4

Creating Packages

Objectives

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

- Describe packages and list their components
- Create a package to group together related variables, cursors, constants, exceptions, procedures, and functions
- Designate a package construct as either public or private
- Invoke a package construct
- Describe the use of a bodiless package

Lesson Agenda

- Identifying the benefits and the components of packages
- Working with packages:
  - Creating the package specification and body
  - Invoking the package subprograms
  - Displaying the package information
  - Removing a package

What Are PL/SQL Packages?

- A package is a schema object that groups logically related PL/SQL types, variables, and subprograms.
- Packages usually have two parts:
  - A specification (spec)
  - A body
- The specification is the interface to the package. It declares the types, variables, constants, exceptions, cursors, and subprograms that can be referenced from outside the package.
- The body defines the queries for the cursors and the code for the subprograms.
- Enable the Oracle server to read multiple objects into memory at once.

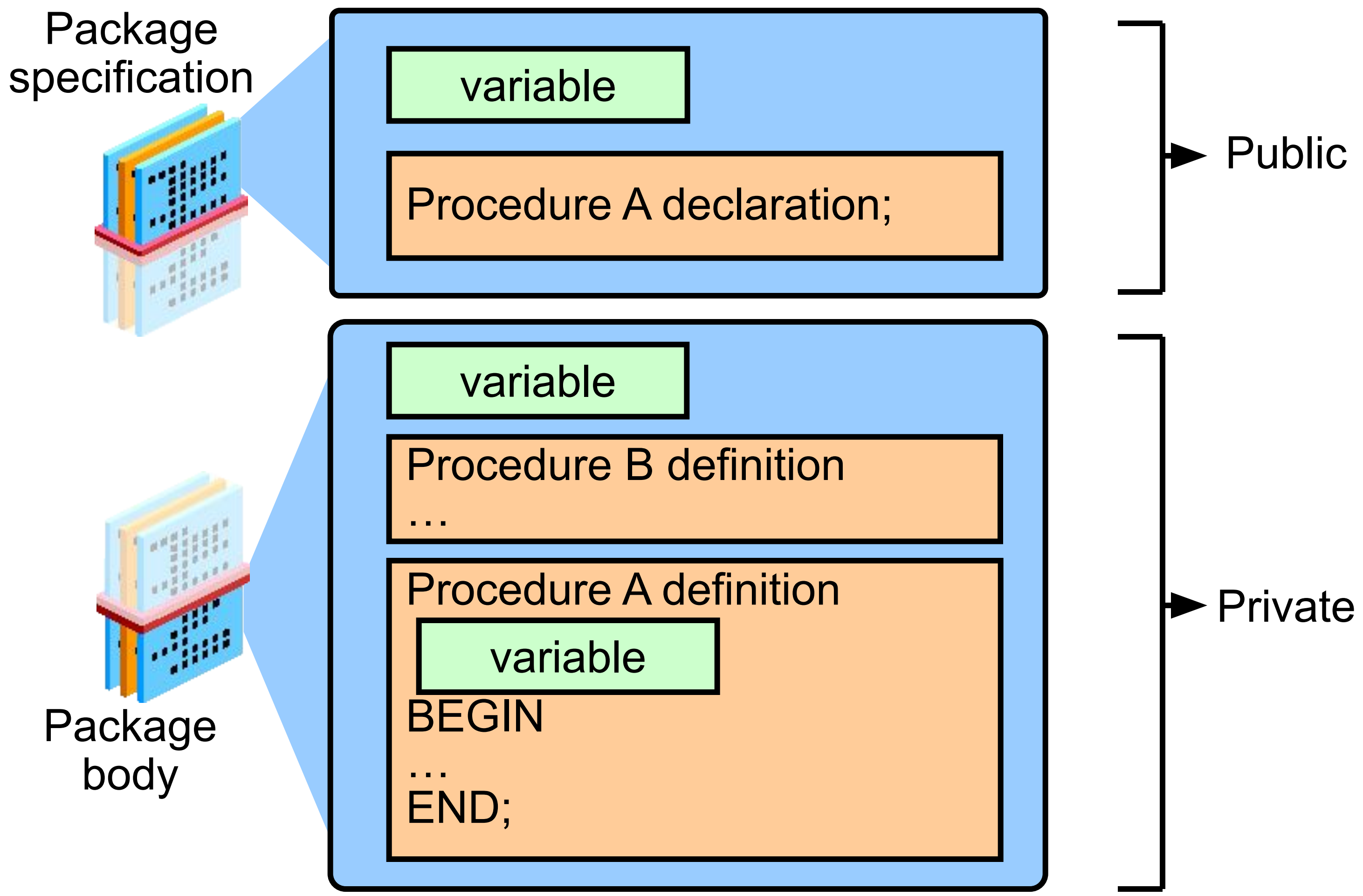
Advantages of Using Packages

- Modularity: Encapsulating related constructs
- Easy maintenance: Keeping logically related functionality together
- Easier application design: Coding and compiling the specification and body separately
- Provision for hiding information:
  - Only the declarations in the package specification are visible and accessible to applications.
  - Private constructs in the package body are hidden and inaccessible.
  - All coding is hidden in the package body.

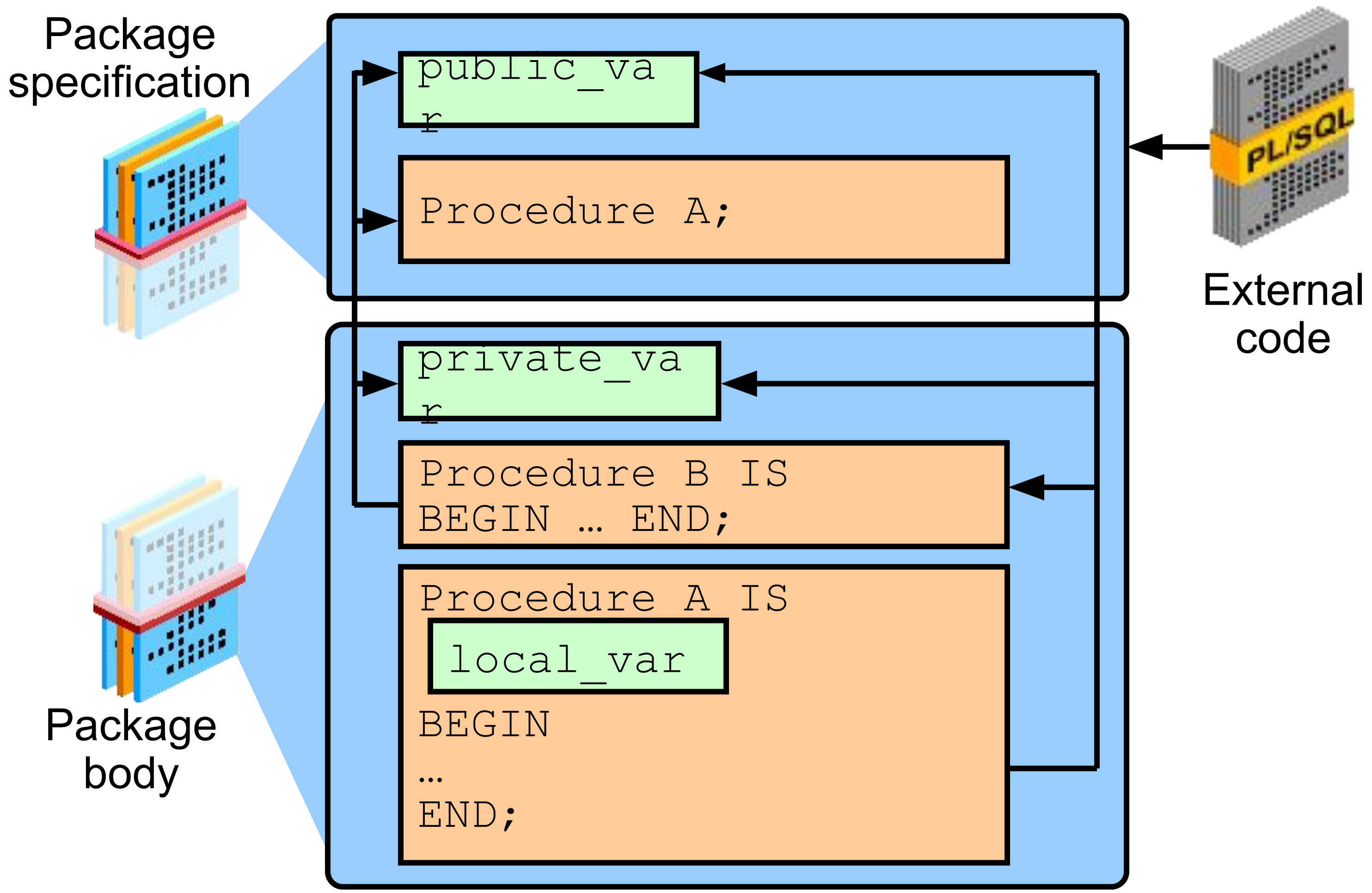
Advantages of Using Packages

- Added functionality: Persistency of public variables and cursors
- Better performance:
  - The entire package is loaded into memory when the package is first referenced.
  - There is only one copy in memory for all users.
  - The dependency hierarchy is simplified.
- Overloading: Multiple subprograms of the same name

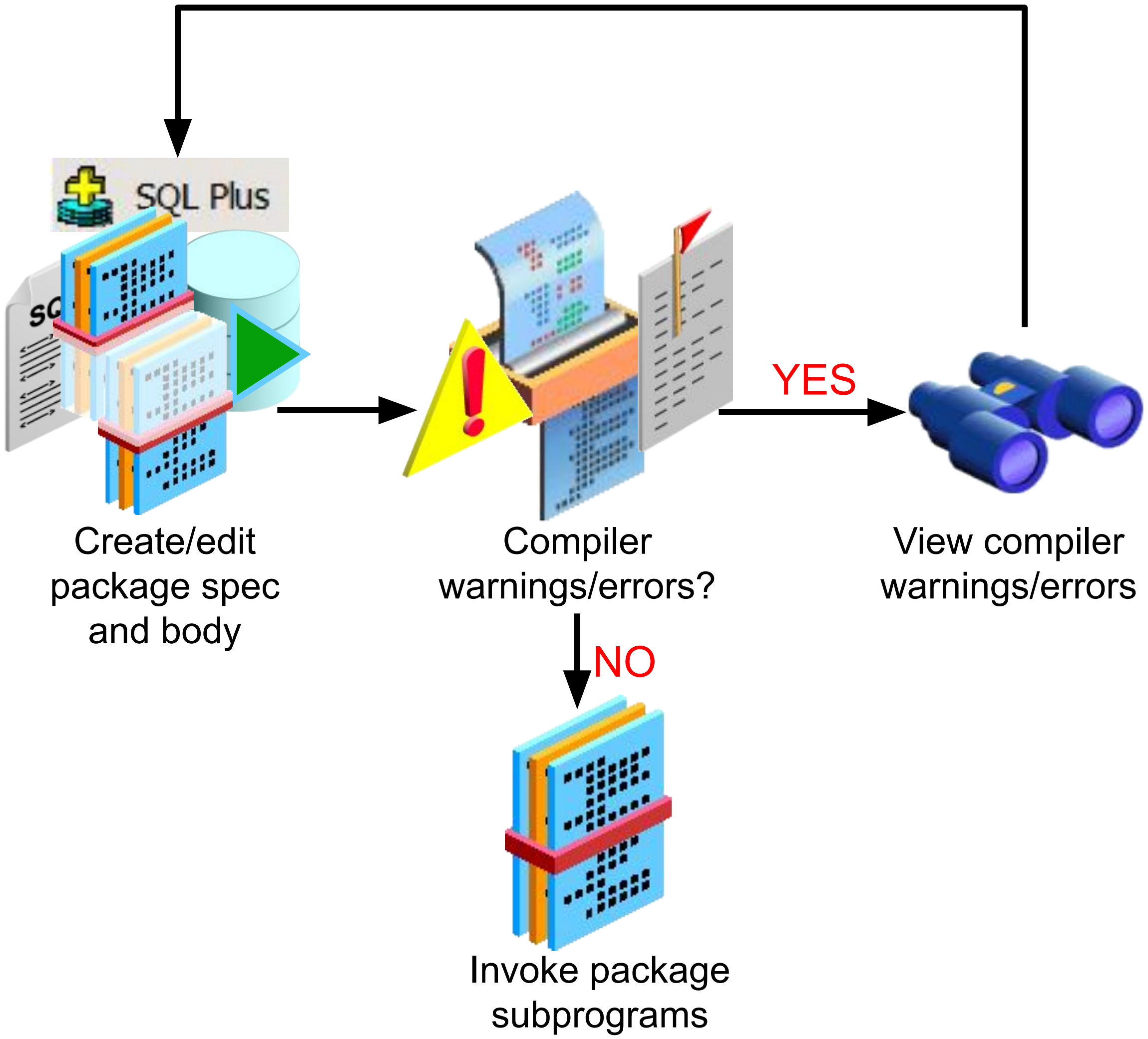
Components of a PL/SQL Package

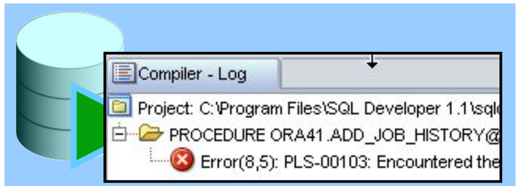


Internal and External Visibility of a Package's Components



Developing PL/SQL Packages: Overview


SQL Plus
Create/edit package spec and body
Compiler warnings/errors?
YES
View compiler warnings/errors
NO
Invoke package subprograms


Compiler - Log
Project: C:\Program Files\SQL Developer 1.1\sql
PROCEDURE ORA41.ADD_JOB_HISTORY@
Error(8,5): PLS-00103: Encountered the

View errors/warnings in SQL Developer

Use `SHOW ERRORS` command in SQL*Plus

Use `USER/ALL/DBA_ ERRORS` views

Lesson Agenda

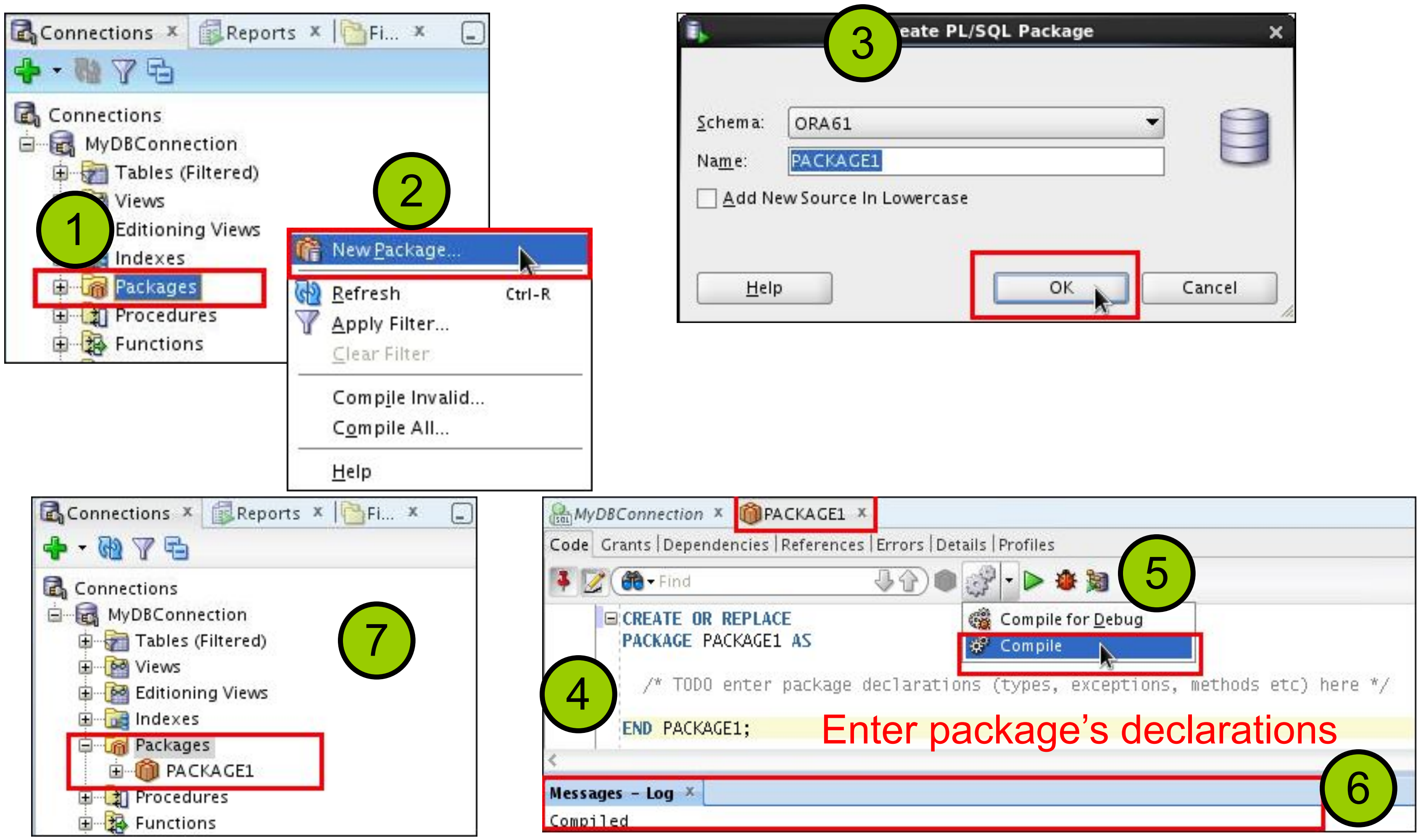
- Identifying the benefits and the components of packages
- Working with packages:
  - Creating the package specification and body
  - Invoking the package subprograms
  - Displaying the package information
  - Removing a package

Creating the Package Specification: Using the `CREATE PACKAGE` Statement

```
CREATE [OR REPLACE] PACKAGE package_name IS|AS
    public type and variable declarations
    subprogram specifications
END [package_name];
```

- The `OR REPLACE` option drops and re-creates the package specification.
- Variables declared in the package specification are initialized to `NULL` by default.
- All the constructs declared in a package specification are visible to users who are granted privileges on the package.

Creating the Package Specification: Using SQL Developer



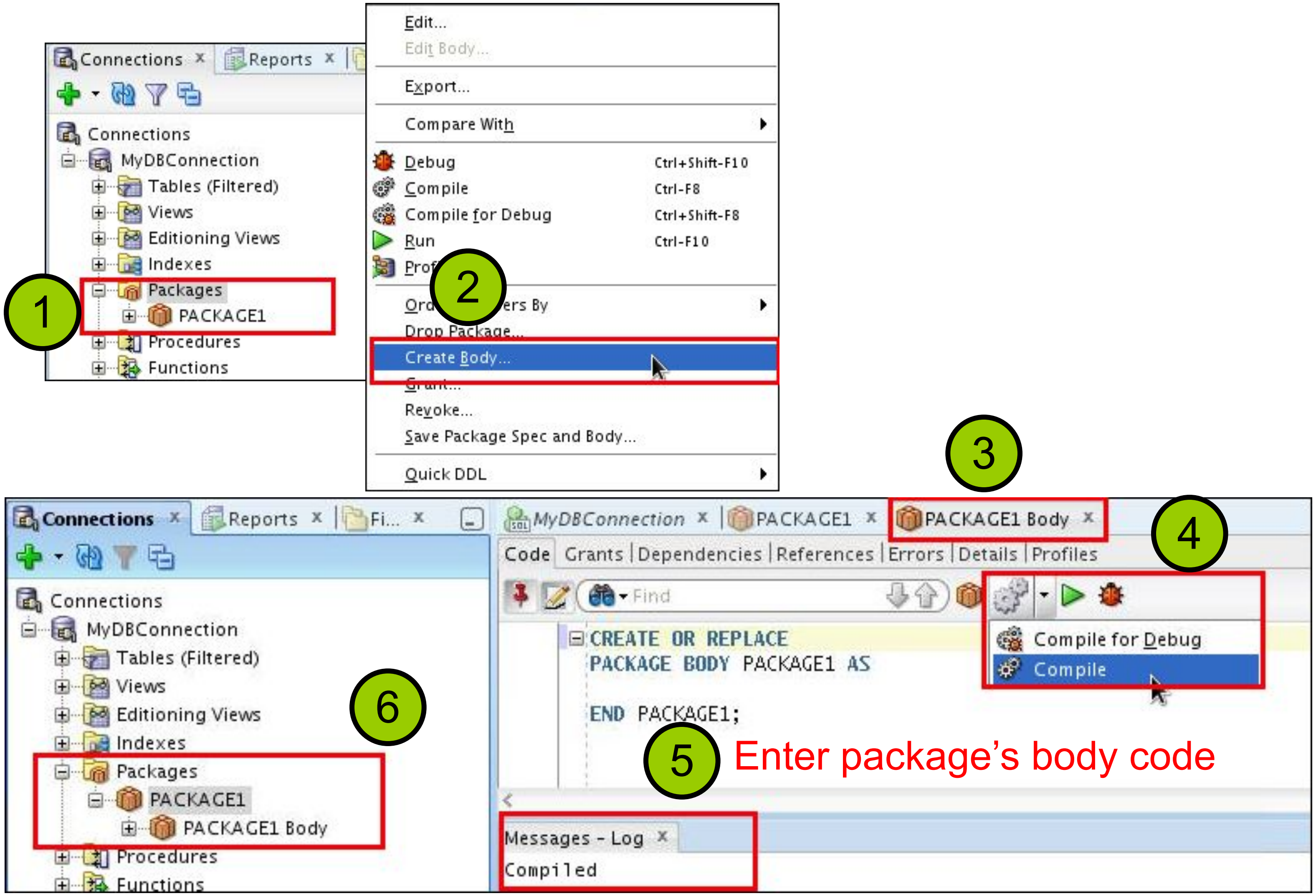
```
CREATE OR REPLACE
PACKAGE PACKAGE1 AS

  /* TODO enter package declarations (types, exceptions, methods etc) here */

END PACKAGE1;
```

```
Compiled
```

Creating the Package Body: Using SQL Developer



```
CREATE OR REPLACE
PACKAGE BODY PACKAGE1 AS

END PACKAGE1;
```

Enter package's body code

```
Compiled
```

Example of a Package Specification: `comm_pkg`

```
-- The package spec with a public variable and a
-- public procedure that are accessible from
-- outside the package.

CREATE OR REPLACE PACKAGE comm_pkg IS
  v_std_comm NUMBER := 0.10;  --initialized to 0.10
  PROCEDURE reset_comm(p_new_comm NUMBER);
END comm_pkg;
/
```

- `V_STD_COMM` is a *public* global variable initialized to `0.10`.
- `RESET_COMM` is a *public* procedure used to reset the standard commission based on some business rules. It is implemented in the package body.

Creating the Package Body

```
CREATE [OR REPLACE] PACKAGE BODY package_name IS|AS
    private type and variable declarations
    subprogram bodies
[BEGIN initialization statements]
END [package_name];
```

- The `OR REPLACE` option drops and re-creates the package body.
- Identifiers defined in the package body are *private* and not visible outside the package body.
- All *private* constructs must be declared before they are referenced.
- Public constructs are visible to the package body.

Example of a Package Body: `comm_pkg`

```
CREATE OR REPLACE PACKAGE BODY comm_pkg IS
  FUNCTION validate(p_comm NUMBER) RETURN BOOLEAN IS
    v_max_comm   employees.commission_pct%type;
  BEGIN
    SELECT MAX(commission_pct) INTO v_max_comm
    FROM    employees;
    RETURN (p_comm BETWEEN 0.0 AND v_max_comm);
  END validate;

  PROCEDURE reset_comm (p_new_comm NUMBER) IS
  BEGIN
    IF validate(p_new_comm) THEN
      v_std_comm := p_new_comm; -- reset public var
    ELSE  RAISE_APPLICATION_ERROR(
            -20210, 'Bad Commission');
    END IF;
  END reset_comm;
END comm_pkg;
```

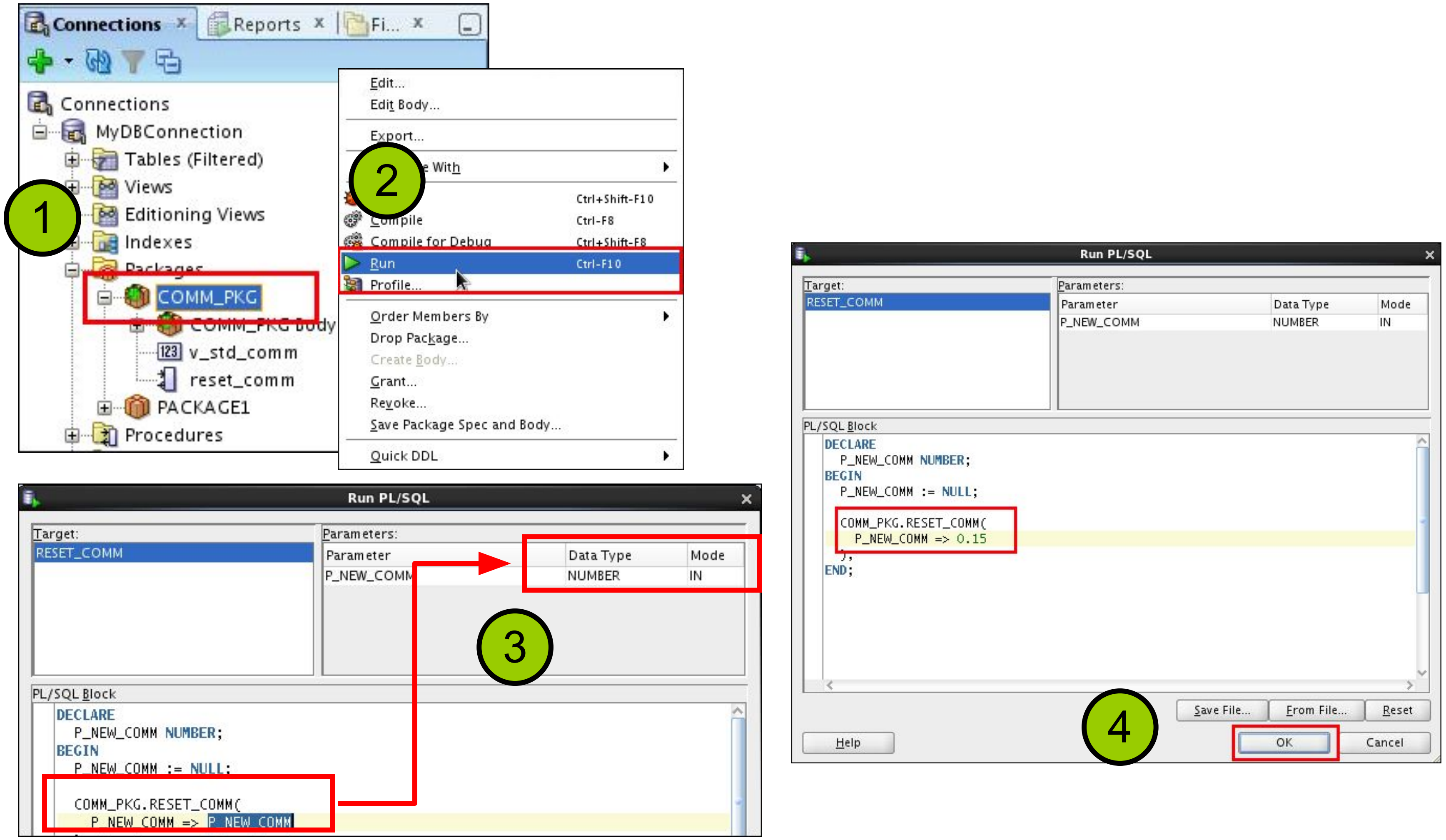
Invoking the Package Subprograms: Examples

```
-- Invoke a function within the same packages:
CREATE OR REPLACE PACKAGE BODY comm_pkg IS ...
  PROCEDURE reset_comm(p_new_comm NUMBER) IS
  BEGIN
    IF validate(p_new_comm) THEN
      v_std_comm := p_new_comm;
    ELSE ...
    END IF;
  END reset_comm;
END comm_pkg;
```

```
-- Invoke a package procedure from SQL*Plus:
EXECUTE comm_pkg.reset_comm(0.15)
```

```
-- Invoke a package procedure in a different schema:
EXECUTE scott.comm_pkg.reset_comm(0.15)
```

Invoking the Package Subprograms: Using SQL Developer



Run PL/SQL

Target:

RESET_COMM

Parameters:

Parameter	Data Type	Mode
P_NEW_COMM	NUMBER	IN

PL/SQL Block

```
DECLARE
  P_NEW_COMM NUMBER;
BEGIN
  P_NEW_COMM := NULL;

  COMM_PKG.RESET_COMM(
    P_NEW_COMM => P_NEW_COMM
```

Run PL/SQL

Target:

RESET_COMM

Parameters:

Parameter	Data Type	Mode
P_NEW_COMM	NUMBER	IN

PL/SQL Block

```
DECLARE
  P_NEW_COMM NUMBER;
BEGIN
  P_NEW_COMM := NULL;

  COMM_PKG.RESET_COMM(
    P_NEW_COMM => 0.15
  );
END;
```

Creating and Using Bodiless Packages

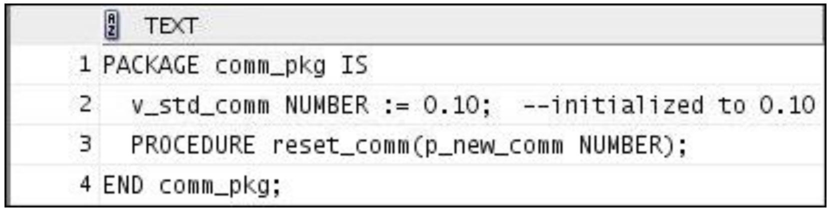
```
CREATE OR REPLACE PACKAGE global_consts IS
  c_mile_2_kilo    CONSTANT  NUMBER  :=  1.6093;
  c_kilo_2_mile    CONSTANT  NUMBER  :=  0.6214;
  c_yard_2_meter   CONSTANT  NUMBER  :=  0.9144;
  c_meter_2_yard   CONSTANT  NUMBER  :=  1.0936;
END global_consts;
```

```
SET SERVEROUTPUT ON
BEGIN
   DBMS_OUTPUT.PUT_LINE('20 miles = ' ||
        20 * global_consts.c_mile_2_kilo || ' km');
END;
```

```
SET SERVEROUTPUT ON
CREATE FUNCTION mtr2yrd(p_m NUMBER) RETURN NUMBER IS
BEGIN
  RETURN (p_m * global_consts.c_meter_2_yard);
END mtr2yrd;
/
EXECUTE DBMS_OUTPUT.PUT_LINE(mtr2yrd(1))
```

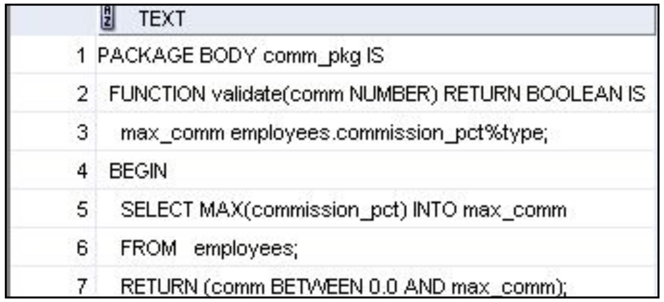
Viewing Packages by Using the Data Dictionary

```
-- View the package specification.
SELECT text
FROM   user_source
WHERE  name = 'COMM_PKG' AND type = 'PACKAGE'
ORDER BY LINE;
```



```
  TEXT
1 PACKAGE comm_pkg IS
2   v_std_comm NUMBER := 0.10;  --initialized to 0.10
3   PROCEDURE reset_comm(p_new_comm NUMBER);
4 END comm_pkg;
```

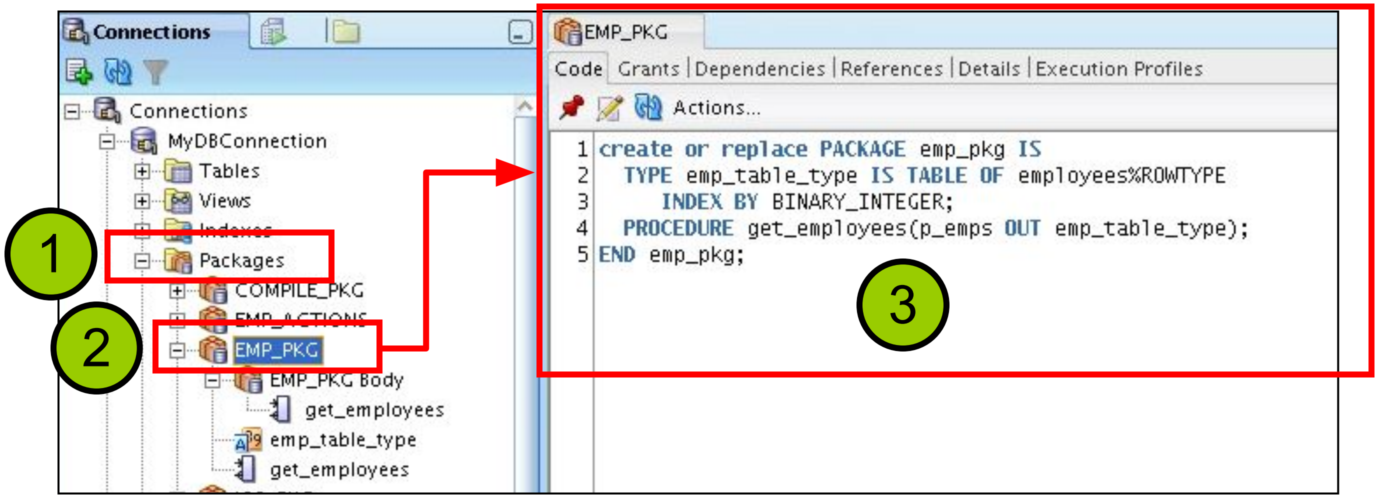
```
-- View the package body.
SELECT text
FROM   user_source
WHERE  name = 'COMM_PKG' AND type = 'PACKAGE BODY'
ORDER BY LINE;
```



```
  TEXT
1 PACKAGE BODY comm_pkg IS
2  FUNCTION validate(comm NUMBER) RETURN BOOLEAN IS
3   max_comm employees.commission_pct%type;
4  BEGIN
5   SELECT MAX(commission_pct) INTO max_comm
6   FROM  employees;
7   RETURN (comm BETWEEN 0.0 AND max_comm);
```

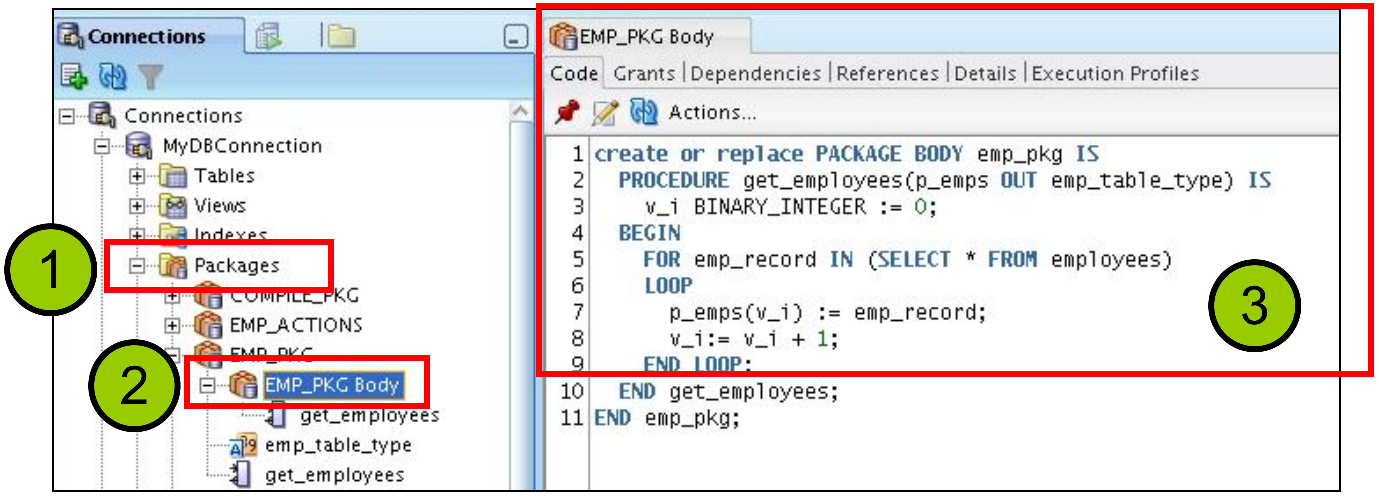
Viewing Packages by Using SQL Developer

To view the package spec, click the package name.



```
create or replace PACKAGE emp_pkg IS
  TYPE emp_table_type IS TABLE OF employees%ROWTYPE
     INDEX BY BINARY_INTEGER;
  PROCEDURE get_employees(p_emps OUT emp_table_type);
END emp_pkg;
```

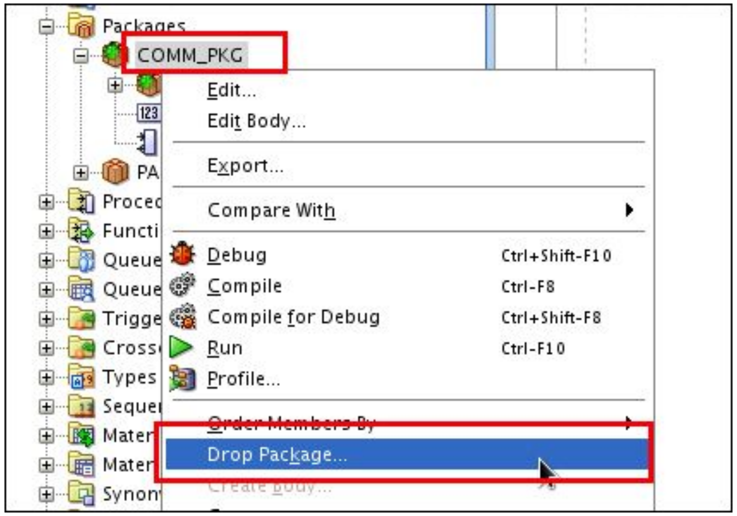
To view the package body, click the package body.



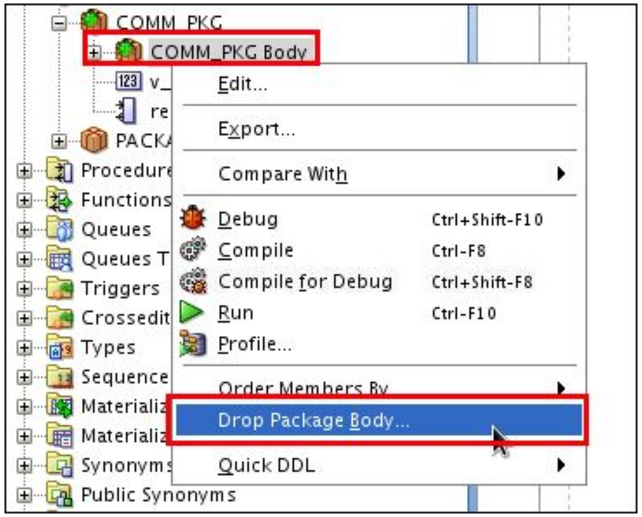
```
create or replace PACKAGE BODY emp_pkg IS
  PROCEDURE get_employees(p_emps OUT emp_table_type) IS
    v_i BINARY_INTEGER := 0;
  BEGIN
    FOR emp_record IN (SELECT * FROM employees)
    LOOP
      p_emps(v_i) := emp_record;
      v_i:= v_i + 1;
    END LOOP;
  END get_employees;
END emp_pkg;
```

Removing Packages by Using SQL Developer or the SQL `DROP` Statement

Drop the package specification and body.



Drop only the package body.



```
-- Remove the package specification and body
DROP PACKAGE package_name;
```

```
-- Remove the package body only
DROP PACKAGE BODY package_name;
```

Guidelines for Writing Packages

- Develop packages for general use.
- Define the package specification before the body.
- The package specification should contain only those constructs that you want to be public.
- Place items in the declaration part of the package body when you must maintain them throughout a session or across transactions.
- The fine-grain dependency management reduces the need to recompile referencing subprograms when a package specification changes.
- The package specification should contain as few constructs as possible.

Quiz

The package specification is the interface to your applications. It declares the public types, variables, constants, exceptions, cursors, and subprograms available for use. The package specification may also include PRAGMAs, which are directives to the compiler.

a. True

b. False

Summary

In this lesson, you should have learned how to:

- Describe packages and list their components
- Create a package to group related variables, cursors, constants, exceptions, procedures, and functions
- Designate a package construct as either public or private
- Invoke a package construct
- Describe the use of a bodiless package

Practice 4 Overview: Creating and Using Packages

This practice covers the following topics:

- Creating packages
- Invoking package program units