**Buscar**

**Visión de conjunto**

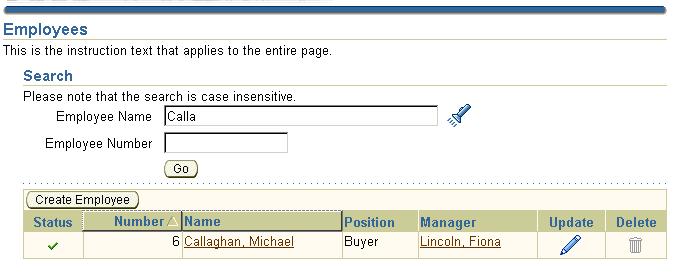
**Regiones de búsqueda principal**

El [Oracle Browser Look-and-feel (BLAF) Pauta de interfaz de usuario: Búsqueda y consulta de plantillas](https://translate.google.com/translate?hl=es&prev=_t&sl=en&tl=es&u=http://uistandards/blaf/guidelines/search.html) [ [Versión OTN](https://translate.google.com/translate?hl=es&prev=_t&sl=en&tl=es&u=http://otn.oracle.com/tech/blaf/specs/search_flow.html) ] se describen las siguientes opciones de diseño básico para la búsqueda:

* [Búsqueda sencilla](https://translate.googleusercontent.com/translate_f#simple) - presenta a los usuarios un número limitado de criterios de búsqueda básico (generalmente 1 - 4 campos) con un control de ocultar / mostrar opcional para ver criterios de búsqueda adicionales.
* [Búsqueda Avanzada](https://translate.googleusercontent.com/translate_f#adv) - presenta a los usuarios con amplios criterios de búsqueda y capacidades de búsqueda de energía, incluyendo la posibilidad de especificar de forma declarativa "Y" o "O" búsquedas y los operadores de tipo de datos específico en campos individuales. Por ejemplo, un valor "Salario" debe ser mayor que 100000.
* [(Personalizable usuario) Vistas](https://translate.googleusercontent.com/translate_f#views) - presenta a los usuarios una lista personalizable de búsquedas guardadas, con nombre.
* [Control de tiempo de ejecución - OAQueryBean](https://translate.googleusercontent.com/translate_f#runtime) - describe las instrucciones especiales para la manipulación de la región de consulta en tiempo de ejecución.
* [Regiones de consulta múltiples](https://translate.googleusercontent.com/translate_f#multiple) - describe las instrucciones especiales para la visualización de múltiples regiones de consulta en una sola página.

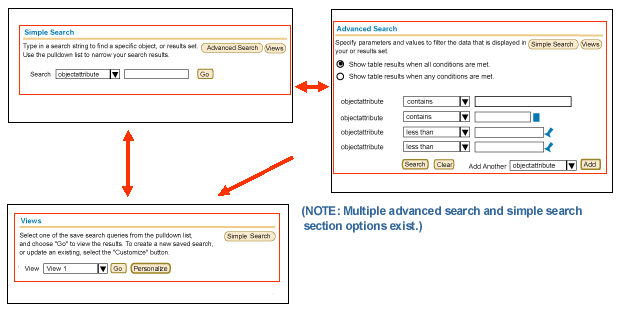
En todos los casos, estas regiones se presentan en la misma página con los resultados asociados, como se muestra en la siguiente ilustración de una búsqueda simple:

Figura 1: Ejemplo de una tabla simple de búsqueda y resultados en la misma página



La búsqueda simple, la búsqueda avanzada y las vistas personalizables por el usuario se pueden mostrar individualmente o en combinación para proporcionar una búsqueda extremadamente flexible en una lista de objetos. La Figura 2 muestra cómo el usuario alterna entre estos paneles de búsqueda si los tres están habilitados en una sola página.

Figura 2: alternar la navegación entre una búsqueda simple, una búsqueda avanzada y una región de vistas personalizables por el usuario en la misma página



**Regiones suplementarias de búsqueda**

Las Directrices de la interfaz de usuario también describen dos diseños para presentar "accesos directos" de búsqueda especiales a los usuarios:

* [Búsqueda rápida](https://translate.googleusercontent.com/translate_f#quick) - como se muestra en la Figura 3, esto hace que inmediatamente debajo del menú y presenta al usuario un único campo de búsqueda y poplist opcional de tipos de búsqueda de objetos
* [De navegación de la búsqueda](https://translate.googleusercontent.com/translate_f#side) - como se muestra en la Figura 4, esto también presenta al usuario un único campo de búsqueda y poplist opcional de tipos de objetos de búsqueda, pero la región de búsqueda hace que la navegación en el lado del lugar.

Figura 3: Configuraciones de búsqueda rápida válidas según las pautas de la interfaz de usuario de BLAF

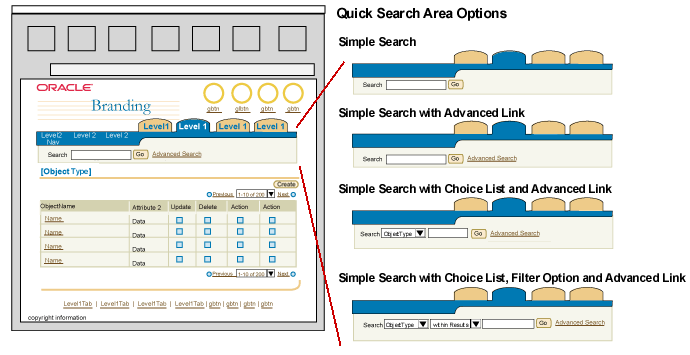
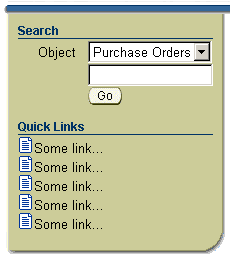


Figura 4: Ejemplo de búsqueda de navegación lateral.



**Contenido**

* [Visión de conjunto](https://translate.googleusercontent.com/translate_f#overview)
  + [Regiones de búsqueda principal](https://translate.googleusercontent.com/translate_f#primary)
  + [Regiones suplementarias de búsqueda](https://translate.googleusercontent.com/translate_f#suppl)
* [Consideraciones de implementación de búsqueda](https://translate.googleusercontent.com/translate_f#srchconsid)
  + [Usar las regiones de consulta](https://translate.googleusercontent.com/translate_f#qregions)
  + [Cómo usar las regiones de anidación](https://translate.googleusercontent.com/translate_f#nregions)
  + [Usando Search Persistence](https://translate.googleusercontent.com/translate_f#persist)
* [Búsqueda simple](https://translate.googleusercontent.com/translate_f#simple)
  + [Implementación declarativa: búsqueda basada en resultados](https://translate.googleusercontent.com/translate_f#simple_rbc)
  + [Implementación declarativa: criterios de personalización de automóviles](https://translate.googleusercontent.com/translate_f#simple_auto)
  + [Búsqueda manual](https://translate.googleusercontent.com/translate_f#simple_manual)
* [Búsqueda Avanzada](https://translate.googleusercontent.com/translate_f#adv)
  + [Implementación declarativa: búsqueda basada en resultados](https://translate.googleusercontent.com/translate_f#adv_rbc)
  + [Implementación declarativa: criterios de personalización de automóviles](https://translate.googleusercontent.com/translate_f#adv_autocc)
  + [Búsqueda manual](https://translate.googleusercontent.com/translate_f#adv_manual)
  + [Control de tiempo de ejecución](https://translate.googleusercontent.com/translate_f#rtc)
    - [Validaciones personalizadas](https://translate.googleusercontent.com/translate_f#cust_validations)
    - [Proceso de manejoFormRequest](https://translate.googleusercontent.com/translate_f#handlepfr)
* [(Personalizable por el usuario) Vistas](https://translate.googleusercontent.com/translate_f#views)
  + [Implementación](https://translate.googleusercontent.com/translate_f#views_impl)
  + [Vistas no personalizables](https://translate.googleusercontent.com/translate_f#nonpersv)
* [Comportamiento OAQueryBean en detalle](https://translate.googleusercontent.com/translate_f#runtime)
  + [Descripción general del objeto Criteria](https://translate.googleusercontent.com/translate_f#criteria_obj)
  + [Generación de cláusula WHERE predeterminada](https://translate.googleusercontent.com/translate_f#where)
  + [Anulando la generación de la cláusula WHERE](https://translate.googleusercontent.com/translate_f#where_override)
* [Múltiples regiones de consulta](https://translate.googleusercontent.com/translate_f#multiple)
* [Búsqueda rápida](https://translate.googleusercontent.com/translate_f#quick)
* [Búsqueda de navegación lateral](https://translate.googleusercontent.com/translate_f#side)
* [Consideraciones de personalización](https://translate.googleusercontent.com/translate_f#pers)
* [Problemas conocidos](https://translate.googleusercontent.com/translate_f#known_issues)
* [Información relacionada](https://translate.googleusercontent.com/translate_f#info)

**Consideraciones de implementación de búsqueda**

En esta sección se discute el uso de [regiones de consulta](https://translate.googleusercontent.com/translate_f#qregions) , [regiones anidadas](https://translate.googleusercontent.com/translate_f#nregions) , y [buscar la persistencia](https://translate.googleusercontent.com/translate_f#persist) en la aplicación de Búsquedas. Ver la [búsqueda simple](https://translate.googleusercontent.com/translate_f#simple) y [la búsqueda avanzada](https://translate.googleusercontent.com/translate_f#adv) secciones de implementación a título de orientación.

**Usar las regiones de consulta**

Esta sección proporciona una visión general de [las regiones de la consulta](https://translate.googleusercontent.com/translate_f#srchmodes) , y discute [las limitaciones de la región de consulta](https://translate.googleusercontent.com/translate_f#queryconstr) y [recomendaciones criterios de búsqueda](https://translate.googleusercontent.com/translate_f#srchcrit) .

Puede crear búsquedas simples y avanzadas como si fueran regiones regulares y luego implementar el botón asociado para ejecutar la consulta. Sin embargo, si es posible, use la región de consulta especial.

**Nota:** *Debe* utilizar la región consulta si desea implementar los puntos de vista-personalizable por el usuario. Por otra parte, se recomienda *encarecidamente* que utilice la región de consulta para poner en práctica la búsqueda en un HGrid como la alternativa requiere un esfuerzo significativo de su parte.

Por último, la [búsqueda rápida](https://translate.googleusercontent.com/translate_f#quick) y [lateral de navegación Buscar](https://translate.googleusercontent.com/translate_f#side) interfaces de usuario no pueden ser implementados utilizando la región consulta. Sus implementaciones manuales se describen a continuación.

**Comprender las regiones de consulta**

Cuando se agrega una región consulta a una región pageLayout, OA Marco genera automáticamente un oracle.apps.fnd.framework.webui.beans.layout.OAQueryBean que, dependiendo de su configuración, trabaja en conjunto con un niño [de mesa](https://translate.googleusercontent.com/feat_tables.htm) , [mesa de avanzada](https://translate.googleusercontent.com/feat_advtables.htm) o [HGrid](https://translate.googleusercontent.com/feat_hgrid.htm) para implementar cualquier combinación de búsqueda simple, búsqueda avanzada y paneles de visualización. OA Framework genera automáticamente botones según corresponda para alternar entre las regiones aplicables.

Los paneles de búsqueda simple y de búsqueda avanzada se pueden construir utilizando tres modos diferentes, que indican el nivel de región y la automatización de la construcción de búsqueda.

|  |  |  |
| --- | --- | --- |
| **Modo de construcción** | **Impacto en la construcción de la región** | **Impacto en la ejecución de búsqueda** |
| resultsBasedSearch | OA Framework representa automáticamente las regiones de búsqueda simple y avanzada en función de los elementos consultables designados en la tabla asociada o HGrid.  **Nota:** Las regiones de búsqueda automáticamente incluyen tanto un un botón Borrar Go y. | OA Marco ejecuta automáticamente la búsqueda subyacente cuando el usuario selecciona el botón Ir.  Si el objeto de vista subyacente está "sucio" (tiene cambios pendientes), OA Framework muestra un mensaje de error en lugar de ejecutar la consulta. |
| autoCustomizationCriteria | OA Framework representa automáticamente tanto las regiones de búsqueda simple como avanzada basándose en las regiones de Búsqueda simple y Búsqueda avanzada correspondientes que usted define y especifica como elementos secundarios con nombre de la región de consulta.  **Nota:** Las regiones de búsqueda incluyen automáticamente un botón Ir. Además, la región de búsqueda avanzada incluye un botón Borrar. | OA Marco ejecuta automáticamente la búsqueda subyacente cuando el usuario selecciona el botón Ir. Sin embargo, los desarrolladores *deben* definir explícitamente las asignaciones entre los elementos del panel de búsqueda y artículos en la región de mesa / HGrid.  Como en el caso resultsBasedSearch, OA Framework muestra un mensaje de error en lugar de ejecutar la consulta si el objeto de vista subyacente tiene cambios pendientes. |
| ninguna | Las regiones de búsqueda se representan según las regiones Búsqueda simple y Búsqueda avanzada que define y especifica como elementos secundarios con nombre de la región de consulta.  **Nota:** Es *necesario* implementar su propio botón Ir en este modo. | La búsqueda subyacente debe ser ejecutada por el desarrollador. |

Las instrucciones detalladas para implementar las opciones de búsqueda simple, búsqueda avanzada y vistas se proporcionan a continuación.

**Restricciones de la región de consulta**

* [LOV Choice](https://translate.googleusercontent.com/feat_lov.htm#choice) componentes *no* son compatibles. *No se* debe marcar una columna de tabla de resultados **messageLovChoice**como "consultable" cuando se utiliza una región de consulta.

**Recomendaciones de criterios de búsqueda**

* [El campo de texto LOV](https://translate.googleusercontent.com/feat_lov.htm#text) componentes (artículos messageLovInput) utilizados como criterios de búsqueda dentro de una región de consultas deben tener su propiedad deshabilitar la validación establece en **True.** Como resultado, los usuarios perderán el comportamiento de validación que se produce cuando salgan del campo de texto LOV (autocompletado). Sin embargo, esta pérdida es menor en comparación con el comportamiento extraño que se produce cuando la propiedad Desactivar validaciónse establece en **False** para el elemento messageLovInput en el contexto de un grano web de consulta.

Cuando la propiedad Desactivar validación se establece en su valor predeterminado de **False,** el comportamiento siguiente:

* + En la búsqueda simple, se realiza una búsqueda "igual". Debido a que no hay una forma de especificar el ID para el formValue messageLovInput en una búsqueda simple, validar-en-submit *no* funciona a pesar de que deshabilitar la validación es **falso.** Además, un "es igual a" búsqueda se realiza para el elemento messageLovInput porque Desactivar validación es **falso.** Esto puede generar confusión para el usuario, ya que no hay ninguna indicación de que el valor del elemento messageLovInput deba ser exacto.
  + En el modo de búsqueda avanzada autoCustomizationCriteria (ACC), se producen errores incómodos. En el modo ACC de búsqueda avanzada, es posible especificar el ID formValue para el mensajeLovInput. Sin embargo, la OA Marco le pide al usuario con torpeza "Seleccionar un valor válido" cuando el usuario introduce un valor parcial en el campo de texto de LOV y presiona el botón Ir. En su lugar, debe mostrar los resultados de la búsqueda en la tabla de resultados sin mostrar ningún error de LOV.
  + Escribiendo un valor en el campo de texto de lista de valores y seleccionando Borrar hace aparecer la ventana modal LOV. Aunque este comportamiento es esperado desde la perspectiva de LOV, es confuso cuando el mensajeLovInput es un criterio de búsqueda en el contexto de una región de consulta. Al establecer la propiedad Desactivar validación en **True,** no se produce este comportamiento.

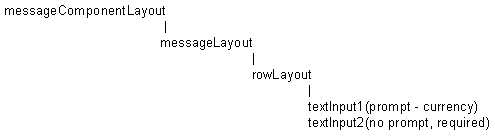
**Usar regiones anidadas**

Para crear un diseño en el panel de búsqueda simple usando un queryBean en modo ACC, como por ejemplo:

C:\oracle\OA_Framework\jdevdoc\WebHelp\devguide\feat\images\feat_nregions1.gif

\* representa una entrada de texto sin un aviso.

define el siguiente diseño:



**Nota:** Los elementos **TextInput** *no* son hijos directos de la **messageComponentLayout** o **messageLayout,** que están dentro de un contenedor de **rowLayout** anidada. (Las regiones de criterios anidados ahora son compatibles).

OA Marco *siempre* hace que la región AdvancedSearch como la región filtro en Crear vista Página, basado en los metadatos especificados en la página de consulta. Si un panel AdvancedSearch *no* está definido, OA Framework utiliza la sencilla metadatos de búsqueda avanzada para crear la región de filtro de búsqueda.

Por lo tanto, si necesita definir un diseño con criterios de regiones anidados *debe* definir una región de búsqueda avanzada en la región de consulta y crear asignaciones adecuadas para los elementos de criterios, que están anidados en el sencillo panel de búsqueda. Esto asegura que la OA Marco *siempre* utiliza los metadatos desde el panel AdvancedSearch definido y no hay problemas de diseño en Crear vista Página.

**Nota:** Si *no* desea que la región avanzada que se mostrará en la página que sigue *debe* definir y establecer la propiedad Incluir panel Avanzadas del grano de consulta a **falso.**

**Usando Search Persistence**

En el pasado, si un usuario realizaba una búsqueda y luego profundizaba en una página de detalles, los criterios de búsqueda se perdían cuando el usuario navegaba de regreso a la página de Búsqueda, a menos que el desarrollador retuviera la AM o creara una implementación personalizada para persistir los criterios de búsqueda. A partir de la versión 12, los desarrolladores ya no necesitan una implementación personalizada de persistir criterios de búsqueda, ya que la región **consulta** ahora con un mecanismo de búsqueda de persistencia automática en su lugar.

En general, aún debe conservar el AM a favor del uso del mecanismo de persistencia de búsqueda automática por motivos de optimización del rendimiento. Sin embargo, si conserva la AM presenta un problema de escalabilidad porque la vista subyacente objeto (VO) tenía cientos de atributos, entonces usted debe apagar retainAM y permitir que la región **de consulta** para manejar automáticamente todos los criterios de unión y utilizar el mecanismo automático de búsqueda de persistencia .

**Nota:** Si usted conserva el estado de la mañana, el mecanismo automático de búsqueda de persistencia no estará activo. El mecanismo de persistencia de búsqueda es automático solo si el estado de AM no se retiene.

Cuando un usuario selecciona el botón Ir para iniciar una búsqueda en la región de consulta, el OAQueryBean almacena automáticamente los criterios de búsqueda en la sesión de usuario. Después de que el usuario profundiza en la página de detalles de los resultados de búsqueda y luego regresa a la página de búsqueda, OAQueryBean restaura los criterios de búsqueda de la caché y vuelve a ejecutar la consulta utilizando esa crítica guardada.

**Nota:** Puesto que se vuelve a ejecutar la consulta, no se conserva ninguna clase, la navegación o la interfaz de usuario transitoria estado de la tabla de resultados. Guardar todo el estado de la tabla presenta una sobrecarga alta, que es lo que esta característica particular está tratando de evitar.

**Control de tiempo de ejecución**

Si [reemplazar el valor predeterminado donde la generación de la cláusula](https://translate.googleusercontent.com/translate_f#where_override) de la región de **consulta** utilizando getCriteria o getNonViewAttrCriteria en su controlador, debe hacer algunos cambios en su código con el fin de persistir los criterios de búsqueda en la sesión. Los cambios que debe realizar se pueden categorizar por tres casos diferentes.

**Caso 1: su controlador maneja un subconjunto de los campos de búsqueda al no definir el mapeo de búsqueda para ellos.**

Para el subconjunto de campos de búsqueda manejados por su controlador, que necesita para obtener los valores de los criterios de búsqueda restauradas de los granos web que corresponden a los campos de búsqueda y se unen a la VO en el método processRequest de su controlador. A su regreso de una página detalle, los criterios de búsqueda introducidos en los campos de búsqueda se restauran en el ciclo de la processRequest OAQueryBean. El siguiente es un ejemplo del cambio de código que debe hacer en este caso:

public void processRequest (OAPageContext pageContext, OAWebBean webBean)

{   
 super.processRequest (pageContext, webBean);   
 OAQueryBean queryBean = (OAQueryBean) webBean;

String value = (String) ((OAMessageTextInputBean)   
 webBean.findChildRecursive ("trabajo")). getValue (pageContext);   
 String currentPanel = queryBean.getCurrentSearchPanel ();

if (SEARCH.equals (currentPanel) && value! = null)   
 {

// Construir la cláusula WHERE apropiada para el trabajo   
 // y configurarlo en su VO usando el método initQuery.   
 OAApplicationModule am = pageContext.getRootApplicationModule ();

Parámetros serializables [] = {valor};   
 am.invokeMethod ("initQuery", parámetros);   
 }   
 }

**Caso 2 - El controlador utiliza getNonViewAttrCriteria.**

Su controlador llama a la API getNonViewAttrCriteria en processFormRequest para devolver un diccionario de criterios para el subconjunto de los campos que desea manejar. En este caso, después de regresar de una página de detalles, puede obtener acceso a los mismos criterios en el diccionario processRequest de su controlador llamando a la siguiente API en OAQueryBean (si el panel de búsqueda actual es un panel simple o avanzado):

Dictionary [] getNonViewAttrCriteriaFromSearchCache (OAPageContext)

Utilice getNonViewAttrCriteria (OAPageContext) si el panel actual es un Puntos de vista   
panel.

Puede entonces el usuario este diccionario criterios para manejar los criterios a sí mismo en processRequest. Usted debe extraer esta lógica a un método personalizado, como "handleCriteria" y llamarlo desde processFormRequest y processRequest. El siguiente es un ejemplo del cambio de código que debería hacer en este caso:

public void processRequest (OAPageContext pageContext, OAWebBean webBean)

{   
 super.processRequest (pageContext, webBean);   
 OAQueryBean queryBean = (OAQueryBean) webBean

Dictionary [] nonViewAttrCriteria = null;

String panel = queryBean.getCurrentSearchPanel ();   
 if (SEARCH.equals (panel) || ADVANCED\_SEARCH.equals (panel))

nonViewAttrCriteria = queryBean.getNonViewAttrCriteriaFromSearchCache (pageContext);   
 else if (CUSTOMIZE.equals (panel))

nonViewAttrCriteria = queryBean.getNonViewAttrCriteria (pageContext);

if (nonViewAttrCriteria! = null)   
 handleCriteria (pageContext, nonViewAttrCriteria);   
 }

**Nota:** Debe utilizar el modo Auto Criterios de personalización del grano de consulta.

**Nota:** No utilice getNonViewAttrCriteriaFromSearchCache si está utilizando getCriteria en processRequest con el fin de suprimir la generación DONDE cláusula y ejecución de la consulta en la página Ver.

**Caso 3 - Su controlador utiliza getCriteria.**

Usted controlador llama a la API getCriteria en processFormRequest para devolver el diccionario criterios enteras y desencadenar OA Framework para omitir la cláusula DONDE generación y consulta de ejecución. En este caso, después de regresar de una página de detalles, puede obtener acceso a los mismos criterios en el diccionario processRequest de su controlador llamando a la siguiente API en el OAQueryBean (si el panel de búsqueda actual es un panel simple o avanzado):

Diccionario [] getCriteriaFromSearchCache (OAPageContext)

Utilice getCriteria (OAPageContext) si el panel actual es un panel Vistas.

A continuación, puede implementar su propio método personalizado como "handleCriteria" para utilizar este diccionario criterios para manejar los criterios y ejecutar la consulta en processRequest. El siguiente es un ejemplo del cambio de código que debería hacer en este caso:

public void processRequest (OAPageContext pageContext, OAWebBean webBean)

{   
 super.processRequest (pageContext, webBean);   
 OAQueryBean queryBean = (OAQueryBean) webBean;   
  
 Diccionario [] searchCriteria = null;   
 String panel = queryBean.getCurrentSearchPanel ();   
 if (SEARCH.equals (panel) || ADVANCED\_SEARCH.equals (panel))

searchCriteria = queryBean.getCriteriaFromSearchCache (pageContext);

else if (CUSTOMIZE.equals (panel))   
 searchCriteria = queryBean.getCriteria (pageContext);

if (searchCriteria! = null)

handleCriteria (pageContext, searchCriteria);   
 }

**Nota:** Debe utilizar el modo Auto Criterios de personalización del grano de consulta.

**Nota:** No utilice getCriteriaFromSearchCache si está utilizando getCriteria en processRequest con el fin de suprimir la generación DONDE cláusula y ejecución de la consulta en la página Ver.

**Contexto de persistencia de búsqueda**

Algunas páginas de búsqueda se utilizan en el contexto de un registro maestro. Por ejemplo, el producto Oracle Sourcing rastrea las negociaciones y las líneas de negociación. La búsqueda realizada dentro de una negociación para líneas de negociación es específica para esa negociación.

Para habilitar la persistencia búsqueda de un contexto específico, puede llamar a la API setSearchPersistenceContext en OAQueryBean:

setSearchPersistenceContext (contexto de cadena)

**Nota:** Se debe llamar a esta API a partir del método processRequest de un controlador definido por encima de la región de **consulta.**

Normalmente, la referencia del documento completo de la región de **consulta** se utiliza como clave de caché para los criterios de búsqueda en una sesión de usuario. Al llamar a esta API, el valor de contexto que se pasa se añade a la clave de caché existente para esta región de **consulta** en particular por lo que los criterios de búsqueda siempre se almacenan en caché basan en ese contexto.

Para el ejemplo de negociaciones descrito anteriormente, dado que la región de búsqueda de líneas de negociación está siempre en el contexto de una línea de negociación particular, el contexto de persistencia de búsqueda debe establecerse en el Id. De línea de negociación.

**Búsqueda simple**

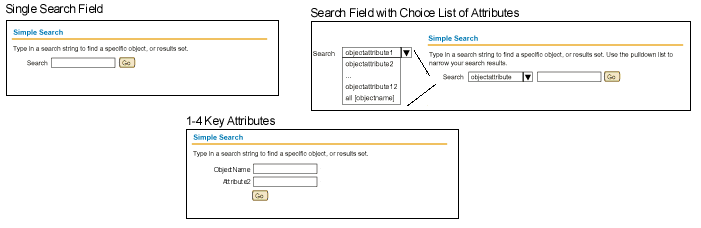
Como se describió anteriormente, una búsqueda simple muestra una pequeña cantidad de elementos del criterio de búsqueda en un formato restringido como se muestra en las Figuras 5 y 6 a continuación.

* Para implementar una lista simple de elementos que se asignan directamente a las columnas de tabla asociadas, aproveche el bean Query para configurar rápidamente una búsqueda completamente automatizada. Ver la [Implementación declarativa: Resultados de Búsqueda en base](https://translate.googleusercontent.com/translate_f#simple_rbc) a las instrucciones.
* Para implementar una búsqueda simple con elementos personalizados, debe manipular la búsqueda en sí. Por ejemplo, el usuario selecciona un valor poplist que *debe* ser traducido a un intervalo de fechas particular. Use el bean Query con su manejo de consultas personalizado. Ver la [Implementación declarativa: Criterios de personalización automática](https://translate.googleusercontent.com/translate_f#simple_auto) .

**Nota:** La costumbre simple panel de búsqueda que usted cree *debe* ser una región messageComponentLayout.

* Para implementar el campo de búsqueda con una lista emergente de atributos de búsqueda, no puede usar el bean Query.Ver la [Implementación declarativa: Búsqueda manual](https://translate.googleusercontent.com/translate_f#simple_manual) de instrucciones.
* El modo Ninguno se proporciona para compatibilidad con versiones anteriores. Para cualquier consulta nuevas regiones que se crean, la *búsqueda basada en los resultados* y las opciones *de personalización automática Criterios* deben satisfacer todas sus necesidades.

Figura 5: Ejemplo de diseños válidos de región de búsqueda simple según las pautas de interfaz de usuario de BLAF.



El botón Ir normalmente se rindió por debajo de los elementos de búsqueda. Sin embargo, como se muestra en los ejemplos anteriores, si una búsqueda simple sólo tiene un **messageTextInput,** o uno **messageTextInput** y uno **messageChoice** (donde **messageChoice** actúa como un poplist condicional), y luego en el botón Ir se rindió junto a la **messageTextInput.**

**Nota:** El **messageChoice** actúa como un poplist condicional cuando el indicador se establece sólo para el **messageChoice** y *se* pone durante la **messageTextInput.**

Para una región de búsqueda que contiene un control Ocultar / Mostrar, el botón Ir hace debajo del control de ocultar / mostrar, tanto en el estado colapsado y el ampliado.

Figura 6: Ejemplo de región de búsqueda simple con el menú desplegable "Mostrar más opciones de búsqueda" ocultar / mostrar control.

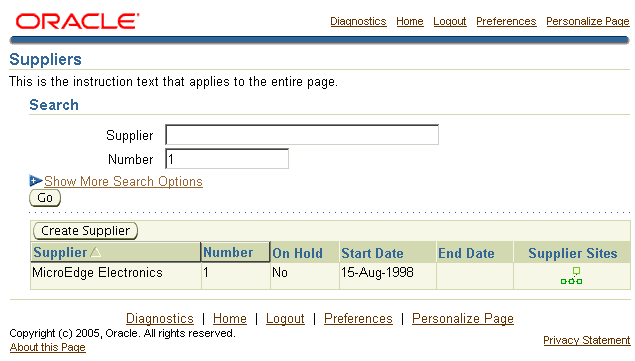
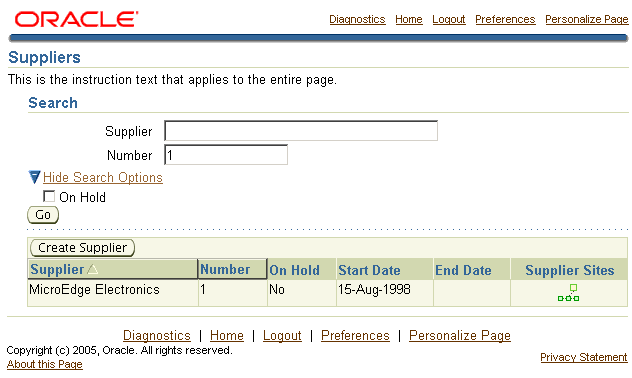


Figura 7: Ejemplo de región de búsqueda simple con la opción "Mostrar más opciones de búsqueda" ocultar / mostrar control.



**Implementación declarativa: búsqueda basada en resultados**

Para crear un panel de búsqueda simple en el modo de búsqueda basado en resultados:

Paso 1: crea el objeto de vista que usarás para buscar. En este modo, no hay necesidad de crear un método initQuery () en el objeto a fin de establecer la cláusula WHERE, se unen los criterios de búsqueda y ejecutar la consulta. OA Framework maneja esto en su nombre.

**Consejo:** Si la definición de la vista del objeto requiere valores vinculados, o se *debe* agregar manualmente una relación cláusula WHERE a sí mismo cada vez que se ejecuta la consulta, se puede proceder de una de dos maneras. En cualquier caso, OA Framework agrega los criterios de búsqueda del usuario a su cláusula WHERE inmediatamente antes de ejecutar la consulta, por lo que se conservan las características de la cláusula WHERE que defina.

1. Reemplazar el método executeQuery () en su \* VOImpl para modificar su cláusula WHERE y luego llamar super.executeQuery ().Esto se recomienda si ejerce la misma lógica cada vez que se consulta el objeto de vista.
2. Manejar la prensa del botón generada Go y llamar a un método initQuery () en el objeto de vista donde se puede modificar la cláusula WHERE y / o conjunto unen valores.

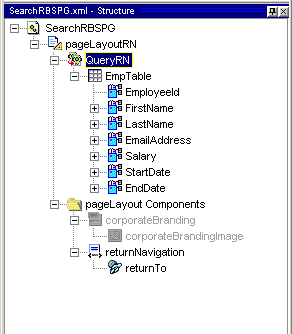
Paso 2: Seleccione su región pageLayout en el panel Estructura JDeveloper, haga clic derecho y seleccione Nuevo> Región. Set Estilo de la región para **consultar,** especifique un ID que cumple con las [Normas Marco de la OA de archivos](https://translate.googleusercontent.com/cs/cs_filedir.htm) , (todas las regiones y objetos que se crean deben cumplir con estas normas) y establezca las siguientes propiedades:

* Modo de construcción - se establece en **resultsBasedSearch.**
* Incluir sencillo panel - se establece en **True.**
* Panel Inicial - si se incluye la búsqueda avanzada y / o la opción Vistas, especifique el panel de la pantalla inicial.
* Las instrucciones simples de búsqueda - texto de la instrucción por el simple región de búsqueda.
* Botón simple etiqueta - si se incluye la búsqueda avanzada y / o la opción Vistas, especifique la etiqueta del botón que hace que la OA Marco para alternar la búsqueda simple. La etiqueta predeterminada es "Búsqueda simple".

**Nota:** En el modo de búsqueda basada en los resultados, la OA Marco genera automáticamente un "Búsqueda sencilla" de cabecera para que en tiempo de ejecución para contener la región de consulta. No es necesario agregar esto manualmente a la jerarquía de componentes.

Paso 3: Seleccione su región de consulta en el panel de estructura, haga clic derecho y seleccione Nuevo> Región mediante el asistente para crear rápidamente su mesa o HGrid regiones de resultados de búsqueda. Recuerde que debe dar a la región un IDcompatible con los estándares y especificar conjuntos de atributos apropiados para los elementos asociados. Ver las [Tablas](https://translate.googleusercontent.com/feat_tables.htm) y [HGrid](https://translate.googleusercontent.com/feat_hgrid.htm)documentación para obtener información adicional sobre cómo crear estos componentes. (Esto es particularmente importante para HGrid ya que hay propiedades adicionales que debe configurar en ese componente para habilitar la búsqueda de forma adecuada).La Figura 8 ilustra una estructura típica de búsqueda basada en resultados.

Figura 8: Ejemplo de una estructura de búsqueda basada en resultados en la biblioteca de muestras ToolBox.



Paso 4: Seleccione la región / HGrid tabla que creó en el paso 3 y ajustar la búsqueda de residencias en **True** para los elementos que se deben utilizar como criterios de búsqueda. Estos artículos se incluyen automáticamente en la región de búsqueda sencilla de acuerdo con las reglas descritas a continuación, y OA Marco ejecuta automáticamente la consulta correspondiente cuando el usuario selecciona el botón Ir.

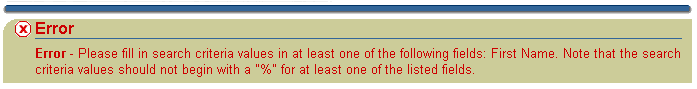
* Si el elemento de la tabla de búsqueda es una lista pop, el elemento de la región de búsqueda correspondiente es la misma lista pop.
* Si el elemento de la tabla de búsqueda es un campo de entrada de texto, el elemento de la región de búsqueda correspondiente es un campo de entrada de texto o un campo de fecha, según corresponda, en función del tipo de datos de la columna.
* Si el elemento de la tabla de búsqueda es un campo de entrada de texto LOV, el elemento de la región de búsqueda correspondiente es el mismo (con la misma definición de LOV).
* Si el elemento de la tabla de búsqueda es un valor de solo visualización (como un mensaje Texto Estimado, un texto sin formato, un texto formateado, etc.), OA Framework genera un campo de entrada de texto o una entrada de fecha archivados según el tipo de datos de la columna.

Paso 5: Establecer la propiedad Criterios de búsqueda selectiva **True** para cualquier criterio de búsqueda elementos que aseguran una consulta performant. Ver la [OA marco Ver estándares de codificación](https://translate.googleusercontent.com/cs/cs_viewgeneral.htm) para obtener información adicional.

**Nota:** OA Marco trata a estos valores como se requiera *de forma individual:* si el usuario entra en uno cualquiera de los elementos requeridos, la consulta puede proceder. Actualmente no puede exigir que el usuario ingrese los criterios de búsqueda en más de uno de estos elementos requeridos.

Como se muestra en la Figura 9 a continuación, si el usuario no puede especificar al menos uno de los valores de búsqueda requeridos, aparece un mensaje de error con la lista de elementos candidatos para que el usuario pueda ingresar un valor y continuar con la búsqueda.

Figura 9: ejemplo de un error se muestra cuando el usuario no introduzca un valor para al menos uno de los "selectivos criterios de búsqueda" artículos en la búsqueda simple (Nótese que el mensaje de error real puede diferir ligeramente).



Paso 6: guarda tu trabajo.

**Nota:** Si crea una búsqueda basada en los resultados, no añada regiones de búsqueda personalizada que se requieren para el modo de "criterios de personalización de automóviles" se describe a continuación.

**Implementación declarativa: criterios de personalización de automóviles**

**Advertencia:** Cuando se implementa tanto una búsqueda simple y avanzada utilizando criterios de personalización automática, la búsqueda simple *debe* ser un subconjunto de la búsqueda avanzada. Esto es necesario porque en el módulo Personalización, si intenta guardar una búsqueda, OA Framework siempre muestra la búsqueda avanzada.

Para crear un panel de búsqueda simple en el modo de criterios de personalización automática:

Paso 1: Crear el objeto vista que va a utilizar para la búsqueda como se describe en los [resultados de búsqueda basados](https://translate.googleusercontent.com/translate_f#simple_rbc) sección anterior.

Paso 2: Seleccione su región pageLayout en el panel Estructura JDeveloper, haga clic derecho y seleccione Nuevo> Región. Set Estilo de la región para **consultar,** especifique un ID que cumple con las [Normas Marco de la OA de archivos](https://translate.googleusercontent.com/cs/cs_filedir.htm) , (todas las regiones y objetos que se crean deben cumplir con estas normas) y establezca las siguientes propiedades:

* Modo de construcción - se establece en **autoCustomizationCriteria.**
* Incluir sencillo panel - se establece en **True.**
* Panel Inicial - si se incluye la búsqueda avanzada y / o la opción Vistas, especifique el panel de la pantalla inicial.
* Las instrucciones simples de búsqueda - texto de la instrucción por el simple región de búsqueda.
* Botón simple etiqueta - si se incluye la búsqueda avanzada y / o la opción Vistas, especifique la etiqueta del botón que hace que la OA Marco para alternar la búsqueda simple. La etiqueta predeterminada es "Búsqueda simple"

Paso 3: Seleccione su región de consulta en el panel Estructura, haga clic derecho y seleccione Nuevo> Región mediante el asistente para crear rápidamente su mesa o HGrid regiones de resultados de búsqueda. Recuerde que debe dar a la región un IDcompatible con los estándares y especificar conjuntos de atributos apropiados para los elementos asociados. Ver las [Tablas](https://translate.googleusercontent.com/feat_tables.htm) y [HGrid](https://translate.googleusercontent.com/feat_hgrid.htm)documentación para obtener información adicional sobre cómo crear estos componentes. (Esto es particularmente importante para HGrid ya que hay propiedades adicionales que debe configurar en ese componente para habilitar la búsqueda de forma adecuada).

Paso 4: crea tu región de búsqueda simple personalizada.

* Paso 4.1 Seleccione su región de consulta en el panel Estructura, haga clic derecho y seleccione Nuevo> simpleSearchPanel.
* Paso 4.2 JDeveloper crea automáticamente dos regiones para usted: un encabezado con la propiedad de texto se establece en Búsqueda sencilla, y debajo de esto, un messageComponentLayout. Dar ambas regiones un documento de identidad con los estándares. Si va a incluir solamente una simple búsqueda en tu región consulta, cambie la propiedad Text de la cabecera para buscar y, a continuación, configurar el [messageComponentLayout](https://translate.googleusercontent.com/feat_layout.htm#mcl) incluir sus campos de criterios de búsqueda.
* Paso 4.3 Establecer la propiedad Criterios de búsqueda selectiva **True** para cualquier criterio de búsqueda elementos que aseguran una consulta performant. Ver la [OA marco Ver estándares de codificación](https://translate.googleusercontent.com/cs/cs_viewgeneral.htm) para obtener información adicional.

**Nota:** OA Marco trata a estos valores como se requiera *de forma individual:* si el usuario entra en uno cualquiera de los elementos requeridos, la consulta puede proceder. Actualmente no puede exigir que el usuario ingrese los criterios de búsqueda en más de uno de estos elementos requeridos.

Como se muestra en [la Figura 9](https://translate.googleusercontent.com/translate_f#fig09) anteriormente, si el usuario no especificar al menos uno de los valores de búsqueda necesarios, un mensaje de error con la lista de elementos candidatos por lo que el usuario puede introducir un valor y continúe con la búsqueda.

Paso 5: Cree las correlaciones de consulta entre su región de búsqueda personalizada simple y las columnas de tabla de modo OA marco puede ejecutar automáticamente la búsqueda cuando el usuario selecciona el botón Ir.

**Nota:** Si necesita realizar el mapeo mediante programación (por ejemplo, que se derivan de un rango de fechas cláusula WHERE basa en un valor poplist de "ANY\_TIME", "LAST\_2\_WEEKS", "last\_month" y así sucesivamente) omita este paso y poner en práctica la ejecución de búsquedas como se describe en [control de Tiempo de ejecución: OAQueryBean](https://translate.googleusercontent.com/translate_f#runtime) continuación.

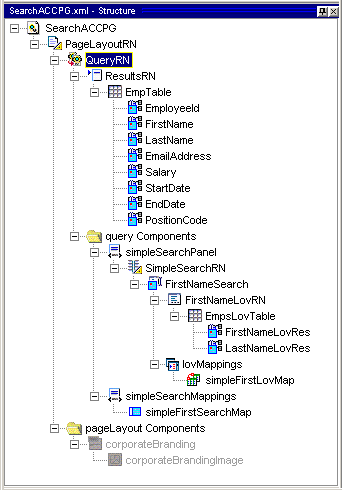
* Paso 5.1 Seleccione su región consulta en el panel Estructura, haga clic derecho y seleccione Nuevo> simpleSearchMappings.
* Paso 5.2 Configure la asignación predeterminada que JDeveloper crea para usted. Esta asignación identifica el artículo en la región de Búsqueda cuyo valor se usa para consultar el artículo correspondiente (columna de la tabla) en la región de Resultados.

**Importante:** Usted está asignando valores de los criterios de búsqueda en este paso, no mostrar los valores. Por ejemplo, la región de búsqueda incluye un LOV que devuelve una clave única a un campo de formulario oculto y un valor de visualización al campo de entrada LOV. *Debe* asignar tanto en el campo de formulario oculto *y* el valor de visualización a sus columnas de las tablas correspondientes.

* Paso 5.3 (opcional) Si tiene más de una asignación, seleccione el nodo simpleSearchMapping en el panel Estructura, haga clic derecho y seleccione Nuevo> queryCriteriaMap. Configure el nodo como lo hizo en el Paso 7.2. Repita según sea necesario hasta que todos los elementos del criterio de búsqueda estén asignados a la tabla de resultados / elementos HGrid.

La Figura 10 ilustra una estructura de página de búsqueda simple creada en el modo de criterios de personalización automática.

Figura 10: Ejemplo de una estructura de búsqueda simple de criterios de personalización automática en la biblioteca de ejemplos de OA Framework ToolBox.



Paso 6: guarda tu trabajo.

**Búsqueda manual**

Para crear una región de búsqueda que no se puede implementar con el bean Query (siempre debe usar el Bean Query si es posible), siga estos pasos:

Paso 1: Cree el objeto de vista que usará para buscar y agréguelo al módulo de aplicación de su página. Enlace los elementos de resultado a este objeto de vista.

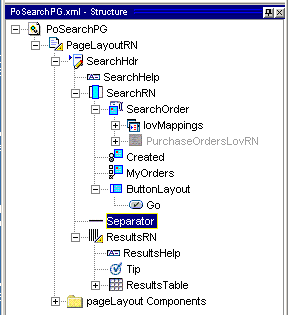
Paso 2: crea el diseño para tu región de búsqueda. Su región debe tener un encabezado con el título "Buscar". Para obtener ayuda con esta tarea, consulte [Diseño de página (Cómo colocar contenido)](https://translate.googleusercontent.com/feat_layout.htm) . Ver [Ocultar / Mostrar](https://translate.googleusercontent.com/feat_hideshow.htm) si desea agregar un "Mostrar más opciones" para su búsqueda. El botón Ir debería ser un submitButton para que el formulario se envía cuando el usuario lo selecciona.

Paso 3: Establecer la propiedad Criterios de búsqueda selectiva **True** para cualquier criterio de búsqueda elementos que aseguran una consulta performant. Ver la [OA marco Ver estándares de codificación](https://translate.googleusercontent.com/cs/cs_viewgeneral.htm) para obtener información adicional.

**Nota:** OA Marco trata a estos valores como se requiera *de forma individual:* si el usuario introduce cualquiera de los elementos requeridos, la consulta puede proceder. Actualmente no puede exigir que el usuario ingrese los criterios de búsqueda en más de uno de estos elementos requeridos.

Paso 4: Entre la región de búsqueda y la región de resultados, agregue un elemento "Separator" tal como lo exigen las pautas de BLAF. La Figura 11 muestra una estructura de página de búsqueda típica.

Figura 11: Ejemplo de región de búsqueda configurada manualmente desde OA Framework ToolBox Tutorial



Paso 5: Añadir un método initQuery () a su objeto vista para manejar la unión de los entrantes criterios de búsqueda para cláusula WHERE de la vista del objeto.

**Nota:** Aunque se puede manipular la cláusula WHERE en su initQuery () método, así como vinculantes parámetros de búsqueda, por razones de rendimiento, debería incluir idealmente un estática cláusula WHERE en la definición del objeto de vista. Hacer cambios dinámicos en la cláusula WHERE *solamente* si es absolutamente necesario debido a la complejidad y variedad de los criterios de búsqueda.

El siguiente ejemplo de la PoSimpleSummaryVOImpl en la OA marco de cuadro de herramientas muestra un caso de uso apropiado para donde la manipulación cláusula. Demuestra la forma correcta de unir una serie dinámica de parámetros mediante la unión de Oracle-estilo. También demuestra el uso de un executeQuery booleana parámetro para controlar si esta subventana debe consultar de forma explícita en sí o diferir a la manipulación tabla de OA Framework. Ver [Ver objetos en detalle -> Directrices de inicialización](https://translate.googleusercontent.com/bus/bus_vo.htm#init)para obtener información adicional:

import java.util.Vector;

oracle.bali.share.util.IntegerUtils de importación;

oracle.jbo.domain.Number importación;

oracle.apps.fnd.framework.OANLSServices de importación;

oracle.apps.fnd.framework.server.OADBTransaction importación;

oracle.apps.fnd.framework.server.OAViewObjectImpl importación;

...

public void initQuery (String orderNumber, cadena creada, showMyOrders String, Boolean executeQuery)

{

StringBuffer whereClause = new StringBuffer (100);

parámetros del vector = new Vector (3);

int clauseCount = 0;

int bindCount = 0;

setWhereClauseParams (null);

// Siempre restablezca

// Tenga en cuenta que el uso de fijaciones de tipo Oracle, mientras que requieren un poco

// más esfuerzo que el de estilo ANSI unión, se requiere de un marco

// código.

if ((orderNumber! = null) && (! ( "" es igual a (orderNumber.trim ()))))

{

Número ORDERNUM = null;

tratar

{

ORDERNUM = new Number (orderNumber);

}

catch (Exception e) {}

whereClause.append ( "ORDER\_NUMBER =:");

whereClause.append (++ bindCount);

parameters.addElement (ORDERNUM);

clauseCount ++;

}

if ((creado! = null) && (! ( "" es igual a (created.trim ()))) &&

(! ( "Cualquiera" .equals (creada))))

{

si (clauseCount> 0)

{

whereClause.append ( "Y");

}

whereClause.append ( "CREATION\_DATE> =:");

whereClause.append (++ bindCount);

whereClause.append ( "-");

whereClause.append (++ bindCount);

parameters.addElement (getClientSysdate ());

parameters.addElement (getDaysToSubtract (creado));

clauseCount ++;

}

if ((showMyOrders! = null) && (! ( "" es igual a (showMyOrders.trim ()))))

{

// Por lo general, se debe establecer este valor en función del usuario actual.

// Desde el tutorial tiene su propio modelo de datos para los usuarios, vamos a

// establecer esto en la identificación del comprador cabeza de serie.

si (clauseCount> 0)

{

whereClause.append ( "Y");

}

whereClause.append ( "buyer\_id =:");

whereClause.append (++ bindCount);

parameters.addElement (IntegerUtils.getInteger (6)); // 6 es el empleado comprador sin semillas

clauseCount ++;

}

setWhereClause (whereClause.toString ());

si (bindCount> 0)

{

Objeto params [] = new Object [bindCount];

// El copyInto () es 1.1.8 compatible que, como de 07/2004, se requiere por ARU

parameters.copyInto (params);

setWhereClauseParams (params);

}

if ((executeQuery! = null) && (executeQuery.booleanValue ()))

{

ejecutar la solicitud();

}

}

// fin initQuery ()

protegida java.sql.Date getClientSysdate ()

{

OADBTransaction txn =

. (OADBTransaction) getApplicationModule () GetTransaction ();

OANLSServices nls = txn.getOANLSServices ();

oracle.jbo.domain.Date ServerDate = txn.getCurrentDBDate ();

java.util.Date javaClientDate = nls.getUserDate (ServerDate);

largo longDate = javaClientDate.getTime ();

volver nuevo java.sql.Date (longDate);

} // fin getclientSysdate ()

protegida Entero getDaysToSubtract (String creado)

{

int día = 0;

si (.equals "hoy" (creada))

{

días = 0;

}

else if (.equals "THIS\_WEEK" (creada))

{

día = 6;

}

else if (.equals "Last\_30\_days" (creada))

{

día = 29;

}

else if (.equals "LAST\_60\_DAYS" (creada))

{

día = 59;

}

volver IntegerUtils.getInteger (días);

} // fin getDaysToSubtract ()

Paso 6: Añadir una initQuery () método para el módulo de aplicación que los delegados a la vista del objeto initQuery () método. Dar a este método un nombre explícito si su módulo de aplicación tiene varios métodos de consulta de inicialización. Por ejemplo, initEmployeeSummary () , initEmployeeDetails () , initDeparementSummary () , initDepartmentDetails () y así sucesivamente. El módulo de aplicación initQuery () método que los delegados a la initQuery () método que se muestra en los pasos anteriores se ve así:

public void initSummary (String orderNumber,

Cadena creada,

showMyOrders de cuerda,

Boolean executeQuery)

{

PoSimpleSummaryVOImpl vo = getPoSimpleSummaryVO1 ();

si (vo == null)

{

MessageToken [] tokens = {new MessageToken ( "OBJECT\_NAME", "PoSimpleSummaryVO1")};

arrojar nueva OAException ( "FWK\_TBX\_OBJECT\_NOT\_FOUND" "AK",, fichas);

}

vo.initQuery (OrderNumber, creado, showMyOrders, executeQuery);

} // fin initSummary ()

Paso 7: Añadir controlador processFormRequest () la lógica de manejar de la región de búsqueda Ir pulse el botón.

El siguiente es un ejemplo de cómo llamar la initSummary () método, que se muestra en el paso anterior. Dado que nuestros resultados de la consulta se muestran en una tabla, tenga en cuenta la llamada al queryData () método en la parte inferior del controlador. (Las razones de esto se describen detalladamente en [Ver objetos en detalle -> Directrices de inicialización](https://translate.googleusercontent.com/bus/bus_vo.htm#init) ).

oracle.apps.fnd.framework.OAApplicationModule importación;

oracle.apps.fnd.framework.webui.OAQueryUtils de importación;

...

processFormRequest pública vacío (OAPageContext pageContext, OAWebBean webBean)

{

super.processFormRequest (pageContext, webBean);

// Si el valor del parámetro "Go" no es nulo, el usuario selecciona la submitButton "Go".

si (pageContext.getParameter ( "Go")! = null)

{

// Para completar la aplicación selectiva de criterios de búsqueda que inició anteriormente,

// también debe realizar la siguiente prueba. Tenga en cuenta que el parámetro webBean

// que se pasa a la checkSelectiveSearchCriteria ()

// método deberán ser de la región matriz directa que contiene los artículos para los que tiene

// establecer el Selectivo criterios de búsqueda propiedad a **Verdadero** . Si esta prueba falla, el

// OA Marco muestra el mensaje de excepción descrito anteriormente y la consulta

// no se ejecuta.

OAQueryUtils.checkSelectiveSearchCriteria (pageContext, webBean);

// Obtener criterios de búsqueda del usuario a partir de la solicitud.

Cadena orderNumber = pageContext.getParameter ( "SearchOrder");

Cadena creada pageContext.getParameter = ( "Creado");

showMyOrders String = pageContext.getParameter ( "MyOrders");

OAApplicationModule am = getApplicationModule (webBean);

// NOTA Se requiere lo siguiente para los estándares de inicialización de visualización de objetos

// alrededor de las mesas.

Boolean executeQuery = BooleanUtils.getBoolean (false);

Serializables [] Parámetros = {ORDERNUMBER, creado, showMyOrders, executeQuery};

// Dado que los parámetros que estamos pasando incluyen un tipo no-cadena, debe

// llamar a la versión de InvokeMethod () que le permite especificar el individuo

// tipos de parámetros, como se muestra.

Clase [] = {paramTypes String.class, String.class, String.class, Boolean.class};

am.invokeMethod ( "initSummary", parámetros, paramTypes);

OAAdvancedTableBean tabla =

(OAAdvancedTableBean) webBean.findChildRecursive ( "ResultsTable");

// Al manejar una búsqueda iniciada por el usuario, siempre hay que ejecutar

// la consulta para que pase "falso" a queryData ().

table.queryData (pageContext, false);

}

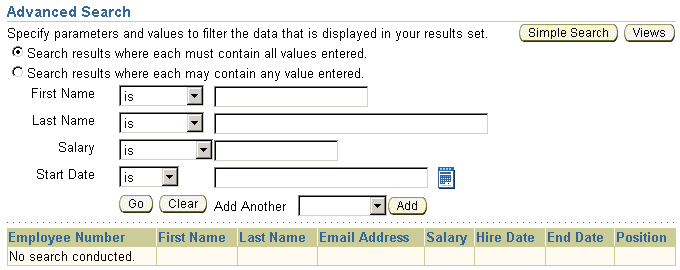
}

**Búsqueda Avanzada**

Como se describió anteriormente, una búsqueda avanzada muestra amplios criterios de búsqueda y capacidades de búsqueda de energía, incluyendo la posibilidad de especificar de forma declarativa "Y" o "O" búsquedas y los operadores de tipo de datos específico en campos individuales. Por ejemplo, un valor de "salario" debe ser superior a 100.000.

**Nota:** OA Marco implementa esta región de búsqueda utilizando el oracle.apps.fnd.framework.webui.beans.layout.OAAdvancedSearchBean . La interfaz de usuario resultante aparece como se muestra en la Figura 12.

Figura 12: Ejemplo de un panel de búsqueda avanzada de la Biblioteca de muestra OA Marco ToolBox



* Para llevar a cabo búsquedas en artículos que se asignan directamente a las columnas de las tablas asociadas, puede aprovechar el grano de consulta para configurar rápidamente una búsqueda completamente automatizado. Ver la [Implementación declarativa: Resultados de Búsqueda en base](https://translate.googleusercontent.com/translate_f#adv_rbc) a las instrucciones.
* Para llevar a cabo una búsqueda avanzada de serie con la capacidad de manipular la propia búsqueda, tales como búsqueda intermedia, utilizar el grano de la consulta con su manejo consulta personalizada. Ver la [Implementación declarativa: Criterios de personalización automática](https://translate.googleusercontent.com/translate_f#simple_auto) .
* Para implementar un diseño de búsqueda avanzada que se diferencia de la OA Marco OAAdvancedSearchBean presentación, se *debe* aplicar de forma manual.

**Nota para los desarrolladores de E-Business Suite de Oracle internos:** Si es necesario implementar una búsqueda de Oracle Intermedia, en contacto con el equipo de Oracle E-Business Suite Ajuste del rendimiento.

**Buscar Resultados: En base declarativas de implementación**

Para crear un panel de búsqueda avanzada en el modo de búsqueda basada en los resultados:

Paso 1: Crear el objeto vista que va a utilizar para la búsqueda. En este modo, no hay necesidad de crear un initQuery () método en el objeto a fin de establecer la cláusula WHERE, se unen los criterios de búsqueda y ejecutar la consulta. OA marco se encarga de esto en su nombre.

**Consejo:** Si la definición de la vista del objeto requiere valores vinculados, o se *debe* agregar manualmente una relación cláusula WHERE cada vez que se ejecuta la consulta, se puede proceder de una de dos maneras. En cualquiera de los casos, la OA Marco añade criterios de búsqueda del usuario a la cláusula WHERE inmediatamente antes de ejecutar la consulta, por lo que las características de cualquier cláusula WHERE que defina se conservan.

1. Anular el executeQuery () método en su \* VOImpl para modificar su cláusula WHERE y luego llamar super.executeQuery () . Esto es recomendable si se ejercita la misma lógica cada vez que el objeto de vista se consulta.
2. Manejar la generada Ir prensa de botón y llamar a una initQuery () método en el objeto de vista donde se puede modificar la cláusula WHERE y / o conjunto unen valores.

Paso 2: Seleccione su región pageLayout en el panel Estructura JDeveloper, haga clic derecho y seleccione Nuevo> Región . Ajuste el estilo de **consulta** , especifique un ID que cumpla con las [Normas Marco de la OA de archivos](https://translate.googleusercontent.com/cs/cs_filedir.htm) , (todas las regiones y objetos que se crean deben cumplir con estas normas) y establezca las siguientes propiedades:

* Modo de construcción - se establece en **resultsBasedSearch** .
* Incluir Panel Avanzado - establece en **Verdadero** .
* Panel Inicial - si se incluye la búsqueda simple y / o la opción Vistas, especifique el panel de la pantalla inicial.
* Instrucciones del Panel avanzadas - especifican el texto de instrucciones para la región de búsqueda avanzada.
* Avanzada la etiqueta del botón - si se incluye la búsqueda simple y / o la opción Vistas, especifique la etiqueta del botón que hace que la OA Marco para alternar la búsqueda simple. La etiqueta predeterminada es "Búsqueda avanzada".

Paso 3: Seleccione su región de consulta en el panel de estructura, haga clic derecho y seleccione Nuevo> Región mediante el asistente para crear rápidamente su mesa o HGrid regiones de resultados de búsqueda. Dar a la región un compatible con los estándares de identificación y especificar conjuntos de atributos apropiados para los elementos asociados. Ver las [Tablas](https://translate.googleusercontent.com/feat_tables.htm) y [HGrid](https://translate.googleusercontent.com/feat_hgrid.htm)documentación para obtener información adicional sobre cómo crear estos componentes. (Esto es particularmente importante para el HGrid ya que hay otras propiedades que debe establecer en ese componente para permitir adecuadamente la búsqueda).

**Nota:** En el modo de búsqueda basada en los resultados, la OA Marco genera automáticamente una "avanzada" de cabecera en tiempo de ejecución para contener la región de consulta. No hay necesidad de añadir esto a la jerarquía de componentes de forma manual.

Paso 4: Seleccione la región / HGrid tabla que creó en el paso 3 y se ajusta la búsqueda Más propiedad a **Verdadero** para los elementos que se utilizan como criterios de búsqueda. Estos artículos se incluyen automáticamente en criterios de búsqueda avanzada de poplists la región de búsqueda, y OA Marco ejecuta automáticamente la consulta correspondiente cuando el usuario selecciona el Go botón.

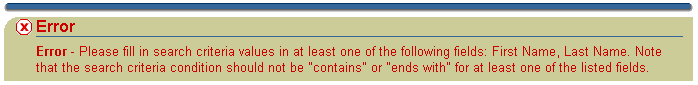
Paso 5: Establecer el Criterios de búsqueda selectiva propiedad a **Verdadero** para cualquier criterio de búsqueda elementos que aseguran una consulta performant. Ver la [OA marco Ver estándares de codificación](https://translate.googleusercontent.com/cs/cs_viewgeneral.htm) para obtener información adicional.

**Nota:** OA Marco trata a estos valores como ser *individual* requerida: si el usuario entra en uno cualquiera de los elementos requeridos, la consulta puede proceder. Actualmente no se puede exigir que el usuario introduzca los criterios de búsqueda en más de uno de estos elementos necesarios.

Como se muestra en la Figura 13 a continuación, si el usuario falla al especificar al menos uno de los valores de búsqueda necesarios, un mensaje de error con la lista de elementos candidatos por lo que el usuario puede introducir un valor y continúe con la búsqueda.

**Nota:** El mensaje de error para la búsqueda avanzada difiere de la búsqueda simple.

Figura 13: Búsqueda avanzada selectiva mensaje de error criterios de búsqueda (tenga en cuenta que el mensaje de error real puede diferir ligeramente).



Paso 6: Guarde su trabajo.

**Nota:** Si crea una búsqueda basada en los resultados, no añada regiones de búsqueda personalizados como se requiere para los "criterios de personalización automático" que se describen a continuación.

**La aplicación declarativa: Criterios para requisitos particulares auto**

Para crear una búsqueda avanzada del panel en el modo automático criterios de personalización:

Paso 1: Crear el objeto vista que va a utilizar para la búsqueda como se describe en los [resultados de búsqueda basados](https://translate.googleusercontent.com/translate_f#adv_rbc) sección anterior.

Paso 2: Seleccione su región pageLayout en el panel Estructura JDeveloper, haga clic derecho y seleccione Nuevo> Región . Ajuste de la región Estilo de **consulta** , especifique un ID que cumpla con las [Normas Marco de la OA de archivos,](https://translate.googleusercontent.com/cs/cs_filedir.htm) (todas las regiones y objetos que se crean deben cumplir con estas normas) y establezca las siguientes propiedades:

* Modo de construcción - se establece en **autoCustomizationCriteria** .
* Incluir Panel Avanzado - establece en **Verdadero** .
* Panel Inicial - si se incluye la búsqueda simple y / o la opción Vistas, especifique el panel de la pantalla inicial.
* Instrucciones del Panel avanzadas - especifican el texto de instrucciones para la región de búsqueda avanzada.
* Avanzada la etiqueta del botón - si se incluye la búsqueda simple y / o la opción Vistas, especifique la etiqueta del botón que hace que el Marco de la OA para alternar la búsqueda simple. La etiqueta predeterminada es "Búsqueda avanzada".

Paso 3: Seleccione su región de consulta en el panel Estructura, haga clic derecho y seleccione Nuevo> Región mediante el asistente para crear rápidamente su mesa o HGrid regiones de resultados de búsqueda. Asignar a la región una compatible con los estándares de identificación y especificar atributo apropiado establece para los elementos asociados. Ver las [Tablas](https://translate.googleusercontent.com/feat_tables.htm) y [HGrid](https://translate.googleusercontent.com/feat_hgrid.htm)documentación para obtener información adicional sobre cómo crear estos componentes. (Esto es particularmente importante para el HGrid ya que hay otras propiedades que debe establecer en ese componente para permitir adecuadamente la búsqueda).

Paso 4: Crear su región de búsqueda personalizada avanzado.

* Paso 4.1 Seleccione su región consulta en el panel Estructura, haga clic derecho y seleccione Nuevo> advancedSearchPanel . JDeveloper crea automáticamente una región de cabecera con el texto propiedad establecida en Búsqueda avanzada. Dar a esta región una compatible con los estándares de identificación .
* Paso 4.2 Seleccionar la región de cabecera en el panel Estructura, haga clic derecho y seleccione Nuevo> AdvancedSearch .Dar la región AdvancedSearch resultando una queja normas- ID . JDeveloper crea un nodo y un nodo Criterios criteriaRow defecto con dos elementos por debajo de ella.
* Paso 4.3 Configurar el criteriaRow por defecto para la primera búsqueda criterios de valor que desea visualizar:
  + Dar al nodo criteriaRow una significativa ID como se muestra en la Figura 14 a continuación.
  + En el primer elemento que JDeveloper crea para usted, especifique los criterios de búsqueda Prompt .
  + Configurar la segunda **messageTextInput** artículo para el valor de criterios. Especificar una compatible con los estándares de identificación y aplicar un adecuado conjunto de atributos . Asegúrese de seleccionar el correcto tipo de datos de valor, ya que impulsa el poplist "Condiciones" que OA Marco crea para usted. Además, establecer el Criterios de búsqueda selectiva propiedad a **Verdadero** para cualquier criterio de búsqueda elementos que aseguran una consulta performant. Ver la [OA marco Ver estándares de codificación](https://translate.googleusercontent.com/cs/cs_viewgeneral.htm) para obtener información adicional.

**Nota:** OA Marco trata a estos valores como ser *individual* requerida: si el usuario entra en uno cualquiera de los elementos requeridos, la consulta puede proceder. Actualmente no se puede exigir que el usuario introduzca los criterios de búsqueda en más de uno de estos elementos necesarios.

Por último, establecer el texto Addtional valor en el mismo valor que ha establecido para los criterios de búsqueda Prompt . Esto es requerido por las [directrices de accesibilidad](https://translate.googleusercontent.com/cs/cs_viewgeneral.htm#508) ).

Como se muestra en [la Figura 13](https://translate.googleusercontent.com/translate_f#fig13) anterior, si el usuario no especificar al menos uno de los valores de búsqueda necesarios, un mensaje de error con la lista de elementos candidatos por lo que el usuario puede introducir un valor y continúe con la búsqueda.

* Paso 4.3 Si tiene criterios de búsqueda adicionales, seleccione el nodo de Criteria en el panel Estructura, haga clic derecho y seleccione Nuevo> criteriaRow para cada elemento criterios de búsqueda que desee agregar. Repita el paso 4.3 para configurar cada fila de criterios.

Paso 5: Cree las correlaciones de consulta entre la región de búsqueda personalizada avanzado y las columnas de tabla, por lo que la OA marco puede ejecutar automáticamente la búsqueda cuando el usuario selecciona el Go botón.

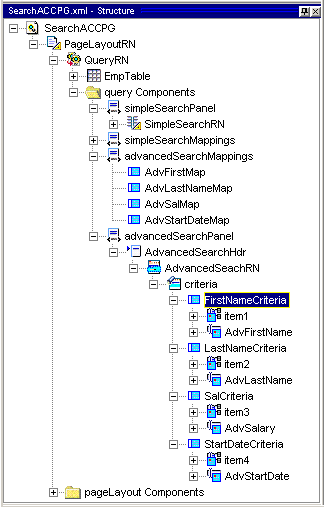
* Paso 5.1 Seleccione su región consulta en el panel Estructura, haga clic derecho y seleccione Nuevo> AdvancedSearch Asignaciones.
* Paso 5.2 Configurar la asignación predeterminada que JDeveloper crea para usted. Este mapeo identifica el elemento en la región de búsqueda cuyo valor se utiliza para consultar el elemento correspondiente (columna de la tabla) en la región de resultados.

**Importante:** Usted está asignando valores de los criterios de búsqueda en este paso, no mostrar los valores. Por ejemplo, la región de búsqueda incluye una lista de valores que devuelve una clave única para un campo de formulario oculto y un valor de visualización para el campo de entrada LOV. Debe asignar tanto en el campo de formulario oculto y el valor de visualización a sus columnas de las tablas correspondientes.

* Paso 5.3 (opcional) Si tiene más de una asignación, seleccione el nodo advancedSearchMapping en el panel Estructura, haga clic derecho y seleccione Nuevo> queryCriteriaMap . Para configurar el nodo como lo hizo en el paso 5.2. Repita hasta que todos sus artículos criterios de búsqueda se asignan a los resultados de la tabla artículos / HGrid.

La figura 14 ilustra una estructura de la página de búsqueda avanzada creada en modo de criterios de personalización automática.

Figura 14: ejemplo de una personalización automática Criterios de búsqueda avanzado de la estructura en la Biblioteca de la muestra OA Marco caja de herramientas



Paso 6: Guarde su trabajo.

**Búsqueda manual**

Para construir una región de búsqueda avanzada que no puede ser implementado con el grano de consulta, (siempre se debe utilizar el grano de consulta si es posible), siga los pasos descritos en el [Manual de Búsqueda simple](https://translate.googleusercontent.com/translate_f#simple_manual) sección anterior.

**control de tiempo de ejecución**

En esta sección se discute:

* [Las validaciones personalizadas](https://translate.googleusercontent.com/translate_f#cust_validations)
* [Manipulación processFormRequest](https://translate.googleusercontent.com/translate_f#handlepfr)

**Las validaciones personalizadas**

Dos nuevas constantes CUST\_VIEW\_UPDATE\_APPLY y CUST\_VIEW\_APPLY\_VIEW\_RESULTS se han añadido en OAWebBeanConstants para la celebración de la ID de la Aplicar y Aplicar y ver los resultados botones. Para ver si los botones se han hecho clic en la página Crear / Actualizar vista, la verificación de estos atributos en el controlador conectado a AdvancedSearchRegion usando pageContext.getParameter (CUST\_VIEW\_UPDATE\_APPLY) o pageContext.getParameter (CUST\_VIEW\_APPLY\_VIEW\_RESULTS) .

Validaciones personalizados *deben* ser realizadas en el processFormData método de controlador conectado a la AdvancedSearchRegion. Esto se debe a que en el momento en el processFormRequest método se llama, se guardará la vista y cualquier excepción lanzada desde el processFormReques método t no impedirá la vista desde ser salvo.

El siguiente código de ejemplo se puede utilizar para lanzar una excepción si no se introduce un valor en el primer elemento de búsqueda en el AdvancedSearchRegion antes de pulsar el Aplicar o Aplicar y ver los resultados botón:

processFormData public void (OAPageContext pageContext, OAWebBean webBean)

{

super.processFormData (pageContext, webBean);

si (pageContext.getParameter (CUST\_VIEW\_UPDATE\_APPLY)! = null ||

pageContext.getParameter (CUST\_VIEW\_APPLY\_VIEW\_RESULTS)! = null)

{

OAMessageTextInputBean mtiBean = (OAMessageTextInputBean)

webBean.findChildRecursive ( "VALUE\_0");

si (mtiBean.getValue (pageContext) == null)

{

arrojar nueva OAException ( "El valor debe ser introducido para Elemento1",

OAException.ERROR);

}

}

}

**Manipulación processFormRequest**

En general, es necesario manejar las formulario envía de ningún botón sembradas en sus artículos región. El único botón que no es necesario para manejar es el Agregar botón, que se encarga de la avanzada de Web Bean búsqueda.

Dado que los elementos duplicados se pueden añadir a la lista de elementos que se muestran y la lista cambia dinámicamente, utilizar la siguiente convención de nomenclatura para identificar los granos web en la región de búsqueda avanzada:

* El nombre del elemento de la / O grupo de radio y está OAWebBeanConstants.ADVANCED\_SEARCH\_RADIO\_GROUP . Devuelve un valor de Y o O en base a lo que se elija.
* Todas las condiciones (condiciones poplist) mostrados en la página tienen UINodeNames establecidos como Condition\_0 a Condition\_n .
* Todos los frijoles web reales, que son artículos en su región, tienen sus UINodeNames establecen en VALUE\_0 a Valor\_N .
* Utilice los getOriginalUINodeName métodos en el grano de búsqueda avanzada para recuperar el nombre del artículo original de cada grano web.
* Usted puede utilizar el método getDisplayedCriteriaCount en el grano de búsqueda avanzada para obtener el número de condiciones / criterios que aparecen en la página en cualquier punto en el tiempo.
* Las condiciones poplists derivan sus valores a partir de las operaciones de búsqueda FND ICX\_CONDITIONS , donde LookupCode es el atributo de valor y Significado es el atributo de visualización. Usted puede utilizar el método getConditionen oracle.apps.fnd.framework.OAFwkUtils para obtener el equivalente SQL de la afección que se transmite. Por ejemplo, si sus condiciones poplist devuelve un atributo de valor de CCONTAINS (y un atributo de visualización de Contiene ), a continuación, OAFwkUtils.getCondition ( "CCONTAINS") devuelve la cadena "así como" que se puede utilizar directamente en su cláusula WHERE.

Para manejar somete botones específicos de la región, el processFormRequest método en su búsqueda avanzado controlador de frijol web debe ser similar al código de ejemplo a continuación:

processFormRequest pública vacío (OAPageContext pageContext, OAWebBean webBean)

{

// obtener los criterios que se muestran cuentan en primer lugar.

int displayedCriteriaCount = webBean.getDisplayedCriteriaCount ();

// La siguiente le da la Y O valor de grupo / radio

Cadena radioGroupValue = pageContext.getParameter

(OAWebBeanConstants.ADVANCED\_SEARCH\_RADIO\_GROUP);

for (int i = 0; i <displayedCriteriaCount; i ++)

{

Cadena newItemName = OAWebBeanConstants.VALUE +

Integer.toString (i); // devuelve una cadena como "VALUE\_0"

Cadena ConditionName = OAWebBeanConstants.CONDITION +

Integer.toString (i); // devuelve una cadena como "Condition\_0"

// comprobar si se ha introducido algún valor y la condición primera

// por el usuario por este concepto.

if ((pageContext.getParameter (newItemName)! = null) &&

(PageContext.getParameter (ConditionName)! = Null))

{

// esto le da el nombre del artículo original, que eligió en AK.

Cadena originalChildName = webBean.getOriginalUINodeName (newChildName);

// Con el fin de obtener la "condición de SQL" que se utiliza puede utilizar

// el siguiente método

// esto devuelve una cadena como: "=", o "me gusta", etc.

Cadena condición = OAFwkUtils.getCondition (ConditionName);

// Esto le da el valor elegido

valor String = pageContext.getParameter (newItemName);

// Con el fin de obtener el valor de "SQL", puede utilizar el siguiente método

// Por lo tanto, si usted tenía una condición de "CCONTAINS" (similares) y un valor de

// "XYZ" este método volvería "XYZ%%";

// Esto puede usarse para unir directamente a su whereClause.

value = OAFwkUtils.getValue (ConditionName, valor);

// Ahora que tiene la condición, el valor y sabes

// el niño, manejar su lógica específica del producto aquí.

}

}

}

En cuanto a páginas con múltiples regiones de consulta, (que incluye a más de una región consulta definida bajo el mismo diseño pestaña secundaria o bajo diferentes subpestañas en la misma página). La segunda y siguientes granos de consulta tienen la llave de su región de queryBean antepone al Valor\_N . Si la llamada nonFirstQueryBean.findChildRecursive ( "VALUE\_ ") Se hace, volverá **nula** se requieren equipos de productos para manejar la segunda región y la posterior consulta múltiple. Valor\_N siendo nula utilizando la siguiente lógica;

OAQueryBean nonFirstQueryBean = (OAQueryBean) webBean.findChildRecursive (<nonFirstQueryRN>);

OAAdvancedSearchBean advBean = (OAAdvancedSearchBean) nonFirstQueryBean.findChildRecursive ( "advancedSearchRN");

nonFirstQueryBean.findChildRecursive (advBean.getRegionKey (advBean, ...) + "Valor \_ '<n>'");

**(Usuario Personalizable) Vistas**

Para apoyar las búsquedas de usuario personalizable, que se apareció en un panel Vistas, debe aprovechar la región de consulta como se describe a continuación.

**Nota:** La primera vez que el usuario accede a una página de vistas habilitada (sin búsquedas para ejecutar salvados) OA Marco hace lo siguiente:

* Si sólo hay un panel Vistas, esta muestra con un poplist Vistas vacía. Se espera que el usuario seleccione el botón Personalizar para crear una o más búsquedas guardadas.
* Si hay un panel de vistas y una simple búsqueda, la búsqueda simple hace. Se espera que el usuario ejecute una consulta y seleccionar la Memoria de búsqueda botón.
* Si hay un panel de vistas y una búsqueda avanzada, la búsqueda avanzada hace. Se espera que el usuario ejecute una consulta y seleccionar la Memoria de búsqueda botón.

**Implementación**

Paso 1: Crear el objeto vista que va a utilizar para la búsqueda. En este modo, no hay necesidad de crear un initQuery () método en el objeto a fin de establecer la cláusula WHERE, se unen los criterios de búsqueda y ejecutar la consulta. OA marco se encarga de esto en su nombre.

**Consejo:** Si la definición de la vista del objeto requiere valores vinculados, o se *debe* agregar manualmente una relación cláusula WHERE cada vez que se ejecuta la consulta, se puede proceder de una de dos maneras. En cualquiera de los casos, la OA Marco añade criterios de búsqueda del usuario a la cláusula WHERE inmediatamente antes de ejecutar la consulta, por lo que las características de cualquier cláusula WHERE que defina se conservan.

1. Anular el executeQuery () método en su \* VOImpl para modificar su cláusula WHERE y luego llamar super.executeQuery () . Esto es recomendable si se ejercita la misma lógica cada vez que el objeto de vista se consulta.
2. Manejar la generada Ir prensa de botón y llamar a una initQuery () método en el objeto de vista donde se puede modificar la cláusula WHERE y / o conjunto unen valores.

Paso 2: Seleccione su región pageLayout en el panel Estructura JDeveloper, haga clic derecho y seleccione Nuevo> Región . Ajuste el estilo de **consulta** , especifique un ID que cumpla con las [Normas Marco de la OA de archivos](https://translate.googleusercontent.com/cs/cs_filedir.htm) , (todas las regiones y objetos que se crean deben cumplir con estas normas) y establezca las siguientes propiedades:

* Modo de construcción - Esto no tiene ningún efecto en el panel Vistas y se aplica a la simple y sólo los paneles avanzadas de búsqueda.

**Nota:** Cuando la construcción Modo valor es **Ninguno** , la sección Criterios se oculta en la página de creación de vistas.

* Incluir panel Vistas - se establece en **Verdadero** .
* Panel Inicial - si se incluye la búsqueda simple o opciones de búsqueda avanzada, especifique el panel de la pantalla inicial.
* Vistas etiqueta del botón -si se incluye la búsqueda avanzada o búsqueda simple, especifique la etiqueta del botón que hace que la OA Marco para alternar a la región vistas
* Instrucciones del Panel vistas - especifican el texto de instrucciones para la región vistas
* Vistas Grupo Título - especifique un título alternativo si es necesario. El título predeterminado es "Vistas".
* Guardar Botón de búsqueda de texto - especificar el texto que se mostrará en la Memoria de búsqueda botón. La etiqueta predeterminada es "Guardar búsqueda".

Paso 3: Seleccione su región de consulta en el panel de estructura, haga clic derecho y seleccione Nuevo> Región mediante el asistente para crear rápidamente su mesa o HGrid regiones de resultados de búsqueda. Asignar a la región una compatible con los estándares de identificación y especificar atributo apropiado establece para los elementos asociados. Ver las [Tablas](https://translate.googleusercontent.com/feat_tables.htm) y [HGrid](https://translate.googleusercontent.com/feat_hgrid.htm)documentación para obtener información adicional sobre cómo crear estos componentes. (Esto es particularmente importante para el HGrid ya que hay otras propiedades que debe establecer en ese componente para permitir adecuadamente la búsqueda).

Paso 4: Seleccione la región / HGrid tabla que creó en el paso 3 y configura su personalización de usuario propiedad a **Verdadero** .

Paso 5: Seleccione la región / HGrid tabla que creó anteriormente y establecer cada elemento de personalización de usuariopropiedad a **Verdadero** si el artículo que debe ser personalizable. Por ejemplo, el usuario puede configurar sus propiedades de pantalla en la tabla subyacente / HGrid y especificar criterios de búsqueda para el valor de la columna asociada. Fijar su búsqueda Más propiedad a **Verdadero** si desea que el usuario sea capaz de filtrar la búsqueda sobre el valor del artículo.

**Nota:** Si el artículo es administrador personalizable, el usuario puede cambiar la búsqueda Más propiedad a **Verdadero** , incluso si se establece a **Falso** .

Paso 6: Guarde su trabajo.

Paso 7 (opcional): Si tiene que hacer un procesamiento especial punto de vista, tales como la unión criterios de consulta adicionales, manejar esto de processFormRequest () como se muestra en el [control de tiempo de ejecución](https://translate.googleusercontent.com/translate_f#runtime) siguiente sección. Sin embargo, si el usuario crea una nueva visión y opta por utilizar inmediatamente seleccionando la Aplicar y ver los resultados botón en el módulo de personalización, también hay que añadir el siguiente processRequest () lógica que comprueba el CUST\_ANVR\_VIEWID valor de atributo:

processRequest pública vacío (OAPageContext pageContext, OAWebBean webBean)

{

super.processRequest (pageContext, webBean);

...

OAQueryBean queryBean =

(OAQueryBean) pageContext.findIndexedChildRecursive ( "Query");

Cadena viewId = queryBean.getAttributeValue (OAWebBeanConstants.CUST\_ANVR\_VIEWID);

// Si viewId no es nulo, entonces el usuario selecciona los "aplicar y ver los resultados"

// botón y la nueva vista que acaba de crear es ahora la vista actual. Tenga en cuenta que

// esto es el mismo valor viewId que se devuelve cuando llama

// queryBean.getCurrentCustomization () en processFormRequest (). Si el

// valor viewId es nulo, esto es el flujo normal de la página.

si (viewId! = null)

{

...

}

}

Paso 8: Asegúrese de que la [opción de perfil personal Desactivar autoservicio](https://translate.googleusercontent.com/options.htm#personalization) está establecido en **No** a nivel de sitio o aplicación de acuerdo con su página. Si se establece en **Sí** , la opción Vistas no será accesible para el usuario, incluso si está correctamente activada. (Si la región de búsqueda no incluye simples o avanzadas opciones de búsqueda, la opción Vistas rendirá pero el poplist Vistas estará vacía).

**Nota:** Cuando un usuario selecciona una [vista personalizada](https://translate.googleusercontent.com/persguide/T401443T401451.htm#cust_persuser_searchpg) para mostrar, OA Personalización no retiene la vista de selección actual para la página de consulta cuando se desplaza a otra página. Por ejemplo, suponga que muestra una vista personalizada de la página de consulta mediante la selección de una vista (como Vista X) desde el panel Vistas. A continuación, se selecciona el Personalizar botón para mostrar la página Vistas Personalizar, pero elige Cancelar sin hacer ningún cambio. Cuando regresa a la página de consulta, la definición de base se muestra y no ven a X. Si desea conservar la vista, se debe modificar el controlador de la página para pasar la vista seleccionada por el usuario como un parámetro y establecer este punto de vista como el personalización actual cada vez que el usuario vuelve a la página de consulta.

**No Personalizable Vistas**

Para mostrar un panel Vistas en su página de búsqueda que sólo muestra vistas cabezas de serie, (ya sea de Oracle-nivel o de nivel de administrador), y no permite a los usuarios crear sus propios puntos de vista personalizable, siga las instrucciones de implementación descritos anteriormente para crear un [(Usuario Personalizable ) Vistas](https://translate.googleusercontent.com/translate_f#views) del panel. A continuación, mediante programación ocultar la Memoria de búsqueda botón en el panel de búsqueda y el Personalizar botón en el panel Vistas usando los siguientes OAQueryBean API en el controlador de processRequest método:

* hideSaveSearchButton (ocultar boolean)
* hidePersonalizeButton (ocultar boolean)

Por defecto, el valor para ocultar está establecido en **Falso** y la Memoria de búsqueda y personalizan se muestran los botones.

**Nota:** En la versión 12.1 hay una nueva característica que permite la misma funcionalidad para ocultar los botones a través de personalizaciones administrativas en lugar de utilizar las API a través del método processRequest.

**Vistas compartibles**

OA Marco le permite ampliar una región personalizable consulta de usuario, de modo que es compartido por varias páginas. Anteriormente, sólo los administradores de personalización se les permitió crear una vista personalizada de una región compartida que también fue visible a todas las páginas que comparten esa región. Una vista personalizada de una región compartida creada por un usuario final, sin embargo, siempre se guarda como una vista por instancia y sólo era visible en la página desde la que se creó / actualizó la vista. En la versión 12, ahora se puede establecer una propiedad en la región de consulta que permite una vista creada por un usuario final que se almacenan directamente en la región compartida, en lugar de como una [vista por instancia](https://translate.googleusercontent.com/persguide/T401443T401450.htm#cust_persadmin_perinstance) . Esto permite al usuario final para ver la vista personalizada en todas las páginas en las que se comparte la región consulta.

**Nota:***Sólo se puede habilitar la característica de las vistas que se pueden compartir para las nuevas regiones de consulta que se crean en la versión 12.*

Para activar esta función, llame a la oracle.apps.fnd.framework.webui.beans.layout.OAQueryBean storeViewsOnSharedRegion API de la siguiente manera: (Esta API se debe establecer en cada región de consulta tal como se define en cada página que se extiende él).

queryBean.storeViewsOnSharedRegion (true);

Esta API almacena ninguna vista creada por el usuario final en la región de consulta compartida. Como resultado, cualquier cambio que un usuario realiza a una vista (crear, actualizar o eliminar) de esa región de consulta se reflejará en todas las páginas que comparten esa región consulta.

**Restricciones**

* Esta es una nueva característica introducida en la versión 12 y sólo debe ser habilitado para que las nuevas páginas que no tienen ningún existente por ejemplo vistas creadas para la misma. Esto generalmente significa que la región de consulta recién creada nunca ha enviado a los clientes previamente. OA marco no ofrece soporte de compatibilidad con versiones anteriores de esta característica para las regiones de consulta previamente enviado en las versiones anteriores, que pueden haber existentes por ejemplo vistas creadas para la misma. Por tanto, es obligatorio que no implementa esta característica en las regiones de la consulta pre-existentes.
* Cuando esta función está habilitada para una región compartida consulta, una vista creada por un usuario final se almacenará en la región compartida. OA Framework no admite administradores de personalización Creación de una vista por cada instancia de esa región compartida. Como resultado, un administrador de personalización debe abstenerse de crear una vista por ejemplo en una región de consulta compartida que tiene la característica de vistas compartible habilitado. En otras palabras, cuando un administrador crea una personalización, él o ella debe establecer el alcance del contexto de personalización a la región compartida y no a la propia página. **Es responsabilidad del equipo de producto que implementa esta función para documentar esta restricción para sus usuarios.** La administración Personalización de interfaz de usuario se actualizará en una versión posterior para hacer cumplir esta restricción.
* OA Marco emite un error si usted o un administrador de personalización lo ignora cualquiera de las dos restricciones anteriores. Por ejemplo:
  + Si todavía intenta poner en práctica las vistas que se pueden compartir disponen de una página que ha enviado previamente a los clientes (y ha existente por vistas instancia creada por ella en las instalaciones del cliente), y alguien trata de duplicar una de las vistas existentes por ejemplo en la página donde se lleva a cabo la función de vistas compartible, OA marco genera un error, advirtiendo que detecte esta discrepancia.
  + Si habilita las vistas que se pueden compartir cuentan para una región compartida y un administrador de personalización hace caso omiso de su restricción documentada y procede a crear una vista por instancia de la región de consulta compartida (donde el alcance contexto de personalización se ajusta a la página), OA marco genera un error cuando un usuario final intenta duplicar el por instancia opinión de que el administrador crea.

**Notas de uso**

**Filas que se mostrarán** - Si se define una región de consulta que incluye un panel Vistas, los usuarios finales de esa región consulta podrán guardar su busca y personalizar sus resultados de búsqueda como vistas. En [la definición de sus vistas personalizadas](https://translate.googleusercontent.com/persguide/T401443T401451.htm#cust_persuser_ucudv2) , pueden especificar el número de filas que se mostrarán en la tabla de resultados de búsqueda. Si desea anular el número de filas que se muestran, según la definición de la vista, y en su lugar, utilice el valor establecido en los metadatos de la base de la actual [mesa](https://translate.googleusercontent.com/feat_tables.htm#nor) o [tabla avanzada](https://translate.googleusercontent.com/feat_advtables.htm#nor) , oa un valor establecido mediante programación-, puede utilizar el atributo BYPASS\_NOR\_ON\_VIEW en el oracle.apps.fnd.framework.webui.OAWebBeanConstants interfaz. Ajuste el BYPASS\_NOR\_ON\_VIEWatributo mediante programación en la tabla secundaria de la consulta web de frijol o mesa avanzada. Mediante el uso de este atributo, la tabla de resultados de búsqueda pasará por alto el número de filas que se muestran según lo establecido por la vista, y hacer que el número de filas que se mostrarán como se especifica en los metadatos de la tabla o mesa de avanzada.

**Comportamiento OAQueryBean en detalle**

OA Marco genera la cláusula WHERE de forma automática para sus búsquedas, tanto en los resultados de búsqueda en base y los criterios de modos de personalización de automóviles. También genera la cláusula WHERE de forma automática para los criterios introducidos a través del Marco Personalización del panel Vistas en los dos modos anteriores.

Esta sección analiza las reglas utilizadas por OA Framework para generar su cláusula WHERE y los diversos métodos que se pueden utilizar para anularlo.

**Criterios de objetos Información general**

OA Marco almacena la información de metadatos necesarios para generar la cláusula WHERE en un criterio objetivo. El objeto criterios es una matriz de java.util.Dictionary . Cada elemento de la matriz identifica un *criterio* o una porción de su cláusula WHERE. Cada criterio se compone de los siguientes pares de nombre y valor. Todas las constantes están disponibles en el oracle.apps.fnd.framework.OAViewObject clase.

|  |  |
| --- | --- |
| **Nombre constante** | **Valor** |
| CRITERIA\_ITEM\_NAME | El nombre del artículo o la identificación del tema que el criterio se define para. |
| CRITERIA\_VIEW\_ATTRIBUTE\_NAME | La vista nombre de atributo del tema que el criterio se define para. |
| CRITERIA\_CONDITION | La condición real que se utiliza para la unión en la consulta SQL. Ejemplos: "=", "como" etc. |
| CRITERIA\_VALUE | El valor transformado del artículo como computada por Marco OA basado en la condición elegida por el usuario. |
| CRITERIA\_JOIN\_CONDITION | La condición de unión utilizado para unir los diversos criterio dentro de una cláusula WHERE. Su valor se establece en "Y" si "Que todas" se usa y "O" si "coincide con ninguno" se utiliza en el panel de búsqueda avanzada.  Para el panel de búsqueda simple, su valor se establece en "Y". |

La tabla siguiente describe cómo OA Marco calcula la CRITERIA\_VALUE y la CRITERIA\_CONDITION basado en la condición de elegido y el valor introducido por el usuario en el panel de búsqueda avanzada.

**Nota:** De las diversas opciones de abajo, el panel de búsqueda simple utiliza los *arranques con* opción internamente para cuerdas y *es* la opción para los números y fechas.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Condición**  **(Como elegido por el usuario)** | **Aplicable** **Tipos de datos** | **CRITERIOS \_CONDITION (calculado por el Marco OA)** | **Valor (como introducidos por el usuario)** | **CRITERIA\_VALUE (calculado por el Marco OA)** |
| es | Cadena, número, fecha | = | Jamie Frost | "Jamie Frost" |
| empieza  con | Cuerda | me gusta | Ja | "Ja%" |
| termina con | Cuerda | me gusta | Ja | "Ja%" |
| contiene | Cuerda | me gusta | Ja | Ja%% |
| antes de | Fecha | < | 01.12.03 | 01.12.03 |
| después | Fecha | > | 01.12.03 | 01.12.03 |
| mas grande que | Número | > | 123 | 123 |
| menos que | Número | < | 123 | 123 |

**Por defecto donde la generación de la Cláusula**

OA Marco genera la cláusula WHERE utilizando los criterios objeto descrito anteriormente y lo añade a cualquier otra cláusula WHERE que es probable que haya utilizando una condición AND.

La cláusula WHERE se genera utilizando las reglas siguientes. Las reglas se explican usando un ejemplo de un objeto de criterios que se utiliza en los "proyectos" páginas de búsqueda.

Nota: La búsqueda se puede hacer entre mayúsculas y minúsculas llamando al método setCaseSensitive (verdadero) en el grano de consulta en el controlador.   
queryBean.setCaseSensitive (true); y la prueba utilizando isCaseSensitive ().

El objeto criterios se puede representar en una forma tabular como:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CRITERIA\_ ITEM\_NAME** | **Selectivos criterios de búsqueda** | **CRITERIA\_** **VIEW\_ ATTRIBUTE\_**  **NOMBRE** | **CRITERIA\_ CONDICIÓN** | **CRITERIA\_ VALOR** | **REAL** **nombre de la columna DB** |
| Projecto ID | falso | projectId (tipo de datos: NÚMERO) | = | "101" | PROJECTO ID |
| estado del proyecto | falso | projectStatus (tipo de datos: VARCHAR2) | me gusta | "%Aprobado%" | ESTADO DEL PROYECTO |
| nombre del proyecto | cierto | projectName (tipo de datos VARCHAR2) | me gusta | "Seguridad%" | SEGURIDAD |
| fecha de inicio | falso | startDate (tipo de datos de fecha) | > | "1/1/04" | FECHA DE INICIO |

**Regla 1** : Si su viewattribute es del tipo Numérico, Fecha o Fecha y hora y, a continuación OA marco genera un simple cláusula WHERE utilizando el CRITERIA\_CONDITION basado en la CRITERIA\_VALUE . Así, en el ejemplo anterior, la cláusula WHERE para el "projectId" atributo de vista se genera como:

PROJECT\_ID = 101

y la cláusula WHERE para el "startDate" atributo vista se genera como:

START\_DATE> "1/1/03"

OA Marco se encuentra la columna de la derecha base de datos que corresponde a un atributo de vista, incluso si utiliza los alias.

**Regla 2:** Si el atributo de vista es de tipo String (VARCHAR2), entonces OA Marco genera una cláusula WHERE entre mayúsculas y minúsculas utilizando una de cuatro vías se unen para garantizar que se utiliza el índice.   
es decir, la generación de la cláusula WHERE es independiente de la propiedad criterios de búsqueda que se pone a falso o verdadero

**Anulación de la Generación cláusula WHERE**

Para generar una cláusula WHERE que es diferente de la generada por Marco OA, utilice el getCriteria () y la getNonViewAttrCriteria () métodos en OAQueryBean . Debido a cuestiones de tiempo, utilizar estos métodos en el controlador asociado con sólo la región de consulta.

Los métodos son diferentes de las siguientes maneras:

|  |  |  |
| --- | --- | --- |
|  | **getCriteria ()** | **getNonViewAttrCriteria ()** |
| Generación de cláusula WHERE | Va a obtener un identificador para todo el diccionario de criterios. | Va a conseguir una manija sólo para un subconjunto del diccionario (criterios de los artículos que no tienen atributos de vista) |
| La ejecución de la consulta | Llamar executeQuery () después de construir los criterios. | OA Marco añade cualesquiera otras cláusulas adicionales WHERE y ejecuta la consulta para usted.  No llame executeQuery () en este caso. |

Siga estos pasos para utilizar uno de los métodos anteriores:

Paso 1: Determinar el método a utilizar.

Determine si desea manejar los criterios completos o los criterios para un subconjunto de atributos que en realidad no pertenecen a la tabla, pero es algo que desea buscar. Utilice la getCriteria () método para el primer caso y la getNonViewAttrCriteria () método para el último caso.

Paso 2: Definir el campo de búsqueda en los paneles de búsqueda.

Definir el artículo como un campo de búsqueda en los paneles de búsqueda simple o avanzada. Para un ejemplo, definir los siguientes campos:

* projectIdSimpleSearch en el panel de búsqueda simple.
* projectIdAdvSearch en el panel de búsqueda avanzada.

Paso 3: Uso de la getNonViewAttrCriteria () método.

Modificar el ejemplo anterior y el estado que projectId es un campo que no está presente en el viewObject, no se muestran en la tabla de resultados, pero algo que desea buscar.

**Opción 1***-* Utilice esta opción si desea buscar en un campo específico, pero *no* guardar su búsqueda como una vista de personalización de usuario.

Procesar este campo sólo en el panel de búsqueda simple o avanzada.

Paso 3.1: No defina las asignaciones de búsqueda simple o avanzada para este campo.

Paso 3.2: Manejar el Go pulse el botón en el controlador de su región de consulta processFormRequest () usando:

processFormRequest (OAPageContext pageContext, OAWebBean webBean)

{

// Asegúrese de que usted está en el panel de la derecha

OAQueryBean queryBean = (OAQueryBean) webBean;

queryBean.getCurrentSearchPanel ();

// Manejar el botón "Go" clic en el sencillo panel de búsqueda

if (SEARCH.equals (currentPanel) &&

queryBean.getGoButtonName ()! = null)

{

// recuperar su artículo y procesar aquí.

Valor String = pageContext.getParameter (< *projectIdSimpleSearch* >);

// Construir la cláusula WHERE apropiada para projectIdSimpleSearch

// y la puso en su vo y ejecutar la consulta

// utilizando el método initQuery.

...

}

// Manejar el botón "Go", haga clic en el panel de búsqueda avanzada

if (ADVANCED\_SEARCH.equals (currentPanel) &&

queryBean.getGoButtonName ()! = null)

{

// recuperar su artículo y procesar aquí.

Valor String = pageContext.getParameter (< *projectIdAdvSearch* >);

// construir la cláusula WHERE apropiada para projectIdAdvSearch

// y la puso en su vo y ejecutar la consulta

// utilizando el método initQuery.

...

}

}

La siguiente tabla muestra las constantes que se pueden utilizar para determinar su actual panel de búsqueda. Todas las constantes se definen en la oracle.apps.fnd.framework.webui.OAWebBeanConstants clase.

|  |  |
| --- | --- |
| **Nombre constante** | **Panel actual** |
| OAWebBeanConstants.SEARCH | Sencillo panel de búsqueda |
| OAWebBeanConstants.ADVANCED\_SEARCH | Panel de búsqueda avanzada |
| OAWebBeanConstants.CUSTOMIZE | panel de vistas. |

**Opción 2***-* Utilice esta opción si desea buscar en un campo específico y guardar la búsqueda como una vista de personalización de usuario.

Procesar este campo en el panel Vistas, además del panel de búsqueda simple o avanzada.

Paso 3.1: Definir un campo oculto asociado ( OA: formValue ) en la región de mesa, con su userCustomizable propiedad establece en **verdadera** y viewUsageName y viewAttributeName establece en nulo.

Para nuestro ejemplo, definir un campo oculto con su conjunto de identificación para projectId en la región mesa.

Paso 3.2: Añadir las asignaciones apropiadas para este campo, incluyendo la simple y / o las asignaciones avanzadas de búsqueda. Para nuestro ejemplo, añadir tanto un simple y un mapeo de búsqueda avanzada de la projectId campo de tabla:

* Simple mapeo de búsqueda - Buscar material: projectIdSimpleSearch ; Resultados material: projectId
* Búsqueda avanzada de mapeo - Buscar material: projectIdAdvSearch ; Resultados material: projectId

Paso 3.4: Manejar el diccionario criterios y construir su cláusula WHERE para la projectId atributo:

processFormRequest pública vacío (OAPageContext pageContext, OAWebBean webBean)

{

// Buscar el panel actual

OAQueryBean queryBean = (OAQueryBean) webBean;

queryBean.getCurrentSearchPanel ();

// si usted está en la búsqueda o los paneles de búsqueda avanzada,

// manejar los criterios de la prensa botón Ir.

if (((SEARCH.equals currentPanel) || ADVANCED\_SEARCH.equals (currentPanel)) &&

queryBean.getGoButtonName ()! = null)

{

handleCriteria (pageContext, webBean);

}

// Si se encuentra en el panel de vistas, manejar los criterios en la personalización ir

// pulse el botón.

if ((CUSTOMIZE.equals (currentPanel) &&

pageContext.getPersonalizeGoButtonName ()! = null)

{

handleCriteria (pageContext, webBean);

}

}

handleCriteria public void (OAPageContext pageContext, OAWebBean webBean)

{

OAQueryBean queryBean = (OAQueryBean) webBean;

// Esto le da a los criterios actuales de atributos no-view

Diccionario [] DIC = queryBean.getNonViewAttrCriteria (pageContext);

// Si el diccionario está vacío, entonces significa que no hay criterios no vista

// está disponible, por lo que volver.

si (DIC nula == || dic.isEmpty ())

regreso;

// procesar lo contrario, el diccionario para construir su cláusula where.

tamaño int = dic.length;

// Iterar a través del diccionario para establecer sus cláusulas where

for (int i = 0; i <dictSize; i ++)

{

// del artículo para el que se define los criterios.

Cadena itemName = (String) DIC [i] .get (OAViewObject.CRITERIA\_ITEM\_NAME);

// La condición es la condición SQL - ejemplos: como, etc =

condición String = (String) DIC [i] .get (OAViewObject.CRITERIA\_CONDITION);

// El valor es el valor introducido con el% adecuado en función de la condición

valor Object = DIC [i] .get (OAViewObject.CRITERIA\_VALUE);

// condición de unión es AND u OR dependiendo de lo que el usuario elige

Cadena joinCondition = (String) DIC [i] .get (OAViewObject.CRITERIA\_JOIN\_CONDITION);

// Se pueden utilizar los siguientes fragmentos de código si es necesario encontrar el real

// base de datos de nombre de columna del elemento.

Cadena viewAttributeName = (String) DIC [i] .get (CRITERIA\_VIEW\_ATTRIBUTE\_NAME);

Cadena columnName = vo.findAttributeDef (viewAttributeName) .getColumnNameForQuery ();

// Ahora usar la información anterior para construir su cláusula where.

Cadena whereClause = ...

// Por último invocar un método personalizado en su objeto vista para establecer la cláusula where

// no se debe ejecutar la consulta si se llama getNonViewAttrCriteria.

// dónde cláusula

}

}

Paso 3.5: Repita el código anterior que maneja los criterios en el panel Vistas en su processRequest () con el fin de manejar los criterios asociados con la personalización por defecto (si lo hay).

processRequest pública vacío (OAPageContext pageContext, OAWebBean webBean)

{

OAQueryBean queryBean = (OAQueryBean) webBean;

// Buscar el panel actual

OAQueryBean queryBean = (OAQueryBean) webBean;

queryBean.getCurrentSearchPanel ();

// Si se encuentra en el panel de vistas, manejar los criterios

// para la personalización por defecto

if ((CUSTOMIZE.equals (currentPanel) &&

queryBean.getDefaultCustomization ()! = null)

{

handleCriteria (pageContext, webBean);

}

}

Paso 4: Usar la getCriteria () método.

Esto es muy similar a la Opción 2 - getNonViewAttrCriteria . La diferencia clave es que el getCriteria () método devuelve los criterios completos definidos por el usuario. El ejemplo de código anterior es válido también.

**Regiones de consulta múltiples**

**Caso 1: Regiones de consulta múltiples *no* construye con el modo Auto Criterios de personalización**

Para mostrar varias consultas ( OAQueryBean regiones) que no se construyen con el modo de criterios de personalización automática al mismo tiempo en una sola página, incluir lo siguiente processRequest () código.

**Importante:** Si va a incluir un HGrid dentro de una de sus regiones de consulta, este código debe añadirse a un controlador que se encuentra por encima de la queryBean en la jerarquía.

**Nota:** Crear las regiones de consulta declarativa como se describe en la [búsqueda avanzada](https://translate.googleusercontent.com/translate_f#adv) , [búsqueda simple](https://translate.googleusercontent.com/translate_f#simple) y [usuario de Personalizable Vistas](https://translate.googleusercontent.com/translate_f#views) secciones anteriores.

processRequest pública vacío (OAPageContext pageContext, OAWebBean webBean)

{

super.processRequest (pageContext, webBean);

OAQueryBean consulta1 = (OAQueryBean)

webBean.findIndexedChildRecursive ( "queryRegion1");

query1.setAdvancedSearchButtonName ( "AdvancedSearch1");

query1.setClearButtonName ( "ClearButton1");

query1.setGoButtonName ( "GoButton1");

query1.setPersonalizationsPoplistName ( "PersPoplist1");

query1.setPersonalizeButtonName ( "PersButton1");

query1.setPersonalizeGoButtonName ( "PersGoButton1");

query1.setSaveSearchButtonName ( "SaveSearchButton1");

query1.setSaveSearchButtonText ( "Memoria de búsqueda 1");

query1.setSimpleSearchButtonName ( "SimpleSearch1");

query1.setViewPersonalizationsButtonName ( "ViewPersButton1");

query1.setAddPoplistName ( "AddPoplist1");

query1.setAddButtonName ( "AddButton1");

query1.setAdvRadioGroupName ( "RadioGroup1");

OAQueryBean Query2 = (OAQueryBean)

webBean.findIndexedChildRecursive ( "queryRegion2");

query2.setAdvancedSearchButtonName ( "AdvancedSearch2");

query2.setClearButtonName ( "ClearButton2");

query2.setGoButtonName ( "GoButton2");

query2.setPersonalizationsPoplistName ( "PersPoplist2");

query2.setPersonalizeButtonName ( "PersButton2");

query2.setPersonalizeGoButtonName ( "PersGoButton2");

query2.setSaveSearchButtonName ( "SaveSearchButton2");

query2.setSaveSearchButtonText ( "Memoria de búsqueda 2");

query2.setSimpleSearchButtonName ( "SimpleSearch2");

query2.setViewPersonalizationsButtonName ( "ViewPersButton2");

query2.setAddPoplistName ( "AddPoplist2");

query2.setAddButtonName ( "AddButton2");

query2.setAdvRadioGroupName ( "RadioGroup2");

// En cada región consulta posterior después de la primera, agregue esta línea de código

// para asegurar que la OA Marco para nombrar correctamente los elementos de criterios.

query2.setAttributeValue (OAWebBeanConstants.IS\_NON\_FIRST, Boolean.TRUE);

}

**Caso 2: regiones de consulta múltiples, construidos con el modo Auto Criterios de personalización**

Para mostrar varias regiones de consulta que se construyen con el modo de criterios de personalización automática al mismo tiempo en una sola página, debe realizar los siguientes pasos para cada una de las regiones de consulta secundarias. (Todas las regiones de consultas excepto el primero en una página de consulta se consideran regiones secundarias.)

**Nota:** Crear las regiones de consulta declarativa como se describe en la [búsqueda avanzada](https://translate.googleusercontent.com/translate_f#adv) , [búsqueda simple](https://translate.googleusercontent.com/translate_f#simple) y [usuario de Personalizable Vistas](https://translate.googleusercontent.com/translate_f#views) secciones anteriores.

Paso 1: Establecer el Agregar indexado Niños propiedad del contenedor para el frijol consulta a falsa .

Paso 2: Llame a la oracle.apps.fnd.framework.webui.OAQueryUtils createSecondaryQueryBean API en el processRequest método de su controlador. Esta llamada crea el grano de consulta web secundaria, lo añade a la matriz que contiene frijol web, y ajusta todos los parámetros necesarios, tales como IS\_NON\_FIRST . No es necesario configurar manualmente estos parámetros, como en [el Caso 1](https://translate.googleusercontent.com/translate_f#mq_case1) anterior. De hecho, si ha realizado las llamadas anteriores para configurar manualmente estos parámetros, debe eliminarlos para evitar conflictos con el createSecondaryQueryBean llamada a la API.

**Importante:** Si va a incluir un HGrid dentro de una de sus regiones de consulta, este código debe añadirse a un controlador que se encuentra por encima de la queryBean en la jerarquía.

**Búsqueda rápida**

**La aplicación declarativa**

No se puede poner en práctica la búsqueda rápida de forma declarativa, no puede seleccionar los granos apropiados para la región de búsqueda rápida, y no se puede establecer la región de búsqueda rápida en la pageLayout.

**Control de tiempo de ejecución**

**Crear una instancia de búsqueda rápida**

Para crear una búsqueda rápida y asociarlo con una página, haga lo siguiente:

processRequest pública vacío (OAPageContext pageContext, OAWebBean webBean)

{

super.processRequest (pageContext, webBean);

OAPageLayoutBean pageLayout = pageContext.getPageLayout ();

// Tenga en cuenta que no hay necesidad de llamar a prepareForRendering () cuando se ajusta el

// componente de búsqueda rápida ya que el marco OA no manipula en

// de todas formas.

pageLayout.setQuickSearch (quickSearchRN);

}

**Manejar Va el botón Press**

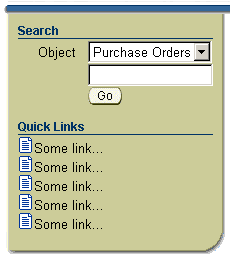
Ver las instrucciones de implementación de búsqueda básicas de la simple búsqueda - [Búsqueda manual](https://translate.googleusercontent.com/translate_f#simple_manual) sección.

**Lado Navegación Buscar**

Actualmente, no hay OAQueryBean apoyo a una búsqueda implementado en la barra de navegación.

**Nota:** Se puede definir la región de búsqueda de navegación de la forma declarativa, sin embargo, debe asociarla con el componente de navegación de la programación.

Figura 15: Lado de navegación búsqueda del cuadro de herramientas de aplicaciones OA Framework Tutorial.



Para implementar la región de búsqueda de navegación lado como se muestra en la Figura 15 anterior:

Paso 1: Crear la región de búsqueda reutilizable. Seleccione Archivo> Nuevo desde el menú principal JDeveloper.

* Paso 1.1: En el nuevo cuadro de diálogo, expanda el nodo de nivel Web, seleccione Componentes de OA y seleccione RegiónDel artículos lista. Seleccione el Aceptar botón.
* Paso 1.2: En el cuadro de diálogo Nueva Región, introduzca un nombre que cumpla con las [Normas Marco de la OA de archivos](https://translate.googleusercontent.com/cs/cs_filedir.htm) , a continuación, seleccione el paquete y establecer el estilo de **cabecera** . Seleccione el Aceptar botón.

Paso 2: Seleccione su región de cabecera en el panel Estructura JDeveloper, haga clic derecho y seleccione Nuevo> Región . Dar a esta región una [compatible con los estándares](https://translate.googleusercontent.com/cs/cs_filedir.htm)de identificación y establecer su estilo de región a [messageComponentLayout](https://translate.googleusercontent.com/feat_layout.htm#mcl) .

Paso 3: Seleccione su región messageComponentLayout, haga clic derecho y seleccione Nuevo> Elemento . Dar a este tema una [compatible con los estándares](https://translate.googleusercontent.com/cs/cs_filedir.htm)de identificación y establecer su estilo de **messageChoice** . Configurar el poplist para consultar una lista de objetos, (véase la [Norma Web Widgets](https://translate.googleusercontent.com/feat_widgets.htm) para obtener información sobre la creación de poplists), y establecer su Prompt a **objetos** .

Paso 4: Seleccione la región messageComponentLayout de nuevo, haga clic derecho y seleccione Nuevo> Elemento . Configurar este **messageTextInput** campo como se describe en el [estándar Web Widgets](https://translate.googleusercontent.com/feat_widgets.htm) . Usted debe *no* especificar un símbolo del sistemaviviendas en este campo, pero se debe establecer su texto adicional valor para el [cumplimiento de accesibilidad](https://translate.googleusercontent.com/cs/cs_viewgeneral.htm#508) . Por último, establecer la longitud de un valor razonable (este ejemplo es **15** ), de modo que la navegación lateral no hace excesivamente amplia.

Paso 5: Seleccione su región messageComponentLayout en el panel Estructura JDeveloper, haga clic derecho y seleccione Nuevo> Región . Dar a esta región una [compatible con los estándares](https://translate.googleusercontent.com/cs/cs_filedir.htm)de identificación y establecer su estilo de región a **messageLayout** .

Paso 6: Seleccione su región messageLayout, haga clic derecho y seleccione Nuevo> Elemento . Dar a este tema una [compatible con los estándares](https://translate.googleusercontent.com/cs/cs_filedir.htm)de identificación y establecer su estilo de **submitButton** . Establecer el conjunto de atributos para el atributo estándar Botón OA Framework fijan **/ Oracle / aplicaciones / enc / attributesets / Botones / Ir** .

Paso 7: Para asociar una región de búsqueda mediante declaración definida con una exploración lateral, haga lo siguiente en un controlador de nivel de página:

processRequest pública vacío (OAPageContext pageContext, OAWebBean webBean)

{

super.processRequest (pageContext, webBean);

// instancia de un componente de navegación lateral.

OASideNavBean sideNav =

(OASideNavBean) createWebBean (pageContext,

OAWebBeanConstants.SIDE\_NAV\_BEAN,

null, "hpSideNav");

// instancia de la región de búsqueda de forma declarativa definido. Tenga en cuenta que el segundo

// "crear" parámetro es el nombre completo de la región de búsqueda reutilizable.

Búsqueda OAHeaderBean = (OAHeaderBean) createWebBean (pageContext,

"/ Oracle / aplicaciones / enc / marco / caja de herramientas / tutorial / webui / HomeSearchRN"

"SearchRN", true);

// cada componente necesita un nombre

// usando OA Extensión JDeveloper

sideNav.addIndexedChild (búsqueda);

// Usted no tiene que hacer nada para conseguir el estilo CSS para la correcta

// texto de cabecera / línea (esto hace automáticamente basándose en el color de la

// fondo). Usted tiene que establecer el tamaño del texto de cabecera (esto no puede

// ser cambiado mediante el uso de un estilo CSS).

search.setSize (2); // 2 es el tamaño más pequeño

// obtener un identificador para el componente de diseño de página y establecer su "arranque" a la

// de navegación de la que hemos creado.

OAPageLayoutBean pageLayout = pageContext.getPageLayoutBean;

// Tenga en cuenta que debe llamar prepareForRendering () antes de configurar el inicio o

// no quede inutilizada.

pageLayout.prepareForRendering ();

pageLayout.setStart (sideNav);

}

Paso 8: Si la consulta subyacente no está restringido (la consulta incluye una cláusula WHERE de los resultados de una consulta performant), se *debe* aplicar selectivos criterio de búsqueda. Para ello, siga el procedimiento descrito en la Búsqueda sencilla - [Búsqueda manual](https://translate.googleusercontent.com/translate_f#simple_manual) sección.

Paso 9: Para manejar el Go pulse el botón, consulte las instrucciones de aplicación de búsqueda básicas de la simple búsqueda - [Búsqueda manual](https://translate.googleusercontent.com/translate_f#simple_manual) sección.

**Consideraciones de personalización**

Como se discutió en la [Regiones de consulta múltiple](https://translate.googleusercontent.com/translate_f#multiple) sección, todas las regiones secundarias de consulta de una página (es decir, todas las regiones de consultas excepto el primero de una página) deben tener el atributo OAWebBeanConstants.IS\_NON\_FIRSTestablece en Boolean.TRUE , mientras que la primera región de consulta en la página no debería tener este conjunto de atributos o lo han ajustado a Boolean.FALSE .  
  
Si utiliza la OA marco Personalización para agregar una región secundaria consulta predefinida a una página que no tiene una región de consulta existente, entonces la región de consulta recién añadido se convierte en la primera región de consulta en esa página. Esto, sin embargo, conduce a un comportamiento inesperado, ya que el código del controlador existente asociado con la región de consulta recién añadido tendrá el atributo OAWebBeanConstants.IS\_NON\_FIRST establece en Boolean.TRUE .   
  
Para solucionar este problema, se extienden controlador de la región de consultas que actualmente establece el atributo OAWebBeanConstants.IS\_NON\_FIRST a Boolean.TRUE . Después de la super.processRequest llamada, obtener un identificador para el queryBean y establecer el atributo OAWebBeanConstants.IS\_NON\_FIRSTa Boolean.FALSE como se muestra:

processRequest pública vacío (OAPageContext pageContext, OAWebBean webBean)

{

super.processRequest (pageContext, webBean);

OAQueryBean consulta1 = (OAQueryBean)

webBean.findIndexedChildRecursive ( "queryRN1");

query1.setAttributeValue (OAWebBeanConstants.IS\_NON\_FIRST, Boolean.FALSE);

}

Utilice la OA marco Personalización de actualizar la propiedad clase controlador de la región consulta con su nuevo controlador.

**Problemas conocidos**

* Cuando se implementa el modo autoCustomizationCriteria para una región de consulta, la costumbre simples y avanzadas regiones de búsqueda deben residir en el mismo XML como las regiones de consulta y de mesa. Se requiere esta restricción porque las asignaciones de las regiones de búsqueda no utilizan un ID completo; los ID se supone que son válidos sólo dentro del documento actual.
* Si utiliza setQuery () en un objeto vista asociada con la región de resultados de consulta de frijol, entonces también debería llamar vo.setFullSqlMode (FULLSQL\_MODE\_AUGMENTATION) en el objeto de vista. Esto asegura que el orden o por la cláusula WHERE, que se genera por el marco de la OA, se pueden añadir correctamente a su VO. El comportamiento con FULLSQL\_MODE\_AUGMENTATION es el siguiente:
  1. La nueva consulta que ha establecido mediante programación entra en vigor cuando se llama setQuery y ejecutar la consulta.
  2. Si llama setWhereClause o si un cliente personaliza los criterios para su región, BC4J aumenta la whereClause en la consulta mediante programación que ha establecido. Por ejemplo:
  3. seleccionar \* de (la consulta programático establecido a través setQuery)
  4. donde (su cláusula programática donde establece a través setWhereClause)

ordenado por (su orden programático establecido a través SetOrderby)

La misma consulta se cambia de la siguiente manera si un cliente añade un criterio de uso de personalización:

seleccionar \* de (la consulta programático establecido a través setQuery)

donde (su programática cláusula where establece a través setWhereClause) Y (personalización adicional cláusula where)

ordenado por (su orden programático establecido a través SetOrderby)

**Advertencia:** Si no se establece FULLSQL\_MODE\_AUGMENTATION , la whereClause y / o la orderBy , que fue creado mediante programación, no va a aumentar en la nueva consulta que se configuran utilizando setQuery . En su lugar, va a aumentar en su consulta VO tiempo de diseño.

* Debido a cuestiones de tiempo, utilice siempre el controlador en la región de consulta para cualquier OAQueryBean métodos específicos.
* Al utilizar regiones de consulta en un flujo de varias páginas, las regiones de consulta *deben* tener identificadores únicos. Por ejemplo, los ID de región consulta para el EmpSearchPG y DeptSearchPG pueden ser EmpQueryRN y DeptQueryRN respectivamente.

**Información relacionada**

* Directrices de interfaz de usuario BLAF
  + [Buscar y plantillas de consulta](https://translate.google.com/translate?hl=es&prev=_t&sl=en&tl=es&u=http://uistandards/blaf/guidelines/search.html) [ [OTN Versión](https://translate.google.com/translate?hl=es&prev=_t&sl=en&tl=es&u=http://otn.oracle.com/tech/blaf/specs/search_flow.html) ]
* Javadoc
  + oracle.apps.fnd.framework.webui.beans.layout.OAQueryBean
  + oracle.apps.fnd.framework.webui.beans.layout.OAHeaderBean
  + oracle.apps.fnd.framework.webui.beans.layout.OAAdvancedSearchBean
  + oracle.apps.fnd.framework.webui.OAQueryUtils
* OA marco Caja de Herramientas Tutorial / Biblioteca Muestra
  + [Buscar Lab](https://translate.googleusercontent.com/toolbox/labs/lessons/search/exercise.htm)
  + Oracle / aplicaciones / enc / marco / caja de herramientas / samplelib / webui / SearchRBSPG
  + Oracle / aplicaciones / enc / marco / caja de herramientas / samplelib / webui / SearchACCPG
  + Oracle / aplicaciones / enc / marco / caja de herramientas / tutorial / webui / PoSearchPG
  + Oracle / aplicaciones / enc / marco / caja de herramientas / tutorial / servidor / PoSimmpleSummaryVOImpl
  + Oracle / aplicaciones / enc / marco / caja de herramientas / tutorial / servidor / SearchAMImpl

[Copyright © 2000,2010, Oracle y / o sus afiliados.](https://translate.googleusercontent.com/copyright.htm) [Todos los derechos reservados.](https://translate.googleusercontent.com/copyright.htm)