Role Based Access Manager

Role Based Access Control Management

for the Yii PHP Framework

Developed for the Yii Community by

PBM Web Development

PBM

Role Based Access Manager

Contents

Figures

Role Based Access Manager

Introduction

Role Based Access Manager (RBAM) is a Yii module that provides complete management of Authorisation Data (Authorisation Items, Authorisation Hierarchy, and Authorisation Assignments) for Yii's Role Based Access Control system via a browser interface; it is intended for use in development and end-user administration environments.

RBAM has an intuitive "Web 2.0" interface to easily manage Authorisation Items (Roles, Tasks, and Operations), their hierarchy, and Authorisation Assignments. It presents all of an Authorisation Item's information in one place providing a comprehensive overview and complete management of the item.

RBAM's "Drill-down" and "Drill-up" features quickly show an item's position in the Authorisation Hierarchy, what permissions it inherits (Drill-down) and which Roles inherit its permissions (Drill-up).

RBAM is built on top of Yii's CAuthManager component and supports both of Yii's built-in Authorisation Managers, CDbAuthManager and CPhpAuthManager, and authorisation managers extended from them.

Features

- Complete management of Authorisation Data
    - Manages Authorisation Items (Roles, Tasks, and Operations), their business rules and data
    - Manages Authorisation Item hierarchy
    - Manages Authorisation Assignments, their business rules and data, and assignment/revocation of Roles to/from users
- Drag & drop interface for Authorisation Hierarchy management
- Drill-down and drill-up
    - Drill-down from an Authorisation Item to see what permissions it inherits
    - Drill-up from an Authorisation Item to see which Roles inherit its permissions
- Filtering, paging, and sorting of Authorisation Items, Authorisation Assignments, and users
- Confirmation dialogs for all changes to Authorisation Data
- Integrated help
- Breadcrumb support
- Integrates with CDbAuthManager, CPhpAuthManager, and authorisation managers extended from them
- Compound attribute and related model support for user names
- Internationalization (I18N)
- Initialisation of Authorisation Data using built-in or user-supplied data (can import CPhpAuthManager data)
- Generate Authorisation Data based on the application's modules, controllers, and actions
- CMenu integration
- Simple installation and configuration

Requirements

- JavaScript enabled browser
- CDbAuthManager, CPhpAuthManager, or an authorisation manager component extended from them
- A User model with an attribute that is the model's primary-key and an attribute or attributes that provide the names of users

Tip: To enable quick evaluation of RBAM, you can install RBAM in Yii's "*testdrive*" demo application and add a User model (see Appendix A for the code and Appendix B – Minimal Schema for the User Table for the minimal schema RBAM requires).

Log in as "demo/demo" or "admin/admin". The logged user will be assigned the "RBAC Manager" role on initialising RBAM, but will *not* appear in the list of users as they are not contained in the user model.

License

Compatibility

	Yii	Chrome	Firefox	MSIE	Opera	Safari	OS
Tested with	1.1.5	8.0	3.68	8.0	10.63	5.0	Windows 7
Should work with	1.x.x	All	2.0+	6.0+	9.0+	3.0+	All

Installation & Configuration

1. Download RBAM from http://www.yiiframework.com/extension/rbam/
2. Extract the module and place in the required folder. RBAM can be installed as a "top level" module (i.e. under *protected/modules*) or a nested module (i.e. in the "modules" directory of a parent module); RBAM can be nested at any depth.

3. Edit your application configuration file (Yii's default configuration file is */protected/config/main.php*) and add:
    a. If a top level module

```
'modules'=>array(
  'rbam'=>array(
    // RBAM Configuration
  ),
),
```

    b. If a nested module:

```
'parentModule'=>array(
  // Parent Module Configuration
  'modules'=>array(
    'rbam'=>array(
      // RBAM Configuration
    ),
  ),
),
```

4. RBAM Configuration consists of setting the required values of the module's public properties.

Default Configuration Values

The default values are intended to allow RBAM to work "out of the box"; it is only necessary to provide configuration values for those options where the default value does not meet the needs of your application.

```
'rbam'=>array(
   'applicationLayout'=>'application.views.layouts.main',
   'authAssignmentsManagerRole'=>' Auth Assignments Manager',
   'authenticatedRole'=>'Authenticated',
   'authItemsManagerRole'=>'Auth Items Manager',
   'baseScriptUrl'=>null,
   'baseUrl'=>null,
   'cssFile'=>null,
   'development'=>false,
   'exclude'=>'rbam',
   'guestRole'=>'Guest',
   'initialise'=>null,
   'layout'=>'rbam.views.layouts.main',
   'juiCssFile'=>'jquery-ui.css',
   'juiHide'=>'puff',
   'juiScriptFile'=>'jquery-ui.min.js',
   'juiScriptUrl'=>null,
   'juiShow'=>'fade',
   'juiTheme'=>base',
   'juiThemeUrl'=>null,
   'pageSize'=>10,
   'rbacManagerRole'=>'RBAC Manager',
   'relationshipsPageSize'=>5,
   'showConfirmation'=>3000,
   'showMenu'=>true,
   'userClass'=>'User',
   'userCriteria=>array(),
   'userIdAttribute=>'id',
   'userNameAttribute=>'username',
)
```

Figure 1 - Default Configuration Values

Note: If your application is *"ended"*, e.g. has front-end and back-end applications, you will need to configure `applicationLayout` for RBAM to render correctly.

Note: RBAM itself is not "*ended*".

RBAM

Inheritance	RbamModule » CWebModule » CModule » CComponent

Public Properties

See CWebModule for inherited properties.

Name	Type	Description
applicationLayout	string	Path alias to the application's layout file. RBAM's content and layout are the content for this layout file.
authAssignmentsManagerRole	string	Name of the role that grants permission to manage user role assignments.
authenticatedRole	string	Name of the role that grants permissions to users that are logged in.
authItemsManagerRole	string	Name of the role that grants permission to manage authorisation items.
baseScriptUrl	string	The base script URL for all module resources (e.g. JavaScript, CSS file, images).
baseUrl	string	The base URL used to access RBAM.
cssFile	string	The URL of the CSS file used by this module.
development	boolean	Set TRUE to enable development mode.
exclude	mixed	Modules to exclude when generating authorisation data.
guestRole	string	Name of the role that grants permissions to users that are not logged in.
initialise	mixed	Defines whether RBAM should initialise the RBAC system and if so with what values.
juiCssFile	string	The JUI theme CSS file name.
juiHide	string	The effect used to hide JUI dialogs.
juiScriptFile	string	The main JUI JavaScript file.
juiScriptUrl	string	The root URL that contains all JUI JavaScript files.
juiShow	string	The effect used to show JUI dialogs.
juiTheme	string	The JUI theme name.
juiThemeUrl	string	The root URL that contains all JUI theme folders.
layout	string	Path alias to layout file.
pageSize	integer	The number of auth items displayed on a page.
rbacManagerRole	string	Name of the role that grants permission to manage authorisation items and user role assignments.

Name	Type	Description
relationshipsPageSize	integer	The number of auth items displayed in the relationship areas of the *Manage Item* screen.
showConfirmation	integer	The number of milliseconds to display confirmation dialogs.
showMenu	boolean	Whether to show the RBAM menu.
userClass	string	The class name of the user model.
userCriteria	array	The criteria applied in order to filter the list of users.
userIdAttribute	string	The name of the attribute that provides a unique id in the user model.
userNameAttribute	mixed	Attribute(s) in the user or related models used to display the user's name.

Public methods

See CWebModule for inherited methods.

Name	Description
getMenuItem()	Returns the module menu item.
getMenuItems()	Returns the module menu items.

Property Details

applicationLayout

```
public string $applicationLayout;
```

Path alias to the application's layout file.

RBAM's content and layout are the content for this layout file.

This will need configuring for "ended" applications, e.g. has front-end and back-end applications.

Defaults to 'application.views.layouts.main'

authAssignmentsManagerRole

```
public string $authAssignmentsManagerRole;
```

Name of the role that grants permission to manage user role assignments.

Defaults to 'Auth Assignments Manager'

authenticatedRole

```
public string $authenticatedRole;
```

Name of the role that grants permissions to users that are logged in. This will be added to CAuthManager::defaultRoles.

Note: It is not necessary to declare this role in your AuthManager configuration.

Defaults to 'Authenticated'

authItemsManagerRole

```
public string $authItemsManagerRole;
```

Name of the role that grants permission to manage authorisation items.

Defaults to 'Auth Items Manager'

baseScriptUrl

```
public string $baseScriptUrl;
```

The base script URL for all module resources (e.g. JavaScript, CSS file, images).

If NULL (default) the integrated module resources (which are published as assets) are used.

Defaults to NULL

baseUrl

`public string $baseUrl;`

The base URL used to access RBAM.

If NULL (default) the baseUrl is:

'/parentModule1Id/.../parentModuleNId/rbam'.

Do not append any slash character to the URL.

Defaults to NULL

cssFile

`public string $cssFile;`

The URL of the CSS file used by this module.

If NULL (default) the integrated CSS file is used.

If === FALSE a CSS file must be explicitly included, e.g. in the layout.

Defaults to NULL

development

`public boolean $development;`

Set TRUE to enable development mode.

In development mode assets (e.g. JavaScript and CSS files) are published on each access and options to initialise (if RbamModule::initialise is not empty) and generate authorisation data are shown.

Defaults to FALSE

exclude

`public mixed $exclude;`

Modules to exclude when generating authorisation data. Either an array or comma delimited string of module ids.

Defaults to 'rbam'

guestRole

`public string $guestRole;`

Name of the role that grants permissions to users that are not logged in. This will be added to CAuthManager::defaultRoles.

Note: It is not necessary to declare this role in your AuthManager configuration.

Defaults to 'Guest'

initialise

`public mixed $initialise;`

Defines whether RBAM should initialise the RBAC system and if so, with what values.

WARNING: Initialising the RBAC system will clear existing authorisation data (auth items, auth item children, and assignments).

String: the path to a file that returns an array that defines the authorisation data that RBAM will use to initialise the RBAC system. The array format is that used by CPhpAuthManager, meaning that this option can be used to import authorisation data if changing to CDbAuthManager; see below for the array format.

Array: defines the authorisation data that RBAM will use to initialise the RBAC system. The array format is that used by CPhpAuthManager; see below for the array format.

Boolean: If === TRUE RBAM will initialise the RBAC system with default auth items and auth item children; the current user will be assigned the *RBAC Manager* role.

If empty (default) no initialisation is performed.

Defaults to NULL

juiCssFile

`public string $juiCssFile;`

The JUI theme CSS file name.

The file ***must*** exist under the URL specified by *juiThemeUrl*/*juiTheme*.

To include multiple theme CSS files (e.g. during development, to include individual plugin CSS files), set this property as an array of the CSS file names.

If === FALSE a JUI CSS file must be explicitly included, e.g. in the layout.

Defaults to 'jquery-ui.css'

juiHide

`public string $juiHide;`

The effect used to hide JUI dialogs.

The following effects are available: blind, bounce, clip, drop, explode, fold, highlight, puff, pulsate, scale, shake, size, and slide.

Set to empty string for no effect.

Defaults to 'puff'

juiScriptFile

```
public string $juiScriptFile;
```

The main JUI JavaScript file.

The file ***must*** exist under the URL specified by *juiScriptUrl*.

To include multiple script files (e.g. during development, to include individual plugin script files rather than the minimized JUI script file), set this property as array of the script file names.

If === FALSE a JUI script file must be explicitly included, e.g. in the layout.

Defaults to 'jquery-ui.min.js'

juiScriptUrl

```
public string $juiScriptUrl;
```

The root URL that contains all JUI JavaScript files.

If NULL (default) the JUI package included with Yii is published and used to infer the root script URL. You should set this property if you intend to use a JUI package whose version is different from the one included in Yii.

There must be a file whose name is specified by *juiScriptFile* under this URL.

Do not append any slash character to the URL.

Defaults to NULL

juiShow

```
public string $juiShow;
```

The effect used to show JUI dialogs.

The following effects are available: blind, bounce, clip, drop, explode, fold, highlight, puff, pulsate, scale, shake, size, and slide.

Set to empty string for no effect.

Defaults to 'fade'

juiTheme

```
public string $juiTheme;
```

The JUI theme name.

There must be a directory whose name is the same as this property value (case-sensitive) under the URL specified by *juiThemeUrl*.

Defaults to 'base'

juiThemeUrl

`public string $juiThemeUrl;`

The root URL that contains all JUI theme folders.

If NULL (default) the JUI package included with Yii is published and used to infer the root theme URL. You should set this property if you intend to use a theme that is not found in the JUI package included in Yii.

There must be a directory (case-sensitive) whose name is specified by *juiTheme* under this URL.

Do not append any slash character to the URL.

Defaults to NULL

layout

`public string $layout;`

Path alias to layout file.

Defaults to 'rbam.views.layouts.main'

pageSize

`public integer $pageSize;`

The number of auth items displayed on a page.

Defaults to 10

rbacManagerRole

`public string $rbacManagerRole;`

Name of the role that grants permission to manage authorisation items and user role assignments.

Defaults to 'RBAC Manager'

relationshipsPageSize

`public integer $relationshipsPageSize;`

The number of auth items displayed in the relationship areas of the *Manage Item* screen. If empty defaults to *pageSize*.

Defaults to 5

showConfirmation

`public integer $showConfirmation;`

The number of milliseconds to display confirmation dialogs.

Dialogs can be closed before this time by clicking the "OK" button.

Defaults to 3000 (3 seconds)

showMenu

```
public boolean $showMenu;
```

Whether to show the RBAM menu. If true the RBAM module renders the RBAM menu. Set FALSE if the menu is rendered by the application.

See getMenuItem()

Defaults to TRUE

userClass

```
public string $userClass;
```

The class name of the user model

Defaults to 'User'

userCriteria

```
public array $userCriteria;
```

The criteria applied in order to filter the list of users.

CDbCriteria property values indexed by property name.

Defaults to array()

userIdAttribute

```
public string $userIdAttribute;
```

The name of the attribute that provides a unique id in the user model.

Defaults to 'id'

userNameAttribute

```
public mixed $userNameAttribute;
```

Attribute(s) in the user or related models used to display the user's name; compound attributes are supported, e.g. the user's given and family names, e.g. "Angela Other".

String: if a single attribute, the name of the attribute.

If multiple attributes, a comma delimited list of the join string followed by attribute names. If a comma is used in the join string escape it with a backslash, e.g. '\, ,given_name,family_name'

Array: the first element is the join string, subsequent elements are attribute names, e.g. array(', ','profile.given_name','profile.family_name')

If using compound elements you can specify which, if any, are to be rendered as initials only by adding a comma delimited string or array as the final element. This has a similar format as above; the first element is rendered after an initial and the following elements are the attributes to be rendered as initials, e.g. array(', ','profile.family_name','profile.given_name', array('.','profile.given_name')) will render as "Other, A."

NOTE: if using a comma delimited string within a comma delimited string, the delimiting commas in the internal string must be escaped. If you wish to use a comma as the character after initials it must be escaped with a backslash, not forgetting to escape the backslash if using a string in a string, i.e. '\\\,'.

The following are valid and equivalent:

- array(', ','family_name','given_name', array('.','given_name'))
- array(', ','family_name','given_name', '.,given_name')
- '\, ,given_name,family_name,.\,given_name'

Defaults to 'username'

Method Details

getMenuItem()

```
public array getMenuItem(array $item=array())
```

$item	array	The menu item. Merged recursively with the defaults.

Returns the module menu item, including sub-items.

This method should be used if adding RBAM as an item in a CMenu with other items.

getMenuItems()

```
public array getMenuItems(array $items=array())
```

$item	array	Menu items. Merged recursively with the defaults.

Returns the module menu items and sub-items.

This method should be used if the RBAM menu is "stand-alone".

Using RBAM

RBAM has two main functions:

- Management of Authorisation Items
- Management of Authorisation Assignments

In addition RBAM can:

- Initialise RBAC authorisation data using built-in or user supplied authorisation data
- Generate authorisation data based on your application's modules, controllers, and actions

> **Note:** Management of Authorisation Items and Generation of Authorisation Data require authItemsManager permission, Management of Authorisation Assignments requires authAssignmentsManager permission, Initialisation of RBAC authorisation data required either that none currently exists or rbacManagr permission.

Access RBAM at: *http://your.domain/index.php?r=[parent module/]*rbam*

Unless you are initialising the RBAC system, you will see the RBAM home page.

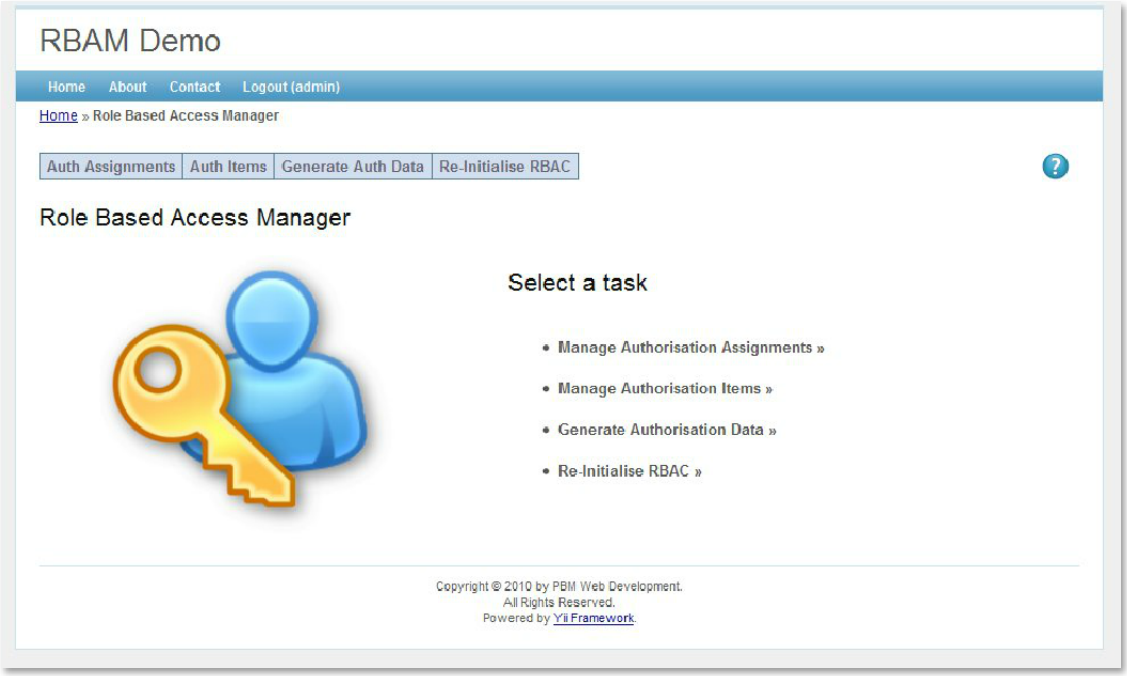

Figure 2 - RBAM Home Page

> **Note:** The menu options and tasks available depend on what permissions you have and whether RBAM is in development mode.

Menu

RBAM can render a menu (showMenu==TRUE) or you can integrate the RBAM menu into your application's menu using the module's `getMenuItem()`or `getMenuItems()` methods. The following assumes that RBAM is rendering the menu; the details of the menu items are the same for both cases.

At the top of each page is RBAM's main menu.

Figure 3 - RBAM Menu

The menu has up to four items:

- Auth Assignments – only visible if you have Auth Assignments Manager permission
- Auth Items – only visible if you have Auth Items Manager permission
  - Create Role
  - Create Task
  - Create Operation
- Generate Auth Data – only visible if you have Auth items Manager permission and RBAM is in development mode
- Re-Initialise RBAC – only visible if you have RBAC Manager permission and RBAM is in development mode

Button Icons

These are the icons used on buttons in RBAM

Icon	Meaning	Description
	Help	At the top right of each page in RBAM. Show help for the page
	Add Authorisation Item	Add a new Authorisation Item to the RBAC system
	Manage Authorisation Item	Update the item and/or its relationships
	Delete Authorisation Item	Permanently remove the item from the RBAC system
	Add Authorisation Assignment	Add a new Authorisation Assignment to the RBAC system, granting the user permissions of the role subject to the assignment's business rule and data
	Update Authorisation Assignment	Edit the business rule and/or data for the assignment
	Revoke Authorisation Assignment	Remove the assignment from the RBAC system, removing the permissions of the role from the user
	View Role	Show users with the role assigned
	View user	Show the roles that the user has assigned

Figure 4 - RBAM's Button Icons

Integrated Help

RBAM has help for every page (except the initialisation page). The "Help" icon is at the top right of each page; click to open the help for that page.

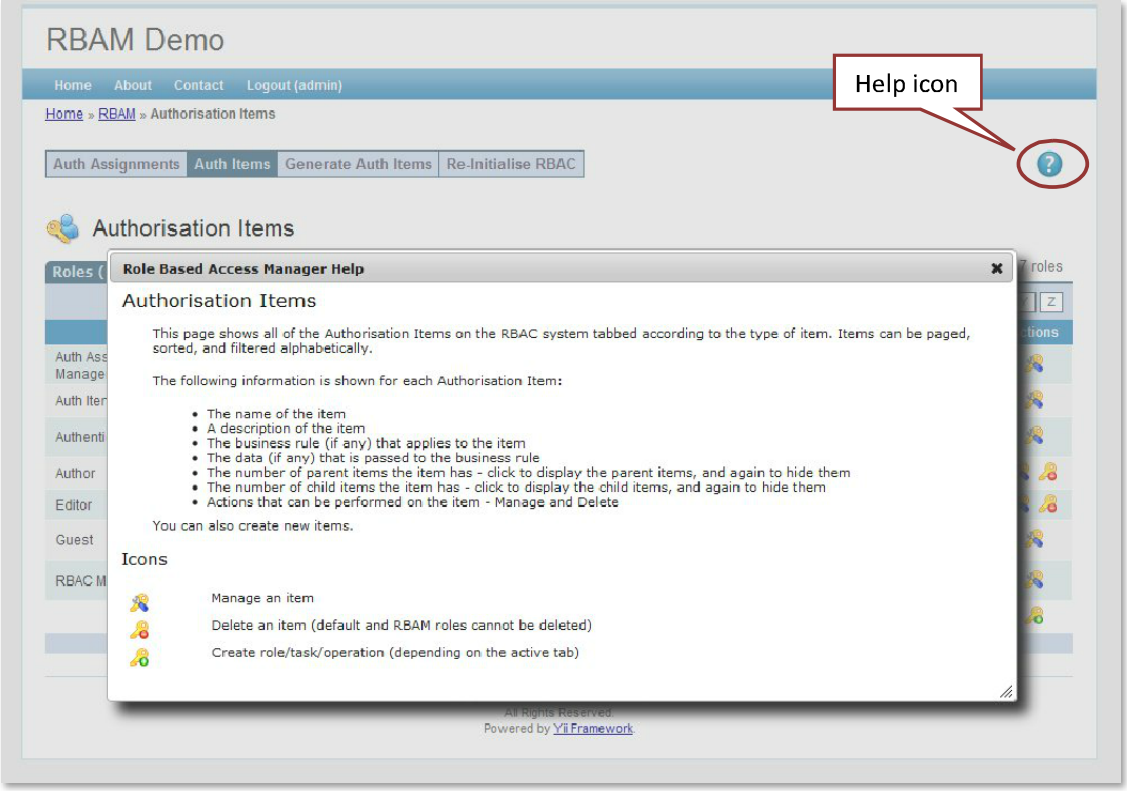

Figure 5 - Page Help

Authorisation Items

There are three views that allow you to manage authorisation items:

- Authorisation Items Overview
- Manage Authorisation Item
- Create Authorisation Item

Authorisation Items Overview

This lists all the authorisation items in the system by type – Role, Task, or Operation – on their corresponding tab; items within a tab can be paged, sorted, and filtered alphabetically, allowing you to quickly get to the item(s) you wish to mange.

You can also add new authorisation items by clicking the icon at the bottom of the Actions column.

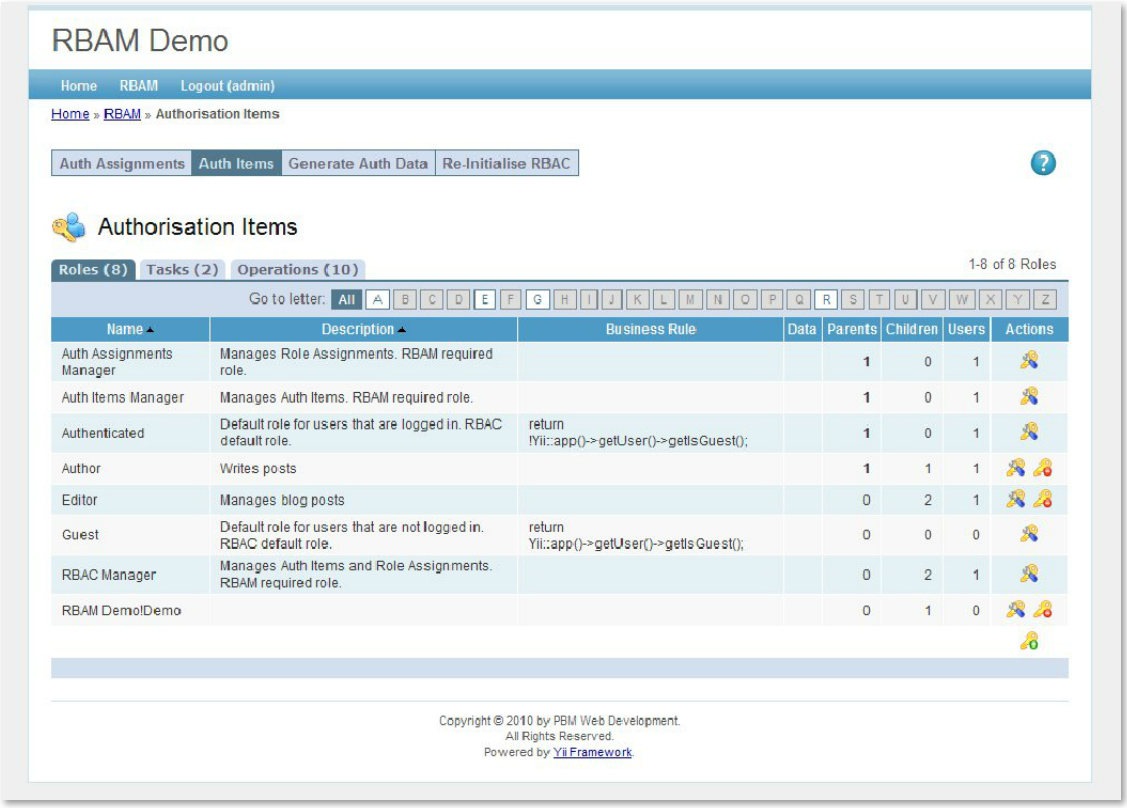

Figure 6 - Authorisation Items Overview

The following information is shown for each item:

- Item name – this uniquely identifies the item.
- Description – a brief description of the item
- Business Rule – the business rule (if any) for the item
- Data – the data (if any) supplied to the business rule
- Parent count – the number of items to which the item is a child. Click to see the parent items, and again to hide them
- Child count – the number of child items the item has. Click to see the child items, and again to hide them
- User count – the number of users with permission for the item
- Actions – buttons to manage and delete the item. In the footer of the Actions column is the link to create a new authorisation item of the current type.

Note: RBAM and CAuthManager default roles (rbacManagerRole, authItemsManagerRole, authAssignmentsManagerRole, authenticatedRole, and guestRole) cannot be deleted.

Manage Authorisation Item

From here you can view and manage all aspects of an authorisation item. When you have made all the required changes click the "***Done***" button to return to the Authorisation Items overview.

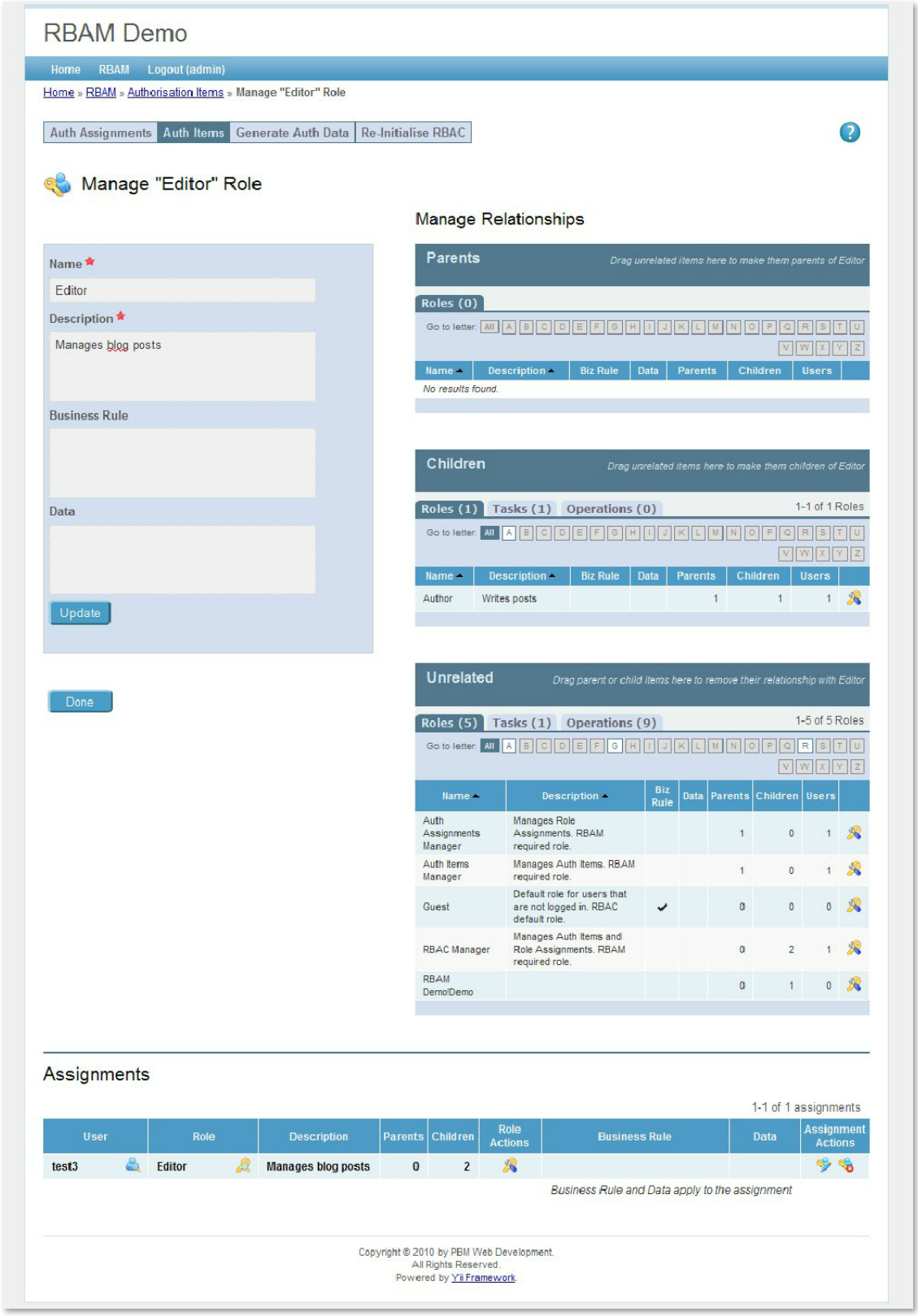

Figure 7 - Manage Authorisation Item

Update an Item

On the left is a form to update the details of the item. Change the details as required (name and description are required) and click the "***Update***" button.

> **Note:** Renaming the default and RBAM roles (rbacManagerRole, authItemsManagerRole, authAssignmentsManagerRole, authenticatedRole, and guestRole) is ***not*** recommended.
> By default the name of these roles is read only. If you wish to change the name of one of these roles, double click the name to make it editable.

> **WARNING: Renaming the RBAM roles will make RBAM unusable until the configuration data is changed to the new name(s).**

Manage Relationships

On the right you can view and manage the relationships – parents and children – of the item. There are three relationship areas:

- Parents – items to which the current item is a child
- Children – items that are children of the current item
- Unrelated – items that are not related to the current item , i.e. not an ancestor or descendant of the current item

Managing relationships is simply a case of dragging and dropping items to or from the *Unrelated* area onto or out of the *Parents* or *Children* areas. The drag handle of an item is its name which will have an orange background while being dragged. Area(s) where the item can be dropped will turn yellow then green when an item is in the area.

The relationship areas are drop targets - not the tabs - and it does not matter which tab is active in the drop area

Each area displays its items in tabs according to type; this means that not all types may be shown for parents and/or children as it depends on the type of the current item; for example, if the current item is a Role, only other Roles can be its parents.

The content of each tab can be paged, sorted, and filtered alphabetically.

The following information is shown for each Authorisation Item:

- Name – name of the item. Use this to drag and drop to manage relationships
- Description – a brief description of the item
- Business Rule – a check mark is shown if the item has a business rule; hover to see the rule
- Data – a check mark is shown if the item has data to be supplied to the business rule; hover to see the data
- Parent count – the number of items to which the item is a child. Click to see the parent items, and again to hide them
- Child count – the number of child items the item has. Click to see the child items, and again to hide them
- User count – the number of users with permission for the item
- Actions – button to manage the item.

Add a Relationship

To add an item as a child or parent of the current item, drag an unrelated item (click and drag on the name) to the *Parents* or *Children* area of the screen; the relationship is created immediately and the relationships and permission areas updated accordingly.

Remove a Relationship

To remove a relationship, drag an item from the *Parents* or *Children* areas to the *Unrelated* area; the relationship is removed immediately and the relationships and permission areas updated.

Note: When adding or removing relationships, Auth Item counts of *Unrelated* items may change for types of items other than that of the item added/removed as a child/parent, and its type count may change by more than one. This is because relations (descendants and ancestors) of the item are taken into account.
For example, if you add *Task T* as a child of *Role R*, *Role R* inherits all the permissions of *Task T* and its descendants ; i.e. *Task T's* descendants become descendants of *Role R* meaning that they are now related to it and so removed from the list of unrelated items.

Assignments

This shows the users that have permission for the current item and the role assignments by which they are granted permission.

The following information is shown for each Authorisation Assignment:

- User – the name of the user with permission for the current item. Clicking the "view" button (*only available if you are assigned the Auth Assignments Manager role*) will display the roles assigned to the user
- Role – the name of the role by which the user is granted permission for the current item. Clicking the "view" button (*only available if you are assigned the Auth Assignments Manager role*) will display the users with the role assigned
- Description – a brief description of the roles
- Parent count – the number of roles to which this role is a child. Click to see the parent items, and again to hide them
- Child count – the number of child roles the role has. Click to see the child items, and again to hide them
- Role Action – a button to manage the role
- Business rule – the business rule (if any) applicable to the user/role assignment
- Data – the data (if any) supplied to the business rule
- Assignment Actions (*only available if you are assigned the Auth Assignments Manager role*) – buttons to update the business rule and/or data for the assignment, and revoke the assignment

Create Authorisation Item

This page allows you to create a new Authorisation Item.

The item name and a description are required, and the name must be unique among all Authorisation Items (RBAM will verify this).

Enter the business rule and data if required and click "***Create***". You will see a confirmation message and be taken to the new item's management page.

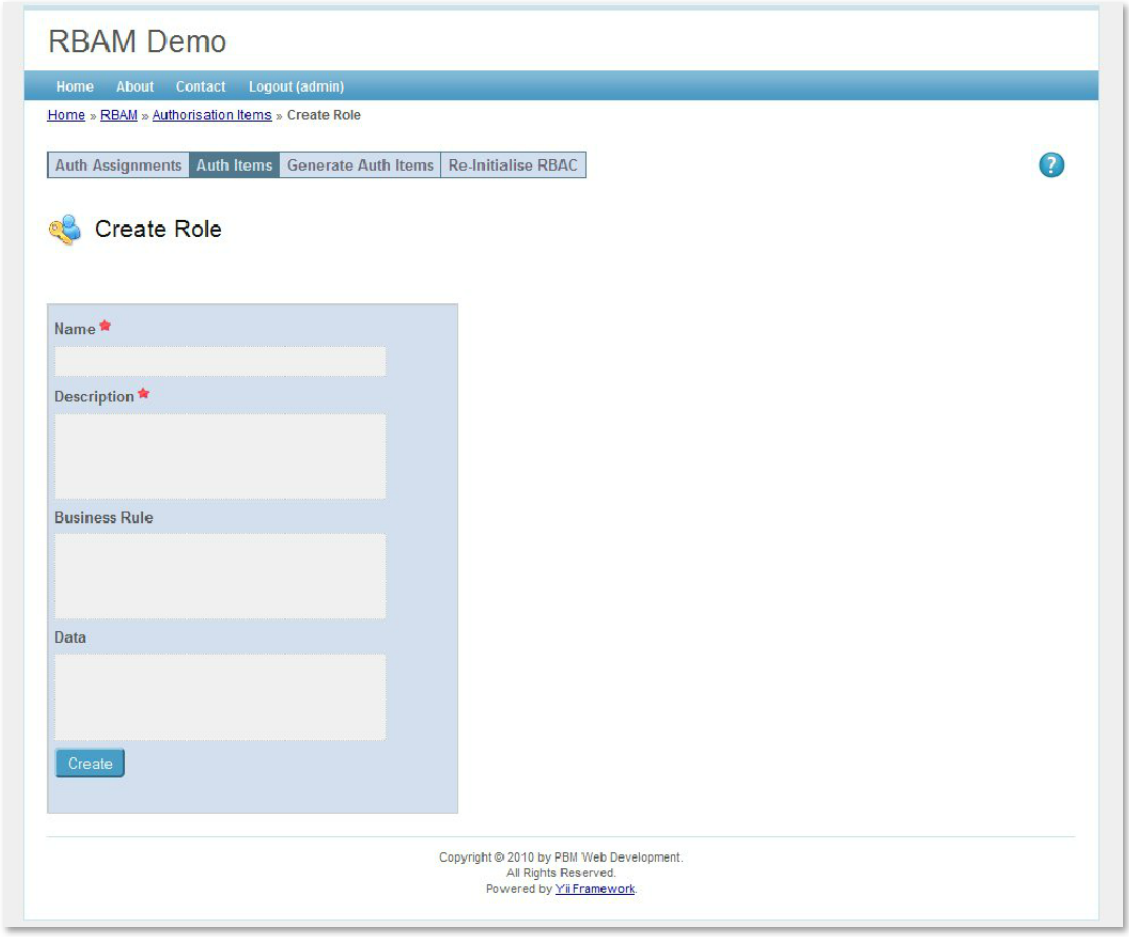

Figure 8 - Create Authorisation Item (Role)

Authorisation Assignments

There are 4 views that allow you to manage authorisation assignments:

- Users
- Assign Roles to a User
- Roles Assigned to a User
- Users with a Role Assigned

Users

Users that can be assigned roles are shown here; the list is all users in the system filtered using the *userCriteria* property. Users can be paged, sorted, and filtered alphabetically.

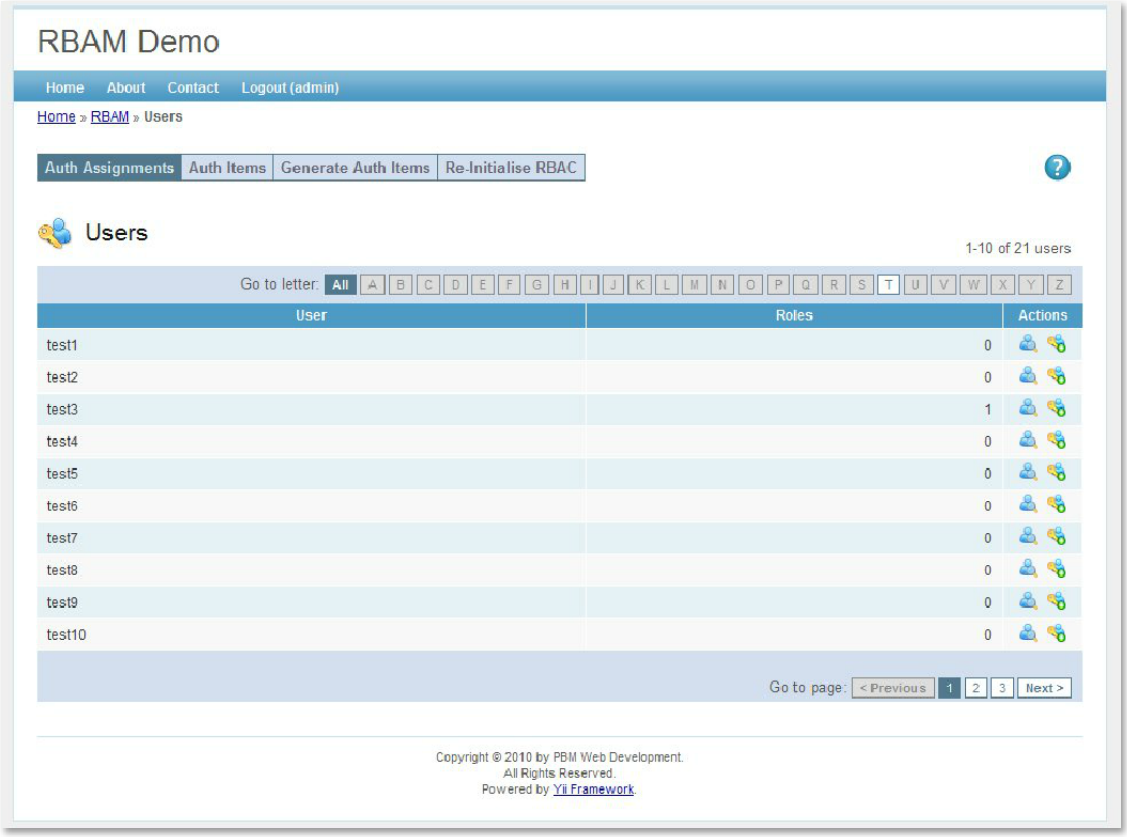

Figure 9 - Users (User "test3" has one role)

The following information is shown for each user:

- Name – the name of the user
- Role count – the number of roles currently assigned to the user
- Actions – buttons view the roles assigned to the user and to assign roles to the user

Assign Roles to a User

Available roles, i.e. roles that are not assigned to and are not children of roles assigned to the user, are shown; they can be paged, sorted, and filtered alphabetically.

Note: CAuthManager::defaultRoles are not shown as they cannot be assigned to a user.

The following information is shown for each role:

- Role – the name of the role.
- Description – a brief description of the role
- Business Rule – the business rule (if any) for the role
- Data – the data (if any) to be supplied to the business rule
- Parent count – the number of rule to which the item is a child. Click to see the parent items, and again to hide them
- Child count – the number of child items the rule has. Click to see the child items, and again to hide them
- Checkbox – click the checkbox to assign the role to the current user

On assigning a role (clicking its checkbox) a pop-up form is displayed where the business rule and data for the assignment are entered if required. Click "***Assign***" to complete the assignment, or "***Cancel***" to cancel it. The list of unassigned roles is updated when a role is assigned.

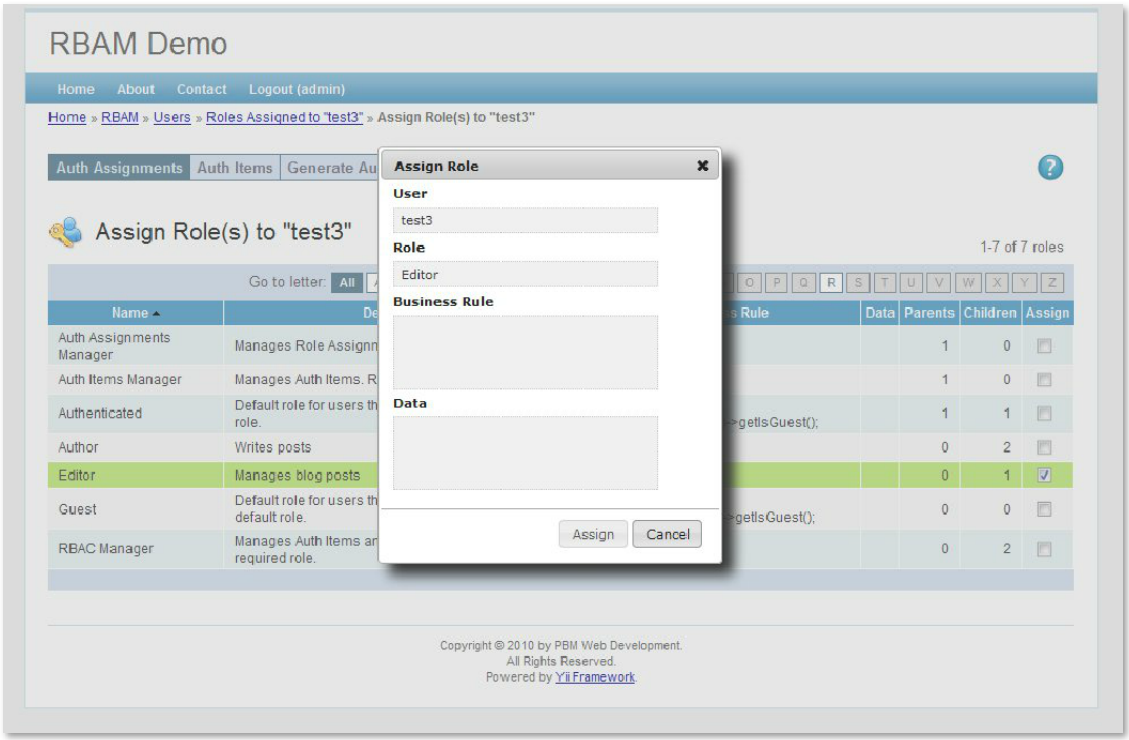

Figure 10 - Role Assignment Dialog (Assigning the "Editor" role to the user "test3")

Note: Roles other than the assigned role will be removed from the list if they are children of the assigned role.

Roles Assigned to a User

This shows the roles assigned to a user. The roles can be paged, sorted, and filtered alphabetically.

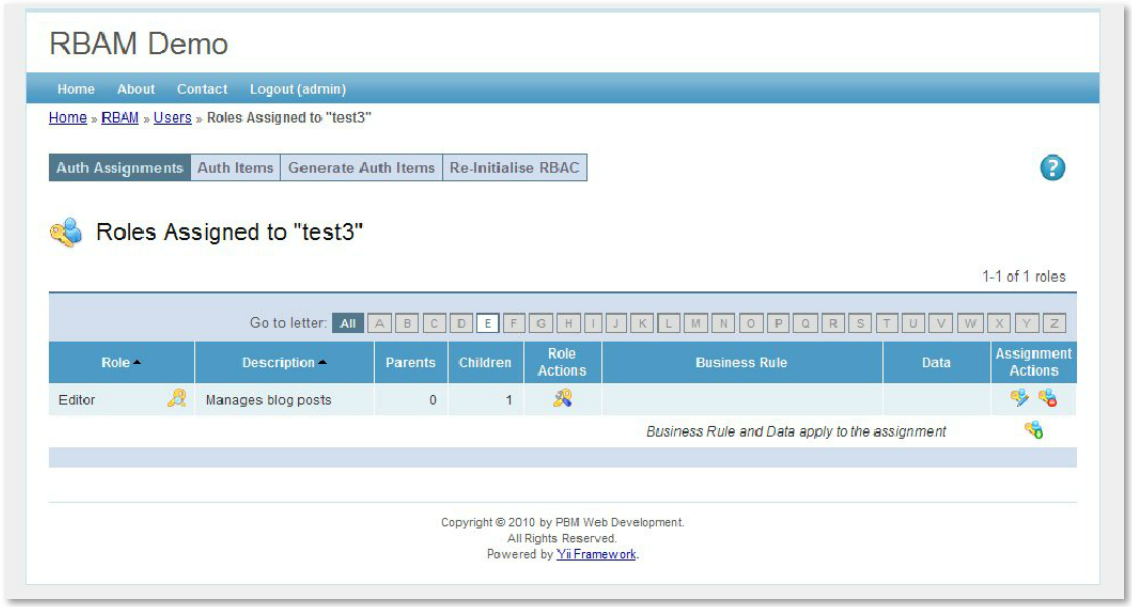

Figure 11 - Roles Assigned to a User

The following information is shown for each role:

- Role – the name of the role. The "view" button shows the users with the role assigned
- Description – a brief description of the role
- Parent count – the number of items to which this role is a child. Click to see the parent items, and again to hide them
- Child count – the number of child items the role has. Click to see the child items, and again to hide them
- Role Actions – button to manage the role (only available if you have Auth Items Manager permission)
- Business rule – the business rule(if any) applicable to the assignment
- Data – the data (if any) supplied to the business rule
- Assignment Actions – buttons to update the business rule and/or data for the assignment, and revoke the assignment. The footer of the Actions column contains the button to assign additional roles to the current user.

Users Assigned to a Role

This shows the users that are directly assigned to the current role and users that inherit the role's permissions and the role(s) by which they do so; direct assignments are in bold.

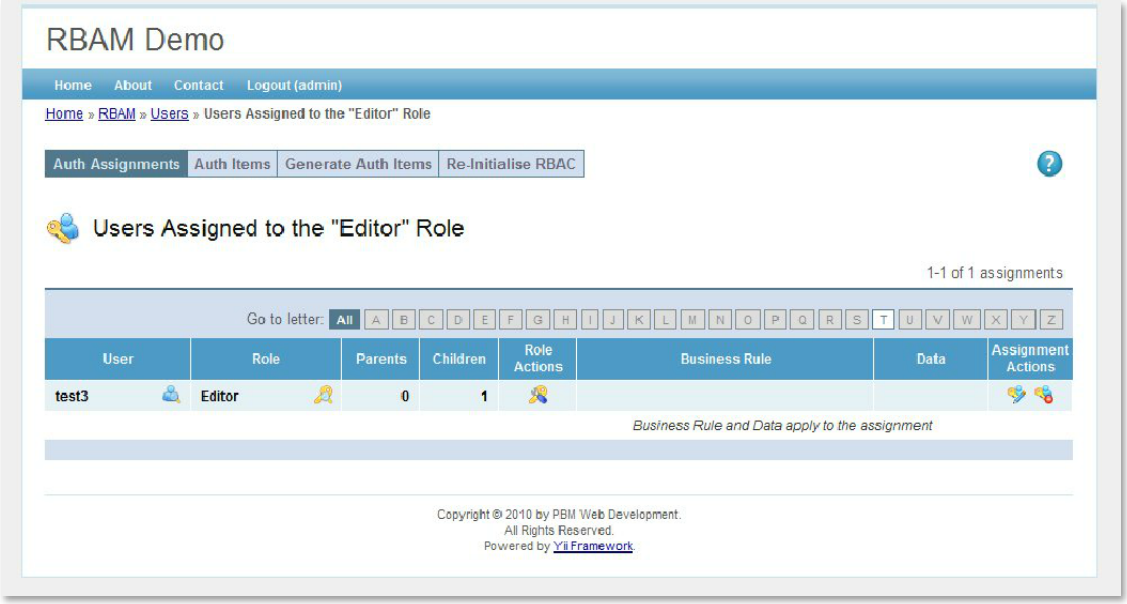

Figure 12 - Users Assigned to a Role

The following information is shown for each Authorisation Assignment:

- User – the name of the user with permission for the current item. Clicking the "view" button will display the roles assigned to the user
- Role – the name of the role by which the user is granted permission for the current item. Clicking the "view" button will display the users with the role assigned
- Description – a brief description of the role
- Parent count – the number of roles to which this role is a child. Click to see the parent items, and again to hide them
- Child count – the number of child roles the role has. Click to see the child items, and again to hide them
- Role Action – a button to manage the role (*only available if you are assigned the Auth Items Manager role*)
- Business rule – the business rule if any) applicable to the user/role assignment
- Data – the data (if any) supplied to the business rule
- Assignment Actions – buttons to update the business rule and/or data for the assignment, and revoke the assignment

Drill-down & Drill-up

Key to understanding the position of an Authorisation Item in the Authorisation Hierarchy, and therefore what permissions an Authorisation Item has and which roles inherit its permissions, are RBAM's "Drill-down" and "Drill-up" features. By clicking on the parent/child count of an

Authorisation Item you can see its parent/child items; drilling up/down from those items allow you to see the authorisation hierarchy.

Drill-down

To see the children of an item, click on the item's child count; the item is highlighted and its children displayed below it. Click on the item's child count again to hide the children.

You can click on the children's child count to see their children, and so on; allowing you to drill down and see what permissions an item has and which Authorisation Items grant them. Each level of drill-down is indented.

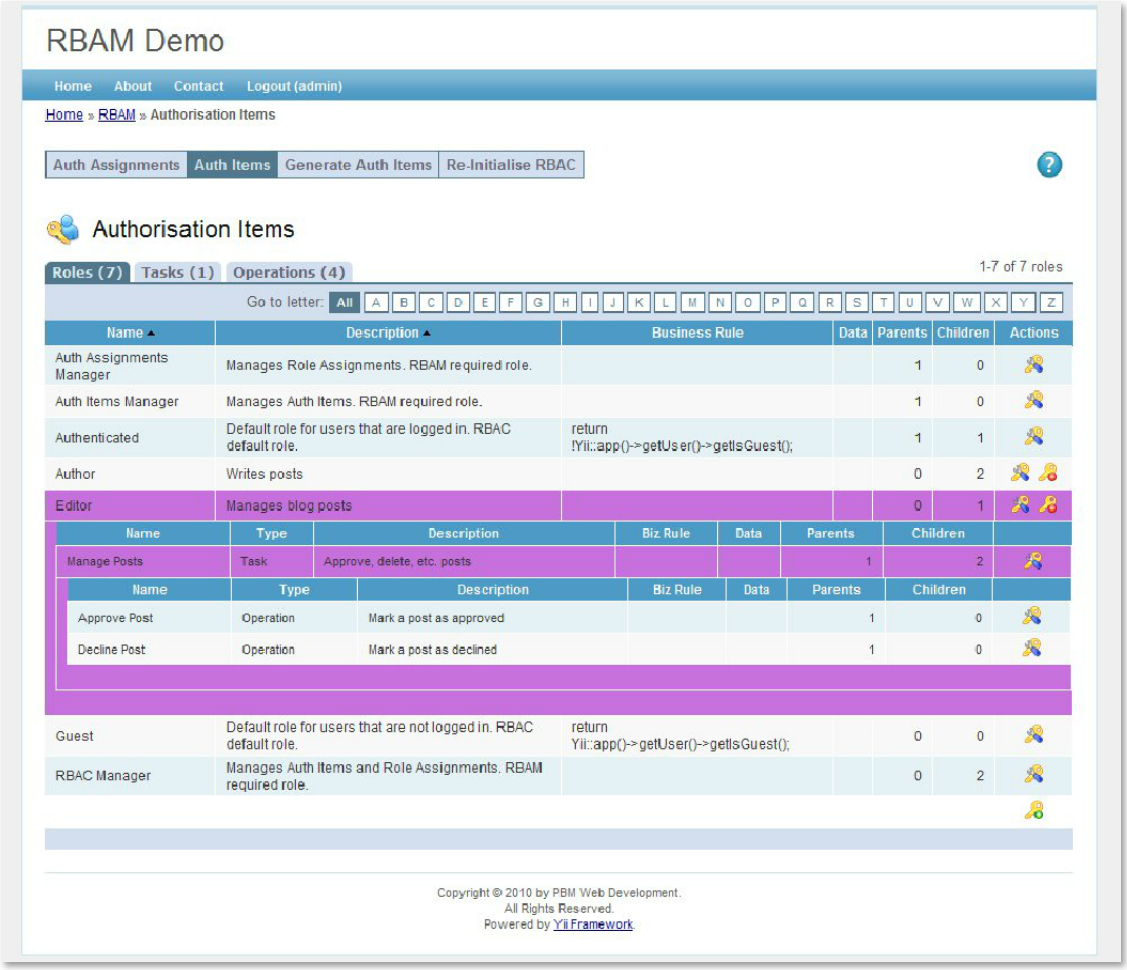

Figure 13 - Drill-Down (showing two levels of drill-down)

Drill-up

To see the parents of an item, click on the item's parent count; the item is highlighted and its parents displayed below it. Click on the item's parent count again to hide the parents.

You can click on the parent's parent count to see their parents, and so on; allowing you to drill up and see which roles inherit the items permissions. Each level of drill-up is indented.

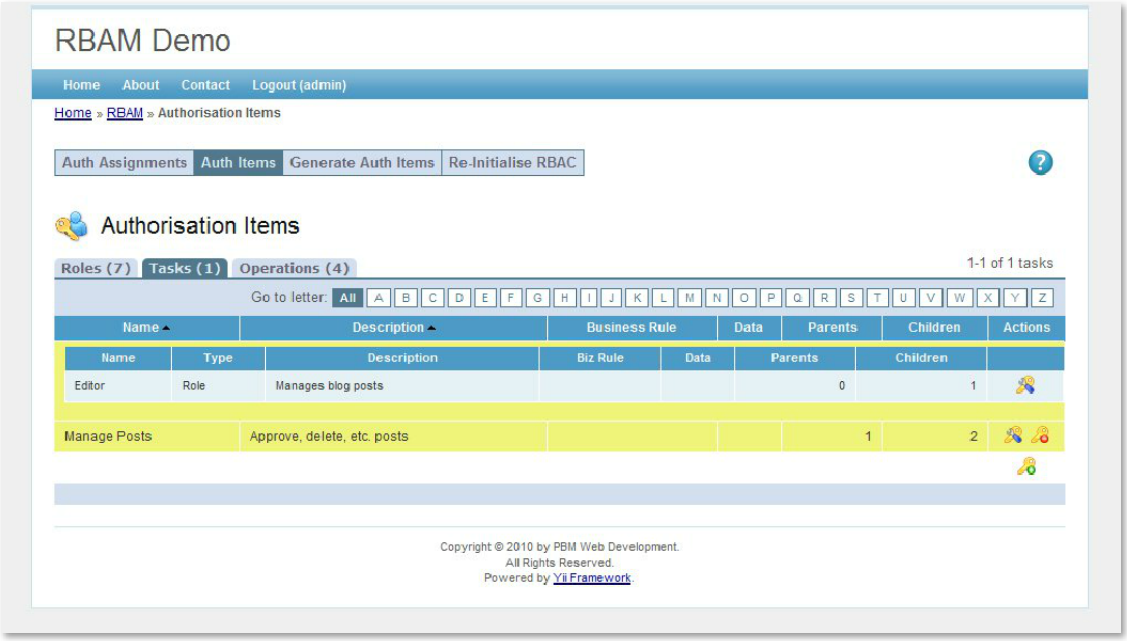

Figure 14 - Drill-Up

The following information is shown for child and parent Authorisation Items:

- Name – name of the item. Use this to drag and drop to manage relationships
- Type – type of item; Role, Task, or Operation
- Description – a brief description of the item
- Business Rule – a check mark is shown if the item has a business rule; hover to see the rule
- Data – a check mark is shown if the item has data to be supplied to the business rule; hover to see the data
- Parent count – the number of items to which the item is a child. Click to see the parent items, and again to hide them
- Child count – the number of child items the item has. Click to see the child items, and again to hide them
- Actions – button to manage the item (*only available if you are assigned the Auth Items Manager role*)

Tip: You can display an item's parents and children at the same time

Initialisation

RBAM can initialise your RBAC system using built-in or user supplied authorisation data. The built-in authorisation data creates the roles needed by RBAM ("rbacManagerRole", "authAssignmentsManagerRole", and "authItemsManagerRole") and the" Guest" and "Authorised" CAuthManager default roles.

Initialisation only needs to be performed once. If you wish to re-initialise an RBAC system that has authorisation data you will be asked to confirm that you wish to do so.

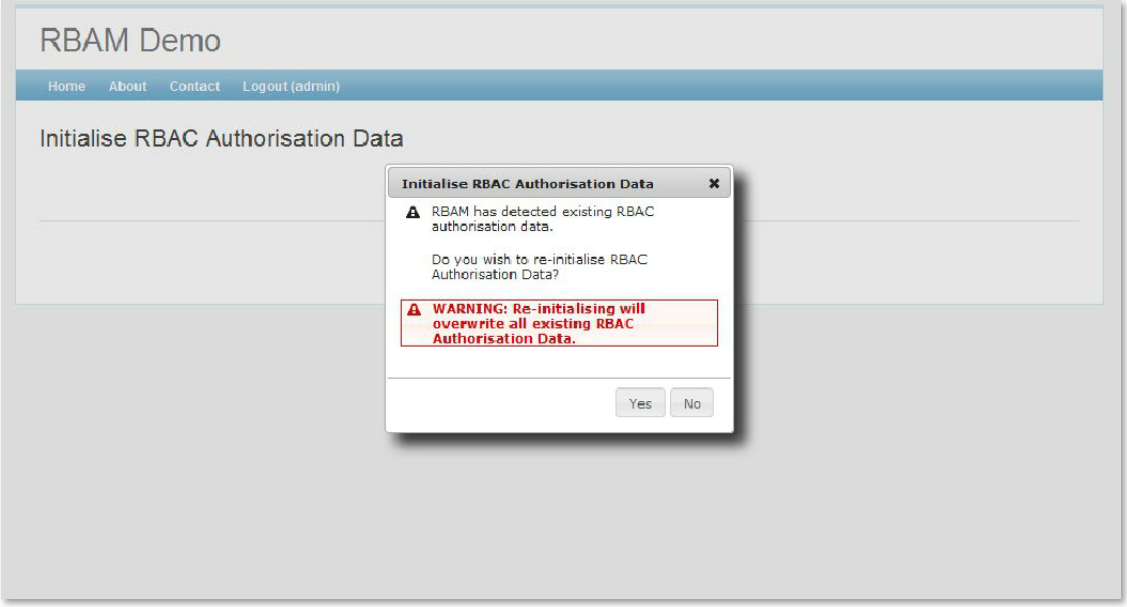

Figure 15 - Confirm Re-initialise RBAC Authorisation Data Dialog

WARNING: Initialising RBAC will clear all existing authorisation data.

If you are supplying authorisation data it can be an array or the path to a PHP file that returns an array. The format of the array is that used by CPhpAuthManager, hence RBAM can import an existing CPhpAuthManager file.

Note: If not specified in the authorisation data, RBAM will create the Authorisation Items that allow it to operate and CAuthManager default roles "Guest" and "Authorised".

Note: If you choose not to initialise RBAC, RBAM will create the Authorisation Items that allow it to operate and CAuthManager default roles "Guest" and "Authorised" if they do not already exist.

Authorisation Data Array Format

```
array(
  'itemName0'=>array(
    'type'=>[CAuthItem::TYPE_OPERATION|TYPE_TASK|TYPE_ROLE],
    'description'=>'Description of item',
    'bizRule'=>null,
    'data'=>null,
    'children'=>array(
      'childName0',
        …
      'childNameN'
    ),
    'assignments'=>array(
      userId0=>array(
        'bizRule'=>null,
        'data'=>null
      ),
        …
      userIdN=>array(…)
    )
  ),
    …
  'itemNameN'=>array(…)
);
```

Figure 16 - Authorisation Data Array Format

Tip: The array format is that used by CPhpAuthManager.

Note: ***bizrule*** and ***data*** must be present for auth items and assignments (if specified), even if null.

Tip: Auth Item ***children*** and ***assignments*** are optional.

Built-in Authorisation Data

```
array(
   $this->rbacManagerRole=>array(
      'type'=>CAuthItem::TYPE_ROLE,
     'description'=>Yii::t('rbac','Manages Auth Items and Role
Assignments. RBAM required role.'),
        'bizRule'=>null,
        'data'=>null,
        'children'=>array(
          $this->authItemsManagerRole,
          $this->authAssignmentsManagerRole
        ),
     ),
     $this->authItemsManagerRole=>array(
        'type'=>CAuthItem::TYPE_ROLE,
        'description'=> Yii::t('rbac','Manages Auth Items. RBAM required
role.'),
        'bizRule'=>null,
        'data'=>null,
     ),
     $this->authAssignmentsManagerRole=>array(
        'type'=>CAuthItem::TYPE_ROLE,
        'description'=> Yii::t('rbac','Manages Role Assignments. RBAM
required role.'),
        'bizRule'=>null,
        'data'=>null,
     ),
     $this->authenticatedRole=>array(
        'type'=>CAuthItem::TYPE_ROLE,
        'description'=> Yii::t('rbac','Default role for users that are
logged in. RBAC default role.'),
        'bizRule'=>'return !Yii::app()->getUser()->getIsGuest();',
        'data'=>null,
     ),
     $this->guestRole=>array(
        'type'=>CAuthItem::TYPE_ROLE,
        'description'=> Yii::t('rbac','Default role for users that are
not logged in. RBAC default role.'),
        'bizRule'=>'return Yii::app()->getUser()->getIsGuest();',
        'data'=>null,
     ),
   );
```

Figure 17 - Built-in Authorisation Data

Tip: If user supplied data does not specify any of the default roles they will be added automatically.

Tip: If user supplied data does not assign a user to the RBAC Manager role the current user will be assigned.

Generate Authorisation Data

RBAM can analyse your application and generate authorisation data – authorisation items and hierarchy - based on the modules, controllers, and actions you select.

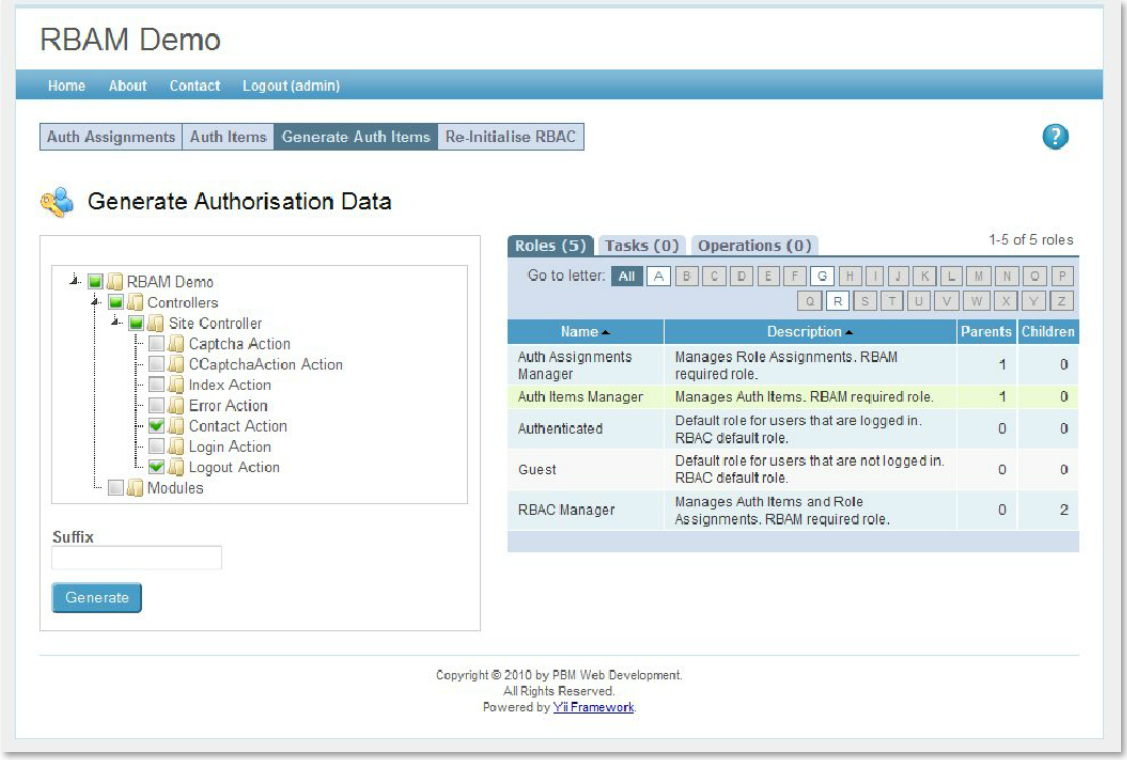

Figure 18 - Generate Authorisation Data

RBAM lists your application's modules, controllers, and their actions in a tree where you can select/deselect individual actions, controllers, and/or modules. All child nodes of a selected node are also selected.

RBAM also shows the current authorisation items to help you decide which, if any, new items need generating.

RBAM creates roles from module ids, tasks from controller ids, and operations from action ids. Items are named according to their "path"; e.g. the *Blog* module containing the *Post* controller which has the create *Action* will result in a role named "*Blog*", a task named "*Blog:Post*", and an operation named "*Blog:Post:Create*". RBAM also creates the appropriate child relationships; using

the above example the "*Blog:Post:Create*" operation will be a child of the "*Blog:Post*" task, which in turn will be a child of the "*Blog*" role.

If your application is "ended", e.g. has front-end and back-end applications, you should run the generator for each end and supply a suffix to avoid name conflicts; e.g. if the suffix "*Back*" is given the previous examples will be named "*Blog:Post:Create!Back*", "*Blog:Post!Back*", and "*Blog! Back*" respectively (a *suffix is used to allow sorting by name to be effective)*.

CMenu Integration

RBAM has two utility methods that provide menu integration:

1. getMenuItem() - Use this method to include RBAM as an item in a menu structure. Typical use is to include RBAM into the application's backend main menu.

```
$this->widget('zii.widgets.CMenu', array(
  …
  'items'=>array(
    …
    $this->getModule('rbam')->getMenuItem(),
    …
  )
  …
));
```

Figure 19 – RbamModule->getMenuItem() Example

2. getMenuItems() - Use this method to include RBAM's menu items in a menu structure. Typical use is to include RBAM's menu items in a "standalone" menu (the default menu uses this method).

```
$this->widget('zii.widgets.CMenu', array(
  'id'=>'rbam-menu',
  'firstItemCssClass'=>'first',
  'items'=>$this->getModule()->getMenuItems()
));
```

Figure 20 - RbamModule->getMenuItems() Example

Credits

The following have helped make RBAM better and/or its development easier.

Yii	http://www.yiiframework.com/	The best PHP framework
AlphaPager	http://www.yiiframework.com/extension/alphapager/	Excellent extension. Used extensively throughout RBAM
directory-independent extensions	http://www.yiiframework.com/wiki/105/directory-independent-extensions/	Great idea. Should be mandatory/in the core
Rights extension	http://www.yiiframework.com/extension/rights/	For the idea of automatically generating authorisation data from the application and setting the standard for documentation
Metadata extension	http://www.yiiframework.com/extension/metadata/	For ideas on how to analyse the application for generating authorisation data
jsTree extension	http://www.yiiframework.com/extension/jstree	jsTree extension, modified to work with jsTree 1.0rc2
jsTree plugin	http://www.jstree.com/	jsTree jQuery plugin

Requests & Bugs

If you have an idea that you think will improve RBAM, or you find a bug, please post in the RBAM thread on the Yii forum; http://www.yiiframework.com/forum/index.php?/topic/14235-rbam-role-based-access-control-manager/

Appendix A – Yii Demo Application User Model

```
<?php
class User extends CActiveRecord {
   public function tableName() {
      return 'tbl_user';
   }
}
```

Figure 21 – User Model Class for Yii Demo Application

Appendix B – Minimal Schema for the User Table

```
CREATE TABLE tbl_user (
   id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
   username VARCHAR(128) NOT NULL,
);
```

Figure 22 – Minimal Schema for the User Table

Note: This is the minimal schema for the user table required by RBAM. The table will typically have other fields, e.g. for password, email address, etc. (See the schema in the "*/protected/data*" directory of Yii's "*testdrive*" application), and may have other items in the statement if supported by the database, e.g. table type.