UNIDAD

8

DIPLOMATURA EN PROGRAMACION ABAP

MÓDULO 8: PANTALLAS

PANTALLAS

Este módulo introduce los conceptos relacionados con la construcción por código de las pantallas de usuario y de selección de datos. Se desarrolla también el manejo de mensajes en ABAP. Más adelante, en la Unidad 10 y en la Unidad 11, se abordará el tema de la creación de pantallas en modo gráfico y en modo interactivo en ABAP.

Objetivos de la Unidad

Después de completar esta Unidad, se debería ser capaz de:

- Codificar pantallas de selección que interactúen con el usuario.
- Entender cómo las claves foráneas pueden ayudar a validar la entrada del usuario.
- Entender cómo se utilizan los *matchcodes* en los programas ABAP / 4.
- Utilizar elementos de formato para crear pantallas de selección eficientes y bien diseñadas.
- Utilizar las pantallas de selección para mantener la integridad de los datos.
- Utilizar mensajes para proporcionar una comunicación eficaz.

Programación orientada a eventos

El código del programa debe ser capaz de interactuar y comunicarse con el usuario final. Esto se hace en ABAP / 4 utilizando los eventos, que son invocados por las acciones de los usuarios.

Los bloques de procesamiento se definen por *palabras clave de evento* y por lo tanto se ejecutan ante la invocación de ciertos eventos relevantes.

De forma predeterminada, el evento `start-of-selection` se une a todos los eventos en ABAP / 4. En sus programas se puede definir un bloque de procesamiento y adjuntar este bloque a una palabra clave de evento.

Para un código generalmente fácil de leer, es una buena práctica definir los bloques de procesamiento en el orden secuencial en el que muy probablemente se activarán durante la ejecución de la pantalla de selección. También es una buena práctica utilizar los eventos más importantes para la programación de la pantalla de selección. Estos eventos son los siguientes:

- evento `initialization`
- evento `at selection-screen`
- evento `at user-command`

El uso del evento `initialization`

El siguiente código muestra la sintaxis para **el evento** `initialization` :

```
report ywhatyr.
* .. (más código)
data: p_year for sy-datum.
initialization.
if sy-datum >= '01012014'
p_year = '2014'.
else.
p_year = 'Año Anterior'.
endif.
```

En este ejemplo, cuando se ejecuta el programa para el que se define una pantalla de selección, el bloque de procesamiento `initialization` se ejecuta, estableciendo el campo de parámetro `p_year` igual a un valor en función de la fecha del sistema en el momento de la ejecución.

Este es el bloque en el que se especifican los valores iniciales por defecto de la pantalla de su selección sobre la base de cualquier criterio es necesario para mantener la integridad de los datos de entrada del usuario.

Algunos ejemplos incluyen barras de configuración de título, la asignación de elementos de texto a la interfaz gráfica de usuario (GUI) y el estado de los elementos del código de alguna función.

Se ha visto ya cómo funciona el evento initialization , por lo que ahora es el momento de echar un vistazo más de cerca a algunos de los usos de evento `at selection-screen` y del evento `at user-command` .

El uso del evento `at selection-screen`

El evento `at selection-screen` se procesa después de la entrada del usuario en la pantalla de selección activa. Esto puede ocurrir cuando el usuario pulsa una tecla de función o hace clic en un botón pulsador, así como una serie de otros elementos que pueden ser interactuados por el usuario.

Además, los controles de validación de datos, los mensajes de advertencia, los cambios de estado de interfaz gráfica de usuario, o incluso las ventanas pop-up que se puede llamar mediante el evento `at selection-screen` .

Se van a ver más ejemplos de esto más adelante, cuando nos fijemos en el formato de las pantallas de selección y los eventos involucrados con los elementos de formato de pantalla. Por ahora, sin embargo, vamos a echar un vistazo a un botón en una pantalla de selección y a cómo los eventos `at selection-screen` y `at user-command` se utilizan, con el próximo ejemplo.

El uso del evento `at user-command`

Los pulsadores, así como muchas otras opciones de la pantalla de selección basados en eventos, pueden ser muy útiles en el mantenimiento de la interacción del usuario y la validación de entrada del usuario.

En el próximo ejemplo, se explorará cómo utilizar pulsadores para invocar el evento `at user-command` y ver un ejemplo de cómo se pueden utilizar pulsadores (botones "*pushbutton*") para procesar la entrada del usuario.

Sintaxis para el evento `selection-screen pushbutton`

Se muestra ahora la sintaxis para el evento `selection-screen pushbutton`:

```
selection-screen pushbutton example1 user-command 1234.
```

Aquí el pushbutton se llama *example1* (este es el nombre del objeto) y el valor que almacenará cuando el usuario lo puse será *1234*.

Esta declaración, cuando se utiliza con el evento `at selection-screen` , es una gran manera de interactuar con el usuario cuando entra datos. La sintaxis es similar a la de `selection-screen comment`, excepto que los datos se pasan cuando el usuario presiona el botón. Al pulsar el botón, se determina el valor del campo `sy-ucomm` en el evento `at selection-screen`, y los campos de entrada son entonces importados.

Estos datos pueden ser validados y el usuario emite un mensaje dependiendo del propósito del botón.

Para clarificar más todo lo dicho, veremos a continuación un ejemplo de cómo se pueden utilizar dos pulsadores para determinar qué idioma reportar los datos seleccionados.

El siguiente código muestra, entonces, la sintaxis para dos pulsadores que se utilizan para elegir un idioma, creando dos botones, uno en la columna *10* y otro en la columna *50*, ambos de ancho *20* caracteres, llamados respectivamente *text-003* y *text-004*, que almacenan respectivamente los valores *engl* y *germ*:

```
selection-screen pushbutton 10(20) text-003 user-command engl.
selection-screen pushbutton 50(20) text-004 user-command germ.
at selection-screen.
at user-command.
case sy-ucomm.
when 'engl'.
     lang-english = 'Y'.
when 'germ'.
     lang-german = 'Y'.
endcase.
```

En este ejemplo, se puede comprobar cuál de los dos botones se presiona por parte del usuario mediante el uso de una declaración *case,* de acuerdo a la figura:

Cuando se dispara el evento `at user-command`, el campo `sy-ucomm` contiene el nombre de cuatro bytes único del elemento que el usuario ha seleccionado (en este caso, *'engl'* o *'germ'*). De esta manera, puede codificar varias validaciones de datos o entradas de usuario de comandos basado en la combinación de los datos introducidos y los elementos, en este caso los pulsadores, seleccionados por el usuario.

Ahora que se ha aprendido un poco acerca de la validación de datos mediante eventos en ABAP / 4, es hora de mirar a algunas de las técnicas de validación de datos definidas, utilizadas y mantenidas por el sistema. Vamos a ver primero las claves externas. Una clave externa (foreign key), como ya se mencionó anteriormente, es el conjunto de uno o más campos que representan la clave principal de una segunda tabla (por ejemplo, el CUIT del cliente en la tabla de compras es clave foránea en la misma, siendo clave principal en la tabla de clientes).

Validación de datos mediante el uso de claves externas

El modelo de datos relacional del entorno SAP puede contener muchas tablas, vistas, estructuras, y tablas vinculadas. Las claves externas, se puede decir, definen estas relaciones entre varias tablas. Las funciones que realizan las claves externas incluyen el suministro de datos de ayuda y la creación de objetos de diccionario, pero nos centraremos en la validación de datos. Después de todo,

mantener la integridad de los datos es uno de los objetivos principales en la definición de las pantallas de selección y el uso más importante de las claves externas.

Un campo de clave externa está, por definición, restringido a los valores que corresponden a los de la clave primaria de la tabla de verificación del campo de entrada. Así es como se hace el enlace entre las dos tablas. Una tabla puede ser considerada como tabla externa (o dependiente) o tabla de valores, principalmente debido a que incluye campos de clave externa que se asignan a los campos de clave principal de otra tabla, la que se conoce como la tabla de referencia, tabla primaria, tabla de chequeo o *check table*.

El campo de clave externa y la clave principal de una tabla primaria deben compartir el mismo dominio y también debe ser especificada una tabla de valores para ese dominio. Esto extiende, en cierto modo, la mera funcionalidad de comprobación de integridad de datos por lo general proporcionada por la tabla de valores.

Algunas tablas de chequeo pueden tener varios campos como clave principal. En tales casos, las asignaciones deben hacerse para cada campo cuando se inicia una relación de clave externa.

Las tres opciones son las siguientes:

- *Utilice una clave externa parcial.* En este caso, algunos campos no serán un factor al validar los valores aceptables para entradas en el campo de clave externa. Algunos campos se marcan como genérico (*generic*) en este caso y por lo tanto son ignorados por el sistema de validación.
- *Utilice una clave externa constante.* Para que la entrada de campo sea válida, el valor debe coincidir con el de un valor *constante* predefinido en la tabla de chequeo.
- *Crear una asignación de campo a campo.* Esta es la más profunda de las tres opciones. Cada campo de clave principal en la tabla de chequeo de entrada se corresponde con un campo de la tabla de clave externa y todos los campos clave sirven entonces para determinar las entradas válidas en la tabla de clave externa.

Básicamente, la forma en que funciona una clave externa se asemeja a la de una declaración `select` directa contra la tabla de chequeo, del campo con la clave externa. Más específicamente, cuando un campo de comprobación de clave externa se llena, la declaración `select` que se generó por el sistema SAP cuando se definió la clave foránea, es enviada por el programa. Si la tabla devuelve un

valor a partir de esa selección, la entrada es válida. Si no se encuentra el registro, la entrada de campo no es válida.

El siguiente código muestra la sintaxis de una declaración `select` generada por el sistema :

```
select * from table_1 where table_1-exam1 = fk_exam1
                        and table_1-exam2 = fk_exam2.
```

Este fragmento de código muestra un ejemplo de una declaración `select` generada por el sistema, que se llama cuando se introducen los datos en el campo con la definición de clave externa. En este escenario, una entrada en este campo de la pantalla sólo se permite si la sentencia `select` produce datos válidos de la tabla de verificación utilizando las entradas de datos realizadas en los campos `fk_exam1` y `fk_exam2` como claves.

Una cosa que se debe tener en cuenta al crear relaciones de clave externa es la cardinalidad. La cardinalidad de una clave externa, junto con su tipo, se conocen como los atributos semánticos de la clave externa. La cardinalidad de una clave externa es opcional. La cardinalidad no debe pasarse por alto, sin embargo, como buena práctica.

La *cardinalidad* es una descripción de la relación entre uno o más elementos de datos a uno o más de otro elemento de datos. Si se ha definido una clave externa que une dos tablas, el registro de la tabla de clave externa se refiere a un registro de la tabla de chequeo. Esta referencia se construye mediante la asignación de campos de una tabla, la tabla de clave externa, a los campos de clave principal de la otra tabla, la tabla de chequeo.

Existe la relación sólo si los campos de clave externa son del mismo tipo de datos y longitud que los campos de clave principal correspondientes. Una tabla primaria es la tabla que hace referencia a la misma clave externa. Esto es por lo general una tabla de valores, pero también puede ser una tabla que consta de un subconjunto de los contenidos de una tabla de valores. Una tabla de valores fija los valores válidos que se asignan al dominio del dato.

En algunos casos, una clave externa constante podría adaptarse mejor a sus necesidades. Este es el caso cuando todas las entradas válidas en el campo de entrada contienen un valor específico en el campo clave de la tabla de chequeo utilizada. Tras la consulta `select`, el campo *constante* se comprueba con el campo de clave principal que contiene el valor fijo.

Validación de datos usando *Matchcodes*

Otra forma útil de mantener la integridad de los datos durante la entrada del usuario es utilizar *matchcodes*. Un *matchcode* es un objeto que se utiliza para buscar los registros de datos en el diccionario de datos SAP. Más específicamente, los matchcodes acceden los registros de datos de una manera similar a un índice en el sentido que no son necesarios todos los campos clave, pero difieren de los índices en que los matchcodes puede contener más de una tabla en el rango de selección.

Así es como funciona. En primer lugar un objeto matchcode se define con las tablas primarias y secundarias pertinentes y campos significativos designados. Este objeto identifica entonces todos los caminos posibles a los registros de datos requeridos. A continuación, se crean los matchcode ID mediante la asignación de una ruta definida por el objeto matchcode. Los únicos campos de datos que son permitidos en este ID se basan totalmente en el objeto matchcode. Por lo menos un ID debe ser declarado para cada objeto matchcode. Los objetos matchcode se almacenan en el diccionario de SAP como cualquier otro objeto.

Un matchcode se puede almacenar de dos maneras diferentes:

- *Físicamente.* En este contexto, los datos se almacenan en una tabla separada en el sistema SAP
- *Lógicamente.* Esta opción configura los datos matchcode temporalmente durante el acceso. Este acceso es administrado por una vista de base de datos.

Los objetos matchcode se pueden crear sencillamente como cualquier otro objeto del DDIC. Se le asigna un nombre que empieza con Y o Z. Introduzca un nombre para el nuevo objeto, introduzca un texto descriptivo al entrar en la pantalla de atributos, junto con la tabla principal del matchcode, y pulse el botón Guardar. La tabla principal representa la tabla de fuente primaria para la búsqueda de campo posterior. Tablas secundarias también se agregar haciendo doble clic en la tabla principal o pulsando el botón Tablas. Estas tablas deben ligarse a la tabla principal a través de claves externas. A continuación se siguen los pasos tradicionales de creación de objetos de DDIC (activación, salvado, generación, transporte). Con estos sencillos pasos se ha creado un objeto matchcode que ahora se puede utilizar para mantener la integridad de los datos. Existen cinco tipos de matchcodes (al crear uno, se le preguntará cuál asigna). He aquí una descripción de cada uno de estos cinco tipos:

- I-Arreglado por la base de datos.
- S-Organizado por la interfaz de base de datos.
- A-Auto-organizado por el programa de sistema SAPMACO.
- P-Arreglado por un programa de aplicación.
- K-También organizado por un programa de aplicación.

La diferencia entre *P* y *K* es que el primero corresponde a almacenamiento físico y el segundo a almacenamiento lógico.

Luego de crear el objeto matchcode, se creará entonces un matchcode ID con un nombre de 4 caracteres que empiece con Y o Z, y uno de los 5 tipos recién descriptos.

Sintaxis de uso de un *matchcode*

La siguiente es la sintaxis para usar matchcodes para la validación de parámetros de entrada de una pantalla de selección:

```
tables: saptab
parameter: example like saptab-field matchcode object exam.
select single field from saptab where field = example.
```

En este ejemplo se utiliza el objeto matchcode *exam*, que ya se ha definido con un matchcode ID, para validar la entrada de datos en el campo de parámetro *example*. Este objeto *exam* contiene los datos de relaciones necesarias para mantener la integridad de la entrada del usuario.

Formateo de pantallas de selección

Ahora que se han explorado algunas herramientas para validar los datos de entrada en ABAP / 4, vamos a tomar algún tiempo para aprender más sobre el diseño de formatos eficaces para una pantalla de selección. Al definir pantallas de entrada de de datos para usar la información para reportes, varios elementos de selección se pueden combinar para mantener la integridad de los datos, y ser todavía fácil de usar. Estos elementos incluyen los campos de entrada estándar, tales como parámetros, así como otros tales como casillas de verificación, botones radio y botones de usuario. La siguiente sección le echa un vistazo a cada una de ellas por separado y da algunos ejemplos de dónde son mejor utilizados. Tendremos la oportunidad de combinarlos en un ejercicio.

Usar instrucciones de pantalla de selección (*selection screen*)

Los elementos de la pantalla de selección (*selection screen*) se pueden combinar en unidades cohesivas llamados *bloques* . Estos bloques lógicos son, en esencia, una característica cosmética de la pantalla, que encapsula una combinación de elementos de entrada de la pantalla y se pueden crear con un título de marco descriptivo. Los bloques lógicos ayudan a que las opciones de selección sean más fácil de entender y usar.

Sintaxis de `selection-screen ... block ... with frame ...`

La siguiente es la sintaxis para una sentencia
`selection-screen ... block ... with frame ...:`

```
selection-screen begin of block block0 with frame title text-000.
```

Además de la declaración `block` , las pantallas de selección se pueden personalizar mediante la utilización de elementos de formato, tales como los siguientes declaraciones `selection-screen`:

- `selection-screen comment` -Esto colocará un comentario en la pantalla de selección.
- `selection-screen uline` -Esto colocará un subrayado en la pantalla en un lugar determinado con una longitud determinada.
- `selection-screen position` -Esta es una herramienta muy útil que especifica un comentario para el siguiente parámetro en la pantalla. se puede usar tanto con `selection-options` , así como con `parameters`.
- `selection-screen begin-of-line` y `selection-screen end-of-line`-Todos los campos de entrada definidos entre estos dos estados se colocan uno junto al otro en la misma línea.
- `selection-screen skip` *`n`* -Esta sentencia crea una línea en blanco para *`n`* líneas en la pantalla de selección.

La mayor parte de los elementos incluidos en la sección anterior eran cosméticos, en la siguiente sección se verán algunos de los elementos de la pantalla de selección que son más específicos para el procesamiento de la entrada de datos.

Selection Screen *parameters*

El declaración `parameters` (o `parameter`, que es lo mismo) en pantallas de selección crea una estructura de datos en el programa que mantiene la entrada introducida por el usuario en tiempo de ejecución en el campo de parámetro.

Los parámetros se utilizan cuando la entrada de campo requerida es un solo valor, en lugar de un rango de valores. El rango válido de valores puede ser determinado por el usuario haciendo clic en la flecha hacia abajo (lista desplegable o combo) o presionando *F4* mientras el cursor está en el campo generado por la sentencia `parameters` . Esta tabla de valores puede ser mantenida por el programador para incluir valores personalizados para las transacciones o informes específicos. La forma más sencilla de crear ayuda para la tecla F4 es mediante la asignación de una tabla de chequeo para el dominio del campo en el diccionario ABAP / 4.

Un *dominio*, como ya se dijo, describe las propiedades de los campos de una tabla. Define el rango de valores de datos válidos para un campo y características específicas del campo.

Los dominios determinan atributos como el tipo de campo, longitud, y las posibles relaciones de clave externa. La modificación de un dominio cambiará automáticamente los atributos de todos los elementos de datos adjuntos a ese dominio. Esto es debido al hecho de que los elementos de datos en un dominio heredan todas las propiedades del dominio al que se hace referencia.

Para ver o realizar acciones en un dominio, comience en la pantalla de diccionario de datos. Marque el botón de radio correspondiente a Dominios y seleccione el botón de las acciones que desea invocar (por ejemplo, "Visualizar" o "Display"). Como se podrá ver, las características de dominio se muestran en la pantalla.

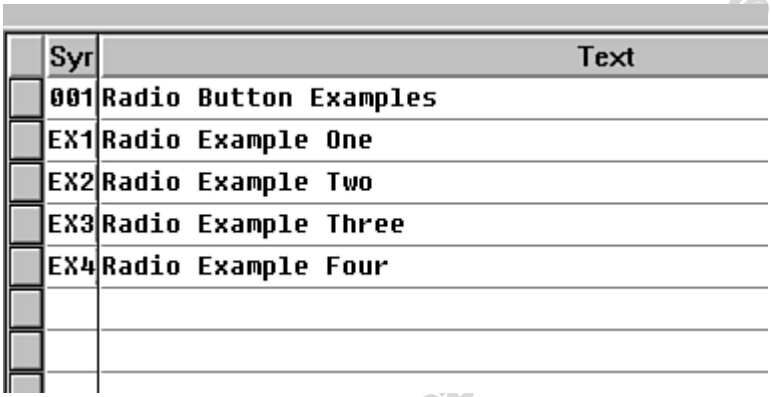
Con la declaración `parameters`, se puede incluir palabras clave que pueden restringir la entrada con ciertas opciones. Estas tres palabras clave son los siguientes:

- `lower case` -Con esto incluido en la declaración `selection`, el valor en minúsculas que se entró en el campo de entrada, no se convierte automáticamente a mayúsculas, que es lo que le sucede a todos los campos de entrada en ABAP / 4 por defecto en tiempo de ejecución.
- `obligatory` -Esto prohíbe campos de entrada en blanco.
- `default` -Este término mantiene un valor inicial predeterminado para el parámetro.

Se deben manejar *elementos de texto* para los parámetros para proporcionar al usuario una buena explicación de lo que el campo de entrada de la pantalla de selección representa en el contexto del programa. Estos elementos de texto son

independientes del lenguaje y se muestran en el idioma de inicio de sesión del usuario.

Para mantener los elementos para sus pantallas de selección, elija la ruta del menú *Ir a -> Elementos de texto* del Editor ABAP / 4. Se puede mantener los elementos de texto para su programa desde aquí. Ahora vamos a ver un ejemplo de edición de símbolos de texto (*text symbols*), que es uno de los tres tipos de elementos de texto que existen en ABAP / 4, y es el que se usa con *parameters*. Veamos como ejemplo la siguiente figura:



Syr	Text
001	Radio Button Examples
EX1	Radio Example One
EX2	Radio Example Two
EX3	Radio Example Three
EX4	Radio Example Four

La figura muestra la pantalla que permite editar los símbolos de texto que creó en su programa. Si desea añadir más, escriba un número y el texto y haga clic en el botón *Guardar*.

Selection Screen *checkbox*

Los parámetros pueden ser creados como campos de datos que contienen sólo un valor de entrada y también se pueden crear como casillas de verificación. Cuando los parámetros toman la forma de casillas de verificación (*checkbox*), se declaran como tipo `c` y mantienen el valor de `x` cuando están marcadas y el espacio en blanco como valor cuando están desactivadas. Un buen uso del parámetro `checkbox` es para pedir al usuario, por ejemplo, que indique si quiere que ciertos componentes de un informe se muestren o no.

Sintaxis de selection-screen *checkbox*

La siguiente es la sintaxis para `selection-screen checkbox`:

```
parameters: testparm as checkbox default 'X'.
```

En este ejemplo, el valor inicial se establece a `X` (marcado), para el procesamiento lógico del programa. Las casillas de verificación, a diferencia de los botones de radio, no son excluyentes entre sí, por lo que el usuario puede tener tantas casillas marcadas como las que se generen en la pantalla de selección.

Como se ha aprendido, es mejor utilizar la declaración `parameters` al solicitar un único valor de entrada. Si la entrada requerida es mejor representada por un rango o una serie de valores, es más eficiente utilizar la declaración `select-options`.

La sentencia `select-options` genera una tabla interna de selección que contiene la entrada para los campos involucrados.

Selection Screen *select-options*

La instrucción `select-options` se utiliza de manera similar a la declaración `parameters` en el sentido que se crea un criterio de selección para un campo de base de datos. La principal diferencia entre los dos es que la sentencia `select-options` crea dos campos de entrada que contienen tanto un campo `from` *(de)* y un campo to *(a)* , en lugar de sólo una única entrada de campo.

Sintaxis para `select-options`

La sintaxis de esta sentencia es la siguiente:

```
select-options ex sele for table-field default 'VALUELOW' to 'VALUEHI'.
```

La declaración `select-options` muestra una línea que por lo general tiene dos campos de datos de entrada. Esto puede ser restringido a uno solo mediante el uso de la cláusula `no-intervals` en la sintaxis de `select-options`. Por ejemplo, si el programa no requiere una `A` (*to*) para el campo de entrada en `select-options`, pero aún desea utilizar la declaración `select-options`, debe incluir esta cláusula `no-intervals` . Esta cláusula, así como la cláusula `no-extension`, se estudiarán más en el ejemplo al final de esta unidad. Por ahora, sin embargo, vamos a realizar a una discusión sobre el formato de la tabla interna de selección de `select-options`.

Esta tabla interna se mantiene con el formato del siguiente campo si se hace clic en la flecha situada a la derecha del campo `to` de `select-options` . Al hacer clic en esta flecha aparece una pantalla de entrada de selecciones múltiples que llena

la tabla de selección interna. Esta tabla a continuación contiene los atributos clave de los datos de entrada, incluyendo los valores de `SIGN`, `OPTION` ,`LOW` y `HIGH`.

Estas características de la declaración `select-options` pueden tener las siguientes condiciones:

- `SIGN` -Puede significar `INCLUSIVE` , que es el predeterminado, o `EXCLUSIVE`.
- `OPTION` -pueden contener valores `de BT` (entre), `CP` (contiene el patrón), `EQ` (igual) y `GE` (mayor o igual que).
- `LOW` : mantiene el valor de entrada en el campo `FROM` .
- `HIGH`: mantiene el valor de entrada en el campo `TO`.

Botones de radio en la pantalla de selección

Además de los campos `parameters` y `select-options`, la declaración `radiobutton` en la pantalla de selección es una gran manera de mantener la integridad de los datos al procesar la entrada del usuario en tiempo de ejecución. Con el fin de crear parámetros como botones de radio, debe utilizarse la cláusula `radiobutton group`.

Sintaxis para `radiobutton group`

La siguiente es la sintaxis para `radiobutton group`:

```
selection-screen begin of block rad_blk with frame title text-000.
parameters: rad_ex1 radiobutton group one,
            rad_ex2 radiobutton group one,
            rad_ex3 radiobutton group one.
selection-screen end of block rad_blk.
```

En este ejemplo se genera un grupo de tres parámetros como botones de radio. Como se puede ver, estos botones de radio se agrupan en un solo bloque en la pantalla. Esta es una buena práctica de programación, ya que ayuda al usuario a darse cuenta de que todos ellos pertenecen al mismo grupo de datos de entrada. Estos parámetros se utilizan mejor para seleccionar un valor único de una opción múltiple de valores, y debe contener al menos dos botones por grupo.

Sólo uno de estos tres botones de los ejemplos se puede elegir en tiempo de ejecución debido a su inclusión en el grupo *one*. La integridad de los datos se mantiene, dado que esta es una gran manera de solicitar datos mutuamente excluyentes por parte del usuario.

Esto se demostrará de nuevo en la siguiente sección donde hay un ejemplo de una pantalla de selección simple que utiliza muchos elementos de la pantalla de selección acerca de los que hemos aprendido.

Ejemplo de programa con pantalla de selección

El siguiente es un ejemplo de código para una pantalla de selección que contiene la mayoría de los elementos básicos estudiados:

```
report Zpant message-id Y6.
* Database Table Definitions
tables: spfli.
selection-screen skip 1.
 selection-screen begin of block block0 with frame title text-000.
 selection-screen skip 1.
selection-screen begin of line.
selection-screen pushbutton 10(20) text-003 user-command engl.
selection-screen pushbutton 50(20) text-004 user-command germ.
selection-screen end of line.
 selection-screen end of block block0.
* Selection parameters
selection-screen skip 2.
selection-screen begin of block block1 with frame title text-001
                                                 no intervals.
selection-screen begin of line.
parameters: p_ex1 radiobutton group rad1 .
selection-screen comment 5(30) text-ex1.
selection-screen end of line.
parameters: p_jdate1 type d default sy-datum.
selection-screen skip 1.
selection-screen begin of line.
parameters: p_ex2 radiobutton group rad1 .
selection-screen comment 5(30) text-ex2.
selection-screen end of line.
select-options: s_jcity for spfli-cityfrom.
selection-screen skip 1.
selection-screen begin of line.
parameters: p_ex3 radiobutton group rad1.
selection-screen comment 5(20) text-ex3.
selection-screen end of line.
parameters: p_jcity2 like spfli-cityto.
selection-screen skip 1.
selection-screen begin of line.
parameters: p_ex4 radiobutton group rad1 .
selection-screen comment 5(30) text-ex4.
selection-screen end of line.
select-options: s_jair for spfli-airpfrom no-extension no intervals.
selection-screen end of block block1.
selection-screen skip.
```

```
selection-screen begin of block block2 with frame title text-002
no intervals.
selection-screen begin of line.
parameters: P_ex5 as checkbox.
selection-screen comment 5(30) text-ex5.
selection-screen end of line.
selection-screen skip.
selection-screen begin of line.
parameters: P_ex6 as checkbox.
selection-screen comment 5(30) text-ex6.
selection-screen end of line.
selection-screen skip.
selection-screen begin of line.
parameters: P_ex7 as checkbox.
selection-screen comment 5(30) text-ex7.
selection-screen end of line.
selection-screen end of block block2.
* AT selection-screen.
AT selection-screen.
  if ( p_ex1 = 'X' ).
    message E017 with '1'.
  endif.
  if ( p_ex2 = 'X').
    message E017 with '2'.
  endif.
  if ( p_ex3 = 'X' ) .
message E017 with '3'.
endif.
  if ( p_ex4 = 'X' ) .
message E017 with '4'.
  endif.
  if p_ex5 = 'X'.
  perform get_price.
  else.
    message I017 with 'Sin Precio de Venta Informado'.
  endif.
  if p_ex6 = 'X'.
  perform get_cost.
  else.
    message I017 with 'Sin Costo Informado'.
  endif.
  if p_ex7 = 'X'.
  perform get_revenue.
  else.
    message I017 with 'Sin Ganancia Informada'.
  endif.
form get_cost.
...
endform.
form get_revenue.
...
endform.
form get_price.
...
endform.
```

Debido a que los botones de opción son excluyentes entre sí, es una buena idea mantener a todos en el mismo bloque de procesamiento en la pantalla de selección.

Como se menciona en este último ejemplo, es una buena práctica utilizar declaraciones *message* como avisos de validación de datos de entrada en tiempo de ejecución, lo cual veremos a continuación.

Uso de la instrucción *message*

Los mensajes se mantienen y almacenan en la tabla `T100` y se puede acceder desde el *Workbench* ABAP. Un programador eficaz emite mensajes que son descriptivos y ayudan al usuario a comprender la naturaleza del flujo del programa.

A cada declaración *message* individual se le puede asignar tipos de mensajes que tienen diversos efectos sobre el resultado del programa.

- S-Success, este mensaje se muestra en la pantalla siguiente en la línea de estado.
- A-Abend, se detiene la operación actual y se muestra un mensaje hasta que el usuario lo confirma.
- E-Error, los datos de usuario de entrada deben continuar.
- W-Warning, el usuario puede modificar cualquiera de los datos de entrada o pulsar *Enter* para continuar.
- I-Information, el usuario debe pulsar *Enter* para continuar en el mismo punto en el programa.

Debe especificar una declaración `message-id` al comienzo de su programa, junto a `report`.

Sintaxis para la declaración `message-id`

La siguiente es la sintaxis para la declaración `message-id` en `report`:

```
report example line-count 65 line-size 132 message-id fs.
```

Este ID de dos caracteres *fs* junto a un número de mensaje de tres dígitos, y junto con el lenguaje de logon, determinan las clases de mensaje.

SAP ha hecho fácil incluir declaraciones `message` en un programa mediante el uso de la declaración `insert` en el editor de ABAP / 4.

Una clase de mensajes se puede crear, como cualquier otro objeto, con el Diccionario de Datos de ABAP / 4.

Algunos consejos sobre el uso de mensajes:

- Recuerde utilizar los mensajes de una manera que ayude al usuario a manejar los datos de entrada de manera más eficiente.
- Recuerde hacer un buen uso de las clases de mensaje existentes en el sistema antes de crear una nueva.
- **NO** olvidar especificar el `message-id` en el comienzo de su informe, en la sentencia `report`.

Resumen

- ABAP / 4 es un lenguaje de programación orientado a eventos, alguno de los cuales son `at selection-screen` e `initialization`, así como `at user-command`, para validar la entrada del usuario, utilizando una serie de técnicas.
- Las claves externas se pueden utilizar para comprobar la integridad de los datos de entrada utilizando los campos de clave principal de una tabla de verificación.
- Los *matchcodes* utilizan datos agrupados lógicamente para validar la entrada del usuario sobre la base de criterios de selección especificados cuando se utiliza junto con parámetros de la pantalla de selección y opciones de selección.
- Los mensajes pueden ser utilizados para comunicar el estado del programa y la validez de los datos introducidos, así como para interactuar con el usuario de una manera que pueda procesar datos de forma más eficiente.
- Los elementos de texto deben ser mantenidos por elementos de la pantalla de selección de modo que el usuario tenga una buena comprensión de los usos de los diferentes criterios de selección de los campos en la pantalla.