# Validation des formulaires

<
ORM IgnitedRecord >>

- Aide codeIgniter sur la librairie form_validation

La bibliothèque **form_validation** prend en charge le contrôle et la validation des formulaires, et l'affichage de messages d'erreur.
**form_validation** doit être chargée :

- soit automatiquement avec autoload.php :

```
$autoload['libraries'] = array('form_validation');
```

- Soit dans le code d'un contrôleur par exemple :

```
$this->load->library('form_validation');
```

## Formulaire

Créer le formulaire myForm.php dans application/views/

|h application/views/myForm.php

```
<html>
<head>
<title>My Form</title>
</head>
<body>

<?php echo validation_errors(); ?>

<?php echo form_open('form'); ?>

<h5>Nom d'utilisateur</h5>
<input type="text" name="username" value="" size="50" />

<h5>Mot de passe</h5>
<input type="text" name="password" value="" size="50" />

<h5>Confirmation de mot de passe</h5>
<input type="text" name="passconf" value="" size="50" />

<h5>Adresse email</h5>
<input type="text" name="email" value="" size="50" />

<div><input type="submit" value="Submit" /></div>

</form>

</body>
```

```
</html>
```

Le formulaire à afficher en cas de succès :

|h application/views/formSuccess.php

```
<html>
<head>
<title>My Form</title>
</head>
<body>

<h3>Le formulaire a été correctement soumis !</h3>

<p><?php echo anchor('form', 'Essayer à nouveau'); ?></p>

</body>
</html>
```

Créer un contrôleur pour accéder et contrôler la vue :

|h application/controllers/form.php

```
<?php

class Form extends CI_Controller {

    function index()
    {
        $this->load->helper(array('form', 'url'));

        $this->load->library('form_validation');

        if ($this->form_validation->run() == FALSE)
        {
            $this->load->view('myform');
        }
        else
        {
            $this->load->view('formsuccess');
        }
    }
}
?>
```

Accéder à l'url : http://localhost/testPhp/form/ pour tester le formulaire. Pour l'instant, en l'absence de règles de validation, la soumission du formulaire retourne toujours false et affiche à nouveau le formulaire.

## Ajout de règles de validation

Avec ajout de règles de validation, la validation du formulaire fonctionne complétement.

```
<?php
class Form extends CI_Controller {

    function index()
    {
        $this->load->helper(array('form', 'url'));

        $this->load->library('form_validation');

        $this->form_validation->set_rules('username', 'Nom d\'utilisateur',
'required');
        $this->form_validation->set_rules('password', 'Mot de passe', 'required');
        $this->form_validation->set_rules('passconf', 'Confirmation de mot de
passe', 'required');
        $this->form_validation->set_rules('email', 'Email', 'required');
        if ($this->form_validation->run() == FALSE)
        {
            $this->load->view('myform');
        }
        else
        {
            $this->load->view('formsuccess');
        }
    }
}
?>
```

### Ajout de règles en cascade

Remplacer les règles précédentes par :

```
$this->form_validation->set_rules('username', 'Nom d''utilisateur',
'trim|required|min_length[5]|max_length[12]|xss_clean');
$this->form_validation->set_rules('password', 'Mot de passe',
'trim|required|matches[passconf]|md5');
$this->form_validation->set_rules('passconf', 'Confirmation', 'trim|required');
$this->form_validation->set_rules('email', 'Email', 'trim|required|valid_email');
```

Le contrôle inclut cette fois une transformation de la valeur des champs avant validation :

- trim supprime les espaces avant et après
- md5 crypte en md5
- xss_clean supprime tous les éléments susceptibles de permettre une attaque xss (Cross Site Scripting)

### Ajout d'une fonction de validation utilisateur

|h form.php

```
<?php

class Form extends CI_Controller {

    public function index()
```

```
    {
        $this->load->helper(array('form', 'url'));

        $this->load->library('form_validation');

        $this->form_validation->set_rules('username', 'Nom d''utilisateur',
'callback_username_check');
        $this->form_validation->set_rules('password', 'Mot de passe',
'required');
        $this->form_validation->set_rules('passconf', 'Confirmation',
'required');
        $this->form_validation->set_rules('email', 'Email',
'required|is_unique[users.email]');

        if ($this->form_validation->run() == FALSE)
        {
            $this->load->view('myform');
        }
        else
        {
            $this->load->view('formsuccess');
        }
    }

    public function username_check($str)
    {
        if ($str == 'test')
        {
            $this->form_validation->set_message('username_check', 'Le champ %s
ne peut pas être égal à "test"');
            return FALSE;
        }
        else
        {
            return TRUE;
        }
    }

}
?>
```

### Affichage des valeur après échec de validation

On utilise **set_value('nomChamp')** pour afficher à nouveau la valeur de chacun des champs (excepté sur les mots de passe).

|h myForm.php

```
<html>
<head>
<title>My Form</title>
</head>
<body>
```

```
<?php echo validation_errors(); ?>

<?php echo form_open('form'); ?>

<h5>Username</h5>
<input type="text" name="username" value="<?php echo set_value('username');
?>" size="50" />

<h5>Password</h5>
<input type="password" name="password" value="" size="50" />

<h5>Password Confirm</h5>
<input type="password" name="passconf" value="" size="50" />

<h5>Email Address</h5>
<input type="text" name="email" value="<?php echo set_value('email'); ?>"
size="50" />

<div><input type="submit" value="Submit" /></div>

</form>

</body>
</html>
```