Login - Copied

Customize Authentication On Yii Framework Using MySQL Database

Just managed to find some time to play around with Yii. Yii is powerful but there is still a long way to go if we are talking about documentation for Yii framework. In Yii framework, we can see that it is very different from CodeIgniter where documentation is really structured and well understood. Nonetheless, i still feel that Yii framework is worth to explore. I managed to get my own customized authentication on Yii by adding some secure feature such as hashing, salt, key and etc. So here i am writing this tutorial to share with you more about Yii framework using MySQL database.

Requirement

Since this is more like a follow up tutorial, there are a few requirements before you start reading this tutorial.

1. Installed Yii with a MySQL database.
2. Setup Gii and get the user CRUD completed.

Customize Authentication - Database

Now here comes the tricky part. We need a database that stored our hashed password which is 128 bits since i am using sha512. Our data schema should looks like this,

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

```
    email VARCHAR(128) NOT NULL
);
```

Well, it looks the same as the demo one so just ignore me lol. Create this table and we are ready to do some MVC. If you are following the tutorial you would most likely get a CRUD user setup. But we only have 'admin' and 'demo' login account. We would definitely want something better.

Customize Authentication - Model

In order to validate a user, we need to create a few methods in the model folder in order to authenticate and store user password. We will need to create these functions on our User.php file on our model folder.

```
/**
 * @return boolean validate user
 */
public function validatePassword($password, $username){
    return $this->hashPassword($password, $username) === $this->password;
}
/**
 * @return hashed value
 */
DEFINE('SALT_LENGTH', 10);
public function hashPassword($phrase, $salt = null){
    $key = 'Gf;B&yXL|beJUf-K*PPiU{wf|@9K9j5?d+YW}?VAZOS%e2c -:11ii<}ZM?PO!96';
    if($salt == '')
        $salt = substr(hash('sha512', $key), 0, SALT_LENGTH);
    else
        $salt = substr($salt, 0, SALT_LENGTH);
    return hash('sha512', $salt . $key . $phrase);
}
```

the two methods above is used to validate the user password during login and the other method returns a hashed password given the original plain password value. Once these two methods are pasted into the user.php file. We are done with our modal!

Customize Authentication - Controller

In controller, we need to modify the create and update handler but i will just demonstrate the create user handler. Go to your controller folder and look for UserController.php. Change the method actionCreate to the following

```
/**
 * Creates a new model.
 * If creation is successful, the browser will be redirected to the 'view' page.
 */
public function actionCreate()
{
    $model=new User;

    // Uncomment the following line if AJAX validation is needed
    // $this->performAjaxValidation($model);

    if(isset($_POST['User']))
    {
        $model->attributes=$_POST['User'];
        $model->password = $model->hashPassword($_POST['User']['password'], $_POST['User']['email']);
        if($model->save())
            $this->redirect(array('view','id'=>$model->ID));
        else
            $model->password = $_POST['User']['password'];
    }

    $this->render('create',array(
        'model'=>$model,
    ));
}
```

This way, we can create user with hashed password instead of plain password stored in our database.

Customize Authentication - Component

Next we need to authenticate our users. The original one just defined 'demo' and 'admin' as the only users that we are able to login. But now we have a database and a list of user and password. We should really secure our login. Here we modify our original authentication method to the following one.

```
public function authenticate()
{
    $username = $this->username;
    $user = User::model()->find('username=?', array($username));
    if($user === NULL)
        $this->errorCode=self::ERROR_USERNAME_INVALID;
    else if(!$user->validatePassword($this->password, $this->username))
        $this->errorCode=self::ERROR_PASSWORD_INVALID;
    else{
        $this->username = $user->username;
        $this->errorCode=self::ERROR_NONE;

    }
    return !$this->errorCode;
}
```

this allowed us to go through the users in our database table instead of the hardcoded one by using the method we wrote previously on the modal folder.

Now, our user will be authenticate using our customized authentication process rather than using the default one!

No related posts.

Revision #3
Created 5 March 2021 19:58:44 by Max
Updated 24 November 2021 08:54:43 by Max