FREE eBook

# LEARNING phalcon

Free unaffiliated eBook created from
**Stack Overflow contributors.**

#phalcon

# Table of Contents

# About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: phalcon

It is an unofficial and free phalcon ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official phalcon.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# Chapter 1: Getting started with phalcon

## Remarks

Phalcon is an open source, full stack framework for PHP.

Functionality is exposed as PHP classes ready to be used. Written as a C extension, it is optimized for extremely high performance, being the fastest possible framework available for PHP developers.

## Useful links:

| Resource | Link |
|---|---|
| Team | https://phalconphp.com/en/team |
| Documentation | https://docs.phalconphp.com/ |
| Download & Installation instructions | https://phalconphp.com/en/download |
| Forum | https://forum.phalconphp.com/ |
| Blog | https://blog.phalconphp.com/ |
| GitHub | https://github.com/phalcon/cphalcon |
| Roadmap | https://github.com/phalcon/cphalcon/wiki/Roadmap |
| Built with Phalcon | https://builtwith.phalconphp.com/ |

## Versions

| Version | Release Date |
|---|---|
| 2.0.0 | 2014-04-17 |
| 2.0.1 | 2015-05-08 |
| 2.0.2 | 2015-05-25 |
| 2.0.3 | 2015-06-10 |
| 2.0.4 | 2015-07-07 |

| Version | Release Date |
| --- | --- |
| 2.0.5 | 2015-07-14 |
| 2.0.6 | 2015-07-21 |
| 2.0.7 | 2015-08-17 |
| 2.0.8 | 2015-09-25 |
| 2.0.9 | 2015-11-23 |
| 2.0.10 | 2016-02-04 |
| 2.0.11 | 2016-05-04 |
| 2.0.12 | 2016-05-16 |
| 2.0.13 | 2016-05-24 |
| 3.0.0 | 2016-07-29 |
| 3.0.1 | 2016-08-24 |

## Examples

### Installation

Download installation files from Phalcon dedicated download page, as well as finding manuals on making Phalcon work with popular platforms.

# Windows

Put the actual DLL files in a directory proper to extend PHP functionality. For XAMPP use `xampp\php\ext\` - and for WAMP use `wamp\bin\php\php*\ext\` derectory. Then enable Phalcon by adding `extension=php_phalcon.dll` to the appropriate `php.ini` file. Restart the web server and Phalcon should become available.

# Linux platforms

To compile the desired version of Phalcon, first install PHP sources along with some other necessary tools:

```
#Ubuntu
    sudo apt-get install php5-dev php5-mysql gcc libpcre3-dev
```

```
#Fedora
    sudo yum install php-devel php-mysqlnd gcc libtool

#RHEL
    sudo yum install php-devel php-mysql gcc libtool

#Suse
    yast2 -i php5-pear php5-devel php5-mysql gcc

#OS X (Using Homebrew)
    brew tap homebrew/dupes
    brew tap homebrew/versions
    brew tap homebrew/php
    brew install php5x php5x-phalcon # Where "x" - minor number of PHP
```

After they are all properly installed, Phalcon can be compiled:

```
git clone --depth=1 git://github.com/phalcon/cphalcon.git
cd cphalcon/build
sudo ./install
```

(Pick the desired version instead of using just `git://github.com/phalcon/cphalcon.git`) Afterwards the Phalcon extension should be available in the PHP directories. All that's left is to include `extension=phalcon.so` in the desired `php.ini` file. Restart the web server and it should be available.

## Ubuntu users

It is possible to install Phalcon directly from repositories using following commands:

```
sudo apt-add-repository ppa:phalcon/stable
sudo apt-get update
sudo apt-get install php5-phalcon
```

# Mac OS X

## Homebrew

If you have brew installed you first need to tap homebrew-php:

```
brew tap homebrew/homebrew-php
```

After that you need to determine your PHP version. This can be done via the command:

```
php -v
```

The command will output something similar to `PHP 5.6.22` you want the first and second numbers, which are `5` and `6` in this case. Then you run the following command to install the proper version (replacing `5` and `6` with the version you have):

```
brew install php56-phalcon
```

Sources:

- https://docs.phalconphp.com/en/latest/reference/install.html#mac-os-x

Read Getting started with phalcon online: https://riptutorial.com/phalcon/topic/4559/getting-started-with-phalcon

# Chapter 2: Database Management

## Examples

### Using standard SQL directly with models

To use SQL syntax with model, that would transfer result to proper instantions, you should use directly one of Phalcon\Mvc\Model\Resultset classes:

```
$users = new \Application\Models\Users();

// bitwise operation on `flag` field
$sql = 'SELECT * FROM phorum.users WHERE
        (15 & (1 << (flag - 1))) > 0  ORDER BY login DESC';

// as a result you will have a Resultset\Simple with Models\Users instances.
$result = new \Phalcon\Mvc\Model\Resultset\Simple(
    null,

    // what model to use for data returned from SQL
    $users,

    // setting result via "read connection" proper for this model.
    $users->getReadConnection()->query($sql)
);
```

### Database management using Phalcon Model

A model for a new table can be created by running the following commend from the terminal root location:

```
phalcon model <table-name>
```

Let us take the Model Users.

**SELECT**

There are two default functions to do select operation in phalcon, `find()` and `findFirst()`

`findFirst()` is used to get the first row which satisfies the conditions that we are passing. It returns a single object with the data in first row.

Example:

```
$user = Users::findFirst("active=1 AND verified=1 AND email='a@a.com'");
```

This returns the user with the given email and the value of the column verified and active is 1

`find()` is used to get all rows which satisfies the conditions we are passing.

Example:

```
$users = Users::find("active=1 AND verified=1");
```

This returns the users with the value of the column verified and active is 1

**INSERT**

Insert can be done using the following code:

```
$user = new Users();

$user->name = "Arun";
$user->email = "abc@gmail.com";
$user->verified = 1;
$user->active = 1;

$user->save();
```

A new row with the these values will be inserted.

**UPDATE**

Update can be done using the following code:

First we have to select the row we have to update using `findFirst()`

```
$user = Users::findFirst("email='a@a.com'");

$user->verified = 0;
$user->active = 0;

$user->save();
```

This will change the values for the column verified and active for the row with given email.

**DELETE** Delete can also be done using the findFirst()

Example:

```
Users::findFirst("email='a@a.com'")->delete();
```

This will delete the row with given email.

You can also execute custom sql commands with models using the following code:

```
$query = $this->modelsManager->createQuery("SELECT * FROM Users WHERE email='a@a.com'");

$user = $query->execute();
```

## Setting up default connection service

Phalcon uses `db` service by default to obtain connection to databases.

Assuming you have an conguration file with `database` field set up, you can include or autoload following code to obtain connection with database for your project:

```
$di->set('db', function () use ($config) {
    $dbconf = $config->database;
    switch(strtolower($dbconf->adapter)) {

        case 'mysql':
            return new \Phalcon\Db\Adapter\Pdo\Mysql(array(
                'host' => $dbconf->host,
                'username' => $dbconf->username,
                'password' => $dbconf->password,
                // default database to work with
                'dbname' => $dbconf->dbname,
                // default character set
                'charset' => $dbconf->charset,
                // connection warm-up commands for PDO
                'options' => array(
                    PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES "' . $dbconf->charset . '"',
                    PDO::ATTR_CASE => PDO::CASE_LOWER
                )
            ));

        case 'postgresql':
            return new \Phalcon\Db\Adapter\Pdo\Postgresql(array(
                'host' => $dbconf->host,
                'username' => $dbconf->username,
                'password' => $dbconf->password,
                'dbname' => $dbconf->dbname,
                'options' => array(
                )
            ));

        default:
            throw new \Exception('Unimplemented database::adapter in config.ini');
    }
});
```

## Caching Models Meta-Data.

Phalcon builds up some information about tables it is using, so it is possible to validate data being inserted to them without implementing everything by hand. Those are meta data for models. To speed up and prevent Phalcon from building Meta Data every time page is refreshed, it is possible to cache them. To do so, you need to implement `metaData` service for it to use:

```
$di->set('modelsMetadata', function() use ($config)
{
    // assuming that you have a $config var with
    // models.metadata.adapter field declared
    switch (strtolower($config->models->metadata->adapter)) {
        case 'apc':
            $metaData = new MetaDataApcAdapter([
                'lifetime' => $config->models->metadata->lifetime,
                'suffix' => $config->models->metadata->suffix,
            ]);
```

```
            break;
        case 'xcache':
            $metaData = new MetaDataXCacheAdapter([
                'lifetime' => $config->models->metadata->lifetime,
                'prefix' => $config->models->metadata->suffix,
            ]);
            break;
        case 'memory':
            $metaData = new MetaDataMemoryAdapter();
            break;
        default:
            throw new \Exception('Unimplemented models::metadata.adapter in config.ini');
    }

    return $metaData;
});
```

Further documentation available at Phalcons' dedicated page.

Read Database Management online: https://riptutorial.com/phalcon/topic/5294/database-management

# Chapter 3: Events Manager

## Examples

### Dynamic ACL check

Create a Security class to run your ACL logic.

```
<?php

namespace Plugins;

use Phalcon\Events\Event;
use Phalcon\Mvc\Dispatcher;
use Phalcon\Acl;
use Phalcon\Acl\Role;
use Phalcon\Acl\Resource;
use Phalcon\Acl\Adapter\Memory as AclList;

class Security extends \Phalcon\Mvc\User\Plugin
{
    public function beforeExecuteRoute(Event $event, Dispatcher $dispatcher)
    {
        // your acl logic here
    }
}
```

Hook the Security class to the dispatcher, to run on *beforeExecuteRoute*.

```
$di = new \Phalcon\DI\FactoryDefault();
$eventsManager = $di['eventsManager'];

$di->setShared('dispatcher', function() use ($eventsManager) {
    $eventsManager->attach('dispatch:beforeExecuteRoute', new \Plugins\Security);
    $dispatcher = new \Phalcon\Mvc\Dispatcher;
    $dispatcher->setEventsManager($eventsManager);
    return $dispatcher;
});
```

Read Events Manager online: https://riptutorial.com/phalcon/topic/5293/events-manager

# Chapter 4: Filtering and Sanitizing

## Examples

### Convenient in-model sanitizing

Set a convenience method in your base model

```
namespace Base;

class Model extends \Phalcon\Mvc\Model
{
    public function sanitize($attr, $filterName)
    {
        $filter = $this->getDI()->get('filter');
        $this->$attr = $filter->sanitize($this->$attr, $filterName);
    }
}
```

Then use like so

```
class User extends \Base\Model
{
    public function beforeValidation()
    {
        $this->sanitize('id', 'int');
        // input $this->id: 123abc
        // output: 123

        $this->sanitize('email', 'email');
        // input $this->email: youre(-)mail@dom/ain.com
        // output: youremail@domain.com

        $this->sanitize('wage', 'float');
        // input $this->wage: +1234ab.56cd
        // output: 1234.56

        $this->sanitize('name', 'string');
        // input $this->name: <john>
        // output: john
    }
}
```

Read Filtering and Sanitizing online: https://riptutorial.com/phalcon/topic/4917/filtering-and-sanitizing

# Chapter 5: Incubator

## Examples

### Introduction

The Phalcon Incubator can be used by the community to experiment with new features or expand onto the existing Phalcon adapters, prototypes or functionalities.

Anything in the Incubator can be potentially corporated into the framework.

Github repository: https://github.com/phalcon/incubator

### Installation

# Installation via Composer

The easiest way to install the Incubator is by using Composer.

Install Composer and create a new `composer.json` file in the root of your project.

```
|-- app
|-- public
|   `-- index.php
|-- vendor
|-- composer.json
```

Add the following content to the `composer.json` file. If you are still using Phalcon 2.0.x

```
{
    "require": {
        "phalcon/incubator": "^2.0"
    }
}
```

If you are using Phalcon 3.0.0

```
{
    "require": {
        "phalcon/incubator": "~3.0"
    }
}
```

After altering the `composer.json` file, you need to run the following command, from the root of your project.

```
$ php composer.phar install
```

If you already installed your files and you would like to update them instead. Then use `update` instead of `install`.
By default Composer will create a new folder named `vendor` in your project root and download all the requested files into this directory.

After composer has been installed, your document structure should look something like this:

```
|-- app
|-- public
|   `-- index.php
|-- vendor
|   `-- phalcon
|       `-- incubator
|           `-- docs
|           `-- Library
|           `-- tests
|-- composer.json
```

# Installation via Github

Create a folder named `vendor` in your project root directory. And also create the folder `phalcon` inside this folder.

```
|-- app
|-- public
|   `-- index.php
|-- vendor
|   `-- phalcon
```

Now navigate inside the `phalcon` folder and clone the Incubator from the Github repository.

```
git clone https://github.com/phalcon/incubator.git
```

By default, the above command will download the latest version of Phalcon. If you'd like to download an earlier version you can simply add the `--branch` parameter to the command, followed by the required branch version.

```
git clone https://github.com/phalcon/incubator.git --branch 2.0.9
```

# Installation via the manual way

If the above methods are confusing for you and you like to do stuff manually, you can easily download the repository from Github and place the files inside the `vendor/phalcon/`, in your project root.

```
|-- app
|-- public
|   `-- index.php
```

```
|-- vendor
|   `-- phalcon
```

### Usage

# Loading the Incubator into your project

Add the following lines of code to your loader file

```
$loader = new Phalcon\Loader();

$loader->registerNamespaces([
    'Phalcon' => '/path/to/your/vendor/phalcon/incubator/Library/Phalcon/',
    // any other namespaces you have loaded
    // ...
]);

$loader->register();
```

Now you can access all the Incubator functionalities by using the normal Phalcon namespaces:

```
\Phalcon\Acl\Adapter\Database;
```

Read Incubator online: https://riptutorial.com/phalcon/topic/5354/incubator

# Chapter 6: Routing and dispatching

## Examples

### RESTful API Routes for Multi Module Application

```
// Define new router group
$api = new \Phalcon\Mvc\Router\Group([
    'module' => 'api',
]);
$api->setPrefix('/api/v1');

// API routes (Maps to Cotnroller::Action)
$api->addGet('/users', 'Users::index');
$api->addGet('/users/search/{query}', 'Users::search');
$api->addGet('/users/{id:[0-9]+}', 'Users::fetch');
$api->addPost('/users', 'Users::add');
$api->addPut('/users/{id:[0-9]+}', 'Users::edit');
$api->addDelete('/users/{id:[0-9]+}', 'Users::delete');

// Add API routes to main router
$router->mount($api);
```

Example of creating a user:

```
curl -i -X POST -d
    '{"name": "John Snow", "title": "King of the North"}'
    http://example.com/api/v1/users
```

### Dynamically set module routes

```
$router = new \Phalcon\Mvc\Router(false);
$router->removeExtraSlashes(true);
$request = new \Phalcon\Http\Request();
$action = strtolower($request->getMethod()); // get, post, etc.
$modules = ['calendar', 'main', 'user']; // names of the modules you create

// you can define other static routes here

foreach ($modules as $module) {
    // must match what you register with the Loader service
    $namespace = 'App\\' . ucfirst($module) . '\Controllers';

    // make a group to avoid setting namespace and module for every route definition
    $moduleGroup = new \Phalcon\Mvc\Router\Group([
        'namespace' => $namespace,
        'module' => $module
    ]);

    // this will match a route like /calendar/index/save
    $moduleGroup->add("/{$module}/:controller/:action", [
        'controller' => 1,
        'action' => 2
```

```
    ]);

    // setting a prefix will apply it to all routes below
    $moduleGroup->setPrefix('/api');

    // this will match a route like /api/calendar/index/save
    $moduleGroup->add("/{$module}/([a-zA-Z_]+)/:action", [
        'controller' => 1,
        'action' => 2
    ]);

    // this will match a route like /api/calendar/123
    $moduleGroup->add("/{$module}/:int", [
        'moduleId' => 1,
        'controller' => 'index',
        'action' => $action // defined at the top of example
    ]);

    $router->mount($moduleGroup);
}

// you can define other static routes here

return $router;
```

Read Routing and dispatching online: https://riptutorial.com/phalcon/topic/5035/routing-and-dispatching

# Chapter 7: Validation

## Remarks

- API reference to the validation class can be found here: https://docs.phalconphp.com/en/latest/api/Phalcon_Validation.html
- If there is entity provided in `\Phalcon\Validation` you don't need to pass model key in `\Phalcon\Validation\Validator\Uniqueness`

## Examples

### Built in Validators

**PresenceOf** - Validates that a value is not null or empty string

```
$validator->add('name', new \Phalcon\Validation\Validator\PresenceOf([
   'message' => 'The name is required'
]));
```

**Email** - Checks if a value has a correct e-mail format

```
$validator->add('email', new \Phalcon\Validation\Validator\Email([
   'message' => 'The e-mail is not valid'
]));
```

**Identical** - Checks if a value is identical to other

```
$validator->add('terms', new \Phalcon\Validation\Validator\Identical([
   'accepted' => 'yes',
   'message' => 'Terms and conditions must be accepted'
]));
```

**Url** - Checks if a value has a url format

```
$validator->add('url', new \Phalcon\Validation\Validator\Url([
   'message' => ':field must be a url'
]));
```

**Confirmation** - Checks that two values have the same value

```
$validator->add('password', new \Phalcon\Validation\Validator\Confirmation([
   'message' => 'Password doesn\'t match confirmation',
   'with' => 'confirmPassword'
]));
```

**StringLength** - Validates that a string has the specified maximum and minimum constraints The test is passed if for a string's length L, min<=L<=max, i.e. L must be at least min, and at most max.

```
$validation->add('name_last', new \Phalcon\Validation\Validator\StringLength([
      'max' => 50,
      'min' => 2,
      'messageMaximum' => 'We don\'t like really long names',
      'messageMinimum' => 'We want more than just their initials'
]));
```

**Regex** - Allows validate if the value of a field matches a regular expression

```
$validator->add('created_at', new \Phalcon\Validation\Validator\Regex([
   'pattern' => '/^[0-9]{4}[-\/](0[1-9]|1[12])[-\/](0[1-9]|[12][0-9]|3[01])$/',
   'message' => 'The creation date is invalid'
]));
```

**CreditCard** - Checks if a value has a valid creditcard number

```
$validator->add('creditcard', new \Phalcon\Validation\Validator\CreditCard([
   'message' => 'The credit card number is not valid'
]));
```

**Between** - Validates that a value is between an inclusive range of two values. For a value x, the test is passed if minimum<=x<=maximum.

```
$validator->add('name', new \Phalcon\Validation\Validator\Between([
   'minimum' => 0,
   'maximum' => 100,
   'message' => 'The price must be between 0 and 100'
]));
```

**ExclusionIn** - Check if a value is not included into a list of values

```
$validator->add('status', new \Phalcon\Validation\Validator\ExclusionIn([
   'message' => 'The status must not be A or B',
   'domain' => ['A', 'B']
]));
```

**InclusionIn** - Check if a value is included into a list of values

```
$validator->add('status', new \Phalcon\Validation\Validator\InclusionIn([
   'message' => 'The status must be A or B',
   'domain' => ['A', 'B']
]));
```

**Uniqueness** - Check if a value is uniqueness

```
$validator->add('login', new \Phalcon\Validation\Validator\Uniqueness([
    'message' => 'The login must be unique',
    'model' => new Users()
]));
```

## Google reCaptcha custom validation component

### The class

```
use Phalcon\Validation\Validator;
use Phalcon\Validation\ValidatorInterface;
use Phalcon\Validation\Message;

class RecaptchaValidator extends Validator implements ValidatorInterface
{
    public function validate(\Phalcon\Validation $validation, $attribute)
    {
        $value = $validation->getValue('g-recaptcha-response');
        $ip = $validation->request->getClientAddress();
        if (!$this->verify($value, $ip)) {
            $validation->appendMessage(new Message($this->getOption('message'), $attribute,
'Recaptcha'));
            return false;
        }
        return true;
    }

    protected function verify($value, $ip)
    {
        $params = [
            'secret' => 'YOUR_RECAPTCHA_SECRET_KEY',
            'response' => $value,
            'remoteip' => $ip
        ];
        $response =
json_decode(file_get_contents('https://www.google.com/recaptcha/api/siteverify?' .
http_build_query($params)));
        return (bool) $response->success;
    }
}
```

### Example usage in a Phalcon form:

```
$reCaptchaField->addValidator(new \RecaptchaValidator([
    'message' => 'Your reCaptcha error message'
]));
```

Read Validation online: https://riptutorial.com/phalcon/topic/4722/validation

# Chapter 8: Working with ACL

## Syntax

- You can use '*' as second and third parameter in `Phalcon\Acl::allow` and `Phalcon\Acl::deny` methods. This will mean any resource and action respectively.
- Second argument in `Phalcon\Acl::addRole` tells from which role inheritance access.

## Remarks

- You should serialize your ACL to some file or cache backend instead of creating it on each request.
- Also it's good idea to keep acl in seperated file.
- `Phalcon\Acl` is able to send events to event manager, there are two events - `beforeCheckAccess` and `afterCheckAccess`.
- You can use `Phalcon\Acl\AdapterInterface` to implement your own acl adapter.
- You can protect your routes using acl with combination of proper listener in dispatcher

## Examples

### Creating an ACL

You can create ACL by using `Phalcon\Acl\Adapter\Memory` class:

```
$acl = new Phalcon\Acl\Adapter\Memory();
```

By default phalcon allows action to resource which has not been defined, to change this you can use:

```
$acl->setDefaultAction(Phalcon\Acl::DENY);
```

Roles can be added in two ways - using `Phalcon\Acl\Role` or just plain string:

```
$roleAdministrator = new Phalcon\Acl\Role('Administrator');
$acl->addRole($roleAdministrator);
$acl->addRole('Customer');
```

Resources can be added in two ways too, you can add actions as single action or as array:

```
$resourceCategories = new Phalcon\Acl\Resource('categories');
$acl->addResource($resourceCategories, 'create');
$acl->addResource('products', ['create', 'update']);
```

### Defining Access Control and querying an ACL

You can allow role to access some action on resource by:

```
$acl->allow('Administrator', 'products', 'create');
```

You can deny role to access some action on resource by:

```
$acl->deny('Customer', 'categories', 'create');
```

You can check if role is allowed to some action on resource by using:

```
$acl->isAllowed('Administrator', 'products', 'create');
```

## Additional condition in ACL

You can add also add some more logic which has to be checked to your ACL using anonymous functions. They will be executed when using `Phalcon\Acl\Adapter\Memory::allow()` or `Phalcon\Acl\Adapter\Memory::deny()`, if they will return true, they role will be allowed to access certain action on resource.

```
$acl->allow('Customer', 'products', 'create', function($parameter) {
    return $parameter % 2 == 0;
});
$acl->isAllowed('Customer', 'products', 'create', ['parameter' => 1]); // this will return
false
$acl->isAllowed('Customer', 'products', 'create', ['parameter' => 2]); // this will return
true
```

Notice how parameters are passed to function. Your key in array needs to have the same name as in function. Also default parameters parameters can be passed, as well as objects.

## Objects as roles and resources

By implementing `Phalcon\Acl\RoleAware` or `Phalcon\Acl\ResourceAware` you can use them as objects in `Phalcon\Acl\Adapter\Memory::isAllowed()`.

```
// Create our class which will be used as roleName
class UserRole implements Phalcon\Acl\RoleAware
{
    protected $id;
    protected $roleName;

    public function __construct($id, $roleName)
    {
        $this->id = $id;
        $this->roleName = $roleName;
    }

    public function getId()
    {
        return $this->id;
    }
```

```
    // Implemented function from RoleAware Interface
    public function getRoleName()
    {
        return $this->roleName;
    }
}
```

```
// Create our class which will be used as resourceName
class ModelResource implements Phalcon\Acl\ResourceAware
{
    protected $id;
    protected $resourceName;
    protected $userId;

    public function __construct($id, $resourceName, $userId)
    {
        $this->id = $id;
        $this->resourceName = $resourceName;
        $this->userId = $userId;
    }

    public function getId()
    {
        return $this->id;
    }

    public function getUserId()
    {
        return $this->userId;
    }

    // Implemented function from ResourceAware Interface
    public function getResourceName()
    {
        return $this->resourceName;
    }
}
```

```
$customer = new ModelResource(1, "products", 2);
$administrator = new UserRole(1, "Administrator");
$acl->isAllowed($administrator, $customer, 'create');
```

Also ability to use objects can be combined with additional condition in acl:

```
$acl->allow('Administrator', 'products', 'update', function(UserRole $user, ModelResource
$model) {
    return $user->getId == $model->getUserId();
});
$product = new ModelResource(1, 'products', 2);
$administrator = new UserRole(1, 'Administrator');
$anotherAdministrator = new UserRole(2, 'Administrator');
$acl->isAllowed($administrator, $product, 'update'); // this will return false
$acl->isAllowed($anotherAdministrator, $product, 'update'); // this will return true
```

Notice that with additional condition and using objects in `isAllowed` method you don't need to pass those objects as arguments. They are passed automatically only if there are correct types before arguments in function. This gives you huge ability to control if certain users can edit for example

certain models in your application and when they can do it.

Read Working with ACL online: https://riptutorial.com/phalcon/topic/5202/working-with-acl

# Credits

| S. No | Chapters | Contributors |
|---|---|---|
| 1 | Getting started with phalcon | 4444, Community, Goke Obasa, Magnie Mozios, Nikolay Mihaylov, Timothy, yergo |
| 2 | Database Management | Arun D Nambissan, yergo |
| 3 | Events Manager | galki |
| 4 | Filtering and Sanitizing | galki, Timothy |
| 5 | Incubator | Timothy |
| 6 | Routing and dispatching | galki, Nikolay Mihaylov, Timothy |
| 7 | Validation | Juri, Nikolay Mihaylov, Timothy |
| 8 | Working with ACL | Juri |