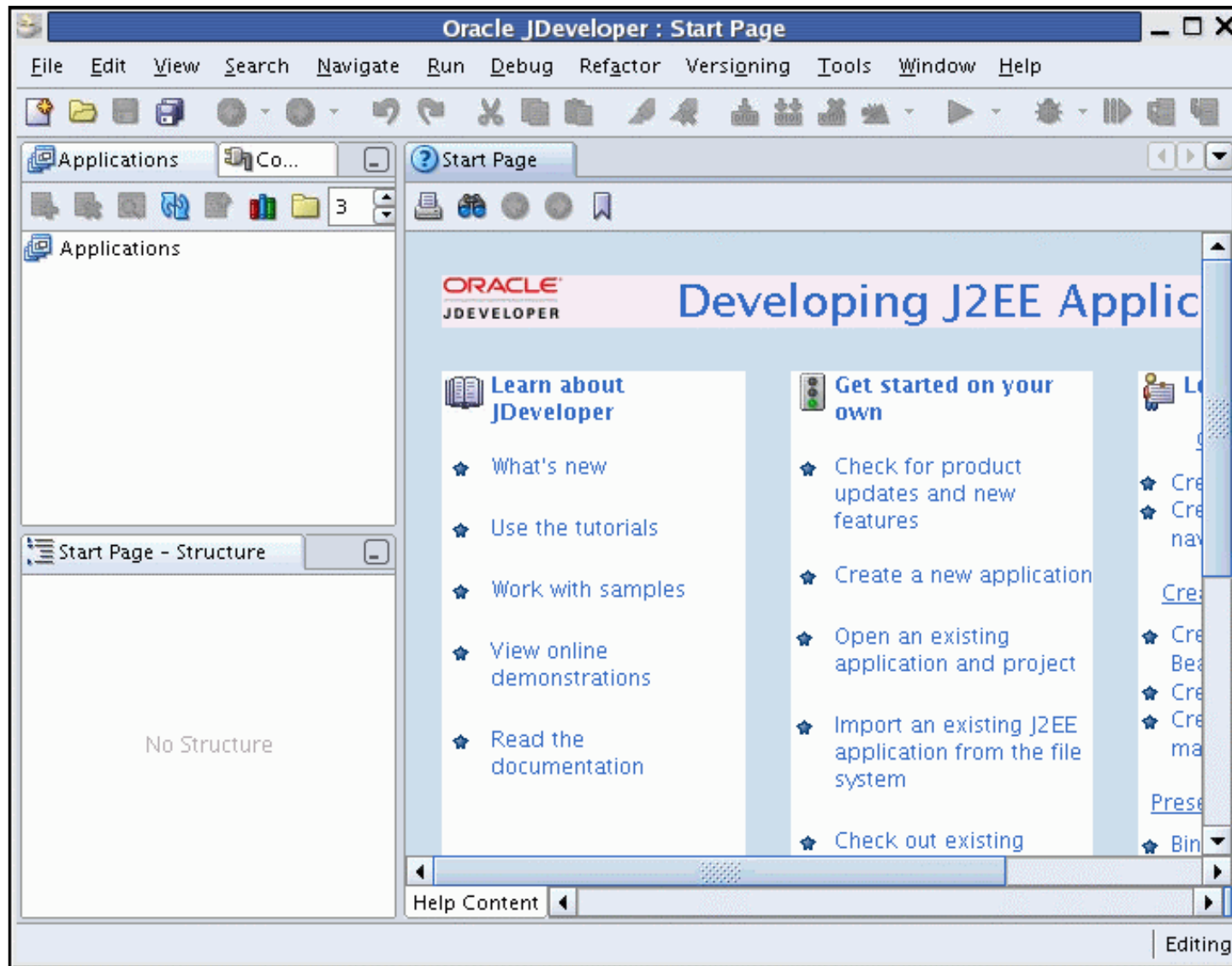


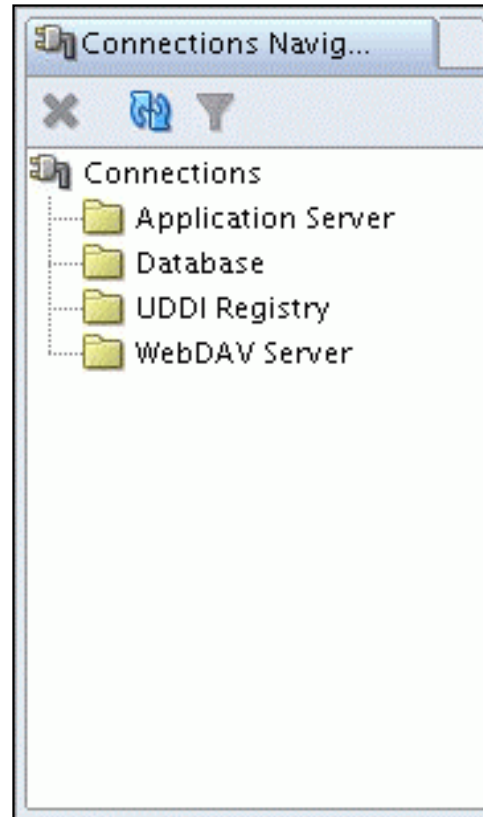


# Oracle JDeveloper

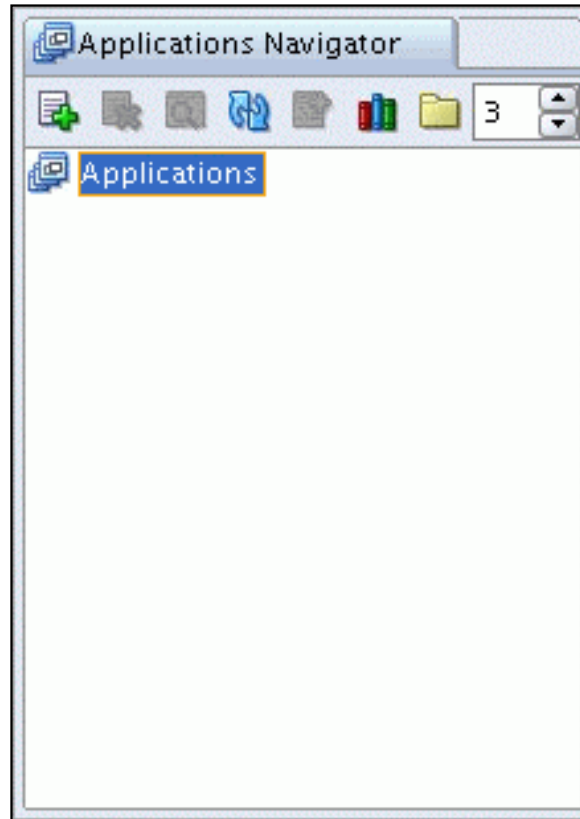
# Oracle JDeveloper 10g



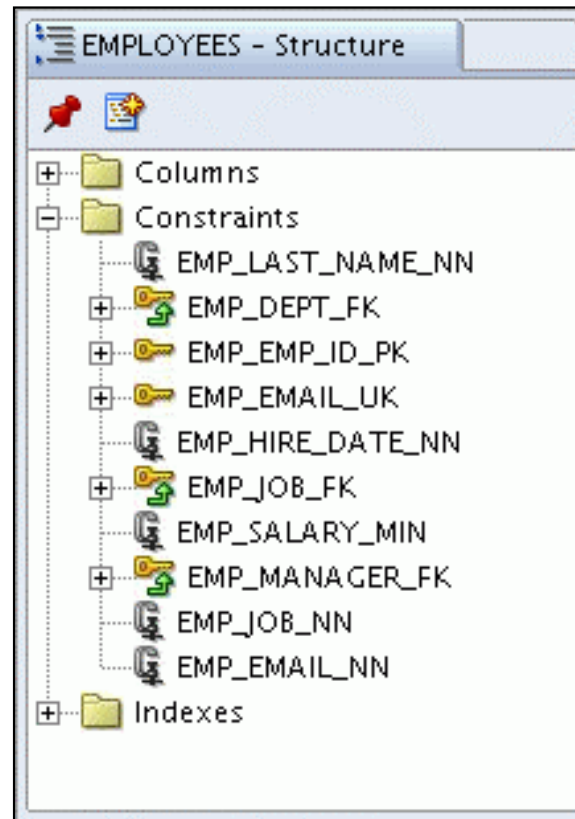
# Connection Navigator



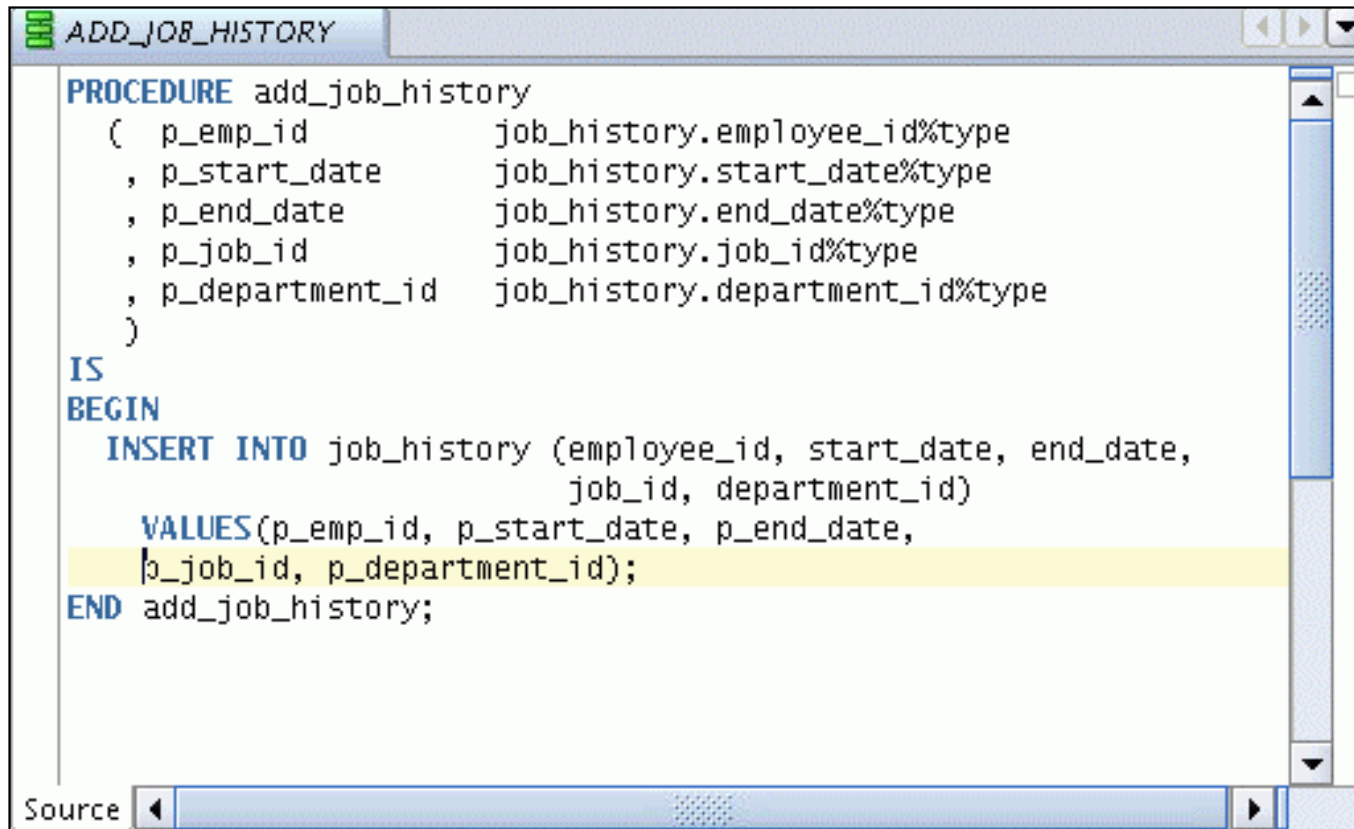
# Application Navigator



# Structure Window



# Editor Window



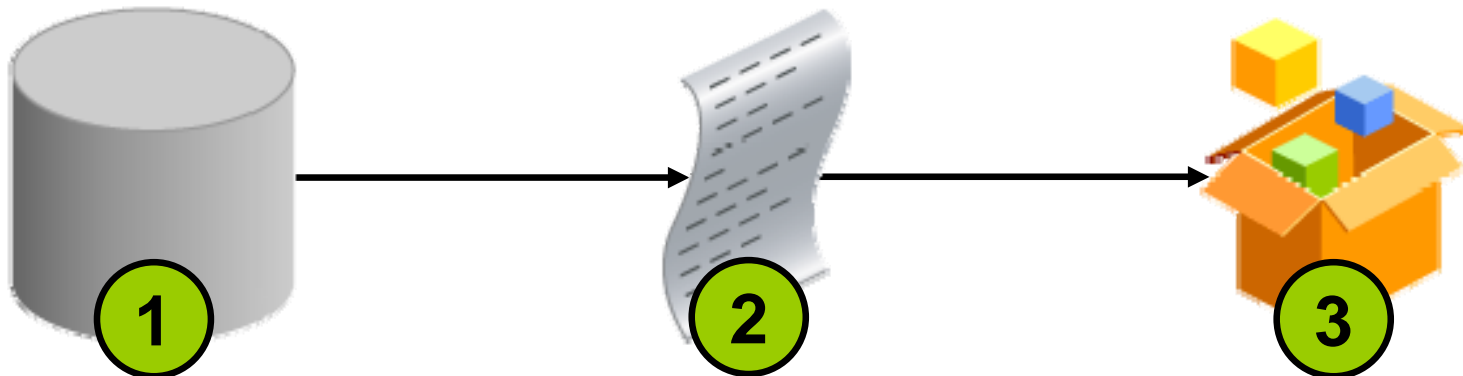
```
ADD_JOB_HISTORY

PROCEDURE add_job_history
(
  p_emp_id          job_history.employee_id%type
, p_start_date      job_history.start_date%type
, p_end_date        job_history.end_date%type
, p_job_id          job_history.job_id%type
, p_department_id   job_history.department_id%type
)
IS
BEGIN
  INSERT INTO job_history (employee_id, start_date, end_date,
                           job_id, department_id)
    VALUES(p_emp_id, p_start_date, p_end_date,
            p_job_id, p_department_id);
END add_job_history;
```


# Deploying Java Stored Procedures

Before deploying Java stored procedures, perform the following steps:

1. Create a database connection.
2. Create a deployment profile.
3. Deploy the objects.



# Creating Program Units



The screenshot shows a window titled 'ADD\_JOB\_HISTORY' with a tab for 'TEST\_JDEV'. The main text area contains the following SQL code:

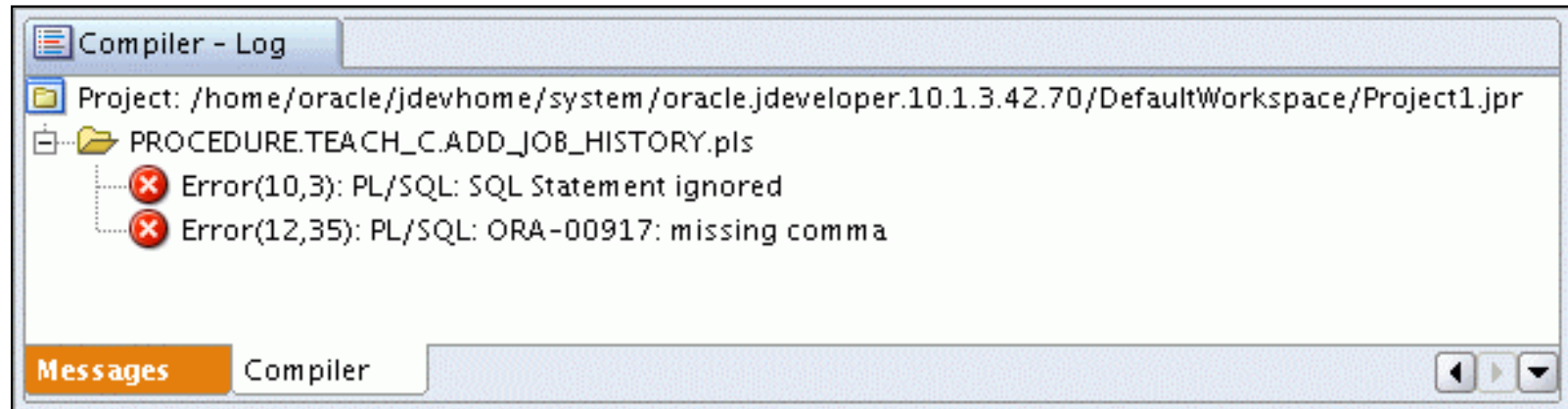
```
FUNCTION "TEST_JDEV"  
RETURN VARCHAR2  
AS  
BEGIN  
    RETURN(' ');  
END;
```

The code is displayed in a monospaced font. The 'BEGIN' and 'END' keywords are in blue. The 'RETURN(' ');' line is indented. A yellow highlight is visible on the line following the 'END;' statement. The window has a standard toolbar at the top and a status bar at the bottom labeled 'Source'.

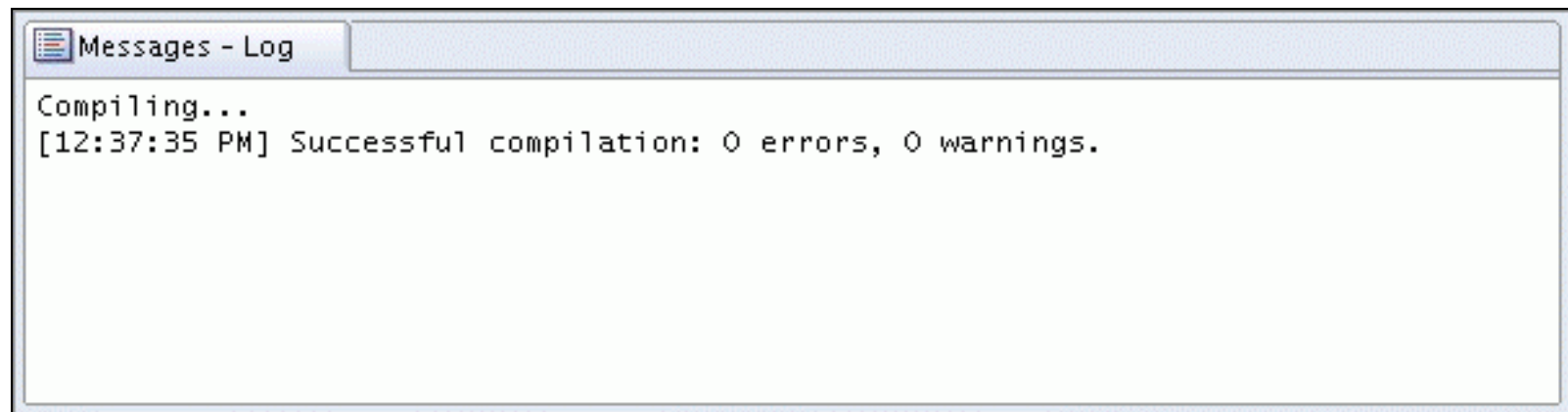
**Skeleton of the function**



# Compiling

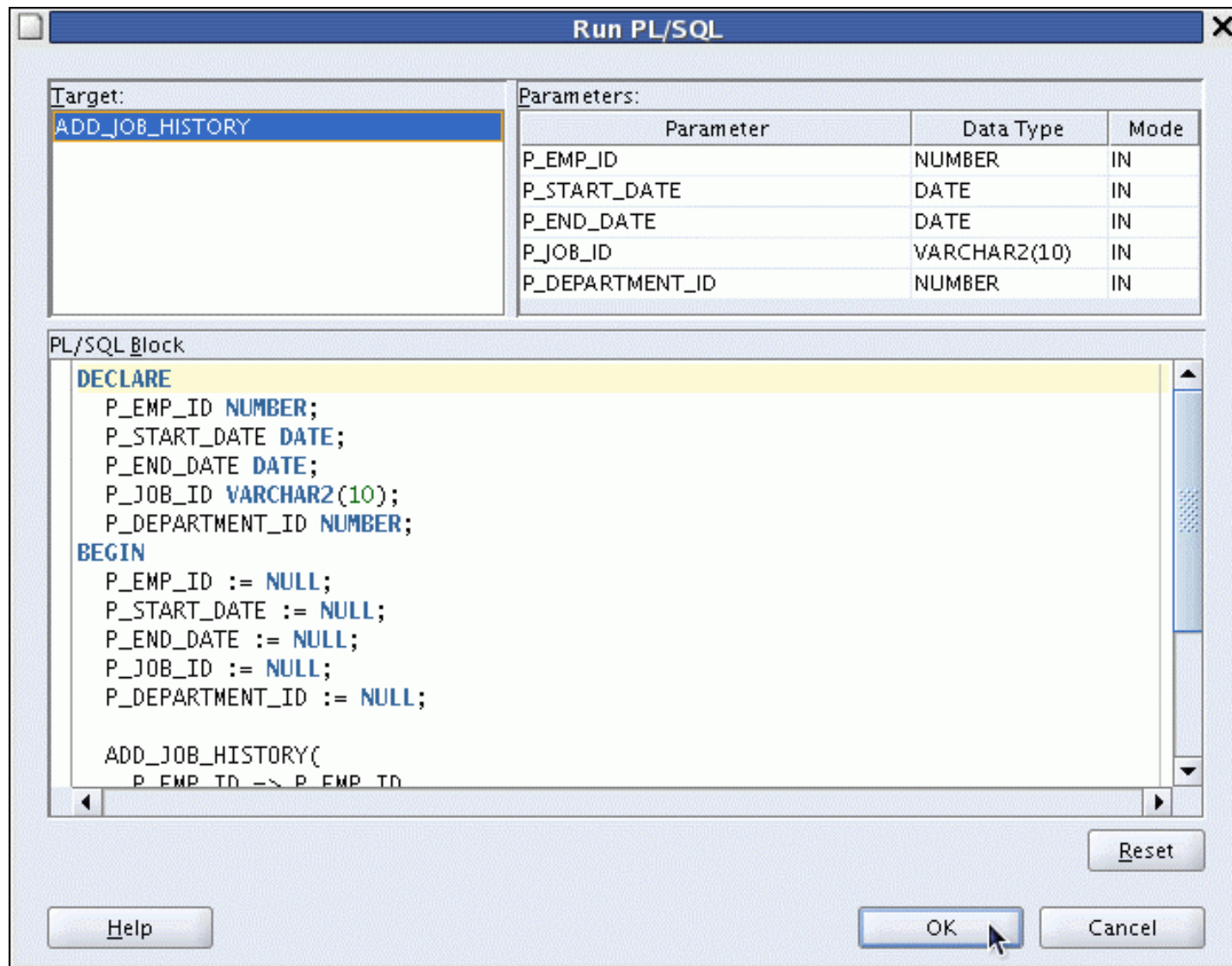


## Compilation with errors

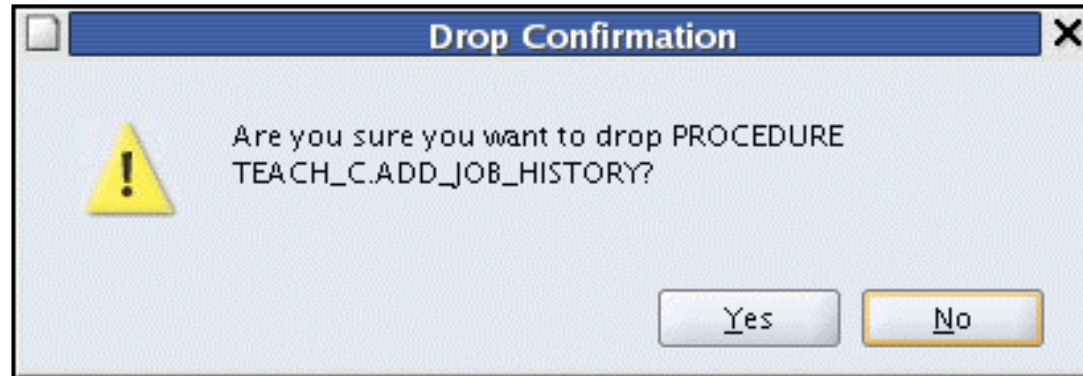


## Compilation without errors

# Running a Program Unit



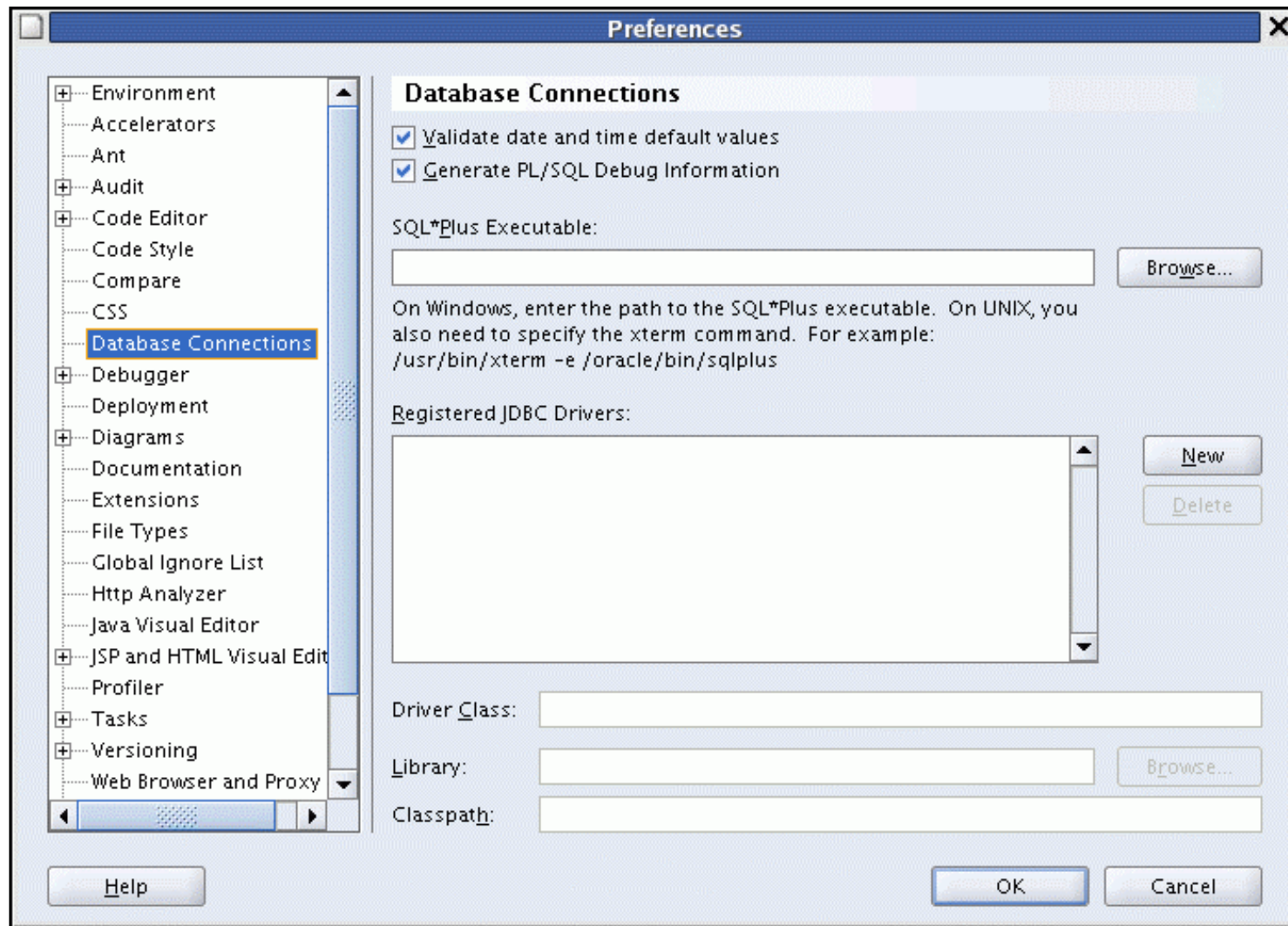
# Dropping a Program Unit



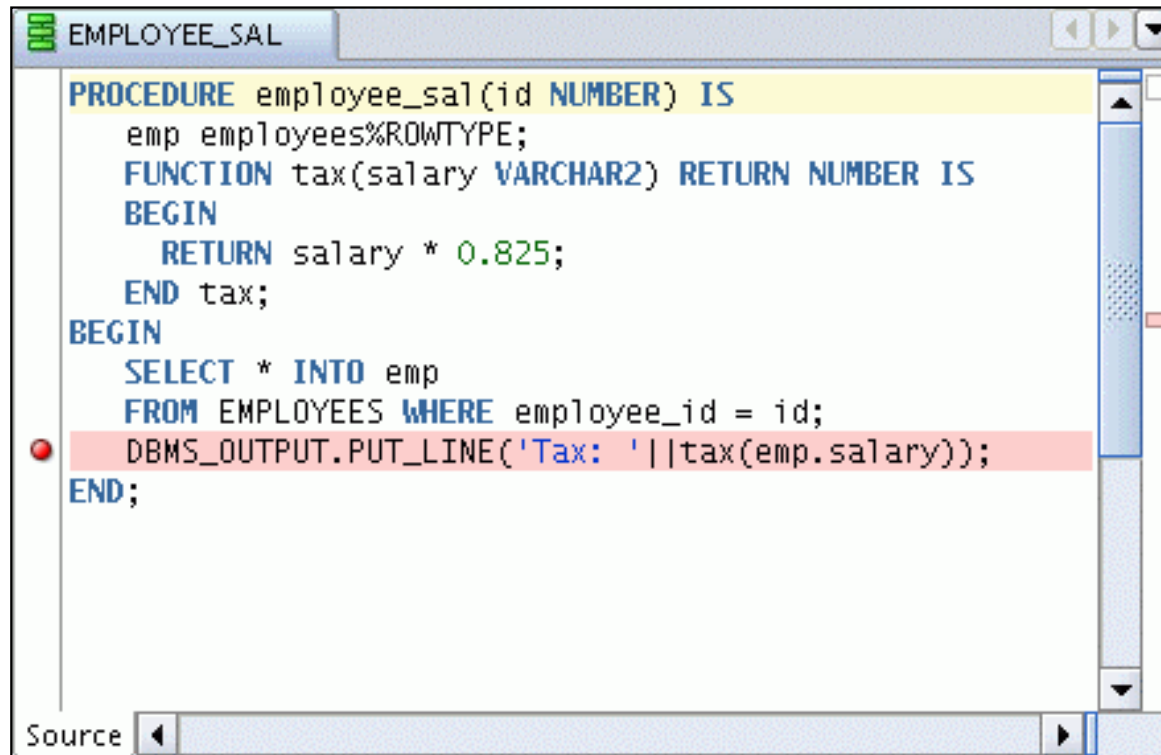
# Debugging PL/SQL Programs

- JDeveloper supports two types of debugging:
  - Local
  - Remote
- You need the following privileges to perform PL/SQL debugging:
  - `DEBUG ANY PROCEDURE`
  - `DEBUG CONNECT SESSION`

# Debugging PL/SQL Programs



# Setting Breakpoints



The screenshot shows a window titled 'EMPLOYEE\_SAL' containing the following PL/SQL code:

```
PROCEDURE employee_sal(id NUMBER) IS
  emp employees%ROWTYPE;
  FUNCTION tax(salary VARCHAR2) RETURN NUMBER IS
  BEGIN
    RETURN salary * 0.825;
  END tax;
BEGIN
  SELECT * INTO emp
  FROM EMPLOYEES WHERE employee_id = id;
  DBMS_OUTPUT.PUT_LINE('Tax: ' || tax(emp.salary));
END;
```

A red circular breakpoint icon is positioned to the left of the line containing the `DBMS_OUTPUT.PUT_LINE` statement. The code is displayed in a monospaced font with syntax highlighting: keywords in blue, identifiers in black, and literals in green.



# Stepping Through Code

Debug 

