**Implementando el Modelo**

**Visión de conjunto**

Este documento describe cómo implementar sus objetos modelo en términos genéricos.

**Contenido**

* [Diseño de objetos modelo](https://translate.googleusercontent.com/translate_f#design)
* [Enfoque de construcción recomendado](https://translate.googleusercontent.com/translate_f#approach)
* [Paquetes de componentes de negocios](https://translate.googleusercontent.com/translate_f#bc4j_package)
* [Objetos de la entidad](https://translate.googleusercontent.com/translate_f#eo)
* [Asociaciones de entidades (objetos de asociación)](https://translate.googleusercontent.com/translate_f#ao)
* [Ver objetos y ver filas](https://translate.googleusercontent.com/translate_f#vo)
* [Ver enlaces](https://translate.googleusercontent.com/translate_f#vl)
* [Módulos de aplicación](https://translate.googleusercontent.com/translate_f#am)
* [Objetos de entidad, expertos de entidad, módulos de aplicación de 'validación' y objetos de vista de 'validación'](https://translate.googleusercontent.com/translate_f#eo_ee)
  + [Objetos de vista de validación (VVO)](https://translate.googleusercontent.com/translate_f#vvo)
  + [Módulos de aplicación de validación (VAM)](https://translate.googleusercontent.com/translate_f#vam)
  + [Expertos de la entidad](https://translate.googleusercontent.com/translate_f#ee)
* [Reutilizando Business Objects](https://translate.googleusercontent.com/translate_f#reusebc4j)

**Requisito de lectura**

Este documento asume que ha leído lo siguiente en la Guía del Desarrollador de OA Framework:

* [Edificio '¡Hola, mundo!'](https://translate.googleusercontent.com/gs/gs_hello.htm)
* [JSP Application Primer](https://translate.googleusercontent.com/ess/ess_jspprimer.htm)
* [Anatomía de una página de Marco OA](https://translate.googleusercontent.com/ess/ess_anatomy.htm)
* [OA Framework State Management](https://translate.googleusercontent.com/ess/ess_state.htm)

**Diseño de objetos modelo**

**Separación del código 'Cliente' / 'Servidor'**

Dentro de la arquitectura Modelo-Vista-Controlador, OA Marco establece una clara distinción entre las clases "servidor", una distinción que en la superficie puede parecer estar en conflicto con la arquitectura de la aplicación JSP como se describe en el "cliente" y [JSP aplicación de la imprimación](https://translate.googleusercontent.com/ess/ess_jspprimer.htm) . En el capítulo 2, se dice que una aplicación *típica* JSP tiene 3 niveles físicos:

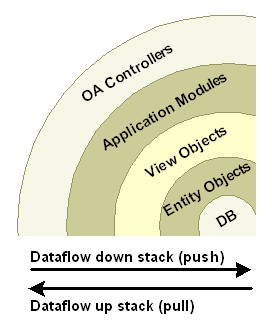
* El navegador (el cliente donde nuestros HTML renderiza y los usuarios interactúan con la interfaz de usuario)
* El servidor de aplicaciones web (el nivel medio donde se construye nuestra jerarquía de bean web UI y se ejecuta nuestra lógica de negocios de aplicaciones)
* El servidor de base de datos

Dentro del nivel medio, OA Framework establece una distinción adicional entre las clases "cliente" y "servidor" de la siguiente manera:

* Cliente las clases (código de vista y controlador) manejan la interfaz de usuario HTML
* Servidor clases (código de modelo) admite cualquier interfaz de usuario de cliente (no solo de OA Framework)

Esta distinción es importante porque conserva la capacidad de usar el código del servidor con diferentes clientes. Considere la siguiente imagen que ilustra las relaciones entre los controladores de IU de una página, módulos de aplicación, objetos de vista específicos de IU y los objetos de entidad subyacentes:

Figura 1: Marco de OA "cebolla" que muestra las capas de código y los límites



En general, para habilitar reutilice en los límites de la capa, los objetos referencian hacia abajo la pila de flujo de datos, pero no hacia arriba.

* El código del modelo nunca debe hacer referencia directamente al código del controlador. Por ejemplo, objetos de vista y módulos de aplicación *no* deben llamar a métodos en los controladores de la interfaz de usuario, y objetos de entidad *no*deben hacer referencia a los módulos de aplicación y objetos de vista de interfaz de usuario, como se discute más adelante.Sin embargo, los objetos de entidad pueden utilizar, y lo hacen, los módulos de aplicación de validación del lado del servidor y los objetos de visualización.
* Nunca haga referencia / importe ninguna implementación en el servidor clases o interfaces en el lado del cliente (clases / interfaces que están en el oracle.apps.fnd.framework.server paquete). Por ejemplo, *no* llamar a métodos en un oracle.apps.fnd.framework.OADBTransaction en el controlador de interfaz de usuario.
* Si necesita que el código del servidor le sirva, siempre enrute sus llamadas a través del módulo de aplicación radicular utilizando el genérico "remoteable" InvokeMethod () método en la interfaz oracle.apps.fnd.framework.OAApplicationModule, o crear una interfaz para sus módulos de aplicación para que pueda llamar a los métodos escritos, cuya invocación se puede comprobar en tiempo de compilación. El módulo de aplicación puede delegar o implementar la lógica según sea necesario.

**Nota:** El OAApplicationModule interfaz está en el paquete oracle.apps.fnd.framework y por lo tanto, no viola la norma establecida en la primera frase de este punto de bala. Todas las clases, interfaces y excepciones en el paquete oracle.apps.fnd.framework con seguridad se pueden utilizar tanto en código de cliente y servidor.

Si opta por crear una interfaz para su módulo de aplicación en lugar de utilizar InvokeMethod (), crear este interfaz en el directorio del paquete inmediatamente anterior a su aplicación. Por ejemplo, la interfaz EmployeeAM para el módulo de aplicación oracle.apps.fnd.framework.toolbox.labsolutions.server.EmployeeAMImpl se debe crear en el paquete oracle.apps.fnd.framework.toolbox.labsolutions.

* Nunca incluya JDBC u otro procesamiento del lado del servidor directamente en su código de cliente. De nuevo, si el cliente de UI necesita información de este servidor, debe consultar el módulo de la aplicación, que luego puede delegar o implementar la solicitud de manera apropiada.

**Codificación de cumplimiento de normas**

Antes de definir cualquier objeto modelo o escribir un código de apoyo, lea los siguientes documentos cuidadosamente. Si bien este tema menciona varios estándares de modelos clave, no pretende ser una lista de verificación exhaustiva. Para cualquier código OA Framework que escribe, los documentos en el capítulo 8 deben considerarse la *única fuente de la verdad* para los estándares de codificación.

* [Capítulo 8: Estándares de codificación Java de Oracle E-Business Suite](https://translate.googleusercontent.com/cs/cs_javafunc.htm)
* [Capítulo 8: Naming Framework / File / Package / Directory Structure Standards](https://translate.googleusercontent.com/cs/cs_filedir.htm)
* [Capítulo 8: Estándares de codificación del modelo de marco de OA](https://translate.googleusercontent.com/cs/cs_mdlgeneral.htm)

**Enfoque de construcción recomendado**

Si se está preparando para construir su primera aplicación OA Framework , es posible que le resulte más fácil proceder de la siguiente manera para un módulo pequeño (una sola página o algunas páginas simples relacionadas que participan en la misma transacción).

**Nota:** Esto supone que haya completado su trabajo de diseño, y están listos para su implementación. También asume que está construyendo un módulo completo, no solo la interfaz de usuario, o simplemente la lógica de negocios.

1. Cree los paquetes de componentes comerciales que necesite para sus objetos de modelo BC4J.
2. Implemente el módulo de aplicación declarativa BC4J, el objeto de entidad, la asociación de entidad, el objeto de vista y vea las definiciones de enlace necesarias para su (s) página (s). Agregue objetos de vista a su (s) módulo (s) de aplicación raíz según corresponda. No se preocupe por escribir código en esta etapa.
3. Crear la definición de menú para su aplicación (discutido en [Implementación de la vista](https://translate.googleusercontent.com/build_view.htm) ).
4. Crear los componentes de la interfaz de usuario OA para su página (s) (discutido en [Implementación de la vista](https://translate.googleusercontent.com/build_view.htm) ).
5. Crear e implementar código del controlador (discutido en [Implementación del Controlador](https://translate.googleusercontent.com/build_control.htm) ).
6. Implemente el código del módulo de aplicación de interfaz de usuario que respalda sus páginas.
7. Implementar la lógica de negocio del objeto de la entidad.

**Paquetes de componentes de negocios**

Todos los componentes del modelo BC4J deben pertenecer a un componente empresarial (BC4J) paquete.

**Nota:** En esta sección se supone que se ha configurado el entorno de desarrollo, creado un área de trabajo de la OA, y se define un origen de datos de trabajo como se describe en [la creación y ejecución "Hola, mundo!"](https://translate.googleusercontent.com/gs/gs_hello.htm) .

1. En JDeveloper Navigator, seleccione el proyecto OA donde desea crear su paquete.
2. En el menú principal, seleccione Archivo> Nuevo para abrir la nueva galería de objetos.
3. En el árbol de categorías, expanda el nodo capa de negocio, y seleccione Componentes de Negocio (BC4J).
4. En la lista Elementos, seleccione Componentes paquete de negocios para abrir el Asistente del paquete de componentes de negocio. También puede hacer clic en el Proyecto OA y seleccione Componentes Nuevo paquete de negocios para navegar directamente a la Asistente del paquete de componentes de negocio.
5. En el paso 1 de 3, escriba el nombre del paquete que cumpla con los [OA Marco de archivo / directorio / estructura de paquetes](https://translate.googleusercontent.com/cs/cs_filedir.htm) estándares. También seleccione Entidad objetos asignados a los objetos del esquema de base de datos en la sección de modelado entidad comercial. Seleccione el botón Siguiente.
6. En el paso 2 de 3, verificar el nombre de conexión (debe apuntar a la base de datos en la que desea trabajar; JDeveloper utiliza esta información en toda la BC4J asistentes componentes). Ajuste el tipo de SQL y el tipo de mapa de "Oracle".
7. Seleccione el botón Finalizar para guardar los cambios.

Para cambiar la conexión de la base de datos asociado con un paquete BC4J existente, que debe hacer si cambia su entorno de desarrollo:

1. Seleccione el proyecto de OA con el paquete de componentes de negocio que desea editar.
2. Haga clic derecho y seleccione Editar negocios Componentes del proyecto.
3. En el Asistente para proyectos Business Components, seleccione Conexión.
4. Especifique su nueva base de datos.
5. Seleccione Aceptar para guardar los cambios.

**Objetos de la entidad**

Esto introduce los conceptos básicos de creación / uso de objetos de entidad. Ver [Java Entidad objetos](https://translate.googleusercontent.com/bus/bus_java.htm) y [PL / SQL Entidad objetos](https://translate.googleusercontent.com/bus/bus_plsql.htm)en el Capítulo 5 para obtener información adicional acerca de cómo trabajar con estos objetos.

Como se describe en [Anatomía de una OA marco de páginas](https://translate.googleusercontent.com/ess/ess_anatomy.htm) , un objeto entidad aplica las reglas de negocio para una tabla, una vista o instantánea dada. Los objetos de entidad están destinados a ser utilizados en muchos clientes (no solo en la interfaz de usuario de OA Framework) y, como tales, deben consolidar toda la validación / comportamiento relevante para la tabla.

* Cada tabla debe tener como máximo un objeto de entidad.
* Los objetos de entidad deben incluir atributos para todas las columnas en su tabla.
* Usted puede subclase sus propios objetos de entidad común (por ejemplo, ver los [temas avanzado modelo de desarrollo](https://translate.googleusercontent.com/adv/adv_model.htm) en el capítulo 6 para aprender cómo crear objetos de entidad polimórficos que se extienden de una clase base común).
* Normalmente agregará la inicialización de objetos, la validación de atributos, la validación de nivel de entidad y otros comportamientos funcionales a los objetos de su entidad.
* También puede crear singletons de "experto en entidades" para realizar trabajos compartidos por múltiples objetos de entidad relacionados en un objeto comercial compuesto, como una orden de compra que tiene un encabezado, líneas y envíos. Otros objetos de entidad de referencia también pueden utilizar el experto entidad para realizar validaciones de peso ligero (por ejemplo, una orden de compra podría pedir experto entidad del proveedor si un ID de proveedor que quiere hacer referencia es válida). Los expertos de la entidad se discuten más adelante.
* Finalmente, puede crear objetos e interfaces de "ayudante" de servidor según sea necesario para crear un código modular.Por ejemplo, como se ilustra en OA Framework ToolBox Tutorial, puede crear uno o más objetos auxiliares para realizar el procesamiento en múltiples tipos de objetos de entidades.

**Implementación declarativa**

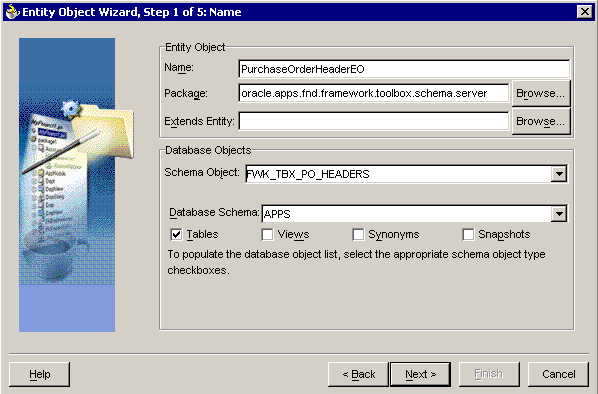
Para obtener información adicional sobre Entity Object Wizard propiedades que no se describen específicamente aquí, consulte la documentación de JDeveloper.

**Nota:** Puede acceder contextual Ayuda en cualquier BC4J asistentes mediante la selección de la tecla F1 del teclado.

Para crear un nuevo objeto de entidad en un paquete de componentes de negocio (BC4J):

1. En JDeveloper Navigator, seleccione el paquete BC4J donde desea crear su objeto de entidad.
2. En el menú principal, seleccione Archivo> Nuevo para abrir la nueva galería de objetos.
3. En el árbol de categorías, expanda el nodo capa de negocio, y seleccione Componentes de Negocio (BC4J).
4. En la lista Elementos, seleccione Entidad Objeto para abrir el Asistente para objetos Entidad. También puede hacer clic en el paquete de BC4J y seleccione nueva entidad objeto de navegar directamente al Asistente para objetos Entidad.
5. En la página Nombre (Paso 1 de 5):

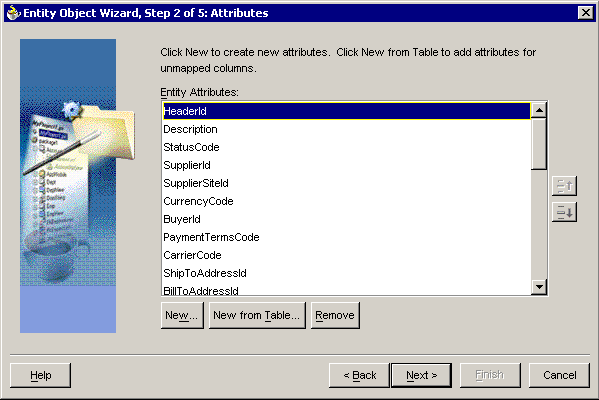
Figura 2: Página del nombre del asistente de objetos de la entidad (paso 1 de 5)



* + Especifique el nombre del objeto entidad, de acuerdo con los [OA Marco de archivo / directorio / estructura de paquetes](https://translate.googleusercontent.com/cs/cs_filedir.htm)estándares.
  + Compruebe que ha seleccionado el paquete BC4J derecha.
  + *No* introduzca un valor en el campo Objeto Entidad amplía a menos que esté de subclases deliberadamente uno de sus propios objetos de entidad.
  + Especificar un objeto de esquema (el nombre exacto de la tabla para el objeto de entidad), como se muestra en la Figura 2, para mejorar el rendimiento del asistente. *No es* necesario comprobar los sinónimos o casillas de verificación Tablas.
  + Seleccione Siguiente para continuar.

1. En la página Atributos (Paso 2 de 5), debería ver todas las columnas de la tabla que ha especificado en la página Nombre.

Figura 3: Página de atributos del asistente de objetos de la entidad (paso 2 de 5)



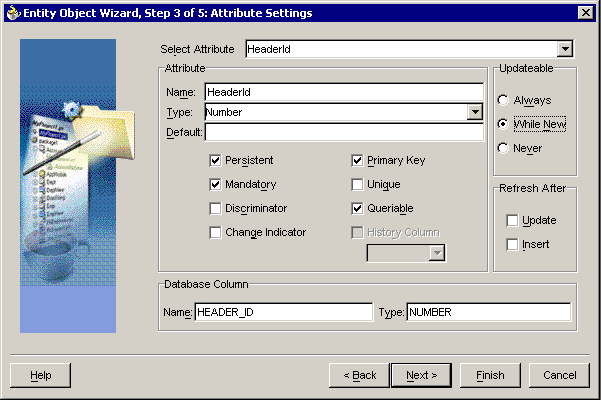
* + *No* eliminar cualquier atributo de entidad con el fin de incluir todas las columnas de la tabla,
  + Seleccione Nuevo ... para crear un atributo transitoria que se utiliza en la lógica de negocio, tales como OrderTotal calculado en una orden de compra que se utiliza para el control de su aprobación.

**Nota:** Para mostrar un valor calculado para un objeto entidad que no es realmente relevante para la lógica de negocio en sí (que es muy específica-UI), siempre se puede crear un atributo para que en el objeto de vista como se describe a continuación.

* + Seleccione Siguiente para continuar.

1. En las configuraciones de atributos página (Paso 3 de 5), verifique o establezca la siguiente información para cada uno de los atributos del objeto de la entidad:

Figura 4: Página de configuración de atributos del asistente de objetos de la entidad (paso 3 de 5)



* + Las propiedades de los atributos y base de datos de la columna Nombre y tipo predeterminado correctamente desde la definición de la tabla.
  + Todos los atributos de columna de base de datos tienen las casillas de verificación y persistentes queriableseleccionados como se muestra.
  + por columnas de clave principal, asegúrese de que la clave principal y casillas de verificación obligatorios son seleccionados.

**Advertencia:** Si usted no puede definir sus claves primarias, BC4J generará una clave ROWID para usted. *Debe definir explícitamente su clave principal en lugar de depender de ROWID como la clave principal ya que esto puede conducir a problemas.* Por ejemplo, en un clúster de aplicación real con particiones de base de datos, la misma fila de la base de datos puede tener un ROWID diferente en diferentes particiones. De forma similar, si una tabla está particionada, ROWID puede cambiar dentro de una transacción si la fila se migra.

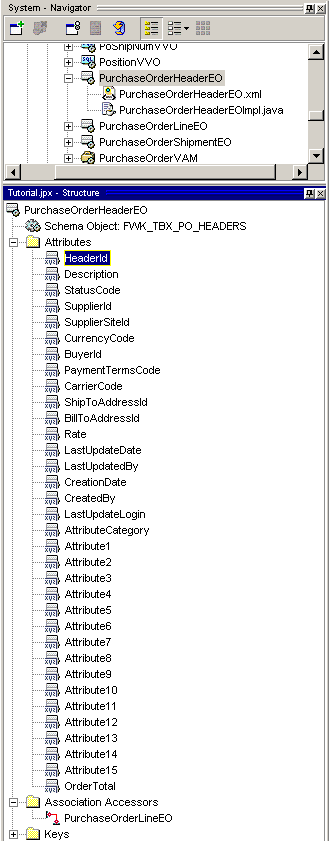
* + Para las columnas que no son actualizables, o actualizable sólo cuando nueva, seleccione el botón de radio actualizableapropiado.
  + Para las columnas cuyos valores cambian después de trigger ejecutar, seleccione la actualización Después de actualizar o insertar en su caso.
  + Nunca seleccione la casilla de verificación único; la comprobación de exclusividad siempre debe realizarse mediante programación (el estado de esta casilla de verificación no tiene ningún impacto en el comportamiento del tiempo de ejecución, BC4J lo usa para crear una restricción de base de datos si genera una tabla desde el asistente de EO).

**Nota:** El único no tiene efecto si se crea el EO en una mesa de base de datos ya existente.

* + El discriminador la columna se usa para polimórficos objetos de entidad tal como se describe en el Capítulo 6 [Temas avanzado modelo de desarrollo](https://translate.googleusercontent.com/adv/adv_model.htm) .
  + Si estás usando un Columna Número de versión del objeto, seleccione la Cambiar la casilla de verificación Indicador de ella. Ver el Capítulo 5 para información sobre esto.
  + Seleccione Siguiente para continuar.

1. En la página de Java (Paso 4 de 5) página:
   * Marque la opción para la generación de una clase de objeto de entidad. En el cuadro Generar métodos, optar porgenerar descriptores de acceso, un método de creación y un método Delete.
   * Se extiende seleccione el botón ... para verificar que se le subclassing oracle.apps.fnd.framework.server.OAEntityImplpara la fila (el objeto entidad).
2. Seleccione Finalizar para guardar la definición de objeto de entidad e implementación. BC4J creará un archivo de definición XML y un archivo de implementación Java para su objeto entidad como se muestra en la Figura 5. Tenga en cuenta que puede ver rápidamente la tabla subyacente y los atributos incluidos en el objeto entidad simplemente seleccionándolo en el Navegador del sistema.
3. Seleccione el objeto de entidad, haga clic derecho y seleccione Editar *<EntityName>* ... navegar a la sintonización página y comprobar el uso Actualizar los lotes. Establecer el valor umbral de **100.** Ver la [codificación de marco Modelo OA estándar M68](https://translate.googleusercontent.com/cs/cs_mdlgeneral.htm#M68) para obtener información adicional acerca de las actualizaciones de DML por lotes, incluyendo una lista de situaciones en las que esta función no se puede utilizar.

Figura 5: Navegador del sistema JDeveloper y panel de estructura que muestra un objeto de entidad seleccionado



**Objetos de entidad "\_TL" multilenguaje**

Para crear un multilenguaje "\_TL" objeto de entidad, siga las instrucciones especiales publicados en [Java Entidad objetos para tablas traducibles](https://translate.googleusercontent.com/bus/bus_java.htm#tl) .

**Objetos de entidad PL / SQL**

Para crear un PL objeto de entidad / SQL, siga las instrucciones especiales publicados en [PL / SQL Objetos de entidad](https://translate.googleusercontent.com/bus/bus_plsql.htm) .

**Control programático**

Para obtener información detallada sobre la codificación del objeto de entidad la lógica de negocio, ver [Java Entidad objetos](https://translate.googleusercontent.com/bus/bus_java.htm) y [PL / SQL Objetos de entidad](https://translate.googleusercontent.com/bus/bus_plsql.htm) .

**Asociaciones de entidades (objetos de asociación)**

Asociaciones permitirte crear relaciones declarativas entre objetos de entidad. En tiempo de ejecución, BC4J usa estas relaciones para coordinar los objetos relacionados. Hay dos tipos básicos de asociaciones:

* **Composición** - Un fuerte asociación donde el objeto entidad de origen *posee* el objeto *de* entidad de destino. En otras palabras, el destino no puede existir independientemente de su origen. Por ejemplo, un encabezado de orden de compra se compone de líneas de orden de compra, que no tienen significado o duración de vida fuera del contexto de su encabezado.
* **Referencia** - Un débil asociación donde el objeto de la entidad de origen solo hace referencia al objeto de la entidad de destino. Por ejemplo, un encabezado de orden de compra hace referencia a un proveedor, pero el proveedor puede seguir existiendo independientemente de si una orden de compra lo hace referencia o no.

Cree asociaciones de composición según corresponda para todos sus objetos de entidad y asegúrese de que se creen, inicialicen y administren correctamente en tiempo de ejecución. BC4J trata automáticamente las composiciones como una unidad lógica, por lo que, por ejemplo, un encabezado de orden de compra se bloquea automáticamente incluso si realiza cambios solo en sus líneas).

Cree asociaciones de referencia para cualquier objeto de entidad que pueda actualizar o instanciar en tiempo de ejecución. Por ejemplo, cree asociaciones entre un encabezado de orden de compra y su proveedor si puede actualizar un proveedor mientras edita un pedido de compra, pero no cree asociaciones entre órdenes de compra y un objeto de entidad de código de búsqueda de términos de carga.

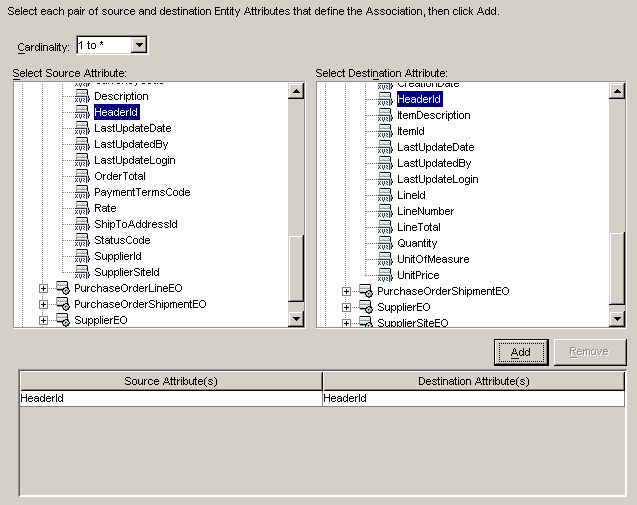
**Implementación declarativa**

Para información adicional acerca de Las propiedades de Association Wizard que no se describen específicamente aquí, consulte la documentación de JDeveloper.

**Nota:** Puede acceder a la ayuda específica al contexto, mientras que en cualquiera de los asistentes BC4J seleccionando la tecla F1del teclado.

1. En JDeveloper Navigator, seleccione el paquete BC4J donde desea crear su objeto de asociación.
2. En el menú principal, seleccione Archivo> Nuevo para abrir la nueva galería de objetos.
3. En el árbol de categorías, expanda el nodo capa de negocio, y seleccione Componentes de Negocio (BC4J).
4. En la lista Elementos, seleccione Asociación para abrir el Asistente Asociación. También puede hacer clic en el paquete de BC4J y seleccione Nuevo objeto de asociación para navegar directamente al Asistente de Asociación.
5. En la página Nombre (Paso 1 de 3):
   * Especificar nombre de la asociación, de acuerdo con los [OA Marco de archivo / directorio / estructura de paquetes](https://translate.googleusercontent.com/cs/cs_filedir.htm)estándares.
   * Compruebe que ha seleccionado el paquete BC4J derecha.
   * No introduce un valor en el campo Asociación Extiende a menos que esté de subclases deliberadamente una de sus propias asociaciones.
   * Seleccione Siguiente para continuar.
6. En la Entidad página de objetos (Paso 2 de 3), especifica la cardinalidad de la asociación (por ejemplo, ¿es una relación de uno a muchos?) Y seleccione la fuente y el destino unirse a atributos como se muestra en la Figura 5. Seleccione el botón Agregarpara cree la unión (repita según sea necesario para una relación de clave múltiple). Seleccione Siguiente para continuar.

Figura 5: Selección de objetos y atributos de entidad de origen y destino en la página Objeto de entidad (paso 1 de 3)



1. En la página Propiedades de Asociación (Paso 3 de 3):
   * Comprobar el descriptor de acceso Expose opciones según corresponda para los objetos de origen y destino (un descriptor de acceso permite que el objeto obtenga una referencia al otro extremo de la asociación).
   * Seleccione la casilla de verificación Asociación Composición si el objeto de destino no puede existir fuera del contexto del objeto de origen.

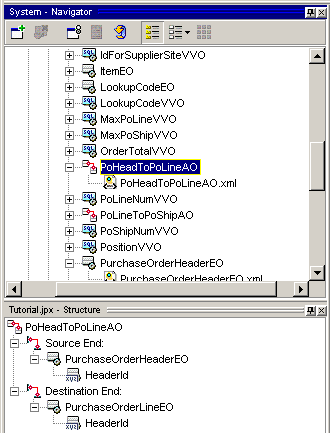
**Nota:** Para las composiciones, compruebe siempre la opción de descriptor de acceso en el objeto de destino Exposé.Opcionalmente, habilite esto para el objeto fuente como se requiere en su código.

* + No seleccione ninguna de las otras opciones de la página.

1. Seleccione Finalizar para guardar la asociación. BC4J crea un archivo de definición XML como se muestra en la Figura 6.

**Nota:** Se puede ver rápidamente la relación subyacente, simplemente seleccionando la asociación en el Navigator sistema.

Figura 6: Navegador del sistema JDeveloper y panel de estructura que muestra un objeto de asociación seleccionado



**Control programático**

Los objetos de asociación no tienen implementación, por lo que no se escribirá ningún código para ellos. En el Capítulo 5, discutimos cómo acceder a un objeto de entidad utilizando una asociación.

**Ver objetos y ver filas**

Esto introduce los conceptos básicos del objeto de vista y ver la creación / uso de fila. Ver [Ver objetos en detalle](https://translate.googleusercontent.com/bus/bus_vo.htm) en el Capítulo 5 para obtener información adicional acerca de cómo trabajar con estos objetos.

**Consideraciones de diseño**

Como se describe en [Anatomía de una OA marco de páginas](https://translate.googleusercontent.com/build_model.htm) , objetos de vista encapsulan consultas de base de datos y proporcionar acceso a objetos de entidades asociadas. Una de las decisiones de diseño de objeto de vista más importantes es si debe basarse en SQL simple o en objetos de entidad.

* Todos los objetos triviales de vista de UI para cosas como listas de valores (LOV) y poplists se basan en SQL simple.
* Cualquier punto de vista *de validación de* objetos, creados para poner en práctica las reglas de negocio sencillas para un objeto de entidad, se basan en SQL sin formato. Ver los [objetos de entidad, Expertos entidad, de la aplicación "validación" y "validación" objetos de vista](https://translate.googleusercontent.com/translate_f#eo_ee) de los siguientes temas para obtener información adicional.
* Todos los demás objetos de vista creados para la interfaz de usuario, independientemente de si son actualizables, se basan en objetos de entidad.

por Por motivos de rendimiento, los objetos de vista deben optimizarse para un caso de uso dado. Es preferible crear varios objetos de vista pequeños y específicos de la tarea para compartir un solo objeto de vista grande en un rango de UI. Los objetos de vista deben considerarse específicos de UI.

Evitar el uso de dinámico donde cláusulas siempre que sea posible (objetos de vista apoyan la capacidad de modificar sus definiciones declarativas programación). Si es posible, definir estáticamente 3 vista diferentes objetos de la misma, seleccione - cada uno con un declarativa cláusula WHERE a la que puede simplemente enlazar en tiempo de ejecución. Sin embargo, es apropiado para modificar la cláusula WHERE en vista de objetos utilizados con criterios de consulta complejo, ya que es poco práctico para crear definiciones individuales para cada combinación posible criterios de consulta.

Los objetos de vista, como cualquier objeto BC4J, se pueden crear declarativa y programáticamente. Por razones de rendimiento, es preferible, si puede, declarativamente define el objeto de vista.

**Implementación declarativa**

Para obtener información adicional sobre View Object Wizard propiedades que no se describen específicamente aquí, consulte la documentación de JDeveloper.

**Nota:** Puede acceder a la ayuda específica al contexto, mientras que en cualquiera de los asistentes BC4J seleccionando la tecla F1 del teclado.

**Importante:** Siempre que se crea un objeto de vista, *siempre* generar una clase de implementación de vista de primera fila. Debe generar una clase de implementación de objeto de vista solo si tiene la intención de escribir cualquier código para el objeto de vista.

Se puede crear un objeto de vista compartido, que oracle.apps.fnd.framework.server.OAViewObjectImpl subclases, que luego puede subclase para crear comportamientos más específicos.

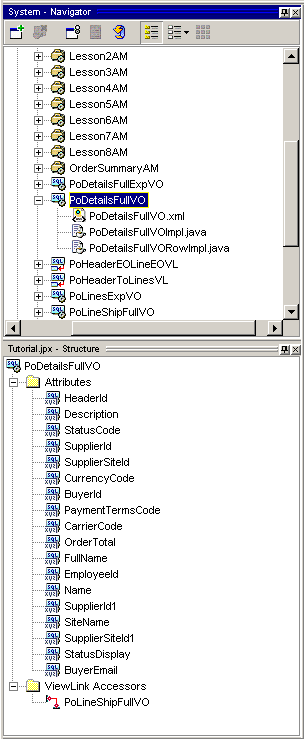
**Objetos de vista SQL**

Para crear un nuevo objeto de vista en un paquete de componentes de negocio (BC4J) que se basa completamente en una consulta SQL:

1. En JDeveloper Navigator, seleccione el paquete BC4J donde desea crear su objeto de vista.
2. En el menú principal, seleccione Archivo> Nuevo para abrir la nueva galería de objetos.
3. En el árbol de categorías, expanda el nodo capa de negocio, y seleccione Componentes de Negocio (BC4J).
4. En la lista Elementos, seleccione Ver Objeto para abrir el Asistente de visualización de objetos. Tenga en cuenta que también puede hacer clic en el paquete de BC4J y seleccione Nueva visualización de objetos para navegar directamente al Asistente de visualización de objetos.
5. En la página Nombre (Paso 1 de 7):
   * Especificar nombre del objeto vista de acuerdo con las [OA Marco de archivo / directorio / estructura de paquetes](https://translate.googleusercontent.com/cs/cs_filedir.htm)estándares.
   * Verifique que ha seleccionado el paquete BC4J correcto.
   * No introduce un valor en el campo Objeto vista se extiende a menos que esté de subclases deliberadamente uno de sus propios objetos de vista.
   * Seleccione Siguiente hasta llegar al Paso 5.
6. En la página de consulta (Paso 5 de 7):
   * Introduzca su consulta en el campo de instrucción Select (no incluya un punto y coma). Tenga en cuenta que siempre hay que utilizar fijaciones de estilo de Oracle (seleccione EMP\_NAME de emp donde emp\_id =: 1) si va a obligar a las variables en tiempo de ejecución.
   * Seleccione la prueba ... botón para verificar que su consulta es correcta.
   * Seleccione Siguiente para continuar.
7. En la página de asignaciones de atributos (Paso 6 de 7):
   * Verificar que las columnas de la consulta que ha definido en su partido de seleccionar la vista de atributos. Si no coinciden, haga clic en el valor Ver atributo que está en error para activar una lista emergente. Seleccione el atributo correcto.
   * Seleccione Siguiente para continuar.
8. En la página de Java (Paso 7 de 7):
   * Compruebe siempre la opción de generar una vista de primera fila de clase (incluyendo métodos de acceso).
   * Compruebe la opción de generar una Clase de visualización de objetos sólo si usted anticipa que escribir ningún código para su objeto vista (siempre se puede generar esta clase más adelante, si es necesario, o eliminarlo si es que generamos ahora y encuentre más tarde que no haya una necesidad para ello).
   * Se extiende seleccione el botón ... para asegurarse de que está de subclases las clases OA Framework oracle.apps.fnd.framework.server.OAViewObjectImpl y oracle.apps.fnd.framework.server.OAViewRowImpl según corresponda. Si necesita corregir los valores por defecto, seleccione Examinar ... para abrir la ventana Buscar superclase.
9. Seleccione Finalizar para guardar el nuevo objeto de vista. BC4J creará un archivo de definición XML e implementaciones Java como se muestra en la Figura 7.

**Nota:** Se puede ver rápidamente los atributos subyacentes y los vínculos de vista seleccionando la vista de objetos en el Navigator sistema.

Figura 7: Navegador del sistema JDeveloper y panel de estructura que muestra un objeto de vista seleccionado



En este punto, no has terminado el proceso de creación. Para continuar, debe editar el objeto de vista de la siguiente manera:

1. En el JDeveloper Navigator, seleccione la vista de objetos que acaba de crear, haga clic derecho y seleccione Editar *<viewobject\_name>* ....
2. En el Asistente para objetos Ver, seleccionar la sintonización.
3. En la página de optimización, desactive la casilla de verificación Habilitar pasivación. Seleccione Aceptar para guardar los cambios.

**Objetos de vista de objeto de entidad**

Para crear un nuevo objeto de vista en un paquete de componentes de negocio (BC4J) que se basa completamente en objetos de entidad:

1. En JDeveloper Navigator, seleccione el paquete BC4J donde desea crear su objeto de vista.
2. En el menú principal, seleccione Archivo> Nuevo para abrir la nueva galería de objetos.
3. En el árbol de categorías, expanda el nodo capa de negocio, y seleccione Componentes de Negocio (BC4J).
4. En la lista Elementos, seleccione Ver Objeto para abrir el Asistente de visualización de objetos. Tenga en cuenta que también puede hacer clic en el paquete de BC4J y seleccione Nueva visualización de objetos para navegar directamente al Asistente de visualización de objetos.
5. En la página Nombre (Paso 1 de 6):
   * Especificar nombre del objeto vista de acuerdo con las [OA Marco de archivo / directorio / estructura de paquetes](https://translate.googleusercontent.com/cs/cs_filedir.htm) estándares.
   * Verifique que ha seleccionado el paquete BC4J correcto.
   * No introduce un valor en el campo Objeto vista se extiende a menos que esté de subclases deliberadamente uno de sus propios objetos de vista.
   * Seleccione Siguiente para continuar.
6. En la página de objetos de entidad (Paso 2 de 6):
   * En la lista Disponible, seleccione los objetos de entidad que desee incluir en el objeto de vista y les traslado a la lista Seleccionado.
   * Indican si los objetos de entidad son de sólo lectura, y si deben ser tratados como una referencia (véase la documentación JDeveloper para obtener información adicional acerca de esta página).
   * Seleccione Siguiente para continuar.
7. En la página Atributos (Paso 3 de 6) seleccionar los atributos que desea incluir en la lista disponible y les traslado a la lista Seleccionado. Seleccione Siguiente para continuar.
8. En la página de configuración de atributo (paso 4 de 6), verificar que la información predeterminada es la correcta. Seleccione Siguiente para continuar.
9. En la página de consulta (Paso 5 de 6):
   * Verifique que la consulta BC4J generada para usted sea correcta. Si no es así, seleccione la casilla de verificación Modo experto para hacer la consulta editable.

**Nota:** Para expertos subventanas modo, no trate de establecer un valor en la instrucción SELECT para un atributo EO.Por ejemplo, no asuma la columna de la *bandera* se basa en un atributo EO ya que esto da lugar a un fallo de bloqueo debido BC4J trata de comparar este valor con el valor de base de datos original y se queja de que son diferentes. Ver [Java objetos de entidad](https://translate.googleusercontent.com/bus/bus_java.htm) para los enfoques válidos para establecer los valores por defecto de los atributos.

SELECCIONE x pk1, y pk2, estado z, indicador 'Y', ....

* + Seleccione la prueba ... botón para verificar que su consulta es correcta.
  + Seleccione Siguiente para continuar.

1. En la página de Java (Paso 6 de 6):
   * Compruebe la opción de generar una clase de vista de primera fila (incluyendo métodos de acceso).
   * Compruebe la opción de generar una Clase de visualización de objetos sólo si usted anticipa que escribir ningún código para su objeto vista (siempre se puede generar esta clase más adelante, si es necesario, o eliminarlo si es que generamos ahora y encuentre más tarde que no haya una necesidad para ello).
   * Se extiende seleccione el botón ... para asegurarse de que está de subclases las clases OA Framework oracle.apps.fnd.framework.server.OAViewObjectImpl y oracle.apps.fnd.framework.server.OAViewRowImpl según corresponda. Si necesita corregir los valores por defecto, seleccione Examinar ... para abrir la ventana Buscar superclase.
2. Seleccione Finalizar para guardar el nuevo objeto de vista.

Una vez que haya creado un objeto de vista basado en objeto de entidad, debe editarlo para ajustarlo propiedades de pasivación como se describió anteriormente. Por ejemplo, para un objeto de vista utiliza para actualizar objetos de entidad, la opción de pasivación se puede consultar en la página de sintonización. Véase el Capítulo 6 [OA Marco Estado Persistencia Modelo (pasivación)](https://translate.googleusercontent.com/adv/adv_passive.htm)para obtener información adicional.

**Objetos de vista híbrida (objetos de vista en modo experto)**

También puede crear objetos de visualización basados ​​en objetos de entidad e incluir Atributos de SQL. En este caso, cree el objeto de vista como se describe en el caso de objeto de entidad anterior, con algunas modificaciones pequeñas:

* En la página Atributos, seleccione el botón Nuevo para crear atributos para los valores de los objetos que no son de entidad que desea consultar directamente.
* En la página de consulta, seleccione el modo experto para editar el SQL generado, según sea necesario para recuperar estos valores "calculados".
* En la página de asignaciones de atributos (se muestra sólo si tiene atributos basados en SQL), asegúrese de que las columnas de la consulta y Atributos de vista partido.

**Llaves primarias**

Por la [Codificación Marco Modelo OA estándar M39](https://translate.googleusercontent.com/cs/cs_mdlgeneral.htm#M39) , casi todos los objetos de vista requieren llaves principales. Puede especificar las claves primarias de forma declarativa en la definición de atributos, o puede establecer mediante programación llamando setKeyAttributeDefs () en OAViewObjectImpl.

**Control programático**

**Manejo de consultas**

Cada objeto de vista implementa su propia búsqueda, y si es necesario, debe ser capaz de traducir los parámetros de entrada para unir las variables y donde frases cláusula. Como práctica de codificación general, todos los métodos que realizan este trabajo se deben nombrar initQuery () o alguna variante descriptivo como initNewEmployeesQuery () si necesita varios métodos "init".

**Nota:** También se debe utilizar "al estilo de Oracle" vinculante (FOO> =: 1) la unión en lugar de ANSI-estilo (FOO> =?). Aunque el código es un poco más complejo, el equipo de OA Framework planea desproteger los enlaces de estilo ANSI en la versión de Fusion.

El siguiente ejemplo muestra cómo modificar una DONDE criterios de búsqueda cláusula y se unen sobre la base de los valores pasados al initQuery método.

// Inicializa y ejecuta la consulta   
 public void initQuery (String name, String onHold, String number)   
 {   
  
 StringBuffer whereClause = new StringBuffer (100);   
 Parámetros del vector = nuevo Vector (3);   
 int cláusulaCount = 0;   
 int bindCount = 0;   
  
 setWhereClauseParams (null); // Siempre restablecer

if ((nombre! = nulo) && (! ("". equals (nombre.trim ()))))

{

whereClause.append ("NAME like:");

whereClause.append (++ bindCount);

parameters.addElement (nombre + "%");

cláusulaCount ++;

}

if ((number! = null) && (! ("" Igual (number.trim ()))))

{

Número supplierId = null;

// SUPPLIER\_ID es un NÚMERO; los tipos de datos siempre deben

// match, y el parámetro pasado a este método es un

// Cuerda.

tratar

{

supplierId = new Number (número);

}

catch (Exception e) {}

if (clauseCount> 0)

{

whereClause.append ("Y");

}

whereClause.append ("SUPPLIER\_ID =:");

whereClause.append (++ bindCount);

parameters.addElement (supplierId);

cláusulaCount ++;

}

if ((onHold! = null) && (! ("" Igual (onHold.trim ()))))

{

if (clauseCount> 0)

{

whereClause.append ("Y");

}

whereClause.append ("ON\_HOLD\_FLAG =:");

whereClause.append (++ bindCount);

parameters.addElement ("Y");

cláusulaCount ++;

}

setWhereClause (whereClause.toString ());

if (bindCount> 0)

{

Object [] params = new Object [bindCount];

// el copyInto () es 1.1.8 obediente que, a partir del 4/02/03, es requerido por ARU

parameters.copyInto (params);

setWhereClauseParams (params);

}

ejecutar la solicitud();

} // finalizar initQuery ()

**Lógica de negocios**

Los objetos de vista no son un hogar apropiado para lógica de negocios; no debería escribir reglas de validación en sus objetos de vista o ver filas.

**Ver filas**

Aunque siempre debes crear una fila de vista como se mencionó anteriormente, en su mayor parte, no será necesario escribir la vista de código fila. Ver código de fila es útil en los casos en los que desea calcular un valor de atributo transitoria, por ejemplo, pero que no puede o no desea incluir la lógica de la consulta (tal vez el costo de rendimiento es muy alto). También puede utilizar la vista de código seguidos para realizar validaciones de atributos simples transitorios utilizados en la interfaz de usuario, o llamar a los métodos del objeto entidad personalizada (véase el ejemplo "Aceptar" en la sección Módulo de Aplicación a continuación para obtener información adicional).

Personalizado métodos vista fila no se puede acceder directamente en el cliente. El cliente debe invocar por primera vez un método en el módulo de aplicación, que delega en el objeto vista. El objeto de la vista proporciona acceso a la fila vista. Por otra parte, al darse cuenta de la ventaja de rendimiento de tener la clase de vista fila, siempre llama a los setters / captadores generados (por ejemplo, setSupplier () ) en la fila si necesita acceder a los valores o establecer mediante programación. Esto se debe a que es mucho más rápido que llamar al genérico setAttribute ( "< *AttributeName* >") y getAttribute ( "< *AttributeName* >") . Por ejemplo, la [Entidad Eliminar objeto Ejemplo](https://translate.googleusercontent.com/translate_f#eo_delete) en la sección Módulo de Aplicación de abajo muestra cómo recuperar correctamente un valor de atributo de vista de primera fila.

**Ver enlaces**

Como se describió anteriormente, una asociación define una relación entre dos objetos de entidad. Del mismo modo, un acoplamiento de la visión define una relación entre dos objetos de vista que BC4J utiliza para consultar automáticamente y coordinar el objeto de vista de destino basado en la vista de origen actual objeto.

Consulte los enlaces pueden estar basadas en una asociación o declarativa se unen relación entre dos objetos de visualización. Por ejemplo, supongamos que dos tablas tienen una relación maestro-detalle en base a una clave externa. Los correspondientes objetos de entidad están relacionadas a través de una asociación, y ver objetos en función de los objetos de entidad pueden estar relacionados por un acoplamiento de la visión sobre la base de la asociación.

Aunque vínculos de vista puede ser muy conveniente, usarlos con moderación en las páginas de aplicaciones web, ya que ambos guardan en caché los registros maestra y detalle que el usuario se desplaza de una fila maestro al lado - y esto puede ser costoso. Utilice vínculos de vista solamente en los siguientes casos:

* Cuando los frijoles específicos (como el HGrid) les exigen.
* Cuando se tiene actualizables maestro / detalle objetos de vista (en el mismo o diferentes páginas) cuya entidad subyacente objetos están relacionados con el uso de la composición, *debe definir un vínculo de vista entre ellos* (se discuten más a fondo en el capítulo 5).
* Cuando se tiene un maestro y detalle objeto de vista de sólo lectura en la misma página, y navegar a una fila maestro debe causar a los niños para consultar automáticamente.

**La aplicación declarativa**

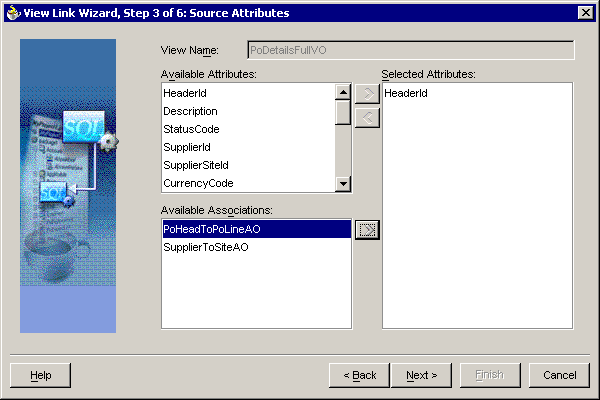
Para información adicional acerca de Ver las propiedades del Asistente Enlace no específicamente descritas aquí, consulte la documentación de JDeveloper.

**Nota:** Puede acceder a la ayuda específica al contexto, mientras que en cualquiera de los asistentes BC4J mediante la selección de la F1 en su teclado.

Para crear un nuevo acoplamiento de la visión en un paquete Business Components (BC4J):

1. En el JDeveloper Navigator, seleccione el paquete BC4J donde desea crear su vista enlace.
2. En el menú principal, seleccione Archivo> Nuevo para abrir el nuevo objeto de la Galería .
3. En el Categorías de árbol, expanda la capa de negocio nodo y seleccione Componentes de Negocio (BC4J) .
4. En el Artículos lista, seleccione Ver enlace para abrir el Asistente Enlace Vista . Tenga en cuenta que también puede hacer clic en el paquete de BC4J y seleccione Nueva vista Enlace para navegar directamente al Asistente Enlace Vista .
5. En el Nombre de página (paso 1 de 6):
   * Especificar el enlace de vista Nombre de acuerdo con las [OA Marco de archivo / directorio / estructura de paquetes](https://translate.googleusercontent.com/cs/cs_filedir.htm)estándares.
   * Compruebe que ha seleccionado el paquete BC4J derecha.
   * No introduce un valor en el Ver enlace amplía el campo a menos que esté de subclases deliberadamente uno de sus propios objetos de vista.
   * Seleccione Siguiente para continuar.
6. En el objetos de vista de página (Paso 2 de 6), seleccione la Fuente y Destino objetos de vista. Seleccione Siguiente para continuar.
7. En la Fuente Atributos página (paso 3 de 6), especifica el atributo o asociación objeto del objeto de origen se unen (si está disponible) como se muestra en la Figura 8. Seleccione Siguiente para proceder.

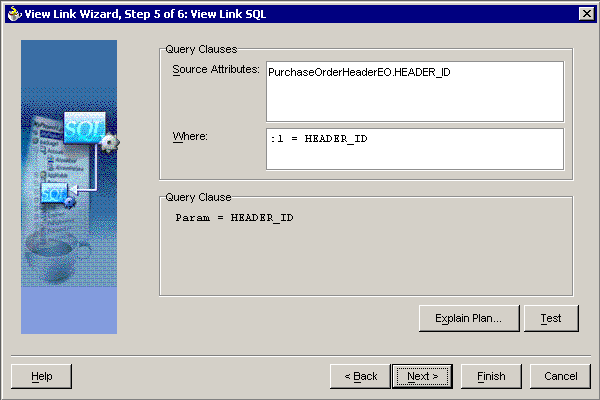
Figura 8: Vista Asistente Enlace que muestra el uso de una asociación para obtener el objeto vista de origen unirse atributo



1. En el Destino atributos de página (Paso 4 de 6), especificar el atributo unirse o asociación del objeto de destino. Seleccione Siguiente para continuar.
2. En la Ver enlace SQL página (Paso 5 de 6), revisar el DONDE cláusula que BC4J va a crear para usted para asegurarse de que es correcta.

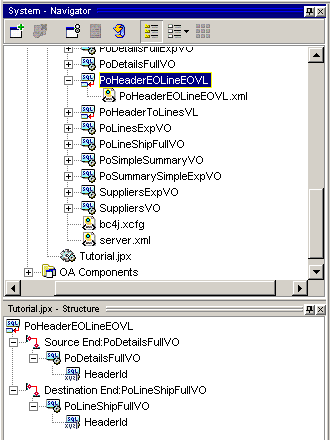
**Consejo:** Si no hay claves primarias especificado en el origen y ver su destino objetos, BC4J no puede crear correctamente una DONDE cláusula. Si estos campos están desactivados, revisar sus definiciones de vista del objeto.

Figura 8: Vista Asistente de Enlace que muestra una cláusula WHERE generada



1. En el Propiedades del vínculo Ver página (Paso 6 de 6,) especificar la cardinalidad entre los objetos de vista e indique si desea generar descriptores de acceso desde el origen al destino y viceversa.
2. Seleccione Finalizar para crear la vista enlace. BC4J creará un archivo de definición XML como se muestra en la Figura 9. Tenga en cuenta que puede ver rápidamente la relación subyacente, seleccionando el enlace de vista en el navegador del sistema.

Figura 9: JDeveloper Sistema Navigator y panel Estructura vista de un acoplamiento de la visión seleccionado



**Control programático**

Desde vínculos de vista no tienen aplicación, no escribir ningún código para ellos. En el capítulo 5, se discute cómo acceder a un objeto de vista usando un enlace de vista.

Sin embargo, puede crear vínculos de vista dinámicamente mediante el uso de la oracle.jbo.ApplicationModule createViewLinkBetweenViewObjects API. Consulte la JavaDoc correspondiente para un ejemplo de cómo utilizar este método.

**Nota:** Tanto el capitán y ver detalles objetos que participan en un enlace de vista de programación creado deben pertenecer a la misma instancia de módulo de aplicación.

**Módulos de aplicación**

Esto introduce los conceptos básicos de la creación de módulos de aplicación / uso. Ver [módulos de aplicación en detalle](https://translate.googleusercontent.com/bus/bus_am.htm) en el Capítulo 5 para obtener información adicional acerca de cómo trabajar con estos objetos.

**Usos módulo de aplicación**

La siguiente es una lista de las distintas funciones que los módulos de aplicación pueden desempeñar en su aplicación:

* Módulo de interfaz de usuario de aplicación de la raíz - Establece el contexto de transacción para una o varias páginas de interfaz de usuario relacionados. Cada página tiene un módulo de aplicación de la raíz que incluye cualquier objeto de vista y módulos de aplicación anidados utilizados por la página.
* Módulo de aplicación de interfaz de usuario región compartida - Cualquier región de la interfaz de usuario con el *creado para su uso* en múltiples páginas debe tener su propio módulo de aplicación que contiene. Cuando se utiliza esta región en una página, la OA Marco anida automáticamente su módulo de aplicación bajo módulo de aplicación radicular de la página. Ver [La implementación de la vista](https://translate.googleusercontent.com/build_view.htm#region) para obtener información adicional sobre la creación de regiones compartidas.
* Lista de Valores de interfaz de usuario Módulo de Aplicación - Se trata de un caso especial de la función anterior. Al crear la lista de objetos de vista valores (LOV), se añaden estos componentes a un módulo de aplicación dedicada a la tarea de agregación de listas de valores para un paquete dado.
* Módulo de Aplicación de validación - A *validación* agregados módulos de la aplicación relacionadas Ver objetos creados con el propósito de realizar validaciones de SQL ligeros. Objetos de entidad y los expertos hacen uso de los módulos de aplicación de validación, que no tienen nada que ver con la interfaz de usuario. Ver los [objetos de entidad, Entidad expertos, módulos y la aplicación 'validación' 'validación' objetos de vista](https://translate.googleusercontent.com/translate_f#eo_ee) de los siguientes temas para obtener información adicional.

**La aplicación declarativa**

Para información adicional acerca de las propiedades del Asistente Módulo de Aplicación no específicamente descritos aquí, véase la documentación JDeveloper.

**Nota:** Puede acceder a la ayuda específica al contexto, mientras que en cualquiera de los asistentes BC4J mediante la selección de la F1 en su teclado.

**Creación de nuevos módulos de aplicación**

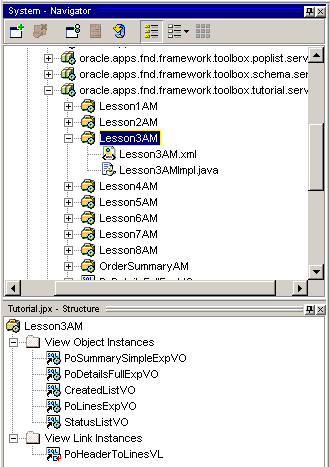
**Nota:** Crear un módulo de aplicación compartida que subclases oracle.apps.fnd.framework.server.OAApplicationModuleImpl , que luego subclase para crear comportamientos más específicos.

Para crear un nuevo módulo de aplicación en un paquete Business Components (BC4J):

1. En el JDeveloper Navigator, seleccione el paquete BC4J donde desea crear su módulo de aplicación.
2. En el menú principal, seleccione Archivo> Nuevo para abrir el nuevo objeto de la Galería .
3. En el Categorías de árbol, expanda la capa de negocio nodo y seleccione Componentes de Negocio (BC4J) .
4. En el Artículos lista, seleccione Módulo de Aplicación para abrir el Asistente para módulo de aplicación . Tenga en cuenta que también puede hacer clic en el paquete de BC4J y seleccione Nuevo módulo de aplicación para navegar directamente al Asistente Módulo de Aplicación .
5. En el Nombre de la página (paso 1 de 5):
   * Especificar del módulo de aplicación Nombre de acuerdo con las [OA Marco de archivo / directorio / estructura de paquetes](https://translate.googleusercontent.com/cs/cs_filedir.htm) estándares.
   * Compruebe que ha seleccionado el paquete BC4J derecha.
   * No introduce un valor en el módulo de aplicación se extiende el campo a menos que esté de subclases deliberadamente uno de sus propios módulos de aplicación.
   * Seleccione Siguiente hasta llegar al Paso 4.
6. En el Java página (Paso 4 de 5), anule la selección de archivos Java (s) Generar casilla de verificación sólo si está seguro de que no va a escribir ningún código para su módulo de aplicación (siempre se puede eliminar la clase más adelante, si usted encuentra que que no lo necesita, por lo que es probablemente el mejor para generar simplemente que en este punto menos que esté creando un envase simple para los objetos de vista LOV). Si desea generar una clase de implementación para su módulo de aplicación, seleccione el Extiende ... botón para verificar que se le subclassing oracle.apps.fnd.framework.server.OAApplicationModuleImpl .
7. Seleccione Finalizar para crear el módulo de aplicación. BC4J creará un archivo XML de definición e implementación, como se muestra en la Figura 10.

**Nota:** Puede ver rápidamente el contenido subyacente seleccionando el módulo de aplicación en el navegador del sistema.

Figura 10: JDeveloper Sistema Navigator y panel Estructura vista de un módulo de aplicación seleccionado



En este punto, no está terminado ya con el proceso de creación. Para continuar, modificar el módulo de aplicación de la siguiente manera:

**Importante:** No *, no* fijar el módulo de aplicación de Retención Nivel de MANAGE\_STATE (pasos 2 y 3 a continuación) si aún no está listo para aplicar plenamente y certificar su página para el apoyo de pasivación. Ver [Modelo de la codificación estándar M8](https://translate.googleusercontent.com/cs/cs_mdlgeneral.htm#M8) para más detalles.

1. En el JDeveloper Navigator, seleccione el módulo de aplicación que acaba de crear, haga clic derecho y seleccione Editar < *appmodule\_name* > ... .
2. En el Asistente para módulo de aplicación , seleccione Propiedades .
3. En el Propiedades de la página, crear una propiedad de pasivación como se describe en [la OA Marco Estado Persistencia Modelo (pasivación)](https://translate.googleusercontent.com/adv/adv_passive.htm) . Por ejemplo, la configuración más común de pasivación módulo de aplicación consiste en la creación del módulo de aplicación Retención Nivel para MANAGE\_STATE . Para hacer esto:
   1. Tipo RETENTION\_LEVEL en la propiedad Nombre de campo.
   2. Tipo MANAGE\_STATE en la propiedad Valor de campo.
   3. Seleccione Añadir para crear la propiedad.
4. Por último, mientras está todavía en el asistente módulo de aplicación, vaya a la sintonización página. Verificar que el comportamiento de tiempo de ejecución de instancias Personalizar casilla de verificación está marcada, y la carga lenta se selecciona el botón de radio (en cuenta que también debe revisar [módulos de aplicación en detalle](https://translate.googleusercontent.com/bus/bus_am.htm) para una descripción detallada de la función de carga diferida y varias consideraciones de casos de uso).
5. Seleccione Aceptar para guardar los cambios.

**Generación de Interfaces módulo de aplicación**

A generar un módulo de interfaz de aplicación para que pueda invocar métodos escrito directamente (con control de tiempo de compilación) en lugar de llamar InvokeMethod () , primero debe crear los métodos que desee exponer al cliente. Entonces:

1. En el JDeveloper Navigator, seleccione el módulo de aplicación que acaba de crear, haga clic derecho y seleccione Editar < *appmodule\_name* > ... .
2. En el Asistente Módulo de Aplicación , seleccione Métodos de cliente.
3. Seleccionar los métodos que desea ser capaz de invocar de forma remota en el Disponible lista y traslado a la seleccionadalista.
4. Seleccione Aceptar para crear la interfaz.

JDeveloper crea automáticamente una interfaz en el paquete correcto y con el nombre correcto de acuerdo con las [Normas Marco de la OA de archivos](https://translate.googleusercontent.com/cs/cs_filedir.htm) .

**Añadiendo Ver objetos**

**Consejo:** Cuando se crea un objeto de vista para un fin determinado, inmediatamente agregarlo al módulo de aplicación adecuada para que no se olvide de hacerlo más adelante.

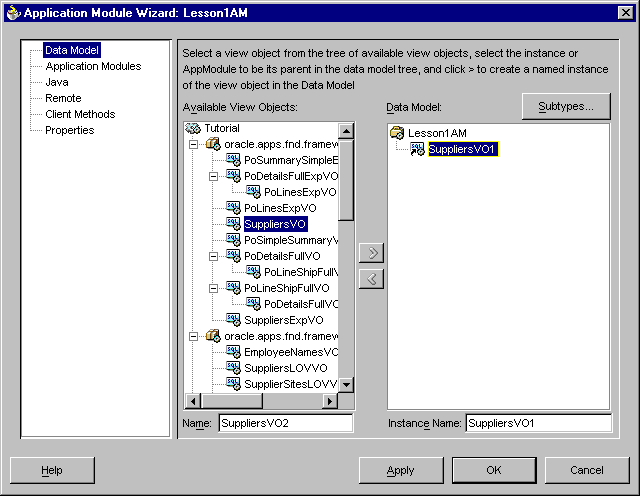
Todos los objetos de vista se utilizan en el contexto de un módulo de aplicación dada.

Ver objetos se instancian en un "como sea necesario" (en versiones anteriores, BC4J instancia todos los objetos de vista asociados a un módulo de aplicación determinada cuando se creó el módulo de aplicación). Por ejemplo, si se escribe código para encontrar una visión objeto específico para que pueda consultarlo, o una página de interfaz de usuario hace que con los objetos unidos a ver atributos de los objetos, BC4J automáticamente crea la instancia de los objetos de vista necesarios. Si un objeto de vista que está asociado a un módulo de aplicación no es necesaria en tiempo de ejecución, no se crea una instancia.

Para crear esta relación de forma declarativa de un objeto y la aplicación dada módulo de vista de:

1. En el JDeveloper Navigator, seleccione el módulo de aplicación que acaba de crear, haga clic derecho y seleccione Editar < *appmodule\_name* > ... .
2. En el Asistente para módulo de aplicación , seleccione Modelo de Datos .
3. En el modelo de datos de la página, seleccione los objetos de vista que desee incluir en el Disponible Subventanas lista y traslado al modelo de datos de lista.
4. Opcionalmente cambiar la vista predeterminada de nombre de instancia . Un objeto de vista individual se puede agregar al mismo módulo de aplicación varias veces (por ejemplo, se podría realizar la misma consulta en múltiples regiones dentro de la misma interfaz de usuario de la tarea / módulo). Cada instancia de objeto de vista tiene un identificador único; este identificador único es el nombre de la instancia . Cuando se agrega un objeto de vista de un módulo de aplicación, BC4J crea el nombre de instancia predeterminado añadiendo un entero a nombre del objeto vista como se muestra en la Figura 11. Para modificar este valor, sólo tiene que seleccionar la vista de objetos en el modelo de datos de la lista y hacer los cambios en el campo actualizable por debajo de la lista.

Figura 11: Modelo de la página Módulo de Aplicación Asistente de Datos que muestra una vista por defecto Nombre de instancia

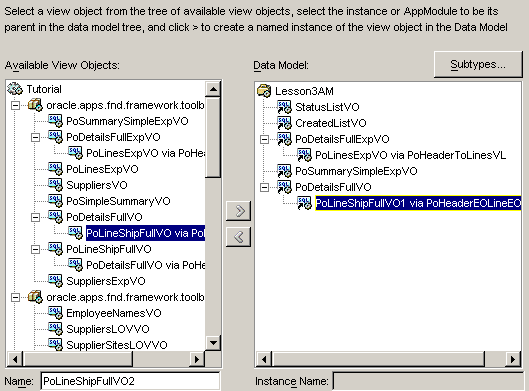


**Nota:** Para agregar un objeto de vista de detalle (se accede a través de un enlace de vista) para el módulo de aplicación, siga estos pasos en el Editor de Módulo de aplicación de diálogo. Usted debe cumplir con estas instrucciones para acceder adecuadamente el objeto de vista de detalle; que no es suficiente simplemente agregar el objeto de la vista de detalle de igual a igual con el objeto vista maestra.

1. Seleccione la vista principal objeto en el modelo de datos de vista
2. Seleccione la vista de detalle en el objeto Disponible Subventanas vista y servicio de transporte al modelo de datos de vista

Si ha añadido el objeto de vista de detalle correctamente, aparecerá como se muestra en la Figura 12.

Figura 12: página Modelo Módulo de Aplicación Asistente de Datos que muestra un objeto de la vista de detalle añadió a su maestro a través de un acoplamiento de la visión



**Adición de Nested módulos de aplicación**

Puede módulos de aplicación nido a cualquier nivel arbitrario.

Los módulos de aplicación anidadas se instancian sobre una base "según sea necesario" (en las versiones anteriores, BC4J instancia todos los módulos de la aplicación anidados cuando se creó el módulo de aplicación que contiene). Por ejemplo, si usted hace una findApplicationModule, BC4J será una instancia del objeto. Si un módulo de aplicación anidada no se accede, no se crea.

Para agregar un módulo de aplicación anidada a su módulo de aplicación:

1. En el JDeveloper Navigator, seleccione el módulo de aplicación que acaba de crear, haga clic derecho y seleccione Editar < *appmodule\_name* > ... .
2. En el Asistente para módulo de aplicación , seleccione módulos de aplicación .
3. En el módulos de aplicación página, seleccionar el módulo (s) de aplicaciones que desea incluir en el Disponible lista y traslado a la seleccionada de datos de lista.
4. Opcionalmente cambiar el módulo de aplicación por defecto Nombre de instancia como se describe para objetos de vista anteriormente.

**Control programático**

NO hacer código de lógica de negocio, tales como actividades de validación, en los módulos de aplicación; este debe ser codificada en la entidad subyacente objetos en su lugar. El módulo de aplicación es un lugar apropiado para la lógica que:

* Proporciona acceso a cualquier objeto BC4J asociados. Por ejemplo, en la [Implementación del controlador](https://translate.googleusercontent.com/build_control.htm) , verá que los controladores no deben tener acceso a objetos de vista directamente cuando necesitan para ejecutar una consulta. En su lugar, deben llamar a los métodos de módulo de aplicación de la página para pedir una consulta en particular a ejecutar.
* Realiza múltiples acciones del lado del servidor, o se extiende por ver varios objetos como el resultado de un solo evento o método de invocación. Por ejemplo, el código que copia todas las filas de visualización de objetos de A a B de visualización de objetos pertenece a esta clase.
* Devuelve los valores del lado del servidor al cliente que no se puede acceder desde un OAPageContext. Si, por ejemplo, la página necesita un valor específico del servidor para determinar si una región se dictarán o, el módulo de aplicación debe proporcionar esta información un artículo debe ser de sólo lectura.
* Las llamadas rutinas especiales de PL / SQL.

**Consejo:** Si se utilizan las rutinas de PL / SQL para la validación y procesamiento de filas individuales (o un conjunto de filas), entonces debería usar PL / SQL entidad basada en objetos en su lugar. Véase el Capítulo 5 para obtener información adicional sobre el uso de objetos / entidad SQL PL.

**Naming método**

Alguna métodos de módulos de programa que apoya directamente a la interfaz de usuario deben ser nombrados por los correspondientes de interfaz de usuario "eventos". Por ejemplo, si el usuario pulsa una Crear botón, el método del módulo de aplicación debe ser nombrado "crear" y se muestra en los siguientes ejemplos.

**Nota:** las invocaciones de controlador correspondiente de todos los ejemplos siguientes se incluyen en la [Aplicación de la Controller](https://translate.googleusercontent.com/build_control.htm) .

**Entidad Objeto Crear Ejemplo**

El siguiente ejemplo ilustra un método de módulo de aplicación que crea e inserta una fila en la SuppliersVO objeto de vista. Este punto de vista objeto en particular se basa en la SupplierEOImpl objeto de entidad, por lo que esta instancia BC4J detrás de las escenas cuando se crea la fila.

createSupplier public void ()

{

OAViewObject vo = getSuppliersVO ();

Fila fila = vo.createRow ();

vo.insertRow ();

// Tal como se especifica en la OA Modelo Marco de Estándares de Codificación,   
// establecer el nuevo estado fila para STATUS\_INITIALIZED.

row.setNewRowState (Row.STATUS\_INITIALIZED);

}

**Ejemplos de consultas de visualización de objetos**

Esto muestra un método de módulo de aplicación que consulta la SuppliersVO vista de objetos utilizando criterios de búsqueda pasa del cliente.

consulta public void (String supplierName, Cadena onHoldFlag, Cadena supplierNumber)

{

SuppliersExpVOImpl vo = getSuppliersExpVO ();

si (vo == null)

{

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

arrojar nueva OAException ( "ICX", "FWK\_TBX\_OBJECT\_NOT\_FOUND", tokens);

}

vo.initQuery (supplierName, onHoldFlag, supplierNumber);

} // fin de consulta ()

Este ejemplo ilustra una consulta que inicializa una página cuando el usuario navega a la misma. Tenga en cuenta que el navegador Volver cheque botón para asegurarse de que una consulta no se ejecuta sin necesidad. Véase el Capítulo 6 [Apoyar el botón atrás del navegador](https://translate.googleusercontent.com/adv/adv_backbutton.htm) para obtener información adicional).

public void init (estado de cadena)

{

PoSimpleSummaryVOImpl vo = getPoSimpleSummaryVO ();

si (vo == null)

{

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

arrojar nueva OAException ( "ICX", "FWK\_TBX\_OBJECT\_NOT\_FOUND", tokens);

}

// Sigue Volver estándar Botón de no realizar una consulta ciega y sin

// comprobación para ver si esto es necesario.

if (! vo.isPreparedForExecution ())

{

vo.initQuery (status);

}

} // fin init ()

**Entidad Eliminar objeto Ejemplo**

Esto ilustra cómo buscar una fila objeto vista establecido para un único objeto seleccionado para que el objeto entidad se puede eliminar.

/ \*\*

\* Elimina una orden de compra de la PoSimpleSummaryVO utilizando el

\* Parámetro poHeaderId.

\* /

public void eliminar (String poHeaderId)

{

// En primer lugar, tenemos que encontrar la orden de compra seleccionada en nuestra VO.

// Cuando lo encontramos, que llamamos remove () en la fila que a su vez

// eliminar las llamadas en el objeto PurchaseOrderHeaderEOImpl asociado.

int poToDelete = Integer.parseInt (poHeaderId);

OAViewObject vo = getPoSimpleSummaryVO ();

fila PoSimpleSummaryVORowImpl = null;

// Esto nos dice que el número de filas que se han exagerado en el

// conjunto de filas, y no va a tirar de filas adicionales en como algunos de los

// obtener otros "contar" métodos.

int fetchedRowCount = vo.getFetchedRowCount ();

// Usamos un iterador por separado - a pesar de que podríamos paso a través de la

// filas sin ella - porque no queremos afectar a la moneda fila.

// Tenga en cuenta que también existen métodos de conveniencia para encontrar las filas coincidentes

// en un objeto de vista (ver Javadoc).

RowSetIterator deleteIter = vo.createRowSetIterator ( "deleteIter");

si (fetchedRowCount> 0)

{

deleteIter.setRangeStart (0);

deleteIter.setRangeSize (fetchedRowCount);

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

{

fila = (PoSimpleSummaryVORowImpl) deleteIter.getRowAtRangeIndex (i);

// Por motivos de rendimiento, generamos ViewRowImpls para todos

// Ver objetos. Cuando necesitamos para obtener un valor de atributo,

// usamos los descriptores de acceso con nombre en lugar de una búsqueda de cadena genérico.

// Número primaryKey = (Número) row.getAttribute ( "HeaderId");

Número primaryKey = row.getHeaderId ();

si (primaryKey.compareTo (poToDelete) == 0)

{

row.remove ();

GetTransaction () commit ().;

descanso; // único posible fila seleccionada en este caso

}

}

}

deleteIter.closeRowSetIterator ();

} // fin deletePurchaseOrder ()

**Ejemplo acción personalizada ( "Aprobar")**

Esto ilustra cómo buscar un objeto de vista de conjunto de filas de uno o varios objetos seleccionados para llamar a un evento entidad objeto personalizado.

/ \*\*

\* Pasos a través de la POSimpleSummaryVO para buscar las filas seleccionadas. por

\* Cada fila seleccionada, esta llama al método aprobar () en el

\* Clase PurchaseOrderHeaderEOImpl.

\* /

approvePurchaseOrders public void ()

{

// Para llamar a un método personalizado en un objeto de entidad se debe añadir un envoltorio

// en la clase \* RowImpl del VO (véase

// oracle.apps.fnd.framework.toolbox.schema.server.PoSimpleSumaryVORowImpl).

OAViewObject vo = getPoSimpleSummaryVO ();

fila PoSimpleSummaryVORowImpl = null;

int coincide = 0;

// Esto nos dice que el número de filas que se han exagerado en el

// conjunto de filas, y no va a tirar de filas adicionales en como algunos de los

// obtener otros "contar" métodos.

// Tenga en cuenta que también existen métodos de conveniencia para encontrar las filas coincidentes

// en un objeto de vista (ver Javadoc).

int fetchedRowCount = vo.getFetchedRowCount ();

// Usamos un iterador por separado - a pesar de que podríamos paso a través de la

// filas sin ella - porque no queremos afectar a la moneda fila.

RowSetIterator approveIter = vo.createRowSetIterator ( "approveIter");

si (fetchedRowCount> 0)

{

approveIter.setRangeStart (0);

approveIter.setRangeSize (fetchedRowCount);

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

{

// Por cada fila con una casilla de verificación seleccionada, queremos llamar

// el () envoltorio aprobar en el cual POSimpleSummaryVORowImpl

// a su vez llama al método aprobar) en el PurchaseOrderHeaderEOImpl.

fila = (PoSimpleSummaryVORowImpl) approveIter.getRowAtRangeIndex (i);

// Por motivos de rendimiento, generamos ViewRowImpls para todos

// Ver objetos. Cuando necesitamos para obtener un valor de atributo,

// usamos los descriptores de acceso con nombre en lugar de una búsqueda de cadena genérico.

// Cadena selectFlag = (String) row.getAttribute ( "SelectFlag");

Cadena selectFlag = row.getSelectFlag ();

si ( "Y" es igual a (selectFlag))

{

row.approve ();

partidos ++;

}

}

}

approveIter.closeRowSetIterator ();

// Si el usuario no ha hecho ningún seleccionar filas, mostrará un mensaje de error.

si (partidos> 0)

{

GetTransaction () commit ().;

}

más

{

arrojar nueva OAException ( "ICX", "FWK\_TBX\_T\_SELECT\_FOR\_APPROVE");

}

} // fin de aprobar ()

**Commit Ejemplo**

/ \*\*

\* Proporciona un "commit" envoltorio para que el código del controlador de interfaz de usuario no necesita

\* Obtener un identificador para la propia transacción que es una violación de la

\* Normas de separación de nivel de cliente / severas.

\* /

public void apply ()

{

GetTransaction () commit ().;

} // aplica end ()

**Comprobación de los módulos de aplicación**

Una vez que termine de añadir sus objetos de vista a sus módulos de aplicación, puede utilizar el Explorador de Componentes de Negocio ( BC4J Tester) para ejecutar sus objetos de vista antes de construir un marco OA interfaz de usuario para ellos, ni escribir código para apoyar sus objetos BC4J. Por ejemplo, se puede consultar objetos de vista (incluyendo la capacidad de navegar a través de las filas maestros para consultar los niños vinculados con un acoplamiento de la visión).

Ver [Aplicaciones Prueba de OA Framework](https://translate.googleusercontent.com/test/test_test.htm#bc4j_test) para obtener instrucciones sobre cómo activar esta utilidad.

**Entidad objetos, entidad Expertos, módulos de aplicación 'validación' y 'validación' Ver objetos**

Para obtener información detallada sobre el uso de objetos de entidad, los expertos de la entidad, los módulos de aplicación y validación de vista de la validación objetos entre sí, véase el Capítulo 5. En esta sección se presenta simplemente los objetos y las funciones que desempeñan en una aplicación.

**Validación Subventanas**

Cuando se implementa la lógica de negocio en sus objetos de entidad, se encontrará frecuentemente que es necesario ejecutar sentencias SQL simples, y no sólo para fines de validación puros. Por ejemplo, un jefe de la orden de compra tiene muchas líneas. Cada línea se le asigna un número de línea única. Este número se define como *el número de línea de corriente máxima para toda la orden de compra + 1* . En tiempo de ejecución, lo que necesitamos para consultar la base de datos para averiguar lo que el número de línea máxima es de cabecera de la orden de compra dada:

SELECT MAX (LINE\_NUMBER) a partir de FWK\_TBX\_PO\_LINES DONDE HEADER\_ID =: 1;

Siempre que necesite para ejecutar SQL como esto, se puede crear un objeto de vista dinámica a partir de una sentencia SQL, o se puede predefinir un objeto de vista declarativo para él. Dicho esto, [Codificación OA Marco Modelo Normas](https://translate.googleusercontent.com/cs/cs_mdlgeneral.htm) requieren el uso de la estrategia declarativa en este caso, ya que es más performante: un objeto de vista se almacena en caché en su respectivo módulo de aplicación, lo que permite que el código objeto de entidad para reutilizarlo (y el JDBC subyacente preparada declaración) simplemente revinculación y volver a ejecutar la consulta. Se trata de una importante mejora en el rendimiento, pues las rutinas de validación se llaman repetidamente.

**Implementación**

Desde el punto de vista de implementación, *validación de* objetos de vista no son diferentes de *regulares* subventanas; que se diferencian sólo por el caso de uso. Sin embargo, siempre desactivar pasivación para estos objetos de vista, que nunca debería haber estado asociados y siempre debe ser recreables. Ver [Codificación OA Marco de Normas Modelo](https://translate.googleusercontent.com/cs/cs_mdlgeneral.htm) .

**Módulos de Aplicación de validación (VAM)**

subventanas predefinidos deben ser asignados a un módulo de aplicación para que puedan tener acceso en tiempo de ejecución. En otras palabras, objetos de vista no existen fuera del contexto de un módulo de aplicación.

Desde objetos de entidad (y sus asociados en la vista de validación objetos) pueden ser compartidos por varios clientes de interfaz de usuario (y los módulos de aplicación de la raíz deben ser considerados-UI específica), no es apropiado utilizar el módulo de aplicación de la raíz de una página en particular para mantener su validación Ver objetos. En cambio, para agrupar estos objetos vista utilidad en unidades significativas, reutilizables, crear una *validación* módulo de aplicación por cada objeto de negocio para sostenerlos. Un objeto de negocio se define la entidad objeto de nivel superior en una composición, o un solo objeto de entidad si está solo. Por ejemplo, la orden de compra OA Framework Tutorial caja de herramientas se compone de 3 objetos de entidad, pero el PurchaseOrderHeaderEOImpl clase representa el objeto de negocio de la orden de compra.

Por ejemplo, en el marco de la OA Caja de Herramientas Tutorial, hemos creado un módulo de aplicación de validación a nivel de objeto de empresa denominada PurchaseOrderVAM y añadimos todos los objetos de vista de validación de la orden de compra a la misma.

**Implementación**

Desde el punto de vista de implementación, *validación de* objetos de aplicación no son diferentes de *regulares* objetos de aplicación; que se diferencian sólo por el caso de uso. Crear su módulo de aplicación de validación declarativa y asociar los objetos de vista de validación apropiados con él.

**Los expertos de la entidad**

El experto entidad es un singleton definido para ser un afiliado especial de un objeto de negocio (ya sea el objeto de entidad parte superior en una composición, o un objeto de entidad independiente). Incluye código común llamado por el objeto de negocio de poseer, o rutinas de validación simples llamados por otros objetos de entidad que no quieren el costo de crear instancias de la propia entidad objeto. Por ejemplo, un PurchaseOrderHeaderEOImpl clase no quiere crear una instancia de un todo SupplierEOImplclase sólo para averiguar si la clave externa IdProveedor se trata de establecer es válido. En su lugar, se llama a un isSupplierIdValue (Número IdProveedor) método de Singleton experto en la entidad del proveedor - una operación de peso mucho más ligero.

**Implementación**

Para crear un experto entidad, crear primero una clase que se extiende oracle.apps.fnd.framework.server.OAEntityExpert . La clase debe ser nombrado y empaquetado de acuerdo con las normas publicadas en los [OA Marco Archivo / Paquete estructura de directorios /](https://translate.googleusercontent.com/cs/cs_filedir.htm) estándares. En segundo lugar, es necesario asociar esta clase con un objeto de entidad:

**Nota:** Para los objetos de negocio de composición, asociar el experto con el objeto de nivel superior. De lo contrario, sólo tiene que asociar con el objeto de entidad independiente.

1. En el sistema Navigator de JDeveloper, seleccione el objeto entidad a la que desea adjuntar el experto. Haga clic derecho y seleccione Editar < *EntityOjectName* > ...
2. Navegue hasta la Propiedades de página para crear una nueva propiedad con las siguientes características:
   * Establecer el nombre de ExpertClass .
   * Establecer el valor al nombre completo de su clase experto. Por ejemplo: oracle.apps.fnd.framework.toolbox.schema.server.PurchaseOrderEntityExpert .
3. Seleccione Añadir para crear su propiedad.
4. Seleccione Aceptar para guardar los cambios.

**La reutilización de Business Objects**

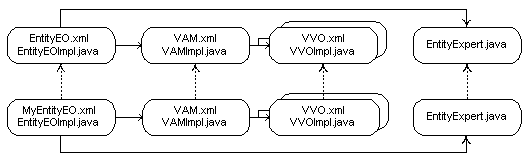
**Objetos de entidades, asociaciones, Validación Validación de AMS, Vos, Expertos entidad**

Si se desea crear Entidad objeto, Asociación, Validación AM, Validación VO, o Entidad Expertos objetos de negocio para su reutilización, se debe tener en cuenta las siguientes normas:

* El propietario del esquema de datos subyacente posee únicamente los objetos de entidad correspondientes, Asociaciones, módulos de aplicación de validación, la validación de objetos de vista, y los expertos de la entidad.
* El equipo de producto propietario de estos objetos de negocio debe documentar y publicar para otros equipos de producto de reutilizar.
* Cuando se crea una asociación compuesta que involucra a su EO (YourEO) y EO de otro equipo de producto (OtherBaseEO), debe extender la primera OtherBaseEO en su espacio de producto (OtherExtendedEO) antes de crear la asociación entre YourEO y OtherExtendedEO.
* Otro equipo de producto que está consumiendo el negocio de objetos que es el propietario lo desea, puede ampliar las validaciones que usted proporciona. En tal caso, el equipo de producto de consumo debería ampliar el Objeto de Entidad, Validación Módulo de Aplicación, Validación de visualización de objetos y de las Entidades de expertos e incluyen definiciones personalizadas y el código en las extensiones. Al extender los métodos de validación, asegúrese de que se invoca a super ()en la parte superior de los métodos de validación extendida. Por favor, vea el [negocio Extendiendo Objetos](https://translate.googleusercontent.com/translate_f#extendbo) subsección siguiente para más detalles.

**La extensión de Business Objects**

El siguiente diagrama muestra los objetos que usted se ocupa de la hora de crear objetos de negocio y las extensiones asociadas que desea hacer para extender las validaciones.

  
La primera fila del diagrama anterior representa una lista exhaustiva de todos los posibles objetos que se van a crear en la definición de un objeto entidad. El primer cuadro muestra que cuando se crea un objeto de entidad, dos archivos se generan: el archivo XML de definición de metadatos y el archivo de clase Java aplicación. Objetos entidad manejan atribuir validaciones nivel y nivel de registro. Estas validaciones menudo necesitan utilizar Validación Ver objetos (VVO). Validación objetos se agrupan bajo Validación Módulo de Aplicación (VAM). Como Entidad de objetos, creación de VVO y VAM del, genera un archivo XML de definición de metadatos y un archivo de clase de implementación de Java para cada objeto. Por último, la Entidad objetos a veces se basan en una clase que ayuda a ofrecer entre otros servicios, un servicio de validación optimizado para ser usado por otros objetos de entidad.Esta clase es la ayuda de expertos Entidad y vinculado a la entidad a través de objetos de una propiedad de objeto entidad.

El diagrama anterior ilustra un caso en el que se extienden todos los objetos de negocio. Eso no es necesariamente siempre el caso. En la mayoría de los casos, puede estar satisfecho con la extensión de algunos de estos objetos. Tenga en cuenta que nunca se debe modificar la definición de base de un objeto o hacer una copia de un objeto base. Siempre se debe ampliar el objeto relevante y utilizar mecanismos de sustitución / fábrica para hacer referencia a los objetos extendidos.

Por ejemplo, puede estar satisfecho con la ampliación de la Experto Entidad para anular un método de validación tal como isSupplierValid . En este caso, tenga en cuenta que no es prudente hacer referencia a la Entidad de expertos extendida (MyEntityExpert) directamente desde el objeto de entidad de base (EntityEO.XML) ya que este planteamiento no sobrevive a las actualizaciones. Un mejor enfoque requiere extender el objeto de entidad de base utilizando el asistente de creación de objetos entidad JDeveloper y actualizar la propiedad experto en la entidad objeto de entidad extendida para apuntar a la Entidad de expertos extendida.

Otro enfoque es extender simplemente el objeto de entidad a través del asistente de creación de objeto de entidad JDeveloper para agregar una validación adicional a la OtherExtendedEOImpl clase (asegúrese de que invoque super () en primer lugar) que no requiere ningún objeto vista de validación adicionales.

Tenga en cuenta que las extensiones de objeto de negocio se colocan debajo de la jerarquía paquete de los equipos de producto de consumo.

**Ver objetos, Ver vínculos**

Si se desea crear Vista vínculo de objeto o los objetos de negocio para su reutilización, se debe tener en cuenta las siguientes normas:

* Ver objetos y Ver vínculos se crean para una lógica de interfaz de usuario o negocio en particular. Una vez dicho esto, un equipo de producto que posee estos objetos puede optar por publicar cierta Ver objetos y Ver vínculos para el consumo público.
* El equipo de producto de la posesión de un objeto de la vista del público debe proporcionar la documentación necesaria y debe garantizar el contrato de los objetos (cláusula WHERE, atributos, y así sucesivamente).
* Otros equipos de producto pueden ampliar las capacidades de una base de la visión de objetos pública mediante la vista JDeveloper asistente de creación de objeto.
* También puede extender una vista pública definición de objeto para incluir atributos adicionales, si se desea.

Para obtener más información sobre la extensibilidad, consulte [el Capítulo 9: Ampliación y Despliegue de Aplicaciones de OA Framework](https://translate.googleusercontent.com/cust/cust_ext.htm) .

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