Diseño y Desarrollo de Sistemas de Información.

Sesión Práctica 6.
Desarrollo de Aplicaciones con Oracle ADF

Objetivo

En este tutorial se muestra de forma breve cómo desarrollar una aplicación empleado *Oracle Application Developments Framework* (Oracle ADF). Para ello, se emplearán las tecnologías Oracle ADF, Struts y JSP (JavaServer Pages).

La aplicación a desarrollar será una aplicación basada en web que permitirá la gestión de los de clientes de una compañía. Se comenzará el proceso desarrollando la capa de servicios de negocio basándose en Oracle ADF Business Components y posteriormente se desarrollará la capa cliente que permitirá la interacción con los usuarios.

Contenido

El tutorial abordará los siguientes contenidos:

- Introducción.
- Descripción de la aplicación a desarrollar.
- Creación de los servicios de negocio.
- Definición del flujo básico entre páginas.
- Añadir la funcionalidad de borrado con una página de confirmación.
- Creación y empleo de mensajes de usuario.
- Habilitación de la página de confirmación de borrado.

Introducción

Este tutorial demuestra cómo crear una aplicación para el mantenimiento de los datos de clientes de una compañía. La aplicación ofrecerá mediante un interfaz de páginas web la capacidad de añadir nuevos registros, editar los registros existentes, y eliminar registros. Durante el desarrollo de este tutorial aprenderá a crear dichas páginas y hacer que éstas sean amigables para el usuario o lo suficientemente robustas como para funcionar en un entorno multiusuario.

Para que la aplicación a desarrollar sea amigable con el usuario, ésta debe de ser fácil de usar y de entender, así como ofrecer mensajes útiles para el usuario del tal forma que éste sepa qué está ocurriendo en todo momento. El usuario debe de poder editar y eliminar los datos de los clientes así como crear nuevos registros de clientes. La navegación debe de ser sencilla e intuitiva, y los mensajes a los usuarios deben de ser claros y obvios. Por ejemplo, cuando un registro sea actualizado, el usuario debería de recibir un mensaje indicando el éxito de la operación. La misma técnica debe de ser aplicada cuando un usuario elimina o crea un registro así como para el tratamiento de cualquier error que pueda darse. Los mensajes que se

muestran al usuario deben de ser internacionalizables para que la aplicación pueda funcionar bajo múltiples lenguajes.

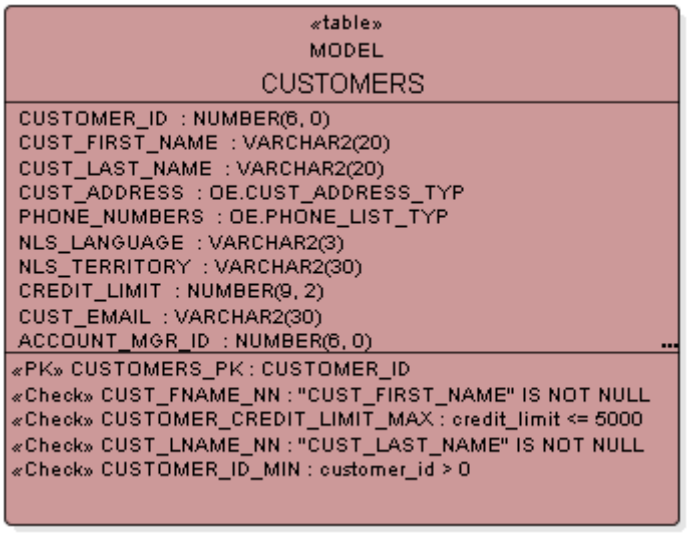
Antes de comenzar a desarrollar la aplicación es necesario realizar ciertas modificaciones en la base de datos. Para ello, abra la pestaña **Database Navigator** y abra la conexión de base de datos en la que tiene las tablas de ejemplo (utilizaremos el esquema de ejemplo OE). Si no tiene acceso a este esquema, habilítelo. Una vez conectado ejecute las siguientes sentencias:

```
create sequence CUSTOMERS_SEQ start with 1000;

create trigger CUSTOMER_INS_TRIG before insert on CUSTOMERS for each row
begin
        select CUSTOMERS_SEQ.nextval into :new.CUSTOMER_ID from dual;
end;
/
```

Estas sentencias crean una secuencia para generar claves primarias basadas en números enteros y posteriormente define un disparador que establece la clave primaria de la tabla *Customers* (*Customer_ID*) a un valor obtenido de la secuencia. Este sistema evitará que los usuarios de la aplicación tenga que generar identificadores únicos cada vez que deseen crear un nuevo registro.

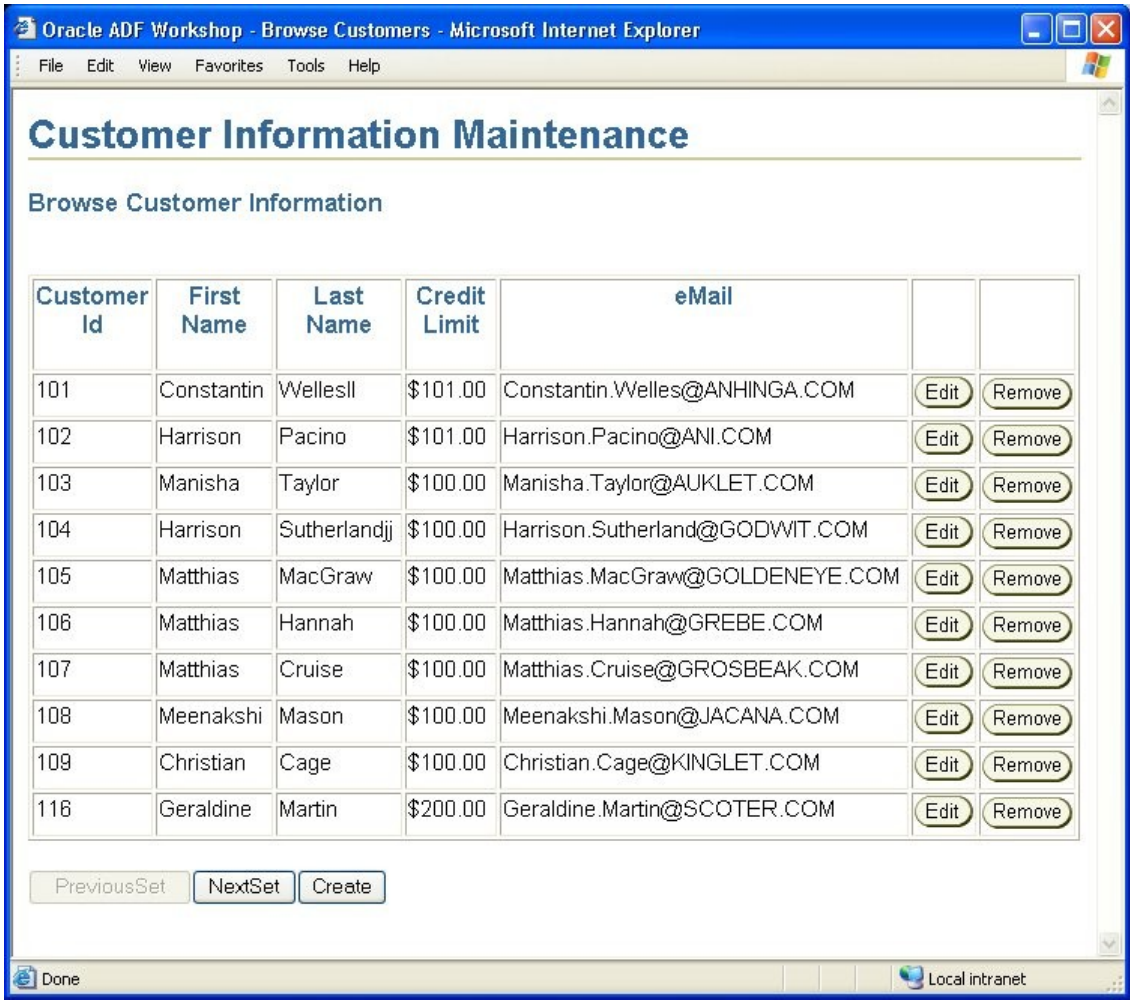
La tabla *Customers*, con la que trabajará la aplicación tiene la siguiente estructura:



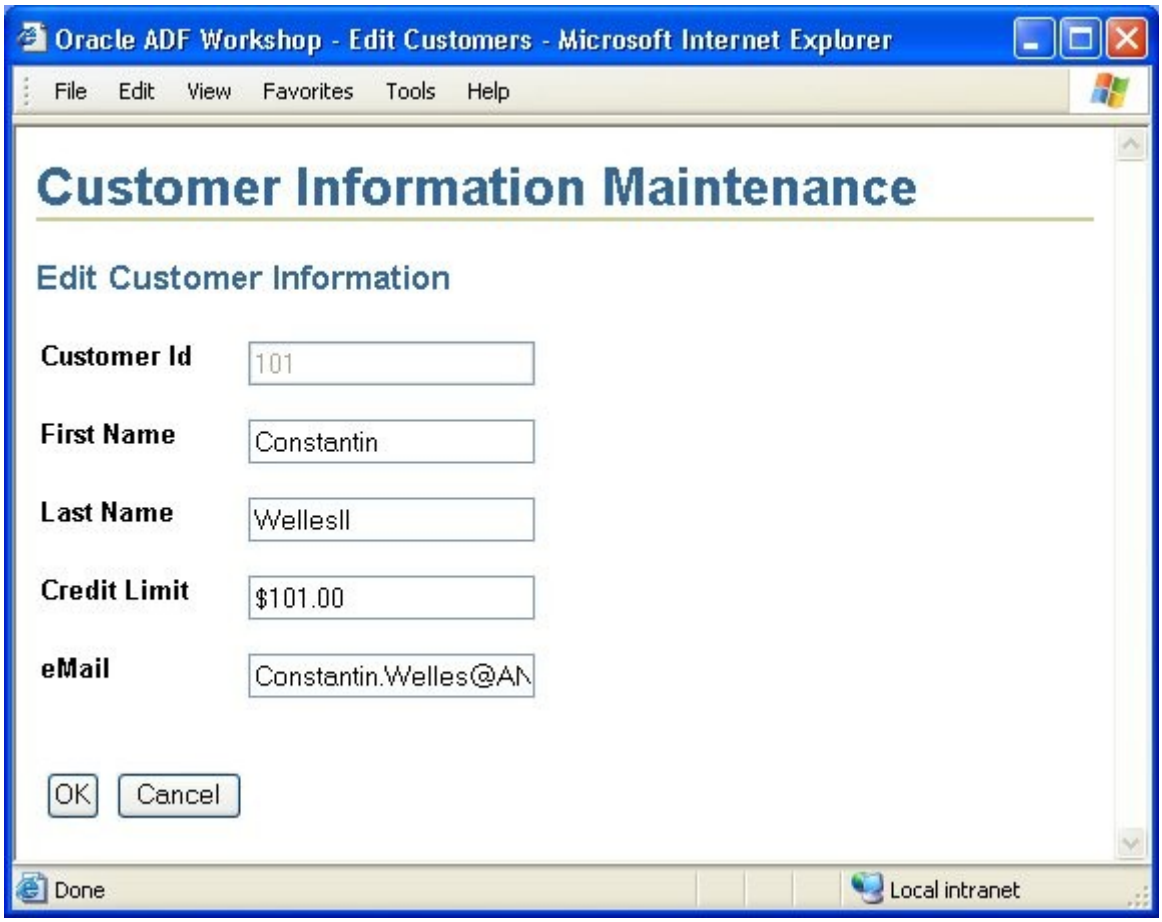
Descripción de la aplicación a desarrollar

Como ya se ha comentado, la aplicación proporcionará la funcionalidad necesaria para mantener la información de los clientes de una compañía. Esto implica crear funcionalidad para la creación, recuperación, actualización y borrado de registros de clientes. La aplicación dispondrá de cuatro páginas web:

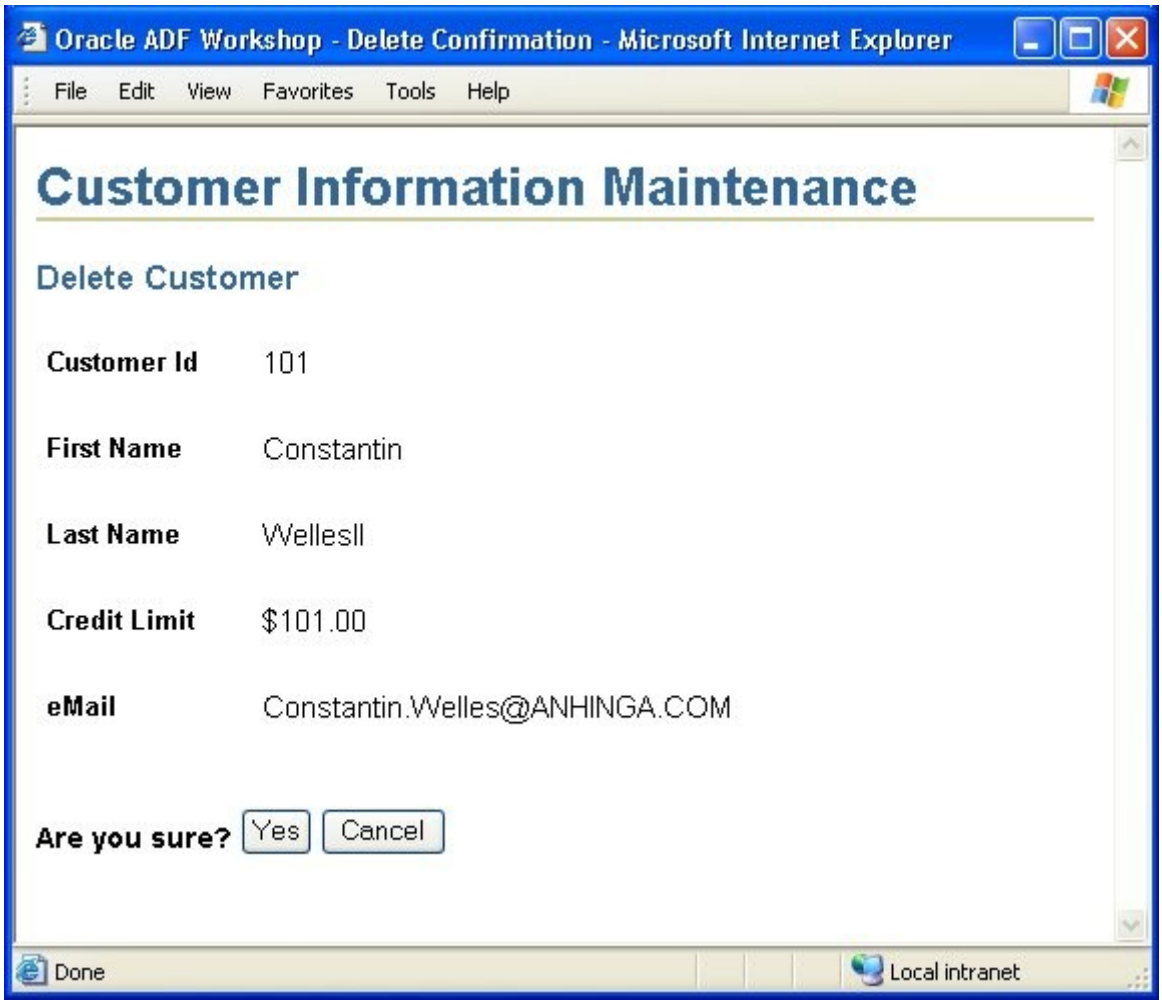
La página de visualización **"*Browse page*"** servirá como punto de partida de la aplicación y permitirá ver una lista de los registros de los clientes. El usuario podrá seleccionar un registro y editarlo o borrarlo. Esta página además proporciona un botón que permitirá al usuario crear un nuevo registro.



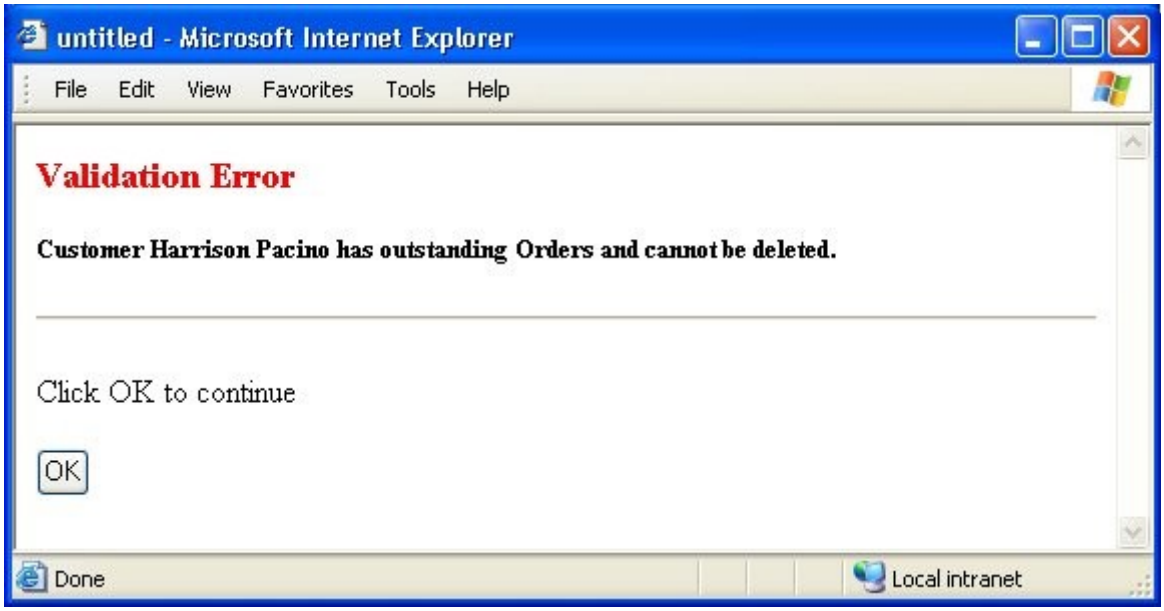
La página de edición **"*Edit page*"** ofrece al usuario todos los campos que éste puede modificar de un registro. Esta página también se empleará para crear nuevos registros. Cuando la página se use para crear un nuevo registro el identificador del cliente no se mostrará ya que este se genera a partir de una secuencia en la base de datos, para lo cual se debe de haber ejecutado el código de creación de la secuencia y el disparador que se incluía en la introducción.



La página de confirmación **"*Sure page*"** proporciona una oportunidad al usuario para confirmar su intención antes de que sea borrado un registro. Esta página muestra los datos del cliente que se ha pedido eliminar. El usuario podrá aceptar la operación de borrado o cancelar dicha operación. En ambos casos de deberá de mostrar un mensaje adecuado según la operación realizada cuando se vuelva la página principal.



La página de errores **"*Error page*"** mostrará cualquier mensaje de error que se genere durante el proceso de borrado.



Creación de servicios de negocio

La aplicación para el mantenimiento de los datos de clientes que vamos a desarrollar está basada en un modelo de datos bastante simple. Este modelo sólo incluye una tabla, la tabla *Customers* que se describió en la introducción.

Oracle ADF proporciona una infraestructura que hace que la creación y el manejo de la capa de persistencia de la aplicación sean sencillos. Esta infraestructura está compuesta por componentes de negocio de Oracle ADF (*Oracle ADF Business Components*). La infraestructura proporciona reglas de validación incorporadas y se adapta a las reglas personalizadas que se definan y a comportamientos estándar. En la mayor parte de los casos, estos comportamientos estándar son suficientes, y cuando éstos no sean capaces de satisfacer las necesidades del usuario se pueden anular y definir el comportamiento que se desee.

Debido a que en la aplicación que vamos a desarrollar emplearemos los comportamientos por defecto, podemos emplear el asistente para la creación de componentes de negocio de Oracle ADF para crear de forma automática los componentes que necesita la aplicación. En los próximos pasos se crearán los componentes de negocio necesarios para la aplicación.

Creación de un espacio de trabajo y los proyectos para la aplicación

Comenzaremos creando una nueva aplicación.

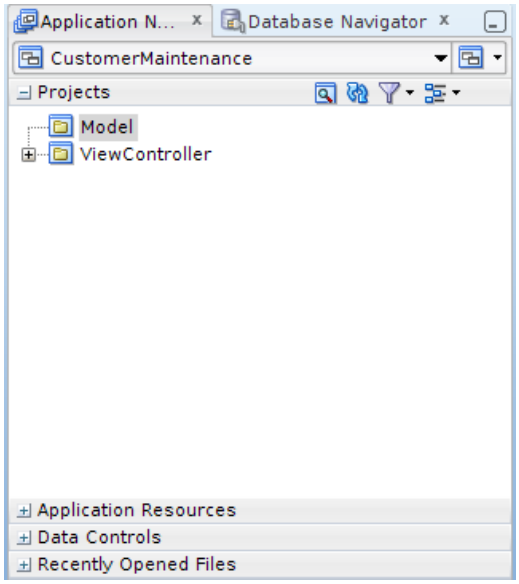
1. Cree una nueva aplicación que contenga los componentes de negocio.

    a. Seleccione en la pestaña de aplicaciones New Application... en la New Gallery seleccione ***General → Applications*** en el menú de la izquierda y seleccione el nodo ***Fusion Web Application (ADF)***.

    b. Indique como nombre de la aplicación *CustomerMaintenance* o el nombre que usted desee.

    c. Pulse ***Finish*** para terminar. Se crearán los elementos necesarios de forma automática.

2. Una vez terminado el asistente, debería de tener en el navegador una aplicación llamada ***CustomerMaintenance*** y dos proyectos llamados ***Model*** y ***ViewController*** respectivamente. En esta parte del tutorial sólo emplearemos el proyecto ***Model*** que contendrá los componentes de negocio.



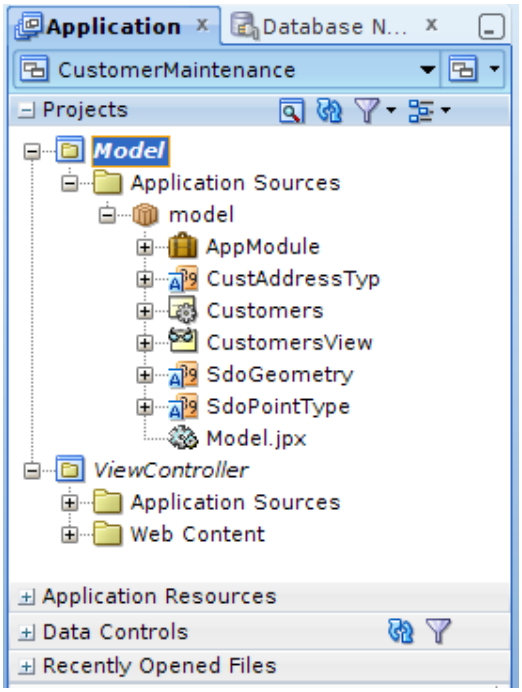
Creación de los componentes de negocio por defecto

1. Haga clic con el botón derecho el nodo del proyecto ***Model*** en el navegador y seleccione la opción ***New*** del menú contextual.

2. Seleccionaremos en la "***New Gallery***" la categoría ***Business Tier → ADF Business Components*** y el elemento ***Business Components from Tables*** de dicha categoría.

3. El primer paso del asistente que se nos muestra es establecer una conexión a la base de datos donde están las tablas con los datos que manejará la aplicación. Seleccione una conexión que conecte con el esquema en el que disponemos de la tabla ***Customers*** (normalmente el esquema OE). Para ello pulse sobre el icono de la lupa en la página del asistente y seleccione la conexión que tenga configurada y que contenga la tabla solicitada, pulse *Copy Connection*.

4. En los siguientes pasos del asistente seleccione la tabla *Customers* y cree los componentes de entidad y vista (*entity/view* objects) y el *módulo de aplicación por defecto*. Una vez finalizado el asistente el navegador de aplicaciones se muestra con el siguiente contenido:



Personalizando los componentes de vista

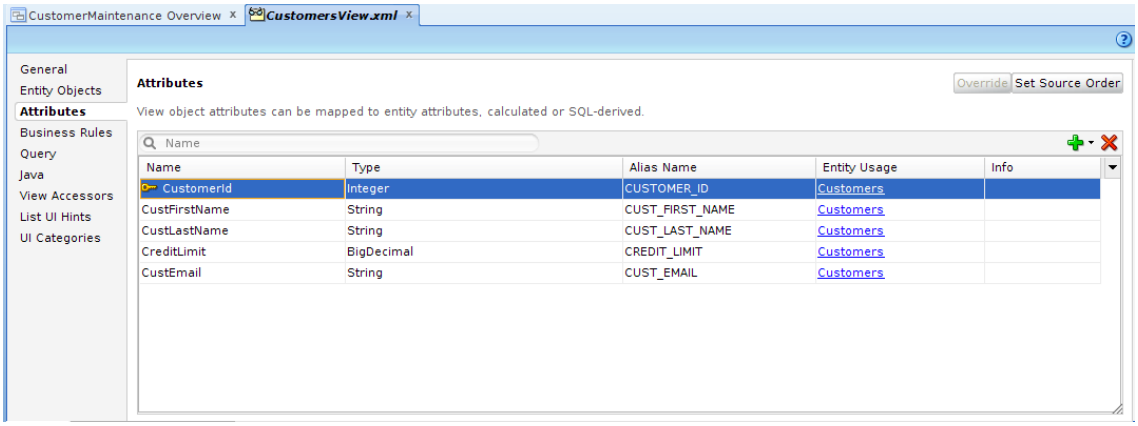
1. Como puede observar ha sido sencillo crear la capa de servicios de negocio empleada por defecto. Después de crear los componentes por defecto, suele ser necesario personalizar los componentes de vista para adaptarlos a las necesidades de la aplicación.

La aplicación, de todos los atributos de la tabla ***Customers***, sólo empleará los siguientes:

- CustomerId
- CustFirstName
- CustLastName
- CreditLimit
- CustEmail

Haga doble clic en el nodo ***CustomersView*** que representa el componente de vista de la tabla ***Customers***. En el editor de los objetos de vista seleccione en la parte izquierda la pestaña del nodo de los atributos (***Attributes***).

Modifique la lista de atributos seleccionados para que sólo queden en dicha lista los atributos mencionados anteriormente. Los atributos se eliminan seleccionándolos y luego pulsando sobre el icono del aspa roja. Una vez terminado, la lista de atributos seleccionados tiene este aspecto:



2. Haga clic en ***Guardar todo*** para aceptar los cambios realizados.

Probar los componentes de negocio

1. JDeveloper incluye un visor para componentes de negocio que permite realizar pruebas sobre dichos componentes sin necesidad de que exista todavía un interfaz gráfico para la aplicación. Para acceder al visor, haga clic con el botón derecho en el nodo del módulo de aplicación (***AppModule***), y seleccione la opción ***Run*** del menú principal.

2. El visor tomará por defecto la conexión de base de datos que indicamos para crear los componentes de negocio usando el asistente.

3. Una vez abierto el visor haga doble clic sobre el nodo del objeto de vista que hemos creado (***CustomersView1***) y observe los datos que ofrece los componentes que acaba de crear. Puede utilizar los iconos de la barra superior para moverse por los datos de la tabla, modificarlos, etc...

Creación de las páginas

A continuación vamos a crear las páginas y las conectaremos con los componentes de negocio que hemos definido anteriormente.

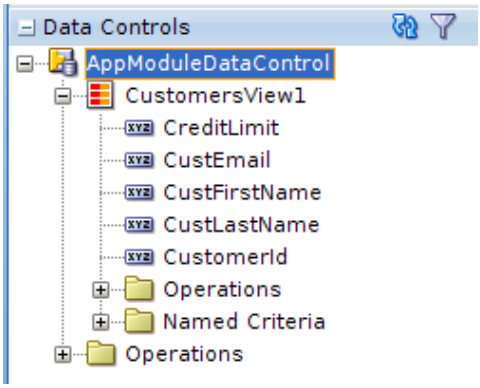
1. Comenzaremos creando la página web ***browseCustomers***. Pulse sobre el proyecto *ViewController* con el botón derecho del ratón muestre el menú contextual y pulse **New...** Esto abrirá la New Gallery, donde seleccionamos **Web Tier → *JSP***. Seleccionamos el elemento **JSP**. Llamamos a la página ***browseCustomers.jsp***.

Al terminar, disponemos de una página JSP en blanco llamada ***browseCustomers.jsp***. El siguiente paso incorporará en dicha página algunos componentes para el acceso a datos.

2. JDeveloper y la tecnología Oracle ADF hacen sencillo incorporar componentes para el acceso a datos en páginas web. En la ventana de la aplicación existen varias pestañas apilada

en la parte inferior. Una de esas pestañas es ***Data Controls***. Expandiendo esta pestaña se nos muestran componentes que nos permiten el acceso a datos.

Expanda la pestaña ***Data Controls*** y expanda el nodo ***AppModuleDataControl*** para ver los componentes de vista disponibles en la aplicación. En este caso solo hay un componente de vista llamado ***CustomersView1*** que creamos en pasos anteriores. Si expendimos este nodo se nos muestran cada uno de los controles de datos que tenemos disponibles:



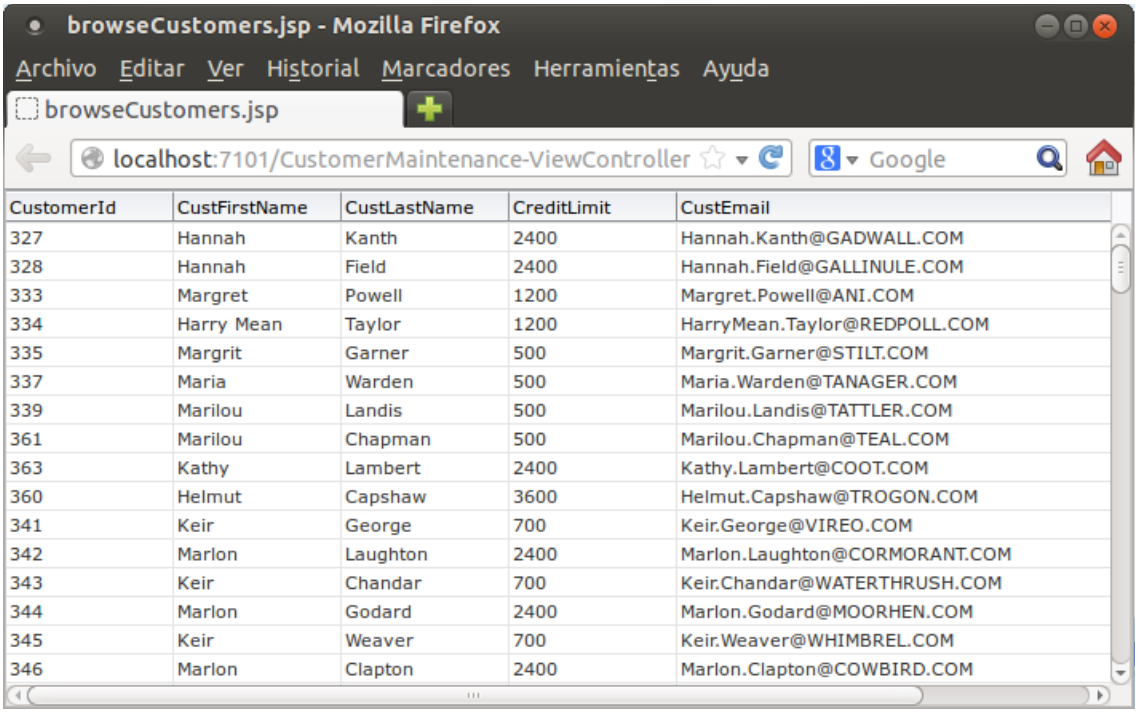
3. La página ***browseCustomers*** mostrará varios registros de clientes en forma de tabla. JDeveloper proporciona funciones que se podrán arrastrar en la página y nos permitirán crear el componente necesario para este tipo de acceso a datos.

Asegúrese de que está situado en la ventana de edición de la página ***browseCustormers.jsp***.

Seleccione el nodo ***CustomersView1*** de la paleta de componentes en su pestaña ***DataControl***. Arrastre el componente ***CustomersView1*** a la ventana de edición de la página ***browseCustomers.jsp***. Observe que aparece una lista desplegable. Esta lista muestra los distintos estilos que posee este componente. En este caso seleccionaremos el estilo ***Table → ADF Read-Only Table*** ya que deseamos crear una página web solo para mostrar información, no editarla. Indicamos que queremos que se pueda ordenar y filtrar marcando las casillas correspondientes. También indicamos que queremos que se pueda seleccionar una fila. Aceptamos el resto de la configuración por defecto, y aceptamos que cree un formulario alrededor del componente.

4. Una vez incluido el componente la página JSP mostrará los datos. Para probar el funcionamiento de dicha página haga clic con el botón derecho sobre el elemento componente ***browseCustomers.jsp*** y seleccione la opción ***Run*** del menú contextual.

Al ejecutarse, la página debe de mostrarse según la siguiente figura:



CustomerId	CustFirstName	CustLastName	CreditLimit	CustEmail
327	Hannah	Kanth	2400	Hannah.Kanth@GADWALL.COM
328	Hannah	Field	2400	Hannah.Field@GALLINULE.COM
333	Margret	Powell	1200	Margret.Powell@ANI.COM
334	Harry Mean	Taylor	1200	HarryMean.Taylor@REDPOLL.COM
335	Margrit	Garner	500	Margrit.Garner@STILT.COM
337	Maria	Warden	500	Maria.Warden@TANAGER.COM
339	Marilou	Landis	500	Marilou.Landis@TATTLER.COM
361	Marilou	Chapman	500	Marilou.Chapman@TEAL.COM
363	Kathy	Lambert	2400	Kathy.Lambert@COOT.COM
360	Helmut	Capshaw	3600	Helmut.Capshaw@TROGON.COM
341	Keir	George	700	Keir.George@VIREO.COM
342	Marlon	Laughton	2400	Marlon.Laughton@CORMORANT.COM
343	Keir	Chandar	700	Keir.Chandar@WATERTHRUSH.COM
344	Marlon	Godard	2400	Marlon.Godard@MOORHEN.COM
345	Keir	Weaver	700	Keir.Weaver@WHIMBREL.COM
346	Marlon	Clapton	2400	Marlon.Clapton@COWBIRD.COM

5. Ahora que disponemos de una página básica, añadiremos a la misma algunos botones para hacerla algo más útil. Los botones se añaden de la misma forma que hemos añadido el componente para mostrar los datos.

Expanda el nodo ***AppModuleDataControl*** en la pestaña ***Data Controls***. A continuación, expanda el nodo ***CustomersView1*** y finalmente el nodo ***Operations***. Arrastre la operación ***Previous Set*** a la página JSP (hasta el formulario) y seleccione el estilo ***ADF Buttom***. Reubique el componente en la página JSP justo debajo de la tabla de datos. Repita los pasos anteriores para añadir un componente ***Next Set***.

Una vez añadidos los botones, seleccione la etiqueta del formulario que se muestra como un recuadro punteado en naranja/rojo. Cambie el valor de la propiedad ***Action*** a ***browseCustomers.do***, empleando para ello el inspector de propiedades que se encuentra bajo el panel de componentes. Esta propiedad le indica a Struts que página se debe de ejecutar cuando se pulsa un botón del formulario, siendo esta página la que contiene la funcionalidad para ejecutar las acciones asociadas a los botones.

6. Ejecute la página JSP para probar el funcionamiento de los nuevos botones, siguiendo el mismo procedimiento que se ha indicado anteriormente.

Añadiendo componentes de datos a la página de edición de clientes

En este apartado vamos a añadir los componentes de acceso a datos para la página de edición de los datos de lo clientes ***editCustomers***. La única diferencia con la página de visualización de clientes será el componente de acceso a datos que va a ser empleado. En la página de visualización se empleó una tabla de solo lectura, y en este caso se empleará un formulario de entrada de datos.

1. El primero paso consiste en crear la página JSP correspondiente a ***editCustomers***. Se creará de forma similar a la página de visualización, pero indicando el nombre ***editCustomers.jsp***.

2. Para añadir un formulario de entrada de datos, seleccione el nodo **CustomersView1** de la paleta de componentes ***Data Control*** y arrástrelo a la página, selecciones de la lista ***Form → ADF Form***. Se nos pedirán los elementos que queremos que se incluyan en el Form, los dejamos todos.

3. Añada dos botones a la página. Estos botones será los componentes ***Commit*** y ***Rollback***.

Para añadir esos botones empelaremos las operaciones ***Commit*** y ***Rollback*** que cuelgan del nodo ***Operations*** que depende del nodo ***AppModuleDataControl***. Seleccione el componente ***Commit*** y arrastre dicho componente bajo el formulario de entrada de datos, asegúrese de que el estilo es ***ADF Button*** y que quede dentro de la etiqueta ***form*** que está marcada con un recuadro de línea punteada naranja/roja. Repita estos mismos pasos para el componente ***Rollback***.

4. Renombre los botones par hacer la página algo más amigable. La forma mas sencilla de cambiar la etiqueta de un botón es seleccionándolo y luego volviendo a hacer clic sobre dicho botón. Otra forma es cambiando la propiedad ***Text***, en el inspector de propiedades.

a. Cambie la etiqueta del botón ***Commit*** por ***OK***.

b. Cambie la etiqueta del botón ***Rollback*** por ***Cancel***.

5. Por defecto, el código que se crea para estos botones incluye una expresión que comprueba si el botón debería de estar o no activado. Debido a que deseamos que estos botones estén activos todo el tiempo, deberemos de eliminar el código de comprobación, que para los distintos botones es el siguiente:

Para el botón ***OK*** eliminar el código: ***disabled="#{!bindings.Commit.enabled}"***

Para el botón ***Cancel*** eliminar: ***disabled="#{!bindings.Rollback.enabled}"***

Otra forma de deshabilitar la activación del botón es pulsando sobre el botón, ir al inspector de propiedades y modificar la propiedad **Disabled**. Pulsamos la lista deplegable y elegimos Reset to Default, esto pondrá el valor a falso y activará el botón.

Si deseamos editar el código de los botones, podemos hacerlo seleccionando el botón y posteriormente haciendo clic en la pestaña ***Source*** de la ventana de edición de la página.

5. Pruebe el funcionamiento de la página con los nuevos botones. Recuerde que para probar una página debe de hacer clic con el botón derecho sobre la página y seleccionar la opción ***Run***.

Tenga en cuenta que en esta página no hay posibilidad de variar el registro que se está editando. Este hecho es deliberado ya que conectaremos la página de visualización con la página de edición del tal forma que sea en la página de visualización en la que seleccionamos el registro a editar.

Conexión de las páginas

Hasta el momento se ha creado un par de páginas que permiten en acceso a los mismos datos en diferentes formas. El siguiente paso consistirá en conectar estas dos páginas. La página ***browseCustomers*** servirá como punto de inicio de la aplicación para los usuarios. En dicha página el usuario seleccionará el registro que desea editar, y se abrirá la página de edición con los datos de dicho registro. La página de edición también servirá para insertar nuevos registros, permitiendo al usuario especificar el valor para los diferentes campos del registro a insertar. Para que la aplicación funcione de esta manera es necesario crear ciertas conexiones, llamadas ***forwards***, entre las dos páginas.

En este apartado se añadirán algunos botones a la página ***browseCustomers*** para que el usuario pueda editar o crear registros y se abra automáticamente la página ***editCustomers***. También se añadirán ***forwards*** asociados a los botones ***OK*** y ***Cancel*** de la página de edición para que una vez terminada la tarea se vuelva automáticamente a la página de visualización.

1. JDeveloper y Oracle ADF proporcionan cierta funcionalidad para permitir una navegación sencilla por los datos que se muestran en los componentes de datos. En los próximos pasos se añadirán algunos comportamientos por defecto, se probarán, y se modificarán para ajustarlos a las necesidades específicas de la aplicación.

La primera tarea será añadir una función que permita fijar como registro actual cualquier registro en el que haga clic el usuario. El modelo de componentes de negocio que se creó anteriormente sincronizará los datos sin necesidad de que sea necesario añadir código alguno por parte del desarrollador.

Para añadir esta función, siga los siguientes pasos:

a. Abra la ventana de edición de la página ***browseCustomers***.
b. Muévase a la parte derecha de la página.
c. Seleccione la última columna de la tabla.
d. Aparecerá una flecha en la parte superior, púlsela.
e. Seleccione la opción Inser ***Insert as Sibling → Column*** del menú contextual.
f. Repita la operación para insertar un total de dos columnas.

Esta dos columnas que acaba de crear contendrán los botones que permitirán borrar y editar los registros.

2. A continuación, añada el método ***setCurrentRowWithKey*** del modelo de datos como un enlace. Para ello:

a. Expanda el nodo ***CustomersView1*** del la paleta de controles de datos.
b. Expanda el nodo ***Operations***.
c. Seleccione el control ***setCurrentRowWithKey***.
d. Arrastre el componente a la primera columna que acaba de crear, soltándolo en la celda de la fila inferior de la tabla para esta columna, indique se muestre como un enlace ADF con imagen.

Ahora dispondrá de un enlace con el texto ***setCurrentRowWithKey*** en dicha columna. Este enlace no llevará al usuario a ninguna otra página todavía, pero permitirá seleccionar la fila haciendo clic sobre él. Podemos cambiar la etiqueta del enlace si modificamos la propiedad **Text** en el inspector de propiedades.

3. Ejecute la página para probar que el funcionamiento del nuevo componente. Al hacer clic sobre el debería de mostrase la fila en un color resaltado indicando el registro que se ha seleccionado.

4. Una vez comprobado el control para seleccionar el registro, modificaremos este enlace para que sea visualmente más agradable, sustituyendo el texto por un botón. Pulse sobre el enlace creado anteriormente, vaya al inspector de propiedades y modifique la propiedad **Icon**.

Añada el botón 

JDeveloper le preguntará si desea añadir la imagen a la aplicación. Acepte, y una copia de la imagen quedará incorporada al proyecto.

El resultado de esta operación tiene aproximadamente el siguiente aspecto:

5. A continuación, borre los fragmentos del texto que hay alrededor del botón. Puede hacerlo en la propiedad **Text** del inspector de propiedades.

6. Una vez que ya disponemos de botón para realizar la edición, debemos de añadir los enlaces entre páginas en el diagrama Struts.

Definición del flujo básico entre páginas

Una vez definido el modelo de servicios de negocio, la capa de acceso a los datos, es posible definir la estructura de flujo entre las páginas que forman parte de la aplicación. Desarrollar una aplicación resulta un proceso iterativo. Se desarrolla la capa de servicios de negocio, y posteriormente el flujo básico entre páginas. Una vez definida la aplicación de manera básica, volveremos a repetir el proceso de desarrollo para ir incluyendo en la aplicación nuevos elementos y cambios que la hagan más robusta.

Creación de componentes Datapage

Los components ***DataPage*** son los elementos básicos de una aplicación web que emplee la tecnología Oracle ADF. Las ***Datapage*** son páginas web que permitirán tratar datos procedentes de una base de datos. La primera tarea será crear la estructura básica de la aplicación creando un componente ***Datapage*** por cada una de las páginas de la aplicación. Estas páginas, como se comentó al inicio del tutorial, son las siguientes:

- Página de visualización "***Browse page***".
- Página de edición "***Edit page***".
- Página de confirmación "***Sure page***".
- Página de errores "***Errors page***".

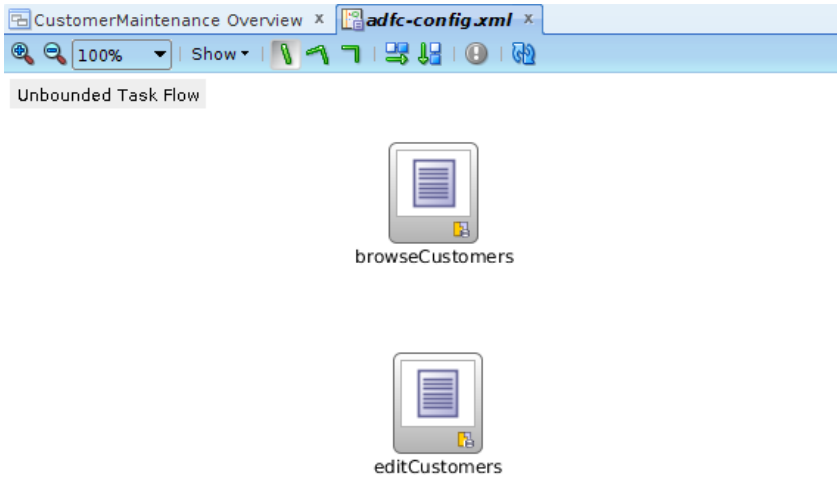
Para añadir estas ***DataPages*** empleando el diagrama ***Struts*** siga los siguientes pasos:

1. Abra el digrama ***Struts*** de la aplicación (si no está ya abierto) de la siguiente forma, haga doble clic sobre el elemento del proyecto ***ViewController*** → **Web content** → **Page Flows**.

2. Cree las ***DataPage*** arrastrando las páginas JSP al diagrama.

- ***browseCustomers*** (la página de visualización).
- ***editCustomers*** (la página de edición e inserción).

El diagrama debe de quedar de una forma similar a la siguiente:



El controlador de Struts gestiona la navegación entre páginas mediante eventos y ***forwards***. La notificación de eventos está implementada en JSP, mientras que los ***forwards*** se definen la configuración de Struts. El diagrama de flujo de páginas Struts gestiona automáticamente la gestión de la configuración de Struts por lo que no será necesario modificar el fichero directamente.

En los siguientes pasos añadiremos ***forwards*** en el diagrama de flujo entre páginas de Struts y modificaremos el enlace anterior para añadirle un evento.

8. Comencemos añadiendo los ***forwards*** en el diagrama de flujo de páginas. Abra dicho diagrama y seleccione el componente ***Control Flow Case*** de la paleta de componentes. Para definir un ***forward*** entre las páginas ***browseCustomers*** y ***editCustomers*** haga clic sobre el icono de la primera y posteriormente sobre el icono de la segunda. Entre esos dos clic puede hacer clic en otras partes del diagrama para definir cambios de dirección en la línea.

El nombre por defecto de los ***forwards*** es *******. Cámbielo a ***Edit***.

9. una vez que disponemos del ***forward Edit***, debemos de añadir un evento que lo invoque.

Para ello, seleccionaremos el botón ***Edit*** de la página **browseCustomers.jsp**, y examinaremos el código fuente que le corresponde. Vemos que ha definida una propiedad **actionListener** que indica las operaciones de datos a realizar. Debemos definir otra propiedad **action** que nos va a indicar la acción a realizar desde el punto de vista del flujo de control. Lo que debemos hacer es asociar el forward ***Edit*** que creamos antes con el botón. Esta asociación se puede hacer en el código fuente añadiendo al elmento del botón el siguiente código:

action=”Edit”

Quedando el código de dicho enlace de la siguiente forma:

```
<af:commandLink actionListener="#{bindings.setCurrentRowWithKey.execute}"
                text="Select"  action="Edit"/>
```

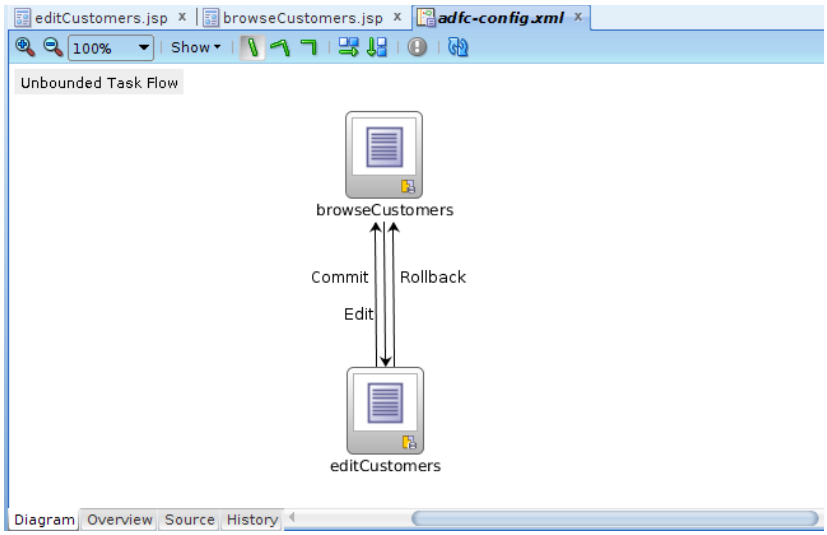
Otra forma de hacer esto, es seleccionar el botón y en el inspector de propiedades nos vamos a la propiedad **Action** y seleccionamos del desplegable **Edit**. Vemos que al hacer esto se genera el código que antes habíamos incluido a mano.

10. Al añadir esta referencia al evento, Struts llamará a la función ***setCurrentRowWithKey***, que establecerá el registro actual, y posteriormente aplicará el ***forward Edit*** transportando al usuario a la página ***editCustomers***.

De la misma manera que se ha hecho anteriormente, añada dos ***forwards*** desde la página ***editCustomers*** a la página ***browseCustomers*** con los nombres ***Commit*** y ***Rollback***.

Haga que el código de los botones ***OK*** y ***Cancel*** de la página de edición lancen un evento para aplicar los ***forwards Commit*** y ***Rollback*** respectivamente.

El diagrama Struts debe de tener el siguiente aspecto:



11. Pruebe la aplicación. Ejecute la página ***browseCustomers*** y compruebe que al pulsar el botón ***Edit***, se selecciona el registro y se abre la página de edición. De la misma forma compruebe el funcionamiento de los botones de la página edición, comprobando que al terminar su función se vuelve a la página de visualización.

Añadir un función para crear registros

Hasta el momento, han sido creadas un par de páginas enlazadas mediante ***forwards*** de Struts. Esas páginas permiten visualizar los datos de los clientes y editar los mismos. El siguiente paso será añadir a la aplicación la posibilidad de crear nuevos registros de clientes. Una vez más, JDeveloper combinado con Oracle ADF incluyen funcionalidades que hacen que esta tarea resulte sencilla.

1. Abra la ventana de edición de la página ***browseCustomers.jsp***. Seleccione el componente ***Create*** de la paleta de componentes de datos. Arrástrelo a la ventana de edición y suéltelo en la misma etiqueta de formulario con los botones ***Previous Set*** y ***Next Set***.

2. Ejecute la página para probar el funcionamiento del nuevo botón **Insert**. Comprobará que al pulsar el botón se inserta en el listado un registro en blanco. La inserción funciona, pero como no se pueden modificar los datos del nuevo registro, sirve de poco.

Lo que se busca precisamente es que ocurran dos cosas: que se inserte el nuevo registro en blanco, y que se vaya automáticamente a la página de edición para añadir los valores a cada uno de los campos.

Para provocar el cambio automático de página podemos emplear Struts. Añada un ***forward*** en el diagrama de flujo de páginas entre la página ***browseCustomers*** y ***editCustomers*** y cambie su nombre a ***Create***. Asocie este ***forward*** al botón ***Insert***.

3. Ahora ya la aplicación nos permite crear un nuevo registro y editar los datos del mismo. Sin embargo, falta un detalle: deseamos que la clave primaria de la tabla, ***customerId***, tenga un valor obtenido de una secuencia en la base de datos. Esto evitaría a los usuario tener que buscar una clave primaria única cada vez que desean insertar un nuevo registro.

Como se han empleado los componentes de negocio de Oracle para crear el modelo de datos, esta tarea es bastante sencilla.

Expanda el nodo ***Model*** en el navegador de aplicaciones. Colgando de ese nodo, expanda el nodo ***Application Sources*** y posteriormente el nodo ***model***. Haga doble clic en el nodo que representa el objeto de entidad ***Customers***.

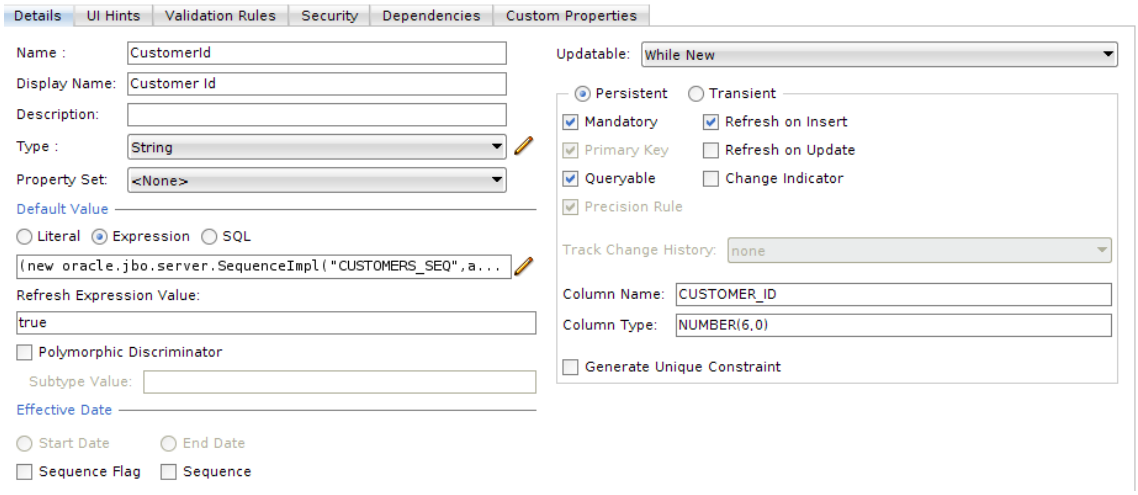
Lo que vamos a hacer, es que cuando se vaya a insertar un nuevo registro, copiaremos en el campo de la clave el valor que nos diga la secuencia ***CUSTOMERS_SEQ*** que creamos al principio en la base de datos.

En el cuadro de edición de objetos de entidad expanda el nodo ***Attributes*** y seleccione el atributo ***CustomerId***. Cambie el tipo del atributo (propiedad ***Type***) a ***String***. Pulse botón derecho sobre **CustomerId** y seleccione ***Change Type...*** Compruebe que la opción ***Updateable*** está marcada a ***While New*** (esto permitirá introducir un valor diferente al usuario, si así lo desea), y marque las opciones ***Persistent***, ***Mandatory***, ***Queryable***, ***Refresh on Insert.***

Hasta el momento le hemos dicho que va a ser una cadena, y cómo debe comportarse el campo, pero no le hemos indicado de dónde debe tomar los datos de la secuencia. Para hacer esto, en la sección de valores por defecto seleccionamos Expression, e introducimos el siguiente código:

(new oracle.jbo.server.SequenceImpl("CUSTOMERS_SEQ",adf.object.getDBTransaction())).getSequenceNumber()

Con esto accedemos a la secuencia en la base de datos y cargamos el valor en un objeto Java, que luego mostraremos en el campo de la clave. Al finalizar la configuración del atributo debe quedar de la siguiente forma:



4. Pruebe la nueva funcionalidad ejecutando la página ***browseCustomers.jsp*** y creando un nuevo registro. Observe que al crear una fila el campo ***CustomerId*** está relleno.

Posteriormente en este tutorial cambiaremos las propiedades de visualización de este campo para que se muestre de forma más amigable.

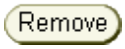
Añadir la funcionalidad de borrado con una página de confirmación

En este punto añadiremos a la aplicación la capacidad de borrar registros. Esta capacidad se puede añadir de una forma muy similar a la que se empleó para añadir la funcionalidad de crear registros, pero el comportamiento por defecto no permitiría a los usuarios confirmar el borrado. La mayoría de las aplicaciones requieren dar al usuario una oportunidad para confirmar un borrado antes de hacer este definitivo, y esta aplicación no será una excepción.

La tarea en este punto se dividirá en añadir un enlace para el borrado de registros, y crear una página de confirmación.

Añadir un enlace para eliminar registros

El primer paso sera crear un botón en la página ***browseCustomers.jsp***. Este botón tendrá el siguiente aspecto:



Esta imagen tendrá el aspecto de un botón pero en realidad no eliminará un registro. Su funcionamiento será similar al del botón ***Edit***, es decir, establecerá el registro al que está asociado el botón como registros actual. Una vez hecho esto, se enviará al usuario a la página de confirmación, que será la que realmente realice el borrado del registro.

1. Añada el botón ***Remove*** de la misma forma que añadió el botón ***Edit*** anteriormente en este tutorial:

    a. Abra la ventana de edición de la página ***browseCustormer.jsp***.

b. Arrastre la operación ***setCurrentRowWithKey*** a la última columna de la tabla en la celda de la última fila de la misma. Indique que desea crear un enlace ADF con imagen.

c. Añada la imagen del botón ***Remove*** a través del inspector de propiedades, en la propiedad ***Icon.*** JDeveloper le preguntará si desea incorporar la imagen al directorio de la aplicación. Acepte para que la imagen quedará incorporada.

El resultado de los pasos anteriores debe tener un aspecto similar a este:

2. Puede borrar el texto del enlace cambiando la propiedad ***Text*** a vacío.

3. Ahora que el botón que acabamos de insertar establece el registro al que está asociado como registro actual, es necesario añadir un evento al enlace del botón para que cuando el usuario haga clic sobre él Struts lo envíe a la página de confirmación.

4. Tal y como hicimos con el botón de edición debemos crear un ***forward***. Este ***forward*** se dirigirá a la página de confirmación. Por tanto primero debemos crear dicha página.

Añadir la funcionalidad de borrado con una página de confirmación

La siguiente tarea consiste en crear una página de confirmación e incorporar ésta al flujo de páginas de la aplicación. Para ello, siga los siguientes pasos:

1. La página de confirmación, ***sure.jsp***, será una página en la que se mostrarán los datos del registro que se está a punto de borrar. Esta página estará compuesta por un formulario de solo lectura basado en el objeto de vista **CustomersView1**.

Cree esta página de la misma forma en que se creó la página edición, pero indicando que se cree como ***ADF Read-Only Form***.

2. La página necesita dos botones más para estar completa, uno de confirmación y otro de cancelación, que permitan al usuario indicar si realmente se desea borrar el registro.

El botón de confirmación será un botón enlazado con los datos que los borrará. Aunque en realidad el botón no confirmará nada si no que borrará el registro, cambiaremos la etiqueta a “Yes” para que el usuario tenga la sensación de estar confirmando la operación.

Para añadir este botón seleccione del nodo ***Operations*** que cuelga del nodo ***CustomersView1***, en la paleta de componentes de datos, la operación ***Delete***. Arrastre el componente y suéltelo dentro del elemento de formulario como un botón ADF. Posteriormente cambie su propiedad ***Text*** a “Yes”. Cambie la propiedad ***Disabled*** para que esté activado por defecto.

A continuación, añadiremos un botón de cancelación. Una manera muy sencilla de crear un nuevo botón será copiar y pegar el botón que acabamos de crear. Después de haberlo pegado, haga doble clic en el botón para abrir el cuadro de diálogo de edición. Cambie el valor a ***Cancel*** para que se muestre esa etiqueta en el botón. Elimine el valor de la propiedad ***ActionListener***, porque no queremos que este botón borre nada.

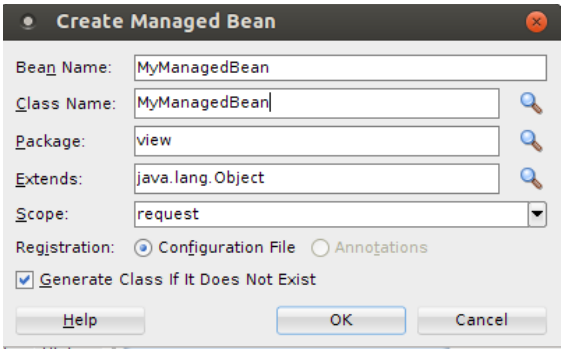
Para terminar añada sobre los botones que acaba de crear algún texto para orientar al usuario, como por ejemplo “Are you sure?”. Establezca el estilo del texto a Heading 4.

3. Además de realizar la operación de borrado, se han de cometer los cambios para que le borrado sea efectivo. Debido a esto debe de añadir una ligadura a acción llamada (action binding) ‘Commit’ que está asociada al la operación que comete los cambios en nuestro modelo de interfaz de usuario UI model.

Para añadir esta acción:

a. Con la página JSP de confirmación abierta y en la vista de diseño, hacemos doble clic sobre el botón. Aparece el dialogo **Bind Action Property**. Este diálogo nos permitirá crear un ManagedBean, una clase java que se encargará de las acciones que deben realizar los elementos activos de la página. Pulsamos ***New...*** para crear un nuevo Bean.

b. Vamos a crear un nuevo Bean de nombre **MyManagedBean** que almacenaremos en el paquete **view**. Rellenamos la pantalla para que quede como en la figura y aceptamos.



c. Ahora que hemos creado el Bean debemos introducir el nombre del método que se va a ejecutar cuando pulsemos el botón, le damos el nombre de **deleteAndCommit**.

d. Ahora se ha generado el siguiente código para el método:

```
public String deleteAndCommit() {
        BindingContainer bindings = getBindings();
        OperationBinding operationBinding = bindings.getOperationBinding("Delete");
        Object result = operationBinding.execute();
        if (!operationBinding.getErrors().isEmpty()) {
            return null;
        }
        return null;
}
```

El código indica que se va a tomar la operación de borrado y se va a ejecutar. Luego se comprobará si ha habido errores.

e. En la página sure.jsp pulsamos en la pestaña bindings y creamos el binding commit. Pulsamos en la cruz verde, para crear una nueva acción ***action***. Seleccionamos **AppModuleDataControl** y en el desplegable la operación ***Commit****, y aceptamos.*

f. Ahora debemos acceder al binding desde el código que se había generado para el botón. Para ello actualizamos el código para que quede de la siguiente forma:

```
public String deleteAndCommit() {
    BindingContainer bindings = getBindings();
    OperationBinding operationBinding = bindings.getOperationBinding("Delete");
    Object result = operationBinding.execute();
    if (!operationBinding.getErrors().isEmpty()) {
       return null;
    }
    OperationBinding commitBinding = bindings.getOperationBinding("Commit");
    Object commitResult = commitBinding.execute();
    if (!commitBinding.getErrors().isEmpty()) {
       return null;
    }
    return "Sure";
}
```

Se han resaltado en negrita las nuevas líneas que se han añadido al código. Ese código llama al *binding* de la operación *commit* que hemos definido y la ejecuta, comprueba si ha habido errores, y si todo va bien, devuelve *"Sure"*. Recordemos que mediante el atributo ***action*** podíamos decirle al botón el forward que debía seguir al terminar la operación. Sin embargo ahora ese atributo para el botón de confirmación tiene el siguiente código:

```
<af:commandButton text="Yes" action="#{MyManagedBean.deleteAndCommit}"/>
```

Vemos como ahora, en dicho atributo se realiza una llamada al *bean* que va a manejar el comportamiento del botón. Para poder seguir usando los ***forward*** el método que maneja el comportamiento del botón debe devolver el nombre del ***forward*** que se debe seguir tras realizar la acción. Vemos como de esta forma podemos seguir un ***forward*** si la operación ha tenido éxito y otro distinto si no ha ocurrido así.

4. Pruebe la página para comprobar el funcionamiento de la confirmación del borrado. Recuerde que todo registro que borre no podrá ser recuperado.

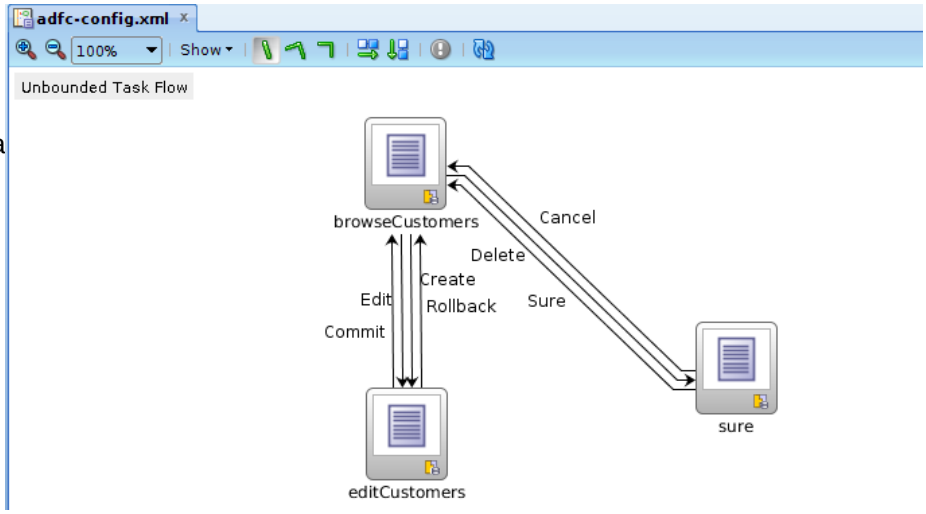
5. Ahora que la página de confirmación borra los registros, solo queda incorporarla al flujo de páginas para que se vuelva automáticamente a la página de visualización.
Abra el diagrama de flujo de páginas (Struts), arrastre ***sure.jsp*** para crear la página y añada un ***forward*** llamado ***Delete*** de la página ***browseCustomers*** a la página ***sure***. A continuación añada el ***forward*** llamado ***Sure*** de la página ***sure*** (el cual usamos desde el *ManagedBean*) a la página ***browseCustomers***.

El primer ***forward*** que ha añadido llevará automáticamente a la página de confirmación cuando se pulse sobre el botón borrar de la página de visualización. El segundo, llevará automáticamente de la página de confirmación a la de visualización en caso de que se confirme un borrado.

Por último, falta añadir un ***forward*** llamado ***Cancel*** que irá desde la página ***sure*** a la página ***browseCustomers***. Este ***forward*** llevará al usuario automáticamente desde la página de confirmación a la página de visualización en caso de que se haya cancelado el borrado.

6. Una vez terminado, el diagrama de flujo entre páginas debería de tener el siguiente aspecto:

7. Ahora



debemos de colocar el correspondiente ***forward*** a cada botón editando la propiedad Action y seleccionando del desplegable la que corresponda.

8. Ejecute la aplicación desde la página ***browseCustomers***. Ahora, con la funcionalidad que se ha ido añadiendo, ya puede editar registros, crear nuevos registros y eliminar registros.
Además, con la página de confirmación el usuario tiene la oportunidad de corregir un borrado accidental.

Creación y empleo de mensajes de usuario

La aplicación que hemos creado, cubre hasta ahora toda la funcionalidad básica que necesitamos. A continuación añadiremos a ésta algunos mensajes que ayuden a los usuarios a conocer el estado de las acciones que han realizado. Por ejemplo, si el usuario borra un registro, se indicará al usuario el registro que ha borrado mediante un mensaje. Cuando el usuario cree un nuevo registro, se le indicará que la inserción se ha completado con éxito. De la misma manera, si el usuario modifica los datos de un registro se le mostrará un mensaje apropiado.

Adición de código en la página browseCustomers para mostrar los mensajes de usuario

En el paso anterior se ha añadido a la página ***editCustormers*** el código que genera de forma adecuada al tipo de transacción los mensajes que se mostrarán al usuario. Dichos mensajes, se mostrarán cuando se finalice la transacción, cuando se vuelva a la página ***browseCustomers***. Por tanto, se deberá a añadir a esta última página la funcionalidad necesaria para que se muestren dichos mensajes. Para ello, siga los siguientes pasos:

1. Una de las ventajas de Struts es que proporciona etiquetas para incluir en páginas JSP que hacen fácil la ejecución de operaciones comunes. En esta parta del tutorial se añadirán etiquetas para recuperar y mostrar los mensajes para el usuario que se generaron en páginas anteriores. Estas etiquetas recuperarán e interpretarán los mensajes almacenados en un objeto de la clase **FacesContext**. Para comenzar abra la página ***browseCustomers*** en el editor visual.

2. Abra la paleta de componentes, seleccione la lista ***ADF Faces***, luego expanda **Text and Selection** y arrastre el elemento ***Messages*** a la posición de la página donde desea que se muestren los mensajes. Por defecto los mensajes se mostrarán en una ventana emergente,

para que se muestren dentro de la propia página, seleccionamos el elemento ***Messages*** in la paleta de propiedades cambiamos el atributo ***Inline*** y lo ponemos a verdadero.

Habilitación de la página de confirmación de borrado

En pasos anteriores se ha creado la página de confirmación de borrado, permitiendo al usuario pulsar sobre los botones ***Yes*** y ***Cancel*** según quisiese o no confirmar el borrado de un registro. Para hacer la aplicación más completa, deberemos de añadir el código necesario para que desde esta página también se creen mensajes para informar al usuario del resultado de la operación.

En las próximas tareas añadiremos el código necesario para aumentar la función ***delete*** de tal forma que genere un mensaje y que cometa la transacción. Adicionalmente, añadiremos también código para generar un mensaje cuando la transacción se cancele.

Aumentar la función Delete

En pasos anteriores se añadió a la página de confirmación un botón etiquetado como ***Yes***. Este botón simplemente realizaba una llamada la función de borrado, ***delete***, incluida por defecto en este tipo de páginas. Sería deseable que esta página genere, al igual que la página de edición, mensajes para informar al usuario sobre el resultado de sus operaciones. Para ello, siga los siguientes pasos:

1. El código que vamos a añadir debe de ir asociado la página ***sure.jsp***. Como en apartados anteriores ya definimos el ManagedBean ***MyManagedBean***, asociado al botón ***Yes*** de la página, podemos incluir el nuevo código en el método ***deleteAndCommit*** que ya habíamos definido.

2. Lo primero que vamos a hacer es crear un objeto de tipo FacesContext que nos permita enviar mensajes. Luego creamos el mensaje, e indicamos que tipo de mensaje va a ser, error, alerta, en nuestro caso mostraremos un aviso informativo. Finalmente añadimos el mensaje a una cola en el contexto, y éste se mostrará en cuanto el control llegue a una página que contenga un objeto Faces para mostrar mensajes. Incluimos el siguiente código en el método ***deleteAndCommit***, usando la ayudas que proporciona JDeveloper importamos las clases que nos falten.

```
FacesContext fctx = FacesContext.getCurrentInstance();
FacesMessage msg = new FacesMessage("The row has been deleted.");
msg.setSeverity(FacesMessage.SEVERITY_INFO);
fctx.addMessage(null, msg);
```

2. Como este método eliminará el registro marcado, la primera tarea será obtener el nombre y apellido del cliente para poder generar el mensaje que se mostrará al usuario. Esta tarea se lleva a cabo de la misma forma que se hizo en la clase edtiCustomerAction en el método XXX. Copie el siguiente código antes del que borra el registro. Importe los paquetes necesarios:

```
BindingContext bctx = BindingContext.getCurrent();
BindingContainer custBindings = BindingContext.getCurrent().getCurrentBindingsEntry();

ControlBinding binding = custBindings.getControlBinding("CustFirstName");
String firstName = (binding != null) ? binding.toString() : "";

binding = custBindings.getControlBinding("CustLastName");
```

```
String lastName = (binding != null) ? binding.toString() : "";
```

4. Una vez que hemos recuperado el nombre y apellido del cliente, se puede ejecutar la función de borrado tal y como habíamos hecho anteriormente. El código de la función quedaría:

```
public String deleteAndCommit() {

    BindingContext bctx = BindingContext.getCurrent();
    BindingContainer custBindings = BindingContext.getCurrent().getCurrentBindingsEntry();

    ControlBinding binding = custBindings.getControlBinding("CustFirstName");

    String firstName = (binding != null) ? binding.toString() : "";
    binding = custBindings.getControlBinding("CustLastName");

    String lastName = (binding != null) ? binding.toString() : "";

    BindingContainer bindings = getBindings();
    OperationBinding operationBinding = bindings.getOperationBinding("Delete");
    Object result = operationBinding.execute();
    if (!operationBinding.getErrors().isEmpty()) {
        return null;
    }
    OperationBinding commitBinding = bindings.getOperationBinding("Commit");
    Object commitResult = commitBinding.execute();
    if (!commitBinding.getErrors().isEmpty()) {
        return null;
    }

    FacesContext fctx = FacesContext.getCurrentInstance();

    FacesMessage msg=new FacesMessage("Customer " + firstName + " " + lastName + " deleted.");
    msg.setSeverity(FacesMessage.SEVERITY_INFO);
    fctx.addMessage(null, msg);

    return "Sure";
}
```

5. Se ha de comprobar si ha ocurrido algún error durante el borrado, y en función de esto crear el mensaje adecuado.

6. Ejecute la aplicación y compruebe los mensajes que se muestran cuando se borra el registro de un cliente.

Ejercicio

Cree mensajes de confirmación para las operaciones de inserción y modificación.