Academia Omnis

Documentación para acompañar la creación de la aplicación Cucina Piccola de la Academia Omnis.

Revisión 1.1

Créditos y derechos de autor

Autor: Andreas Pfeiffer
Traducción al español: Carlos Marcelo Vallejo

Registro de revisiones

Revisión	**Descripción**
1.0 Feb 2018	Versión inicial producida por Andreas Pfeiffer
1.1 Ene 2019	Actualización para Omnis Studio 10 por Andreas Pfeiffer

Tabla de contenido

Parte 1: primeros pasos y creación de la forma remota para personal de servicio

Instalación de la versión de desarrollo de Omnis Studio

Omnis Studio es una herramienta para el desarrollo de aplicaciones web y móviles. No necesita otras herramientas. Puede descargar la versión de desarrollo de Omnis Studio en http://www.omnis.net/developers/download

Allí está la última versión para Windows y Mac OS. Simplemente descargue el instalador e instale Omnis Studio. Puede obtener un número de serie de evaluación, de prueba, en: https://www.omnis.net/developers/download/free-trial/.

La primera vez que inicie Omnis se le solicitará que ingrese el número de serie.

Estructura de las carpetas de Omnis Studio

La instalación en ambas plataformas (Windows y Mac OS) se divide en dos partes: directorio del programa y directorio de datos. Este último contiene todos los archivos para los que Omnis necesita tener permisos de escritura.

El directorio de datos se lo puede encontrar en:

- Windows: c:\users\%USERNAME%\AppData\Local\Omnis Software\OSx.x
  Nota: es posible que deba ingresar el texto "AppData" porque la carpeta podría estar oculta.
- Mac OS: esta carpeta también está oculta. Abra el buscador y haga clic en el menú "Goto". Luego presione la Tecla "opción". Esto debería hacer que el elemento "Library" aparezca en el menú Goto. Haga clic en Library - Application support - Omnis y luego en la instalación de Omnis Studio.

Hay una serie de carpetas de apoyo allí. Estas son solo las más importantes:

- Studio: contiene librerías para el Omnis Studio IDE y otros archivos del IDE, por ejemplo: config.json contiene algunas configuraciones básicas de Omnis Studio. El omnis.cfg contiene el número de serie que ingresó, así como la posición de las ventanas del IDE.
- Startup: contiene las librerías que se iniciarán automáticamente cuando inicie Omnis. Contiene una cantidad de librerías adicionales para el IDE. Puede usted poner su librería en esta carpeta en el Servidor de aplicaciones de Omnis Studio para que se inicie automáticamente.
- Html: es la carpeta más importante para el desarrollo web y móvil en Omnis Studio. Usted deberá poner esta carpeta y sus subcarpetas en la carpeta htdocs (si usa Apache como servidor web) o inetpub (si usa IIS) cuando vaya a implementar su aplicación utilizando un servidor web más adelante. También contiene la carpeta de imágenes en la que puede poner las imágenes utilizadas por su aplicación.

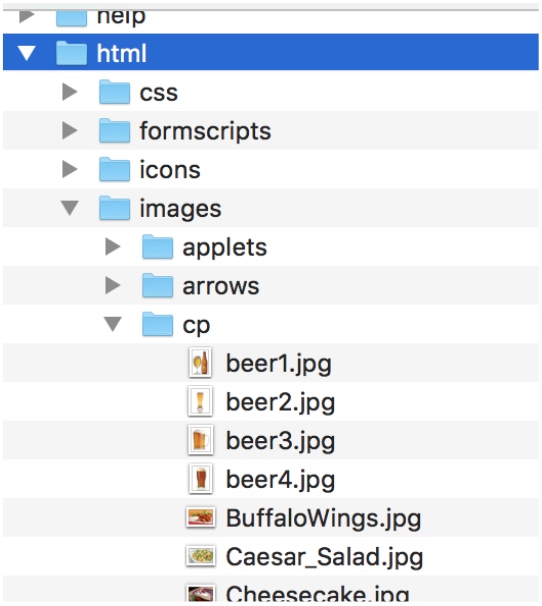
Archivos de ejemplo

Descomprima el archivo cucina_piccola.zip. Contiene lo siguiente:

- Carpeta CP: contiene todas las imágenes que se requieren para esta aplicación.
- Cp_final.lbs: versión final del proyecto
- Resource.lbs: librería de Omnis que contiene código de ejemplo que se copiará posteriormente durante el curso.
- Cucina_piccola.db y Cucina_piccola.db-journal: base de datos SQLite con datos de ejemplo

Ejercicio 1 - Copiar imágenes

- Copie la carpeta CP que contiene todas las imágenes de Cucina Piccola en la carpeta /html/images de su instalación de Omnis en el directorio de datos.



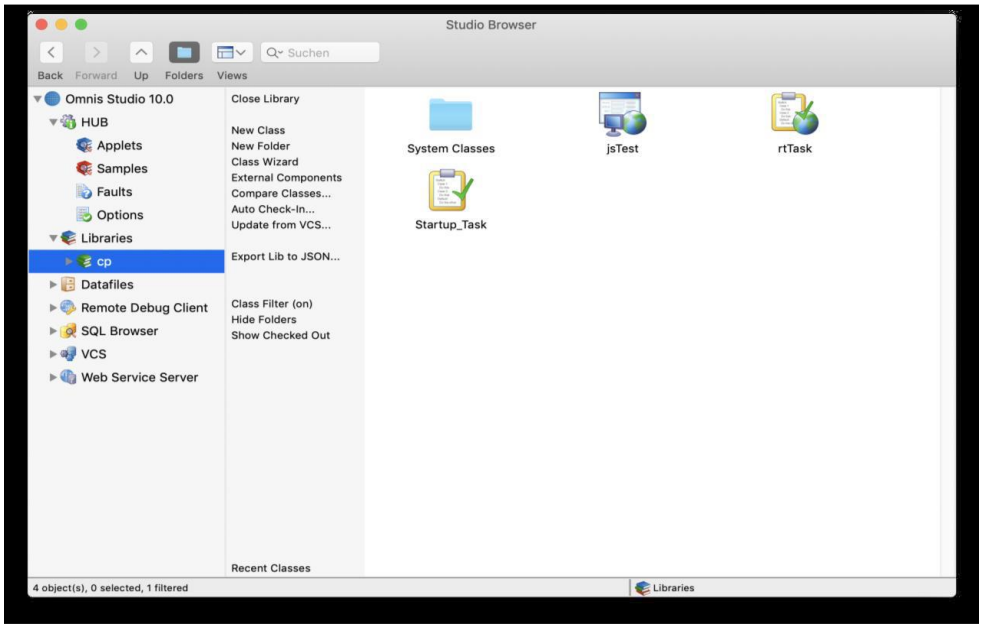
Estructura de carpetas

Ejercicio 2: iniciar Omnis y crear un proyecto

1. Inicie la versión de desarrollo de Omnis Studio. Favor ingresar su nombre, el nombre de su compañía y el número de serie que obtuvo para Omnis, cuando se lo soliciten.
2. Debería ver la ventana del **Studio Browser** (En algunas versiones de Omnis puede aparecer una pantalla de bienvenida - solo ciérrela).
3. Haga clic en la opción **librerías** de la lista de árbol y luego en **Nueva librería** a la derecha de la lista de árboles y finalmente en **Librería en blanco** en la ventana del asistente.
4. Aparecerá un cuadro de diálogo de archivo y debe navegar a su carpeta de ejercicios de Cucina Piccola (donde están los archivos cucina_piccola.db) y dele el nombre **cp.lbs** a su nueva librería.
5. Esto creará un nuevo proyecto de Omnis - llamado "librería" y también abrirá la librería para que usted la pueda ver en la ventana del **Studio Browser** en el lado izquierdo de la lista de árbol.

Studio Browser

★ Puede usar F2 (Windows) o Cmd-2 (Mac OS) para abrir rápidamente el Studio Browser si por alguna razón se ha cerrado.

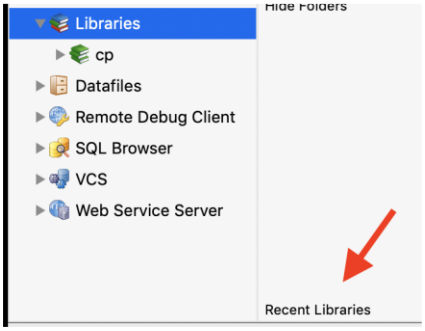


El Omnis Studio Browser (F2)

El Studio Browser le permite mantener sus librerías y las clases dentro de las librerías. La sección central (al lado del árbol) le permite crear nuevas clases y carpetas cuando está seleccionada una librería en el árbol.

Esta sección central cambia sus enlaces en función del objeto seleccionado en el árbol. Por ejemplo, cuando selecciona Librerías en el árbol, puede crear una nueva o abrir una ya existente

★ Hay un enlace en la parte inferior para abrir librerías utilizadas recientemente

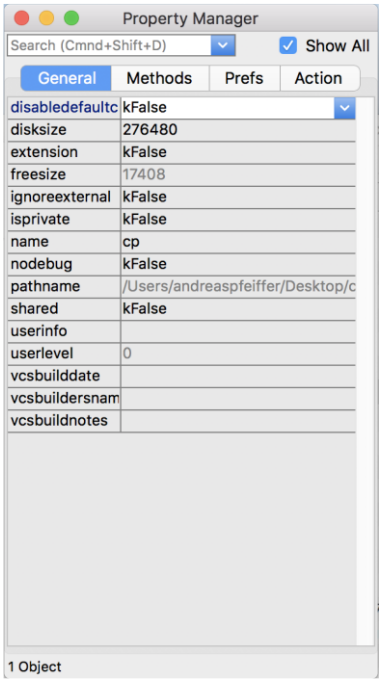


Librerías recientes

El Property Manager o Gestor de propiedades

El gestor de propiedades muestra las propiedades disponibles de la librería, clase o componente seleccionado y es sensible al contexto. Para abrirlo puede usar el menú contextual en un elemento o usar F6 (Windows) o Cmd-6 (Mac OS).

Asegúrese de tener marcada la casilla “Mostrar todo” para ver todas las propiedades del objeto. Estas se distribuyen en páginas lógicas en las pestañas del panel. (general, métodos, etc.)



Gestor de propiedades (F6)

★ Tiene un cuadro de búsqueda que le permite encontrar rápidamente una propiedad en todos los paneles si conoce el nombre de la misma. ¡Pruébalo!

Ayuda

La ventana de ayuda se puede abrir con F1 (Windows) o Cmd-1 (Mac OS). Le permite buscar cualquier palabra clave y explica todas las funciones y comandos disponibles en Omnis Studio.

★ Puede usar Shift-F1 (Windows) o Shift-Cmd-1 (Mac OS). Esto convertirá tu cursor en un signo de interrogación. Luego haga clic en cualquier elemento, es decir, en el Catálogo o en el Navegador de Studio para obtener Ayuda para el elemento apuntado.

Ayuda (F1)

Catálogo

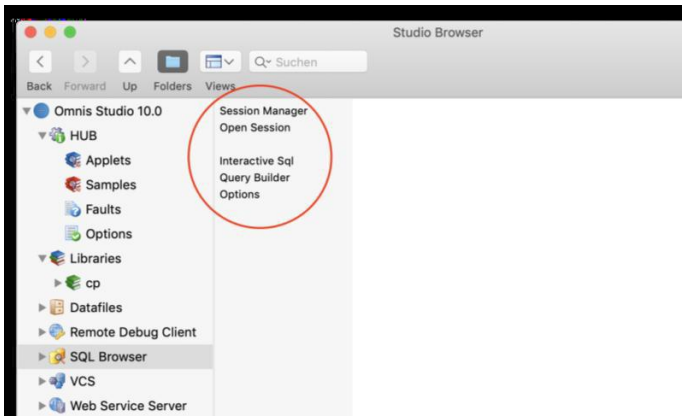
El catálogo se puede abrir con F9 (Windows) o Cmd-9 (Mac OS). Se usa para enumerar variables, esquemas, funciones, constantes y mucho más. Puede arrastrar y soltar el elemento desde Catálogo a la casilla de cálculo en su código o puede hacer doble clic en él cuando el cursor está en la casilla de cálculo para colocar el elemento en la línea del código

★ Use Shift-F1 (Windows) o Shift-Cmd-1 (Mac OS) y haga clic en una función para obtener ayuda detallada acerca de ella.

Catálogo

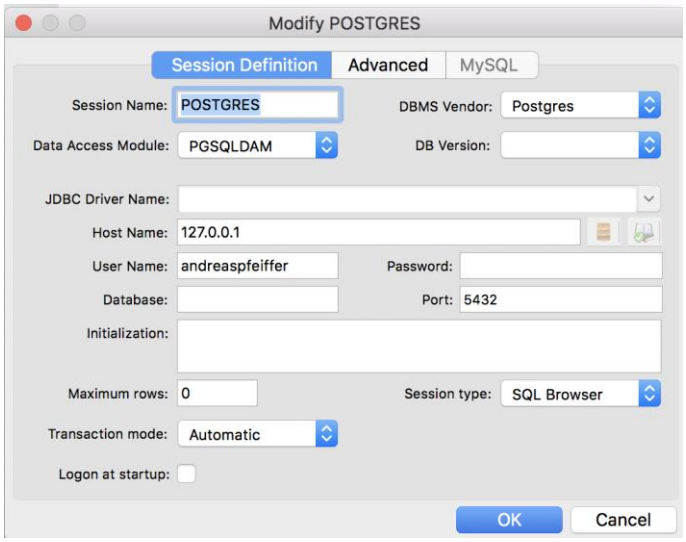
El Navegador SQL (Sql Browser)

El Omnis Studio Browser tiene un nodo en la lista de árbol, llamado "SQL Browser" que le permite abrir conexiones a su base de datos.



Enlaces en el navegador de SQL

Encontrará un enlaces al Administrador de sesiones que le permite definir una nueva sesión con la base de datos o modificar las definiciones de sesiones ya existentes.



Manejador de sesiones

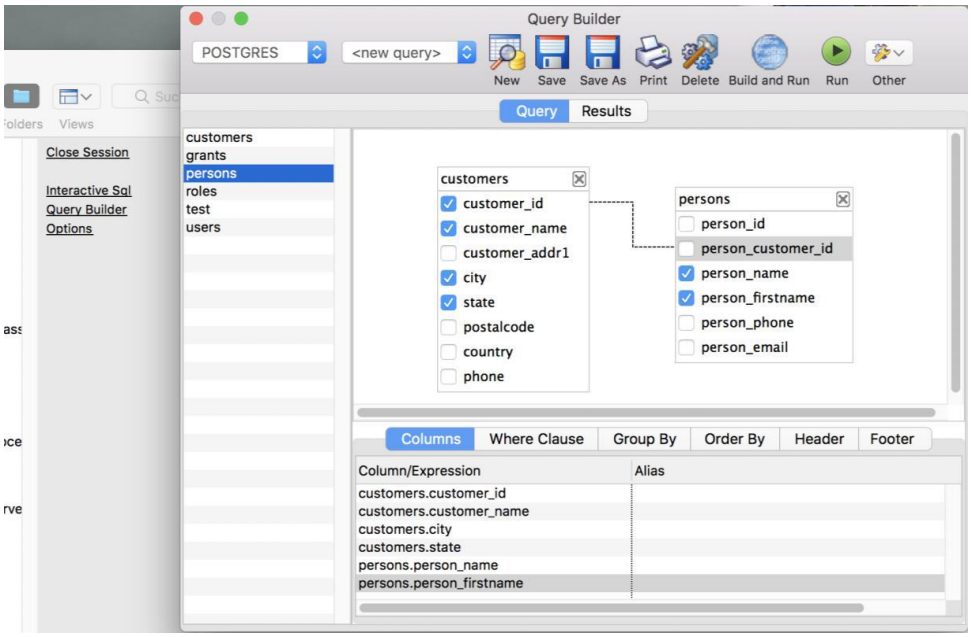
Una vez que haya ingresado la definición de sesión, puede abrir la sesión con la base de datos (haga clic en volver para ver el enlace abierto y seleccionar la sesión definida)

Luego podrá examinar en el navegador SQL todas las tablas, vistas, disparadores (lo que sea soportado por la base de datos).

Puede usar el menú contextual en la tabla de la base de datos para ejecutar una instrucción SELECT o usar la ventana SQL interactiva para ejecutar cualquier instrucción SQL manualmente.

El Constructor de consultas (Query Builder)

El generador de consultas le permite construir sentencias Select utilizando una o varias tablas de la base de datos. Se muestran todas las tablas de la sesión con la base de datos y se pueden arrastrar tablas desde allí hacia el área de diseño en el lado derecho. Luego puede marcar las columnas que desea recibir y hasta vincular tablas arrastrando la clave foranea de la tabla hija hacia la clave primaria de la tabla padre.



Constructor de consultas (Query Builder)

★ Se puede cambiar el tipo de unión utilizando el menú contextual sobre la línea de enlace.

Cuando presione "Construir y ejecutar (Build and run)", Omnis genera y ejecuta la sentencia SQL. Se pueden ver los resultados en el panel "Resultados".

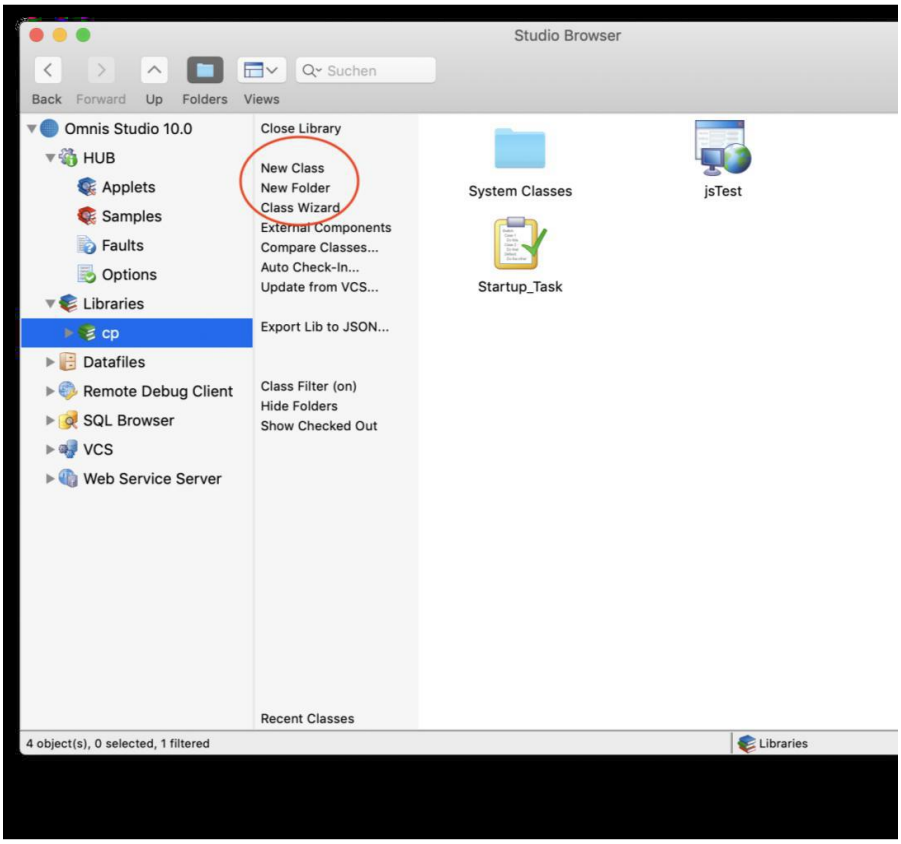
★ El menú "Otro" permite exportar los datos o generar código Omnis en el portapapeles para pegar ese código en su aplicación o abrir un asistente llamado ("Crear clase tabla ...") que permite generar el código y crear una nueva clase tabla.

Ejercicio 3: configuración de las propiedades de la librería

1. Seleccione la librería "CP" en el navegador Omnis Studio y use el menú contextual o F6 (Windows) o Cmd-6 (Mac OS) para abrir el Gestor de propiedades.
2. Asegúrese de marcar la casilla de verificación "Mostrar todo" en el Gestor de propiedades.
3. Cuando se selecciona la librería, el gestor de propiedades muestra todas sus propiedades
4. En el panel PREFS encontrará la propiedad "nombre predeterminado".
5. Establezca el "nombre predeterminado" como "CP". Esto asegurará que el nombre interno de la librería sea siempre "CP" incluso cuando a la librería se le cambie el nombre en el disco duro.

Creación de una nueva clase

Para crear una nueva clase o una carpeta dentro de su librería, necesita primero seleccionar la librería en el navegador de Studio. Luego puede usar los enlaces junto a la lista de árbol o el menú contextual en el área de clase a la derecha para agregar una nueva carpeta o clase.



Tipos de clases

Hay varios tipos de clases en Omnis Studio. Los más importantes para desarrollar aplicaciones web o móviles son:

- **Tareas remotas**
  Se utilizan para encapsular la conexión con el cliente. Normalmente solo necesita una tarea remota.Una tarea remota tiene propiedades que permiten averiguar la dirección IP (dirección de cliente) o configurar una conexión SSL (segura) si su servidor web tiene un certificado SSL válido. La tarea remota permanece abierta hasta que el cliente cierre la instancia del navegador o si se agota el tiempo de espera (timeout). Puede ver todas las propiedades o modificar algunas de ellas. Por ejemplo, si establece la propiedad tiempo de espera en 30, la tarea se eliminará si el usuario permanece inactivo durante 30 minutos.

- **Forma remota**
  Esta es la clase que permite dibujar la interface del usuario con la aplicación. Es una buena práctica usar varias de esas clases con no muchos campos en ellas. Puede usar un campos subforma para anidar formas remotas. Además, puede usar un campo subforma y cambiar dinámicamente el nombre de la clase para construir un menú que controle diferentes formas remotas que se mostrarían en un campo subforma.

- **Clase tabla**
  Esta clase tiene métodos incorporados como $select, $update, $insert, etc. Si va a utilizar métodos internos, debe vincular la clase tabla a una clase esquema (o a una clase query). Usted puede poner sus propias sentencias select y sus propias declaraciones y reglas comerciales en esta clase. Permite separar sentencias SQL complejas de la IU (interface del usuario) para que los métodos puedan reutilizarse en diferentes contextos. Se crea una instancia de una clase tabla utilizando una variable tipo lista o tipo fila. De esa manera, la lista o fila no solo contiene datos sino también métodos.La lista o fila se convierte en una instancia de la clase de tabla.

- **Clase objeto**
  Esta es otra clase que permite encapsular métodos. Puede usar esta clase para implementar objetos que pueden contener funciones utilizadas más globalmente. Por ejemplo: leer archivos, enviar correos electrónicos, analizar JSON, iniciar un temporizador, etc. Las clases objeto son similares a las clases de tabla pero no vienen con métodos incorporados. Para crear una instancia de una clase objeto, deberá declarar una variable objeto Puede asignar un subtipo a la clase objeto. De esa manera la variable objeto es usada como la instancia de la clase objeto.

- **Clase reportes**
  Esta clase permite imprimir sus datos en una impresora. También puede especificar diferentes tipos de salidas por ejemplo un dispositivo PDF. Esto es útil para aplicaciones web cuando el PDF se debe enviar al cliente.

- **Tarea de inicio (Startup task)**
  Esta clase es global para el lado del servidor de una aplicación web o móvil. Controla cualquier instancia que se ejecute en el servidor, como las clases ventana. Podríamos por ejemplo abrir una ventana monitor que muestre todas las conexiones o agregar algún código al método $construct del Startup Task para construir un pool de sesiones cuando se abra la librería y, por lo tanto, la tarea de inicio.

Ejercicio 4: agregar una tarea remota

1. Cree una tarea remota y llámela rtTask.
2. Busque las propiedades "dirección de cliente", "issecure" y "timeout" de la tarea remota en el gestor de propiedades (Property manager).
3. Cambie la propiedad "timeout” a 30 minutos.

Ejercicio 5: inicie sesión en la base de datos de demostración de Cucina Piccola

1. En el navegador SQL, haga clic en el Administrador de sesión y luego en Nueva sesión para definir una nueva sesión con la base de datos.
2. Ingrese como nombre para la nueva sesión "CP".
3. Elija SQLite como proveedor de la base de datos y SQLITEDAM como el Módulo de acceso a datos.
4. Hay dos botones al lado del nombre del host. Use el de la derecha que al apuntarlo dice. "Select Datafile", navegue a la carpeta del ejemplo de Cucina Piccola y elija el Archivo Cucina_Piccola.db.
5. Confirme con OK para guardar esta configuración.
6. Haga clic en "Atrás" y luego en "Abrir sesión". Debería aparecer listada la nueva sesión de la base de datos CP.
7. Haga clic en CP y la sesión de la base de datos debería abrirse y hacerse visible en la lista de árbol.
8. Debería poder hacer doble clic en "Tablas" y verr todas las tablas que esta base de datos exhibe allí.

Luego puede usar el menú contextual en cualquiera de las tablas para mostrar sus datos.

Ejercicio 6: crear una clase tabla con el constructor de consultas

1. Abra el constructor de consultas
2. Arrastre la tabla del servidor "freeTables" al área de diseño.
3. Seleccione ambas columnas con la casilla de verificación.
4. Haga clic en "Build and run (construir y ejecutar)"
5. Modifique la instrucción SQL generada en la siguiente forma (Nota: los dos símbolos || sirven para concatenar el texto "Mesa" con la columna "table_id"):

```
SELECT 'Mesa '||table_id as tablename,
  CASE table_taken
    WHEN 1 THEN 'invitados en mesa'
    ELSE 'mesa libre'
  END as status
FROM
  freeTables"
```

6. 6. Pruebe su sentencia usando el botón "Ejecutar (Run)" y corrija cualquier tipo de error.
7. Si el resultado es satisfactorio, utilice el botón "Otro" para crear una nueva clase tabla en su librería CP, clase a la que pondrá el nombre "taTables"

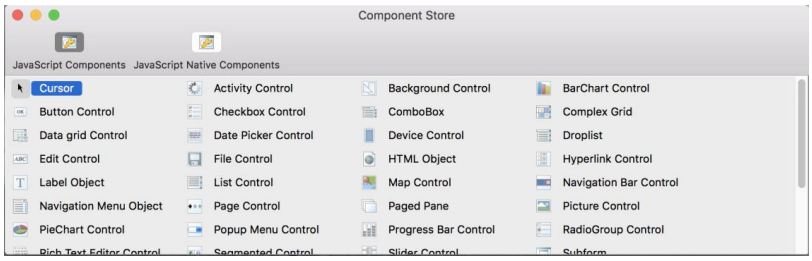
Crear una clase tabla desde un query

8. Luego puede cerrar el generador de consultas. Se le preguntará si desea guardar la consulta. Presione "No".

1. Vaya a su librería CP e inspeccione las nuevas tablas clase tabla. El método $construct de esta clase contiene código que garantiza que la instancia de esta clase utilizará la sesión "CP" con la base de datos, que hemos abierto en la versión de desarrollo de Omnis Studio. Además hay un método $load que contiene el código SQL generado. También tiene una línea que ejecuta la declaración del objeto sesión de la base de datos y obtiene los datos en la instancia actual.

Diseño de la forma remota

Haga doble clic en una forma remota para abrir su ventana de diseño. Verá paneles de pestañas en la parte superior de la ventana de diseño que representan los llamados "puntos de corte de diseño" (Layout Breakpoints). Puede cambiar entre ellos para mostrar diferentes resoluciones de pantalla. Puede agregar o eliminar esos puntos de corte de diseño utilizando el signo más o x, pero no recomendamos tener más de 4. Los campos que coloque en la forma remota pueden tener diferentes tamaños y posiciones para cada uno de los puntos de corte. Hay una propiedad "animatelayouttransitions" en el grupo "acción" de la forma remota que permite animar los campos cuando cambie el tamaño de la pantalla; es decir, cuando el usuario inclina el dispositivo por ejemplo.



Tienda de componentes

Puede agregar campos a la forma remota desde la tienda de componentes que aparece automáticamente cuando abre la ventana. Alternativamente, puede usar F3 (Windows) o Cmd-3 (Mac OS) para abrirlo. Simplemente arrastre y suelte el campo requerido desde la tienda de componentes hacia la forma remota. Alternativamente, puede seleccionar un campo en la tienda de componentes y luego dibujar un rectángulo en el formulario. Esto creará un componente del tamaño dibujado. Después de añadir un campos, enseguida normalmente se suele cambiar sus propiedades ya que el campo está seleccionado.

★ Puede personalizar la tienda de componentes utilizando su menú contextual. Por ejemplo para mostrar texto en vez de iconos. Asegúrese de guardar la configuración con otro clic contextual: Guardar Configuración de la ventana. Esto asegura que los cambios sean permanentes.

Estas son las propiedades más importantes de los campos:

- nombre: ingrese un nombre significativo, por ejemplo, "orderBtn", "street", etc. Esta propiedad podría utilizarse más adelante para tener acceso a ese campo .
- nombre de datos: para todos los campos controlados por datos, como campos de entrada, botones de opción, cuadrículas de datos, etc. esta propiedad se utiliza para vincular el campo a una variable de instancia.
- edgefloat: permite que el campo cambie de tamaño o flote automáticamente cuando cambie el tamaño de la pantalla. Esto permite usar un Layout Breakpoint

para teléfonos móviles de diferentes tamaños. Por ejemplo un iPhone o iPhone Plus y también diferentes teléfonos Android usarían el mismo Layout Breakpoint. Con esta propiedad puede pedir que el borde de una cuadrícula se vuelva flotante y, por lo tanto, agrande la cuadrícula para un teléfono de mayor tamaño.

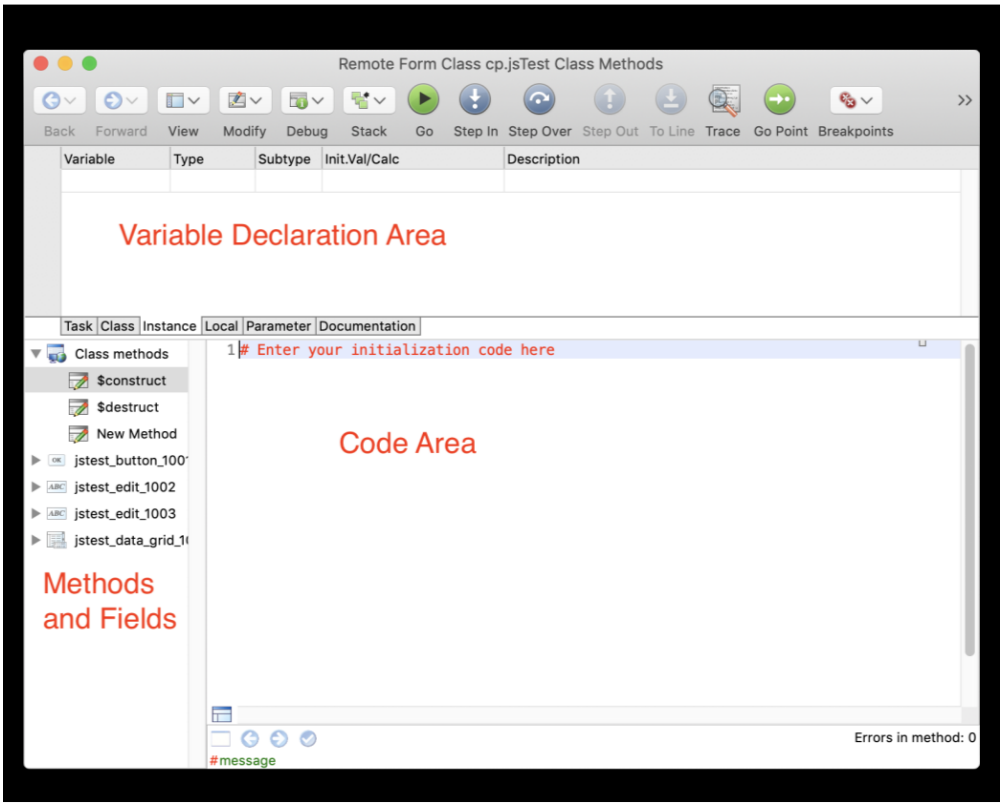
★ Cuando abra la lista desplegable de la propiedad edgefloat hay una casilla marcada "Establecer para todos los puntos de ruptura"en la parte inferior de la lista lo que permite especificar borde flotante para todos los puntos de ruptura a la vez.

- texto: algunos campos, como botones o etiquetas, tienen una propiedad texto que permite ingresar el texto mostrado en ese campo.
- visible: si necesita ocultar el campo, puede configurarlo como kFalse.
- visibleinbreakpoint: permite ocultar un campo para un punto de corte específico.
- izquierda, arriba, altura y ancho son el tamaño y la posición del campo.
- evento: permite definir que se recibirá cualquiera de los eventos enumerados. Debería agregar código a cada método de cada evento habilitado para ejecutar cualquier acción cuando se reciba ese evento.

★ Puede probar en cualquier momento usando Ctr-T (Windows) o Cmd-T (Mac OS) o con el menú contextual sobre el fondo del formulario. El navegador web debe abrirse y mostrar el formulario diseñado.

El editor de métodos

Puede abrir el editor de métodos (o Editor de código Omnis) para la forma remota utilizando el menú contextual (y eligiendo "Métodos" o "Métodos de la clase") en la clase, o haciendo doble clic en el fondo de la forma o en uno de los campos. En el último caso, aterrizaría en el método $event del campo marcado. De lo contrario, estaría en el método $construct de la clase.

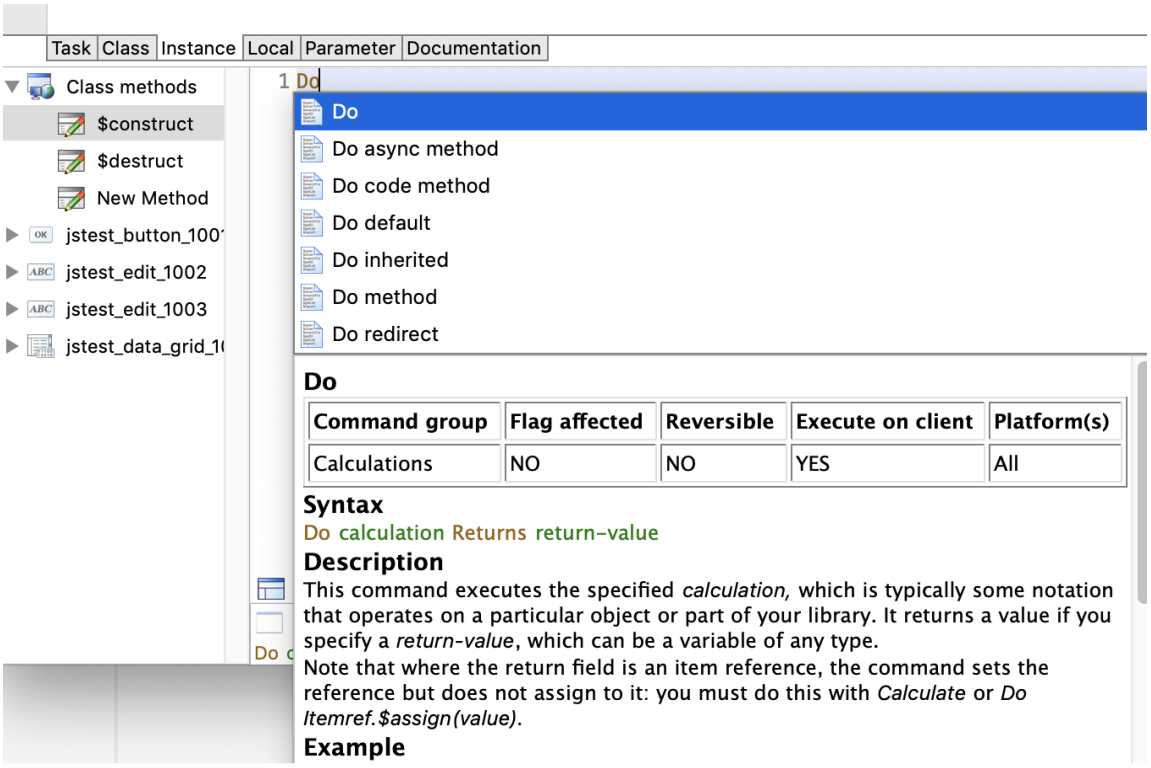


El editor de métodos

Si ha puesto nombre a sus campos, los encontrará nuevamente en el área marcada como Métodos y campos. Cada campo normalmente tiene un método $event por sí mismo pero puede tener métodos adicionales.

El área de declaración de variables permite declarar variables. Primero elija el alcance de la variable usando las pestañas mostradas en la parte inferior de esta área y luego escriba el nombre de la variable, su tipo y subtipo. Para datos que desee enlazarlos a la propiedad dataname de un campo, debe utilizar una variable con alcance instancia.

Para ingresar código, primero seleccione una línea en el área de Código. Luego digite el comando que desea: por ejemplo, "D". Aparecerá el **Asistente de codificación** para mostrarle los comandos que empiezan con “D” Puede continuar digitando y el asistente le indicará resultados mas finos. Cuando usa las teclas del cursor para navegar a un comando, el asistente brindará mas ayuda especifica para el comando escogido.



Asistente de codificación

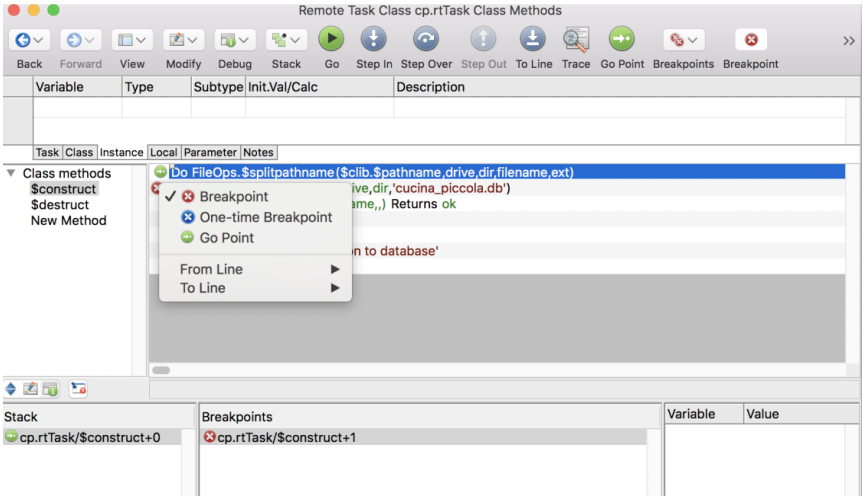
Ejercicio 7: creación de la forma remota para personal de servicio

1. Cree una nueva forma remota y asígnele el nombre "jsService"
2. Abra métodos, utilizando el menú contextual sobre el fondo de la nueva forma remota.
3. Agregue una variable con alcance instancia "iTableList" escribiendo esto en el primer cuadro de la zona de declaración de variables. Luego configure el tipo para que sea "Lista" y el subtipo para que sea taTables.
4. Ahora ingrese el comando "Do" colocando el cursor en la primera línea de código y digitando "D"
5. Abra el Interface Manager con un clic contextual en la variable iTableList - Interface Manager y busque el método $load que generamos anteriormente.
6. Arrastre $load de la iTableList al cuadro de cálculo del comando "Do".
7. Use F3 (Windows) o Cmd-3 (Mac OS) o haga doble clic en la forma remota para abrir su ventana de diseño.
8. En la ventana de la tienda de componentes, escoja la pestaña Componentes nativos de JavaScript y arrastre el Control lista nativa hacia la forma remota.
9. Busque la propiedad "dataname" para este campo y escriba "iTableList" que debería aparecer cuando empiece a escribir.
10. Cambie la propiedad name para que diga "cuadrícula".
11. Vaya a la pestaña "Data" del Administrador de propiedades y asigne los siguientes valores:
    accesorytypecol = 3
    text1col = 1
    text2col = 2
12. Pruebe el formulario (usando ya sea Ctr-T (Windows) o Cmd-T (Mac OS) o el menú contextual sobre el fondo de la Forma. Esta debe aparecer en el navegador web y mostrar los datos. La tercera columna (accesorytypecol) utiliza la columna 3 de la lista, (que siempre es 1) y, por lo tanto, muestra el ícono flecha. Este ícono indica que el usuario puede navegar al siguiente nivel.

Parte 2: Mejorar la forma remota para personal de servicio mostrando imágenes

Depuración en Omnis Studio

El depurador (The debugger)



Depurador

El **depurador** de Omnis Studio es muy poderoso. Se puede establecer breakpoints para detener la ejecución del código usando el menú contextual, o mediante los botones de la cinta de encabezado o usando el menú "Depurar" de la cinta. También hay atajos de teclado disponibles, vealos en el menú Depurar.

Hay diferentes tipos de breakpoints o puntos de interrupción disponibles. El rojo permanecerá en el código mientras usted no lo quite. Desaparecerá al cerrar la librería. El punto de interrupción azul es un punto de interrupción de una sola vez, lo que significa que desaparecerá automáticamente después del primer uso. Además puede codificar un punto de interrupción utilizando el comando "Breakpoint". Ese nunca desaparece mientras no lo borre de su código. Pero no tendrá ningún efecto a tiempo de ejecución. Finalmente hay puntos de interrupción sobre variables, que puede habilitar usando el menú contextual sobre una variable. Por ejemplo puede hacer que el código se detenga cuando el contenido de la variable cambie o cuando alcance un cierto límite.

Si desea depurar su código, necesita colocar uno de los puntos de interrupción (por ejemplo, el rojo) en su código y luego inicie su aplicación probando la forma remoto. Cuando el código llegue al punto de interrupción traerá Omnis al frente y luego usted podrá recorrer su código usando uno de los Botones "Step ..." o las teclas de acceso directo para avanzar (vealas en el menú Depurar).

★ Puede comenzar a depurar y abrir un stack de ejecución simplemente haciendo doble clic en una línea de código. Esto funcionará siempre que no esté utilizando ninguna variable instance o task en este código. La razón es que el doble clic no crea una instancia de su clase.

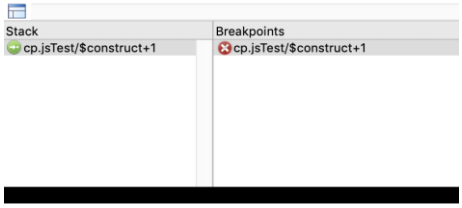
Verá una bolita verde con una flecha blanca cuando su depurador llegue al punto de ruptura por primera vez. A esto le llamamos el punto Go. Esa línea se ejecutará la próxima vez que toque uno de los botones Step o el botón Go (Ir).

Entonces, ¿cómo depurar el código desde el primer punto de interrupción?

- Go: este botón ejecuta todas las línea siguientes hasta que se alcance otro punto de interrupción o se llegue al final del stack del método. Se recomienda utilizar este botón como la última acción para resolver el stack.
- Step In: inicia y ejecuta una línea. Si se llama a otro método, Step In entrará a la subrutina.
- Step Over: avanza y ejecuta una línea. También ejecuta subrutinas pero no entra en ellas.
- Step Out: sale de una subrutina y se detiene en la siguiente línea del método exterior.
- To line: puede seleccionar una línea de abajo, se ejecutan todas las líneas entre el punto Go y la línea seleccionada y el depurador se detiene en la línea seleccionada.

Una vez que comience a depurar verá el stack de métodos en la esquina inferior izquierda. Para cada subrutina se agregará otra línea a la pila de métodos y puede navegar entre ellos haciendo clic en una línea de la pila. Puede borrar la pila (stack) de métodos utilizando el menú contextual.

★ Puede configurar el punto Go a otra línea de código si lo desea.Por ejemplo, si hace cambios a su código mientras depura, puede establecer el punto Go a la línea que ha cambiado y volver a ejecutar esa línea otra vez. Para editar el código durante la depuración hay pequeños botones arriba de la lista del stack que le permiten cambiar entre modo de edición y modo de stack.



Lista del stack con botones

Automatizar el inicio de sesión

Actualmente estamos utilizando el SQL Browser para iniciar sesión en la base de datos. Sin embargo, esto funciona en la versión de Desarrollo de Omnis Studio pero no está disponible cuando la aplicación corre en versiones run time. Por esta razón, necesitamos desarrollar un método de inicio de sesión que se ejecute cuando un nuevo usuario se conecte.

Cuando el usuario inicie sesión en la primera instancia de la forma remota, Omnis encapsulará automáticamente esa instancia dentro de la instancia del remote task. Cuando una instancia está a punto de instanciarse, su método $construct se ejecuta automáticamente. Por lo tanto, el método $construct del remote task es el primer método que se ejecuta.

Para escribir el nuevo código de inicio de sesión necesitamos el nombre del host, que, para la base de datos SQLite, es solo la ruta del archivo Cucina_Piccola.db. Dado que este archivo está en la misma carpeta que la biblioteca, se puede escribir un poco de código para descubrir la ruta de la biblioteca actual. Podemos usar el alias "$clib" que se refiere a la librería actual y agregar la propiedad $pathname. De ahí que $clib.$pathname retornará la ruta completa de la librería.

Sin embargo, tendríamos que partirla para usar la ruta de acceso a la carpeta y agregarle "Cucina_Piccola.db". Para eso podemos usar la función "FileOps.$split-pathname" y la función "con" para concatenar textos, ambas las encuentra en el Catálogo.

Ejercicio 8: escribir el método de inicio de sesión

1. Haga doble clic en el Remote Task rtTask para abrir su editor de métodos.
2. Asegúrese de estar en el método $construct y agregue el comando "Do".
3. Necesitamos partir el nombre de ruta de la librería actual. Por lo tanto, ingrese lo siguiente después del comando Do (Obtendrá ayuda del Asistente de notación mientras escribe):

   FileOps.$splitpathname($clib.$pathname,drive,dir,filename,ext)

   Nota: Observe que los nombres de las variables que ingresó tienen un subrayado rojo ondulado. Esto indica que puede haber un error de digitación o, en este caso, que las variables aún no han sido declaradas. Usted puede corregir esto usando el pequeño botón de verificación en la parte inferior de su editor. Esto abrirá un mensaje que le permite declarar la variable sobre la marcha. Asegúrese de hacer esto para todas las variables que haya agregado: "drive", "dir", "filename" y "ext". Asegúre también que cada una de ellas sean variables locales tipo caracteres.

   Nota: Encontrará las nuevas variables dentro del Área de declaración de variables en la sección Local.
4. Agregue otra variable local "hostname" tipo carácter agregando otra línea en el área de declaración de variables.
5. Ahora presione Mayúsculas - clic (shift click) en el lado izquierdo de esta línea en el área de código para iniciar el depurador. Debiera ver ahora el punto Go en la línea de código.
6. Haga clic en Step In o Step Over para ejecutar esta línea. Debería ver la ruta cuando coloca el mouse sobre la variable "dir" en el código.
7. Agregue otra línea usando Ctr-N (Windows) o Cmd-N (Mac OS).
8. Escriba "C" para obtener el comando "Calcular". Este se usa normalmente para asignar un valor. Podría también haber usado la tecla tabulación para obtener el comando "Calcular", ya que es el primero de la lista.
9. Luego escriba "h" y el Asistente de Código mostrará la nueva variable "hostname" que puede elejirla también con la tecla de tabulación.
10. Un tab adicional aceptará el "as" y pondrá el cursor al final del comando para que pueda ingresar la función: con(drive,dir,'cucina_piccola.db')
    Ahora debería tener el código en esa línea:

    Calcule hostname as con(drive,dir,'cucina_piccola.db')
11. Ahora salte esa línea y debería ver el hostname calculado al pasar el cursor sobre la variable "hostname".

12. Haga clic en Go o borre el stack de métodos utilizando el menú contextual en la lista del stack.

13. Declare una variable task "tSessionObj" de tipo objeto. Como subtipo, seleccione SQLITESESS de la lista desplegable en subtipo, dentro del grupo Objetos externos.
    Nota: Esta será la variable objeto de sesión que vamos a usar para conectarnos a la base de datos.

14. Ahora agregue lo siguiente en la siguiente línea:
    Do tSessionObj. $logon (hostname,,) returns ok

    Nota: Las dos comas representan los parámetros de nombre de usuario y contraseña. Son obligatorios para el método $logon pero deben estar vacíos porque SQLite no soporta usuarios.

15. El valor de retorno del comando "Do" será la nueva variable local: "ok". Se supone que debe declarar esta variable usando el pequeño botón de verificación en la parte inferior. Seleccione "boolean" como tipo.

16. Finalmente ingrese las siguientes líneas de código:

    ```
    If ok
    Else
    Quit method 'could not logon to database'
    End If
    ```

17. Cierre el editor en la tarea remota.

Ejercicio 9: permitir que el objeto de datos use la nueva sesión de la base de datos

1. Seleccione la clase tabla "taTables"
2. Busque la propiedad "designtaskname" y asigne la tarea remota allí utilizando la lista desplegable mostrada al lado de dicha propiedad

   Nota: esto asegura que la variable de tarea se haga visible al momento de diseño.
3. Haga doble clic en "taTables" para abrir su editor.
4. Encontrará código en su método $construct que asigna el objeto de sesión a la instancia actual. Utiliza la sesión global que abrimos en el IDE. Reemplace esta sesión usando la variable de tarea en su lugar. Su código debería verse así:

   Do $cinst.$sessionobject.$assign(tSessionObj)
5. Pruebe su forma remota "jsService". Si no ve ningún dato o si recibe error use un breakpoint (rojo) para depurar su código.

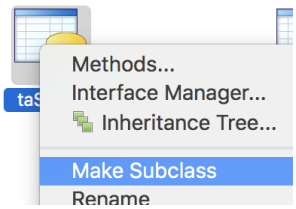
Ventajas de usar una clase supertabla

La construcción de nuestra clase tabla usa código para asignar el objeto de sesión a la instancia actual que, en el caso de una clase tabla, es ya sea una variable lista o una variable de fila, en realidad un objeto de datos. Asumiendo que todos nuestros objetos de datos harán uso de la misma sesión de base de datos, necesitaríamos repetir esto para cada nueva clase tabla que agreguemos a nuestro proyecto. Además, ¿qué pasa si quisieramos implementar un comportamiento común para todos los objetos de datos, como escribir en la columna "insert_date" o "user_name" de la tabla?

Para evitar duplicar código, simplemente podríamos usar una superclase y heredar todos sus métodos a otras clase tablas.

¿Cómo heredar una clase?

Hay dos maneras de hacer esto. Si tiene una super clase y desea crear una nueva clase heredada de ella simplemente haga clic derecho en la super clase en el Navegador de clases y seleccione "Crear subclase" en su menú contextual. En este caso Omnis intenta heredar tantos métodos y propiedades como se pueda.



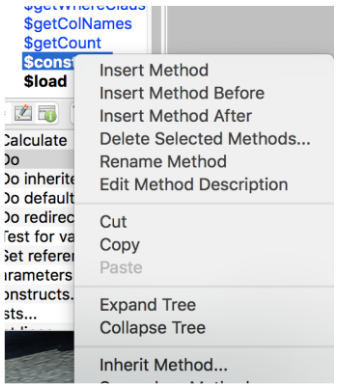
Hacer subclases

Como alternativa, puede asignar a la propiedad $superclass de la subclase el nombre de la superclase.

¿Cómo anular o heredar métodos y propiedades?

Los métodos y propiedades heredados se muestran en color azul para indicar que están determinados por la super clase. Puede heredar usando el menú contextual sobre el nombre del método o nombre de la propiedad. A la inversa puede anular (desheredar) un método o propiedad utilizando el menú contextual sobre el mismo.

★ Puede hacer doble clic en un método heredado azul para saltar al método de superclase.



Heredar un método

Ejercicio 10: presentación de la tabla superclase

Necesitamos ordenar un poco nuestra librería. Por esta razón haremos una carpeta que va a contener clases vinculadas a datos.

1. Cree una carpeta en su librería llamada "DatabaseLayer".
2. Arrastre la tabla taTables a la nueva carpeta DatabaseLayer.
3. Abra la librería resource.lbs
4. Mueva la tabla taSuper a la carpeta DatabaseLayer de su librería. Esto copiará la clase a su librería.
5. Ahora seleccione taTables dentro de su librería cp.lbs y encuentre la propiedad $superclass.
6. Asigne el nombre de la superclase "taSuper" a esta propiedad. (Esto hace que la tabla taSuper sea una superclase de taTables.)
7. Abra el editor de clases para taTables (con doble clic en el Navegador sobre taTables).

   Nota: ahora verá todos los métodos públicos de la superclase en azul. El método $construct no ha sido heredado porque ya tenía código.
8. Herede el método $construct: haga clic con el botón derecho del mouse en el nombre del método y elija “Inherit Method (Heredar Método)". Se le preguntará si desea eliminar el código existente. Confirme con "Sí".
9. Herede la propiedad "designtaskname" de taTables.
10. Ahora copie todas las demás clases tabla, las dos clases esquema "facturas" y "pedidos", así como el objeto oNavegación desde resource.lbs a la carpeta DatabaseLayer de su librería.
11. Copie la clase de sistema #ICONS de la carpeta "System Classes" a la carpeta "System Classes" de su librería. (Allí están algunos íconos que usaremos)
12. Cierre la librería resource.lbs.

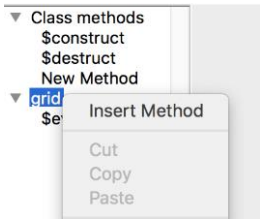
Ejercicio 11: clase objeto oNavegation

1. Consulte oNavigation que es de clase objeto. Haga doble clic para ver sus métodos.
2. Seleccione $getGroupList. Verá que devuelve una lista anidada ya que la variable "mainList" está usando una "subList" en una de sus columnas.
3. Eche un vistazo a $getProductList. Este método devuelve una lista de productos.
4. Consulte la declaración de las variables locales tipo lista. Verá que utiliza métodos de las clase tabla importadas. Este método carga productos para cada grupo de productos y devuelve una lista anidada también.

Ejercicio 12: escribir código para el componente que hemos llamado cuadrícula

1. Abra el editor de métodos de jsService.

2. Encuentre el componente cuadrícula y añada un nuevo método a dicho componente usando el menú contextual sobre el nombre del componente dentro de la lista de métodos. Llámelo $setMenu



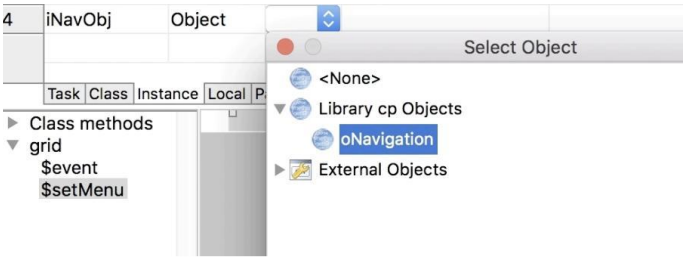
Añadir un método a un campo

3. Declare un parámetro "pOffset" en el panel de declaración del método $setMenu. El tipo debe ser Integer y el subtipo 32 bits Int. Se usará para navegar al siguiente nivel (+1) cuando el usuario haga clic en un elemento de la cuadrícula o retroceda un nivel (-1) cuando el usuario haga clic con el botón Regresar.

1. Agregue una variable de instancia "iMenuLevel" - Short Integer. Se usará para mantener el nivel del menú corriente. Ponga como valor inicial "1".

2. Agregue una variable de instancia "iHeader" del tipo "character". Se usará para mostrar el texto del encabezado. Por ejemplo: el nombre de la tabla que se eligió en el primer nivel.

3. Agregue la variable de instancia "iNavObj" tipo "objeto". Como subtipo, seleccione oNavigation dentro de su librería.



Asignación del subtipo en una variable objeto

7. Agregue otra variable de instancia "iCurrentListName" de tipo carácter.

1. Agregue una variable de instancia "iGroupList" tipo lista; no necesita ningún subtipo.

2. Agregue una variable de instancia "iProductList" tipo lista; no necesita subtipo.

3. Agregue una variable de instancia "iOrderRow" tipo fila basada en la clase tabla (subtipo) taOrders.

4. Agregue una variable de instancia "iSummaryList" tipo lista, sin subtipo.

5. Establezca el foco en el área de código del editor de métodos y agregue las siguientes líneas de código:


```
Calcule iMenuLevel as iMenuLevel + pOffset
Switch iMenuLevel
Case 1
  Calculate iHeader as ''
  Calculate iCurrentListName as 'iTableList'
Case 2
  Calculate iHeader as con('Table ',iTableList.$line)
  calculate iCurrentListName as 'iGroupList'          (use comillas en 'iGroupList').
  Do iNavObj.$getGroupList(iTableList.$line) Returns iGroupList

(Nota: iTableList.$line contiene el número de la linea corriente de iTableList)

  Case 3

(Nota: debemos determinar si estamos en el primer grupo. Digite lo siguiente:)

    If iGroupList.$line=1
      Do iNavObj.$getProductList(iGroupList.subList.$line=2) Returns iProductList
      Calculate iCurrentListName as "iProductList"
    Else
      If iGrouplist.subList.$line=1

(Nota: Esto es para chequear si se ha usado el botón "summary")

        Do iOrderRow.$getGroupedList(iTableList.$line) Returns iSummaryList
        Calculate iCurrentListName as 'iSummaryList'    (use comillas en
'iSummaryList').
      End if
    End if
  Default
    Calculate iMenuLevel as iMenuLevel-pOffset
End Switch
```

(En la siguiente línea asigne un nuevo dataname al campo cuadricula:)

```
Do $cinst.$objs.cuadricula.$dataname.$assign(iCurrentListName)
```

Para su conveniencia aquí está el código completo tal como lo debería haber digitado:

```
Calculate iMenuLevel as iMenuLevel+pOffset
Switch iMenuLevel
  Case 1
    Calculate iHeader as ''
    Calculate iCurrentListName as 'iTableList'
  Case 2
    Calculate iHeader as con('Table ',iTableList.$line)
```



```
    Calculate iCurrentListName as 'iGroupList'
    Do iNavObj.$getGroupList(iTableList.$line) Returns iGroupList
  Case 3
    If iGroupList.$line=1
      Do iNavObj.$getProductList(iGroupList.subList.$line=2) Returns iProductList
      Calculate iCurrentListName as 'iProductList'
    Else
      If iGroupList.subList.$line=1
        Do iOrderRow.$getGroupedList(iTableList.$line) Returns iSummaryList
        Calculate iCurrentListName as 'iSummaryList'
      End If
    End If
  Default
    Calculate iMenuLevel as iMenuLevel-pOffset
End Switch
Do $cfield.$dataname.$assign(iCurrentListName)
```


Manejador de eventos

Puede implementar un manejador de eventos para cada campo. Debe ser un método $event que lo debe buscar directamente en el nombre del campo dentro del editor de la forma remota.

Cada método $event comienza con el comando "On" utilizando una constante de evento, por ejemplo, "evClick", "EvDrop", etc. El código que sigue se ejecutará cuando ocurra el evento seleccionado.

El campo en sí tiene una propiedad llamada "events". Active el o los eventos que desea utilizar. En el caso de botones, el evento evClick ya viene activado por defecto.

Ejercicio 13: escribir el manejador de eventos

Necesitamos ejecutar este código cuando el usuario haga clic en la cuadrícula.

1. Vaya al método $event del campo cuadrícula.
2. Ya debería estar "activado" un "evClick" en la primera línea de código.
3. En la siguiente línea ingrese:

   Do $cfield.$setMenu(1)

   Nota: $cfield (se refiere al campo actual) es decir la cuadrícula y, por lo tanto ejecutará el método $setMenu. El parámetro (1) es el desplazamiento que indicará al método que queremos ir un nivel más abajo.
4. Antes que nada necesitamos activar el evento para el campo cuadrícula. Use F3 (Windows) o Cmd-3 (Mac OS) para abrir la ventana de diseño de la forma remota y vaya a propiedades. Hay una propiedad "event" en la que necesita activar el evento evClick.

   Nota: Por defecto, los eventos están generalmente desactivados. Es para minimizar el tráfico generado por eventos entre el navegador y el servidor de aplicaciones Omnis.

Ejercicio 14: agregar más campos a la forma de servicio

Podemos usar un componente "panel paginado" (Paged pane) como contenedor.

1. Abra la forma remota jsService en modo de diseño. Si todavía está en el editor de métodos, puede usar Shift-F8 (Windows) o Cmd-Shift-8 (Mac OS) para abrirlo.
2. Agregue un componente PagedPane de la tienda de componentes a la forma remota.
3. Asigne a la propiedad nombre "top_container".
4. Abra la lista desplegable de la propiedad "edgefloat" en la pestaña Apariencia. Seleccione la casilla de verificación "establecer para todos los puntos de interrupción del diseño" y elija kEFPosnTopToolbar. Esto hará que este componente aparezca siempre en la parte superior de su forma remota.
5. Cambie la altura del contenedor en cada corte de diseño para que quepa dentro de él fácilmente un botón.
6. Seleccione el componente cuadrícula y asignele kEFPosnClient para que ocupe todo lo que sobra (hasta el borde) en todos los puntos de interrupción del diseño.
7. Agregue un componente “Edit control” en el medio del contenedor (como si fuera título). Queremos que este campo muestre el contenido de la variable iHeader. Así que ingresemos esto en la propiedad dataname.
8. Visualmente centre el campo dentro del contenedor para cada diseño.

   Nota: Si el campo no está visible en el diseño más pequeño, es posible que desee utilizar la lista de campos (Field list en el menu contextual del formulario) para seleccionar el campo y asignarle un valor a la propiedad "left" utilizando el Manejador de propiedades.
9. Agregue un componente botón a la izquierda del campo Edit control que queremos sirva como botón "Regresar".
10. Vamos a llamarlo "backBtn" y asignarle "Regresar" como texto para el botón.
11. Haga doble clic en el botón para abrir el Editor de métodos. Esto le lleva al método $event del botón.
12. Luego de On evClick, ingrese lo siguiente para ejecutar el método $setMenu de la cuadrícula:

    Do $cinst.$objs.cuadrícula.$setMenu (-1)

Ejercicio 15: pruebe su forma

Ahora debería poder probar su forma. Es posible que desee agregar un breakpoint al método $event de la cuadrícula para depurar el código e inspeccionar la variable. Asegúrese de no agregar el punto de interrupción en la línea del comando "On", sino más bien en la siguiente línea después del comando "On".

Agregar etiquetas de texto e imágenes a la cuadrícula

Es posible que haya notado que no hay etiquetas de texto en los botones dentro de la cuadrícula y tampoco imágenes de los producto todavía. Los datos necesarios para esto ya están cargados en la lista pero no los podemos ver porque aún no hemos indicado cuales son las columnas de la lista que contienen dicha información.

Ejercicio 16: habilitación de las propiedades de íconos y etiquetas de texto

1. En la forma remota jsService, seleccione el campo cuadrícula y cambie las siguientes propiedades (en la pestaña Data):

   accesorycontentcol = 5
   accesoryvalcol = 6
   imagecol = 4

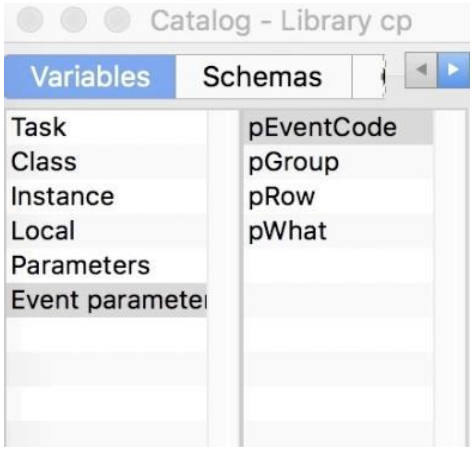
   Nota: estos valores apuntan a las columnas correspondientes de la lista que está enlazada mediante la propiedad $dataname

2. Pruebe de nuevo. Ahora debería ya ver las imágenes para el grupo del segundo nivel y el texto para el botón checkout en el 3er nivel, y por ejemplo, cuando haga clic en el grupo "Bebidas", ya debería ver imágenes de cada producto y también texto en cada botón de pedido.

Parámetros de eventos

Los métodos de eventos pueden tener sus propios parámetros. Si selecciona una línea de código de un método $event de un campo justo después (no sobre) On evClick, por ejemplo, puede usar F9 (Windows) o Cmd-9 (Mac OS) para mostrar el catálogo y busca la pestaña Variables, allí encontrará un grupo llamado "Event parameter".

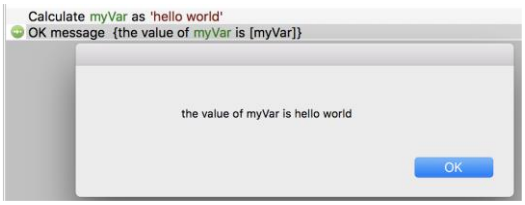
★ Durante depuración puede inspeccionar el contenido de esos parámetros de evento volando sobre ese parámetro o usando el menú contextual sobre la variable.



Event parameters en el Catálogo

Corchetes

Los corchetes le permiten forzar la evaluación de una variable. Por ejemplo cuando Omnis espera recibir solo texto se puede evaluar ese texto usando una variable dentro de corchetes.



Corchetes en el mensaje de Ok

Función jst ()

La función jst () se puede usar para dar formato a una cadena de texto. Consulte el manual en http://developer.omnis.net/documentation/functionref/index.jsp?detail=jst.html#jst o la Ventana de Ayuda para descubrir su rica funcionalidad.

Para este curso necesitamos el argumento "P" que se usará para llenar el resto de la cadena con el argumento que sigue a la P (es decir con ceros). Usamos el signo menos para indicar que el llenado sea en el lado izquierdo.

Por ejemplo:

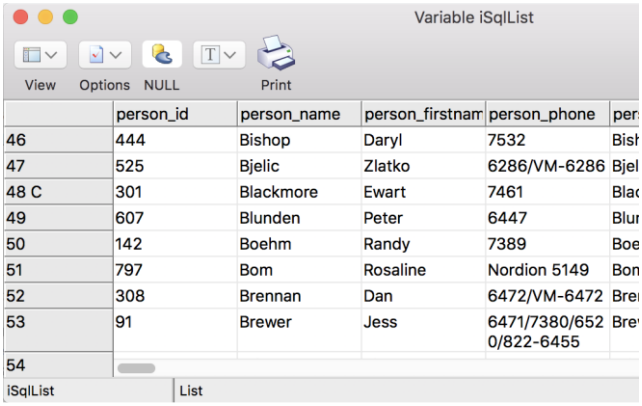
jst (999,'-5P0') se convertirá en: 00999

Notación de listas

Las variables tipo lista son variables estructuradas que utilizan columnas y filas. El conteo de las líneas de la lista comienza con 1.

Puede acceder al contenido de una celda dentro de una variable de lista utilizando el nombre de la lista seguido por el número de línea y el nombre de la columna (en ese orden) separados por un punto.

Por ejemplo, la imagen muestra el contenido de la variable de lista iSqlList.



	person_id	person_name	person_firstnam	person_phone	per
46	444	Bishop	Daryl	7532	Bish
47	525	Bjelic	Zlatko	6286/VM-6286	Bjel
48 C	301	Blackmore	Ewart	7461	Blac
49	607	Blunden	Peter	6447	Blur
50	142	Boehm	Randy	7389	Boe
51	797	Bom	Rosaline	Nordion 5149	Bon
52	308	Brennan	Dan	6472/VM-6472	Bre
53	91	Brewer	Jess	6471/7380/652 0/822-6455	Bre
54					

iSqlList | List

Variable lista

iSqlList.52.person_name devolvería "Brennan" ya que está en la línea 52 y en la columna person_name.

En algunos casos, una línea puede convertirse en la línea actual porque el usuario podría haber hecho clic en una línea de un componente cuya lista se asigna con su dataname o porque realizó una búsqueda en esa lista o simplemente asignando el $line de la lista: Do iSqlList.$execute(48)

Como puede ver, hay una "C"en la línea número 48 de la lista de ejemplo. Esto indica que 48 es la línea actual. O en otras palabras: iSqlList.$line=48

Si necesita hacer referencia a una celda de la lista dentro de la línea actual, simplemente puede omitir la línea número: iSqlList.person_name devolvería "Blackmore".

También puede usar corchetes si tiene el número de lista o el nombre de la columna dentro de una variable o parámetro. Por ejemplo, si el número de fila está en un parámetro "pRow", puede usar:

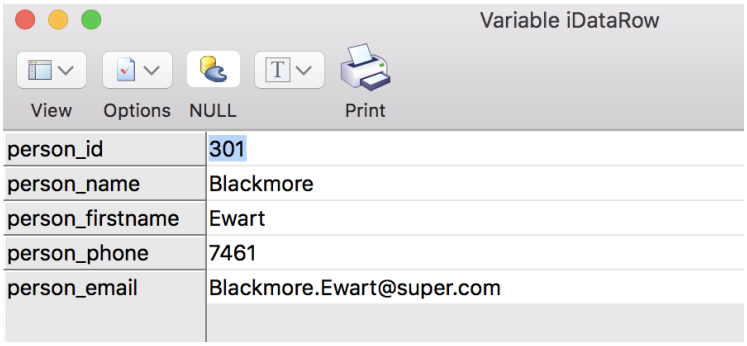
iSqlList.[pRow].person_name o, alternativamente, asigne a $line (la línea actual) el valor de pRow.

También puede acceder a una celda sin conocer el nombre de la columna si conoce el número de columna. En en este caso, puede usar una "c" como prefijo para el número de columna: iSQLList.c2 se referiría a la columna dos de la línea actual en iSQList. Nota: La "c" es obligatoria para distinguirla de la línea número.

Notación de filas

Una variable fila (row) es simplemente una lista con una sola línea y la línea siempre es la actual. Por lo tanto usted debería omitir el número de línea: Do iDataRow.person_name.$execute('Duck') asignaría "Duck" a la columna "person_name" de la variable de fila "iDataRow".

Tenga en cuenta que al depurar, la fila se muestra verticalmente.



Variable tipo fila

Así que, en este ejemplo, iDataRow.person_name devolvería "Blackmore".

Ejercicio 17: reacción al botón de pedido

1. Vaya al método $event de la cuadrícula.
2. Agregue algunas líneas entre "On evClick" y "Do $cfield.$setMenu(1)"
3. Justo abajo de "On evClick" ingrese lo siguiente:

   If pWhat=kJSNativeListPartAccessory

   Nota: esto es para averiguar si se ha hecho clic en uno de los botones de pedido
4. Agregue una línea vacía a continuación.
5. Agregue el comando "Else".
6. Después del "Do $cfield.$setMenu (1)" agregue el comando "End if".
7. Entre "If" y el "Else" ingrese lo siguiente:

```
Switch iCurrentListName
  Case ’iProductList’
    Do iProductList.$line.$assign(pGroup)
    Do iProductList.subList.[pRow].amount.$assign(iProductList.subList.[pRow].
amount+1)
  Case ’iGroupList’
  # Checkout
End Switch
```

   Nota: Si se hace clic en uno de los botones de la lista, debemos averiguar si se trata del botón "checkout" en iGroupList o si es el botón "order" en iProductList. Para este último debemos contar la cantidad en la sublista anidada.
8. Para recibir realimentación, en el formulario cuando el usuario haga clic en uno de los botones de pedido, debemos mostrar la cantidad en el botón de pedido. Para esto, agreguemos una variable local "myString" tipo caracteres.
9. Luego agregue una línea de código antes del Case ‘iGroupList‘:

   Calcule myString as iProductList.subList.[PRow].amount
   Do iProductList.subList.[PRow].buttontext.$assign(myString)
10. Pruebe su código. Al hacer clic en un botón de pedido, el texto del botón debe mostrar cuantas veces se ha dado clic en el botón. Desafortunamente, el ancho del botón cambia dependiendo del largo del texto.
11. Para solucionar esto usaremos la función jst (). Cambie la última línea que ingresamos de la siguiente manera (los cambios están en negrita):

    Do iProductList.subList.[PRow].buttontext.$assign(**jst(**myString**,’- 4P0’)**)
12. Pruebe nuevamente para ver los cambios.

Para su comodidad, aquí está el código completo tal como debía haberlo ingresado en el método $event de la cuadrícula:

```
On evClick
 If pWhat=kJSNativeListPartAccessory
  Switch iCurrentListName
   Case 'iProductList'
    Do iProductList.$line.$assign(pGroup)
    Do iProductList.subList.[pRow].amount.$assign(iProductList.subList.[pRow].amount+1)
    Calculate string as iProductList.[pGroup].subList.[pRow].amount
    Do iProductList.[pGroup].subList.[pRow].buttontext.$assign(jst(string,'-4P0'))
    Do $cinst.$objs.orderBtn.$visible.$assign(kTrue)
   Case 'iGroupList' ## click on the checkout butto
  End Switch
 Else
  Do $cfield.$setMenu(1)
 End If
```


Ejercicio 18: Ocultar el encabezado si no es necesario

No necesitamos mostrar el botón de retroceso ni el encabezado si estamos en la primera tabla que muestra la lista de tablas. Por lo tanto, vamos a ocultar el top_container cuando estemos en el primer nivel:

1. Vaya al método $setMenu de la cuadrícula.
2. Al final del método, ingrese:

   Do $cinst.$objs.top_container.$visible.$assign(iMenuLevel>1)
3. Ahora también establezca la propiedad $visible del top_container en kFalse porque no queremos que se muestre inicialmente cuando se abra el formulario por primera vez.
4. Pruebe su forma.

Parte 3: hacer el botón de pedido

Escribir el registro en la base de datos

Ejercicio 19: manejador de eventos para actualizar el pedido

Necesitamos un botón para enviar a la cocina los pedidos que han sido marcados con los botones de pedido. Guardemos esa información en la base de datos.

1. Agregue un botón en el contenedor superior al lado del campo de encabezado.
2. Asignele "orderBtn" como nombre y "order" como texto a mostrar.
3. Haga el botón inicialmente invisible con la propiedad "visible" en kFalse.
4. Verifique posicionamiento en todos los puntos de interrupción del diseño.
5. Haga doble clic en el botón en modo de diseño para abrir el método $event.
6. Agregue una variable local "ok" de tipo boolean.
7. Después de activar evClick, ingrese un bucle For; usaremos iProductList.$line como contador del bucle que cuenta las iteraciones. El valor inicial será "1" y el valor final el número total de líneas en la lista: iProductList.$linecount:

```
For iProductList.$line from 1 to iProductList.$linecount step 1
 Do iOrderRow.$save(iProductList.sublist,iTableList.$line) Returns ok
End for
If ok
 Do $cfield.$visible.$assign(kFalse)
 Do iTableList.$updateStatus(iTableList.$line,kTrue)
 Do $cinst.$objs.grid.$setMenu(-1)
Else
 Do $cinst.$showmessage(iOrderRow.$getErrorText ())
End if
```

   Nota: El método $save de iOrderRow recibe como primer parámetro la sublista anidada en iProductList que contiene la cantidad pedida y como segundo parámetro el número de tabla que resulta ser el número de línea de iTableList.

   Si el pedido fue exitoso, hacemos que el botón ($cfield) sea invisible para evitar que se presione de nuevo y actualizamos el estado de la tabla usando un nuevo método $updateStatus que aún no se ha implementado. Por lo tanto, no será mostrado por el Asistente de código cuando empecemos a escribir el código.

   $cinst.$showmessage es una forma de mostrar un mensaje en el navegador. Utiliza el método $getErrorText

8. En el método $event de la cuadrícula, debemos hacer que el botón de pedido sea visible cuando se haya hecho clic sobre un botón de pedido de la lista. Para ello, agregue una línea de código después de asignar el texto del botón (y antes del siguiente comando "Case"):

```
Do $cinst.$objs.orderBtn.$visible.$assign(kTrue)
```


Variables de enlace

Puede usar variables de enlace para conseguir que el objeto de acceso a datos evalúe el contenido de esa variable. Por ejemplo, al enviar una instrucción SELECT en la cláusula WHERE, necesitaría poner cualquier texto entre comillas simples:

SELECT * FROM CUSTOMERS WHERE NAME = 'Scotty'

Si usted tuviera el texto "Scotty" dentro de una variable "MyVar", usaría corchetes para evaluar la variable:

SELECT * FROM CUSTOMERS WHERE NAME = [MyVar]

Desafortunadamente, se perderían las comillas simples, ya que no forman parte de los contenidos variables. Por lo tanto, necesitaría hacer algo como esto:

SELECT * FROM CUSTOMERS WHERE NAME = '[MyVar]'

La mejor manera es usar una variable de enlace, lo que se hace agregando el prefijo @ justo antes del primer corchete:

SELECT * FROM CUSTOMERS WHERE NAME = @[MyVar]

Esto instruye a Omnis que **no** evalúe la variable, sino que transfiera esa información al Omnis Data Access Module (DAM) que esté utilizando. Haciendolo así es el DAM quien decide cómo formatear el contenido variable apropiadamente para la base de datos específica empleada.

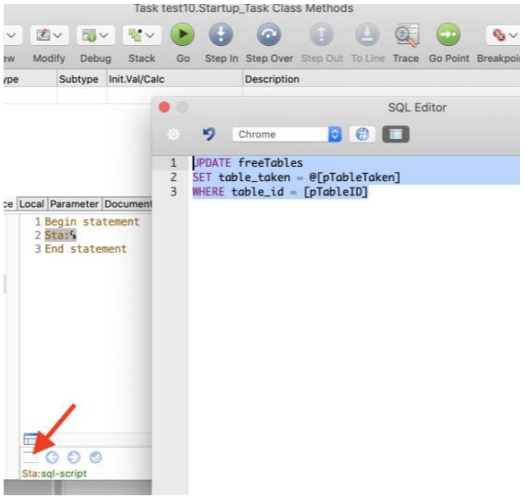
★ Esto es especialmente útil cuando se usen variables de fecha ya que el DAM usa el formato de datos específico para la base de datos en uso.

Editor de SQL

En Omnis se puede ingresar código SQL en un llamado bloque de instrucciones. Hay comandos como "Begin statement "y "End statement". Entre esos dos comandos, hay normalmente una o más líneas que comienzan con „Sta:“

Para facilitar la entrada de este código, puede abrir el Editor de SQL cuando el cursor esté en una línea que comienza con "Sta:". Simplemente haga clic en el pequeño botón en el lado izquierdo y en la parte inferior del editor.

Cuando cierre el editor, transferirá el SQL al código Omnis y viceversa cuando vuelva a abrir el editor.



Editor de SQL

Ejercicio 20: escribir código para actualizar el estado de la tabla

Necesitamos desarrollar el método $updateStatus que llamamos en el método del controlador de eventos del botón de pedidos

1. Cree el método "$updateStatus" para las tabla taTables. Debe estar en la carpeta DataLayer. Puede hacer doble clic en ella para abrir el editor.
2. Agregue un nuevo método "$updateStatus" debajo del método $load. Para ello puede usar el menú contextual dentro de la lista de métodos.
3. En este método ingresar dos parámetros:

   pTableID - Integer - 32bit
   pTableTaken - Boolean
4. Agregue una línea de comando: "Begin Statement"
5. En la línea siguiente, agregue el comando "Sta:". Nota: puede usar el pequeño botón en el borde inferior de la ventana de código para abrir el editor SQL para ingresar el siguiente código SQL. Cuando cierres el editor, el código SQL se transferirá automáticamente a los comandos de instrucción en Omnis:

   ```
   UPDATE freeTables
   SET table_taken = @[pTableTaken]
   WHERE table_id = [pTableID]
   ```

   Nota: Las variables enteras como pTableID no necesitan el uso de una variable de enlace porque el servidor acepta valores numéricos de todos modos.
6. Si la "End Statement" aún no está digitada, utilice el puntero del mouse para agregar otra línea y evitar el comando automatico "Sta:":

   End Statement
7. Las siguientes líneas verificarán el código SQL y lo ejecutarán. Si la preparación falla, llamará al método de $sqlerror. Además, cambiará el texto de la lista según el valor del parámetro „pTableTaken“:

   ```
   If $cinst.$statementobject().$prepare()
    Do $cinst.$statementobject().$execute()
    Calculate $cinst.c2 as pick(pTableTaken,'table free','guests on table')
    Quit method kTrue
   Else
    Do $cinst.$sqlerror()
    Quit method kFalse
   End If
   ```

   Nota: la instancia de una clase de tabla tiene un objeto $statement incorporado. Podemos referirnos a el usando $cinst $statementobject. Los paréntesis después del objeto Statement se usan para resolver esta parte de la notación primero. Luego usa el método: $prepare().

$prepare usa el bloque Statement para verificar la sintaxis y devuelve false si se detecta error. Por eso, en caso de error, llamamos al método $sqlerror que tenemos en la supertabla taSuper.

Si $prepare tiene éxito, el $execute del objeto statement envía la declaración al servidor de base de datos para ejecución.

Para su comodidad, aquí está el código completo tal como debía haberlo ingresado en el método $updateStatus de la clase tabla taTables:

```
Begin statement
 Sta: UPDATE freeTables
 Sta: SET table_taken = @[pTableTaken]
 Sta: WHERE table_id = [pTableID]
End statement
If $cinst.$statementobject().$prepare()
 Do $cinst.$statementobject().$execute()
 Calculate $cinst.c2 as pick(pTableTaken,'table free','guests on table')
 Quit method kTrue
Else
 Do $cinst.$sqlerror()
 Quit method kFalse
End If
```


Ejercicio 21: probar el botón de pedido

Ahora ya debería poder escribir órdenes en la base de datos. Si uno de los siguientes pasos falla necesitaría colocar un breakpoint en su código y descubrir lo que está mal.

1. Abra la forma remota en su navegador
2. Seleccione una de las mesas.
3. Elija “Bebidas” o “Comida” y use el botón de pedido de la lista para pedir artículos. Este botón se supone que muestra el número de pedidos. Debería aparecer el botón de orden principal en el encabezado.
4. Luego haga clic en el botón de orden en el encabezado. Esto debería escribir las órdenes en la tabla del servidor. El botón de pedido debería desaparecer y la cuadrícula debería mostrar el nivel de menú anterior.
5. Haga clic en "Resumen". Ahora debería ver la orden que se ha realizado para esta mesa.

Parte 4: Implementación de la identidad corporativa y desarrollo de una segunda forma remota para cocina y personal del bar

Implementación de la identidad corporativa utilizando una forma remota como superclase

Ya que vamos a usar más de una forma remota, podemos usar una forma remota superclase para implementar la Identidad Corporativa (CI). Todos los métodos y campos que implementemos en esa clase pueden ser heredados por otras formas remotas. De esa manera, cualquier funcionalidad y UI estarán disponibles en todas las clases heredadas y solo necesitamos implementar esto una vez.

Ejercicio 22: hacer una forma que muestre la identidad corporativa

Ahora vamos a agregar una forma remota que mostrará una imagen dentro de la ruta /html/images de la carpeta AppData (Windows) o Application Support (Mac OS).

1. Cree una forma remota "jsSuper" en su librería.
2. Abra jsSuper en modo de diseño y agregue un PagedPane. Será usado para asignar algunos espacio en la parte superior del formulario.
3. Llámelo "top_pane" en la propiedad nombre.
4. Asigne kEFposnMenuBar en la propiedad "edgefloat". Asegúrese de haber seleccionado "Establecer para todos puntos de corte de diseño ".
5. Agregue un control de imagen en el top_container en la esquina izquierda. Llámelo "picture".
6. En el punto de corte de diseño 320, elija un ancho y una altura de 70 para el componente de imagen. Asegúrese de que la altura del top_container sea de 70 como mínimo.
7. Para el diseño más grande, puede hacer que la imagen y el top_container sean un poco más grandes ya que hay más espacio disponible.
8. Vaya a la propiedad "dataname" del componente picture. Ingrese iPictureURL. Cuando digite entrar se le preguntará si desea declarar esta variable. El alcance se fija en "instancia" porque este es el único alcance que se puede usar dentro de "dataname". El tipo "caracteres" está bien, así como el subtipo que es la longitud. No cambie nada allí. Pero agregue lo siguiente en el campo Init Val/Calc: "Images/cp/cucina_form_top.png"

   Nota: Es la ruta relativa a la carpeta "html". Recuerde que ha copiado la carpeta "cp" que contiene la imagen "cucina_form_top.png" a la carpeta /html/images. Puede reemplazar esta imagen con su propio archivo png o jpeg si lo desea.
9. Busque la propiedad "noscale" para el campo de imagen y configúrelo en kFalse. De esta manera la imagen será escalada al tamaño del campo imagen.
10. Compruebe si "keepaspectratio" está configurado en kTrue. Esto asegurará que la relación de aspecto de la imagen sea la correcta.
11. Establezca "disablesystemfocus" en kFalse. De lo contrario, pondría un marco alrededor de la imagen cuando el usuario la selecciona.
12. Pruebe el formulario. Debería mostrar la imagen (si la ruta es correcta) en diferentes tamaños cuando cambie el tamaño de la ventana del navegador.

Ejercicio 23: hacer que jsSuper se convierta en superclase

Ahora haremos que jsSuper se convierta en la superclase de la forma remota jsService en la que trabajamos.

1. Cierre la ventana de diseño y el editor de métodos de jsService si aún está abierto.
2. Seleccione jsService y vaya a la propiedad "superclase".
3. Asigne "jsSuper" en la propiedad "superclase".
4. Abra jsService en modo de diseño. Verá que ahora contiene el "top_container", incluido el campo "imagen" de la superclase.
5. Pruebe jsService. Ahora debería mostrar la identidad corporativa allí también.

Agregar un forma remota para personal de cocina y bar

El pedido ahora se guarda en la base de datos desde jsService. Lamentablemente, no tenemos una forma remota que enumere todas las órdenes abiertas para que el personal de cocina pueda verlas y proporcionar la comida o bebidas

Ejercicio 24: agregar la forma remota para cocina y bar

Ahora vamos a hacer una segunda forma remota que heredará de jsSuper. Debe-mostrar todas las órdenes abiertas y queremos que permita mostrar todas las órdenes, solo las de bebidas o solo las órdenes de comida y que además permita marcar los pedidos como "servidos".

1. En el navegador de Omnis Studio, haga clic con el botón derecho en jsSuper y seleccione "Hacer Subclase ". Esto crea una nueva forma remota subclase de jsSuper. Llámelo "jsOutlet".
2. Agregue un componente de "datagrid" desde el Almacén de componentes. Se utilizará para listar los pedidos.
3. Nómbrelo "cuadrícula"
4. Escriba "iDataList" en la propiedad "dataname". Se le pedirá que declare esta variable. El tipo será "Lista" y el subtipo: taOrderList
5. Haga doble clic en el fondo del formulario para abrir los métodos de la clase. Va a observar que ha heredado métodos, es decir, $construct se muestra en azul para indicar que es parte de la super clase.
6. Agregue un nuevo método a esta clase "$load"
7. Agregue un comando "Do" al método $load.

   Do iDataList.$load(iType)

   Nota: Puede notar que iType está subrayado con una línea roja rizada. Eso nos dice que la expresión no es válida; en ese caso, es porque la variable aún ha sido declarada. Puede que desee llevar el cursor a este texto ya sea con el mouse o usando una de las teclas de flecha en la parte inferior y luego usando el pequeño botón de verificación al lado de los botones de flecha. Esto abre un cuadro de diálogo que permite declarar la variable Debe ser una variable de in-stancia de tipo Integer. Puede elegir "Short (0-255) ".
8. Queremos ejecutar el método $load de jsOutlet cuando se abra el formulario. Por lo tanto lo que haríamos sería llamarlo desde el método $construct. Ya que este método es heredado, podemos anularlo. Para ello, seleccione $construct de jsOutlet y abra su menú contextual. Ahí encontrará el "Override Method". Nota: el nombre del método se vuelve negro y el texto del método es accesible.
9. Ingrese el comando "Do inherit". Tenga en cuenta que esto garantiza que el código heredado de la super clase se ejecute en este método.
10. En la siguiente línea ingresamos:

    Do $cinst.$load()

    Nota: Esto llama al método $load que acabamos de escribir.
11. Ahora vaya a la ventana de diseño, es decir, Shift F8 (Windows) o Cmd-Shift-8

(Mac OS)

12. Seleccione el componente cuadrícula y establezca la propiedad "designcols" en 7. Esto hará que la cuadrícula muestre siete columnas.
13. Establezca la propiedad "userdefined" en kTrue. De esta manera tiene control total para cada columna por separado.

    Nota: una de las columnas de la cuadrícula ahora debe tener un marco de selección rojo en la ventana de diseño. Puede seleccionar cualquiera de las columnas para cambiar las propiedades de cada una.
14. Seleccione la primera columna y busque el panel "columna" en el Administrador de propiedades. Asigne "iconpath" a la propiedad "columndatacol" y "status" en "columnname".
15. Ahora configure las otras columnas en consecuencia:

Columna	$columndatacol	$columnname
1	iconpath	status
2	order_table_num	table
3	order_date	order at
4	product_id	product
5	product_name	text
6	order_size	size
7	order_amount	amount

16. Pruebe su forma. Ahora debería ver los datos en la cuadrícula, pero note que la primera columna muestra un nombre de ruta que hace referencia a un icono pero todavía no lo muestra. Usted necesita establecer la propiedad "columnstyledtext" en la pestaña "column" de la primera columna de la cuadrícula a kTrue. Con otra prueba debería mostrar ahora viñetas rojas, verdes o grises como ícono de estado.
17. Cambie la propiedad "height" de la cuadrícula a 2500 para cualquiera de los diseños y cambie el tamaño de ancho y posición para que se ajuste al tamaño del diseño.
18. Establezca la propiedad "edgefloat" en kEfRight. Esto aumentará la cuadrícula automáticamente para tamaños intermedios como teléfonos más grandes o si usa un navegador en pantallas más grandes.
19. Cambie la propiedad "rowheight" a 40. Esto permite un uso más fácil si usa pantallas táctiles.

Ejercicio 25: configurar un filtro para la lista de pedidos

En el último ejercicio, agregamos una variable de instancia para determinar si todas las órdenes estarán en la lista o solo bebidas o solo comida.

1. Agregue un componente "radio group" de la tienda de componentes a la forma entre el top_container y la cuadrícula. Puede ser que desee mover la cuadrícula un poco hacia abajo.
2. Nómbrelo "selector"
3. Ponga "iType" en la propiedad "dataname" del radio group.
4. En la propiedad "texto", ingrese: "Todas, solo bebidas, solo comida"
5. Cambie la propiedad ":: horizontal" en la pestaña Apariencia a kTrue
6. Establezca "columncount" en 3
7. En la propiedad $event habilite el evento evClick.
8. Haga doble clic en el campo y debería estar en su método $event.
9. Debajo de la línea "On evClick", ingrese:

   Do $cinst.$load()
10. Cuando pruebe, debe poder cambiar entre todas, solo bebidas y solo comida.

Ejercicio 26: Marcar pedidos como servidos

Si el personal prepara un pedido, sería bueno que pueda marcar pedidos individuales como "servidos" para que cambie el estado. Para esto implementaremos un manejador de eventos en la cuadrícula.

1. Habilite el "evento" evClick para la cuadrícula en el Administrador de propiedades.
2. Haga doble clic en el control cuadrícula para acceder a su método $event.
3. Elimine todos los comandos "On" excepto el que tiene "On evClick". No necesitamos ningún otro evento.
4. Abra el Catálogo para ver los parámetros del evento. Existe el parámetro de evento "pVertCell" que informa en qué línea ha hecho clic el usuario.
5. Debajo de "On evClick" ingresaremos:

   Do iDataList.$line.$assign(pVertCell)

   Nota: esto es para hacer que la linea corriente de la lista use el número de línea que obtenemos del parámetro del evento.
6. Ahora necesitamos descubrir si el usuario ha hecho clic en la primera columna, la columna de "estado" que contiene el ícono:

   ```
   If pHorzCell = 1
    Do $cfield.$updateOrder()
   End if
   ```

   Nota: $cfield es la referencia al mismo campo: la cuadrícula, donde también está el método $event. Actualmente no tenemos un método "$updateOrder" en esa cuadrícula, por lo que no recibiremos ayuda del Ayudante de notación.
7. Agregue un método a la cuadrícula utilizando el menú contextual en la "cuadrícula" en la lista de métodos. Nombre este método como "$updateOrder".
8. Agregue una variable de instancia "iOrder" de tipo "row" y de subtipo "taOrders".
9. Agregue la siguiente línea al $updateOrder:

   If iOrder.$updateStatus(iDataList.order_id)

   Nota: Podemos omitir el número de línea de la lista porque establecemos la línea actual en el método $event de la cuadrícula.
10. Luego agregue:

    Do $cinst.$load ()
11. Añada un comentario. Puede hacer esto escribiendo un signo "#". Puede ingresar: “escribiremos codigo para push posteriormente". Esto es para recordarnos que agregaremos código más adelante.

12. Agregue una línea de comando "End if".
13. Para su conveniencia aquí está el código completo del $updateStatus de cuadrícula

```
If iOrder.$updateStatus(iDataList.order_id)
 Do $cinst.$load() ## reload this instance
 # we do a push here later
End If
```


Ejercicio 27: solucionar un problema de redibujo

Al probar el clic en el icono, puede notar que la cuadrícula se desplaza hasta el final de la lista. Esto es porque el método $load de la clase recarga la lista completa y eso hace que la última línea sea la línea actual. Por lo tanto, el componente cuadrícula intenta mostrar la línea actual.

1. Para solucionar este problema, vaya al método $load de jsOutlet.
2. Agregue una variable local "currentLine" de tipo Integer y subtipo 32bit.
3. Agregue una nueva línea encima de "Do iDataList.$load(iType). Puede hacer esto cuando selecciona la línea y presione Ctr.-i (Windows) o Cmd-i (Mac OS).
4. Ingrese: Calculate currentLine como iDataList.$line

   Nota: La variable local se usará para recordar el número de línea antes de volver a cargar la lista.
5. Después de volver a cargar la lista, podemos asignar el número de línea:

   Do iDataList.$line.$assign(currentLine) o alternativamente:
   Calculate iDataList.$line as currentLine
6. Pruebe su código. Las órdenes "rojas" aún no se entregan. Cuando hace clic en una viñeta roja, debería cambiar el estado de su pedido en la base de datos y volver a cargar la lista y luego mostrar el pedido usando una viñeta amarilla en su lugar. Si se hace clic en un icono amarillo, se invertirá y el icono se convertirá rojo de nuevo. Esto permite deshacer una acción si se hizo clic accidentalmente en la orden equivocada.

Para su comodidad, aquí está el código completo del método $load de la clase jsOutlet:

```
Calculate currentLine as iDataList.$line
Do iDataList.$load(iType)
Calculate iDataList.$line as currentLine
```


Parte 5: Agregar un servicio push para sincronizar todos los dispositivos, imprimir un reporte e instalar el Servidor de aplicaciones Omnis

Servicio push

El servicio push le permite activar un cierto método $pushed a una forma remota. Para hacer que una instancia esté lista para recibir cualquier push se necesita habilitar este comportamiento, generalmente se lo hace dentro del método $construct de la forma remota que va a recibir el envío:

Do $cinst.$Clientcommand('openpush',row())

El segundo parámetro del comando $client es obligatorio pero "openpush"no lo necesita. De ahí que toca pasar una función row() vacía.
Luego puede referenciar la instancia de la forma remota y enviarle un mensaje $pushdata(row ()). Típicamente necesitaría enviar como parámetro una variable de fila. Esta variable de fila debe ser declarada como parámetro de tipo row en el método $push de la instancia receptora. La variable row es obligatoria, de modo que si no necesita enviar ningún parámetro, debe usar una row vacía ().

Puede usar $iremotetasks.[NameOfTheTask].$Iremoteforms.[NameOfTheForm] para referenciar una instancia de forma remota específica dentro de una instancia de la tarea remota específica.

Alternativamente, si desea enviar mensajes a todas las instancias de una forma remota específica, usted puede usar: $iremoteforms.$ sendall($ref.$ senddata (row))

Ejercicio 28: habilite el servicio push en jsSuper

Para recibir notificaciones push, necesitamos activar esta recepción en nuestras dos formas remotas ya que cualquiera de ellas podrían recibir este push. Imaginese que recargaríamos automáticamente la lista de pedidos en el formulario jsOutlet, así como recargaríamos la lista de resumen en el formulario jsService cuando el estado de un pedido haya cambiado en la salida.

1. Agregue un nuevo método $load en la superclase jsSuper.

Nota: este se utilizará como el contenedor para definir nuestra interface en las clases heredadas.

2. Vaya al método $construct de jsSuper y active el servicio push:

```
Do $cinst.$clientcommand('openpush',row())
```

3. En la siguiente línea ingrese:

```
Do $cinst.$load()
```

4. Vaya al método $construct de jsOutlet. Ya no necesitamos nada de este código en adelante. Por tanto, herede el método completo utilizando el menú contextual en el nombre del método y seleccionando "Heredar método". Se le preguntará si desea anular este código. Por favor confirmar. El nombre $construct de este método aparecerá ahora en azul.

5. Abra el editor de métodos de ls forms remots jsService. Anule el método azul $load de allí usando el menú contextual en el nombre del método. El nombre del método ahora debería estar en negro.

6. Vaya al $construct de la forma remota jsService. Verá que ahí existe el comando que llama al $load de iTableList. Corte esta línea del $construct y péguela en el método $load que hemos desheredado.

7. Luego herede el método $construct como lo hizo anteriormente en jsOutlet.

Nota: Ambas formas remotas que heredan de jsSuper están ahora listas para recibir un push. Adicionalmente hemos movido código para cargar los datos iniciales en el método $load de ambas clases.

Ejercicio 29: envío del push

Ahora estamos listos para enviar el push.

1. Vaya al botón "ordenar" de jsService. En el método $event justo después de llamar al $setMenu (-1) del componente cuadrícula agreguemos otra línea:

   Do $iremoteforms.$sendall($ref.$pushdata(row()))

   Nota: $ref se refiere a cada instancia individual cubierta por el método $sendall. Así que básicamente vamos a enviar $pushdata a todas nuestras instancias de formas remotas.

   Nota 2: Dado que no queremos pasar ninguna información, simplemente podemos enviar una fila vacía usando la función row ().

2. Vaya al método $updateOrder del componente cuadrícula dentro de jsOutlet. Justo después del "Do $cinst.$load()”agreguemos otra línea que también activa el push

   Do $iremoteforms.$sendall($ref.$pushdata(row()))

   Nota: esto actualizará los pedidos en los otros terminales, así como las formas remotas para personal de servicio, es decir, la lista resumen.

3. Agregue un nuevo método a jsOutlet: $pushed

   Nota: El nombre del método tiene ahora un color rosa. Esto nos dice que va a ser un método ejecutado donde el cliente lo cual es obligatorio para el método $pushed.

4. Agregue una línea de código:

   ```
   Do $cinst.$load()
   ```

   Nota: Entonces, cuando se recibe el push, el método $load se ejecutará automáticamente y la cuadrícula de datos se volverá a cargar.

5. Ahora regrese a la forma remota jsService y agregue un nuevo método de clase privada "loadSummary".

6. En el componente cuadrícula, vaya al método $setMenu y busque la línea

   "Do iOrderRow.$getGroupedList(iTableList.$line) Returns iSummaryList ".

   Copie esta línea en el método "loadSummary" y reemplácelo dentro de $ setMenu con el comando:

   Do method loadSummary

   .. para llamar a este método.

   Nota: El comando "Do method" es diferente del comando "Do" y se usa para llamar métodos privados

7. Ahora podemos agregar un método $pushed en la clase jsService y llamar al $loadSummary también ahí.
8. Actualice el navegador para ambas formas remotas para volver a cargar el código que va a ser ejecutado por el cliente.

   Nota: esto es necesario porque el código ejecutado por el cliente se traduce a JavaScript y por lo tanto, debería volver a cargarse desde el navegador cuando se haya cambiado el código.

9. Prueba tu trabajo.

Cuando ahora haga un pedido en la forma remota jsService y presione el botón principal de pedido (order), dicho pedido debería aparecer mágicamente en la forma remota jsOutlet si está abierta.

También cuando su forma remota de servicio muestre la lista de resumen de una mesa y en la forma outlet uno de los pedidos sea marcado como servido, el color del icono para este pedido debería cambiar en la lista resumida de la otra forma remota.

Finalmente, también podría abrir varias instancias de las formas remotas y ver esos cambios en cualquiera de ellas.

Ejercicio 30: Marcar un pedido como cancelado

Es posible que haya notado que existe un botón deshacer en la lista de resumen de jsService. Este botón aparece mientras el pedido no haya sido entregado. Permite al usuario cancelar un pedido sujeto a confirmación del personal de outlet.

1. Vaya al método $event del componente cuadrícula dentro de la forma remota jsService.
2. Necesitamos agregar otro "Case" para el "iSummaryList":

   Case "iSummaryList"
3. Luego agregue una línea para calcular la línea actual dentro de la lista iSummaryList:

   Do iSummaryList.$line.$assign(pGroup)
4. Ahora usaremos una forma indirecta de llamar a la solicitud de cancelación. Agregue el siguiente código:

   If iOrderRow.$SetCancelRequest(iSummaryList.sublist.[PRow].order_id….

   Note que obtenemos el código de producto de la lista anidada en la lista resumen.

   Note que el número de línea viene del parámetro pRow y debe estar en corchetes para hacer que Omnis lo evalúe.

   El segundo parámetro es booleano e indica si se acepta la solicitud de cancelación (0) o se la rechaza (1).

   Podemos usar el texto del botón para averiguar el estado:

   iSummaryList.sublist.[pRow].buttontext = "redo"

   Aquí está la línea de código completa:

   If iOrderRow.$setCancelRequest(iSummaryList.sublist.[PRow].order_id, iSummaryList.sublist.[pRow].buttontext='redo')
5. La siguiente línea es para asignar un nuevo valor para el texto del botón:

   Calculate iSummaryList.subList.[PRow].buttontext as pick(iSummaryList.subList.[pRow].buttontext= 'undo', 'undo', 'redo')

   Nota: Utiliza la función pick () para cambiar el valor.
6. Ahora agregue una línea que envíe un push a todos los demás clientes:

   Do $iremoteforms.$sendall($ref.$pushdata(row ()))
7. Agregue un comando "End if" para cerrar la instrucción If.
8. Vuelva a cargar el navegador y pruebe su formulario.

Si todo va bien, ahora debería poder hacer un pedido de cancelación en la lista de resumen de la forma remota de servicio. El color del ícono del pedido debe volverse gris en ambas formas. La solicitud de cancelación ahora se puede confirmar desde la forma outlet al hacer clic en el icono gris. La orden entonces será eliminada de todas las formas.

Agregar un reporte

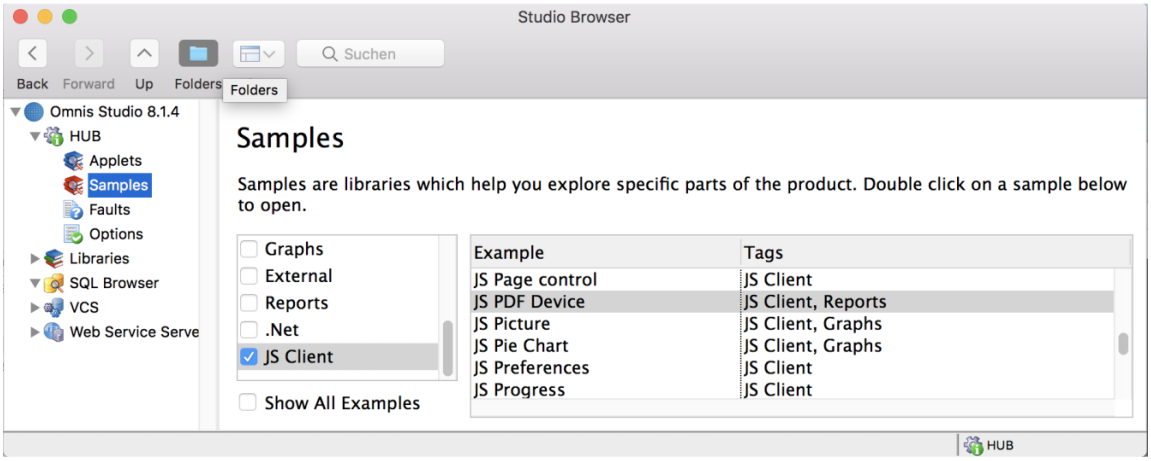
Puede crear clases de reportes que le permitan imprimir cualquier dato. En este ejemplo estamos usando un reporte diseñado que ya está listo, pero siéntase libre de personalizarlo si lo desea.

Hay comandos para establecer el destino donde desea imprimir el reporte.

```
Prompt for destination
If flag true
    Select printer
    Set report name rTableBill
    Print report
End If
```

Comandos print

Por lo general, en una aplicación web típica, no imprimiría un reporte directamente en la impresora. En cambio usted deseara generar un archivo PDF y mostrarlo dentro de la aplicación. Esto puede hacerse utilizando el dispositivo JS PDF. También hay una librería de ejemplo disponible en la sección Muestras del Omnis HUB.



Para la aplicación Cucina Piccola está bien imprimir la factura directamente en el servidor. Asumimos que el servidor es solo una computadora ubicada en el restaurante y que tiene conectada una impresora .

Ejercicio 31: hacer el check out

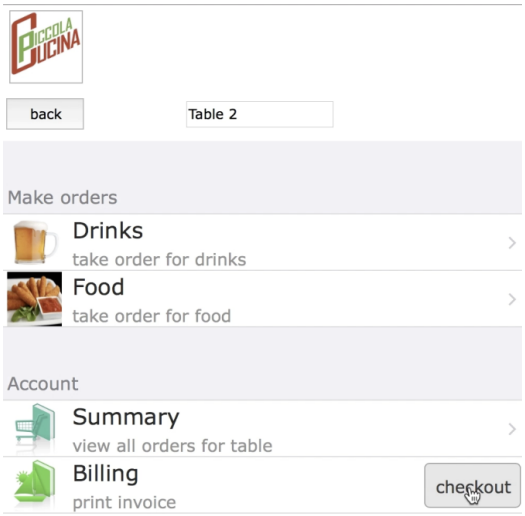
Finalmente, nos gustaría permitir que el cliente realice el pago. El sistema creará automáticamente la factura con la lista de pedidos de la mesa específica. Ya hay un botón de "pago" (checkout) en el menú.

1. Asegúrese de copiar la clase reporte"rTableBill" desde resource.lbs.
2. Agregue un nuevo método "checkOut" a la jsService.
3. Agregue una nueva variable de instancia "iInvoiceRow" de tipo Row y subtipo "taInvoices"

   Nota: asegúrese de tener esta clase tabla y la clase esquema que se usa "invoices" copiándolos de la librería de recursos.
4. Agregue el siguiente código:

   If iInvoiceRow.$save(iTableList.$line)
5. La siguiente línea va a actualizar el estado de la mesa:

   Do iTableList.$updateStatus(iTableList.$line,kFalse)



Menú con botón check-out

Nota: el segundo parámetro es para liberar el bloqueo de la mesa.

6. Agregue otro comando "Set report name" y use rTableBill como nombre del reporte.
7. La siguiente línea es el comando "Print report". El asterisco le dice a Omnis que genere automáticamente un nombre de instancia. Luego, pase como parámetro la identificación de la factura:

   Print report * (iInvoiceRow.$getIdent())
8. En la siguiente línea, ejecute un push:

   Do $iremoteforms.$sendall($ref.$pushdata(row()))

   Nota: esto es para que las otras formas remotas de servicio vean que el pedido ya ha sido pagado y que la mesa está disponible de nuevo.
9. Ahora agreguemos un comando "Else". Es para manejar el caso de que la row de la factura no haya podido realizar el $save por cualquier motivo.
10. Agregue un mensaje al cliente:

    Do $cinst.$showmessage(iInvoiceRow.$getErrorText())

11. Agregue un "End if" para cerrar la declaración.

Aquí está el código completo del método checkOut:

```
If iInvoiceRow.$save(iTableList.$line)
  Do iTableList.$updateStatus(iTableList.$line;kFalse)
  Set report name rTableBill
  Print report * (iInvoiceRow.$getIdent ())
  Do $iremoteforms.$sendall($ref.$pushdata(row()))
Else
  Do $cinst.$showmessage(iInvoiceRow.$getErrorText())
End if
```

12. Vaya al método $event de la cuadrícula en jsService.

13. Si aún no lo ha hecho, agregue otro "Case" para el pago:

```
Case "iGroupList"
  Do method checkOut
```

14. Probar su trabajo!

Ahora debería poder imprimir facturas para una mesa que tenga pedidos. Los iconos de los pedidos en la forma outlet deberían cambiar de verde y desaparecer de esta lista al día siguiente. Esta mesa debe mostrarse como disponible en la lista de mesas de las otras formas remotas de servicio.

Servidor de aplicaciones

Arquitectura web de Omnis Studio

El Omnis Studio Server Runtime (App Server) tiene un servidor HTTP incorporado que le permite a Omnis comunicarse con un servidor web a través de un módulo Apache o CGI.

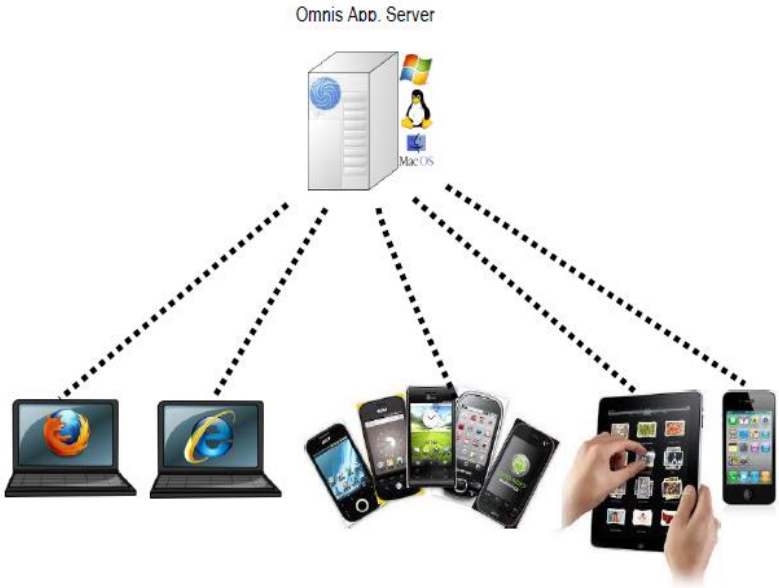
Esta arquitectura es la mejor práctica para aplicaciones web y móviles que acceden al servidor de aplicaciones Omnis a través de Internet.

Puede escalar el servidor de aplicaciones Omnis mediante un proceso de carga compartida (LSP). En ese caso, el LSP recibe las solicitudes HTTP del servidor web y distribuye la solicitud al menos ocupado Omnis App Server.

Nota: Siga los pasos de la siguiente nota técnica en nuestro portal de desarrolladores para configurar el servidor de aplicaciones Omnis junto con el servidor web: http://developer.omnis.net/technotes/tnjs0003.jsp

Omnis Studio Intranet Architecture

Para pequeñas aplicaciones internas, como Cucina Piccola, podemos usar el servidor de aplicaciones Omnis directamente sin usar un servidor web "real". Omnis funciona entonces como un servidor HTTP y puede directamente tener acceso a él desde su red de área local.



Ejercicio 32: instalar y configurar el servidor de aplicaciones

Si desea que su aplicación Cucina Piccola se ejecute en su restaurante sin usar la versión de desarrollo, necesitaría instalar un servidor de aplicaciones Omnis.

Tenga en cuenta que para probar y depurar puede usted usar la versión de desarrollo de Omnis Studio, pero para la implementación, debe usar el servidor de aplicaciones Omnis y adquirir una licencia diferente.

1. Descargue e instale el servidor de aplicaciones Omnis desde: http://www.omnis.net/developers/downloads/
2. Elimine cualquier librería innecesaria de la carpeta /Startup del Omnis App Server y coloque en esa carpeta su propia librería y la base de datos (cp.lbs + cucina_piccola.db). Esto garantiza que su librería se inicie cuando se inicie el servidor de aplicaciones Omnis.

   Nota: para ejecutar el Omnis App Server en modo de subprocesos múltiples, deberá ejecutar el comando "Start server" por ejemplo en el $construct del Startup_Task de su librería. Consulte el manual "Aplicaciones web y móviles" en el Portal del desarrollador para más información.
3. Al iniciar el servidor de aplicaciones Omnis, se le solicitará su número de serie, número que puede comprar de Omnis Software.
4. Configure su servidor. Se lo puede hacer a través del menú "Archivo" o mediante el archivo config.json que se encuentra dentro de la carpeta /Studio del servidor de aplicaciones Omnis. Asegúrese de usar un puerto que otras aplicaciones no usen en su máquina (por ejemplo, 5912).
5. Coloque todos los recursos que se requieran dentro del directorio /HTML del servidor Omnis. Recuerde que copiamos las imágenes necesarias en una subcarpeta de /html/images en la versión de desarrollo al comienzo de este curso. En realidad se necesitan los mismos recursos dentro del servidor de aplicaciones.

   Nota: la carpeta /html se encuentra dentro del "\AppData" oculto (Windows) o en la carpeta "/library/ApplicationSupport" (Mac OS).
6. Finalmente copie los dos archivos HTML (jsService.htm y jsOutlet.htm) de la carpeta /html de la versión de desarrollo de Omnis Studio a la carpeta /html del servidor de aplicaciones Omnis.
7. Pruebe su trabajo!

Ahora debería poder conectar sus formas remotas del servidor de aplicaciones Omnis utilizando el siguiente formato. Supongamos que la dirección IP del servidor sea 192.168.0.10 y el número de puerto que ha elegido es 5912:

http://192.168.0.10:5912/jschtml/jsService.htm

y

http://192.168.0.10:5912/jschtml/jsOutlet.htm

Observaciones

Espero que se haya divertido desarrollando la aplicación Cucina Piccola de este curso.

Si tuvo problemas en algún momento, hágamelo saber, realmente me gustaría ayudarlo. Envíeme un correo electrónico a: andreas.pfeiffer@omnis.net
Además, si continúa desarrollando sus propias aplicaciones, hágamelo saber. Siempre tengo curiosidad por saber de otros desarrolladores que crean usando Omnis Studio.

Mis mejores deseos,

Andreas

