**Anatomía de una página de Marco OA**

Este documento describe los elementos básicos de una página típica de OA Framework.

**Contenido**

* [Conceptos básicos de la página](https://translate.googleusercontent.com/translate_f#pagebasics)
* [El modelo](https://translate.googleusercontent.com/translate_f#model)
* [La vista](https://translate.googleusercontent.com/translate_f#view)
* [El controlador](https://translate.googleusercontent.com/translate_f#controller)
* [Arquitectura Web Bean](https://translate.googleusercontent.com/translate_f#webbean_arch)
* [Guía de OA Framework Javadoc](https://translate.googleusercontent.com/translate_f#javadoc)

**Requisito de lectura**

Si eres nuevo en el desarrollo de aplicaciones web, por favor, lea el corto [JSP Application Primer](003_JSP_Application_Primer.docx) antes de proceder. La Guía del desarrollador de OA Framework asume que está familiarizado con los conceptos y el vocabulario presentados en JSP Primer.

**Conceptos básicos de la página**

En el nivel del navegador, un OA Framework página, como cualquier otra página web, se representa como HTML estándar.

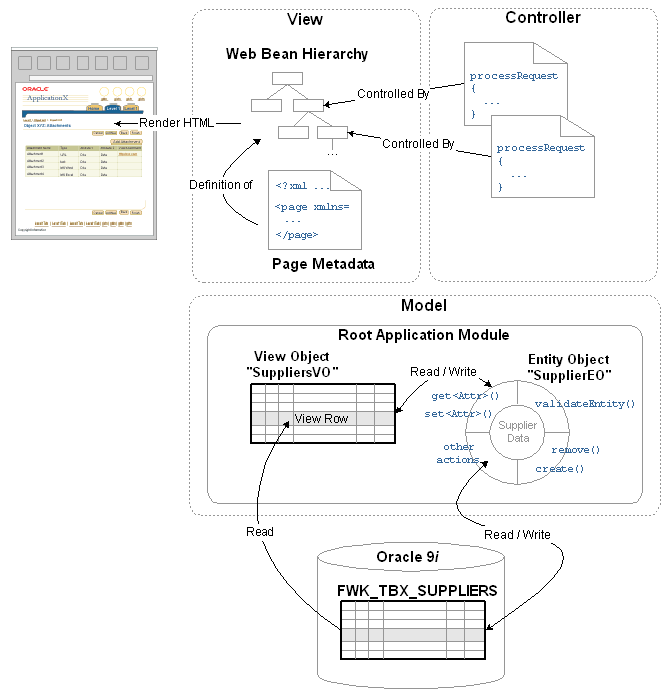
En el nivel medio, sin embargo, esta página se implementa en la memoria como una jerarquía de beans Java, muy similar a una interfaz de usuario clásica de Java. Cada widget UI, como botones, una tabla, las pestañas, la imagen de marca de la aplicación, etc., que los resultados en la página corresponden a uno o más beans web en la jerarquía.

Cuando el navegador emite una solicitud para una nueva página, OA Framework lee la definición de metadatos declarativos de la página para crear la jerarquía del bean web. Para cada bean con un controlador de UI asociado, OA Framework llama al código que escribe para inicializar la página. Cuando se completa el procesamiento de la página, OA Framework entrega la jerarquía del bean web al marco UIX para que pueda generar y enviar HTML al navegador.

Cuando el navegador emite un formulario, envíe (si, por ejemplo, el usuario selecciona el botón Ir de una región de búsqueda), OA Marco recrea la jerarquía de bean web, si es necesario (la jerarquía se almacena en caché entre las peticiones, y por lo general tiene que ser recreada sólo en casos excepcionales que vamos a discutir en detalle más adelante), y luego llama a cualquier código de manejo de eventos que haya escrito para los beans de página. Cuando finaliza el procesamiento de la página, el HTML de la página se genera de nuevo y se envía al navegador.

El resto de este documento presenta el modelo específico, la vista y el código del controlador y las definiciones declarativas que usted crea para implementar una página.

Figura 1: Una ilustración conceptual de la arquitectura modelo-vista-controlador de OA Framework



**El modelo**

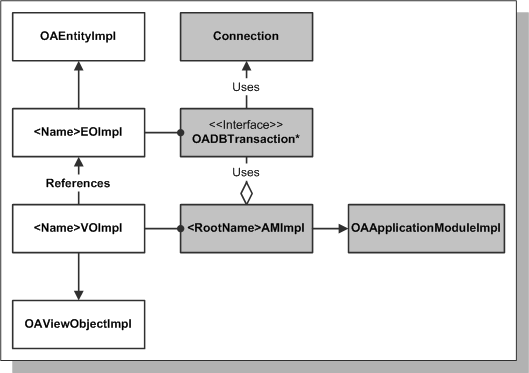
El modelo encapsula los datos subyacentes y la lógica comercial de la aplicación. También proporciona una abstracción de los objetos comerciales del mundo real y los servicios de aplicación que proporciona.

La [implementación del modelo de](https://translate.googleusercontent.com/build/build_model.htm) documento en el Capítulo 3 discute todo lo siguiente en detalle.

**Módulos de aplicación**

Un módulo de aplicación BC4J es esencialmente un contenedor que administra y proporciona acceso a objetos de modelo BC4J "relacionados". En este contexto, los objetos están "relacionados" en virtud de participar en la misma tarea. Por ejemplo, todos los objetos BC4J que comprenden una sola transacción participan en la misma tarea, incluso si la interfaz de usuario correspondiente requiere que el usuario visite varias páginas.

Figura 2: Arquitectura básica del modelo - Asociaciones del módulo de aplicación



Todos los módulos de aplicación que crea subclase de la clase oracle.apps.fnd.framework.server.OAApplicationModuleImpl.

Cada página de OA Framework tiene un módulo de aplicación "raíz", los cuales está asociado con la región de la página de nivel superior (la región **pageLayout).** El módulo de aplicación raíz proporciona el contexto de transacción y establece una conexión de base de datos.

* Si varias páginas participan en la misma transacción física o virtual, deben compartir el mismo módulo de aplicación raíz.
* Si una página funciona independientemente de cualquier otra, debe tener su propio módulo de aplicación.

La [administración de estados de OA Framework](https://translate.googleusercontent.com/ess_state.htm) documento en este capítulo se analiza la relación entre los módulos de la aplicación de la raíz y diferentes tipos de páginas en detalle.

**Nota:** También es posible que un módulo de aplicación de la raíz para contener uno o más módulos "anidados" aplicación, que puede anidar mismos niños a cualquier nivel arbitrario. En este escenario, el módulo de la aplicación raíz tiene acceso a todos los datos / objetos que tienen sus hijos, y todos los niños participan en la misma transacción establecida por la raíz. Utilizará esta función siempre que quiera crear una región UI reutilizable que interactúe con la base de datos.

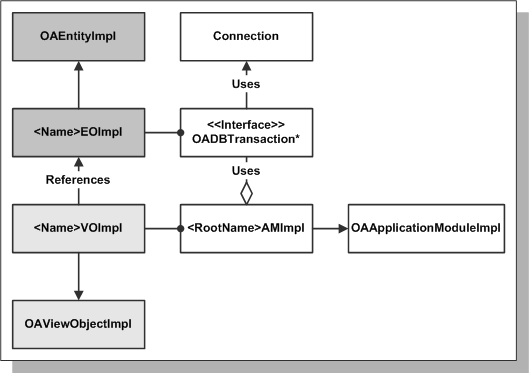
**Objetos de entidad (y objetos de asociación)**

Objetos de entidad BC4J encapsular las reglas de negocio (validaciones, acciones, etc.) asociadas con una fila en una tabla de base de datos. Por ejemplo, la Biblioteca Marco OA ToolBox muestra incluye una mesa FWK\_TBX\_SUPPLIERS para almacenar las definiciones de los proveedores. También se define un objeto de entidad para esta tabla (SupplierEO) que implementa todas las reglas de negocio para la inserción, actualización y eliminación de un proveedor.

**Nota: Los** objetos entidad también pueden basarse en las vistas, sinónimos o instantáneas.

OA Framework admite objetos de entidades tanto de Java como de PL / SQL (el Capítulo 5 analiza el diseño y la implementación de la lógica de negocios en detalle, incluido el asesoramiento sobre la elección de objetos de entidad de Java frente a PL / SQL).

Figura 3: Arquitectura básica del modelo - asociaciones de objetos de entidad



La mayoría de los objetos de entidad que se crea una subclase de la clase oracle.apps.fnd.framework.server.OAEntityImpl (verá un poco más tarde que los objetos PL / SQL entidad se extienden versiones especializadas de OAEntityImpl).

Existe una correspondencia de uno a uno entre una tabla y un objeto de entidad, y todos los objetos de entidad de Oracle E-Business Suite deben incluir todas las columnas en sus tablas asociadas. Los objetos de entidad utilizan una asignación declarativa entre sus atributos y las columnas de bases de datos subyacentes para implementar automáticamente consultas, inserciones, actualizaciones y eliminaciones. En la mayoría de los casos, todo lo que necesita hacer es agregar la lógica de validación.

Un objeto entidad está destinado a ser utilizado por *cualquier* programa (no sólo un cliente de OA Framework) que necesita para interactuar con una tabla dada. Como tal, debe consolidar toda la lógica de validación para el objeto de entidad, de modo que las reglas de negocio se implementen sistemáticamente, independientemente de qué cliente las utilice.

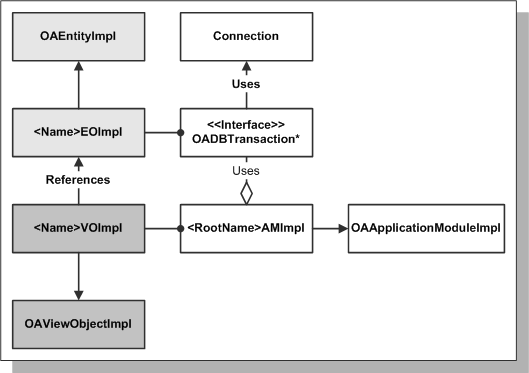
**Objetos de asociación**

Si tiene objetos complejos (como una orden de compra de 3 niveles con un 1: muchos entre los encabezados, líneas y envíos) también se puede definir las relaciones entre los correspondientes objetos de entidad mediante la creación de *objetos de asociación.* Puede crear asociaciones débiles (un encabezado de orden de compra "hace referencia" a un proveedor que existe independientemente de cualquier orden de compra determinada) y asociaciones de composición fuertes (un encabezado de orden de compra "posee" sus líneas, que no pueden existir fuera del contexto de su encabezado).

**Ver objetos (y ver enlaces)**

En los términos más simples, un objeto de vista BC4J encapsula una consulta de base de datos. Después de que se ejecuta una consulta, un objeto de vista proporciona iteración y acceso a su conjunto de resultados. El conjunto de resultados contiene una o más filas de vista, donde una fila de vista compuesta de atributos individuales corresponde a una fila devuelta por una consulta de base de datos.

Figura 4: Arquitectura básica del modelo - Ver asociaciones de objetos



Todos los objetos que se crean vista subclase de la clase oracle.apps.fnd.framework.server.OAViewObjectImpl.

Cada objeto de vista se puede configurar para consultar datos utilizando una de las siguientes estrategias:

* Sus atributos se asignan a columnas en una declaración SQL simple (comúnmente utilizada para objetos de vista pequeños y de solo lectura)
* Sus atributos se correlacionan con atributos de objeto de entidad (utilizados para insertar, actualizar y eliminar objetos de entidad)
* Algunos atributos se asignan a objetos de entidad, y algunos se completan directamente con SQL (se utilizan para aumentar los datos de objeto de entidad con columnas transitorias que no se pueden consultar a través de un objeto de entidad; un ejemplo calculado es el valor calculado utilizado exclusivamente para visualización de IU)

En una aplicación de OA Framework, usará objetos de vista en cada uno de los siguientes escenarios (todos los cuales se describen en detalle en los temas posteriores):

* Presentar datos que están optimizados para una interfaz de usuario en particular. Si la interfaz de usuario admite inserciones de objetos de entidad, actualizaciones y eliminaciones, interactuarás con el objeto de vista para realizar estas tareas.
* Cree consultas simples para listas pop, listas de valores y otros componentes de interfaz de usuario de soporte
* Cree "consultas de validación" eficientes que se puedan usar en su lógica comercial. Por ejemplo, en un objeto de entidad de encabezado de orden de compra puede usar un objeto de vista de validación para obtener el número de línea de orden de compra máximo actual de la base de datos para que pueda almacenar en caché e incrementar este valor a medida que se crean líneas nuevas.

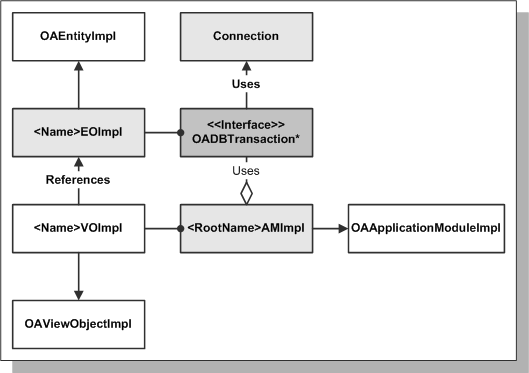
Finalmente, no solo definirá objetos de vista de manera declarativa, sino que escribirá código para ellos. En un caso típico, el código implementará enlaces de datos para consultas complejas y ejecutará la consulta (para que un objeto de vista sepa cómo "consultar").

**Ver enlaces**

Del mismo modo que se puede relacionar objetos de entidad, también se pueden crear asociaciones objeto Ver los llamados *vínculos de vista.* Por ejemplo, puede crear un enlace de vista entre un objeto de vista de encabezado de orden de compra y un objeto de vista de línea de orden de compra. Esto se puede usar en tiempo de ejecución para consultar automáticamente las líneas cuando se accede a un encabezado.

**OADBTransaction**

Figura 5: Modelo de arquitectura básica - OADBTransaction



**\* Nota:** Para ser totalmente exacta y consistente, este diagrama debe incluir la aplicación oracle.apps.fnd.framework.server.OADBTransactionImpl lugar de la interfaz oracle.apps.fnd.framework.OADBTransaction, sin embargo, hemos optado por incluir esta última ya utilizará exclusivamente la interfaz en su código.

Como se muestra en el diagrama anterior, el OADBTransaction juega un papel central en el código del modelo, ya que encapsula la sesión de conexión JDBC / base de datos asociada con un módulo de aplicación de la raíz, y directamente posee cualquier entidad objetos que cree (sus objetos de vista, propiedad de la módulo de aplicación raíz, mantenga referencias a sus objetos de entidad en sus filas de vista). También hará que el uso regular de la OADBTransaction en su código de modelo para las siguientes acciones comunes:

* Crear una instrucción invocable para ejecutar funciones y procedimientos de PL / SQL
* Acceder a la información de contexto de las aplicaciones a nivel de sesión, como el nombre del usuario, identificación, responsabilidad actual, etc.
* Acceso a una oracle.apps.fnd.framework.OANLSServices objeto si necesita realizar operaciones de NLS como la conversión de la fecha del servidor / hora en la fecha de usuario / hora, etc.

El acceso a la OADBTransaction es proporcionada por el módulo de aplicación radicular.

**La vista**

La vista formatea y presenta los datos del modelo al usuario.

La [implementación de la vista](https://translate.googleusercontent.com/build/build_view.htm) del documento en el Capítulo 3 discute todo lo siguiente en detalle.

**Definiendo la página**

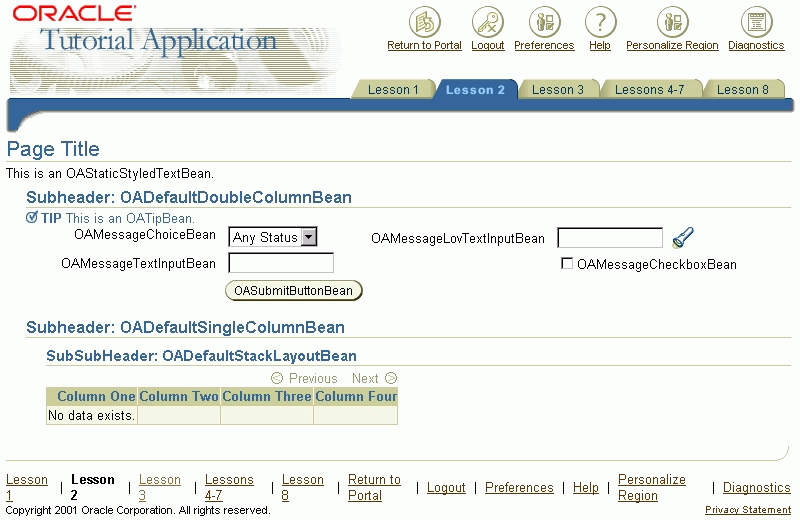
En el tiempo de desarrollo, se especifica la jerarquía de bean para cada página utilizando la herramienta JDeveloper declarativa que hemos introducido en el [edificio "Hola, mundo!"](https://translate.googleusercontent.com/gs/gs_hello.htm) . En el desarrollo de Oracle E-Business Suite, trabajará con (y con el control de origen) definiciones de páginas de archivos XML. Cuando su producto se implementa en el sitio de un cliente, OA Framework ejecuta las definiciones de página desde un repositorio de base de datos.

Para recapitular rápidamente, se utiliza para definir JDeveloper *páginas* compuestas de *regiones* y *artículos.*

* Los elementos son widgets simples como botones, campos, imágenes, etc. que no contienen elementos secundarios.
* Las regiones son objetos contenedores que pueden contener elementos y otras regiones. Los ejemplos de regiones incluyen encabezados, tablas y componentes especiales de diseño.
* Cada región y cada elemento que defina tiene una propiedad *de estilo* que le dice a la OA Framework qué objeto bean web para crear una instancia para que en tiempo de ejecución (y esto a su vez dicta lo que se genera HTML para el bean). Por ejemplo, si se define una región cuya propiedad es de estilo "mesa", el Marco OA una instancia de un oracle.apps.fnd.framework.webui.beans.table.OATableBean.
* Todas las páginas deben tener una sola región de nivel superior (a menudo llamada la "zona de la raíz") cuyo estilo es **pageLayout**. Esto se instancia como una oracle.apps.fnd.framework.webui.beans.layout.OAPageLayoutBean.
* La secuencia en la que las regiones y los elementos aparecen en el árbol de páginas de JDeveloper (y el archivo XML correspondiente) le dice al Framework dónde agregar estos objetos a la jerarquía de bean en tiempo de ejecución.

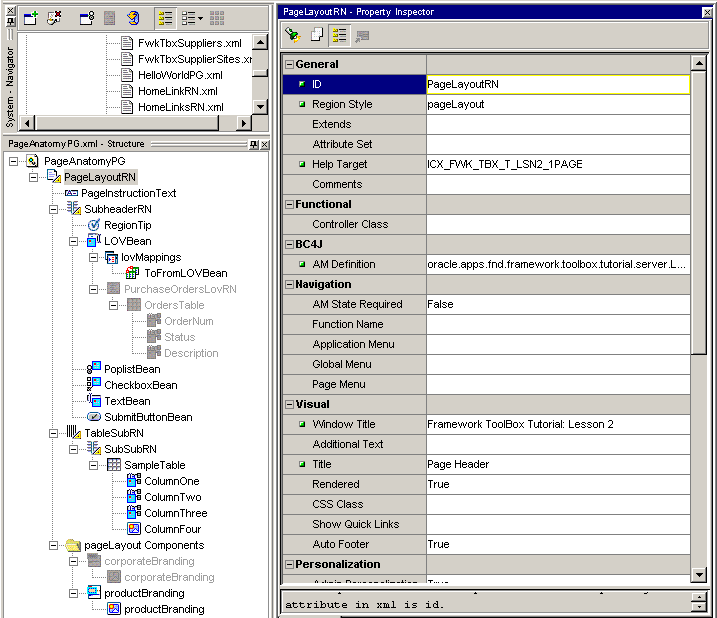
La Figura 3 a continuación brinda una mirada entre bastidores a los tipos de beans web que se crean instancias para una página simple. Las etiquetas que veas nombran los beans web subyacentes. Por ejemplo, un poplist se instancia como una oracle.apps.fnd.framework.webui.beans.message.OAMessageChoiceBean, y un botón de envío se instancia como una oracle.apps.fnd.framework.webui.beans.form.OASubmitButtonBean. La Figura 4 muestra la definición de página correspondiente.

Figura 3: Página con componentes de UI que muestran los nombres de los beans web correspondientes



**Nota:** Los nombres de las regiones y el tema se muestran a continuación no cumplen con el [Oracle E-Business Suite estándares de nomenclatura](https://translate.googleusercontent.com/cs/cs_filedir.htm) ; en cambio, están destinados a ayudarlo a traducir de la estructura a los beans web correspondientes.

Figura 4: estructura de página en JDeveloper



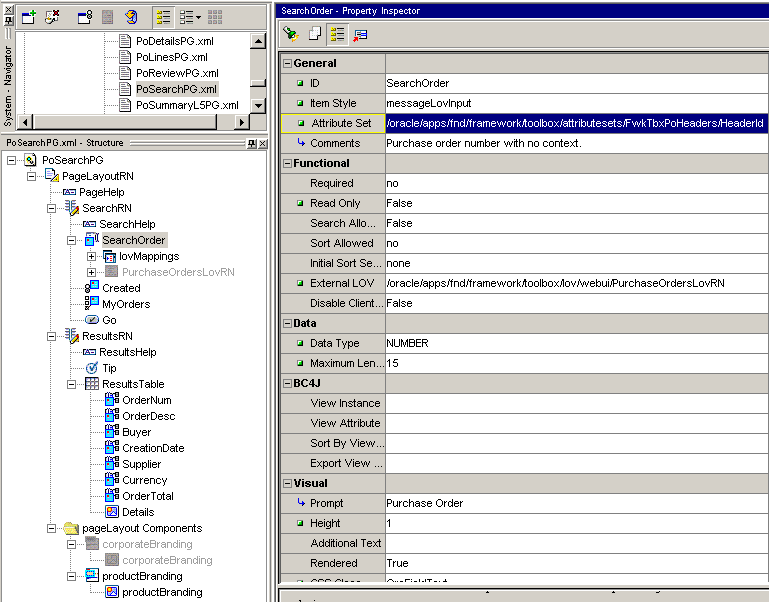
**Conjuntos de atributos**

Cada región o elemento puede heredar grupos de configuraciones de propiedad utilizando conjuntos de atributos. Un *atributo establecido* es un llamado colección, reutilizable de propiedades que pueden ser utilizados por cualquier tipo de objeto de interfaz de usuario, incluyendo las regiones, elementos y otros conjuntos de atributos. Cada vez que crea una interfaz de usuario que usa conjuntos de atributos, puede anular las propiedades heredadas (aunque esto se desaconseja en los estándares de codificación de OA Framework).

Para ilustrar este concepto, en el desarrollo de Aplicaciones, cada tabla debe tener conjuntos de atributos asociados para cada columna desplegable. Estos conjuntos de atributos incluyen propiedades como solicitud, ancho de visualización, etc.

* En el Cuadro de la muestra OA Framework Biblioteca / Tutorial, tenemos una tabla de órdenes de compra (FWK\_TBX\_PO\_HEADERS) con un HEADER\_ID columna de clave principal de tipo NUMBER que también se muestra a los usuarios como el número de orden de compra.
* Esta tabla tiene un atributo asociado establece archivo del paquete XML llamado **FwkTbxPoHeaders** que incluye todos los conjuntos de atributos para las columnas que se pueden visualizar de la tabla (un atributo establecido por columna). Uno de los conjuntos de atributos se llama **HeaderId.**
* El atributo de **HeaderId** tiene la propiedad Prompt establece en **orden de número** y la duración de presentación establecido en algo razonable como **15.**
* Cuando creamos una página que incluye el elemento número de orden de compra, que especifique el conjunto de atributos alojamiento hasta el completo conjunto de atributos Nombre / Oracle / aplicaciones / enc / marco / caja de herramientas / attributesets / FwkTbxPoheaders / Headerid

Figura 5: Uso de un conjunto de atributos en JDeveloper



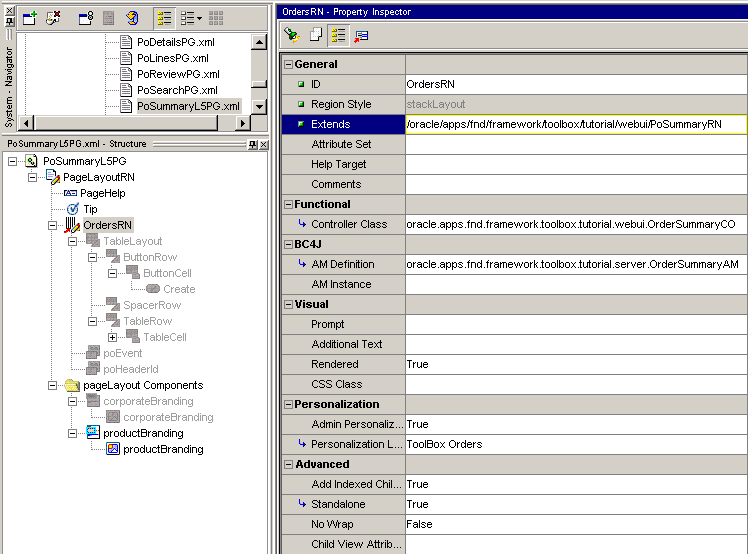
**Reutilización de componentes**

Si desea incorporar objetos compartidos en su página, puede simplemente *extenderlas.*

Por ejemplo, en la muestra Biblioteca / Tutorial OA Marco caja de herramientas que hemos creado una región común (llamado **PoSummaryRN)** por lo que el mismo contenido puede incluirse en múltiples páginas sin recodificación. Para agregar esta región compartida a una página, simplemente crea una nueva región, y después fijar su Extiende propiedad para el nombre completo de la región compartida: / oracle / aplicaciones / enc / marco / caja de herramientas / tutorial / webui / PoSummaryRN

**Nota:** La región compartida no se puede editar en la página de referencia, por lo que sus artículos aparecen en gris en el panel Estructura JDeveloper.

Figura 6: Ampliación de una región JDeveloper



**Encuadernación de fuente de datos**

Para los beans con cualquier interacción de base de datos (consulta, inserción, actualización y / o eliminar), también se especifica una fuente de datos la unión a una vista de instancia Nombre y asociado Ver nombre del atributo. Este enlace es crucial porque OA Framework lo utiliza para obtener datos consultados y escribir datos ingresados ​​por el usuario en el objeto de vista subyacente.

* La vista Nombre de instancia hace referencia a la vista de objetos subyacente dentro del contexto de su módulo de aplicación que contiene (todo Ver objetos "en vivo" dentro de un módulo de aplicación y se identifica por un nombre de instancia en su contenedor). Por ejemplo, si un objeto vista SuppliersVO se identifica por el nombre de instancia "MySupVO" dentro del módulo de aplicación de la raíz de su página, "MySupVO" es el nombre que especifique aquí.
* El nombre de vista Atributo referencia a la vista atributo de objeto subyacente que se asigna a una columna. Por ejemplo, si su SuppliersVO tiene un atributo "IdProveedor" (que se asigna a la columna SUPPLIER\_ID subyacente), "IdProveedor" es el nombre que especifique aquí.

**Definiendo el Menú**

Todas las aplicaciones de OA Framework incluyen menús como se describe en el [Look and Feel de Oracle Browser (BLAF) Pauta de interfaz de usuario: aquí / Navegación](https://translate.google.com/translate?hl=es&prev=_t&sl=en&tl=es&u=http://uistandards/blaf/guidelines/tabs.html) [ [OTN Versión](https://translate.google.com/translate?hl=es&prev=_t&sl=en&tl=es&u=http://otn.oracle.com/tech/blaf/specs/tabs.html) ]. Defina estas estructuras de menú de forma declarativa utilizando el menú de Oracle E-Business Suite y los formularios de definición de funciones. Discutiremos esto en detalle más adelante en la Guía del desarrollador.

Del mismo modo que OA Framework traduce su diseño declarativo de UI en la jerarquía de beans en tiempo de ejecución, también incluye web beans para la definición del menú declarativo.

**Definición de flujo de página**

Cuando se trata de flujos de transacciones de varias páginas, el OA Framework proporciona una alternativa declarativa (por lo tanto, personalizable) a la lógica de controlador compleja y codificada. Ver [Capítulo 4: Uso de pageflow declarativa de flujo de trabajo](https://translate.googleusercontent.com/feat/feat_pageflow.htm) para obtener información adicional acerca de esta característica.

**Personalizando páginas**

El OA Framework también incluye una infraestructura de personalización declarativa llamada OA Personalization Framework. Esto está destinado a respaldar las necesidades de personalización de los usuarios finales y la cadena de entrega del producto (cambios para la localización, verticalización, etc.).

**Nota:** Como se verá a lo largo de la Guía del desarrollador, la creación de regiones y artículos declarativa es siempre preferible a la creación de ellos mediante programación. De hecho, debe crear componentes programáticamente ÚNICAMENTE si no puede crearlos declarativamente para que los clientes puedan personalizar su trabajo.

**El controlador**

El controlador responde a las acciones del usuario y dirige el flujo de la aplicación.

La [Implementación del controlador](https://translate.googleusercontent.com/build/build_control.htm) documento en el Capítulo 3 discute todo lo siguiente en detalle.

Los controladores pueden estar asociados con la vista en el nivel de región (en términos más generales, cualquier Web Beans OA Framework que implementan la interfaz oracle.apps.fnd.framework.webui.beans.OAWebBeanContainer pueden tener controladores asociados).

Todos los controladores que se crean subclase oracle.apps.fnd.framework.webui.OAControllerImpl como se muestra en la Figura 7 a continuación.

La clase de controlador es donde se define cómo se comportan los beans web. Específicamente, usted escribe el código del controlador para:

* Manipule / inicialice la IU en tiempo de ejecución (incluida cualquier distribución programática que no pueda realizar de forma declarativa) y
* Interceptar y manejar eventos de usuario como pulsaciones de botones

**Gestión de solicitudes**

Cuando el explorador emite una solicitud OA.jsp para una de sus páginas:

1. El oracle.apps.fnd.framework.webui.OAPageBean (la clase principal OA Framework procesamiento de páginas) utiliza el nombre de la página para determinar qué módulo de aplicación de la raíz que necesita para que pueda comprobar que funciona desde la piscina módulo de aplicación. Este módulo de aplicación también verifica una conexión JDBC desde el grupo de conexiones, y se establece el contexto de transacción para la página.
2. La sesión del usuario está validada; si no es válido, se muestra una página de inicio de sesión (tenga en cuenta que esto es una simplificación, detalles adicionales se proporcionan más adelante en la Guía del desarrollador).
3. Suponiendo que el usuario es válido, el OAPageBean evalúa parámetros de la petición de averiguar si se está tratando con un HTTP POST o GET.

**Manejo de una solicitud GET**

Cuando el explorador emite una solicitud GET al servidor de una página (o manualmente hacia adelante a la misma), la OA Framework utiliza la definición de la interfaz de usuario declarativa para construir la jerarquía de Web Bean:

1. El OAPageBean llama processRequest () en bean pageLayout de nivel superior de la página, y toda la jerarquía de bean web se procesa de forma recursiva del siguiente modo para inicializar los granos de web (incluyendo cualquier componentes del modelo asociados):
   1. Cada grano de web crea una instancia su controlador - si tiene uno - y pide processRequest (OAPageContext pageContext, OAWebBean webBean) en el controlador. Este es el método que usa para construir / modificar el diseño de su página, establecer propiedades de bean web y realizar cualquier inicialización manual de datos (si, por ejemplo, necesita realizar una autoconsulta cuando navega a la página).
   2. Algunos granos web complicadas (como el oracle.apps.fnd.framework.webui.beans.table.OATableBean y oracle.apps.fnd.framework.webui.beans.layout.OAPageLayoutBean) realizan el procesamiento post-controlador llamando a su prepareForRendering () métodos (este método se describe en el correspondiente Javadoc bean).
   3. Cada grano de Web llama processRequest () en sus hijos.
2. El oracle.apps.fnd.framework.webui.OAPageBean da la jerarquía de Web Bean a UIX para hacer y enviar al navegador.

Manejo de una solicitud POST

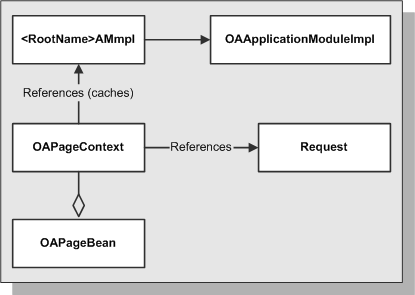
Cuando el explorador emite una solicitud POST al servidor de una página:

1. El OAPageBean comprueba si la jerarquía de Web Bean está en la memoria. Si no es así (porque los recursos fueron recuperados, el usuario navega con el botón Atrás del navegador, o un POST se emite a la página principal de una página de un mensaje de diálogo), que recrea la jerarquía tal como se describe en el procesamiento GET anteriormente.
2. El OAPageBean llama processFormData (OAPageContext pageContext, OAWebBean webBean) en todos los granos en la jerarquía de escribir los datos del formulario al modelo (específicamente, se llama processFormData () en la región pageLayout, y luego cada uno de Web Bean llama de forma recursiva processFormData () en sus hijos). Escribir los datos del formulario en el modelo subyacente invoca automáticamente las validaciones a nivel de atributo y de entidad, y si lanza cualquier excepción de validación, el proceso se detiene y los mensajes de error se muestran al usuario.
3. Si no se lanzan excepciones durante la fase de processFormData (), llama OAPageBean processFormRequest (OAPageContext pageContext, OAWebBean webBean) en todos los granos en la jerarquía utilizando el mismo método descrito anteriormente.Este pase le da a su código de controlador la oportunidad de responder a las acciones del usuario.
4. Si no hay delante o página JSP redirecciones se emitieron - o excepciones fueron lanzados en processFormRequest () - a continuación, se actualiza la página.

**OAPageContext**

Cuando la OA Framework recibe una solicitud OA.jsp, la OAPageBean crea una oracle.apps.fnd.framework.webui.OAPageContext,una clase que existe sólo durante la duración del procesamiento de la página. Cada uno de los tres principales métodos descritos anteriormente (processRequest (), processFormData () y processFormRequest ()) realiza un OAPageContext como parámetro, y cualquier código de controlador que se escribe, invariablemente, hacer uso de esta clase crucial.

Figura 7: Relación entre OAPageContext y otras clases de claves



Como se ilustra en el diagrama anterior, el OAPageContext tiene una referencia tanto a la petición y el módulo de aplicación de la raíz. Teniendo en cuenta estas relaciones, y el hecho de que un OAPageContext se pasa a cada uno de sus métodos de procesamiento de la respuesta del controlador, se puede ver cómo se usaría el OAPageContext para la siguiente lista de tareas comunes:

**Acceder a los parámetros de solicitud**

Tal vez lo más importante, esta es la clase que se utiliza para leer los valores de solicitud llamando a un método getParametersencilla (String nombre) (recordar que la solicitud incluye ningún parámetro de URL más - si se trata de un POST - todos los valores de campo de formulario más la nombres y eventos asociados con cualquier widgets de acción / control seleccionados por el usuario).

**Consejo:** Para los beans web individuales en su página (botones, campos, y así sucesivamente) el valor del nombre pasó a getParameter () es el identificador único correspondiente que se asigna al definir su página. Entonces, por ejemplo, puede decir si el usuario presionó un botón que llamó "GoButton" en JDeveloper escribiendo el siguiente código en un controlador:

processFormRequest (OAPageContext pageContext, OAWebBean webBean)

{

si **(pageContext** .getParameter ( "GoButton")! = null)

{

// El usuario presionó el botón "Ir", hacer algo ...

}

}

**Accediendo al Módulo de Aplicación Raíz**

El OAPageContext almacena una referencia al módulo de aplicación de la raíz, que a su vez proporciona acceso a sus objetos de vista y la transacción. Si necesita acceder a un módulo de aplicación, pedir al OAPageContext:

processFormRequest (OAPageContext pageContext, OAWebBean webBean)

{

OAAplicationModule am =

(OAApplicationModule) **pageContext** .getRootApplicationModule ();

}

**Emitir instrucciones de navegación**

Utiliza métodos en esta clase para indicar al OA Framework que realice un reenvío JSP o un redireccionamiento del cliente. Por ejemplo (revisaremos este método con mayor detalle más adelante en la Guía del desarrollador):

processFormRequest (OAPageContext pageContext, OAWebBean webBean)

{

if (pageContext.getParameter ("CreateButton")! = null)

{

// El usuario presionó el botón "Crear proveedor", ahora realiza un adelanto JSP para

// la página "Crear proveedor".

**pageContext** .setForwardURL ( "página OA.jsp? = / oracle / aplicaciones / DEM / empleado / webui / EmpDetailsPG",

nulo,

OAWebBeanConstants.KEEP\_MENU\_CONTEXT,

nulo,

nulo,

cierto, // Retener AM

OAWebBeanConstants.ADD\_BREAD\_CRUMB\_YES, // Mostrar las migas de pan

OAWebBeanConstants.IGNORE\_MESSAGES);

}

}

**Acceder a la información del contexto de la aplicación**

Al igual que el OADBTransaction en su código de modelo, el OAPageContext proporciona acceso a servlet a nivel de sesión de Oracle E-Business Suite información de contexto como el nombre, la identificación, la responsabilidad actual del usuario y así sucesivamente. Por ejemplo, el siguiente fragmento de código muestra cómo obtener el nombre del usuario:

processRequest (OAPageContext pageContext, OAWebBean webBean)

{

Cadena usuario = **pageContext** .getUserName ();

}

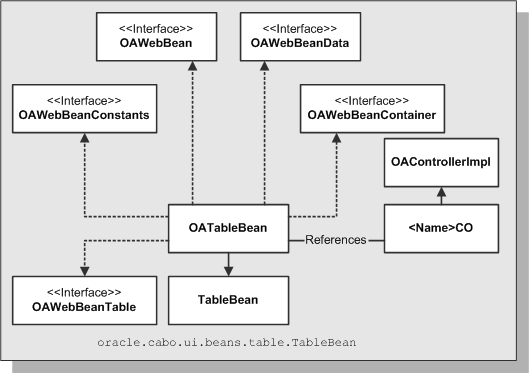
**Arquitectura Web Bean**

En primer lugar, todos los beans web de OA Framework clasifican los beans correspondientes en el marco UIX. Por ejemplo, un OATableBean extiende una oracle.cabo.ui.beans.table.TableBean ( "cabo" era un nombre anterior para el marco UIX, y las definiciones de paquetes todavía utiliza este nombre de edad).

Cada bean web de OA Framework también implementa un grupo de interfaces cuyas implementaciones definen colectivamente los comportamientos que OA Framework agrega a los beans UIX base.

* oracle.appps.fnd.framework.webui.beans.OAWebBean - define el comportamiento básico común a todos los granos web (por ejemplo, entre otros comportamientos clave, esto define los métodos processRequest, processFormData y processFormRequest que los granos individuales implementan por sí mismos).
* oracle.apps.fnd.framework.webui.OAWebBeanConstants - una colección de constantes utilizadas en la vista módulos / controlador
* oracle.apps.fnd.framework.webui.beans.OAWebBeanData - define el comportamiento de la gestión de la personalización definición común y datos de origen
* oracle.apps.fnd.framework.webui.beans.OAWebBeanContainer - define las características de todos los granos de la tela que pueden actuar como contenedores para otros granos web. Por ejemplo, todos los beans web de diseño implementan esta interfaz. Solo los beans que implementan esta interfaz pueden tener controladores asociados.
* OAWebBean <tipo> - define los comportamientos inherentes de un Bean dentro del contexto del Marco de la OA. Por ejemplo, el OATableBean implementa la interfaz oracle.apps.fnd.framework.webui.beans.OAWebBeanTable.

Figura 8: Ejemplo de un bean web contenedor (OATableBean)

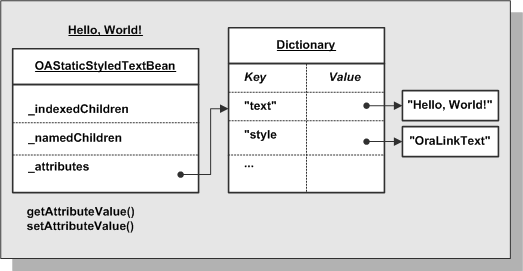


**Estructura interna del bean**

Cada web bean mantiene la siguiente información sobre sí mismo:

* \_indexedChildren - Web Beans niño
* \_namedChildren - Web Beans niño que las etiquetas UIX para un comportamiento especial
* \_attributes - características de bean web (propiedades descriptivas) como se ilustra en la Figura 9 a continuación

Figura 9: Ilustración del uso de un diccionario en la web bean para rastrear los pares clave / valor de sus atributos



**Valores enlazados a datos**

En lugar de los valores literales como se ilustra en la Figura 9 anterior, los atributos de bean web de OA Framework se implementan realmente como valores enlazados a datos, lo que significa que el valor es proporcionado por una fuente de datos subyacente que se resuelve al componente en el tiempo de renderizado. Verá un poco más adelante cómo definir y usar valores encuadernados personalizados en su código.

**Representación**

En el momento de la representación de la página, el marco UIX procesa la jerarquía del bean web para generar el HTML de la página.

Para cada atributo de Web Bean, UIX llama a su método getAttributeValue () mientras pasa un *contexto de representación* (un contexto de representación es básicamente toda la información que necesita para resolver UIX valores ligados). Para un atributo dado, por ejemplo, el contexto de representación sabe cuál es la instancia del objeto de vista subyacente, el atributo de vista y la fila actual. El valor enlazado a datos utiliza la información proporcionada por el contexto de representación para interrogar a su fuente de datos y devuelve el valor real a UIX para que pueda generar el HTML correspondiente.

**Guía de OA Framework Javadoc**

Probablemente ya hayas notado que la Guía del Desarrollador enlaza directamente con el OA Framework Javadoc para clases individuales. Dado que ahora tiene un conocimiento de alto nivel de los componentes que utilizará cuando construye una página, esta sección resume brevemente el propósito de cada paquete de OA Framework y describe cuándo es probable que utilice Javadoc para el correspondiente clases / interfaces.

**oracle.apps.fnd.framework**

Contiene clases e interfaces a las que se puede acceder de forma segura desde el modelo (servidor) y el controlador de interfaz de usuario o código de vista (cliente). Por ejemplo, si necesita acceder a un módulo de aplicación de la raíz en su página, que va a utilizar la interfaz oracle.apps.fnd.framework.OAApplicationModule (que nunca acceder a una aplicación en el cliente).

Entre otras cosas, este paquete también incluye:

* Todas las excepciones de OA Framework que pueda tener ocasión de lanzar
* La clase OANLSServices que va a utilizar para llevar a cabo las operaciones de internacionalización

**oracle.apps.fnd.framework.server**

Contiene clases e interfaces para implementar el modelo en una aplicación Modelo-Vista-Controlador de OA Framework.

Estas clases están destinados a ser utilizados con cualquier interfaz de usuario del cliente (no sólo las páginas HTML OA Marco), y como tal no debería tener referencias directas a las clases e interfaces publicadas en el paquete oracle.apps.fnd.framework.webui y sub-paquetes, o en cualquier paquete y subpaquetes webui específicos de la aplicación.

Al construir un modelo de aplicación de OA Framework, siempre debe trabajar con las clases de este paquete en lugar de las clases BC4J que extienden.

**Advertencia:** Nunca llame a las clases de este paquete de controlador o código de la vista.

**oracle.apps.fnd.framwork.webui**

Contiene clases principales para compilar y manipular las interfaces de usuario HTML de OA Framework.

Algunas de las clases / interfaces más comúnmente usadas en este paquete incluyen:

* OAController
* OAPageContext
* Cualquier clase en los subpaquetes de beans que se describe a continuación

**Advertencia:** Nunca se refiera a ninguna clase en este paquete de código del modelo.

**oracle.apps.fnd.framework.webui.beans**

Contiene clases de bean web para los componentes de la interfaz de usuario que no encajan perfectamente en los diversos subpaquetes de bean (por ejemplo: imagen, conmutador, texto de instrucciones estáticas, keyfield de teclas, flexfield descriptivo y más). Debe usar estas clases cuando escriba un controlador de interfaz de usuario que necesite manipular los beans web mediante programación.

Este paquete también contiene interfaces principales de OA Framework implementadas por todos los beans web.

Las clases de este paquete y sus subpaquetes corresponden a los componentes UIX que se extienden, como se muestra a continuación. Al compilar las páginas de la aplicación OA Framework, siempre debe trabajar con las clases de OA Framework a menos que se haya introducido una nueva característica en UIX y aún no sea compatible con el marco.

**Nota: Las** clases OA Framework siempre se crean instancias para las páginas MDS que se genera de forma declarativa en JDeveloper.

|  |  |
| --- | --- |
| **Paquete UIX** | **Paquete OA** |
| oracle.cabo.ui.beans | oracle.apps.fnd.framework.webui.beans |
| oracle.cabo.ui.beans.form | oracle.apps.fnd.framework.webui.beans.form |
| oracle.cabo.ui.beans.include | oracle.apps.fnd.framework.webui.beans.include |
| oracle.cabo.ui.beans.layout | oracle.apps.fnd.framework.webui.beans.layout |
| oracle.cabo.ui.beans.message | oracle.apps.fnd.framework.webui.beans.message |
| oracle.cabo.ui.beans.nav | oracle.apps.fnd.framework.webui.beans.nav |
| oracle.cabo.ui.beans.table | oracle.apps.fnd.framework.webui.beans.table |

**Advertencia:** Nunca se refiera a ninguna clase en este paquete de código del modelo.

**oracle.apps.fnd.framework.webui.beans.form**

Contiene clases de bean web para los componentes de formulario HTML que incluyen un botón de enviar y diversos datos de entrada / controles de especificación (casilla de verificación, el grupo de radio, lanzadera, campo de entrada de texto y más). Debe utilizar estas clases para escribir un controlador de interfaz de usuario que necesita manipular mediante programación los granos web.

Para muchos de los Web Beans en este paquete hay variantes en el oracle.apps.fnd.framework.webui.beans.message paquete (los granos web de mensajes tienen la capacidad de mostrar el error, la información y los iconos de advertencia con un mensaje explicativo que, correspondientes Web Beans de datos de entrada / especificación en este paquete no lo hacen). Al crear sus páginas de forma declarativa en JDeveloper, el Marco de la OA instancia automáticamente los granos de mensajes para cualquiera de los componentes que existen en ambos paquetes. Debe utilizar las clases de este paquete sólo en los siguientes casos:

* La clase no tiene una alternativa bean controlado por mensajes (por ejemplo, el OASubmitButtonBean sólo existe en este paquete)
* No se puede utilizar la alternativa bean controlado por mensajes

**Advertencia:** Nunca se refiera a ninguna clase en este paquete de código del modelo.

**oracle.apps.fnd.framework.webui.beans.include**

Contiene clases de bean web para la inclusión de fragmentos de interfaz de usuario de fuentes externas (servlets, páginas JSP y HTML plano) en las páginas de aplicación OA Framework. Debe utilizar estas clases para escribir un controlador de interfaz de usuario que necesita manipular mediante programación los granos web.

**Advertencia:** Nunca se refiera a ninguna clase en este paquete de código del modelo.

**oracle.apps.fnd.framework.webui.beans.layout**

Contiene clases de Web Bean para el tendido de contenido en una página de aplicación OA Framework, incluyendo componentes especiales de composición como ocultar / mostrar, recipientes de contenido, listas con viñetas, encabezados, plantillas estandarizadas para los diseños individuales / dobles columnas y mucho más. Debe utilizar estas clases para escribir un controlador de interfaz de usuario que necesita manipular mediante programación los granos web.

**Advertencia:** Nunca se refiera a ninguna clase en este paquete de código del modelo.

**oracle.apps.fnd.framework.webui.beans.message**

Contiene clases de bean web para los datos de formularios componentes de entrada / especificación de HTML que son capaces de mostrar asociado de error, advertencia o icono (s) de información con un mensaje explicativo (por ejemplo, si un usuario introduce un valor incorrecto en un campo de entrada de texto en un icono de error hace que junto a su pronta). Debe utilizar estas clases para escribir un controlador de interfaz de usuario que necesita manipular mediante programación los granos web.

Muchos de los granos web en este paquete también se incluyen en el oracle.apps.fnd.framework.webui.beans.form paquete sin la capacidad de mostrar los iconos de mensaje y el texto suplementario. Al crear sus páginas de forma declarativa en JDeveloper, el Marco de la OA instancia automáticamente los granos de mensajes para cualquiera de los componentes que existen en ambos paquetes. Debe utilizar las clases sin la capacidad mensaje sólo si no se puede incluir granos de mensajes en su página.

**Advertencia:** Nunca se refiera a ninguna clase en este paquete de código del modelo.

**oracle.apps.fnd.framework.webui.beans.nav**

Contiene clases de bean web para los componentes de interfaz de usuario de navegación HTML (links, árboles, elementos de menú, enlaces rápidos, pan rallado y más). Debe utilizar estas clases para escribir un controlador de interfaz de usuario que necesita manipular mediante programación los granos web.

**Advertencia:** Nunca se refiera a ninguna clase en este paquete de código del modelo.

**oracle.apps.fnd.framework.webui.beans.table**

Contiene clases de bean web para las tablas (que presentan datos al usuario en un formato tabular) y los componentes HGrid (un híbrido de visualización de datos tabulares con la estructura jerárquica en forma de árbol). Debe utilizar estas clases para escribir un controlador de interfaz de usuario que necesita manipular mediante programación los granos web.

**Advertencia:** Nunca se refiera a ninguna clase en este paquete de código del modelo.

**oracle.apps.fnd.framework.webui.laf**

Contiene utilidades que se pueden utilizar para controlar las características de representación HTML de la página, incluyendo look-and-feel y el comportamiento específico del contexto (por ejemplo, el contenido puede ser optimizado para imprimir frente a la pantalla en un navegador regular o en un correo electrónico).

**Advertencia:** Nunca se refiera a ninguna clase en este paquete de código del modelo.

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