# NetQuarry, Inc.

# Training

# 200 - Metadata Basics

# Mappers

Mappers are the most important objects you will work with in the NetQuarry Environment. The mapper is an object that virtualises read and write access to the database. It also used as a base for how the data is presented to the user on pages. It's up to the page to layout controls, but it's the mapper that tells the page what controls to display and how they should be displayed. The mapper even provides a rudimentary set of business rules that allows you to create a rich user interface without writing any code.

There are many attributes and properties of mappers and we will only be discussing the most commonly used

## Attributes

The most commonly used attributes for a mapper are the DenyInsert, DenyUpdate, DenyDelete settings. These control whether the user is allowed to modify or delete records from the database through this mapper. DenyInsert means that a user is not allowed to create new records through the mapper. In the UI this would manifest itself as the New button being disabled or hidden. DenyUpdate means the user is not allowed to make changes to an existing record in the mapper. In the UI this would be evident from the Save button being disabled or hidden. For DenyDelete, the UI would not show the delete button (or would be disabled). Since the mapper is the lowest level object of presentation, when you specify these attributes, you cannot override them at the page level.

Importantly, although the UI is affected by these attributes, programmatically, you can still add, update and delete from a mapper, even if it is marked with any or all of these attributes.

The available Mapper Attributes are…

| Attribute | Description |
|---|---|
| AppendPKSort | This mapper should append to the sort the primary key to ensure deterministic paging. |
| BiDirectional | Allow forward and backward dataset access. By default access is forward-only. |
| DebugMode | The mapper should be configured for debugging. |
| DenyDelete | Do not allow deletions. |
| DenyInsert | Do not allow inserts. |
| DenyUpdate | Do not allow updates. |
| FastStart | For performance the datasheet should not query unless a user filter has been supplied. |
| LogicalDelete | Records should be logically deleted, rather than physically deleted, based on the LogicalDelete property. |

| Attribute | Description |
|---|---|
| LogTransactions | Log SQL transactions to the xmt_transactions table. |
| NoClientValidation | No client-side validation should be performed on the mapper's values. |
| NoCount | This mapper should not implicitly query for count information (for performance tuning). |
| NoCSSOverride | Do not override the Cascading Style Sheet for the table. |
| NoDupeCheck | This mapper should not perform duplicate checking (ignoring any field DupeCheck attributes). |
| NoHeaderFreezing | Do not freeze the column or row headers when scrolling datasheet. |
| NoRowRequery | Do not perform a row requery after a save. |
| NoTabOrdering | Do not apply automatic tab ordering/indexing on detail forms. |
| PopupDetail | DEPRECATED (see Page attributes)! When a drilldown to detail is performed, pop up the detail. |
| SkipCodeGeneration | This mapper should be skipped during the generation of TypedMapper derived classes. |
| UseDistinct | Query using the SQL DISTINCT keyword. |

## Properties

### Mapper Properties

| Property | Description |
|---|---|

## Mapper Properties

| Property | Description |
|---|---|
| AggregateDisplay | Specifies if and how a mapper (currently datasheet only) should display aggregate data. Note that aggregate data display requires an additionalquery by the mapper.<br><br>**Attribute** / **Description**<br>AllowPageToDisplay: Allow page to enable aggregates by setting the page's DisplayAttributes.DisplayAggregates attribute.<br>Display: Aggregate data should be displayed by default.<br>DisplaySelectable: The user can select whether or not to display aggregate data. The default is specified by the Display option.<br>PersistForPageInstance: If the user selects to show/hide aggregate data, that selection should persiste for the life of the page only.<br>PersistForSession: If the user selects to show/hide aggregated data, that selection should persiste for the life of the session only.<br>PersistPermanently: If user selects to show/hide aggregate data, that selection should persist across sessions until the user changes it. |
| AjaxValueFields | A semi-colon separated list of keys of fields should be added to Ajax events such as Button AjaxClick. |
| Caption | The caption of this mapper. May contain literal text or field expressions. |
| ClassName | Overrides the default name of the class generated by the typed mapper generated code. By default the class name is the same as the mapper's key name with the spaces stripped out. |
| ClientHtmlBlock | A block of HTML that will be added to the page using the .Net RegisterClientScriptBlock() method. |
| ClientScriptBlock | A block of JavaScript (typically function declarations) that will be added to the page using the .Net RegisterClientScriptBlock() method, but wrapped in a <script></script> block. |
| ClientStartupScript | A block of JavaScript (to be run at page startup) that will be added to the page using the .Net RegisterStartupScript() method, but wrapped in a <script></script> block. |
| Columns | The number of columns to use when laying out a detail control. |
| CommandTimeout | The time (in seconds) to wait for the command to execute. The minimum value is 30 seconds. This overrides the CommandTimeout property on the Data Source associated with the mapper. |
| DefaultGrouping | The grouping (a column name) to use when a page containing the mapper is opened. |
| DefaultSort | The sort (typically a column name) to use when a page containing the mapper is opened. |
| FitWidth | Fit the columns to the width of the text. |
| InitialFocus | Identifies the field to be given initial focus in detail views of the mapper. |
| ItemDescription | A description of a single record in the mapper (may include field references). |
| LogicalDelete | Specifies the field to use for logical delete handling. Ignored unless the LogicalDelete attribute is set. |
| ObjectType | |
| PhantomCellWidth | The width of each column in a detail layout. |
| PhantomColumns | The number of columns to use when laying out a detail control. |

### Mapper Properties

| Property | Description |
|---|---|
| QuerySQL | A SQL query used to retrieve data for the mapper. This overrides the SQL of the view name. This is also the location to specify a stored procedure to use for the data source instead of a table/view. Can be overridden per page and/or NavTarget |
| RelatedPicklists | A semi-colon delimted list of picklists (SQL and cached) that should be refreshed when this mapper is changed. |
| RowCssFields | A semi-colon delimited list of keys of fields whose values should be added as CSS styles on the row. |
| RowRevision | Specifies the field to use for optimistic locking via row revisioning. Field must be an integer. |
| RowsPerPage | The number of rows to display on a page in list view. Can be overridden per page and/or NavTarget. |
| TabOrderAttributes | Specifies how the mapper should tab order controls in detail forms, Ignores if the NoTabOrdering attribute is set. Overridden by any non-zero page-level TabOrderAttributes setting. |

| Attribute | Description |
|---|---|
| IgnoreGrouping | Do not calculate tab order based on control grouping. |
| IncludeLabels | Include label control in the tab order. |
| InlcudeLockedFields | Include locked fields in the tab ordering (overrides TabOrderLockedFields AppAppearance setting). |
| IncludeUnpositioned | Include elements that are not absolutely positioned. |
| OmitLockedFields | Do not include locked fields in the tab ordering (override TabOrderLockedFields AppAppearance setting). |
| TopToBottom | Fields should be ordered top-to-bottom vs. the default of left-to-right. |

### Extensions

The extension subform is where you associate a mapper extension component with a mapper. A mapper may have zero or more extensions associated with it. You can prioritise the order in which an extension is fired in case there are extensions that handled events that must be handled first, or last. The extension with the highest priority number gets first crack at handling the event. The priority of extensions is set up in the studio to order automatically by priority. The highest priority extensions appear first in the list. The lowest priority is 1. Higher priority is > 1.

Extensions can also be set up to pay attention to permissions, so extensions can be loaded based on who is logged in.

### Permissions

By default, on a mapper, all the permissions are set and available so that all roles are allowed all methods of viewing/modification.

| Permission | Description |
|---|---|
| Delete | Controls ability to delete recrods |
| Insert | Controls ability to add new records |
| Read | Controls ability to read data |

| | |
|---|---|
| Update | Controls ability to modify existing records |

### Filters

By default a mapper has no filtering, so that when the mapper selects data from the view, it returns all the data from the view. Any additional filters added from pages or programmatically, are added to these filters. The filters added here are applied every time data is queried by the mapper.

When applying a filter criteria, we can specify a SQL statement, or refer to one of the pre-defined named filter definitions. If both SQL and NamedFilters are specified, the Named Filter setting takes precedence.

When applying filters, it is typical to specify a permission as to which role is the filter to be applied. It's important to understand that the permissions for filters work slightly differently than other permissions that if you want a filter to be applied, then the permission is to be UNCHECKED. If you want the filter to be omitted or ignored for a particular role type, then the permission needs to be checked.

## Fields

Fields are equivalent to the columns in a dataset. The fields define how the data will be presented to the user on a page. As with mapper objects, fields have attributes and properties and a set of permissions to control role based permissions.

### Attributes

| Attribute | Description |
|---|---|
| AlternateKey | This field is an alternate primary key. This is used when mappers are to be used for WebServices ("Remoted"). You would mark a field as the alternate key that provides an equivalent uniqueness to the primary key. The remote web service user may not use the applications internal primary key as an identifier, so you would match their primary key to another field in the mapper. Only one field on the mapper should be marked as the alternate key. |
| Audit | This field should be audited by the mapper's audit extension. |
| AutoAssign | The field's value should be set automatically on create. |
| DefaultFromParent | This field's default value should be set from it's mapper's parent mapper. |
| DupeCheck | On insert, verify that no record exists having values matching those of all fields marked as DupeCheck fields. |
| EditInList | The field should be editable by the user in list views. |
| ExcludeFromSelect | The field should not be queried. Generally used when there is no corresponding column in the underlying view. |
| Filterer | This field is used as a filtering element within the mapper. |
| FreezeColumn | Freeze this field's column in the datasheet. |
| HtmlCaption | The field's caption is HTML and should not be encoded. |
| HtmlContent | The field's content is HTML and should not be encoded. |
| Identity | The field is a database 'IDENTITY' or 'AutoIncrement' column. |

| Attribute | Description |
|---|---|
| InitOnly | This field should be locked on existing records. |
| Locked | The field is readonly in the UI (but can still be set programmatically). |
| NoClientValidation | No client-side validation should be performed on the field's value. |
| NonNullable | This field's underlying data column does not support null values. |
| PK | The field is a primary key for its table. |
| QuickCreate | The field should be part of a minimal record creator. |
| Required | The field cannot be null. |
| ReservedWord | This field's column name is a reserved word in the DBMS and should be escaped accordingly. |
| Search | The field should be used when searching using this Mapper. |
| Sensitive | The field's data is sensitive and, when not visible, must not be allowed in page source. This will prevent javascript access. |
| SubformKey | This field's value is required by subforms. |
| UniqueKey | This field uniquely identifies a row in this mapper. If no field is marked as the UniqueKey, then the key will be formed from the set of fields marked as PK (primary key). |
| UseDefaultOnUpdate | The field's default value should also be used on updates (in addition to inserts). |

### *PK (PrimaryKey)/UniqueKey*

When a mapper is created and loaded from a view, the database slurper identifies the underlying tables of each field in the view and whether that field is the primary key of that table. Without the primary key, we cannot guarantee that we update the records we expect to during a save. In views with multiple tables, there may be the same number of primary keys in the mapper as there are tables in the underlying view.

Because of this, the mapper will construct a rowkey based on each of the primary key values on that row. Although the basics of this are supported, we recommend that one field in the mapper is defined as a unique key. Normally that unique key is one of the fields marked as the primary key. The unique key field is effectively the single primary key field for the view. The purpose of the unique key is to ensure we select only the data we expect. The primary key fields are still required and used for insert and updates even when a unique key field is defined.

#### What primary keys control

During a save, the mapper identifies which fields have been modified and then groups those fields into a collection based on the name of the underlying table. From that collection, we build an insert or update statement as appropriate. If there is no primary key for a table on which we've changed a field value, no insert or update statement will be generated nor executed.

If this does happen, we do write a devlog entry indicating that this has occurred. Typically you will see behavior where you've modified a field value but after the save, the value hasn't actually changed.

To reduce the potential for this happening, we provide a visual clue in the UI if a field has no primary key for the underlying table. We lock the field making it uneditable as would all the fields be locked that are associated with that primary key less table. This is not uncommon when you've had to add a field to an existing mapper after adding it to a view.

In extreme cases (no primary key marked, or not configured correctly) the entire mapper would be locked and uneditable.

### *Other Slurper Specified Attributes*

AutoAssign (on PK fields), Identity, NonNullable

### When would I set some of these attributes?

| | |
|---|---|
| You always want a field to be locked | Set the Locked attribute. |
| You want a field to be editable on a new record, but locked on an existing record | Set the InitOnly attribute |
| You want to be able to edit a value of the field in the list view as well as detail | Set the EditInList attribute |
| You want to have a field visible in the filter criteria of a Find Widget | Set the Search attribute. Also fields with the Required attribute will appear in the Filter Widget criteria |
| You always want a field to have a value (blank entry is forbidden) | Set the Required attribute |
| You want to add a field to the mapper that isn't a real field in the underlying table or view. | Set the ExcludeFromSelect attribute on the field. Use in conjunction with the Table property on the property sheet and setting '+' as property value to make the field editable |
| You want to a value to be copied from a parent when you insert a new record in the subform. | Set the DefaultFromParent attribute. Use in conjunction with the ParentField property on the property sheet if the parent field has a different name to the subform's field. |
| You want the default value to be applied to a field, even when the record is updated. For example, setting an update by, or updated date field and you don't want to write code | Set the UseDefaultOnUpdate attribute and the value specified in the DefaultValue property on the property sheet will be set on the field during inserts and updates. |

### Properties (List View)

| Column | Description |
|---|---|
| Key Name | The name of the field in the mapper. This is typically the name of the field in the underlying table or view, but doesn't have to be, as that can be overridden. However, this is the name of the field in the mapper that you refer to in code and in meta data. |

### Properties (List View)

| Column | Description |
|---|---|
| Column Order | The order in which the fields are displayed on the list view and in the phantom detail layout |
| Picklist | One of the three types of picklist that can be associated with a field. The picklist might be there to allow different selections, or just to transform the key value to a display text |
| Column Width | The width (approximately) of the column in characters. A column width of 0 means the field will not be visible on the list view, nor on the detail. |
| Cell Type | The type of field to display. Each field has a common set of properties on the property sheet, but changing the cell type, you also get a set of cell type specific properties. See below for more details. |
| .NET Type | The .NET data type of the field. |
| Data Type | The DBMS data type of the field. The .Net data type for the field has to be consistent with the database data type otherwise there are likely to be type conversion errors or data truncation problems that are hard to track down. The best way to add fields to mappers is to use the slurper functionality, rather than manually adding a field. If you must add a field manually, or modify an existing field, be careful not to mismatch the data types. |
| Include/Exclude Flavor | Bit flags that control when a field should or should not be included in the mapper. See below for more information |
| Version/ Min Version/ Max Version | Version control properties that determine when a field should be in the mapper, or not. Used to manage addition or removal of features based on a version. See below for more information. |
| Attributes | The numeric value of the attributes for the field. See above for more information. |

### Properties (Property Sheet)

The properties on the property sheet provide an extensible way of adding custom properties to an object and also provide a context sensitive capability. For example, there are a common set of properties for all fields, like Table, ColumnName, but there are additional properties that are visible or hidden depending on the type of the field the properties are related to.

This section will discuss the common set of properties for all field types and the next section will discuss the field type specific properties available.

### Common Properties (Property Sheet)

| Property | Description |
|---|---|
| AccessKey | Not used in web UI |
| AggregateType | The aggregate function to use when calculating row values. If not specified, then Sum is used. If you want to specify aggregation, then set one of the Mapper's AggregateDisplay property attributes. This is only going to be valid for fields containing numeric values. |

## Common Properties (Property Sheet)

| Property | Description |
|---|---|
| AliasName | This provides a way for the mapper to perform aliasing in the select statement if there is one field in the table/view that you want to refer to in two separate mapper fields. For example, in a view there is one date field, “relevant_date”, but in the mapper we want to display separately, a date and time portion. The mapper has two fields “relevant_date”, “relevant_time”. You would set up the relevant_time field to have an alias to “relevant_date” in the mapper. When the mapper selects from the view, it will build an alias clause “relevant_date as relevant_time”.<br><br>When you do define two fields to be aliased, one of those fields must not have a TableName property. If you do, any insert or update statement would contain two instances of the same field name and is ambiguous. |
| CalculatedValue | Javascript expression that represents the value for this field. The expression may include field references. The fields must exist on the page for the field references to function. |
| Caption | Used as the display text for the fields lable. The value of this property may be a plain string, or refer to a vocabulary item using the !fnVocab() syntax. See section on Embedded Functions. |
| CellTypeAttributes | Provides a set of behaviour attributes specific to a cell type. See below for details on what attributes are available for each specific cell type. |
| ColumnName | The actual column name in the underlying table. If you want a mapper to be able to write data to the database, then it must have a column name associated with the field. If there is no column name for a field, the field is read only. There is no validation of the ColumnName at design time. |
| ControlName | The name of the control in the UserControl, to bind to the field. By default it’s the keyname. |
| CssClass | The CssClass to use. All the cell types have inbuilt CssClasses associated with them and for the most part this is fine. However, you can override the default. Examples of overridden styles are on print buttons where we specify “np” so the print button is not visible when the page is printed. |
| DefaultValue | The default value for the field. On a new record, the default values are populated into the mapper and are displayed to the user. The DefaultValue can be a literal value, or an embedded function that resolves to a string. Additionally the default value will be set when an existing record is updated when the “UseDefaultOnUpdate” field attribute is specified. |
| Description | The description for the field and not used in the web UI. |
| Discrim | The value or field reference to use as the Picklist discriminator. The discriminator is the value that would filter a picklist down to a few records. For example a list of divisions would be discriminated by a company to only show divisions for that company. Picklists are discussed in more detail in the section about picklists. |

## Common Properties (Property Sheet)

| Property | Description |
|---|---|
| FieldBehaviorAttributes | Implemented to collapse Boolean based field properties into one manageable property. This property is a catch all for many different behaviours. |

| Attribute | Description |
|---|---|
| AllowLinkInFindResults | By default, all links within find results are disallowed. You can force a link to be active on a field by field basis. Note that this attribute also activates field links on a WizardPhantomList pages. |
| AllowListWrap | In a list view, if the content of the field is longer than the column width, then the content is wrapped onto as many lines as necessary to display the content. |
| DefaultMakesDirty | Normally the setting of a default value does not make a field dirty, but you can force it to do so if you need a wizard to save a change even though the user has made no modifications on any wizard page. |
| FixedTextboxSize | Render the textbox of a BndLkup style control so that the width of the control is the same whether the control is locked or unlocked. |
| IgnoreRowLock | Override whether the field is locked from mapper level Row Locking |
| LinkContainsParentInfo | When you use the Link… properties to construct a link on a field, you would set this attribute to have the link also include context information about the originator of the navigation (parent info). |
| NoBorderWhenLocked | When the field is locked it should not be displayed with a border. |
| NoColumnResize | Prevents the column from being resized in the list view. |
| NoColumnSort | Prevents the column from being sorted in the list view. |
| ShowHoverSummary | Shows a popup summary when the user hovers over the field, or label. If this is specified you must set the SummaryKey and SummaryMOP properties. The target SummaryMOP must implement a Summary page element. |
| SortAndGroup | Set this if you want the list to group on the sort field when the user clicks on the column header to sort. |
| SortDescInit | If the field is sorted, this property menas the property should be sorted descending when first clicked. |
| StaticContentFromDefault | The value of the field is statically populated from the fields DefaultValue. Even if the field is editable and modified, the default value is written to the database. |
| TrimLeft | Trim leading whitespace from field when value is set. |
| TrimRight | Trim trailing whitespace from field when value is set. |

## Common Properties (Property Sheet)

| Property | Description |
|---|---|
| FilterOptions | Modifies the way in which the field can participate in filtering. |

| Attribute | Description |
|---|---|
| DefaultContains | If not explicitly specified, a search by default with use a "contains" criteria instead of the normal "starts with" critieria |
| DefaultHideInList | By default this field should not be shown in list views, but it can be restored/removed from the list view via the Action menu commands. It provides a way to cut down on the number of columns initially displayed, yet allows users to customize (add back in) the columns they want to see. |
| DefaultShowInList | By default this field should be shown in list views, but it can be removed/restored from the list view via the Action menu commands. |
| Disabled | The field cannot be filtered. |
| Filterable | Overrides any option that might disabled filtering. When set, filtering is always allowed. |
| PreFilter | The field will be used to construct a 'pre-filter'that will be provided to a stored procedure, or UDF query. The stored procedure definition is specified in the QuerySQL property of the mapper and you would write<br>mySP [[FILTER]], [[PREFILTER]], [[SORT]], [[OPTIONS]], <embedded func1>, <[field ref1>, etc.<br><br>Any fields marked with the PreFilter option (with a filter value) will be collected in to the [[PREFILTER]] parameter. All other field filters, page filters, etc will go into the [[FILTER]] parameter. It is up to the writer of the stored procedure to make use of both the [[FILTER]] and [[PREFILTER]] parameters in the appropriate places. |
| Required | When filtering, a filter criteria must be provided for this field. |
| TextFilterWidget | Use a simple text filter widget instead of a cell type specific widget. For example the default for a field with a picklist is to popup a widget containing the available picklist items. |
| UnlimitedWidgetList | Do not limit the number of items in a picklist filter widget. The default limit is 120. The unlimited widget actually has a limit of 100,000 |
| UserFilterDefault | Use the fields DefaultValue as the default filtering value. For Find and MultiFind criteria. |

| | |
|---|---|
| Format | The .Net data format for the field. |
| Group | Specified the name of a group to which this field belongs. If the name corresponds to the key of a GroupBox field in the mapper, then the fields will be rendered within that GroupBox. |
| GroupingSpec | Specified how rows are to be grouped when grouping in the datasheet. This is mainly used for numeric and date fields. You can specify a semi-colon delimited list of values in which to group the values into buckets. e.g. "1000;5000;15000", which defines four buckets: 0-1000, 1001-5000, 5001-15000, 15000+ |
| HeaderIcon | Provides an icon in the header of the list instead of using the caption for the header column |
| HideFlavor | Provides the ability to hide a field based on the flavour of the mapper. Flavours are discussed in detail, below. |
| IgnoreRowLock | Don't use. Use the FieldBehaviorAttribute flag instead. |

## Common Properties (Property Sheet)

| Property | Description |
|---|---|
| KeyFormat | Used for picklists when the underlying data for the key is in a format may not be natively recognized. For example a DateTime picklist |
| ListOptions | Affects the behaviour of fields in list views<br>**Attribute** — **Description**<br>LinkIfTruncated — Provide a link to the detail record if content is truncated.<br>RestrictLength — Restrict the length of the content (current to approximately 500 characters).<br>RetainLineBreaks — Retain line breaks in the content in list views.<br>Trim — Trim leading and trailing whitespace. |
| LockFlavor | Provides the ability to lock a field based on the flavour of the mapper. Flavours are discussed in detail, below. |
| Max | Maximum allowed value for a field that may include field references, although typically for numeric fields |
| MaxLength | The maximum number of characters accepted in a field. When the limit is reached, no more characters can be entered. The MaxLength property is normally populated during the slurp process. |
| MaxLengthTruncate | If the MaxLength is specified, when the field is saved, the length is automatically truncated to the maximum length. |
| Min | Minimum allowed value for a field that may include field references, although typically for numeric fields |
| OnChange | Javascript placed in the control's onchange event. |
| ParentField | The name of the field in the parent mapper when filling default values. Used when the DefaultFromParent attribute is set on the field |
| PropertyName | Overrides the default name of the property in the generated typed mapper if the key name is the same as a .NET reserved word. |
| SortDescInit | Don't use. Use the FieldBehaviorAttribute flag instead. |
| SourceField | Another field in the mapper from which to copy any non-null value on insert. Typically used on a foreign key to identify the corresponding join value. |
| Style | Any HTML style expression to include on the element |
| SummaryKey | The RowKey for a HoverSummary |
| SummaryMOP | The MOP/Page to use as the HoverSummary that contains a Summary Page Element |
| Table | The underlying table for the field. The table property is tied to the editability of the field in a mapper via the primary key, as explained. But sometimes you may want to add a field to the mapper that is not actually in the database, or is in the database, but you don't want the value entered to be saved to the database. In this case there is a special table property value called "+" which means no table store but allow it to be editable. For such fields (if not actually in the database) the field should be marked with the "Exclude From Select" attribute. If you don't care if the field is editable and the value isn't saved to the database, you can omit a table property, or use a "-" and the field will be locked. |
| TagAttributes | HTML tag attributes to be added literally to the tag. |

## Common Properties (Property Sheet)

| Property | Description |
|---|---|
| Timezone | The timezone format. Leave blank to store the time as written. The majority of cases where you want a timezone specification is to alays store the time in GMT and then display the time in the user's local timezone. For this, you would specify 0,. as the property value. |
| ToolTip | A string that displays a tooltip to the user when hovering over the control. |
| ToolTipOptions | Options specifying how the field's tooltip is to be presented. |

| Attribute | Description |
|---|---|
| NoBubbleTipIfLocked | |
| PermAggBubbleTip | Add a permanently visible bubble tip to this field's aggregate cell. Ignored when no aggregate shown. |
| PlaceTipBelow | When combined with ShowTipInPlace, specifies that the tooltip should be displayed below its field. |
| PlaceTipOnRight | When combined with ShowTipInPlace, specifies that the tooltip should be displayed to the right of its field. |
| ShowInList | This field's tooltip should also be displayed in list views. |
| ShowTipInPlace | The tooltip should be displayed in place on the page |
| SuppressTip | The tooltip should be suppressed. |
| TipOnTextBox | On BndLkup based controls, the tooltip is associated with the textbox, rather than the icon button. |

| | |
|---|---|
| ValidationRegEx | A regular expression used to validate the field value. The field will be considered valid if the expression matches. |
| ValidationScript | A custom JavaScript validation function for the field of the form func(fld, val, full, caption) returning an error message string if validation failed, else a null or blank string. Just specify the name of the function in the property value. |
| ValidationText | The text to display in the case of a field value validation failure |

## Cell Types

Defines how the field looks and behaves. The following tables describe the type of field that are supported. Listed with each field type are the specific Cell Attributes and Field properties associated with that cell type. Some of the fields are links to online help

| Button Cell Type | | |
|---|---|---|
| Cell Attributes | AjaxClick | Use AJAX to fire the button click event on the server instead of posting the page back. |
| | AjaxClickAndClose | Use AJAX to fire the button click event on the server, instead of posting the page back and then close the page. |
| | AlignLeft | Align the button to the left instead of centering it |
| | DisabledOnClick | The button should be disabled after being clicked |
| | IconButton | The button should be presented as an icon button, rather than a text buttonor link. If an IconButton is specified, then the Icon property should also be specified, pointing to an image file |
| | LockPerMapper | Normally buttons are not locked on existing records if the mapper has DenyUpdate specified. Set this attribute to override that behaviour. |
| | Popup | The results of the click should be displayed in a new popup window. |
| | Postback | Perform a postback on a button click even though a ClickScript is specified. |
| | Requery | The button click handler affects the mapper's underlying table/view data and a post click requery is needed. |
| | ValueAsButtonText | The fields value, rather than its caption should be used as the button text. |
| | ValueDisabled | The button should only be enabled when the field's value is null, blank, or zero. |
| | ValueEnabled | The button should only be enabled when the field's value is not null, blank, or zero. |
| | ValueHidden | The button should only be visible when the field's value is not null, blank, or zero. |
| Field Properties | ButtonType | The way in which the button should be rendered in the list view and detail view.<br>ListButtonDetailButton<br>ListButtonDetailLink<br>ListLinkDetailButton<br>ListLinkDetailLink |
| | Icon | Specified the icon image to display when the Button is using the IconButton CellTypeAttribute. |
| | LinkKey | The rowkey to the related item. |

| Button Cell Type | | |
|---|---|---|
| | LinkMop | The page to link to. |
| | LinkSource | An expression that resolves to a link for a button. The expression can contain field references to refer to value of field in the mapper.<br><br>`~/Handler.ashx?req=nav&mop=candidate_activity!detail&pk=[activity_id]` |
| | OnClick | JavaScript placed on the button's onclick event. By default no postback is performed. Set the PostBack CellTypeAttribute if a postback is also desired. |
| | WindowOpenOptions | Options controlling the format of new browser windows, opened by the click of the button. |

| Attribute | Description |
|---|---|
| AtEventXY | Position the window at the X, Y coordinate of the current event. |
| AtRight | Position the window at the right side of the screen. |
| Center | Vertically and Horizontally center the window on the screen |
| CenterHorizontal | Horizontally center the window on the screen |
| CenterVertical | Vertically center the window on the screen. |
| FullScreen | Full-screen mode (IE 4+ only) |
| Location | Whether to display the address bar, which enables users to navigate the window to a new URL |
| MenuBar | Include the menubar on the window |
| Resizable | Make the window resizable |
| Scrollbars | Include scrollbars on the window |
| Standard | Standard browser settings |
| Status | Include the status bar on the window |
| Toolbar | Include the toolbar on the window. |

| Checkbox Cell Type | | |
|---|---|---|
| Cell Attributes | InvertSense | The checkbox sense is inverted from the data (e.g., a field that stores the enabled state, but whose text is 'disabled'). Therefore a checked value would be stored in the database as false value and vice versa. |
| | TextOnLeft | The checkbox's text should be placed to the left of the checkbox. |
| Field Properties | FalseValues | A semi colon delimited list of values that cause the checkbox to treat the underlying value as False |
| | TrueValues | A semi colon delimited list of values that cause the checkbox to treat the underlying value as True |

| ComboboxCell Type |
|---|

| Combobox Cell Type | | |
|---|---|---|
| Cell Attributes | AllowListWrap | Allow the field to wrap in list views |
| | DynamicValues | Allow user to enter values not found in the picklist while still using a standard combobox |
| | LockNoDiscrim | Control should be locked when there is no discrim value |
| | NoPicklistEdit | Do not allow users to edit picklist values. This overrides the AllowUserAdd/Update/Delete picklist attributes. |
| | NumericDropdown | The picklist values will be generated per the field's Min/Max properties |
| | PicklistOptional | Picklist values are optional and the user should be allowed to enter free-form entry or pick from list. |
| | Typeahead | Provide improved typeahead (vs. standard HTML SELECT) using JavaScript |
| | UpdatePicklist | When DynamicValues is specified, new user entered values are automatically added to the picklist. |
| Field Properties | Discrim | The value or field reference to use as the Picklist discriminator. |
| | Max | Maximum allowed value for a picklist range if NumericDropdown attribute specified. May include field references. |
| | Min | Minimum allowed value for a picklist range if NumericDropdown attribute specified. May include field references. |

| Currency Cell Type | | |
|---|---|---|
| Cell Attributes | LCID | The currency type is specified as a Windows Locale ID (LCID) rather than a .Net culture name. e.g 2057 instead of en-GB, or 1033 as en-US. When specifying currency fields, it is required to set the Culture property to either hard code the culture value, or specify a field reference to a field containing the appropriate culture id for the record. |

| Currency Cell Type | | |
|---|---|---|
| | UseNumberCulture | Format the number part of the currency value in the user's current Number culture setting, instead of the number culture associated with the currency symbol. If this attribute was not set, then in a list of currencies, you might see the formats..<br><br>`£ 1,000.34`<br>`€ 1.000,34`<br>`CHF 1'000.34`<br><br>With the attribute set (and assuming the person logged on preferred UK English number format), they would see<br><br>`£ 1,000.34`<br>`€ 1,000.34`<br>`CHF 1,000.34` |
| Field Properties | Culture | The Windows locale ID (LCID) or a .Net culture name for the currency. The value can be specified directly or using a field reference. |
| | Watermark | Text to display in the field when the field value is not set. |

| DatePicker Cell Type | | |
|---|---|---|
| Cell Attributes | LockTextBox | Lock the cell's Textbox. This forces all input via calendar widget |
| | NoLiveLook | Do not present the button with a 'live' look where the button image livens when user hover's over it. |
| Field Properties | Watermark | Text to display in the field when the field value is not set. |

| EmailAddr Cell Type | | |
|---|---|---|
| Cell Attributes | ContentAsLink | Use the content itself (rather than the lable) to serve as a the link when the field is locked. |
| | EmailForm | Use the standard Email form (which supports journaling)instead of the mailto protocol |
| | HideFrom | Hide the From field on the standard email form when creating emails, so user cannot override the normal sender address. |
| | NoMailIcon | Do not show the mail icon on the mail link. |
| | PlainText | Use plain text (vs HTML) in email body. |

| EmailAddr Cell Type | | |
|---|---|---|
| | UseServiceDefaultSender | Use the email service's default sender address, not the user's email address when originating email. |
| Field Properties | Bcc | The literal text or field expression to use as the BCC. |
| | Body | The literal text or field expression (or combinat) to use as the Body |
| | Cc | The literal text or field expression to use as the CC |
| | EmailFromField | The field (not expression) to use as the from address when sending an email |
| | EmailNameField | The field (not expression) to use as the display name when sending an email |
| | Subject | The literal text or field expression (or combination) to use as the Subject |
| | To | The literal text or field expression to use as the BCC. Note that names should be delimited with semi-colons |
| | Watermark | Text to display in the field when the field value is not set. |

| FilePath Cell Type | | |
|---|---|---|
| Cell Attributes | DelFileOnFileUpdate | Delete the related file when a new file is specified. |
| | DelFileOnRecDelete | Delete the related file when the record is deleted. |
| | FileInDB | The uploaded file is stored in the database vs. a path to the file. |
| | NoFileExtDisplay | Do not display the file extension to the user. |
| | NoLiveLook | Do not present the button with a 'live' look where button image livens when user hovers over it. |
| | NoTextBoxLock | Do not lock the cell's text box. |
| | NoUploadDialog | Do not present a popup dialog box for the user to select an upload file. |
| | NoUploadHints | Do not provide user with upload performance hints in the file browser dialog. |
| | OpenInline | Open the file inline in the application instead of as a standard getf. |
| | OpenInlinePopup | Open the file inline in the application in a popup window instead of as a standard getf. |
| | PhysicalPath | The folder specified for this field is a physical path, not a virtual directory. |
| | ReadableFileName | When storing the file on the server, use a readable filename(while still making filenames unique). |

| FilePath Cell Type | | |
|---|---|---|
| | ResizeUploadedImage | If an uploaded file is an image, the image will be resized and/or an equivalent thumbnail will be created. |
| Field Properties | CreateThumbnail | Determines whether a thumbnail should be created for an image. The default is to always create a thumbnail. |
| | DontResizeSmallerThan | Don't resize images that are smaller than the given size in kB. Default is 120kB. |
| | FileDBField | The binary field in the mapper that stores the file. |
| | FileMimeTypeField | The name of the field specifying the mime type of the content. If not specified, defaults to application/octet-stream. |
| | FilePathField | The name of the field specifying the file name. |
| | Folder | The folder to store the files. Ignored if FileInDB attribute set. Normally this property is the virtual folder, but defines the physical folder path if the PhysicalPath attribute is set. |
| | ImageDBField | The binary field in the mapper that stores the resized image. |
| | ImageFormat | The format of the output image. The default format is jpg. The supported image formats are:- bmp, emf, exif, gif, icon, jpg, png, tif, wmf |
| | ImageOutputPath | The path where converted images are to be located after conversion. |
| | ImageQuality | The quality of the converted image. The default quality for images is Medium. The quality is related to the amount of image compression. |
| | ImageSize | The size of the output image. The size is constrained on the length of the longest side. The available sizes are small (640), medium (800), large (1024), very large (1280). The default size is large. |
| | MaxFileKB | The maximum upload file size in kB. |
| | ThumbDBField | The binary field in the mapper that stores the thumbnail image. |
| | ThumbFormat | The format of the thumbnail image. The default format is jpg. The supported image formats are:- bmp, emf, exif, gif, icon, jpg, png, tif, wmf |
| | ThumbPath | The relative path to the output path where thumbnails are created. The default value is a subfolder called Thumb. |
| | ThumbQuality | The quality of the generated thumbnail. The default quality is High. |
| | ThumbSize | The size of the thumbnail image. The size is constrained on the length of the longest side. The available sizes are small (50), medium (100), large (200), very large (300). The default size is large. |
| | Watermark | Text to display in the field when the field value is not set. |

| Find Cell Type | | |
|---|---|---|
| Cell Attributes | AutoSuggest | (Find Criteria) Auto suggest selection based on partial entries typed by user. |
| | CascadingCombos | (Find Criteria) Use cascading combo boxes, instead of text boxes. Combobox fields should have discrim values set so each parent field can discriminate the picklist of the child combobox. |
| | HideTextBox | (Find Control) Hide the Find's text box. Ths is intended for use when the Find is used as a client-side programming element. |
| | NoLabelLink | (Find Control) Do not make the field lable a link when there is a related item. |
| | NoLiveLook | (Find Control) Do not present the button with a 'live' look where button image livens when user hovers over it. |
| | NoUserSort | (Find Results) Do not allow user to choose a sort order when finding records. |
| | RepeatedSelected | (Find Action) Allow user to select an item then repeatedly select additional items. Normally when an item is selected from the find results, the Find window closes immediately. |
| | ShowLockedButtons | (Find Control) Show locked buttons (which are normally hidden). This is typically used when the control is locked/unlocked in script. |
| Field Properties | FindDefaults | (Find Criteria) A semi-colon delimited list of field=value pairs specifying default search criteria. The values may include field references and/or embedded functions. |
| | FindFields | (Find Action) A tag string of field in this format <destination field>=[source field]; <destination field>=[source field], etc. If the destination and source are the same you may omit the source field. |
| | FindFilter | (Find Results) Filter to apply to the Find results in addition to any find criteria specified by the user. |
| | FindFlavor | (Find Criteria/ Find Results) Flavor to apply to both find criteria and find results |
| | FindKey | (Find Control) The key name of the field that provides a rowkey value to the related item. Used in providing a navigation link to the thing you're trying to find. |
| | FindMOP | (Find Control/FindCriteria) This provides two functions. It defines which mapper is going to be used for the find (looked up via the page) and also the target Page to which you can navigate to look at the thing you've found. |

| Find Cell Type | | |
|---|---|---|
| | FindNoSearch | (Find Criteria) A semi-colon separated list of fields, normally searchable in the search mapper, that should not be searchable in the Find Critieria (that is, they are hidden from the user). Note that each field can include an =<text> qualifier for display purposes. |
| | FindPageInstanceValues | (Find Results) A semi-colon delimited list of fields that exist on the same page as the find control. The value of these fields on the same page instance will be passed to the find dialog and potentially used for filtering. References to these instance value can be placed on FindFilter criteria using {} braces instead of the usual reference [] brackets. Standard reference syntax applies. |
| | NoTransaction | (Multi-find Action) By default, the multi-find select action puts a transaction around the insert of items. Set this property to skip having transactions around the multiple inserts. |
| | Watermark | Text to display in the field when the field value is not set. |

| GroupBox Cell Type | | |
|---|---|---|
| Cell Attributes | AllowListWrap | When ShowInList is set, allow items in this field to wrap in list views. |
| | DynamicLayout | Any controls contained by this grouper are laid out dynamically, ignoring normal positioning. |
| | Popup | Show fields in this group, only in a popup window |
| | ShowInList | Use this grouper to group its fields in a single column in list views |
| | SizeDynamically | Size the group box to its contents ignoring any layout sizing. |
| Field Properties | | |

| Html Cell Type | | |
|---|---|---|
| Cell Attributes | AllowListWrap | Allow this field to wrap content in list views. |
| Field Properties | Height | The height of the control expressed in HTM units. |

| Icon Cell Type | | |
|---|---|---|
| Cell Attributes | HideIfMissing | The icon should be hidden if the image file can't be found. |
| Field Properties | | |

| Label Cell Type | | |
|---|---|---|
| Cell Attributes | NoHtmlEncode | The text should not be HTML encoded. Used when the content of the field is HTML and you want it to be displayed based on the HTML encoding |
| | StaticFromCaption | Always use the field's caption text for the field's display text. |
| Field Properties | | |

| Link Cell Type | | |
|---|---|---|
| Cell Attributes | ButtonInDetail | Present the cell as a button in detail views instead of a link |
| | ButtonInList | Present the cell as a button in list views instead of a link |
| Field Properties | LinkKey | The keyname of the field providing the rowkey of the item to navigate to. |
| | LinkMOP | The page to navigate to. |

| Listbox Cell Type | | |
|---|---|---|
| Cell Attributes | MultiSelect | Set the list box into multiselect mode. In multiselect mode the list box stores the selected values as a semi-colon delimited list. |
| Field Properties | Rows | The number of rows to display in a list box. This can be a non-integer value e.g. 1.5 |

| MapperSummary Cell Type | | |
|---|---|---|
| Cell Attributes | | |
| Field Properties | Filter | The filter to use on the summary's mapper. The filter may contain references to the fields on the current mapper. |
| | Flavor | The flavor to use on the summary mapper. The summary flavor will always be included. |

| MapperSummary Cell Type | | |
|---|---|---|
| | HTMLSummaryFlags | Options controlling the look and feel of the summary. See the section on using Summaries for more information. |
| | Mapper | The mapper to use for the summary. |

| Attribute | Description |
|---|---|
| HeaderOnly | Include the header captions only (applies to horizontal layout only). |
| Horizontal | Summary of the mapper will be laid out horizontally like a list view |
| IgnoreTopN | In horizontal layout, ignores the TopN rowcount and summarizes all the rows of the mapper. |
| IncludeBlankValues | Include blank values in the summary (does not apply to horizontal layout where blank values are always displayed). |
| InitOpen | The summary should be initially open (not collapsed). |
| LockedOnly | Include only locked fields |
| NoFieldStyleOverride | Don't override default summary style with field specific styles. |
| NoHeader | Do not show any header captions in horizontal layout. |
| OmitDefaultValues | Do not include default values in the summary (does not apply to horizontal layout). |
| OneRow | In horizontal layout, forces the summary to be only one row. Either the current row, or the first row. |
| Vertical | The Current, or first row of a mapper will be laid out in a vertical orientation |

| MultiSelect Cell Type | | |
|---|---|---|
| Cell Attributes | CheckBoxWidget | Use the checkbox widget instead of the multi-select list widget. |
| | LockTextBox | Lock the cell's textbox to force all input via the picklist widget. |
| | NoLiveLook | Do not present the button with a 'live' look where button image livens when user hovers over it. |
| | NoPicklistEdit | Do not allow users to edit the picklist values. This overrides the AllowUserAdd/Update/Delete picklist attributes. |
| | OpenWidgetOnLockedClick | If the field is locked and the user clicks on the text box, pop up widget as though the button was clicked. |
| | SizeToList | Size the selection widget ti fit the list. |
| Field Properties | | |

| Password Cell Type | | |
|---|---|---|
| Cell Attributes | NoAutoComplete | Disables the browser's built in Auto Complete feature. |
| Field Properties | | |

| Picture Cell Type | | |
|---|---|---|
| Cell Attributes | AlwaysExtract | If the image is stored in the database, extract the image on each access and don't cache it. |
| | ImageInDB | The image is stored in the database vs a path to a file. If the image is stored in the database, it is stored in this field which must be of type binary. For an image to be stored to a file, the field is a string containing the path. |
| | ScaleToWidth | Scale the image to the column width. |
| Field Properties | ImageType | They type of image stored in the database (bmp, jpg or gif). |
| | PrefixBytes | Number of bytes to ignore when extracting an image stored in the database. (Images in Northwind had a set of prefix bytes that had to be discarded before being displayed). |

| RadioButtons Cell Type | | |
|---|---|---|
| Cell Attributes | Horizontal | Radio buttons should be laid out horizontally (left-to-right) instead of vertically (top-to-bottom). |
| | NoBorder | The list of buttons should not have an border. |
| | NoHtmlEscape | The text of the buttons should not be escaped for HTML. |
| | NoScroll | The field should not be scrollable. |
| | SizeDynamically | If border is used, size the border to the RadioButtons ignoring any layout sizing. |
| Field Properties | | |

| RatingIcon Cell Type | | |
|---|---|---|
| Cell Attributes | DualImage | Use separate enabled/disabled images (disabled image name assumed to be enabled image name with 'x' appended). |
| Field Properties | ImageFile | The relative path to the rating icon image file to use in the "on" state. |

| TabStrip Cell Type | | |
|---|---|---|
| Cell Attributes | | |
| Field Properties | | |

| TextBox Cell Type | | |
|---|---|---|
| Cell Attributes | AllowListWrap | Allow this field to wrap in list views |
| | AutoResizeHorizontal | Auto-resize the field's control horizontally when the window is resized. Currently applies only to popups and is not supported in Opera. |
| | AutoResizeVertical | Auto-resize the field's control vertically when the window is resized. Currently applies only to popups and is not supported in Opera. |
| | AutoSuggest | Auto suggest selections based on partial entries typed by user and matching distinct values in underlying column |
| | AutoSuggestFilter | In the FBF row, auto suggest selections based on partial entries typed by user and matching distinct values in the underlying column. |
| | Multiline | The control should allow multiple lines of text. |
| | NoAutoComplete | Disables the browser's built-in AutoComplete feature. |
| | NoFilterPicklist | Do not use the picklist filter widget for this fielf in the datasheet. Only applies when field has a picklist. |
| Field Properties | Rows | The number of rows to display in a multi-line text box. This can be a non-integer value, e.g. 1.5. |
| | Watermark | Text to display in the field when the field value is not set. |

| TextWithWidget Cell Type | | |
|---|---|---|
| Cell Attributes | LockTextBox | The text portion of the control should not be editable. |
| Field Properties | Watermark | Text to display in the field when the field value is not set. |

| TimeDuration Cell Type | | |
|---|---|---|
| Cell Attributes | | |
| Field Properties | MinuteInterval | The interval, in minutes, between selectable minute values. |

| TimePicker Cell Type | | |
|---|---|---|
| Cell Attributes | LockTextBox | Lock the cell's text box. This forces all input via the time widget. |
| | NoSeparators | Do not show separator characters, if any, between time part controls. |
| | NoTextBoxOnLock | Unless set, the TimePicker renders as a locked TextBox when locked. |
| | UseCombos | Use combo boxes for hour/minute/second selection instead of time widget. |
| Field Properties | MinuteInterval | The interval, in minutes, between selectable minute values. |
| | Watermark | Text to display in the field when the field value is not set. |

| Tree Cell Type | | |
|---|---|---|
| Cell Attributes | NoBorder | The tree should not have a border |
| | NoCounts | The grouper items should not have counts. |
| | NoScroll | The field should not be scrollable |
| | NoWrap | Do not wrap text. |
| Field Properties | CssHoverNode | The CSS class to use as the style for rendering a tree node when the mouse pointer is positioned over it, default is TreeHover. |
| | CssLeafNode | The CSS class to use as the style for rendering leaf nodes, default is TreeLeaf. |
| | CssNode | The CSS class to use as the default style for rendering tree nodes, default is TreeNode. |
| | CssParentNode | The CSS class to use as the style for rendering parent nodes, default is TreeParent. |
| | CssRootNode | The CSS class to use as the style for rendering the root node, default is TreeRoot. |
| | CssSelectedNode | The CSS class to use as the style for rendering selected nodes, default is TreeSelected. |

| Tree Cell Type | | |
|---|---|---|
| | ImageSet | The group of images to use for the TreeView control. See TreeViewImageSet in the .Net documentation |

| Url Cell Type | | |
|---|---|---|
| Cell Attributes | LinkOnly | Render field with link only, no TextBox to allow a user to edit the URL. |
| | NoNavIcon | Do not show a navigation icon for popup or external links. |
| | NoPopup | Do not popup a new window as the target for navigation, navigate in place. |
| Field Properties | MinuteInterval | The interval, in minutes, between selectable minute values. |
| | Watermark | Text to display in the field when the field value is not set. |
| | WindowOpenOptions | Options controlling the format of new browser windows, opened by the click of the button. |

| Attribute | Description |
|---|---|
| AtEventXY | Position the window at the X, Y coordinate of the current event. |
| AtRight | Position the window at the right side of the screen. |
| Center | Vertically and Horizontally center the window on the screen |
| CenterHorizontal | Horizontally center the window on the screen |
| CenterVertical | Vertically center the window on the screen. |
| FullScreen | Full-screen mode (IE 4+ only) |
| Location | Whether to display the address bar, which enables users to navigate the window to a new URL |
| MenuBar | Include the menubar on the window |
| Resizable | Make the window resizable |
| Scrollbars | Include scrollbars on the window |
| Standard | Standard browser settings |
| Status | Include the status bar on the window |
| Toolbar | Include the toolbar on the window. |

| Website Cell Type | | |
|---|---|---|
| Cell Attributes | ShowProtocol | Show the site's protocol (e.g. http:) if present in the URL. |
| Field Properties | TargetFrame | The frame to display the content of the link. |
| | URL | The literal text or field expression ([field_name]) (or combination) to use as the URL. The default is to use the contents of the field. |

## Permissions

Permissions on mappers and mapper fields (or any object really) provide a simple way to modify behavior based on the role of the person who is logged into the application. You have to be careful with permissions and understand what permission options are there before you check or uncheck settings.

Also, our permissions structure is deemed "permissive" rather than "restrictive". Meaning that for any new object created, by default, everyone can see that new object and/or has full access to it. At a mapper level, the permissions for a mapper are easy to understand. For each role, can the user Insert, Update, Delete, or Read?

For fields the options are Read, Write, Configuration, Navigation.

### Read/Write

Read permission is obvious if checked the user can read a field. But if a user can read, but not write, the field is locked. If a user can Write, but not read, it means the field is editable, but it acts like a password type field where the text entry is obscured. If a field is neither read, nor write the field becomes hidden.

### Configuration

This permission is rarely used. Some picklists are allowed to be edited and added to dynamically. A field with such a picklist may only be allowed to be edited/appended for a certain user role so Configuration would be turned off for the users not allowed to edit.

### Navigation

Some fields come with a navigation link. E.g., find fields, File path fields. If navigation is turned off for a user role, then the link is not constructed for that user role. On a button, the Navigation permission controls whether the button fires a click event for a particular role. This may not seem too intuitive but a button can perform either a command, or a navigation, so it was thought appropriate to not create yet another permission specifically for buttons.

## Pages

A NetQuarry page roughly corresponds to a page in the application. Unlike traditional ASP or ASP.Net application, there is generally no source file corresponding to the page. Instead, the page is specified in the NetQuarry meta-data by a Page object. A page object can be built on top of a Base Page. A Base Page is simply a meta-data page definition that is used like a base class for other pages. Each Page has a number of attributes:

- Module – The module to which the page belongs. Each module corresponds to a single .meta file.
- Name – The name of the page.
- MOP – The Module/Page is formed from the Module and page Name in the form <module>!<page>. The MOP is the standard navigation target in NetQuarry applications.
- Moniker – A page name allowing differentiation (for studio users) among similar pages.
- Mapper – The primary mapper used for the page.
- Priority – The page priority in the case where a particular user has access to multiple pages with the same MOP. When such is the case, the page with the highest priority is used for that MOP.
- Template – The template specifies how the page is layed out.

Like most significant NetQuarry objects, pages carry permissioning allowing page access to be specified by user role. Navigation throughout a NetQuarry application is specified by navigation to a MOP (Module/Page). This, in conjunction with page permissioning, allows an application to be build with "smart" navigation because multiple pages can be created with the same MOP with access controlled by role. So, for example, a company record could be displayed differently to internal users vs. external users. In some cases a user will have access to multiple pages with the same MOP. For example, imagine a user!detail page for users that includes basic user information and a user!detail page for administrators that includes additional information where admin users have both a "user" role and an "admin" role. In such cases the page with the highest Priority (in this case presumably the admin page) is used.

### Templates

A Page object requires that a Template be specified. The template specifies how that page is layed out. The NetQuarry platform provides a relatively small number of predefined templates. Each template is a .aspx file. It is possible to develop custom templates, however, custom templates are not common and are not covered in this document. The standard NetQuarry templates are:

- TabbedSubformTemplate – Includes a top pane, a subformbar with tabs, and zero or more subform panes. Each subform pane itself contains fully functional page (without subforms).

- ExpandedSubformTemplate – Includes a top pane and a subform with inline subform panes. Each subform pane contains a page with limited functionality.
- WizTemplate – Provides for a wizard interface with sequential pages. Each page is specified by a PageElement.
- ConsoleTemplate – Includes multiple panes in configurable columns. Each pane contains a PageElement with limited functionality.
- PopupDialogTemplate – Generally NOT specified on a page, but selected automatically as an override for one of the subform templates when in a pop-up situation.

## PageElements

Each template has one or more "slots". A slot is a place where content can be placed. The content of that slot is in the form of a PageElement. The TabbedSubformTemplate and ExpandedSubformTemplate each have a "Main" slot that holds the top pane and subforms, and a "Side" slot that may hold a navigator such as the NavTree. The WizTemplate has a "Side" slot for a navigator. In addition it has an arbitrary number of WizardPage slots corresponding to pages within the wizard, a single "Side" slot for a navigator, and any number of Summary slots. The ConsoleTemplate has a Main slot hosting the primary element, a Side slot for a navigator, and any number of additional ConsolePage slots.

Each PageElement in the template specifies a renderer component to use when rendering the content of that element. The renderer specified on each PageElement is dependent on the template used:

- TabbedSubformTemplate, ExpandedSubformTemplate:
    - phantomdetail – The standard detail renderer.
    - dsctrl – The standard list renderer.
    - SimpleDetail – A simple detail renderer than can be used when a simple, read-only display is desired.
    - GMap – A GoogleMaps-based map renderer.
- WizTemplate
    - WizPhantomDetail – The standard detail renderer in a wizard.
    - WizPhantomEditList – The standard editable list renderer in a wizard.
    - MapperSummary – Used to provide summaries of related data within the wizard.

- ConsoleTemplate
    - MiniDetail – The standard detail renderer in a console.
    - MiniList – The standard list renderer in a console.
    - MiniNav – A BulletList-like navigation link renderer.
    - Graph – A GooleGraphs-based graphing renderer.
    - GMap – A GoogleMaps-based map renderer.

PageElements also carry permissioning. PageElement permissioning is primarily useful in the ConsoleTemplate and, to a lesser degree, in the WizTemplate. It is not generally very meaningful in the subform templates because it does not add any real benefit beyond the page-level permissions.

### Attributes

As with most other NetQuarry objects, Pages support a variety of attributes that affect page behavior:

- AllowMultiPopup – Allow multiple instances of this form to be open at one time. Applies only when PopupDetailDialog is set.
- HasElements – For internal, programmatic use only.
- NoPaging – Do not provide record navigation buttons or related captioning on the toolbar.
- NotPrintable – This page will not support printing.
- PopupDetail – When a drilldown to detail is performed, pop up the detail in a new window.
- PopupDetailDialog – When a drilldown to detail is performed, pop up the detail in a new window using the PopupDialogTemplate.
- SupportsSearch – This page supports, and should be used for, global searching.

### Extensions

Like Mappers, Pages support extensions which are used to provide application-specific programmatic logic. At this time the only page events that are supported are all specific to the WizTemplate temple:

- WizardPageLoad – Fired when a wizard page is loaded.
- WizardNext – Fired when the wizard “next” button is clicked. The primary purpose of the event is to allow override redirection to another page.
- WizardPrev – Fired when the wizard “previous” button is clicked. The primary purpose of the event is to allow override redirection to another page.

- WizardFinish – Fired when the wizard "finish" button is clicked.
- WizardCancel – Fired when the wizard "cancel" button is clicked.
- WizardDataExchange – Fired when the wizard "next" button, or "finish" button is clicked. The primary purpose of this event is to get access to the data from a page just after the data has been transferred from the page to the mapper and into the UserData collection. When this event is fired, the mapper and user data has all the data entered from that page.

## Navigation and Commands

Each page supports several mechanisms for navigation, and one for general commands.

### Subform

The Subform property is available for all template types, but really only meaningful on the subform templates (TabbedSubformTemplate and ExpandedSubformTemplate). The subform is a reference to a Subform-type Navigator that the template should use for constructing its subform. The subform is used to present pages related to the current item which is rendered in the top pane of the page. The item in the top pane is the parent object. The items in the subform are the child objects. For example, the top pane might display a company while the subforms display related people, related branch offices, related addresses, etc.

For a detail form this relationship is probably as expected. What might be somewhat surprising is that this relationship is maintained even when the top pane contains a datasheet. When the top pane is a list, the subform is requeried as the user moves from one row to the next. Note that to maintain the independence of the top and bottom panes, the bottom pane is actually composed of one or more iframes. Also, a draggable horizontal divider is provided to adjust the relative sizes of the top and bottom panes. The position of the divider is persisted per-user, per-page.

The bottom pane in an ExpandedSubformTemplate is a single iframe that is always rendered with data from each subform. The display of each subform can be toggled open or closed (with client-side handling), but the actual rendering pulls data from all related items and can therefore be somewhat expensive. Each subform item is based on a page, but has limited functionality compared to that same page when rendered in a top pane.

The bottom pane in a TabbedSubformTemplate, on the other hand, is rendered with a separate iframe for each subform element. Each iframe is only populated on demand. Typically the first subform element is populated immediately (depending on the *LoadFirstSubformTab* application property which defaults to *true*). Subsequent tabs' iframes are populated only when the user clicks on that tab. This minimizes the server load especially since in the typical case users do **not** click on additional tabs every time they view a page. As a user clicks from subform tab to tab, their corresponding iframes are hidden and displayed. Thus, when a user clicks off a tab and back, no server request is performed, rather the iframe is unhidden. This dramatically reduces the load on the server as users return to previously clicked tabs. The user can double-click on a tab to force a requery. Each subform is itself specified by a page object and fully functional with the ability to insert/update/delete according to that page's

settings. When the top pane is a datasheet and the user moves to a new row, only the currently visible subform is requeried – the others are just cleared. Note that the TabbedSubformTemplate subformbar (the subform tab strip) provides several useful features that might not be obvious at first glance:

- MoveUp icon – Clicking on the MoveUp icon, just to the right of the tab caption, promotes the page in the subform to the top pane. MoveUp can be disabled globally with the *NoMoveUp SubformDisplayAttribute* on the application, or per-subform item on the subform navigator's target.
- Min/Max icon – Alternately minimizes or maximizes the subform.
- Scroller icons – Scrolls the subform tabs left or right one or multiple items to provide access to additional tabs that might not be visible. Note that the scrollers are not provided when the *WrapSubformTabs AppAppearance* property is set.

### Side Slot

As mentioned above, most templates provide for a "Side" slot. The side slot is designed to hold a navigator PageElement. Typically the NavBar renderer is used to render a navigation tree is the side slot. The NavBar PageElement has a Navigator property that specifies which Navigator object to render. Typically the same navigator is used throughout most of the system to provide consistency as the user navigates from page to page in the system. While not widely used, a SearchBar renderer is also provide for use the Side slot. The side slot is separate from the rest of the page by a draggable vertical divider. An open/close toggler is also provided and its state is persisted by user.

### PageCommands

A Page object can support page commands (which may include commands and navigation). Page commands are configured by specifying a Menu Navigator in the page's PageCommands property. Menu navigators can invoke commands passed through to the page's mapper and provide navigation to other pages, external sites, export templates, etc.

### Other Navigation Properties

In addition to navigation provided via the Side slot and PageCommands, several page properties affect navigation behavior.

- DetailTarget – Specifies the MOP to which to navigate when drilling into a datasheet. By default navigation is to the MOP <module>!detail where <module> is the current module. So, for example if the current page is company!list drilling into a list item will navigate to company!detail. By setting a DetailTarget this behavior can be overridden.
- NewTarget – Specifies the MOP to which to navigate when creating a new item from the page. By default navigation is to the MOP <module>!detail (with req=new) where <module> is the current module. By setting a NewTarget this behavior can be overridden.
- DeleteTarget – Specifies the MOP to which to navigate following the deletion of the current item when on a detail page. There is also an application property of the same name which is overridden by any page-level setting. If no application or page DeleteTarget is specified, navigation is to a blank page. When a page-level DeleteTarget is specified, the page's DeleteQueryParams, if any, are added to the navigation request.

The WizTemplate also provides three sets of navigation specifies for particular actions -- Cancel, Delete, and Finish. Below are the properties for the Finish action. There are corresponding properties for Cancel and Delete.

- FinishAction – Specifies the type of action to take upon successful completion of the wizard.
- FinishQueryParams – Query parameters to be added to the destination URL upon successful completion of the wizard. This may include page instance expressions.
- FinishTarget – The target MOP or URL to which to navigate upon successful completion of the wizard.
- FinishTargetInstance – The element instance of the item on which to navigate upon successful completion of the wizard.

### Other Significant Page Properties

Most templates support the following page properties:

- RefreshInterval – If set, the page will be requeried at the specified interval (specified in seconds).
- DetailLayout – The detail layout specification as generated by the detail layout tool.
- DisplayAttributes – Various attribute that specify the presentation of the page:
    - NoCaption – Do not show the caption on the page.
    - NoCheckBoxes – Do not show the item selection checkboxes in the list (applies to datasheet pages only).
    - NoColOrdering – Do not allow users to customize list column order (applies to datasheet pages only).
    - NoCount – Do not show the mapper's record count (applies primarily to datasheet pages). This can dramatically improve performance on large datasheets. Note that the mapper provides a similar attribute.
    - NoDetail – Do not allow navigation to a corresponding detail page (applies primarily to datasheet pages).
    - NoFilterByForm – Do not provide the Filter-By-Form (FBF) filtering row on the list (applies to datasheet pages only).
    - NoFreezing – Do not freeze list row and column headers (applies to datasheet pages only).
    - NoPinning – Do not allow users to "pin" items (applies to datasheet pages only).
    - NoSelection – Do not indicate the selected list row via selector or coloring (applies to datasheet pages only).
    - ProportionalSizing – Use proportional sizing for elements on this page (applies primarily to ConsolTemplate MiniList elements).
- Flavor – The flavor to set on the page's mapper. The flavor can be used to hide/exclude mapper fields in specific contexts.
- RowsPerPage – The maximum number of rows to show in one page when rendering a datasheet.
- ToolbarDisplayAttributes – Various attributes that specify the presentation of the page's toolbar:
    - DenyAdd – Do not provide an "add" button regardless of the page's mapper's permissions.

  - DenyDelete – Do not provide a “delete” button regardless of the page’s mapper’s permissions.
  - DenyInsert – Do not provide a “save” button on new records regardless of the page’s mapper’s permissions.
  - DenyUpdate – Do not provide a “save” button on existing records regardless of the page’s mapper’s permissions.
  - HideButtonsWhenDisabled – Hide any disabled toolbar buttons (rather than showing as disabled).
  - NoRefresh – Do not provide a “refresh” button on the toolbar.
  - NoToolbar – Do not provide a toolbar on the page.
  - PopupNew – Pop up a new window when creating a new item rather than navigating in place.
  - TooltipHelp – Not currently supported.
- PageFilter – A filter for this page’s mapper. This is applied in addition to any other filters on the mapper.
- PageGrouping – A grouping specification that overrides the page’s mapper’s DefaultGrouping, if any.
- PageSort – A sort specification that overrides the page’s mapper’s DefaultSort, if any.
- PageView – A table or view that overrides the page’s mapper’s normal View.
- ClientHtmlBlock – Provides a way to inject raw HTML directly into the page’s HTML. Note that ultimately the .Net Page.ClientScript.RegisterClientScriptBlock () method is called.
- ClientScriptBlock – Provides a way to inject a block of javascript directly into the page’s HTML. The script should not be enclosed in an HTML script tag. Note that ultimately the script is wrapped in an HTML script tag and the .Net Page.ClientScript.RegisterClientScriptBlock() method is called.
- ClientStartupScript – Provides a way to inject, directly into the page’s HTML, a block of javascript to be run at the time the page is loaded. The script should not be enclosed in an HTML script tag. Note that ultimately the script is wrapped in an HTML script tag and the.Net Page.ClientScript.RegisterStartupScript() method is called.

The following page properties are specific to the WizTemplate template:

- CancelScript – Javascript to be set on the wizard’s “cancel” button.
- PromptCancel – Specifies whether or not to prompt for a confirmation when the user clicks on the wizard’s “cancel” button.
- SummaryFlags – Options specifying how the wizard’s running summary should behave. Note that this works in conjunction with the page elements’ WizardPageAttributes.
  - IncludeBlankValues – Include blank values in the summary (applies to detail items only).
  - IncludeDefaultValues – Include default values in the summary (applies to detail items only).
  - IncludeUnlockedFields – Include unlocked fields in the summary (applies to detail items only).
  - InlinePresentation – Present the summary in a simple, inline format.
  - Vertical – Present the summary of each mapper in a vertical orientation.
- SummaryHeight/SummaryWidth – The height and width of the summary expressed in HTML units

The following page properties are specific to the ConsoleTemplate template:

- TemplateAttributes – Attributes specific to the current template:
    - NoElementMove – Do not allow the user to move console panes.
- Columns/ColumnWidths – The number of pane columns to layout in the console, and a semi-colon-separated list of HTML widths for the columns (e.g. “275px;200px”).

## Navigators

There are three types of navigation object that provide ways to navigate from one place to another. Menus, Navbars, Subforms.

### *Adding Javascript, Confirmation, Process Status Indicator*

For all types of navigator, you can specify properties to add confirmation messages. This is specified via JavaScript. JavaScript to be invoked when the target is clicked can be specified for a target using its *Script* property. When an item with javascript is clicked, the script will be executed and the page will **not** be posted back unless the *AllowPostBack CommandAttribute* is set.

Any menu target can include a confirmation message and/or a process status indicator when invoked. To set a confirmation, provide text in the *Confirmation* text item. If confirmation text is provided and an item is clicked, a javascript confirmation box will be presented with the text allowing the user to cancel or continue the action. For actions that might be expected cause a noticeable delay, text can be provided in the *StatusText* text item. If status text is provided and an item is clicked, an animated in-progress indicator will be display along with the specified text until the response page is received. Both confirmation text and status text can be provided on the same item.

### Menus

Menus provide the way to add links to pages in various locations about the page. There are four locations that menus can be located and there are six different types of target to call for each command location.

The three locations are

- MoreMenu (default location if not specified)
- Quicklinks
- Toolbar

The six targets are

- URL
- Page
- Navigator
- HTML
- Command
- Template

Not every combination of location and command are appropriate.

The following matrix gives valid combinations of the two settings.

| | URL | Page | Navigator | HTML | Command | Template |
|---|---|---|---|---|---|---|
| MoreMenu | - | x | - | - | x | x |
| QuickLinks | - | x | - | - | - | - |
| Toolbar | - | x | - | - | x | - |

### *Page*

Navigate to a page. You can specify a page to navigate to from any of the three command locations. You can navigate to another page for two purposes.

1) When you would want to navigate to a wizard, or to a page for creating a new object you'd put a Page target on the MoreMenu. To set this up…

    a. Add the target as a Page target type and specify the MOP

    b. In the target properties set the ParentViewKeySource, ViewKey to indicate how the two pages are to be related.

    c. In the CommandAttributes property, optionally specify whether the navigation should be to an existing record, or to a new record. For a new record, set the NavNew attribute.

2) When you want to navigate to a page that shows related records to the current page you'd put a page navigation on the QuickLinks menu. For example, if you don't want to permanently show a subform under a main page, you can add a link to navigate to the related page and when you navigate, that page would be filtered by the appropriate key values as per subforms. To set this up…

    a. Add the target as a Page target type and specify the MOP

    b. In the target properties set the ParentViewKeySource, ViewKey to indicate how the two pages are to be related.

    c. There's no need to set any CommandAttributes

If you would prefer to add your page navigation to the Toolbar location, i.e. there is a specific toolbar button dedicated for that navigation, specify the Toolbar menu location and importantly, set the location to an image file in the Image32Url property to represents that toolbar navigation. The path to

the image is a url path relative to the application root. If you don't specify the image location, the toolbar item is not presented.

### *Command*

Execute a command through an extension. When you add a command and on the front end, you click on the command, a MapperCommand event is fired to all the extensions on the mapper. It is your job to add a handler for that command.

A command item can be placed on the More Menu and Toolbar. It is not supported on the QuickLinks..

There are a couple of common CommandAttributes that you may want to use for command handlers. These are…

**Existing Only** – The menu only appears on an existing record (not a new record).

**New Only** – The menu item only appears on a new record.

**Requery** – After you've handled the command in your code, the page is automatically refreshed to reflect your changes

**Require One Selection** – When you activate a command on a list view you are required to choose one item and one item only on which to perform the action. On a detail one item is selected by default. If you don't select any items, or select more than one, you are prompted to select only one when you activated the command.

**Require Selection** – When you activate a command on a list view you are required to choose one or more items on which to perform the action. On a detail one item is selected by default. If you don't select any items, you are prompted to select one or more items when you activated the command.

If you don't specify a Require Selection attribute on your command, it allows users to run the selected command against all the records of a list view matching the current filter. If there is no filter applied, that means it would run against all the records in that mapper. Potentially hundreds of thousands of records. Therefore you must decide carefully on how command functionality should be applied.

### *Handling a Command*

If you added a command target then you have to write code in one of the extensions on the mapper to handle that command. Let's assume you added a command to the menu, called "MigrateUsers" . You would code up the extension as follows…

```
public override void MapperCommand(MyCompany.Data.Companies sender, EAPCommandEventArgs e)
{
  switch (e.CommandName)
  {
    case "MigrateUsers":
     UpdateLogonVersion(sender, 4);
     break;
    default:
```

```
      break;
   }
}
```

The cleanest way to do it is create a specific function for each command making the event handler code simpler to read.

```
private void UpdateLogonVersion(MyCompany.Data.Companies sender, int version)
{
    //--- You have to determine whether any rows have been selected.
    //--- If rows are selected, then the count of selected keys will be greater than 0
    //--- If no rows are selected, you will execute this command on all the records
    //--- in the list view
    ArrayList arrKeys = sender.Mapper.SelectedKeys();

    //--- If keys count > 0 create a filter clause based on the list of selected keys
    //--- Use the GetMapperKeyFilter method passing the key field with which to filter.
    //--- If keys count == 0 then use the filter on the list view
    string rowFilter = (arrKeys.Count > 0) ? base.GetMapperKeyFilter(arrKeys, "company_id") :
        this.Mapper.Filters.GetFilter(GetFilterFlags.IncludeAllTypes);


    //--- create your mapper (or typed mapper) on which you want to perform the action.
    //--- Don't use the same mapper object (sender) that is passed to the function.
    //--- Even if it is the same type.
    using (MyCompany.Data.Companies companies =
        MyCompany.Data.Companies.OpenReader(this.Application, rowFilter, 0, MapperAttrs.NoRowRequery))
    {
        while (companies.MoveNext())
        {
            //--- Do work on your mapper or whatever you want to achieve
            //--- If you make mapper changes, remember to save
            companies.Save();
        }
    }
}
```

### *Handling a Command for Navigation*

In some circumstances, it is not possible to define the way you want to navigate to another page just through metadata, so you have to resort to navigating via code. So you handle the mapper command in your extension and then force a navigation to your page. To manually navigate, you use the Navigate method of the Application object.

```
public override void MapperCommand(MyCompany.Data.Orders sender, EAPCommandEventArgs e)
{
 switch (e.CommandName)
 {
  case MyCompany.Data.Assignments.CommandNames.CreateAssignmentButton:

   // need to get the candidate activity for this order.
   string params = string.Format("ordrk={0}&parmop={1}&parrk={2}",
                                 sender.order_id,
                                 sender.Mapper.MOP,
                                 sender.order_id);
   this.Application.Navigate("assignments!wizard", string.Empty, params, "new");
   break;
 }
}
```

More details on manual navigation are provided later.

### *Template*

Perform a mail merge against a fragment of html, or html/xml file containing field replacement codes. To utilize this feature you first have to create a Template and add it to the Templates metadata. Then, you add your target as a Template target type. Specify the name of the Template in the MOP/URL/Command/Navigator column.

A Template target can be located on the MoreMenu, or Toolbar (needs an icon image). The template merge can only work with a single record, so only appropriate for details and if you set the RequireOneSelection.

## Subforms

Subform navigators are consumed by pages using the TabbedSubformTemplate and ExpandedSubformTemplate templates.

Subforms provide a way to show different sets of data related to a parent module.

### *Parent/Child Relationships*

Subform relationships are entirely based by filtering a subform foreign key against a parent record or set of records. Generally the relationship can be inferred from the meta-data by matching the parent's primary key against similarly named field in the subform's mapper. For example, if the parent record is a company with a company_id primary key, and the subform shows employees for that company, the default relationship would be to filter for employees where the employees' company_id matched the parent's company_id. In well structured databases we find that this type of relationship match works in the vast majority of cases.

That said, there are certainly times when such a relationship is not present. For example, if the system maintains notes on lots of different entities throughout the system, but stored in a single table (we use a GUID to provide a key that is unique regardless of what the note is related to), the notes on a company would not be expected to have a company_id column and might have instead a related_id column. To support such cases, the platform provides two properties that can be specified on a subform target item:

ViewKey – The key (field) in the target mapper to use as a filter key instead of the parent's primary key name.

ParentViewKeySource – The key (field) in the parent mapper to filter against instead of the using the parent's primary key value.

For example, to support the case above of providing notes for companies, the subform target's ViewKey would be set to related_id. The ParentViewKeySource would not need to be set because the filtering should indeed be performed against the parent's primary key, company_id.

### *Programmatically Overriding Subform Filtering.*

It is possible to override the standard subform filtering mechanism programmatically to force more complex filtering. The default subform filtering uses a single key from the selected item in the main pane and a single key from the subform's mapper. By default, the primary key is used from the top pane (or ParentViewKeySource) and the same named key on the subform's mapper (or ViewKey). See the NetQuarry wiki "Change the filter on a subform list" article http://support.netquarry.com/default.asp?W23 for detailed information.

### *Dynamic Showing/Hiding of Subform Tabs Using the EnableRule*

Subforms support subform item hiding based on criteria in the parent record. For example, the application might provide a subform to a People module with a UserInfo tab, but only for people who are also users of the system. Rules for dynamic subform tabs are specified in a subform target's EnableRule property. The EnableRule specifies a javascript expression that must resolves to true or false indicating whether or not to show the tab. The expression may include field expressions and is handled entirely on the client with no server load. See the NetQuarry wiki "Dynamic Subform Tabs" article http://support.netquarry.com/default.asp?W67 for detailed information.

There examples of subform hiding on the people subfom and the divisions subform, where data in the parent controls the visibility of the nodes in the subform.

When using the TabbedSubformTemplate, you should endeavour to make the first subform node of the page be one that all types of parent can see. Therefore as you move between rows and tabs are hiding and showing, the first subform is consistently open.

### *Subform Properties and Attributes*

The following properties and attributes are commonly used for subforms and their page targets…

- Disabled (attribute) – The subform is never visible to the user. If you go to the trouble of creating a subform and then you don't need it, it's best to disable the subform target, rather than delete it.

- MultiAdd (attribute) – The subform supports the multi-add feature. Mutli-add is where you can select from a Find popup, one or more records that are then copied automatically to the subform as new rows. The records listed in the popup find window are automatically filtered by the records in the subform so that the same records cannot be selected and added multiple times.

- Popup (attribute) – When you navigate to the detail from the subform, the detil target will popup in a window, rather than in the main window frame.

- RefreshOpender (attribute) – If you popup a window and then make changes to the record and save, setting this property will cause the opening page to requery to reflect the changes just made.

- RefreshParent (attribute) – If you make changes to the subform record, setting this property will cause the parent page of the subform to be requeried to reflect the changes just made.

## NavBars

Navigators provide a way to add links to pages. The trees, main links (across the top) and console/inbox are all created from the navigator metadata. The concept of navigators is similar to menu command and subforms in that you specify a target for that link. Navbars can use all six types of navigation target.

There are a standard set of Navbars that we refer to in the framework and these are…

- AppBar – This comprises the set of links at the very top, right of a NetQuarry application. A NetQuarry application includes a single AppBar which, by default, uses the navigator named "appbar". The default can be overridden using the *Appbar* application property. The application's *Log Out* link is always placed here. Certain system-level link are generally placed here including *Change Password*, *User Preferences*, etc.

- FolderToolbar – This is the main navigator tab set across the top of the application and provides global navigation to major modules within the application. A NetQuarry application includes a single FolderToolbar which, by default, uses the navigator named "main". The default can be overridden using the *Navbar* application property. The FolderToolbar supports a single level of sub-navigation rendered as a horizontal set of links, immediately below the tabs, with the links rendered dependent on the currently selected tab.

- NewBar – The NewBar, if used, is placed directly below the FolderToolbar and generally used to provide links to pages used to create new objects within the system (e.g. a new company record). A NetQuarry application includes a single NewBar which, by default, uses the navigator named "newbar". The default can be overridden using the *Newbar* application property.

- NavBar – The primary "side" slot renderer (typically the tree view) The NavBar is specified per-Page (or per-BasePage) as the page's Side slot PageElement rendering component. The Navigator to use is specified as a property on that PageElement. Typically the same navigator is used thoughout most of the application for a particular role so as to provide a consistent experience for the user as he navigates from page to page in the application. The NavBar provides a tree based on its navigator specification and may include sub-trees specified by sub-navigators.

### *Properties and Attributes of Navigation Targets*

There are some particularly useful properties of Navigation targets (specifically for page navigation targets) that are useful to know.

- Filter (property) – When you navigate to a page, the specified filter is added to any existing page and mapper filtering.

- RequestType (property) – When you navigate to a page you can specify whether the navigation is to an existing record (nav) or should force the navigation to a new record (new)

- UseNavCaptionOnTarget (property) – When you navigate to a page and the navigator has this property set to true, the default caption of the page will be overridden by the caption of the navigator target.
- Expanded (property) – When the target of the navigator is another navigator (therefore a child tree) this property defines whether that sub tree is automatically expanded when viewed.
- AppendCount (attributes) – When the target of the navigator is a page, the count of records matching any filter criteria on the target page is displayed in parentheses to the right of the target caption. This is typically used in the inbox pages. Use this property wisely as it is expensive to perform counts on all links
- NoAppendCount (property) – When the target of the navigator is another navigator. And on that child navigator there are potentially AppendCount attributes set, this NoappendCount blocks the counts from being calculated on the child navigators.

The inbox page uses navbar metadata for each group of links. Those links contain counts. Those same navbars are used by the tree to display the same links, but do not contain counts. This is due to the override NoAppendCount property on the main-tree navbar.

## Navigation Parameters and Query Strings

When a page is navigated to we rely on query string parameters to communicate to the core which page to navigate to, how to filter the data and what state the page should be in. There is a standard set of query params used in the application. You can see what query params have been used for a given page by pressing the F8 button on the page. All the query param information is given at the bottom of the F8 page.

### Query params used by the core

| Parameter | Description |
|---|---|
| parmop | The mop of the immediate parent of this page. parmop appears on subform pages, wizard pages and QuickLink navigated pages |
| parmap | The mapper of mop on the immediate parent of this page. parmap appears on subform pages, wizard pages and QuickLink navigated pages |
| parrk | The row key value of the immediate parent of this page. parrk appears on subform pages, wizard pages and QuickLink navigated pages |
| parkey | The row key name of the immediate parent of this page. parkey appears on subform pages, wizard pages and QuickLink navigated pages |
| pardesc | The description of filter criteria on the mop where the navigation originated. pardesc appears on subform pages, wizard pages and QuickLink navigated pages |
| rk | The value of row key for the current record. |
| fk | The name of the foreign key field in the mapper on the current page. fk appears on subform pages, wizard pages and QuickLink navigated pages |
| fkval | The value of the foreign key field in the mapper on the current page. fkval appears on subform pages, wizard pages and QuickLink navigated pages |
| flt | Custom filter criteria applied to a page in addition to or in place of primary key filtering. Because of the risks of SQL injection attack, the use of the flt parameter can only be used in conjunction with code. |
| flv | An additional flavor added to the page. |

| Parameter | Description |
|---|---|
| req | Request type. Two main types are “nav” and “new”. Most navigations to pages are “anv” which means navigate to an existing record applying any pk filtering, or additional filtering as necessary.<br><br>“new” navigation forces the page to display a new page<br><br>Another support request type is “get” which relates to file support. |
| pop | If the value is 1 then the navigation will open a page in a popup window. |
| mop | The name of the mop that is the target of the navigation |
| crumb | Refers to the level of crumb links in the captions. Crumb level of 1 is the first caption. Set crumb=1 to reset crumbs for your navigation. |
| pk | The value of the primary key or unique key on the page |
| origmop | The mop of the originating parent. origmop appears on wizard pages that are part of chained wizards |
| origmap | The mapper of mop of the originating parent. origmap appears on wizard pages that are part of chained wizards |
| origrk | The row key value of the originating parent. origrk appears on wizard pages that are part of chained wizards |
| origkey | The row key name of the originating parent. origkey appears on wizard pages that are part of chained wizards |

## Using Application.Navigate

The navigate method is declared as

```
// "mop" - The MOP for the page.
// "primaryKey" - The primary key value for the request
// "queryStringParams" - Additional QueryString (GET) parameters.
// "requestType" - The type of the request (by default this is "nav.
// "attrs" - NavigateAttrs for the navigation request
// "paramList" - A semi-colon separated list of param names to take from the current query string
to add to the navigation querystring.
public void Navigate(string mop, string primaryKey, string queryStringParams, string requestType,
NavigateAttrs attrs, string paramList)
```

There are additional methods that take fewer parameters for convenience

### *Navigate to a list view*

```
app.Navigate("assignments!list");
```

### Navigate to a filtered list view

```
string filter = viewKey + " = " + EAPUtil.AnsiQuote(primaryKeyValue);
string navigateURL = string.Format("&flt={0}", EAPEncode.ForUrl(SavedFilter.RegisterReqFilter(appCxt, filter)));
app.Navigate("assignments!list", string.Empty, navigateURL, "nav");
```

### Navigate to a detail

```
app.Navigate("orders!detail", sender.order_id);
```

### Navigate to new detail

```
app.Navigate("order_new!detail", string.Empty, "parrk=" + sender.company_id, "new");
```

### Navigate using query params of existing page

If you wish to navigate to your new page from your command, but pass the query params from the current page directly to the new page, you would use the full declaration (shown above) and specify the attribute value NavigateAttrs.IncludeStandardParameters. and paramList is null.

```
app.Navigate("export_files!wizard", string.Empty, customParam, "new",
NavigateAttrs.IncludeStandardParameters, null);
```

What this does is to extract the following query params from the current query string and rebuilds a new query string from them.

- parmop
- parmap
- parrk
- parkey
- pardesc
- parval
- rk
- fk
- fkval
- origmop
- origmap
- origrk
- origkey

If some of the query params are missing from the current query string they are not included. If you have included one of the listed query params explicitly in the “queryStringParams” parameter, then the navigation function will use your explicitly added value, rather than the default from the current query string.

If you know which query params you want to add to your navigation from the current query string, then you can pass a semi-colon delimited list of those parameters. Same rules apply with regard to “queryStringParams”.

```
string params = "parmop;parrk";
app.Navigate("export_files!wizard", string.Empty, customParam, "new", 0, null);
```

## Creating Modules, Mappers, Pages

For the most part you will be modifying existing modules, mappers and pages. However you should know how to create these objects from scratch. There are three ways to create these objects.

Wizard (from Home Page)

Manually (better when you have more experience)

Duplicate (useful way to create new objects quickly with minimal re-work)

From scratch, to create a page for a new module, you first have to create a new module, then a new mapper, then a new page. So we'll go through the example of creating a new page which describes the entire process.

Before you create your new objects it's advisable to have the underlying schema all configured in the database before you start.

### New Page from Wizard

- Open the NQ Studio and on the home page, click on the New Page wizard link.
- Enter the name of your new module (or existing module)
- Enter the name of an existing mapper, or new mapper.
    - Typically if you are creating a new module, you'll be choosing a new mapper.
    - If you expect to have business rules associated with your new page, uncheck the "Do not generate a typed mapper"
    - Your data source is the datasource for your operational database
    - The majority of your mapper datasources will be views
- For mapper options, you can provide the names of the fields that perform special actions or that contain certain types of data. Setting these values automatically adds special attributes and properties to the fields.
- You can set navigation properties for the page you're creating. You access the new page from the main navigation bar (along the top), or specify it to be a subform of another page.
    - If you specify the page as a subform you have to describe the fields involved in the parent child relationship.
- You can set page options for the page you're creating

- Template defines how the page is rendered on the screen. For most applications you can use the tabbed subform template which is basically a detail or list page on the top half of the screen and below a set of tabs to access related information (subforms)
- Page styles let you decide what types of page should be added. Normally a page would be rendered as a list view to see many records at once, and then as a detail view to see more information.
- Pages can have a default style or what's called a base page. If you use a base page for your pages, the page will render in a way consistent with other pages in the application. Typically a base page will say that the page should be displayed in the main window and the navigation tree be displayed on the left
- If the page is marked as having subitems, it means the page will have subforms and a subform object will be created. The name of the subform will be the same as the page.
- If you want the user to be able to search from the quick search, you would check the box. The module will appear as an option in the quick find drop down. (If it worked).
- For any page you create you should provide a short description for the page. The page description will be visible to the user. If you don't provide a description, the name of the page is used.
- Typically a list view is given a name "list" and detail pages are called "detail". When starting a new module you'd normally want a detail and list page.

### New Page Created Manually

The new object wizards are an effective way to quickly convert database schema into workable user interfaces. Of course once you need to add pages out of the ordinary, you'd need to understand how to create mappers and pages manually. Same with modules, but creating a new module with the wizard is probably quicker at any time and guarantees that you associate the new module with the application. Not having that association configured correctly means that your metadata won't get installed on other machines.

### New Page from Duplicate

Many of the objects in the studio have the ability to be duplicated (except modules). If you have a mapper or a page that is similar to what you need, you can duplicate it. We typically duplicate objects in order to provide slightly different behavior for permissions.

When duplicating an object, not all of the related objects are duplicated. In general, permissions are not duplicated to the copy. For a mapper, the extensions are not duplicated either.

### Put new objects in the build

Be sure to save your metadata following the instructions given above. Especially if you have added a new module. After adding a new module to the application, the metadata file associated with that

module also needs to be added to the build environment. The best way to do that would be to send us an email, letting us know to add the file to the installer.