# Yii-Rights documentation

## Version 1.2.0

**Christoffer Niska**

**1/11/2011**

This document is a detailed description on Yii-Rights, a module which provides a web interface for the Yii's own authorization layer. It also describes how to use it with your Yii application.

# Table of Contents

# Introduction

Rights is a module for Yii web development framework (http://www.yiiframework.com). With Rights you can easily manage user permissions within your Yii application. This extension utilizes Yii's built-in Database Authorization Manager (CDbAuthManager) to provide an extensive web interface for access control.

Project is hosted on Google Code at: http://code.google.com/p/yii-rights

# Requirements

In order to install Rights your application must have a working database connection and a user model (with an id and a name attribute). The user model is required because the authorization items are assigned directly to users.

If you created your application using the Yii console application (yiic) you do not need to worry about this because your application already has a compatible user model.

# Features

- User interface optimized for usability
- Role, task and operation management
- View displaying each role's assigned tasks and operations
- Assigning authorization items to users
- Sorting of authorization items by dragging and dropping
- Installer for easy and quick set up
- Authorization item generation
- Controller filter for checking access
- Support for business rules (and data)
- Runtime caching to increase performance
- Internationalization (I18N)
- Cross-browser and cross-database compatibility
- Easy to extend

## Installation

1. Download the module at http://code.google.com/p/yii-rights/downloads/list.

***Tip:*** *You can also do a checkout. Instructions at:* http://code.google.com/p/yii-rights/source/checkout.

2. Extract the module to your application's **protected/modules/rights** folder.

***Tip:*** *The **modules** folder might not exist, if so create it.*

3. Add the following lines to your application's configuration:

```
'import'=>array(
   ......
   'application.modules.rights.*',
   'application.modules.rights.components.*',       // Correct paths if necessary.
),
......
'components'=>array(
   ......
   'user'=>array(
      'class'=>'RWebUser',                          // Allows super users access implicitly.
      ......
   ),
   'authManager'=>array(
      'class'=>'RDbAuthManager',                    // Provides support authorization item sorting.
       ......
   ),
   ......
),
'modules'=>array(
   'rights'=>array(
      'install'=>true,                              // Enables the installer.
   ),
),
```

*Do not include the dots. They are simply placeholders for additional content in the configuration file.*

navigate to
http://kamran.dev.travelagancy.com:81/index.php?r=rights
to install Rights...

```
navigate to
http://kamran.dev.travelagancy.com:81/index.php?r=rights
to install Rights...
```



## Installer

You can easily install Rights using its installer component. To enable the installer set the **install**-configuration to **true**. The installer will create the required database and insert necessary authorization items and assignments. In case the module is already installed you will be asked to confirm to overwrite the current data. Please note that all existing data will be lost when you reinstall.

**Tip:** If you run into problems with the installer make sure that your database user has write privileges.

## Configuration `// we do not need configuration file at all`

Here is a list of all available configuration values:

```
'rights'=>array(
   'superuserName'=>'Admin',                     // Name of the role with super user privileges.
   'authenticatedName'=>'Authenticated',         // Name of the authenticated user role.
   'userIdColumn'=>'id',                         // Name of the user id column in the database.
   'userNameColumn'=>'username',                 // Name of the user name column in the database.
   'enableBizRule'=>true,                        // Whether to enable authorization item business rules.
   'enableBizRuleData'=>false,                   // Whether to enable data for business rules.
   'displayDescription'=>true,                   // Whether to use item description instead of name.
   'flashSuccessKey'=>'RightsSuccess',           // Key to use for setting success flash messages.
   'flashErrorKey'=>'RightsError',               // Key to use for setting error flash messages.
   'install'=>true,                              // Whether to install rights.
   'baseUrl'=>'/rights',                         // Base URL for Rights. Change if module is nested.
   'layout'=>'rights.views.layouts.main',        // Layout to use for displaying Rights.
   'appLayout'=>'application.views.layouts.main',// Application layout.
   'cssFile'=>'rights.css',                      // Style sheet file to use for Rights.
   'install'=>false,                             // Whether to enable installer.
   'debug'=>false,                               // Whether to enable debug mode.
),
```

*Values used in this example are the default values.*

**Tip:** If your configurations do not differ from the defaults you do NOT need to set them. Usually you do not need to configure Rights at all.

***Note:*** *In case your User model is not called 'User' or if you user name column is not called 'username' or if your user id column is not called 'id' you need to override the respective setting in the module configuration.*

## Variables explained

| Variable | Explanation |
|---|---|
| superuserName | *Name of the role with all privileges. When access is checked for a user that has been assigned this role true is always returned.* |
| authenticatedName | *Name of the role assigned to authenticated users. This can also be null if this role is not needed.* |
| userClass | *Name of the user model class.* |
| userIdColumn | *Name of the id column in the user table.* |
| userNameColumn | *Name of the column in the user table which should be used for displaying the user's name.* |
| enableBizRule | *Whether to enable business rules.* |
| enableBizRuleData | *Whether to enable data for business rules. To enable this business rules needs to be enabled as well.* |
| displayDescription | *Whether to display the description as the authorization item name instead of the name.* |
| flashSuccessKey | *Key to use for displaying success flash messages. Change this if you wish that the Rights success flash message appear in your own flash message region.* |
| flashErrorKey | *Key to use for displaying error flash messages. Change this if you wish that the Rights flash message appear in your own flash message region.* |
| install | *Whether to enable the installer. This should only be enabled when Rights is not installed or you wish to reinstall the module.* |
| baseUrl | *Base URL to Rights. This only needs to be set if Rights is nested within another module.* |
| layout | *Path to the layout to use for displaying Rights.* |
| appLayout | *Path to the application layout.* |
| cssFile | *Name of the cascading style sheet file to use for applying styles to Rights.* |
| debug | *Whether to enable debugging mode. This is used for developing purposes only. Please note that the module performance is decreased while debugging mode is enabled.* |

## Before you start

If you aren't familiar with Yii's Authorization Manager please read the **Role Based Access Control**-section in Yii's guide to Authorization and Authentication at:

http://www.yiiframework.com/doc/guide/topics.auth

Reading this guide might help you understand things like what difference is between tasks and operations, how the authorization hierarchy is built and when to use business rules (and data).

## Using Rights

Once Rights is installed you can access by going to:

```
path/to/application/index.php?r=rights
```

**Tip:** If the main layout is not displaying for Rights go to your **column1.php** (and **column2.php**) in the applications views/layouts folder and make sure that the view path given to beginContent-method is a full path e.g. '**application.views.layouts.main**' instead of only the name of the layout '**main'**.

### Assignments

Under Assignments you can view, assign and revoke each user's permissions.

### Permissions

Under Permissions you can easily assign operations and tasks to your roles and revoke operations and tasks from your roles. In addition you can see which permissions each role has and if they are directly assigned or inherited. If they are inherited you can also see from where they are inherited.

### Operations, Tasks and Roles

Under these you can easily manage each type of authorization item separately. Items can be viewed, reordered, updated and deleted. In addition you can also see how many children each item has. Authorization items can be reordered by dragging and dropping.

### Generator

You can easily generate authorization items for all your controllers and controller actions (even within your modules) using the authorization item generator. It may save you some valuable time as you do not have to create every single authorization items for the ACAC filter (see below) by hand.

# Additional features

## Controller filter for checking access automatically

Rights allows for automatic controller access control by implementing an own filter. In order to use this filter you need to extend your application base controller (normally called Controller) from the RController.

When you do this you can add **rights** to your controller's filters. In addition you may also define which actions should always be allowed by implementing the allowedActions-method.

***Note:*** *Allowed actions are* ***case-insensitive*** *and you may use asterix (*) to allow access to all actions.*

```
public function filters()
{
   return array(
      'rights',
   );
}

public function allowedActions()
{
   return 'index, suggestedTags';
}
```

*Example implementation*

When a request is made the filter will check if the logged in user has the permission required to access requested controller action. Authorization items can be created for the whole controller or for a specific controller action. If an action does not have an authorization item nor is it listed in the allowed actions it will be denied from everyone except the superusers.

The naming policy for the controller authorization items is the following:

```
(Module.)ControllerName.ActionName          // Specific action in the controller.
(Module.)ControllerName.*                   // All actions in the controller.
```

***Note:*** *Authorization items for the controller filter are* ***case-sensitive****!*

## Integration

Rights provide a couple of methods that allow for easy integration with your application. These methods can be found within the Rights class under the components directory.

```
/**
 * Assigns an authorization item to a specific user.
 * @param string the name of the item to assign.
 * @param integer the user id of the user for which to assign the item.
 * @param string business rule associated with the item. This is a piece of
 * PHP code that will be executed when {@link checkAccess} is called for the item.
 * @param mixed additional data associated with the item.
 * @return CAuthItem the authorization item.
 */
public function assign($itemName, $userId, $bizRule=null, $data=null)
{
   ......
}

/**
 * Revokes an authorization item from a specific user.
 * @param string the name of the item to revoke.
 * @param integer the user id of the user for which to revoke the item.
 * @return boolean whether the item was revoked.
 */
public function revoke($itemName, $userId)
{
   ......
}


/**
 * Returns the roles assigned to a specific user.
 * If no user id is provided the logged in user will be used.
 * @param integer the user id of the user for which roles to get.
 * @param boolean whether to sort the items by their weights.
 * @return array the roles.
 */
public function getAssignedRoles($userId, $sort=true)
{
   ......
}


/**
 * Returns the authorization item select options.
 * @param mixed the item type (0: operation, 1: task, 2: role). Defaults to null,
 * meaning returning all items regardless of their type.
 * @param array the items to be excluded.
 * @return array the select options.
 */
public function getAuthItemSelectOptions($type=null, $exclude=array())
{
   ......
}
```

*Methods that can be used when integrating rights into your application.*

## Internationalization

Rights support internationalization. The module will automatically use the same language as your application if a translation is available. To set your application's language to e.g. Brazilian Portuguese add the following line to your application's configuration:

```
'language'=>'pt_br' // CountryId(_RegionId).
```

### Translating Rights

If you wish to translate Rights use the template for translations (found under messages/templates). When the translation is ready place it under messages/CountryId(_RegionId) in a file called core.php (and install.php). If you would like to translate the module into Spanish you would place the translation under *messages/es* and if you would also like to translate the module into Brazilian Portuguese you would place that translation under *messages/pt_br*.

***Tip:*** *Contact me if you need help translating the module or wish to share your translation with the community.* cniska@live.com

## File structure

Here is the complete file structure of the module and a short description for what each file is used for:

```
rights
   assets
      css
         core.css                          // Core style sheet that is always include.
         default.css                       // Default style for the module.
      js
         rights.js                         // Right's JavaScript-functionality.
   components
      behaviors
         RAuthItemBehavior.php             // Rights authorization item behavior.
         RUserBehavior.php                 // Rights user model behavior.
      dataproviders
         RAssignmentDataProvider.php       // Data provider for displaying assignments.
         RAuthItemChildDataProvider.php    // Data provider for displaying item children.
         RAuthItemDataProvider.php         // Data provider for displaying authorization items.
         RAuthItemParentDataProvider.php   // Data provider for displaying item parents.
         RPermissionDataProvider.php       // Data provider for displaying permissions.
      RAuthorizer.php                      // Provides the module's core functionality.
      RController.php                      // Rights controller.
      RDbAuthManager.php                   // Rights authorization manager.
      RGenerator.php                       // Rights authorization item generator.
      Rights.php                           // Provides access to Rights from outside the module.
      RightsFilter.php                     // Controller filter for checking access.
      RInstaller.php                       // Rights installer.
      RWebUser.php                         // Rights web user.
   controllers
      AssignmentController.php             // Handles assignment actions.
      AuthItemController.php               // Handles authorization item actions.
      InstallController.php                // Handles install actions.
   data
      schema.sql                           // Database schema. (Not needed if the installer is used.)
   messages
      templates
         core.php                          // Translation template for the core-category.
         install.php                       // Translation template for the install-category.
   models
      AssignmentForm.php                   // Form model for assigning items to users.
      AuthChildForm.php                    // Form model for adding children to authorization items.
      AuthItemForm.php                     // Form model for creating/updating authorization items.
      GenerateForm.php                     // Form model for generating authorization items.
   views
      assignment
         _form.php                         // Form for assigning authorization items.
         user.php                          // View for displaying a specific user's assignments.
         view.php                          // View for displaying user assignments.
      authItem
         _childForm.php                    // Form for adding authorization item children.
         _form.php                         // Form for creating/updating authorization items.
         _generateItems.php                // Partial view for items that can be generated.
         create.php                        // View for creating authorization items.
         generate.php                      // View for generating authorization items.
         operations.php                    // View for displaying operations.
         permissions.php                   // View for displaying role permission.
         roles.php                         // View for displaying roles.
         tasks.php                         // View for displaying tasks.
         update.php                        // View for updating authorization items.
      install
         confirm.php                       // View for confirming reinstallation.
         ready.php                         // View for the installation ready page.
      layouts
         main.php                          // Default layout for Rights.
      _flash.php                           // Partial view for the flash messages.
      _menu.php                            // Partial view for the menu.
   RightsModule.php                        // Rights module.
```

# Change Log

**Release 1.2.0**

- Assigned items are now divided into three columns (Role, Task and Operation) under Assignments.
- Renamed most of the component classes to be prefixed with 'R' for consistence
- Configuration parameter for whether to display item description instead of name
- Configuration parameter for the application layout.
- Removed the guestName configuration parameter, web user guestName will be used instead
- Changed the forms to not use the form builder for convenience
- Return URLs are now stored with the web user

**Release 1.1.0**

- Optimization by runtime caching authorization items and their children
- Improved the authorization manager and authorizer
- Minor user interface improvements
- Proper support for CSRF validation in authorization item sorting
- Renamed the AuthItemWeight table to Rights

**Release 1.0.0**

- Initial official release

**Release 0.9.11**

- User interface improvements (UI reviewed for usability)
- Description is now mainly used instead of the name
- Even more intensive use of grid view
- Minor generator improvements
- Runtime caching of the module and its components
- Proper support for overriding the module style

**Release 0.9.10**

- Use of grid views and data providers
- An own user behavior
- Proper authorization item sorting according to weights
- Generator now also looks for controllers in subfolders

**Release 0.9.9**

- Improved authorization item generation
- Improved installer
- Improved module configuration
- Rewritten style sheet for easier styling
- French and Hungarian translations

**Release 0.9.8**

- Authorization item generator
- Improved controller  filter
- Installer automation
- Improved support for module nesting

**Release 0.9.7**

- Flash messages
- Module nesting
- Sorting of authorization items
- Improved Installer
- Hover functionality for tables
- German and Italian translations

**Release 0.9.5**

- Support for custom style sheet
- Swedish translation

**Release 0.9.3**

- Rights Installer
- Improved module configuration
- Pagination for Assignments

**Release 0.9.2**

- Internationalization
- Finnish translation

**Release 0.9.1**

- Initial development release

**Pre 0.9.1**

Rights was originally developed as Yii Authorization, but due to a complete rewrite the project name was changed and the module was re-released.

## Glossary

| Term | Description |
|---|---|
| Authenticated name | *Name of the role that is normally assigned to logged in users.* |
| Authorization item | *A permission to do something. Authorization Items can be classified as Operations, Tasks and Roles.* |
| Authorization manager | *A component which manages the Authorization items.* |
| Authorizer | *Rights Authorizer is the core component in the module.* |
| Assignment | *An authorization item assigned to a user.* |
| Business rule | *PHP code that will be executed when performing access checking with respect to the item. Only when the execution returns true, will the user be considered to have the permission represented by the item.* |
| Default roles | *Roles assigned to all users implicitly.* |
| Filter | *Component that is executed before running a controller action.* |
| Flash message | *A message that is displayed only once to the user.* |
| Generator | *Component for generating authorization items for controllers and controller actions.* |
| Guest name | *Name of the role assigned to non-logged in users.* |
| Installer | *Rights Installer is a component that allows for easy installation of the module.* |
| Internationalization | *Support for different languages (and regions).* |
| Operation | *Authorization item that can consist of other Operations.* |
| Permission | *Authorization granted to do something.* |
| Role | *Authorization item that can consist of other Roles, Tasks and Operations.* |
| Superuser | *A user with access to everything.* |
| Superuser name | *Name of the role with superuser privileges.* |
| Task | *Authorization item that can consists of other Tasks and Operations.* |