PERTEMUAN KE – 13

Autentikasi dan Otorisasi User

A. TUJUAN

1. Mampu membuat web, membuat aplkasi web dengan pengaturan akses autentikasi dan otorisasi user dalam Framework Yii.
2. Mempu mengimplementasikan user static, user dimanis.

B. TEORI SINGKAT

Otentikasi adalah proses identifikasi pengguna. Ada beberapa aplikasi web memberikan kombinasi nama user/password atau email, melalui pihak ketiga, seperti lewat akun Twitter atau Facebook.

Pengguna yang tampa melalui otentikasi disebut *anonymous*, atau *guest*/tamu. Terkait dengan otentikasi otorisasi. Otorisasi adalah proses menetukan apakah pengguna saat ini diperbolehkan untuk melakukan tugas tertentu atau tidak.

1. Dasar Otentikasi

Di dalam framework Yii, otentikasi menggunakan kelas yang didefinisikan dalam framework.

2. Komponen kunci

Proses otentikasi ini dirancang dalam Yii menjadi cukup fleksibel, sehingga otentikasi dapat dilakukan dengan cara:

- user statis (misalnya, nama uesr “demo” dan password “demo” dan admin passwornya “admin”.
- user dalam tabel yang ada dalam database
- pakai pihak ketiga (misalnya, Facebook atau Twitter)
- Lightweight Directory Access Protocol (LDAP)

Identifikasi user terdapat pada file `/protected/components/UserIdentity.php`

```
?php
/**
 * UserIdentity represents the data needed to identity a user.
 * It contains the authentication method that checks if the provided
 * data can identity the user.
 */
class UserIdentity extends CUserIdentity
{
	/**
	 * Authenticates a user.
	 * The example implementation makes sure if the username and password
	 * are both 'demo'.
	 * In practical applications, this should be changed to authenticate
	 * against some persistent user identity storage (e.g. database).
	 * @return boolean whether authentication succeeds.
	 */
	public function authenticate()
	{
		$users=array(
			// username => password
			'demo'=>'demo',
			'admin'=>'admin',
		);
```

```
            if(!isset($users[$this->username]))
                  $this->errorCode=self::ERROR_USERNAME_INVALID;
            elseif($users[$this->username]!==$this->password)
                  $this->errorCode=self::ERROR_PASSWORD_INVALID;
            else
                  $this->errorCode=self::ERROR_NONE;
            return !$this->errorCode;
      }
}
```

User statis `'demo'=>'demo',` adalah user ‘admin’ password ‘admin’

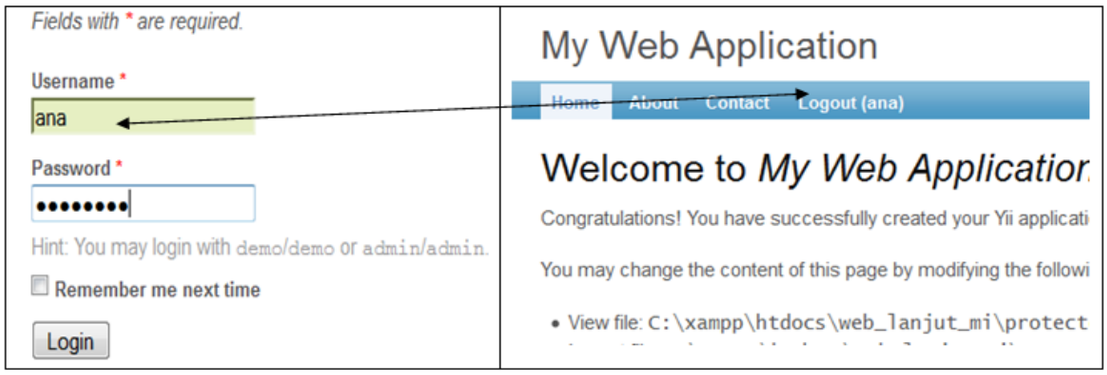
C. PRAKTIK :

1. Menambahkan user statik

Buka /protected/components/UserIdentity.php pada bagian function `authenticate()` tambahkan seperti skrip berikut :

```
public function authenticate()
      {
       $users=array(
                  // username => password
                  'demo'=>'demo',
                  'admin'=>'admin',
                  'ana'=>'ana123?',
                  'budi'=>’budi123?',

             );
      if(!isset($users[$this->username]))
                  $this->errorCode=self::ERROR_USERNAME_INVALID;
      elseif($users[$this->username]!==$this->password)
                  $this->errorCode=self::ERROR_PASSWORD_INVALID;
      else
                  $this->errorCode=self::ERROR_NONE;
            return !$this->errorCode;
      }
```

2. Lakukan pengujian dengan memilih menu Login



3. User Menggunakan tabel

Buatlah tabel seperti pada struktur berikut :

	Field	Type	Comment
🔑	id	int(11)	
	full_name	char(50)	
	username	varchar(128)	
	password	varchar(128)	
	email	varchar(128)	
	active	tinyint(1)	

Index Information

	Indexes	Columns	Index_Type
🔑	PRIMARY	id	Unique

4. Buatlah model dan Controller, view

Buka dan ubah bagian model `User.php`, seperti pada kode program berikut :

file User.php

```
class User extends CActiveRecord
{
   public $password_repeat;

   /**
    * @return string the associated database table name
    */
   public function tableName()
   {
         return 'user';
   }

   /**
    * @return array validation rules for model attributes.
    */
   public function rules()
   {
    // NOTE: you should only define rules for those attributes that
    // will receive user inputs.
     return array(
         array('full_name, username, password, email', 'required'),
         array('active', 'numerical', 'integerOnly'=>true),
         array('full_name', 'length', 'max'=>50),
         array('password, password_repeat', 'length',
               'min'=>2, 'max'=>40),
         array('password', 'compare'),
               array('password_repeat', 'safe'),
         array('username, email', 'length', 'max'=>128),
         // The following rule is used by search().
         // @todo Please remove those attributes that should not be searched.
         array('id, full_name, username, password, email,
               active', 'safe', 'on'=>'search'),
         );
   }

   /**
    * @return array relational rules.
    */
   public function relations()
   {
         // NOTE: you may need to adjust the relation name and the related
         // class name for the relations automatically generated below .
         return array(
         );
   }

   /**
```

```
     * @return array customized attribute labels (name=>label)
     */
    public function attributeLabels()
    {
          return array(
                'id' => 'ID',
                'full_name' => 'Full Name',
                'username' => 'Username',
                'password' => 'Password',
                'email' => 'Email',
                'active' => 'Active',
          );
    }

// enkripsi Password //

protected function beforeSave()
    {
          if(parent::beforeSave())
          {
                if($this->isNewRecord)
                {
                      $this->password=MD5($this->password);
                }
                else
                    $this->password=MD5($this->password);
                return true;
          }
          else
                return false;
    }

…..
}
```

akhir file User.php

Perintah :

`public $password_repeat;` varibel untuk mengulangi password. dan perintah **`array('password', 'compare'),`** adalah validasi mebandingkan untuk mengulangi penulisan password.

methode **`protected function beforeSave()`** trigger yang dijalankan sebelum menyimpan rekaman, dan **`$this->password=MD5($this->password);`** penyimpanan password dienkripsi

5. Ubah bagian `views/user/_form.php` dan tambahkan seperti pada kode program berkut :

file _form.php

```
<?php
/* @var $this UserController */
/* @var $model User */
/* @var $form CActiveForm */
?>

<div class="wide form">

<?php $form=$this->beginWidget('CActiveForm', array(
    'id'=>'user-form',

    'enableAjaxValidation'=>true,
    'enableClientValidation'=>true,
    'clientOptions' => array(
```

```
                    'validateOnType'=>true,
                    'validateOnChange'=>true,
                   ),
)); ?>

   <p class="note">Yang Bertanda <span class="required">*
      </span> tidak boleh kosong.</p>

   <?php echo $form->errorSummary($model); ?>

   <div class="row">
         <?php echo $form->labelEx($model,'full_name'); ?>
         <?php echo $form->textField($model,'full_name',
               array('size'=>50,'maxlength'=>50)); ?>
         <?php echo $form->error($model,'full_name'); ?>
   </div>

   <div class="row">
         <?php echo $form->labelEx($model,'username'); ?>
         <?php echo $form->textField($model,'username',
               array('size'=>60,'maxlength'=>128)); ?>
         <?php echo $form->error($model,'username'); ?>
   </div>

   <div class="row">
         <?php echo $form->labelEx($model,'password'); ?>
         <?php echo $form->passwordField($model,'password',
               array('size'=>40,'maxlength'=>128)); ?>

         <?php echo $form->error($model,'password'); ?>
   </div>

    <div class="row">
         <?php echo $form->labelEx($model,'password_repeat'); ?>
         <?php echo $form->passwordField($model,'password_repeat',
               array('size'=>40,'maxlength'=>40)); ?>
         <?php echo $form->error($model,'password_repeat'); ?>
   </div>

   <div class="row">
         <?php echo $form->labelEx($model,'email'); ?>
         <?php echo $form->textField($model,'email',
               array('size'=>60,'maxlength'=>128)); ?>
         <?php echo $form->error($model,'email'); ?>
   </div>

   <div class="row">
         <?php echo $form->labelEx($model,'active'); ?>
         <?php echo $form->textField($model,'active'); ?>
         <?php echo $form->error($model,'active'); ?>
   </div>

   <div class="row buttons">
         <?php echo CHtml::submitButton($model->isNewRecord ? 'Create' :
'Save'); ?>
   </div>

<?php $this->endWidget(); ?>

</div><!-- form -->
```

file _form.php

perintah :

```
'enableAjaxValidation'=>true,
   'enableClientValidation'=>true,
```

```
'clientOptions' => array(
                    'validateOnType'=>true,
                    'validateOnChange'=>true,
                  ),
```

pengecekan validasi masukan menggunakan ajax.

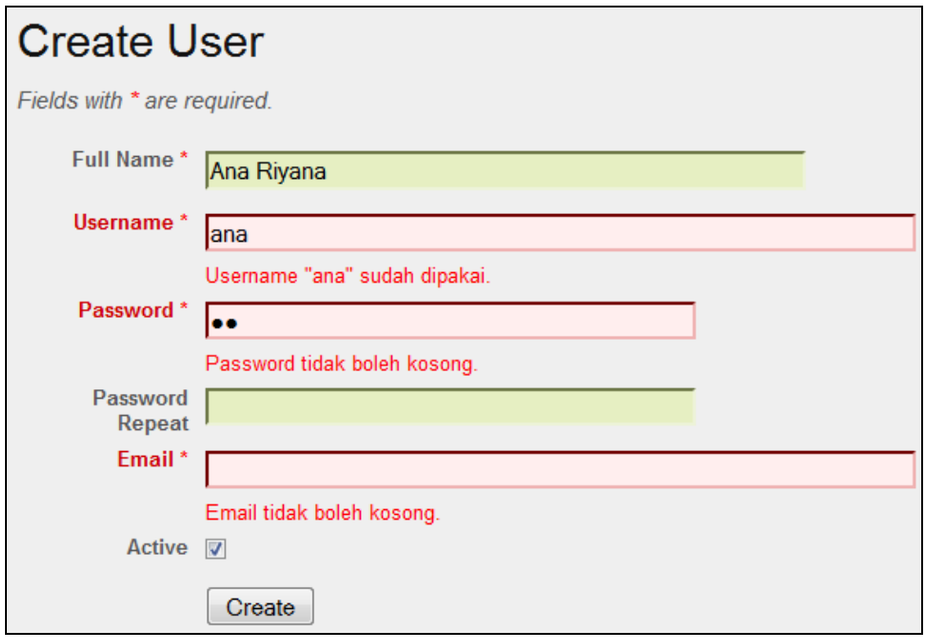
6. Ubah di bagian controller `UserController.php`, pada function actionCreate() seperti pada skrip berikut :

skrip function actionCreate()

```
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'];

            if($model->save())
                $this->redirect(array('view','id'=>$model->id));
        }

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

akhir skrip function actionCreate()

7. uji hasilnya
http://localhost/pertemuan13/index.php?r=user/create
isikan beberapa user sesuai dengan kebutuhan



Menghubungkan UserIdentity dengan User, tujuannya agar rekaman dalam tabel user bisa digunakan untuk login ke sistem. Buka file /`protected/components/UserIdentity.php` kemudian diubah menjadi seperti pada kode program berikut :

file UserIdentity.php

```
?php
/**********************************************
 * UserIdentity represents the data needed to
 * identity a user.
 * It contains the authentication method that checks
 * if the provided
 * data can identity the user.
 **********************************************/
class UserIdentity extends CUserIdentity
{
 private $_id;

 public function authenticate()
   {
   $user=User::model()->findByAttributes(array('username'=>$this-
>username));
   if($user===null)
       $this->errorCode=self::ERROR_USERNAME_INVALID;
       else if($user->password!==md5($this->password))
            $this->errorCode = self::ERROR_PASSWORD_INVALID;
       else
        {
           $this->_id=$user->id;
           $this->errorCode=self::ERROR_NONE;
        }
        return !$this->errorCode;

    }

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

}
```

file UserIdentity.php

penjelasan:

`$user=User::model()->findByAttributes(array('username'=>$this->username));` membaca nilai dan atribut model/kelas User

```
if($user===null)
   $this->errorCode=self::ERROR_USERNAME_INVALID;
```

jika hasilnya null user tidak dikenal (user tidak valid).

```
 else if($user->password!==md5($this->password))
          $this->errorCode=self::ERROR_PASSWORD_INVALID;
```

mencocokkan password tidak sama, memberikan nilai `$this->errorCode` sama dengan konstanta `ERROR_PASSWORD_INVALID;` (password tidak benar)

```
  else
      {
         $this->_id=$user->id;
         $this->errorCode=self::ERROR_NONE;
       }
```

jika user dan password benar maka memperoleh nilai $user->id, dan `$this->errorCode = self::ERROR_NONE;` (bernilai 0)

8. Untuk ketika menjalankan aplikasi harus login, buka `SiteController.php` ubah di bagian `function actionIndex()` seperti pada skrip berikut:

function actionIndex()

```
public function actionIndex()
{
      if(Yii::app()->user->is Guest)
        $this->redirect(array('login'));
      else
       {
            $this->layout='column2';
            $this->render('index');
        }
}
```

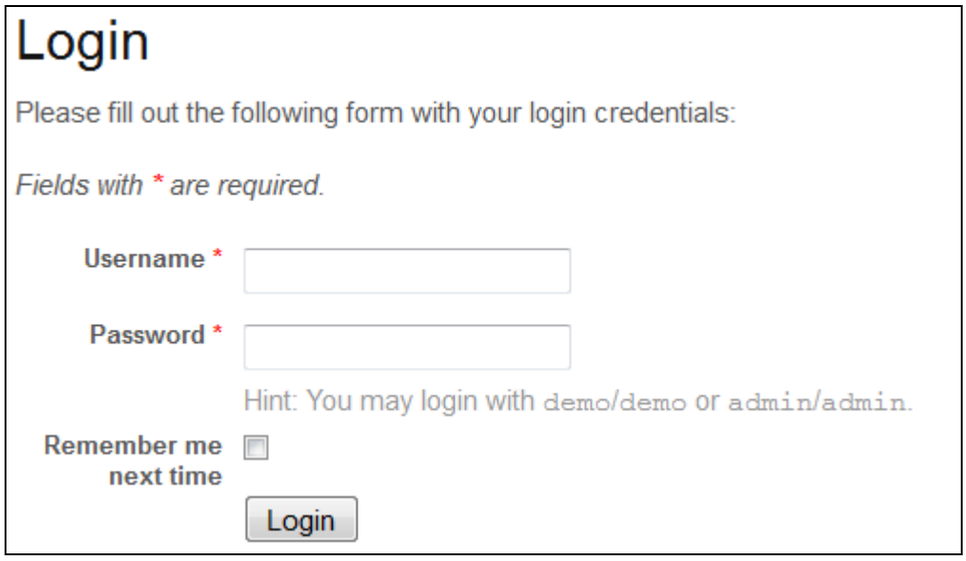
akhir function actionIndex()

penjelasan

```
if(Yii::app()->user->is Guest)
        $this->redirect(array('login'));
```

yii user yang tanpa login dikenal dengan nama guest (tamu).

uji hasilnya dengan menjalankan url : http://localhost/pertemuan13/index.php



D. LATIHAN

Filter kontrol akses merupakan skema awal otorisasi yang memeriksa apakah pengguna saat ini dapat melakukan aksi controller yang diminta. Otorisasi didasarkan pada nama pengguna, alamat IP klien dan jenis permintaan.

1. Ini disediakan sebagai filter bernama "access Control". setiap akses diatur lewat controller, buka salah satu controller yang telah anda buat sebelumnya, misalnya `MahasiswaController.php`

2. function accessRules()

```
public function accessRules()
 {
  return array(
```

```
            array('allow', //semua user bisa akses action 'index' dan 'view'
                  'actions'=>array('index','view'),
                  'users'=>array('*'),
                 ),
            array('allow', //action 'create' dan 'update' diakses dengan
                           //login
                  'actions'=>array('create','update'),
                  'users'=>array('@'),
                  ),
            array('allow', //action 'admin' dan 'delete' hanya boleh diakses
                           //user damin
                 'actions'=>array('admin','delete'),
                  'users'=>array('admin'),
                 ),
            array('deny',  // yang tidak boleh diakses semua user
                         'users'=>array('*'),
            ),
        );
    }
```

3. Buatlah user “ana”, dengan menambahkan lewat model User

4. Ujikan ana diberi akses actions “admin” dan “delete”

5. Apa bedanya user, *, @, ‘admin’

6. Bagaimana caranya create dan update hanya boleh dijalankan oleh “ana” saja

7. Uji login dangan nama “ana” jalankan http://localhost/index.php?r=mahsiswa/create dan http://localhost/index.php?r=mahsiswa/update

8. Coba login dengan user lain, ujikan dan cara yang sama, apa hasilnya?

E. TUGAS

Perhatikan perintah di bawah :

```
return array(
            array('allow','actions'=>array('laporan1','laporan2'),
                  'users'=>array('operator'),
```

Dalam perintah di atas jelasakan penggunaan
allow

1. 'actions'=>array('laporan1','laporan2'),
2. 'users'=>array('operator'),
3. Apa arti Yii::app()->user->isGuest