NetQuarry

The Enterprise Application Software Development Platform for Microsoft.NET

IssueTrak Tutorial –2 – Create IssueTrak

Table of Contents

Purpose of this document

This document is a tutorial that explains how to build an application using the NetQuarry Enterprise Application Platform. The document is split into three parts.

Part 1 – Explains how the NetQuarry Platform works and how the platform should be installed and configured.

Part 2 – Takes you through a set of steps to create a functional bug tracking application. The steps will be deal entirely with manipulating metadata to create the IssueTrak application.

Part 3 – Enhances the basic functionality of the IssueTrak application by adding complex business rules with C# extensions, creating scheduled tasks and debugging.

Creating the Application

The center of the metadata is an application object. (Represented at runtime by the IAppContext interface.) In general, you create only one application object per metadata, but there is future support planned for multiple application support.

The Application (or Context) holds some important information at runtime about the application. First, it holds global properties that apply to all users of the application. These are set in the application's property sheet using the NetQuarry Studio.

Second, the application object specifies which datasources should be loaded and made available at runtime by name. A data source must have its connection specified in the web.config file. It is NEVER specified in metadata. *(Note that at this time, e.g. Version 1.0, connection strings are stored as plain text in the web.config file. For this reason, it is recommended that you use integrated security to connect to the data sources.)*

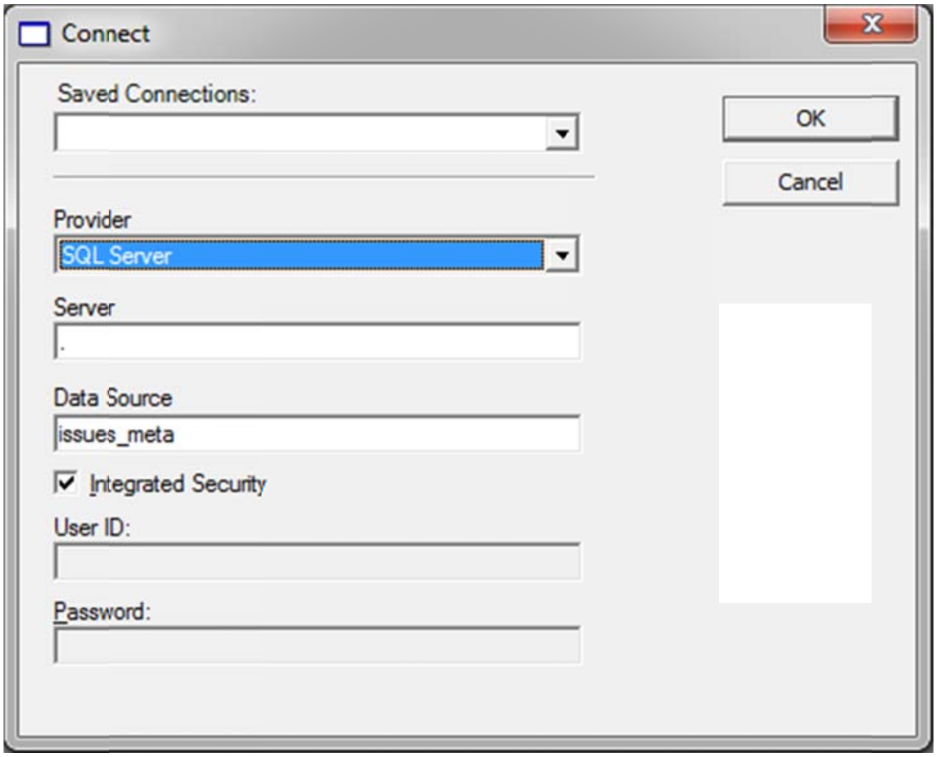
Third, you must specify at the application level which modules are associated with the application. In general, you should add ALL modules into your application. (Note that the wizards, both application and page, do this automatically).

Running the Application Wizard

After your database is initially designed, or already exists as in this example, you should run the application wizard to rapidly create your application's starting metadata. If you don't have all of the tables, views, or columns defined, don't worry, you can always add them later!

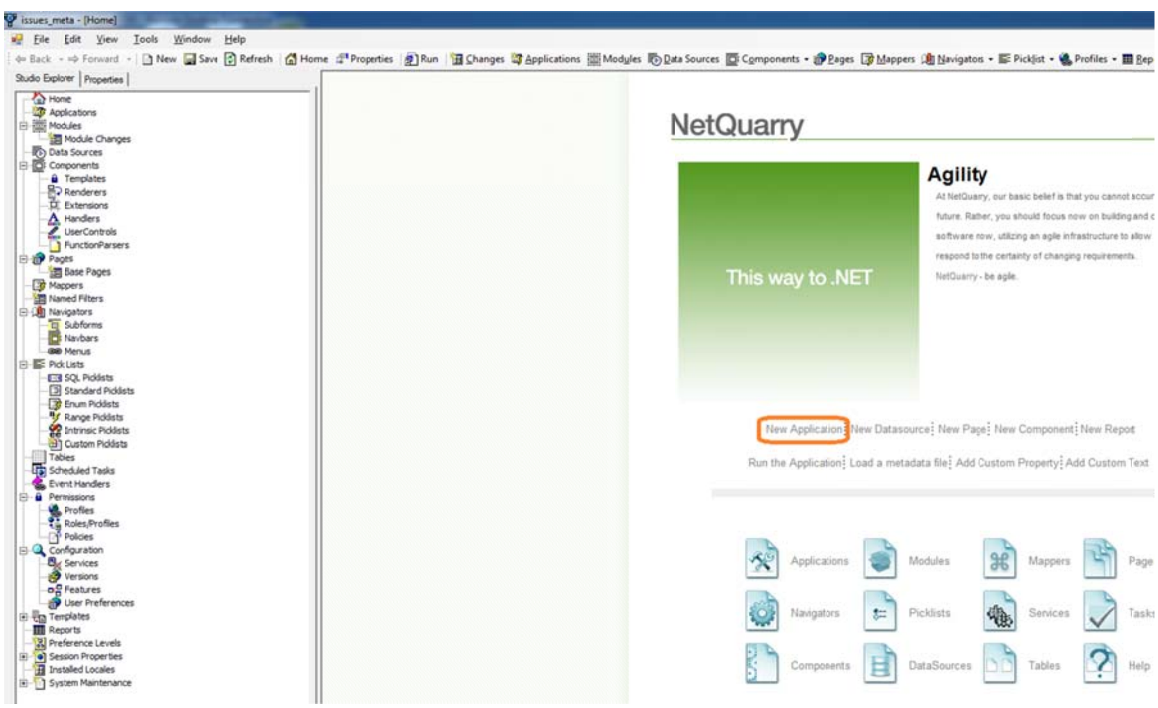
Using the QuickLinks toolbar, go to NQ Apps, NQ Studio link to start the NQ Studio. You are prompted to connect to a database. The database you are connecting is the metadata repository database. In this case issues_meta.

Step 1 – Specify your metadata database in the connection dialog:



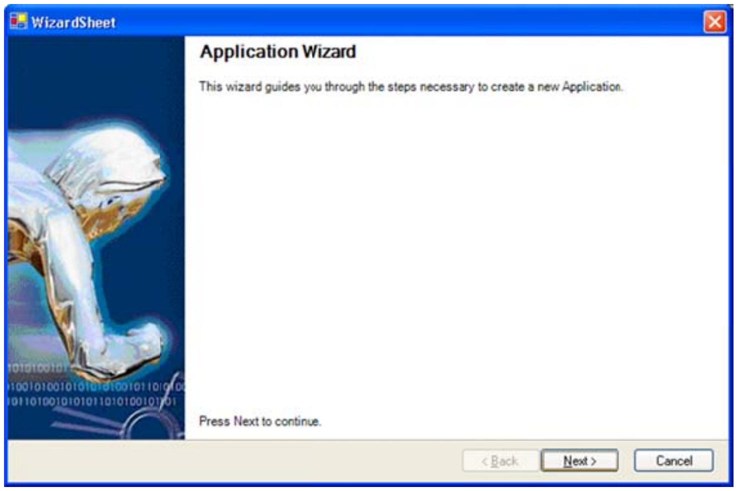
NetQuarry Studio Connection Dialog

Step 2 — click on the "New Application" link under Wizards on the Home Page of the Studio:



Launch New Application Wizard

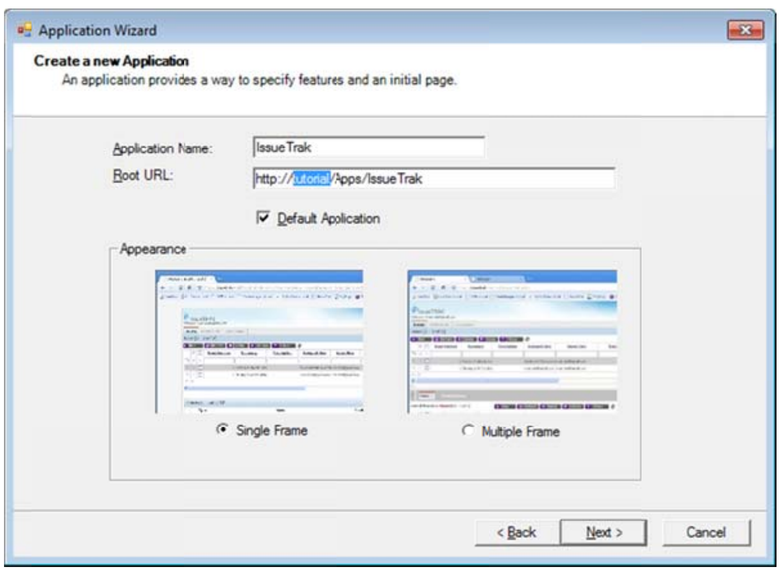
Step 3 – Click Next on the Wizard and type "IssueTrak" as the name of the application. For the "RootURL" property, you need to enter the machine name and virtual folder (if you installed with a virtual folder) of the NetQuarry application.



Application Wizard

Step 3 – Click Next on the Wizard and type “IssueTrak” as the name of the application. For the “RootURL” property, you need to enter the machine name and virtual folder (if you installed with a virtual folder) of the NetQuarry application. Following this example, the Root URL should be http://<your machine name>/apps/IssueTrak.

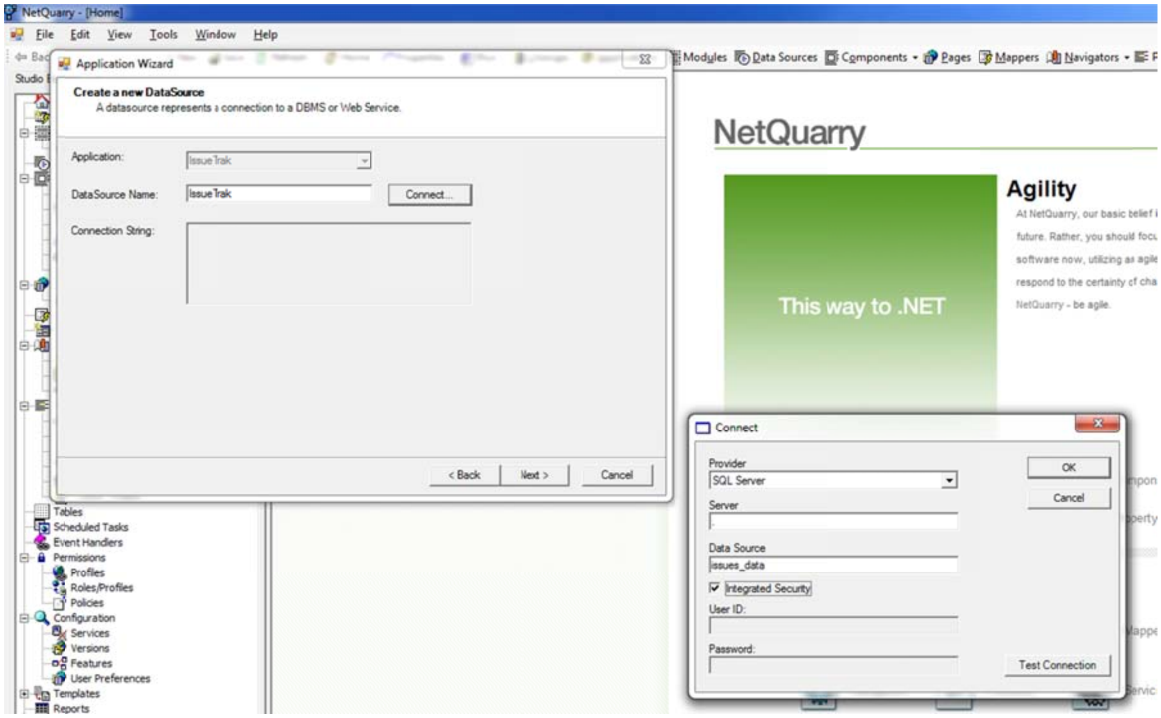
In this tutorial, we’ve mapped the name “tutorial” to 127.0.0.1 in the hosts file of the development computer. (C:\windows\system32\drivers\etc\hosts). If you wish to do this. There is a shortcut to open the hosts file in the Tools folder of NQ Links.



Application Options

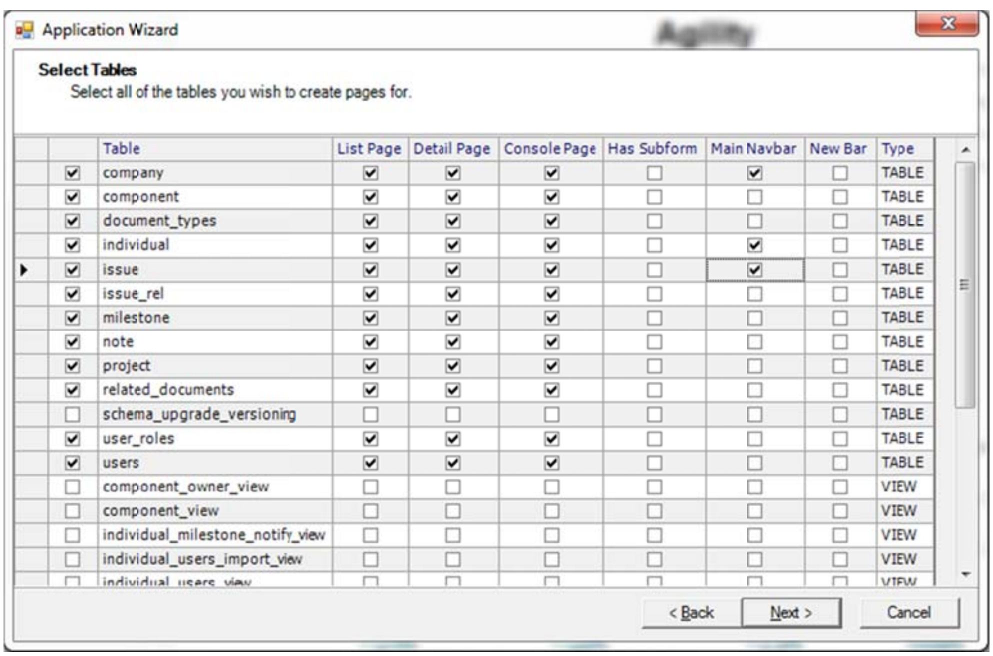
Next, you need to identify the “main” data source for the application where the operational data is located. While there is support for more than one data source in all NetQuarry applications, for the purposes of the wizard you only need to specify one. To specify a data source, click on the “Connect” button to launch the Studio Connect dialog.

You should connect to the database named “issues_data” that has been installed by the database scripts. Test the connection settings before continuing.



Data Source Page

Next, you will select that tables and views that should be used to create the initial data mapping objects. This page presents a grid that allows you to include or exclude a table and specify basic options about the object. For the IssueTrak application, only the tables named “company”, “individual”, “issues should be set to be on the “Main Navbar”. There should be no pages with “Has Subform”. The table schema_upgrade_versioning should be completely excluded from selection. Ultimately, your selected options should match as shown in the figure below. When you are done, press Next.

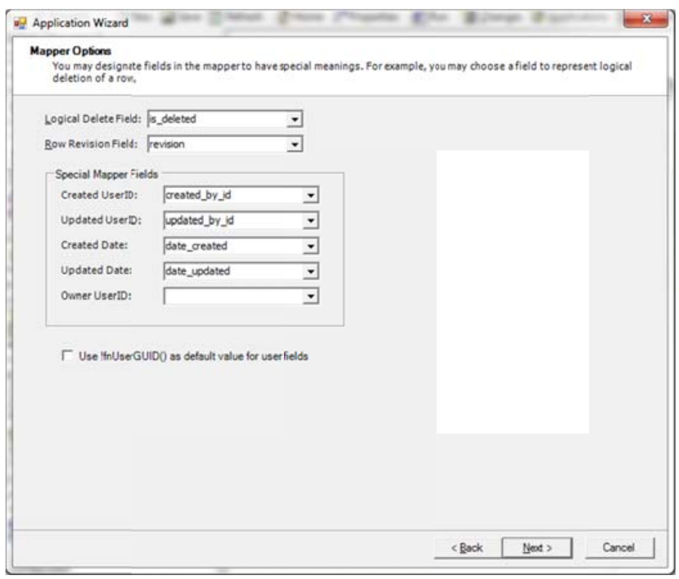


Tab Selection Page

The next screen of the wizard allows you to select names of fields that have special meanings. The screen allows you to specify the fields that store

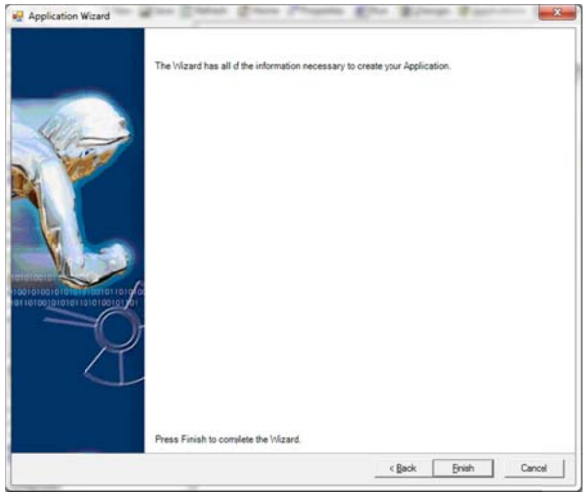
- Logical delete field
- Row revision field
- Created UserID field
- Modified UserID field
- Created date field
- Modified date field
- Owner UserID field

As the wizard “slurps” the schema information from the database and creates the mapper and field metadata, the wizard will compare the names of the fields in each mapper with those specified in the wizard dialog. When a name match is found, specific properties are applied to those mapper fields. For example, a field specified as a “Created UserID” field, will be assigned a default value. This step is entirely optional, but if the fields are populated as shown, will save time configuring the mapper fields later.



Mapper Options Page

At this point you are done with the wizard options. If you are satisfied with your choices, click the Next, then Finish button. The application’s initial metadata will be created.



Finish Page

Main Navbar, NewBar, and Has Subform explained

In the Table page of the wizard, you are asked to select options that make the pages you are creating behave and look a certain way. If you select "Main Navbar" next to the table, it will cause the page's list version (pages have both a detail and list by default) to show up as a tab across the top of the application. If you select "Newbar" a link to the page's detail version will be displayed on the bar directly below the main tab navigation bar. If you select "Has Subform" the wizard will create a navigator (type subform) and connect it to both the detail and list versions of the page.

About Views

You will notice that you didn't select any views as top-level objects in the wizard. Views are fully supported by the data mapping layer, but occasionally will require additional metadata configuration.

First, you need to understand how a NetQuarry Mapper object interacts with a DBMS. When a NetQuarry Mapper object needs to update, insert, or delete data, it does so by building a DBMS specific SQL statement against the underlying *tables.* For example, if you join the users table with the individual table in this tutorial on the user_id column, you would expect that there would be 2 insert, update, or delete statements for each transaction; one against table individual and one against table users.

In order to determine what the underlying tables are, the NetQuarry Studio asks the DBMS for each column's source table. However, and this is why views are a little tricky, some of the supported DBMS (SQL Server being an unfortunate example) don't provide accurate information on columns that have been aliased. So, when the tools ask for the underlying table, the result is that the view is returned by the DBMS.

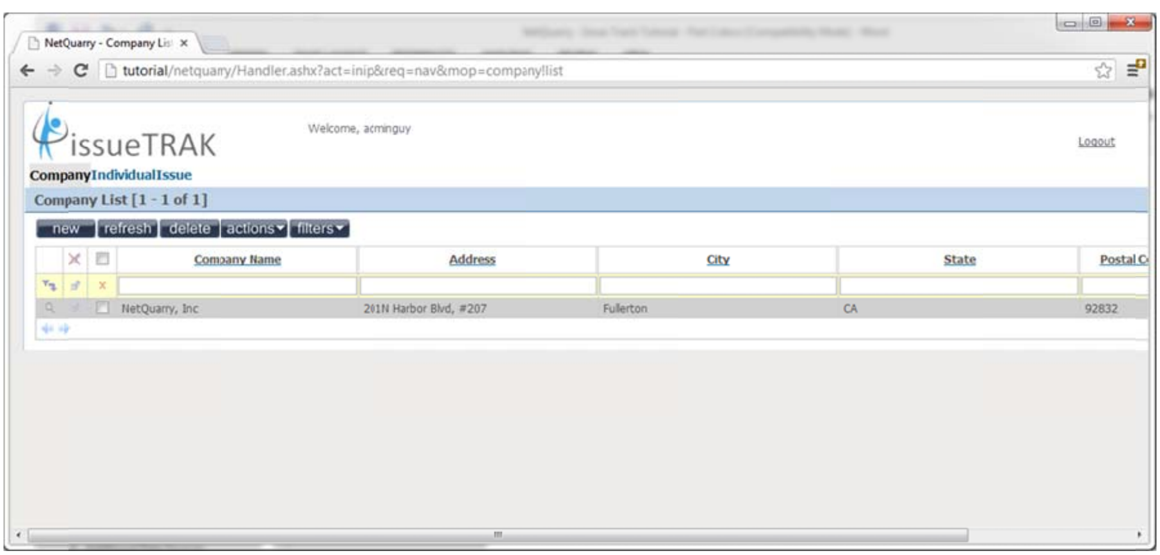
Each field in a Mapper has a property called "Table." If you use a view as the source for your Mapper (as we will in this tutorial), you will have to update the columns that have been aliased in the view with the proper table name.

What just happened?

You now have a somewhat functional application. You have 3 top level objects on the main navigation tabs (company, individual, users), and all of these objects can be inserted into, deleted from, and updated. You can also search using the filter by form row.

What the wizard has done is create a metadata definition of some basic NetQuarry objects. The metadata and the NetQuarry runtime will work together to produce your application. To better understand what you have, it is helpful to start with what you see when you run the application. To run the application, Open your favorite browser and navigate to `http://tutorial/apps/issuetrak/default.htm`

To login to the application you can use any user name you like with or without a password. (We will add an authentication layer later). You should see something like this:



Templates, Pages, and Slots

In this example, the page that is displayed is the company list page. The page is identified by its module/page (MOP) which is the module and name concatenated together with a "!" character—like this: "**company!list"** The MOP serves a couple of important purposes. First, it allows NetQuarry to have multiple pages with the same basic purpose, but different behavior or looks based on the particular set of permissions that the current user has. Second, it is a human-readable name and not an opaque identifier. You can use a MOP in a link, for example, and not have to know that the page you are referring to has an ID of 7843, for example.

Since the MOP isn't required to be unique (for example, you may have multiple pages with the same MOP but assigned different user permissions), there is an additional name called a "moniker" that is defined by the developer and normally used only in the Studio.

Each page is based on a template. A template has a set of regions, called Slots (with associated behaviours), hosting Page Elements. Different templates may support different sets of slots. Each slot in the template hosts "renderer" that renders its contents into HTML. A renderer is specified as a field on the slot. A number of standard renderers are supplied with the Platform and you may write your own.

The template used in the above picture is called the "Expanded Subform" template. This template support 2 slots: the "Main" slot and the "Side" slot. However for single frame applications, the side slot is not typically used.

While all this may sound a little confusing, it may be helpful to think of it in terms of function. A slot is simply a place on the template; its function is to hold a control. A renderer is a control (ASCX UserControl) that renders some interesting HTML. A template is the glue that holds all of this together.

Mapper

As mentioned above, a Mapper is the primary means of mapping object-oriented programming objects to data sources managed by relational databases or other sources of data. The simplest way to think of a Mapper is to imagine a super charged .NET DataTable that is created programmatically.

Mappers are incredibly important objects in the Platform. They allow you to define behaviors, provide look and feel hints to the renderer, change column mappings, and set permissions (access control) on its fields. Mappers also support a rich set of events that you can program against.

Properties

As much as possible, NetQuarry has been defined as a declarative programming Platform. Much in the same way that managed classes use attributes, NetQuarry uses properties on objects to modify many aspects of behavior. Most of this is fairly discoverable. Click on an object in the list and look at the properties in the property sheet; most of the time you will see the property that controls the behavior you want to change. In some cases, the attribute is *in* the list or specified as an attribute instead.

Modifying the application

Now that the application has been started, you need to make a few changes to adjust the layout and styling of the application and glue some objects together. Most of these changes are simple changes made using the NetQuarry Studio. The other few business rules will be written using standard .NET code.

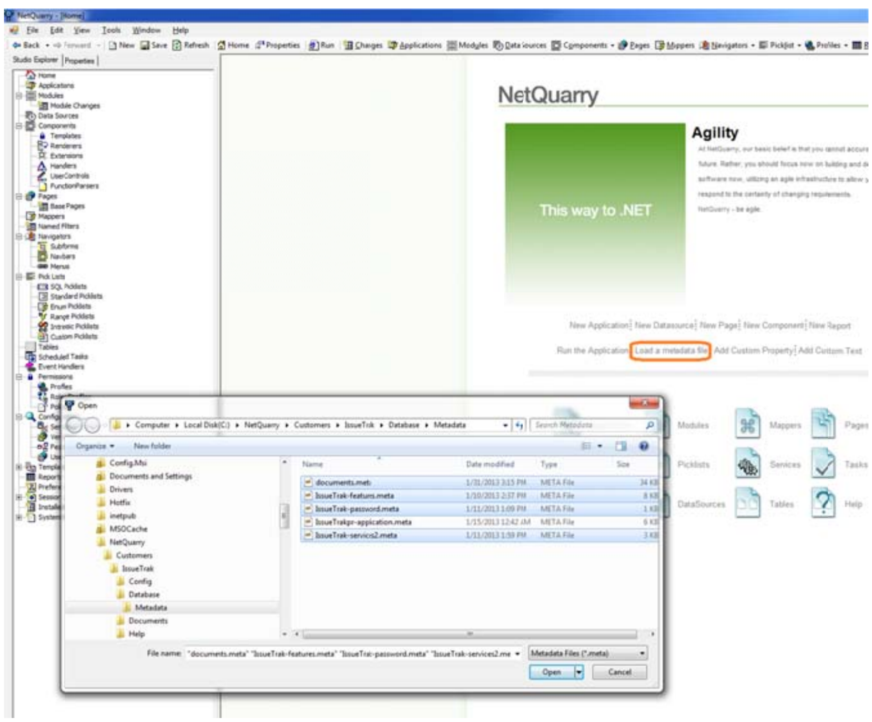
Extra Modules Adding Functionality

The tutorial comes with a set of additional modules that we should now load from the metadata files into the studio (and metadata repository database, issues_meta).

On the home page of studio, click on the link to "Load a metadata file".

In the file dialog that opens, navigate to the C:\NetQuarry\Customers\IssueTrak\Database\Metadata folder.

The file open dialog supports multiple selections, so select all the .meta files and click on the "Open" button.

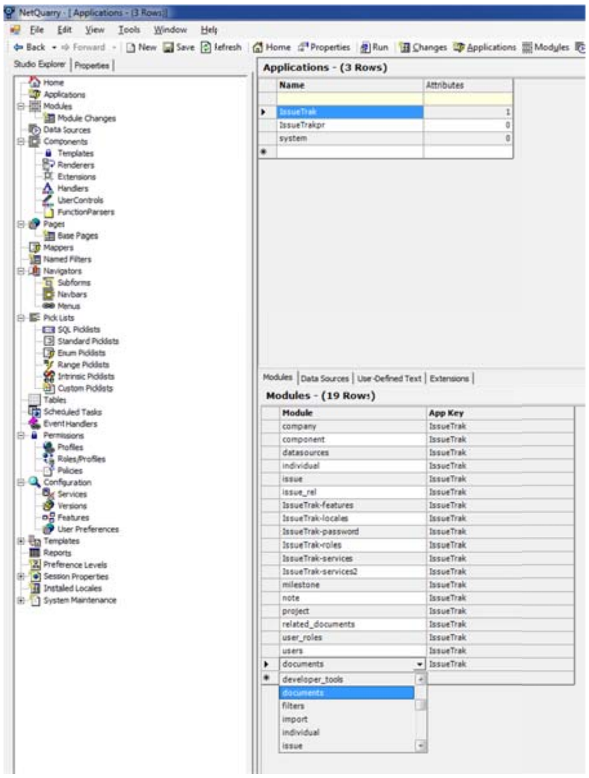


Loaded Meta File	Module Name	Purpose
documents.meta	documents	Document storage mapper.
IssueTrak-features.meta	IssueTrak-features	Application customization
IssueTrak-password.meta	IssueTrak-password	Password change functionality
IssueTrak-services2.meta	IssueTrak-services2	Document storage services

Associating new modules with an application

On the Studio Explorer, click on the Applications link (or type Alt+A shortcut).

We have to associate the newly loaded modules with the application container. Ensure that in the parent list, the IssueTrak application is selected. In the Modules subform, on the new insert row, select and add each of the modules we added.

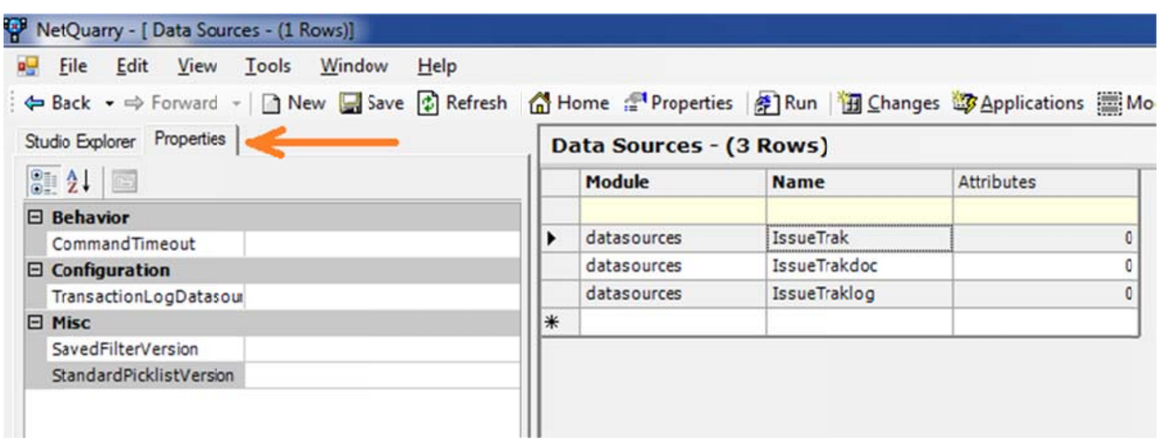


Additional Data Sources

On the Studio Explorer, click on the Datasources link (or type Alt+D shortcut).

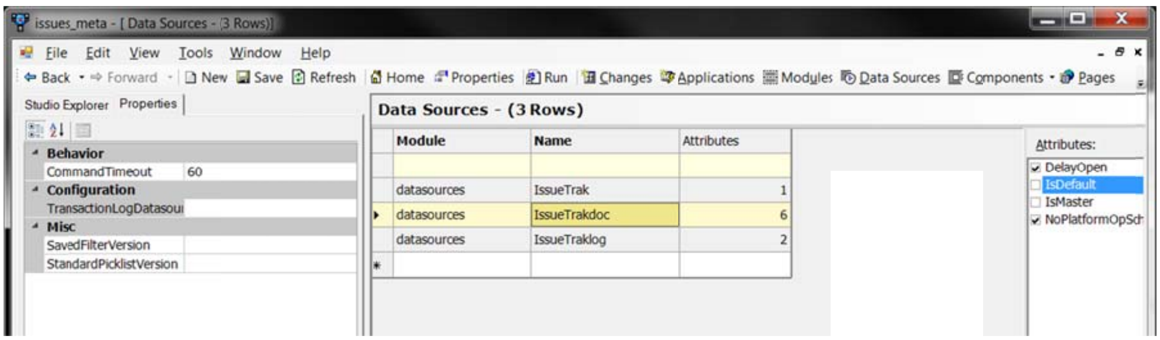
Currently there is one datasource, IssueTrak, that was generated during the application wizard. To support additional features, we need to add two more datasources.

So on the insert row of the datasources list, add IssueTrakdoc and then IssueTraklog.



Once they are added we have to set some properties and attributes for these datasources. Attributes for a datasource are on the right of the studio window. Properties are in the tab next to the Studio Explorer.

Datasource	**Property**	**Attributes**
IssueTrak	CommandTimeout=60 TransactionLogDatasource=IssueTrakLog	1 (IsDefault)
IssueTrakdoc	CommandTimeout=60	6 (2 DelayOpen, 4 NoPlatformOpSchema)
IssueTraklog	CommandTimeout=60	2 (DelayOpen)

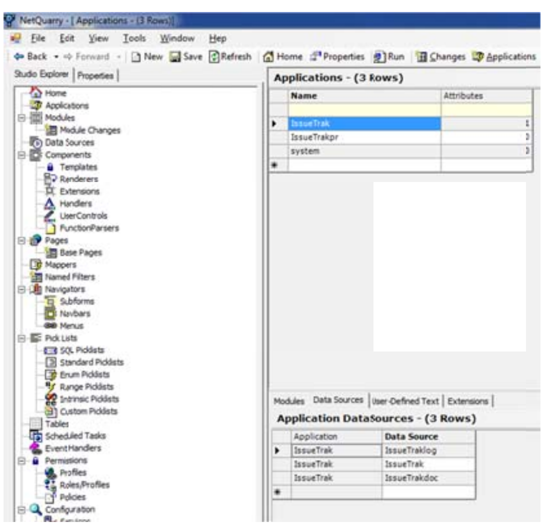


Associating new datasources with an application

On the Studio Explorer, click on the Applications link (or type Alt+A shortcut).

We have to associate the newly created datasources with the application container. Ensure that in the parent list, the IssueTrak application is selected. In the Datasources subform, on the new insert row, select and add each of the datasources we added.

If the Data Sources are not displayed in the drop down, click in the drop down cell and press F9 to refresh the drop down picklist.

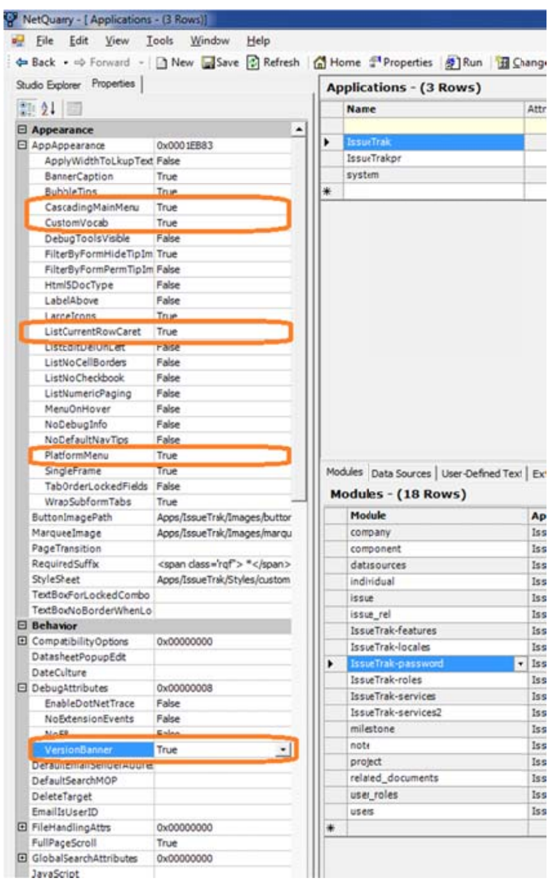


Configuration Application Properties

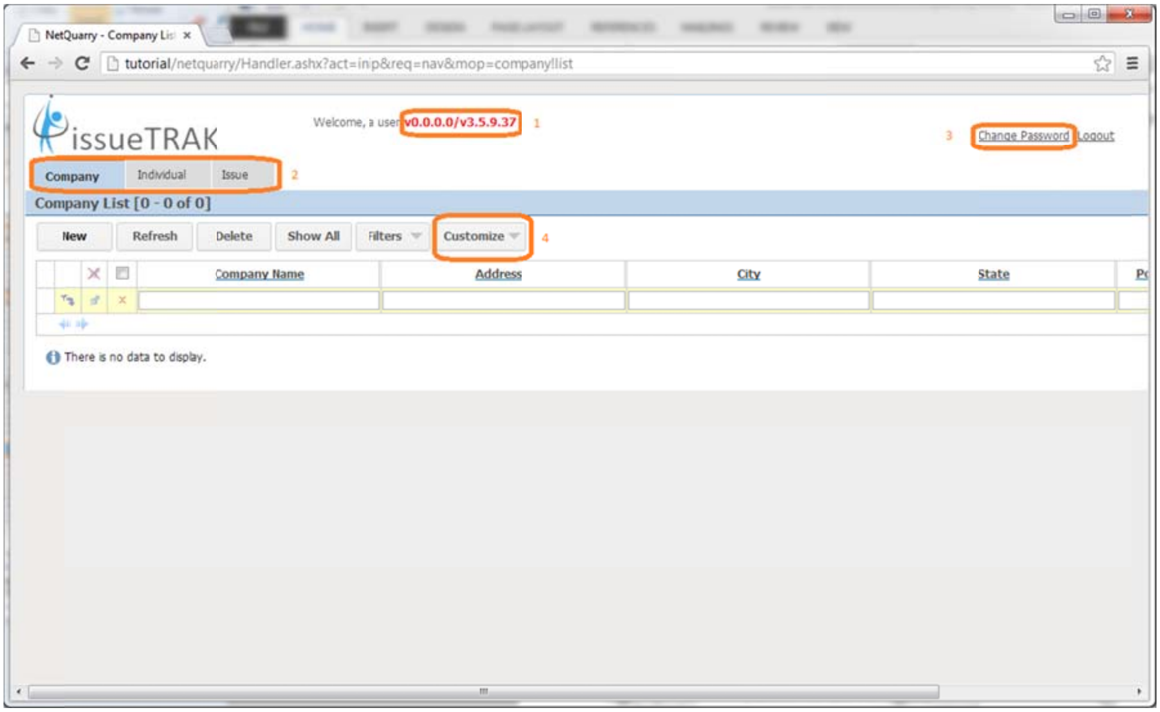
On the Studio Explorer, click on the Applications link (or type Alt+A shortcut). Ensure that in the parent list, the IssueTrak application is selected.

Make sure the Properties window is open and make the following property changes

Category	Property	Value
Appearance	AppAppearance	(expand options) CascadingMainMenu: True CustomVocab: True ListCurrentRowCaret: True PlatformMenu: True
Behavior	DebugAttributes	(expand options) VersionBanner: True
Request Auditing	LogPageActivity	True
	PageActivityDatasource	IssueTraklog



Having made these changes you can log out of the application and log in again. You will notice the user interface is a little cleaner and contains a couple of new items.'



Item #	Visible Changed Caused By
1	DebugAttributes:VersionBanner
2	AppAppearance:CascadingMainMenu
3	IssueTrak-password.meta
4	AppAppearance:PlatformMenu

Users/Individuals

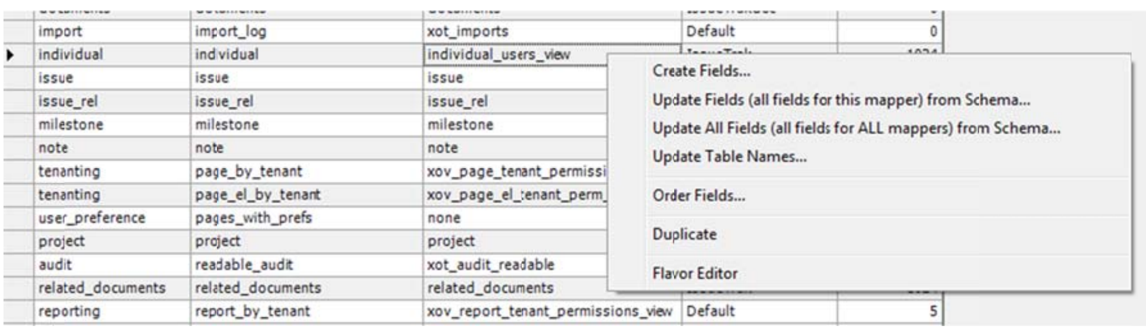
In this application we are modeling a user in two tables. The reason for this is that the users table is used by the database authentication provider and is referred to by name. While you *can* add columns to the users table, we consider it a best practice to build another table and refer to the users table via a foreign key.

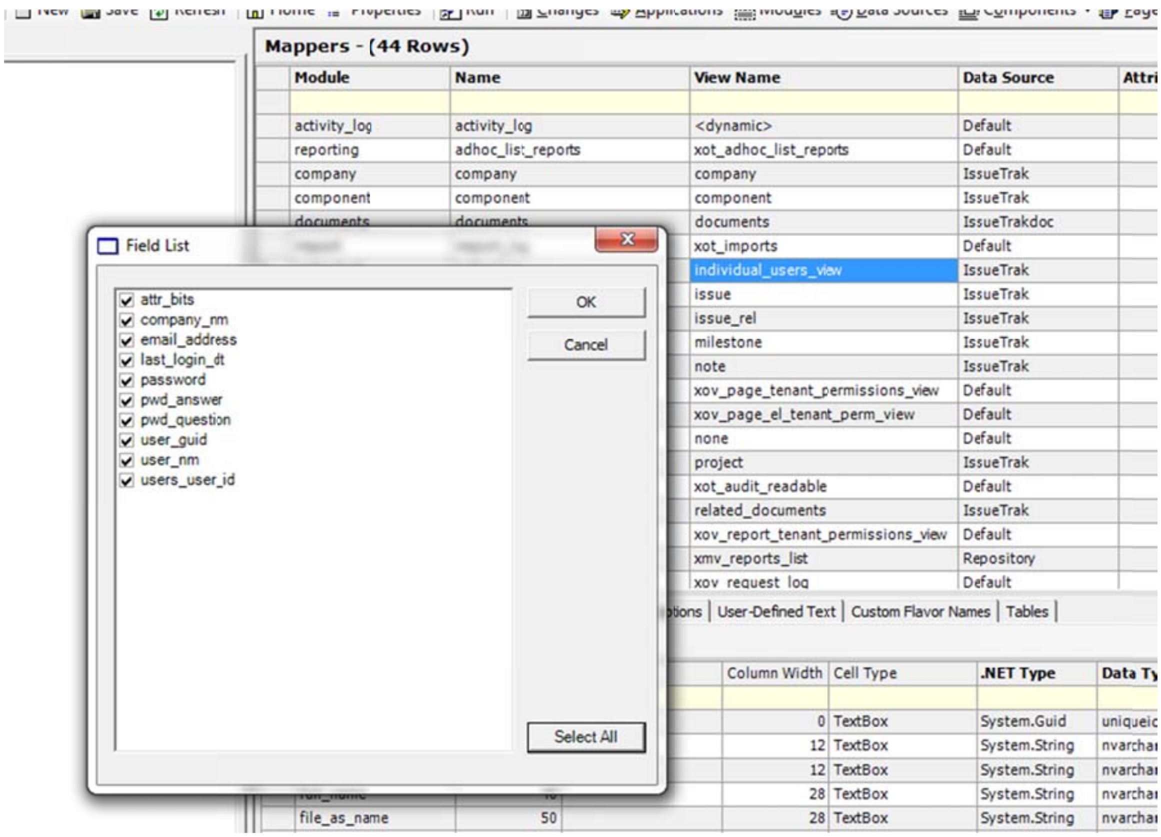
To accomplish this, we have created the view called "individual_users_view." This view joins the tables individual and users on the user_id column. If you look at the design of the view, you will notice that the foreign key column user_id in the individual table has been aliased.

Changing the Mapper's View

Follow these steps to change the underlying view for the individual Mapper:

- Open the Home Page.
- Click on the Mappers link on the Home Page (or press Alt+M)
- Select the individual mapper in the list.
- In the View Name column, change the name of the view from individual to **individual_users_view**.
- Move to the next row to save your changes (or click the Save button on the toolbar).
- Select the individual mapper again in the list.
- Right click and choose "Create Fields…" In the create fields dialog, click the Select All button and then click OK. The additional fields from the users table joined into the view will be added to the mapper.





Configuring Mapper Functionality

In order to make the mapper function correctly, we need to define how the data will be saved into the database. Although the mapper selects the data from a view, the save of data back to the database takes place through each table in the underlying view. In order to save data through a table, that table must have a primary key defined for it. Once the primary key is defined, then the fields in the mapper from that table can be saved.

The mapper is based on the following schema structure.

individual_users_view			
individual table		users table	
View field (pk)	Table field (pk)	View field (pk)	Table field (pk)
individual_id	individual_id	users_user_id	User_id

When creating an individual record in the individual table, we also need to create an equivalent user record.

Looking at the fields in the individual table, there is a user_id field which is used as the join field to users. But there is also a user_id field on the users table which is the primary key of that table. For the user interface, we don't want to specify the user_id value twice, so we have to set up the mapper to have field values copied from one 'side' (table) of the mapper to the other.

Here's a table of the likely mappings

individual table fields	Equivalent user table fields (in mapper)
individual_id	user_guid
user_id	users_user_id
full_name	user_name

Also, there are some fields that contain concatenations of two other fields. E.g. full_name which is the concatenation of "first_name last_name", and the file_as_name field which is a concatenation of the "last_name, first_name" fields

We strongly discourage the use of database integrity rules to control whether certain fields are required (non nullable) or whether there are default value constraints. These concepts can easily be handled through the platform metadata.

Use the following table to set up the individual mapper fields to ensure when an individual record is created, an equivalent user record is created.

Field	Settings	Notes
individual_id	Attributes: AutoAssign, NonNullable, PK, UniqueKey Column Width:0	AutoAssign is automatically added to a field by the platform when the field is a PK and has an underlying type of uniqueidentifier, or string. UniqueKey tells the platform that this field in the mapper is a value that defines the uniqueness of each row. If a mapper has multiple fields marked with the PK attribute. One of these fields should be marked as the unique key. A column width of 0 means the field is hidden.
first_name	Attributes: Required	
last_name	Attributes: Required	
full_name	Calculated Value: [first_name] + " " + [last_name] Attributes: Locked	This is a piece of javascript that takes the values of the first_name and last_name field and concatenates. The field is locked since this is a calculated value
file_as_name	Calculated Value: [last_name] + " " + [first_name] Attributes: Locked	
gender_id	Cell Type: ComboBox	This will have a picklist to select gender but we'll add that later.

Field	Settings	Notes
user_id	Attributes: Required Column Width: 20 Caption: User ID	
company_id	Attributes: Required Column Width: 25 Cell Type: ComboBox	This will have a picklist to select a company from one of the companies added to the IssueTrak application.
users_user_id	Column Width: 0 Attributes: PK ColumnName: user_id Table: users SourceField: user_id	Hide the pk field of the users table Set the field to be the PK of the users table Set the mapper field to save to the underlying user_id field. Set the name of the table we're saving to. Obtaining the value of the user_id field from the user_id value entered on the individual table's user_id
password	Attributes: Required Cell Type: Password	Make the password field with masked characters
user_nm	Column Width: 0 SourceField: full_name	Hide the user_name field of the users table Although hidden, this field is still required. Set the value from the individual table full_name field
attr_bits	Column Width: 0 DefaultValue: 4	Hide the attributes field Set the default value to 1 (forcing a password reset on first logon)
pwd_question	Attributes: Required ColumnWidth:15 Cell Type: ComboBox Picklist: secret_question	Choose a password reset question from a platform provided list of questions.

Field	Settings	Notes
pwd_answer	Attributes: Required ColumnWidth:25 Cell Type: Password	Make the password answer field with masked characters
email_address	Attributes: Required ColumnWidth:25 Cell Type: EmailAddr	Forces a regex validation of an entered email address by setting the EmailAddr type field
last_login_dt	Column Width: 0	Hide field on this mapper
company_nm	Column Width: 0	Hide field on this mapper
user_guid	Column Width: 0 SourceField: individual_id	Hide user_guid field of the users table Although hidden, this field is still required. Set the value from the individual table individual_id field

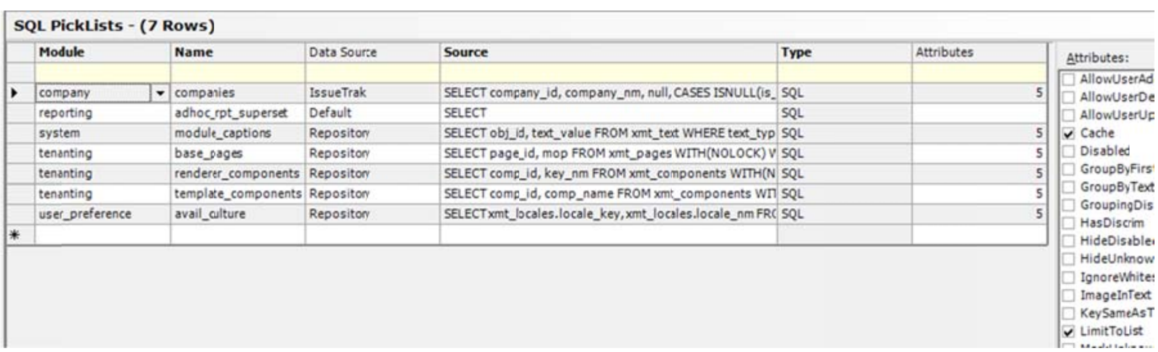
Before we can create a new individual, remember we have forced (by setting the company_id field to be required) an individual/user to be associated with an existing company.

Creating a Company Picklist From Operational Data

In the Studio Explorer click on the SQL Picklist Link (or Alt+L, Q)

Create a picklist with the following settings

Item	Value	Notes
Module	company	All metadata is associated with a module. Associate items within the module that makes most sense.
Name	companies	This name will appear in the list of picklists so it should be meaningful
Data Source	IssueTrak	The datasource from which to execute the SQL statement to retrieve the picklist values
Source	SELECT company_id, company_nm, null, CASE ISNULL(is_deleted,0) WHEN 0 THEN 1 ELSE 0 END from company with(nolock) order by 2	The SQL statement for populating this picklist. It contains a column selection that marks whether an item can be selected but will still display if necessary.
Type	SQL	This is a SQL picklist
Attributes	Cache, LimitToList	Cached for performance purposes and the items are fixed by the SQL statement and cannot be added to.



Creating a Gender Picklist in Metadata

In the Studio Explorer, click on the Standard Picklist Link (or Alt+L, S)

Create a picklist with the following settings

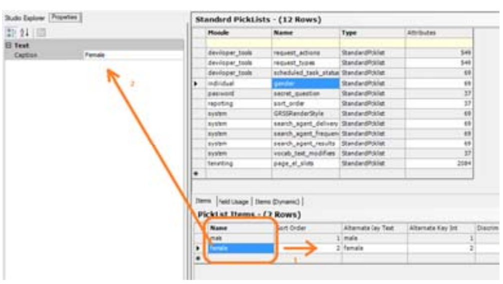
Item	Value	Notes
Module	individual	
Name	gender	
Type	StandardPicklist	
Attributes	Cache, LimitToList, StoreAltInt	Cached for performance purposes and the items are fixed by the SQL statement and cannot be added to. The key value of the picklist is an integer type.

And then in the Items subform add these items

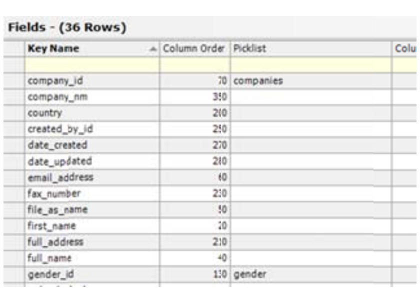
Name	Notes
male	
female	

1) Note that the remaining data items are populated automatically by the studio, but you can override those

2) The display value of the picklist items are from the value in the Caption property.



Now go back to the mappers list (Alt+M). Select the Individual Mapper, and for the gender_id field, select the picklist to be "gender" and on the company_id field, set the picklist to be "companies"


Fields - (36 Rows)

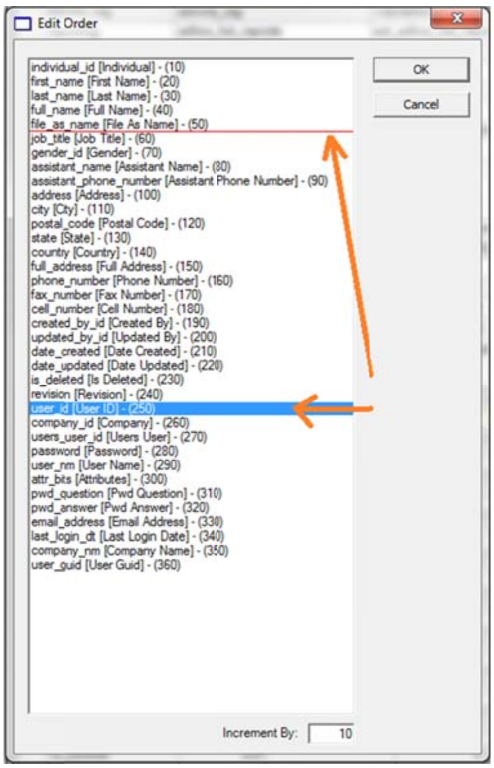
Key Name	Column Order	Picklist
company_id	70	companies
company_nm	350	
country	200	
created_by_id	250	
date_created	270	
date_updated	280	
email_address	60	
fax_number	230	
file_as_name	50	
first_name	20	
full_address	210	
full_name	40	
gender_id	130	gender

Re-arranging Columns in the Mapper

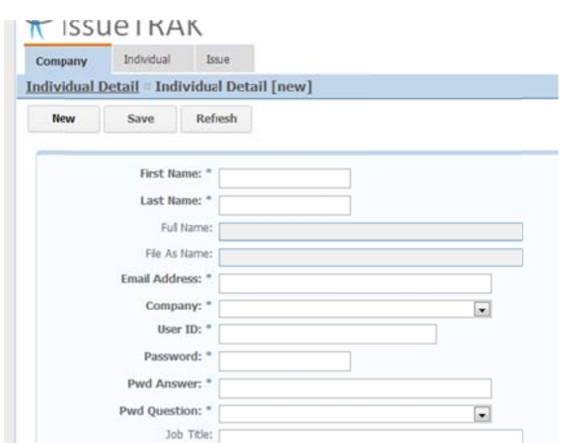
The column order defines the order in which a mapper's fields are displayed in the list view pages and the default 'flow' layout of detail pages. You can re-arrange the order of columns by manually modifying the column order values. However there is a visual tool in the studio to assist this process.

In the list of mappers (Alt+M), select the individual mapper and right click. Select the "Order Fields…" option.

In the dialog window that pops up you can drag and drop the fields to any position you would like.



You can see the effects of the changes by logging into the application. This example the fields have been rearranged with required fields at the top of the details.

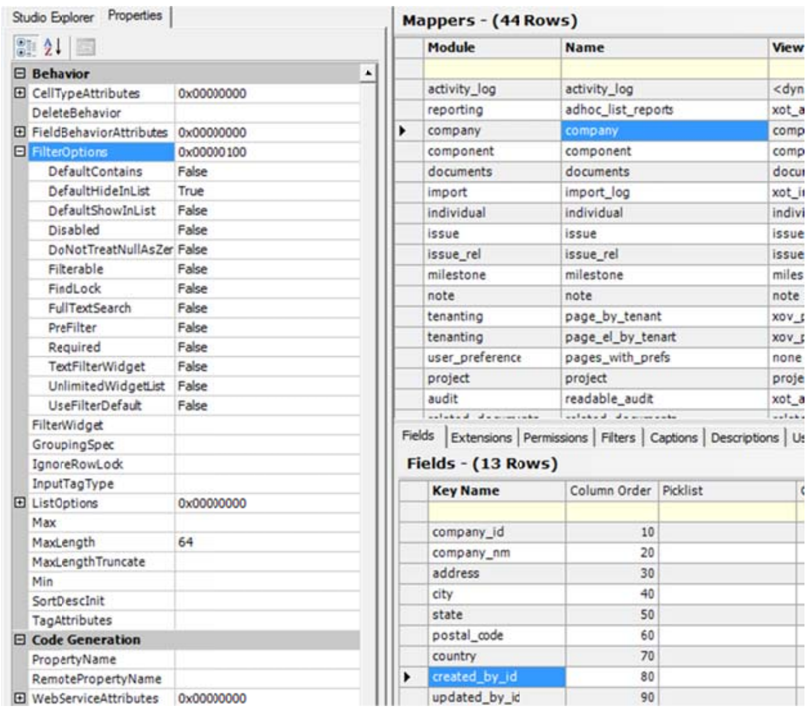


Showing Fields under Certain Circumstances

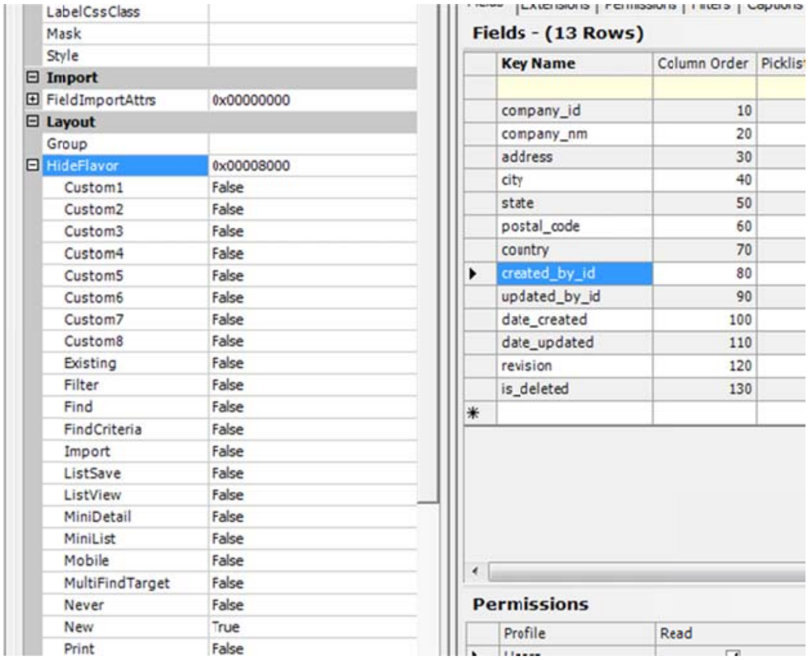
For every page displayed, the platform applies a set of one or more "Flavors" to a mapper which can be used to modify the way Fields are presented to the user. If you remember back to the application wizard. We selected a set of fields that represented specific pieces of information. For example. created_by_id represents who created a specific record.

The wizard applied certain modifying behaviors to the field.

The field is hidden when creating a NEW record, but is visible on an existing record.

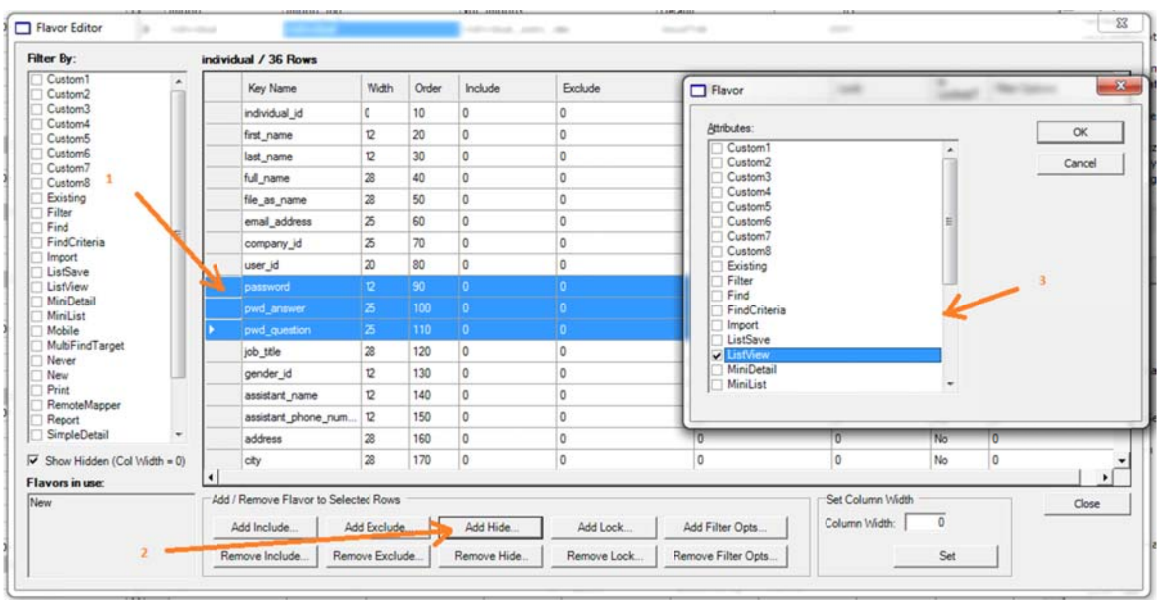


The field is hidden by default on a list, but can be made visible on the list.



You can individually make changes to the fields behavior with respect to flavoring and visibility on the list. However, there is a tool that allows you to make bulk changes to these properties.

In the list of mappers (Alt+M), select the individual mapper and right click. Select the "Flavor Editor" option.

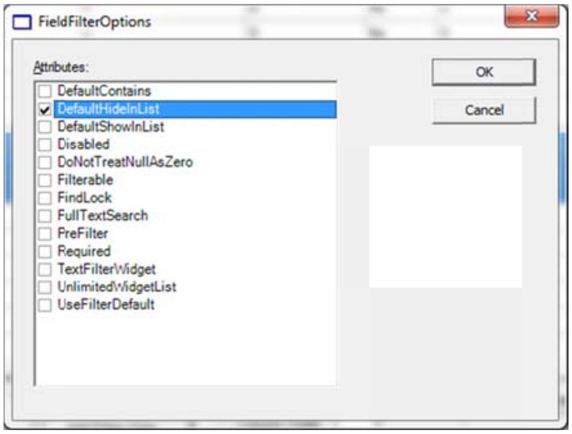


1) Select one or more rows from the list of fields in the mapper by clicking on the header row of the grid. This enables the Add/Remove buttons

2) Choose one of the types of Flavor behavior to select. This example is selecting a "Hide" flavor which means when the mapper has that flavor applied by the platform, the selected fields will be hidden.

3) Choose one or more Flavor attributes.

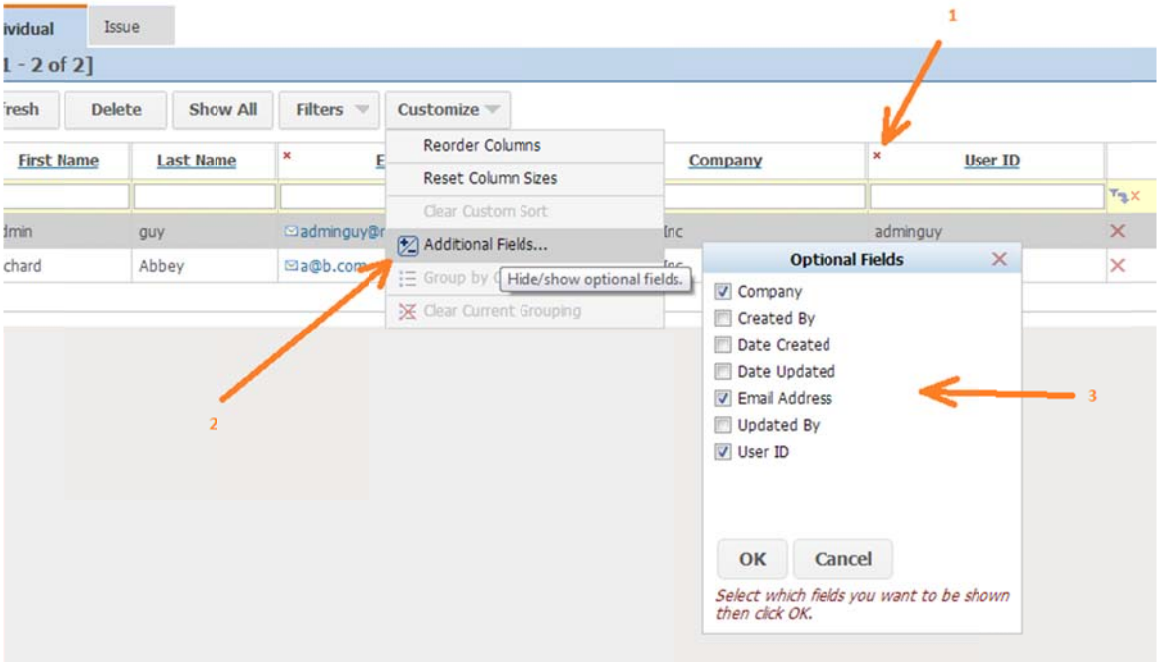
To make a field optionally hidden on the list (but can be displayed), you select the field(s) in question and click on the Add Filter Opts… button. Presenting a different set of options. Select "DefaultHideInList"



Use the Flavor Editor to make the following changes

Attributes	**Flavor**	**Fields**
Set Hide Flavor	ListView	password, pwd_question, pwd_answer
Set Filter Options	DefaultHideInList	full_name, file_as_name, job_title, gender_id, assistant_name, assistant_phone_num, address, city, postal_code, state, country, full_address, phone_number, fax_number, cell_number
Set Filter Options	DefaultShowInList	email_address, user_id, company_id

Logon to the application and see the effects of these changes.



1) Fields that are "optional" have a small red 'x' that you can click on to hide the column

2) Optionally visible fields are managed through the Customize button, Additional Fields.

3) This pops up a window to check (visible) or uncheck (hide) fields from the list.

Adding Authentication Support

The application currently allows you to logon by providing any user name and any password (even blank). Now, however, you can create individuals/users with a password so we should now take advantage of this functionality.

One important piece of functionality we need to add is the encryption of user's passwords because they are currently stored in plain text in the database even though the values are masked by the platform.

Encrypting the Passwords

The platform comes with an extension for user administration that can perform the encryption of passwords. This extension works because it expects to see certain fields in the mapper it is attached to. These fields are provided by the users table.

An extension is an object that responds to one or more commands fired by the mapper, or other platform objects where you can perform custom actions. Typically business rules that cannot easily be handled through metadata settings.

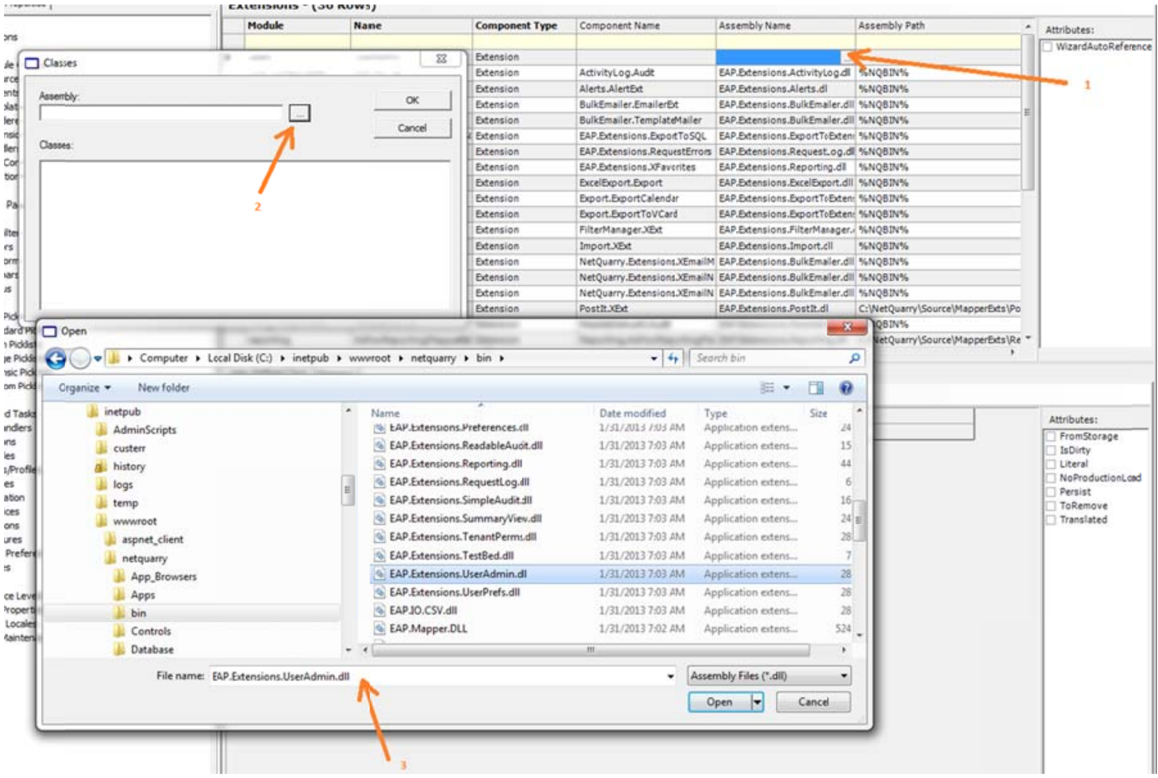
You need to attach a platform provided extension to your individual mapper. The name of the extension is EAP.Extensions.UserAdmin.dll. Before you can attach the extension to a mapper, it has to be registered as a component in the metadata repository. To do this, follow these steps:

In the studio go to the Extensions list under Components and add the following item

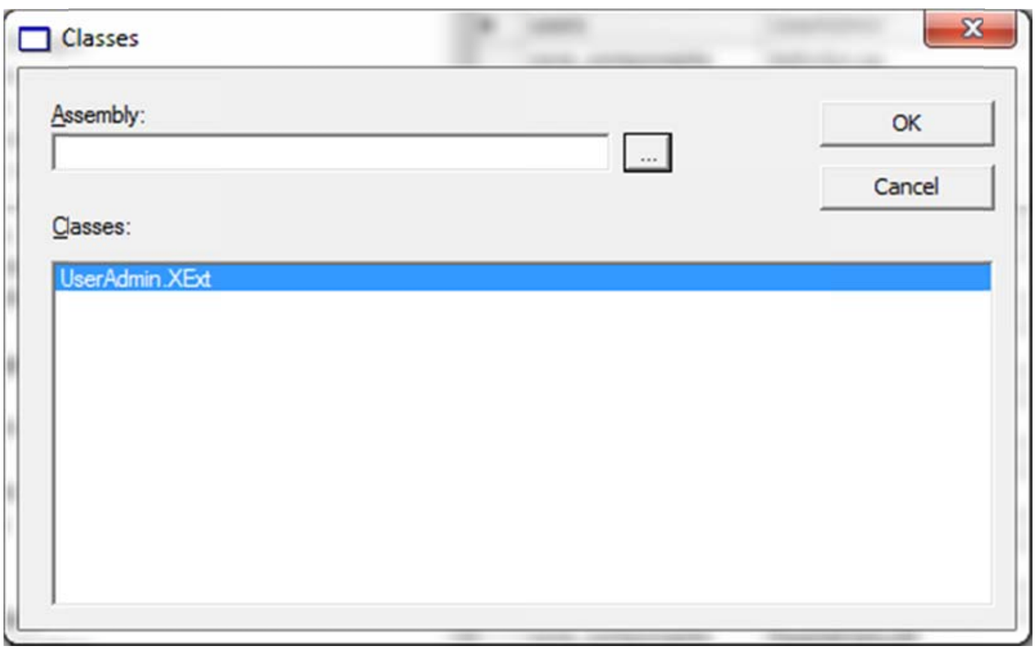
Item	Value	Notes
Module	users	
Name	UserAdmin	
Component Type	Extension	

Move off the row to save, and then click on the same row to select it. Referring to the screenshot below,

1) Click on the small button on the right side of the Assembly Name cell to open the "Browse For Classes" dialog.
2) Then click on the button to the right of the Assembly field, to open the File Open window.
3) In the File Name box, enter %NQBIN% and hit enter, then browse down the list of assemblies, selecting the assembly "EAP.Extensions.UserAdmin.dll"

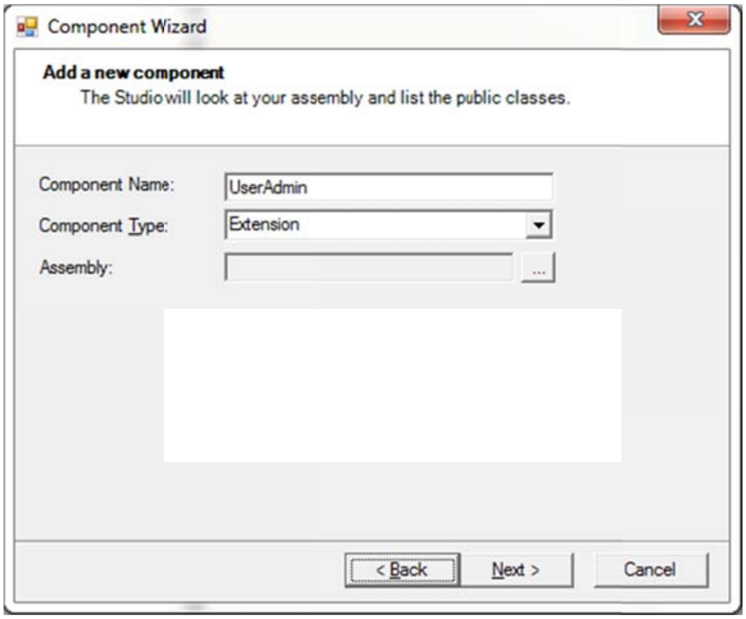


- In the list that is returned, select the class "UserAdmin.XExt.



If the extension contains multiple classes, then each class will be listed in the classes window.

Alternatively, you can use the New Component wizard available from the Home Page.



New Component Wizard

Now that the component is loaded, you need to add it to the list of extensions that will handle events for the individual mapper. To do this, follow these steps:

- Open the Home Page
- In the list of mappers (Alt+M), select the individual mapper
- Select the Extensions tab on the subform.
- Click on the cell of the Component column and press F9 to refresh the pick list to include the newly added extension.
- Add the extension called UserAdmin.XExt to the list and move down to save the row.

Add Extension to individual mapper

- Now add the same extension to the users mapper.

Setup Database Authentication

In the list of Applications (Alt+A), select the IssueTrak application. Then click on the Properties sheet

We're going to set the properties associated with the Security category.

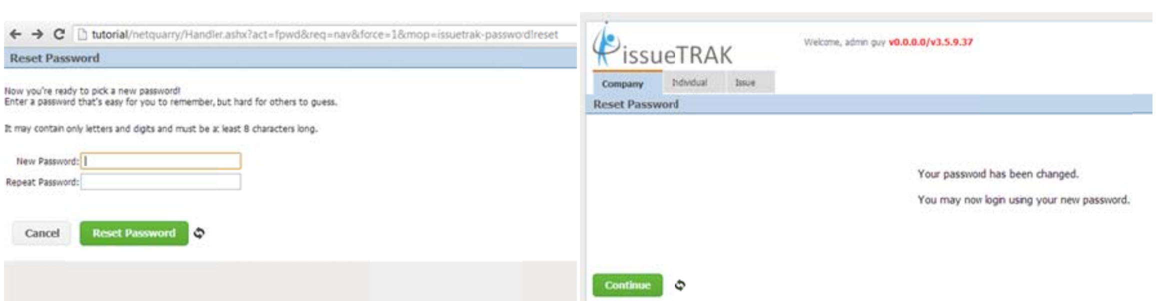
Property	**Value**	**Notes**
AuthenticationProvider	DatabaseAuthenticationProvider	Tells the platform to use the built in authentication against the users table
AutoLogout	True	Will auto logout a users session when the user's token has expired (set by TokenValidHours)
MaxFailedLogons	6	Maximum number of failed logon attempt before account is locked out
ResetPasswordPage	IssueTrak-password!reset	
TokenValidHours	1	Hours for validity of the user's session
UserAuthDatabase	IssueTrak	Which datasource is to be used to lookup the authentication information

Now log back into the application with the following user credentials

UserID: adminguy

Do not provide a password.

You will be forced to reset the password and then asked to login again.



The adminguy user was seeded into the database when you ran the database scripts to create the IssueTrak schema.

User Roles and Permissions

We have nearly completed the initial set up of the application. Getting the basic infrastructure in place. The last step is to setup the application to associate users with roles. Roles are used as a way to assign permissions to a user. The permissions associated with the roles are controlled at the level of an object. E.g. Whether a page is visible, or a mapper allows new records to be created..

Before we consider permissions, we have to add roles to the metadata and provide a way to assign roles to a user. Without an associated role, a user cannot login.

Adding Roles to the Application

In the Studio Explorer, click on the Profiles link under Permissions.

Add the following records

Module	Name
IssueTrak-roles	System Admin
IssueTrak-roles	Manager
IssueTrak-roles	Developer

You should now have four profiles including the Users profile that was already there.

Associating Roles to a User

We will create a new page for administering application data and create a new navigation target on the main bar, to get to that page. The first item on this admin page will present a list of users. From that list we navigate to a user detail page where we can assign roles to a user.

Create a User Roles Picklist

Create a SQL Picklist with the following settings

Item	Value	Notes
Module	Users	
Name	user_roles	
Data Source	Repository	The Repository datasource is a special built in datasource name that always refers to the metadata database. In this case issues_meta
Source	SELECT role_nm, role_nm FROM xmt_profile_roles WITH(NOLOCK) ORDER BY 1	
Type	SQL	This is a SQL picklist
Attributes	Cache, LimitToList	

Apply User Roles Picklist to user_roles mapper

Now go to the mappers list and select the user_roles mapper.

Set the following values.

Field	Settings	Notes
user_id	Column Width:0	user_id is the foreign key to the users table so is not needed to be visible
role_nm	Attributes: Required Picklist: user_roles Cell Type: ComboBox	

Create an Admin specific page

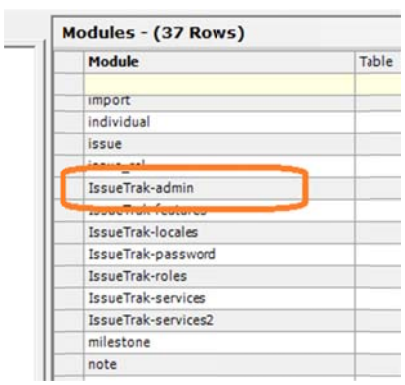
The pages we have created so far are based on an ExpandedSubformTemplate. Which presents either a list, or an editable detail. This new admin page will be presenting an array of disparate administrative information. For this we will use a page that is based off a ConsoleTemplate. You may have already seen an example of this type of page after saving records in the application. The save navigates you to a screen with static information.

Create a New Module

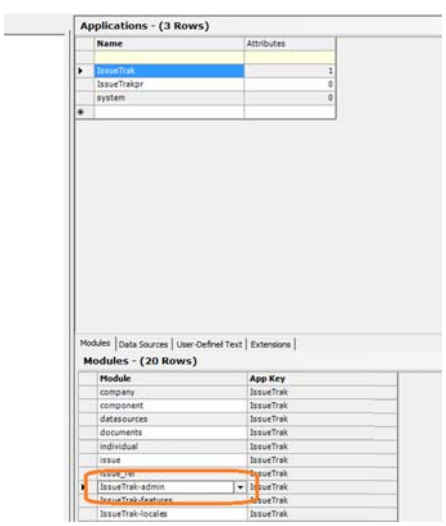
We are going to associate the page and some of its elements with a brand new module, called "IssueTrak-admin". Associating the page with an existing module would not quite work because this page will be hosting more than one category of data.

In the Studio, click on the Module link (or type Alt+U).

On the new row create a module with the name "IssueTrak-admin" and move off the row to save.



Now go to the Applications list (Alt+A). Select the IssueTrak application and click on the Modules subform. You should see that the module you just added has automatically been associated with the default application of IssueTrak (you may need to refresh the subform to see it, pressing F5)



Create an Admin Page

In the Studio, click on the Pages link (or type Alt+P).

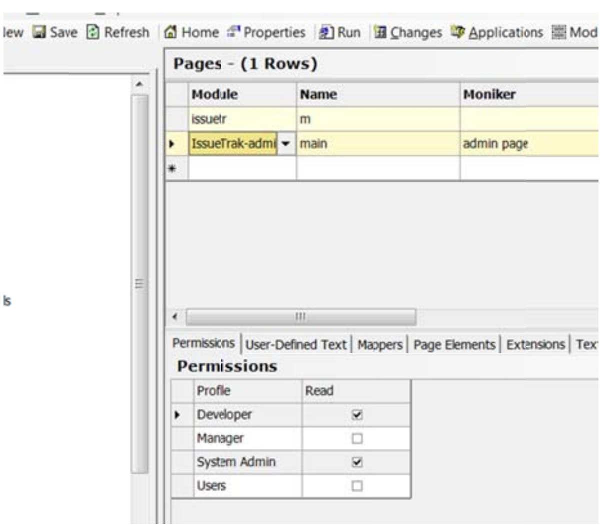
On the new row, create a new page with the following values.

Item	Value	Notes
Module	IssueTrak-admin	You may need to refresh the picklist by clicking on cell and pressing F9
Name	main	
Moniker	admin page	The moniker is a human readable name for the page that can be used to distinguish pages with the same module and name.
Template	ConsoleTemplate.aspx	
Mapper	<nothing>	This page will host content from different sources. So a single mapper to retrieve data does not apply.

With that page selected, set the following properties

The next screenshot shows a couple things

1) Using the filter by form row to restrict the list of pages in a list
2) how we restrict access to a page based on permissions. You give read access to System Admin and Developer role and deny read access to all other roles.



With that new Page selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Caption	Administration Tools	
Columns	2	
ColumnWidths	100%	Although there are two columns, we have only specified a single value for column width. The two columns of data will be sized automatically to 100% of the available width of the page. Which means that each column separately will be 50% of the page width. If you wanted to specify individual column widths, you could have specified “50%;50%”. The widths of each column separated by a semi-colon. Alternatively you can specify a fixed width with pixels”500px;450px”. or mixture of fixed width and floating “300px:100%”

Now we add a page element to the console page to display the list of users.

Click on the Page Elements subform.

Create a new Page Element with the following settings

Item	Value	Notes
Slot Name	Main	Every page has to have ONE page element that is in the Main slot
Name	users list	
Component	MiniList	Presents the data as a static list of items
Order	10	

When you move off the row to save, the Order value may change. Just change it back to 10.

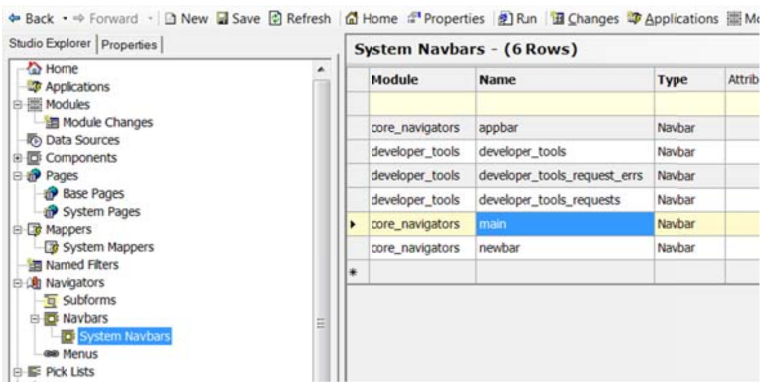
With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
PaneAttributes	FixedAtTop	The pane is fixed at the top of the page and extends across the width of the page
Mapper	users	Defines what data will be displayed by this page element
FieldList	user_id;user_nm;last_login_dt;email_address;attr_bits;user_agent;login_count;failed_logins;pwd_change_dt	The fields to be displayed in the list. You can copy and past from this document or click on the button in the cell of the property to display a field chooser
RowsPerPage	10	The number of user records to display in the list
DetailTarget	users!main	The target page to navigate to when you click on a specific row in the list of users.
LinkMOP	users!list	The target page to navigate to when you click on the caption at the top of the pane.
Caption	Users	The caption to use when referring to the page element when managing panes.
LinkCaption	Users (Click to View All)	The caption to use on the page element that links to the full set of data. If not specified, the Caption property is used in its place.

We will now add a way to get to this page via a new navigation target.

Create a Navigation Target

Go to the Navbars link under Navigators and then click on the System Navbars link, below that.



Select the "main" navbar and then in the Targets subform below, add a new Target with the following settings.

Item	Value	Notes
Module	IssueTrak-admin	
Name	admin page	
Target Type	Page	
Target	IssueTrak-admin!main	It is important to enter the correct target MOP. If you enter this data with a typo, and/or does not match the name of a page that exists in the application, then the navigation link will not appear.
Sort Order	40	

With that new Target selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Caption	Administration	

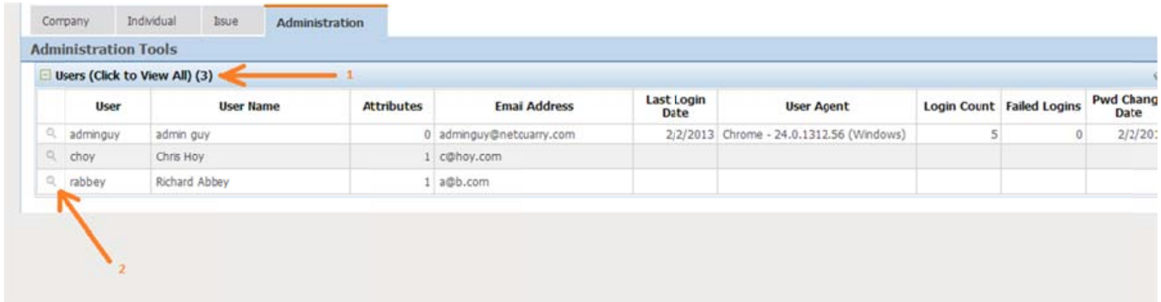
Now log out of the application and login again.

You should see the Administration link on the main bar and when you click on it, it will take you to the page you just created.

You can click on

1) The pane caption to take you to the full list

2) The detail of a particular user



Adding Developer Tools

Let's go back to the admin page and create a new pane with access to developer tools. This set of tools will only be available to a user with a developer role.

In the Studio, click on the Pages link (or type Alt+P). Make sure the IssueTrak-admin!main page is selected, then click on the Page Elements subform.

Create a new Page Element with the following settings

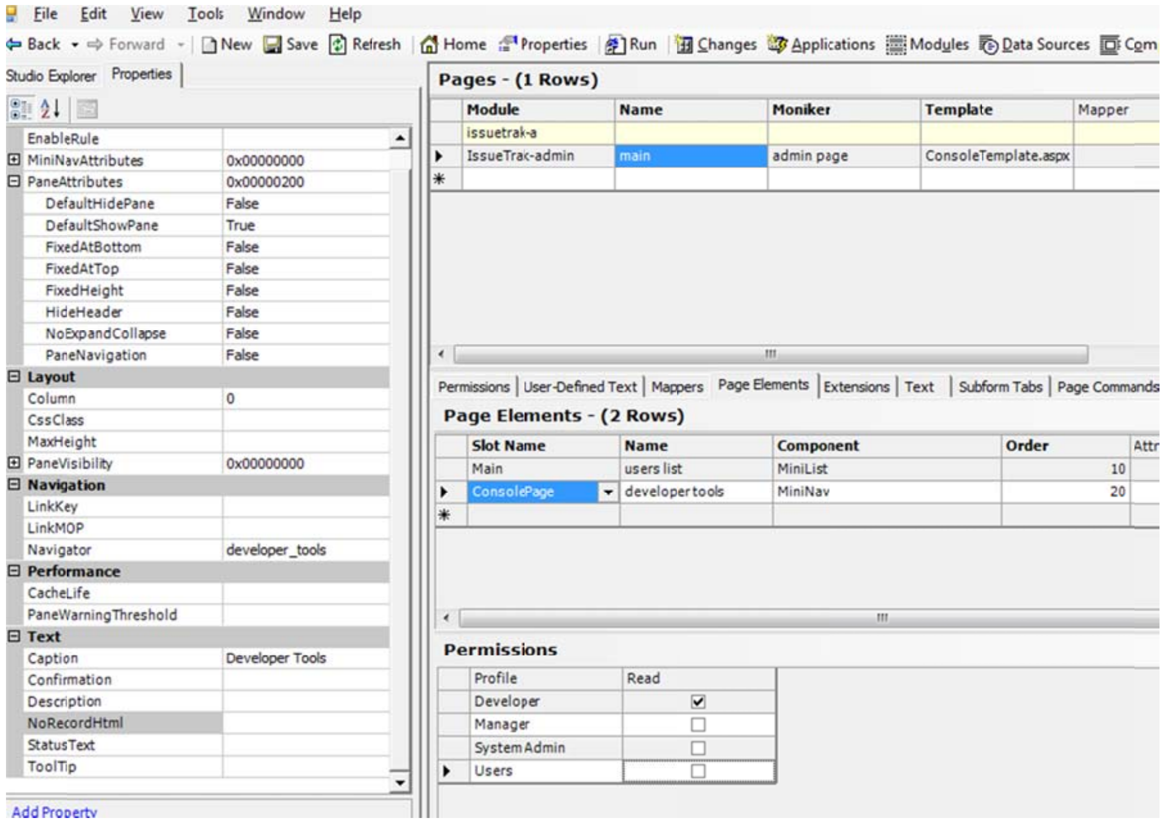
Item	**Value**	**Notes**
Slot Name	ConsolePage	Other elements on the console page have the ConsolePage slot
Name	developer tools	
Component	MiniNav	This will present a list of links from an existing navigator object.
Order	20	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	**Value**	**Notes**
Pane Attributes	DefaultShowPane	Similar concept to DefaultShowInList property of FilterOptions property of a field. You can have some panes be optionally visible on a page
Column	0	A console page by default has two layout columns and the page elements are laid out in left to right top to bottom order. You can force a pane to appear in a column by setting the value of the zero based index column property.
Navigator	developer_tools	The name of the Navigator which will provide the links for this page element
Caption	Developer Tools	

The next step is to set the permissions specifically for the Developer Tool page element.

In the subform below Page Elements, uncheck Read permissions for all profiles except Developer.



Adding a Policy

The last step to take, which is a developer tool specific requirement, is to add a Policy to allow certain roles to see the developer tools.

You can restrict access to the entire contents of a module by setting a RequiredPolicy property on the Module itself. To see this, go to the Modules list (Alt+U). Select the developer_tools module and click on the property sheet. You will see the RequiredPolicy property set to "DeveloperTools"

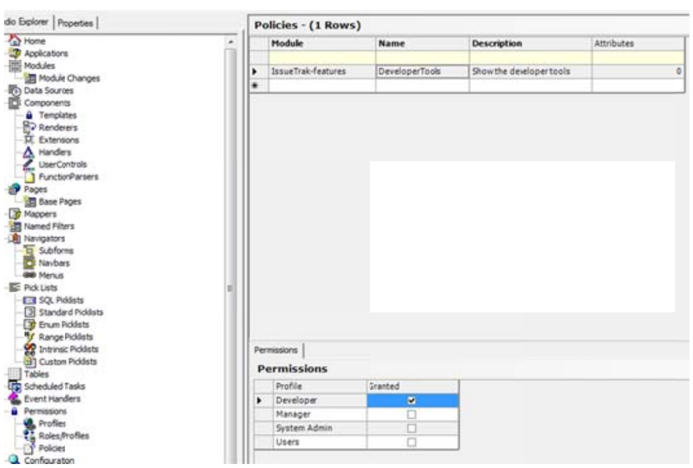
We have to add a policy with this name and grant access to that policy for certain roles.

Click on the Policies link under Permissions.

Add a policy with the following settings

Item	**Value**	**Notes**
Module	IssueTrak-features	
Name	DeveloperTools	This has to match the name in the RequiredPolicy property of the module
Description	Allow access to developer tools	

In the Permissions subform, uncheck all roles except for Developer. Then click the Save button.



If you logout and login and navigate to the Administration page, you will NOT see the Developer Tools pane because the "adminguy" user does not have the Developer role.

We will now add the necessary UI to the users page to manage the user roles

Relating User Roles to Users

In the Studio, click on the Pages link (or type Alt+P). Make sure the users!main page is selected, then click on the Page Elements subform.

Create a new Page Element with the following settings

Item	Value	Notes
Slot Name	ConsolePage	
Name	user roles	
Component	MiniList	
Order	20	

With that new Page Element selected, click on the Property sheet and set the following properties.

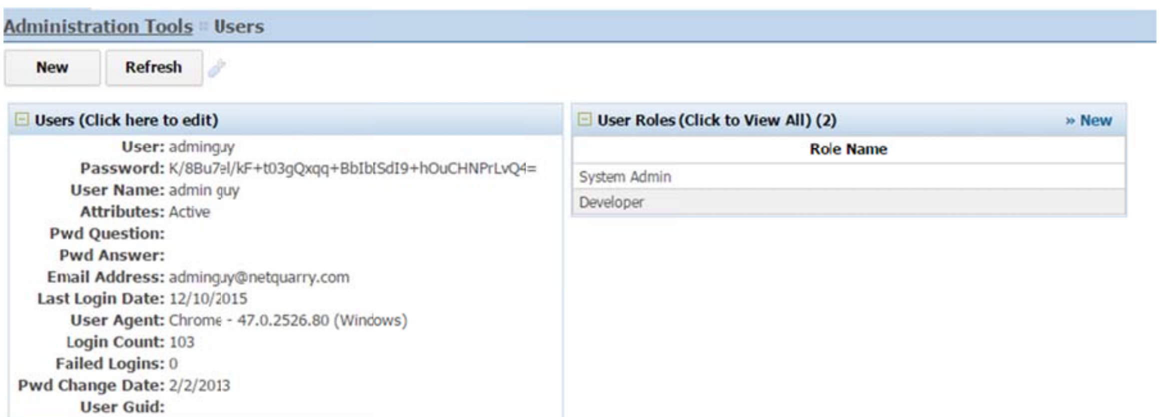
Property	Value	Notes
Pane Attributes	DefaultShowPane	Similar concept to DefaultShowInList property of FilterOptions property of a field. You can have some panes be optionally visible on a page
Mapper	user_roles	
ParentViewKeySource	user_id	We are going to show related user role information to a user record. We tell the platform how a parent user record relates to a set of user_role records. You define the parent field in the relationship (from the users mapper because that is the mapper used in the MAIN slot). The default parent field relationship comes from the primary key of the parent mapper (here, the user_id field). So we could not set the property, but it's always best to explicitly specify any relationships between objects
ViewKey	user_id	The foreign key side to the parent child data relationship. this is a field on the user_roles table that is filtered to give the correct user roles record. If nothing is provided, the platform assumes it is the same name as the parent key in the relationship. Again, we set the value explicitly for clarity.
RowsPerPage	10	
Sort	role_nm	This will override any default sort on the mapper. to sort descending, use role_nm DESC
LinkKey	user_id	
LinkMOP	user_roles!list	
NavigationAttributes	PopupDialog, PopupNew	
NewTarget	user_roles!detail	

Property	**Value**	**Notes**
Caption	User Roles	
Link Caption	User Roles (Click to View All)	
NoRecordHtml	This user has no roles. At least one role is required before the user can login.	

Logout of the application and login again.

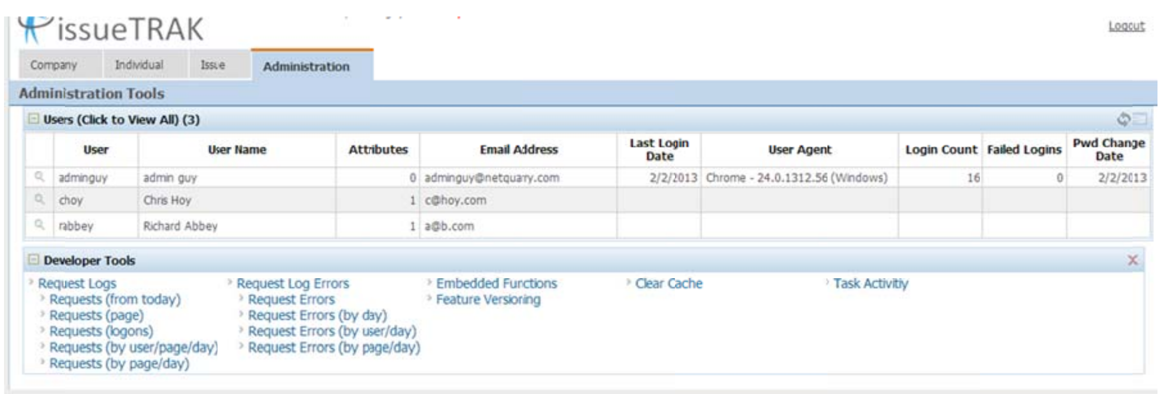
Go to the Administration page, and drill down onto the "adminguy" user.

You now see the detailed user information and next to it a list of user roles which currently lists System Admin and Users.



To add a new role, click on the "New" link on the right of the pane. In the popup, select the Developer role and Save.

Once saved. Logout and login again. Navigate to the Administration tab. You will now see the Developer Tools links.



Fixing the Users Interface

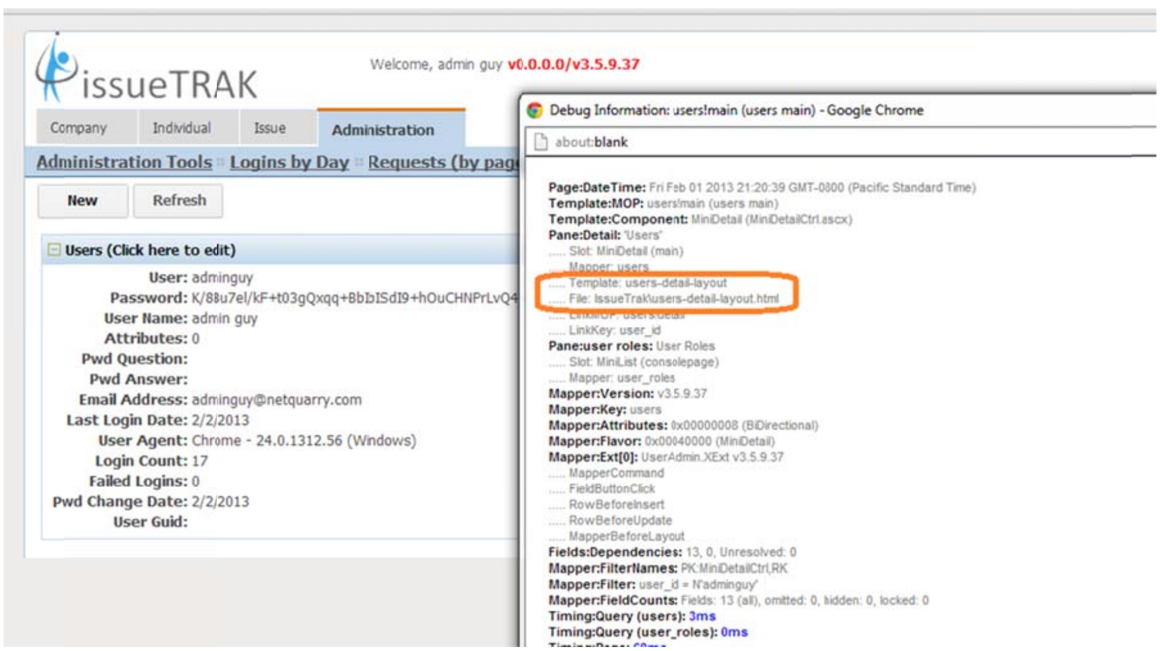
You may have noticed that the users interface displayed the encrypted password and secret question answers plus some other unnecessary fields or unformatted fields.

Modifying Templates

The UI for the user main screen (and all of the main object screens) is based on an HTML template file. When the Application wizard was executed, it saved a set of templates into the %NQROOT%\Templates\Issuetrak folder. These can be customized and modified to suit your own requirements.

To find the template used for a particular page you can either look in the studio at the Page, then page element properties. Find the right pane, look at the Template property. Then go to the Templates list in the studio, find the template and look at the FileName property of the template.

A quicker way to find out the template in use is to take advantage of the built in debugging tools. Click on a area of white space on the users!main page. Then press the F8 key. A popup of debug information appears. (you may need to disable popup blocking)



This instantly shows the name of the template object and the name of the template file.

Open the template file %NQROOT%\Templates\Issuetrak\users\users-detail-layout.html in an application you feel comfortable editing html content.

You will see that the content of the template is just a simple tabular layout with token strings representing the fields in the mapper. These tokens are called fields references and for templates, references to fields are enclosed with {{ and }}.

The {{&…}} syntax uses the special '&' modifier to change the meaning of the token. Instead of meaning replace the token with the display text of the field. It means replace the token with string representing the caption of the field. With this syntax we can create template files that dynamically pay attention to changes in field captions.

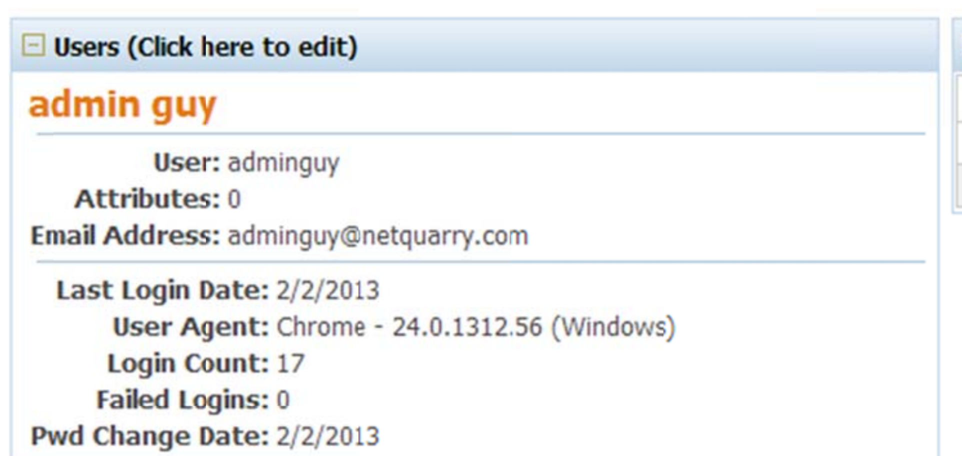
Delete the elements for password, pwd_question, pwd_answer and user_guid

```
<div class="mini-detail">
<table>
<tr><td class="mini-cap">{{&user_id}}:</td><td>{{user_id}}</td></tr>
<tr><td class="mini-cap">{{&password}}:</td><td>{{password}}</td></tr>
<tr><td class="mini-cap">{{&user_nm}}:</td><td>{{user_nm}}</td></tr>
<tr><td class="mini-cap">{{&attr_bits}}:</td><td>{{attr_bits}}</td></tr>
<tr><td class="mini-cap">{{&pwd_question}}:</td><td>{{pwd_question}}</td></tr>
<tr><td class="mini-cap">{{&pwd_answer}}:</td><td>{{pwd_answer}}</td></tr>
<tr><td class="mini-cap">{{&email_address}}:</td><td>{{email_address}}</td></tr>
<tr><td class="mini-cap">{{&last_login_dt}}:</td><td>{{last_login_dt}}</td></tr>
<tr><td class="mini-cap">{{&user_agent}}:</td><td>{{user_agent}}</td></tr>
<tr><td class="mini-cap">{{&login_count}}:</td><td>{{login_count}}</td></tr>
<tr><td class="mini-cap">{{&failed_logins}}:</td><td>{{failed_logins}}</td></tr>
<tr><td class="mini-cap">{{&pwd_change_dt}}:</td><td>{{pwd_change_dt}}</td></tr>
<tr><td class="mini-cap">{{&user_guid}}:</td><td>{{user_guid}}</td></tr>
</table>
</div>
```

Make any pretty layout changes to your template

```
<div class="mini-detail" "hover">
<h1>{{user_nm}}</h1>
<hr />
<table>
<tr><td class="mini-cap">{{&user_id}}:</td><td>{{user_id}}</td></tr>
<tr><td class="mini-cap">{{&attr_bits}}:</td><td>{{attr_bits}}</td></tr>
<tr><td class="mini-cap">{{&email_address}}:</td><td>{{email_address}}</td></tr>
</table>
<hr />
<table>
<tr><td class="mini-cap">{{&last_login_dt}}:</td><td>{{last_login_dt}}</td></tr>
<tr><td class="mini-cap">{{&user_agent}}:</td><td>{{user_agent}}</td></tr>
<tr><td class="mini-cap">{{&login_count}}:</td><td>{{login_count}}</td></tr>
<tr><td class="mini-cap">{{&failed_logins}}:</td><td>{{failed_logins}}</td></tr>
<tr><td class="mini-cap">{{&pwd_change_dt}}:</td><td>{{pwd_change_dt}}</td></tr>
</table>
</div>
```

And view the results by logging out and back into the application. Navigating to a user detail.



Create a user_attrs Standard Picklist with the following items, and associate the picklist to the attr_bits field on the users mapper.

Item	Value	Notes
Module	users	
Name	user_attrs	
Type	StandardPicklist	
Attributes	Cache, LimitToList, StoreAltInt	

And then in the Items subform add these items

Name	Alternate Key Int	Caption	Notes
active	0		
disabled	1		
force_password_change	4		
disabled_force_password_change	5	Disabed | Force Password Change	Modify the default generated caption

Saving Meta Data Changes

Now that we have started to add some real functionality, it would be a good time to save the metadata we have been adding and modifying. The metadata modules in the Studio are saved to XML files. One file per module. These files are checked in to source control and shared across your development team.

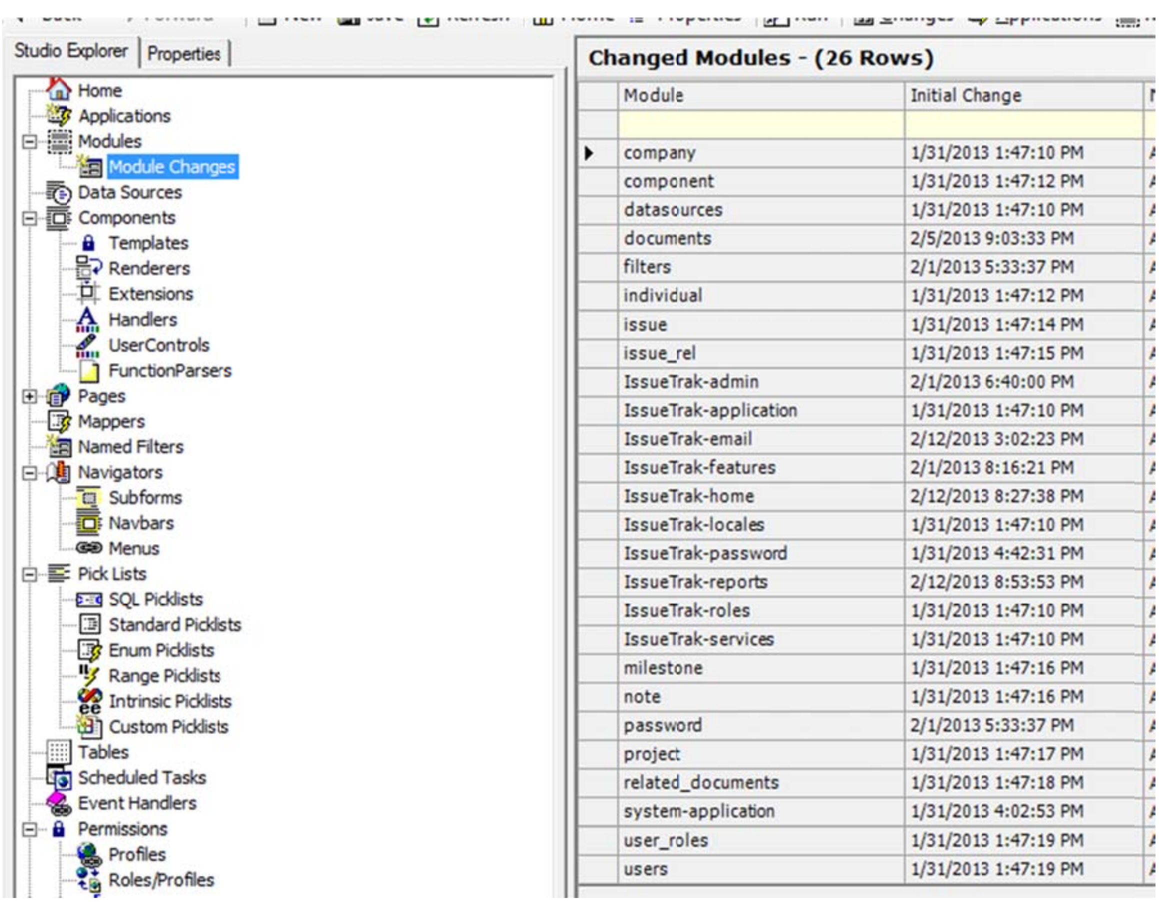
After saving the metadata and checking in the files, your colleagues will download the latest files from source control and run the Update IssueTrak DB batch file to execute the latest schema scripts against the operational database and load the metadata from the files into the metadata database.

It's always a good idea to make meatdata changes in small but discreet changes (meaning that they work and don't break other functionality). That way if you do make a mistake, you can go back to a previous state without losing too much work.

Saving Modules

The NetQuarry Studio keeps a track of which modules have been loaded since the last time the metadata was reloaded. It's not a rich set of history. It just records which modules were touched and when they were first touched.

This list is available from the Module Changes link under Modules



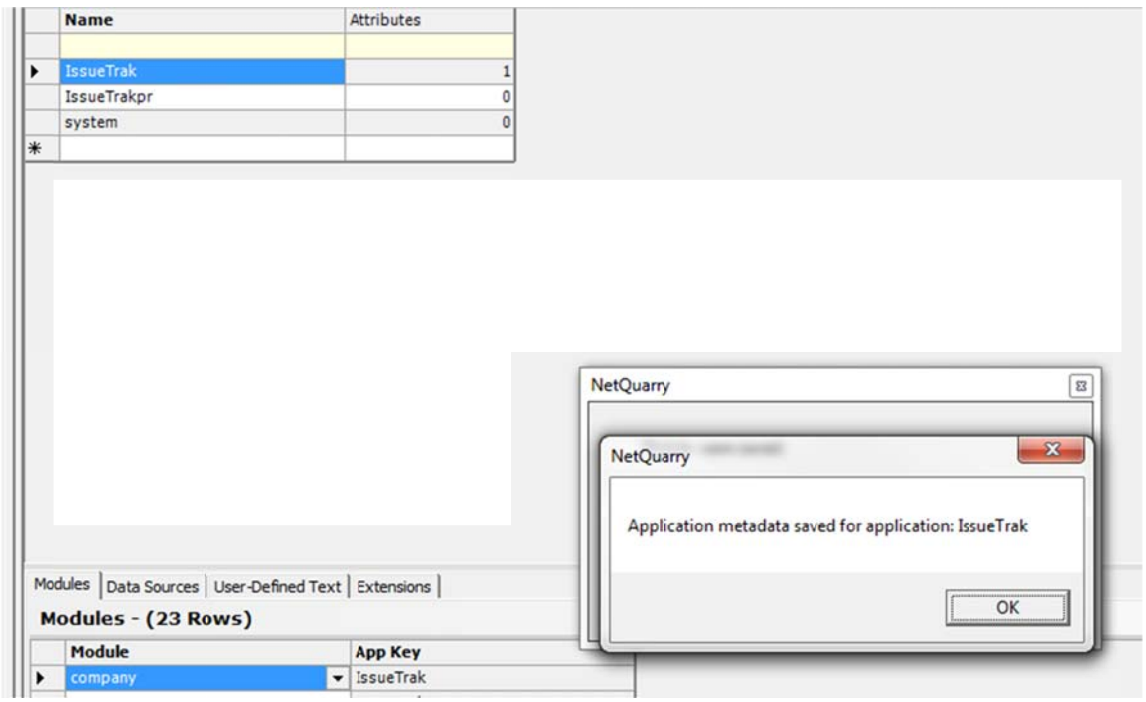
By now, your list will likely be as large as this one. However, this is NOT the ideal approach to creating content in the Studio. It's recommended that you work on a small subset of modules at a time and if possible regularly save your changes and check in.

From this list, you can individually save each module, by selecting a module, then right clicking on it and choosing "Save Module". You will be prompted for a location to save your metadata. You should choose the folder C:\Netquarry\Customer\IssueTrak\Database\Metadata

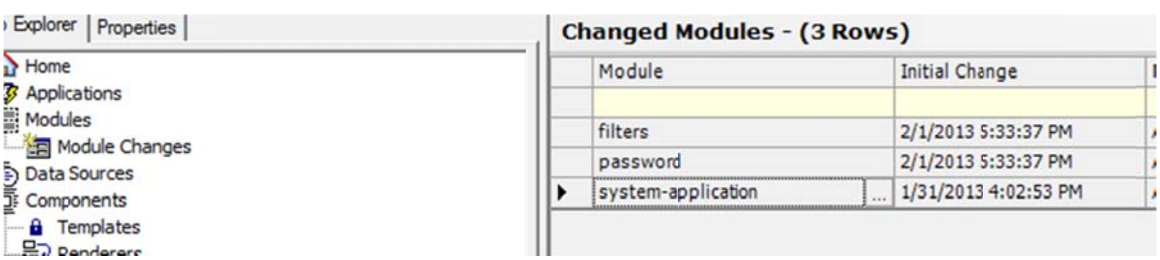
Once saved, the module drops out of the changed Module list.

To save all the modules in one pass, click on the Applications link (above Modules).

Then select the IssueTrak application. Right click and choose "Save All Changed Modules". You are again asked to select the save location. If you did that for one of the modules already, then the save folder location is remembered. Just click OK.



Having saved all the modules for your application, you may find that you still have some unsaved modules in the Changed Modules list.



If you have any modules in this list. It means you have modified metadata for system modules over which you have no control. The next time you reload the metadata, the system modules will be reverted back to their original state.

Creating Issues

We can now look at the task of creating isseus. To get issues created, we need to create some picklists and adjust some of the properties of the fields to force them to be required.

The database creation process seeded some data into the database to set up pickilsts used for selecting projects, components and milestones. This data will provide the content for SQL picklists.

Create SQL picklists with the following settings

The projects pickilst is an example of a 'grouped' picklist

Item	Value	Notes
Module	project	
Name	projects	
Data Source	IssueTrak	
Source	SELECT project_id, project_name, null, display, company_nm FROM project_view WITH(NOLOCK) ORDER BY company_nm, project_name	We are using a view to get to the project data where we have aliased a computed field to a display attribute based on whether both the company and project is not deleted. If a company is deleted, that meanse we can't be allowed to select any projects for that company. The additional select column denotes we want the picklist data to be grouped by company.
Type	SQL	
Attributes	Cache, LimitToList	

The components picklist is an example of a 'discrim' picklist

Item	Value	Notes
Module	component	
Name	components	
Data Source	IssueTrak	
Source	SELECT component_id, component_name, project_id, display FROM component_view WITH(NOLOCK) ORDER BY component_name	Similarly to component, we are getting the data from a view to derive the visibility of the record. Additionally the contents of this picklist are derived, or 'discrimmed' from another value. In this case the list of components depends on which project is selected. the discriminator is configured in the third column of the select statement.
Type	SQL	
Attributes	Cache, LimitToList, HasDiscrim	

The milestones picklist is an example of a 'discrim' picklist

Item	Value	Notes
Module	milestone	
Name	milestones	
Data Source	IssueTrak	
Source	SELECT milestone_id, milestone_name, project_id, display FROM milestone_view WITH(NOLOCK) ORDER BY CASE WHEN milestone_due_date IS NULL THEN 2 ELSE 1 END, milestone_due_date, milestone_name	We are sorting this picklist by the date but for milestones with no date, we force those to the end of the list and then those at end are sorted alphabetically
Type	SQL	
Attributes	Cache, LimitToList, HasDiscrim	

The users picklist is an example of a basic limit to list picklist

Item	Value	Notes
Module	users	
Name	user_list	
Data Source	IssueTrak	
Source	SELECT user_id, user_nm, null, CASE WHEN ISNULL(attr_bits,0) & 1 = 1 THEN 0 ELSE 1 END from users_view WITH(NOLOCK) ORDER BY 2	
Type	SQL	
Attributes	Cache, LimitToList	

Create Standard Picklists with the following settings

Item	Value	Notes
Module	issue	
Name	issue_category	
Type	StandardPicklist	
Attributes	Cache, LimitToList, StoreAltInt	
Items	bug, feature, inquiry	

Item	Value	Notes
Module	issue	
Name	issue_priority	
Type	StandardPicklist	
Attributes	Cache, LimitToList, StoreAltInt	
Items	critical, severe, moderate, low	

Item	Value	Notes
Module	issue	
Name	issue_status	
Type	StandardPicklist	
Attributes	Cache, LimitToList, StoreAltInt	
Items	open, resolved, closed	

Item	Value	Notes
Module	issue	
Name	issue_severity	
Type	StandardPicklist	
Attributes	Cache, LimitToList, StoreAltInt	
Items	system_down system_impaired_no_workaround system_impaired_with_workaround minor_problem workflow_improvement	

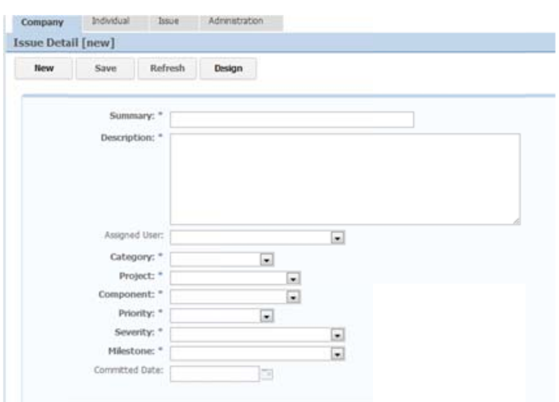
Tidy up Issues Mapper

In the list of mappers (Alt+M), select the issue mapper. And make the following changes

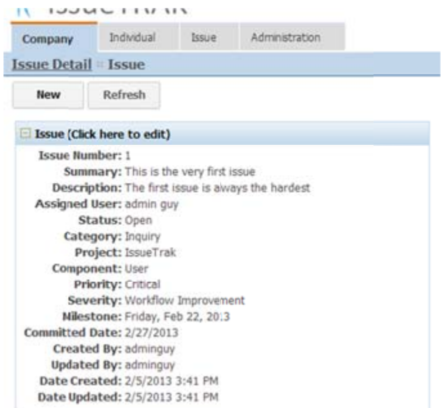
Field	Settings	Notes
issue_number	HideFlavor: New Format:0	The issue number is an identity field and the value is generated by SQL server. So the field is not necessary to be visible when creating a new issue. Setting the format to 0 (zero) means there is no thousands separator.
summary	Attributes: Required MaxLengthTruncate: True	Force the field to truncate to max length if more data is assigned to the field. Possible when value set programmatically.
description	Attribute: Required MaxLengthTruncate: True	
assigned_user_id	Column Width: 20 Cell Type: ComboBox Picklist: user_list	May need to click on the picklist cell for assigned_user_id field and press F9 to refresh the lists of picklists to get the user_list picklist to appear. Assigned to should probably be required, but later we will write an extension to default the value of the assigned_user_id if it is not explicitly specified.
status_id	Attributes: Required, Locked DefaultValue: 1 Cell Type: ComboBox Picklist: issue_status HideFlavor: New	The status_id field should not be modified directly. We will only change value of this field via workflow wizards.
category_id	Attributes: Required Cell Type: ComboBox Picklist: issue_category	
project_id	Column Width: 15 Attributes: Required Cell Type: ComboBox Picklist: projects	

Field	Settings	Notes
component_id	Column Width: 15 Attributes: Required Cell Type: ComboBox Picklist: components Discrim: [project_id]	The Discrim property specifies the value that is passed to this field and the picklist handling to filter the picklist items to display.
priority_id	Attributes: Required Cell Type: ComboBox Picklist: issue_priority	
severity_id	Column Width: 20 Attributes: Required Cell Type: ComboBox Picklist: issue_severity	
milestone_id	Column Width: 20 Attributes: Required Cell Type: ComboBox Picklist: milestones Discrim: [project_id]	
committed_date	Cell Type: DatePicker	

Having made these changes, you can log into the application, navigate to the Issues list and click on the New Button



Enter some data and save. This takes you to the issue main screen



Feel free to edit the html layout template of the issue screen to make the layout more presentable.

We are going to add more panes to this page to provide additional functionality. Such as adding notes, documents and managing the workflow (status) of the issue

Adding Notes to Issues

Tidy up Notes Mapper

In the list of mappers (Alt+M), select the note mapper. And make the following changes

Field	Settings	Notes
related_id	Column Width: 0	
type_id	Column Width: 0	You could make this field visible and set via a picklist. You'd have to create the picklist and associate with this field.
note_text	Attribute: Required	

Creating Notes UI

This process is similar to how we added user roles to the users page.

In the Studio, click on the Pages link (or type Alt+P). Make sure the issue!main page is selected, then click on the Page Elements subform.

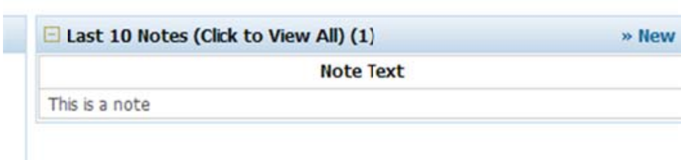
Create a new Page Element with the following settings

Item	Value	Notes
Slot Name	ConsolePage	
Name	notes	
Component	MiniList	
Order	20	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Pane Attributes	DefaultShowPane	Similar concept to DefaultShowInList property of FilterOptions property of a field. You can have some panes be optionally visible on a page
Mapper	note	
ParentViewKeySource	issue_id	
ViewKey	related_id	The note table is a generic table that could be applied to any main object, so we have the foreign key relationship to related_id field.
FieldList	note_text;date_created;created_by_id	note_text
RowsPerPage	10	
Sort	date_created DESC	See the most recent notes at the top of the list
LinkKey	issue_id	
LinkMOP	note!list	
NavigationAttributes	PopupDialog, PopupNew	
NewTarget	note!detail	
Caption	Notes	
Link Caption	Last 10 Notes (Click to View All)	

Log out and log into the application. Go to the issues you created (or create a new one). Then, from the issue!main page, you'll see the note pane. Click on ">> New" link and create a new note. When saved, you should see this…



If you remember, we actually wanted to see the date_created and created_by_id fields but they do not appear. This is because those fields were marked as DefaultHideInList on their FilterOptions property (by the Application creation wizard). The "List" meaning regular list, or mini list. There is no UI element to individually manage field visibility on different mini list panes.

You could simply take that attribute off the two fields, to see it here, but you would then see it on the regular list. So what we can do here is to add the same fields to the mapper but make it behave differently based on the Flavor applied by the platform to the mapper when it is rendered.

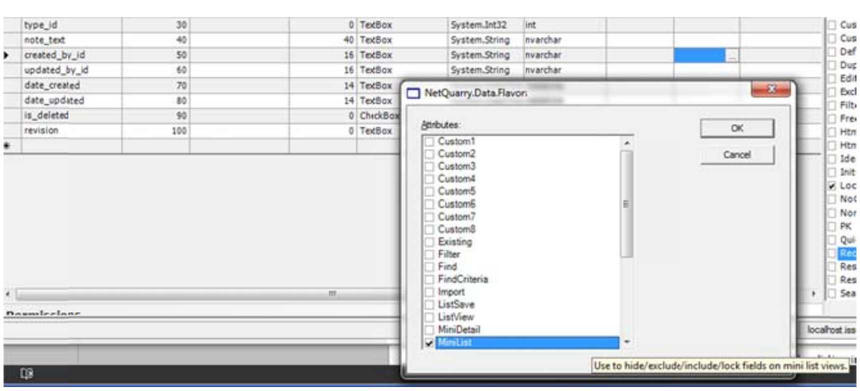
On the header of the Note pane, click on the link to 'View All'. This is what you will see

No Fields! The note field is

In the list of mappers (Alt+M), select the note mapper.

In the list of fields for the note mapper, click on the created_by_id field, then right click and choose "Duplicate". The list will refresh immediately to show the newly duplicated field. There is no obvious indication which field was the original and which was the duplicate. However, the original field is the one that still has the selected row indicator next to it.

On that field with the selector, click on the button in the Exclude Flavor cell and choose "MiniList"



And click OK.

Then select the date_created field, then right click and choose "Duplicate". On that same field set the Exclude Flavor to MiniList.

You will now see two created_by_id fields and two date_created fields. The order in which the fields appear in your Studio may not be the same as in this screenshot .

In the fields that are duplicated that have no attributes in the Exclude Flavor cell, enter the following settings

Field	Settings	Notes
created_by_id	Include Flavor: MiniList (131072) FilterOptions: 0 (Remove DefaultHideInList)	
date_created	Include Flavor: MiniList (131072) FilterOptions: 0 (Remove DefaultHideInList)	

Logout and login to the application to confirm the date_created and created_by_id fields are visible on the issue main page and still (default) hidden on full notes list.

Adding Documents to Issues

Set up the Related Documents Mapper

In the Studio, click on the Mappers link (or Type Alt+M). Select the related_documents mapper.

Make the following changes to the fields.

Field	Settings	Notes
file_name	Attributes: Required	
	CellType: FileUploader	The cell type is changed to a file upload type which gives you the necessary UI to attach a file either from disk, or from a physical scanner.
	FileContentField: document_id	The FileContentField in this case stores the document_id of the document from the document storiage service
	FileUploadSource: Browse, Scanner	The types of document upload source available
	StorageService: DocStorage	
document_type_id	Attributes: Required Picklist: document_types CellType: ComboBox	

Add Documents UI

In the Studio, click on the Pages link (or type Alt+P). Make sure the issue!main page is selected, then click on the Page Elements subform.

Create a new Page Element with the following settings

Item	Value	Notes
Slot Name	ConsolePage	
Name	documents	
Component	MiniList	
Order	30	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Mapper	related_documents	
ParentViewKeySource	issue_id	
ViewKey	related_id	
FieldList	file_name;date_created;created_by_id	
RowsPerPage	10	
Sort	date_created DESC	
LinkKey	issue_id	
LinkMOP	related_documents!list	
NavigationAttributes	PopupDialog, PopupNew	
NewTarget	related_documents!detail	
Caption	Documents	
Link Caption	Last 10 Documents (Click to View All)	

You now have the ability to attach documents to your issue.

Auditing Changes

On many occasions you are required to keep a track of who change what and when. A simple form of that tracking is via the date created, date_updated, created_by_id and updated_by_id fields. However, the platform comes with a more refined set of auditing functionality.

To add auditing, you add the platform extension, ReadableAudit.Audit to a mapper. Then on the fields for that mapper, set the Audit attribute.

Then provide a UI element to display the audited information.

Add Auditing to the Issues Mapper

In the list of mappers (Alt+M), select the issue mapper. Click on the Extensions subform. In the new row, choose the ReadableAudit.Audit extension from the picklist. Move off the row to save.

Select the issue mapper again and set the following field attributes.

Field	**Settings**	**Notes**
summary	Attributes: Audit	
description	Attributes: Audit	
assigned_user_id	Attributes: Audit	
status_id	Attributes: Audit	
category_id	Attributes: Audit	
project_id	Attributes: Audit	
component_id	Attributes: Audit	
priority_id	Attributes: Audit	
severity_id	Attributes: Audit	
milestone_id	Attributes: Audit	
committed_date	Attributes: Audit	

Now create the UI on the issues main page for the audit information.

In the Studio, click on the Pages link (or type Alt+P). Make sure the issue!main page is selected, then click on the Page Elements subform.

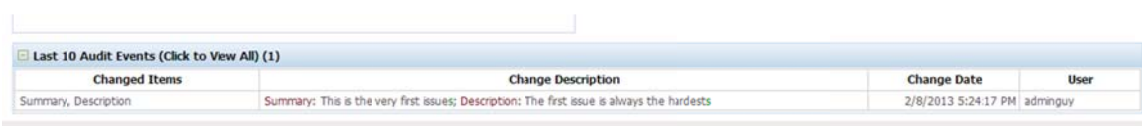
Create a new Page Element with the following settings

Item	Value	Notes
Slot Name	ConsolePage	
Name	audit	
Component	MiniList	
Order	40	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
PaneAttributes	FixedAtBottom	
Mapper	readable_audit	
ParentViewKeySource	issue_id	
ViewKey	rel_id	
RowsPerPage	10	
Sort	audit_dt DESC	
LinkKey	issue_id	
LinkMOP	audit!list	
Caption	Audit	
Link Caption	Last 10 Audit Events (Click to View All)	

Now go to the issue main page. Click on the Issue (Click Here to Edit) link on the caption of the issue pane. Make one or more changes to the issue information. Then save. When the page navigates back to the issue main page, the changes you made are recorded in the auditing information.


Last 10 Audit Events (Click to View All) (1)

Changed Items	Change Description	Change Date	User
Summary, Description	Summary: This is the very first issues; Description: The first issue is always the hardests	2/8/2013 5:24:17 PM	adminguy

Add Auditing to Objects Related to Issues.

If you add/modify or delete items related to the issue. Like Notes and Documents, you want those also to be audited. But audited with respect to the issue.

So add the ReadableAudit.Audit extension to the note mapper and the related_documents mapper. After adding the extension to these mappers, remember to go to the list of fields on these mappers and set the "Audit" attribute on the fields you wish to be audited.

E.g., on note mapper, set audit attribute on fields: note_text, is_deleted

on related_documents, set audit attribute on fields: document_type_id, file_name, description, is_deleted.

By default the audit logic looks at the field on the audited mapper that is the UniqueKey for that mapper. So when we make changes to issues, the audit record is related to the parent issue_id. Making changes to note, the audit record is related to the parent note_id. However, for auditing notes, we want the audit to be associated with the parent issue of the note. If you remember, we know the relationship between an issue and its notes. The foreign/parent key for a note is stored in the related_id field of a note. So we have to tell the auditing logic to use the related_id field of the note to associate to the audit record.

Go to the notes mapper, and click on the Extensions subform. Select the ReadableAudit.Audit extension and view the property sheet.

Set the following properties to modify the behavior of the auditing

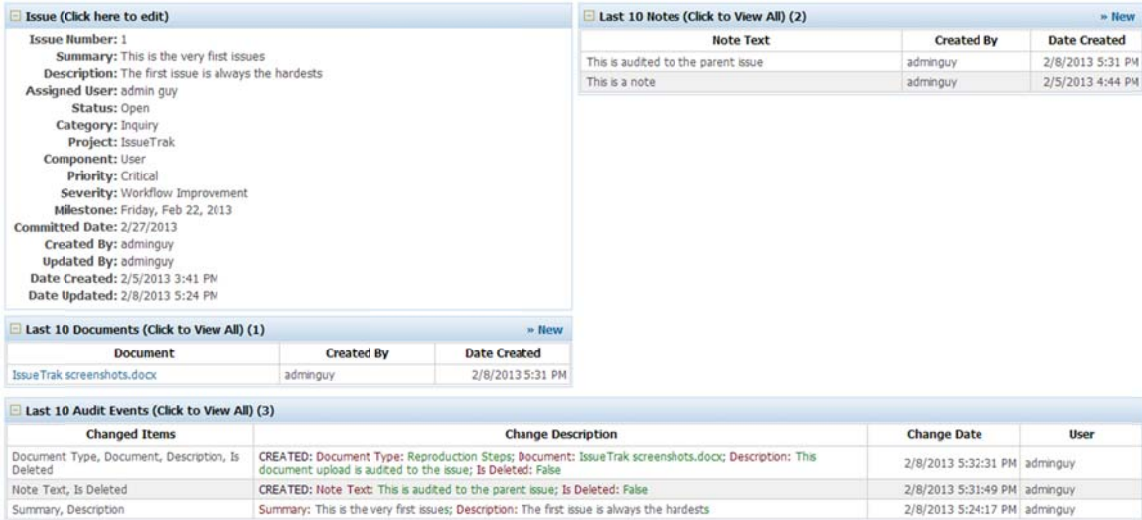
Property	**Value**	**Notes**
PKOVerride	related_id	

Go to the related_documents mapper, and click on the Extensions subform. Select the ReadableAudit.Audit extension and view the property sheet.

Set the following properties to modify the behavior of the auditing

Property	**Value**	**Notes**
AtomicFields	file_name	
PKOVerride	related_id	

Now when you create notes, or add documents to an issue, those modifications will be audited on the parent note.


Issue (Click here to edit)
Issue Number: 1
Summary: This is the very first issues
Description: The first issue is always the hardests
Assigned User: admin guy
Status: Open
Category: Inquiry
Project: IssueTrak
Component: User
Priority: Critical
Severity: Workflow Improvement
Milestone: Friday, Feb 22, 2013
Committed Date: 2/27/2013
Created By: adminguy
Updated By: adminguy
Date Created: 2/5/2013 3:41 PM
Date Updated: 2/8/2013 5:24 PM
Last 10 Notes (Click to View All) (2)
» New
Note Text
Created By
Date Created
This is audited to the parent issue
adminguy
2/8/2013 5:31 PM
This is a note
adminguy
2/5/2013 4:44 PM
Last 10 Documents (Click to View All) (1)
» New
Document
Created By
Date Created
IssueTrak screenshots.docx
adminguy
2/8/2013 5:31 PM
Last 10 Audit Events (Click to View All) (3)
Changed Items
Change Description
Change Date
User
Document Type, Document, Description, Is Deleted
CREATED: Document Type: Reproduction Steps; Document: IssueTrak screenshots.docx; Description: This document upload is audited to the issue; Is Deleted: False
2/8/2013 5:32:31 PM
adminguy
Note Text, Is Deleted
CREATED: Note Text: This is audited to the parent issue; Is Deleted: False
2/8/2013 5:31:49 PM
adminguy
Summary, Description
Summary: This is the very first issues; Description: The first issue is always the hardests
2/8/2013 5:24:17 PM
adminguy

Using Wizards to Manage Workflow

At the moment, all we can do is create an issue. There is no way to resolve, or close an issue, since we have locked the editing of the status_id field. So we will create simple wizards to resolve and close an issue.

To resolve an issue, we want to allow a user to modify a small subset of issue fields. Then force a user to provide a detail resolution note and then upload a "Solution" document.

To close an issue, we want to allow a user to modify a small subset of issue fields. Then force a user to provide a detail closure note and then upload a "Release Note" document.

Resolving an Issue

In the Studio, click on the Pages link (or type Alt+P). On the new row of Pages, create a new page with the following settings

Item	Value	Notes
Module	issue	
Name	wiz_resolve	You don't have to prefix the page name with the phrase "wiz" It simply helps for wizard pages to stand out from the crowd
Moniker	resolve an issue	
Template	WizTemplate.aspx	
Mapper	<nothing>	The wizard will host different pages so there is no need to specify a mapper

With that new Page selected, click on the Property sheet and set the following properties.

Property	Value	Notes
CancelAction	Return to Caller	The action to perform when the user clicks on the Cancel Button. The Action determines how to navigate away from the wizard.
FinishAction	Return to Caller	The action to perform when the user clicks on the Finish/Save Button. The Action determines how to navigate away from the wizard.
Caption	Resolve Issue	This will be the caption displayed on every wizard page of the wizard to give the user a context of the function of the wizard.
Finish_Caption	Resolve	This changes the default caption on the Save button on the last page of the wizard.

You've created Console pages with multiple page elements, where each page element is displayed on the same single page. Wizards provide a way to proceed through a set of steps where each step is represented by a page element on the wizard page.

We will create a three page wizard for resolving an issue.

Make sure the issue!wiz_resolve page is selected, then click on the Page Elements subform.

Create a new Page Element with the following settings

Item	**Value**	**Notes**
Slot Name	WizardPage	
Name	issue	
Component	WizPhantomDetail	
Order	10	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	**Value**	**Notes**
CopyValues	status_id=2	We specify a way to set the value of the status_id field on the mapper of this page to the value of 2. Which is the status_id of the value "Resolved"
InstanceName	resolve_issue	The instance name is a way of linking multiple pages of a wizard to point to the same mapper record. Pages that are linked have the same instance name and same mapper. If there are multiple pages with the same mapper defined, but have different instance names, will result in a wizard that acts on different physical records. The instance name can be anything. Even the same as the mapper name. For clarity of this tutorial, we set a specific instance name that is different to the mapper name.
Mapper	issue	issue
FieldList	assigned_user_id;category_id;committed_date;milestone_id	
Caption	Change Issue Details	This is the caption displayed for the specific page of the wizard and gives the user a context of the functionality of the current page.

Create the next Page Element for adding a note

Item	Value	Notes
Slot Name	WizardPage	
Name	note	
Component	WizPhantomDetail	
Order	20	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
CopyValues	related_id=[resolve_issue.issue_id]	On the previous page we used CopValues to set a literal value into the mapper instance of the first page. This time we are setting a value into the mapper of the note page. This time we are setting this value to a value from the "resolve_issue" instance. Specifically, the issue_id of the issue record will be set as the related_id on a new note record.
InstanceName	note	
Mapper	note	
FieldList	note_text	
Caption	Provide a Resolution	

Create the next Page Element for adding a document

Item	Value	Notes
Slot Name	WizardPage	
Name	document	
Component	WizPhantomDetail	
Order	30	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
CopyValues	related_id=[resolve_issue.issue_id];document_type_id=2	This example shows setting multiple values by separating with semi-colon. We're setting value from a different instance and a literal value.
InstanceName	document	
Mapper	related_documents	
FieldList	description;document_type_id;file_name	
Caption	Upload a Solution Document	

We now need a way to navigate to the Resolve Wizard.

In the Studio, click on the Navigators, NavBars link. On the new row of Navbars, create a new navbar with the following settings.

Item	Value	Notes
Module	issue	
Name	issue_workflow	
Attributes	AddParentInfo	This forces parent information on to all of the navbar's targets. This ensures that when you click on the workflow link it takes you to the wizard filtered to the correct issue.

Select the "issue_workflow" navbar and then in the Targets subform below, add a new Target with the following settings.

Item	Value	Notes
Module	issue	
Name	resolve	
Target Type	Page	
Target	issue!wiz_resolve	
Sort Order	10	

With that new Target selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Caption	Resolve Issue	

Click on the Pages link (or type Alt+P). Make sure the issue!main page is selected, then click on the Page Elements subform.

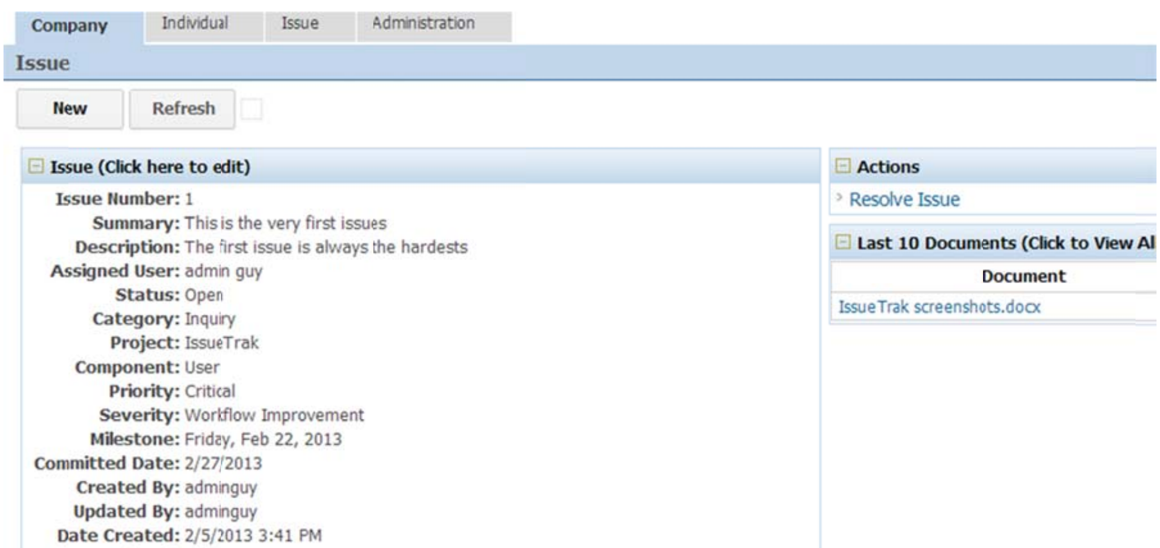
Create a new Page Element with the following settings

Item	Value	Notes
Slot Name	ConsolePage	
Name	workflow_actions	
Component	MiniNav	
Order	15	

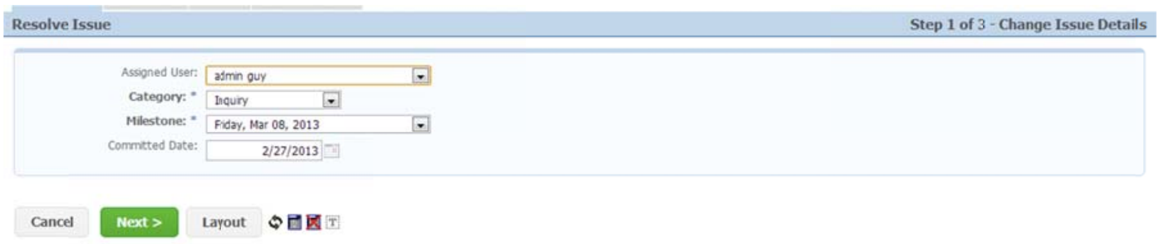
With that new Page Element selected, click on the Property sheet and set the following properties.

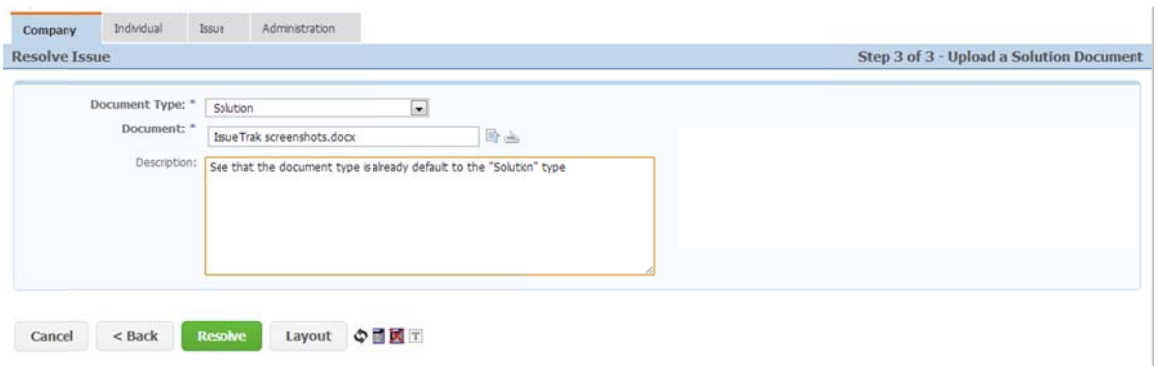
Property	Value	Notes
Column	1	
Navigator	issue_workflow	You may need to refresh the navigator picklist by clicking and pressing F9)
Caption	Actions	

Log out of the application and log in. Navigate to an issue detail. The actions menu will be visible with the "Resolve Issue" link

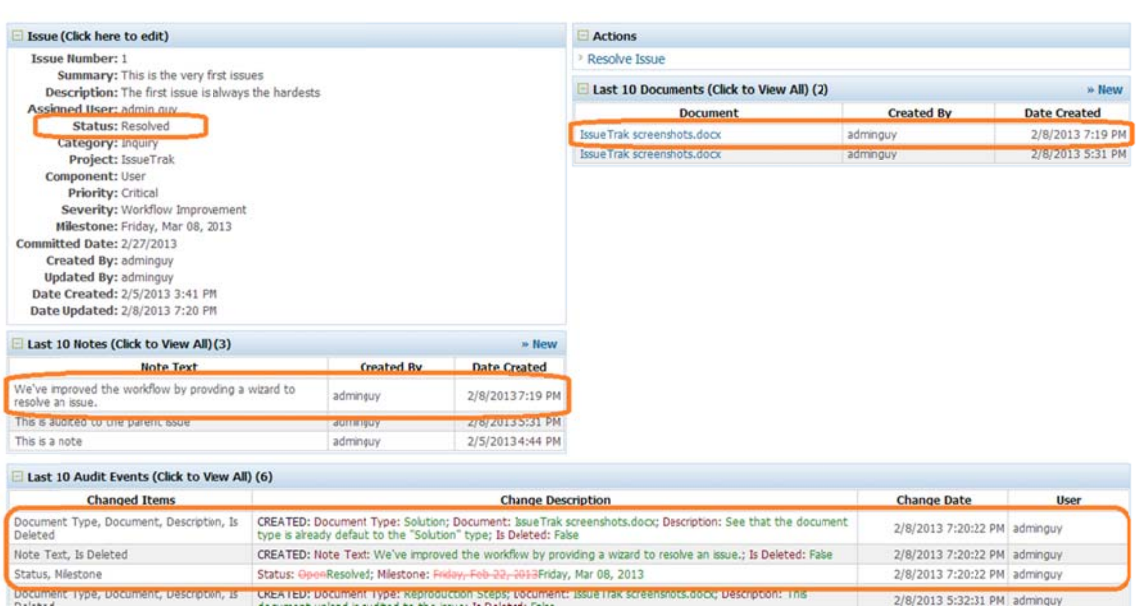


Click on the link to resolve an issue.





Log out and log into the application. Go to the issue you created (or create a new one). Then, from the issue!main page, you'll see the note pane. click on new and create a new note. When saved, you should see this…

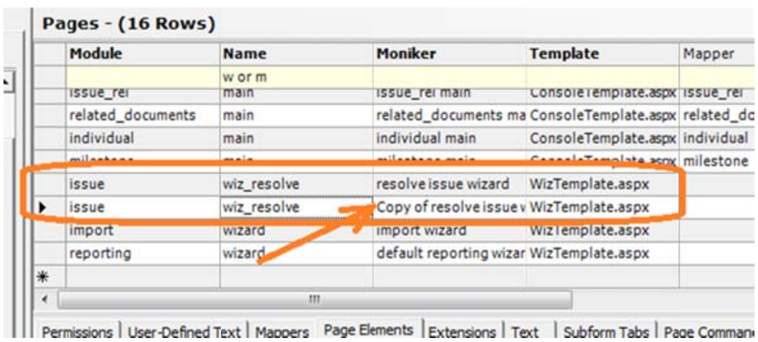


Closing an Issue

Closing an issue requires an almost identical set of steps to resolving an issue. Provide a way to change a few of the issue fields, provide a note and upload a document (for a release note)

You can manually create the page and navigation target, but this time we'll use the duplicate shortcuts.

In the Studio, click on the Pages link (or type Alt+P). Select the page issue!wiz_resolve. Right click and choose "Duplicate". Then press the F5 key to refresh the list. You'll see another record with the same page name, but a modified Moniker. The copied record will be Copy of resolve issue wizard.



For this duplicate record. Make the following changes

Item	Value	Notes
Name	wiz_close	
Moniker	close an issue	

With that new Page selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Caption	Close Issue	
Finish_Caption	Close	

With the issue!wiz_close page selected, now make modifications to the Page Elements properties. There is no need to change any of the actual main page element data.

With the issues Page Element selected, click on the Property sheet and set the following properties.

Property	**Value**	**Notes**
CopyValues	status_id=3	setting the status to "Closed"
InstanceName	close_issue	
Caption	Change Issue Details	

With the note Page Element selected, click on the Property sheet and set the following properties.

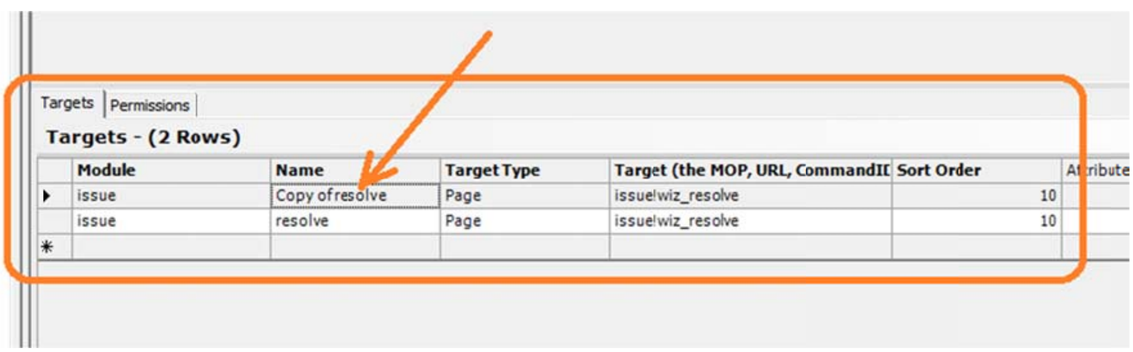
Property	**Value**	**Notes**
CopyValues	related_id=[close_issue.issue_id]	
Caption	Provide Closing Comments	

With the document Page Element selected, click on the Property sheet and set the following properties.

Property	**Value**	**Notes**
CopyValues	related_id=[close_issue.issue_id]; document_type_id=3	
FieldList	description;document_type_id;file_name	
Caption	Upload a Release Note Document	

Now we have to create a link to get to the Close Issue Wizard that is available on the issue workflow navigator.

In the Studio, click on the Navigators, NavBars link. Select the issue_workflow navigator. In the Targets subform, select the "resolve" item. Right click on the row and choose "Duplicate". Press F5 to refresh the screen.



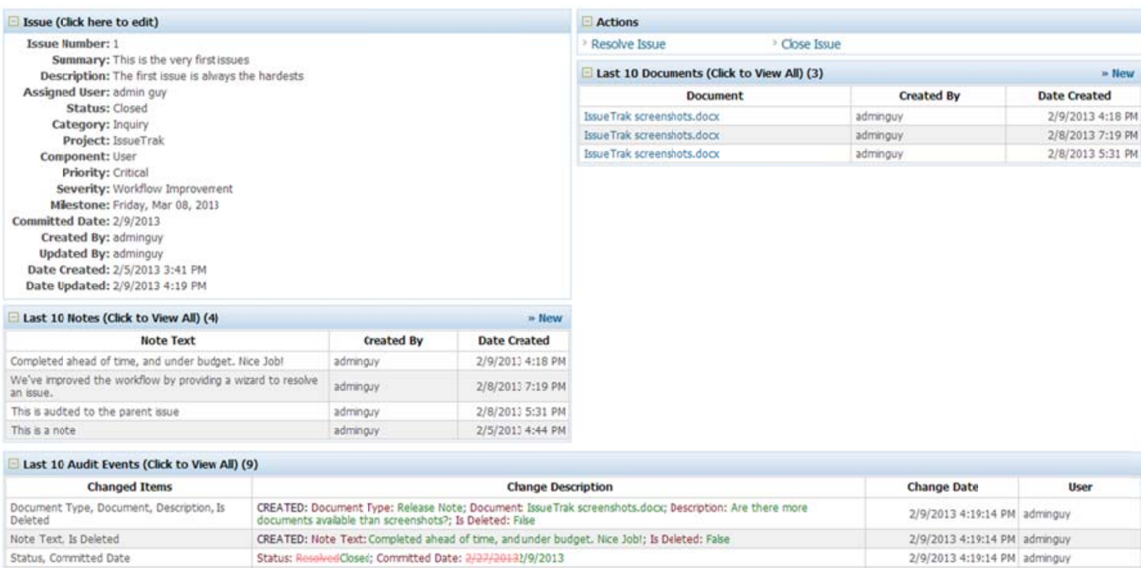
Make the following changes to the duplicated record “Copy of resolve”

Item	**Value**	**Notes**
Name	close	
Target	issue!wiz_close	
Sort Order	20	

With that new Target selected, click on the Property sheet and set the following properties.

Property	**Value**	**Notes**
Caption	Close Issue	

Log into the application and navigate to an issue main screen. You will now see an additional option to close an issue. Test the wizard to ensure you can close an issue.



Re-open an Issue

Now you should provide an additional wizard page to re-open an issue. This wizard is slightly different to the previous two wizards in that there will be only two pages. A page to edit the issue information, and a page to provide additional notes.

You can create the new pages and navigation links in the same way using the Duplicate right click menu option.

Hints:

An open issue has a status of 1

If you duplicate another page, you can select the document page element and delete that row.

Reject an Issue

Now create a wizard to reject an issue. This can be a duplicate of the Re-open Issue wizard requiring only two pages. For modifying issue data and providing a note.

Hints:

An open issue has a status of 1

Once you have completed the creation of the Re-open and Reject issue wizards, you will find the Actions Pane always has four actions available regardless of the status of the issue. Only some actions are relevant for an issue in a particular state. Which are…

Issue State	**Available Actions**
Open	Resolve
Resolved	Close, Reject
Closed	Re-Open

This type of state/workflow management is handled in code. We will come back to this problem and solve it when we look at the extension model of the NetQuarry platform.

Save Your Metadata

Now would be a good time to save your metadata changes.

Bulk Importing Data

The platform comes with the ability to import large amounts of data from spreadsheets or text files. Providing the ability to import data is as simple as adding an extension to a mapper. However, for a full feature rich import process, you will likely have to write code to support such additional functionality.

For this tutorial we are going to describe how to import a set of users into the application (with passwords and roles). And import a set of issues assigned to some of those users.

Importing Users

We have already modified the individual mapper to use a view that joins individual to a user. Although creating an individual automatically creates a user record. It does not create a user role record. At least one user role record is required to allow a user to login.

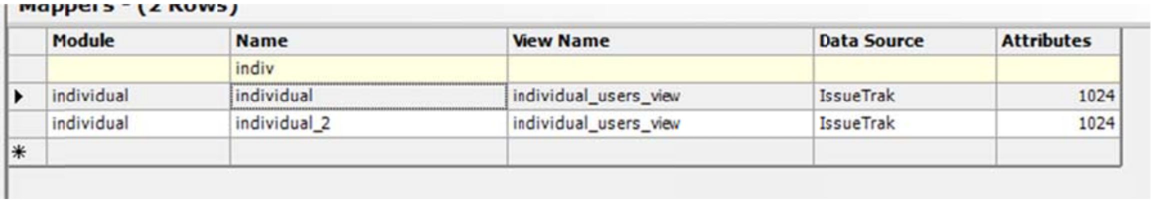
Using the same idea. We could add user_roles into the individual_user_view and create a new user role record when creating a new user. However, this will have an unwanted side effect when a user has more than one role. If a user has three roles, then the individual list would display three duplicate individual records (one for each user role).

To solve this, we will create an additional mapper based off the individual mapper, with the user_roles joined in. But ONLY use this for individual/user import.

Create the Individual Import mapper

In the studio go to the mappers list (Alt+M) and select the Individual Mapper.

Right click on the individual mapper and choose "Duplicate". Hit F5 to refresh the list.



Module	Name	View Name	Data Source	Attributes
	indiv			
individual	individual	individual_users_view	IssueTrak	1024
individual	individual_2	individual_users_view	IssueTrak	1024

Make the following changes to the individual_2 mapper

Item	Value	Notes
Name	individual_import	
View	individual_users_import_view	This view already exists and simply joins user_role table to the existing individual_users_view

Now move off the row and back to the individual_import mapper. Right click and choose "Create Fields…".

Select the three additional fields and click OK.

On the individual_import mapper. Make sure these fields have the following settings

Field	Settings	Notes
pwd_question	Attributes: None	Remove the Required Attribute
pwd_answer	Attributes: None	Remove the Required Attribute
user_role_id	Attributes: AutoAssign, NonNullable, PK Column Width:0 Table: user-roles	AutoAssign is automatically added to a field by the platform when the field is a PK and has an underlying type of uniqueidentifier, or string. A column width of 0 means the field is hidden.
user_role_user_id	ColumnWidth: 0 ColumnName: user_id Table: user_roles SourceField: user_id	This is the foreign key field user_id in the user_roles table that has been aliased in the view. We're copying the user_id value from the "individual" user_id field.
role_nm	Attributes: Required Table: user_roles	

Now use the Flavor Editor on the individual_import mapper to hide the following set of fields from appearing on the list of fields to import. (Check "Import" Attribute on hide flavor

pwd_question, pwd_answer, created_by_id, updated_by_id, date_created, date_updated, is_deleted, revision, users_user_id, user_nm, attr_bits, last_login_dt, company_nm, user_guid, user_role_id, user_role_user_id

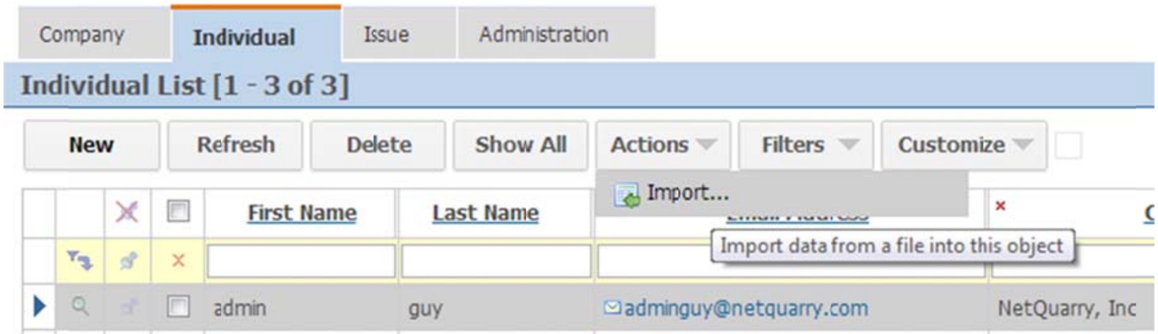
Finally. Click on the Extensions and add the UserAdmin.XExt extension. This will ensure the imported passwords are encrypted during the import.

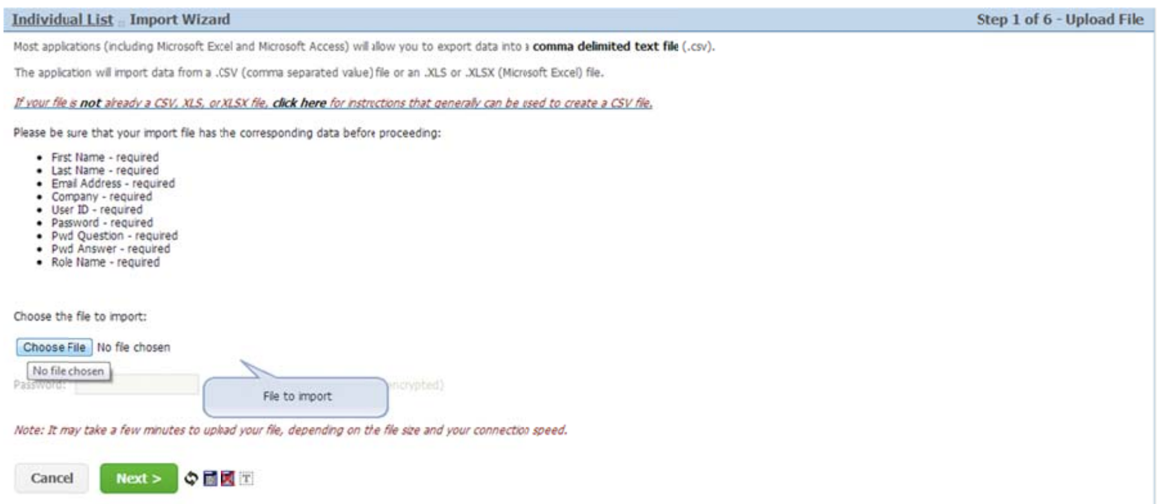
Now select the individual mapper (not the individual_import mapper) and click on the Extensions subform.

Add the Import.XExt extension to the mapper. Then, with the Import.XExt extension selected, click on the Property sheet and set the following properties.

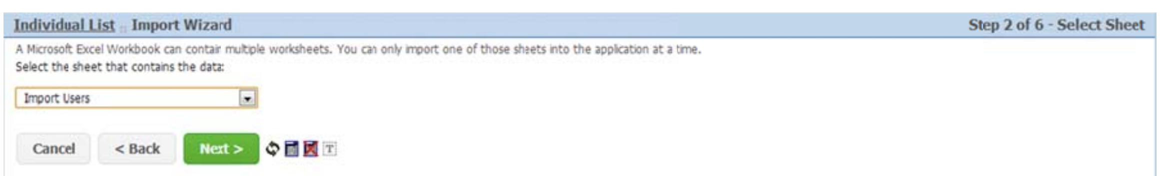
Property	Value	Notes
ImportWizardAttrs	NoPopup	The wizard will be opened in place and not as a popup
ImportWizardMapper	individual_import	If you don't specify a mapper, the import is performed on the same mapper as the page you executed the import (which in this case is individual).

Now log in to the application and navigate to the Individual mapper. Click on the "Actions" Menu. You will see the Import option has been added.

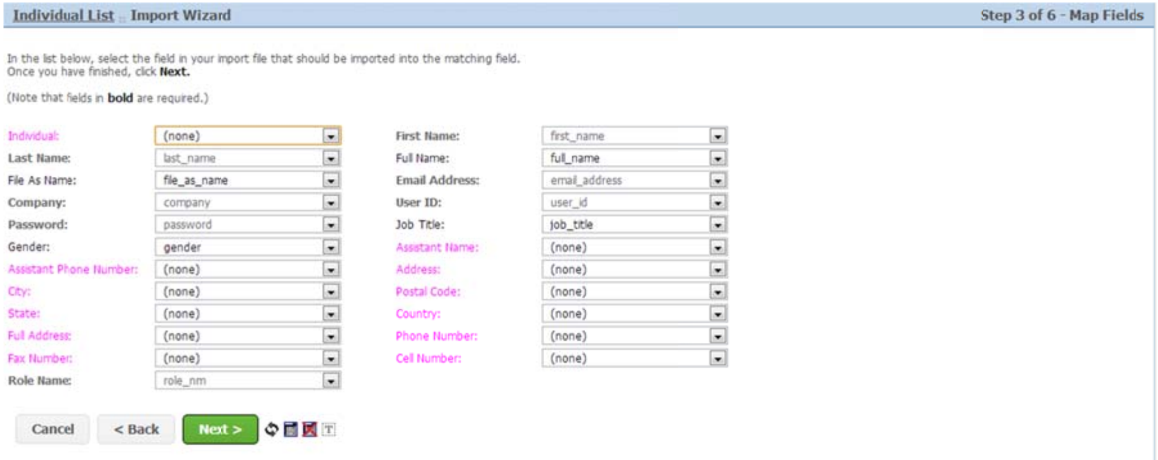




The installation came with a spreadsheet of users to import. You can find the spreadsheet by navigating to %NQROOT%\Documents\Tutorials\IssueTrak\Imports\Users.xlsx or C:\NetQuarry\Customers\IssueTrak\Imports\Users.xlsx



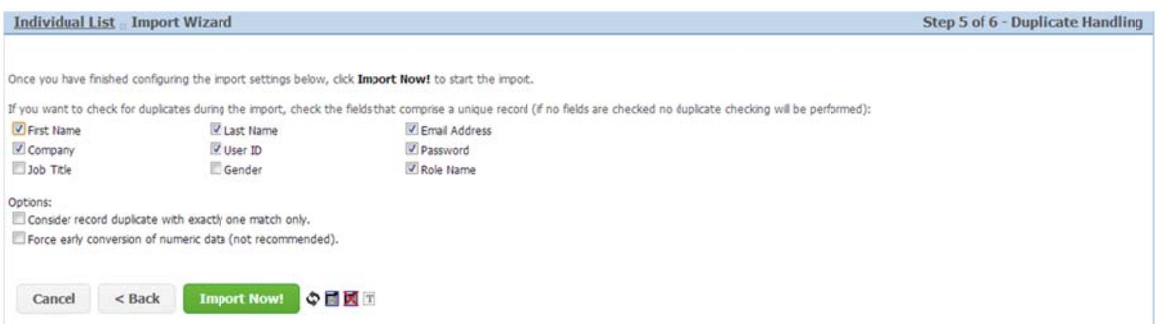
The data is located in "Import Users" sheet.



This page is where the mapping from spreadsheet columns to mapper columns is made. Required mapper fields are Bold and HAVE to be mapped to an equivalent spreadsheet field (from the dropdowns).

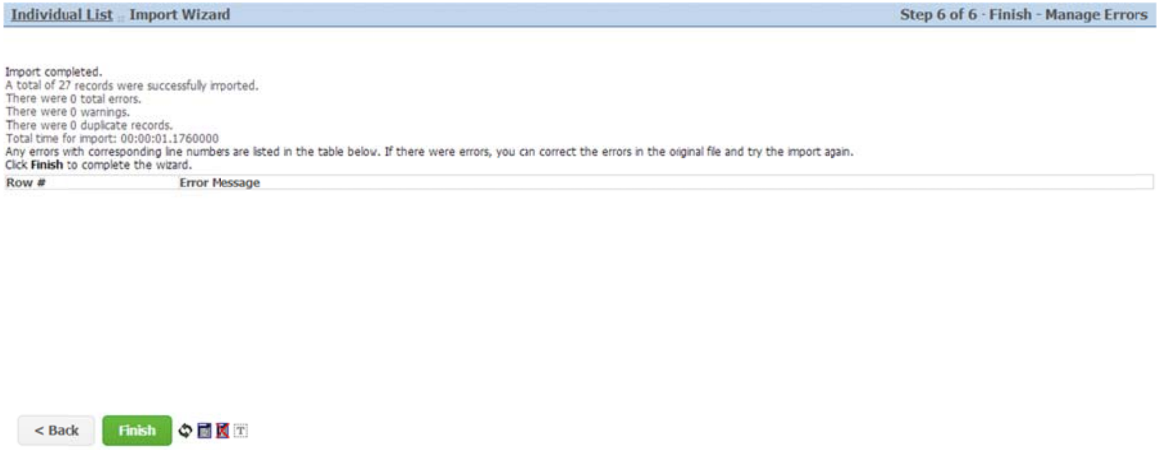
Fields not matched are in pink!

This page is just a status indication if any of the fields of the spreadsheet are not mapped to mapper fields. In this case there are no unmapped fields. But it's OK to have some fields listed here as some spreadsheet data may not be relevant to the import.



This page lets you configure how you want to detect any potential duplicate records. The list of fields presented is based on

Click on the "Import Now" button.



You will be given a summary of the results. There should be no failed import records. Click on the Finish button and you will be navigated back to the individual list with all of the users now visible. Go to the Administration tab and verify the imported users have been imported with a role.

Importing Issues

We’ll now set up import on the issues page. It won’t require as much work as for individual import. Additionally we will navigate to the import wizard as a popup window, rather than navigating in place.

In the list of mappers (Alt+M), select the issue mapper. In the list of Fields, we have to change an attribute of the status_id field specifically for supporting import. This is because the status_id field has been specified with the “Locked” attribute. We did that because we wanted to prevent a user from manually changing the issue status. But a locked field cannot be mapped when importing.

To change the behavior of fields depending on certain situations, or usages, we rely on responding to flavors. In this type of requirement we can have two versions of the field on a mapper, with different attributes that are "mutually exclusive" due to flavoring.

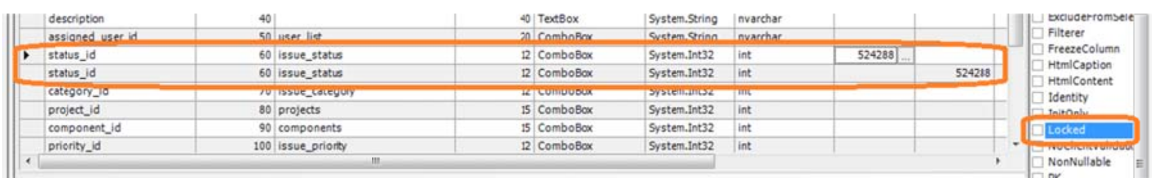
Select the status_id field and right click and choose "Duplicate". On the field you just duplicated, set the following value.

Field	**Settings**	**Notes**
status_id	Exclude Flavor: Import	Popup flavor widget by clicking on the button in the Exclude Flavor cell.

Now press F5, which will refresh the list of fields. You will now see the duplicate field. Select the status_id field which does not have the Exclude Flavor specified and set the following values.

Field	**Settings**	**Notes**
status_id	Include Flavor: Import Attributes: REMOVE Locked attribute.	Popup flavor widget by clicking on the button in the Exclude Flavor cell.

Your should field list should look like this.



Similarly, the milestone_id field has been marked as required. But the imported data does not contain any milestone information. So we need to have a version of the milestone field that is not required for importing.

Select the milestone _id field and right click and choose "Duplicate". On the field you just duplicated, set the following value.

Field	Settings	Notes
milestone _id	Exclude Flavor: Import	

Now press F5, which will refresh the list of fields. You will now see the duplicate field. Select the milestone _id field which does not have the Exclude Flavor specified and set the following values.

Field	Settings	Notes
milestone _id	Include Flavor: Import Attributes: REMOVE Required attribute.	

One further modification to the mapper is required to ensure the assigned user information is imported correctly.

The excel spreadsheet contains user id's associated with a user, rather than a resolved name. However, we are mapping those user id's to the assigned_user_id field of the mapper. If we try and import like this, the import tool will generate an error for each row imported because it could not resolve a user_id value to one of the user name.

There are a couple of ways to solve this. We could duplicate the assigned_user_id field with the import version of the field not having a picklist. Or, we can use a field property specifically for manipulating import behavior.

Select the assigned_user_id field and make the following property change.

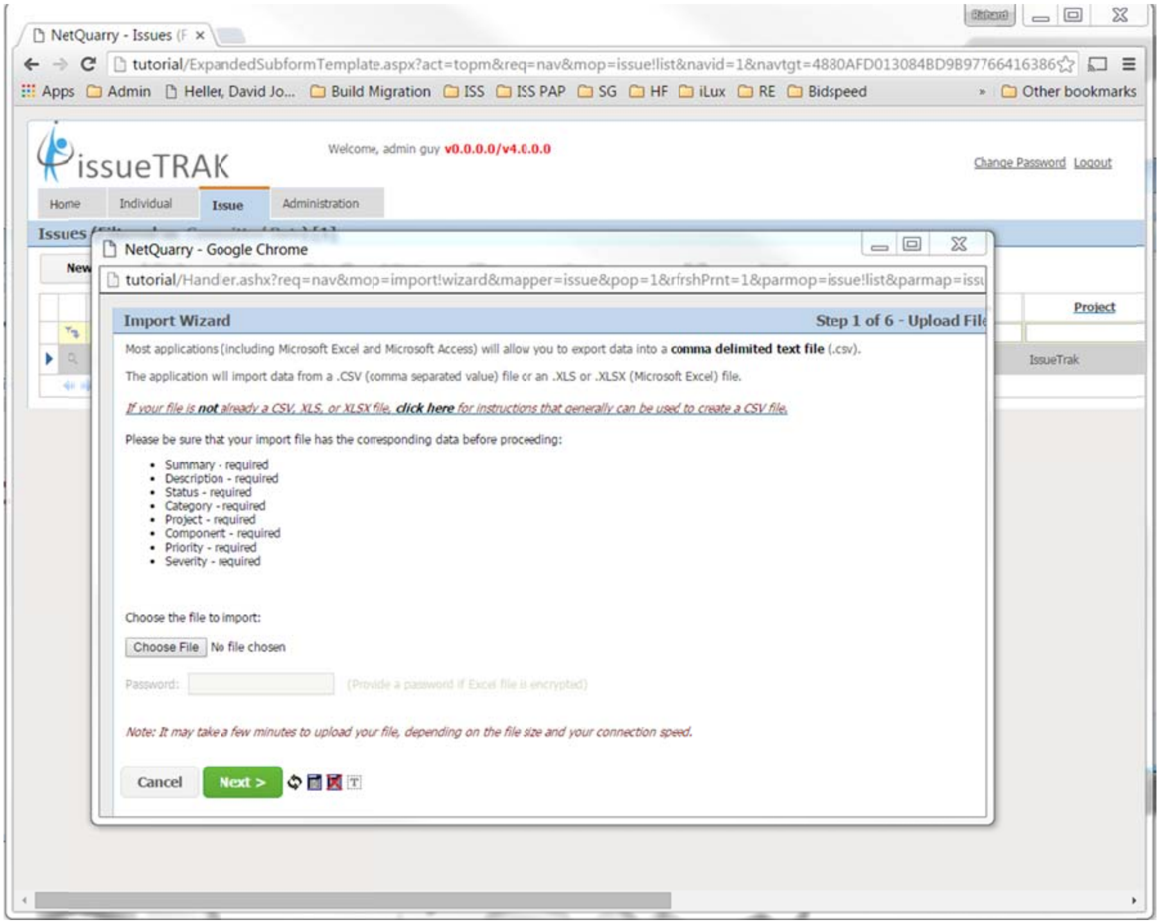
Property	Value	Notes
FieldImportAttrs	RawValueIfUnknown	We know the values being imported are the raw user_id value so we can ignore the picklist resolution requirement with this setting.

Now click on the Extensions subform.

Add the Import.XExt extension to the mapper. Then, with the Import.XExt extension selected, click on the Property sheet and set the following properties.

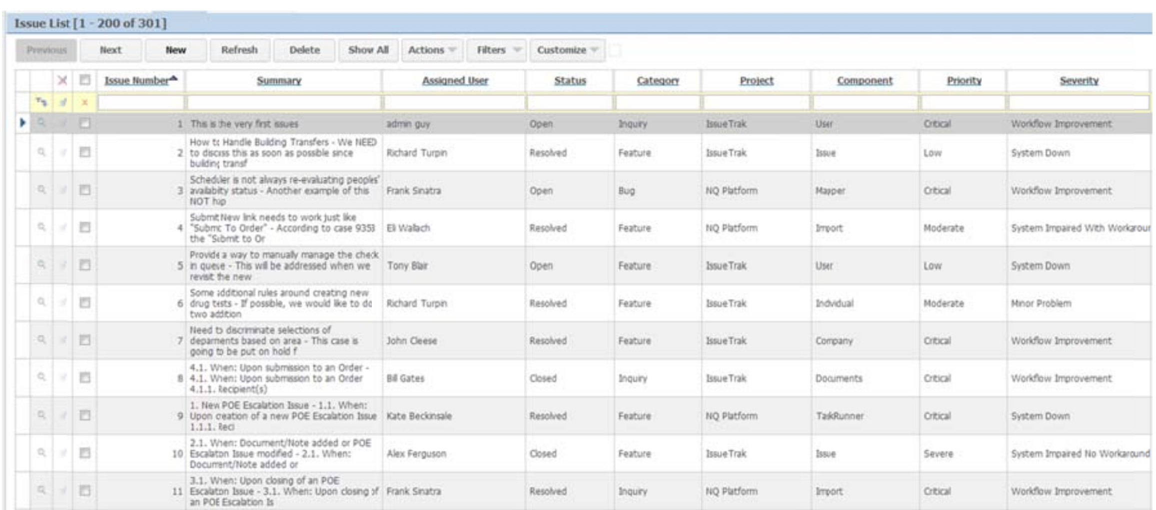
Property	Value	Notes
ImportWizardAttrs	PopupRefreshParent	The wizard will navigate to a popup, but when the wizard is complete and the page closes, the parent page will be refreshed.

Now log into the application and navigate to the Issue list. Click on the "Actions" Menu. You will see the Import option has been added.



Note this time the import wizard is hosted in a popup window. The installation came with a spreadsheet of issues to import that are assigned to one or more of the users you have just imported. You can find the spreadsheet by navigating to %NQROOT%\Documents\Tutorials\IssueTrak\Imports\Issues.xlsx or C:\NetQuarry\Customers\IssueTrak\Imports\Issues.xlsx

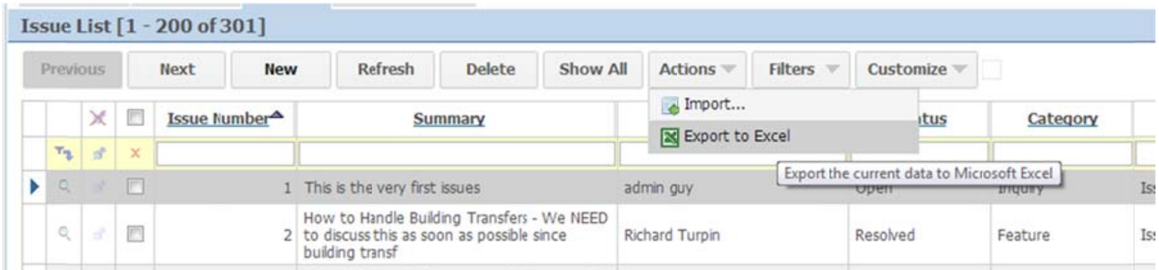
Proceed through the wizard to completion. Once finished, the Popup import wizard should close and the list of issues should be automatically refreshed.



Exporting Data

To export data from the application, all you have to do is add the ExcelExport.Export extension to any mapper (and therefore page) that you wish to support export.

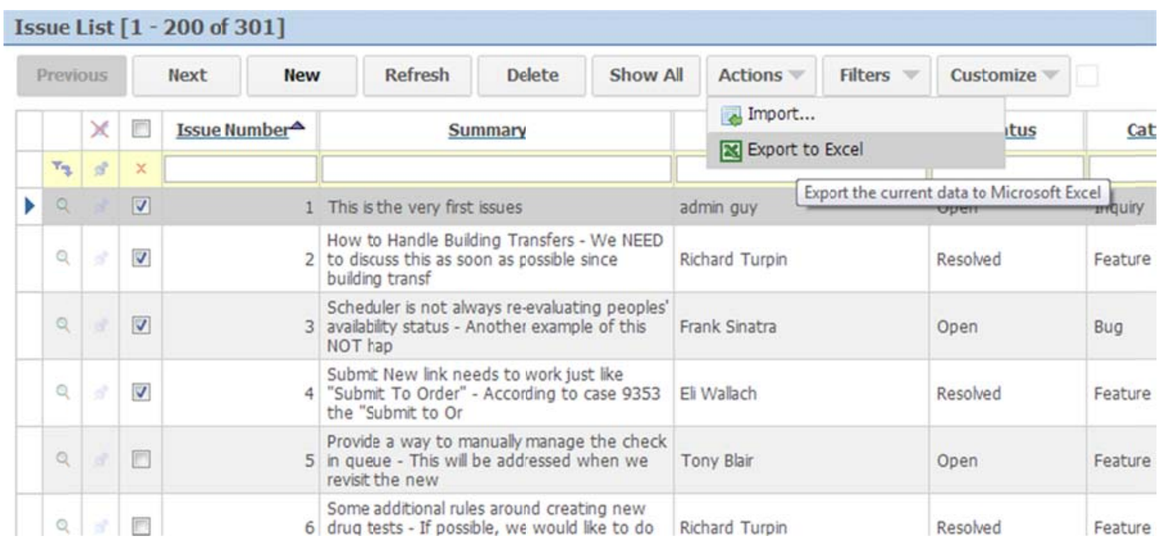
Adding this extension, adds the "Export" option to the "Actions" menu.



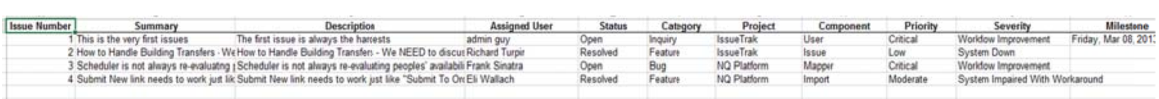
Add the ExcelExport.Export extension to the issue mapper, and then log into the application. Navigate to the issues list and export the data.

The data exported will match on any filtering criteria that has been applied to the mapper/page. E.g. security filtering, user filters.

You can also manually select a subset of the list items to export, by checking one or more of the selection check boxes on the right of the list. If you select 4 issues from the list, and "Export" the spreadsheet will only contain those items that were selected.



Generates this output…



Issue Number	Summary	Description	Assigned User	Status	Category	Project	Component	Priority	Severity	Milestone
1	This is the very first issues	The first issue is always the harrests	admin guy	Open	Inquiry	IssueTrak	User	Critical	Workflow Improvement	Friday, Mar 08, 2013
2	How to Handle Building Transfers - We	How to Handle Building Transfers - We NEED to discus	Richard Turpin	Resolved	Feature	IssueTrak	Issue	Low	System Down	
3	Scheduler is not always re-evaluating	Scheduler is not always re-evaluating peoples' availabili	Frank Sinatra	Open	Bug	NQ Platform	Mapper	Critical	Workflow Improvement	
4	Submit New link needs to work just lik	Submit New link needs to work just like "Submit To Or	Eli Wallach	Resolved	Feature	NQ Platform	Import	Moderate	System Impaired With Workaround	

Save Your Metadata

Now would be a good time to save your metadata changes.

Sending Bulk Email

The platform comes with functionality that allows you to bulk email one or more recipients. There are a couple of requirements you need to meet before being able to send email in bulk. You need to configure the provided SMTP relay service to relay emails to the recipients. You need a mapper that contains a field of Cell Type "EmailAddr". You need one or more templates to select from, to provide the email content.

SMTP Service

You must have configured an SMTP service on your application. This tutorial comes with an SMTP service, but it is not configured to relay emails. That is something you have to provide.

The recommended approach to setting up the SMTP service for testing email delivery in development is to configure the service to relay email to either a local or remote spam trap application/service.

A useful local smtp listener service installed on your computer is the SMTP4DEV tool (http://smtp4dev.codeplex.com/)

Alternatively you can set up an account at Mailtrap (http://mailtrap.io/). This service provides a way to have your email delivered to a safe non-relaying location. You can inspect the content and headers of the email to confirm/debug requirements. It even allows emails to be relayed to recipients.

Setting up SMTP Service for local SMTP4DEV

In the Studio, click on the “Services” link under “Configuration”

Make sure the service using the NetQuarry.Services.MailService component is selected.

Click on the property sheet and make the following changes.

Property	**Value**	**Notes**
CheckNoSendList	True	The service can check against a built in no send list.
DefaultSenderAddress	system@issuetrak.com	
Port	25	Default SMTP port
Server	localhost	Your local development machine
AuthenticationType	Anonymous	
NoSendListCheckDuration	30	Polls the no send list every 30 minutes.

Setting up SMTP Service for Mailtrap

In the Studio, click on the “Services” link under “Configuration”

Make sure the service using the NetQuarry.Services.MailService component is selected.

Click on the property sheet and make the following changes.

Property	Value	Notes
CheckNoSendList	True	The service can check against a built in no send list.
DefaultSenderAddress	system@issuetrak.com	
Port	2525	Port provided by mailtrap.io
Server	mailtrap.io	
AuthenticationType	Basic	
User	<>	Use your mailtrap.io provided user id
Password	<>	Use your mailtrap.io provided password
NoSendListCheckDuration	30	Polls the no send list every 30 minutes.
UseSSL	False	

An Email Address Field

The mapper you wish to send bulk email from must have at least one field that has the Cell Type of "EmailAddr"

Template Mailer Templates

We've already touched on templates for rendering data in console pages. Now, we're going add two templates for use in the template mailer.

Two template files for email have been pre-installed into the tutorial for use by the Template Mailer. Ultimately the files should be located in the following folder %NQROOT%\Templates\IssueTrak\email.
However, they may need to be copied from one of two locations…

C:\NetQuarry\Customers\IssueTrak\Templates, or
%NQROOT%\Documents\Tutorials\IssueTrak\Templates

First create a new module called IssueTrak-email.

In the Studio, click on the Templates link (not Templates under Components). In the new row create a template with the following settings

Item	Value	Notes
Module	IssueTrak-email	
Name	email-welcome-to-issuetrak	
Type	File	
Attributes	TemplateMailer	This is so the template appears in the list of templates to select from.
Category	Static Templates	

With this template selected, click on the Property Sheet and set the following properties

Property	Value	Notes
FileName	IssueTrak\email\welcome-to-issuetrak.html	
Mapper	individual	Selecting a mapper means that the template is only available for selection when the template mailer is launched from apage that uses the same mapper. If no mapper property is specified, the template will be visible on all instances of the Template Mailer.
Subject	Welcome To IssueTrak!	This value is automatically copied into the subject field of the template mailer, when a template is selected.

Create an additional template with the following settings

In the Studio, click on the Templates link. In the new row create a template with the following settings

Item	Value	Notes
Module	IssueTrak-email	
Name	email-welcome-to-issuetrak-body	
Type	File	
Attributes	TemplateMailer	
Category	Dynamic Templates	

With this template selected, click on the Property Sheet and set the following properties

Property	Value	Notes
FileName	IssueTrak\email\welcome-to-issuetrak-body.html	
Mapper	individual	
Subject	Welcome To IssueTrak!	

Add the Template Mailer Extension

Once you have satisfied the minimum requirements, you simply add the BulkEmailer.TemplateMailer extension to your mapper. In this case the individual mapper.

Once done, click on the property sheet and set these properties

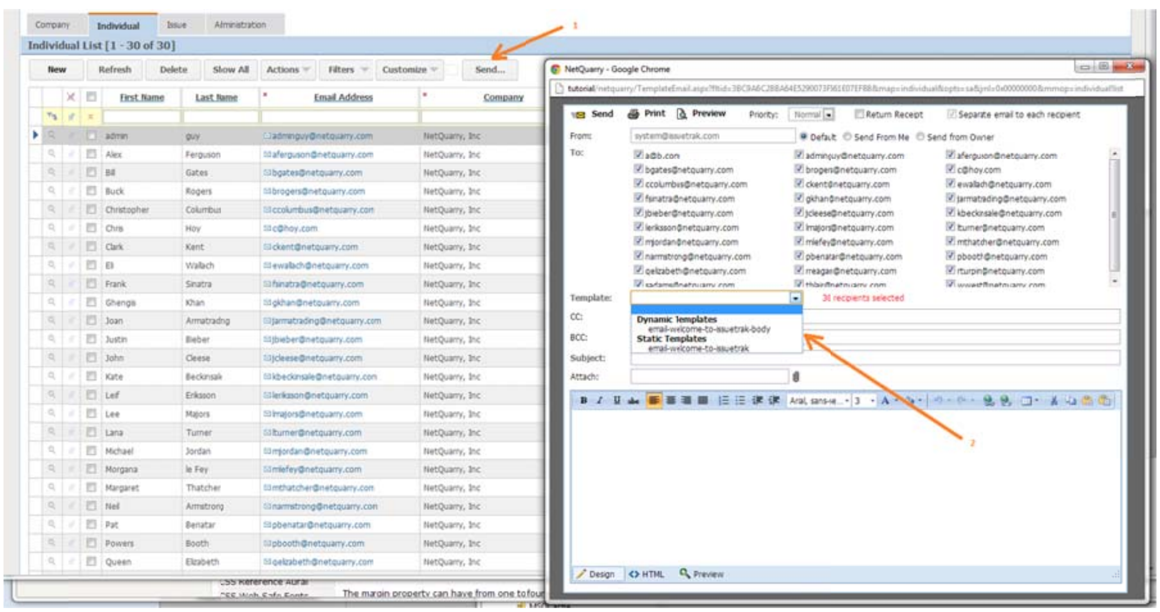
Property	Value	Notes
AllowHtmlEditing	True	Allows the user to send fully custom email content without being forced to pick from a template

Using the Template Mailer

Log in to the application and navigate to the Individual list.

1) The Template Mailer extension has added a new "Send"button to the toolbar

2) Click on the Send button, you can choose from either of the two categorized templates.

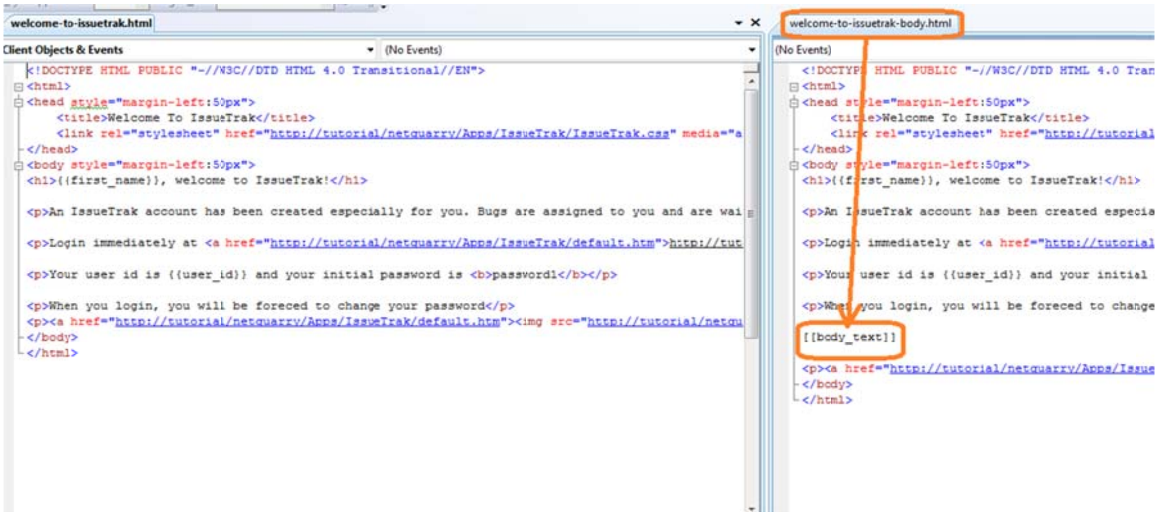


If you do not choose a template, you are free to provide your own subject and body content. As well as attaching a document.

If you select the "email-welcome-to-issuetrak" template. You will see that the subject is copied over from the template data and the body editor of the template mailer is locked.

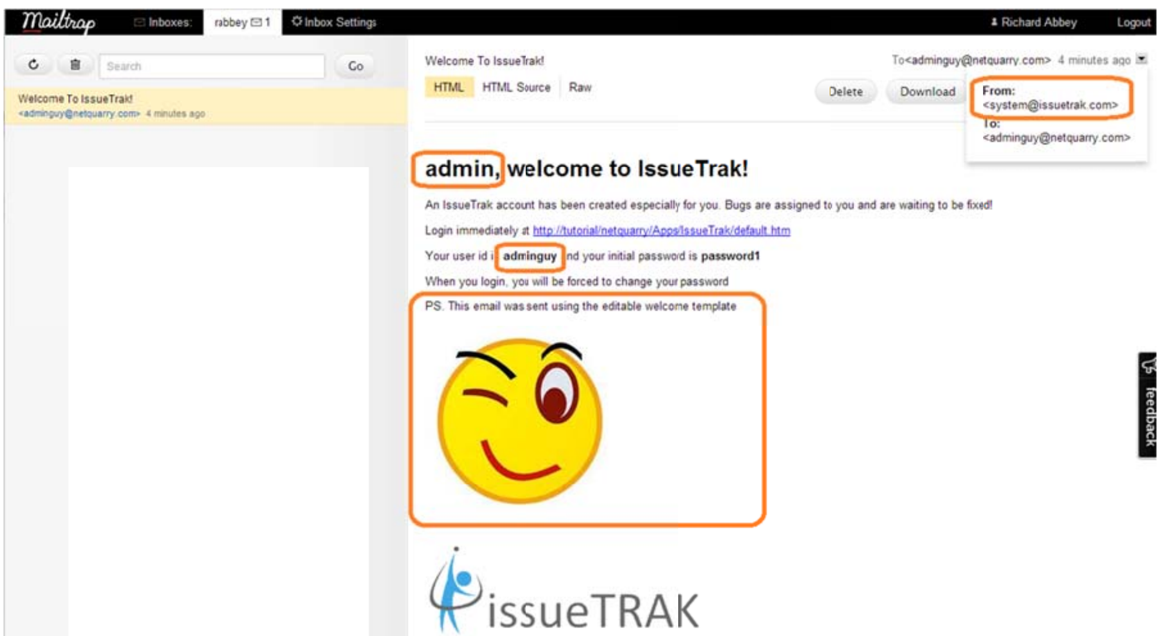
If you select the "email-welcome-to-issuetrak-body" template, you will see that the subject is copied over from the template data and the body editor of the template mailer is editable.

When you choose a template. If you want the body content to be editable, you have to include a special replacement token into the body of your template. If you inspect the "templates\email\email-welcome-to-issuetrak-body.html" template you can see the additional token.



One thing to note is that the replacement token is enclosed in [[ ]] brackets, rather than the expected {{ }} brackets that you've already been introduced to. Content replaced within [[ ]] brackets is not HTML escaped by the platform during the rendering.

Here's an example of an email sent to MailTrap.io (apologies for the custom content).



Other UI Tweaks

Linking From Fields to Pages

We've shown how to create navigation targets to open lists of pages. We also have a way to directly link to specific records, via field properties. To set up links from a field to a record, you specify values for just two properties. A "LinkMOP" which tells the platform which page to navigate to and a "LinkKey" which tells the platform which specific record to display on the target page. You may remember setting these properties on console page elements earlier in the tutorial.

Add links to the Individual List

In the studio, go the list of mappers and select the individual mapper. In the list of fields, set the following properties on the given fields.

Field	**Settings**	**Notes**
company_id	LinkKey: company_id LinkMOP: company!main	
user_id	LinkKey: user_id LinkMOP: users!main	

On the issue mapper, we have to make a slight tweak to the mapper.

Select the issue mapper and change the "View Name" to issue_view. Then move off the row and back on to save the change. Then right click on the issue mapper and click "Create Fields…". Select and add the "assigned_user_guid" field. Set the Column Width of the field to 0. Then make the following property changes on listed fields

Field	**Settings**	**Notes**
assigned_user_id	LinkKey: assigned_user_guid LinkMOP: individual!main	We have to provide the right value of the navigation. It has to be the value that is the recognized primary key value of the data you want to display.
issue_number	LinkKey: issue_id LinkMOP: issue!main	

Log into the application and navigate to the individual and issue lists. You will see the fields have now link decorations and navigate to the specific detail pages.

Add Filtered Links to a List

Use "Alpha" Filtering

The platform provides a way to automatically filter the contents of a list by an "alpha" filter.

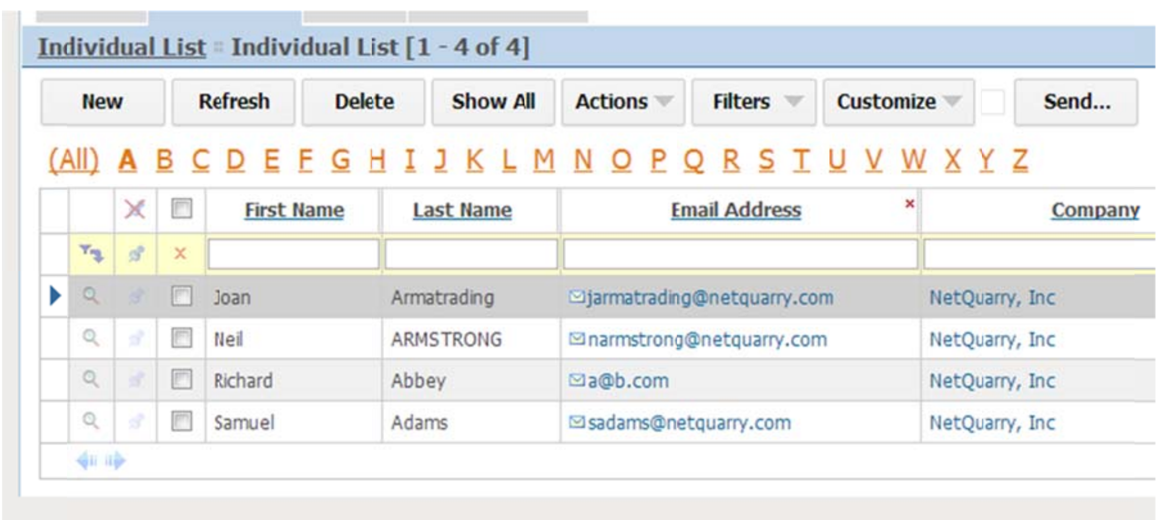
Go to the list of Pages (Alt+P). Select the page individual!list and then click on the Page Element subform.

There is only one page element listed, for the Main slot, with a control type of "dsctrl".

Select that page element and open the property sheet and set the following options.

Property	Value	Notes
AlphaFilterAttributes	IncludeShowAllLink, UseWCharFilter	Adds the (All) link to the alpha filter element so you can quickly go back to the unfiltered list. UseWCharFilter will force the filtering specification to add the wide char prefix N to the filter clause.
AlphaFilterField	last_name	Specified which field on the list (in the underlying mapper) that will be filtered by the alpha filter, e.g. last_name like N'a%'

Log out and login to the application and navigate to the Individual list page. Click on a letter to filter by Last Name.



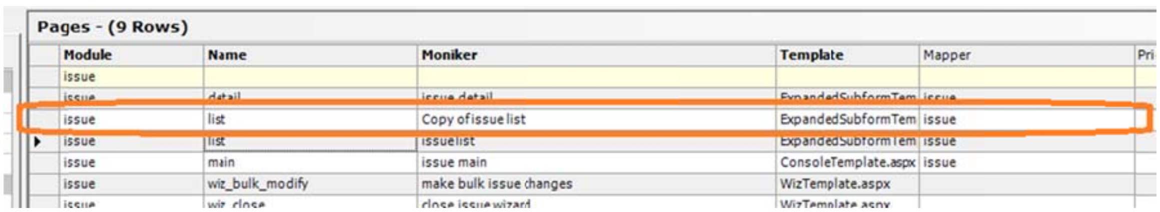
Provide a Set of Specific Filtering Links

We can also provide a set of links that take you to the same lists but filtered by some static criteria. For example, All Issues, Issues assigned to me, Open Issues assigned to me, Resolved Issues assigned to me.

Adding this capability can be achieved by assigning a Navbar to "dsctrl" Page Elements.

For each of links we'll need to create a specific page that is a list page, with a filter applied on the page.

Go to the list of Pages (Alt+P). Select the page issue!list and then right click on the page and select "Duplicate". Press F5 to refresh the list.



Pages - (9 Rows)

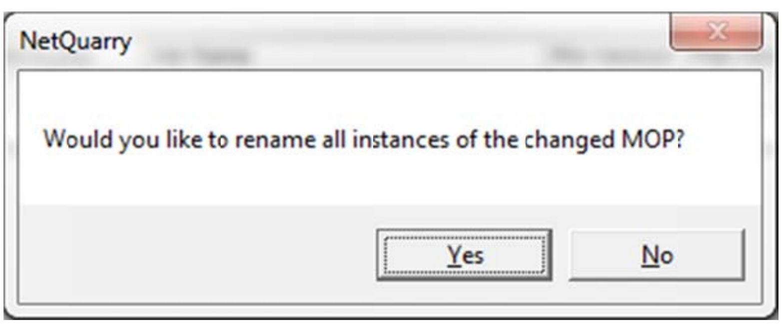
Module	Name	Moniker	Template	Mapper	Pri
issue					
issue	detail	issue detail	ExpandedSubformTem	issue	
issue	list	Copy of issue list	ExpandedSubformTem	issue	
issue	list	issuelist	ExpandedSubformTem	issue	
issue	main	issue main	ConsoleTemplate.aspx	issue	
issue	wiz_bulk_modify	make bulk issue changes	WizTemplate.aspx		
issue	wiz_close	close issue wizard	WizTemplate.aspx		

Select the duplicate page (the Moniker says "Copy of issue list")

Make the flowing changes to the duplicate page

Item	Value	Notes
Name	list-my-issues	
Moniker	issues assigned to logged in user	

Move off the row to save the changes. You will be prompted with the following message



The simple answer to this prompt is "NO". On the duplicate page row, the page name is still issue!list. We do clearly want to change the name of this page, but not to have the name issue!list be changed to issue!-my-issue-list everywhere it is referenced. The issue!list page is still needed as it is. It is only appropriate to say yes to this prompt when you want to change a page name due to a legitimate name change. Not a duplication and renaming when you want to quickly create a new page similar to an existing page.

With the new page selected, make the following property changes.

Property	Value	Notes
PageFilter	assigned_user_id=!fnUserID$() AND status_id <> 3	Here we use an embedded function to filter on the currently logged in user which is obtained from !fnUserID. The $ character is a modifying token for the returned value. It tells the platform to return the value as an Ansi Quoted string. The status_id filter is to return issues that are not closed.
Caption	Issues Assigned to Me	

Now repeat this process, duplicating the issue list page two more times and on each duplicated page, make the following changes. (Remember not to rename instances of the MOP).

First duplicate page

Item	Value	Notes
Name	list-my-open-issues	
Moniker	open issues assigned to logged in user	

Move off the row to save the changes

With the new page selected, make the following property changes.

Property	Value	Notes
PageFilter	assigned_user_id=!fnUserID$() AND status_id = 1	
Caption	Open Issues Assigned to Me	

Second duplicate page

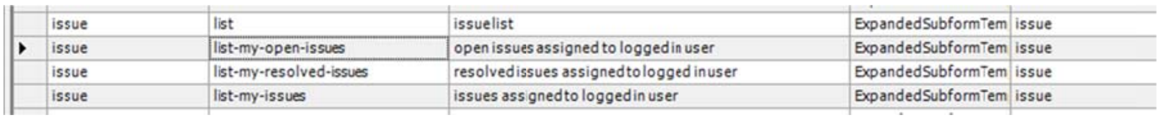
Item	Value	Notes
Name	list-my-resolved-issues	
Moniker	resolved issues assigned to logged in user	

Move off the row to save the changes

With the new page selected, make the following property changes.

Property	Value	Notes
PageFilter	assigned_user_id=!fnUserID$() AND status_id = 2	
Caption	Resolved Issues Assigned to Me	

Your list of pages should look something like this



issue	list	issuelist	ExpandedSubformTem	issue
issue	list-my-open-issues	open issues assigned to logged in user	ExpandedSubformTem	issue
issue	list-my-resolved-issues	resolved issues assigned to logged in user	ExpandedSubformTem	issue
issue	list-my-issues	issues assigned to logged in user	ExpandedSubformTem	issue

We now have to create a Navbar that links to these pages.

Go to the list of NavBar's under Navigators and create the following Navbar

Item	**Value**	**Notes**
Module	issue	
Name	issue-ds	

There are no properties to set.

Move off the row to save the changes and select the issue-ds Navbar

In the Targets subform, create the following four Nav Targets.

1) For viewing all issues…

Item	**Value**	**Notes**
Module	issue	
Name	all	
Target Type	Page	
Target	issue!list	
Sort Order	10	

With that new Target selected, click on the Property sheet and set the following properties.

Property	**Value**	**Notes**
Caption	All Issues	

2) For viewing "my" issues…

Item	Value	Notes
Module	issue	
Name	my issues	
Target Type	Page	
Target	issue!list-my-issues	
Sort Order	20	

With that new Target selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Caption	My Issues	

3) For viewing "my open" issues…

Item	Value	Notes
Module	issue	
Name	my open issues	
Target Type	Page	
Target	issue!list-my-open-issues	
Sort Order	30	

With that new Target selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Caption	My Open Issues	

4) For viewing "my resolved" issues…

Item	Value	Notes
Module	issue	
Name	my resolved issues	
Target Type	Page	
Target	issue!list-my-resolved-issues	
Sort Order	30	

With that new Target selected, click on the Property sheet and set the following properties.

Property	Value	Notes
UseNavCaptionOnTarget	True	Forces the caption of the nav target element to be used as the caption of the target page, overriding the caption property of the target page.
Caption	My Resolved Issues	

Now go back to the list of pages (Alt+P). We have to associate this navbar with each of the pages the navbar target link to.

Select the "issue!list" page and click on the Page Element tab. Select the single "Main" slot record which is a dsctrl and set the following property.

Property	Value	Notes
Navigator	issue-ds	

Select the "issue!list-my-issues" page and click on the Page Element tab. Select the single "Main" slot record which is a dsctrl and set the following property.

Property	Value	Notes
Navigator	issue-ds	

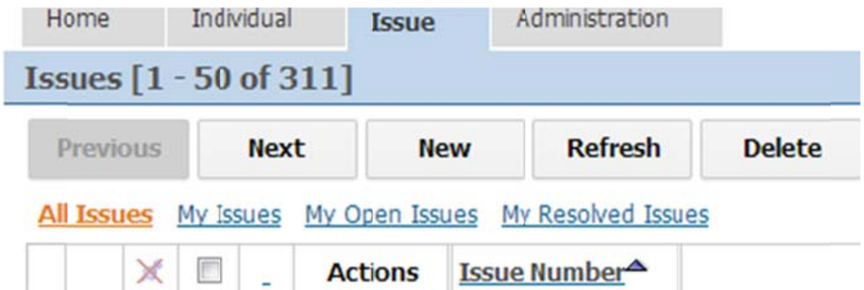
Select the "issue!list-my-open-issues" page and click on the Page Element tab. Select the single "Main" slot record which is a dsctrl and set the following property.

Property	Value	Notes
Navigator	issue-ds	

Select the "issue!list-my-resolved-issues" page and click on the Page Element tab. Select the single "Main" slot record which is a dsctrl and set the following property.

Property	Value	Notes
Navigator	issue-ds	

Log out of the application and log in. Navigate to the issue list to see the effects of this change. You may need to login as a user that is not the "adminguy" to better see the behavior of these links



If you click on a link, see how the Caption of the page matches with the caption property of the page. However, if you click on the "My Resolved Issues" link, the page caption is "My Resolved Issues" instead of "Resolved Issues Assigned to Me". All due to the "UseNavCaptionOnTarget" property of the nav target.

Creating a Report with Individual Drill Down Fields

We will now create a dashboard page that acts as the landing page for all users. On the dashboard will be a set of metrics reports that summarize issue information. The dashboard will be based on a console page template. The cells of the report fields will navigate to filtered lists matching the criteria of the values being represented.

Create a Dashboard Page

Create a new module, called "IssueTrak-home" and create a new Page based on the ConsoleTemplate with the following settings

Item	**Value**	**Notes**
Module	IssueTrak-home	
Name	dashboard	
Moniker	home dashboard	
Template	ConsoleTemplate.aspx	

Set the following properties

Property	**Value**	**Notes**
Caption	Dashboard	

Add a new NavTarget on the "main" bar (under System Navbars), making sure it will be the first target.

Item	**Value**	**Notes**
Module	IssueTrak-home	
Name	dashboard	
TargetType	Page	
Target	IssueTrak-home!dashboard	
Sort Order	0	Sort order affects position on navbar.

Set the following properties

Property	**Value**	**Notes**
Caption	Home	

Create the Summary Contents and Pages

There are some summary views already created in the issues_data database. We need to slurp those in via the page creation wizard to quickly create the necessary content.

In the Studio, on the Home page, click on the New Page link. Run through the page wizard setting the following values

Page	**Item**	**Value**	**Notes**
Module	New Module	IssueTrak-reports	
Mapper	New Mapper	issue_assign_priority_pvt	
	DataSource	IssueTrak	
	View	issue_assign_priority_pvt_view	
Mapper Options		Skip Mapper Options Screen	
Navigation		Skip Navigation Screen	
Page Options	Template	TabbedSubformTemplate.aspx	
	Page Styles	UNCHECK Detail UNCHECK Console UNCHECK Has SubItems	There is no detail or main to drill down to. The drill down targets are already created.
	Text/ List Caption	Issue Priority By Assignee List	
	Page Names/ List Name	issue_assign_priority_list	

Repeat the page wizard again with the following values

Page	**Item**	**Value**	**Notes**
Module	Existing Module	IssueTrak-reports	
Mapper	New Mapper	issue_assign_status_pvt	
	DataSource	IssueTrak	
	View	issue_assign_status_pvt_view	
Mapper Options		Skip Mapper Options Screen	
Navigation		Skip Navigation Screen	
Page Options	Template	TabbedSubformTemplate.aspx	
	Page Styles	UNCHECK Detail UNCHECK Console UNCHECK Has SubItems	There is no detail or main to drill down to. The drill down targets are already created.
	Text/ List Caption	Issue Status By Assignee List	
	Page Names/ List Name	issue_assign_status_list	

Add Contents to the Dasboard

In the Studio, go to the Pages list and select the IssueTrak-home!dashboard page. In the Page Elements subform add two page elements as follows

Item	Value	Notes
Slot Name	Main	
Name	issue priority by assignee	
Component	MiniList	
Order	10	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Mapper	issue_assign_priority_pvt	
RowsPerPage	30	
Sort	assigned_to	
LinkMOP	issuetrak-reports!issue_assign_priority_list	
Caption	Issue Priority By Assignee	
Link Caption	Issue Priority By Assignee (Click to View All)	

And…

Item	Value	Notes
Slot Name	ConsolePage	
Name	issue status by assignee	
Component	MiniList	
Order	20	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Mapper	issue_assign_status_pvt	
RowsPerPage	30	
Sort	assigned_to	
LinkMOP	issuetrak-reports!issue_assign_status_list	
Caption	Issue Status By Assignee	
Link Caption	Issue Status By Assignee (Click to View All)	

Add Filtered Drill Down Links to Priority Report

Now go to the list of Mappers. We will need to make some UI changes to the fields to add the necessary drill down features.

Select the issue_assign_priority_pvt mapper and make the following changes to the Fields

Field	Settings	Notes
assigned_to	CellType: Button CellTypeAttributes: ValueAsButtonText Style: text-align:left; LinkBehaviorAttributes: UseLinkFilter LinkFilterExpr: assigned_user_id=['assigned_user_id] AND status_id<>3 LinkMOP: issue!list	By default button cell types show the field caption on the button, but you can override this with the ValueAsButtonText attribute. Which forces the button text to show the underlying value. The style override forces the default center justification of button text to be left justified. The link behavior setting tells the platform to perform a link and pass filter criteria, rather than a specific single key value. The LinkFilterExpr specified how the target list page should be filtered. We use any valid SQL WHERE clause syntax (without the WHERE). Here we are filtering the list of issue by the person assigned to those issues.
assigned_user_id	Column Width: 0 Attributes: PK, UniqueKey	This is required for the drill down navigation to work. the drill downs will show a list of issues in a particular status, or across status filtered by the assigned_user_id

Field	Settings	Notes
Critical	CellType: Button CellTypeAttributes: ValueAsButtonText AggregateType: Sum CssClass: num LinkBehaviorAttributes: UseLinkFilter, UseColumnFilter LinkFilterExpr: priority_id=1 AND status_id<>3 LinkMOP: issue!list	We will be providing an aggregate row at the bottom of the report to allow drill down to the entire set of issues matching a certain priority. There is a special CSS class to apply to numbers to force the text alignment to the right. the standard alignment for numeric values. The additional attribute here is UseColumnFilter. the platform takes advantage of some known information about the mapper. Specifically know what the primary key value is (assigned_user_id) and automatically applies that filtering to the drill down link. However to support drill down on the Aggregate row, we don't have a primary key. So in that special case the primary key filtering is not applied. What is consistent between the two types of navigation is the filtering by a specific priority_id that matches the selected column.
Severe	CellType: Button CellTypeAttributes: ValueAsButtonText AggregateType: Sum CssClass: num LinkBehaviorAttributes: UseLinkFilter, UseColumnFilter LinkFilterExpr: priority_id=2 AND status_id<>3 LinkMOP: issue!list	

Field	Settings	Notes
Moderate	CellType: Button CellTypeAttributes: ValueAsButtonText AggregateType: Sum CssClass: num LinkBehaviorAttributes: UseLinkFilter, UseColumnFilter LinkFilterExpr: priority_id=3 AND status_id<>3 LinkMOP: issue!list	
Low	CellType: Button CellTypeAttributes: ValueAsButtonText AggregateType: Sum CssClass: num LinkBehaviorAttributes: UseLinkFilter, UseColumnFilter LinkFilterExpr: priority_id=4 AND status_id<>3 LinkMOP: issue!list	

Add Filtered Drill Down Links to Status Report

Now select the issue_assign_status_pvt mapper and make the following changes to the Fields

Field	Settings	Notes
assigned_to	CellType: Button CellTypeAttributes: ValueAsButtonText Style: text-align:left; LinkBehaviorAttributes: UseLinkFilter LinkFilterExpr: assigned_user_id=['assigned_user_id] AND status_id<>3 LinkMOP: issue!list	
assigned_user_id	Column Width: 0 Attributes: PK, UniqueKey	
IssueOpen	CellType: Button CellTypeAttributes: ValueAsButtonText AggregateType: Sum CssClass: num LinkBehaviorAttributes: UseLinkFilter, UseColumnFilter LinkFilterExpr: status_id=1 LinkMOP: issue!list Caption: Open	
Resolved	CellType: Button CellTypeAttributes: ValueAsButtonText AggregateType: Sum CssClass: num LinkBehaviorAttributes: UseLinkFilter, UseColumnFilter LinkFilterExpr: status_id=2 LinkMOP: issue!list	

The last step is to make a couple of changes to the mappers

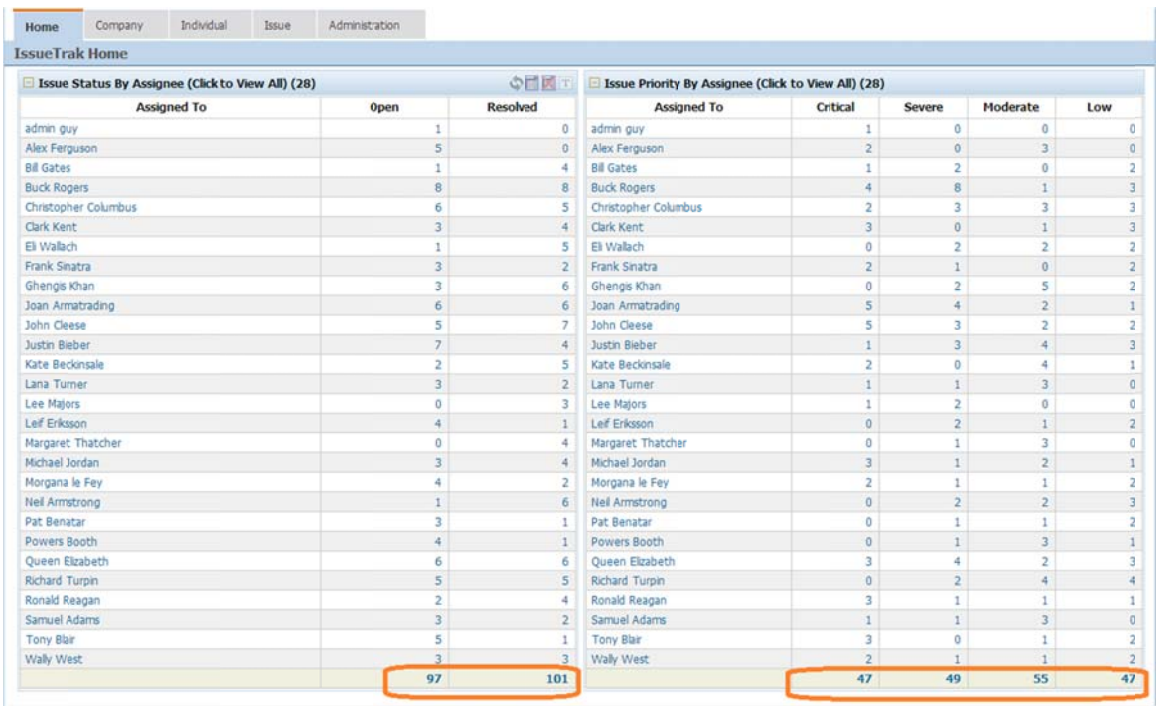
On both mappers, issue_assign_priority_pvt and issue_assign_status_pvt make the following changes.

Item	**Value**	**Notes**
Attributes	DenyDelete, DenyInsert, DenyUpdate	The data for these mappers is actually generated through a pivot so is not updatable, but setting these attributes removes any UI that the user will be tempted to use to create new, or delete items!

Then make the same property sheet changes to both mappers

Property	**Value**	**Notes**
AggregateDisplay	Display	Force the mapper to generate an aggregate row.

Log in to the application to see the effects of these changes. You should see that the landing page is now the "Home" page and there are two reports of issues, with individual drill down links including on the aggregate row.



Assigned To	Open	Resolved
admin guy	1	0
Alex Ferguson	5	0
Bill Gates	1	4
Buck Rogers	8	8
Christopher Columbus	6	5
Clark Kent	3	4
Eli Wallach	1	5
Frank Sinatra	3	2
Ghengis Khan	3	6
Joan Armatrading	6	6
John Cleese	5	7
Justin Bieber	7	4
Kate Beckinsale	2	5
Lana Turner	3	2
Lee Majors	0	3
Leif Eriksson	4	1
Margaret Thatcher	0	4
Michael Jordan	3	4
Morgana le Fey	4	2
Neil Armstrong	1	6
Pat Benatar	3	1
Powers Booth	4	1
Queen Elizabeth	6	6
Richard Turpin	5	5
Ronald Reagan	2	4
Samuel Adams	3	2
Tony Blair	5	1
Wally West	3	3
	97	101

Assigned To	Critical	Severe	Moderate	Low
admin guy	1	0	0	0
Alex Ferguson	2	0	3	0
Bill Gates	1	2	0	2
Buck Rogers	4	8	1	3
Christopher Columbus	2	3	3	3
Clark Kent	3	0	1	3
Eli Wallach	0	2	2	2
Frank Sinatra	2	1	0	2
Ghengis Khan	0	2	5	2
Joan Armatrading	5	4	2	1
John Cleese	5	3	2	2
Justin Bieber	1	3	4	3
Kate Beckinsale	2	0	4	1
Lana Turner	1	1	3	0
Lee Majors	1	2	0	0
Leif Eriksson	0	2	1	2
Margaret Thatcher	0	1	3	0
Michael Jordan	3	1	2	1
Morgana le Fey	2	1	1	2
Neil Armstrong	0	2	2	3
Pat Benatar	0	1	1	2
Powers Booth	0	1	3	1
Queen Elizabeth	3	4	2	3
Richard Turpin	0	2	4	4
Ronald Reagan	3	1	1	1
Samuel Adams	1	1	3	0
Tony Blair	3	0	1	2
Wally West	2	1	1	2
	47	49	55	47

Styling Content Based on Data

Row Styling

What we will do here is add the following row styling to a row based on the issue's priority.

Open Critical Priority Issues will be Bold Red

Open Severe Priority Issues will be Red

Open Moderate Priority Issues will be Blue

Open Low Priority Issues will be Grey

All Resolved Issues regardless of priority will be Green

All Closed Issues regardless of priority will have a Strike Through

First step is to add a calculated field to the underlying view of the issue mapper which will represent a style name based on the above logic.

In the SQL Enterprise Manager, open the file C:\NetQuarry\Customers\IssueTrak\Database\issues_data-views.sql

Locate the script for creating the view, "issue_view" and add the highlighted sql to the existing script as shown

<table>
<tr><th>Script</th><th>Notes</th></tr>
<tr><td>
<pre>
IF OBJECT_ID('[dbo].[issue_view]') IS NOT NULL DROP VIEW
[dbo].[issue_view]
GO

CREATE VIEW [dbo].[issue_view]
AS
SELECT i.issue_id, i.issue_number, i.summary,
i.description, i.assigned_user_id, i.status_id,
i.category_id, i.component_id, i.priority_id,
i.severity_id, i.project_id, milestone_id,
i.committed_date, i.created_by_id, i.updated_by_id,
i.date_created, i.date_updated, i.is_deleted,
i.revision, iuv.user_guid,
   CASE status_id
    WHEN 3 THEN 'iss_closed'
    WHEN 2 THEN 'iss_resolved'
    ELSE
     CASE priority_id
      WHEN 1 THEN 'iss_crit'
      WHEN 2 THEN 'iss_sev'
      WHEN 3 THEN 'iss_mod'
      ELSE 'iss_low'
     END
   END as row_css
FROM dbo.issue i WITH(NOLOCK) LEFT OUTER JOIN
individual_users_view iuv WITH(NOLOCK)
ON i.assigned_user_id=iuv.user_id
GO
</pre>
</td><td>The change you make is highlighted which converts the status and priority to a string representing a css style..</td></tr>
</table>

Execute the issues_data-views.sql script to change the view.

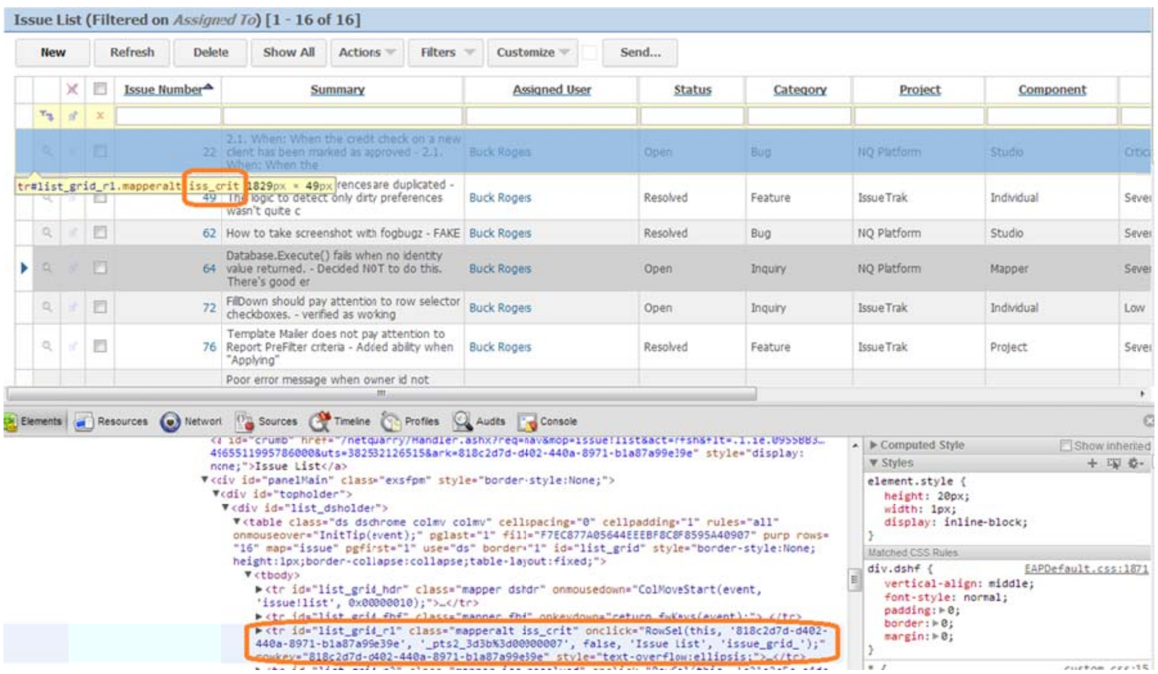
Back in the NetQuarry Studio, go to the "issue" mapper. Select it and right click and choose "Create Fields…". Add the field row_css to the mapper.

And on the row_css field, set the Column Width to zero.

Now add this property to the MAPPER

Property	**Value**	**Notes**
RowCssFields	row_css	Specifies which fields provide values for row styling

Log into the application and go to the issue list. If you inspect the row elements, you can see the additional css class added based on the context of the data.



Now we create the styling parameters by editing the css file %NQROOT%\Apps\IssueTrak\Styles\custom.css

Add the following style elements at the end of the file an save the changes

CSS	Notes

```css
tr.iss_crit td, tr.iss_crit td a, tr.iss_crit td span
{
   font-weight:bold;
   color:Red;
}

tr.iss_sev td, tr.iss_sev td a, tr.iss_sev td span
{
   color:Red;
}

tr.iss_mod td, tr.iss_mod td a, tr.iss_mod td span
{
   color:Blue;
}

tr.iss_low td, tr.iss_low td a, tr.iss_low td span
{
   color:Gray;
}

tr.iss_resolved td, tr.iss_resolved td a, tr.iss_resolved td span
{
   color:Green;
}

tr.iss_closed td, tr.iss_closed td a, tr.iss_closed td span
{
   text-decoration:line-through;
}
```

In the application, click on the Issue tab on the main bar to see the rainbow of colors

68	RowRevision field lists fields from first Summary field - Reopened just to set milestone.	Joan Armatrading	Open	Inquiry	IssueTrak	Audit	Severe
69	Provide the ability to perform a drilldown in the total row of a list - Perfect, thanks	Powers Booth	Open	Feature	IssueTrak	Company	Moderate
~~70~~	~~Textarea not supported in datasheet Fldown. - verified as working~~	~~Morgana le Fey~~	~~Closed~~	~~Bug~~	~~NQ Platform~~	~~Page~~	~~Low~~
71	**Field SelectExpr causes error in aggregates. - Verified by Mark Wilson under case 11411.**	**John Cleese**	**Open**	**Inquiry**	**NQ Platform**	**TaskRunner**	**Critical**
72	FlDown should pay attention to row selector checkboxes. - verified as working	Buck Rogers	Open	Inquiry	IssueTrak	Individual	Low
73	Provide option to suppress hidder fields in lists. - Verified by Mark.	Joan Armatrading	Resolved	Inquiry	IssueTrak	Company	Critical

IMPORTANT

The style sheet you modified is part of the platform installation. Therefore, if you reinstall a new platform version, the changes you made to the css file will be lost forever.

To avoid this loss, you should copy your modified css file to the folder C:\NetQuarry\Customers\IssueTrak\Config\web\Styles

After the install has been completed, the Install.bat file contains a step to copy the backed up custom.css style sheet to the %NQROOT%\Apps\IssueTrak\Styles folder.

Image Styling

We will now add a field to the issue list that displays an icon based on the value of issue_category_id. We need an additional issue_category_id field on the mapper. So there are two ways to approach this. Add an aliased field on the issue_view, or perform the aliasing in the metadata. Which is what we'll do here.

In the Studio, select the issue mapper and go to the Fields subform. In the new row, create a new field with the following values

Field	Settings	Notes
category_icon	Column Order: 0 Column Width: 6 CellType: Icon .NET Type: System.Int32 Data Type: int AliasName: category_id HideFlavor: StandardDetail LabelOptions: NoLabel Tooltip: [$category_id]	

To have the underlying field value be converted from an integer value to an image, we have to create a pickilst which has a display text value that is a path to the relevant image file.

Create a Standard Picklist with the following values.

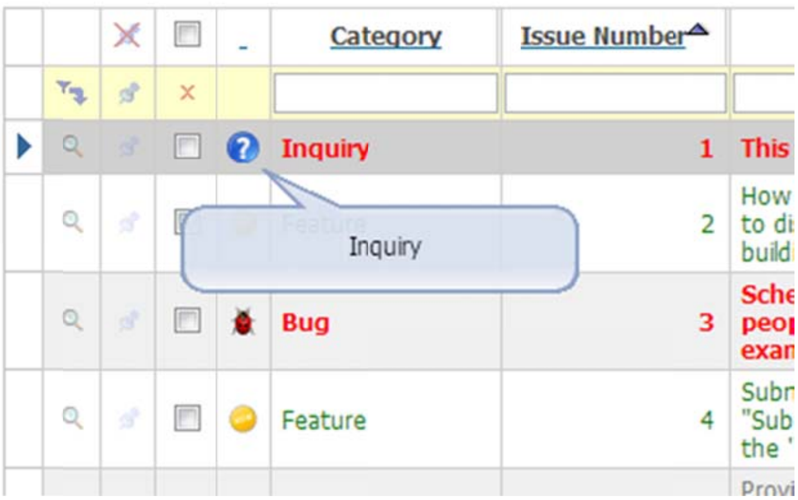
Item	Value	Notes
Module	issue	
Name	issue_category_icon	
Type	StandardPicklist	
Attributes	Cache, LimitToList, StoreAltInt, ImageInText	

And then in the Items subform add these items

Name	Caption	Notes
bug	apps/issuetrak/images/bug24.png	
feature	apps/issuetrak/images/feature24.png	
inquiry	apps/issuetrak/images/inquiry24.png	

Go back to the issue mapper and the category_icon field and set the picklist to issue_category_icon.

Log into the application and you can now see the icon with an image and tooltip relevant to the issue category.



Adding Actions Based on Data

We'll now provide additional functionality that will allow the user to Resolve, Close, Re-open and Reject an issue. The buttons will link to relevant issue workflow wizards and only the relevant action buttons will be visible on the list. E.g. an open issue can only be resolved. A closed issue can only be re-opened, etc.

The visibility of the buttons is derived from the current value of the status_id field on the issue record.

On the issue mapper, add the following fields

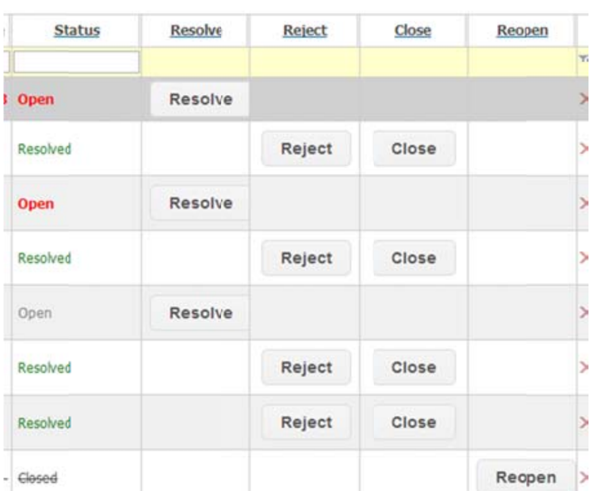
Field	Settings	Notes
issue_resolve	Key Name: issue_resolve Column Order: 220 Column Width: 10 CellType: Button .NET Type: System.Int32 Data Type: int CellTypeAttributes: ValueHidden AliasName: status_id FalseValues: 2;3 TrueValues: 1 Buttontype: ListButtonDetailButton HideFlavor: StandardDetail LinkKey: issue_id LinkMOP: issue!wiz_resolve Caption: Resolve Tooltip: Resolve Issue	

Field	Settings	Notes
issue_reject	Key Name: issue_reject Column Order: 230 Column Width: 10 CellType: Button .NET Type: System.Int32 Data Type: int CellTypeAttributes: ValueHidden AliasName: status_id FalseValues: 1;3 TrueValues: 2 Buttontype: ListButtonDetailButton HideFlavor: StandardDetail LinkKey: issue_id LinkMOP: issue!wiz_reject Caption: Reject Tooltip: Reject Issue	

Field	**Settings**	**Notes**
issue_close	Key Name: issue_close Column Order: 240 Column Width: 10 CellType: Button .NET Type: System.Int32 Data Type: int CellTypeAttributes: ValueHidden AliasName: status_id FalseValues: 1;3 TrueValues: 2 Buttontype: ListButtonDetailButton HideFlavor: StandardDetail LinkKey: issue_id LinkMOP: issue!wiz_close Caption: Close Tooltip: Close Issue	

Field	Settings	Notes
issue_reopen	Key Name: issue_reopen Column Order: 250 Column Width: 10 CellType: Button .NET Type: System.Int32 Data Type: int CellTypeAttributes: ValueHidden AliasName: status_id FalseValues: 1;2 TrueValues: 3 Buttontype: ListButtonDetailButton HideFlavor: StandardDetail LinkKey: issue_id LinkMOP: issue!wiz_reopen Caption: ReOpen Tooltip: ReOpen Issue	

We're not quite finished, but if you logon to the application you can see the effects of these fields with their "ValueHidden" effects.



What we will do now however, is group the action button fields into a single "Actions" field. We are going to add a new field to the mapper which will not have any data selected from the database. These are called "Exclude from select fields" named after the Field attribute associate with that behavior.

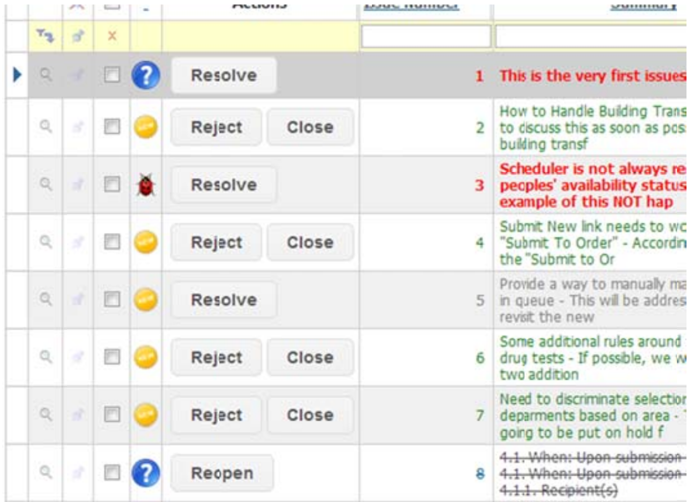
Add the following field to the issue mapper

Field	**Settings**	**Notes**
issue_actions	Key Name: issue_actions Column Order: 5 Column Width: 22 CellType: GroupBox .NET Type: System.String Data Type: varchar Attributes: ExcludeFromSelect CellTypeAttributes: ShowInList Table: + HideFlavor: StandardDetail Caption: Actions	

And revisit the action button fields to attach/associate those button fields with the "Action" GroupBox field

Field	**Settings**	**Notes**
issue_resolve issue_reject issue_close issue_reopen	Group: issue_actions	

Log into the application and navigate to the issues list. The Actions buttons are now grouped into a single column, with hidden buttons not contributing to unnecessary real estate.

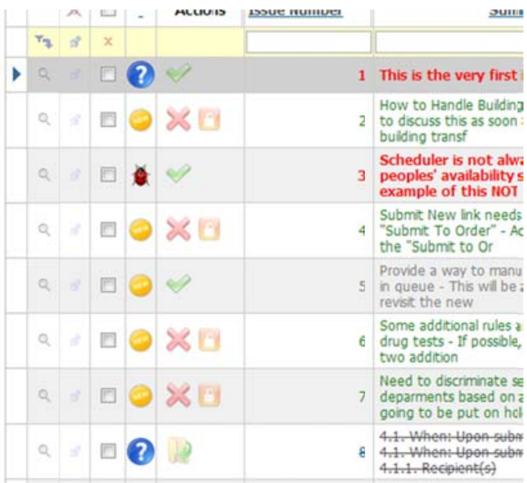


And we can also change the buttons to display images as opposed to text.

On the issue mapper, make the following field changes.

Field	Settings	Notes
issue_resolve	CellTypeAttributes: IconButton Icon: apps/issuetrak/images/resolve24.png	
issue_reject	CellTypeAttributes: IconButton Icon: apps/issuetrak/images/reject24.png	
issue_close	CellTypeAttributes: IconButton Icon: apps/issuetrak/images/close24.png	
issue_reopen	CellTypeAttributes: IconButton Icon: apps/issuetrak/images/reopen24.png	
issue_actions	Column Width: 8	

Log into the application and go to the Issue List to see the buttons displayed as images.


Actions
Issue Number
1 This is the very first
How to Handle Building
2 to discuss this as soon
building transf
Scheduler is not alw
3 peoples' availability
example of this NOT
Submit New link needs
4 "Submit To Order" -
the "Submit to Or
Provide a way to manu
5 in queue - This will be
revisit the new
Some additional rules
6 drug tests - If possible,
two addition
Need to discriminate
7 deparments based on
going to be put on ho
8 ~~4.1. When: Upon subm~~
~~4.1. When: Upon subm~~
~~4.1.1. Recipient(s)~~

Alternative Ways of Selecting Records

The most common control for selecting a specific item from a set of items is the combobox with a picklist. However when the items in the picklist grows beyond a certain size (say 30), it becomes more difficult for the user to quickly select an item. When the number of picklist items is approaching hundreds of items, not only does it directly impact user experience from the sheer volume of items to scroll through, but you then impact memory resource on the application server.

A typical 300 item picklist may represent about 18kB of memory consumption. If the application server is currently handling requests for 200 sessions, that represents a total memory load on one server of 3.5MB. A typical scenario is the growth of people added to the database (or issues!) As the application matures you could have a picklist of users with 100,000 or more items. If you used a picklist like this, it might take 5-10 minutes for the page to actually load a picklist with that many items!

To avoid problems with pickilst size, we can choose records using Find controls. They popup a search window where a user can provide search criteria to restrict a list of possible matches and then select a record back (and possibly additional, related data) onto the main page.

Finding a Single Item

The most appropriate place to utilize the find control is on the issue mapper, where we select a user to assign a case to. Right now it's just a picklist. But after the IssueTrak application becomes a global phenomenon a picklist will no longer be adequate.

The Find cell type requires the 'pairing' of two fields. One field is the "ID" field which is a field you would normally associate a picklist with, in this case the assigned_user_id field. The other field is the "DISPLAY" field which displays the text representation of the "ID" field. You obtain a display field in the mapper by selecting that field in the underlying view. You join the main table/view to a related table/view joining on the "ID" field to retrieve the "DISPLAY" field from the related table/view.

We're going to work on the issue_view and make another modification

In the SQL Enterprise Manager, open the file C:\NetQuarry\Customers\IssueTrak\Database\issues_data-views.sql

Locate the script for creating the view, "issue_view" and change the script as shown

Script

```
IF OBJECT_ID('[dbo].[issue_view]') IS NOT NULL DROP VIEW
[dbo].[issue_view]
GO

CREATE VIEW [dbo].[issue_view]
AS
SELECT i.issue_id, i.issue_number, i.summary,
i.description, i.assigned_user_id, i.status_id,
i.category_id, i.component_id, i.priority_id,
i.severity_id, i.project_id, milestone_id,
i.committed_date, i.created_by_id, i.updated_by_id,
i.date_created, i.date_updated, i.is_deleted,
i.revision, iuv.user_guid,
iuv.full_name AS assigned_user, iuv.email_address,
   CASE status_id
    WHEN 3 THEN 'iss_closed'
    WHEN 2 THEN 'iss_resolved'
    ELSE
     CASE priority_id
      WHEN 1 THEN 'iss_crit'
      WHEN 2 THEN 'iss_sev'
      WHEN 3 THEN 'iss_mod'
      ELSE 'iss_low'
     END
   END as row_css
FROM dbo.issue i WITH(NOLOCK) LEFT OUTER JOIN
individual_users_view iuv WITH(NOLOCK)
ON i.assigned_user_id=iuv.user_id
GO
```

Notes

Add the full_name and email_address field from the individual_users_view

Execute the issues_data-views.sql script to change the view.

Back in the NetQuarry Studio, go to the "issue" mapper. Select it and right click and choose "Create Fields…". Add the field assigned_user and email_address to the mapper.

Then make the following changes

Field	Settings	Notes
assigned_user_id	Picklist: <remove user_list and make blank> Column Width: 0	
assigned_user	Column Order: 50 Column Width: 20 Attributes: Required, Audit CellType: Find CellTypeAttributes: FirstFieldAsSelector, SizeSearchFields	
email_address	Column Width 0	

We also need to make additional property changes to the assigned_user field but are described separately for the sake of clarity

Property	Value	Notes
FindKey	assigned_user_guid	Provide a link to the main page for the current assigned user
FindMOP	individual!main	Satisfies two requirements. One is the target page for a navigation. The other is telling the platform which mapper is to be used when searching for a specific user. The mapper is defined by the page.
FindFields	assigned_user_id=user_id;assigned_user=full_name	This property defines which fields from the mapper (individual) of the FindMOP are mapped to fields of the target mapper (issue). We are pulling over two fields from the find mapper. The user_id to the assigned_user_id field on the issue mapper and the full_name field of the individual mapper to the assigned_user field of the issue_mapper.
FindFilter	user_id IN (SELECT DISTINCT user_id FROM user_roles WITH(NOLOCK) WHERE role_nm IN ('Users','Developer','System Admin'))	This property provides additional filtering to the searching on the Find mapper (individual). We are adding an additional filter to only display individuals who have a "User, Developer or Admin" role. Someone who is only a manager cannot be assigned an issue.

Now there's a little tidying up to perform. We have to fix the template used by the issue!main to render the assigned_user field, instead of the assigned_user_id field.

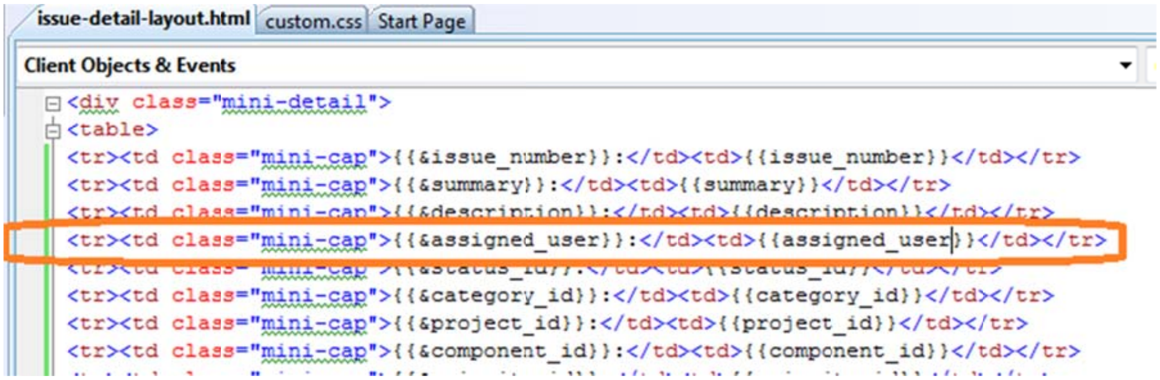
We also need to modify the individual mapper to improve the layout of the find results and provide fields for filtering down the search results.

We need to modify the workflow wizards to use the assigned_user field and remove the assigned_user_id field

Fix the Template

Open the template %NQROOT%\Templates\IssueTrak\issue\issue-detail-layout.html

Change the two occurrence of field “assigned_user_id” to “assigned_user”



Configure the Find UI

Go to the individual mapper and set the “Search” attribute on the following fields

first_name, last_name, email_address, company_id, user_id

Set all of the fields (with the Flavor Editor) with the HideFlavor:“Find”. Except for the fields we have selected as being searchable.

first_name, last_name, email_address, company_id, user_id

Remember, this list of fields should NOT have a HideFlavor of Find. The rest of the fields in the mapper will have the HideFlavor of Find.

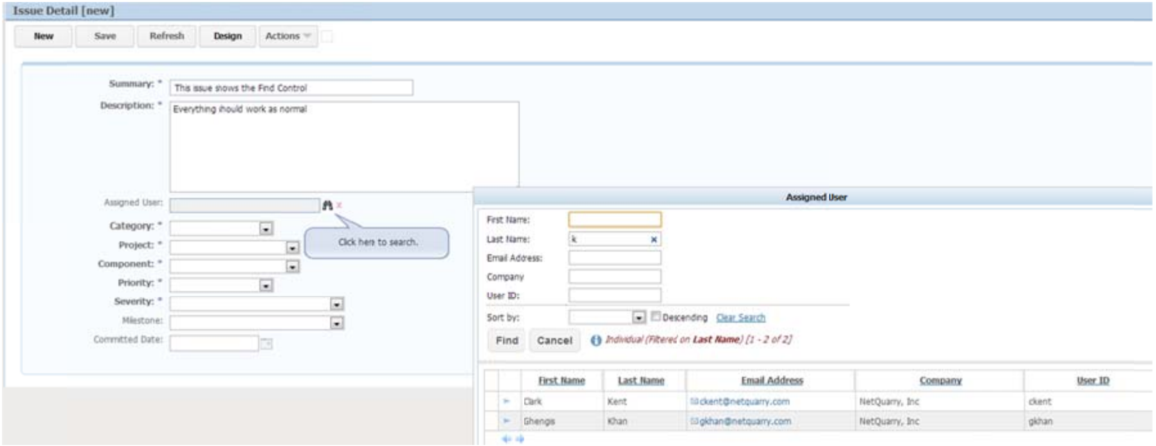
Correct the Issues Workflow Wizards

For each of issue workflow wizards (there are four of these). Go to the respective “issue” Page Elements. Change the FieldList property by replacing the reference of assigned_user_id to assigned_user

Log into the application and navigate to the Issue list.

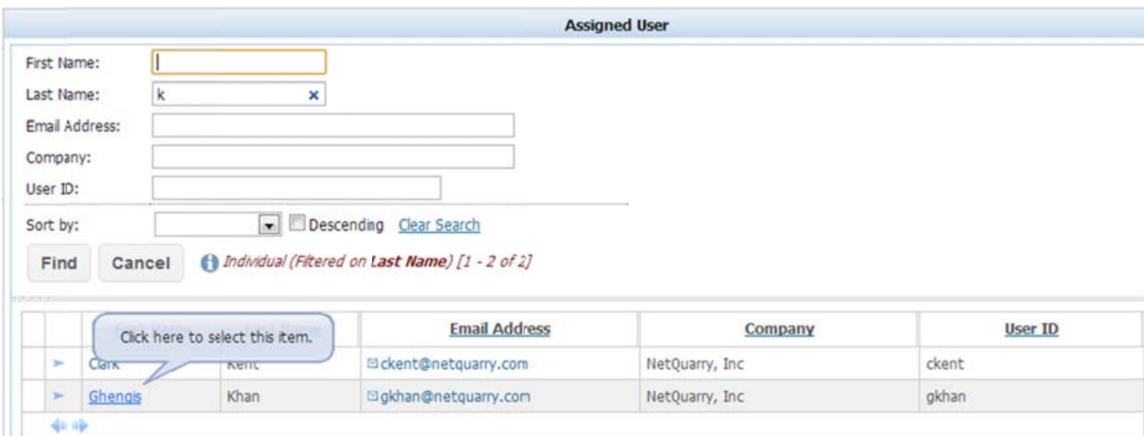
Then click on the “New” button to create a new issue.

When choosing a user, click on the binoculars icon to pop open the find. Enter some criteria to find some users matching your criteria



To select a user, click on the small arrow, or the link on the first column.

Once selected, the values are copied to the main page and the popup window automatically closes.



Finding Multiple Items

A feature related to single find control is to allow multiple items to be related to another object using a find type control which allows multiple selections.

We'll utilize this feature on issues, where we will provide the ability to relate one or more other issues to the current issue. We will have to add a new page element to the issue!main page to display the related issues.

There is already a view that we can use for this purpose and a mapper we can modify slightly to use that view.

In the list of mappers, select the mapper "issue_rel" and change the View Name to issue_rel_view. Move off the row to save and then select the same mapper and right click to "Create Fields…". Select the new fields and click OK.

Make the following changes to the fields

Field	Settings	Notes
issue_id	Column Width: 0 Attribute: DefaultFromParent, Required	
related_issue_id	Column Width: 0 Attribute: DefaultFromParent, Required	
rel_type_id	Column Width: 0	
summary	Column Order: 60 Column Width: 40	
issue_number	Column Order: 50 Column Width: 10 CellType: Find CellTypeAttributes:FirstFieldAsSelector, SizeSearchFields FieldBehaviorAttributes: AllowLinkInFindResults, ShowHoverSumary FindFields: related_issue_id=issue_id;issue_number=issue_number FindFilter: issue_id<>['issue_id] AND project_id=['project_id] FindKey: related_issue_id FindMOP: issue!main SummaryKey: related_issue_id SummaryMOP: issue!main Caption: Issue #	The Findfilter property is defined to specifically exclude being able to assign an issue to itself and only link an issue to another issue in the same project

Now add the Related Issue page element to the issue!main page.

In the Page Element subform, add the following Page Element information

Item	Value	Notes
Slot Name	ConsolePage	
Name	related_to	
Component	MiniList	
Order	50	
Attributes	MultiAdd	Required for providing the necessary UI on the page element.

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Mapper	issue_rel	
ParentViewKeySource	issue_id	
ViewKey	issue_id	
FieldList	issue_number;summary	
RowsPerPage	10	
Sort	issue_number	
LinkKey	issue_ id	
LinkMOP	issue_rel!detail	
MultiAddField	issue_number	
Caption	Related Issues	
LinkCaption	Related Issues (Click to View All)	

We had to tweak the individual mapper to better support using the Find. Adding fields to search on, hiding fields from the search results. We must do the same for the issue search.

On the issue mapper, apply the "Search" attribute to the following fields

issue_number, summary, description, assigned_user, component_id

Duplicate the assigned_user field on the issue mapper. The current field is a Find field with a locked text box and this field is locked on the find criteria. We want to be able to enter a name to search on.

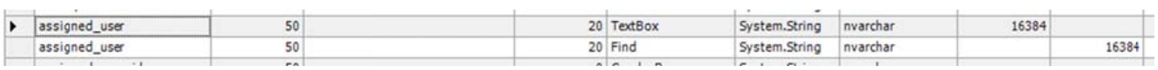
On the original assigned_user field, set the Exclude Flavor to “FindCritieria”. Press F5 to refresh the list of fields

On the duplicated assigned_user field, perform the following actions.

Property	Value	Notes
FindFields	<clear values	
FindFilter	<clear value>	
FindKey	<clear value>	
FindMOP	<clear value>	

Then change the cell type of the duplicated field to TextBox and continue setting properties as follows

Field	Property	Notes
assigned_user (Textbox)	IncludeFlavor: FindCriteria	

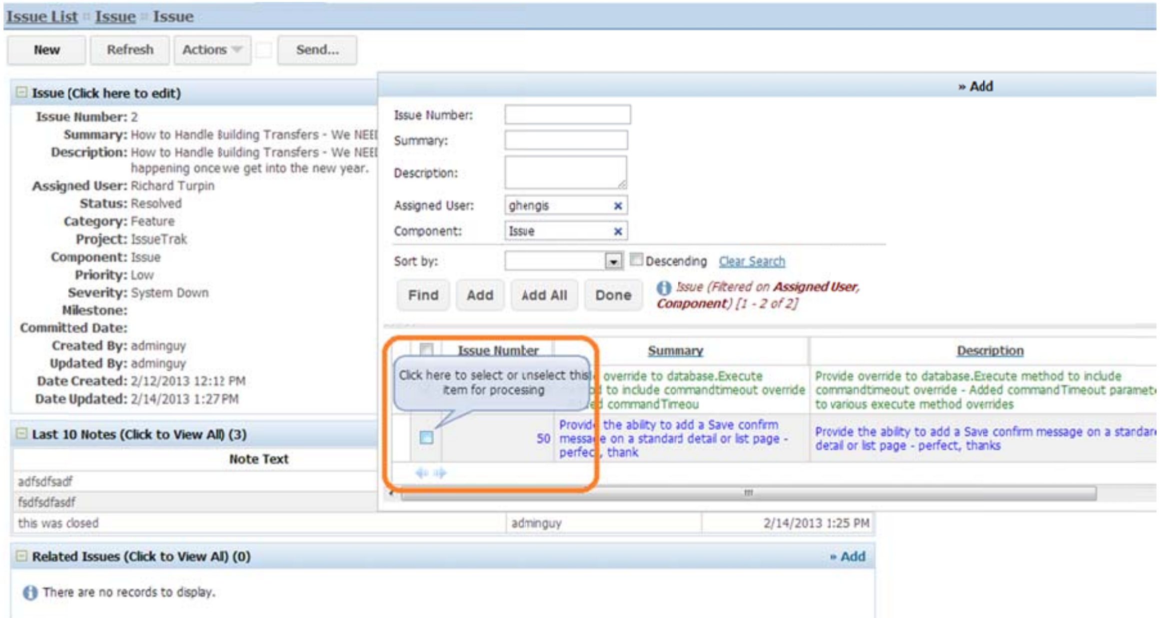


Apply the HideFlavor for Find on the following fields.

category_icon, issue_actions, created_by_id, updated_by_id, date_created, date_updated,

Now log in to the application and navigate to an existing issue.

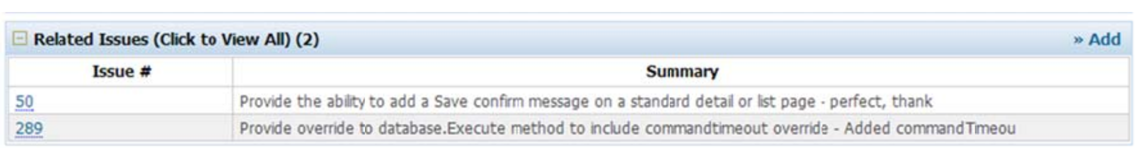
You will see the Related Issues pane on the screen and on the right of the pane there is an “Add” link. Click on that link and it will popup up a find window. You can provide criteria as before. The major difference for the multi-find is addition of check boxes that you can use to select one or more items and then click the “Add” button to associate the selected items with the parent record. Alternatively you can click on the “Add All” button to add all the items that match the filtered search results.



If you “Add All” a message appears to tell you how many items match the criteria.

Having added your selected items (or all) the criteria is re-queried to eliminate the added items from the list. The platform builds a filter criteria to exclude items already added to the related issue table.

When you then click on the “Done” button, the window closes and the parent window refreshes to display the related issues.



Save Your Metadata

Now would be a good time to save your metadata changes.

More Administrative Tools

We should now add a set of administrative tools to manage Companies (although we currently do have a company list, it will be moved from the main toolbar) and related Projects. And the Projects, with their related Components and Milestones.

We've looked at pages that use the Console Template where the data on the pages are static and you navigate to another page to make edits. Here, we'll create these admin pages using the Console Template but in a mode that supports in place navigation.

First we'll create the Company admin page. From the Company List, we drill down to a company detail where we can edit the Company Details, view Projects related to the company, individuals related to the company and audit history for the company record. In addition, we need to create a way to navigate "in place".

Create Company Admin Page

Go to the list of Pages (Alt+P) and create a new page with the following settings

Item	Value	Notes
Module	company	
Name	main	We will make this version of the page only visible to admin users.
Moniker	company admin page	
Template	ConsoleTemplate.aspx	
Mapper	<nothing>	
Priority	2	Pages are loaded by priority. If there are multiple pages with the same name and permissions, the page with the highest priority is the winner

Move off the row to save and select the new company admin page.

With that new Page selected, click on the Property sheet and set the following properties.

Property	Value	Notes
TemplateAttributes	RequiresMainRecord, ShowToolbar	
Columns	2	
ColumnWidths	150px;100%	This column width specification is set up for a fixed width of column 0. Column 1 has a floating width of 100% and will therefore size to the remaining real estate.
HomePane	detail	The name of a pane to open if the companYmain page is navigated to without a pane being specified.
Caption	Company - [company_nm]	

Add this page element

Item	Value	Notes
Slot Name	Main	
Name	hidden	
Component	MiniDetail	
Order	0	
Attributes	Hidden	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Mapper	company	
Caption	Company	

Add this page element

Item	Value	Notes
Slot Name	ConsolePage	
Name	detail	
Component	ConsoleDetail	
Order	10	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
PaneAttributes	PaneNavigation	
Column	1	
Mapper	company	
NewTarget	company!detail	
Caption	Company Details – [company_nm]	

Add this page element

Item	Value	Notes
Slot Name	ConsolePage	
Name	company-links	
Component	MiniNav	
Order	20	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Column	0	
Navigator		Do not set yet,we haven't created a navigator yet
Caption	Links	

Add this page element

Item	Value	Notes
Slot Name	ConsolePage	
Name	project-list	
Component	ConsoleDatasheet	
Order	30	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
PaneAttributes	PaneNavigation	
Column	1	
Mapper	project	
Sort	project_name	
DetailTarget	project!main	
NewTarget	project!detail	
Caption	Projects	

Add this page element

Item	Value	Notes
Slot Name	ConsolePage	
Name	individual-list	
Component	ConsoleDatasheet	
Order	40	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
PaneAttributes	PaneNavigation	
Column	1	
Mapper	individual	
Sort	last_name	
DetailTarget	individual!main	
NewTarget	individual!detail	
Caption	Individuals	

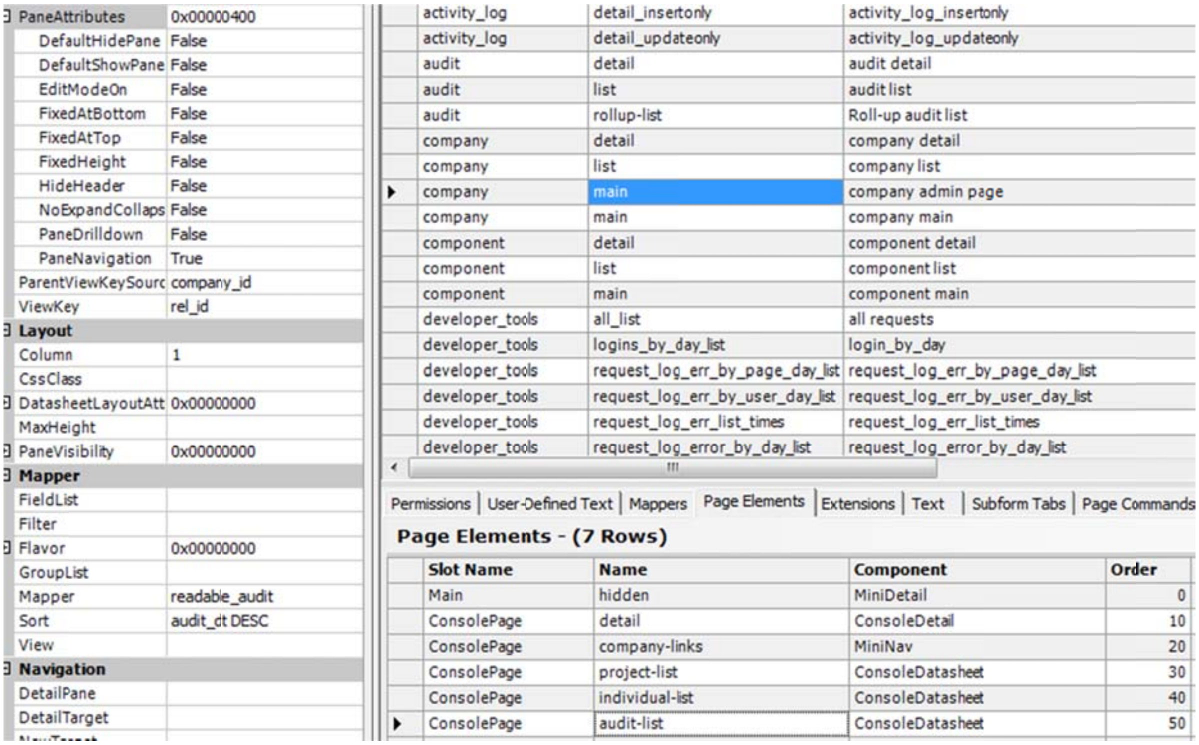
Add this page element

Item	Value	Notes
Slot Name	ConsolePage	
Name	audit-list	
Component	ConsoleDatasheet	
Order	50	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
PaneAttributes	PaneNavigation	
ParentViewKeySource	company_id	
ViewKey	rel_id	
Column	1	
Mapper	readable_audit	
Sort	audit_dt DESC	
Caption	Audit History	

Now you should have something that looks like this



Create Navigation Links on Company Admin Page

Now we have created the panes of the page, we can create the navigation element that links to those panes. And once we have created that navigation element, we can set the Navigator property of the MiniNav page element on the company admin page.

Go to the NavBars list under navigators and add the following navbar.

Item	Value	Notes
Module	company	
Name	company_links	
Attributes	AddParentInfo	This forces parent information on to all of the navbar's targets. This ensures that when you click on the workflow link it takes you to the wizard filtered to the correct issue.

Select the "company_links" navbar and then in the Targets subform below, add the following Targets.

Item	Value	Notes
Module	company	
Name	details	
Target Type	Pane	
Target	detail	
Sort Order	10	

With that new Target selected, click on the Property sheet and set the following properties.

Property	Value	Notes
ParentViewKeySource	company_id	
ViewKey	company_id	
Caption	Company Detail	

Add this target

Item	Value	Notes
Module	company	
Name	projects	
Target Type	Pane	
Target	project-list	
Sort Order	20	

With that new Target selected, click on the Property sheet and set the following properties.

Property	Value	Notes
ParentViewKeySource	company_id	
ViewKey	company_id	
Caption	Projects	

Add this target

Item	Value	Notes
Module	company	
Name	individuals	
Target Type	Pane	
Target	individual-list	
Sort Order	30	

With that new Target selected, click on the Property sheet and set the following properties.

Property	Value	Notes
ParentViewKeySource	company_id	
ViewKey	company_id	
Caption	Individuals	

Add this target

Item	Value	Notes
Module	company	
Name	audit	
Target Type	Pane	
Target	audit-list	
Sort Order	40	

With that new Target selected, click on the Property sheet and set the following properties.

Property	Value	Notes
ParentViewKeySource	company_id	
ViewKey	company_id	
Caption	Audit	

Go back to the Pages list and select the company admin page. Select the company-links page element and set the following property

Property	Value	Notes
Navigator	company_links	

Create a Navbar for Admin Links

Go back to the Navbars list and create a new navbar with the following settings

Item	Value	Notes
Module	IssueTrak-admin	
Name	admin_links	

Select the “admin_links” navbar and then in the Targets subform below, add the following Targets.

Item	**Value**	**Notes**
Module	company	
Name	company-list	
Target Type	Page	
Target	company!list	
Sort Order	10	

Put the Admin Links on the Admin Page

With that new Target selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Caption	Companies	

Go back to Pages list and select the issuetrak-admin!main page.

Add the following page element

Item	Value	Notes
Slot Name	ConsolePage	
Name	admin_links	
Component	MiniNav	
Order	30	

With that new Page Element selected, click on the Property sheet and set the following properties.

Property	Value	Notes
Navigator	admin-links	
Caption	Admin Links	

Remove the Company list link from the Main NavBar

Go back to the Navbars list and select the "main" bar. Select the company_list page element. You can either delete the company_list target, or set the "Disabled" attribute. It's always easier to re-enable the meta later if you need it back. We don't want to see the list of companies on the main navigation area.

Add Auditing to Company

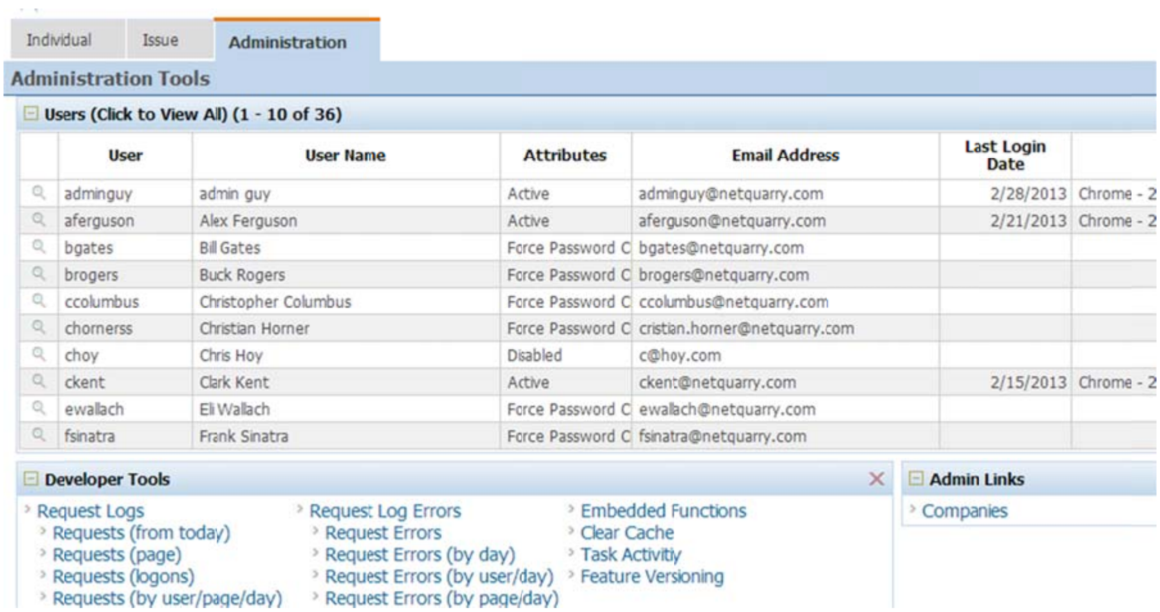
Go to the Mappers list and add the Readable.Audit extension to the company mapper.

Set the audit attribute on user editable fields of the company mapper.

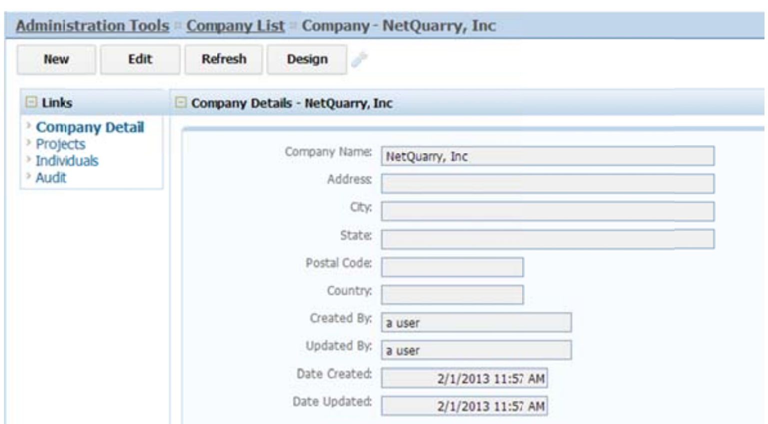
Test the Company Admin Page

Login to the application. The Company link should no longer be there.

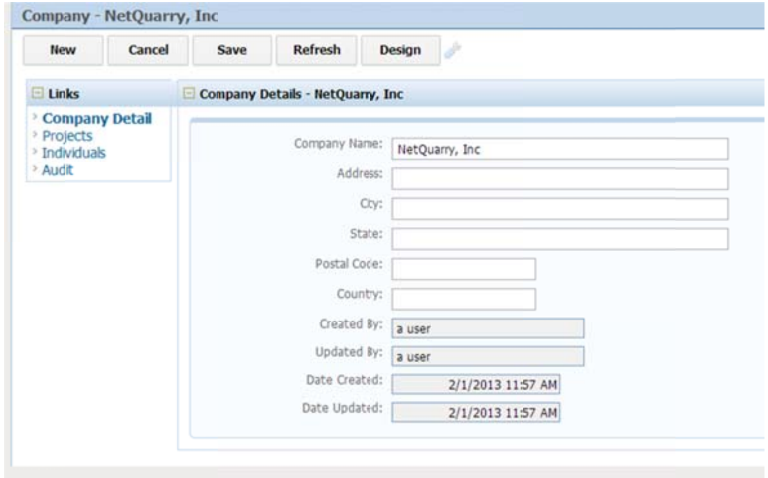
And navigate to the Admin page and the companies list should be visible on the Admin Links pane.



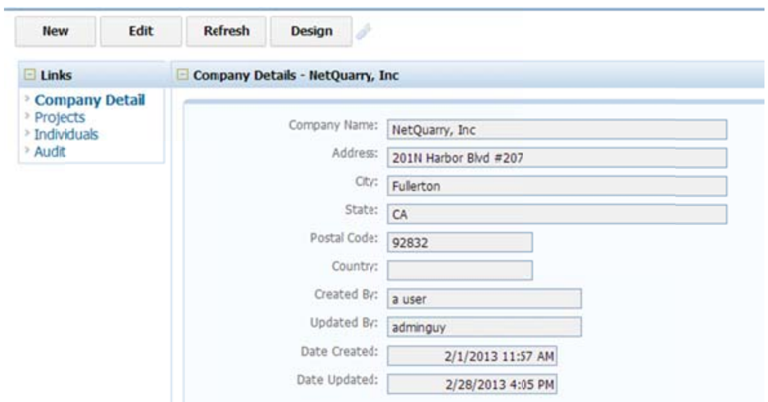
Go to the companies list and drill down to a company detail. The page renders the company detail but notice it is locked, or read only.



This is the default behavior for a pane navigation console. To make edits, you click on the Edit button, which navigates back to the page with the additional “edit=1” query string parameter. The “Edit” button, changing to a “Save” button.

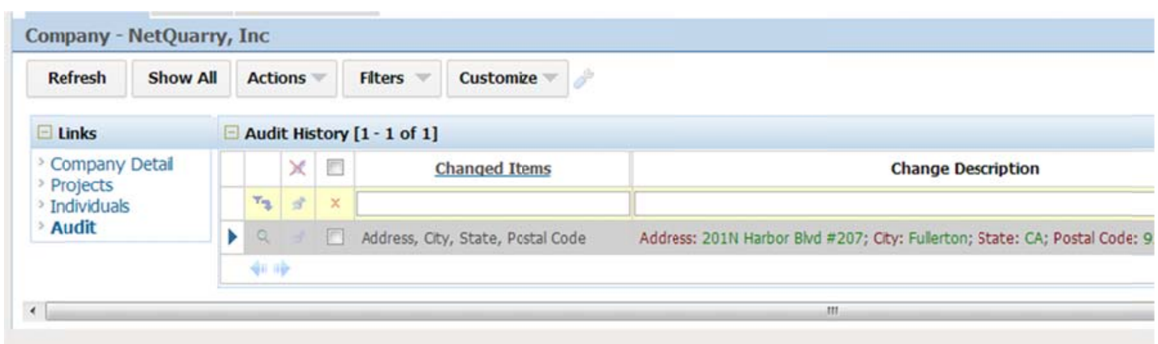


Making any changes and saving will revert back to the read only mode.



Clicking on the other links takes you to the relevant list pages. On these pages you can click on the New button, or New link to create a new item.

The audit link shows audit information for the company.



Save Your Metadata

In the following section we are going to make some Permissions changes to highlight how we can control access to various user interface elements. It is STRONGLY recommended that you save your metadata at this point.

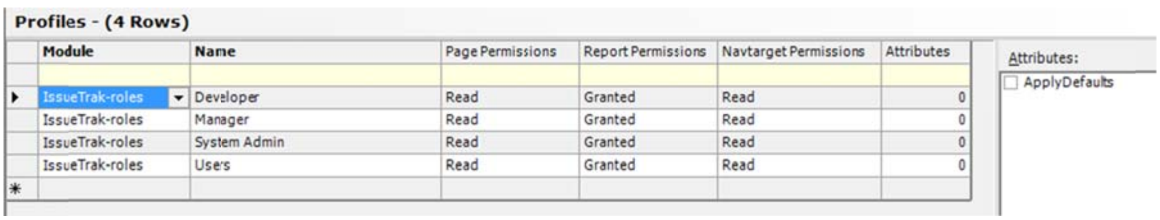
Permissions

We've now created a number of pages in the application and imported a set of users with different roles. However, we have not yet touched on the permissions handling provided by the platform.

The Platform operates in a permissive style. Meaning that by default, if you make no changes to behavior all user roles have access to all object controlled by permissions. This means Pages, Reports, Nav Targets, Page Elements, Mappers, Fields, Filters, and Templates.

There is a way to control the default behavior of permissions but not for all permissioned objects. Default behaviors for permissions (whether permissive or restrictive) is only supported on three main UI objects. Pages, Reports, Nav Targets.

In the NetQuarry Studio, click on the Profiles link under Permissions.



If you remember from earlier, you added three roles, Developer, Manager and System Admin. They were automatically created with permissive options (Read/Granted). Then when the associated objects were created they were given read permissions for all roles.

So why should you change the default behavior?

As your application grows you may suddenly find you have literally thousands of pages and by allowing all user roles to see all pages, you are potentially giving unauthorized access to data.

For example. If you can guess the name of a page, you can enter that into the address bar of the browser and get to that page. Even if there is no explicit way to navigate to that page.

So the best approach is to set up permissions to deny access to certain roles by default. When pages are created, you have to make an explicit decision to allow roles to see that page.

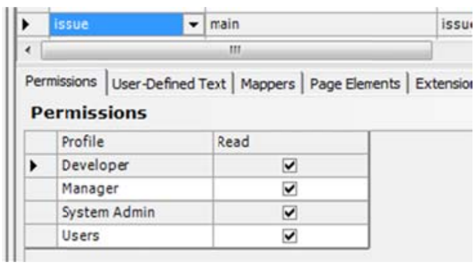
Change the default permissions for Pages and Reports

For User and Manager roles, set permissions to None AND set the attribute ApplyDefaults.

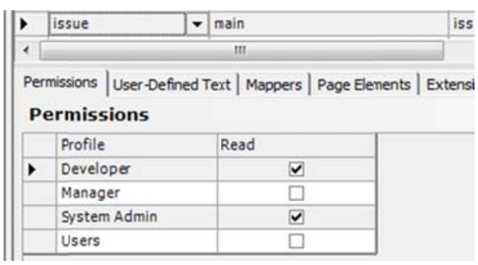
What we have done is change the default behavior in the studio for page and report security.

If you look at an existing page in the studio you will see the effects of this change.

Before…



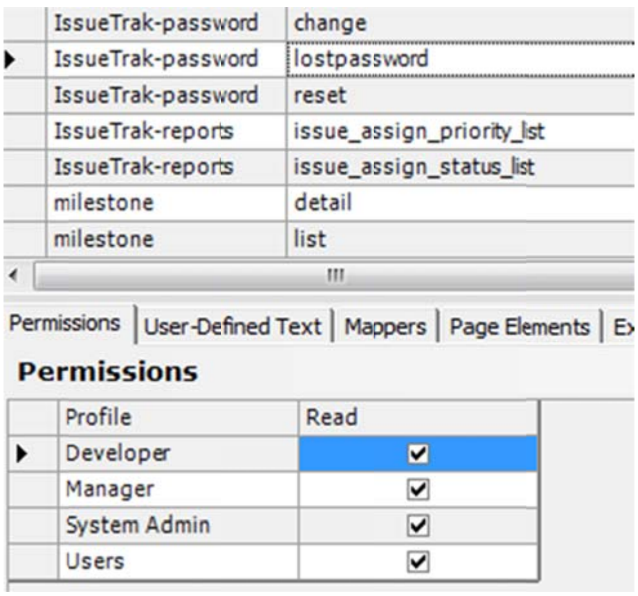
After…



Let's logon to the application as the Admin Guy user. You should not see any differences. All the pages you were able to access should still be there and all functionality is still available.

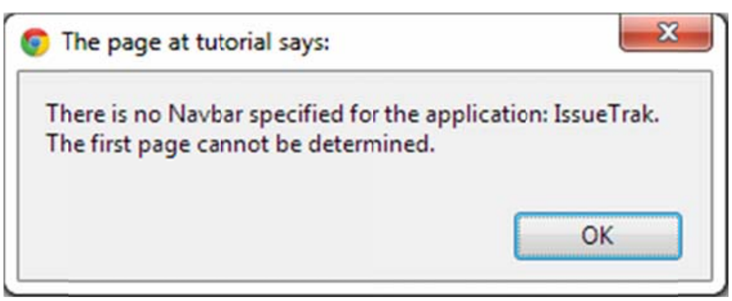
Go back to the studio and make the following permission changes on the listed pages

Page	Permissions	Notes
IssueTrak-password!change	System Admin, Developer, Users, Manager	
IssueTrak-password!lostpassword	System Admin, Developer, Users, Manager	
IssueTrak-password!reset	System Admin, Developer, Users, Manager	



Log out and log in as a different user. Let's say ckent with the initial password of password1.

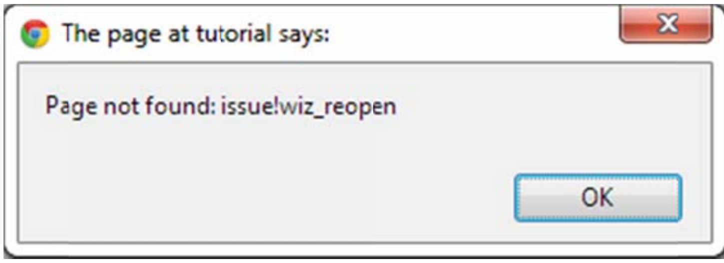
Once you change the password and log in again, you get the following error message



You haven't touched any navbar permissions, only page permissions. What's happened is that the platform first loads the pages that a particular user is allowed to see. Nav Targets of type "page" are not displayed when the target page is not available. And if all the Nav Targets for a navigator are not visible, then the parent navbar is not visible.

Go to the Studio and the list of Pages. Select the page issue!list. Change the permissions of the Page, adding Read permission for Users.

Now you can login as "ckent" and ONLY the list of issues is visible. Note there is no ability to drill down to an issue detail page. The link to navigate to an individual page from the Assigned User field is not available. The workflow action buttons are visible, but when you click on one, you get the error



\

This gives you an indication of the effects of setting restrictive default permissions. There may be unintended consequences of your further development in that while developing as a developer you can automatically see all pages. However you also need to test your work logged on as other less privileged users. To confirm they have the necessary access to pages.

Back in the studio. Make the following permissions changes

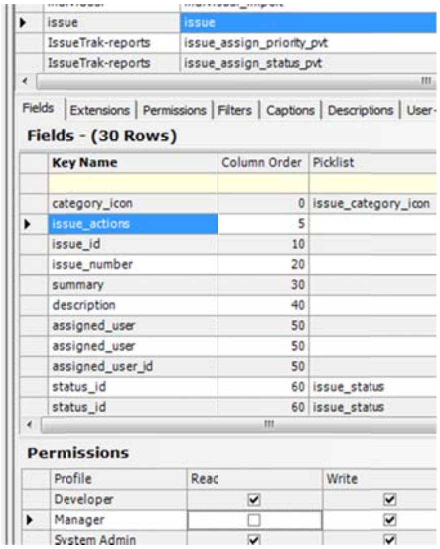
Page	Permissions	Notes
documents!related_doc_list	Users	
documents!related_doc_detail	Users	
individual!main	Users, Manager	
individual!list	Manager	
individual!detail	Manager	
issue!list	User, Manager	
issue!detail	Users	Managers are not allowed to edit cases
issue!main	Users, Manager	
issue!list-my-issues	Users	
issue!list-my-open-issues	Users	
issue!list-my-resolved-issues	Users	
issue!wiz_resolve	Users	
issue!wiz_reopen	Users	
issue!wiz_reject	Users	
issue!wiz_close	Users	
issue_rel!list	Users	
issuetrak-home!dashboard	Manager	
note!list	Users	
note!main	Users	
note!detail	Users	

Now login as a manager. For example aferguson (password1)

You will login to the Home page with the reports Dashboard, Individual list and Issue list as available navigation options.

If you go to the issue list you will see the action buttons are available. But if you click on them they don't work because a manager is not allowed to see any of the workflow wizards.

In the studio, go to the list of mappers and select the issue mapper. Then in the Fields subform, select the issue_actions field. Uncheck the "Read" permission for Manager Role.



Log in as a manager. The Actions column will no longer be visible.

Save Your Metadata

Save your metadata before continuing with the next part of the tutorial.

Next Steps

You have now created the basics of the IssueTrak application. Proceed to the next document, NetQuarry – IssueTrak Tutorial – Part3, where we will describe how to use the object and event model of the NetQuarry platform.