



EBook Gratis

APRENDIZAJE acumatica

Free unaffiliated eBook created from
Stack Overflow contributors.

#acumatica

Tabla de contenido

Acerca de.....	1
Capítulo 1: Empezando con acumatica.....	2
Observaciones.....	2
Examples.....	2
Instalación o configuración.....	2
Capítulo 2: Acumatica BQL Reference.....	3
Examples.....	3
BQL Parse y Verify.....	3
Analizar gramaticalmente.....	3
Verificar.....	3
Conclusión.....	4
Capítulo 3: Acumatica Platform Atributos de Referencia.....	6
Examples.....	6
Atributo PXFormula.....	6
Descripción general.....	6
Modos de uso.....	6
Propiedades de PXFormulaAttribute y parámetros del constructor.....	7
Uso.....	8
Orden de los campos.....	8
Contexto de la fórmula y sus modificadores.....	9
Current<TRecord.field> y Current2<TRecord.field>.....	9
Parent<TParent.field>.....	10
IsEmptyTable<TRecord>.....	10
Selector<KeyField, ForeignOperand>.....	10
Obtiene un PXSelectorAttribute definido en el campo de clave foránea (KeyField) del DAC ac.....	10
Obtiene el registro de datos externos actualmente referenciados por el selector.....	10
Calcula y devuelve una expresión en ese registro de datos como lo define ForeignOperand.....	11
Uso de fórmulas en los campos sin límites.....	11
Lista de fórmulas comunes incorporadas.....	11

Referencias circulares directas y mediadas en fórmulas	11
Flujo de control en fórmulas condicionales	11
Uso de múltiples fórmulas en un campo	11
Atributo de PXRrestringido	12
Introducción	12
Detalles	12
Opciones	13
Anulación de los limitadores heredados	13
Caching global	14
Recomendaciones para el uso	14
Utilizar solo condiciones de restricción	14
Capítulo 4: Adición de soporte de atributos a una entidad de pedido de ventas lista para u	16
Introducción	16
Observaciones	16
Examples	16
Este artículo proporciona una guía práctica para agregar el soporte Acumatica ERP Attribut	16
Capítulo 5: Ampliación de la lista de entidades apoyadas por tareas, eventos y actividades	19
Introducción	19
Examples	19
Agregar órdenes de trabajo de prueba al campo de descripción de entidad relacionada	19
Capítulo 6: Cálculo de fletes	25
Introducción	25
Examples	25
Monto de flete anulado en el envío y la factura	25
FreightCalculator	25
Ordenes de venta	25
Envíos	26
Importe de flete anulado	26
Comprensión de la implementación de la clase FreightCalculatorCst en el ejemplo anterior	27
Capítulo 7: Cambio de subtítulos dinámicamente usando campos DAC de solo lectura	29

Introducción.....	29
Examples.....	29
Cómo.....	29
Capítulo 8: Cambio de tamaño de la ventana desplegable del selector.....	32
Introducción.....	32
Examples.....	32
Cambio de rangos de tamaño por defecto para la ventana desplegable del selector.....	32
Para expandir el ancho de la ventana desplegable del selector de clientes.....	32
Capítulo 9: Cambios significativos en la API entre versiones.....	35
Examples.....	35
PXSelectGroupBy y Bit Values en Acumatica 5.1 y 5.2+.....	35
Acumatica Framework 5.2 y posteriores.....	35
Acumatica Framework 5.1 y anteriores.....	35
Explicación.....	36
Capítulo 10: Creando campos de fecha y hora en Acumatica.....	37
Introducción.....	37
Examples.....	37
El atributo DateAndTime de PX (DB).....	37
El atributo PXDBTime.....	38
El atributo de atributo de fecha PX (DB).....	39
El atributo PXDBTimeSpan.....	40
El atributo PXTimeList.....	40
Capítulo 11: Descarga de archivos adjuntos a una entidad de detalle mediante API basada en.....	42
Introducción.....	42
Observaciones.....	42
Examples.....	42
Encabezado de cookie HTTP de una respuesta SOAP compartida por clientes SOAP y REST.....	42
Capítulo 12: Exportación de registros a través de la API basada en pantalla.....	45
Introducción.....	45
Observaciones.....	45
Examples.....	45

Exportación de datos desde un formulario de entrada con una sola clave primaria.....	45
Para exportar todos los artículos en stock en una sola llamada de servicio web:.....	46
Para exportar artículos en stock en lotes de 10 registros:.....	47
Exportación de datos desde un formulario de entrada con una clave primaria compuesta.....	48
Para solicitar todo tipo de pedidos existentes:.....	49
Para exportar registros de cada tipo de forma independiente en lotes:.....	50
Para exportar registros de un tipo específico:.....	51
Capítulo 13: Exportación de registros a través de REST API basada en contrato.....	53
Introducción.....	53
Observaciones.....	53
Examples.....	55
Exportación de datos en una sola llamada REST.....	55
Para exportar todos los artículos en stock en una sola llamada REST:.....	55
Para exportar todos los pedidos de venta del tipo IN en una sola llamada REST:.....	56
Implementación de paginación en múltiples solicitudes REST.....	56
Para exportar artículos en stock en lotes de 10 registros con múltiples llamadas REST:.....	56
Para exportar todos los pedidos de ventas en lotes de 100 registros con múltiples llamadas.....	57
Capítulo 14: Filtrado con valor múltiple con un solo selector.....	59
Introducción.....	59
Examples.....	59
Recuperación de pedido de cliente para cliente múltiple.....	59
Capítulo 15: Mecanismos de personalización.....	62
Examples.....	62
Uso de CacheAttached para reemplazar atributos de DAC en el gráfico.....	62
Reemplazo de todos los atributos.....	62
Anexando un nuevo atributo al campo DAC.....	62
Anulando una propiedad única de un atributo.....	63
Reemplazo de un atributo con otro atributo.....	64
Orden de aplicación de los atributos de personalización de atributos.....	64
Capítulo 16: Modificaciones a la información de contacto y dirección a través del código.....	65

Introducción.....	65
Examples.....	65
Especifique la información de contacto y dirección para un nuevo empleado.....	65
Anular información de contacto de facturación y de dirección de facturación para un client.....	65
Anular información de contacto de facturación y de dirección de facturación para una orden.....	67
Capítulo 17: Modificaciones a las vistas de datos base.....	69
Introducción.....	69
Examples.....	69
APIInvoiceEntry BLC: agregue restricciones adicionales a la vista de datos poReceiptLinesSe.....	69
Capítulo 18: Modificar elementos en una lista desplegable.....	72
Introducción.....	72
Observaciones.....	72
Examples.....	72
Modificación de los estados maritales.....	72
Para agregar nuevos elementos al sucesor de PXStringListAttribute.....	73
Para eliminar elementos declarados en el sucesor de PXStringListAttribute.....	75
Para reemplazar los elementos declarados en el sucesor PXStringListAttribute.....	76
Capítulo 19: Pestañas de ocultación condicional.....	78
Introducción.....	78
Examples.....	78
Propiedad VisibleExp del control PXTab en Aspx.....	78
Para ocultar la pestaña Actividades para clientes potenciales con nuevo estado.....	78
AllowSelect Property on Data Views.....	79
Para ocultar la pestaña Referencia cruzada de los artículos en stock que no se pueden vend ...	81
Para ocultar la pestaña Atributos de los artículos en stock inactivos.....	83
Capítulo 20: Publicación omitida de contenido de personalización ya aplicado.....	87
Introducción.....	87
Examples.....	87
Publica con limpieza desde la pantalla de personalización.....	87
Publicar con la limpieza desde dentro de un proyecto de personalización.....	88
Capítulo 21: Reemplazo de imágenes en la página de inicio de sesión.....	90

Introducción.....	90
Examples.....	90
Uso de personalización para reemplazar imágenes en la página de inicio de sesión.....	90
Capítulo 22: Rellenando informe con datos a través de código.....	95
Examples.....	95
Este artículo cubre un ejemplo que muestra cómo crear un informe usando registros de memor.....	95
Capítulo 23: Técnicas de interfaz de usuario.....	101
Examples.....	101
Creando un menú desplegable para una pantalla.....	101
Opción 1: Crear un menú desplegable en ASPX.....	101
Opción 2: Crear un menú en el gráfico.....	102
Capítulo 24: Uso del complemento de personalización para realizar cambios en varias empres104	
Introducción.....	104
Examples.....	104
Implementación de un plug-in de personalización para actualizar múltiples empresas.....	104
Capítulo 25: Visualización de un error que requiere ingresar datos de entidad.....	108
Examples.....	108
Visualización de un error que requiere que el usuario ingrese datos de la entidad.....	108
Creditos.....	110

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [acumatica](#)

It is an unofficial and free acumatica ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official acumatica.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con acumatica

Observaciones

Esta sección proporciona una descripción general de qué es acumatica y por qué un desarrollador puede querer usarla.

También debe mencionar cualquier tema grande dentro de acumatica, y vincular a los temas relacionados. Dado que la Documentación para acumatica es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Examples

Instalación o configuración

Instrucciones detalladas sobre cómo configurar o instalar acumatica.

Lea [Empezando con acumatica en línea](#):

<https://riptutorial.com/es/acumatica/topic/7152/empezando-con-acumatica>

Capítulo 2: Acumatica BQL Reference

Examples

BQL Parse y Verify

Cualquier desarrollador de aplicaciones Acumatica dedica gran parte de su tiempo a escribir código BQL. Al mismo tiempo, no todos conocen los detalles subyacentes de cómo funcionan los tipos de BQL bajo el capó.

En el corazón de BQL se encuentran dos métodos clave: `Parse()` y `Verify()`, declarados por la interfaz `IBqlCreator`. La mayoría de los tipos de BQL utilizados comúnmente, como `Where<>`, `And<>`, `Or<>` etc., se derivan de esta interfaz.

Se debe admitir que los nombres con los que estos métodos históricamente están atascados no son muy descriptivos. Posiblemente mejores nombres alternativos para ellos serían

`PrepareCommandText` y `Evaluate`.

Analizar gramaticalmente

```
public void Parse(  
    PXGraph graph,  
    List<IBqlParameter> pars,  
    List<Type> tables,  
    List<Type> fields,  
    List<IBqlSortColumn> sortColumns,  
    StringBuilder text,  
    BqlCommand.Selection selection)
```

El único propósito de `Parse()` es traducir BQL en un comando SQL para ser enviado a DBMS. Por lo tanto, este método acepta un parámetro `StringBuilder` que representa el comando SQL que se está construyendo actualmente, al que el creador BQL anexa la representación de texto SQL de sí mismo.

Por ejemplo, el método `Parse()` del predicado `And<>` agregará " AND " al texto del comando, y solicitará recursivamente la traducción de todos los creadores BQL anidados.

En particular, `And<ARRegister.docType, Equal<ARDocType.invoice>>` se traducirá en algo como "AND "ARRegister.DocType = 'AR'".

Verificar

```
public void Verify(  
    PXCache cache,  
    object item,
```

```
List<object> pars,  
ref bool? result,  
ref object value)
```

A diferencia de `Parse()`, `Verify()` funciona únicamente en el nivel de la aplicación.

Dado un registro (por ejemplo, un objeto `ARRegister`), se puede usar para calcular expresiones en él, incluyendo cálculos de fórmulas y condiciones de evaluación.

El parámetro de `result` se utiliza para almacenar el resultado de la evaluación de la condición booleana. Es utilizado principalmente por creadores de BQL *predicados*, como `Where<>`.

El parámetro de `value` se utiliza para almacenar el resultado del cálculo de la expresión. Por ejemplo, el `value` de una `Constant<string>` BQL `Constant<string>` es la representación de cadena de esa constante.

La mayoría de las veces, los creadores de BQL afectarán el resultado o el valor, pero rara vez los dos.

Un uso notable del método `Verify()` está en el método `BqlCommand.Meet()` estático, utilizado por `PXCache` para determinar si un elemento dado satisface el comando BQL:

```
public bool Meet(PXCache cache, object item, params object[] parameters)  
{  
    List<object> pars = new List<object>(parameters);  
    bool? result = null;  
    object value = null;  
    try {  
        Verify(cache, item, pars, ref result, ref value);  
    }  
    catch (SystemException ex) {  
        throw new PXException(String.Format("BQL verification failed! {0}", this.ToString()),  
ex);  
    }  
    return result == null || result == true;  
}
```

Conclusión

El verdadero poder y la belleza de los creadores de BQL radica en que la mayoría de ellos se pueden usar tanto en la base de datos como en el nivel de la aplicación, lo que permite el mecanismo de fusión de caché de Acumatica y ofrece una gran oportunidad para la reutilización del código.

Por ejemplo, cuando selecciona registros de la base de datos, la cláusula `Where<>` del comando BQL:

- Proporcionará `Parse()` para traducirlo a texto SQL durante la preparación del comando.
- Proporcionará `Verify()` durante la fusión de la memoria caché para determinar qué elementos ya residen en la memoria caché `Meet()` las condiciones de la cláusula `Where<>`

para incluir dichos elementos almacenados en caché en el conjunto de resultados.

Lea Acumatica BQL Reference en línea:

<https://riptutorial.com/es/acumatica/topic/9690/acumatica-bql-reference>

Capítulo 3: Acumatica Platform Atributos de Referencia

Examples

Atributo PXFormula

Descripción general

Una fórmula en Acumatica es un campo DAC que se calcula basándose en los valores de otros campos de objeto.

Para calcular una fórmula, el marco de Acumatica proporciona un conjunto de varias operaciones y funciones (como operaciones aritméticas, lógicas y de comparación y procesamiento de cadenas; consulte la *Lista de fórmulas comunes incorporadas*). Además de los valores de campo, una fórmula puede usar varias constantes proporcionadas por el núcleo de Acumatica y las soluciones de la aplicación. Además, una fórmula puede obtener valores para el cálculo no solo del registro actual sino también de otras fuentes (consulte *Contexto de la fórmula y sus modificadores*).

La belleza de las fórmulas es que volverán a calcular automáticamente el valor en el momento adecuado:

- En la `FieldDefaulting` campo (insertar una nueva fila; controlador de eventos `FieldDefaulting` del campo de fórmula)
- Al actualizar los campos dependientes (controlador de eventos `FieldUpdated` de cada campo dependiente)
- En la selección de la base de datos (solo para campos no `RowSelecting`; `RowSelecting` event handler)
- En la base de datos, persistir si es necesario (el desarrollador debe especificarlo explícitamente; `RowPersisted` eventos `RowPersisted`)

El recálculo de un valor de campo de fórmula en la actualización de un campo dependiente genera un evento `FieldUpdated` para el campo de fórmula. Esto le permite hacer una cadena de fórmulas dependientes (consulte *Referencias circulares directas y mediadas en fórmulas*).

Los desarrolladores de aplicaciones pueden escribir sus propias fórmulas del lado de la aplicación.

Modos de uso

Una fórmula se puede utilizar en tres modos principales:

- Simplemente calculando el valor y asignándolo al campo de fórmula (consulte *Uso básico*)
- Calcular el valor agregado a partir de los valores existentes de los campos de fórmula y asignarlo a un campo especificado en el objeto principal (ver *Uso agregado*)
- Modo mixto: calcular el valor de la fórmula, asignarlo al campo de la fórmula, calcular el valor agregado y asignarlo al campo en el objeto principal (ver *Uso combinado*)

Hay otro modo auxiliar, fórmulas no vinculadas, que es muy similar al modo mixto, pero los valores calculados de la fórmula no se asignan al campo de fórmula. El valor agregado se calcula de inmediato y se asigna al campo del objeto principal. Consulte *Uso de fórmulas sin restricciones* para obtener más información.

Propiedades de `PXFormulaAttribute` y parámetros del constructor

La funcionalidad de la fórmula es implementada por `PXFormulaAttribute`. El constructor de `PXFormulaAttribute` tiene las siguientes firmas:

```
public PXFormulaAttribute(Type formulaType)
{
    // ...
}
```

El único parámetro `formulaType` es un tipo de expresión de fórmula para calcular el valor de campo de otros campos del mismo registro de datos. Este parámetro debe cumplir una de las siguientes condiciones:

- Debe implementar la interfaz `IBqlField`
- Debe ser una constante BQL
- Debe implementar la interfaz `IBqlCreator` (ver *Lista de fórmulas comunes incorporadas*)

```
public PXFormulaAttribute(Type formulaType, Type aggregateType)
{
    // ...
}
```

El primer parámetro, `formulaType`, es el mismo que en el primer constructor. El segundo parámetro, `aggregateType`, es un tipo de fórmula de agregación para calcular el campo de registro de datos principal a partir de los campos de registro de datos secundarios. Se puede usar una función de agregación, como `SumCalc`, `CountCalc`, `MinCalc` y `MaxCalc`. Los desarrolladores de aplicaciones pueden crear sus propias fórmulas de agregación.

Un tipo de fórmula agregada debe ser un tipo genérico y debe implementar la interfaz `IBqlAggregateCalculator`. El primer parámetro genérico del tipo de fórmula agregada debe implementar la interfaz `IBqlField` y debe tener el tipo de campo del objeto principal.

```
public virtual bool Persistent { get; set; }
```

La propiedad `PXFormulaAttribute.Persistent` indica si el atributo vuelve a calcular la fórmula después de guardar los cambios en la base de datos. Es posible que necesite un nuevo cálculo si los campos de los que depende la fórmula se actualizan en el evento `RowPersisting`. Por defecto, la propiedad es igual a `false`.

Uso

En la mayoría de los casos, las fórmulas se utilizan para el cálculo directo del valor del campo de fórmula de otros campos del mismo registro de datos.

El ejemplo más simple de uso de fórmula:

```
[PXDBDate]
[PXFormula(typeof(FADetails.receiptDate))]
[PXDefault]
[PXUIField(DisplayName = Messages.PlacedInServiceDate)]
public virtual DateTime? DepreciateFromDate { get; set; }
```

En este ejemplo, el valor del campo `ReceiptDate` se asigna al campo `DepreciateFromDate` en la inserción de un nuevo registro y en la actualización del campo `ReceiptDate`.

Un ejemplo un poco más complejo:

```
[PXCurrency(typeof(APPayment.curyInfoID), typeof(APPayment.unappliedBal))]
[PXUIField(DisplayName = "Unapplied Balance", Visibility = PXUIVisibility.Visible, Enabled = false)]
[PXFormula(typeof(Sub<APPayment.curyDocBal, APPayment.curyApplAmt>))]
public virtual Decimal? CuryUnappliedBal { get; set; }
```

Aquí, el saldo no aplicado del documento se calcula como la diferencia entre el saldo del documento y la cantidad aplicada.

Ejemplo de opción múltiple con un valor predeterminado:

```
[PXUIField(DisplayName = "Class Icon", IsReadOnly = true)]
[PXImage]
[PXFormula(typeof(Switch<
    Case<Where<EPAActivity.classID, Equal<CRAActivityClass.task>>, EPAActivity.classIcon.task,
    Case<Where<EPAActivity.classID, Equal<CRAActivityClass.events>>,
EPAActivity.classIcon.events,
    Case<Where<EPAActivity.classID, Equal<CRAActivityClass.email>,
    And<EPAActivity.isIncome, NotEqual<True>>>, EPAActivity.classIcon.email,
    Case<Where<EPAActivity.classID, Equal<CRAActivityClass.email>,
    And<EPAActivity.isIncome, Equal<True>>>, EPAActivity.classIcon.emailResponse,
    Case<Where<EPAActivity.classID, Equal<CRAActivityClass.history>>,
EPAActivity.classIcon.history>>>>,
    Selector<Current2<EPAActivity.type>, EPAActivityType.imageUrl>>))]
public virtual string ClassIcon { get; set; }
```

Orden de los campos

El orden de los campos en el DAC es importante para corregir el cálculo de la fórmula. Todos los campos de origen (a partir de los cuales se calcula la fórmula), incluidas otras fórmulas, deben definirse en el DAC antes del campo de fórmula. De lo contrario, el campo se puede calcular incorrectamente o puede causar un error de tiempo de ejecución.

Contexto de la fórmula y sus modificadores

De forma predeterminada, el contexto del cálculo de fórmula está restringido por el objeto actual (registro) de la clase que contiene la declaración de fórmula. También se permite usar constantes (descendientes de la clase `Constant<>`).

Una fórmula que usa solo los campos de su objeto:

```
public partial class Contract : IBqlTable, IAttributeSupport
{
    //...
    [PXDecimal(4)]
    [PXDefault(TypeCode.Decimal, "0.0", PersistingCheck = PXPersistingCheck.Nothing)]
    [PXFormula(typeof(Add<Contract.pendingRecurring, Add<Contract.pendingRenewal,
Contract.pendingSetup>>))]
    [PXUIField(DisplayName = "Total Pending", Enabled=false)]
    public virtual decimal? TotalPending { get; set; }
    //...
}
```

Sin embargo, es posible obtener valores de entrada para el cálculo de la fórmula de otras fuentes:

- Un registro actual de cualquier caché en el BLC (si está asignado).
- Un registro extranjero especificado por `PXSelectorAttribute`.
- Un registro padre especificado por `PXParentAttribute`.

La fórmula soporta los siguientes modificadores de contexto.

`Current<TRecord.field>` **y** `Current2<TRecord.field>`

Obtiene el valor del campo del registro almacenado en la propiedad `Current` de la caché de `TRecord`.

Si la propiedad `Current` del caché **o el campo en sí** contiene nulo:

- `Actual <>` fuerza el valor predeterminado del campo y devuelve el valor predeterminado del campo.
- `Current2 <>` devuelve null.

Ejemplo:

```
[PXFormula(typeof(Switch<
    Case<Where<
        ARAdjust.adjgDocType, Equal<Current<ARPayment.docType>>,
        And<ARAdjust.adjgRefNbr, Equal<Current<ARPayment.refNbr>>>>,
```



```

        ARAdjust.classIcon.outgoing>,
        ARAdjust.classIcon.incoming>))]
protected virtual void ARAdjust_ClassIcon_CacheAttached(PXCache sender)

```

Parent<TParent.field>

Obtiene el valor del campo del registro de datos principal, según lo definido por PXParentAttribute que reside en el DAC actual.

```

public class INTran : IBqlTable
{
    [PXParent(typeof(Select<
        INRegister,
        Where<
            INRegister.docType, Equal<Current<INTran.docType>>,
            And<INRegister.refNbr, Equal<Current<INTran.refNbr>>>>))]
    public virtual String RefNbr { ... }

    [PXFormula(typeof(Parent<INRegister.origModule>))]
    public virtual String OrigModule { ... }
}

```

IsTableEmpty<TRecord>

Devuelve `true` si la tabla de DB correspondiente al DAC especificado no contiene registros, de lo contrario es `false`.

```

public class APRegister : IBqlTable
{
    [PXFormula(typeof(Switch<
        Case<Where<
            IsTableEmpty<APSetupApproval>, Equal<True>>,
            True,
        Case<Where<
            APRegister.requestApproval, Equal<True>>,
            False>>,
        True>))]
    public virtual bool? DontApprove { get; set; }
}

```

Selector<KeyField, ForeignOperand>

Obtiene un PXSelectorAttribute definido en el campo de clave foránea (KeyField) del DAC actual.

Obtiene el registro de datos externos

actualmente referenciados por el selector.

Calcula y devuelve una expresión en ese registro de datos como lo define ForeignOperand.

```
public class APVendorPrice : IBqlTable
{
    // Note: inventory attribute is an
    // aggregate containing a PXSelectorAttribute
    // inside, which is also valid for Selector<>.
    // -
    [Inventory(DisplayName = "Inventory ID")]
    public virtual int? InventoryID

    [PXFormula(typeof(Selector<
        APVendorPrice.inventoryID,
        InventoryItem.purchaseUnit>))]
    public virtual string UOM { get; set; }
}
```

Uso de fórmulas en los campos sin límites

Si el campo de fórmula es un campo no `PXFieldAttribute` marcado con uno de los descendientes de `PXFieldAttribute` (como `PXIntAttribute` o `PXStringAttribute`), entonces su cálculo se activa adicionalmente durante el evento `RowSelecting`.

Lista de fórmulas comunes incorporadas

TBD

Referencias circulares directas y mediadas en fórmulas

TBD

Flujo de control en fórmulas condicionales

TBD

Uso de múltiples fórmulas en un campo

TBD

Atributo de PXRestrictor

Introducción

El atributo PXSelectorAttribute (también conocido como el selector), aunque es vital y se usa con frecuencia, tiene, sin embargo, dos inconvenientes principales:

- Da un mensaje no informativo "<object_name> cannot be found in the system" si no se encuentran elementos que satisfagan la condición del selector.
- El mensaje genera el mismo mensaje de error si actualiza *otros* campos del registro, pero el objeto al que hace referencia el selector ya ha cambiado y ya no cumple con su condición. Este comportamiento es claramente erróneo porque la ley no debe ser retroactiva.

El PXRestrictorAttribute (también conocido como el restrictor) se puede utilizar para resolver estos problemas.

Detalles

PXRestrictorAttribute no funciona solo; siempre debe estar emparejado con un PXSelectorAttribute . Usar el restrictor sin el selector no tendrá efecto.

El restrictor encuentra el selector en el mismo campo, inyectando en él una condición adicional y el mensaje de error correspondiente. La condición del restrictor se agrega a la condición del selector a través de un AND booleano, y se genera un mensaje de error apropiado si el objeto al que se hace referencia viola la restricción del restrictor. Además, si el objeto al que se hace referencia ha cambiado y ya no cumple con la condición de restricción, no se producen mensajes de error cuando cambia **cualquier otro** campo del objeto de referencia.

Uso general:

```
[PXDBInt]
[PXSelector(typeof(Search<FAClass.assetID, Where<FAClass.recordType,
Equal<FARecordType.classType>>>),
    typeof(FAClass.assetCD), typeof(FAClass.assetTypeID), typeof(FAClass.description),
    typeof(FAClass.usefulLife),
    SubstituteKey = typeof(FAClass.assetCD),
    DescriptionField = typeof(FAClass.description), CacheGlobal = true)]
[PXRestrictor(typeof(Where<FAClass.active, Equal<True>>), Messages.InactiveFAClass,
typeof(FAClass.assetCD))]
[PXUIField(DisplayName = "Asset Class", Visibility = PXUIVisibility.Visible)]
public virtual int? ClassID { get; set; }
```

Se pueden usar múltiples restrictores con un atributo selector. En este caso, todas las condiciones de restricción adicionales se aplican en un orden no determinado. Una vez que se viola cualquier condición, se genera el mensaje de error apropiado.

La condición `Where<>` del propio selector se aplica **después de** todas las condiciones del restrictor.

```
[PXDefault]
// An aggregate attribute containing the selector inside.
// -
[ContractTemplate(Required = true)]
[PXRestrictor(typeof(Where<ContractTemplate.status, Equal<Contract.status.active>>),
Messages.TemplateIsNotActivated, typeof(ContractTemplate.contractCD))]
[PXRestrictor(typeof(Where<ContractTemplate.effectiveFrom,
LessEqual<Current<AccessInfo.businessDate>>,
Or<ContractTemplate.effectiveFrom, IsNull>>), Messages.TemplateIsNotStarted)]
[PXRestrictor(typeof(Where<ContractTemplate.discontinueAfter,
GreaterEqual<Current<AccessInfo.businessDate>>,
Or<ContractTemplate.discontinueAfter, IsNull>>), Messages.TemplateIsExpired)]
public virtual int? TemplateID { get; set; }
```

Opciones

El constructor de `PXRestrictorAttribute` toma tres parámetros:

1. Condición adicional del restrictor. Este tipo de BQL debe implementar la interfaz `IBqlWhere`.
2. El mensaje de error apropiado. El mensaje puede contener elementos de formato (llaves) para mostrar el contexto. El mensaje debe ser una constante de cadena definida en una clase estática localizable (como `PX.Objects.GL.Messages`).
3. Una matriz de tipos de campo. Estos campos deben pertenecer al objeto actual y deben implementar la interfaz `IBqlField`. Los valores de los campos se utilizarán para el formato de los mensajes de error.

Además, hay varias opciones que especifican el comportamiento del restrictor.

Anulación de los limitadores heredados

La propiedad `ReplaceInherited` indica si el restrictor actual debe anular los restrictores heredados. Si esta propiedad se establece en verdadero, se reemplazarán todos los restrictores heredados (ubicados en cualquier atributo agregado o atributo base).

Reemplazo de restrictores heredados:

```
[CustomerActive(Visibility = PXUIVisibility.SelectorVisible, Filterable = true, TabOrder =
2)]
[PXRestrictor(typeof(Where<Customer.status, Equal<CR.BAccount.status.active>,
Or<Customer.status, Equal<CR.BAccount.status.oneTime>,
Or<Customer.status, Equal<CR.BAccount.status.hold>,
Or<Customer.status, Equal<CR.BAccount.status.creditHold>>>>>),
Messages.CustomerIsInStatus, typeof(Customer.status),
ReplaceInherited = true)] // Replaced all restrictors from CustomerActiveAttribute
```

```
[PXUIField(DisplayName = "Customer")]
[PXDefault()]
public override int? CustomerID { get; set; }
```

Tenga en cuenta que no recomendamos que utilice la propiedad `ReplaceInherited` en el código de la aplicación cuando existan alternativas razonables. Esta propiedad está destinada principalmente para ser utilizada en personalizaciones.

Caching global

`CacheGlobal` admite la funcionalidad de diccionario global de la misma manera que en `PXSelectorAttribute`.

Recomendaciones para el uso

Utilizar solo condiciones de restricción

Cuando los restrictores y un selector se usan juntos, este último no debe contener la cláusula `IBqlWhere`. Idealmente, todas las condiciones deberían ser movidas a restrictores. Este enfoque proporciona mensajes de error más fáciles de usar y elimina errores retroactivos innecesarios.

Un ejemplo ideal:

```
[PXDBString(5, IsFixed = true, IsUnicode = false)]
[PXUIField(DisplayName = "Type", Required = true)]
[PXSelector(typeof(EPActivityType.type), DescriptionField =
typeof(EPActivityType.description))]
[PXRestrictor(typeof(Where<EPActivityType.active, Equal<True>>),
Messages.InactiveActivityType, typeof(EPActivityType.type))]
[PXRestrictor(typeof(Where<EPActivityType.isInternal, Equal<True>>),
Messages.ExternalActivityType, typeof(EPActivityType.type))]
public virtual string Type { get; set; }
```

Posibles errores retroactivos:

```
[PXDBInt]
[PXUIField(DisplayName = "Contract")]
[PXSelector(typeof(Search2<Contract.contractID,
LeftJoin<ContractBillingSchedule, On<Contract.contractID,
Equal<ContractBillingSchedule.contractID>>>,
Where<Contract.isTemplate, NotEqual<True>,
And<Contract.baseType, Equal<Contract.ContractBaseType>,
And<Where<Current<CRCase.customerID>, IsNull,
Or2<Where<Contract.customerID, Equal<Current<CRCase.customerID>>,
And<Current<CRCase.locationID>, IsNull>>,
Or2<Where<ContractBillingSchedule.accountID, Equal<Current<CRCase.customerID>>,
And<Current<CRCase.locationID>, IsNull>>,
Or2<Where<Contract.customerID, Equal<Current<CRCase.customerID>>,
And<Contract.locationID, Equal<Current<CRCase.locationID>>>>>,
Or<Where<ContractBillingSchedule.accountID, Equal<Current<CRCase.customerID>>,
And<ContractBillingSchedule.locationID,
```

Capítulo 4: Adición de soporte de atributos a una entidad de pedido de ventas lista para usar

Introducción

Acumatica ERP le permite definir atributos para una clasificación flexible y significativa de una Entidad (clientes potenciales, acciones / no-acciones, etc.) según sea necesario para las necesidades específicas de su empresa. Un atributo es una propiedad que le permite especificar información adicional para los objetos en el sistema. Los atributos se definen en el contexto de una clase que es una agrupación de las cuentas comerciales (incluidos clientes potenciales, oportunidades, clientes y casos), acciones y artículos no en existencia por una o más de sus propiedades.

Observaciones

Este ejemplo es aplicable a la serie Acumatica 6.0.

Examples

Este artículo proporciona una guía práctica para agregar el soporte Acumatica ERP Attribute a una entidad de pedido de ventas lista para usar

En el núcleo, el DAC principal de su entidad debe tener una columna GUID (`NoteID`) para hacer referencia a la tabla de `CSAnswers` y debe tener un campo que identifique la clase de la entidad.

Haremos uso del `Order Type` de `Order Type` para definir la lista de atributos para recopilar información específica del tipo de orden.

Cree una Extensión de gráfico para `SOOrderTypeMaint` Gráfico y declare la vista de datos para definir la lista de atributos para un tipo de orden particular. Usaremos

`CSAttributeGroupList<TEntityClass, TEntity>` fuera de la caja `CSAttributeGroupList<TEntityClass, TEntity>`

```
public class SOOrderTypeMaintPXExt : PXGraphExtension<SOOrderTypeMaint>
{
    [PXViewName(PX.Objects.CR.Messages.Attributes)]
    public CSAttributeGroupList<SOOrderType, SOOrder> Mapping;
}
```

Cree una extensión de gráfico para `SOOrderEntry` Graph y declare la vista de datos para los atributos específicos del tipo de orden actual.

```

public class SOOrderEntryPXExt : PXGraphExtension<SOOrderEntry>
{
    public CRAttributeList<SOOrder> Answers;
}

```

Cree la extensión DAC para `SOOrder` DAC y declare el campo definido por el usuario decorado con el atributo `CRAttributesField` y especifique el campo `ClassID` ; en nuestro caso, es `OrderType` .

```

public class SOOrderPXExt : PXCacheExtension<SOOrder>
{
    #region UsrAttributes

    public abstract class usrAttributes : IBqlField { }

    [CRAttributesField(typeof(SOOrder.orderType))]
    public virtual string[] UsrAttributes { get; set; }

    #endregion
}

```

Modifique la página de `Order Types (SO201000)` como se muestra a continuación utilizando el motor de personalización

```

<px:PXTabItem Text="Attributes">
    <Template>
        <px:PXGrid runat="server" BorderWidth="0px" Height="150px" SkinID="Details" Width="100%"
ID="AttributesGrid"
            MatrixMode="True" DataSourceID="ds">
            <AutoSize Enabled="True" Container="Window" MinHeight="150" />
            <Levels>
                <px:PXGridLevel DataMember="Mapping">
                    <RowTemplate>
                        <px:PXSelector runat="server" DataField="AttributeID"
FilterByAllFields="True" AllowEdit="True"
                            CommitChanges="True" ID="edAttributeID" /></RowTemplate>
                        <Columns>
                            <px:PXGridColumn DataField="AttributeID" Width="81px" AutoCallBack="True"
LinkCommand="ShowDetails" />
                            <px:PXGridColumn DataField="Description" Width="351px" AllowNull="False"
/>
                            <px:PXGridColumn DataField="SortOrder" TextAlign="Right" Width="81px" />
                            <px:PXGridColumn DataField="Required" Type="CheckBox" TextAlign="Center"
AllowNull="False" />
                            <px:PXGridColumn DataField="CSAttribute__IsInternal" Type="CheckBox"
TextAlign="Center" AllowNull="True" />
                            <px:PXGridColumn DataField="ControlType" Type="DropDownList" Width="81px"
AllowNull="False" />
                            <px:PXGridColumn DataField="DefaultValue" RenderEditorText="False"
Width="100px" AllowNull="True" />
                        </Columns>
                    </px:PXGridLevel>
                </Levels>
            </px:PXGrid>
        </Template>
    </px:PXTabItem>

```

Modifique la página de `Sales Orders (SO301000)` como se muestra a continuación utilizando el

motor de personalización

```
<px:PXTabItem Text="Attributes">
  <Template>
    <px:PXGrid runat="server" ID="PXGridAnswers" Height="200px" SkinID="Inquire"
      Width="100%" MatrixMode="True" DataSourceID="ds">
      <AutoSize Enabled="True" MinHeight="200" />
      <ActionBar>
        <Actions>
          <Search Enabled="False" />
        </Actions>
      </ActionBar>
      <Mode AllowAddNew="False" AllowDelete="False" AllowColMoving="False" />
      <Levels>
        <px:PXGridLevel DataMember="Answers">
          <Columns>
            <px:PXGridColumn TextAlign="Left" DataField="AttributeID"
              TextField="AttributeID_description"
              Width="250px" AllowShowHide="False" />
            <px:PXGridColumn Type="CheckBox" TextAlign="Center" DataField="isRequired"
              Width="80px" />
            <px:PXGridColumn DataField="Value" Width="300px" AllowSort="False"
              AllowShowHide="False" />
          </Columns>
        </px:PXGridLevel>
      </Levels>
    </px:PXGrid>
  </Template>
</px:PXTabItem>
```

Descargar paquete de despliegue

Lea Adición de soporte de atributos a una entidad de pedido de ventas lista para usar en línea:
<https://riptutorial.com/es/acumatica/topic/8666/adicion-de-soporte-de-atributos-a-una-entidad-de-pedido-de-ventas-lista-para-usar>

Capítulo 5: Ampliación de la lista de entidades apoyadas por tareas, eventos y actividades

Introducción

En este tema, aprenderá cómo ampliar el campo Descripción de entidad relacionada con una entidad personalizada para tareas, eventos y actividades.

Examples

Agregar órdenes de trabajo de prueba al campo de descripción de entidad relacionada

Supongamos que ya ha creado la pantalla de **órdenes de trabajo de prueba** personalizadas para administrar las órdenes de trabajo de prueba en su aplicación Acumatica ERP:

The screenshot shows the Acumatica ERP interface for a 'Test Work Order'. The form includes fields for ITWO Nbr. (000003), Purchase Order (000102), Order Date (2/6/2017), Status (Closed), Created By (admin), and Last Modified On (3/3/2017 12:30:48 PM). Below the form is a table with the following data:

Status	Item ID	Description	Manufacturer	Received Date	Received Qty
Closed	AALEGO500	Lego 500 piece set	DSFD-2324	1/18/2017	5.00
Cancelled	CONCRIB02	Graco Stylus Classic Travel S...	3243-FDEW56	1/16/2017	2.00
Closed Short	ELEBOSE1	Bose Quiet Comfort Noise Ca...	BDS-432456-...	1/20/2017	20.00

Ya `NoteID` campo `NoteID` declarado en el DAC de `TestWorkOrder` , administrado en la pantalla de **órdenes de trabajo de prueba** :

```
[Serializable]
public class TestWorkOrder : IBqlTable
{
    ...

    #region NoteID
    public abstract class noteID : IBqlField { }
```

```
[PXNote]
public virtual Guid? NoteID { get; set; }
#endregion

...
}
```

y la propiedad `ActivityIndicator` se establece en **True** para el contenedor `PXForm` nivel `PXForm` :

```
<px:PXFormView ID="form" runat="server" ActivityIndicator="true" DataSourceID="ds" Style="z-
index: 100" DataMember="ITWO" Width="100%" >
```

Sin embargo, cuando se crea una nueva tarea, evento o actividad para una orden de trabajo de prueba, el control **Descripción de la entidad relacionada** está siempre vacío:

The image shows a multi-step process in a software application:

- Step 1:** In the main application window, the 'ACTIVITIES' tab is selected.
- Step 2:** The 'Tasks & Activities' dialog box is open, and the 'ADD TASK' button is highlighted.
- Step 3:** In the 'Task - Mozilla Firefox' window, the 'Related Entity' field is being edited, and the 'Select Entity' dialog box is open, showing the entity '000003' selected.

Para agregar la entidad **Orden de trabajo de prueba** al selector **Descripción de entidad relacionada** , debe completar los siguientes pasos:

1. Para el `PXNoteAttribute` en el campo `TestWorkOrder.NoteID`, establecer `ShowInReferenceSelector` propiedad en **True** y definir la expresión BQL para seleccionar registros de datos que se muestran en las operaciones de búsqueda **Entidad**:

```
[PXNote (
```

```

        ShowInReferenceSelector = true,
        Selector = typeof(Search<TestWorkOrder.orderNbr>))]
public virtual Guid? NoteID { get; set; }

```

2. Decore el DAC de `TestWorkOrder` con el `PXCacheNameAttribute` y el `PXPrimaryGraphAttribute` :

```

[PXLocalizable]
public static class Messages
{
    public const string Opportunity = "Test Work Order";
}

[Serializable]
[PXCacheName(Messages.Opportunity)]
[PXPrimaryGraph(typeof(TestWorkOrderEntry))]
public class TestWorkOrder : IBqlTable
{
    ...
}

```

El atributo `PXCacheName` define el nombre fácil de usar para el `TestWorkOrder` DAC (**Orden de trabajo de prueba** en este caso), que estará disponible en el menú desplegable **Tipo** . El atributo `PXPrimaryGraph` determina la página de entrada donde se redirige al usuario para editar una orden de trabajo de prueba, que es la pantalla de **Órdenes de trabajo de prueba** en el ejemplo dado.

3. Decore algunos campos de `TestWorkOrder` con el `PXFieldDescriptionAttribute` . Esos valores de campo se concatenarán en una sola etiqueta de texto, que representa la orden de trabajo de prueba referenciada dentro del campo **Descripción de la entidad relacionada** :

```

...
[PXFieldDescription]
public virtual string OrderNbr { get; set; }

...
[PXFieldDescription]
public virtual String Status { get; set; }

...
[PXFieldDescription]
public virtual string POOrderNbr { get; set; }

```

4. Definir la lista de columnas que se muestran en las operaciones de búsqueda **Entidad** por la elección de uno de los enfoques siguientes:

a. Use la propiedad `PXNoteAttribute.FieldList` (obtiene la prioridad más alta):

```

public abstract class noteID : IBqlField { }
[PXNote(
    ShowInReferenceSelector = true,
    Selector = typeof(Search<TestWorkOrder.orderNbr>),
    FieldList = new Type[]
    {
        typeof(TestWorkOrder.orderNbr),

```

```

        typeof(TestWorkOrder.orderDate),
        typeof(TestWorkOrder.status),
        typeof(TestWorkOrder.poOrderNbr)
    })]
    public virtual Guid? NoteID { get; set; }

```

segundo. Pida **prestada** la lista de columnas definidas para la búsqueda **OrderNbr** :

```

public abstract class orderNbr : IBqlField { }
[PXDBString(15, IsKey = true, IsUnicode = true, InputMask = ">CCCCCCCCCCCCCCC")]
[PXDefault()]
[PXUIField(DisplayName = "ITWO Nbr.", Visibility = PXUIVisibility.SelectorVisible)]
[PXSelector(typeof(Search<TestWorkOrder.orderNbr>),
    typeof(TestWorkOrder.orderNbr),
    typeof(TestWorkOrder.orderDate),
    typeof(TestWorkOrder.status),
    typeof(TestWorkOrder.poOrderNbr))]
[PXFieldDescription]
public virtual string OrderNbr { get; set; }

```

do. Mostrar todos los campos de `TestWorkOrder` con la **visibilidad** establecida en `PXUIVisibility.SelectorVisible` :

```

...
[PXUIField(DisplayName = "ITWO Nbr.", Visibility = PXUIVisibility.SelectorVisible)]
public virtual string OrderNbr { get; set; }

...
[PXUIField(DisplayName = "Order Date", Visibility = PXUIVisibility.SelectorVisible)]
public virtual DateTime? OrderDate { get; set; }

...
[PXUIField(DisplayName = "Status", Visibility = PXUIVisibility.SelectorVisible)]
public virtual String Status { get; set; }

...
[PXUIField(DisplayName = "Purchase Order", Visibility = PXUIVisibility.SelectorVisible)]
public virtual string POOrderNbr { get; set; }

```

Después de completar los 4 pasos anteriores, las **Órdenes de Trabajo de Prueba** deben ser totalmente compatibles con el campo **Descripción de la Entidad Relacionada** en Tareas, Eventos y Actividades

Revision Two HQ Test Work Order

ITWO Nbr.: 000003 Status: Closed
 Purchase Order: 000102 Created By: admin
 Order Date: 2/6/2017 Last Modified On: 3/3/2017 12:43:43 PM

Tasks & Activities

ADD TASK ADD EVENT ADD ACTIVITY

Status	Item ID	Description
Closed	AALEGO500	Lego 500 piece

Task - Mozilla Firefox

localhost/StackOverflow/(W(10000))/pages/cr/cr306020.aspx?timeStamp=5bb8b94af373d81d31a9ac8d175642213

Revision Two HQ Task

SAVE & CLOSE COMPLETE COMPLETE & FOLLOW-UP CANCEL

Details Related Activities Related Tasks

* Summary:

Start Date: 3/3/2017 Internal: Priority: Normal

Due Date:

Completion (%): 0

Workgroup:

Owner: EP00000002 - Baker Maxwell Reminder

Remind at (Dat...)

Related Entity ... 000003, Closed, 000102

* Project: X - Non-Project Code.

Project Task:

VISUAL Paragraph B I U

Select Entity

Type: * Test Work Order
 Entity: * 000003

Select - Entity

ITWO Nbr.	Order Date	Status	Purchase Or
000003	2/6/2017	Closed	000102
000004	2/6/2017	Locked	000146
000005	2/15/2017	In Progress	000111
000006	2/21/2017	Open	000154

Lea Ampliación de la lista de entidades apoyadas por tareas, eventos y actividades en línea:
<https://riptutorial.com/es/acumatica/topic/9342/ampliacion-de-la-lista-de-entidades-apoyadas-por-tareas--eventos-y-actividades>

Capítulo 6: Cálculo de fletes

Introducción

Acumatica ERP le permite administrar el flete para controlar mejor los costos e ingresos adicionales en las transacciones de ventas. La cantidad de flete que cobra a sus clientes puede incluir no solo el flete que su compañía cobra a los transportistas, sino también las tarifas de seguro, manejo y embalaje definidas por sus términos de envío y flete premium.

Examples

Monto de flete anulado en el envío y la factura

Fuera de la caja, Acumatica permite crear y mantener la lista de términos de envío en el sistema. Los términos de envío se utilizan para definir los costos de envío, embalaje y manejo, según el monto del envío.

En este ejemplo, mostraré cómo calcular el monto de flete para un envío según el monto de la orden de venta, lo que permitiría a los usuarios crear múltiples envíos por orden de venta con los mismos términos de envío que se aplican automáticamente a todos los envíos.

FreightCalculator

La clase `FreightCalculator` es responsable del cálculo del costo de flete y los términos de flete. A los efectos de este ejemplo, nuestro interés se centrará únicamente en el método `GetFreightTerms` :

```
public class FreightCalculator
{
    ...

    protected virtual ShipTermsDetail GetFreightTerms(string shipTermsID, decimal? lineTotal)
    {
        return PXSelect<ShipTermsDetail,
            Where<ShipTermsDetail.shipTermsID, Equal<Required<SOOrder.shipTermsID>>>,
            And<ShipTermsDetail.breakAmount, LessEqual<Required<SOOrder.lineTotal>>>>,
            OrderBy<Desc<ShipTermsDetail.breakAmount>>>>.Select(graph, shipTermsID, lineTotal);
    }

    ...
}
```

Tanto las pantallas de **pedidos de venta** como las de **envío** utilizan la clase `FreightCalculator` para calcular el monto de flete según el monto del pedido de venta y del envío, respectivamente:

Ordenes de venta


```

public class SOOrderEntry : PXGraph<SOOrderEntry, SOOrder>, PXImportAttribute.IPXPrepareItems
{
    ...

    public virtual FreightCalculator CreateFreightCalculator()
    {
        return new FreightCalculator(this);
    }

    ...

    protected virtual void SOOrder_RowUpdated(PXCache sender, PXRowUpdatedEventArgs e)
    {
        ...

        PXResultSet<SOLine> res = Transactions.Select();
        FreightCalculator fc = CreateFreightCalculator();
        fc.CalcFreight<SOOrder, SOOrder.curyFreightCost, SOOrder.curyFreightAmt>(sender,
(SOOrder)e.Row, res.Count);

        ...
    }

    ...
}

```

Envíos

```

public class SOShipmentEntry : PXGraph<SOShipmentEntry, SOShipment>
{
    ...

    protected virtual FreightCalculator CreateFreightCalculator()
    {
        return new FreightCalculator(this);
    }

    ...

    protected virtual void SOShipment_RowUpdated(PXCache sender, PXRowUpdatedEventArgs e)
    {
        ...

        PXResultSet<SOShipLine> res = Transactions.Select();
        ...
        FreightCalculator fc = CreateFreightCalculator();
        fc.CalcFreight<SOShipment, SOShipment.curyFreightCost,
SOShipment.curyFreightAmt>(sender, (SOShipment)e.Row, res.Count);

        ...
    }

    ...
}

```

Importe de flete anulado

Para personalizar cómo Acumatica calcula el monto del flete en la pantalla **Envíos** , declararé la clase `FreightCalculatorCst` heredada de `FreightCalculator` y `GetFreightTerms` método `GetFreightTerms` :

```
public class FreightCalculatorCst : FreightCalculator
{
    public FreightCalculatorCst(PXGraph graph)
        : base(graph)
    {
    }

    protected override ShipTermsDetail GetFreightTerms(string shipTermsID, decimal? lineTotal)
    {
        if (graph is SOShipmentEntry)
        {
            var shipmentEntry = graph as SOShipmentEntry;
            int orderCount = 0;
            decimal? lineTotalTemp = null;

            foreach (PXResult<SOOrderShipment, SOOrder, CurrencyInfo, SOAddress, SOContact,
SOOrderType> orderRec in
                shipmentEntry.OrderList.SelectWindowed(0, 2))
            {
                orderCount++;
                SOOrder order = (SOOrder)orderRec;
                if (orderCount == 1)
                    lineTotalTemp = order.LineTotal;
                else
                    break;
            }

            if (orderCount == 1)
            {
                lineTotal = lineTotalTemp;
            }
        }

        return base.GetFreightTerms(shipTermsID, lineTotal);
    }
}
```

Después de eso, implementaré una extensión para el BLC de `SOShipmentEntry` y

`CreateFreightCalculator` método `CreateFreightCalculator` para reemplazar `FreightCalculator` con mi clase personalizada `FreightCalculatorCst` en la pantalla de **Envíos** :

```
public class SOShipmentEntryExt : PXGraphExtension<SOShipmentEntry>
{
    [PXOverride]
    public FreightCalculator CreateFreightCalculator()
    {
        return new FreightCalculatorCst(Base);
    }
}
```

Comprensión de la implementación de la

clase FreightCalculatorCst en el ejemplo anterior

En el método `GetFreightTerms` anulado, `GetFreightTerms` monto del pedido de venta en lugar del monto del envío para invocar el método `GetFreightTerms` básico y recibiré los términos del envío:

```
foreach (PXResult<SOOrderShipment, SOOrder, CurrencyInfo, SOAddress, SOContact, SOOrderType>
orderRec in
    shipmentEntry.OrderList.SelectWindowed(0, 2))
{
    orderCount++;
    SOOrder order = (SOOrder)orderRec;
    if (orderCount == 1)
        lineTotalTemp = order.LineTotal;
    else
        break;
}

if (orderCount == 1)
{
    lineTotal = lineTotalTemp;
}
```

Obviamente, solo es posible utilizar el importe del pedido de ventas para calcular el importe del flete para los envíos, que solo cumplen 1 pedido. Si un envío cumple con varios pedidos, tendríamos que seguir el comportamiento del producto base y calcular el monto de flete según el monto del envío. Para verificar el número de pedidos que cumple el envío, `SelectWindowed` método `SelectWindowed` en la vista de datos de `OrderList` y `OrderList` los primeros 2 pedidos realizados por el envío actual. Podría haber solicitado todos los pedidos cumplidos por el envío, pero esto llevaría mucho más tiempo para ejecutar y devolver muchos registros de los necesarios para verificar si se puede usar el monto del pedido de venta en lugar del monto del envío para calcular el flete.

Lea Cálculo de fletes en línea: <https://riptutorial.com/es/acumatica/topic/9044/calculo-de-fletes>

Capítulo 7: Cambio de subtítulos dinámicamente usando campos DAC de solo lectura.

Introducción

Este ejemplo muestra cómo cambiar dinámicamente el campo Título / Etiqueta del nombre del cliente en Customer ScreenID AR303000 en Acumatica ERP, dependiendo de la ID del cliente actual seleccionada en el mismo formulario. Podríamos:

Examples

Cómo

Agregar nuevo campo sin unir al DAC. (como solo lectura)

```
[PXString(60, IsUnicode = true)]
[PXUIField(Enabled = false, IsReadOnly = true)]
public virtual string UsrReadOnlyAcctName{get;set;}
public abstract class usrReadOnlyAcctName : IBqlField{}
```

Modificar su valor en función de las condiciones que utilicen los manejadores. (En el ID del ciclo del cliente seleccionado)

```
public class CustomerMaint_Extension:PXGraphExtension<CustomerMaint>
{
    protected void Customer_RowSelected(PXCache sender, PXRowSelectedEventArgs e)
    {
        var customer = (BAccount)e.Row;
        var customerExt = customer.GetExtension<BAccountExt>();
        if (customerExt != null)
        {
            customerExt.UsrReadOnlyAcctName = customer.AcctName;
        }
    }
}
```

SuppressLabel (verdadero) para los nuevos campos no vinculados y los campos existentes cuya etiqueta se reemplazará.

Layout Editor: AR303000 (Customers)

PREVIEW CHANGES ACTIONS ▾

Override	Property
<input type="checkbox"/>	Base Properties
<input type="checkbox"/>	CommitChanges
<input checked="" type="checkbox"/>	DataField
<input checked="" type="checkbox"/>	ID
<input type="checkbox"/>	Size
<input type="checkbox"/>	SkinID
<input type="checkbox"/>	Ext Properties
<input type="checkbox"/>	AutoCallback
<input type="checkbox"/>	AutoSize
<input type="checkbox"/>	DisableSpellcheck
<input checked="" type="checkbox"/>	Enabled
<input type="checkbox"/>	Height
<input type="checkbox"/>	LabelWidth
<input type="checkbox"/>	LinkCommand
<input checked="" type="checkbox"/>	SuppressLabel
<input type="checkbox"/>	SyncStateWithCommand
<input type="checkbox"/>	TextAlign
<input type="checkbox"/>	TextMode

Coloque el campo sin unir añadido antes del campo existente.

Resultados:

Customer ID:

Status:

Active Staffing Service *

Lea Cambio de subtítulos dinámicamente usando campos DAC de solo lectura. en línea:
<https://riptutorial.com/es/acumatica/topic/8858/cambio-de-subtitulos-dinamicamente-usando-campos-dac-de-solo-lectura->

Capítulo 8: Cambio de tamaño de la ventana desplegable del selector

Introducción

En este tema, aprenderá cómo cambiar el tamaño de la ventana desplegable del selector. Cada control de selección en Acumatica tiene un botón indicado con un icono de lupa. Al hacer clic en este botón, los usuarios pueden abrir una ventana desplegable que muestra una lista de objetos disponibles para su selección.

Examples

Cambio de rangos de tamaño por defecto para la ventana desplegable del selector

Las siguientes 4 propiedades están disponibles para los controles de entrada **PXSelector** y **PXSegmentMask** para definir el rango de tamaño para una ventana desplegable:

- **MinDropWidth** : obtiene o establece el ancho de control desplegable mínimo
- **MinDropHeight** : obtiene o establece la altura mínima de control desplegable
- **MaxDropWidth** : obtiene o establece el ancho máximo del control desplegable
- **MaxDropHeight** : obtiene o establece la altura máxima de control desplegable

Tenga en cuenta que las 4 propiedades enumeradas anteriormente están ocultas de la ventana Propiedades y IntelliSense no las sugerirá mientras edita páginas Aspx en Visual Studio.

Para expandir el ancho de la ventana desplegable del selector de clientes

El diseño predeterminado de 13 columnas definido para el selector de **clientes** en la pantalla **Pedidos de ventas** (SO.30.10.00) no se ajusta al rango de tamaño predeterminado especificado para la ventana desplegable del selector. Para ayudar a los usuarios a explorar la mayor cantidad de información posible y ahorrar tiempo en el desplazamiento horizontal para ver todas las columnas, debe aumentar el ancho máximo del control desplegable asignando un número mayor a la propiedad **MaxDropWidth** para el selector de **clientes** .

Para establecer el valor de la propiedad **MaxDropWidth** en el Editor de diseño, desmarque el botón de **opción Ocultar propiedades avanzadas** como se muestra en la siguiente captura de pantalla:

Layout Editor: SO301000 (Sales Orders)

PREVIEW CHANGES ACTIONS ▾

Override	Property	Value
<input type="checkbox"/>	EditPageUrl	
<input type="checkbox"/>	EnableClientScript	
<input type="checkbox"/>	EnableTheming	
<input type="checkbox"/>	EnableViewState	
<input type="checkbox"/>	ForeColor	
<input type="checkbox"/>	GridSkin	
<input type="checkbox"/>	Height	
<input type="checkbox"/>	height	
<input type="checkbox"/>	Hidden	
<input type="checkbox"/>	HideEnterKey	
<input type="checkbox"/>	HintField	
<input type="checkbox"/>	HintLabelID	
<input type="checkbox"/>	LabelID	
<input type="checkbox"/>	LabelPostfix	
<input type="checkbox"/>	LabelText	
<input type="checkbox"/>	MaxDropHeight	
<input checked="" type="checkbox"/>	MaxDropWidth	2000
<input type="checkbox"/>	MenuImages	
<input type="checkbox"/>	MenuStyles	
<input type="checkbox"/>	MinDropHeight	

Después de publicar la personalización, los usuarios pueden disfrutar del nuevo diseño del selector de **clientes** , ahora ampliado en todo el marco de trabajo:

* Order Type: * Customer: Ordered Qty:

Order Nbr.: **Select - Customer**

Status:

* Date:

* Requested

Customer C

External Re

Document De

🔄 +

📄 🗑️ 📄 *Bra

Customer ID	Customer Name	Address Line 1	Address Line 2
ABARTENDE	USA Bartending School	203 Lower Notch Rd	
ABCHOLDING	ABC Holdings Inc	65 Broadway	
ABCSTUDIOS	ABC Studios Inc	77 W 66th St # 13	
ABCVENTURE	ABC Capital Ventures	601 W Girard Ave	
▶ ACTIVESTAF	Active Staffing Service	460 W 34th St	
ALPHABETLD	Alphabetland School Center	17575 Newbridge Rd	
AMROBANK	AMRO Bank N.V.	351th Fl., Atago Green Hills ...	55-12, Atago 26-chome, Min...
ANTUNSWEST	Antun's of Westchester	15 S Central Ave	
APOSTELSCH	Church of The Apostles	406 Willet Dr	
ARTCAGES	Artcages	22112 Clay Spring Loop	
ASAHISUNTR	Asahi Sun Tours	45-87, Shiba Daimon 17-ch...	Hamamatsucho Seiwa Bldg.
ASBLBAR	Nautilus Bar SABL	1216 Rue Lamartine	
AVACUST1	Avalara Customer	101 E Front St	
BEAUTYSCH	New York International Beau...	143-53 South Dr	
BESTYPEIMG	Bestype Image	4580 Broadway	
BIBIMBAB	Bibimbab Korean Restaurant	153 Lower Notch Rd	
BORDERSHOP	Borders Books, Music & Cafe	3111 McCommas Blvd	
BOULDERCR	Boulder Couriers Denver	1899 Wynkoop St	Suite 700
BRASSKEY	Brass Key Bar	11749 Livernois Ave	

Lea Cambio de tamaño de la ventana desplegable del selector en línea:

<https://riptutorial.com/es/acumatica/topic/9524/cambio-de-tamano-de-la-ventana-desplegable-del-selector>

Capítulo 9: Cambios significativos en la API entre versiones

Examples

PXSelectGroupBy y Bit Values en Acumatica 5.1 y 5.2+

El método de generación de SQL a partir de las vistas de datos BQL `PXSelectGroupBy<>` se ha cambiado en Acumatica Framework 5.2.

Las siguientes secciones ilustran las diferencias usando el ejemplo de `PXSelectGroupBy<FinYear, Aggregate<GroupBy<FinYear.finPeriods>>>.Select(graph)` :

Acumatica Framework 5.2 y posteriores

```
SELECT Max([finyear].[year]),
       Max([finyear].[startdate]),
       Max([finyear].[enddate]),
       [finyear].[finperiods],
       -- Attention!
       CONVERT (BIT, Max([finyear].[customperiods] + 0)),
       --
       Max([finyear].[begfinyearhist]),
       Max([finyear].[periodsstartdatehist]),
       Max([finyear].[noteid]),
       ( NULL ),
       ( NULL ),
       ( NULL ),
       Max([finyear].[tstamp]),
       Max([finyear].[createdbyid]),
       Max([finyear].[createdbyscreenid]),
       Max([finyear].[createddatetime]),
       Max([finyear].[lastmodifiedbyid]),
       Max([finyear].[lastmodifiedbyscreenid]),
       Max([finyear].[lastmodifieddatetime])
FROM   finyear FinYear
WHERE  ( finyear.companyid = 2 )
GROUP BY [finyear].[finperiods]
ORDER BY Max([finyear].[year])
```

Acumatica Framework 5.1 y anteriores

```
SELECT Max([finyear].[year]),
       Max([finyear].[startdate]),
       Max([finyear].[enddate]),
       [finyear].[finperiods],
       -- Attention!
       ( NULL ),
```

```

--
Max([finyear].[begfinyearhist]),
Max([finyear].[periodsstartdatehist]),
( NULL ),
( NULL ),
( NULL ),
Max([finyear].[tstamp]),
( NULL ),
Max([finyear].[createdbyscreenid]),
Max([finyear].[createddatetime]),
( NULL ),
Max([finyear].[lastmodifiedbyscreenid]),
Max([finyear].[lastmodifieddatetime])
FROM   finyear FinYear
WHERE  ( finyear.companyid = 2 )
GROUP BY [finyear].[finperiods]
ORDER BY Max([finyear].[year])

```

Explicación

De forma predeterminada, el agregado `Max()` se aplica a todos los campos que no se mencionan explícitamente en una declaración BQL.

Sin embargo, en Acumatica 5.1 y anteriores, excluye los campos `CreatedByID`, `LastModifiedByID` y `bool`. Cuando se traducen a SQL, estos campos siempre serán `null` menos que los agrupe explícitamente.

A partir de la versión 5.2, `Max()` también se aplicará de forma predeterminada para ellos.

Lea Cambios significativos en la API entre versiones en línea:

<https://riptutorial.com/es/acumatica/topic/9697/cambios-significativos-en-la-api-entre-versiones>

Capítulo 10: Creando campos de fecha y hora en Acumatica

Introducción

Este tema lo guiará a través de las diferentes opciones disponibles en Acumatica Framework para crear campos de fecha y hora en una clase de acceso a datos (DAC).

Examples

El atributo DateAndTime de PX (DB)

El atributo **PXDBDateAndTime** y el atributo **PXDateAndTime** están diseñados para funcionar con un campo DAC del `Nullable<DateTime>` (`DateTime?`) De `Nullable<DateTime>` y almacenan las partes de valor de fecha y hora dentro de un solo campo:

```
#region UsrDateAndTime
public abstract class usrDateAndTimeAttribute : IBqlField
{
}

[PXDBDateAndTime(
    DisplayNameDate = "Date Value Part",
    DisplayNameTime = "Time Value Part")]
public DateTime? UsrDateAndTime { get; set; }
#endregion
```

Desde la perspectiva de la interfaz de usuario, para un campo decorado con **PXDBDateAndTimeAttribute** o **PXDateAndTimeAttribute**, se espera que uno cree controles de entrada separados para las partes de valor de fecha y hora:

DATE AND TIME FIELD

Date Value Part: Time Value Part:

```
<px:PXDateTimeEdit runat="server" ID="edUsrDate" DataField="UsrDateAndTime_Date" />
<px:PXDateTimeEdit runat="server" ID="edUsrTime" DataField="UsrDateAndTime_Time"
TimeMode="True" />
```

o columnas de cuadrícula separadas para ingresar y mostrar los valores de fecha y hora:

Date Value Part	Time Value Part
7/14/2017	9:30 AM

```
<Columns>
...
<px:PXGridColumn DataField="UsrDateAndTime_Date" Width="90px" />
```

```

    <px:PXGridColumn DataField="UsrDateAndTime_Time" Width="90px" TimeMode="True" />
    ...
</Columns>

```

El atributo PXDBTime

El atributo **PXDBTime** está diseñado para funcionar con un campo DAC del tipo `Nullable<DateTime>` (`DateTime?`) Y almacenar solo la parte de tiempo sin fecha dentro de un campo DAC:

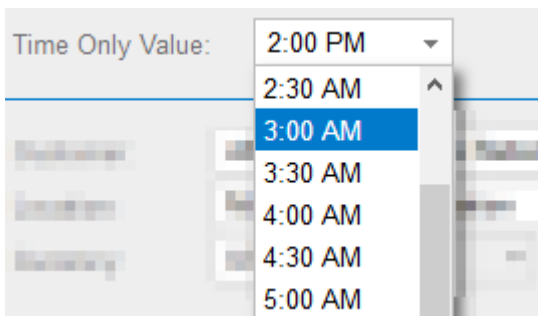
```

#region UsrTime
public abstract class usrTime : IBqlField
{
}

[PXDBTime(DisplayMask = "t", InputMask = "t")]
[PXUIField(DisplayName = "Time Only Value")]
public DateTime? UsrTime { get; set; }
#endregion

```

En la interfaz de usuario, para un campo decorado con **PXDBTimeAttribute**, el sistema crea un control de entrada que acepta solo los valores de tiempo en un formulario:

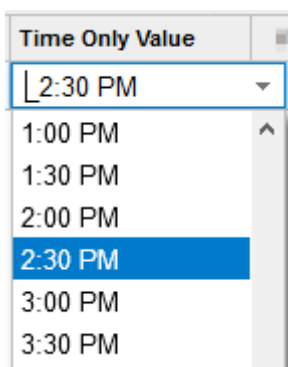


```

<px:PXDateTimeEdit runat="server" ID="edUsrTime" DataField="UsrTime" TimeMode="True" />

```

y dentro de una celda de cuadrícula:



```

<Columns>
    ...
    <px:PXGridColumn DataField="UsrTime" Width="120px" TimeMode="True" />
    ...
</Columns>

```

El atributo de atributo de fecha PX (DB)

El atributo **PXDBDate** y el atributo **PXDate** están diseñados para funcionar con un campo DAC del tipo de `Nullable<DateTime>` (`DateTime?`) Y almacenan el valor de la fecha con una parte de tiempo opcional dentro de un solo campo. Whether **PX (DB) DateAttribute** debería ahorrar tiempo además de que la fecha en un campo DAC está definida por la propiedad **PreserveTime** : cuando **PreserveTime** se establece en **Verdadero** , la parte de tiempo de un valor de campo se conserva, de lo contrario, solo se guarda la parte de fecha en un Campo DAC:

```
#region UsrDateTime
public abstract class usrDateTime : IBqlField
{
}

[PXDBDate(PreserveTime = true, InputMask = "g")]
[PXUIField(DisplayName = "DateTime Value")]
public DateTime? UsrDateTime { get; set; }
#endregion

#region UsrDate
public abstract class usrDate : IBqlField
{
}

[PXDBDate]
[PXUIField(DisplayName = "Date Value")]
public DateTime? UsrDate { get; set; }
#endregion
```

En la interfaz de usuario, para un campo decorado con **PXDBDateAttribute** o **PXDateAttribute**, el sistema crea un control de entrada que acepta solo valores de fecha o valores de fecha y hora según el valor de la propiedad **PreserveTime** . Este concepto funciona exactamente igual en un formulario:

DATE AND OPTIONAL TIME FIELD

DateTime Value: Date Value:

```
<px:PXDateTimeEdit runat="server" ID="edUsrDateTime" DataField="UsrDateTime" Size="SM" />
<px:PXDateTimeEdit runat="server" ID="edUsrDate" DataField="UsrDate" />
```

y dentro de una celda de cuadrícula:

DateTime Value	Date Value
7/11/2017 2:45 PM	7/19/2017

```
<Columns>
...
<px:PXGridColumn DataField="UsrDateTime" Width="130px" />
<px:PXGridColumn DataField="UsrDate" Width="90px" />
...
</Columns>
```

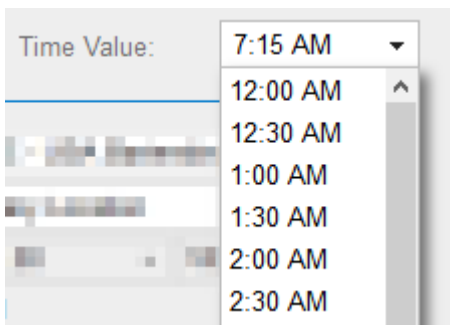
El atributo PXDBTimeSpan

El atributo **PXDBTimeSpan** está diseñado para funcionar con un campo DAC del tipo `Nullable<int> (int?)` Y almacenar el valor de tiempo dentro de un campo DAC como el número de minutos transcurridos desde la medianoche:

```
#region UsrTimeInt
public abstract class usrTimeInt : IBqlField
{
}

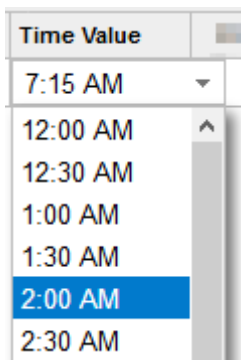
[PXDBTimeSpan(DisplayMask = "t", InputMask = "t")]
[PXUIField(DisplayName = "Time Value")]
public int? UsrTimeInt { get; set; }
#endregion
```

En la interfaz de usuario, para un campo decorado con **PXDBTimeSpanAttribute**, el sistema crea un menú desplegable con valores de intervalo de media hora en un formulario:



y dentro de una celda de cuadrícula:

```
<px:PXDateTimeEdit runat="server" ID="edUsrTimeInt" DataField="UsrTimeInt" TimeMode="true" />
```



```
<px:PXGridColumn DataField="UsrTimeInt" Width="90px" TimeMode="true" />
```

El atributo PXTimeList

El atributo **PXTimeList** está diseñado para trabajar con un campo DAC del tipo `Nullable<int> (int?)` Y almacenar el valor del intervalo de tiempo dentro de un campo DAC como un número de minutos:

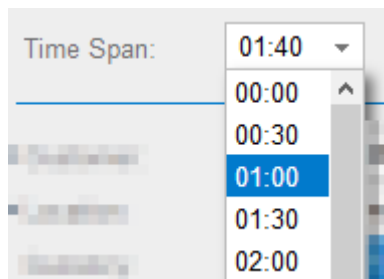
```

#region UsrTimeSpan
public abstract class usrTimeSpan : IBqlField
{ }

[PXDBInt]
[PXTimeList]
[PXUIField(DisplayName = "Time Span")]
public int? UsrTimeSpan { get; set; }
#endregion

```

En la interfaz de usuario, para un campo decorado con **PXTimeListAttribute**, el sistema crea un menú desplegable con valores de intervalo de 30 minutos en un formulario:

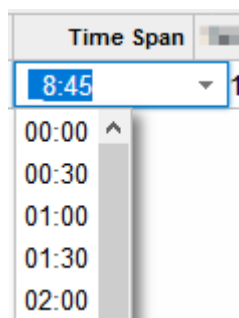


```

<px:PXTimeSpan ID="edUsrTimeSpan" runat="server" DataField="UsrTimeSpan" InputMask="hh:mm" />

```

y dentro de una celda de cuadrícula:



```

<RowTemplate>
    ...
    <px:PXTimeSpan ID="edgUsrTimeSpan" runat="server" DataField="UsrTimeSpan"
InputMask="hh:mm" />
    ...
</RowTemplate>
<Columns>
    ...
    <px:PXGridColumn DataField="UsrTimeSpan" Width="90px" RenderEditorText="True" />
    ...
</Columns>

```

Lea [Creando campos de fecha y hora en Acumatica en línea](https://riptutorial.com/es/acumatica/topic/10783/creando-campos-de-fecha-y-hora-en-acumatica):

<https://riptutorial.com/es/acumatica/topic/10783/creando-campos-de-fecha-y-hora-en-acumatica>

Capítulo 11: Descarga de archivos adjuntos a una entidad de detalle mediante API basada en contrato

Introducción

Este tema mostrará cómo descargar archivos adjuntos a una entidad de detalle dentro de Acumatica ERP mediante el uso de la API basada en contratos.

Observaciones

El fragmento de código anterior se creó utilizando el [marco Json.NET](#) (`Newtonsoft.Json.dll`).

Para obtener el encabezado de la cookie HTTP a partir de una respuesta SOAP, agregue una referencia a los ensamblados .Net Framework `System.ServiceModel` y `System.ServiceModel.Web` y los siguientes 2 usando directivas en su archivo de código:

```
using System.ServiceModel;  
using System.ServiceModel.Web;
```

Examples

Encabezado de cookie HTTP de una respuesta SOAP compartida por clientes SOAP y REST

Existe una limitación en la API basada en contrato SOAP de Acumatica que permite descargar archivos adjuntos solo para una entidad de nivel superior. [Desafortunadamente](#), cualquier intento de usar el método `GetFiles ()` para obtener los adjuntos de una entidad detallada dará como resultado el error "La **entidad sin enlace de pantalla no se puede usar como entidad de nivel superior** ". Nos dice que solo se puede usar con un top Entidad de nivel definida en el punto final del servicio web.

Otra limitación con el método `GetFiles ()` es que siempre devuelve el contenido de todos los archivos adjuntos a una entidad. No hay opción de recuperar primero solo los nombres de archivos y luego decidir qué archivos en particular se descargarán de Acumatica.

Afortunadamente, existe una forma mejor y más controlable de trabajar con los archivos adjuntos que se proporcionan con la API REST basada en el contrato. La matriz de `files` devuelta como parte de cada entidad exportada por la API REST basada en contrato solo contiene:

- nombres de archivos (la propiedad de **nombre de archivo**)
- identificadores de archivo (la propiedad **id**)
- Referencias de hipertexto (la propiedad **href**), que pueden usarse más adelante para

descargar el contenido del archivo

Para obtener un ejemplo de cómo obtener una lista de archivos adjuntos a cualquier entidad desde el punto final del servicio web y recuperar el contenido de un archivo en particular a través de la API REST basada en el contrato, consulte la [Ayuda del producto Acumatica](#)

¿Cómo se pueden descargar los archivos adjuntos a una entidad de detalle si el proyecto de integración completo se desarrolló con la API basada en contrato SOAP? Como se muestra en el fragmento de código a continuación, es posible pasar el encabezado de la cookie HTTP desde una respuesta SOAP al cliente de la API REST que se utiliza exclusivamente para trabajar con los archivos adjuntos:

```
using (var soapClient = new DefaultSoapClient())
{
    var address = new Uri("http://localhost/AcumaticaERP/entity/Default/6.00.001/");
    CookieContainer cookieContainer;
    using (new OperationContextScope(soapClient.InnerChannel))
    {
        soapClient.Login(login, password, null, null, null);
        string sharedCookie = WebOperationContext.Current.IncomingResponse.Headers["Set-
Cookie"];
        cookieContainer = new CookieContainer();
        cookieContainer.SetCookies(address, sharedCookie);
    }
    try
    {
        var shipment = new Shipment()
        {
            ShipmentNbr = new StringSearch { Value = "001301" },
            ReturnBehavior = ReturnBehavior.OnlySpecified
        };
        shipment = soapClient.Get(shipment) as Shipment;

        var restClient = new HttpClient(
            new HttpClientHandler
            {
                UseCookies = true,
                CookieContainer = cookieContainer
            });
        restClient.BaseAddress = address;// new
        Uri("http://localhost/059678/entity/Default/6.00.001/");

        var res = restClient.GetAsync("Shipment/" + shipment.ID + "?$expand=Packages")
            .Result.EnsureSuccessStatusCode();
        var shipmentWithPackages = res.Content.ReadAsStringAsync().Result;

        JObject jShipment = JObject.Parse(shipmentWithPackages);
        JArray jPackages = jShipment.Value<JArray>("Packages");
        foreach (var jPackage in jPackages)
        {
            JArray jFiles = jPackage.Value<JArray>("files");
            string outputDirectory = ".\\Output\\";
            if (!Directory.Exists(outputDirectory))
            {
                Directory.CreateDirectory(outputDirectory);
            }

            foreach (var jFile in jFiles)
```

```
        {
            string fullFileName = jFile.Value<string>("filename");
            string fileName = Path.GetFileName(fullFileName);
            string href = jFile.Value<string>("href");

            res = restClient.GetAsync(href).Result.EnsureSuccessStatusCode();
            byte[] file = res.Content.ReadAsByteArrayAsync().Result;
            System.IO.File.WriteAllBytes(outputDirectory + fileName, file);
        }
    }
}
finally
{
    soapClient.Logout();
}
}
```

Lea Descarga de archivos adjuntos a una entidad de detalle mediante API basada en contrato en línea: <https://riptutorial.com/es/acumatica/topic/10692/descarga-de-archivos-adjuntos-a-una-entidad-de-detalle-mediante-api-basada-en-contrato>

Capítulo 12: Exportación de registros a través de la API basada en pantalla

Introducción

Este tema mostrará cómo exportar registros de Acumatica ERP a través de la API basada en pantalla. La API basada en pantalla de Acumatica ERP proporciona solo la interfaz SOAP. Si su plataforma de desarrollo tiene soporte limitado para servicios web SOAP, considere la API basada en el contrato que proporciona las interfaces SOAP y REST. Para obtener más información sobre la API basada en pantalla, consulte la [documentación de Acumatica ERP](#)

Observaciones

Todos los ejemplos proporcionados en este tema se crearon con el contenedor de API basado en pantalla. Si desea que su aplicación cliente no dependa de los cambios en la interfaz de usuario de la aplicación ERP de Acumatica, debe usar el contenedor de API basado en pantalla, que se describe en la [documentación de Acumatica ERP](#).

Examples

Exportación de datos desde un formulario de entrada con una sola clave primaria

La pantalla **Artículos en stock** (IN.20.25.00) es uno de los formularios de ingreso de datos más utilizados de Acumatica ERP para exportar datos. **El ID de inventario** es la única clave principal en la pantalla **Artículos en existencia** :

* Inventory ID:	AACOMPUT01	Product Workgroup:	
Item Status:	Active	Product Manager:	
Description:	Acer Laptop Computer		

General Settings	Price/Cost Info	Warehouse Details	Vendor Details	Attributes	Packaging	Cross-Reference	Replenish
------------------	-----------------	-------------------	----------------	------------	-----------	-----------------	-----------

ITEM DEFAULTS

Item Class:	ELECCOMP - Electronics & Compute	* Base Unit:	EA
Type:	Finished Good	* Sales Unit:	EA
<input type="checkbox"/> Is a Kit		* Purchase Unit:	EA
Valuation Method:	Average		
* Tax Category:	TAXABLE - Taxable Goods and Servic		
* Posting Class:	ELE - Electronics & Computers		
* Lot/Serial Class:	NOTTRACKED - Not Tracked		
Auto-Incremental Value:			

WAREHOUSE DEFAULTS

Default Warehouse:	WHOLESALE - HQ Wholesale Wareh
Default Issue From:	R1S1 - Row 1 Shelf 1
Default Receipt To:	RECEIVING - Receiving

UNIT OF MEASURE

* From Unit	Multiply/Divide	Co
-------------	-----------------	----

PHYSICAL INVENTORY

PI Cycle:	
ABC Code:	
<input type="checkbox"/> Fixed ABC Code	
Movement Class:	
<input type="checkbox"/> Fixed Movement	

Para exportar registros desde un formulario de ingreso de datos, su solicitud SOAP siempre debe comenzar con el comando `ServiceCommands.Every[Key]`, donde `[Key]` debe reemplazarse con el nombre de la clave principal.

Para exportar todos los artículos en stock en una sola llamada de servicio web:

```
Screen context = new Screen();
context.CookieContainer = new System.Net.CookieContainer();
context.Url = "http://localhost/AcumaticaERP/Soap/IN202500.asmx";
```

```

context.Login(username, password);
try
{
    Content stockItemsSchema = PX.Soap.Helper.GetSchema<Content>(context);
    Field lastModifiedField = new Field
    {
        ObjectName = stockItemsSchema.StockItemSummary.InventoryID.ObjectName,
        FieldName = "LastModifiedDateTime"
    };
    var commands = new Command[]
    {
        stockItemsSchema.StockItemSummary.ServiceCommands.EveryInventoryID,
        stockItemsSchema.StockItemSummary.InventoryID,
        stockItemsSchema.StockItemSummary.Description,
        stockItemsSchema.GeneralSettingsItemDefaults.ItemClass,
        stockItemsSchema.GeneralSettingsUnitOfMeasureBaseUnit.BaseUnit,
        lastModifiedField
    };
    var items = context.Export(commands, null, 0, false, false);
}
finally
{
    context.Logout();
}

```

Con el tiempo, la cantidad de datos en cualquier aplicación de ERP tiende a crecer en tamaño. Si va a exportar todos los registros de su instancia de Acumatica ERP en una sola llamada de servicio web, muy pronto podría notar errores de tiempo de espera. Aumentar el tiempo de espera es una posible solución de una sola vez, pero no muy buena a largo plazo. Su mejor opción para enfrentar este desafío es exportar artículos en stock en lotes de varios registros.

Para exportar artículos en stock en lotes de 10 registros:

```

Screen context = new Screen();
context.CookieContainer = new System.Net.CookieContainer();
context.Url = "http://localhost/AcumaticaERP/Soap/IN202500.asmx";
context.Login(username, password);
try
{
    Content stockItemsSchema = PX.Soap.Helper.GetSchema<Content>(context);
    Field lastModifiedField = new Field
    {
        ObjectName = stockItemsSchema.StockItemSummary.InventoryID.ObjectName,
        FieldName = "LastModifiedDateTime"
    };
    var commands = new Command[]
    {
        stockItemsSchema.StockItemSummary.ServiceCommands.EveryInventoryID,
        stockItemsSchema.StockItemSummary.InventoryID,
        stockItemsSchema.StockItemSummary.Description,
        stockItemsSchema.GeneralSettingsItemDefaults.ItemClass,
        stockItemsSchema.GeneralSettingsUnitOfMeasureBaseUnit.BaseUnit,
        lastModifiedField
    };
}

```

```

var items = context.Export(commands, null, 10, false, false);

while (items.Length == 10)
{
    var filters = new Filter[]
    {
        new Filter
        {
            Field = stockItemsSchema.StockItemSummary.InventoryID,
            Condition = FilterCondition.Greater,
            Value = items[items.Length - 1][0]
        }
    };
    items = context.Export(commands, filters, 10, false, false);
}
}
finally
{
    context.Logout();
}
}

```

Existen 2 diferencias principales entre el enfoque de llamada única y la exportación en lotes:

- El parámetro **topCount** del comando **Exportar** siempre se estableció en 0 en el enfoque de llamada única
- al exportar registros en lotes, el tamaño de un lote se configura a través del parámetro **topCount** complementado por la matriz de **filtros** para solicitar el siguiente conjunto de resultados

Exportación de datos desde un formulario de entrada con una clave primaria compuesta

La pantalla **Pedidos de ventas** (SO.30.10.00) es un ejemplo perfecto de un formulario de entrada de datos con una clave principal compuesta. La clave principal en la pantalla **Pedidos de ventas** está compuesta por el **tipo de pedido** y el **número de pedido** :

* Order Type:	SO	* Customer:	FDITAMPA - Tampa Bay Food Distributor	Ordered Qty
Order Nbr.:	SO003724	* Location:	MAIN - Primary Location	VAT Exempt
	<input type="checkbox"/> Hold	Currency:	USD 1.00	VAT Taxable
Status:	Completed	<input type="checkbox"/> Credit Hold	VIEW BASE	Tax Total:
* Date:	1/1/2017	* Project:	X - Non-Project Code.	Order Total:
* Requested On:	1/1/2017	Description:	Food distribution	
Customer Order:	FDITAMPA201			
External Refer...				

Document Details	Tax Details	Commissions	Financial Settings	Payment Settings	Shipping Settings	Discount Details
------------------	-------------	-------------	--------------------	------------------	-------------------	------------------

				ALLOCATIONS	ADD INVOICE	ADD STOCK ITEM	PO LINK	INVENTORY SUMM
--	--	--	--	-------------	-------------	----------------	---------	----------------

		* Branch	* Inventory ID	Free Item	Warehous	Line Description	* UOM	Quantity	Q Ship
>		VA	AAPOW...	<input type="checkbox"/>	RETAIL	Poweraid 32 Oz - lot numb...	EA	3,880.00	3,880.00

La estrategia recomendada de 2 pasos para exportar datos desde la pantalla **Pedidos de ventas** o cualquier otro formulario de ingreso de datos con una clave primaria compuesta a través de la API basada en pantalla:

- en el paso 1 solicita todos los tipos de pedidos creados previamente en su aplicación Acumatica ERP
- El segundo paso es exportar pedidos de cada tipo de forma independiente, ya sea en una sola llamada o en lotes

Para solicitar todo tipo de pedidos existentes:

```
Screen context = new Screen();
context.CookieContainer = new System.Net.CookieContainer();
context.Url = "http://localhost/AcumaticaERP/Soap/SO301000.asmx";
context.Login(username, password);
try
{
    Content orderSchema = PX.Soap.Helper.GetSchema<Content>(context);
    var commands = new Command[]
    {
        orderSchema.OrderSummary.ServiceCommands. EveryOrderType,
```



```

        orderSchema.OrderSummary.OrderType,
    };

    var types = context.Export(commands, null, 1, false, false);
}
finally
{
    context.Logout();
}

```

En la llamada SOAP anterior, observe el parámetro **topCount** del comando **Exportar** establecido en 1. El propósito de esta solicitud es solo para obtener todos los tipos de pedidos creados previamente en su aplicación Acumatica ERP, no para exportar datos.

Para exportar registros de cada tipo de forma independiente en lotes:

```

Screen context = new Screen();
context.CookieContainer = new System.Net.CookieContainer();
context.Url = "http://localhost/AcumaticaERP/Soap/SO301000.asmx";
context.Login(username, password);
try
{
    Content orderSchema = PX.SoaP.Helper.GetSchema<Content>(context);
    var commands = new Command[]
    {
        orderSchema.OrderSummary.ServiceCommands.EveryOrderType,
        orderSchema.OrderSummary.OrderType,
    };
    var types = context.Export(commands, null, 1, false, false);

    for (int i = 0; i < types.Length; i++)
    {
        commands = new Command[]
        {
            new Value
            {
                LinkedCommand = orderSchema.OrderSummary.OrderType,
                Value = types[i][0]
            },
            orderSchema.OrderSummary.ServiceCommands.EveryOrderNbr,
            orderSchema.OrderSummary.OrderType,
            orderSchema.OrderSummary.OrderNbr,
            orderSchema.OrderSummary.Customer,
            orderSchema.OrderSummary.CustomerOrder,
            orderSchema.OrderSummary.Date,
            orderSchema.OrderSummary.OrderedQty,
            orderSchema.OrderSummary.OrderTotal
        };
        var orders = context.Export(commands, null, 100, false, false);
        while (orders.Length == 100)
        {
            var filters = new Filter[]
            {
                new Filter

```

```

        {
            Field = orderSchema.OrderSummary.OrderNbr,
            Condition = FilterCondition.Greater,
            Value = orders[orders.Length - 1][1]
        }
    };
    orders = context.Export(commands, filters, 100, false, false);
}
}
}
finally
{
    context.Logout();
}
}

```

El ejemplo anterior muestra cómo exportar todos los pedidos de venta de Acumatica ERP en lotes de 100 registros. Para exportar pedidos de ventas de cada tipo de manera independiente, su solicitud SOAP siempre debe comenzar con el comando `Value`, que determina el tipo de pedidos que se exportarán. Después del comando `Value` utilizado para establecer el primer valor de la clave, se `ServiceCommands.Every[Key]` comando `ServiceCommands.Every[Key]`, donde `[Key]` debe reemplazarse con el nombre de la segunda clave.

Para exportar registros de un tipo específico:

En caso de que necesite exportar pedidos de ventas de un tipo específico, es posible definir explícitamente el tipo de pedidos con el comando `Value` al comienzo de su solicitud de SOAP seguido del enfoque de llamada única o la exportación en lotes.

Para exportar todos los pedidos de venta del tipo **IN** en una llamada:

```

Screen context = new Screen();
context.CookieContainer = new System.Net.CookieContainer();
context.Url = "http://localhost/AcumaticaERP/Soap/SO301000.asmx";
context.Login(username, password);
try
{
    Content orderSchema = PX.SoaP.Helper.GetSchema<Content>(context);
    var commands = new Command[]
    {
        new Value
        {
            LinkedCommand = orderSchema.OrderSummary.OrderType,
            Value = "IN"
        },
        orderSchema.OrderSummary.ServiceCommands.EveryOrderNbr,
        orderSchema.OrderSummary.OrderType,
        orderSchema.OrderSummary.OrderNbr,
        orderSchema.OrderSummary.Customer,
        orderSchema.OrderSummary.CustomerOrder,
        orderSchema.OrderSummary.Date,
        orderSchema.OrderSummary.OrderedQty,
        orderSchema.OrderSummary.OrderTotal
    };
    var orders = context.Export(commands, null, 0, false, false);
}
}

```

```
finally
{
    context.Logout ();
}
```

Lea [Exportación de registros a través de la API basada en pantalla en línea](https://riptutorial.com/es/acumatica/topic/9288/exportacion-de-registros-a-traves-de-la-api-basada-en-pantalla):

<https://riptutorial.com/es/acumatica/topic/9288/exportacion-de-registros-a-traves-de-la-api-basada-en-pantalla>

Capítulo 13: Exportación de registros a través de REST API basada en contrato

Introducción

Este tema demostrará cómo exportar registros de Acumatica ERP a través de la API basada en el contrato REST. A diferencia de la API basada en pantalla de Acumatica ERP, la API basada en contrato proporciona interfaces SOAP y REST. Para obtener más información sobre la API basada en el contrato, consulte la [documentación de Acumatica ERP](#)

Observaciones

Para comunicarse con la API basada en contratos REST de Acumatica ERP, su aplicación cliente siempre debe realizar los siguientes 3 pasos:

1. inicie sesión en la instancia de Acumatica ERP y obtenga una cookie con la información de la sesión del usuario
2. interactúe con uno de los puntos finales API basados en contrato disponibles en la instancia de Acumatica ERP
3. cerrar sesión en Acumatica ERP para cerrar sesión de usuario

Todas las muestras proporcionadas en este tema se crearon con el punto final **predeterminado**, siempre implementado como parte del proceso de instalación estándar de Acumatica ERP. En la pantalla de puntos finales del **servicio web** (SM.20.70.60) puede ver los detalles de los puntos finales existentes o configurar sus puntos finales personalizados de los servicios web basados en contratos de Acumatica ERP:



VIEW ENDPOINT SERVICE

VIEW MAINTENANCE SERVICE

EXTEND ENDP

* Endpoint Name:

* Endpoint Version:

+ INSERT

ENDPOINT

- Adjustment
- AttributeDefinition
- Contact
- CurrencyRateHistoryInquiry
- Customer
- CustomerPaymentMethod
- Employee
- InventoryAllocationInquiry
- InventorySummaryInquiry
- ItemClass
- ItemSalesCategory

Endpoint Properties

* Endpoint Name:	Default	Base Endpo
* Endpoint Version:	6.00.001	Base Endpo
System Contract:	2	

Para su referencia, a continuación se muestra la implementación de la clase **RestService** utilizada en todas las muestras anteriores para interactuar con el servicio web basado en contratos de Acumatica ERP:

```
public class RestService : IDisposable
{
    private readonly HttpClient _httpClient;
    private readonly string _acumaticaBaseUrl;
    private readonly string _acumaticaEndpointUrl;

    public RestService(string acumaticaBaseUrl, string endpoint,
        string userName, string password,
        string company, string branch)
    {
        _acumaticaBaseUrl = acumaticaBaseUrl;
        _acumaticaEndpointUrl = _acumaticaBaseUrl + "/entity/" + endpoint + "/";
        _httpClient = new HttpClient(
            new HttpClientHandler
            {
                UseCookies = true,
                CookieContainer = new CookieContainer()
            })
        {
            BaseAddress = new Uri(_acumaticaEndpointUrl),
            DefaultRequestHeaders =
            {
                Accept = {MediaTypeWithQualityHeaderValue.Parse("text/json")}
            }
        };
    }
};
```

```

var str = new StringContent(
    new JavaScriptSerializer()
        .Serialize(
            new
            {
                name = userName,
                password = password,
                company = company,
                branch = branch
            }
        ),
    Encoding.UTF8, "application/json");

_httpClient.PostAsync(acumaticaBaseUrl + "/entity/auth/login", str)
    .Result.EnsureSuccessStatusCode();
}

void IDisposable.Dispose()
{
    _httpClient.PostAsync(_acumaticaBaseUrl + "/entity/auth/logout",
        new ByteArrayContent(new byte[0])).Wait();
    _httpClient.Dispose();
}

public string GetList(string entityName)
{
    var res = _httpClient.GetAsync(_acumaticaEndpointUrl + entityName)
        .Result.EnsureSuccessStatusCode();

    return res.Content.ReadAsStringAsync().Result;
}

public string GetList(string entityName, string parameters)
{
    var res = _httpClient.GetAsync(_acumaticaEndpointUrl + entityName + "?" + parameters)
        .Result.EnsureSuccessStatusCode();

    return res.Content.ReadAsStringAsync().Result;
}
}

```

Examples

Exportación de datos en una sola llamada REST

En este ejemplo, explorará cómo exportar los siguientes datos de Acumatica ERP en una sola llamada a través de la API basada en el contrato REST:

- Todos los artículos de stock existentes en la aplicación.
- Todo pedido de venta del tipo IN.

Si necesita exportar registros de Acumatica ERP, use la siguiente URL: `http://<Acumatica ERP instance URL>/entity/<Endpoint name>/<Endpoint version>/<Top-level entity>`

`<Top-level entity>` es el nombre de la entidad que va a exportar

Para exportar todos los artículos en stock en una sola llamada REST:

Para exportar registros de artículos de stock desde una instancia local de `AcumaticaERP` utilizando el punto final **predeterminado** de la versión **6.00.001** , debe usar la siguiente URL:

`http://localhost/AcumaticaERP/entity/Default/6.00.001/StockItem`

A continuación se muestra el código de muestra escrito en C # para exportar todos los artículos en stock enviando una sola llamada REST al punto final **predeterminado** de la versión **6.00.001** :

```
using (RestService rs = new RestService(
    @"http://localhost/AcumaticaERP/", "Default/6.00.001",
    username, password, company, branch))
{
    string stockItems = rs.GetList("StockItem");
}
```

Para exportar todos los pedidos de venta del tipo IN en una sola llamada REST:

Para exportar pedidos de venta del tipo **IN** desde una instancia local de `AcumaticaERP` utilizando el punto final **predeterminado** de la versión **6.00.001** , debe usar la siguiente URL:

`http://localhost/AcumaticaERP/entity/Default/6.00.001/SalesOrder?$filter=OrderType eq 'IN'`

A continuación se muestra el código de muestra escrito en C # para exportar todos los pedidos de venta del tipo **IN** enviando una sola llamada REST al punto final **predeterminado** de la versión **6.00.001** :

```
using (RestService rs = new RestService(
    @"http://localhost/StackOverflow/", "Default/6.00.001",
    username, password, company, branch))
{
    var parameters = "$filter=OrderType eq 'IN'";
    string inSalesOrders = rs.GetList("SalesOrder", parameters);
}
```

Implementación de paginación en múltiples solicitudes REST

En este ejemplo, explorará cómo exportar los siguientes datos de Acumatica ERP en lotes a través de la API basada en el contrato REST:

- Stock de artículos existentes en la aplicación en lotes de 10 registros.
- Todas las órdenes de venta en lotes de 100 registros.

Para exportar artículos en stock en lotes de 10 registros con múltiples llamadas REST:

Para exportar los primeros 10 artículos en stock de una instancia local de `AcumaticaERP` utilizando el punto final **predeterminado** de la versión **6.00.001**, debe usar la siguiente URL:

```
http://localhost/AcumaticaERP/entity/Default/6.00.001/StockItem?$stop=10
```

En consecuencia, para solicitar artículos de 10 a 20, simplemente extienda la URL anterior con el parámetro de **filtro**:

```
http://localhost/AcumaticaERP/entity/Default/6.00.001/StockItem?$stop=10&$filter=InventoryID gt '<InventoryID>'
```

`<InventoryID>` es el ID del último artículo de stock exportado con una llamada REST anterior

A continuación se muestra el código de muestra escrito en C# para exportar todos los artículos en stock en lotes de 10 registros mediante el envío de varias llamadas REST al punto final **predeterminado** de la versión **6.00.001**:

```
using (RestService rs = new RestService(
    @"http://localhost/StackOverflow/", "Default/6.00.001",
    username, password, company, branch))
{
    var json = new JavaScriptSerializer();
    string parameters = "$stop=10";
    string items = rs.GetList("StockItem", parameters);
    var records = json.Deserialize<List<Dictionary<string, object>>>(items);

    while (records.Count == 10)
    {
        var inventoryID = records[records.Count - 1]["InventoryID"] as Dictionary<string,
object>;
        var inventoryIDValue = inventoryID.Values.First();
        string nextParameters = parameters + "&" +
            "$filter=" + string.Format("InventoryID gt '{0}'", inventoryIDValue);
        items = rs.GetList("StockItem", nextParameters);
        records = json.Deserialize<List<Dictionary<string, object>>>(items);
    }
}
```

Para exportar todos los pedidos de ventas en lotes de 100 registros con múltiples llamadas REST:

Para exportar las primeras 100 órdenes de venta desde una instancia local de `AcumaticaERP` utilizando el punto final **predeterminado** de la versión **6.00.001**, debe usar la siguiente URL:

```
http://localhost/AcumaticaERP/entity/Default/6.00.001/SalesOrder?$stop=100
```

Dado que la clave principal de la entidad **Pedido de venta** está compuesta por el **Tipo de pedido**

y el **Número de pedido** , en este ejemplo, utilizará una combinación de parámetros de **filtro** para los campos **Tipo de pedido** y **Número de pedido** :

- para solicitar pedidos de ventas de 100 a 200 del tipo **SO** , debe usar la siguiente URL:
`http://localhost/AcumaticaERP/entity/Default/6.00.001/SalesOrder?$top=100&$filter=OrderType eq 'SO' and OrderNbr gt '<OrderNbr>'`

<OrderNbr> es el número del último pedido de venta exportado con una llamada REST anterior

- en consecuencia, para solicitar las primeras 100 órdenes de venta del tipo **SO** próximo, debe usar la siguiente URL:
`http://localhost/AcumaticaERP/entity/Default/6.00.001/SalesOrder?$top=100&$filter=OrderType gt 'SO' and OrderNbr gt ''`

A continuación se muestra el código de muestra escrito en C # para exportar todos los pedidos de ventas en lotes de 100 registros con múltiples llamadas REST al punto final **predeterminado** de la versión **6.00.001** :

```
using (RestService rs = new RestService(
    @"http://localhost/StackOverflow/", "Default/6.00.001",
    username, password, company, branch))
{
    var json = new JavaScriptSerializer();
    string parameters = "$top=100";
    string items = rs.GetList("SalesOrder", parameters);
    var records = json.Deserialize<List<Dictionary<string, object>>>(items);

    bool sameOrderType = true;
    while (records.Count > 0 && (records.Count == 100 || !sameOrderType))
    {
        var orderType = records[records.Count - 1]["OrderType"] as Dictionary<string, object>;
        var orderTypeValue = orderType.Values.First();
        var orderNbr = records[records.Count - 1]["OrderNbr"] as Dictionary<string, object>;
        var orderNbrValue = orderNbr.Values.First();

        string nextParameters = parameters + "&" + "$filter=" +
            string.Format("OrderType {0} '{1}'", sameOrderType ? "eq" : "gt", orderTypeValue)
        + " and " +
            string.Format("OrderNbr gt '{0}'", sameOrderType ? orderNbrValue : "" );
        items = rs.GetList("SalesOrder", nextParameters);
        records = json.Deserialize<List<Dictionary<string, object>>>(items);
        sameOrderType = records.Count == 100;
    }
}
```

Lea **Exportación de registros a través de REST API basada en contrato en línea:**

<https://riptutorial.com/es/acumatica/topic/9298/exportacion-de-registros-a-traves-de-rest-api-basada-en-contrato>

Capítulo 14: Filtrado con valor múltiple con un solo selector.

Introducción

Esta es una forma de tener un valor múltiple dentro de un selector para filtrar una cuadrícula.

Examples

Recuperación de pedido de cliente para cliente múltiple

Cuando se trata de filtrar algún registro utilizando varios valores en un selector. Primero debe usar el `px:PXMultiSelector` en la página `aspx` en lugar del `px` normal: `PXSelector`. Luego, después de crear una gráfica que contenga al menos tres vistas y un delegado de vista. También necesitará al menos un DAC básico no unido.

Aquí hay una página de muestra con el `px:PXMultiSelector`:

```
<%@ Page Language="C#" MasterPageFile="~/MasterPages/FormDetail.master" AutoEventWireup="true"
ValidateRequest="false" CodeFile="TT000000.aspx.cs" Inherits="Page_TT000000" Title="Untitled
Page" %>

<%@ MasterType VirtualPath="~/MasterPages/FormDetail.master" %>

<asp:Content ID="cont1" ContentPlaceHolderID="phDS" runat="Server">
<px:PXDataSource ID="ds" runat="server" Visible="True" Width="100%"
    TypeName="MultiSelector.MultiInquiry"
    PrimaryView="MasterView">
    <CallbackCommands>
    </CallbackCommands>
</px:PXDataSource>
</asp:Content>
<asp:Content ID="cont2" ContentPlaceHolderID="phF" runat="Server">
<px:PXFormView ID="form" runat="server" DataSourceID="ds" DataMember="MasterView" Width="100%"
Height="100px" AllowAutoHide="false">
    <Template>
        <px:PXMultiSelector ID="edInventoryID" runat="server" Width="100%" DataSourceID="ds"
DataField="Customer" CommitChanges="True"></px:PXMultiSelector>
    </Template>
</px:PXFormView>
</asp:Content>
<asp:Content ID="cont3" ContentPlaceHolderID="phG" runat="Server">
<px:PXGrid ID="grid" runat="server" DataSourceID="ds" Width="100%" Height="150px"
SkinID="Details" AllowAutoHide="false">
    <Levels>
        <px:PXGridLevel DataMember="DetailsView">
            <Columns>
                <px:PXGridColumn DataField="OrderType" Width="70"></px:PXGridColumn>
                <px:PXGridColumn DataField="OrderNbr" Width="200"></px:PXGridColumn>
                <px:PXGridColumn DataField="OrderDesc" Width="100"></px:PXGridColumn>
                <px:PXGridColumn DataField="CustomerOrderNbr" Width="100"></px:PXGridColumn>
            </Columns>
        </px:PXGridLevel>
    </Levels>
</px:PXGrid>
</asp:Content>
```

```

        <px:PXGridColumn DataField="Status" Width="100"></px:PXGridColumn>
        <px:PXGridColumn DataField="RequestDate" Width="100"></px:PXGridColumn>
        <px:PXGridColumn DataField="ShipDate" Width="100"></px:PXGridColumn>
        <px:PXGridColumn DataField="CustomerID" Width="100"></px:PXGridColumn>
    </Columns>
</px:PXGridLevel>
</Levels>
<AutoSize Container="Window" Enabled="True" MinHeight="150" />
<ActionBar>
</ActionBar>
</px:PXGrid>
</asp:Content>

```

Aquí está el ejemplo de gráfico con las vistas y el delegado.

```

public class MultiInquiry : PXGraph<MultiInquiry>
{
    public PXCancel<MasterTable> Cancel;
    public PXFilter<MasterTable> MasterView;
    public PXSelect<SOOrder> DetailsView;

    public PXSelectJoin<SOOrder, LeftJoin<BAccount, On<SOOrder.customerID,
Equal<BAccount.bAccountID>>>, Where<BAccount.acctCD, In<Required<BAccount.acctCD>>>>> Orders2;

    protected virtual IEnumerable detailsView()
    {
        var list = new List<SOOrder>();
        var customers = MasterView.Current.Customer;
        if (customers != null)
        {
            List<string> customerList = new List<string>();
            customerList.AddRange(customers.Split(new string[] { "; " },
StringSplitOptions.None));
            object[] val = new object[] { customerList.ToArray() };

            foreach (PXResult<SOOrder> res in Orders2.Select(val))
            {
                SOOrder order = res;
                list.Add(order);
            }
        }
        return list;
    }
}

```

A esto le agregamos el DAC que contiene la definición del campo utilizado en el MultiSelector y la constante para seleccionar solo las cuentas de clientes.

```

[Serializable]
public class MasterTable : IBqlTable
{
    #region InventoryID
    public abstract class customer : IBqlField { }
    [PXString()]
    [PXUIField(DisplayName = "Customer")]
    [PXSelector(typeof(Search<BAccount.acctCD, Where<BAccount.type,
Equal<CustomerType>>>), ValidateValue = false)]
    public virtual string Customer { get; set; }

```

```

#endregion






}













public class CustomerType : Constant<string> { public CustomerType() : base("CU") { } }

```

Y el resultado de este ejemplo podría ser algo como esto:

Customer: ABCHOLDING × ABCSTUDIOS × ABARTENDE ×

			*Order Type	Order Nbr.	Description	Customer Order	Status
>			CS	001558	Warehouse ...		Complete
			CS	001559	Warehouse ...		Complete
			CS	001569	Warehouse ...		Complete
			CS	001570	Warehouse ...		Complete
			CS	001580	Warehouse ...		Complete
			CS	001584	Warehouse ...		Complete

Lea [Filtrado con valor múltiple con un solo selector](https://riptutorial.com/es/acumatica/topic/10707/filtrado-con-valor-multiple-con-un-solo-selector). en línea:

[https://riptutorial.com/es/acumatica/topic/10707/filtrado-con-valor-multiple-con-un-solo-selector-](https://riptutorial.com/es/acumatica/topic/10707/filtrado-con-valor-multiple-con-un-solo-selector)

Capítulo 15: Mecanismos de personalización

Examples

Uso de CacheAttached para reemplazar atributos de DAC en el gráfico

A veces, debe anular uno o más atributos de un campo de clase de acceso a datos (DAC) en particular para una pantalla en particular, sin cambiar el comportamiento existente para otras pantallas.

Reemplazo de todos los atributos

Supongamos que los atributos del campo DAC original se declaran como se muestra a continuación:

```
public class ARInvoice : IBqlTable
{
    [PXDBDecimal(4)]
    [PXDefault(TypeCode.Decimal, "0.0")]
    [PXUIField(DisplayName = "Commission Amount")]
    public virtual Decimal? CommnAmt
    {
        get;
        set;
    }
}
```

La forma básica de anular los atributos del campo en el gráfico es declarar un controlador de eventos `CacheAttached` en el gráfico que sigue la convención estándar para nombrar eventos del gráfico (tenga en cuenta la ausencia del argumento `EventArgs`). El cuerpo del controlador de eventos no se ejecutará, pero cualquier *atributo* que coloque en el controlador reemplazará los atributos en el campo DAC correspondiente:

```
[PXDBDecimal(4)]
[PXDefault(TypeCode.Decimal, "0.0")]
[PXUIField(DisplayName = "Commission Amount")]
[PXAdditionalAttribute(NecessaryProperty = true)]
protected virtual void ARInvoice_CommnnAmt_CacheAttached(PXCache sender) { }
```

Anexando un nuevo atributo al campo DAC

El conjunto de atributos colocado en el controlador `CacheAttached` **redefinirá el conjunto completo de los atributos** ubicados en el campo en el DAC. Esto es casi siempre una exageración; observe cómo en el ejemplo anterior, para agregar un solo atributo al campo, tuvo que copiar todas las demás declaraciones de atributos del DAC. Esto conduce a una duplicación de código no deseado, así como a la posibilidad de que DAC y el gráfico se desincronicen. Es

muy fácil imaginar una situación cuando alguien cambia la lógica `PXDefaultAttribute` de, por ejemplo, `PXDefaultAttribute` en el DAC, pero se olvida de actualizar todos los atributos correspondientes colocados en los controladores de `CacheAttached` en varios gráficos.

Para solucionar este problema, Acumatica Framework proporciona un atributo especial llamado `PXMergeAttributesAttribute`. Cuando este atributo se coloca en un controlador `CacheAttached`, puede reutilizar los atributos existentes definidos en el DAC.

Anexando un atributo utilizando `PXMergeAttributesAttribute`:

```
[PXMergeAttributes(Method = MergeMethod.Append)]
[PXAdditionalAttribute(NecessaryProperty = true)]
protected virtual void ARInvoice_CommnmAmt_CacheAttached(PXCache sender) { }
```

En el ejemplo anterior, todo el conjunto de atributos del DAC original se reutilizará, junto con cualquier atributo que haya declarado en el controlador de eventos `CacheAttached`.

`PXMergeAttributesAttribute` tiene otros comportamientos de fusión, de acuerdo con los siguientes valores posibles para la propiedad `Method`:

- `MergeMethod.Replace` reemplaza los atributos del DAC completamente (equivalente a la ausencia de `PXMergeAttributesAttribute`).
- `MergeMethod.Append` agrega los atributos del controlador `CacheAttached` a los atributos DAC originales.
- `MergeMethod.Merge` es similar a `Append`; sin embargo, también verifica si hay atributos en conflicto entre los atributos del manejador y los atributos del campo DAC. Si hay un conflicto, el atributo `CacheAttached` tiene prioridad y el atributo DAC correspondiente se descarta.

Anulando una propiedad única de un atributo

Un escenario de desarrollo de aplicaciones muy común ocurre cuando tiene que redefinir solo una propiedad del atributo de un DAC para una pantalla en particular; considerar la situación cuando se tiene que definir el `DisplayName` propiedad del `PXUIFieldAttribute`.

Para ese propósito, puede usar otro atributo especial provisto por Acumatica Framework:

`PXCustomizeBaseAttributeAttribute`. Su constructor acepta tres valores:

- El tipo del atributo DAC cuya propiedad debe ser anulada
- El nombre de la propiedad del atributo a anular (use el operador `nameof` en C # 6.0 para mantener el código)
- El nuevo valor para la propiedad especificada.

Supongamos que existe un requisito para cambiar el nombre de visualización de la UI de *Importe de la Comisión* a la *Comisión de moneda base* para una sola pantalla. El siguiente ejemplo de código muestra cómo implementar el comportamiento deseado.

```
[PXMergeAttributes(Method = MergeMethod.Append) ]
[PXCustomizeBaseAttribute(typeof(PXUIFieldAttribute), nameof(PXUIFieldAttribute.DisplayName),
"Base Currency Commission")]
protected virtual void ARInvoice_CommnnAmt_CacheAttached(PXCache sender) { }
```

En este ejemplo, `PXMergeAttributes` garantiza que los atributos DAC originales se conservan, y `PXCustomizeBaseAttribute` permite que el ingeniero de software anule el nombre de visualización del campo UI para el gráfico en cuestión.

Reemplazo de un atributo con otro atributo

Supongamos que hay un requisito para reemplazar el `PXDefaultAttribute` `PXDBDefaultAttribute` de un campo DAC con `PXDBDefaultAttribute` para una sola pantalla.

Esto se puede lograr de la siguiente manera:

```
[PXMergeAttributes(Method = MergeMethod.Append) ]
[PXRemoveBaseAttribute(typeof(PXDefaultAttribute))]
[PXDBDefault(typeof(SOShipment.siteID), PersistingCheck = PXPersistingCheck.Nothing)]
protected void SOOrderShipment_SiteID_CacheAttached(PXCache sender) { }
```

Orden de aplicación de los atributos de personalización de atributos

1. `PXCustomizeBaseAttribute`
2. `PXRemoveBaseAttribute`
3. `PXMergeAttributes`

Lea Mecanismos de personalización en línea:

<https://riptutorial.com/es/acumatica/topic/9751/mecanismos-de-personalizacion>

Capítulo 16: Modificaciones a la información de contacto y dirección a través del código

Introducción

En este tema, aprenderá cómo modificar la información de Contacto y Dirección a través del código en diferentes pantallas dentro de Acumatica.

Examples

Especifique la información de contacto y dirección para un nuevo empleado

Para especificar la información de contacto y dirección de un empleado, siempre debe invocar el método `Select()` en las vistas de datos de **contacto** y **dirección** antes de asignar cualquier valor de campo. También se recomienda asignar el resultado del método `Select()` a la propiedad **actual de** las vistas de Datos de **Contacto** y **Dirección** para garantizar que su código modifique el registro actual en **Contacto** y **Dirección** PXCaché respectivamente.

```
EmployeeMaint employeeMaintGraph = PXGraph.CreateInstance<EmployeeMaint>();
EPEmployee epEmployeeRow = new EPEmployee();
epEmployeeRow.AcctCD = "EMPLOYEE1";
epEmployeeRow = employeeMaintGraph.Employee.Insert(epEmployeeRow);

Contact contactRow = employeeMaintGraph.Contact.Current = employeeMaintGraph.Contact.Select();
contactRow.FirstName = "John";
contactRow.LastName = "Green";
employeeMaintGraph.Contact.Update(contactRow);

Address addressRow = employeeMaintGraph.Address.Current = employeeMaintGraph.Address.Select();
addressRow.CountryID = "US";
addressRow = employeeMaintGraph.Address.Update(addressRow);
addressRow.State = "DC";
employeeMaintGraph.Address.Update(addressRow);

epEmployeeRow.VendorClassID = "EMPSTAND";
epEmployeeRow.DepartmentID = "FINANCE";
employeeMaintGraph.Employee.Update(epEmployeeRow);

employeeMaintGraph.Actions.PressSave();
```

Al insertar un nuevo empleado, `employeeMaintGraph.Contact.Current` siempre devolverá el registro de contacto principal a medida que el registro de contacto se inserta automáticamente en el caché y, por lo tanto, queda disponible a través de la propiedad **Actual** de PXCaché / Data View. El uso del método `Select()` es un poco más genérico, ya que funcionará en todos los escenarios posibles, ya sea que necesite insertar un nuevo empleado o actualizar uno existente.

Anular información de contacto de facturación y de dirección de facturación para un cliente

Cuando necesita anular la información del contacto de **facturación** y la dirección de **facturación** de un cliente, el primer paso es establecer los valores correctos para las propiedades **IsBillContSameAsMain** e **IsBillSameAsMain** del DAC del **cliente** . No olvide invocar el método `Update()` en la memoria caché del **cliente** inmediatamente después de actualizar la propiedad **IsBillContSameAsMain** o **IsBillSameAsMain** para confirmar el valor actual **del** campo **Igual que la principal** en la memoria caché.

El siguiente paso es invocar el método `Select()` en las vistas de datos **BillContact** y **BillAddress** antes de asignar cualquier valor de campo. También se recomienda asignar el resultado del método `Select()` a la propiedad **actual de** las vistas de datos **BillContact** y **BillAddress** para garantizar que su código modifique el registro actual en **Contacto** y **Dirección** PXCaché respectivamente.

```
public class CustomerMaintExt : PXGraphExtension<CustomerMaint>
{
    public PXAction<Customer> UpdateBillingAddress;
    [PXButton(CommitChanges = true)]
    [PXUIField(DisplayName = "Update Bill-To Info")]
    protected void updateBillingAddress()
    {
        Customer currentCustomer = Base.BAccount.Current;

        if (currentCustomer.IsBillContSameAsMain != true)
        {
            currentCustomer.IsBillContSameAsMain = true;
            Base.BAccount.Update(currentCustomer);
        }
        else
        {
            currentCustomer.IsBillContSameAsMain = false;
            Base.BAccount.Update(currentCustomer);
        }

        Contact billContact = Base.BillContact.Current = Base.BillContact.Select();
        billContact.FullName = "ABC Holdings Inc";
        billContact.Phone1 = "+1 (212) 532-9574";
        Base.BillContact.Update(billContact);
    }

    if (currentCustomer.IsBillSameAsMain != true)
    {
        currentCustomer.IsBillSameAsMain = true;
        Base.CurrentCustomer.Update(currentCustomer);
    }
    else
    {
        currentCustomer.IsBillSameAsMain = false;
        Base.CurrentCustomer.Update(currentCustomer);
    }

    Address billAddress = Base.BillAddress.Current = Base.BillAddress.Select();
    billAddress.AddressLine1 = "65 Broadway";
    billAddress.AddressLine2 = "Office Suite 187";
    billAddress.City = "New York";
    billAddress.CountryID = "US";
    billAddress = Base.BillAddress.Update(billAddress);
    billAddress.State = "NY";
    billAddress.PostalCode = "10004";
    Base.BillAddress.Update(billAddress);
}
```

```

    }

    Base.Actions.PressSave();
}
}

```

Anular información de contacto de facturación y de dirección de facturación para una orden de venta

Para especificar la información del contacto de **facturación** y la dirección de **facturación** de un pedido de venta, siempre debe invocar primero el método `Select()` en las vistas de datos **Billing_Contact** y **Billing_Address** antes de asignar cualquier valor de campo. También se recomienda asignar el resultado del método `Select()` a la propiedad **actual de** las vistas de datos **Billing_Contact** y **Billing_Address** para garantizar que su código modifique el registro actual en **SOBillingContact** y **SOBillingAddress** PXCACHE respectivamente.

Cuando necesite anular la información de contacto y dirección de facturación para una orden de venta, establezca los valores correctos para las propiedades **OverrideContact** y **OverrideAddress** en el DAC de **SOBillingContact** y el DAC de **SOBillingAddress**. No olvide invocar el método `Update()` en los cachés **SOBillingContact** y **SOBillingAddress** justo después de actualizar las propiedades **OverrideContact** y **OverrideAddress** para confirmar los cambios. Una vez hecho esto, puede continuar y especificar la información de contacto y dirección de facturación para un pedido de venta.

```

public class SOOrderEntryExt : PXGraphExtension<SOOrderEntry>
{
    public PXAction<SOOrder> UpdateBillingAddress;
    [PXButton(CommitChanges = true)]
    [PXUIField(DisplayName = "Update Bill-To Info")]
    protected void updateBillingAddress()
    {
        SOBillingContact contact = Base.Billing_Contact.Current =
Base.Billing_Contact.Select();
        if (contact.OverrideContact == true)
        {
            contact.OverrideContact = false;
            Base.Billing_Contact.Update(contact);
        }
        else
        {
            contact.OverrideContact = true;
            contact = Base.Billing_Contact.Update(contact);
            if (contact == null)
            {
                contact = Base.Billing_Contact.Current;
            }

            contact.Phone1 = "+1 (908) 643-0281";
            contact.Email = "sales@usabartend.com";
            Base.Billing_Contact.Update(contact);
        }

        SOBillingAddress address = Base.Billing_Address.Current =
Base.Billing_Address.Select();
        if (address.OverrideAddress == true)
    }
}

```

```

    {
        address.OverrideAddress = false;
        Base.Billing_Address.Update(address);
    }
    else
    {
        address.OverrideAddress = true;
        address = Base.Billing_Address.Update(address);
        if (address == null)
        {
            address = Base.Billing_Address.Current;
        }

        address.AddressLine1 = "201 Lower Notch Rd";
        address.AddressLine2 = "Office Suite 1936";
        address.City = "Little Falls";
        address.CountryID = "US";
        address = Base.Billing_Address.Update(address);
        address.State = "NJ";
        address.PostalCode = "07425";
        Base.Billing_Address.Update(address);
    }

    Base.Actions.PressSave();
}
}

```

Lea Modificaciones a la información de contacto y dirección a través del código en línea:
<https://riptutorial.com/es/acumatica/topic/10617/modificaciones-a-la-informacion-de-contacto-y-direccion-a-traves-del-codigo>

Capítulo 17: Modificaciones a las vistas de datos base

Introducción

Este tema pretende mostrar varios patrones y prácticas disponibles para modificar las vistas de datos básicos en Acumatica.

Examples

APInvoiceEntry BLC: agregue restricciones adicionales a la vista de datos poReceiptLinesSelection

Para agregar una restricción adicional a la vista de datos **poReceiptLinesSelection**, debe invocar el método `Select` en la vista base e inspeccionar cada elemento en el `PXResultSet` devuelto independientemente para decidir si cumple con las restricciones adicionales:

```
public class APInvoiceEntryExt : PXGraphExtension<APInvoiceEntry>
{
    [PXCOPYPASTE_HIDDEN_VIEW]
    public PXSelect<APInvoiceEntry.PORceiptLineAdd> poReceiptLinesSelection;

    public virtual IEnumerable PORceiptLinesSelection()
    {
        foreach (var record in Base.poReceiptLinesSelection.Select())
        {
            // Additional restriction goes here
            if (true == true)
            {
                yield return record;
            }
        }
    }
}
```

Este enfoque funciona perfectamente con la vista de datos **poReceiptLinesSelection**, debido a la falta de paginación y agregación en la implementación de la vista base. Para componer el conjunto de resultados, **poReceiptLinesSelection** solicita los datos necesarios de la base de datos y realiza todos los cálculos en el lado de la aplicación.

```
public class APInvoiceEntry : APDataEntryGraph<APInvoiceEntry, APInvoice>,
    PXImportAttribute.IPXPrepareItems
{
    ...

    [PXCOPYPASTE_HIDDEN_VIEW]
    public PXSelect<PORceiptLineAdd> poReceiptLinesSelection;

    public virtual IEnumerable PORceiptLinesSelection()
    {
        ...
    }
}
```

```

{
    APInvoice doc = this.Document.Current;
    if (doc == null || doc.VendorID == null || doc.VendorLocationID == null) yield break;
    if (doc.DocType != APDocType.Invoice && doc.DocType != APDocType.DebitAdj)
        yield break;

    string poReceiptType = (doc.DocType == APDocType.Invoice) ? POReceiptType.POREceipt :
    POReceiptType.POReturn;

    HashSet<APTran> usedReceiptLine = new HashSet<APTran>(new POReceiptLineComparer());
    HashSet<APTran> unusedReceiptLine = new HashSet<APTran>(new POReceiptLineComparer());

    foreach (APTran aPTran in Transactions.Cache.Inserted)
    {
        if (aPTran.ReceiptNbr != null && aPTran.ReceiptType != null &&
aPTran.ReceiptLineNbr != null)
            usedReceiptLine.Add(aPTran);
    }

    foreach (APTran aPTran in Transactions.Cache.Deleted)
    {
        if (aPTran.ReceiptNbr != null && aPTran.ReceiptType != null &&
aPTran.ReceiptLineNbr != null && Transactions.Cache.GetStatus(aPTran) !=
PXEntryStatus.InsertedDeleted)
            if (!usedReceiptLine.Remove(aPTran))
                unusedReceiptLine.Add(aPTran);
    }

    foreach (APTran aPTran in Transactions.Cache.Updated)
    {
        APTran originAPTran = new APTran();
        originAPTran.ReceiptNbr =
(String)Transactions.Cache.GetValueOriginal<APTran.receiptNbr>(aPTran);
        originAPTran.ReceiptType =
(String)Transactions.Cache.GetValueOriginal<APTran.receiptType>(aPTran);
        originAPTran.ReceiptLineNbr =
(Int32?)Transactions.Cache.GetValueOriginal<APTran.receiptLineNbr>(aPTran);

        if (originAPTran.ReceiptNbr != null && originAPTran.ReceiptType != null &&
originAPTran.ReceiptLineNbr != null)
        {
            if (!usedReceiptLine.Remove(originAPTran))
                unusedReceiptLine.Add(originAPTran);
        }

        if (aPTran.ReceiptNbr != null && aPTran.ReceiptType != null &&
aPTran.ReceiptLineNbr != null)
        {
            if (!unusedReceiptLine.Remove(aPTran))
                usedReceiptLine.Add(aPTran);
        }
    }

    foreach (LinkLineReceipt item in PXSelect<LinkLineReceipt,
Where<LinkLineReceipt.vendorID, Equal<Current<APInvoice.vendorID>>,
And<LinkLineReceipt.vendorLocationID, Equal<Current<APInvoice.vendorLocationID>>,
And<LinkLineReceipt.receiptCuryID, Equal<Current<APInvoice.curyID>>,
And<LinkLineReceipt.receiptType, Equal<Required<POReceipt.receiptType>>,
And<Where<LinkLineReceipt.orderNbr, Equal<Current<POReceiptFilter.orderNbr>>,
Or<Current<POReceiptFilter.orderNbr>, IsNull>>>>
>>>>.SelectMultiBound(this, new object[] { doc }, poReceiptType)

```

```

    {
        APTran aPTran = new APTran();
        aPTran.ReceiptType = item.ReceiptType;
        aPTran.ReceiptNbr = item.ReceiptNbr;
        aPTran.ReceiptLineNbr = item.ReceiptLineNbr;
        if (!usedReceiptLine.Contains(aPTran))
            yield return (PXResult<POReceiptLineAdd,
POReceipt>)ReceiptLineAdd.Select(item.ReceiptNbr, item.ReceiptType, item.ReceiptLineNbr);
    }

    foreach (APTran item in unusedReceiptLine)
    {
        yield return (PXResult<POReceiptLineAdd,
POReceipt>)ReceiptLineAdd.Select(item.ReceiptNbr, item.ReceiptType, item.ReceiptLineNbr);
    }
}

...
}

```

Lea Modificaciones a las vistas de datos base en línea:

<https://riptutorial.com/es/acumatica/topic/9101/modificaciones-a-las-vistas-de-datos-base>

Capítulo 18: Modificar elementos en una lista desplegable

Introducción

En este tema, aprenderá cómo modificar los atributos de campo heredados de los atributos PXStringList o PXIntList. El enfoque demostrado se asegurará de no romper la funcionalidad del producto base Acumatica ERP y requerirá un mantenimiento mínimo, en caso de que exista, mientras se actualizan sus personalizaciones a una versión más nueva de Acumatica.

Observaciones





En todos los ejemplos anteriores, realizó cambios en las matrices `_AllowedValues` y `_AllowedLabels`. Si su tarea es modificar solo la etiqueta (valor externo) de un elemento desplegable, considere usar los diccionarios de traducción. Para obtener más información sobre los diccionarios de traducción, consulte la [documentación de Acumatica ERP](#).


Examples

Modificación de los estados maritales

En este ejemplo, modificará la lista desplegable **Estado civil que se** encuentra en el formulario de **Contactos** (CR302000):

Revision Two HQ ▾ Contacts ★ My Contacts x


← SAVE & CLOSE   +   ▾ | < < > > | ACTIONS ▾

Contact ID:  Workgroup:
 Type: Owner:
 Active

Details Additional Info Attributes Activities Relations Opportunities Cases Campaigns Marketing Lists Notifications

COMMON

Gender:

Marital Status: 

Spouse/Partner Name:

LEAD HISTORY

Source:

Campaign ID:

Status:

Reason:

Converted By:


Qualification Date:

SYNCHRONIZATION

Synchronize

PHOTO

Select an image to upload.



Drag and drop the image here

Para agregar nuevos elementos al sucesor de PXStringListAttribute

La mejor manera de extender los elementos desplegados codificados dentro de un atributo heredado de los atributos PXStringList o PXIntList es aumentar el tamaño de las matrices `_AllowedValues` y `_AllowedLabels` en el constructor de su atributo de campo personalizado:

```
[PXLocalizable(Messages.Prefix)]
public static class MaritalStatusesMessages
{
    public const string CommonLaw = "Living common law";
    public const string Separated = "Separated (not living common law)";
    public const string DivorcedNoCommonLaw = "Divorced (not living common law)";
    public const string NeverMarried = "Never Married";
}

public class MaritalStatusesCst1Attribute : MaritalStatusesAttribute
{
    public const string CommonLaw = "L";
}
```



```

public const string Separated = "P";
public const string NeverMarried = "N";

public MaritalStatusesCst1Attribute()
{
    Array.Resize(ref _AllowedValues, _AllowedValues.Length + 3);
    _AllowedValues[_AllowedValues.Length - 3] = CommonLaw;
    _AllowedValues[_AllowedValues.Length - 2] = Separated;
    _AllowedValues[_AllowedValues.Length - 1] = NeverMarried;
    Array.Resize(ref _AllowedLabels, _AllowedLabels.Length + 3);
    _AllowedLabels[_AllowedLabels.Length - 3] = MaritalStatusesMessages.CommonLaw;
    _AllowedLabels[_AllowedLabels.Length - 2] = MaritalStatusesMessages.Separated;
    _AllowedLabels[_AllowedLabels.Length - 1] = MaritalStatusesMessages.NeverMarried;
}
}

```

En el ejemplo anterior, aumentó el tamaño de las matrices `_AllowedValues` y `_AllowedLabels` para agregar 3 elementos adicionales a la lista desplegable **Estado civil**. Los valores internos, almacenados en la matriz `_AllowedValues`, se asignarán a los campos DAC y se guardarán en la base de datos, y los valores externos de la matriz `_AllowedLabels` representan valores internos en la interfaz de usuario.

Para probar los resultados, en la extensión Contact DAC, decora el campo **MaritalStatus** con el `MaritalStatusesCst1Attribute`:

```

public class ContactExt : PXCacheExtension<Contact>
{
    [PXRemoveBaseAttribute(typeof(MaritalStatusesAttribute))]
    [PXMergeAttributes(Method = MergeMethod.Append)]
    [MaritalStatusesCst1]
    public string MaritalStatus { get; set; }
}

```

Ahora hay 7 elementos en la lista desplegable **Estado civil**:

Revision Two HQ - Contacts ★

← SAVE & CLOSE [Icons]

Contact ID: Air, Jane [Search]

Type: Contact

Active

Details Additional Info Attributes Activities Relations Opportunities

COMMON

Gender: [Dropdown]

Marital Status: [Dropdown]

Spouse/Partner Name: [Text]

LEAD HISTORY

Source: [Text]

Campaign ID: [Text]

Status: [Text]

Reason: [Text]

Single
Married
Divorced
Widowed
Living common law
Separated (not living common law)
Never Married

Para eliminar elementos declarados en el sucesor de PXStringListAttribute

Para eliminar un elemento desplegable específico, que fue codificado dentro de un atributo heredado de los atributos PXStringList o PXIntList, debe reducir el tamaño de las matrices `_AllowedValues` y `_AllowedLabels` en el constructor de su atributo de campo personalizado:

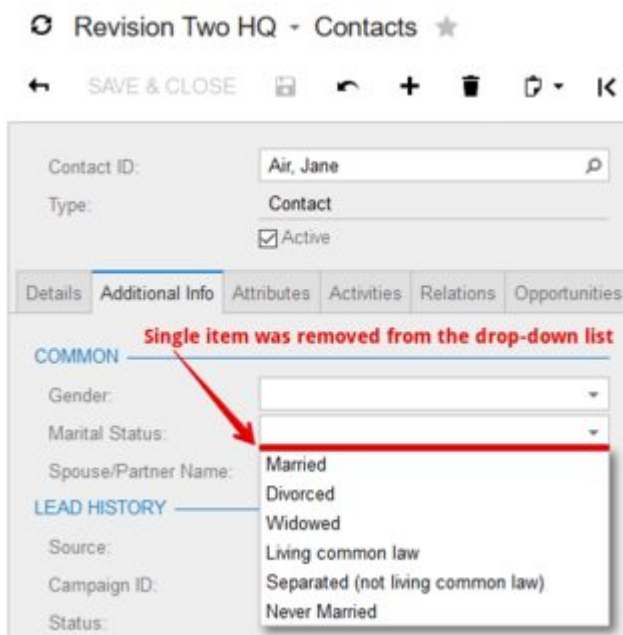
```
public class MaritalStatusesCst2Attribute : MaritalStatusesCst1Attribute
{
    public MaritalStatusesCst2Attribute()
    {
        string[] allowedValues = new string[_AllowedValues.Length - 1];
        string[] allowedLabels = new string[_AllowedLabels.Length - 1];
        Array.Copy(_AllowedValues, 1, allowedValues, 0, _AllowedValues.Length - 1);
        Array.Copy(_AllowedLabels, 1, allowedLabels, 0, _AllowedLabels.Length - 1);
        _AllowedValues = allowedValues;
        _AllowedLabels = allowedLabels;
    }
}
```

En el ejemplo anterior, redujo el tamaño de las matrices `_AllowedValues` y `_AllowedLabels` para eliminar **un** elemento **individual** de la lista desplegable **Estado civil**.

Para probar los resultados, en la extensión Contact DAC, decora el campo **MaritalStatus** con el `MaritalStatusesCst2Attribute`:

```
public class ContactExt : PXCacheExtension<Contact>
{
    [PXRemoveBaseAttribute(typeof(MaritalStatusesAttribute))]
    [PXMergeAttributes(Method = MergeMethod.Append)]
    [MaritalStatusesCst2]
    public string MaritalStatus { get; set; }
}
```

Ahora solo hay 6 elementos: 3 originales y 3 personalizados, en la lista desplegable **Estado civil**:



Para reemplazar los elementos declarados en el sucesor PXStringListAttribute

Para reemplazar un elemento desplegable específico, originalmente codificado dentro de un atributo heredado de los atributos PXStringList o PXIntList, debe actualizar el valor apropiado en las matrices `_AllowedValues` y `_AllowedLabels` en el constructor de su atributo de campo personalizado:

```
public class MaritalStatusesCst3Attribute : MaritalStatusesCst2Attribute
{
    public const string DivorcedNoCommonLaw = "V";

    public MaritalStatusesCst3Attribute()
    {
        _AllowedValues[Array.IndexOf(_AllowedValues, Divorced)] = DivorcedNoCommonLaw;
        _AllowedLabels[Array.IndexOf(_AllowedLabels, Messages.Divorced)] =
        MaritalStatusesMessages.DivorcedNoCommonLaw;
    }
}
```

En el ejemplo anterior, reemplazó **D** - Artículo **divorciado** con **V - Divorciado (no tiene derecho común)** en las matrices `_AllowedValues` y `_AllowedLabels` respectivamente. Al hacerlo, reemplazamos los valores internos y externos de un elemento desplegable.

Para probar los resultados, en la extensión Contact DAC, decora el campo **MaritalStatus** con el `MaritalStatusesCst3Attribute`:

```
public class ContactExt : PXCacheExtension<Contact>
{
    [PXRemoveBaseAttribute(typeof(MaritalStatusesAttribute))]
    [PXMergeAttributes(Method = MergeMethod.Append)]
    [MaritalStatusesCst3]
    public string MaritalStatus { get; set; }
}
```

Ahora solo hay 6 elementos: 2 originales, 3 personalizados y 1 reemplazado, en la lista desplegable **Estado civil** :

Revision Two HQ - Contacts ★

SAVE & CLOSE

Contact ID: Air, Jane

Type: Contact

Active

Details Additional Info Attributes Activities Relations Opportunities

COMMON

Gender:

Marital Status:

Spouse/Partner Name: Married

LEAD HISTORY

Source:

Campaign ID:

Status:

- Married
- Divorced (not living common law)**
- Widowed
- Living common law
- Separated (not living common law)
- Never Married

Lea [Modificar elementos en una lista desplegable en línea](https://riptutorial.com/es/acumatica/topic/9392/modificar-elementos-en-una-lista-desplegable):

<https://riptutorial.com/es/acumatica/topic/9392/modificar-elementos-en-una-lista-desplegable>

Capítulo 19: Pestañas de ocultación condicional

Introducción

En este tema, explorará dos enfoques para ocultar de forma condicional las pestañas en las pantallas de ingreso de datos en Acumatica.

Examples

Propiedad VisibleExp del control PXTab en Aspx

La propiedad **VisibleExp** es una expresión booleana, que determina si la pestaña dada es visible (cuando la expresión lógica es VERDADERA) u oculta. Usted especifica la propiedad **VisibleExp** para los controles PXTab en la página Aspx:

```
<px:PXTabItem Text="Credit Card Processing Info" BindingContext="form"
  VisibleExp="DataControls[&quot;chkIsCCPayment&quot;].Value = 1">
```

VisibleExp se compone de controles de entrada ubicados dentro del contenedor con el ID especificado en la propiedad **BindingContext** del control PXTab. No está permitido usar controles de entrada de más de un contenedor. El acceso a un control de entrada específico se proporciona a través del diccionario `DataControls` por su ID, no por el nombre de un campo DAC.

Generalmente, la propiedad **VisibleExp** se usa para componer expresiones booleanas bastante simples con valores de control de entrada codificados, que es poco probable que cambien con el tiempo. Por ejemplo, la siguiente expresión se usa en la pantalla **Pedidos de venta** (SO.30.10.00) para ocultar la pestaña **Configuración de pago** para pedidos del tipo de **transferencia** :

```
<px:PXTabItem Text="Payment Settings"
  VisibleExp="DataControls[&quot;edOrderType&quot;].Value!=TR" BindingContext="form">
```

Para ocultar la pestaña Actividades para clientes potenciales con nuevo estado

Para ocultar la pestaña **Actividades** de la pantalla **Leads** (CR.30.10.00), configure la propiedad **BindingContext** para que se **forme** (el **formulario de resumen de leads** de nivel superior contiene la ID del **formulario**) y defina **VisibleExp** para que devuelva FALSE si el estado del lead está Abierto (la lista desplegable *Status* tiene la ID del *estado de ed*) :

```
<px:PXTabItem Text="Activities" LoadOnDemand="True"
  BindingContext="form" VisibleExp="DataControls[&quot;edStatus&quot;].Value != H">
```

Revision Two HQ ▾ Leads ★ New ×

← SAVE & CLOSE [Disk Icon] [Refresh Icon] + [Trash Icon] [Clipboard Icon] [Left Arrow] [Right Arrow] >| ACTIONS ▾

Lead ID: Boyd, Ellis [Search Icon] Workgroup: [Text Box]
* Status: New [Dropdown] Owner: [Text Box]
Reason: Assign [Dropdown]

Details [Selected] Attributes [] Relations [] Campaigns [] Marketing Lists []

Activities tab used to be here →

SUMMARY

First Name: [Dropdown] Ellis [Text Box]
* Last Name: Boyd [Text Box]
Position: Developer [Text Box]
Business Account: [Text Box] [Search Icon] [Edit Icon]
Company Name: Officemax North America Inc [Text Box]
Parent Business Accou... [Text Box] [Search Icon] [Edit Icon]

CONTACT

Email: eu.tempor.erat@velitPellentesqueultric [Text Box] [Envelope Icon]
Web: [Text Box] [Globe Icon]
Phone 1: Business 1 [Dropdown] [Text Box]
Phone 2: Business 2 [Dropdown] [Text Box]
Phone 3: Home [Dropdown] [Text Box]
Fax: Business Fa [Dropdown] [Text Box]

CRM

Lead Class: LEADBUS - Sales [Text Box]
Source: Web [Text Box]
Campaign ID: [Text Box]
Contact Method: Any [Text Box]
 Do Not Call
 Do Not Email
 No Mass Mail

Last Incoming Activity: [Text Box]
Last Outgoing Activity: [Text Box]

ADDRESS

Same As In Acco... [Text Box]
Address Line 1: [Text Box]
Address Line 2: [Text Box]
City: Kansas City [Text Box]
* Country: US - UNITED STAT [Text Box]
State: MO - MISSOURI [Text Box]
Postal Code: 95216 [Text Box]

AllowSelect Property on Data Views

A diferencia de la propiedad **VisibleExp** , definida en Aspx, usted manipula la propiedad **AllowSelect** de una vista de datos a través del código de extensión BLC o BLC. La propiedad **AllowSelect** permite utilizar expresiones booleanas más complejas (en comparación con la propiedad **VisibleExp**) y, si es necesario, recuperar información adicional de la base de datos u otras fuentes no disponibles en una página web.

A continuación se muestran los 3 escenarios más comunes para trabajar con la propiedad **AllowSelect** :

- **Controlador de eventos RowSelected** para entidades de nivel superior para ocultar la pestaña **Aplicaciones** para facturas de tipos de **Venta en efectivo** y **Devolución de efectivo** :

```
public class SOInvoiceEntry : ARInvoiceEntry
{
    ...
    protected override void ARInvoice_RowSelected(PXCache cache, PXRowSelectedEventArgs e)
    {
        ...

        Adjustments.AllowSelect =
            doc.DocType != ARDocType.CashSale &&
            doc.DocType != ARDocType.CashReturn;
    }
    ...
}
```

- El constructor de BLC muestra la pestaña **Información de reabastecimiento de subelementos** en la pantalla **Detalles del almacén de artículos** solo cuando están activadas las funciones de *Reposición de inventario* y *Subtemas de inventario* :

```
public class INItemSiteMaint : PXGraph<INItemSiteMaint, INItemSite>
{
    ...
    public INItemSiteMaint()
    {
        ...

        bool enableSubItemReplenishment =
            PXAccess.FeatureInstalled<FeaturesSet.replenishment>() &&
            PXAccess.FeatureInstalled<FeaturesSet.subItem>();
        subitemrecords.AllowSelect = enableSubItemReplenishment;
    }
    ...
}
```

- **Controlador RowSelected** para que la entidad de nivel superior oculte la pestaña **Historial de depreciación**, a menos que el activo actual sea depreciable y la **Vista del historial de depreciación** esté configurada en **Lado a lado** en las Preferencias de Activos fijos:

```
public class AssetMaint : PXGraph<AssetMaint, FixedAsset>
{
    ...
    protected virtual void FixedAsset_RowSelected(PXCache sender, PXRowSelectedEventArgs e)
    {
        ...

        AssetHistory.AllowSelect = asset.Depreciable == true &&
            fasetup.Current.DeprHistoryView == FASetup.deprHistoryView.SideBySide;
    }
}
```

```
}  
...  
}
```

Cada vez que se **utiliza la** propiedad **AllowSelect** para cambiar condicionalmente la visibilidad de la pestaña a través del código de extensión BLC o BLC, debe establecer la propiedad **RepaintOnDemand** en **false** en Aspx para el contenedor PXTab correspondiente:

```
<px:PXTabItem Text="Depreciation History" RepaintOnDemand="false">
```

La propiedad **RepaintOnDemand** es **verdadera** de forma predeterminada. Esta propiedad controla la inicialización del contenedor PXTab: cuando se establece en **verdadero**, PXTab no se inicializará hasta que sea seleccionado por un usuario. Obviamente, necesita **RepaintOnDemand** configurado en **false** para garantizar el comportamiento correcto del contenedor PXTab dado a pesar de que haya sido seleccionado o no.

Para ocultar la pestaña Referencia cruzada de los artículos en stock que no se pueden vender

Para ocultar la pestaña **Referencia cruzada** de la pantalla **Artículos en inventario** (IN.20.25.00) para artículos con estado **Sin ventas**, proceda de la siguiente manera:

1. implementar el controlador **InventoryItem_RowSelected** en la extensión BLC **InventoryItemMaint** para establecer la propiedad **AllowSelect** en **false** para la vista de datos de `itemxrefrecords` si el **Estado del artículo** se configuró en **Sin ventas**:

```
public class InventoryItemMaintExt : PXGraphExtension<InventoryItemMaint>  
{  
    protected void InventoryItem_RowSelected(PXCache sender, PXRowSelectedEventArgs e)  
    {  
        InventoryItem item = (InventoryItem)e.Row;  
        if (item == null) return;  
  
        Base.itemxrefrecords.AllowSelect = (item.ItemStatus !=  
InventoryItemStatus.NoSales);  
    }  
}
```

2. en el Administrador de personalización, establezca la propiedad **RepaintOnDemand** en **false** para la pestaña **Referencia cruzada** y publique la personalización:

Layout Editor: IN202500 (Stock Items)

PREVIEW CHANGES ACTIONS

The screenshot shows the Layout Editor interface for 'IN202500 (Stock Items)'. On the left, a tree view displays the layout structure, with 'Cross-Reference' under 'Tab: ItemSettings' highlighted with a red box. On the right, the Properties window is open, showing a table of properties for the selected 'Cross-Reference' tab. The 'RepaintOnDemand' property is checked, also highlighted with a red box.

Override	Property
<input type="checkbox"/>	Base Properties
<input type="checkbox"/>	BindingContext
<input type="checkbox"/>	ContentLayout
<input type="checkbox"/>	LoadOnDemand
<input checked="" type="checkbox"/>	RepaintOnDemand
<input type="checkbox"/>	Text
<input type="checkbox"/>	Visible
<input type="checkbox"/>	VisibleExp
<input type="checkbox"/>	Ext Properties
<input type="checkbox"/>	AutoCallBack

Indicates whether the tab item repaints its content from the s

Después de completar los 2 pasos bastante simples mencionados anteriormente, la pestaña **Referencia cruzada** no debe estar disponible para los artículos en existencia *sin* estado de **ventas** :

Revision Two HQ ▾ Stock Items ★

ACTIONS ▾ INQUIRIES ▾

* Inventory ID: AACOMPUT01
 Item Status: No Sales ▾
 Description: Acer Laptop Computer

Product Workgroup:
 Product Manager:

General Settings | Price/Cost Info | Warehouse Details | Vendor Details | Attributes | Packaging | Replenishment Info | Deferr

ITEM DEFAULTS

Item Class: ELECCOMP - Electronics & Compute
 Type: Finished Good
 Is a Kit
 Valuation Method: Average
 * Tax Category: TAXABLE - Taxable Goods and Servic
 * Posting Class: ELE - Electronics & Computers
 * Lot/Serial Class: NOTTRACKED - Not Tracked
 Auto-Incremental Value:

WAREHOUSE DEFAULTS

Default Warehouse: WHOLESALE - HQ Wholesale Wareh
 Default Issue From: R1S1 - Row 1 Shelf 1
 Default Receipt To: RECEIVING - Receiving

UNIT OF MEASURE

* Base Unit: EA
 * Sales Unit: EA
 * Purchase Unit: EA

C + X

* From Unit	Multiply/Divide	Co

PHYSICAL INVENTORY

PI Cycle:
 ABC Code:
 Fixed ABC Code
 Movement Class:
 Fixed Movement

Para ocultar la pestaña Atributos de los artículos en stock inactivos

Para ocultar de forma condicional la pestaña ** Atributos ** de la pantalla **Artículos en stock** (IN.20.25.00), proceda de la siguiente manera:

1. implementar el controlador **InventoryItem_RowSelected** en la extensión BLC de **InventoryItemMaint** para establecer la propiedad **AllowSelect** en **false** para las vistas de datos de **Answers** y **Category** si el **Estado del elemento** se configuró como **Inactivo**. También observe la propiedad **Visible** establecida en **false** para **PXUIFieldAttribute**

agregado en el campo `InventoryItem.ImageUrl` por el controlador **CacheAttached** :

```
public class InventoryItemMaintExt : PXGraphExtension<InventoryItemMaint>
{
    protected void InventoryItem_RowSelected(PXCache sender, PXRowSelectedEventArgs e)
    {
        InventoryItem item = (InventoryItem)e.Row;
        if (item == null) return;

        bool showAttributesTab = item.ItemStatus != InventoryItemStatus.Inactive;
        Base.Answers.AllowSelect = Base.Category.AllowSelect = showAttributesTab;
        PXUIFieldAttribute.SetVisible<InventoryItem.imageUrl>(sender, item,
showAttributesTab);
    }

    [PXMergeAttributes(Method = MergeMethod.Append)]
    [PXUIField(DisplayName = "Image")]
    protected void InventoryItem_ImageURL_CacheAttached(PXCache sender)
    { }
}
```

2. en el Administrador de personalización, establezca la propiedad **RepaintOnDemand** en **false** para la pestaña **Atributos** y publique la personalización:

Layout Editor: IN202500 (Stock Items)

PREVIEW CHANGES ACTIONS

The screenshot shows the Layout Editor interface for 'IN202500 (Stock Items)'. On the left, a tree view displays the layout structure, with 'Attributes' highlighted in red. On the right, the 'Properties' tab is active, showing a table of properties. The 'RepaintOnDemand' property is checked and highlighted in red.

Override	Property
<input type="checkbox"/>	Base Properties
<input type="checkbox"/>	BindingContext
<input type="checkbox"/>	ContentLayout
<input type="checkbox"/>	LoadOnDemand
<input checked="" type="checkbox"/>	RepaintOnDemand
<input type="checkbox"/>	Text
<input type="checkbox"/>	Visible
<input type="checkbox"/>	VisibleExp
<input type="checkbox"/>	Ext Properties
<input type="checkbox"/>	AutoCallBack

Indicates whether the tab item repaints its content from the s

Después de completar los 2 pasos anteriores, la pestaña **Atributos** no debe estar accesible para los artículos en inventario con estado **inactivo** :

* Inventory ID: AALEGO500
Item Status: **Inactive**
Description: Lego 500 piece set

Product Workgroup:
Product Manager:
General Settings | Price/Cost Info | Warehouse Details | Vendor Details | **Packaging** | Cross-Reference | Replenishment Info

ITEM DEFAULTS

Item Class: CONSUMER - Consumer Goods
Type: Finished Good
 Is a Kit
Valuation Method: Average
* Tax Category: TAXABLE - Taxable Goods and Services
* Posting Class: CON - Consumer Goods
* Lot/Serial Class: NOTTRACKED - Not Tracked
Auto-Incremental Value:
WAREHOUSE DEFAULTS
Default Warehouse: WHOLESALE - HQ Wholesale Warehouse
Default Issue From: R1S1 - Row 1 Shelf 1
Default Receipt To: RECEIVING - Receiving

UNIT OF MEASURE

* Base Unit: EA
* Sales Unit: EA
* Purchase Unit: EA

* From Unit	Multiply/Divide	Co

PHYSICAL INVENTORY

PI Cycle:
ABC Code:
 Fixed ABC Code
Movement Class:
 Fixed Movement

Lea Pestañas de ocultación condicional en línea:
<https://riptutorial.com/es/acumatica/topic/9506/pestanas-de-ocultacion-condicional>

Capítulo 20: Publicación omitida de contenido de personalización ya aplicado.

Introducción

Al publicar un proyecto de personalización, es posible que vea algún elemento omitido por el motivo de que ya se ha aplicado. Ex:

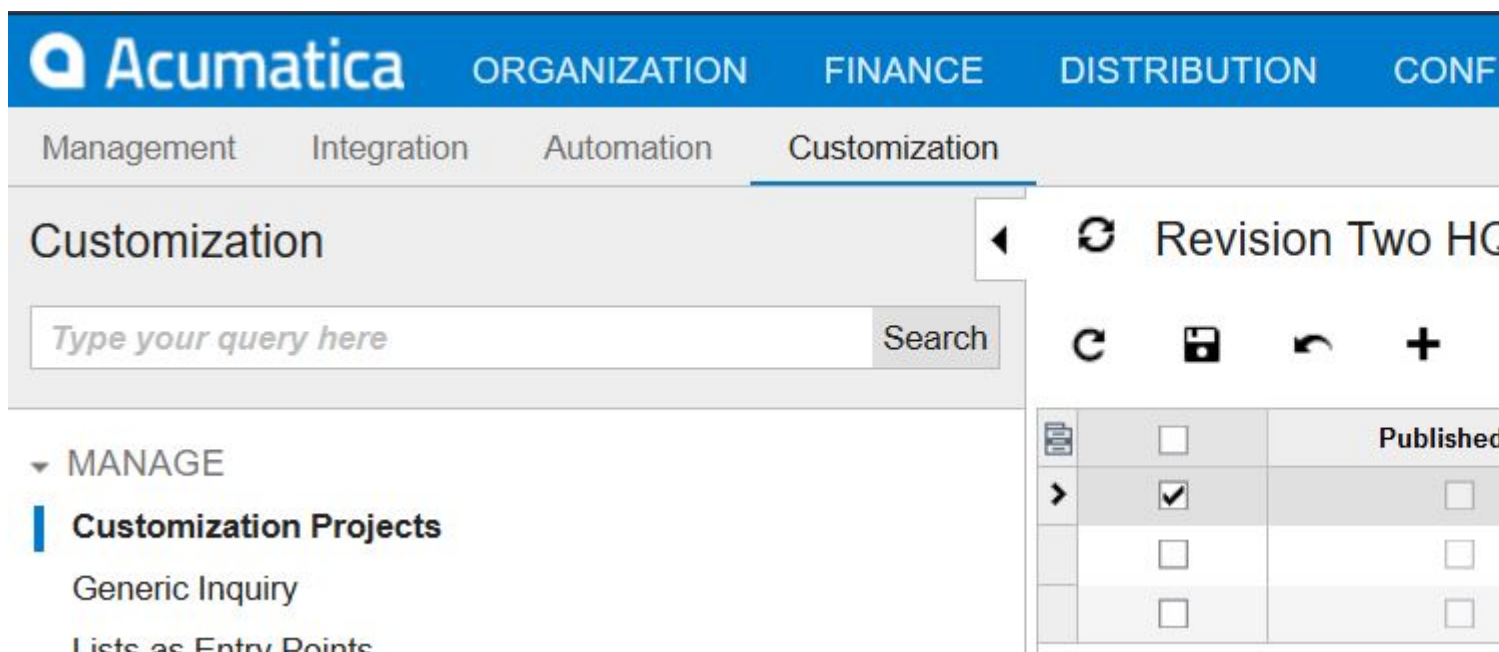
EntityEndpoint EntityEndpoint # 6.00.001\$DefaultPlus (omitido, ya aplicado)

Esto puede suceder con cualquier elemento contenido guardado en la base de datos. Ej .: consultas genéricas, informes, nodos de mapas del sitio, scripts de base de datos, configuraciones regionales del sistema, escenarios de importación / exportación, filtros compartidos, derechos de acceso, wikis, puntos finales de servicios web e informes analíticos.

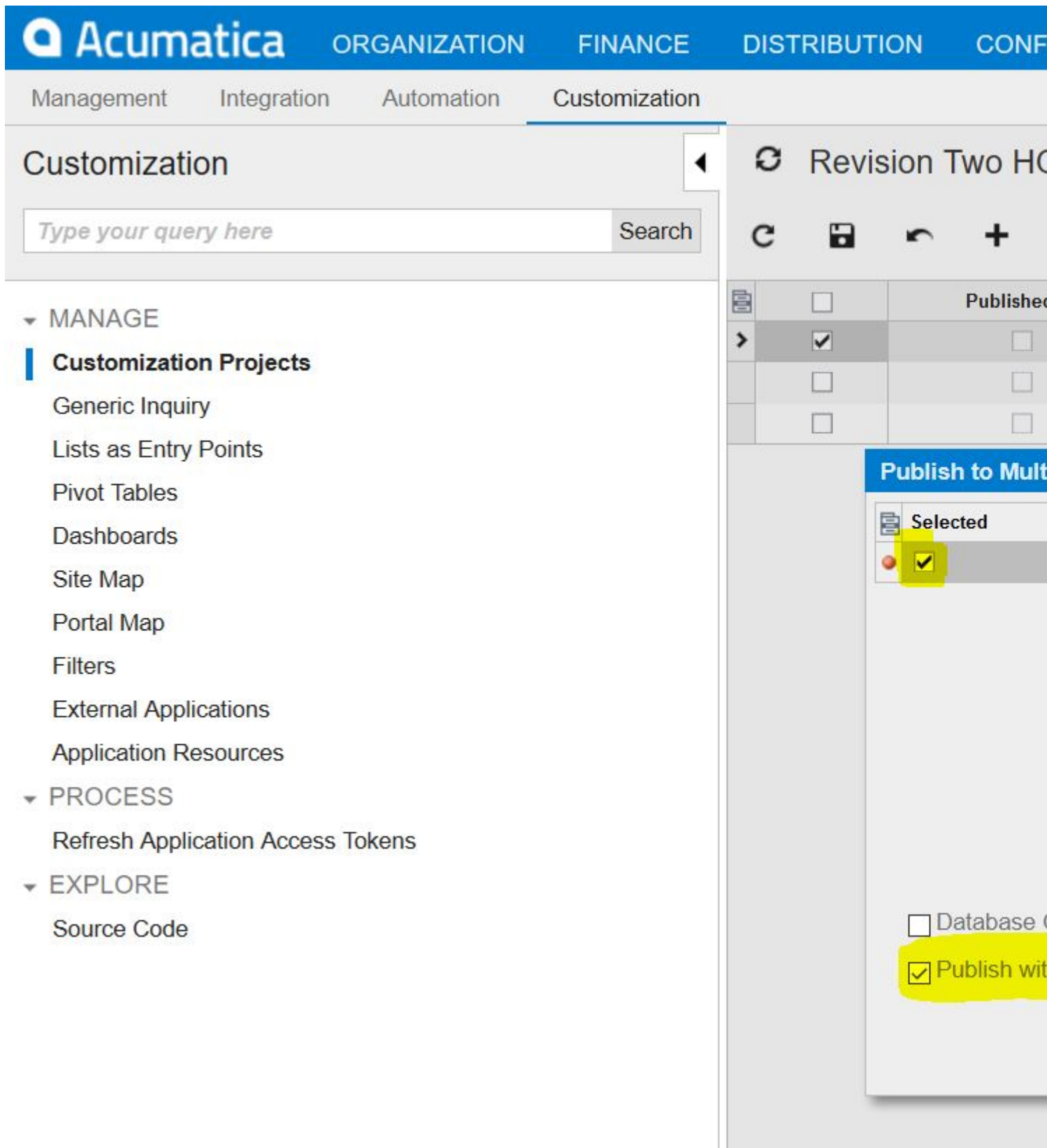
Examples

Publica con limpieza desde la pantalla de personalización.

1. Obviamente debes seleccionar el proyecto que quieres publicar.
2. Debe hacer clic en la flecha pequeña que se encuentra junto al botón "Publicar".
3. Debe hacer clic en la opción "Publicar en varias empresas".



4. En el panel inteligente que aparecerá, debe seleccionar las compañías en las que desea publicar los proyectos. Sólo una empresa es también una posibilidad.
5. Marque la casilla de verificación que indica "Publicar con limpieza", esto se asegurará de volver a aplicar todos los elementos presentes en el proyecto de personalización, reemplazando el ya existente con su versión más nueva.



Publicar con la limpieza desde dentro de un proyecto de personalización

1. Abra el proyecto de personalización que desea publicar con este método.
2. Abra el menú de publicación en la parte superior y seleccione la opción "Publicar con limpieza".

Custom

Publish Current Project (Ctrl+Space)

Validate Current Project

Multiple Projects

Publish with Cleanup

Screen

Data

Code

Files

Generic Inquiries (1)

Reports (1)

Site Map (1)

DB Scripts (1)

System Locales (1)

Import/Export Scenarios (1)

Shared Filters (1)

Access Rights (1)

Wikis (1)

Web Service Endpoints (1)

Analytical Reports (1)

Web Service Endpoints



Object Name

> 6.00.001&DefaultPlus

* Tenga en cuenta que todos los proyectos de personalización que se seleccionan en la pantalla de personalización se volverán a publicar incluso si está dentro de un solo proyecto.

Lea [Publicación omitida de contenido de personalización ya aplicado](https://riptutorial.com/es/acumatica/topic/10155/publicacion-omitida-de-contenido-de-personalizacion-ya-aplicado-). en línea:
<https://riptutorial.com/es/acumatica/topic/10155/publicacion-omitida-de-contenido-de-personalizacion-ya-aplicado->

Capítulo 21: Reemplazo de imágenes en la página de inicio de sesión

Introducción

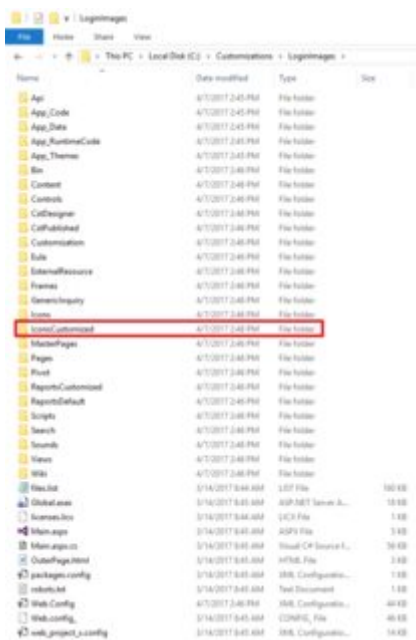
En este tema, aprenderá cómo reemplazar las imágenes estándar de Acumatica en la página de inicio de sesión. El enfoque demostrado se asegurará de mantener sus imágenes personalizadas en la página de inicio de sesión después de la actualización a una versión más reciente y restaurar las imágenes originales, proporcionadas por Acumatica, si en algún momento su personalización aparece sin publicar.

Examples

Uso de personalización para reemplazar imágenes en la página de inicio de sesión

Para crear un paquete de personalización que reemplace las imágenes en la página de inicio de sesión, siga los pasos a continuación en su **instancia local de Acumatica** :

1. Crea una nueva carpeta en la carpeta de instancias de Acumatica. Para este ejemplo, agregué una carpeta llamada **IconsCustomized** en mi instancia de **LoginImages** local:

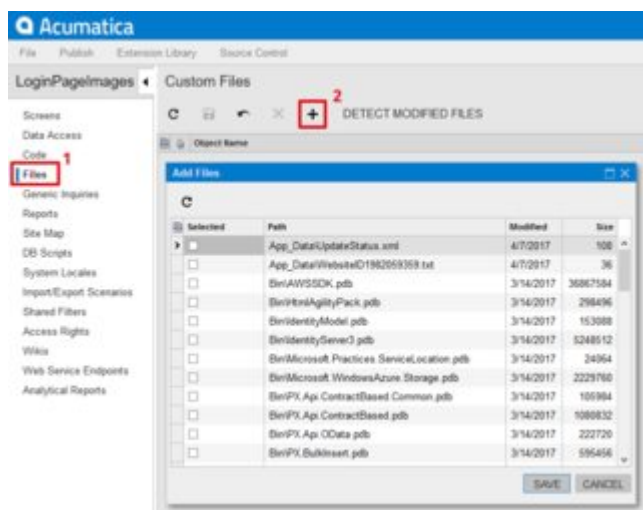


2. Añade tus imágenes personalizadas en esta carpeta. Por el bien de este ejemplo, usé imágenes de la página de inicio de sesión de Acumatica 4.2:

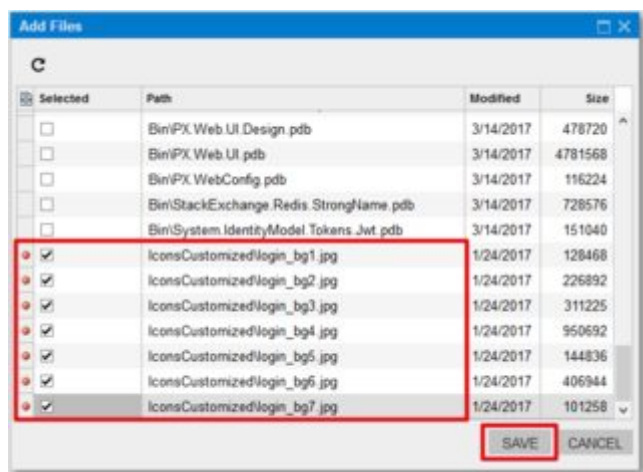


Tenga en cuenta que, **para reemplazar todas las imágenes en la página de inicio de sesión**, debe agregar al menos tantas imágenes personalizadas en su carpeta **IconsCustomized** como el número de archivos `login_bg*.*` Originalmente presentes en la carpeta **Icons** de su sitio web de Acumatica. Es perfectamente correcto usar la misma imagen o imágenes varias veces (al nombrar los archivos de manera diferente), si el número de sus imágenes personalizadas es menor que lo que originalmente proporcionó Acumatica.

- Ahora inicie sesión en su aplicación Acumatica, cree un nuevo proyecto de personalización llamado **LoginPagelImages** y ábralo en el Administrador de personalización.
- En el Administrador de personalización, navegue a la sección **Archivos** y haga clic en el botón **Agregar nuevo registro** para abrir el cuadro de diálogo **Agregar archivos** :



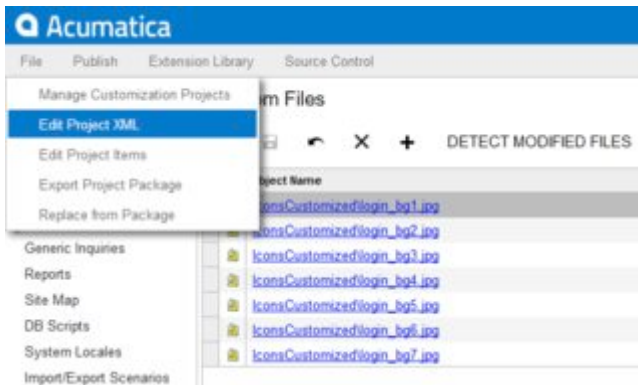
- En el cuadro de diálogo Agregar archivos, seleccione todos los archivos de su carpeta **IconsCustomized** y haga clic en **Guardar** :



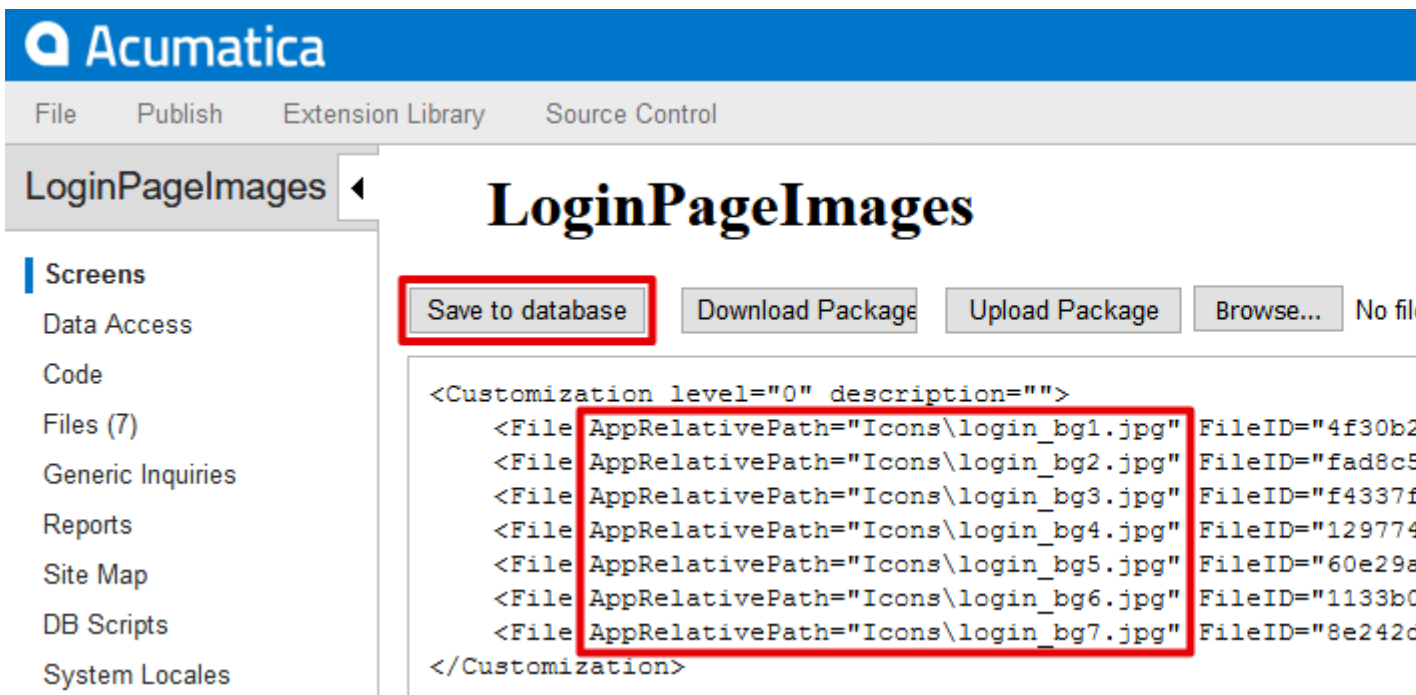
Ahora tiene las imágenes de la página de inicio de sesión personalizadas en el proyecto de personalización, pero aún necesita editar la ruta para que reemplacen

correctamente las imágenes estándar.

6. En el Administrador de personalización, seleccione **Editar proyecto XML** en el menú **Archivo** :

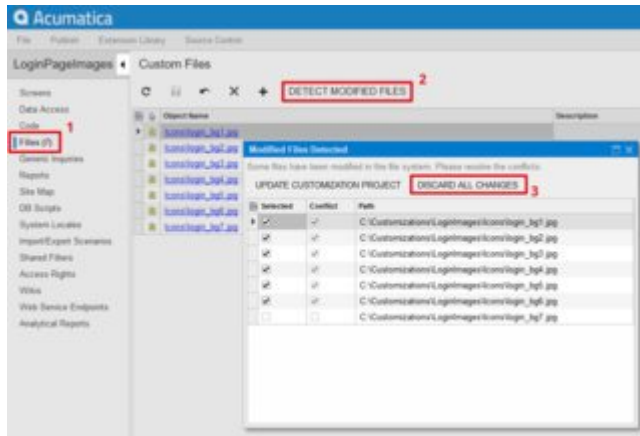


7. Para todas las etiquetas de **archivo** , generadas para sus imágenes personalizadas, cargue el atributo **AppRelativePath** a **AppRelativePath = "Iconos ..."** y configure el atributo **SystemFile** en **Verdadero** para esas imágenes, que actualmente se encuentran en la carpeta **Iconos** , luego haga clic en **Guardar en la base de datos** botón cuando termine:



Si bien la publicación de personalización, se Acumatica archivos de copia de seguridad automática actualmente presentes en la carpeta del sitio web, que son sustituidos por los archivos de la personalización con **SystemFile** conjunto de atributos **verdadera**.

8. Si ahora continúa con la publicación de la personalización, es muy probable que **algunos archivos se hayan modificado en el sistema de archivos**. mensaje de error para aparecer. Para evitar que aparezca este mensaje bastante aterrador, abra su proyecto en el Administrador de personalización, vaya a la sección **Archivos** y haga clic en **Detectar archivos modificados** para abrir el cuadro de diálogo **Detectado de archivos modificados** , luego haga clic en el botón **Descartar todos los cambios** :



9. Ahora puede continuar y publicar la personalización para disfrutar de sus imágenes personalizadas en la página de inicio de sesión:



Lea [Reemplazo de imágenes en la página de inicio de sesión en línea](https://riptutorial.com/es/acumatica/topic/9657/reemplazo-de-imagenes-en-la-pagina-de-inicio-de-sesion):
<https://riptutorial.com/es/acumatica/topic/9657/reemplazo-de-imagenes-en-la-pagina-de-inicio-de-sesion>

Capítulo 22: Rellenando informe con datos a través de código

Examples

Este artículo cubre un ejemplo que muestra cómo crear un informe usando registros de memoria:

Este ejemplo muestra cómo llenar un informe con datos devueltos por un delegado de vista de datos. Durante el ejercicio, desarrollaremos una pantalla de consulta que muestra la lista de pedidos de ventas entre dos fechas. El delegado de vista de datos se utilizará para completar la información de la orden de venta.

Requisitos previos:

1. Comenzamos con la declaración del DAC SOOrderFilter:

```
[Serializable]
public class SOOrderFilter : IBqlTable
{
    public abstract class dateFrom : IBqlField
    {
    }
    [PXDate()]
    [PXUIField(DisplayName = "Date From")]
    public DateTime? DateFrom { get; set; }

    public abstract class dateTo : IBqlField
    {
    }
    [PXDate()]
    [PXUIField(DisplayName = "Date To")]
    public DateTime? DateTo { get; set; }
}
```

2. Continuar con la declaración del SOOrderData DAC:

```
[Serializable]
public class SOOrderData : IBqlTable
{
    #region OrderType
    public abstract class orderType : PX.Data.IBqlField
    {
    }
    [PXString(2, IsKey = true, IsFixed = true)]
    [PXUIField(DisplayName = "Type")]
    public virtual string OrderType { get; set; }
    #endregion
    #region OrderNbr
    public abstract class orderNbr : PX.Data.IBqlField
    {
    }
}
```



```

}
[PXString(15, IsKey = true, IsUnicode = true, InputMask = ">CCCCCCCCCCCCC")]
[PXUIField(DisplayName = "Order Nbr.")]
public virtual string OrderNbr { get; set; }
#endregion
#region OrderDate
public abstract class orderDate : PX.Data.IBqlField
{
}
[PXDate]
[PXUIField(DisplayName = "Date")]
public virtual DateTime? OrderDate { get; set; }
#endregion
#region Status
public abstract class status : PX.Data.IBqlField
{
}
[PXString(1, IsFixed = true)]
[PXUIField(DisplayName = "Status")]
[SOOrderStatus.List()]
public virtual string Status { get; set; }
#endregion
#region OrderDesc
public abstract class orderDesc : PX.Data.IBqlField
{
}
[PXString(60, IsUnicode = true)]
[PXUIField(DisplayName = "Description", Visibility = PXUIVisibility.SelectorVisible)]
public virtual string OrderDesc { get; set; }
#endregion
#region OrderTotal
public abstract class orderTotal : PX.Data.IBqlField
{
}
[PXDecimal(4)]
[PXDefault(TypeCode.Decimal, "0.0")]
public virtual decimal? OrderTotal { get; set; }
#endregion
#region DueDate
public abstract class dueDate : PX.Data.IBqlField
{
}
[PXDate]
[PXUIField(DisplayName = "Due Date")]
public virtual DateTime? DueDate { get; set; }
#endregion
}

```

3. En el espacio de nombres PX.Documentation, cree su BCP de SOOrderInq usando el siguiente fragmento de código para declarar delegado de vista de datos de Resultados, que luego se usará para completar el informe con datos:

```

public class SOOrderInq : PXGraph<SOOrderInq>
{
    public PXCancel<SOOrderFilter> Cancel;
    public PXFilter<SOOrderFilter> Filter;

    [PXFilterable]
    public PXSelectOrderBy<SOOrderData,

```

```

        OrderBy<Desc<SOOrderData.orderNbr>>> Result;
protected virtual IEnumerable result()
{
    BqlCommand cmd = PXSelect<SOOrder,
        Where<SOOrder.orderDate,
            Between<Current<SOOrderFilter.dateFrom>,
                Current<SOOrderFilter.dateTo>>>>.GetCommand();
    PXView inView = new PXView(this, true, cmd);
    int startRow = PXView.StartRow;
    int totalRows = 0;
    foreach (SOOrder order in inView.Select(PXView.Currents, PXView.Parameters,
        PXView.Searches, PXView.SortColumns, PXView.Descendings, PXView.Filters,
        ref startRow, PXView.MaximumRows, ref totalRows))
    {
        yield return new SOOrderData
        {
            OrderType = order.OrderType,
            OrderNbr = order.OrderNbr,
            OrderDate = order.OrderDate,
            Status = order.Status,
            OrderDesc = order.OrderDesc,
            OrderTotal = order.OrderTotal,
            DueDate = order.DueDate,
        };
    }
    PXView.StartRow = 0;
}

public SOOrderInq()
{
    Result.Cache.AllowInsert = false;
    Result.Cache.AllowUpdate = false;
    Result.Cache.AllowDelete = false;
}

public PXAction<SOOrderFilter> Report;
[PXButton]
[PXUIField(DisplayName = "View As Report", MapEnableRights = PXCacheRights.Select,
MapViewRights = PXCacheRights.Select)]
protected virtual void report()
{
    PXReportResultset reportData = new PXReportResultset(typeof(SOOrderData));
    foreach (SOOrderData row in Result.Select())
    {
        reportData.Add(row);
    }
    throw new PXReportRequiredException(reportData, "SO610501",
PXBaseRedirectException.WindowMode.NewWindow, "Report");
}
}

```

4. Cree la página SO401090.aspx seleccionando la plantilla FormDetail y establezca las siguientes propiedades para PXDataSource:

- PrimaryView: Filter
- TypeName: PX.Documentation.SOOrderInq

Después de eso, agregue el control de entrada en el formulario del encabezado del filtro:


```

<px:PXFormView ID="form" runat="server" DataSourceID="ds" Style="z-index: 100"
  Width="100%" DataMember="Filter">
  <Template>
    <px:PXLayoutRule runat="server" StartRow="True" Merge="True" LabelsWidth="XS"
ControlSize="S" />
    <px:PXDateTimeEdit ID="edDateFrom" runat="server" CommitChanges="True"
DataField="DateFrom" />
    <px:PXDateTimeEdit ID="edDateTo" runat="server" CommitChanges="True"
DataField="DateTo" />
    <px:PXLayoutRule runat="server" />
  </Template>
</px:PXFormView>

```

Y cree las siguientes columnas para la cuadrícula de detalles:

```

<px:PXGrid ID="grid" runat="server" DataSourceID="ds" Style="z-index: 100"
  Width="100%" Height="150px" SkinID="Inquire" AllowPaging="True"
AdjustPageSize="Auto">
  <Levels>
    <px:PXGridLevel DataMember="Result">
      <Columns>
        <px:PXGridColumn DataField="OrderType" />
        <px:PXGridColumn DataField="OrderNbr" Width="90px" />
        <px:PXGridColumn DataField="OrderDate" Width="90px" />
        <px:PXGridColumn DataField="Status" />
        <px:PXGridColumn DataField="OrderDesc" Width="200px" />
        <px:PXGridColumn DataField="DueDate" Width="90px" />
      </Columns>
    </px:PXGridLevel>
  </Levels>
  <AutoSize Container="Window" Enabled="True" MinHeight="150" />
</px:PXGrid>

```

5. Añadir pantalla creada al Mapa del sitio

Para rellenar el informe con los datos devueltos por un delegado de vista de datos:

1. Pegue el archivo de informe [SO610501.rpx](#) en la carpeta Informes personalizados de su sitio web de Acumatica, luego agregue el informe de pedidos de ventas en la carpeta **Oculto del mapa del sitio**



Screen ID	Title	Icon
AR.30.60.00	Dunning Letter	
EP.61.20.00	Expense Claim Details	
EP.61.20.00	Expense Claim Details	
GL.00.00.10	BI-Creation Date	
WZ.20.15.20	Welcome Page	
WZ.20.15.01	Scenario History	
CA.20.45.50	Bank Transaction Rules	
SM.20.20.25	Wiki Articles	
GL.30.70.00	Base Form for Voucher Batches	
CA.20.80.00	Update Expiration Dates	
GL.40.50.00	Reclassification History	
SO.61.05.01	Sales Orders	

2. Declare la acción **Ver como informe** en el BLC de SOOrderInq para generar y mostrar el informe de pedidos de ventas. La excepción PXReportRequiredException acepta PXReportResultset preparado dentro de la acción para completar el informe con los datos devueltos por el delegado de la vista de datos del **resultado** :

```
public class SOOrderInq : PXGraph<SOOrderInq>
{
    ...

    public PXAction<SOOrderFilter> Report;
    [PXButton]
    [PXUIField(DisplayName = "View as Report", MapEnableRights = PXCACHERIGHTS.Select,
    MapViewRights = PXCACHERIGHTS.Select)]
    protected virtual void report()
    {
        PXReportResultset reportData = new PXReportResultset(typeof(SOOrderData));
        foreach (SOOrderData row in Result.Select())
        {
            reportData.Add(row);
        }
        throw new PXReportRequiredException(reportData, "SO610501",
        PXBaseRedirectException.WindowMode.NewWindow, "Report");
    }
}
```

VIEW AS REPORT

Date From: 1/1/2010 Date To: 1/1/2020

Sales Orders

localhost/049613-2/(W(10005))/frames/reportlauncher.aspx?id=so

Revision Two HQ Sales Orders

PDF PRINT SEND EXPORT

Type	Ref. Nbr.	Order Date	Order Description
SO	SO003631	10/19/2016	Consumer goods purchase
SO	SO003630	10/16/2016	Food purchase for event
SO	SO003629	10/21/2016	Widget purchase
SO	SO003628	10/29/2016	Electronics - new account
SO	SO003627	10/24/2016	Tech sale with installation
SO	SO003626	11/7/2016	Widget advance order
SO	SO003625	10/30/2016	Industrial lift order
SO	SO003624	10/30/2016	Custom Order with 50% pr
SO	SO003623	10/21/2016	Industrial Equipment - Speci
SO	SO003622	10/12/2016	Electronics - Drop Shipped
SO	SO003621	10/15/2016	Food Order for future delive
SO	SO003620	10/30/2016	Widget Order - Pay with CC
SO	SO003619	10/25/2016	Widget Order
SO	SO003618	10/14/2016	Consumer Good Order
SO	SO003617	10/9/2016	Consumer Good Order
SO	SO003616	10/14/2016	Consumer Good Toy Order
SO	SO003615	10/2/2016	Consumer Good Toy Order
SO	SO003614	10/31/2016	Industrial Equipment Order
SO	SO003613	10/9/2016	Industrial Equipment Order
SO	SO003612	10/5/2016	Industrial Equipment Order
SO	SO003611	10/13/2016	Laptop computer order
SO	SO003610	10/6/2016	Electronics Computing Orde
SO	SO003609	10/31/2016	Electronics Headset Order
SO	SO003608	10/26/2016	Beverage Order
SO	SO003607	10/25/2016	Food Order
SO	SO003606	10/17/2016	Food Order
SO	SO003605	10/10/2016	Food Order
SO	SO003604	10/6/2016	Food Order - Microchip
SO	SO003603	10/3/2016	Food Order
SO	SO003602	9/6/2016	Sell spare widget inventory
SO	SO003601	9/16/2016	Consumer goods purchase
SO	SO003600	9/19/2016	Food purchase for event
SO	SO003599	9/20/2016	Widget purchase
SO	SO003598	9/28/2016	Electronics - new account
SO	SO003597	9/23/2016	Tech sale with installation
SO	SO003596	10/7/2016	Widget advance order
SO	SO003595	9/29/2016	Industrial lift order

Lea Rellenando informe con datos a través de código en línea:

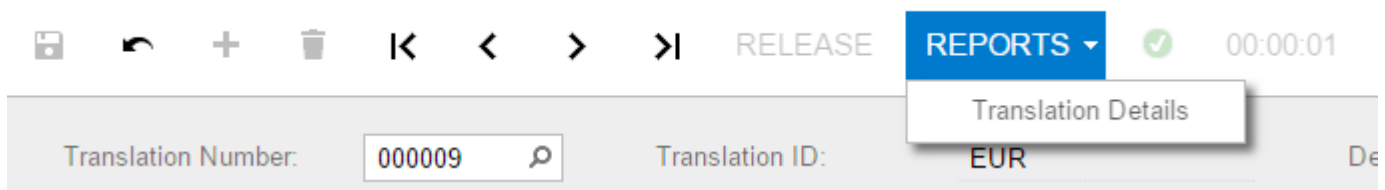
<https://riptutorial.com/es/acumatica/topic/7700/rellenando-informe-con-datos-a-traves-de-codigo>

Capítulo 23: Técnicas de interfaz de usuario

Examples

Creando un menú desplegable para una pantalla

Supongamos que necesita definir un menú desplegable para una pantalla particular de Acumatica, como el menú Informes en la siguiente captura de pantalla.



Esto se puede lograr de tres maneras diferentes:

- Agregando una barra de herramientas con un elemento de menú a ASPX de la pantalla
- Al declarar una acción de "carpeta" especial al gráfico y agregar elementos de menú en el código
- Mediante el uso del subsistema de automatización de Acumatica Framework (no cubierto por este ejemplo)

Opción 1: Crear un menú desplegable en ASPX

En primer lugar, asegúrese de que el elemento PXDataSource de la página ASPX contenga todos los comandos necesarios correspondientes a las acciones del gráfico que le gustaría realizar al hacer clic en un elemento del menú.

```
<px:PXDataSource
  ID="ds" runat="server" Visible="True" PrimaryView="TranslHistRecords"
  TypeName="PX.Objects.CM.TranslationHistoryMaint">
  <CallbackCommands>
    ...
    <px:PXDSCallbackCommand Name="TranslationDetailsReport" Visible="False"/>
    ...
  </CallbackCommands>
</px:PXDataSource>
```

A continuación, agregue un elemento personalizado de la barra de herramientas justo después del elemento PXDataSource. Dentro de él, defina un PXToolBarButton con los elementos del menú desplegable deseados que enlazan con los respectivos comandos de fuente de datos, como se muestra en el siguiente código.

```

<px:PXToolBar ID="toolbar1" runat="server" SkinID="Navigation" BackColor="Transparent"
CommandSourceID="ds">
  <Items>
    <px:PXToolBarButton Text="Reports">
      <MenuItems>
        <px:PXMenuItem Text="Translation Details" CommandSourceID="ds"
CommandName="TranslationDetailsReport"/>
      </MenuItems>
    </px:PXToolBarButton>
  </Items>
  <Layout ItemsAlign="Left" />
</px:PXToolBar>

```

Esta opción puede parecer tentadora debido a su simplicidad; Sin embargo, hay un **inconveniente importante** . Si implementa un menú desplegable de este tipo en una pantalla con un indicador de procesamiento (como una pantalla de publicación de documentos o una pantalla de procesamiento en masa), el indicador aparecerá a la izquierda de su menú desplegable, como se muestra a continuación.

Translation Number:	000007	Translation ID:	EUR	Debit Total:
Status:	Released	Branch:	MAIN - New York	Credit Total:
Translation Date:	1/29/2016	Destination Ledger ID:	TRANSEURO	Control Total:
Currency Effective Date:	10/31/2012	Destination Currency:	EUR	
Fin. Period:	09-2012	Translation Batch Num...:	00004550	
Description:	Translation to Euro			

Si esto no es deseable, considere definir un menú desplegable en el código como se describe en la sección Opción 2 a continuación.

Opción 2: Crear un menú en el gráfico

Primero, en el gráfico de la página, declare una acción de "carpeta" que corresponderá al botón del menú desplegable.

```

public PXAction<TranslationHistory> reportsFolder;
[PXUIField(DisplayName = "Reports", MapEnableRights = PXCachedRights.Select)]
[PXButton(SpecialType = PXSpecialButtonType.Report)]
protected virtual IEnumerable ReportsFolder(PXAdapter adapter)
{
    return adapter.Get();
}

```

A continuación, en el constructor del gráfico, indique que la acción es de hecho un menú desplegable y agregue todas las acciones que deben mostrarse como elementos del menú, como se muestra a continuación.

```
public TranslationHistoryMaint ()
{
    this.reportsFolder.MenuAutoOpen = true;
    this.reportsFolder.AddMenuItem(this.translationDetailsReport);
}
```

Si selecciona este enfoque, el indicador de procesamiento siempre aparecerá a la derecha de su menú, lo que posiblemente sea un UX mejor.

Lea [Técnicas de interfaz de usuario en línea](https://riptutorial.com/es/acumatica/topic/10150/tecnicas-de-interfaz-de-usuario):

<https://riptutorial.com/es/acumatica/topic/10150/tecnicas-de-interfaz-de-usuario>

Capítulo 24: Uso del complemento de personalización para realizar cambios en varias empresas

Introducción

Con las clases derivadas de **CustomizationPlug** , puede utilizar las capacidades de la plataforma de personalización de Acumatica y ejecutar código personalizado una vez que se haya publicado el proyecto de personalización. En este tema, aprenderá cómo se pueden usar los complementos de personalización para realizar cambios en varias compañías.

Puede encontrar más información sobre los complementos de personalización en la [Guía de personalización de Acumatica](#).

Examples

Implementación de un plug-in de personalización para actualizar múltiples empresas.

Para crear un complemento de personalización, simplemente crea una clase derivada de **CustomizationPlug** y la empaqueta en la personalización. Mientras el sistema está publicando el proyecto de personalización, ejecutará los métodos **OnPublished** y **UpdateDatabase** implementados en su complemento de personalización *solo dentro del alcance de la empresa actual* .

Dicho esto, el complemento de personalización nunca realizará cambios en ninguna otra empresa que no sea la actual, a menos que use **PXLoginScope** para iniciar sesión en todas las empresas, una después de la otra, disponible para la personalización de publicación del usuario actual.

A continuación se muestra un ejemplo de complemento de personalización que crea el **rol de usuario MyVerticalSolution** en todas las compañías disponibles para el usuario actual:

```
public class MyVerticalSolutionInit : CustomizationPlugin
{
    public override void UpdateDatabase()
    {
        var companies = PXAccess.GetCompanies();

        foreach (var company in companies)
        {
            using (var loginScope = new PXLoginScope(string.Format("{0}@{1}",
                PXAccess.GetUserLogin(), company)))
            {
                string roleName = "MyVerticalSolution";
                RoleAccess graph = PXGraph.CreateInstance<RoleAccess>();
            }
        }
    }
}
```

```

Roles existingRole = graph.Roles.Search<Roles.rolename>(roleName);
if (existingRole != null)
{
    WriteLog(string.Format("{0} already exists in company '{1}' - skipped",
roleName, company));
    continue;
}

var wmsRole = new Roles();
wmsRole.Rolename = roleName;
wmsRole.Descr = "User Role for MyVerticalSolution";

graph.Roles.Insert(wmsRole);
graph.Save.Press();

WriteLog(string.Format("{0} was succesfully created in company '{1}'",
roleName, company));
}
}
}
}
}

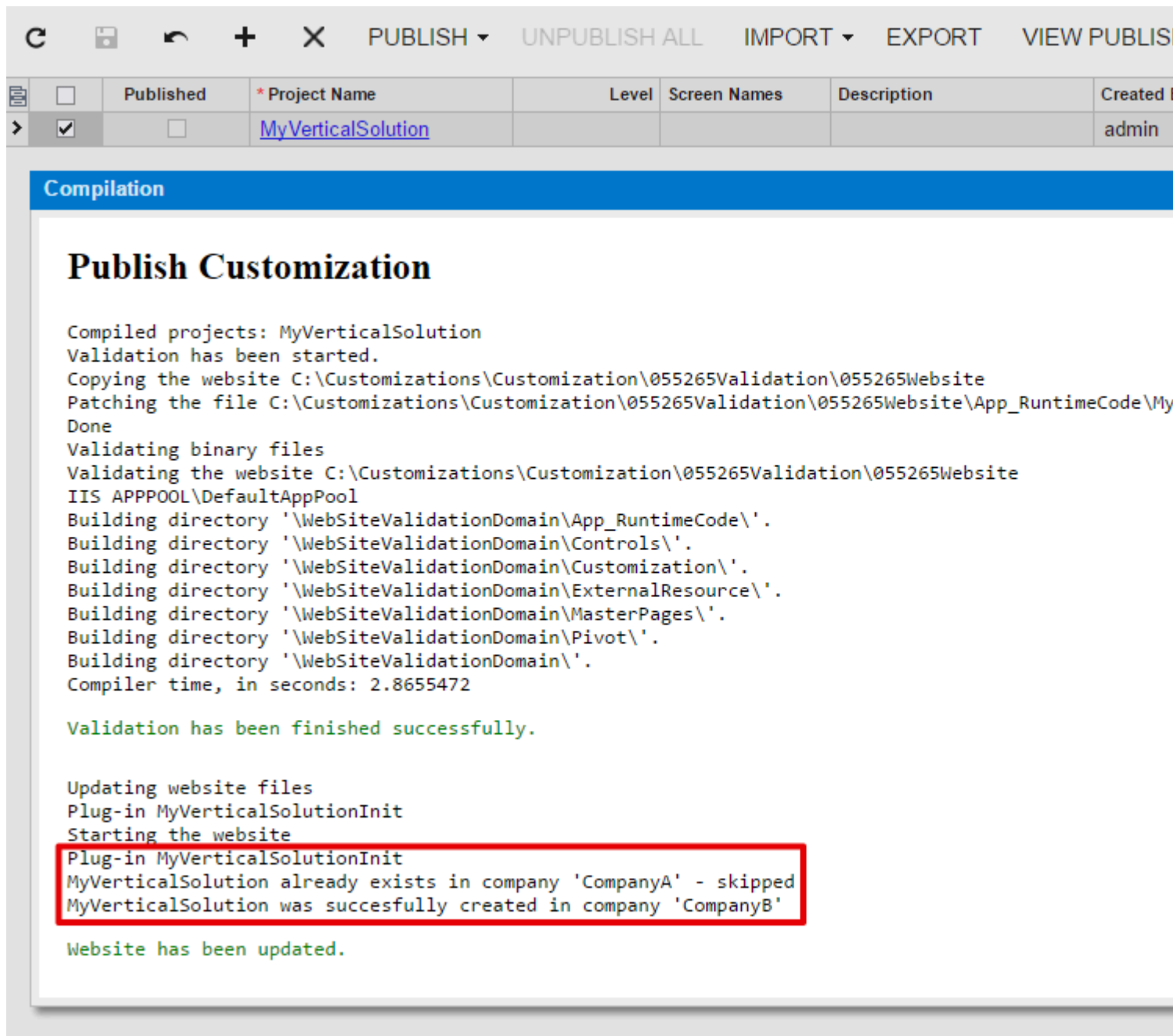
```

Para obtener una lista de compañías disponibles para el usuario actual, simplemente invoque el `PXAccess.GetCompanies()` estático `PXAccess.GetCompanies()` . Luego, **PXLoginScope** se usa para iniciar sesión en cada una de las compañías disponibles para crear el rol de usuario de **MyVerticalSolution** . Observe la instancia del **RoleAccess** BLC reinicializado para cada compañía: este es un paso absolutamente obligatorio para realizar cambios en varias compañías a la vez.

Supongamos que hay 2 empresas en su instancia de Acumatica: CompanyA y CompanyB. El usuario **administrador** , que va a utilizar para publicar la personalización, tiene acceso tanto a las empresas como **al** rol **MyVerticalSolution** , creado por el complemento de personalización, que ya existe en CompanyA:

The screenshot displays the Acumatica user security configuration interface. The top navigation bar includes the Acumatica logo, 'ORGANIZATION', and 'CONFIGURATION' tabs, along with the date and time '3/23/2017 3:38 PM'. Below the navigation bar, there are tabs for 'Common Settings', 'User Security', 'Row-Level Security', 'Document Management', and 'Email'. The 'User Security' tab is active, and the sub-tab 'User Roles' is selected. The main content area shows the configuration for a role named 'MyVerticalSolution - User Role for My'. The role description is 'User Role for MyVerticalSolution'. There is a checkbox for 'Guest Role' which is unchecked. Below the role configuration, there is a 'Membership' section with a table showing columns for '* Username' and 'Display Name'.

Después de publicar la personalización (mientras estaba conectado a CompanyA o CompanyB) con el complemento de personalización desarrollado anteriormente para crear la función **MyVerticalSolution** en todas las compañías disponibles para el usuario actual, observe la función **MyVerticalSolution** omitida para CompanyA y creada con éxito para CompanyB.



The screenshot shows a web application interface with a toolbar at the top containing icons for refresh, save, undo, add, delete, and menu items: PUBLISH, UNPUBLISH ALL, IMPORT, EXPORT, and VIEW PUBLIS. Below the toolbar is a table with columns: Published, *Project Name, Level, Screen Names, Description, and Created. The table has one row with a checked 'Published' checkbox, the project name 'MyVerticalSolution', and 'admin' in the 'Created' column.

Below the table is a blue header for the 'Compilation' section, followed by the title 'Publish Customization'. The main content is a log of compilation steps:

```
Compiled projects: MyVerticalSolution
Validation has been started.
Copying the website C:\Customizations\Customization\055265Validation\055265Website
Patching the file C:\Customizations\Customization\055265Validation\055265Website\App_RuntimeCode\My
Done
Validating binary files
Validating the website C:\Customizations\Customization\055265Validation\055265Website
IIS APPPOOL\DefaultAppPool
Building directory '\WebSiteValidationDomain\App_RuntimeCode\'
Building directory '\WebSiteValidationDomain\Controls\'
Building directory '\WebSiteValidationDomain\Customization\'
Building directory '\WebSiteValidationDomain\ExternalResource\'
Building directory '\WebSiteValidationDomain\MasterPages\'
Building directory '\WebSiteValidationDomain\Pivot\'
Building directory '\WebSiteValidationDomain\'
Compiler time, in seconds: 2.8655472

Validation has been finished successfully.

Updating website files
Plug-in MyVerticalSolutionInit
Starting the website
Plug-in MyVerticalSolutionInit
MyVerticalSolution already exists in company 'CompanyA' - skipped
MyVerticalSolution was succesfully created in company 'CompanyB'

Website has been updated.
```

La próxima vez que publique esta personalización, el rol **MyVerticalSolution** se omitirá para ambas compañías en su aplicación Acumatica:

Navigation icons: Refresh, Save, Undo, Add, Close, PUBLISH, UNPUBLISH ALL, IMPORT, EXPORT, VIEW PUBLIS

<input type="checkbox"/>	Published	* Project Name	Level	Screen Names	Description	Created
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MyVerticalSolution				admin

Compilation

Publish Customization

```
Compiled projects: MyVerticalSolution
Validation has been started.
Copying the website C:\Customizations\Customization\055265Validation\055265Website
Patching the file C:\Customizations\Customization\055265Validation\055265Website\App_RuntimeCode\MyV
Patching the file C:\Customizations\Customization\055265Validation\055265Website\App_RuntimeCode\MyV
Done
Validating binary files
Validating the website C:\Customizations\Customization\055265Validation\055265Website
IIS APPPOOL\DefaultAppPool
Building directory '\WebSiteValidationDomain\App_RuntimeCode\'
Building directory '\WebSiteValidationDomain\Controls\'
Building directory '\WebSiteValidationDomain\Customization\'
Building directory '\WebSiteValidationDomain\ExternalResource\'
Building directory '\WebSiteValidationDomain\MasterPages\'
Building directory '\WebSiteValidationDomain\Pivot\'
Building directory '\WebSiteValidationDomain\'
Compiler time, in seconds: 2.938234

Validation has been finished successfully.

Updating website files
Plug-in MyVerticalSolutionInit
Starting the website
Plug-in MyVerticalSolutionInit
MyVerticalSolution already exists in company 'CompanyA' - skipped
MyVerticalSolution already exists in company 'CompanyB' - skipped

Website has been updated.
```

Lea Uso del complemento de personalización para realizar cambios en varias empresas en línea:
<https://riptutorial.com/es/acumatica/topic/9522/uso-del-complemento-de-personalizacion-para-realizar-cambios-en-varias-empresas>

Capítulo 25: Visualización de un error que requiere ingresar datos de entidad

Examples

Visualización de un error que requiere que el usuario ingrese datos de la entidad

Los usuarios a menudo se presentan en una situación en la que no se puede finalizar un proceso de negocios porque el usuario no ha ingresado toda la información necesaria.

Un ejemplo de esta situación es cuando un usuario intenta crear un pedido de entrega inmediata con la dirección del cliente que falta.

De acuerdo con las mejores prácticas de UX, el sistema debe ser amigable con el usuario y no solo informarle al usuario sobre la situación, sino también guiarlo a la resolución de su problema. Como sabemos, el sistema ya tiene un mecanismo similar activado por `PXSetup<TSetup>.Current` cuando no hay registros en la tabla de `TSetup`. Se implementa internamente lanzando una `PXSetupNotEnteredException`.

Recientemente, se ha agregado una nueva funcionalidad a esta excepción, que permite a un desarrollador de aplicaciones lanzar un error con un enlace a la entidad que debe reconfigurarse:

```
INSite erroneousSite = PXSelect<
    INSite,
    Where<
        INSite.siteID, Equal<Current<SOCreateFilter.siteID>>,
        And<INSite.active, Equal<True>,
        And<Where<INSite.addressID, IsNull, Or<INSite.contactID, IsNull>>>>>>
        .SelectSingleBound(this, new object[] { e.Row });

if (erroneousSite != null)
{
    throw new PXSetupNotEnteredException<INSite, INSite.siteCD>(
        Messages.WarehouseWithoutAddressAndContact,
        erroneousSite.SiteCDlnk,
        erroneousSite.SiteCDinf);
}
```

El resultado se muestra al usuario así:

Error #81



The requested resource is not available. The Multiple Warehouses feature and the Transfer ord

Next step:

Navigate to the [Warehouse](#) form and enter the required configuration data.

- Como primer parámetro de tipo, `PXSetupNotEnteredException` acepta el tipo de entidad a la que se generará el enlace del gráfico predeterminado.
- El segundo parámetro de tipo denota el campo clave del registro que se utilizará para generar el enlace. En el ejemplo anterior, la navegación a la entidad de almacén se realiza mediante la clave del CD.
- El primer argumento del constructor es la cadena de formato para el mensaje de error. La numeración de sus marcadores de posición internos debe comenzar con 1: es decir, la `The Multiple Warehouses feature and the Transfer order type are activated in the system, in this case an address and a contact must be configured for the '{1}' warehouse.`
- El segundo argumento del constructor es el valor del campo clave especificado como el segundo parámetro genérico. En el ejemplo, el enlace que se generaría es `/IN204000.aspx?siteCD=erroneousSite.SiteCDlnk .`
- El tercer argumento del constructor es el valor legible por el hombre que se muestra en el mensaje de error: `...in this case an address and a contact must be configured for the 'erroneousSite.SiteCDinf' warehouse.`

Lea [Visualización de un error que requiere ingresar datos de entidad en línea:](#)

<https://riptutorial.com/es/acumatica/topic/9274/visualizacion-de-un-error-que-requiere-ingresar-datos-de-entidad>

Creditos

S. No	Capítulos	Contributors
1	Empezando con acumatica	Community
2	Acumatica BQL Reference	wh1t3cat1k
3	Acumatica Platform Atributos de Referencia	wh1t3cat1k
4	Adición de soporte de atributos a una entidad de pedido de ventas lista para usar	DChhappgar
5	Ampliación de la lista de entidades apoyadas por tareas, eventos y actividades	RuslanDev
6	Cálculo de fletes	RuslanDev
7	Cambio de subtítulos dinámicamente usando campos DAC de solo lectura.	cbetabeta , Simon ML
8	Cambio de tamaño de la ventana desplegable del selector	Gabriel , RuslanDev
9	Cambios significativos en la API entre versiones	wh1t3cat1k
10	Creando campos de fecha y hora en Acumatica	RuslanDev

11	Descarga de archivos adjuntos a una entidad de detalle mediante API basada en contrato	RuslanDev
12	Exportación de registros a través de la API basada en pantalla	RuslanDev
13	Exportación de registros a través de REST API basada en contrato	RuslanDev
14	Filtrado con valor múltiple con un solo selector.	samol518
15	Mecanismos de personalización	wh1t3cat1k
16	Modificaciones a la información de contacto y dirección a través del código	RuslanDev
17	Modificaciones a las vistas de datos base	RuslanDev
18	Modificar elementos en una lista desplegable	RuslanDev
19	Pestañas de ocultación condicional	RuslanDev
20	Publicación omitida de contenido de personalización ya aplicado.	samol518
21	Reemplazo de imágenes en la página de inicio de sesión	RuslanDev

22	Rellenando informe con datos a través de código	Gabriel, RuslanDev
23	Técnicas de interfaz de usuario	wh1t3cat1k
24	Uso del complemento de personalización para realizar cambios en varias empresas	RuslanDev
25	Visualización de un error que requiere ingresar datos de entidad	wh1t3cat1k