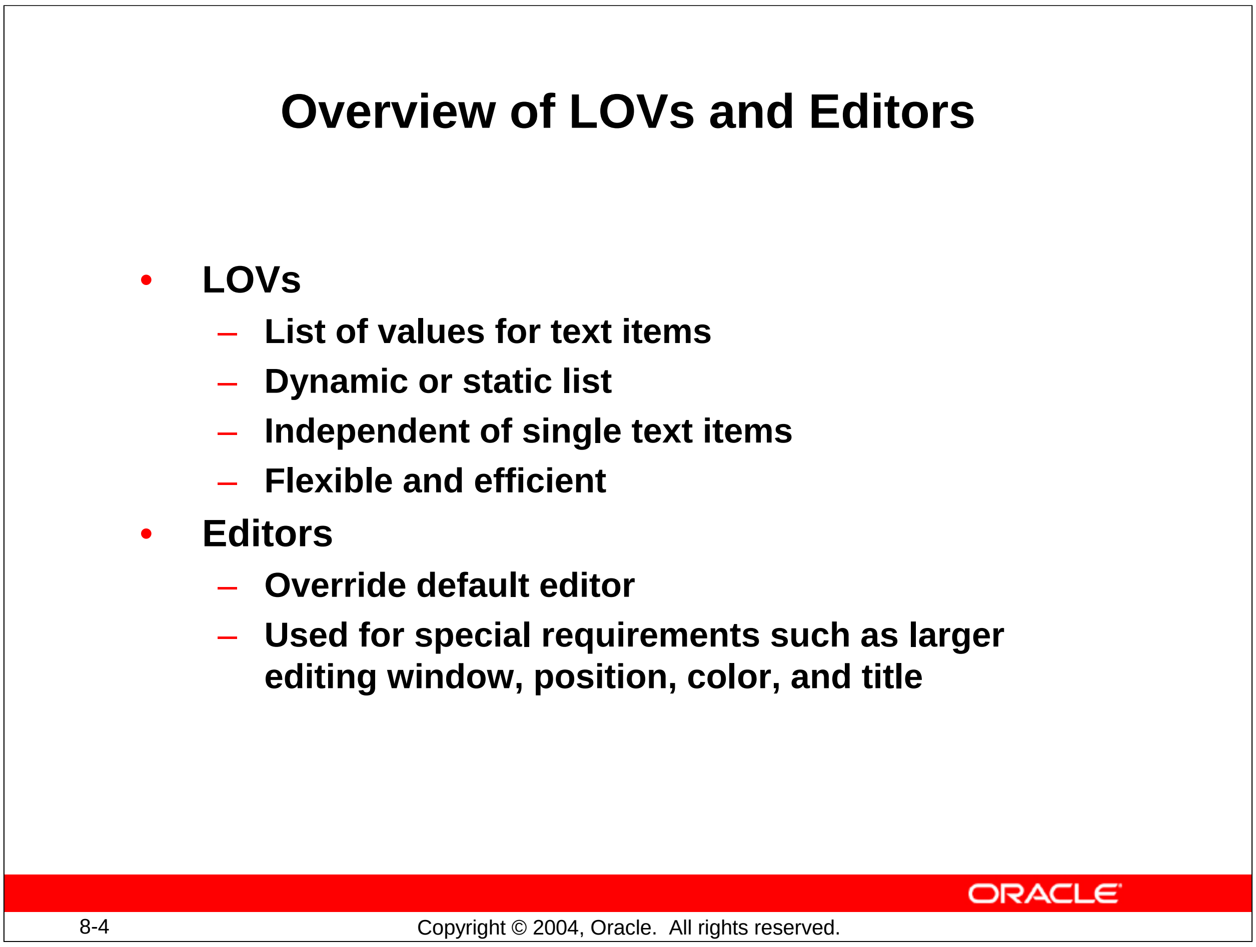
LOVs

An LOV is a scrollable pop-up window that enables a user to pick the value of an item from a multicolumn dynamic list. The user can reduce the lines displayed in the list by simple automatic reduction techniques, or by search strings.

Each line in an LOV can present several field values, with column headings above. You can design your LOV to retrieve some or all of the field values from the line chosen by the user, and place them into form items or variables.

LOVs have the following qualities:

- **Dynamic:** The list entries can change to reflect changes in the source data.
- **Independent:** The designer can invoke an LOV from any text item, or from outside a text item if called programmatically.
- **Flexible:** You can use the same LOV to support several items, if appropriate (for example, product_ID, product_name).
- **Efficient:** You can design LOVs to reuse data already loaded into the form, instead of accessing the database for every call. This is useful where data is relatively static.



Overview of LOVs and Editors

- LOVs
  - List of values for text items
  - Dynamic or static list
  - Independent of single text items
  - Flexible and efficient
- Editors
  - Override default editor
  - Used for special requirements such as larger editing window, position, color, and title

8-4

What Are LOVs and Editors? (continued)

How to use an LOV at run time

When a text item has an LOV attached, the List of Values lamp displays on the status line, while the cursor is in the item.

1. Either press the List of Values key, or select Edit > Display List to invoke the LOV.
2. Select an entry in the displayed list. You can type characters to automatically reduce the list, or enter a search string in the Find field.
3. Click OK to retrieve the line value.

Note: Automatic reduction works by comparing the search string entered with the values displayed in the first column of the LOV. If you start your search criteria with a % symbol, Forms performs a search on all LOV columns.

What Are LOVs and Editors? (continued)

Editors

With a text editor enabled the user can view multiple lines of a text item simultaneously, search and replace text in it, and generally modify the value of an item from this separate window.

You can use one of two editors at run time:

- Forms Builder default editor
- User-named editor

Every text item has the default editor available, but you can design your own replacement editor for those items that have special requirements such as larger editing window, position, color, and title.

By overriding the default editor for a text item, you can provide a larger editing window for items with potentially large textual values.

How to use an editor at run time

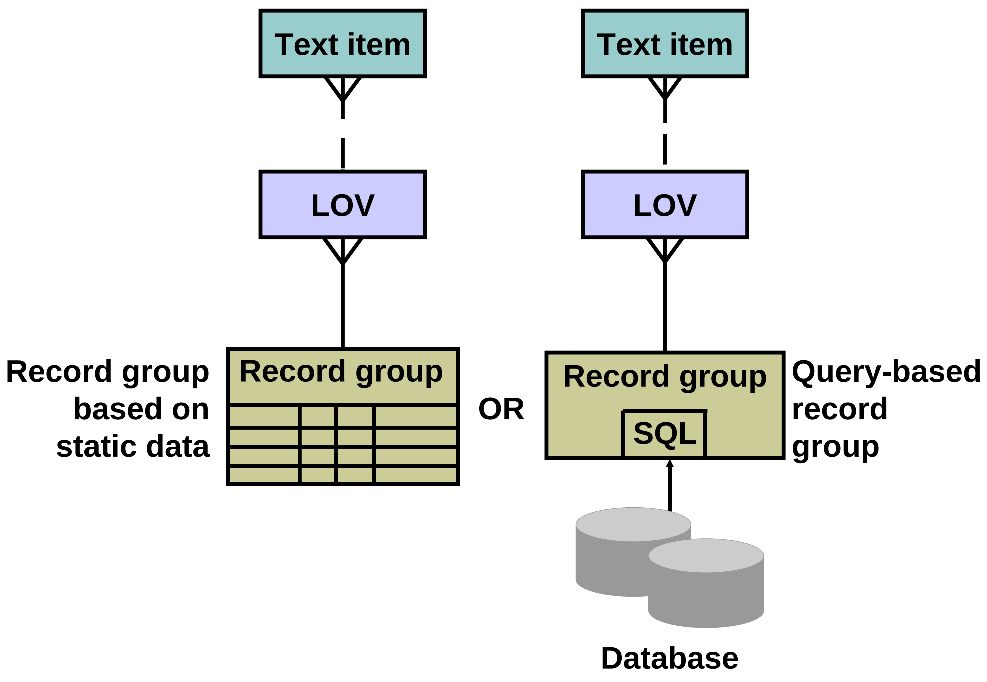
With the cursor in the text item to be edited, follow these steps:

1. Press the Edit key, or select Edit > Edit to invoke the attached editor.
2. Edit the text in the Editor window. Forms Builder editors provide a Search button that invokes an additional search-and-replace dialog box for manipulating text.
3. Click OK to write your changes back to the text item.

Instructor Note

- The `EDIT_TEXTITEM` built-in invokes the editor associated with the current text item. The `SHOW_EDITOR` built-in invokes a user-named editor at the specified display coordinates.
- Previous versions of Forms allowed use of the system editor in client/server mode. However, this is not available for Web-deployed forms.

LOVs and Record Groups



8-6 Copyright © 2004, Oracle. All rights reserved.

LOVs and Record Groups

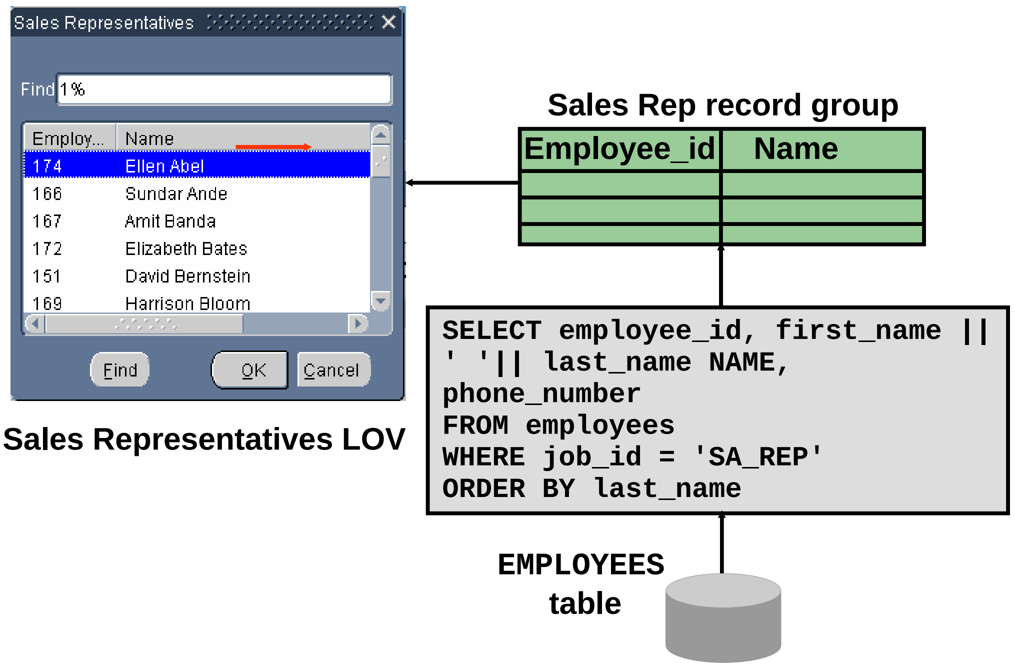
When you build an LOV, consider the following objects:

- **Record group:** A Forms Builder object that is used to store the array of values that are presented by an LOV (The record group can be created first or as part of the LOV creation process if based on a query.)
- **LOV:** The list itself, which presents one or more column values from the supporting record group in the LOV window (It enables the user to select values, and then write values back to specified items or variables.)
- **Text items:** The main text item that you attach to an LOV is usually one that the LOV returns a value to. You can call the LOV from this item to provide possible values for it. A single LOV can return values to several items. You can attach the LOV to any text item from which the same list of values needs to be viewed, whether or not it will receive a value.

Instructor Note

Older versions of Forms supported a different type of LOV not based on a record group, called Old-Style (also known as V2.3-style). This is no longer supported.

LOVs and Record Groups



Sales Representatives

Find 1%

Employ...	Name
174	Ellen Abel
166	Sundar Ande
167	Amit Banda
172	Elizabeth Bates
151	David Bernstein
169	Harrison Bloom

```
SELECT employee_id, first_name ||
' '|| last_name NAME,
phone_number
FROM employees
WHERE job_id = 'SA_REP'
ORDER BY last_name
```

LOVs and Record Groups (continued)

Record Groups

A record group is a column-and-row structure stored within Forms Runtime memory and is similar to the structure of a database table. It holds records that can be reused by other Oracle Forms applications and Oracle Reports applications, hence reducing repeated access to external data.

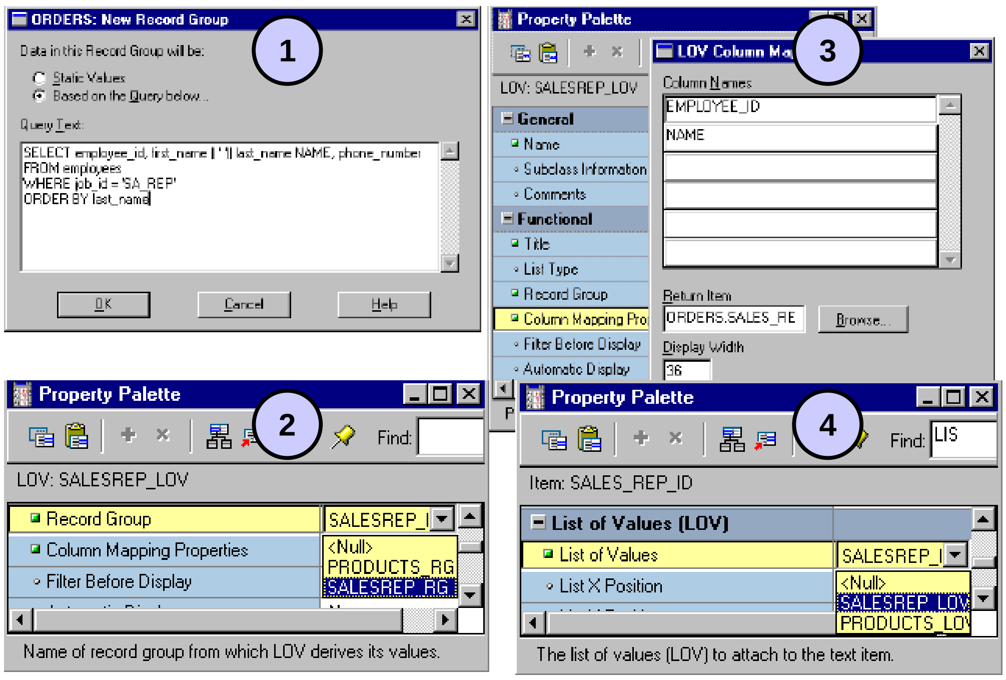
Record groups can be designed to contain static values. Alternatively, they can be populated programmatically at run time or, most commonly, populated by a SQL query. In this lesson, you use record groups to support LOVs.

Record groups can provide the following:

- Data that is presented by LOVs
- Data for dynamic list items
- Other application-defined uses

Note: Because LOVs and record groups are separate objects, you can create multiple LOVs based on the same record group.

Creating an LOV Manually


1

ORDERS: New Record Group

Data in this Record Group will be:

Static Values [✓ Based on the Query below...]

Query Text:

```
SELECT employee_id, first_name || ' '|| last_name NAME, phone_number
FROM employees
WHERE job_id = 'SA_REP'
ORDER BY last_name
```

2

Property Palette

Find:

LOV: SALESREP_LOV

Record Group	SALESREP_I
Column Mapping Properties	<Null> PRODUCTS_RG SALESREP_RG
Filter Before Display	

Name of record group from which LOV derives its values.

3

Property Palette

LOV: SALESREP_LOV

General: Name, Subclass Information, Comments

Functional: Title, List Type, Record Group, Column Mapping Pro, Filter Before Display, Automatic Display

LOV Column Ma

Column Names: EMPLOYEE_ID, NAME

Return Item: ORDERS.SALES_RE

Display Width: 36

4

Property Palette

Find: LIS

Item: SALES_REP_ID

List of Values (LOV)	
List of Values	SALESREP_I
List X Position	<Null> SALESREP_LOV PRODUCTS_LOV

The list of values (LOV) to attach to the text item.

Creating an LOV Manually

Because Forms Builder has an LOV Wizard for you to use in creating LOVs and their associated record groups, you may never need to create an LOV manually. However, knowing how to do so helps you to understand how to set the properties of the record group, the LOV, and the item to which it is attached, even if using the Wizard.

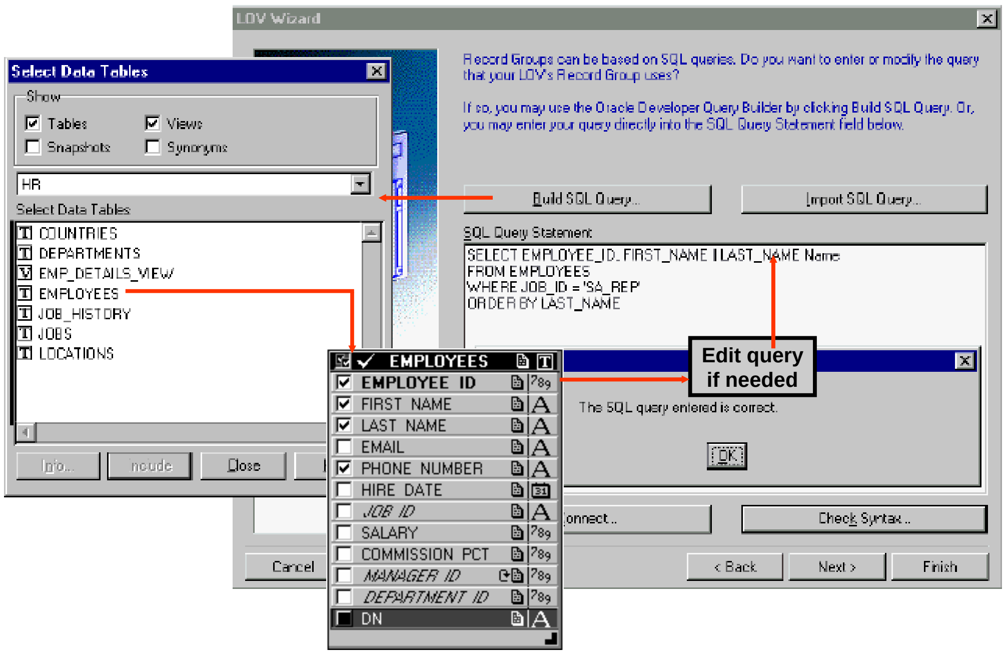
The steps to create an LOV manually are:

1. Create the record group. You will need to type in the query on which the record group is based.
2. Create the LOV and set its Record Group property to the appropriate record group.
3. Set the LOV property Column Mapping. You must type in the columns and their headings, then select a return item for each item that you want to populate from the LOV.
4. Assign the LOV to any text items from which you want the LOV to be available.

Instructor Note

Demonstrate this for the students how an LOV can be created manually, following the above steps. To create the record group and the LOV, select the appropriate node in the Object Navigator and click Create.

Creating an LOV with the LOV Wizard: SQL Query Page



Record Groups can be based on SQL queries. Do you want to enter or modify the query that your LOV's Record Group uses?

If so, you may use the Oracle Developer Query Builder by clicking Build SQL Query. Or, you may enter your query directly into the SQL Query Statement field below.

```
SELECT EMPLOYEE_ID, FIRST_NAME ||LAST_NAME Name
FROM EMPLOYEES
WHERE JOB_ID = 'SA_REP'
ORDER BY LAST_NAME
```

The SQL query entered is correct.

Creating an LOV with the LOV Wizard

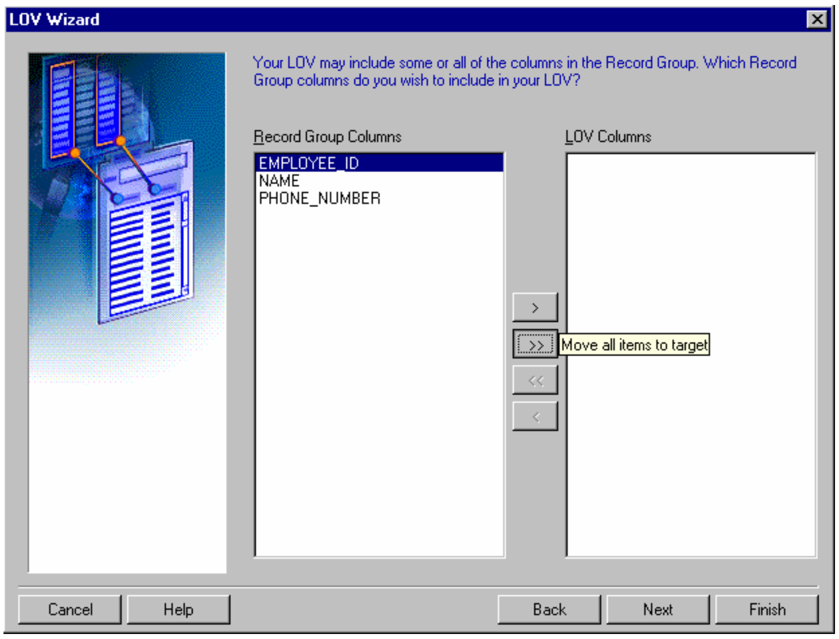
It is easy to make a mistake or to forget one of the manual steps. This can be avoided by using the LOV Wizard, which guides you through the process. To create an LOV with the wizard, perform the following steps:

1. Launch the LOV Wizard.
   The Welcome page appears. Click Next. The LOV Source page appears.
2. Specify the LOV source in the LOV Source page. Choose an existing record group or create a new one based on a query. The default option is New Record Group based on a query. Click Next to select the default. The SQL Query page appears.
3. In the SQL Query page specify the query that is used for the record group. You cannot include a column of a complex object data type. Use one of the following three options for constructing the query:
   - Click Build SQL Query to use Query Builder.
   - Click Import SQL Query to import the query from a file.
   - Type the SQL syntax in the SQL Query Statement field to enter the query directly. Then click Check Syntax.

   After defining the query, click Next. The Column Selection page appears.

Note: See Appendix G for more information about Query Builder.

Creating an LOV with the LOV Wizard: Column Selection Page


LOV Wizard

Your LOV may include some or all of the columns in the Record Group. Which Record Group columns do you wish to include in your LOV?

Record Group Columns

EMPLOYEE_ID
NAME
PHONE_NUMBER

LOV Columns

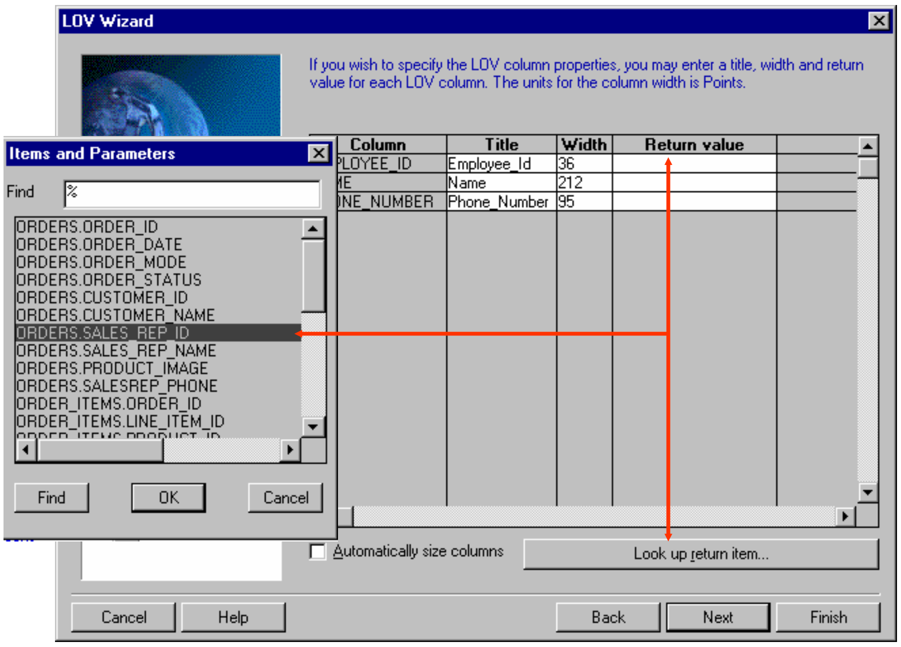
Move all items to target

ORACLE

Creating an LOV with the LOV Wizard (continued)

4. In the Column Selection page, select the record columns that you want to include in the LOV. Click Next. The Column Properties page displays.

Creating an LOV with the LOV Wizard: Column Properties Page



If you wish to specify the LOV column properties, you may enter a title, width and return value for each LOV column. The units for the column width is Points.

Column	Title	Width	Return value
PLOYEE_ID	Employee_Id	36	
IE	Name	212	
INE_NUMBER	Phone_Number	95	

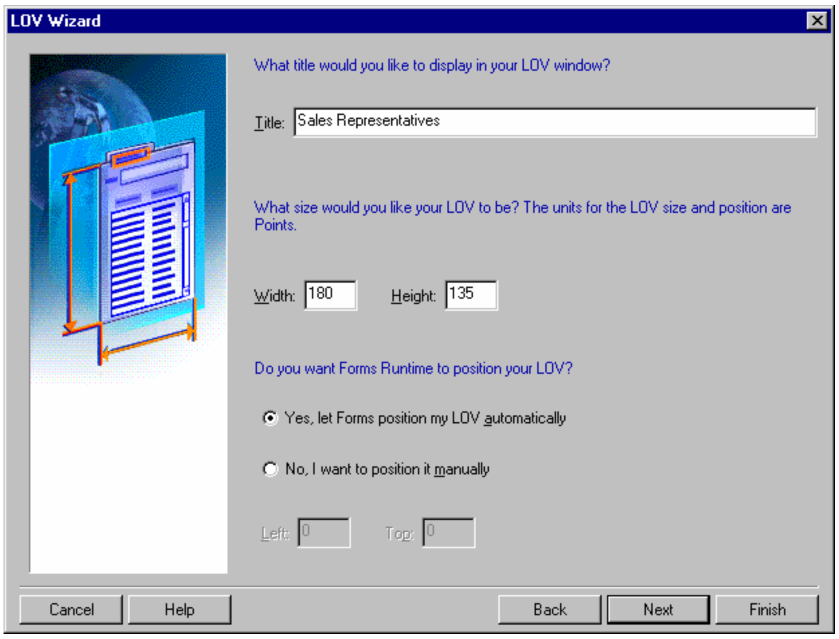
Find: %

ORDERS.ORDER_ID
ORDERS.ORDER_DATE
ORDERS.ORDER_MODE
ORDERS.ORDER_STATUS
ORDERS.CUSTOMER_ID
ORDERS.CUSTOMER_NAME
ORDERS.SALES_REP_ID
ORDERS.SALES_REP_NAME
ORDERS.PRODUCT_IMAGE
ORDERS.SALESREP_PHONE
ORDER_ITEMS.ORDER_ID
ORDER_ITEMS.LINE_ITEM_ID

Creating an LOV with the LOV Wizard (continued)

5. In the Column Properties page, specify the title, width and return value for each LOV column. Note that Return Value is optional. Click Next. The LOV Display page displays.

Creating an LOV with the LOV Wizard: Display Page


LOV Wizard

What title would you like to display in your LOV window?

Title: Sales Representatives

What size would you like your LOV to be? The units for the LOV size and position are Points.

Width: 180 Height: 135

Do you want Forms Runtime to position your LOV?

[✓ Yes, let Forms position my LOV automatically]

No, I want to position it manually

Left: 0 Top: 0

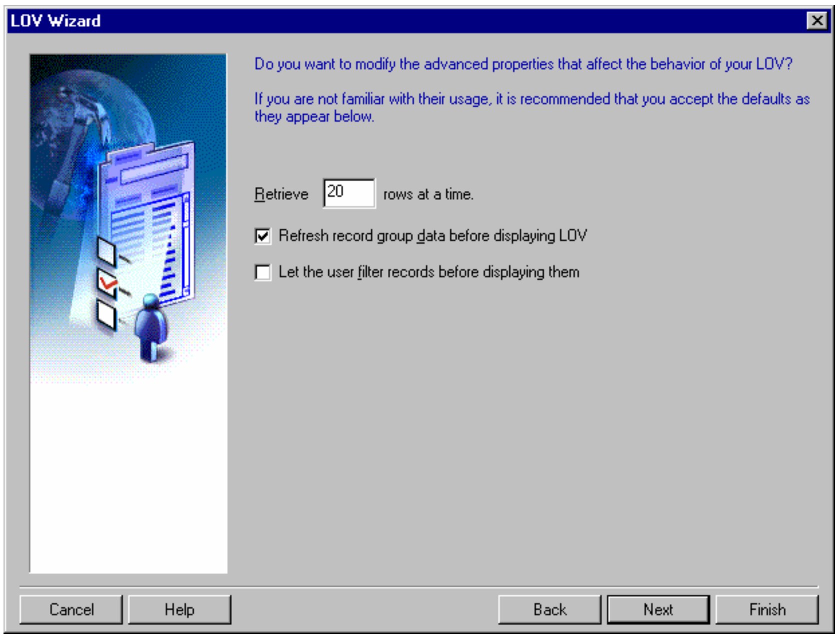
ORACLE

8-12

Creating an LOV with the LOV Wizard (continued)

6. In the LOV Display page, specify the title, the width, and the height of the LOV window. You can choose to display it at a set position that you manually define, or let Forms position it automatically. Click Next. The Advanced Options page displays.

Creating an LOV with the LOV Wizard: Advanced Properties Page



Do you want to modify the advanced properties that affect the behavior of your LOV?

If you are not familiar with their usage, it is recommended that you accept the defaults as they appear below.

Retrieve 20 rows at a time.

☑ Refresh record group data before displaying LOV

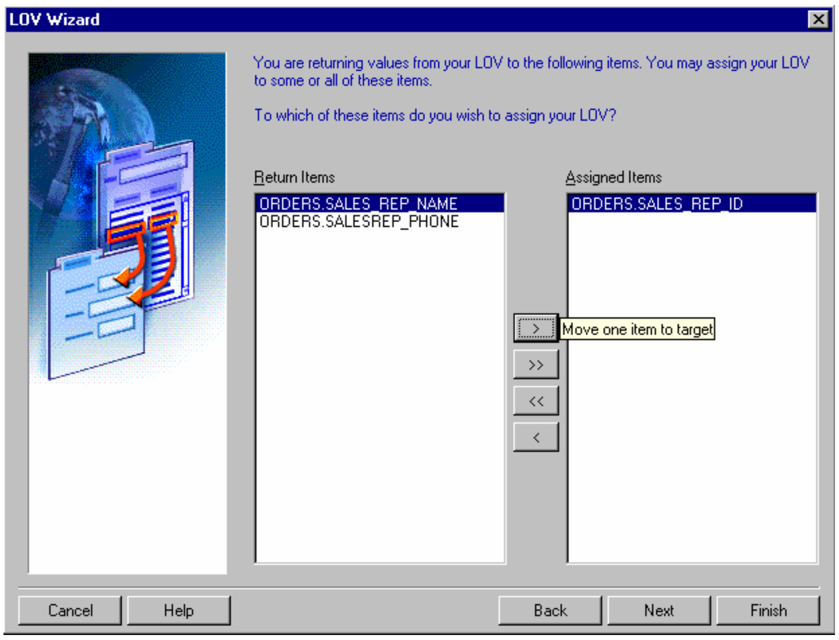
☐ Let the user filter records before displaying them

Creating an LOV with the LOV Wizard (continued)

7. In the Advanced Options page, set the additional advanced properties. Specify:
   - The number of records at a time to be fetched from the database
   - If the LOV records should be queried each time the LOV is invoked
   - If the user should be presented with a dialog box to add criteria before the LOV is displayed

   Click Next. The Assign to Item page displays.

Creating an LOV with the LOV Wizard: Assign to Item Page


LOV Wizard

You are returning values from your LOV to the following items. You may assign your LOV to some or all of these items.

To which of these items do you wish to assign your LOV?

Return Items
- ORDERS.SALES_REP_NAME
- ORDERS.SALESREP_PHONE

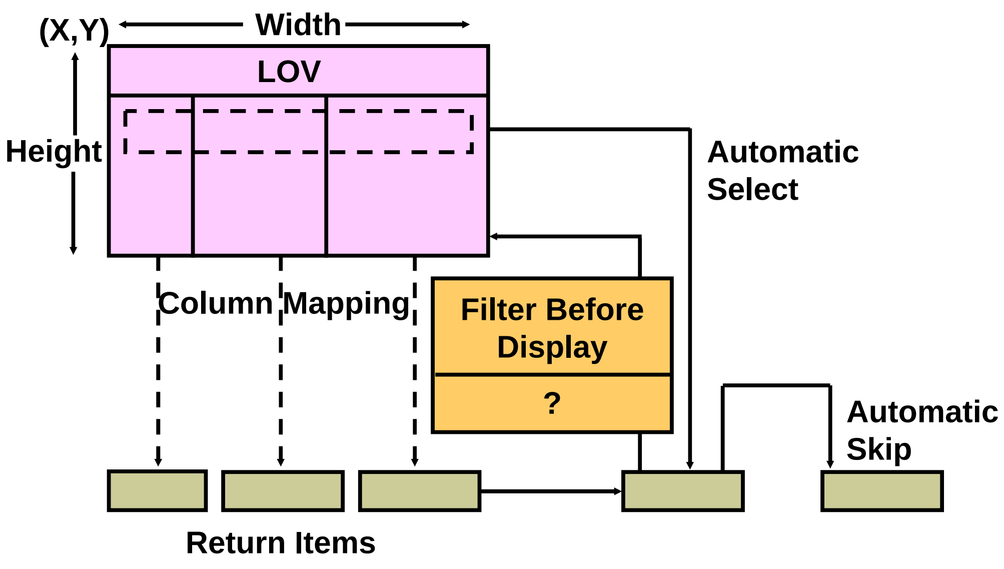
Assigned Items
- ORDERS.SALES_REP_ID

Creating an LOV with the LOV Wizard (continued)

8. In the Assign to Item page, select the items to which your LOV should be attached. At run time, the LOV will be available from these items so that operators may use it while input focus is in one of these items. Click Next. The Finish page displays.
9. On the Finish page, click Finish to complete the LOV creation process.

Note: The LOV Wizard is reentrant, so you can use it to modify the LOV after it is created. In the Object Navigator, click the LOV to be modified, and choose Tools > LOV Wizard from the menu.

LOV Properties

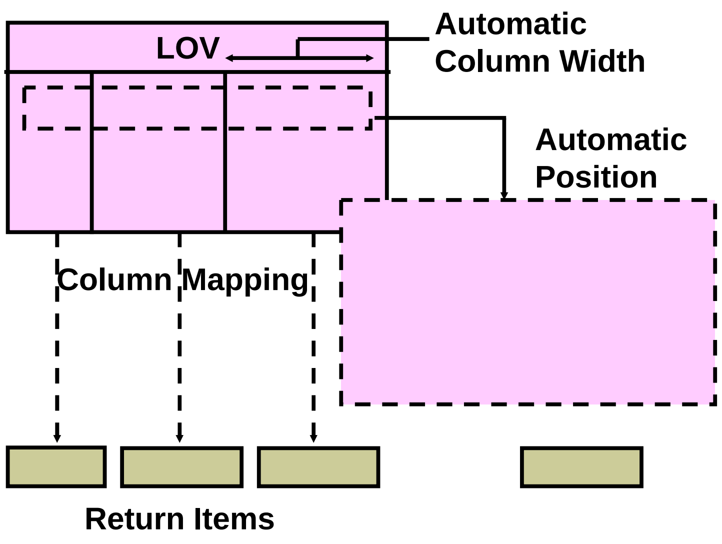


Setting LOV Properties

After you create an LOV, open its Property Palette to define its properties. Some of these properties are the following:

- **X and Y Position:** Specify screen coordinates for the LOV window in the form coordinate units
- **Width and Height:** Define size of the LOV window in the current form coordinate units; can be adjusted by form operator
- **Column Mapping:** Click the button labeled “More…” to open the LOV Column Mapping window
- **Filter Before Display:** Determines whether the user should be prompted with a dialog box that enables them to enter a search value before the LOV is invoked; value will be used as additional restriction on first column in the query
- **Automatic Display:** Controls whether LOV should be invoked automatically when form operator enters an item to which the LOV is attached
- **Automatic Select:** Specifies if LOV should close and return values automatically when reduced to single entry
- **Automatic Skip:** Determines whether cursor skips to next navigable item when operator selects a value from LOV to populate text item

Setting LOV Properties

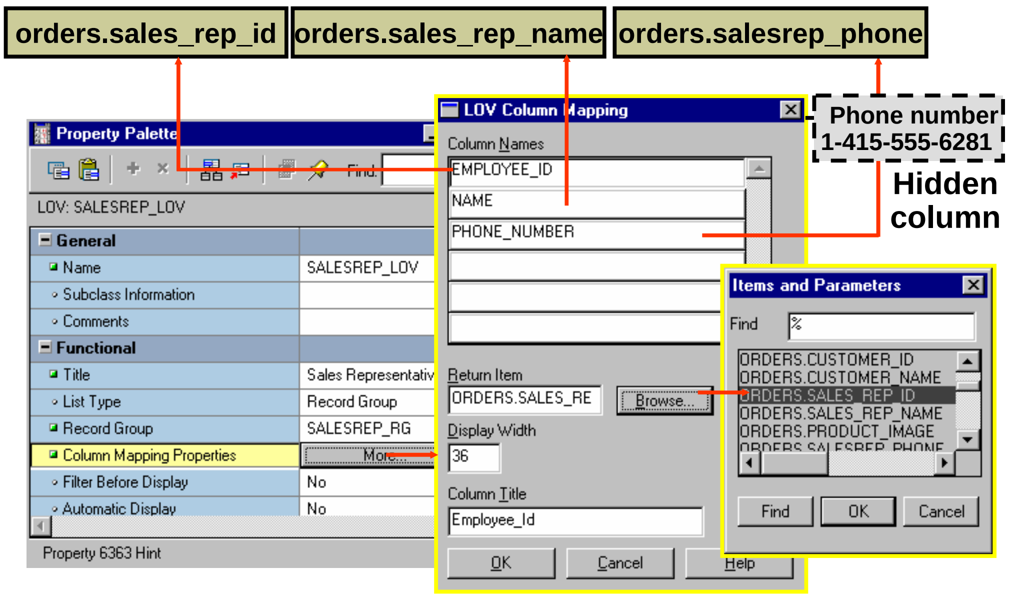


Setting LOV Properties (continued)

- **Automatic Refresh:** If Yes, record group reexecutes query every time LOV is invoked; if No, query fires only first time LOV is invoked in the session
- **Automatic Position:** Determines whether Forms automatically positions LOV near item from which it was invoked
- **Automatic Column Width:** Determines if Forms automatically sets the width of each column so entire title displays if the title is longer than the column display width

Note: More than one LOV can be based on the same record group. When this is the case and you set Automatic Refresh to No, Forms Builder will not reexecute the LOV query once any of the associated LOVs is invoked.

LOVs: Column Mapping



The Column Mapping Properties

When you click the More property control button for Column Mapping Properties, the LOV Column Mapping dialog box opens, with the following elements:

- **Column Names:** Lets you select an LOV column for mapping or defining a column
- **Return Item:** Specifies the name of the form item or variable to which Forms should assign the column value. If null, the column value is not returned from the LOV. If you want to return a value, specify one of the following:
  - `Block_name.item_name`
  - `GLOBAL.variable_name`
  - `PARAMETER.parameter_name`
- **Display Width:** Width of column display in LOV; value of zero causes column to be hidden, but value is available for return
- **Column Title:** Heading for column in LOV window

To set a column mapping in this dialog, first select the column from the Column Names list, then set the other mapping values, as required.

Note: The record group columns and LOV columns must remain compatible.

You can modify the record group query from its own properties list.

The Column Mapping Properties (continued)

Associating an LOV with a Text Item

So that the user can invoke an LOV from a text item, you must specify the LOV name in the Property Palette of the text item.

1. Select the text item in the Object Navigator from which the LOV is to be accessible.
2. In the item Property Palette, set the List of Values property to the required LOV.

Remember that the List of Values lamp is displayed when the user navigates to this text item, indicating that the LOV is available through the List of Values key or menu command.

Instructor Note

- Do not put a colon in front of the Return Item object names.
- To add extra spaces between columns in the LOV window, set the Display Width wider than the column default width. However, you cannot increase the width between a number column and a nonnumber column this way because LOVs display numbers right-justified.
- You can also attach an LOV to a text item programmatically with the `SET_ITEM_PROPERTY` built-in.

Demonstration

Demonstrate how mapping column values to global variables and changing the record group query at run time enable you to use the same LOV for more than one set of items.

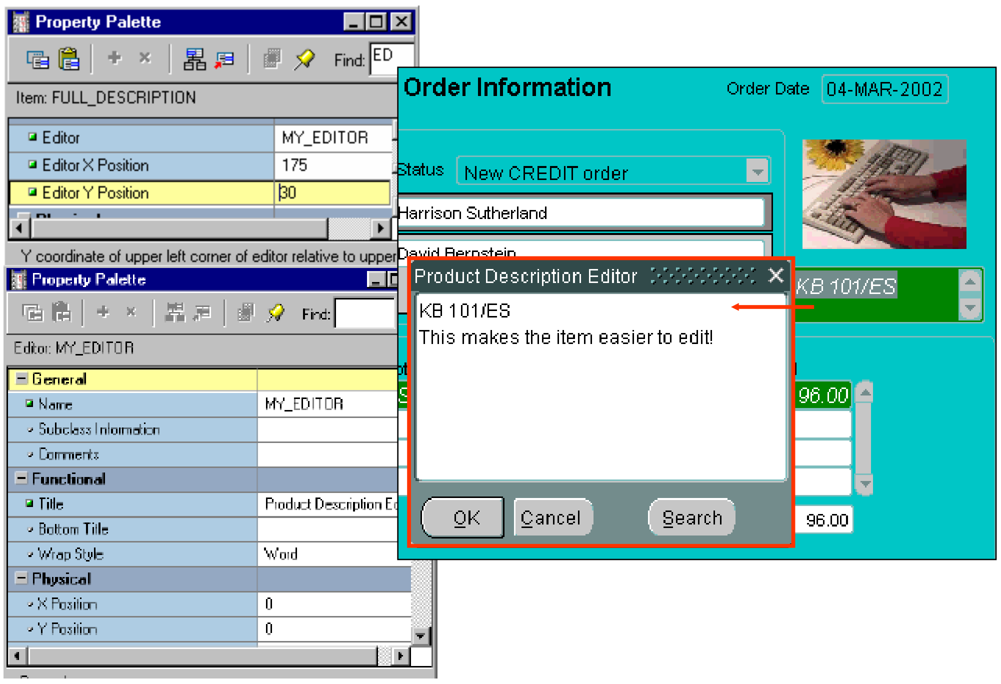
From the `demo` directory, open the form `LOV_Demo.fmb`. Show the students that the form uses one LOV and one record group. The LOV is assigned to both the Sales_Rep_Id and the Manager_Id items. Look at the column mapping property of the LOV to see that the ID and NAME returned by the query are assigned to global variables.

Now examine the code in the Key-Listval triggers for the Sales_Rep_Id and the Manager_Id items. Do not emphasize the code itself to students because they have not learned how to code triggers yet – just summarize what is being accomplished:

- The record group is repopulated with a different query for each item prior to displaying the LOV
- The global variables from the LOV are being assigned to the text items. So indirect mapping to multiple sets of text items can be accomplished by mapping LOV column values to global variables, and then assigning the global variables to the desired text items after invoking the LOV.

Run the form and invoke the LOV from both the Sales Rep ID and the Manager ID items. You can see that two different sets of records are displayed by the same LOV for the two different text items, and when you choose an entry, it is assigned to the correct set of text items (either Sales Rep ID and Sales Rep Name, or Manager ID and Manager Name).

Defining an Editor



Property Palette

Item: FULL_DESCRIPTION

Editor	MY_EDITOR
Editor X Position	175
Editor Y Position	30

Y coordinate of upper left corner of editor relative to upper

Property Palette

Editor: MY_EDITOR

General	
Name	MY_EDITOR
Subclass Information	
Comments	
Functional	
Title	Product Description E
Bottom Title	
Wrap Style	Word
Physical	
X Position	0
Y Position	0

Order Information Order Date 04-MAR-2002

Status New CREDIT order

Harrison Sutherland

David Bernstein

Product Description Editor

KB 101/ES
This makes the item easier to edit!

KB 101/ES

96.00

96.00

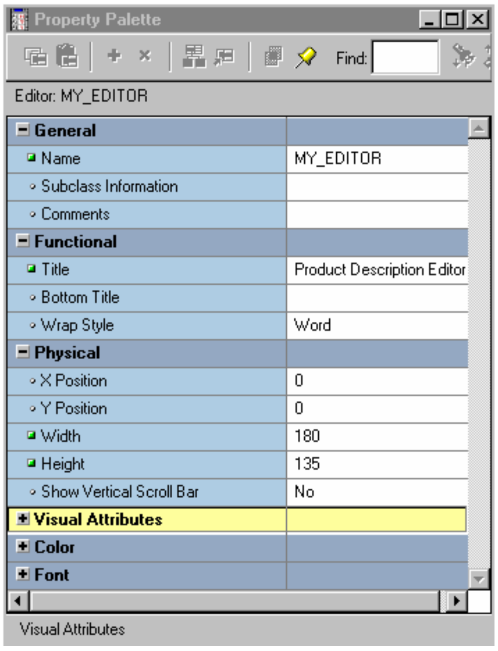
Defining an Editor

If the user needs to use an editor on text values, the default Forms Builder editor is usually sufficient for most items. However, you can design your own customized editor as an object in a form module, and then attach it to the text items that need it.

How to create a customized editor

1. Select the Editors node in the Object Navigator, then click Create. A new editor object is displayed in the list.
2. Select the new editor in the Object Navigator, and then access its Property Palette, where you can set its name and other properties.

Setting Editor Properties


Property Palette

Find:

Editor: MY_EDITOR

General	
Name	MY_EDITOR
Subclass Information	
Comments	
Functional	
Title	Product Description Editor
Bottom Title	
Wrap Style	Word
Physical	
X Position	0
Y Position	0
Width	180
Height	135
Show Vertical Scroll Bar	No
Visual Attributes	
Color	
Font	

Visual Attributes

ORACLE

Defining an Editor (continued)

Setting Editor properties

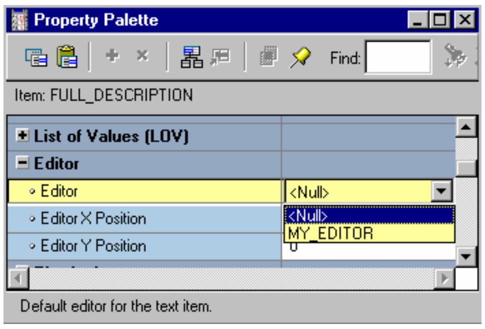
The following properties show the individual tailoring that is possible by creating your own editor:

- **Title/Bottom Title:** Displays at top or bottom of editor window
- **Width/Height:** Control size of editor window and hence its editing area
- **X/Y Position:** Screen position for editor; can be overridden by a text item property
- **Wrap Style:** How text wraps in the window: None, Character, or Word
- **Show Vertical Scrollbar:** Specify Yes to add vertical scroll bar to editor window

Instructor Note

Demonstration: Show how to create an editor.

Associating an Editor with a Text Item



Item: FULL_DESCRIPTION

<Null>

<Null>

MY_EDITOR

Default editor for the text item.

- Associate one of two types of editors with a text item.
- Set text item's Editor property to one of the following:
  - Null (default Forms Builder editor)
  - Editor name (customized editor)

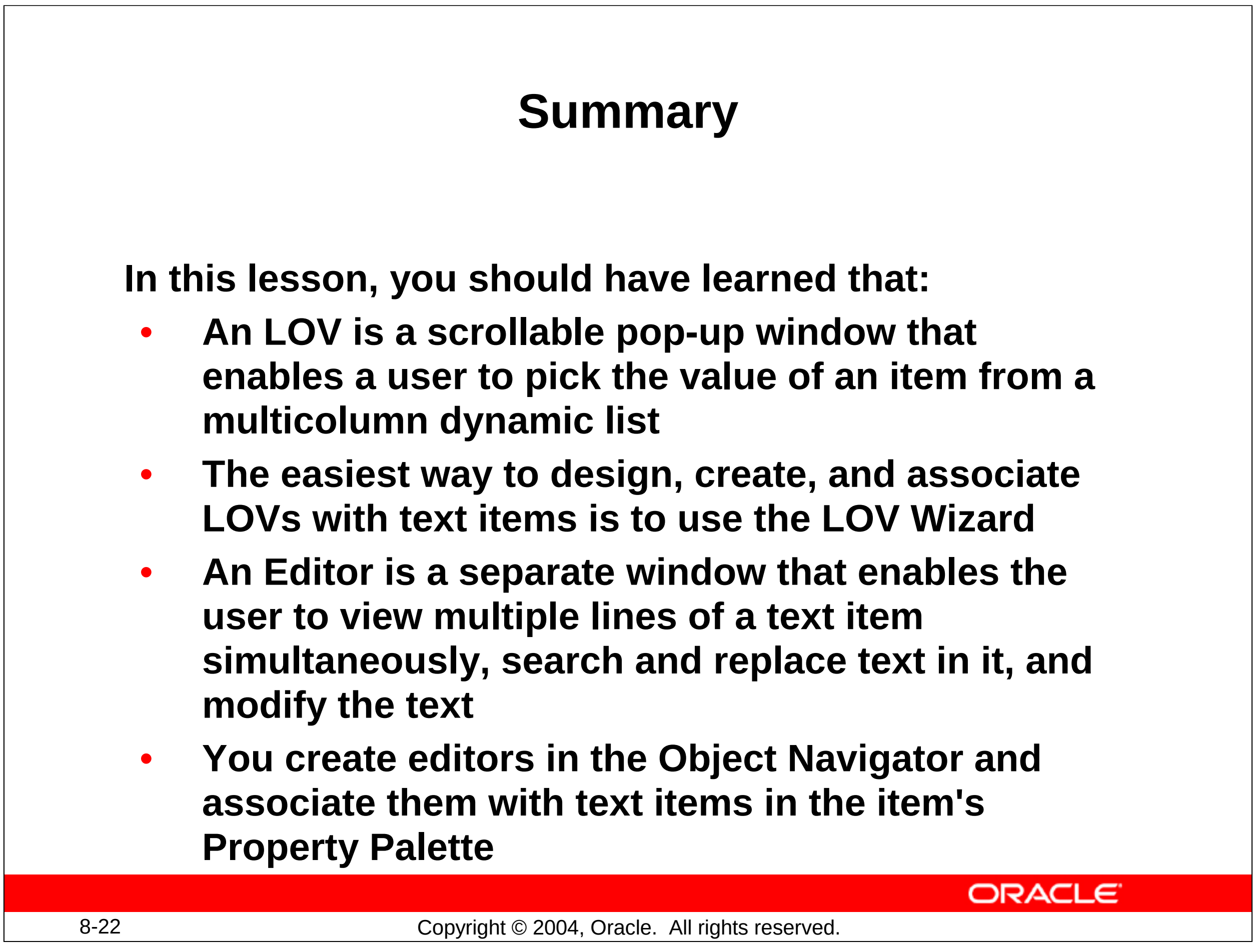
Associating an Editor with a Text Item

To associate an editor with a text item, you must specify the editor in the Property Palette of the text item.

Select the text item in the Object Navigator from which the editor is to be accessible.

In the item Property Palette, set the Editor property to one of the following settings:

- **Null:** The text item uses the default Forms Builder editor.
- **Editor Name:** The text item uses the named editor that you have created and customized in this module.



Summary

In this lesson, you should have learned that:

- An LOV is a scrollable pop-up window that enables a user to pick the value of an item from a multicolumn dynamic list
- The easiest way to design, create, and associate LOVs with text items is to use the LOV Wizard
- An Editor is a separate window that enables the user to view multiple lines of a text item simultaneously, search and replace text in it, and modify the text
- You create editors in the Object Navigator and associate them with text items in the item's Property Palette

8-22

Summary

In the lesson, you learned that lists of values (LOVs) and text editors can be used to support text items. Both LOVs and editors are objects in a form module that open their own window when activated at run time and are used to support text items in any block of the form module.

- LOVs and editors can be shared across text items.
- The steps to implement an LOV are:
  1. Create a new LOV (and record group).
  2. Define column mapping for return items.
  3. Attach the LOV to text items, as required.
- The LOV Wizard performs these steps automatically.
- Text items can use the default editor or a user-named editor.

Practice 8 Overview

This practice covers the following topics:

- Creating an LOV and attaching the LOV to a text item
- Creating an Editor and attaching it to a text item

Practice 8 Overview

In this practice session, you will create three LOVs and an editor.

- Using the LOV Wizard, create an LOV in the ORDERS form to display product numbers and their descriptions. Attach the LOV to the Product_ID item in the `ORDER_ITEMS` data block.
- Using the LOV Wizard, create an LOV in the ORDERS form to display Sales Representatives' IDs and names. Attach the LOV to the Sales_Rep_ID item in the `ORDERS` data block. Save and run the form.
- Using the LOV wizard, create an LOV in the CUSTOMERS form to display sales representatives' numbers and their names. Attach the LOV to the ACCT_MGR_ID item in the `CUSTOMERS` data block. Save and run the form.
- In the CUSTOMERS form, create an editor for the Phone_Numbers item.

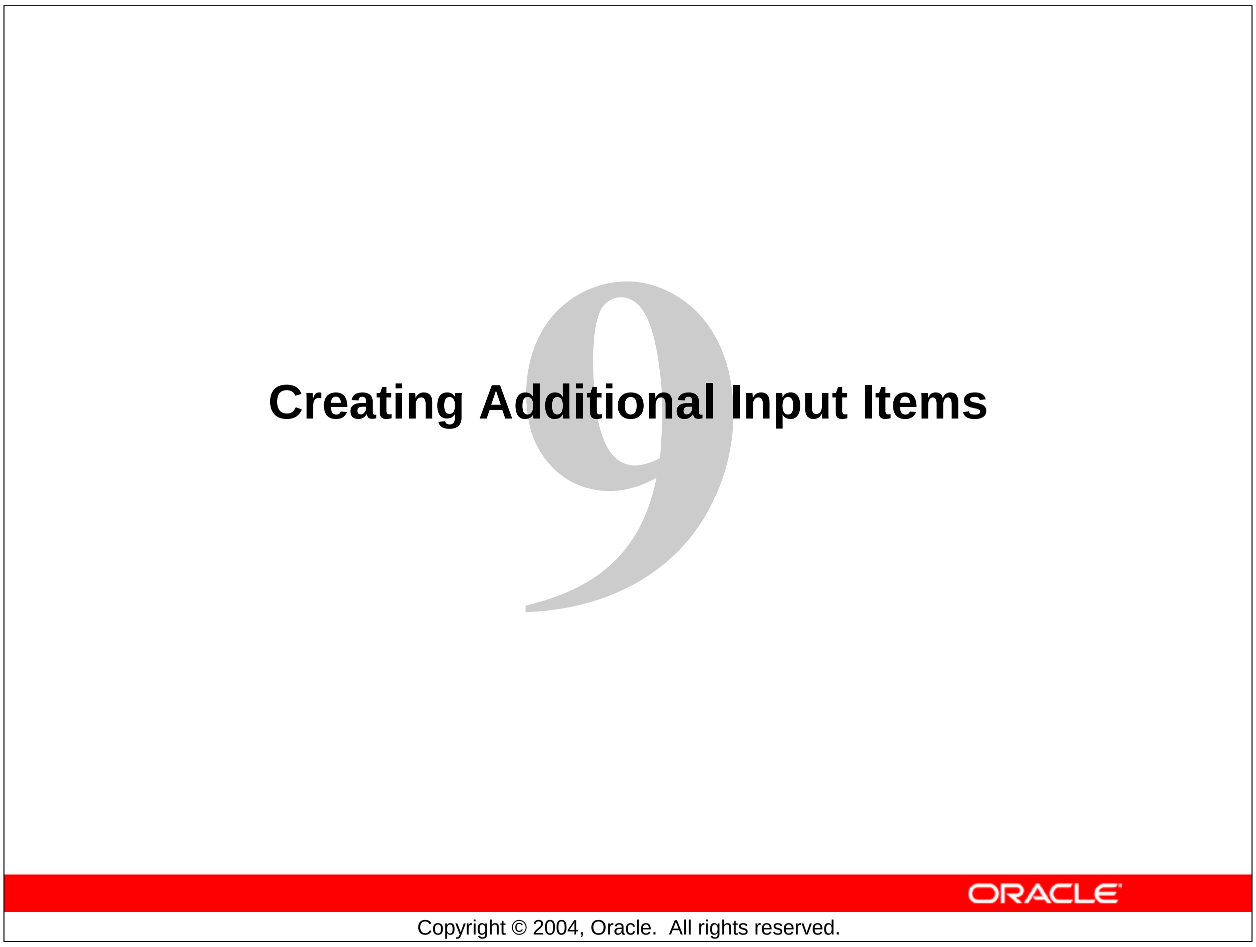
Note: For solutions to this practice, see Practice 8 in Appendix A, "Practice Solutions."

Practice 8

1. In the ORDG*XX* form, create an LOV to display product numbers and descriptions to be used with the Product_Id item in the `ORDER_ITEMS` block.
   Use the `PRODUCTS` table and select the Product_Id and Product_Name columns. Assign a title of Products to the LOV. Sort the list by the product name. Assign a column width of 25 for Product_Id, and assign the LOV a width of 200 and a height of 250. Position the LOV 30 pixels below and to the right of the upper-left corner. For the Product_Id column, set the return item to `ORDER_ITEMS.PRODUCT_ID`. Attach the LOV to the Product_Id item in the `ORDER_ITEMS` block. Change the name of the LOV to `PRODUCTS_LOV` and the name of the record group to `PRODUCTS_RG`.
2. In the ORDG*XX* form, use the LOV Wizard to create an LOV to display sales representatives' numbers and their names. Use the `EMPLOYEES` table, Employee_Id, First_Name, and Last_Name columns. Concatenate the First_Name and the Last_Name columns and give the alias of Name. Select employees whose Job_ID is SA_REP.
   Assign a title of Sales Representatives to the LOV. Assign a column width of 20 for ID, and assign the LOV a width of 200 and a height of 250. Position the LOV 30 pixels below and to the right of the upper-left corner. For the ID column, set the return item to `ORDERS.SALES_REP_ID`; for the Name column, set the return item to `ORDERS.SALES_REP_NAME`. Attach the LOV to the Sales_Rep_Id item in the ORDERS block.
   Change the name of the LOV to `SALES_REP_LOV` and the record group to `SALES_REP_RG`.
3. Save and compile your form.
   Click Run Form to run your form and test the changes.
4. In the CUSTG*XX* form, use the LOV Wizard to create an LOV to display account managers' numbers and their names. Use the `EMPLOYEES` table, Employee_Id, First_Name, and Last_Name columns. Concatenate the First_Name and the Last_Name columns and give the alias of Name. Select employees whose Job_ID is SA_MAN.
   Assign a title of Account Managers to the LOV. Assign a column width of 20 for ID, and assign the LOV a width of 200 and a height of 250. Position the LOV 30 pixels below and to the right of the upper-left corner. For the ID column, set the return item to `CUSTOMERS.ACCOUNT_MGR_ID`. Attach the LOV to the Account_Mgr_Id item in the CUSTOMERS block.
   Change the name of the LOV to `ACCOUNT_MGR_LOV` and the record group to `ACCOUNT_MGR_RG`.
5. In the CUSTG*XX* form, create an editor and attach it to the Phone_Numbers item. Set the title to Phone Numbers, the bottom title to Max 30 Characters, the background color to gray, and the foreground color to yellow.
6. Save, compile, and run the form to test the changes. Resize the window if necessary.

Instructor Note

If students try to include an object type column in the LOV query, they will receive an error. See Bug 2338467.



Creating Additional Input Items

Schedule:	**Timing**	**Topic**
	40 minutes	Lecture
	45 minutes	Guided Practice
	85 minutes	Total

Objectives

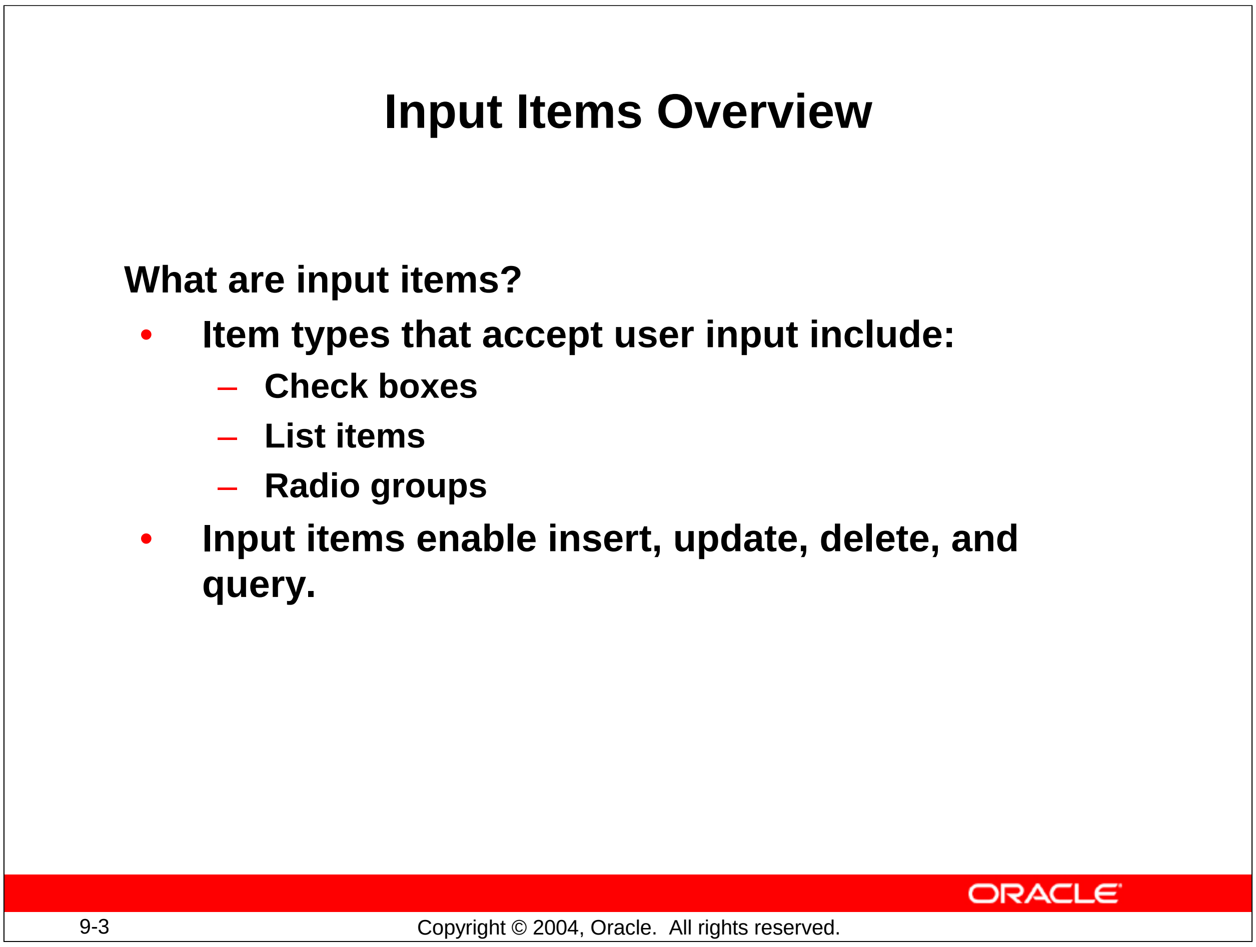
After completing this lesson, you should be able to do the following:

- **Identify the item types that allow input**
- **Create a check box**
- **Create a list item**
- **Create a radio group**

Introduction

Overview

In addition to text items, Oracle Forms Developer provides a variety of other item types. These can be divided into two groups: those that accept input and those that do not. This lesson covers input items and how they are used.



Input Items Overview

What are input items?

- **Item types that accept user input include:**
  - **Check boxes**
  - **List items**
  - **Radio groups**
- **Input items enable insert, update, delete, and query.**

ORACLE

9-3

What Are Input Items?

Input item is a generic term for Forms Builder item types that accept user input.

These item types include the following:

- Check box
- List item
- Radio group

What can you do with Input Items?

When you create input items, they already have some initial functionality. Through items you can interact with the database in the following ways:

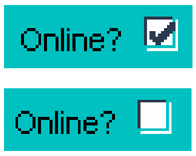
- Insert values
- Update existing values
- Delete existing values
- Query existing values

Note: You can add functionality to input items with triggers and PL/SQL program units.

Check Boxes Overview

What Are Check Boxes?

- **Two-state interface object:**
  - **Checked**
  - **Unchecked**
- **Not limited to two values**



What Is a Check Box?

A check box is a two-state interface object that indicates whether a certain value is ON or OFF. The display state of a check box is always either checked or unchecked.

You can use check boxes to enhance the user interface by converting existing items that have two possible states. Although a check box is limited to two states, it is not limited to just two values. You specify the value to represent Checked, the value to represent Unchecked, and how other values are processed.

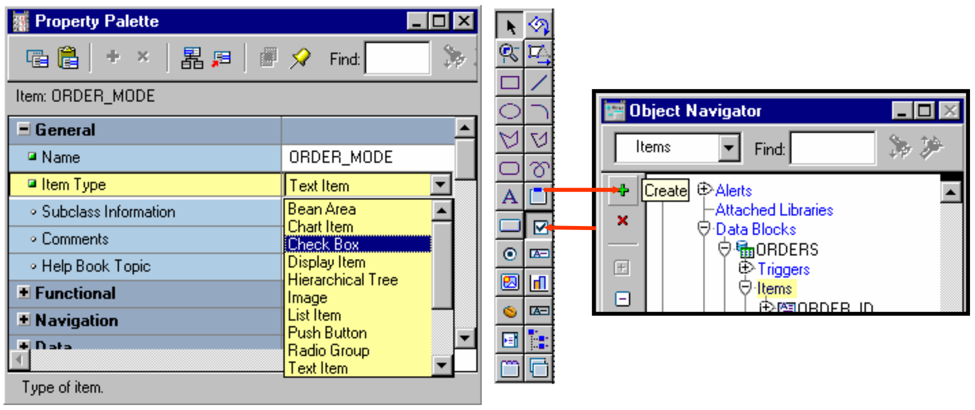
Using a Check Box at Run Time

You can do the following at run time:

- Set check box values either by user input, by means of the Initial Value property, or programmatically
- In Enter Query mode:
  - Query checked values by clicking one or more times until item is checked.
  - Query unchecked values by clicking one or more times until item is unchecked.
  - Ignore check box values in Enter Query mode by not selecting or deselecting the initial displayed value.

Creating a Check Box

- Convert an existing item.
- Use the Check Box tool in the Layout Editor.
- Use the Create icon in the Object Navigator.



ORACLE

9-5 Copyright © 2004, Oracle. All rights reserved.

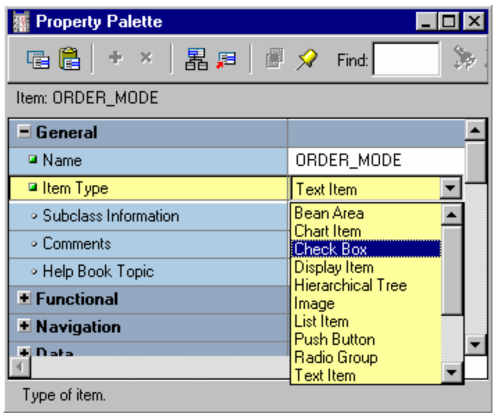
Creating a Check Box

A check box can be created by:

1. Converting an existing item
2. Using the Check Box tool in the Layout Editor
3. Using the Create icon in the Object Navigator (creates a text item which you can convert to a check box)

Converting an Existing Item into a Check Box

Convert text item to check box



Name	ORDER_MODE
Item Type	Text Item

How to Convert an Existing Item into a Check Box

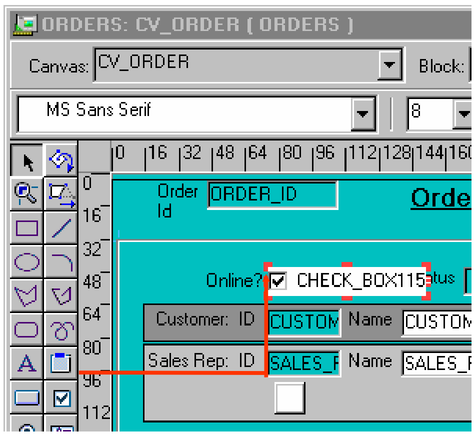
You can convert an existing item into a check box by changing the Item Type property to Check Box in the Property Palette and setting other relevant properties.

1. Invoke the Property Palette for the item that you want to convert.
2. Set the Item Type property to Check Box.
3. Enter a check box label.
4. Enter values for the checked and the unchecked states.
5. Set the Check Box Mapping of Other Values property.
6. Enter an initial value for the check box item.

Note: The check box label that you specify is displayed to the right of the check box element at run time. If the complete label name is not displayed, adjust it in the Layout Editor. If the item already has a prompt, delete it in the item Property Palette.

Creating a Check Box in the Layout Editor

Use check box tool in Layout Editor



How to Create a Check Box in the Layout Editor

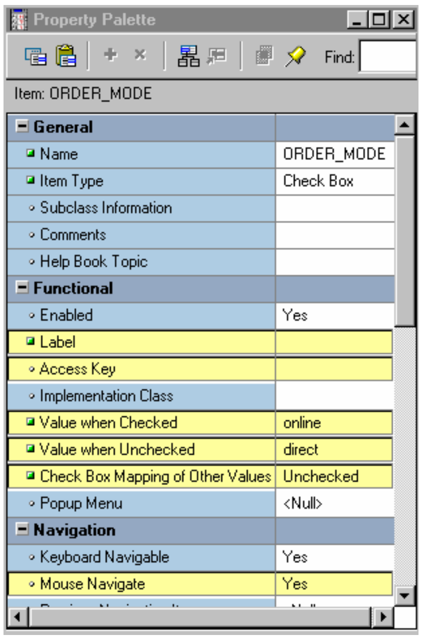
You can also create a check box by using the Check Box tool in the Layout Editor.

1. Invoke the Layout Editor.
2. Set the canvas and block to those on which you want the check box item to be displayed.
3. Click the Check Box tool.
4. Click the canvas in the position where you want the check box to be displayed.
5. Double-click the check box to invoke its Property Palette.
6. Set the properties as required.

Instructor Note

Demonstration: Using the `ordwk07.fmb` file, convert the Order_Mode item to a check box item. Point out that the object icon for this check box item changes automatically in the Object Navigator.

Setting Check Box Properties



Property Palette

Find:

Item: ORDER_MODE

General	
Name	ORDER_MODE
Item Type	Check Box
Subclass Information	
Comments	
Help Book Topic	
Functional	
Enabled	Yes
Label	
Access Key	
Implementation Class	
Value when Checked	online
Value when Unchecked	direct
Check Box Mapping of Other Values	Unchecked
Popup Menu	<Null>
Navigation	
Keyboard Navigable	Yes
Mouse Navigate	Yes

- **Data Type**
- **Label**
- **Access Key**
- **Value When Checked**
- **Value When Unchecked**
- **Check Box Mapping of Other Values**
- **Mouse Navigate**

Setting Check Box Properties

The following properties may be set to affect the appearance and behavior of check boxes:

- **Data Type:** Must be compatible with values specified in the Value properties
- **Label:** Text label displayed next to check box item (independent of check box value)
- **Access Key:** Which combination of keys may be used to navigate to this item and check or uncheck it
- **Initial Value:** Initial value of the item for new record, determining whether check box is initially checked or unchecked
- **Value When Checked:** Value to represent check state of the check box
- **Value When Unchecked:** Value to represent unchecked state of the check box
- **Check Box Mapping of Other Values:** How other values are to be processed (NOT ALLOWED, CHECKED, or UNCHECKED)

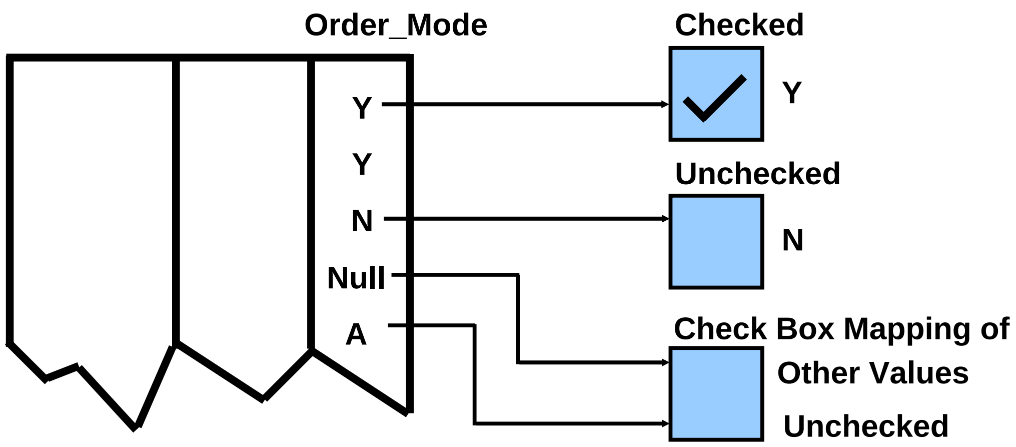
Setting Check Box Properties (continued)

- **Mouse Navigate:** Whether Forms navigates to and moves input focus to the item when the user activates the item with the mouse (default is Yes)

Instructor Note

The Mouse Navigate property is valid only for buttons, check boxes, list items, and radio group items. When Mouse Navigate is set to Yes, Forms navigates to the item, firing any appropriate navigation and validation triggers on the way.

Check Box Mapping of Other Values



Dealing with Other Values

If your base table column accepts other values, then your check box should account for them. You can assign other values to either the checked or unchecked states by using the Check Box Mapping of Other Values property. Alternatively, you can choose not to accept other values with the Not Allowed setting.

Note: If you choose not to accept other values and they exist in the base table column, Forms ignores the entire record during query processing.

Dealing with Null Values

If your base table column accepts null values, you can account for them by one of the following methods:

- Set the Check Box Mapping of Other Values property.
- Set the checked or unchecked state to represent null (leave the value blank).

Instructor Note

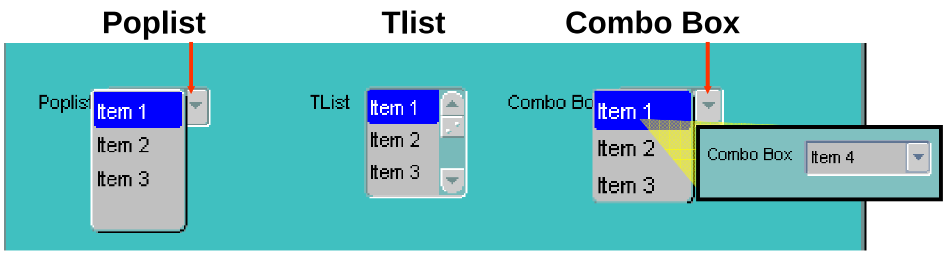
You must specify a valid initial value, except under either of the following conditions:

- The Check Box Mapping of Other Values property is set to Checked or Unchecked.
- The value associated with Checked or Unchecked is NULL.

List Items Overview

What Are List Items?

- Set of mutually exclusive choices, each representing a different value
- Three list styles available:



- Space-saving alternative to a radio group
- Smaller-scale alternative to an LOV

What Are List Items?

A *list item* is an interface object that displays a predefined set of choices, each corresponding to a specific data value. You use the list item at run time to select a single value. List choices or elements are mutually exclusive; one and only one can be selected at a time.

The Three List Item Styles

- **Poplist:** Appears as a field with an iconic button attached to the right side (When you click a poplist, all its list elements are displayed.)
- **Tlist:** Appears as a rectangular box that displays the list elements (When the display area is not big enough to display all the list elements, a scroll bar is automatically attached to the right side to view the remaining list elements.)
- **Combo box:** Appears as a field with a downarrow next to its right side (Use the button to display all the combo box list elements. The combo box accepts user input.)

Instructor Note

Demonstration: Open the form `LIST_ITEMS.fmb` and run it to show the different run time behavior of the three types of list items.

What Are List Items? (continued)

Note: The poplist and combo box take up less space, but end users must open them to see the list elements. A Tlist remains “open,” and end users can see multiple values at a time. Use the attached scroll bar to see more values if the Tlist is not big enough to display all the list elements.

Uses and Benefits of List Items

- Enable display of a defined set of choices
- Display a set of choices without using a vast area of canvas
- Provide an alternative to radio groups
- Provide a Windows-style list of values

Setting the Value for a List Item

The value for a list item can be set in any of the following ways:

- User selection
- User input (combo box style only)
- A default value
- Programmatic control

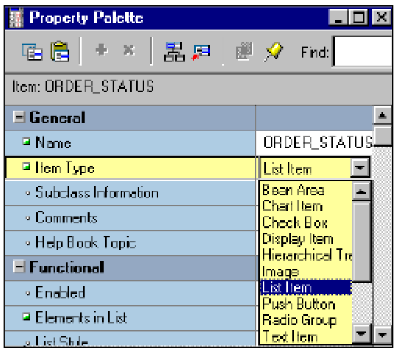
Instructor Note

What are some of the differences between a list item and an LOV?
List items:

- Are generally used for a small number of elements
- Do not have a Find button
- Cannot be attached to other items
- Have choices that are not based on a `SELECT` statement (although they can be if the list elements are created programmatically)

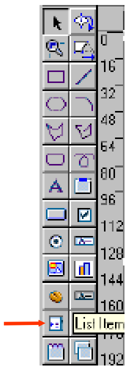
Creating a List Item

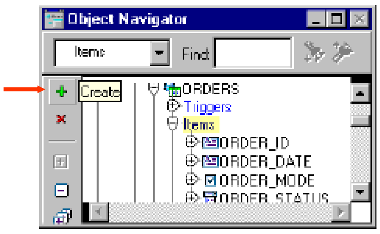
- Convert an existing item.
- Use the List Item tool in the Layout Editor.
- Use the Create icon in the Object Navigator.



Item: ORDER_STATUS

Name	ORDER_STATUS
Item Type	List Item



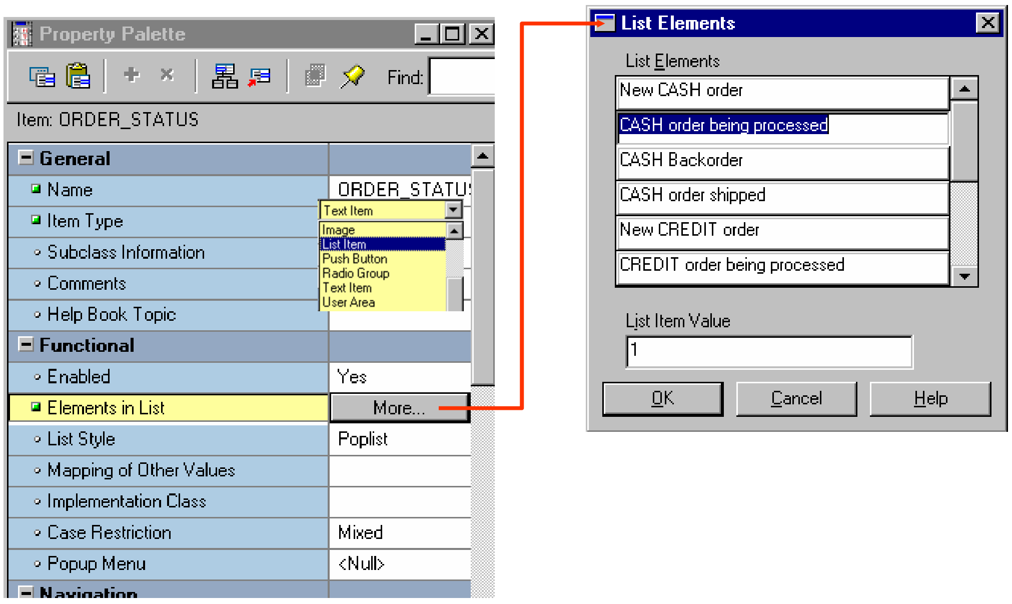


Creating a List Item

A list item can be created by:

- Converting an existing item
- Using the List Item tool in the Layout Editor
- Using the Create icon in the Object Navigator (this creates a text item that you can convert to a list item)

Converting an Existing Item into a List Item



Property Palette

Item: ORDER_STATUS

General	
Name	ORDER_STATU
Item Type	Text Item
Subclass Information	
Comments	
Help Book Topic	
Functional	
Enabled	Yes
Elements in List	More...
List Style	Poplist
Mapping of Other Values	
Implementation Class	
Case Restriction	Mixed
Popup Menu	<Null>
Navigation	

List Elements

List Elements:
- New CASH order
- CASH order being processed
- CASH Backorder
- CASH order shipped
- New CREDIT order
- CREDIT order being processed

List Item Value: 1

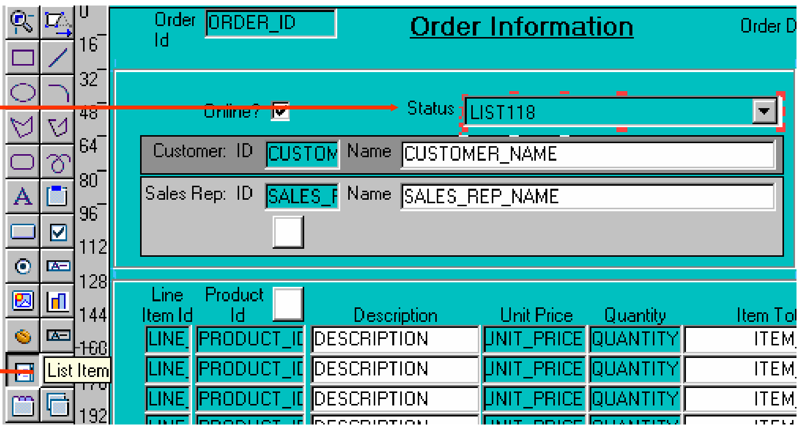
How to Convert an Existing Item into a List Item

You can convert an existing item into a list item by changing its Item Type property to List Item and setting the relevant properties.

1. Invoke the Property Palette for the item that you want to convert.
2. Set the Item Type property to List Item.
3. Select the Elements in List property.
4. Click More.
   The List Item Elements dialog box appears.
5. Enter the element that you want to appear in your list item in the List Elements column.
6. Enter the value for the currently selected list element in the List Item Value field.
7. Create additional list elements and values by repeating steps 5 and 6.
8. Click OK to accept and close the List Item Elements dialog box.
9. Set the Other Values property to do one of the following:
   - Reject values other than those predefined as list values
   - Accept and default all other values to one of the predefined list element values
10. Enter an initial value for the list item.

Creating a List Item in the Layout Editor

Use list item tool in Layout Editor



How to Create a List Item in the Layout Editor

You can also create a list item by using the List Item tool in the Layout Editor.

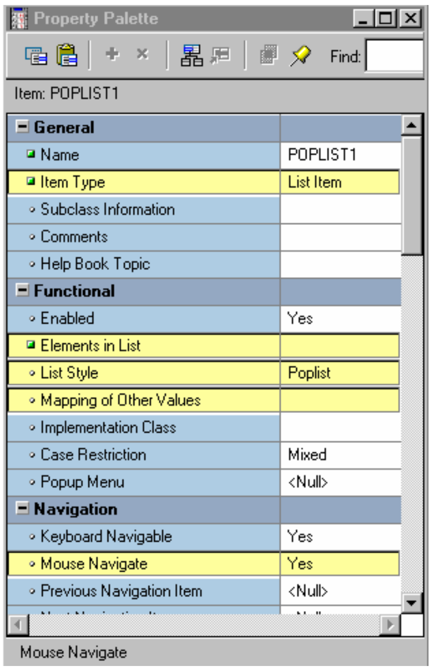
1. Invoke the Layout Editor.
2. Set the canvas and block to those on which you want the list item to be displayed.
3. Select the List Item tool.
4. Click the canvas in the position where you want the list item to be displayed.
5. Double-click the list item to invoke its Property Palette.
6. Set the properties as required.

Technical Note

To obtain a list of available functions when defining list elements, select `[Ctrl] + k` while the input focus is in the List Elements window. The Keys window may pop up behind the List Elements window; if so, just move the List Elements window so that you can see the Keys window.

Setting List Item Properties

- **Elements in List:**
  - **List elements**
  - **List item value**
- **List Style**
- **Mapping of Other Values**
- **Mouse Navigate**



Item: POPLIST1

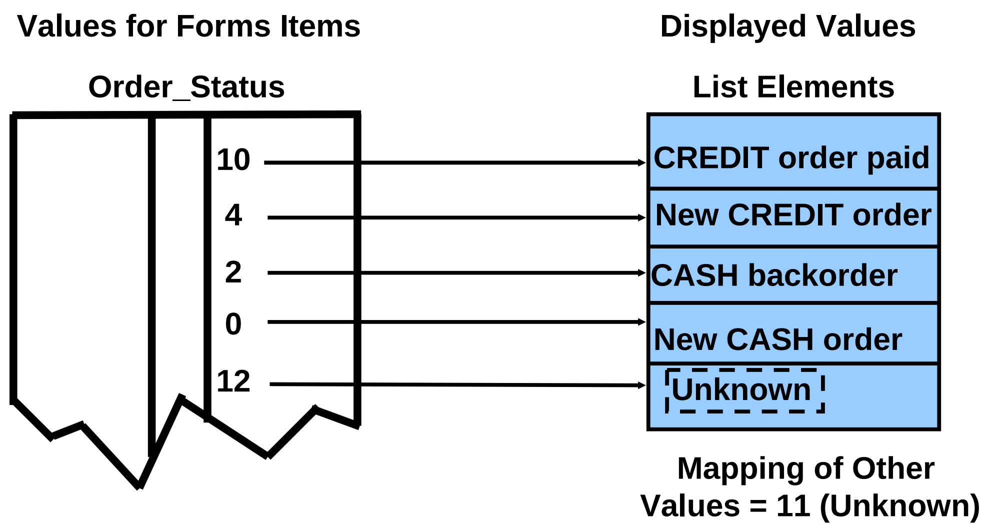
General	
Name	POPLIST1
Item Type	List Item
Subclass Information	
Comments	
Help Book Topic	
Functional	
Enabled	Yes
Elements in List	
List Style	Poplist
Mapping of Other Values	
Implementation Class	
Case Restriction	Mixed
Popup Menu	<Null>
Navigation	
Keyboard Navigable	Yes
Mouse Navigate	Yes
Previous Navigation Item	<Null>

Mouse Navigate

Setting List Item Properties

- **Elements in List:** Clicking More opens List Item Elements dialog window, where you specify:
  - List Elements: List elements that display at run time
  - List Item Value: Actual value that corresponds to the list element
- **List Style:** Display style of list item (Poplist, Tlist, or Combo Box)
- **Mapping of Other Values:** How other values are processed
- **Mouse Navigate:** Whether Forms navigates to the item and moves input focus to it when the user activates the item with a mouse

List Item Mapping of Other Values



NULL Values in a List Item

If the base table column for a list item accepts NULL values, Forms Builder creates a pseudochoice in the list to represent the null.

All three list styles display a blank field if a query returns a NULL value. If the Data Required property is set to No:

- A poplist displays a blank element for a NULL value.
- The user can omit a selection for a TList or can press [Clear Field] to deselect all list elements. This sets the list item to NULL.
- A combo box does not display a blank element. The end user must delete the default value if the default value is not NULL.

Handling Other Values in a List Item

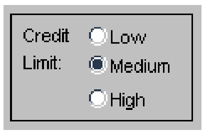
If the base table column for a list item accepts values other than those associated with your list elements, you must specify how you want to handle the values. Do this in one of the following ways:

- Ignore other values by leaving the Mapping of Other Values property blank.
- Associate the other values with one of the existing list elements (by naming either the list element or its associated value) in the Mapping of Other Values property.

Radio Groups Overview

What are radio groups?

- Set of mutually exclusive radio buttons, each representing a value
- Use:
  - To display two or more static choices
  - As an alternative to a list item
  - As an alternative to a check box



What Are Radio Groups?

A *radio group* is an item where a set of radio buttons represents the possible values for the item. These values and hence their corresponding radio buttons are mutually exclusive.

Uses and Benefits of Radio Groups

- Provide a choice between two or more static values
- Provide an alternative to list items with two or three choices
- Provide a choice between two alternatives, where choice is not On/Off or Yes/No; for example, Landscape or Portrait print format

Note: Consider list items instead of radio groups if there are more than four or five choices.

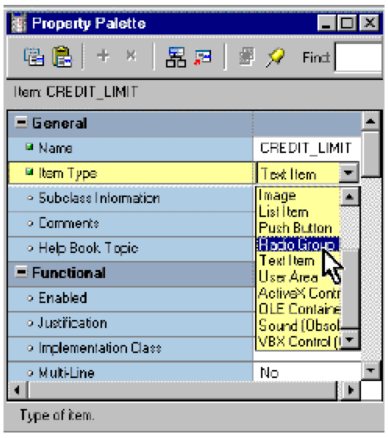
Using a Radio Group at Run Time

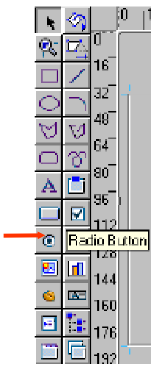
You can do the following at run time:

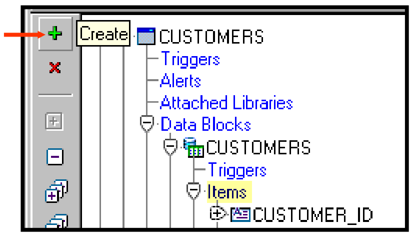
- Set radio group values:
  - By user input
  - By means of the Initial Value property
  - Programmatically
- Query individual radio button values

Creating a Radio Group

- Convert an existing item.
- Create a new radio button in the Layout Editor.
- Use the Create icon in the Object Navigator.





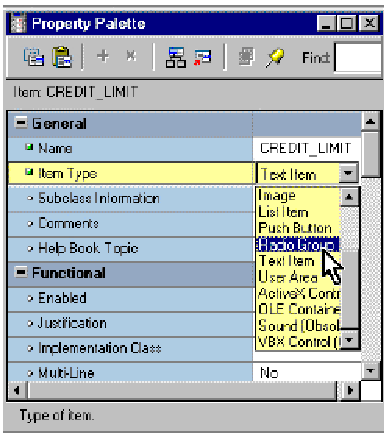


Creating a Radio Group

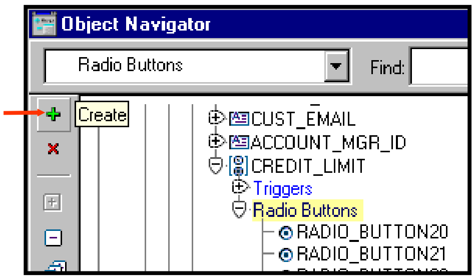
A radio group can be created by:

- Converting an existing item to a radio group
- Creating a new radio button in the Layout Editor (automatically creates a radio group if none exists)
- Using the Create icon in the Object Navigator (this creates a text item that you can convert to a radio group)

Converting Existing Item to Radio Group



Change Item Type and set other properties



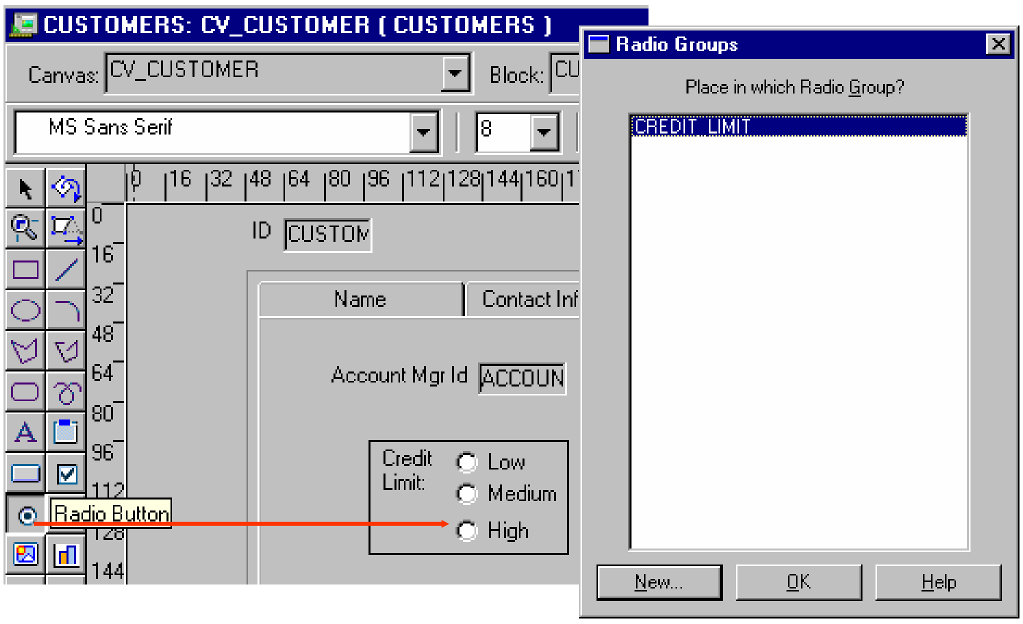
Create radio buttons for the radio group

How to Convert an Existing Item into a Radio Group

You can convert an existing item to a radio group by changing the item type and setting the properties for a radio group.

1. Invoke the Property Palette for the item that you want to convert.
2. Set the Item Type property to Radio Group.
3. Set the Canvas property to the Canvas on which you want the radio buttons to appear.
4. Set the Mapping of Other Values property to specify how the Radio Group should handle any other values.
5. Set the Initial Value property, as required. This should be the name of a radio button.
6. Expand the item node in the Object Navigator.
   The Radio Buttons node appears.
7. Select the Radio Buttons node and click the Create icon.
   A radio button displays in the Object Navigator and the Property Palette takes on its context.
8. Enter a name, value, and a label for the radio button.
9. Specify the display properties of the radio button.
10. Create additional radio buttons by repeating steps 6 through 8.

Creating Radio Group in Layout Editor



How to Create a Radio Group in the Layout Editor

You can also create a radio group by using the Radio Button tool in the Layout Editor.

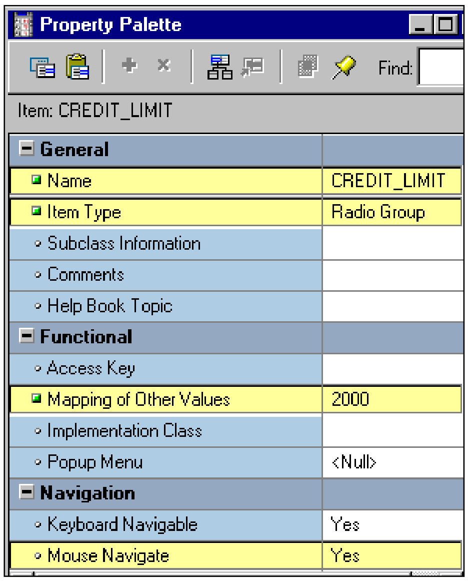
1. Invoke the Layout Editor.
2. Set the canvas and block to those on which you want the radio group to be displayed.
3. Select the Radio Button tool.
4. Position the cursor at the desired location and click.
   If you already have a radio group in the current block, the Radio Groups dialog box appears and you must decide whether the new radio button should appear in the existing group or a new one. If you select New, the new radio group is created implicitly.
5. Double-click the radio button to invoke the Property Palette.
6. Set the radio button properties as required.

Instructor Note

The canvas property for a radio group is set in the Property Palette of the radio group. The individual radio buttons do not have a canvas property.

Setting Radio Properties

Radio group:

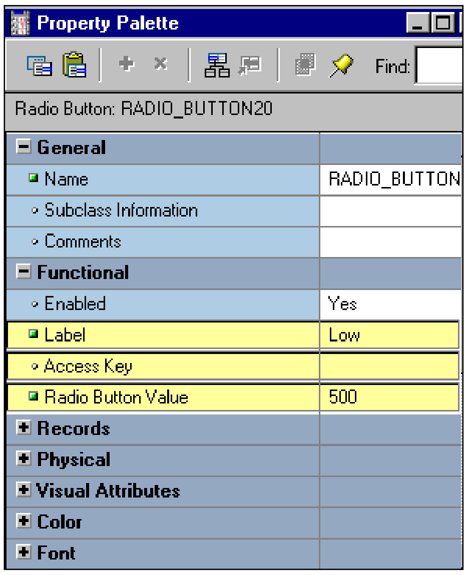


Property Palette

Item: CREDIT_LIMIT

General	
Name	CREDIT_LIMIT
Item Type	Radio Group
Subclass Information	
Comments	
Help Book Topic	
Functional	
Access Key	
Mapping of Other Values	2000
Implementation Class	
Popup Menu	<Null>
Navigation	
Keyboard Navigable	Yes
Mouse Navigate	Yes

Radio button:



Property Palette

Radio Button: RADIO_BUTTON20

General	
Name	RADIO_BUTTON
Subclass Information	
Comments	
Functional	
Enabled	Yes
Label	Low
Access Key	
Radio Button Value	500
Records	
Physical	
Visual Attributes	
Color	
Font	

Setting Item Properties for Radio Group Items and Radio Buttons

You should set the following properties for radio groups:

- **Data Type:** Must be compatible with Mapping of Other Values for the Radio Group, and Radio Button Value for each Radio Button in the group
- **Mapping of Other Values:** How values other than those specified are processed
- **Mouse Navigate:** Whether Forms navigates to the item when the operator activates the item with the mouse

You should set the following properties for radio buttons:

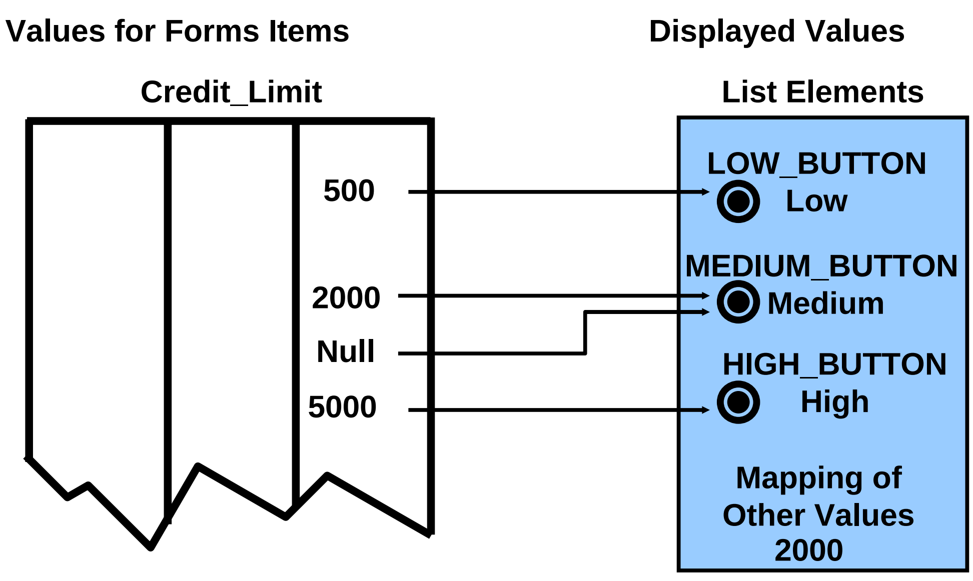
- **Label:** Text that appears adjacent to the radio button (independent of the button value)
- **Access Key:** Which combination of keys can be used to navigate to and manipulate this radio button
- **Radio Button Value:** The value for this item if this radio button is selected

Instructor Note

You must specify a valid initial value, except under either of the following conditions:

- The radio group accepts other values.
- The value associated with one of the radio buttons is NULL.

Radio Group Mapping of Other Values



Handling Other Values in a Radio Group

If the base table column for a radio group accepts values other than those associated with your radio buttons, you must use one of the following methods to specify how you want to handle the values:

- Ignore other values (by leaving the radio group's Mapping of Other Values property blank)
- Associate the other values with one of the existing radio buttons (by naming the associated value of the button in the Mapping of Other Values property)

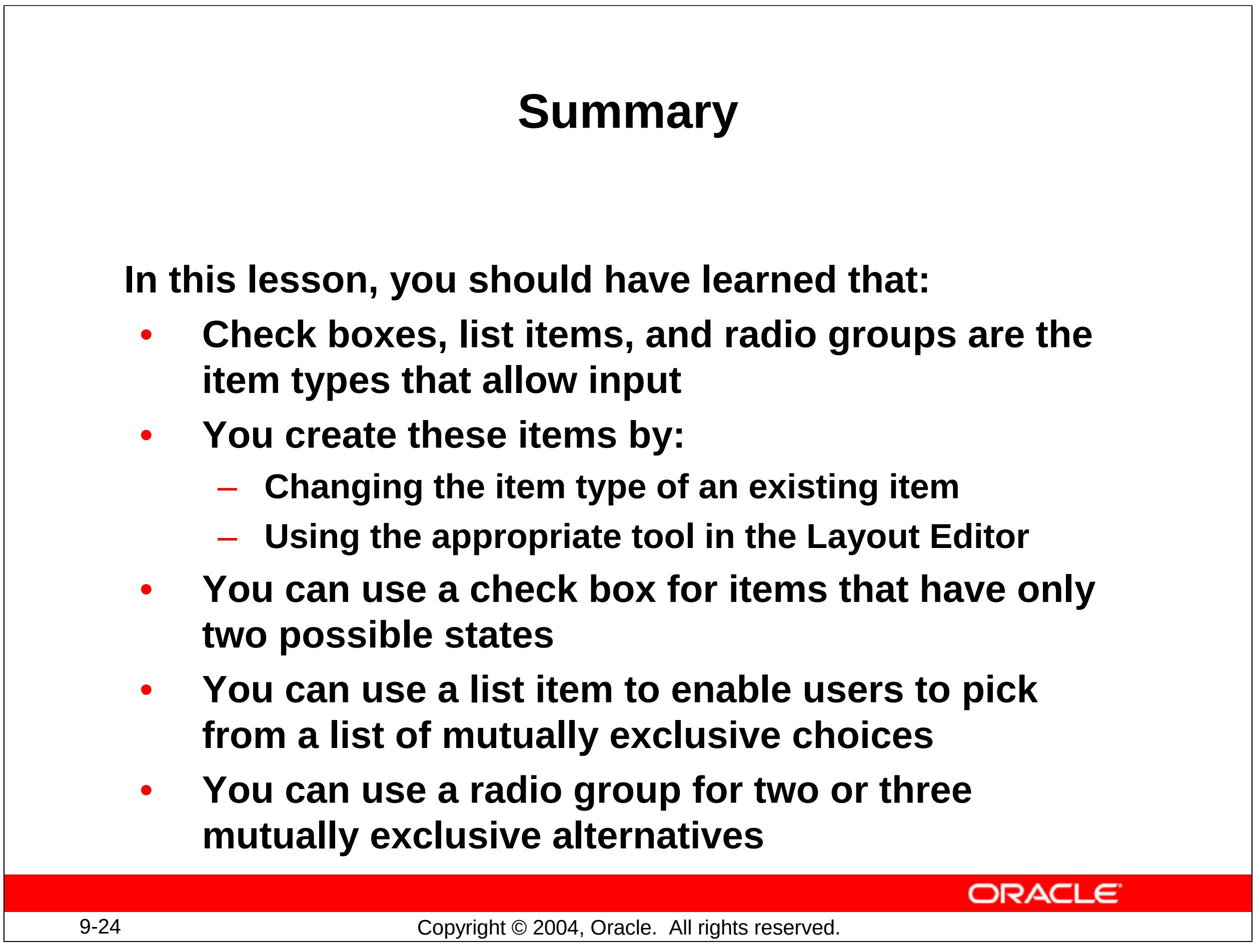
Note: Ignoring other values results in the entire row being ignored during query processing.

NULL Values in a Radio Group

A radio group can treat NULL as a valid value. You should account for the NULL case, if your base table column allows them. Do this in one of the following ways:

- Use the Mapping of Other Values property to implicitly force NULL to a radio button.
- Assign the NULL to its own radio button.

Note: To assign a NULL value, leave the Radio Button Value property blank.



Summary

In this lesson, you should have learned that:

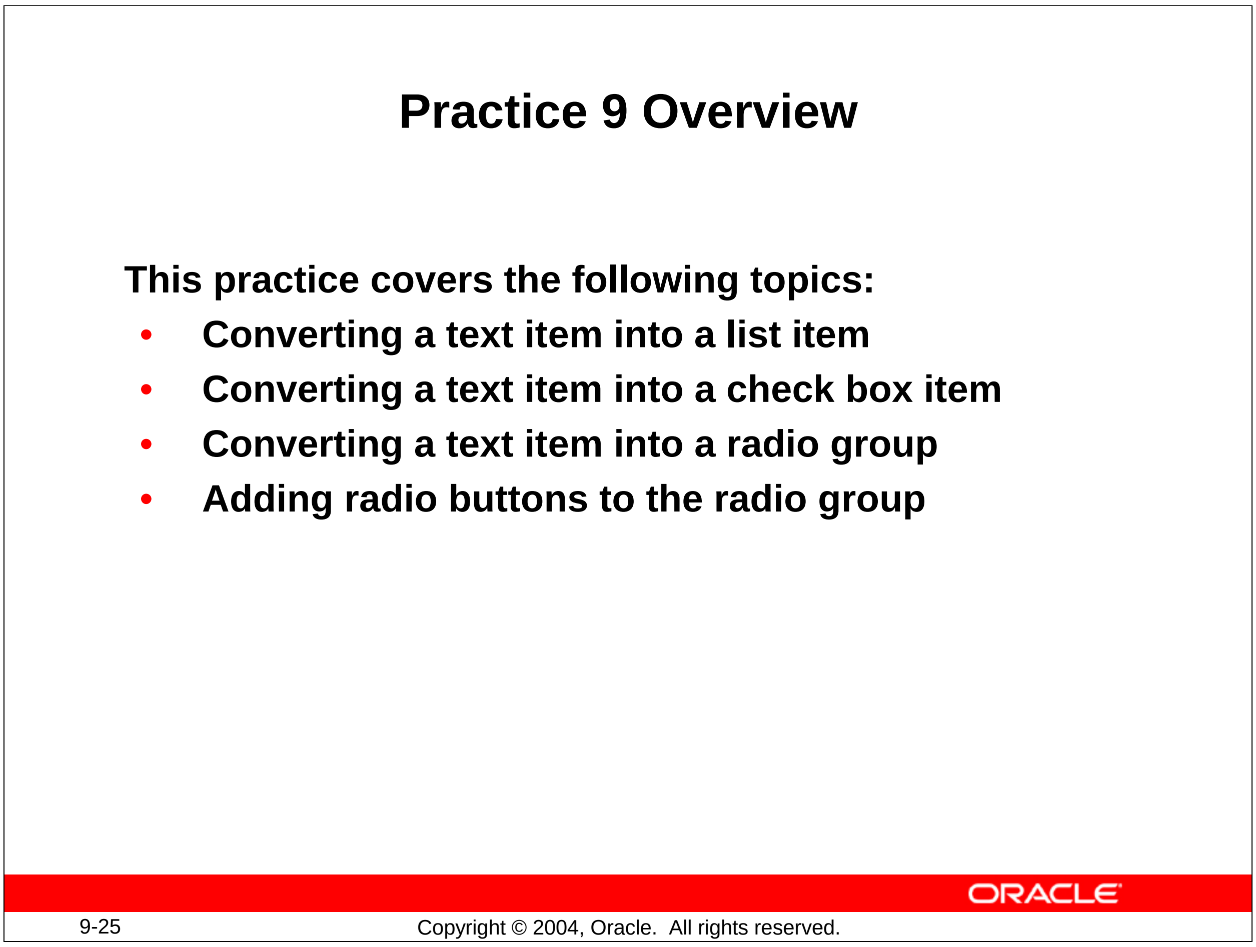
- Check boxes, list items, and radio groups are the item types that allow input
- You create these items by:
  - Changing the item type of an existing item
  - Using the appropriate tool in the Layout Editor
- You can use a check box for items that have only two possible states
- You can use a list item to enable users to pick from a list of mutually exclusive choices
- You can use a radio group for two or three mutually exclusive alternatives

9-24

Summary

In this lesson, you learned how to create items that accept direct user input. Use these items to enhance the user interface:

- **Check boxes:** To convert items that have two possible states
- **List items (Poplists, Tlists, and Combo boxes):** To convert items that have mutually exclusive choices
- **Radio groups:** To convert items with two or three alternatives that are mutually exclusive



Practice 9 Overview

This practice covers the following topics:

- Converting a text item into a list item
- Converting a text item into a check box item
- Converting a text item into a radio group
- Adding radio buttons to the radio group

9-25

Practice 9 Overview

In this practice session, you will convert existing text items into other input item types. You will create a list item, a check box, and a radio group.

- In the `ORDERS` form, convert the Order_Status item into a list item. Save and run the form.
- In the `ORDERS` form, convert the Order_Mode item into a check box item.
- In the `CUSTOMERS` form, convert the Credit_Limit item into a radio group. Add three radio buttons in the radio group. Save and run the form.

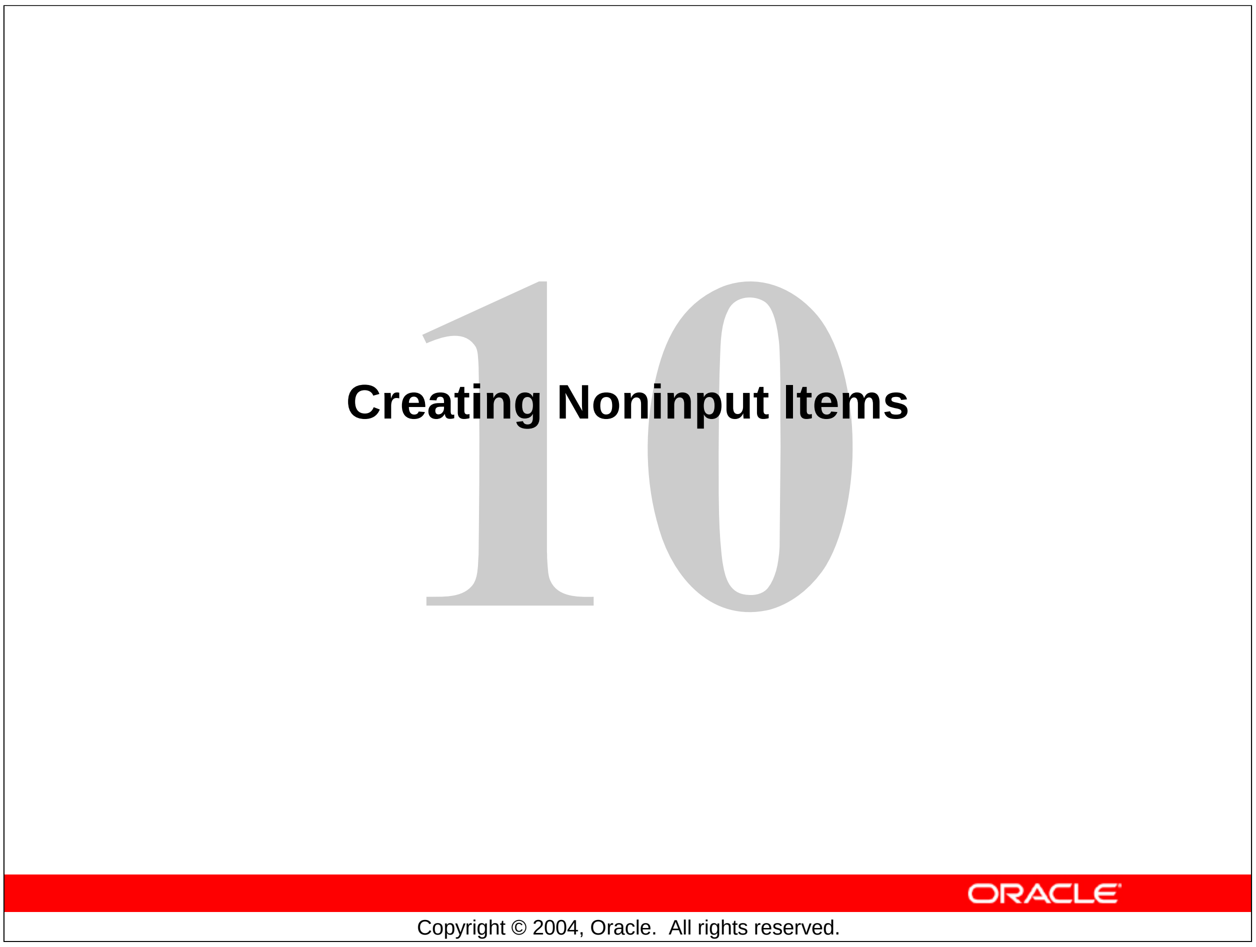
Note: For solutions to this practice, see Practice 9 in Appendix A, “Practice Solutions.”

Practice 9

1. In the ORDG*XX* form, convert the Order_Status item into a pop-up list item.
   Add list elements shown in the table below.
   Display any other values as Unknown.
   Ensure that new records display the initial value New CASH order.
   Resize the poplist item in the Layout Editor, so that the elements do not truncate at run time.

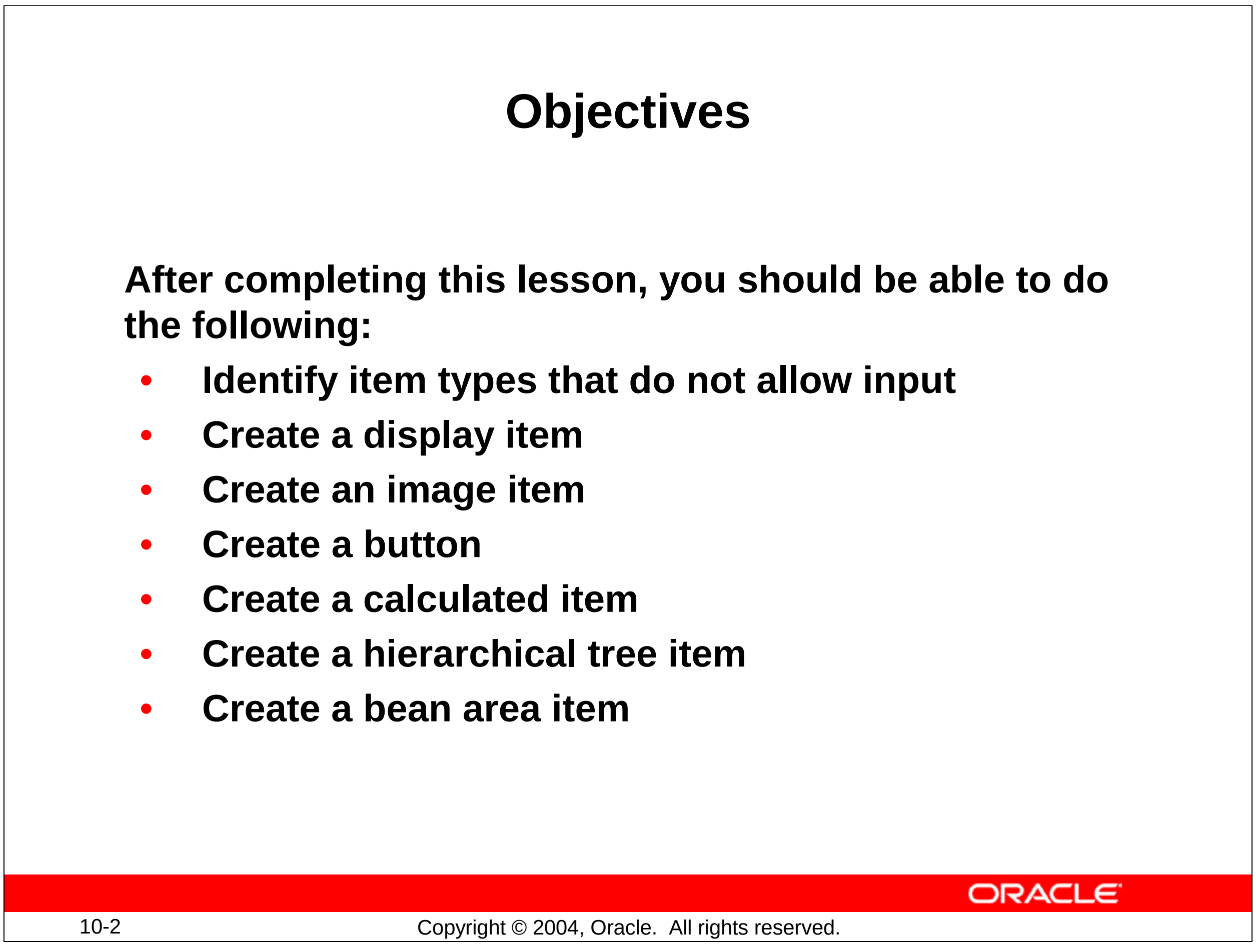
List Element	**List Item Value**
New CASH order	0
CASH order being processed	1
CASH Backorder	2
CASH order shipped	3
New CREDIT order	4
CREDIT order being processed	5
CREDIT Backorder	6
CREDIT order shipped	7
CREDIT order billed	8
CREDIT order past due	9
CREDIT order paid	10
Unknown	11

2. In the ORDG*XX* form, convert the Order_Mode text item to a check box.
   Set the checked state to represent the base table value of `online` and the unchecked state to represent `direct`.
   Ensure that new records are automatically assigned the value `online`.
   Display any other values as unchecked.
   Remove the existing prompt and set label to: Online?
   In the Layout Editor, resize the check box so that its label is fully displayed to the right. Resize it a little longer than needed in Forms Builder so that the label does not truncate at run-time.
3. Save and compile the form.
   Click Run Form to run your form and test the changes.
4. In the CUSTG*XX* form, convert the Credit_Limit text item into a radio group.
   Add radio buttons for Low, Medium, and High to represent database values of 500, 2000, and 5000.
   Define access keys of L for Low, M for Medium, and H for High.
   Add text Credit Limit to describe the radio group’s purpose.
   Set Label to Low for the Low radio button, Medium for the Medium radio button, and High for the High radio button.
   Ensure that new records display the default of Low, and that existing records with other values display as Medium.
5. Save, compile, and run the form to test the changes.



10 Creating Noninput Items

Schedule:	**Timing**	**Topic**
	60 minutes	Lecture
	40 minutes	Guided Practice
	100 minutes	Total



Objectives

After completing this lesson, you should be able to do the following:

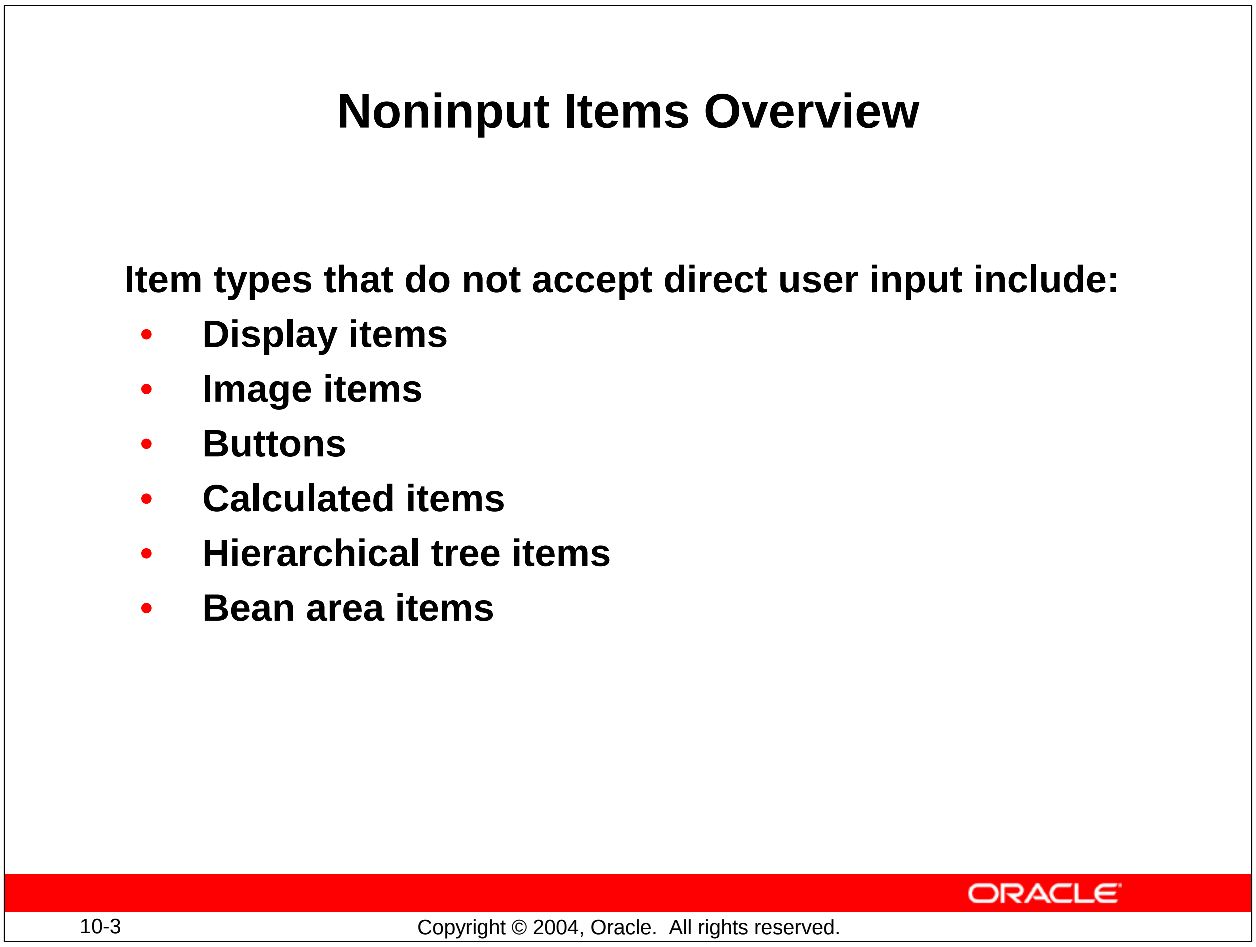
- Identify item types that do not allow input
- Create a display item
- Create an image item
- Create a button
- Create a calculated item
- Create a hierarchical tree item
- Create a bean area item

10-2

Introduction

Overview

Some Oracle Forms Developer item types do not accept user input (noninput items); however, they do provide an effective means of accessing data and initiating actions. This lesson describes how to create and use noninput items.



Noninput Items Overview

Item types that do not accept direct user input include:

- **Display items**
- **Image items**
- **Buttons**
- **Calculated items**
- **Hierarchical tree items**
- **Bean area items**

ORACLE

10-3

Noninput Items

Noninput items is a generic term for item types that do not accept direct user input. However, you can set the value of some noninput items by programmatic control. Noninput items can be divided into two groups—those that can display data and those that cannot.

Noninput Items that can Display Data

- Display items
- Image items
- Calculated items
- Hierarchical tree items

Noninput Items that Cannot Display Data

- Push Buttons

The Bean Area item can fall into either of these categories, depending on the functionality of the implemented JavaBean.

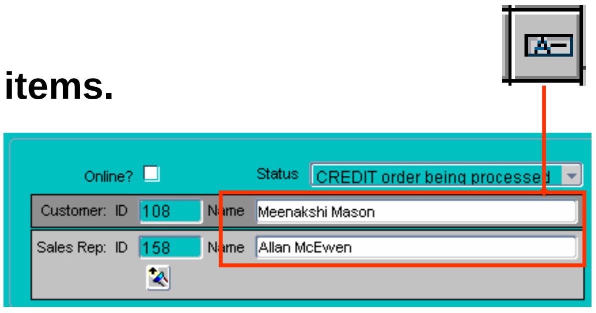
Using Noninput Items

Use noninput items to enhance your application by displaying additional data, often from a nonbase table.

Display Items

Display items:

- **Are similar to text items.**
- **Cannot:**
  - **Be edited**
  - **Be queried**
  - **Be navigated to**
  - **Accept user input**
- **Can display:**
  - **Nonbase table information**
  - **Derived values**



Display Items

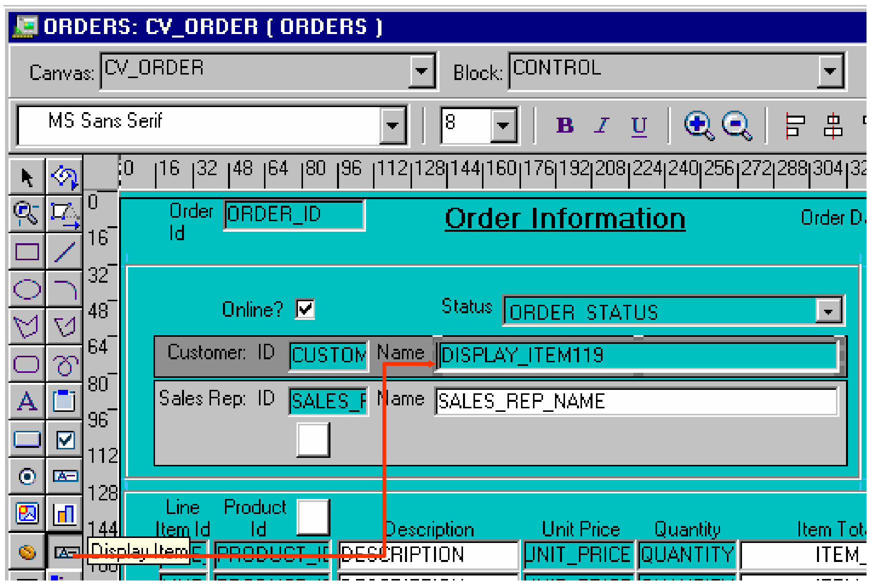
What Are Display Items?

A display item is similar to a text item, except that it cannot be edited or navigated to at runtime. A display item is a read-only text box whose value must be fetched or assigned programmatically.

Display items:

- Display additional, nonbase table information
- Display derived data values

Creating a Display Item



Creating a Display Item

A display item can be created by using:

- The Layout Editor
- The Create icon in the Object Navigator (creates a text item that you can convert to a display item)
- The Item Type property to convert an existing item into a display item

Whichever method you choose, you can set the required item properties in the Property Palette. Set the Database Item property to No for a display item whose value is not stored in the base table.You can assign a format mask to a single-line display item by setting its Format Mask property.

How to Create a Display Item From the Layout Editor

To create a display item in the Layout Editor, perform the following steps:

1. Invoke the Layout Editor.
2. Display the desired canvas and ensure that the correct data block is set.
3. Select the Display Item tool.
4. Click the canvas at the position where the display item is required.
5. Double-click the new display item. The Property Palette displays.
6. Change the name from DISPLAY_ITEMXX to the required name.
7. Specify the other properties as required.

Image Items

Use image items to display images:

- **From file system—supported file type**
- **From database—LONG RAW column or a BLOB column**

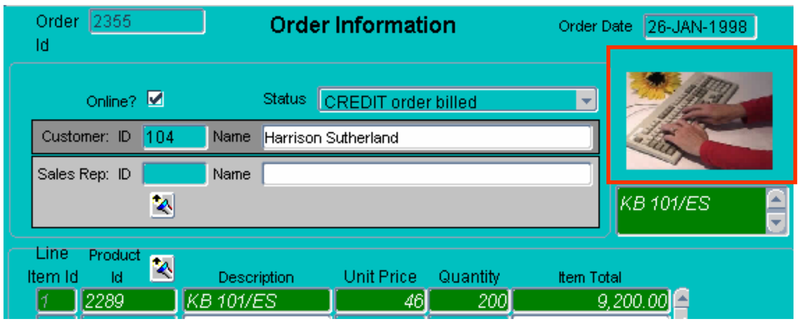


Image Items

Using Images

You can use images as graphic objects within a form module. A graphics image displays automatically and cannot be manipulated at run time. It can be imported from the database or the file system.

Alternatively, you can display images within image items.

What is an Image Item?

An image item is a special interface control that can store and display vector or scanned images. Just as text items store and display VARCHAR2, number, or date values, image items store and display images.

Like other items that store values, image items can be either data block items or control items.

Image Items (continued)

Displaying Image Items

You can populate an image item by using one of the following methods:

- Fetching from a `LONG RAW` or `BLOB` database column
- Using a trigger and a built-in to populate the image item programmatically

Storing Images

You can store images in either the database or the file system.

When you insert images into the database by means of a save (commit), they are automatically compressed using Oracle image compression.

Where Image Is Stored	Description
Database	Long Raw column compressed image that can be up to two gigabytes or BLOB column that can be up to four gigabytes
File	Any supported file format

Note: To conserve application server memory when displaying large image items, reduce the number of records that are buffered by manipulating the Number of Records Buffered data block property.

Technical Note

You can also populate an image item with a BFILE, but you will need to use `DBMS_LOB` to do so.

Image File Formats

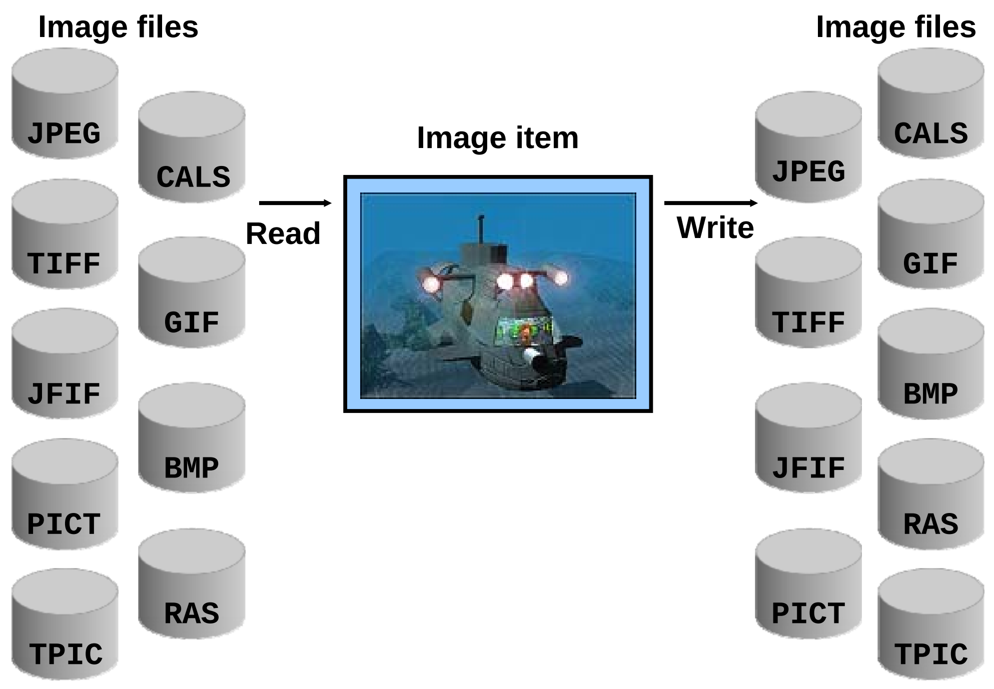


Image File Formats

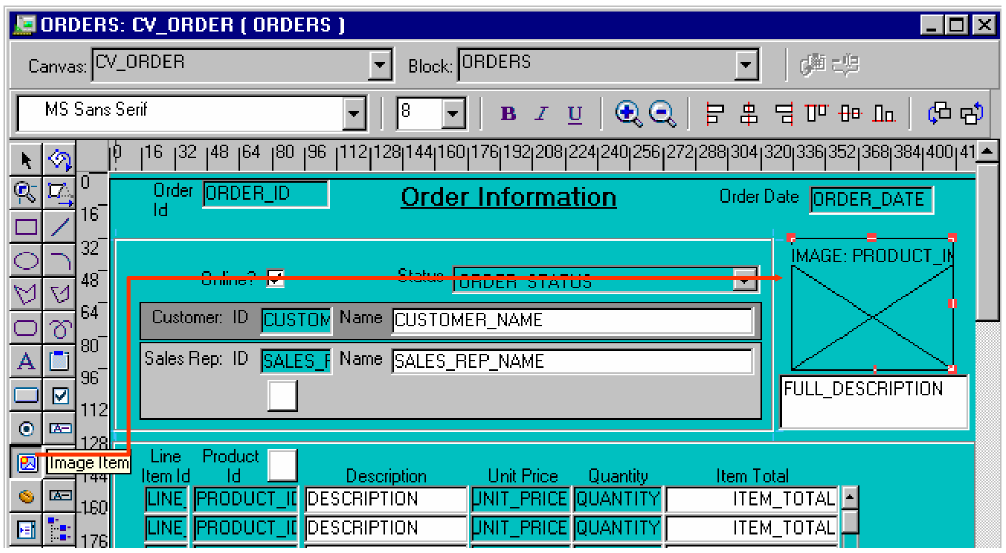
Forms Builder supports the following image formats:

File Suffix	Description	Image Items
BMP	MS Windows and OS/2 BitMap Picture	Read/Write
CALS	CALS type raster	Read/Write
GIF	CompuServe	Read/Write
JFIF	JPEG File Interchange Format	Read/Write
TIFF	Tag Image File Format	Read/Write
JPEG	Joint Photographic Experts Group	Read/Write
PICT	Macintosh Quickdraw Picture	Read/Write
RAS	Sun Raster	Read/Write
TPIC	Truevision Raster Graphics Array Picture	Read/Write

To reduce network traffic, limit the number of image items and background images that must be downloaded from the application server. You can deploy them as JAR files to reduce network traffic, or you can download them in an alternative manner.

For example, to display a company logo in your application, you could include the image in the HTML page that downloads at application startup rather than retrieving the image from the database or the file system. The HTML page can be cached.

Creating an Image Item



ORACLE

Creating an Image Item

An image item can be created in three ways:

- By using the Image Item tool in the Layout Editor (as described in the next section)
- By using the Create icon in the Object Navigator (creates a text item that you can convert to an image item)
- By converting an existing item into an image item

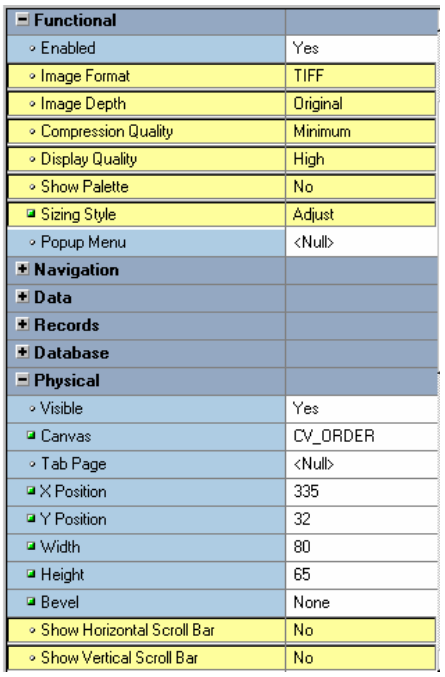
Steps to Create an Image Item from the Layout Editor

1. Invoke the Layout Editor.
2. Set the canvas and block to those on which you require the item to display.
3. Select the Image Item tool.
4. Click the canvas at the position where you want the image item to display.
5. Double-click the image item. The Property Palette displays.
6. Change the name from IMAGEXX to the required name.
7. Specify the other properties as required.

Note: Remember to set the Database Item property to No for an image item whose value is not stored in the base table.

Setting Image-Specific Item Properties

- Image Format
- Image Depth
- Compression Quality
- Display Quality
- Sizing Style
- Show Horizontal Scroll Bar
- Show Vertical Scroll Bar



⊟ Functional	
Enabled	Yes
Image Format	TIFF
Image Depth	Original
Compression Quality	Minimum
Display Quality	High
Show Palette	No
Sizing Style	Adjust
Popup Menu	<Null>
⊞ Navigation	
⊞ Data	
⊞ Records	
⊞ Database	
⊟ Physical	
Visible	Yes
Canvas	CV_ORDER
Tab Page	<Null>
X Position	335
Y Position	32
Width	80
Height	65
Bevel	None
Show Horizontal Scroll Bar	No
Show Vertical Scroll Bar	No

Setting Image Specific Item Properties

Set the following properties to affect the appearance and behavior of the image item:

- **Image Format:** Format in which image will be stored in the database
- **Image Depth:** Depth setting for image being read from or written to a file on the file system (Original, Monochrome, Gray, LUT , or RGB)
- **Compression Quality:** Degree of compressions for an image item being read from or written to a file (None, Minimum, Low, Medium, High, or Maximum)
- **Display Quality:** Resolution used to display the image item; controls trade off between quality and performance (High, Medium, or Low)
- **Sizing Style:** Determines how much of image displays when the image sized does not match the size of the item (Crop – cuts off edges of image so it fits in item display area; Adjust – scales image so it fits within the item display area)
- **Show Horizontal/Vertical Scroll Bar:** Displays scroll bars to enable scrolling of image that does not fit into the item display area

Setting Image Specific Item Properties (continued)

Note: Image items do not have a Data Type property. If you set an image item Database Item property to Yes, Forms Builder understands that the data type is LONG RAW.

Instructor Note

- The Show Palette property has no effect. In prior Forms versions, the image palette enabled the user to manipulate the image at run time in client/server mode only.
- LUT = Lookup Table
- RGB = Red, Green, Blue

Push Buttons

Push buttons:

- **Cannot display or represent data**
- **Are used to initiate an action**
- **Display as:**
  - **Text button** Exit
  - **Iconic**

ORACLE

Push Buttons

What Is a Push Button?

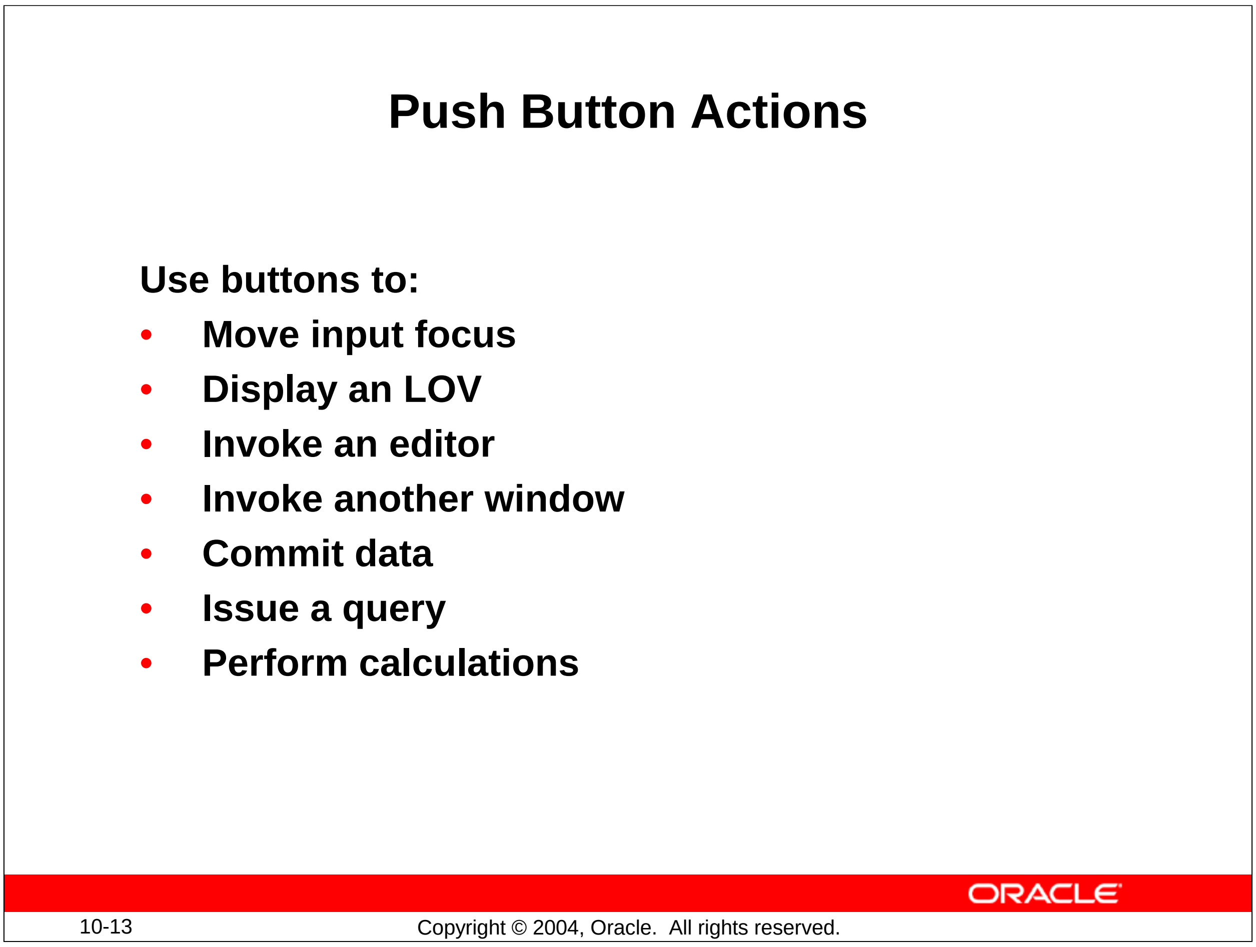
A *push button* is an interface object that you click to initiate an action. A push button is usually displayed as a rectangle with a descriptive label inside. Push buttons cannot store or display values.

You can enhance your form module further by adding push buttons to provide quick and direct access to the most needed operations.

Push Button Styles

Forms Builder supports two push button styles:

- **Text button:** Displayed with a text label on the push button
- **Iconic button:** Displayed with a bitmapped graphic on the push button, and often used in toolbars



Push Button Actions

Use buttons to:

- Move input focus
- Display an LOV
- Invoke an editor
- Invoke another window
- Commit data
- Issue a query
- Perform calculations

10-13

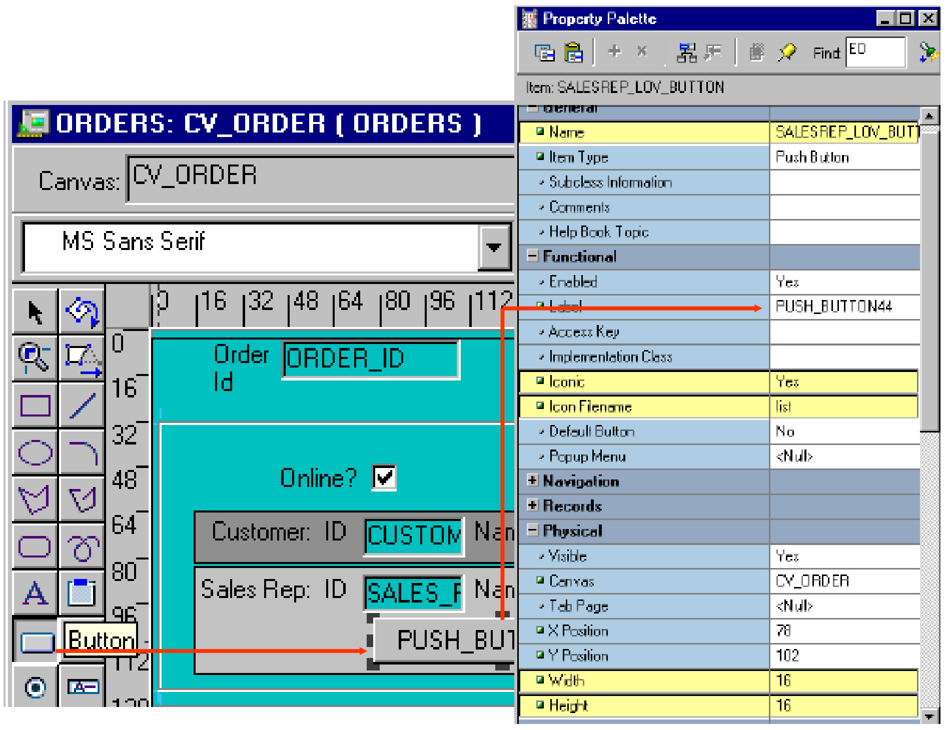
Push Button Actions

Some typical actions that would be invoked by a push button include the following:

- Moving the input focus
- Displaying an LOV
- Invoking an editor
- Invoking another window
- Committing data
- Issuing a query
- Performing calculations

Note: Push buttons do not accept input focus on some window managers. On these platforms, the Keyboard Navigable property has no effect, and users can only interact with the items by using a mouse. Clicking a push button does not move the input focus on these platforms. The input focus remains in the item that was active before user clicked the push button.

Creating a Push Button



Property Palette

Find ED

Item: SALESREP_LOV_BUTTON

General	
Name	SALESREP_LOV_BUTT
Item Type	Push Button
Subclass Information	
Comments	
Help Book Topic	
Functional	
Enabled	Yes
Label	PUSH_BUTTON44
Access Key	
Implementation Class	
Iconic	Yes
Icon Filename	list
Default Button	No
Popup Menu	<Null>
Navigation	
Records	
Physical	
Visible	Yes
Canvas	CV_ORDER
Tab Page	<Null>
X Position	78
Y Position	102
Width	16
Height	16

Creating a Push Button

A push button can be created by using:

- The Push Button tool in the Layout Editor
- The Create icon in the Object Navigator (creates a text item that you can convert to a push button)

How to Create a Push Button From the Layout Editor

To create a push button from the Layout Editor, perform the following steps:

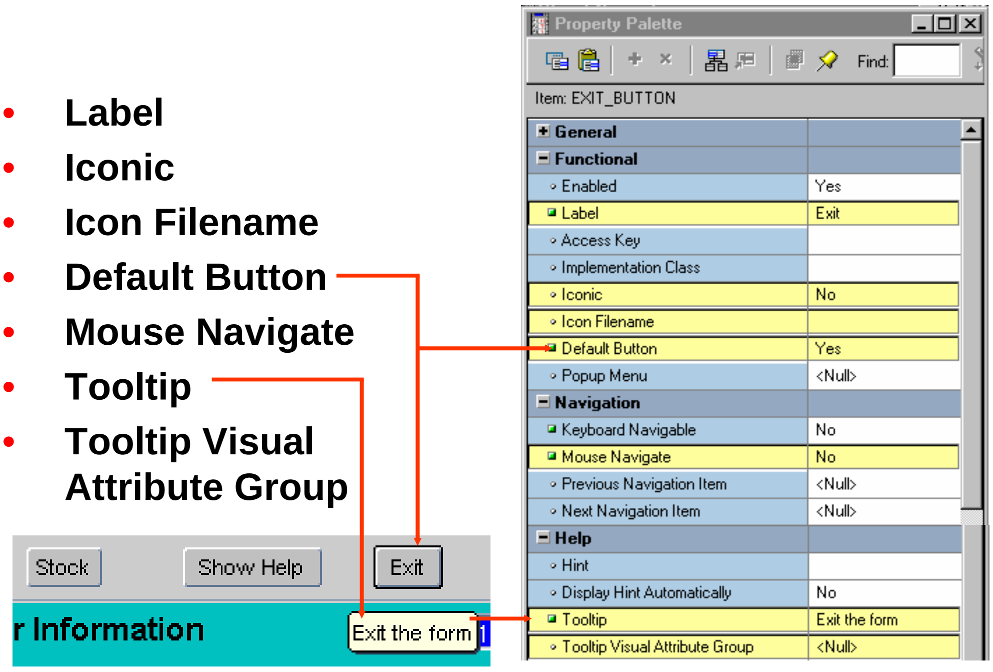
1. Invoke the Layout Editor.
2. Set the canvas and block to those on which you require the push button to display.
3. Select the Push Button tool.
4. Click the canvas at the position where you want the push button to display.
5. Double-click the push button. The Property Palette displays.
6. Change the name from PUSH_BUTTONXX to the required name.
7. Specify the other properties as required.

Note

- You can use the mouse to resize and move the push button once you have created it.
- Icon image files for iconic buttons must be in GIF or JPEG format.

Setting Push Button Properties

- Label
- Iconic
- Icon Filename
- Default Button
- Mouse Navigate
- Tooltip
- Tooltip Visual Attribute Group



Item: EXIT_BUTTON

General	
Functional	
Enabled	Yes
Label	Exit
Access Key	
Implementation Class	
Iconic	No
Icon Filename	
Default Button	Yes
Popup Menu	<Null>
Navigation	
Keyboard Navigable	No
Mouse Navigate	No
Previous Navigation Item	<Null>
Next Navigation Item	<Null>
Help	
Hint	
Display Hint Automatically	No
Tooltip	Exit the form
Tooltip Visual Attribute Group	<Null>

Setting Properties for the Push Button

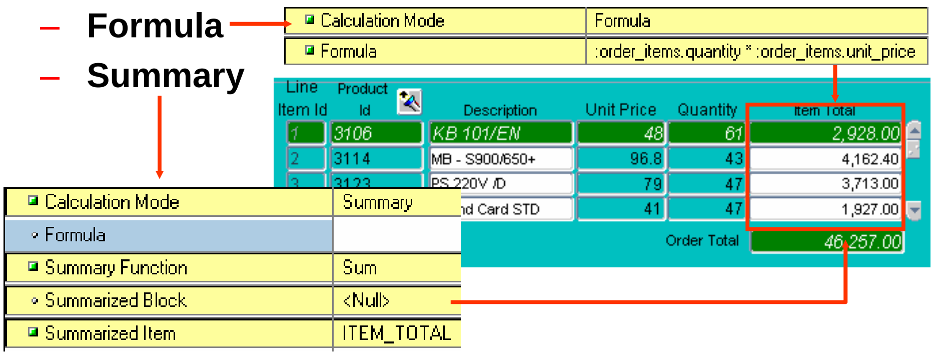
- **Label:** Text label that appears on the button at run time
- **Iconic:** Whether the button displays as an icon instead of as a label
- **Icon Filename**: Name of the file that contains the icon resource (filename only without the extension, such as "`list`", not "`list.gif`")
- **Default Button:** Whether this is the default push button for the block, which can be selected implicitly by pressing `[Select]` without the need to navigate or use the mouse
- **Mouse Navigate:** Whether Forms navigates to the item when you click on it with the mouse
- **Tooltip:** Help text that should appear in a tool tip beneath the button when the mouse moves over it
- **Tooltip Visual Attribute Group:** Named visual attribute to apply to the tool tip at run time

Note: The default push button may be bordered or highlighted in a unique fashion to distinguish it from other push buttons.

Calculated Items

What are calculated items?

- **They accept item values that are based on calculations.**
- **They are read-only.**
- **They can be expressed as:**
  - **Formula**
  - **Summary**



Calculation Mode	Formula
Formula	:order_items.quantity * :order_items.unit_price

Line Item Id	Product Id	Description	Unit Price	Quantity	Item Total
1	3106	KB 101/EN	48	61	2,928.00
2	3114	MB - S900/650+	96.8	43	4,162.40
3	3123	PS 220V /D	79	47	3,713.00
		nd Card STD	41	47	1,927.00

Order Total 46,257.00

Calculation Mode	Summary
Formula	
Summary Function	Sum
Summarized Block	<Null>
Summarized Item	ITEM_TOTAL

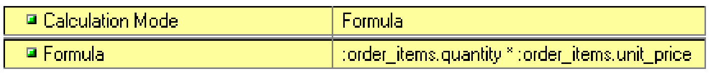
ORACLE

Calculated Items

With a calculated item you can declaratively base item values on calculations involving one or more variable values. For example, you can use a calculated item to display a running total of employees' total compensation.

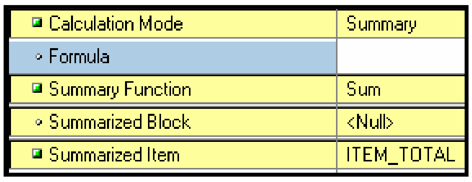
Any item that can store a value can be used as a calculated item by setting its required property values.

Creating a Calculated Item by Setting Properties



Calculation Mode	Formula
Formula	:order_items.quantity * :order_items.unit_price

- **Formula**
  - A calculated item value is the result of a horizontal calculation.
  - It involves bind variables.
- **Summary**
  - A calculated item value is a vertical calculation.
  - A summary is performed on values of a single item over all rows in a block.



Calculation Mode	Summary
Formula	
Summary Function	Sum
Summarized Block	<Null>
Summarized Item	ITEM_TOTAL

Creating a Calculated Item

A calculated item can be created by:

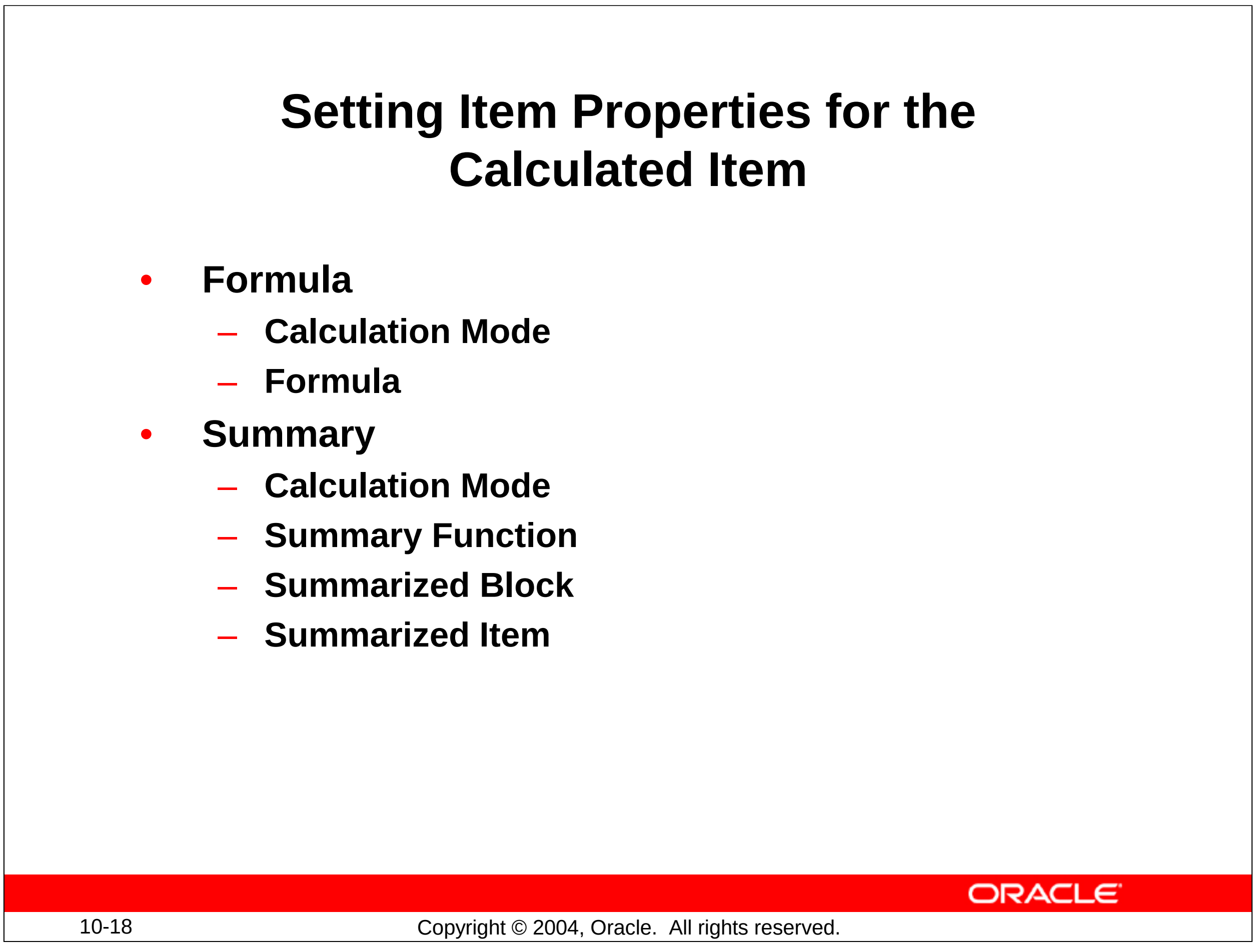
- Setting the calculation specific properties of any existing item that can store a value
- Creating a new item in the Layout Editor and setting its calculation specific properties
- Using the Create icon in the Object Navigator and setting its calculation specific properties

Calculation Modes

Calculations can be expressed as a formula or as a summary of all items in a block. Forms Builder supports the following calculation modes:

- **Formula:** The calculated item value is the result of a horizontal calculation involving one or more bind variables, such as form items, global variables, and parameters.
- **Summary:** The calculated item value is a vertical calculation involving the values of a single item over all rows within a single block.

Note: A calculated item is read only. End users cannot insert or modify calculated items. You should, therefore, generally use display items as calculated items.



Setting Item Properties for the Calculated Item

- **Formula**
  - **Calculation Mode**
  - **Formula**
- **Summary**
  - **Calculation Mode**
  - **Summary Function**
  - **Summarized Block**
  - **Summarized Item**

10-18

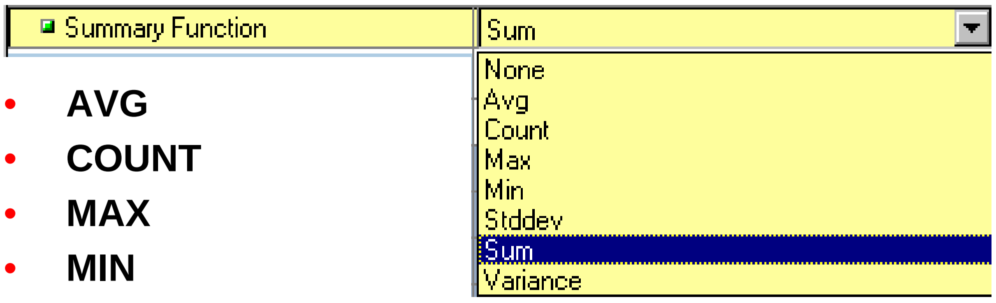
Setting Item Properties for the Calculated Item

Unlike the other items covered so far in this lesson, there is no Item Type property value called Calculated Item. The calculation-specific properties of an item make it a calculated item. Text items and display items both support calculated items.

The following properties are specific to calculated items:

- **Calculation Mode:** The method of computing the calculated item values (None, Formula, or Summary)
- **Formula:** A single PL/SQL expression that determines the calculated value of a formula calculated item; can compute a value or call a subprogram
- **Summary Function:** The type of summary function (discussed on next page) to be performed on the summary calculated item
- **Summarized Block:** The block over which all rows will be summarized in order to assign a value to the summary calculated item; if not specified, current block is assumed
- **Summarized Item:** The item whose value is summarized in order to assign a value to the calculated item; required if the Calculation Mode is Summary

Summary Functions



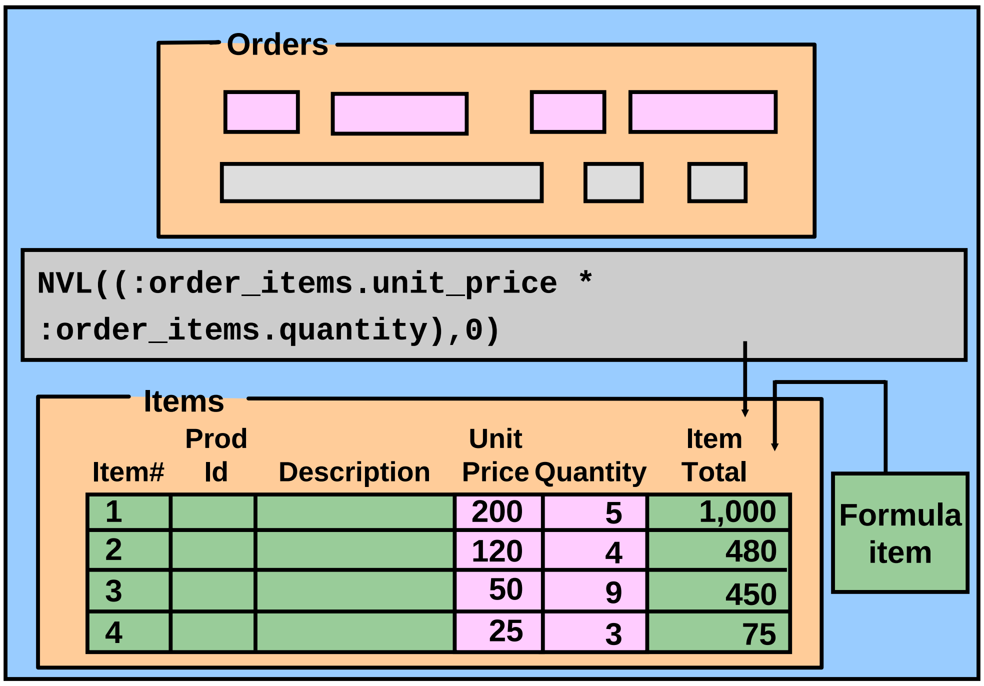
- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE

Summary Functions

You can use the standard SQL aggregate functions for summary items:

- **AVG:** The average value (arithmetic mean) of the summarized item over all records in the block
- **COUNT:** Count of all non-null instances of the summarized item over all records in the block
- **MAX/MIN:** Maximum/minimum value of the summarized item over all records in the block
- **STDDEV:** The standard deviation of the summarized item's values over all records in the block
- **SUM:** Sum of all values of the summarized item over all records in the block
- **VARIANCE:** The variance (square of the standard deviation) of the summarized item's values over all records in the block

Calculated Item Based on a Formula



```
NVL((:order_items.unit_price *
:order_items.quantity),0)
```

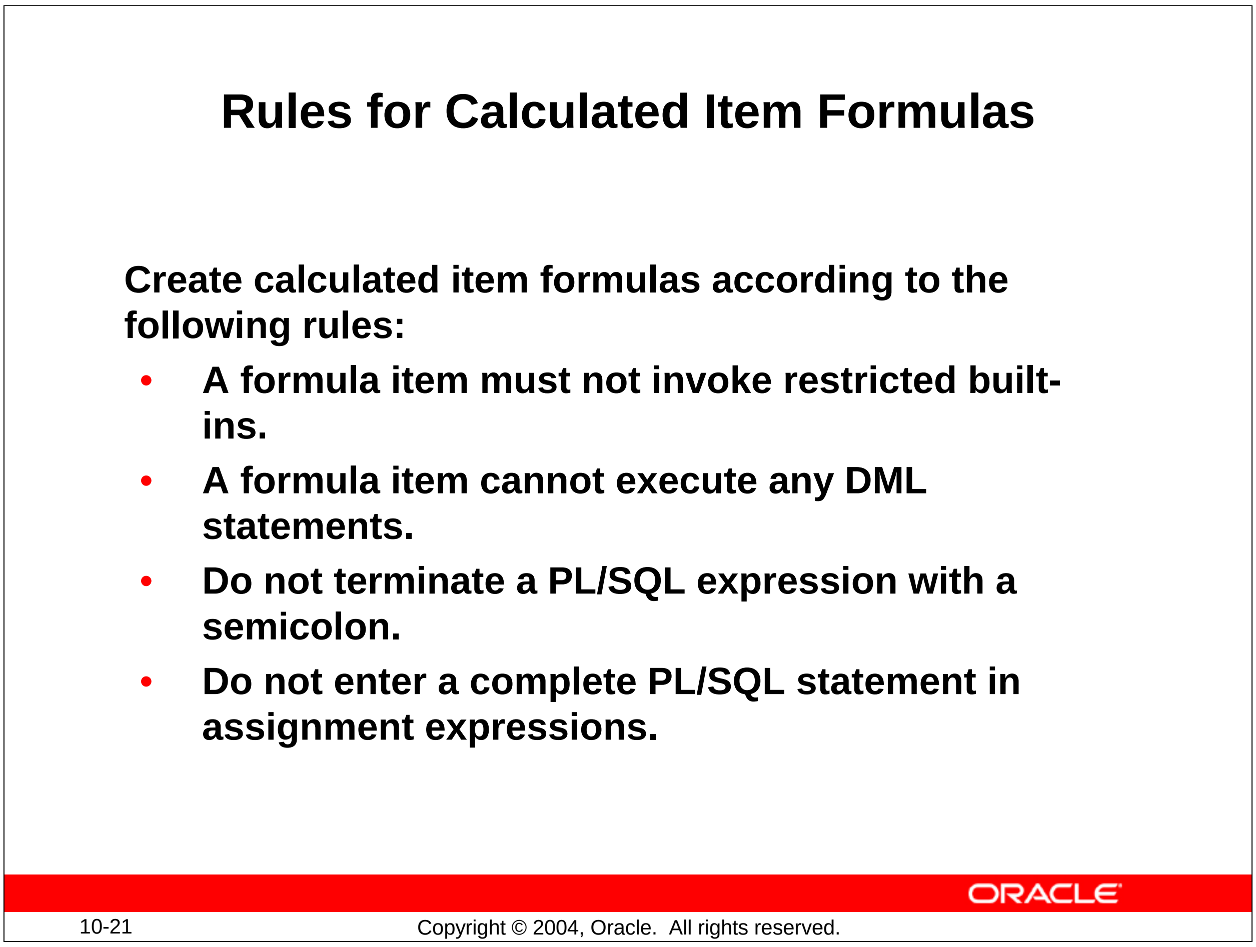
Item#	Prod Id	Description	Unit Price	Quantity	Item Total
1			200	5	1,000
2			120	4	480
3			50	9	450
4			25	3	75

Calculated Item Based on a Formula

To create a calculated item based on a formula, perform the following steps:

1. Create a new item in the Object Navigator.
2. Open the Property Palette of the item.
3. Set the Calculation Mode property to Formula.
4. Click More for the Formula property and enter the PL/SQL expression to define the formula.

Note: A formula item cannot be a database item because its value is computed by Forms, not queried from a database column.



Rules for Calculated Item Formulas

Create calculated item formulas according to the following rules:

- **A formula item must not invoke restricted built-ins.**
- **A formula item cannot execute any DML statements.**
- **Do not terminate a PL/SQL expression with a semicolon.**
- **Do not enter a complete PL/SQL statement in assignment expressions.**

ORACLE

10-21

Rules for Calculated Item Formulas

When writing formulas for calculated items, observe the following rules:

- The formula (and any user-written subprogram that calls it) must not invoke any restricted built-ins.
- The formula (and any user-written subprogram that calls it) cannot execute any DML statements.
- Do not terminate the PL/SQL expression with a semicolon.
- If the PL/SQL expression involves an assignment, do not enter the complete PL/SQL statement. Forms Builder assigns the actual assignment code internally.

Example

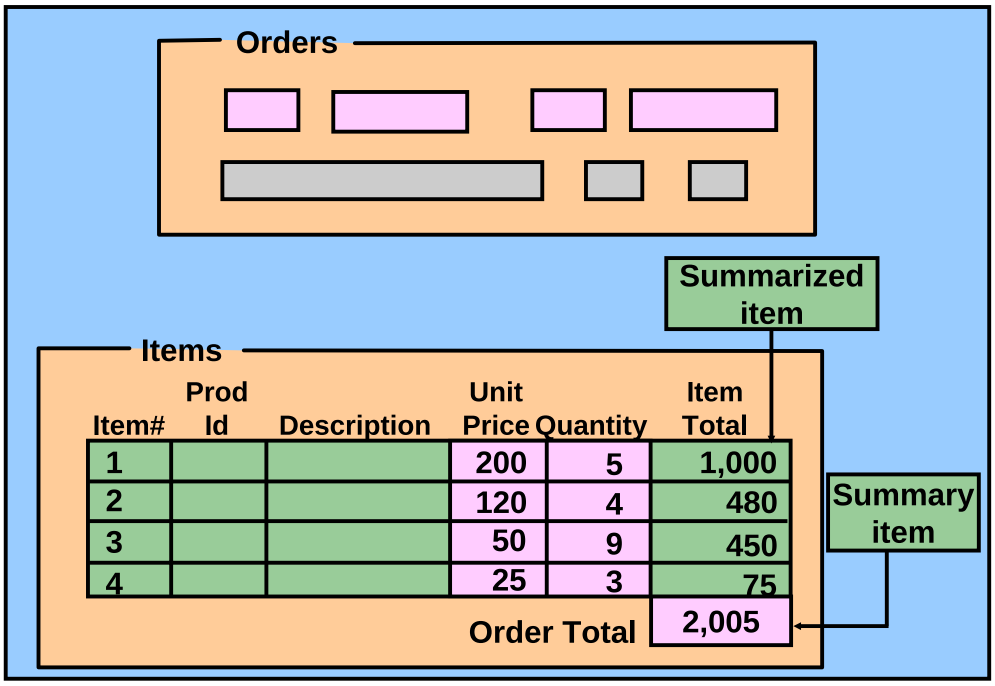
Suppose you have a formula item called Gross_Comp in the EMPLOYEES block of a form. If you set the Formula property to:

```
NVL(:employees.salary,0) *
NVL(:employees.commission_pct,0)
```

Forms Builder will internally convert this expression into a complete statement as:

```
:employees.gross_comp := (NVL(:employees.salary,0) *
     NVL(:employees.commission_pct,0));
```

Calculated Item Based on a Summary



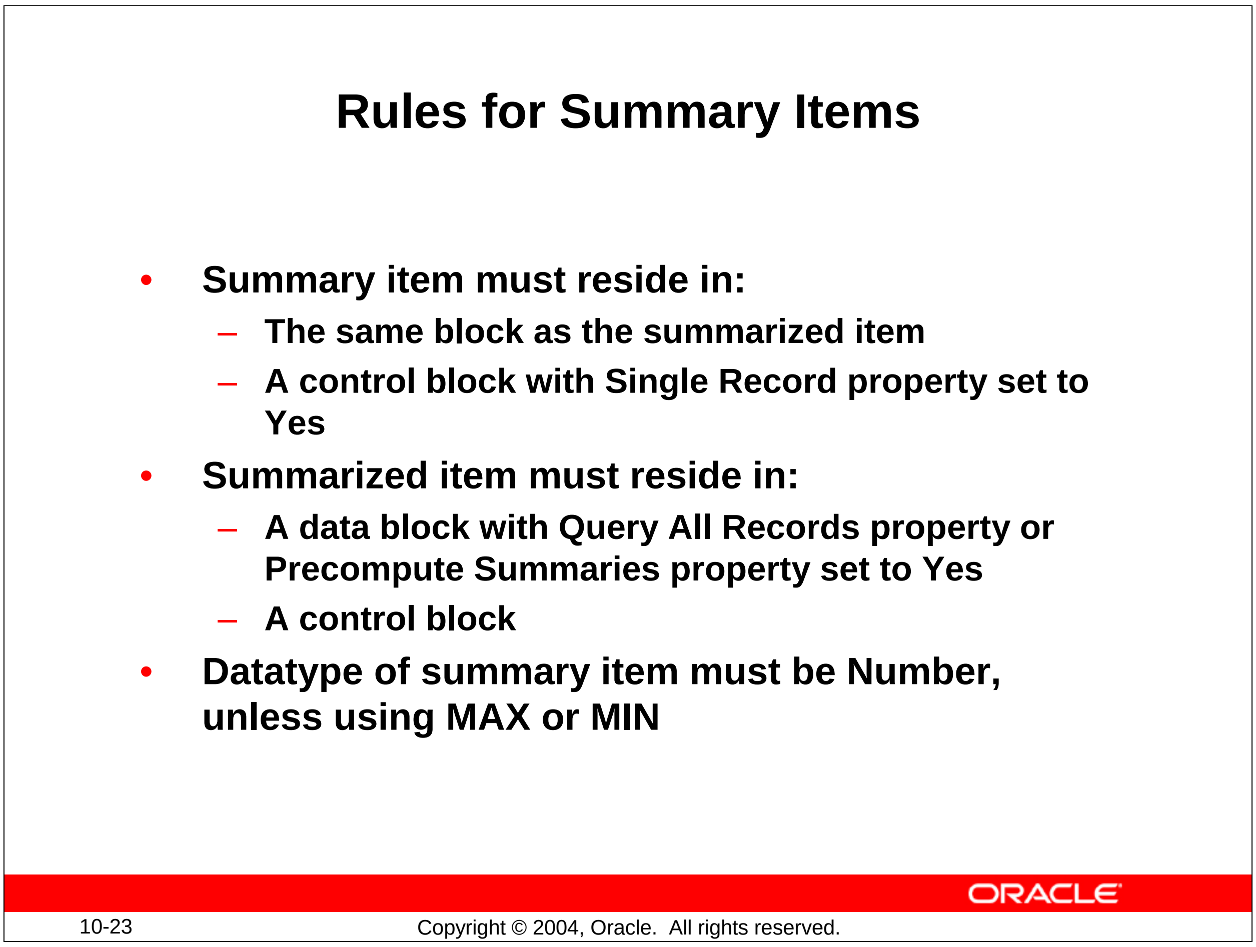
Calculated Item Based on a Summary

To create a calculated item based on a summary, perform the following steps:

1. Create a new item in the Object Navigator.
2. Open the Property Palette of an item.
3. Set the Calculation Mode property to Summary.
4. Select the required function from the Summary Function pop-up list.
5. From the Summarized Block pop-up list, select a block over which all rows will be summarized.
6. From the Summarized Item pop-up list, select an item to be summarized.

Note: A *summary item* is the calculated item to which you assign a value.

A *summarized item* is the item whose values are summarized and then assigned to the summary item.



Rules for Summary Items

- Summary item must reside in:
  - The same block as the summarized item
  - A control block with Single Record property set to Yes
- Summarized item must reside in:
  - A data block with Query All Records property or Precompute Summaries property set to Yes
  - A control block
- Datatype of summary item must be Number, unless using MAX or MIN

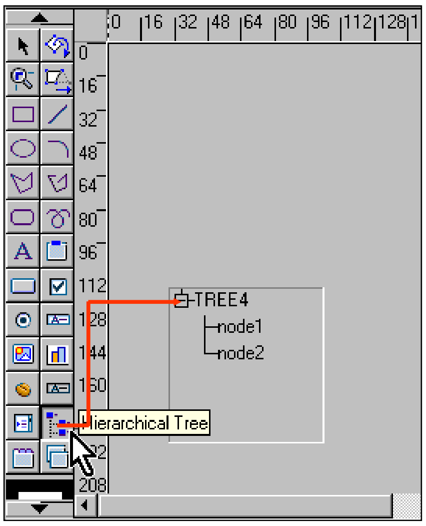
ORACLE

Rules for Summary Items

When creating calculated items based on a summary, observe the following rules:

- The summary item must reside in the same block as the summarized item, or in a control block whose Single Record property is set to Yes.
- The summarized item must reside in a control block, or in a data block whose Query All Records property or the Precompute Summaries property is set to Yes.
  Note: This ensures that records fetched in the block and the summarized value are consistent. Otherwise, another user may possibly update a record that has not been fetched yet.
- Set the Data Type property for a summary item to Number, unless the summary function is Max or Min, in which case the datatype must mirror that of its associated summarized item. For example, a calculated item that displays the most recent (maximum) date in the `HIRE_DATE` column must have a datatype of Date.
- If the summarized item values are based on a formula, the summarized item must reside in a block whose Query All Records property is set to Yes.

Creating a Hierarchical Tree Item



Creating a Hierarchical Tree Item

A hierarchical tree is an item that displays data in the form of a standard navigator.

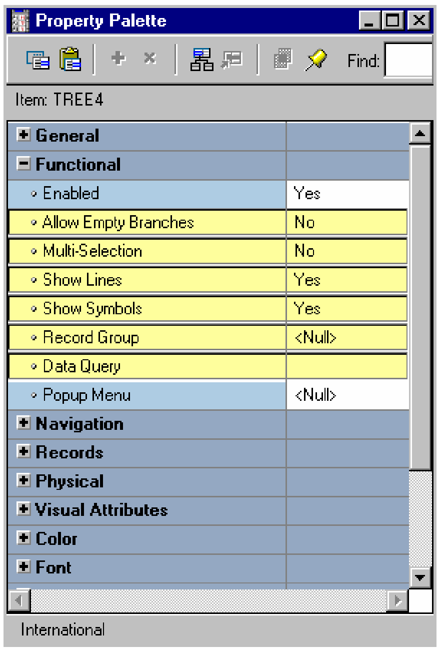
How to Create a Hierarchical Tree Item

To create a hierarchical tree item, do one of the following:

- **In the Layout Editor:**
  - Click the Hierarchical Tree icon.
  - Click and drag the mouse on the canvas to create the hierarchical tree object.
  - Set other hierarchical tree-related properties as required.
- **In the Object Navigator:**
  - Create a new item by using the Create icon.
  - Open the item's Property Palette and set the Item Type property to Hierarchical Tree.
  - Set other hierarchical tree related properties as required.

Setting Hierarchical Tree Item Properties

- Allow empty branches
- Multi selection
- Show lines
- Show symbols
- Record group
- Data query



Item: TREE4

General	
Functional	
Enabled	Yes
Allow Empty Branches	No
Multi-Selection	No
Show Lines	Yes
Show Symbols	Yes
Record Group	<Null>
Data Query	
Popup Menu	<Null>
Navigation	
Records	
Physical	
Visual Attributes	
Color	
Font	

International

ORACLE

Setting Hierarchical Tree Item Properties

Hierarchical Tree properties include the following:

- **Item Type:** Must be set to Hierarchical Tree
- **Allow Empty Branches:** Whether branch nodes may exist with no children (if set to FALSE, branch nodes with no children will be converted to leaf nodes; if set to TRUE, an empty branch will be displayed as a collapsed node)
- **Multi Selection:** Whether multiple nodes may be selected at one time
- **Show Lines:** Whether a hierarchical tree displays lines leading up to each node
- **Show Symbols:** Whether a hierarchical tree should display + or - symbols in front of each branch node
- **Record Group:** Name of the record group from which the hierarchical tree derives its values
- **Data Query:** Specifies the query-based data source

Several built-ins are available to manipulate hierarchical trees. These are discussed in Lesson 15, which discusses how to implement a hierarchical tree in a form.

Note: A hierarchical tree must be the only item in the data block.

Bean Area Items

The Bean Area item enables you to:

- Add a JavaBean to a form
- Extend Forms functionality
- Interact with client machine
- Reduce network traffic

ORACLE

Bean Area Items

A JavaBean is a component written in Java that can plug into any applet or Java application. The bean area item enables you to extend Forms functionality by adding a JavaBean to your form.

With JavaBeans, you can interact with the client machine, performing such functions as:

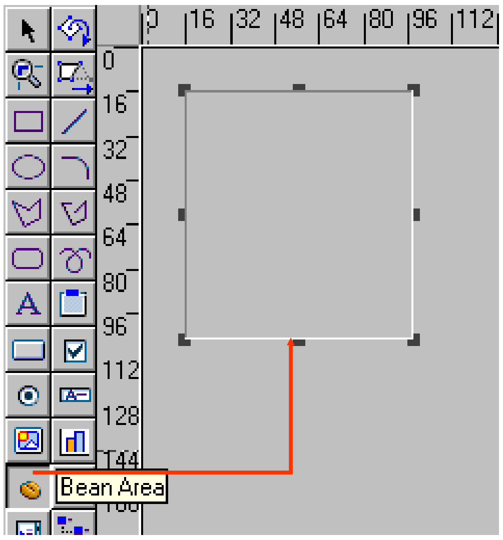
- Obtaining information about the client machine
- Uploading client files to the server machine
- Modifying the user interface on the client
- Checking the spelling of a text item
- Displaying a progress bar, clock, calendar, or color picker with which the operator may be able to interact and select values

Some of this functionality, such as the calendar, is possible using Forms native functionality. However, using a JavaBean enables client interaction without generating network traffic.

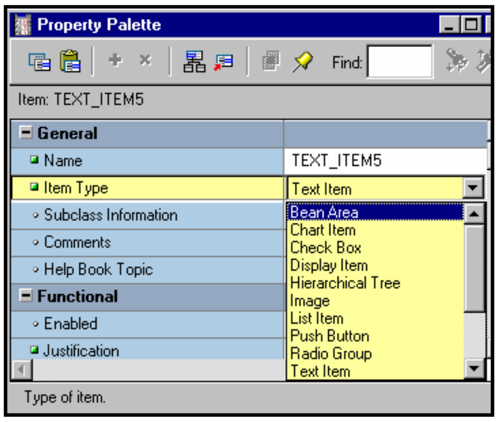
Although JavaBeans can be used to input data, as in the case of the Calendar JavaBean, the bean area item itself does not accept user input.

Creating a Bean Area Item

Create bean area in Layout Editor



Convert existing item to bean area



Property Palette

Find:

Item: TEXT_ITEM5

General	
Name	TEXT_ITEM5
Item Type	Text Item
Subclass Information	
Comments	
Help Book Topic	
Functional	
Enabled	
Justification	

Bean Area
Chart Item
Check Box
Display Item
Hierarchical Tree
Image
List Item
Push Button
Radio Group
Text Item

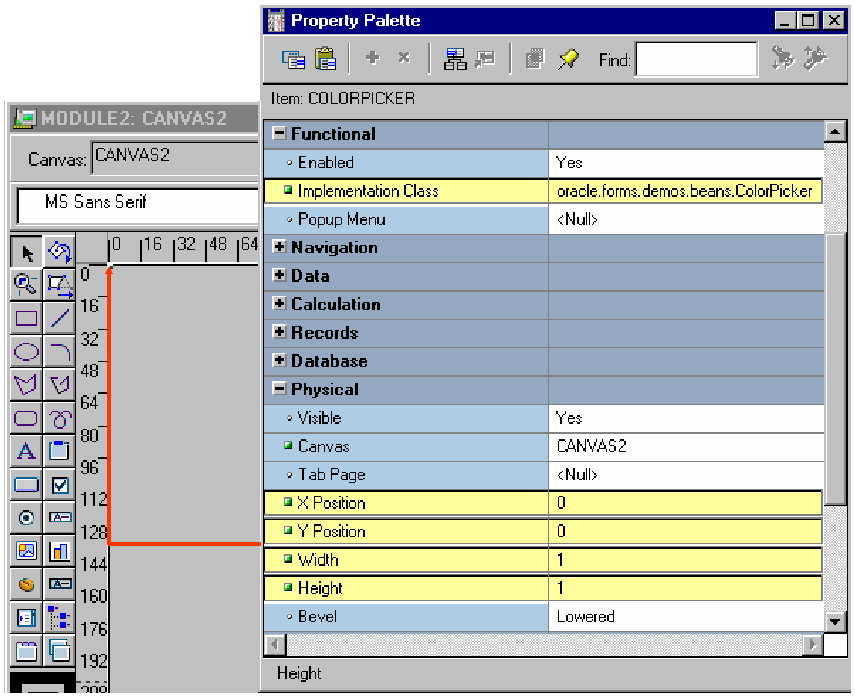
Type of item.

Creating a Bean Area

You create a bean area using the Bean Area tool in the Layout Editor. Click the tool and drag out an area on the canvas. At first the area will look like an empty rectangle.

You can also create a bean area from any existing item by changing its item type to Bean Area in the Property Palette.

Setting Bean Area Item Properties



Item: COLORPICKER

Functional	
Enabled	Yes
Implementation Class	oracle.forms.demos.beans.ColorPicker
Popup Menu	<Null>
Navigation	
Data	
Calculation	
Records	
Database	
Physical	
Visible	Yes
Canvas	CANVAS2
Tab Page	<Null>
X Position	0
Y Position	0
Width	1
Height	1
Bevel	Lowered

Height

Setting Properties for the Bean Area

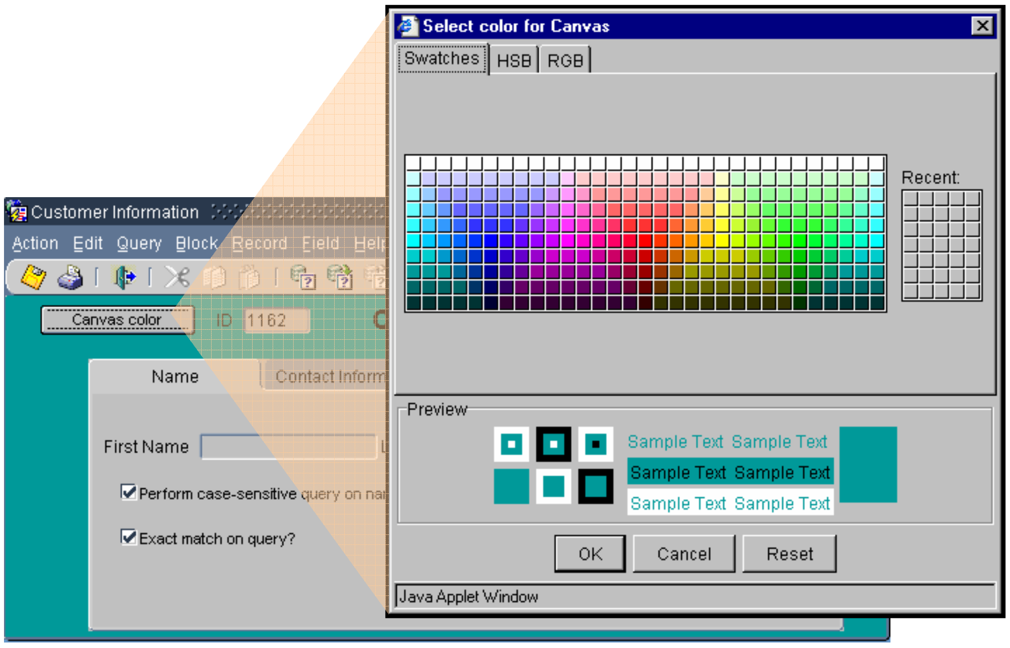
The most important Bean Area property is Implementation Class. This property is used to specify the fully qualified name of the Java class that the Bean Area item should instantiate.

If the JavaBean has a visible component, Forms Builder displays the JavaBean within the Bean Area item in the Layout Editor once the Implementation Class is set. You can set the size and position of the bean area visually in the Layout Editor, or by changing the X/Y Position, Width, and Height properties.

Some JavaBeans have no visible component. To avoid the distraction of a large empty rectangle on the form, you can set the properties so that the bean displays as a tiny dot in the corner of the canvas. Alternatively, you can set its Visible property to No, which ensures that the Bean Area does not display at run time, although you still see it in the Layout Editor.

Note: With enhanced JavaBean, it is not necessary to set the Implementation Class at design time. Instead, you can programmatically register the bean at run time. This is discussed in Lesson 16.

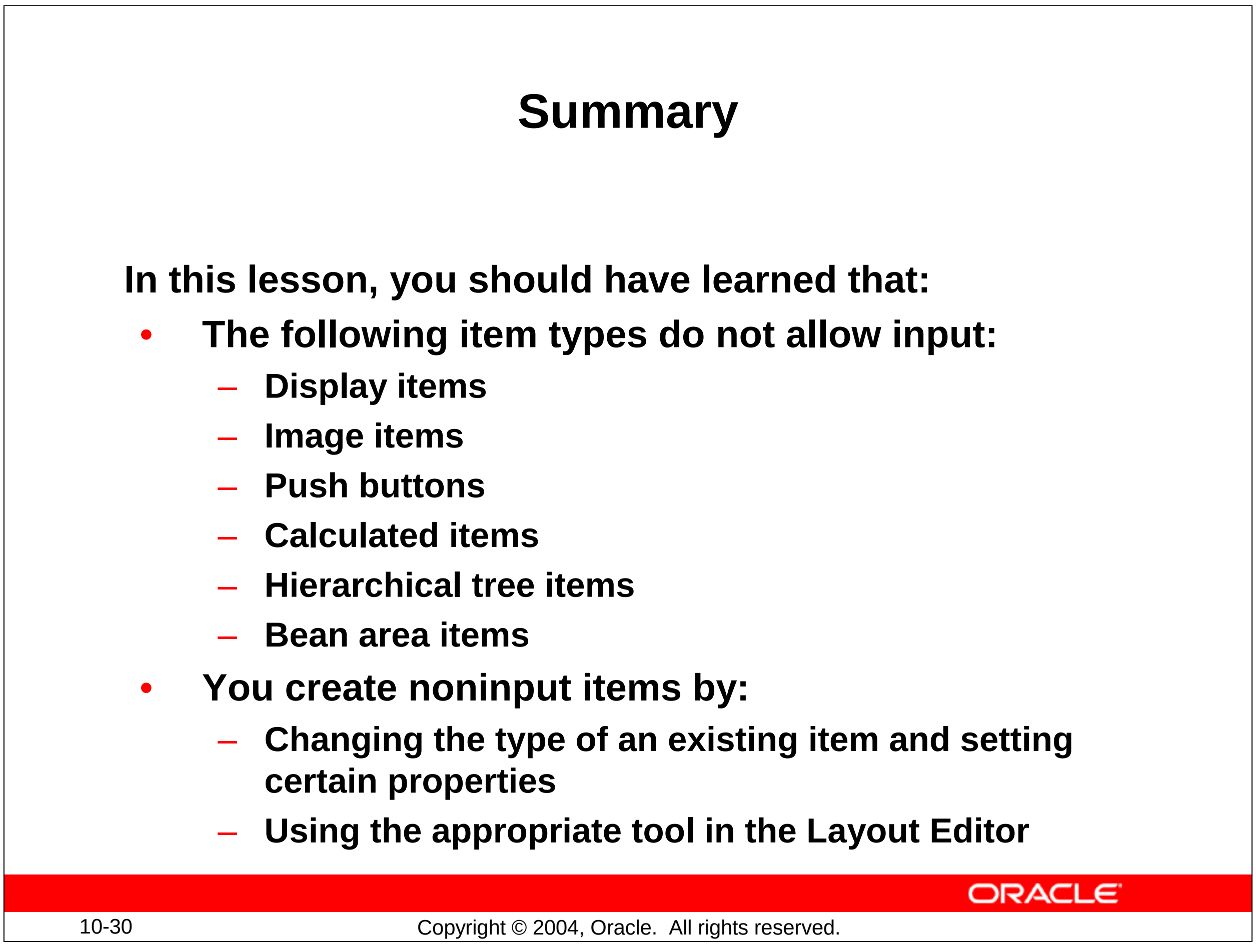
The JavaBean at Run Time



The JavaBean at Run Time

At run time, the user can interact with the JavaBean to perform the functionality that you have programmatically defined. In the preceding example, there is a button that invokes the ColorPicker JavaBean. When the operator selects a color from the color picker, it is used to set the background color of the canvas.

This programmatic interaction is discussed in Lesson 16.



Summary

In this lesson, you should have learned that:

- **The following item types do not allow input:**
  - **Display items**
  - **Image items**
  - **Push buttons**
  - **Calculated items**
  - **Hierarchical tree items**
  - **Bean area items**
- **You create noninput items by:**
  - **Changing the type of an existing item and setting certain properties**
  - **Using the appropriate tool in the Layout Editor**

10-30

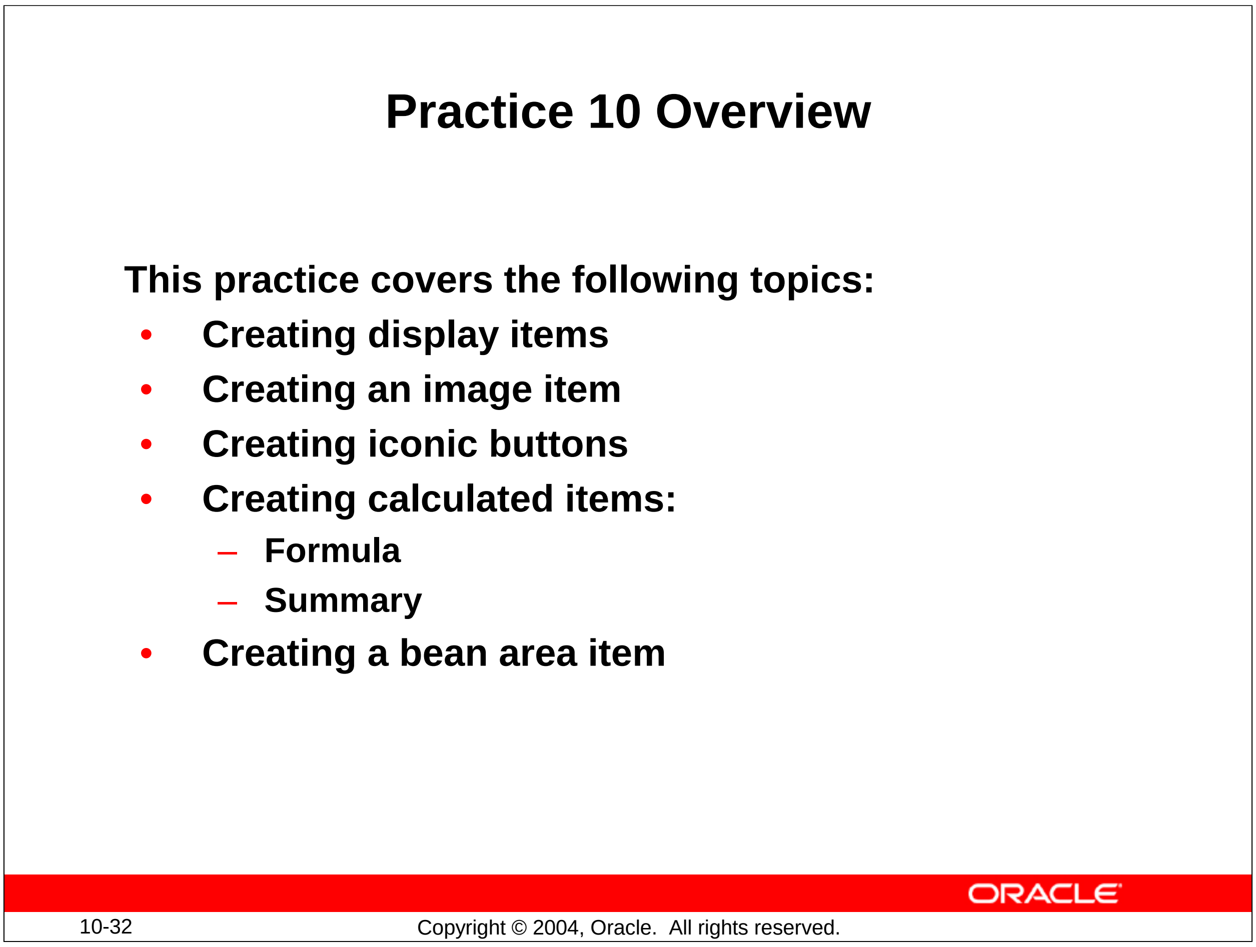
Summary

In this lesson, you should have learned that:

- Display items display nonbase table information or derived values.
- Image items store and display vector or scanned bitmapped images.
- Push Buttons initiate an action.
- Calculated items base item values on calculations. Calculations can be expressed in one of the following modes:
  - Formula
  - Summary
- Hierarchical trees display information in an Object Navigator style display.
- Bean Area items enable you to integrate Java components into your application.
- You create the above item types by:
  - Using the appropriate tool in the Layout Editor
  - Clicking Create in the Object Navigator (this creates a text item that you can covert to a different item type)
  - Changing the item type and/or setting appropriate properties on existing items

Summary

- You can use:
  - A display item to show nonbase table information
  - An image item to display an image
  - A push button to initiate action
  - A calculated item to display the results of a formula or a summary function of another item
  - A hierarchical tree item to display related data in a hierarchical fashion
  - A bean area item to execute client-side Java code



Practice 10 Overview

This practice covers the following topics:

- Creating display items
- Creating an image item
- Creating iconic buttons
- Creating calculated items:
  - Formula
  - Summary
- Creating a bean area item

10-32

Practice 10 Overview

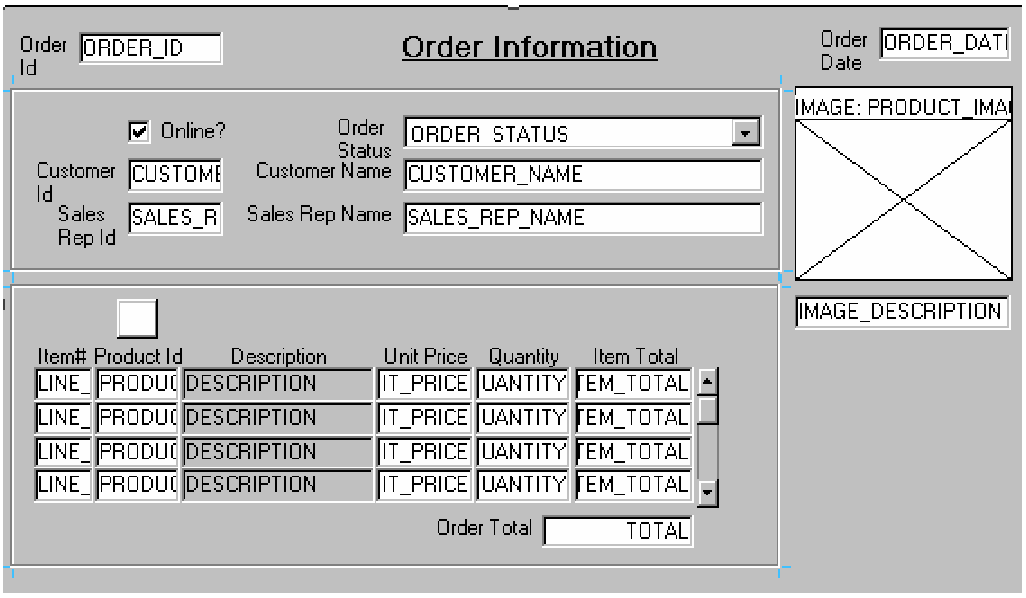
In this practice session, you will add several items in the CUSTOMERS and ORDERS forms: display items, image item, push buttons, and calculated items.

- In the ORDERS form:
  - Create two display items in the `ORDER_ITEMS` block.
  - Create an image item in the `CONTROL` block.
  - Create an iconic button in the `CONTROL` block.
  - Base the Item_Total item in the `ORDER_ITEMS` block on a formula.
  - Create a control item in the `CONTROL` block. Base this item value on a summary that displays the total value of an order.
- In the CUSTOMERS form
  - Create an iconic button in the `CONTROL` block.
  - Create a bean area.
- Save and run the ORDERS and CUSTOMERS forms.

Note: For solutions to this practice, see Practice 10 in Appendix A, “Practice Solutions.”

Practice 10

1. In the `ORDER_ITEMS` block of the ORDG*XX* form, create a display item called Description. Set the Prompt property to Description and display the prompt centered above the item. Rearrange the items in the layout so that all are visible.
2. Create an image item called Product_Image in the `CONTROL` block of the ORDG*XX* form. (Use the Control block because you do not want to pick up the Current Record Visual Attribute Group of the `ORDER_ITEMS` block, and this is a non base table item.) Position this and the other items you create in this practice so that your form looks like the screenshot.
   Note: The image will not display in the image item at this point; you will add code to populate the image item in Practice 20.



Order Id ORDER_ID

Order Information

Order Date ORDER_DATI

☑ Online?

Order Status ORDER_STATUS

Customer Id CUSTOME

Customer Name CUSTOMER_NAME

Sales Rep Id SALES_R

Sales Rep Name SALES_REP_NAME

IMAGE: PRODUCT_IMA

IMAGE_DESCRIPTION

Item#	Product Id	Description	Unit Price	Quantity	Item Total
LINE_	PRODUC	DESCRIPTION	IT_PRICE	UANTITY	TEM_TOTAL
LINE_	PRODUC	DESCRIPTION	IT_PRICE	UANTITY	TEM_TOTAL
LINE_	PRODUC	DESCRIPTION	IT_PRICE	UANTITY	TEM_TOTAL
LINE_	PRODUC	DESCRIPTION	IT_PRICE	UANTITY	TEM_TOTAL

Order Total TOTAL

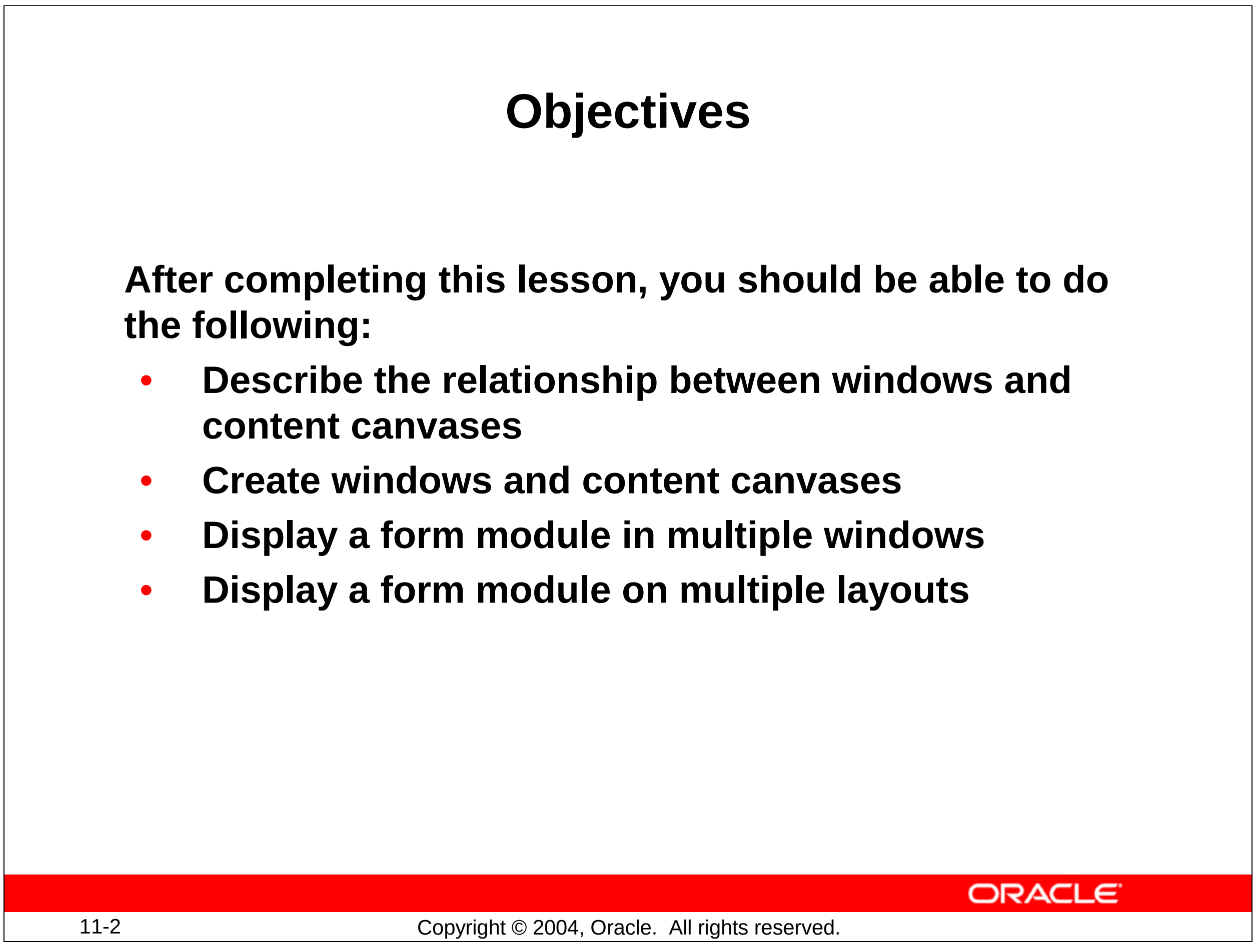
3. Create another display item, Image_Description, in the `ORDER_ITEMS` block. This should synchronize with the Description item. Set the Maximum Length property to the same value as the Description item.
4. In the `CONTROL` block of the ORDG*XX* form, create an iconic button called Product_LOV_Button. Use the `list` file (do not include the `.ico` or `.gif` extension). Set the Keyboard Navigable property and the Mouse Navigate property to No.
5. To display item total information, set the following properties for the Item_Total item in the `ORDER_ITEMS` block:
   Set the Item Type to Display Item.
   Set the Calculation Mode property to Formula.
   Set the Formula property to: `nvl(:ORDER_ITEMS.quantity,0) * nvl(:ORDER_ITEMS.unit_price,0)`

Practice 10 (continued)

6. To display the total of the item totals, create a new nonbase table item in the `CONTROL` block.
   Set the Position, Size, and Prompt properties according to the screenshot.
   Set the Format Mask property to 9G999G990D99.
   Set the Justification property to Right.
   Set the Number of Items Displayed property to 1.
   Set the Keyboard Navigable property to No.
   Make CONTROL.total a summary item and display summaries of the Item_Total values in the `ORDER_ITEMS` block. Ensure that you have set the Query All Records property to Yes for the `ORDER_ITEMS` block.
7. Click Run Form to run your form and test the changes. Perform a query in the ORDG*XX* form to ensure that the new items do not cause an error. Did you remember to switch off the Database Item property for items that do not correspond to columns in the base table? Also, check that the calculated items function correctly.
8. In the CUSTGXX form, create an iconic button similar to the one created in question 4, in the `CONTROL` block. Use the `list` file (do not include the `.ico` or `.gif` extension). Name the push button Account_Mgr_Lov_Button, and place it next to Account_Mgr_ID.
9. In the CUSTGXX form, create a bean area in the `CONTROL` block and name it Colorpicker. This bean has no visible component on the form, so set it to display as a tiny dot in the upper left corner of the canvas. Ensure that users cannot navigate to the bean area item with either the keyboard or mouse. You will not set an implementation class; in Lesson 19 you learn how to programmatically instantiate the JavaBean at run time.
10. Save and compile the form. Click Run Form to run your form and test the changes.

11 Creating Windows and Content Canvases

Schedule:	**Timing**	**Topic**
	30 minutes	Lecture
	20 minutes	Guided Practice
	50 minutes	Total



Objectives

After completing this lesson, you should be able to do the following:

- Describe the relationship between windows and content canvases
- Create windows and content canvases
- Display a form module in multiple windows
- Display a form module on multiple layouts

11-2

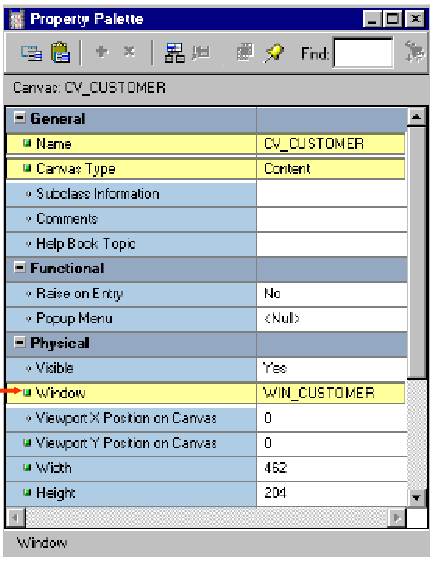
Introduction

Overview

With Oracle Forms Developer you can take advantage of the GUI environment by displaying a form module across several canvases and in multiple windows. This lesson familiarizes you with the window object and the default canvas type, the content canvas.

Windows and Canvases

- Window: Container for Forms Builder visual objects
- Canvas: Surface on which you “paint” visual objects
- To see a canvas and its objects, display the canvas in a window.


Property Palette

Find:

Canvas: CV_CUSTOMER

General	
Name	CV_CUSTOMER
Canvas Type	Content
Subclass Information	
Comments	
Help Book Topic	
Functional	
Raise on Entry	No
Popup Menu	<Null>
Physical	
Visible	Yes
Window	WIN_CUSTOMER
Viewport X Position on Canvas	0
Viewport Y Position on Canvas	0
Width	462
Height	204

Window

Windows and Canvases

With Forms Builder you can display an application in multiple windows by using its display objects—windows and canvases.

What Is a Window?

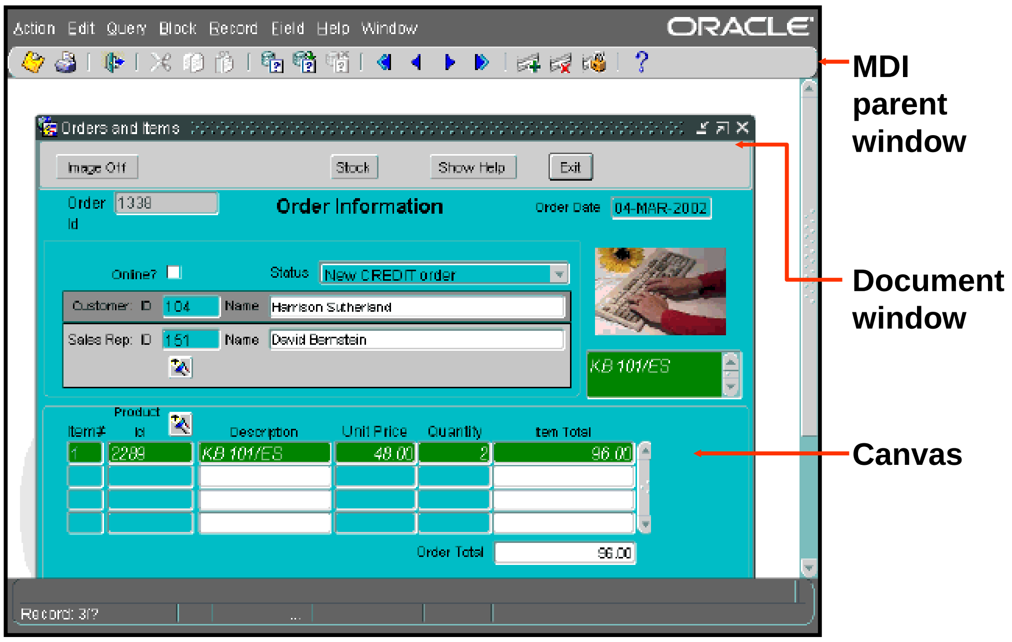
A window is a container for all visual objects that make up a Forms application. It is similar to an empty picture frame. The window manager provides the controls for the window that enable such functionality as scrolling, moving, and resizing. You can minimize a window. A single form may include several windows.

What Is a Canvas?

A canvas is a surface inside a window container on which you place visual objects such as interface items and graphics. It is similar to the canvas upon which a picture is painted. To see a canvas and its contents at run time, you must display it in a window. A canvas always displays in the window to which it is assigned.

Note: Each item in a form must refer to no more than one canvas. An item displays on the canvas to which it is assigned, through its Canvas property. Recall that if the Canvas property for an item is left unspecified, that item is said to be a Null-canvas item and will not display at run time.

Window, Canvas, and Viewport



Windows and Content Canvases (continued)

What Is a Viewport?

A viewport is an attribute of a canvas. It is effectively the visible portion of, or view onto, the canvas.

Instructor Note

You can demonstrate ownership and visual view at this point.

The Content Canvas

- “Base” canvas
- View occupies entire window
- Default canvas type
- Each window should have at least one content canvas

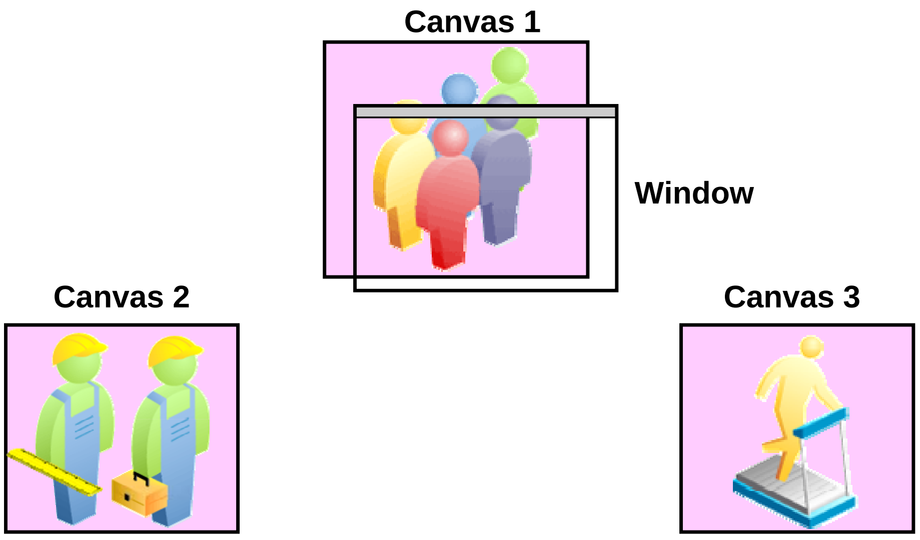
The Content Canvas

Forms Builder offers different types of canvases. A content canvas is the base canvas that occupies the entire content pane of the window in which it displays. The content canvas is the default canvas type. Most canvases are content canvases.

Instructor Note

Point out to the students that the other canvas types will be covered in the next lesson.

Relationship Between Windows and Content Canvases



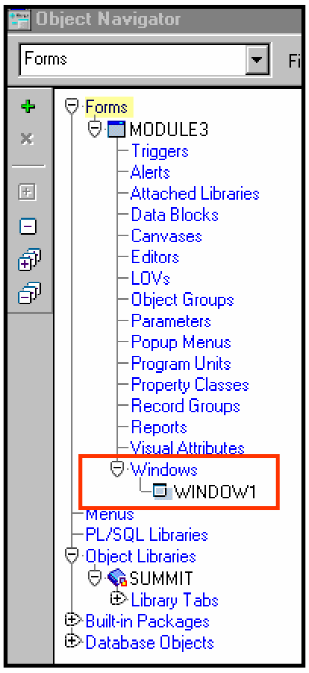
The Relationship Between Windows and Content Canvases

You must create at least one content canvas for each window in your application. When you run a form, only one content canvas can display in a window at a time, even though more than one content canvas can be assigned to the same window at design time.

At run time, a content canvas always completely fills its window. As the user resizes the window, Forms resizes the canvas automatically. If the window is too small to show all items on the canvas, Forms automatically scrolls the canvas to bring the current item into view.

Note: You can assign multiple content canvases to a window; however, only one content canvas at a time can be displayed in a window.

The Default Window



WINDOW1:

- **Created by default with each new form module**
- **Is modeless**
- **You can delete, rename, or change its attributes**

The Default Window

When you create a new form module, Forms Builder creates a new window implicitly. Thus, each new form module has one predefined window, which is called WINDOW1. You can delete or rename WINDOW1, or change its attributes.

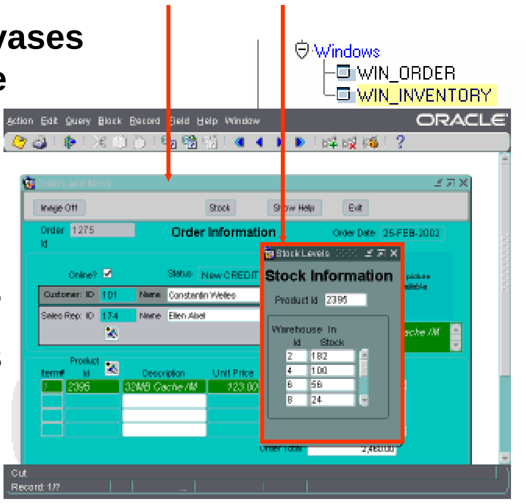
Uses and Benefits of a New Window

You can create additional windows in which to display your form application. A new or second window provides the ability to do the following:

- Display two or more content canvases at once.
- Modularize the form contents.
- Switch between canvases without replacing the initial one.
- Take advantage of window manager functionality such as minimizing.

Displaying a Form Module in Multiple Windows

- Use additional windows to:
  - Display two or more content canvases at once
  - Switch between canvases without replacing the initial one
  - Modularize form contents
  - Take advantage of the window manager
- Two types of windows
  - Modal
  - Modeless



Displaying a Form Module in Multiple Windows

You can create additional windows in a form so that you can:

- Display more than one content canvas at a time
- Switch between content canvases without the need to replace the first one
- Separate the form module into multiple layouts
- Use features of the window manager, such as allowing the user to resize or close a window

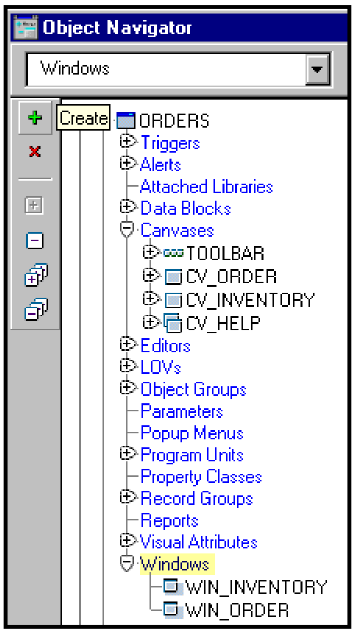
You can create two different types of windows:.

- A *modal window* is a restricted window that the user must respond to before moving the input focus to another window. Modal windows:
  - Must be dismissed before control can be returned to a modeless window
  - Become active as soon as they display
  - Require a means of exit or dismissal
- A *modeless window* is an unrestricted window that the user can exit freely. Modeless windows:
  - Can be displayed at the same time as multiple other modeless windows
  - Are not necessarily active when displayed
  - Are the default window type

Creating a New Window

Object Navigator: Click Create with Windows node selected

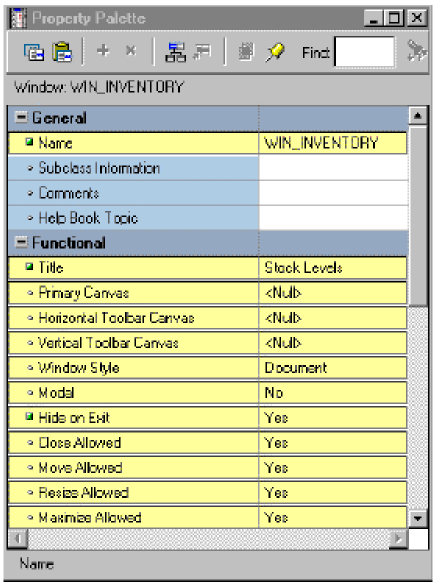
Property Palette: Set properties



Object Navigator

Windows

```
Create ORDERS
  Triggers
  Alerts
  Attached Libraries
  Data Blocks
  Canvases
    TOOLBAR
    CV_ORDER
    CV_INVENTORY
    CV_HELP
  Editors
  LOVs
  Object Groups
  Parameters
  Popup Menus
  Program Units
  Property Classes
  Record Groups
  Reports
  Visual Attributes
  Windows
    WIN_INVENTORY
    WIN_ORDER
```



Property Palette

Find:

Window: WIN_INVENTORY

General	
Name	WIN_INVENTORY
Subclass Information	
Comments	
Help Book Topic	
Functional	
Title	Stock Levels
Primary Canvas	<Null>
Horizontal Toolbar Canvas	<Null>
Vertical Toolbar Canvas	<Null>
Window Style	Document
Modal	No
Hide on Exit	Yes
Close Allowed	Yes
Move Allowed	Yes
Resize Allowed	Yes
Maximize Allowed	Yes

Name

Creating a New Window

To create a new window, perform the following steps:

1. Select the Windows node in the Object Navigator and click Create.
2. A new window entry displays, with a default name of WINDOWXX.
3. If the Property Palette is not already displayed, double-click the window icon to the left of the new window entry.
4. Set the window properties according to your requirements.

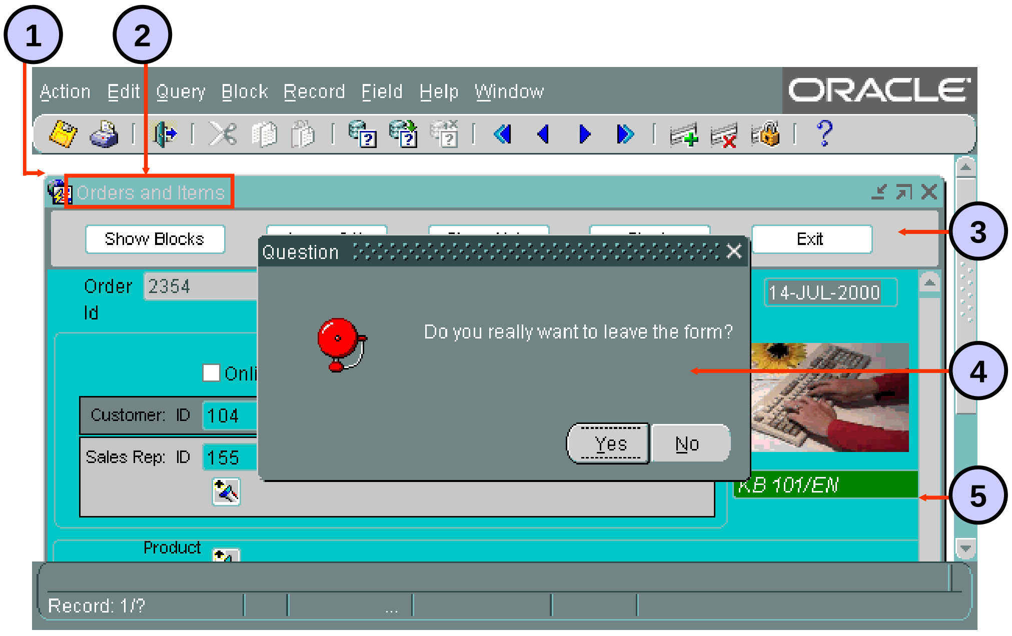
Note: For your new window to display, you must specify its name in the Window property of at least one canvas. To display a console to end users, set the form-level property Console Window to the window in which you want to display the console. To hide the console, set the property to `<Null>`.

Instructor Note

The Window Style of dialog has no effect in a Web environment because windows are always included in the applet area.

Demonstration: Using the ordwk09.fmb file, create a new window in the Orders form. Explain that the new window does not display at run time because it is not specified in the Window property of a canvas.

Setting Window Properties



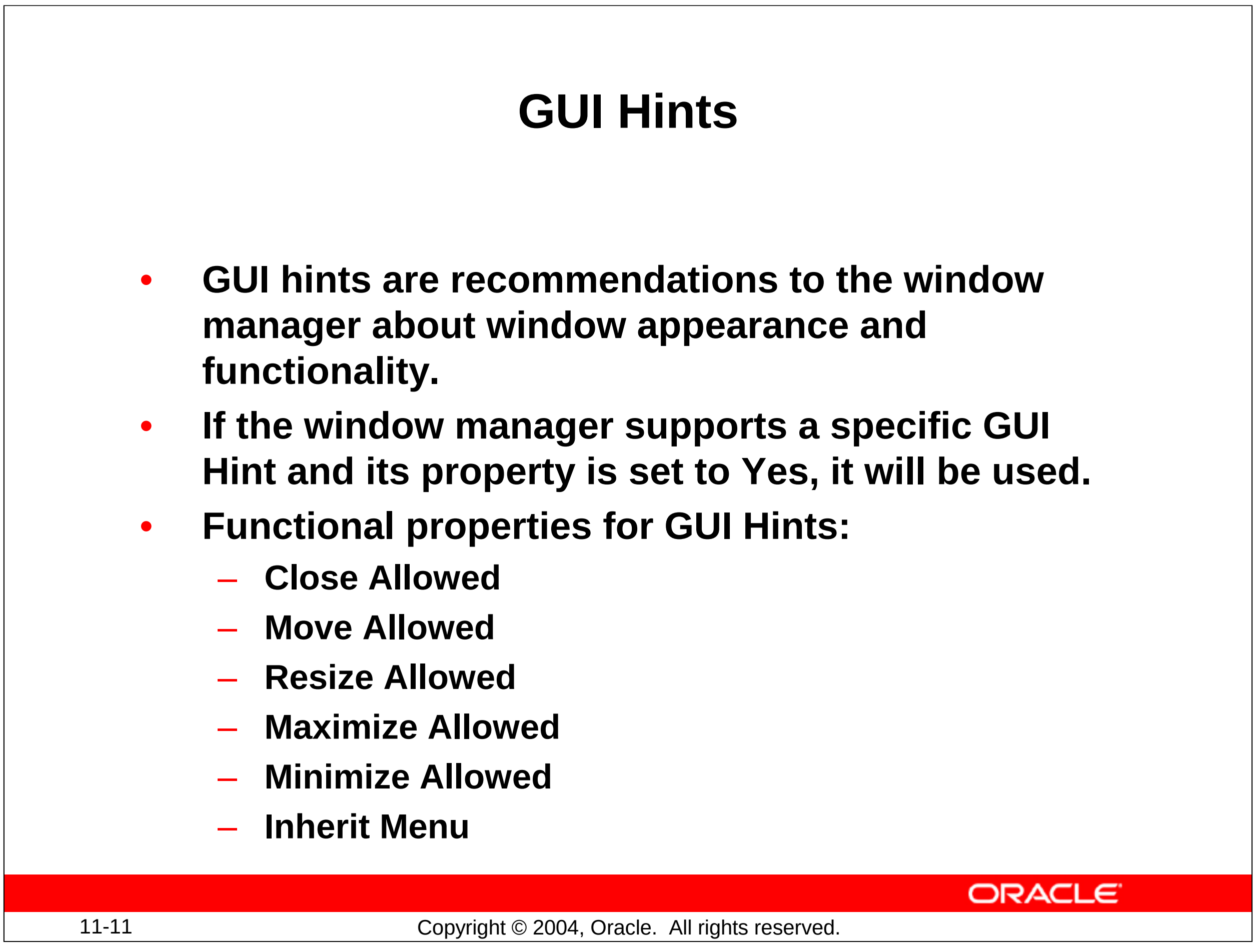
ORACLE

Setting Window Properties

You set properties for windows to determine their behavior and appearance. Some of these properties are:

1. **X/Y Position:** Specifies the location of the window within the containing window.
2. **Title:** The title to be displayed; if not specified, uses the name indicated by the window Name property
3. **Horizontal/Vertical Toolbar Canvas:** Specifies the toolbar canvas to be displayed horizontally across the top or vertically to the left of the window; selected from all horizontal/vertical toolbar canvases assigned to this window
4. **Modal:** Specifies if window is modal, requiring the user to dismiss the window before any other user interaction can continue, like Question window above
5. **Show Horizontal/Vertical Scroll Bar:** Whether a horizontal or vertical scroll bar should display on the window
6. **Hide on Exit:** Whether Forms hides the window automatically when the end user navigates to an item in another window (not shown on slide)

For a description of other properties that affect the behavior of windows, click the property in the Property Palette and press [F1].



GUI Hints

- GUI hints are recommendations to the window manager about window appearance and functionality.
- If the window manager supports a specific GUI Hint and its property is set to Yes, it will be used.
- Functional properties for GUI Hints:
  - Close Allowed
  - Move Allowed
  - Resize Allowed
  - Maximize Allowed
  - Minimize Allowed
  - Inherit Menu

ORACLE

11-11 Copyright © 2004, Oracle. All rights reserved.

GUI Hints

GUI Hints are recommendations to the window manager about the window appearance and functionality. There are certain properties under the Functional group that enable you to make these recommendations:

- **Close Allowed:** Enables the mechanism for closing the window (Forms responds to user attempts to close the window by firing a `WHEN-WINDOW-CLOSED` trigger.)
- **Move/Resize Allowed:** Whether user can move and resize the window
- **Maximize/Minimize Allowed:** Whether user can zoom or iconify the window
- **Inherit Menu:** Whether the window displays the current form menu

Instructor Note

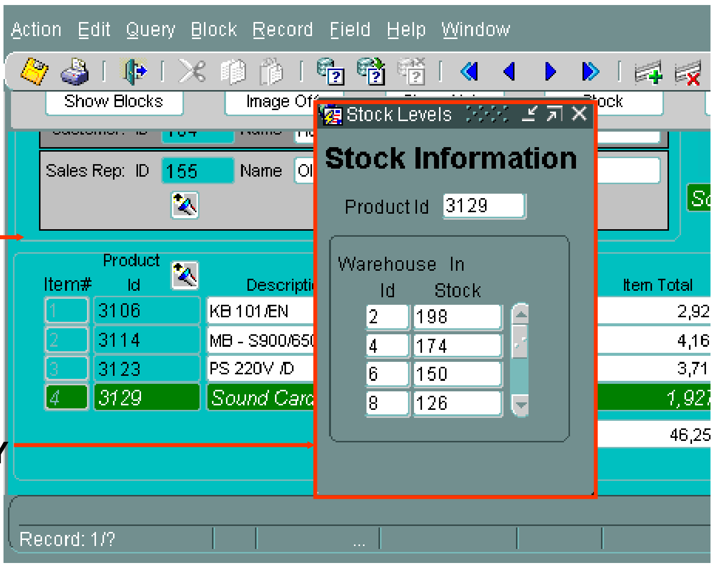
The Resize Allowed property can be set to prevent an end user from resizing the window, but it does not prevent you from resizing the window programmatically.

Displaying a Form Module on Multiple Layouts

PROPERTIES:

Canvas CV_ORDER
Window: WIN_ORDERS

Canvas CV_INVENTORY
Window: WIN_INVENTORY



Warehouse Id	In Stock
2	198
4	174
6	150
8	126

Item#	Product Id	Description	Item Total
1	3106	KB 101/EN	2,92
2	3114	MB - S900/650	4,16
3	3123	PS 220V /D	3,71
4	3129	Sound Card	1,927
			46,25

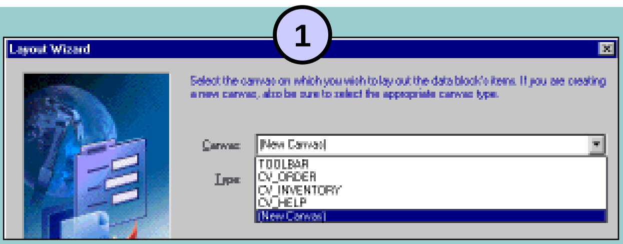
Displaying a Form Module on Multiple Layouts

You can have more than one content canvas in your form application. However, remember that only one content canvas can display in a window at one time. To display more than one content canvas at the same time, you can assign each content canvas to a different window.

Now you can display the form module on multiple layouts or surfaces.

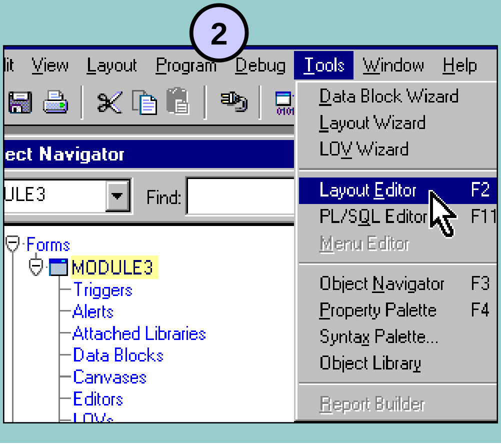
Creating a New Content Canvas

- **Implicitly:**

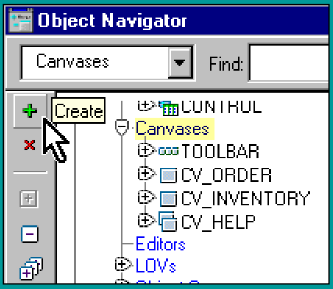

Select the canvas on which you wish to lay out the data block's items. If you are creating a new canvas, also be sure to select the appropriate canvas type.

Canvas: [New Canvas]

Type:



- **Explicitly:**



Creating a New Content Canvas

Creation of a content canvas can be implicit or explicit.

Implicitly Creating a Content Canvas

There are two ways of implicitly creating a new content canvas:

1. **Layout Wizard:** When you use the Layout Wizard to arrange data block items on a canvas, the wizard enables you to select a new canvas on its Canvas page. In this case, the wizard creates a new canvas with a default name of CANVAS*XX*.
2. **Layout Editor:** When there are no canvases in a form module and you invoke the Layout Editor; Forms Builder creates a default canvas on which you can place items.

Explicitly Creating a Content Canvas

You can create a new content canvas explicitly by using the Create icon in the Object Navigator.

Creating a New Content Canvas (continued)

To explicitly create a new content canvas, perform the following steps:

1. Click the Canvases node in the Object Navigator.
2. Click the Create icon.
3. A new canvas entry displays with a default name of CANVAS*XX*.
4. If the Property Palette is not already displayed, click the new canvas entry and select Tools > Property Palette.
5. Set the canvas properties according to your requirements.

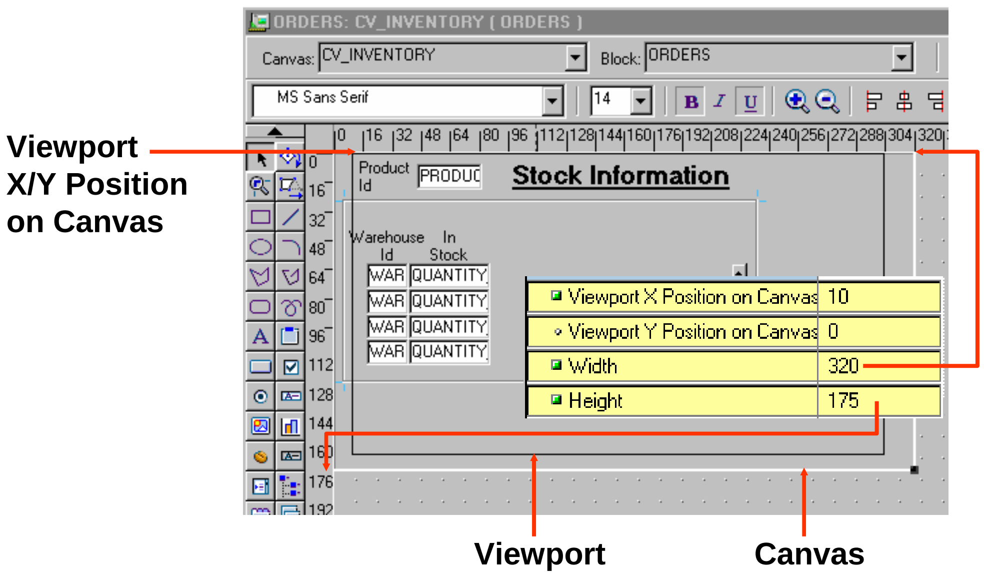
Note: Double-clicking the icon for a canvas in the Object Navigator will invoke the Layout Editor instead of the Property Palette.

Instructor Note

Beginning with Release 6, the Show View option in the View menu of the Layout Editor displays the size of the window for a content canvas.

Demonstration: Display the contents of the `INVENTORIES` data block in the new window created in the previous demonstration.

Setting Content Canvas Properties



Viewport X/Y Position on Canvas
Viewport
Canvas
ORDERS: CV_INVENTORY (ORDERS)
Stock Information

Canvas: CV_INVENTORY
Block: ORDERS

Viewport X Position on Canvas	10
Viewport Y Position on Canvas	0
Width	320
Height	175

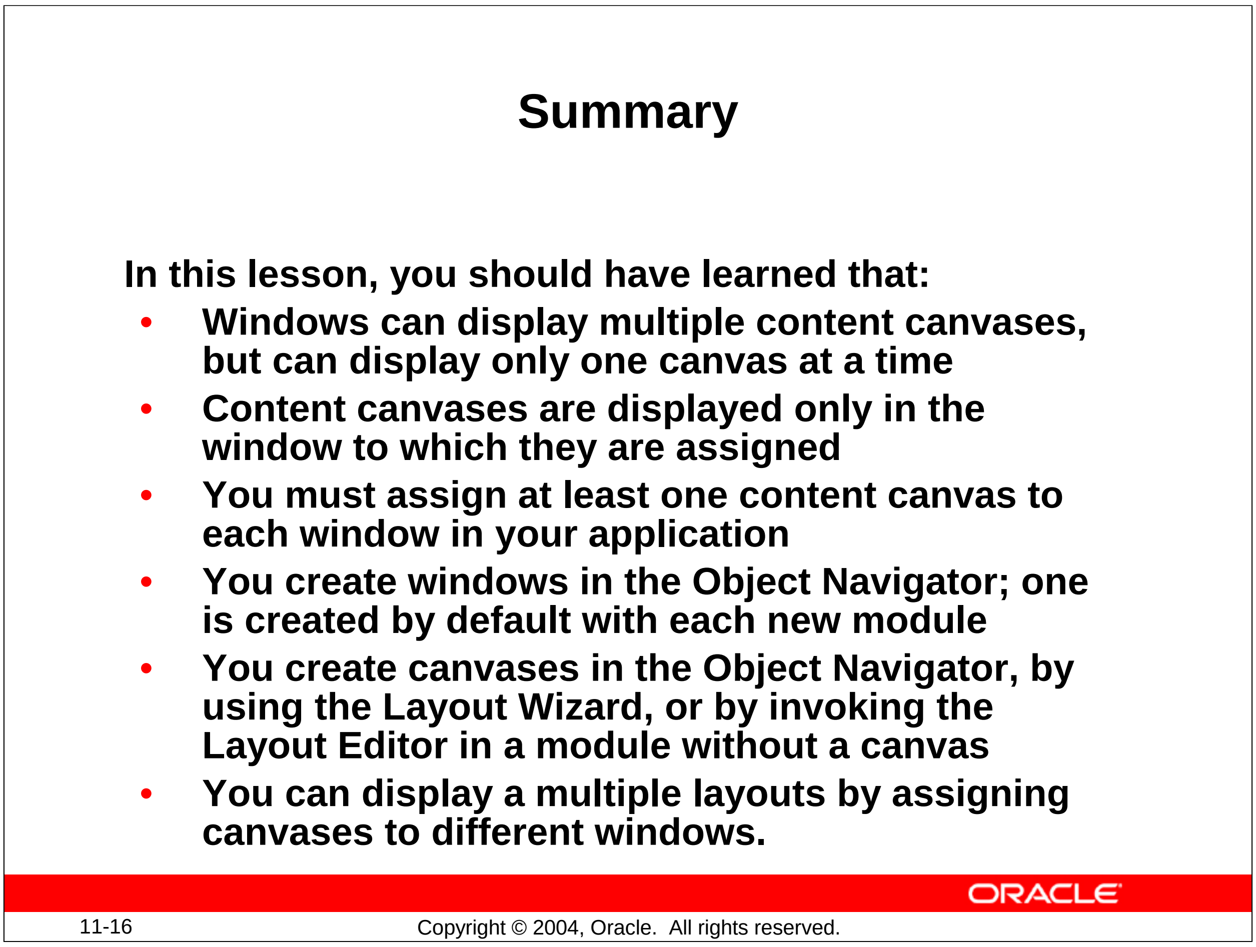
Setting Content Canvas Properties

You can set properties to determine how the canvas is to be displayed. Several properties in the Physical group are depicted in the slide: Width, Height, and Viewport X/Y Position on Canvas. For a canvas to display at run time, its Window property must also be specified.

In the General group, you can choose the Canvas Type. This lesson covers the Content Canvas; other canvas types are presented in the next lesson.

In the Functional group, you can set Raise on Entry to affect whether the canvas is always brought to the front of the window when the user navigates to an item on the canvas. You use this property when the canvas is displayed in the same window with other canvases. Forms always ensures that the current item is visible. Even if Raise on Entry is set to No, Forms will bring the canvas to the front of the window if the user navigates to an item on the canvas that is hidden behind other canvases.

Performance Tip: To reduce the time that is needed to display the initial screen, keep the number of items initially displayed to a minimum. You can hide elements, such as canvases, that are not immediately required. To do this, set the canvas Raise on Entry property to Yes, and set Visible to No.



Summary

In this lesson, you should have learned that:

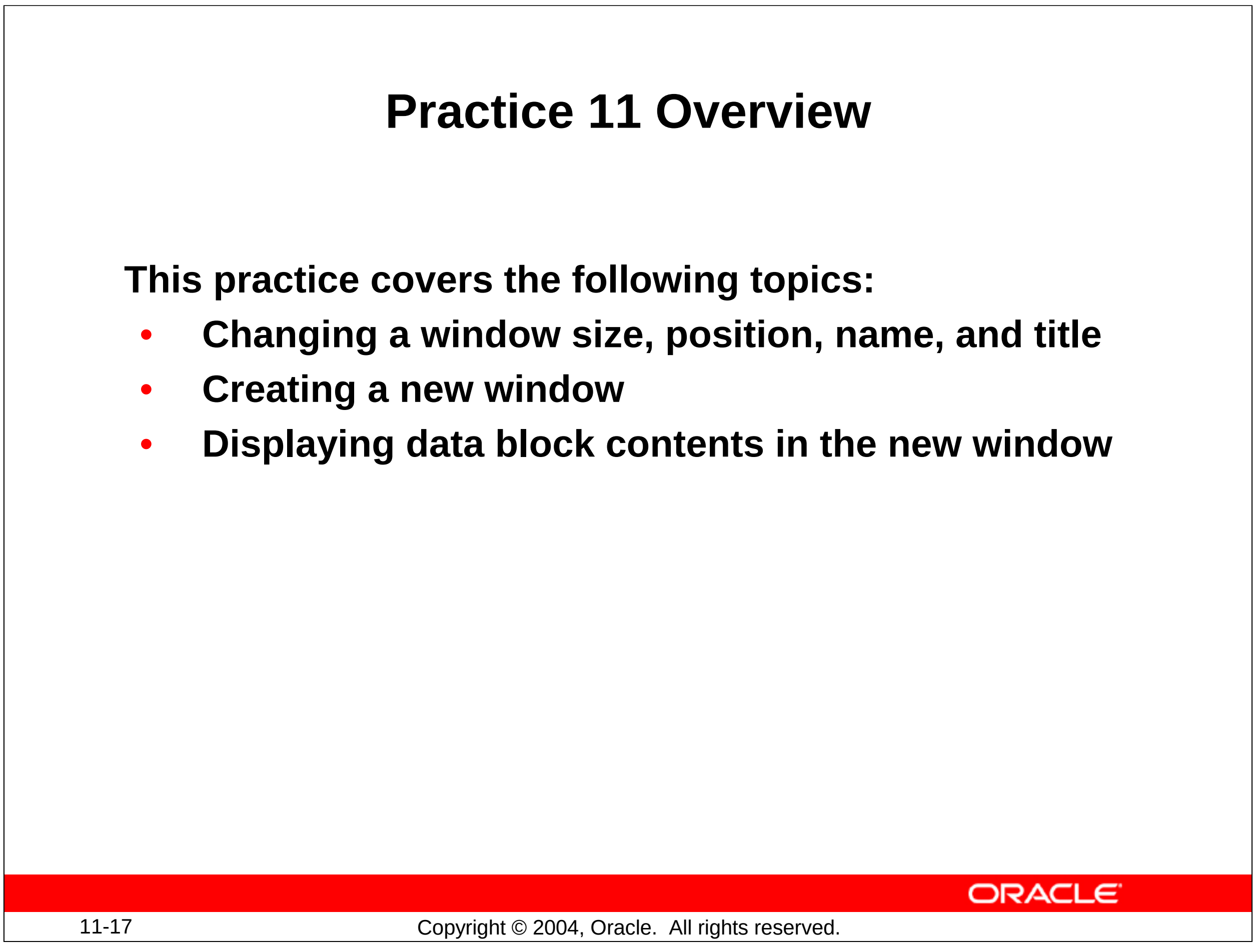
- Windows can display multiple content canvases, but can display only one canvas at a time
- Content canvases are displayed only in the window to which they are assigned
- You must assign at least one content canvas to each window in your application
- You create windows in the Object Navigator; one is created by default with each new module
- You create canvases in the Object Navigator, by using the Layout Wizard, or by invoking the Layout Editor in a module without a canvas
- You can display a multiple layouts by assigning canvases to different windows.

11-16

Summary

In this lesson, you should have learned:

- About the relationship between windows and content canvases
- How to create a new window
- How to create a new content canvas
- How to display a form module on multiple layouts by displaying each canvas in a separate window



Practice 11 Overview

This practice covers the following topics:

- Changing a window size, position, name, and title
- Creating a new window
- Displaying data block contents in the new window

ORACLE

11-17 Copyright © 2004, Oracle. All rights reserved.

Practice 11 Overview

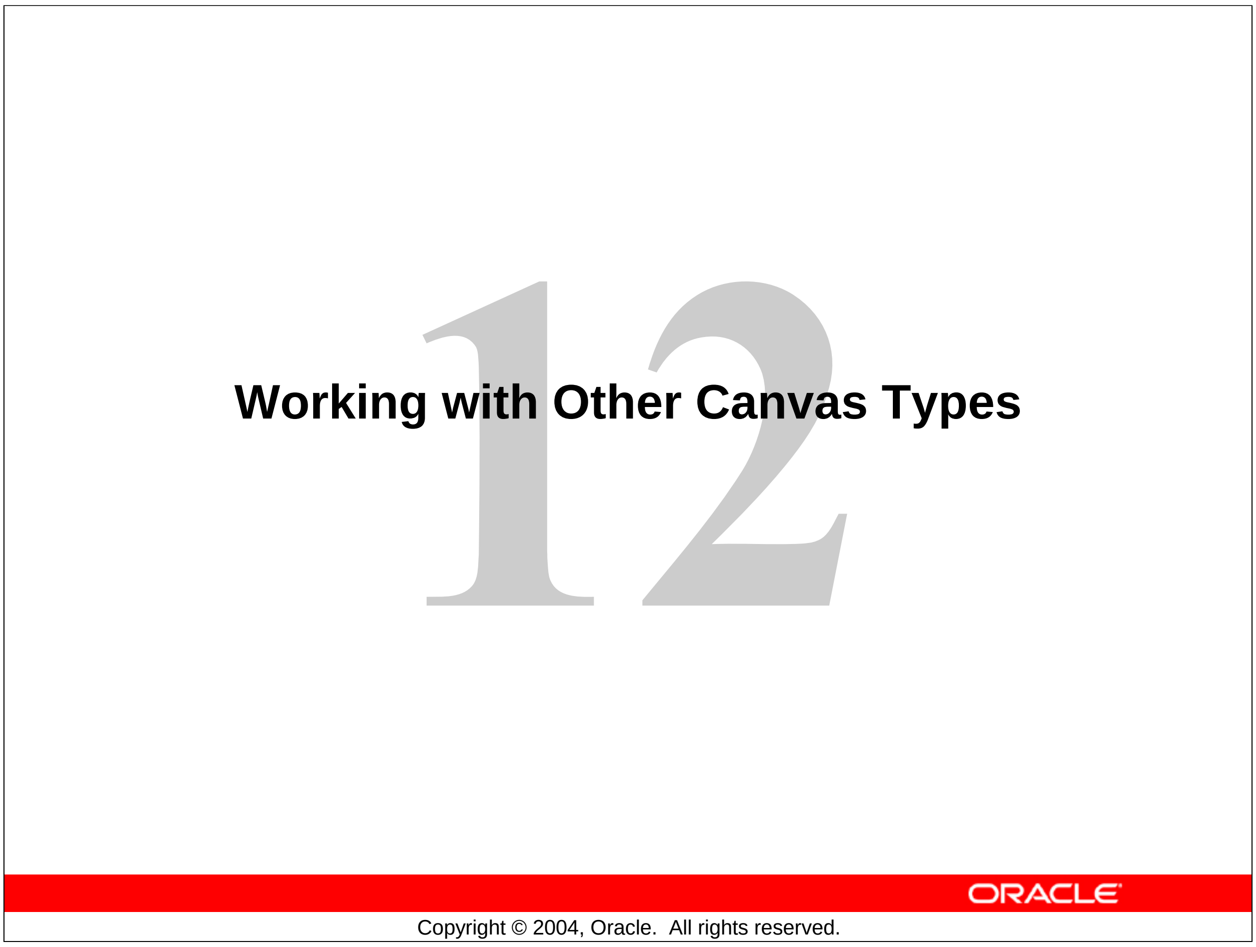
In this practice session, you will customize windows in your form modules. You will resize the windows to make them more suitable for presenting canvas contents. You will also create a new window to display the contents of the `INVENTORIES` block.

- Change the size and position of the window in the CUSTOMERS form. Change its name and title. Save and run the form.
- Modify the name and title of the window in the ORDERS form.
- Create a new window in the ORDERS form. Make sure the contents of the `INVENTORIES` block display in this window. Save and run the form.

Note: For solutions to this practice, see Practice 11 in Appendix A, “Practice Solutions.”

Practice 11

1. Modify the window in the CUSTG*XX* form. Change the name of the window to `WIN_CUSTOMER`, and change its title to Customer Information. Check that the size and position are suitable.
2. Save, compile, and run the form to test the changes.
3. Modify the window in the ORDG*XX* form. Ensure that the window is called `WIN_ORDER`. Also change its title to Orders and Items. Check that the size and position are suitable.
4. In the ORDG*XX* form, create a new window called `WIN_INVENTORY` suitable for displaying the `CV_INVENTORY` canvas. Use the rulers in the Layout Editor to help you plan the height and width of the window. Set the window title to Stock Levels. Place the new window in a suitable position relative to WIN_ORDER. Ensure that the window does not display when the user navigates to an item on a different window.
5. Associate the `CV_INVENTORY` canvas with the window `WIN_INVENTORY`. Compile the form. Click Run Form to run the form. Ensure that the `INVENTORIES` block is displayed in `WIN_INVENTORY` when you navigate to this block. Also make sure that the `WIN_INVENTORY` window is hidden when you navigate to one of the other blocks.
6. Save the form.



Working with Other Canvas Types

Schedule:	Timing	Topic
	45 minutes	Lecture
	40 minutes	Guided Practice
	85 minutes	Total

Objectives

After completing this lesson, you should be able to do the following:

- Describe the different types of canvases and their relationships to each other
- Identify the appropriate canvas type for different scenarios
- Create an overlay effect by using stacked canvases
- Create a toolbar
- Create a tabbed interface

ORACLE

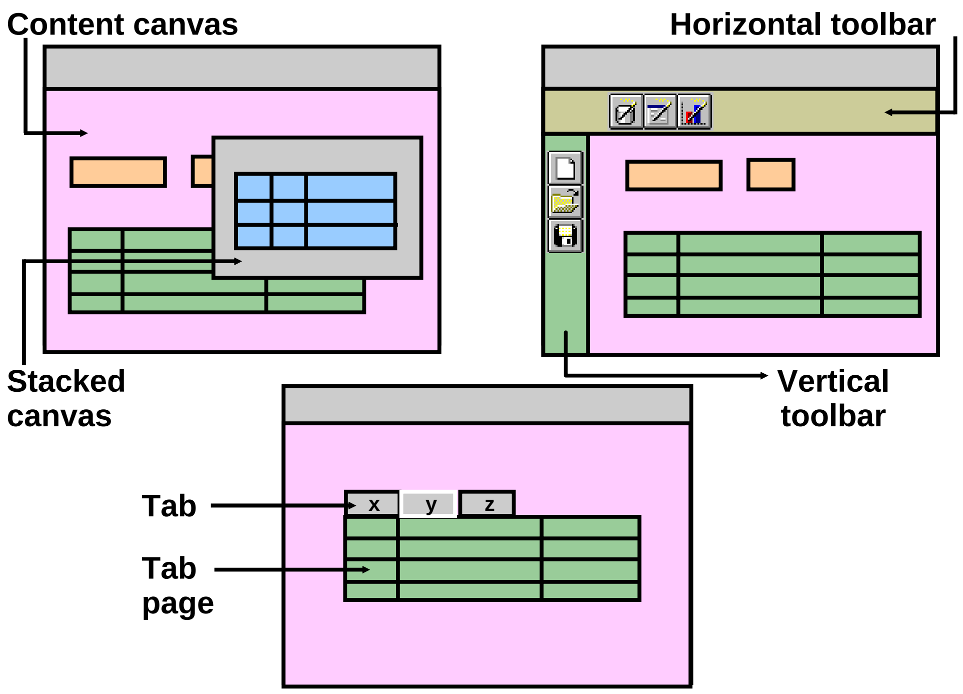
12-2

Introduction

Overview

In addition to the content canvas, Oracle Forms Developer enables you to create three other canvas types. This lesson introduces you to the stacked canvas, which is ideal for creating overlays in your application. It also introduces you to the toolbar canvas and the tabbed canvas, both of which enable you to provide a user-friendly GUI application.

Overview of Canvas Types



Overview of Canvas Types

In addition to the content canvas, Forms Builder provides three other types of canvases that are:

- Stacked canvas
- Toolbar canvas
- Tab canvas

When you create a canvas, you specify its type by setting the Canvas Type property. The type determines how the canvas is displayed in the window to which it is assigned.

Instructor Note

Demonstration

Open the `win_demo.fmb` form and run it.

In the run-time session, move the yellow and purple windows around. In the gray window, point out the gray content canvas and the green toolbar. Scroll the blue and red stacked canvases using the scrollbars. Switch between the different windows by using the mouse. Point out the MDI parent window.

The Stacked Canvas

- Displayed on top of a content canvas
- Shares a window with a content canvas
- Size:
  - Usually smaller than the content canvas in the same window
  - Determined by viewport size
- Created in:
  - Layout Editor
  - Object Navigator

ORACLE

12-4

The Stacked Canvas

What Is a Stacked Canvas?

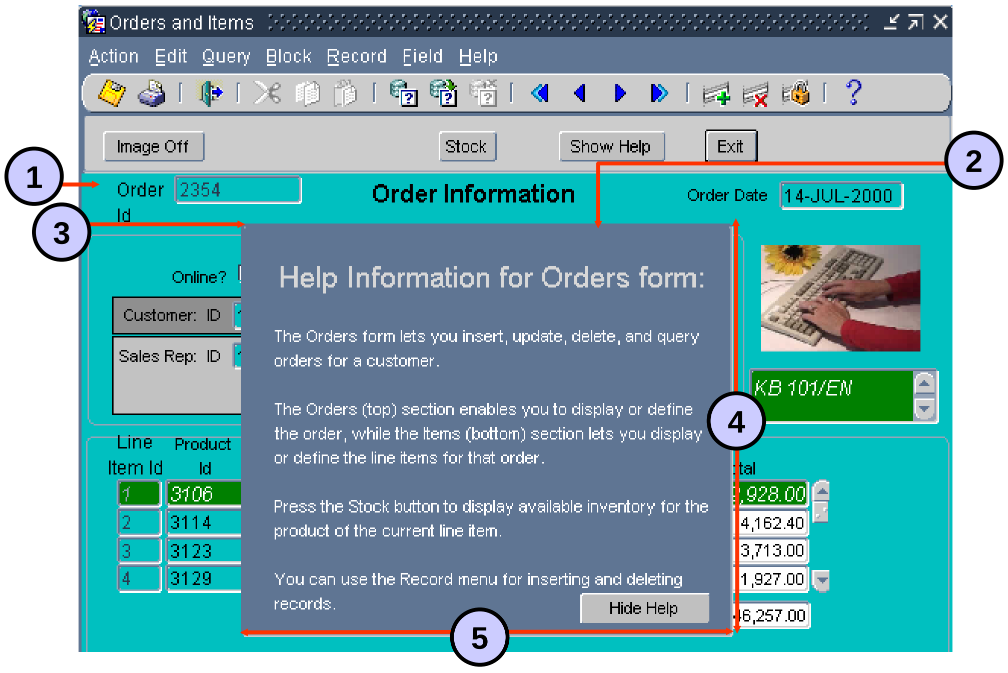
A *stacked canvas* is displayed on top of, or stacked on, the content canvas assigned to a window. It shares a window with a content canvas and any number of other stacked canvases. Stacked canvases are usually smaller than the window in which they display.

Determining the size of a Stacked Canvas

Stacked canvases are typically smaller than the content canvas in the same window. You can determine the stacked canvas dimensions by setting the Width and Height properties. You can determine the view dimensions of the stacked canvas by setting the Viewport Width and Viewport Height properties.

Typical Usage of a Stacked Canvas: If a data block contains more items than the window can display, Forms scrolls the window to display items that were initially not visible. This can cause important items, such as primary key values, to scroll out of view. By placing important items on a content canvas, then placing the items that can be scrolled out of sight on a stacked canvas, the stacked canvas becomes the scrolling region, rather than the window itself.

The Stacked Canvas



Help Information for Orders form:

The Orders form lets you insert, update, delete, and query orders for a customer.

The Orders (top) section enables you to display or define the order, while the Items (bottom) section lets you display or define the line items for that order.

Press the Stock button to display available inventory for the product of the current line item.

You can use the Record menu for inserting and deleting records.

The Stacked Canvas (continued)

The stacked canvas displays on the content canvas. You can see the following elements depicted in the slide:

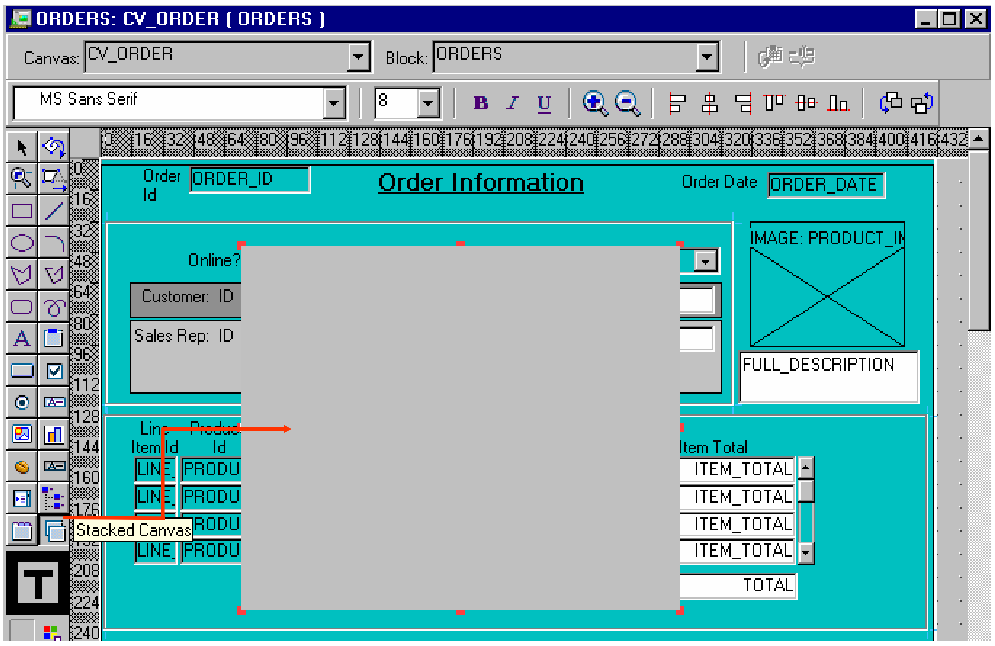
1. Content canvas
2. Stacked canvas
3. Viewport X/Y position
4. Viewport height
5. Viewport width

Uses and Benefits of Stacked Canvases

With stacked canvases you can achieve the following:

- Scrolling views as generated by Oracle Designer
- Creating an overlay effect within a single window
- Displaying headers with constant information, such as company name
- Creating a cascading or a revealing effect within a single window
- Displaying additional information
- Displaying information conditionally
- Displaying context-sensitive help
- Hiding information

Creating a Stacked Canvas



Creating a Stacked Canvas

You can create a stacked canvas in either of the following:

- Layout Editor
- Object Navigator

How to Create a Stacked Canvas in the Layout Editor

To create a stacked canvas in the Layout Editor, perform the following steps:

1. In the Object Navigator, double-click the object icon for the content canvas on which you wish to create a stacked canvas.
   The Layout Editor is displayed.
2. Click the Stacked Canvas tool in the toolbar.
3. Click and drag the mouse in the canvas where you want to position the stacked canvas.
4. Open the Property Palette of the stacked canvas. Set the canvas properties according to your requirements (described later in this lesson).

Creating a Stacked Canvas (continued)

How to Create a Stacked Canvas in the Object Navigator

To create a stacked canvas in the Object Navigator, perform the following steps:

1. Click the Canvases node in the Object Navigator.
2. Click the Create icon.
   A new canvas entry displays with a default name of CANVASXX.
3. If the Property Palette is not already displayed, click the new canvas entry and select Tools > Property Palette.
4. Set the Canvas Type property to Stacked. Additionally, set the properties that are described later in this lesson according to your requirements.
5. Ensure that the stacked canvas is below the content canvas in the Object Navigator.

Note: To convert an existing content canvas to a stacked canvas, change its Canvas Type property value from Content to Stacked.

In order for the stacked canvas to display properly at run time, make sure that its position in the stacking order places it in front of the content canvas assigned to the same window. The stacking order of canvases is defined by the sequence in which they appear in the Object Navigator.

Displaying Stacked Canvases in the Layout Editor

You can display a stacked canvas as it sits over the content canvas in the Layout Editor. Check the display position of stacked canvases by doing the following:

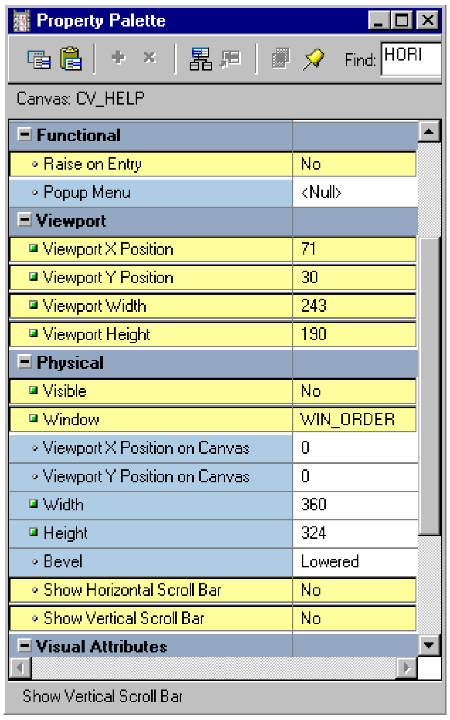
- Select View > Stacked Views in the Layout Editor. The Stacked/Tab Canvases dialog box is displayed, with a list of all the stacked canvases assigned to the same window as the current content canvas.
- Select the stacked canvases you want to display in the Layout Editor.

Note: [Ctrl] + click to clear a stacked canvas that was previously selected.

Instructor Note

Demonstration: Create a stacked canvas. Show the stacked canvas in the Layout Editor. Show the stacked canvas at run time.

Setting Stacked Canvas Properties



Canvas: CV_HELP

Functional	
Raise on Entry	No
Popup Menu	<Null>
Viewport	
Viewport X Position	71
Viewport Y Position	30
Viewport Width	243
Viewport Height	190
Physical	
Visible	No
Window	WIN_ORDER
Viewport X Position on Canvas	0
Viewport Y Position on Canvas	0
Width	360
Height	324
Bevel	Lowered
Show Horizontal Scroll Bar	No
Show Vertical Scroll Bar	No
Visual Attributes	

Show Vertical Scroll Bar

ORACLE

12-8 Copyright © 2004, Oracle. All rights reserved.

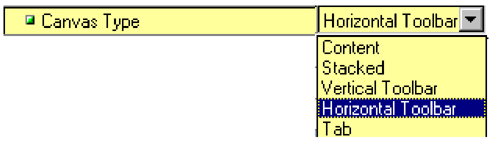
Setting Stacked Canvas Properties

There are several properties you can set to affect the appearance and behavior of stacked canvases:

- **Raise on Entry:** Controls how Forms displays the canvas when the user or the application navigates to an item on the canvas
- **Viewport X/Y Position:** Specifies the point at which the upper left corner of the stacked canvas is located in relation to the content canvas
- **Viewport Width/Height:** Determines the size of the stacked canvas displayed at run time
- **Visible:** Indicates whether the stacked canvas is initially displayed or visible; set to Yes or No to show or hide the stacked canvas
- **Window:** Specifies the window in which the canvas will be displayed
- **Show Horizontal/Vertical Scroll Bar:** Specifies whether scrollbars display with the canvas

The Toolbar Canvas

- Special type of canvas for tool items
- Two types:
  - Vertical toolbar
  - Horizontal toolbar
- Provide:
  - Standard look and feel
  - Alternative to menu or function key operation



The Toolbar Canvas

A *toolbar canvas* is a special type of canvas that you can create to hold buttons and other frequently used GUI elements.

Toolbar Types

- **Vertical toolbar:** Use a vertical toolbar to position all your tool items down the left side of your window.
- **Horizontal toolbar:** Use a horizontal toolbar to position all your tool items and controls across the top of your window under the window's menu bar.

Uses and Benefits of Toolbars

Toolbar canvases offer the following advantages:

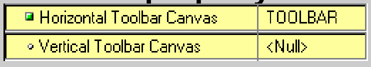
- Provide a standard look and feel across canvases displayed in the same window
- Decrease form module maintenance time
- Increase application usability
- Create applications similar to others used in the same environment
- Provide an alternative to menu or function-key driven applications

The MDI Toolbar

Runtime parameter:

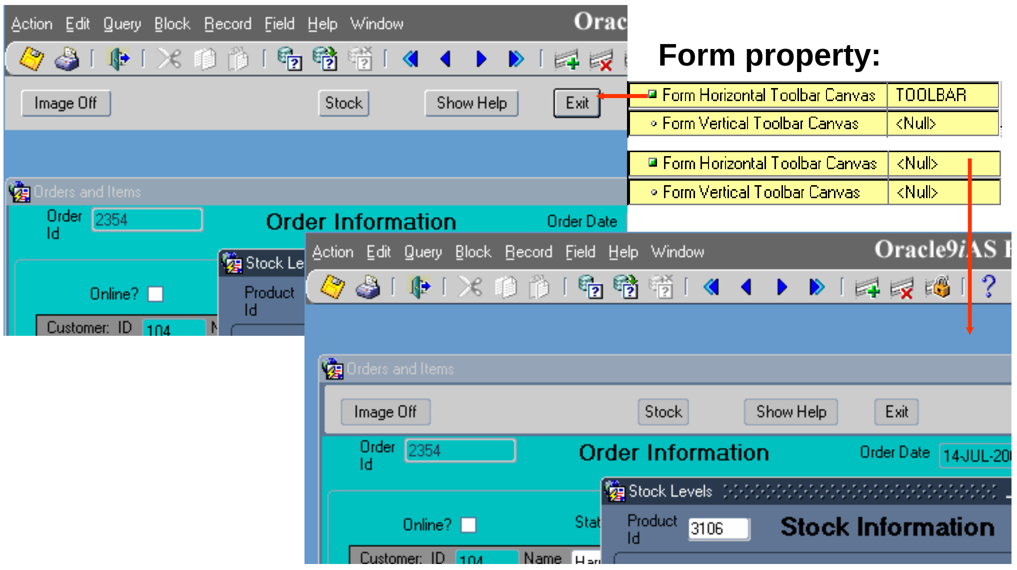
`otherparams=useSDI=no`

Window property:



Horizontal Toolbar Canvas	TOOLBAR
Vertical Toolbar Canvas	<Null>

Form property:



Form Horizontal Toolbar Canvas	TOOLBAR
Form Vertical Toolbar Canvas	<Null>

Form Horizontal Toolbar Canvas	<Null>
Form Vertical Toolbar Canvas	<Null>

The MDI Toolbar

You can attach the toolbar to individual windows, or to the form itself. Attaching a toolbar to a form provides an MDI toolbar, so that you do not need to create more than one toolbar for a Forms application that uses multiple windows. If you display a toolbar in the MDI window, the same toolbar will not be duplicated in the individual windows of the form.

Whether an MDI toolbar is displayed is determined by a combination of Form properties and the `useSDI` run-time parameter, which you set as part of `otherparams`:

Setting	Setting Type	Effect
Form Horizontal or Vertical Toolbar Canvas	Form Property	If useSDI=no, displays MDI toolbar at top or left of MDI window
`otherparams= useSDI=no`	Runtime setting	Displays MDI window and uses MDI toolbar if set as Form property
`otherparams= useSDI=yes`	Runtime setting	MDI window not displayed; Form Horizontal or Vertical Toolbar setting, if any, has no effect.

Creating a Toolbar Canvas

1. Create:
   – Click Create in Object Navigator
   – Change Canvas Type
   – Set other properties as required
2. Add functionality
3. Resize the canvas (not the view)
4. Assign to window and/or form

ORACLE

Creating a Toolbar Canvas

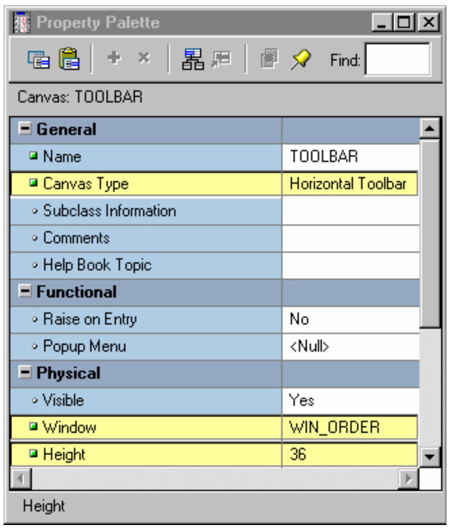
To create a Toolbar canvas, perform the following steps:

1. Create new or modify the existing canvas:
   - With the Canvases node selected, use the Create icon in the Object Navigator.
   - If the Property Palette is not already displayed, click the new canvas entry and select Tools > Property Palette.
   - Set the Canvas Type to either Horizontal or Vertical Toolbar.
2. Add functionality, usually with push buttons. To avoid problems with unwanted navigation, be sure to set the Mouse Navigate property of any toolbar buttons to No.
3. Resize: Use the Layout Editor to resize the toolbar canvas (not just the view) to the smallest size necessary to display its items.
4. Assign to Window or Form: In Property Palette of window or the Form module, set the Horizontal/Vertical Toolbar Canvas properties.

When toolbars are placed on the form or window, the window manager controls shifting the content canvas downward or to the right in order to display the toolbar. You do not need to leave enough space in the content canvas for the placement of the toolbar.

Setting Toolbar Properties

- **Canvas properties:**
  - **Canvas Type**
  - **Window**
  - **Width or Height**
- **Window properties:**
  - **Horizontal Toolbar Canvas**
  - **Vertical Toolbar Canvas**
- **Form Module properties:**
  - **Form Horizontal Toolbar Canvas**
  - **Form Vertical Toolbar Canvas**


Property Palette

Canvas: TOOLBAR

General	
Name	TOOLBAR
Canvas Type	Horizontal Toolbar
Subclass Information	
Comments	
Help Book Topic	
Functional	
Raise on Entry	No
Popup Menu	<Null>
Physical	
Visible	Yes
Window	WIN_ORDER
Height	36

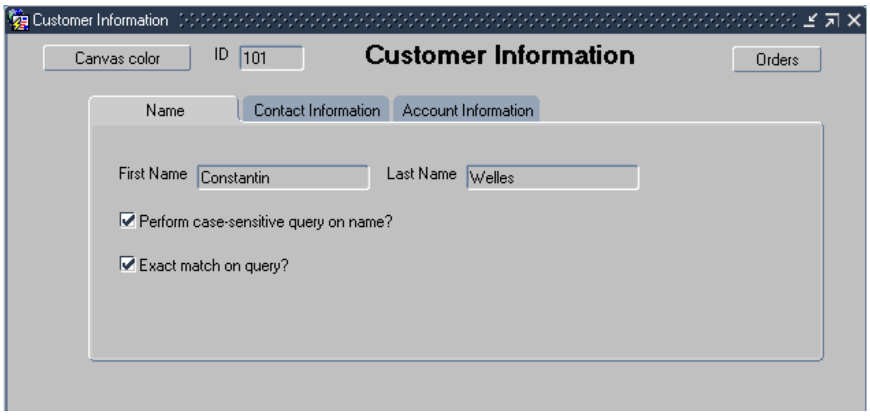
Height

Setting Toolbar Properties

Once you create a toolbar canvas, you can set its properties as well as the properties of the associated window or form module:

- **Canvas Properties:**
  - Canvas Type: Either Horizontal or Vertical Toolbar
  - Window: Specifies in which window the toolbar displays
  - Width/Height: Size of canvas. The width of a horizontal toolbar is set to the width of the window (for example, content canvas). Likewise, the height of a vertical toolbar is set to the height of the window.
- **Window Property:**
  - Horizontal/Vertical Toolbar Canvas: Identifies the horizontal/vertical toolbar to display in this window
- **Form Module Property:**
  - Form Horizontal/Vertical Toolbar Canvas: Identifies the horizontal/vertical toolbar to display in the MDI window (when the useSDI runtime parameter is set to No)

The Tab Canvas



- Enables you to organize and display related information on separate tabs
- Consists of one or more tab pages
- Provides easy access to data

The Tab Canvas

What Is a Tab Canvas?

A *tab canvas* is a a special type of canvas that enables you to organize and display related information on separate tabs. Like stacked canvases, tab canvases are displayed on top of a content canvas.

What Is a Tab Page?

A *tab page* is a subobject of a tab canvas. Each tab canvas is made up of one or more tab pages. A tab page displays a subset of the information in the entire tab canvas. Each tab page has a labeled tab that end users can click to access information on the page.

Each tab page occupies an equal amount of space on the tab canvas.

Uses and Benefits of Tab Canvases

You can use tab canvases to:

- Create an overlay effect within a single window
- Display large amounts of information on a single canvas
- Hide information
- Easily access required information by clicking the tab

Creating a Tab Canvas

- Create in:
  - Object Navigator
  - Layout Editor
- Define tab pages
- Place items on tab pages

12-14

Creating a Tab Canvas

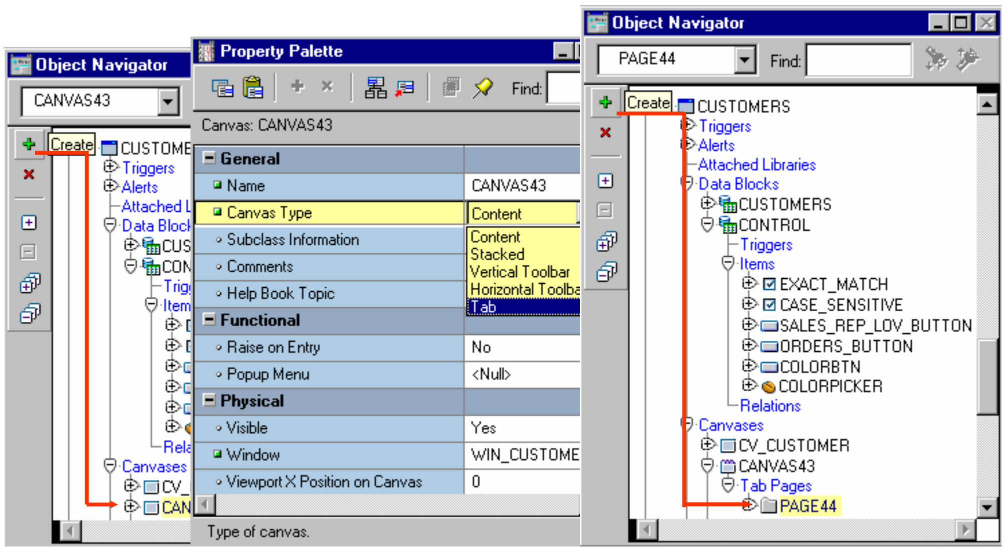
To create a functional tab canvas, you must:

- Create an empty tab canvas in either of the following:
  - Object Navigator
  - Layout Editor
- Define one or more tab pages for the tab canvas.
- Place items on the tab pages.

Instructor Note

Although it is possible to get rid of a content canvas after creating a tab canvas, it is still highly recommended that there be at least one content canvas for each window in your form application.

Creating a Tab Canvas in the Object Navigator



General	
Name	CANVAS43
Canvas Type	Content
Subclass Information	Content / Stacked / Vertical Toolbar / Horizontal Toolba / Tab
Comments	
Help Book Topic	
Functional	
Raise on Entry	No
Popup Menu	<Null>
Physical	
Visible	Yes
Window	WIN_CUSTOME
Viewport X Position on Canvas	0

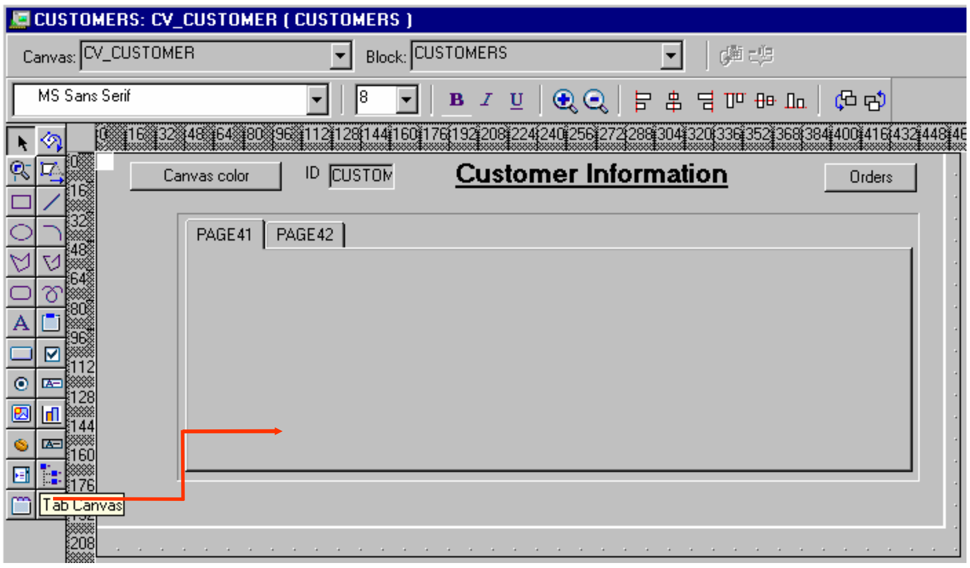
Create new Canvas **Set Canvas Type** **Create Tab Pages**

Creating a Tab Canvas in the Object Navigator

To create a tab canvas in the Object Navigator, perform the following steps:

1. Click the Canvases node in the Object Navigator.
2. Click the Create icon.
   A new canvas entry displays.
3. If the Property Palette is not already displayed, click the new canvas entry and select Tools > Property Palette.
4. Set the Canvas Type property to Tab. Additionally, set the canvas properties according to your requirements (described later in the lesson).
5. Expand the canvas node in the Object Navigator.
   The Tab Pages node displays.
6. Click the Create icon.
   A tab page displays in the Object Navigator, with a default name of PAGE*XX*. The Property Palette takes on its context.
7. Set the tab page properties according to your requirements (described later in the lesson).
8. Create additional tab pages by repeating steps 6 and 7.

Creating a Tab Canvas in the Layout Editor

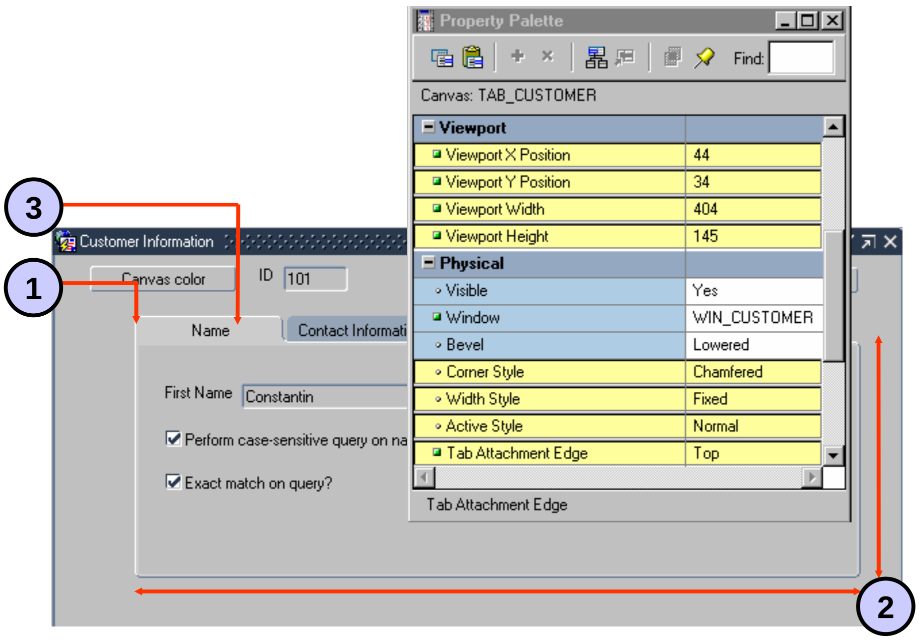


Creating a Tab Canvas in the Layout Editor

To create a tab canvas in the Layout Editor, perform the following steps:

1. In the Object Navigator, double-click the object icon for the content canvas on which you want to create a tab canvas.
   The Layout Editor displays.
2. Click the Tab Canvas tool in the toolbar.
3. Click and drag the mouse in the canvas where you want to position the tab canvas. Forms Builder creates a tab canvas with two tab pages by default.
4. Open the Property Palette of the tab canvas. Set the canvas properties according to your requirements (described later in the lesson).
5. Create additional tab pages, if required, in the Object Navigator.
6. Set the tab page properties according to your requirements (described later in the lesson).

Setting Tab Canvas, Tab Page, and Item Properties



Viewport	
Viewport X Position	44
Viewport Y Position	34
Viewport Width	404
Viewport Height	145
Physical	
Visible	Yes
Window	WIN_CUSTOMER
Bevel	Lowered
Corner Style	Chamfered
Width Style	Fixed
Active Style	Normal
Tab Attachment Edge	Top

Setting Tab Canvas, Tab Page, and Item Properties

Once you create a tab canvas and its tab pages, you must set the required properties for both of these objects. Place items on a tab page by setting the required item properties.

- **Tab Canvas Properties:**
  - Viewport X/Y Position: Specifies the point at which the upper left corner of the tab page is located in relation to the content canvas
  - Viewport Width/Height: Defines the size of tab canvas
  - Corner Style: Specifies the shape of labeled tabs (Chamfered, Square, or Rounded)
  - Tab Attachment Edge: Where tabs are attached (Top, Bottom, Left, Right, Start, or End)
  - Width Style: Whether the width of the tab will vary with the length of the label
  - Active Style: Whether label displays as bold when tab page is active
- **Tab Page Property:** Label specifies label appearing on tab page’s tab
- **Item Properties:**
  - Canvas: The tab canvas on which the item will be displayed
  - Tab Page: The tab page on which the item will be displayed

Placing Items on a Tab Canvas

- Place items on each tab page for user interaction.
- Set the item properties:
  - Canvas
  - Tab Page

ORACLE

Placing Items on a Tab Page

After you create a tab canvas and related tab pages, you place items on the tab pages that the end users can interact with at run time:

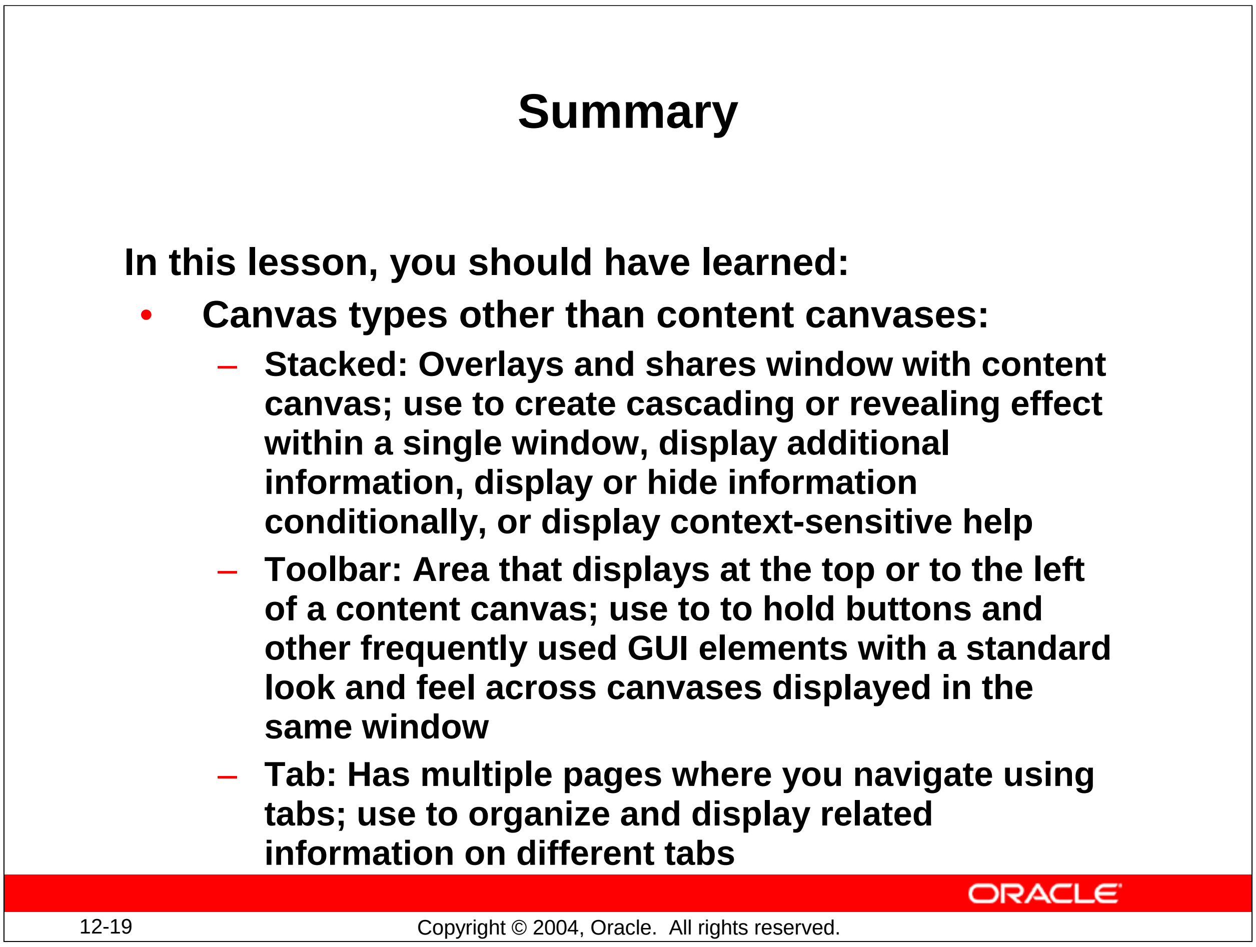
- Open the Property Palette of the item.
- Set the item's Canvas and Tab Page properties of the item to the desired tab canvas and tab page.

Note: Display the tab canvas as it sits on top of the content canvas, by selecting View > Stacked View in the Layout Editor.

Performance Tip: The time taken to load and initialize a TAB canvas at run time is related to all objects on the canvas and not just to those initially visible.

Instructor Note

When you add an item to a tab page, make sure the entire item is completely within the tab page region. You can resize the tab canvas viewport in the Layout Editor, or in the Property Palette of the tab canvas by changing its Viewport Width and Height properties.



Summary

In this lesson, you should have learned:

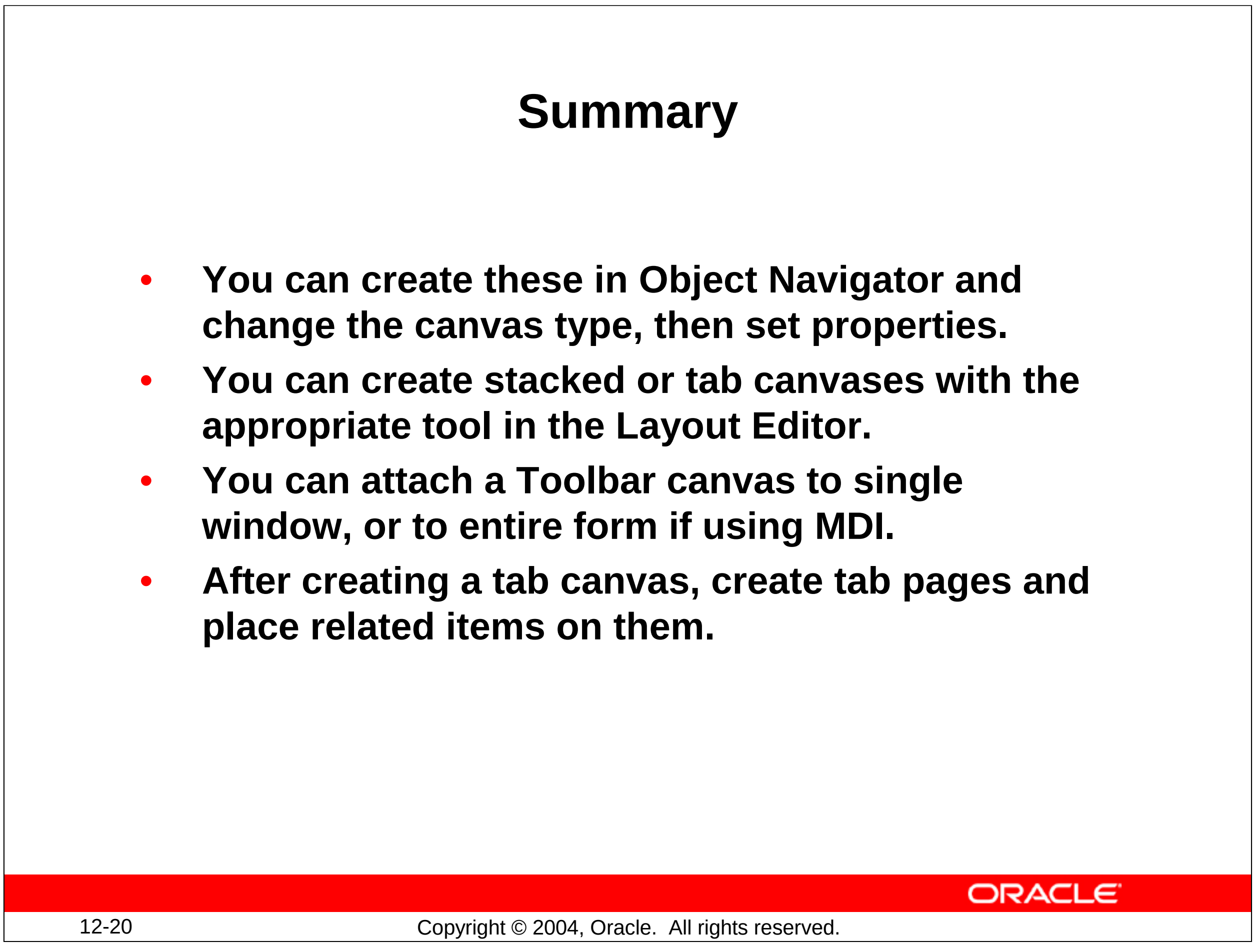
- **Canvas types other than content canvases:**
  - **Stacked: Overlays and shares window with content canvas; use to create cascading or revealing effect within a single window, display additional information, display or hide information conditionally, or display context-sensitive help**
  - **Toolbar: Area that displays at the top or to the left of a content canvas; use to to hold buttons and other frequently used GUI elements with a standard look and feel across canvases displayed in the same window**
  - **Tab: Has multiple pages where you navigate using tabs; use to organize and display related information on different tabs**

12-19

Summary

In this lesson, you should have learned to:

- Create an overlay effect with a stacked canvas
- Create a toolbar
- Create a tabbed canvas

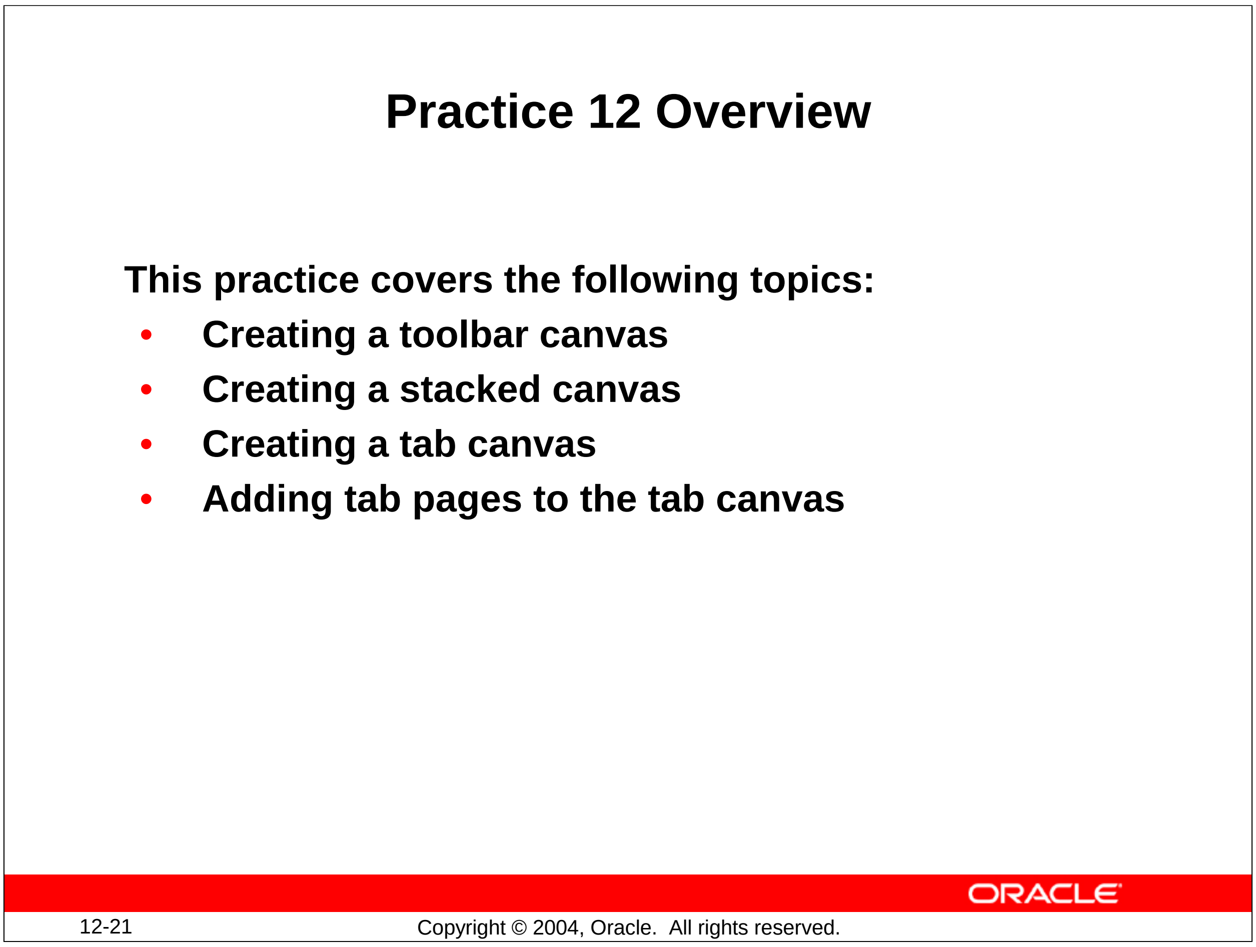


Summary

- You can create these in Object Navigator and change the canvas type, then set properties.
- You can create stacked or tab canvases with the appropriate tool in the Layout Editor.
- You can attach a Toolbar canvas to single window, or to entire form if using MDI.
- After creating a tab canvas, create tab pages and place related items on them.

ORACLE

12-20



Practice 12 Overview

This practice covers the following topics:

- Creating a toolbar canvas
- Creating a stacked canvas
- Creating a tab canvas
- Adding tab pages to the tab canvas

Practice 12 Overview

In this practice session, you will create different types of canvases: stacked canvas, toolbar canvas, and tab canvas.

- Create a horizontal toolbar canvas in the ORDERS form. Create new buttons in the Control block, and place them on the horizontal toolbar. Save and run the form.
- Create a stacked canvas in the ORDERS form to add some help text. Position the canvas in the center of the window. Create a button in the Control block. This button will be used later to display the stacked canvas. Add help text on the stacked canvas. Save and run the form.
- Create a tab canvas in the CUSTOMERS form. Create three tab pages on this canvas, and make sure that each tab page displays the appropriate information. Save and run the form.

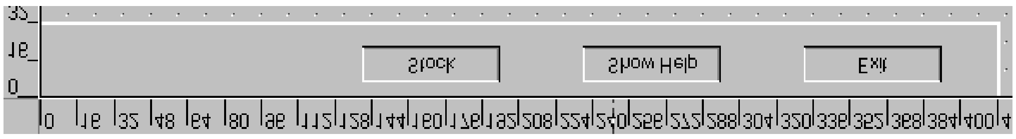
Note: For solutions to this practice, see Practice 12 in Appendix A, “Practice Solutions.”

Practice 12

Toolbar Canvases

1. In the ORDGXX form, create a horizontal toolbar canvas called Toolbar in the `WIN_ORDER` window, and make it the standard toolbar for that window. Suggested height is 30.
2. Save, compile, and run the form to test.
   Notice that the toolbar now uses part of the window space. Adjust the window size accordingly.
3. Create three push buttons in the `CONTROL` block, as shown in the following table, and place them on the Toolbar canvas. Resize the Toolbar if needed.
   Suggested positions for the push buttons are shown in the illustration.

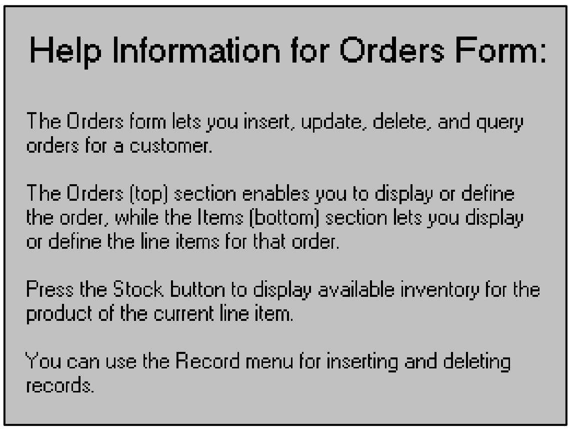
Push Button Name	Details
Stock_Button	Label: Stock Mouse Navigate: No Keyboard Navigable: No Canvas: Toolbar Height: 16 Background Color: white
Show_Help_Button	Label: Show Help Mouse Navigate: No Keyboard Navigable: No Canvas: Toolbar Height: 16 Background Color: white
Exit_Button	Label: Exit Mouse Navigate: No Keyboard Navigable: No Canvas: Toolbar Height: 16 Background Color: white



Practice 12 (continued)

Stacked Canvases

4. Create a stacked canvas named `CV_HELP` to display help in the `WIN_ORDER` window of the ORDG*XX* form. Suggested *visible* size is Viewport Width 250, Viewport Height 225 (points). Select a contrasting color for the canvas. Place some application help text on this canvas, similar to that shown:



Help Information for Orders Form:

The Orders form lets you insert, update, delete, and query orders for a customer.

The Orders (top) section enables you to display or define the order, while the Items (bottom) section lets you display or define the line items for that order.

Press the Stock button to display available inventory for the product of the current line item.

You can use the Record menu for inserting and deleting records.

5. Position the view of the stacked canvas so that it appears in the center of `WIN_ORDER`. Ensure that it will not obscure the first enterable item.
   Do this by planning the top-left position of the view in the Layout Editor, while showing `CV_ORDER`. Define the Viewport X and Viewport Y Positions in the Property Palette. You can move and resize the view in the Layout Editor to set the positions.
6. Organize `CV_HELP` so that it is the last canvas in sequence.
   Do this in the Object Navigator. (This ensures the correct stacking order at run time.)
7. Save and compile the form. Click Run Form to run the form and test the changes. Note that the stacked canvas is initially displayed until it obscures the current item in the form.
8. Switch off the Visible property of `CV_HELP`, and then create a push button in the `CONTROL` block to hide the Help information when it is no longer needed. You will add the code later. Display this push button on the `CV_HELP` canvas.

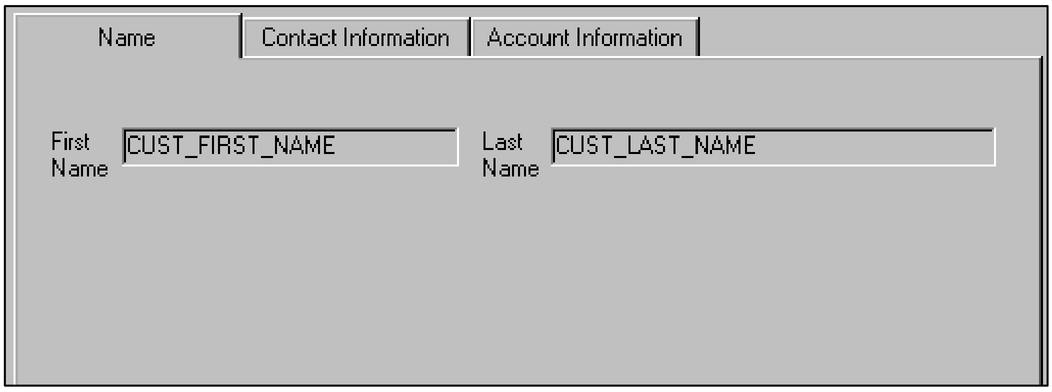
Push Button Name	**Details**
Hide_Help_Button	Label: Hide Help Mouse Navigate: No Keyboard Navigable: Yes Canvas: CV_HELP Width, Height: 65, 16 X, Y Positions: 180, 200

Practice 12 (continued)

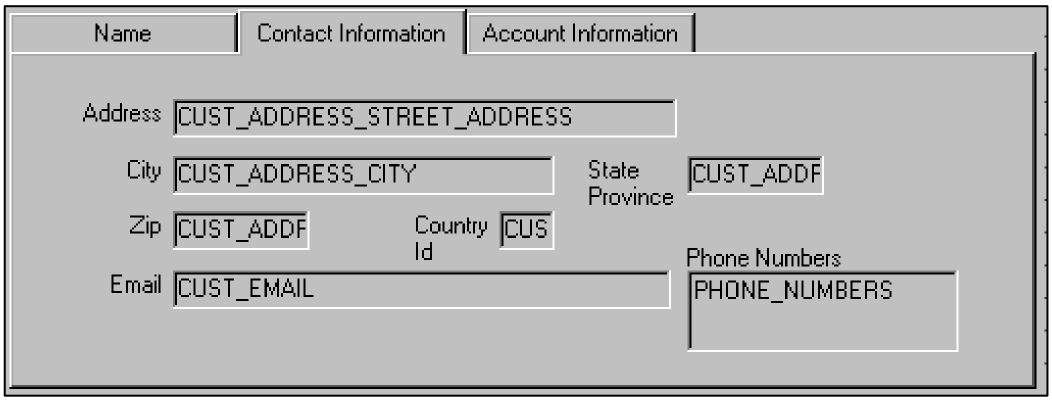
Tab Canvases

Modify the CUSTGXX form in order to use a tab canvas:

9. In the Layout Editor, delete the frame object that surrounds the `CUSTOMERS` block. Create a tab canvas (you may need to first enlarge the content canvas). In the Property Palette, set the Background Color property to gray, Corner Style property to Square, and Bevel property to None.
10. Rename the `TAB_CUSTOMER` tab canvas. Create another tab page in addition to the two tab pages created by default. Name the tab pages as Name, Contact, and Account. Label them as Name, Contact Information, and Account Information.
11. Design the tab pages according to the following screenshots. Set the item properties to make them display on the relevant tab pages.



First Name: CUST_FIRST_NAME
Last Name: CUST_LAST_NAME



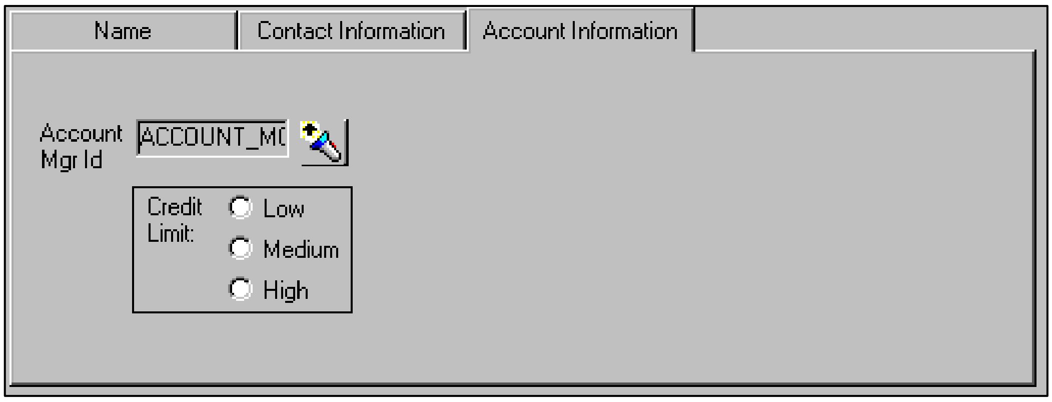
Address: CUST_ADDRESS_STREET_ADDRESS
City: CUST_ADDRESS_CITY
State Province: CUST_ADDF
Zip: CUST_ADDF
Country Id: CUS
Email: CUST_EMAIL
Phone Numbers: PHONE_NUMBERS

Instructor Note

If students appear to “lose” items after setting their Canvas properties, have them set the items’ X and Y positions to 0 to move them to the upper left of the tab page, where they can be seen.

Practice 12 (continued)

Tab Canvases (continued)



12. Reorder the items according to the tab page sequence. Ensure that the user moves smoothly from one tab page to another when tabbing through items. Set the Next Navigation Item and Previous Navigation Item properties to make it impossible to tab to the Customer_Id item once you tab out of that item initially.
    Note: Since Customer_Id is now the only item on the `CV_CUSTOMER` canvas, setting either its Enabled or Keyboard Navigable properties to No has the effect of making the item not visible. This is because Forms must be able to navigate to an item on a canvas in order to display that canvas' items.
13. Save and compile the form. Click Run Form to run the form.