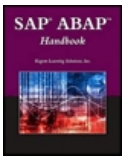

SAP ABAP Handbook

by Kogent Learning Solutions, Inc.

Chapter 4: Understanding ABAP Workbench

Highlights

ABAP Workbench is a graphical programming environment in the SAP R/3 system to develop different applications by using the ABAP language. It provides different tools, such as ABAP Dictionary,ABAP Editor, and Screen Painter, to create ABAP applications. Using these tools, you can perform different tasks, such as defining data structures in ABAP Dictionary, developing data programs in ABAP Editor, and designing interfaces in Screen Painter and Menu Painter. Besides these tasks, you can also create user-defined reports, transactions, and enhancements within ABAP Workbench. Moreover, all the tools of ABAP Workbench are integrated, which means one tool recognizes the objects created by the other tools.

This chapter introduces you to ABAP Workbench. You explore the tools available in ABAP Workbench, such as ABAP Dictionary, ABAP Editor, Screen Painter, and Menu Painter. This chapter also describes the role of Package Builder to create packages for development objects. In addition, you explore the testing tools of ABAP Workbench, such as ABAP Debugger and Performance Analysis, which are used to debug the ABAP program code and trace program performance. You also learn about web services and how ABAP Workbench helps publish, search, and call the web services. Finally, you explore the eXtensible Stylesheet Language Transformation (XSLT) Editor tool of ABAP Workbench, which is used to define XSL transformations.

Overview of ABAP Workbench

ABAP Workbench provides a set of tools and libraries to design, implement, test, and maintain the transactions and reports written in the ABAP language. Some examples of these tools are ABAP Dictionary, Object Navigator, and ABAP Editor. By using these tools of ABAP Workbench, you can create ABAP programs, user interfaces, web services, and access database information. You can use ABAP Workbench to implement additional functionality in ABAP applications in the form of programs, reports, screens, or menus, if the functions of a standard SAP system are not sufficient to meet your requirements.

All the tools provided by ABAP Workbench are integrated with each other. For instance, assume that you write a program in ABAP Editor that uses a user-defined table, ZStudentData, which is already defined in ABAP Dictionary. Now, just by double-clicking the name of this table (ZStudentData), you receive the entire structure of this table in ABAP Editor. The integrated environment of ABAP Workbench provides the Object Navigator tool, which helps you to manage application development and understand the relationship between the objects of a program. You learn more about the Object Navigator tool later in the chapter.

Note An ABAP application is created as a transaction or a report. A transaction represents an end-user application, which retrieves data from users and performs the relevant actions. For instance, an application that creates purchase orders is a transaction. In contrast, a report may be defined as an application that requires less or no user interaction. A monthly report of raw materials imported by a company is an example of a report.

Exploring the ABAP Workbench Tools

The ABAP Workbench tools are used to write ABAP code, design the screens, and create user interfaces. Figure 4.1 shows different tools provided by ABAP Workbench:

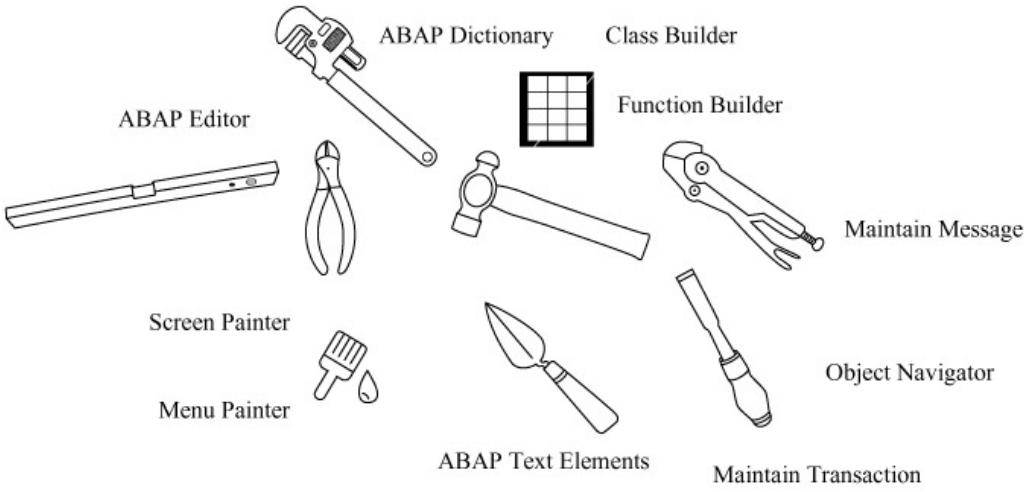

Figure 4.1: ABAP workbench tools

As shown in Figure 4.1, the following are different tools provided by ABAP Workbench:

- `ABAP Dictionary`
- `ABAP Editor`
- `Class Builder`
- `Function Builder`
- `Screen Painter`
- `Menu Painter`
- `Object Navigator`
- `Maintain Message`
- `ABAP Text Elements`
- `Maintain Transaction`

Now, let's discuss each tool in detail.

ABAP Dictionary

ABAP Dictionary is one of the important tools of ABAP Workbench. It is used to create and manage data definitions without redundancies. ABAP Dictionary always provides the updated information of an object to all the system components. The presence and role of ABAP Dictionary ensures that data stored in an SAP system is consistent and secure.

The data definitions stored in ABAP Dictionary allow you to create various objects, such as tables and views. The objects stored in ABAP Dictionary are logically related to each other in the context of an application. ABAP Dictionary also stores some standard functions that help you modify the properties of a field on a screen. For instance, you can assign an input help to a field or restrict the range to accept data from a field.

To open the initial screen of ABAP Dictionary, either select `SAP Menu > Tools > ABAP Workbench > Development > ABAP Dictionary` in the `SAP Easy Access` screen or simply enter the `SE11` transaction code in the `Command` field present on the standard toolbar and press the ENTER key. Figure 4.2 shows the initial screen of ABAP Dictionary:

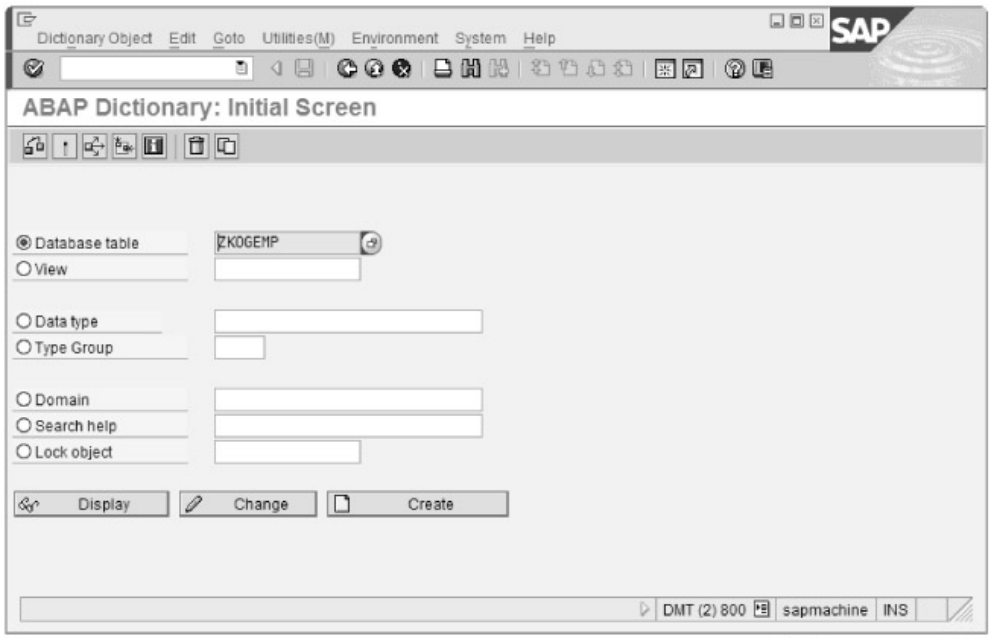

Figure 4.2: Displaying the initial screen of ABAP dictionary

As shown in Figure 4.2, the initial screen of ABAP Dictionary provides the following object types:

- **Database table—** Creates table definitions in ABAP Dictionary, independent of any database. The created table definition can then be used to create a table (with the same table structure) in the database.
- **View—** Creates view definitions in ABAP Dictionary. A view is a virtual table that does not store the data physically. It has a logical structure that represents or points to the data from one or more database tables. From the created view definition, you can create a view (with the same view structure) in the database.
- **Data type—** Creates a definition of a user-defined type in ABAP Dictionary. The created definition of a user-defined type can then be used in a program to define data types and data objects.
- **Type group—** Creates groups of data types in ABAP Dictionary.
- **Domain—** Creates domains in ABAP Dictionary. A domain is used to describe the technical attributes of a field, such as a range of values and the type of data that are acceptable in a field.
- **Search help—** Creates a help document (F4 help) or called input help for fields. A help document related to a field provides you help to enter a set of valid values in the field.
- **Lock object—** Creates a local or lock object that helps synchronize the access of multiple users simultaneously. This is because an SAP system does not allow multiple users to access an object simultaneously. In such a situation, ABAP Dictionary allows you to create a lock or local object.

Note that when you perform any change in ABAP Dictionary, the change is also reflected in the related ABAP program or screen. ABAP Dictionary also facilitates easy navigation between development objects and dictionary definitions. For example, when you double-click the name of a dictionary object in a program, the SAP system displays the definition of that object in ABAP Dictionary. If you make any changes to the dictionary objects, they are automatically reflected in the program, and the program refers to the changed objects when it is executed. Note that ABAP programs are interpreted by the interpreter, and it is not necessary to recompile the reference of changed dictionary objects.

ABAP Editor

ABAP Editor is used to create ABAP programs by writing code. You can use the ABAP Editor to define the class methods, function modules, screen flow logic, type groups, and logical databases for the ABAP programs. To open the initial screen of ABAP Editor, either select `SAP Menu > Tools > ABAP Workbench > Development > ABAP Editor` in the `SAP Easy Access` screen or simply enter the `SE38` transaction code in the `Command` field present on the standard toolbar, and press the `ENTER` key. Figure 4.3 shows the initial screen of ABAP Editor:

Figure 4.3: Displaying the initial screen of ABAP Editor

ABAP Editor is available in three different modes for the front-end and back-end purposes. These modes of ABAP Editor can be accessed from the `ABAP Editor` tab in the `User-Specific Settings` dialog box. You can open te `User-Specific Settings` dialog box by selecting Utilities > Settings in the initial screen of ABAP Editor. Figure 4.4 shows the three modes of ABAP Editor in the `User-Specific Settings` dialog box:

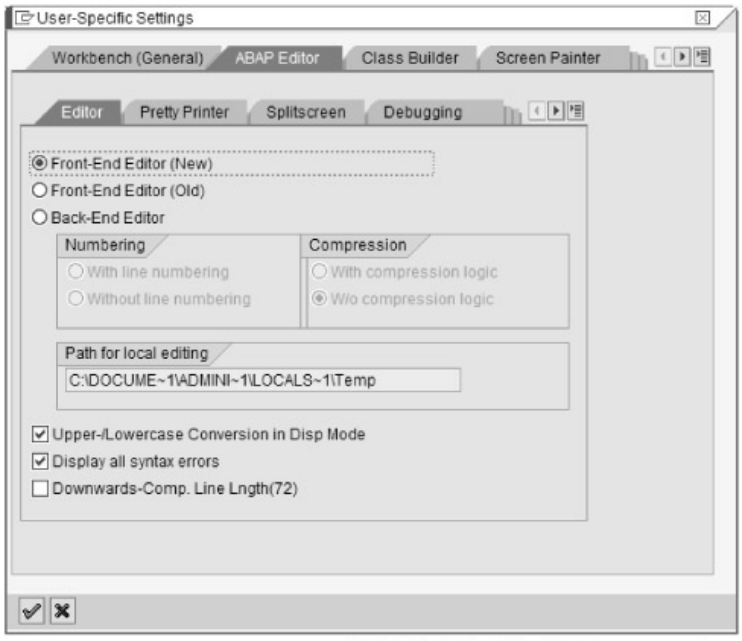

Figure 4.4: Displaying the three modes of ABAP Editor

As shown in Figure 4.4, the following are the three modes of ABAP Editor:

- `Front-End Editor (New)`
- `Front-End Editor (Old)`
- `Back-End Editor`

Note Click the `Transfer` icon to apply the desired mode of ABAP Editor and close the `User-Specific Settings` dialog box to use the editor.

Now, let's discuss these modes of ABAP Editor in detail, one by one.

Front-End Editor (New)

`Front-End Editor (New)` is the latest programming tool to create applica tions in an SAP system. It provides various new features, such as code hints, syntax highlighting, and auto-completion of language structures and elements. When you work in `Front-End Editor (New)`, the source code of your program is loaded in the front end of the SAP system and can be edited easily. Note that `Front-End Editor (New)` is used in situations when editing the source code does not require communication with the back end of the SAP system. Using `Front-End Editor (New)`, you can modify the development objects, such as ABAP programs, method implementations, function module implementations, and screen flow logic.

Figure 4.5 shows the `Front-End Editor (New)` screen:

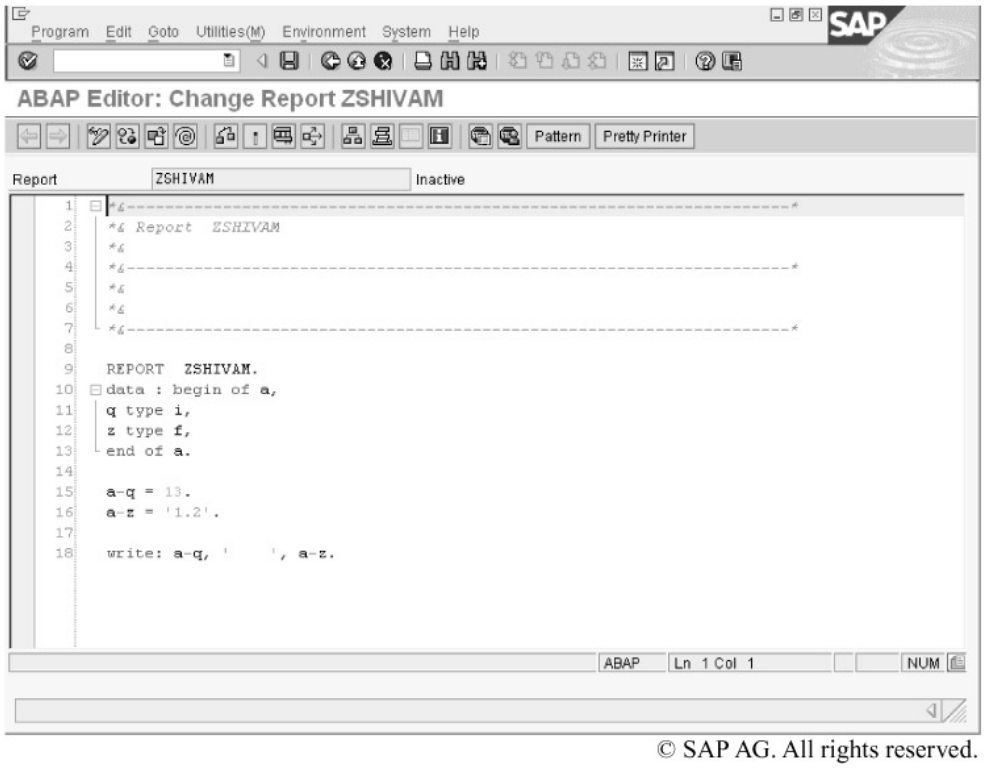

Figure 4.5: Displaying the appearance of `Front-End Editor (New)`

`Front-End Editor (New)` supports various formatting functions, such as automatic code indentation, smart tabs, automatic insertion of brackets, and mistyping correction, which speed up the development process. In addition, `Front-End Editor (New)` offers significantly improved functionalities over the two classical modes of the ABAP Editor, `Front-End Editor (Old)` and `Back-End Editor`.

Front-End Editor (Old)

As stated earlier, `Front-End Editor (New)` is used to edit the source code of the development objects. The classic mode of `Front-End Editor (Old)` is also used to edit the development objects, such as ABAP programs, function modules, and type groups. The source code of a program in `Front-End Editor (Old)` appears in plain text. Figure 4.6 shows the appearance of `Front–End Editor (Old)`:

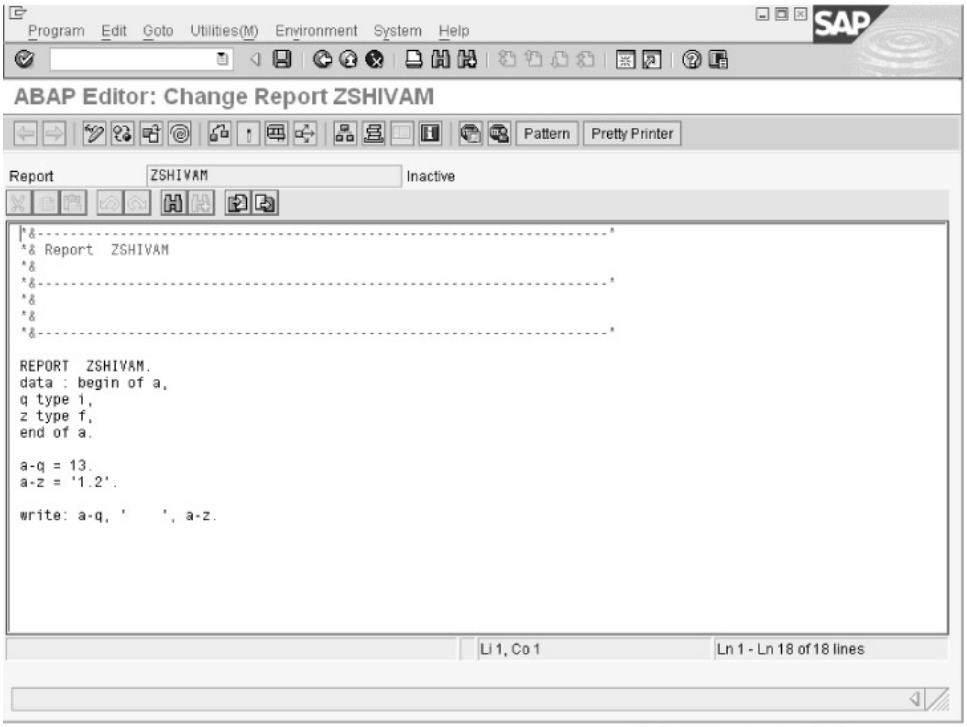

Figure 4.6: Displaying the appearance of `Front-End Editor (Old)`

When you edit the source code of a program in `Front-End Editor (Old)` the functions required for editing do not communicate with the back end of the SAP system. You can perform the following activities when you work with `Front-End Editor (Old)`:

- Performing local scrolling in an SAP window
- Performing cut, copy, and paste for selected text areas
- Dragging any text and dropping at the desired location
- Using the context menu to access editor functions
- Using the Find and Replace facility
- Using the context menu to navigate to a selected line
- Using the context menu to access the buffer and block operations
- Inserting comments on text blocks
- Working with blocks and clipboards
- Using navigation functions (forward navigation)
- Showing the syntax check, error messages, and warnings in a separate window
- Highlighting the comment lines
- Using the automatic line feed feature when the maximum line width is reached
- Using the Insert statement
- Performing undo and redo functions multiple times
- Displaying the current cursor position
- Using Pretty Printer to standardize the layout

- Importing and exporting the local files

Back-End Editor

`Back-End Editor`, another classic editor, allows users to edit ABAP code. This editor is line-based and is used to perform normal editor functions, such as cut, copy, and paste. For this, however, you must first select a block of lines. Figure 4.7 shows the line-based mode of `Back–End Editor`:

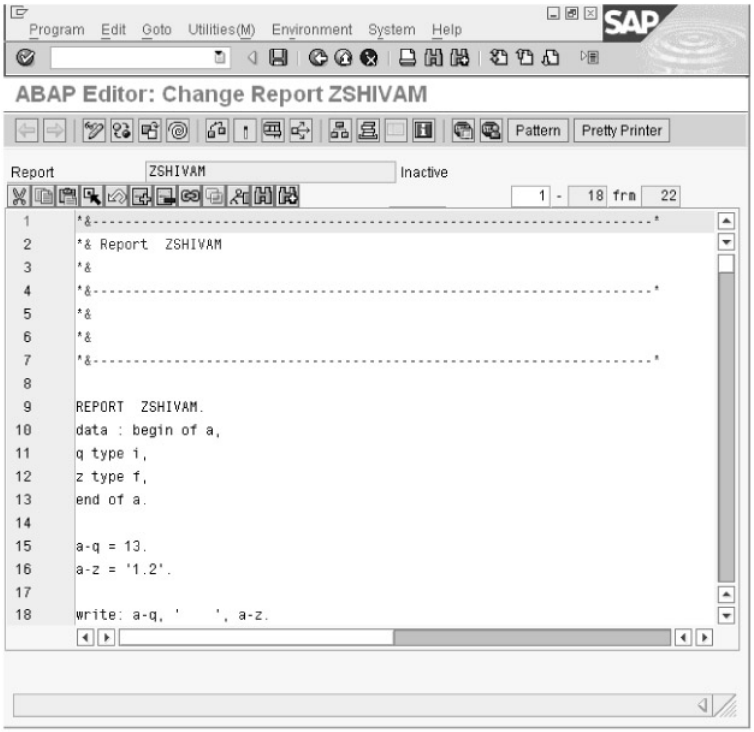

Figure 4.7: Displaying `Back-End Editor`

When you need to write a very long program, especially in the Wide Access Network (WAN) environment, `Back-End Editor` can produce better performance than `Front-End Editor`. Moreover, `Back-End Editor` allows you to edit any development object based on ABAP Editor. You can perform the following activities when you work with `Back-End Editor`:

- Navigating the functions (forward navigation)
- Selecting the block selection easily
- Using the Compression logic feature
- Using the Line Numbering feature
- Using the functions, such as Find and Replace
- Highlighting the comment lines in color
- Using the Insert Statement function
- Including the expansion
- Using the Single-step Undo function
- Converting a normal text block into the comment lines
- Using Pretty Printer to standardize the layout

- Checking the syntax of code that has the possibility of raising an error
- Uploading or downloading the local files

Class Builder

Class Builder is a tool of ABAP Workbench used to create, define, change, and test the global ABAP classes and interfaces. ABAP classes and interfaces are object types defined in Repository Browser. These classes and interfaces are stored in a central class library and are available to all the objects in an SAP system. This class library is used to display all the existing classes and interfaces by using Class Browser. Class Browser, an integral part of Class Builder, is used to display and maintain the existing global object types from the class library. Class Browser can be started either from Class Builder or simply by entering the `CLABAP` transaction code in the `Command` field. You can perform the following actions by using the Class Builder tool:

- Displaying an overview of global object types and their relationships in Class Browser
- Maintaining the already existing global classes or interfaces easily
- Creating new global classes and interfaces
- Implementing inheritance between two or more global classes
- Creating compound interfaces
- Creating and specifying the attributes, methods, and events of global classes and interfaces
- Defining internal types in classes
- Implementing the methods
- Redefining the already existing methods
- Maintaining local auxiliary classes as well
- Testing classes or interfaces in a simulated runtime environment

To open the initial screen of Class Builder, either select SAP `Menu > Tools > ABAP Workbench > Development > Class Builder` in the SAP `Easy Access` screen or enter the `SE24` transaction code in the `Command` field and press the `ENTER` key. You can display the contents of the class library as well as edit a class by using the initial screen of Class Builder. Figure 4.8 shows the intial screen of Class Builder:

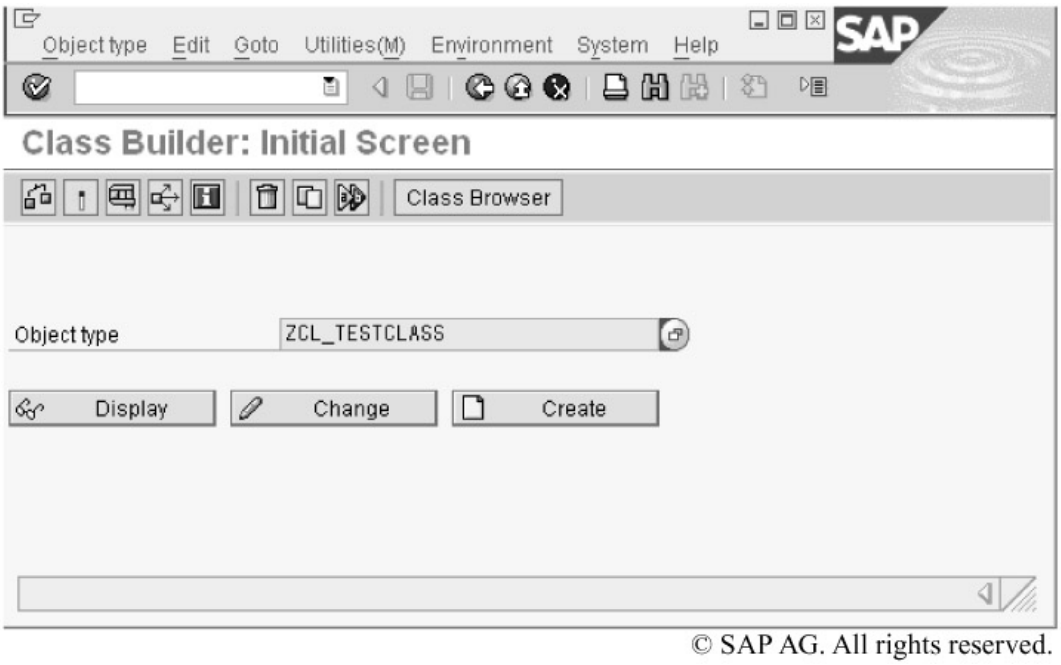

Figure 4.8: Displaying the initial screen of Class Builder

In Figure 4.8, you need to specify the name of a class or interface in the `Object type` field. Note that the `Object type` name can be up to 30 characters long. In our case, `ZCL_TESTCLASS` is the name of a class, specified in the `Object type` field. Now, you can click either the `Display`, `Change`, or `Create` button to perform a particular task. The `Create` button is used to create a new object type; however, the `Display` and `Change` buttons are used to display and modify the content of an already existing object type, respectively.

Function Builder

Function Builder plays an important role in defining and maintaining the ABAP functional modules. Function modules are procedures or routines defined in an ABAP program. They are stored in function groups, which act as containers for function modules that are logically related to each other. The Function Builder tool is used to create a function group and function module. Function modules can accept the values of input parameters from users or take default values for these parameters. In addition, function modules support exception handling. As a result, exceptions can be caught if executing the function modules causes errors. Moreover, you can test the function modules by using Function Builder. An SAP system contains several predefined function modules, such as `CALCULATE_EXCHANGE_RATE` and `CHANGEDOCUMENT_READ_HEADERS`, which can be called from any ABAP program.

To open the initial screen of Function Builder, either select SAP `Menu > Tools > ABAP Workbench > Development > Function Builder` in the SAP `Easy Access` screen or enter the `SE37` transaction code in the `Command` field and press the `ENTER` key. Figure 4.9 shows the initial screen of Function Builder:

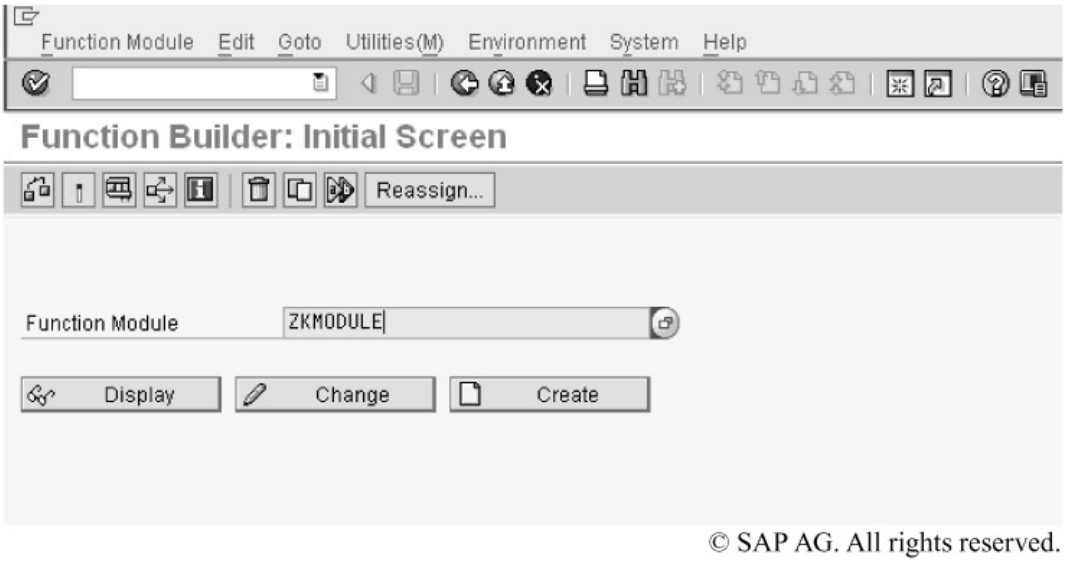

Figure 4.9: Showing the initial screen of function builder

Enter a name for the function module you are creating in the `Function Module` field of the initial screen of Function Builder. In our case, we have entered ZKMODULE, as shown in Figure 4.9. Click the `Create` button to create the given function module. Alternatively, click the `Display` button to display and the `Change` button to change the content of an already existing function module.

Screen Painter

Screen Painter is used to design and manage screens and their elements. It facilitates users to create GUI screens for transactions. To open the initial screen of Screen Painter, either select SAP `Menu > Tools > ABAP Workbench > Development > User Interface > Screen Painter` in the SAP `Easy Access` screen or enter the `SE51` transaction in the `Command` field and press the `ENTER` key. Figure 4.10 shows the initial screen of Screen Painter:

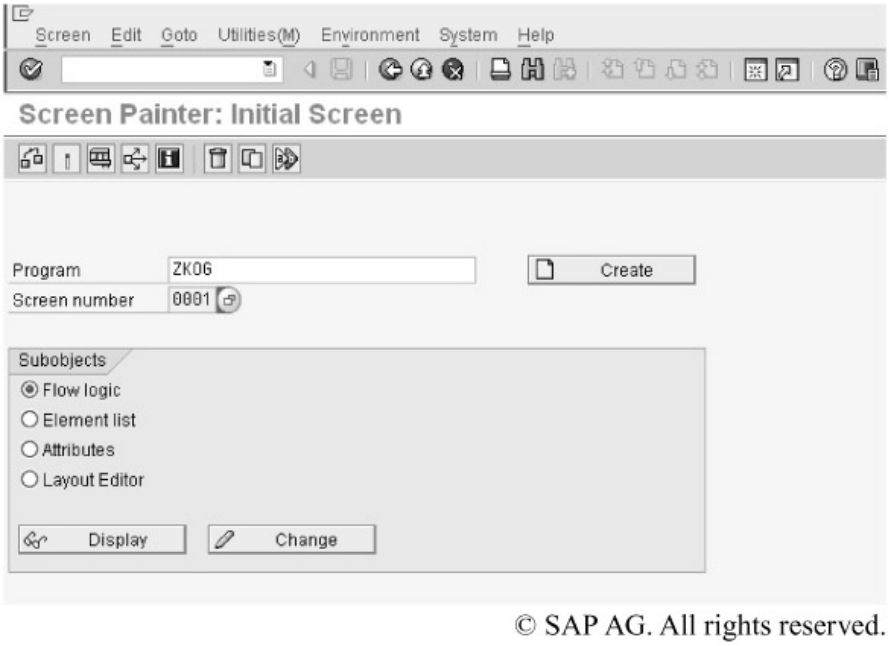

Figure 4.10: Sharing the initial screen of screen painter

In the initial screen of Screen Painter, you provide a name of the program for which the screen has to be created and a number associated with that screen. For example, ZKOG is the program name and 0001 is the screen number, as shown in Figure 4.10. Now, click the `Create` button to create the screen for the given program. Click the `Display` and `Change` buttons to display and modify the screen for the given program, respectively. Note that on the initial screen of Screen Painter, you also need to select any one of the listed subobjects. Table 4.1 describes the listed subobjects in the initial screen of Screen Painter:

Table 4.1: A list of subobjects on the initial screen of screen painter

Subobjects	Description
Layout Editor	Manages a screen's layout. The screen layout describes both the screen elements and their layout.
Element List	Manages the ABAP Dictionary or program fields for a screen, and assigns a program field to the `OK_CODE` (a predefined field name) field in Screen Painter. Screen elements include I/O fields, text fields, check boxes, radio buttons, buttons, and other controls.
Attributes	Manage screen's attributes. Screen attributes determine the type of a screen and the program to which the screen belongs.
Flow Logic	Manage screen's flow logic. The flow logic controls the flow of execution of a program.

Layout Editor of Screen Printer is used to design the layout for a screen. It is available in two modes, Graphical mode (available only on Microsoft Windows platforms) and Alphanumeric mode. Both modes provide the same functionalities, but in a different way. That is, in the graphical mode, you use a drag-and-drop interface similar to a drawing tool, while in the alphanumeric mode, you use the keyboard and menus.

Menu Painter

Menu Painter is used to design user interfaces for ABAP programs. The main components of Menu Painter are GUI status, menu bars, menu lists, F-key settings, functions, and titles. In other words, using Menu Painter, you can create custom menus in SAP. However, to use Menu Painter, you must understand the term GUI status. This status consists of a menu bar, a standard toolbar, an application toolbar, and a collection of function keys. The GUI status differs from the GUI title, which includes the SAP title bar, as shown in Figure 4.11:

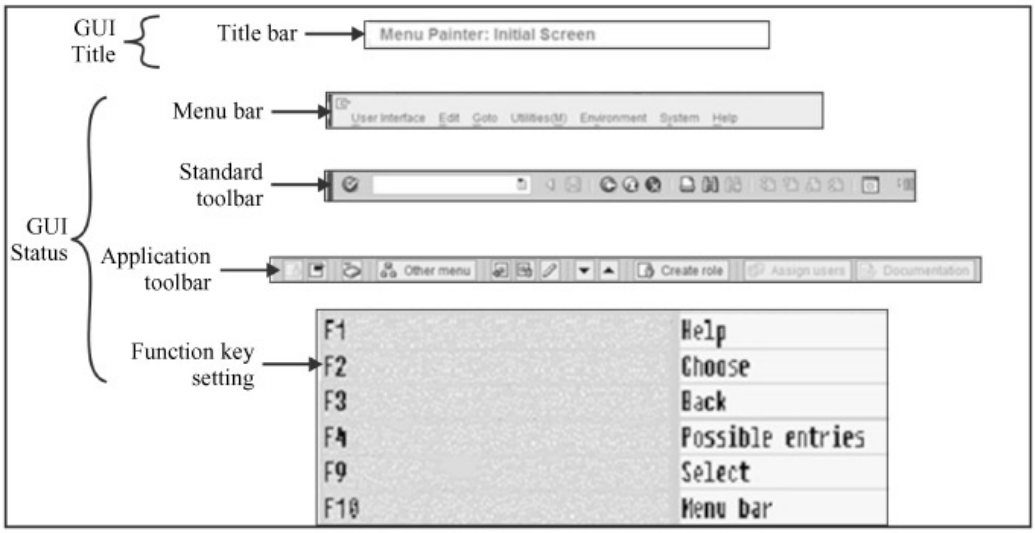

Figure 4.11: Showing the components of menu painter

To open the initial screen of Menu Painter, either select SAP `Menu > Tools > ABAP Workbench > Development > User Interface > Menu Painter` in the SAP `Easy Access` screen or enter the `SE41` transaction code in the `Command` field and press the `ENTER` key. Figure 4.12 shows the initial screen of Menu Painter:

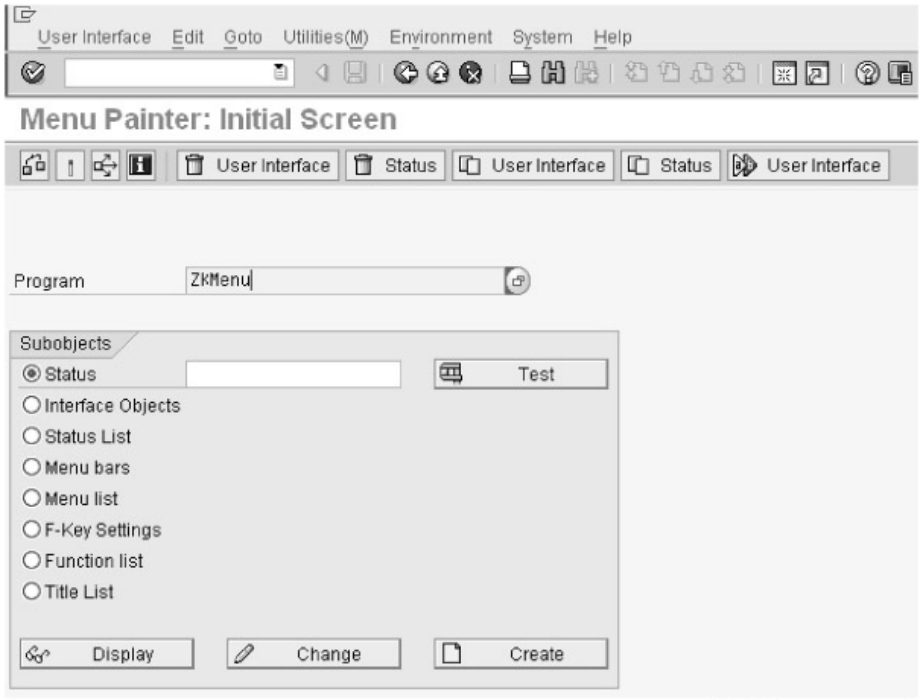

Figure 4.12: Displaying the the initial screen of menu painter

As shown in Figure 4.12, the initial screen of Menu Painter contains a program name, ZKMenu, in the `Program` field. The `Create` button is used to create the program with the given name, and the `Display` and `Change` buttons are used to display and modify the contents of the given program, respectively. The `Test` button is used to test the specified program. Note that before using the `Display` and `Change` buttons, you must specify one of the following subobjects, as given in Table 4.2:

Table 4.2: A list of subobjects for the initial screen of menu painter

Subobjects	Description

Status	Opens the Menu Painter work area
Interface Objects	Displays all the user interface objects for the current program
Status List	Displays a list of all the available GUI statuses for the current program
Menu Bars	Displays a list of menu bars that are sorted by status
Menu List	Displays a list of all the menus
F-Key Settings	Displays a list of function key settings
Function List	Displays a list of the function codes
Title List	Displays a list of all GUI titles for the current program

Object Navigator

Object Navigator is another important tool of ABAP Workbench. It helps create objects in an SAP system and navigate from one object to another. It acts as the central area of ABAP Workbench, where you can access any object of an SAP system. In addition, Object Navigator integrates the programming of various SAP objects in ABAP Workbench. In other words, you can create any SAP object by using Object Navigator, which would then be reflected in all the tools of ABAP Workbench. Object Navigator was also available in the earlier versions of SAP, but it was called Object Browser in Release 3.*x* and Repository Browser in Release 4.*x*. When you create an object or a component of an object in Object Navigator, the object is stored in the repository of SAP and is called a development object or a repository object. Development or repository objects are arranged in a list format, also called an object list. When you open an application, ABAP Workbench automatically opens the application in the development tool with which that object has been created.

Object Navigator is a central area of ABAP Workbench, from where you can open various browers for various development processes, including

- `Multipurpose Internet Mail Extension Files (MIME) Repository`—Displays all directories that store MIME objects, downloaded or imported from external sources in an SAP system.
- `Repository Browser`—Displays repository objects in the form of object lists on the basis of their searching category. The searching category for the repository objects can be a package, program, or class.
- `Repository Information System`—Displays the information of all the repository objects. Unlike the Repository Browser, the Repository Information System is not dependent on the selection of repository objects.
- `Tag Browser`—Displays tags for web applications.
- `Transport Organizer`—Displays the output of a user request sent to it.
- `Test Repository`—Displays the result after testing repository objects.

To open the Object Navigator screen, either select SAP menu menu > Tools > ABAP Workbench > Overview > Object Navigator in the `SAP Easy Access` screen or enter the `SE80` transaction code in the `Command` field and press the ENTER key. Figure 4.13 shows the Object Navigator screen:

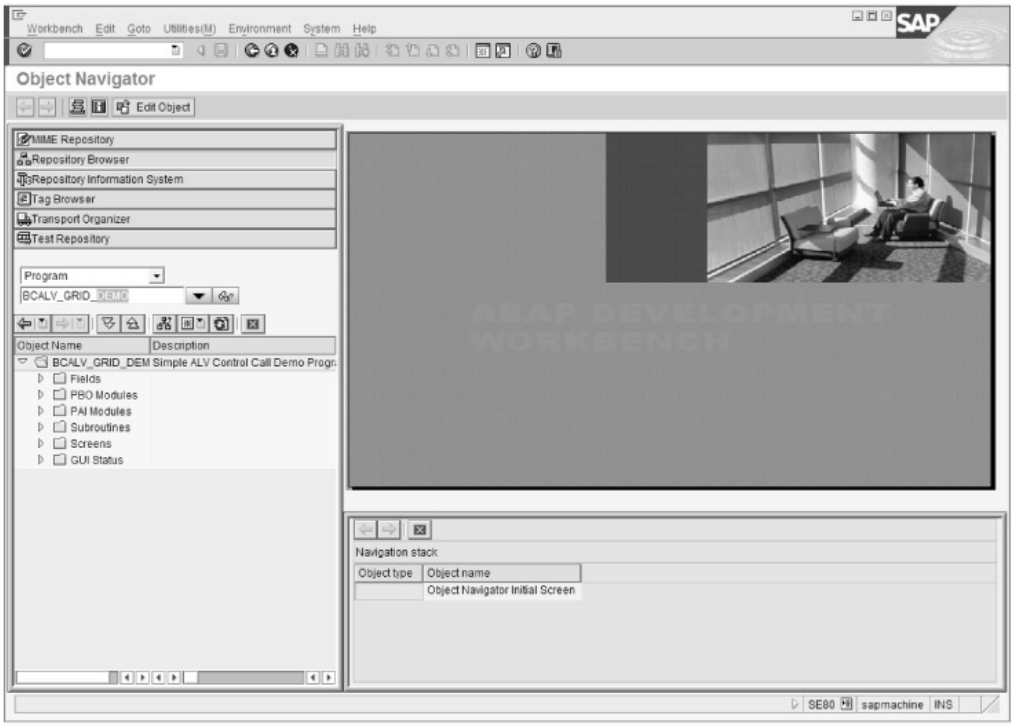

Figure 4.13: Displaying the object navigator screen

The Object Navigator screen acts as a common interface to access any application or object in an SAP system.

Object Navigator Interface

The following user interface areas appear when a user accesses ABAP Workbench through Object Navigator:

- **Navigation area**—Contains different browsers that allow you to display the development objects and open the tools associated with these objects to modify them.
- **Tools area**—Contains necessary tools to edit a development object.
- **Integrated window**—Displays the result after syntax check-ups, navigation stack, and worklist.
- **Additional dialog box**—Appears when you open an object in a new session.

Figure 4.14 shows the different areas of Object Navigator:

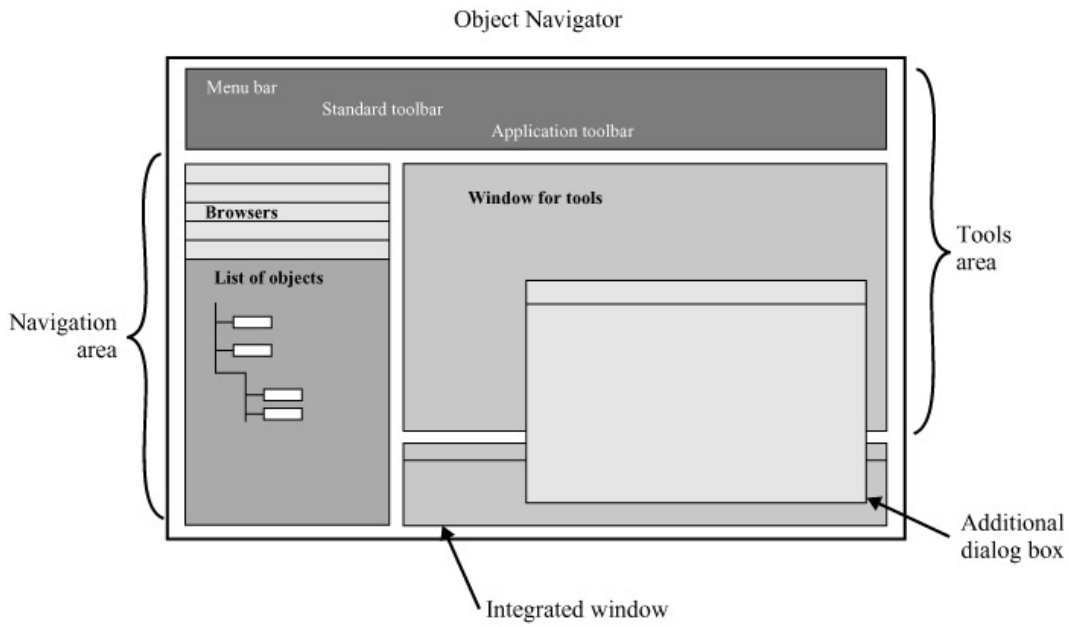

Figure 4.14: Displaying the different areas of object navigator

As shown in Figure 4.14, the two main areas of Object Navigator are the Navigation area and Tools area. The Navigation area is used to display the lists of development objects in a hierarchical order. The Tools area provides a window to open a development object. These areas are not interconnected, meaning that any change in one area is not reflected in the other area. Moreover, in both areas, you can select functions from a context menu. The options available in the context menu are different for the different development objects.

Object Navigator facilitates users to perform the following tasks:

- Select a browser and navigate in the object list
- Use the tools for development objects
- Navigate from one window to another window
- Perform syntax checks in the Integrated window
- Open an object in a new session using an Additional dialog box

As stated earlier, development objects are displayed in Repository Browser of Object Navigator in the form of object lists. An object list arranges the development objects in hierarchical order on the basis of a specified category. In Object Navigator, you can display the development objects by selecting a category from the folowing:

- **Application Hierarchy Tool—** Displays a list of applications in hierarchical order in an SAP system.
- **Package—** Displays a list of all development objects in a package.
- **Program—** Displays a list of elements of an ABAP program.
- **Function group—** Displays a list of function modules and their elements, related to a function group.
- **Class/interface—** Displays a list of all the components of a global class or interface. It also displays a list of superclasses of a class and the methods that are redefined in the class.
- **Internet service—** Displays a list of all elements of an Internet service, which includes service description, themes, Hypertext Markup Language (HTML) templates, and MIME objects.
- **Business Server Pages (BSP) application—** Displays a list of elements of a BSP application.
- **Local objects—** Displays a list of local objects created in a program. All displayed local objects belong to the $TMP package and cannot be transported from one computer to another. In an SAP system, local objects are created for the temporary purpose of testing a program. However, you can assign a local object to a package to transport.

- **Inactive objects—** Displays a list of all inactive objects, including local and transportable objects.

Message Maintenance

Messages help an SAP system communicate with the users. For example, error messages display if the users make invalid entries on a screen, making the users aware that a wrong entry has been made. These messages are displayed by using a message class. Each message class has an ID and usually contains a set of messages. A message contains a single text line and may contain placeholders for variables.

To open the initial screen of Message Maintenance, either select `SAP Menu > Tools > ABAP Workbench > Development > Programming Environment > Messages` in the `SAP Easy Access` screen or enter the `SE91` transaction code in the `Command` field and press the `ENTER` key. Figure 4.15 shows the initial screen of Message Maintenance:

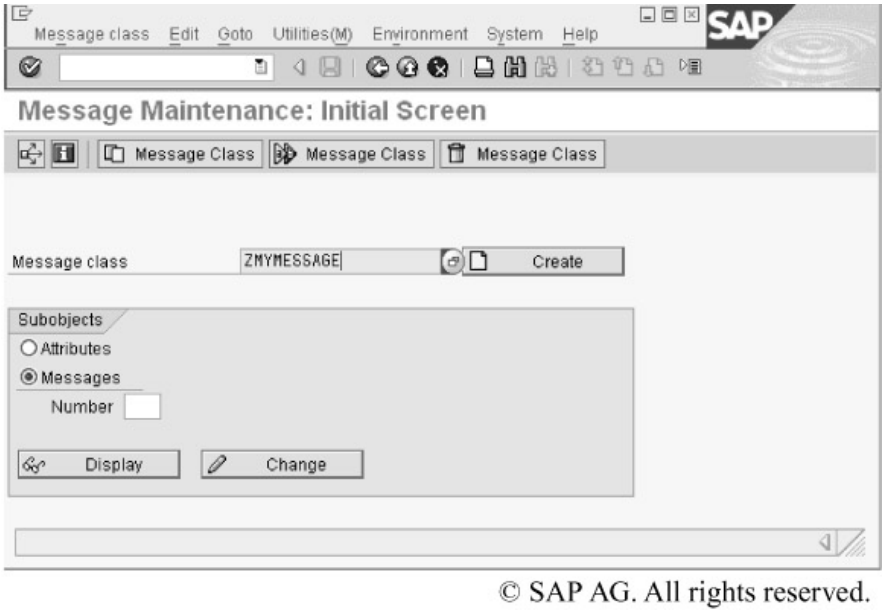

Figure 4.15: Showing the initial screen of message maintenance

After you have created a message, you can use it in a program with the help of the `MESSAGE` statement.

ABAP Text Elements

The text elements of program texts in different languages can be maintained easily with the help of a tool known as Text Elements. The text, which is displayed by a program on a screen, is called a text element. Creation, maintenance, and translation of the text elements can be done with the help of the Text Element Maintenance tool. The following are the various types of text elements:

- **Text symbols—** Used by the Write statement.
- **List and column headers—** Appear in an ABAP list.
- **Selection texts—** Appear on selection screens.

The place where the text elements are stored is known as a Text Pool. You can change the text element without changing the source code of the program. A user can also design some standard text elements to use them in other programs.

The initial screen of Text Elements can be started either from the SAP `Easy Access` screen, or from Repository Browser. To open the screen to create text elements in ABAP, either select `SAP Menu > Tools > ABAP Workbench > Develop-ment > Programming Environment > Text Elements` in the `SAP Easy Access` screen or enter the `SE32` transaction code in the `Command` field and press the `ENTER` key. Figure 4.16 shows the SAP screen to create a text element:

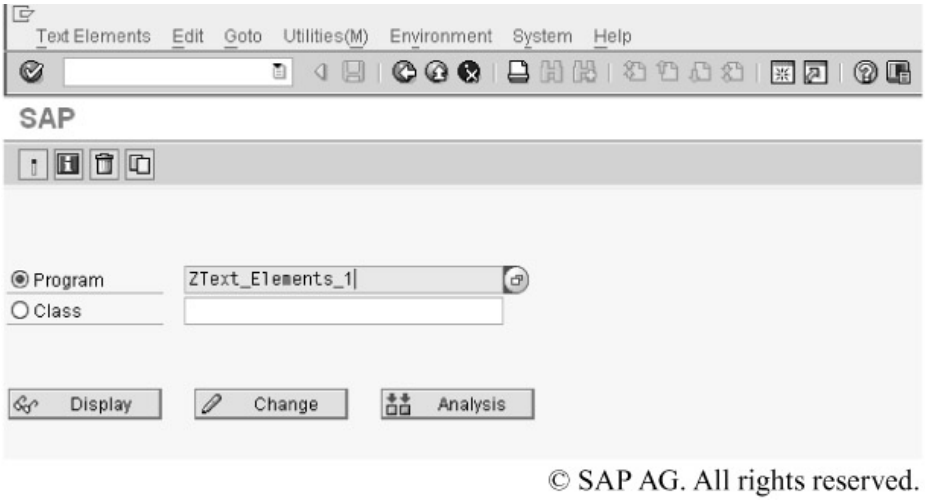

Figure 4.16: Displaying the screen of the text elements

In Figure 4.16, notice that the SAP screen contains the `Program` field, where you enter the name of an ABAP text element. For instance, `ZText_Elements_1` is entered in the `Program` field (see Figure 4.15). You then click either the `Create` button to create the text element or the `Display` button to display and edit the already maintained text elements in the display mode or the `Change` button.

Note In Figure 4.16, you'll notice there is a text field next to the `Class` radio button. This text field accepts the name of an object type (a class or interface). The name can consist of alphanumeric characters with the special characters underscore (_) and forward slash (/). However, the name must not begin with a digit.

Maintain Transaction

Maintain Transaction is an ABAP Workbench tool used to create and manage user-defined transactions. To open the Maintain Transaction screen, either select `SAP Menu > Tools > ABAP Workbench > Development > Other Tools > Transactions` in the `SAP Easy Access` screen or enter the `SE93` transaction code in the `Command` field and press the `ENTER` key. Figure 4.17 shows the `Maintain Transaction` screen:

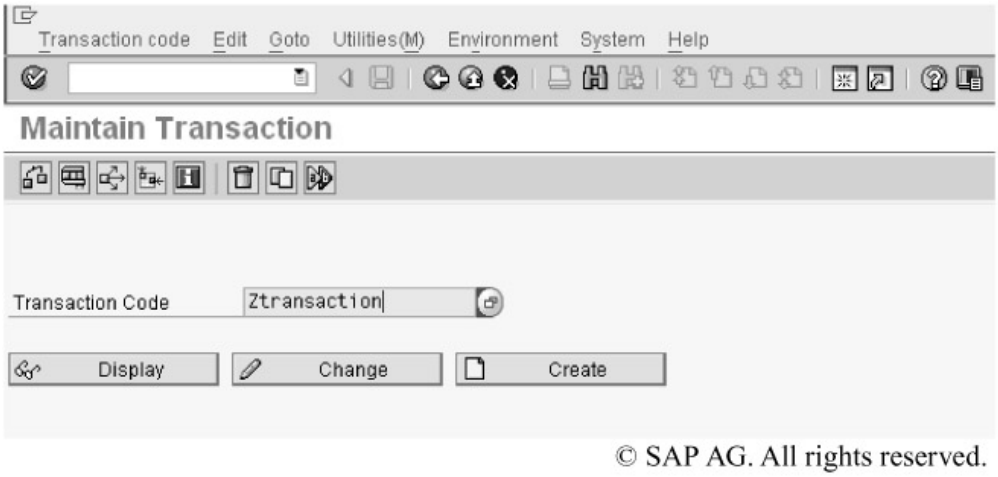

Figure 4.17: Displaying the maintain transaction screen

In Figure 4.17, you can see that the `Maintain Transaction` screen contains a transaction code name, `Ztransaction`, in the `Transaction Code` field. You can either click the `Create` button to create a user-defined tran saction code or the `Display` and `Change` buttons to display and modify the contents of the given transaction code, respectively. In the `Maintain Transaction` screen, you can create the transaction code for various types of transactions. Table 4.3 lists various types of transactions:

Table 4.3: A list of transaction types

Transaction Type	Description
Dialog Transaction	Assigns a transaction code to a dialog program.
Report Transaction	Assigns a transaction code to either an executable program or a report.
Object-Oriented Transaction	Assigns a transaction code to a method of a class in ABAP objects.
Variant Transaction	Executes the dialog transaction with the transaction variant. In Variant Transaction, you can assign default values to the input fields of screens, change the attributes of screen elements, and hide entire screens.
Parameter Transaction	Fills the parameters in the input fields of dialog transactions.

Describing Package Builder

Package Builder is a tool used to implement the concept of a package in ABAP Workbench. A package is a type of development object that acts as a container to store development objects, such as function modules, screens, menus, and transactions. Package Builder, provided by the SAP system, is used to develop and manage the development classes. Package Builder is also used to transfer existing development objects to other existing packages.

Package Builder is started by entering the SPACKAGE or SE21 transaction code in the Command field and pressing the ENTER key. Figure 4.18 shows the initial screen of Package Builder:

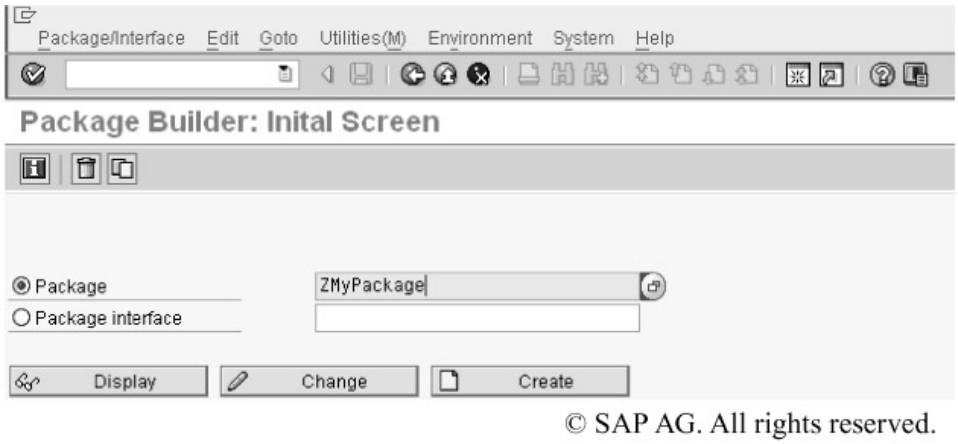

Figure 4.18: Displaying the initial screen of package builder

Some common tasks that are performed by using the Package Builder tool are:

- Creating packages and subpackages
- Specifying a package hierarchy
- Defining package interfaces for potential users
- Adding the elements to package interfaces
- Defining user access for user packages
- Restricting the use of interfaces to selected users
- Creating structure packages and defining filter package interfaces

Types of Packages

Package Builder creates two types of packages: provider (server) packages and user (client) packages. The provider package provides development elements, such as function modules, BAPIs, classes, ABAP programs, and types, to other packages by using one or more interfaces. A provider package can also be a user package, utilizing the services of other packages. In the case of a user package, you define user access for the visible elements of another package allotted to the interface.

Tasks Performed by Provider Packages

The provider package performs two tasks: creating a package and defining a package hierarchy. The main purpose of these tasks is to assign and create a structure for the packages. In the package hierarchy, there are various levels, and the top level is constructed with the help of structure packages that contain several other main packages. In addition, each main package has associated subpackages. You can find the total number of subpackages in a main package and display the levels up to which these subpackages are nested in the main package. Package interfaces are designed to provide easy access to the package elements from outside the package.

The second task is to help access the content of one package from other packages with the help of package interfaces. In other words, the package interfaces help define development objects in different packages.

Tasks Performed by User Packages

Similar to provider packages, the user package provides a structure for packages. In this case, the top level in the package hierarchy is also formed by structure packages, which generally contain main packages, and then the associated subpackages are created within the main packages.

Testing Tools in ABAP Workbench

In ABAP Workbench, testing tools are used to check or test the code entered by the user. The following testing tools are used in ABAP Workbench:

- **ABAP Debugger—** Executes the ABAP program line by line or section by section.
- **ABAP Runtime Analysis—** Shows the overall duration consumed in the processing of the source code.
- **Performance Analysis—** Displays the performance of the system while accessing databases, performing locking activities, and making remote calls to reports and transactions.

Now, let's discuss these testing tools in detail, one by one.

ABAP Debugger

ABAP Debugger enables step-by-step processing of the source code of ABAP programs. ABAP Debugger interrupts the processing of an ABAP program at each step, which allows users to check the processing logic of the ABAP program. Apart from checking the processing logic, users can use ABAP Debugger to check the outcome of the individual statements used in the ABAP program. It also helps the user understand the flow of logic of the programs and displaying the data objects. You can start ABAP Debugger by doing either of the following:

- Setting a breakpoint and then running the ABAP program
- Running the ABAP program directly in debugging mode

Setting a Breakpoint and Then Running the ABAP Program

A breakpoint can be set in an ABAP program (in active mode) either by selecting `Utilities > Breakpoint > Set/Delete` or by using the `BREAK` statement directly. In addition to this, the breakpoint can also be set in the ABAP program by clicking the `Set/Delete Session Breakpoint` icon. After setting the breakpoint, the ABAP program is directly executed either by clicking the `Direct Processing` icon or by pressing the F8 key.

Running the ABAP Program in Debugging Mode

An ABAP program can be executed directly in debugging mode through any of the following ways:

- Writing the `/h` system command in the `Command` field and pressing the `ENTER` key. The debugging mode is switched on.
- Setting the breakpoints at the key statements.

ABAP Runtime Analysis

Runtime Analysis, another tool of ABAP Workbench, is used to trace the time duration and performance of the ABAP source code. To open the initial screen of ABAP Runtime Analysis, either select `SAP Menu > Tools > ABAP`

`Workbench > Test > Runtime Analysis` in the `SAP Easy Access` screen or enter the `SE30` transaction code in the `Command` field and press the `ENTER` key. Figure 4.19 shows the initial screen of ABAP Runtime Analysis:

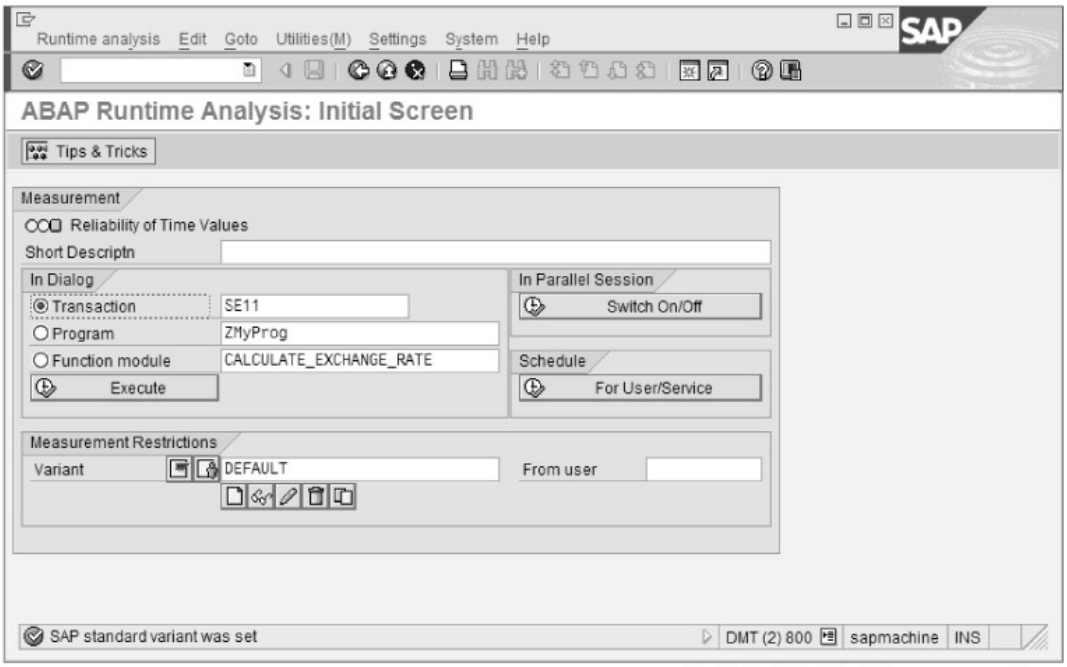

Figure 4.19: Displaying the initial screen of ABAP runtime analysis

The ABAP Runtime Analysis screen allows you to display the following:

- The execution time of each statement in a program.
- The tables accessed at run time.
- The execution time of all the statements. You can also group the statements based on the type of command.
- The execution time of each statement in the order in which they have been executed.
- The level of nesting of statements within subroutines.

With large applications, it is suggested that users analyze the entire application at the end. Now you can carry out the analysis and display, and interpret the results by doing the runtime analysis of programs with the help of the Runtime Analysis tool. Runtime Analysis also makes it possible to analyze the Web Dynpro ABAP and Business Server Pages (BSP) applications.

Performance Analysis

Performance Analysis, another tool of ABAP Workbench, performs a wide range of functions used to supervise and examine the performance of the system while accessing databases, performing locking activities, and remotely calling reports and transactions. It also uses the trace file, which keeps the record of the database accesses, reports or transactions, and then displays the performance log as a list. The Performance Analysis tool also helps analyze a trace file individually.

To open the Performance Analysis screen, either select `SAP Menu > Tools > ABAP Workbench > Test > SQL Trace` in the `SAP Easy Access` screen or enter the `ST05` transaction code in the `Command` field and press the `ENTER` key. Figure 4.20 shows the `Performance Analysis` screen:

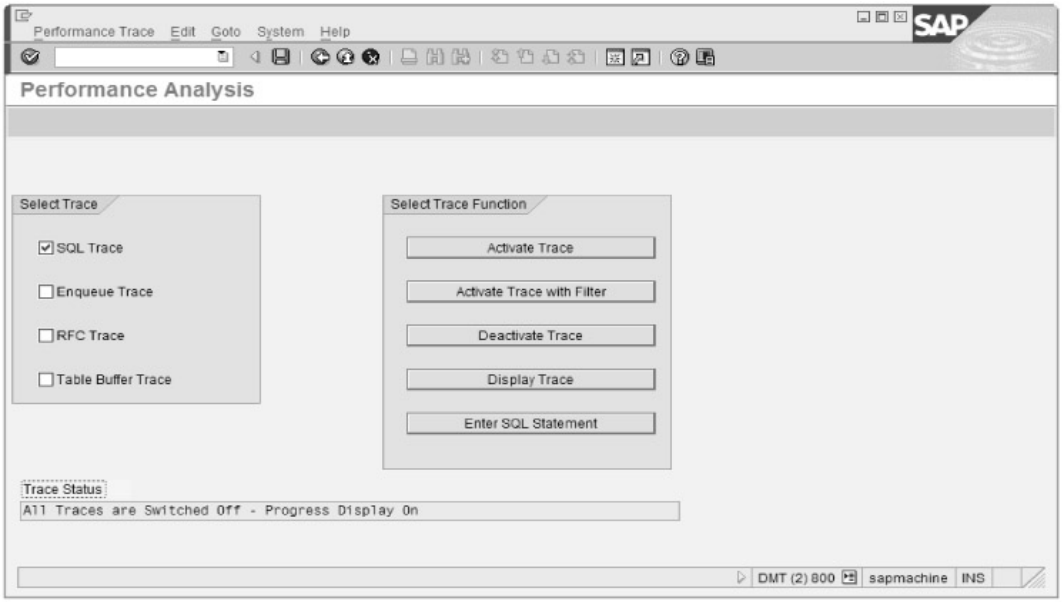

Figure 4.20: Displaying the screen of performance analysis

In Figure 4.20, the following types of traces are available in the `Performance Analysis` screen:

- **SQL Trace—** Keeps track of the various database accesses of reports and transactions.
- **Enqueue Trace—** Monitors the locking system.
- **RFC Trace—** Provides information about the remote function calls.
- **Table Buffer Trace—** Monitors the database calls of reports and transactions made through the table buffer.

Describing Web Services

A web service is defined as an independent and self-sustained unit of a software application hosted on the Internet. These self-sustained units of the software application collectively implement specific functionalities to execute the business logic of the application. You can use a web service to implement various functionalities, such as preparing the pay slips for the employees of an organization, calculating the income tax, and sending a price query to a material provider.

Web services solve the problem of integrating diverse computer applications that have been developed independently and run on a variety of software and hardware platforms. They allow applications to expose business operations to other applications, regardless of their implementation, by using the following standards:

- **Extensible Markup Language (XML)—** Represents the common markup language for communication.
- **Simple Object Access Protocol (SOAP)—** Specifies the common message format to exchange information.
- **Web Service Description Language (WSDL)—** Represents the common service specification format.
- **Universal Description, Discovery, and Integration (UDDI)—** Specifies a platform-independent framework to describe services, discover businesses, and integrate business services by using the Internet.

In addition to a common message format and markup language, there must be a general format that all service providers can use to specify service details, such as the service type, the service parameters, and the method of accessing the service.

The development and deployment of web services is governed by a standardization committee. Some enhanced standards, such as security standards or additional protocols, are continuously being updated and integrated into Web Service Framework by SAP. In the SAP system, ABAP Workbench provides an environment where you can publish a web service easily as well as call the web service. ABAP Workbench enables SAP Web Application Builder to act as a platform for web services, Internet Transactions Services (ITS) based web applications, and Business Service Pages (BSP) applications.

Web Application Builder for ITS

Web Application Builder acts as the platform to develop ITS based web applications. It also allows you to create web development objects, such as service files, HTML templates, and MIME objects. These web development objects are created as Repository objects, which are published on an ITS server. You can perform the following actions by using Web Application Builder for ITS:

- Create the Internet services for existing SAP transactions.
- Create and edit HTML templates for the screens of a transaction. An HTML template contains HTML and HTMLBusiness statements.
- Create objects for JavaScript files, which include MIME objects (icons, graphical images, and Java applets), to improve the layout.
- Create language-specific text.
- Create language-specific MIME objects.
- Execute a web application.

Note To open a Web application by logging on to a SAP system, the Internet Transaction Server (ITS) requires a relevant Internet service.

Note that Web Application Builder neither provides the syntax check facility for HTML or HTMLBusiness, nor does it provide debugger connections for HTMLBusiness.

Web Application Builder for BSP

Release 6.10 of the SAP system introduced another independent platform known as Web Application Builder, which is used to develop web applications based on the mySAP.com application server. web Application Builder is fully integrated into Object Navigator (`SE80`). New types of web applications are developed by Web Application Builder, popularly known as BSP applications. The following actions are supported by Web Application Builder for BSP:

- Creating BSP applications and their pages
- Editing the layout of BSPs by using HTML and the scripting languages such as ABAP and JavaScript
- Declaring the page-based data storage parameters related to the page
- Implementing the event handlers by using ABAP
- Defining page flow by using navigation requests
- Integrating the MIME Repository to store MIME objects
- Creating and defining themes for the layout adjustments
- Implementing the layout by using external tools

Note MIME Repository is a logical container that stores MIME objects, such as graphical images, icons, and style sheets, in an SAP system. All the objects stored in MIME Repository are arranged in a hierarchical order and can be displayed in a browser.

BSP Applications

A BSP application is an SAP application created by some external tools, such as Adobe GoLive, Dreamweaver, and Microsoft FrontPage 2000. The two main components of a BSP application are BSPs and MIME objects.

The user interface of a BSP application includes:

- Static websites
- Dynamically generated websites that are BSPs or templates that contain server-side scripting

- Various MIME objects, such as pictures, icons, sound files, and style sheets, that are part of a typical web application

Figure 4.21 shows the various components of a BSP application:

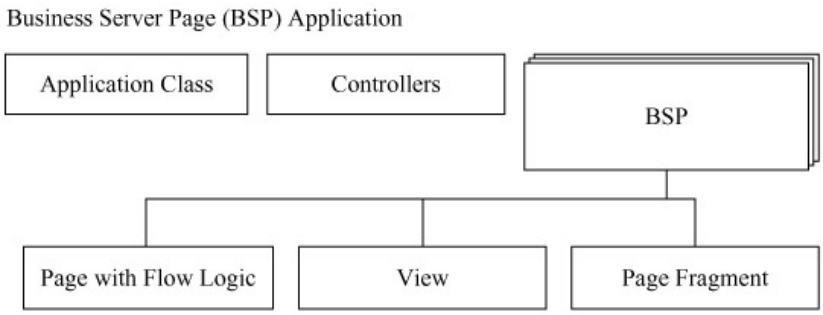

Figure 4.21: Showing the structure of a BSP application

In Figure 4.21, you can see that the structure of a BSP application contains one or more controllers and BSPs (including the elements, such as application classes, MIME objects, and themes).

Note BSPs can have different characteristics, such as a page with flow logic, a view, and a page fragment.

Describing XSLT Editor

Extensible Stylesheet Language Transformations (XSLT) XSLT Editor is a development tool, integrated in to ABAP Workbench, that is used to define Extensible Stylesheet Language (XSL) transformations. XSL transformations are executed on the Application server. Using XSLT Editor, the XML documents can be transformed into either XML or HTML documents, or ABAP data structures. Note that an XSLT program must be a repository object to define the transformation rules.

To call XSLT Editor, start Object Navigator by using the `SE80` transaction, select the package in Repository Browser, and double-click the required XSLT program under XSLT Transformations. XSLT Editor helps perform the following operations:

- Creating XSLT programs to define transformation rules
- Editing the source text by using Tag Browsers (XSLT, HTML, WML, or XHTML)
- Checking and activating XSLT programs
- Testing XSL transformations
- Creating breakpoints to debug XSL transformations

Summary

In this chapter, you have learned about ABAP Workbench, which is a programming environment for ABAP applications. The text also explained the tools available in ABAP Workbench, such as ABAP Dictionary, ABAP Editor, Screen Painter, and Menu Painter. In addition, the chapter described the role of Package Builder and explained the testing tools of ABAP Workbench, such as ABAP Debugger and Runtime Analysis, which are used to test and check the ABAP code written by users. Further, you also learned about web services and that ABAP Workbench offers an environment where you can publish, search, or call the web services. Finally, the chapter described XSLT Editor, which is a development tool to define XSL transformations.