Relationships Extended Demonstrations

This document guides you through setting up all 6 scenarios that are handled by the Relationships Extended module in a step by step fashion. The 6 scenarios covered are:

1. Related Pages (Both Orderable AdHoc Relationships and Unordered Relationships)
2. Node Categories (using CMS.TreeNode)
3. Node Categories (using a Custom Joining Table)
4. Object to Object binding with Ordering
5. Node to Object binding with Ordering
6. Node to Object binding without Ordering.

Quick Navigation

Installation

1. Install the RelationshipsExtended NuGet package into your CMS Project on your site's solution. Make sure to select the RelationshipsExtended package that has a major version # that matches your Kentico Version (ex 10.0.6 = Kentico 10, 11.0.6 = Kentico 11, etc).
   a) https://docs.microsoft.com/en-us/nuget/consume-packages/ways-to-install-a-package
2. Run your Kentico site after the NuGet package finishes installation.
   a) Check the event log after installing, you will see a MODULEINSTALLED if it installed correctly.
   b) Ignore the RelationshipsExtended-ErrorSettingForeignKeys on the first run, as this runs before the class gets an opportunity to install. You should not see this on subsequent application starts.
3. If you are on Kentico version 10, please also go to System -> Macros -> Signatures and resign your macros.

If you have enough privileges, you should now see a “Relationship Name Extended” user interface under the Configuration section in the Kentico Admin menu. You should also now have access to the Macro namespaces RelHelper (along with RelEnums and some other classes which are used internally).

If you wish to install the Demo module, you can go to Sites within the Kentico Menu, [Import Sites or Objects] and select the RelationshipsExtendedDemo_SiteImport.zip file.

If you only want to see the code files themselves, you can unzip the Demo_CodeFilesOnly.zip file.

Setting up Related Pages

Preparations – Creating Banner and Quote Page Types

1. Go to the Page Types in the Kentico Menu
2. Create a new Page Type: Banner
   1. Page Type Display Name: Banner
      Namespace: Demo
      Name: Banner
      [Next]
   2. Add new field:
      Field Name: BannerName
      Data Type: Text
      Size: 200
      Required: True
      Field Caption: Name
      [Save]
   3. Add new field:
      Field Name: BannerText
      Data Type: Text
      Size: 200
      Required: False
      Field Caption: Banner Text
      [Save]
   4. Hit [Next] and set Page Name Source to BannerName.
   5. Hit [Next] add Folder to the Parent Types, then hit Next until you Finish

3. Create a new Page Type: Quote
    1. Page Type Display Name: Quote
       Namespace: Demo
       Name: Quote
       [Next]
    2. Add new field:
       Field Name: QuoteName
       Data Type: Text
       Size: 200
       Required: True
       Field Caption: Name
       [Save]
    3. Add new field:
       Field Name: QuoteText
       Data Type: Text
       Size: 200
       Required: False
       Field Caption: Quote Text
       [Save]
    4. Add new field:
       Field Name: QuoteAuthor
       Data Type: Text
       Size: 50
       Required: False
       Field Caption: Author
       [Save]
    5. Hit [Next] and set Page Name Source to QuoteName.
    6. Hit [Next] add Folder to the Parent Types, then hit [Next] until you [Finish]
4. Go to Pages, and create a folder under the Root “Site Objects”
5. Create two folders underneath Site Objects, “Banners” and “Quotes”
6. Under the Banners Folder, create 3 banners with Names “Banner 1” “Banner 2” “Banner 3” and text “Sample Text 1” “Sample Text 2” and “Sample Text 3”
7. Under the Quotes Folder, create 3 quotes with names “Golden Rule” “Relativity” and “Sonic” with Text “Do unto others as you would have them do unto you” (Author Jesus), “E=mc2” (Author Albert Einstein) and “Yo, buttnik!” (Author “Sonic the Hedgehog”)

Create Relationships

1. Go to Relationship Names Extended in the Kentico Menu
2. Create a new Relationship Name: Banner
    1. Display Name: Banners
       Code Name: Banners
       Relationship Is Adhoc: True
       [Save]
3. Create a new Relationship Name: Quotes

1. Display Name: Quotes
   Code Name: Quotes
   Relationship Is Adhoc: False
   [Save]

4. Go to Modules in the Kentico menu, and create a new Module
    1. Module Display Name: Demo
       Module Code Name: Demo
       Module Version: 1.0.0
    2. Go to the Sites tab and add it to the current site.
5. Go to the User Interfaces tab of the Demo Module
6. Go to CMS-> Administration -> Content Management -> Pages -> Edit (Click on Edit)
7. Click the + Icon to add a new User Interface
    1. DisplayName: Demo
       Page Template: Vertical Tabs
       [Save]
    2. Go to the Properties tab of this UI element
       Tab Extender: RelationshipsExtended
       Tab Extender: RelationshipsExtended.RelationshipVerticalTabExtender (Enables Auto Hide of Relationship UI)
       [Save]
8. Click the + icon to add a new User Interface below the Demo UI you just created
    1. DisplayName: Banners
       Page Template: Edit Relationship
       [Save]
    2. Go to the Properties tab of this UI element
       Relationship Name: Banners
       Allow Switch Sides: False
       Max Relationships: Empty
       Left Side Macro: {% CurrentDocument.ClassName != "Demo.Banner" @%}
       Right Side Macro: {% CurrentDocument.ClassName == "Demo.Banner" @%}
       Auto Hide Tab: True
       Show New Button: True
       New Page Type: Banner
       Parent Node Alias Path: /Site-Objects/Banners
       Selector Type: Tree Selector
       Allowed Page Types: Banner
       Display Name Format: {% BannerName %}
       Hover Over/ToolTip: {% BannerText %}
       Where Condition:
       Starting Paths: /Site-Objects/Banners
       Tree Selector Mode: Add Individually
       [Save]
9. Click on the Demo UI element again and hit the + icon to add another UI element
    1. DisplayName: Quotes
       Page Template: Edit Relationship
       [Save]

2. Go to the Properties tab of this UI element
   Relationship Name: Qutoes
   Allow Switch Sides: False
   Max Relationships: Empty
   Left Side Macro: {% CurrentDocument.ClassName != "Demo.Quote" @%}
   Right Side Macro: {% CurrentDocument.ClassName == "Demo.Quote" @%}
   Auto Hide Tab: True
   Show New Button: False
   Selector Type: Uni Selector
   Allowed Page Types: Banner
   Display Name Format: {% NodeName %}
   Hover Over/ToolTip: Author: {% QuoteAuthor %}, Text: {% QuoteText %}
   Where Condition:
   Object Site Name: #current
   Additional Columns: QuoteName, QuoteAuthor, QuoteText
   Search Columns: QuoteName, QuoteAuthor, QuoteText
   Tree Selector Mode: Add Individually
   [Save]

Relate Banners / Quotes to Page

1. Go to Pages in the Kentico Menu and create a new page under the Root node.
2. Name: “Relationship Test” and click “Create a blank page” so it creates an ad-hoc template.
3. Go to the new Demo tab (right of Properties) and go to Banners
    1. Add the 3 banners by clicking on them.
    2. Drag the Banner 3 to the first position
        1. Also, hover over the Banners and you'll see the Sample Text
4. Go to the Quotes tab next
    1. Hit “Select”
    2. Search for “Jesus” and only the Golden Rule will show (filtering by additional columns)
    3. Hover over the element to see the Author and text
    4. Select all 3 Quotes, and again you can hover and see the text. You can delete and re-add if you wish.
5. FOR PORTAL: Click on the Design Tab
    1. Add a Repeater (for the banners)
        1. Web part Control ID: Banner
           Path: /%
           Page Type: Demo.Banner
           Transformation: New Text/XML ({% BannerName %}: {% BannerText %})
           Main Page: Display Pages Related to the Current Page
           Main page is on the left side: True
           Relationship Name: Banners
           [Save and close]
        2. You'll now see the 3 banners with Banner 3 first (the order that you set)
    2. Add a repeater (For the quotes)

1. Web part Control ID: Qutoes
   Path: /%
   Page Type: Demo.Quote
   Order By: NewID() (random ordering)
   Transformation: New Text/XML ("{%QuoteText%}" ~{%QuoteAuthor%})
   Main Page: Display Pages Related to the Current Page
   Main page is on the left side: True
   Relationship Name: Quotes
   Cache Minutes: 0
   Partial Cache Minutes: 0
2. Now you'll see the Quotes you have selected randomized

6. FOR MVC
   1. To retrieve banners:
      ```
      new DocumentQuery("Demo.Banner").InRelationWith(TheCurrentNodeGUID, "Banners", RelationshipSideEnum.Left)
      ```
   2. To retrieve Quotes:
      ```
      new DocumentQuery("Demo.Quote").InRelationWith(TheCurrentNodeGUID, "Quotes",
      RelationshipSideEnum.Left).OrderBy("NEWID()")
      ```

Node Categories – Using CMS.TreeCategory

This demo assumes you have performed the Create Relationships demo and will leverage the Banners and Quotes that were created during it.

1. Go to Categories in the Kentico Menu
   1. Create a new Category directly under “Categories”
      Category Name: Regions
   2. Under the Regions Category, add “Global” “USA” and “Canada” (with matching CodeNames)
2. Go to Modules in the Kentico Menu and edit the Demo Module
   1. Go to the User Interface tab
   2. Go to CMS → Administration → Content Management → Pages → Edit → Demo (click on Demo)
   3. Hit the + symbol to add a new UI
      Display Name: Regions
      Page Template: Edit Categories
      [Save]
   4. Go to the Properties Tab
      Root Category: Regions
      Display Mode: Searchable List
      Minimum Categories: 1
      Maximum Categories: (leave empty)
      Save Mode: Set Node Categories
      Field Save Type: Category Ids
3. Go to the Pages UI, navigate under your Site-Objects/Banners, and select Banner 1
4. Under the Demo tab, select the new Regions
   Select Global and Save
5. Select Banner 2, and under the Regions tab select USA
6. Select Banner 3, and under the Regions tab select Canada

7. FOR PORTAL
    1. Go to the Test Page and click on the Design Tab
    2. Edit the Banner Repeater
       Where Condition: Hit the Black arrow to enter Macro mode, then put in
       {% RelHelper.GetNodeCategoryWhere("USA,Global") @%}
       [Save]
    3. You will now see only the banners that had either Global or USA (so Banner 3 no longer is showing)
8. For MVC
    1.
```
new DocumentQuery("Demo.Banner").InRelationWith(TheCurrentNodeGUID, "Banner",
RelationshipSideEnum.Left).Where(RelHelper.GetNodeCategoryWhere(new string[] { "Global", "USA" }));
```

Node Categories – Using custom Join Table

This demo assumes you have performed the Create Relationships demo and will leverage the Banners and Quotes that were created during it.

Preparations: Creation of Node-to-Category Joining table

1. Go to Modules in the Kentico Menu and Edit the Demo Module [Created earlier]
2. Go to the Classes Tab and create a new class
    1. Class display Name: Node Regions
       Namespace: Demo
       Class: NodeRegion
       [Next]
    2. Table Name: Demo_NodeRegion
       Primary Key Name: NodeRegionID
       Is M:N table: false
       Include NodeRegionGuid field: False
       Include NodeRegionLastModified field: False
       [Next]
    3. Add a new field:
       Field name: NodeID
       Data Type: Integer number
       Required: True
       Reference to: Node
       Field Caption: Node
       [Save]
    4. Add a new field:
       Field name: RegionCategoryID
       Data Type: Integer number
       Required: True
       Reference to: Content Category
       Field Caption: Region Category
       [Save]
    5. [Next], [Finish]
    6. Go to the Code Tab, and hit [Save Code]

1. Recommended that you store your Module Classes in a Class Library project. https://docs.kentico.com/k12/custom-development/creating-custom-modules#Creatingcustommodules-Compilingmodulecodeintoassemblies
2. You should ensure that your Binding class works properly with Kentico staging, please see Enabling Staging on Node-to-Object Bindings

Adding the custom Node-Category UIs

1. Go back to the Demo Module, and click on the "User Interface" tab
    1. Go to CMS → Administration → Content Management → Pages → Edit → Demo
        1. If you do not have a Demo from the previous examples, create one with a Page Template of Vertical Tabs
    2. Add a new User Interface under Demo
       Display Name: Regions (Custom Binding)
       Page template: Edit Categories
       [Save]
    3. In the Properties tab of this new UI Element
       Root Category: Regions
       Display Mode: Tree Structure
       Expand to Nth Level: 1
       Only Leafs Selectable: True
       Parent Selects Children: True
       Minimum Categories: 1
       Maximum Categories: (blank)
       Save Mode: To Joining Table
       This Object Foreign Key: NodeID
       Foreign Key Source: Current Page
       Join Table Code Name: Demo.NodeRegion
       Join Table Left Field Name: NodeID
       Join Table Right Field Name: RegionCategoryID
       Field Save Type: Category Ids
       [Save]
    4. [Optional] Since we want a minimum of 1 category, we need to also add a Field without Database representation on the Form itself so when they add a new item, it forces them to also add a Category. You do not need to have this unless you want to ensure at least 1 is selected.
        1. Go to Page Types and Edit Banner Page Type
        2. Go to Fields, and Add a new Field
           Field Type: Field without database representation
           Field Name: BannerRegions
           Data Type: Text
           Size: 200
           Field Caption: Banner Regions
           Form Control: Advanced Category Selector
           [Hit the Advanced under the Editing control settings]
           Root Category: Regions
           Display Mode: Tree Structure
           Expand to Nth Level: 1
           Only Leafs Selectable: True
           Parent Selects Children: True
           Minimum Categories: 1

Maximum Categories: (blank)
Save Mode: To Joining Table
This Object Foreign Key: NodeID
Foreign Key Source: Current Page
Join Table Code Name: Demo.NodeRegion
Join Table Left Field Name: NodeID
Join Table Right Field Name: RegionCategoryID
Field Save Type: Category Ids
[Save]

Testing

1. Go to Pages in the Kentico Menu, and navigate to the Site Objects → Banners
2. Create a new Banner
   Name: Banner 4
   Banner Text: Sample Text 4
3. Attempt to save without setting regions, you will see it won't let you save unless you have at least 1 region.
4. Add USA to the Regions, and Save
5. Go to the Demo UI Tab and select Regions (Custom Binding)
6. Attempt to remove the USA Category, again you will see it requires at least 1
7. Select USA and Global and hit Set Categories
8. Back on the Form Tab you'll see the Banner Regions is showing the 2 categories.
9. PORTAL
   1. Go to the Test Page's Design Tab, and Edit the Banners Repeater
      WHERE Condition: {% RelHelper.GetBindingCategoryWhere("Demo.NodeRegion", "NodeID", "NodeID", "RegionCategoryID", "USA,Global") @%}
      Relationship: Do not use Relationships (May have been set from previous demo)
10. MVC:

```
new DocumentQuery("Demo.Banner").Where(RelHelper.GetBindingCategoryWhere("Demo.NodeRegion", "NodeID", "NodeID",
"RegionCategoryID", new string[] { "USA", "Global" }));
```

Binding Object to Object with Ordering

Preparations: Create 2 Object Classes with CRUD UI, and Binding Class

1. Go to Modules in the Kentico Menu and edit the Demo Module
2. Go to the Classes Tab and create a new class
   1. Class display Name: Foo
      Namespace: Demo
      Class: Foo
      [Next]
      [Next]
   2. Add a new Field
      Field name: FooDisplayName
      Data Type: Text
      Size: 200
      Required: True

Field Caption: Foo Name
[Save]

3. Add a new Field
   Field name: FooCodeName
   Data Type: Text
   Size: 200
   Required: True
   Unique: True
   Field Caption: Code Name
   Form Control: Code Name
   Require Identifier Format: True
   [Save]
4. [Next], [Finish]
5. Go to the Code Tab, and hit [Save Code]
    1. Recommended that you store your Module Classes in a Class Library project.
       https://docs.kentico.com/k12/custom-development/creating-custom-modules#Creatingcustommodules-Compilingmodulecodeintoassemblies
    2. You should ensure that your Binding class works properly with Kentico staging, please see
       https://docs.kentico.com/k12/custom-development/creating-custom-modules/setting-the-type-information-for-module-classes/enabling-export-and-staging-for-the-data-of-classes#Enablingexportandstagingforthedataofclasses-Enablingstagingsupport

3. Go to the Classes Tab and create a new class
    1. Class display Name: Bar
       Namespace: Demo
       Class: Bar
       [Next]
       [Next]
    2. Add a new Field
       Field name: BarDisplayName
       Data Type: Text
       Size: 200
       Required: True
       Field Caption: Bar Name
       [Save]
    3. Add a new Field
       Field name: BarCodeName
       Data Type: Text
       Size: 200
       Required: True
       Unique: True
       Field Caption: Code Name
       Form Control: Code Name
       Require Identifier Format: True
       [Save]
    4. [Next], [Finish]
    5. Go to the Code Tab, and hit [Save Code]

    1. Recommended that you store your Module Classes in a Class Library project. https://docs.kentico.com/k12/custom-development/creating-custom-modules#Creatingcustommodules-Compilingmodulecodeintoassemblies
    2. You should ensure that your Binding class works properly with Kentico staging, please see https://docs.kentico.com/k12/custom-development/creating-custom-modules/setting-the-type-information-for-module-classes/enabling-export-and-staging-for-the-data-of-classes#Enablingexportandstagingforthedataofclasses-Enablingstagingsupport
4. Go to the Classes Tab and create a new class
    1. Display Name: Foo Bars
       Namespace: Demo
       Class: FooBar
       [Next]
    2. Is M:N Table: False
       Include FooBarGuid Field: False
       Include FooBarLastModified Field: False
       [Next]
    3. Add a new Field
       Field Name: FooID
       Data Type: Integer Number
       Required: True
       Reference to: ObjectType.Demo_foo
       Reference type: Binding
       Field Caption: Foo
       [Save]
    4. Add a new Field
       Field Name: BarID
       Data Type: Integer Number
       Required: True
       Reference to: ObjectType.Demo_bar
       Reference type: Binding
       Field Caption: Bar
       [Save]
    5. Add a new Field
       FieldName: FooBarOrder
       Data Type: integer Number
       Required: True
       Field Caption: Order
       [Save]
    6. [Next], [Finish]
    7. Go to the Code Tab, and hit [Save Code]
        1. Recommended that you store your Module Classes in a Class Library project. https://docs.kentico.com/k12/custom-development/creating-custom-modules#Creatingcustommodules-Compilingmodulecodeintoassemblies
        2. You should ensure that your Binding class works properly with Kentico staging, please see https://docs.kentico.com/k12/custom-development/creating-custom-modules/setting-the-type-information-

for-module-classes/enabling-export-and-staging-for-the-data-of-classes#Enablingexportandstagingforthedataofclasses-Enablingstagingsupport

5. Go to the User Interfaces tab on the Demo Module
    1. Navigate to CMS → Administration → Custom [click on Custom]
    2. Create a new User Interface
        1. Display Name: Foo
           Page Template: Object Listing
           [Save]
        2. Go to the Properties Tab
           Object type: ObjectType.demo_foo
           [Save]
        3. Create a UniGrid XML in the App_Data\CMSModules\Demo\UI\Grids\Demo_Foo\default.xml

```
<grid>
  <actions parameters="FooID">
    <action name="edit" caption="$general.edit$" fonticonclass="icon-edit" fonticonstyle="allow" />
    <action name="#delete" caption="$general.delete$" fonticonclass="icon-bin" fonticonstyle="critical"
confirmation="$general.confirmdelete$" />
  </actions>
  <columns>
    <column source="FooDisplayName" caption="Name" wrap="false" />
    <column cssclass="filling-column" />
  </columns>
  <objecttype name="demo.foo" />
  <options></options>
</grid>
```

    3. Click on the Foo UI element you just made, and add a new UI element under it
        1. Display Name: New Foo
           Page Template: New/Edit Object
           [Save]
    4. Click on the Foo UI Element you made, and add a new UI element under it
        1. Display Name: Edit Foo
           Page Template: Vertical Tabs
           [Save]
        2. Go to the properties tab
           Display Breadcrumbs: True
           [Save]
    5. Click on the Edit Foo UI Element you just made, add a new UI Element under it
        1. Display Name: General
           Code Name: Foo_General
           Page Template: New/Edit Object
           [Save]
    6. Click on the Custom UI Element and add a new UI Element under it
        1. Display Name: Bar
           Page Template: Object Listing
           [Save]
        2. Go to the Properties Tab
           Object Type: ObjectType.Demo_Bar
           [Save]

3. Create a UniGrid XML Definitionin the App_Data\CMSModules\Demo\UI\Grids\Demo_Bar\default.xml

```
<grid>
  <actions parameters="BarID">
    <action name="edit" caption="$general.edit$" fonticonclass="icon-edit" fonticonstyle="allow" />
    <action name="#delete" caption="$general.delete$" fonticonclass="icon-bin" fonticonstyle="critical"
confirmation="$general.confirmdelete$" />
  </actions>
  <columns>
    <column source="BarDisplayName" caption="Name" wrap="false" />
    <column cssclass="filling-column" />
  </columns>
  <objecttype name="demo.bar" />
  <options></options>
</grid>
```

7. Click on the Bar UI element you just created and add a new UI Element under it

    1. Display Name: New Bar
       Page Template: New/Edit Object
       [Save]

8. Click on the Bar UI Element you just created and add a new UI Element under it

    1. Display Name: Edit Bar
       Page Template: New/Edit Object
       [Save]

    2. Go to the Properties Tab
       Display Breadcrumbs: True
       [Save]

Add Ordering to Binding class

1. In Visual Studios, open your FooBarInfo.cs file

    1. In the New ObjectTypeInfo declaration, add these properties

```
IsBinding = true,
OrderColumn = "FooBarOrder"
```

    2. Override the TypeInfo's GetSiblingsWhereCondition

```
protected override WhereCondition GetSiblingsWhereCondition()
{
  return new WhereCondition(string.Format("FooID = {0}", this.FooID));
}
```

    3. Implement the `IOrderableBaseInfo, IBindingBaseInfo` interfaces with these methods:

```
public void SetObjectOrder(int Order)
{
  Generalized.SetObjectOrder(Order);
  SetObject();
}
public void SetObjectOrderRelative(int PositionChange)
{
  Generalized.SetObjectOrder(PositionChange, true);
  SetObject();
}
public void MoveObjectUp()
{
  Generalized.MoveObjectUp();
  SetObject();
}
public void MoveObjectDown()
{
  Generalized.MoveObjectDown();
  SetObject();
}
public string ParentObjectReferenceColumnName()
{
  return "FooID";
}
```

```
public string BoundObjectReferenceColumnName()
{
  return "BarID";
}
```

2. In Visual Studios, open your FooBarInfoProvider.cs file

    1. Adjust the SetFooBarInfoInternal method to set the Order if not present

```
protected virtual void SetFooBarInfoInternal(FooBarInfo infoObj)
{
  // Customization 1 - On Insert or update, check and set the Order
  if (ValidationHelper.GetInteger(infoObj.GetValue("FooBarOrder"), -1) <= 0)
  {
    infoObj.FooBarOrder = GetFooBars().WhereEquals("FooID", infoObj.FooID).Count + 1;
  }
  SetInfo(infoObj);
}
```

    2. Adjust the `DeleteFooBarInfoInternal` method to reorder upon delete:

```
protected virtual void DeleteFooBarInfoInternal(FooBarInfo infoObj)
{
  DeleteInfo(infoObj);
  // Customization 2, on deletion re-order
  // Initialize Order, the infoObj should still exist in memory and only needed the Generalized portion
  infoObj.Generalized.InitObjectsOrder(null);
}
```

Add Binding UI

1. Go to Modules in the Kentico Menu and edit the Demo module
2. Go to the User Interfaces tab, and navigate to CMS → Administration → Custom → Foo → Edit Foo [Click on Edit Foo]
3. Create a new UI Element under Edit Foo

    1. Display Name: Bars
    Page Template: Edit binding (Tree+Order Support)
    [Save]

    2. Go to the Properties Tab
    Bind on Primary Node Only: False
    Binding Object Type: ObjectType.Demo_FooBar
    Target Object Type: ObjectType.Demo_Bar
    Where Condition: FooID = {% Convert.ToInt(UIContext.ObjectID, 0) @%}
    [Save]

Testing

1. Go to Bar from the Kentico Menu, and add 3 Bar's (Bar 1, Bar 2, Bar 3)
2. Go to Foo from the Kentico Menu, and add 3 Foo's (Foo 1, Foo 2, Foo 3)
3. Edit the Foo 1, and go to the Bars tab

    1. Add some Bars to the Foo 1
    2. Order the Foo's
    3. Try going to another Foo (Foo 2), and you'll see no bars, until you add them.

4. PORTAL

    1. You can find the Foo's with the given Bars using the Macro
    {% RelHelper.GetBindingWhere("Demo.FooBar", "Demo.Bar", "FooID", "FooID", "BarID", "Bar2,Bar3") @%}

5. MVC

1. You can find the Foo's with the given Bars using this

```
new ObjectQuery("Demo.Foo").Where(RelHelper.GetBindingWhere("Demo.FooBar", "Demo.Foo", "FooID", "FooID", "BarID",
new string[] { "Bar2", "Bar3" }));
```

Node to Object Binding w/ Ordering

This Demo assumes you have completed the Binding Object to Object with Ordering steps as it leverages Module classes and UI elements created within it.

Add Orderable binding object

1. Go to Modules in the Kentico menu, and edit the Demo module
2. On the classes tab, create a new class
    1. DisplayName: Node Foos
       Namespace: Demo
       Class: NodeFoo
       [Next]
    2. Is M:N Table: False
       Include NodeFooGuid field: False
       Include NodeFooLastModified field: False
       [Next]
    3. Add a new Field
       Field Name: NodeID
       Data Type: Integer number
       Required: True
       Reference to: Node
       Reference Type: Binding
       Field Caption: Node
       [Save]
    4. Add a new Field
       Field Name: FooID
       Data Type: Integer number
       Required: True
       Reference To: ObjectType.Demo_Foo
       Reference Type: Binding
       Field Caption: Foo
       [Save]
    5. Add a new Field
       Field Name: NodeFooOrder
       Data Type: Integer number
       Required: True
       Field Caption: Order
       [Save]
    6. [Next], [Finish]
    7. Go to the Code Tab, and hit [Save Code]
        1. Recommended that you store your Module Classes in a Class Library project. https://docs.kentico.com/k12/custom-development/creating-custom-modules#Creatingcustommodules-Compilingmodulecodeintoassemblies

2. You should ensure that your Binding class works properly with Kentico staging, please see Enabling Staging on Node-to-Object Bindings

8. In Visual Studios, open your NodeFooInfo.cs file

    1. In the New ObjectTypeInfo declaration, add these properties

    ```
    IsBinding = true,
    OrderColumn = "FooBarOrder"
    ```

    2. Override the TypeInfo's GetSiblingsWhereCondition

    ```
    protected override WhereCondition GetSiblingsWhereCondition()
    {
      return new WhereCondition(string.Format("FooID = {0}", this.FooID));
    }
    ```

    3. Implement the IOrderableBaseInfo, IBindingBaseInfo interfaces with these methods

    ```
    public void SetObjectOrder(int Order)
    {
      Generalized.SetObjectOrder(Order);
      SetObject();
    }
    public void SetObjectOrderRelative(int PositionChange)
    {
      Generalized.SetObjectOrder(PositionChange, true);
      SetObject();
    }
    public void MoveObjectUp()
    {
      Generalized.MoveObjectUp();
      SetObject();
    }
    public void MoveObjectDown()
    {
      Generalized.MoveObjectDown();
      SetObject();
    }
    public string ParentObjectReferenceColumnName()
    {
      return "NodeID";
    }
    public string BoundObjectReferenceColumnName()
    {
      return "FooID";
    }
    ```

    1. In Visual Studios, open the NodeFooInfoProvider.cs file

        1. Adjust the SetNodeFooInfoInternal method to set the Order if not present

        ```
        protected virtual void SetNodeFooInfoInternal (NodeFooInfo infoObj)
        {
          // Customization 1 - On Insert or update, check and set the Order
          if (ValidationHelper.GetInteger(infoObj.GetValue("NodeFooOrder "), -1) <= 0)
          {
            infoObj.NodeFooOrder = GetNodeFoos().WhereEquals("NodeID ", infoObj.NodeID).Count + 1;
          }
          SetInfo(infoObj);
        }
        ```

    2. Adjust the DeleteNodeFooInfoInternal method to reorder upon delete

    ```
    protected virtual void DeleteNodeFooInfoInternal(NodeFooInfo infoObj)
    {
      DeleteInfo(infoObj);
      // Customization 2, on deletion re-order
      // Initialize Order, the infoObj should still exist in memory and only needed the Generalized portion
      infoObj.Generalized.InitObjectsOrder(null);
    }
    ```

Add UI Element

1. Go to Modules in Kentico and edit the Demo Module

2. Go to the User Interfaces and navigate to CMS → Administration → Content Management -> Pages → Edit → Demo (click on Demo)
3. Add a new UI Element
   Display Name: Foos
   Page Template: Edit Bindings (Tree+Order Support)
   [Save]
4. Go to the Properties Tab
   Object type: Node
   Bind on Primary Node Only: True
   Binding object type: ObjectType.demo_nodefoo
   Target Object type: ObjectType.demo_foo
   Where condition: NodeID = {% Convert.ToInt(QueryString.NodeID, -1) %}
   [Save]

Testing

1. Go to the Pages within Kentico
2. Click on your Test Page
3. Under the Demo tab, select Foos

Node to Object Binding w/out Ordering

This Demo assumes you have completed the Binding Object to Object with Ordering steps as it leverages Module classes and UI elements created within it.

Preparations: Create Baz class and Node-Baz binding class

1. Go to Modules in the Kentico Menu and edit the Demo Module
2. Go to Classes and Create a new Class
   1. Display Name: Baz
      Namespace: Demo
      Code Name: Baz
      [Next]
      [Next]
   2. Add a new Field:
      Field name: BazDisplayName
      Data Type: Text
      Size: 200
      Required: True
      Field Caption: Name
      [Save]
   3. Add a new Field:
      Field name: BazCodeName
      Data Type: Text
      Size: 200
      Required: True
      Field Caption: Code Name
      Form control: Code name
      [Save]

4. [Next] [Finish]
5. Go to the Code Tab, and hit [Save Code]
    1. Recommended that you store your Module Classes in a Class Library project. https://docs.kentico.com/k12/custom-development/creating-custom-modules#Creatingcustommodules-Compilingmodulecodeintoassemblies
    2. You should ensure that your Binding class works properly with Kentico staging, please see https://docs.kentico.com/k12/custom-development/creating-custom-modules/setting-the-type-information-for-module-classes/enabling-export-and-staging-for-the-data-of-classes#Enablingexportandstagingforthedataofclasses-Enablingstagingsupport

3. Go to the Classes and Create a new Class
    1. Display Name: Node Bazs
       Namespace: Demo
       Class: NodeBaz
       [Next]
    2. Is M:N table: False
       Include NodeBazGuid field: False
       Include NodeBazLastModified field: False
       [Next]
    3. Create a new Field
       Field Name: NodeID
       Data Type: Integer number
       Required: True
       Reference to: Node
       Reference type: Binding
       Field Caption: Node
       [Save]
    4. Create a new Field
       Field Name: BazID
       Data Type: Integer number
       Required: True
       Reference to: ObjectType.Demo_Baz
       Reference type: Binding
       Field Caption: Baz
       [Save]
    5. [Next] [Finish]
    6. Go to the Code Tab, and hit [Save Code]
        1. Recommended that you store your Module Classes in a Class Library project. https://docs.kentico.com/k12/custom-development/creating-custom-modules#Creatingcustommodules-Compilingmodulecodeintoassemblies
        2. You should ensure that your Binding class works properly with Kentico staging, please see Enabling Staging on Node-to-Object Bindings
    7. In Visual Studios, open the NodeBazInfo.cs class, and add to the new ObjectTypeInfo:
       ```
       IsBinding = true
       ```
4. Go to the User Interface tab
5. Navigate to CMS → Custom, click on Custom

6. Add a new User Interface

    1. Display Name: Baz
       Page Template: Object Listing
       [Save]

    2. Properties
       Object Type: ObjectType.Demo_Baz
       [Save]

7. Add a new User Interface beneath the Baz one you just created

    1. Display Name: New Baz
       Code name: NewBaz
       Page Template: New / Edit Object
       [Save]

8. Add a new User Interface beneath the Baz one you created

    1. Display Name: Edit Baz
       Code Name: EditBaz
       Page Template: New / Edit Object
       [Save]

    2. Go to the properties tab
       Display Breadcrumbs: True

9. Create a UniGrid XML Definition at /App_Data/CMSModules/Demo/UI/Grids/Demo_Baz/default.xml

```
<grid>
  <actions parameters="BazID">
    <action name="edit" caption="$general.edit$" fonticonclass="icon-edit" fonticonstyle="allow" />
    <action name="#delete" caption="$general.delete$" fonticonclass="icon-bin" fonticonstyle="critical"
confirmation="$general.confirmdelete$" />
  </actions>
  <columns>
    <column source="BazDisplayName" caption="Name" wrap="false" />
    <column cssclass="filling-column" />
  </columns>
  <objecttype name="demo.baz" />
  <options></options>
</grid>
```

Add UI Element

1. In the Demo Module, go to User Interface
2. Navigate to CMS → Administration → Content management → Pages → Edit → Demo (Click on Demo)
3. Create a new User Interface

    1. Display Name: Bazs
       Page Template: Edit bindings (Tree+Order Support)
       [Save]

    2. Go to the properties tab
       Object Type: Node (cms.node)
       Bind on Primary Node Only: True
       Binding Object Type: ObjectType.Demo_NodeBaz
       Target Object type: ObjectType.Demo_Baz
       Where condition: NodeID = {% Convert.ToInt(QueryString.NodeID, 0) @%}
       [Save]

Testing

1. Go to the Baz in the Kentico Menu
    1. Create 3 Bazs (Baz 1, Baz 2, Baz 3)
2. Go to the Pages and click on your test page
3. Go to the Demo tab → Bazs
    1. Add a couple Bazs
    2. Remove a couple
    3. Click on another page and you can add more Baz there
4. PORTAL:
    1. You can find the Nodes with the given Bazs using the Macro
       {% RelHelper.GetBindingWhere("Demo.NodeBaz", "demo.baz", "NodeID", "NodeID", "BazID", "Baz2,Baz3") %}
    2. You can get the given Node's Baz's through a simple where condition of
       BazID in (Select BazID from Demo_NodeBaz where NodeID = '{% CurrentDocument.NodeID %}')
5. MVC
    1. You can find the Nodes with the given Bars using this

```
new DocumentQuery().Where(RelHelper.GetBindingWhere("Demo.NodeBaz", "demo.baz", "NodeID", "NodeID", "BazID", new string[] { "Baz2", "Baz3" }));
```

    2. You can find the Baz's for the given NodeID using

```
BazInfoProvider.GetBazes().WhereIn("BazID", NodeBazInfoProvider.GetNodeBazes().WhereEquals("NodeID", TheNodeID).Select(x => x.BazID).ToList());
```

       OR

```
BazInfoProvider.GetBazes().Where(string.Format("BazID in (Select BazID from Demo_NodeBaz where NodeID = {0}", NodeID);
```

Enabling Staging on Node-to-Object Bindings

While normal object to object bindings can be easily staged with their parent through setting the Synchronization Settings properly on the TypeInfo, binding to a parent Node requires manual handling. Below are the steps to enable Staging on Bound objects (such as Node Regions or Node Baz).

Preparations: Add InitializationModule

1. Create a new file at App_Code/CMSModules/Demo/DemoInitializationModule.cs

```
using CMS;
using CMS.Base;
using CMS.DataEngine;
using CMS.DocumentEngine;
using CMS.Synchronization;
using Demo;
using System.Data;
using System.Linq;
using CMS.EventLog;
using System.Collections.Generic;
using RelationshipsExtended;
using CMS.Taxonomy;
[assembly: RegisterModule(typeof(DemoInitializationModule))]
public class DemoInitializationModule : Module
{
  public DemoInitializationModule() : base("DemoInitializationModule") { }
  protected override void OnInit()
  {
    base.OnInit();
```

```
    // Custom Items here
  }
}
```

Set Staging Settings to bind objects to Document

By default, if you set the Binding object's SynchronizationSettings.LogSynchronization to SynchronizationTypeEnum.LogSynchronization, each time you add or remove a binding, an individual staging task is generated. This can be less than desirable though because you can't re-sync a node and its bindings, they are disconnected. The following section shows you how to make it where Node to object bindings will instead append their data to Document Update tasks, so when you push a document, it pushes all the relationships with it, and allows you to re-synchronize a node with its bound objects by simply resynchronizing that page.

Node Baz (Node to Object), Node Foo (Node to Object w/Order), Node Regions (Node to Object)

1. In Visual Studios, open the various info files (NodeBazInfo.cs, NodeFooInfo.cs, NodeRegionInfo.cs)
    1. In the new ObjectTypeInfo declaration, add this:

```
SynchronizationSettings = {
  // Logging is handled separately
  LogSynchronization = SynchronizationTypeEnum.None
},
```

Trigger Document Updates on Binding Inset/Update/Delete

Node Baz & Node Regions (Node to Object w/out Binding)

1. Add the following to the DemoInitializationModule's OnInit();

```
// Manually Trigger document update staging task.
NodeBazInfo.TYPEINFO.Events.Insert.After += NodeBaz_Insert_Or_Delete_After;
NodeBazInfo.TYPEINFO.Events.Delete.After += NodeBaz_Insert_Or_Delete_After;

// Manually Trigger document update staging task.
NodeRegionInfo.TYPEINFO.Events.Insert.After += NodeRegion_Insert_Or_Delete_After;
NodeRegionInfo.TYPEINFO.Events.Delete.After += NodeRegion_Insert_Or_Delete_After;
```

2. Add the following methods to the DemoInitializationModule

```
private void NodeBaz_Insert_Or_Delete_After(object sender, ObjectEventArgs e)
{
  RelHelper.HandleNodeBindingInsertUpdateDeleteEvent(((NodeBazInfo)e.Object).NodeID, "demo.nodebaz");
}
private void NodeFoo_Insert_Or_Update_Or_Delete_After(object sender, ObjectEventArgs e)
{
  RelHelper.HandleNodeBindingInsertUpdateDeleteEvent(((NodeFooInfo)e.Object).NodeID, "demo.nodefoo");
}
```

Node Foo (Node to Object w/ Ordering)

1. Add the following to the DemoInitializationModule's OnInit();

```
// Manually Trigger document update staging task.
NodeFooInfo.TYPEINFO.Events.Insert.After += NodeFoo_Insert_Or_Update_Or_Delete_After;
NodeFooInfo.TYPEINFO.Events.Update.After += NodeFoo_Insert_Or_Update_Or_Delete_After;
NodeFooInfo.TYPEINFO.Events.Delete.After += NodeFoo_Insert_Or_Update_Or_Delete_After;
```

2. Add the following method to the DemoInitializationModule

```
private void NodeFoo_Insert_Or_Update_Or_Delete_After(object sender, ObjectEventArgs e)
{
  RelHelper.HandleNodeBindingInsertUpdateDeleteEvent(((NodeFooInfo)e.Object).NodeID, "demo.nodefoo");
}
```

Add Node-binding data to Document Staging Task Data

Note that in this example, we combined all 3 into one call, you should be able to understand the code and adjust as needed.

1. Add the following to the DemoInitializationModule's OnInit();

```
// Manually add items to Document Update task
StagingEvents.LogTask.Before += LogTask_Before;
```

2. Add the following method to the DemoInitializationModule

```
private void LogTask_Before(object sender, StagingLogTaskEventArgs e)
{
  RelHelper.UpdateTaskDataWithNodeBinding(e, new NodeBinding_DocumentLogTaskBefore_Configuration[]
    {
      new NodeBinding_DocumentLogTaskBefore_Configuration(new NodeBazInfo(), "NodeID = {0}"),
      new NodeBinding_DocumentLogTaskBefore_Configuration(new NodeFooInfo(), "NodeID = {0}"),
      new NodeBinding_DocumentLogTaskBefore_Configuration(new NodeRegionInfo(), "NodeID = {0}")
    }
  );
}
```

Manually handle the Node to Object data on Task Processes

Note that in this example, we again combined all the items into a single method.
Please pay special attention to how the Node-to-Object binding with Ordering has additional logic
Make sure your target environment has this same Code files and database objects for it to processes properly.

1. Add the following to the DemoInitializationModule's OnInit()

```
// Manuall handle the Staging Task and processes our Node bound objects
StagingEvents.ProcessTask.After += ProcessTask_After;
```

2. Add the following method to the DemoInitializationModule

```
private void ProcessTask_After(object sender, StagingSynchronizationEventArgs e)
{
  if (e.TaskType == TaskTypeEnum.UpdateDocument)
  {
    // First table is the Node Data
    DataTable NodeTable = e.TaskData.Tables[0];
    if (NodeTable != null && NodeTable.Columns.Contains("NodeGuid"))
    {
      // Get node ID
      TreeNode NodeObj = new DocumentQuery().WhereEquals("NodeGUID", NodeTable.Rows[0]["NodeGuid"]).FirstObject;
      // Don't want to trigger updates as we set the data in the database, so we won't log synchronziations
      using (new CMSActionContext() {LogSynchronization = false, LogIntegration = false})
      {
        #region "Node Baz (Node object w/out Ordering)"
        // Get NodeBaz and Handle
        List<int> BazIDs = RelHelper.NewBoundObjectIDs(e, "demo.nodebaz", "NodeID", "BazID", BazInfo.TYPEINFO);
        NodeBazInfoProvider.GetNodeBazes().WhereEquals("NodeID", NodeObj.NodeID).WhereNotIn("BazID",
BazIDs).ForEachObject(x => x.Delete());
        List<int> CurrentBazIDs = NodeBazInfoProvider.GetNodeBazes().WhereEquals("NodeID", NodeObj.NodeID).Select(x =>
x.BazID).ToList();
        foreach (int NewBazID in BazIDs.Except(CurrentBazIDs))
        {
          NodeBazInfoProvider.AddTreeToBaz(NodeObj.NodeID, NewBazID);
        }
        #endregion
        #region "Node Region (Node object w/out Ordering)"
        // Get NodeRegion and Handle
        List<int> RegionCategoryIDs = RelHelper.NewBoundObjectIDs(e, "demo.nodeRegion", "NodeID", "RegionCategoryID",
CategoryInfo.TYPEINFO);
        NodeRegionInfoProvider.GetNodeRegions().WhereEquals("NodeID", NodeObj.NodeID).WhereNotIn("RegionCategoryID",
RegionCategoryIDs).ForEachObject(x => x.Delete());
        List<int> CurrentRegionCategoryIDs = NodeRegionInfoProvider.GetNodeRegions().WhereEquals("NodeID",
NodeObj.NodeID).Select(x => x.RegionCategoryID).ToList();
        foreach (int NewRegionCategoryID in RegionCategoryIDs.Except(CurrentRegionCategoryIDs))
        {
          NodeRegionInfoProvider.AddTreeToCategory(NodeObj.NodeID, NewRegionCategoryID);
        }
        #endregion

        #region "Node Foo (Node object with Ordering)"
        // Get NodeFoo and Handle
        List<int> FooIDInOrders = RelHelper.NewOrderedBoundObjectIDs(e, "demo.nodeFoo", "NodeID", "FooID",
"NodeFooOrder", FooInfo.TYPEINFO);
        NodeFooInfoProvider.GetNodeFoos().WhereEquals("NodeID", NodeObj.NodeID).WhereNotIn("FooID",
FooIDInOrders).ForEachObject(x => x.Delete());
        List<int> CurrentFooIDs = NodeFooInfoProvider.GetNodeFoos().WhereEquals("NodeID", NodeObj.NodeID).Select(x =>
x.FooID).ToList();
        foreach (int NewFooID in FooIDInOrders.Except(CurrentFooIDs))
        {
```

```
          NodeFooInfoProvider.AddTreeToFoo(NodeObj.NodeID, NewFooID);
        }
        // Now handle the ordering
        for (int FooIndex = 0; FooIndex < FooIDInOrders.Count; FooIndex++)
        {
          int FooID = FooIDInOrders[FooIndex];
          NodeFooInfo CurrentObj = NodeFooInfoProvider.GetNodeFooInfo(NodeObj.NodeID, FooID);
          if (CurrentObj != null && CurrentObj.NodeFooOrder != (FooIndex + 1))
          {
            CurrentObj.SetObjectOrder(FooIndex + 1);
          }
        }
        #endregion
      }
    } else if (NodeTable == null || !NodeTable.Columns.Contains("NodeGuid"))
    {
      EventLogProvider.LogEvent("E", "DemoProcessTask", "No Node Table Found", eventDescription: "First Table in the
incoming Staging Task did not contain the Node GUID, could not processes.");
    }
  }
}
```