

Professional backend developer



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

**Centro de
e-Learning**

p. 2

Modulo 1: Introducción y nivelación

Unidad 4: Codeigniter 3 (Parte 2)

Centro de e-Learning SCEU UTN - BA.

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148

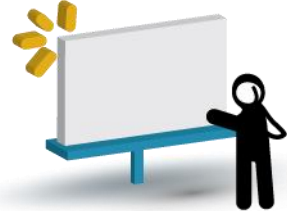
www.sceu.frba.utn.edu.ar/e-learning



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

**Centro de
e-Learning**

p. 3



Presentación:

En la presente unidad aprenderemos mas sobre el framework codeigniter 3, la utilización de constructor y validación de formularios, inclusión de css e imágenes y envíos de email

Centro de e-Learning SCEU UTN - BA.

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148

www.sceu.frba.utn.edu.ar/e-learning



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

**Centro de
e-Learning**

p. 4



Objetivos:

Que los participantes*:

- Aprendan a insertar datos usando codeigniter
- Aprendan la utilización de validación de formularios con form_builder
- Comprendan como incluir hojas de estilos e imágenes

Centro de e-Learning SCEU UTN - BA.

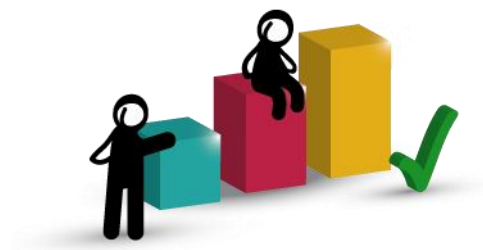
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148

www.sceu.frba.utn.edu.ar/e-learning



Bloques temáticos*:

- Validación de formularios
- Insertar usuario en base de datos
- Enviar emails



Consignas para el aprendizaje colaborativo

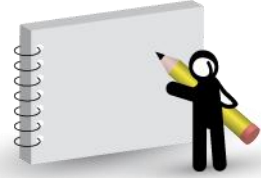
En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.



Tomen nota*

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.



Validación de formularios

A través de form_validation podremos construir y validar formularios de manera simple.

Dicha biblioteca se encuentra en

http://codeigniter.com/userguide3/libraries/form_validation.html

Los pasos a seguir para la utilización de imágenes form_validation serán los siguientes

Autoload

Abrir el archivo application -> config -> autoload.php



```
EXPLORADOR      ...      autoload.php x
EDITORES ABIERTOS
X autoload.php application\config
PHPACODE
v application
  > cache
  v config
    autoload.php
    config.php
    constants.php

54 / $autoload['libraries'] = array('database', 'email', 'session');
55 /
56 / You can also supply an alternative library name to be assigned
57 / in the controller:
58 /
59 / $autoload['libraries'] = array('user_agent' => 'ua');
60 */
61 $autoload['libraries'] = array("database","form_validation");
62
```




Construir formulario HTML

En la vista elegida construimos el formulario HTML, como, por ejemplo:

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');
?>

<h1>Registrarse</h1>
<form action="<? echo base_url()?>registro/save" method="POST">
    <div>
        <label for="">Nombre</label>
        <input type="text" name="nombre">
        <div>
            <? echo form_error("nombre")?>
        </div>
    </div>
    <div>
        <label for="">Apellido</label>
        <input type="text" name="apellido">
        <div>
            <? echo form_error("apellido")?>
        </div>
    </div>
    <div>
        <label for="">Email</label>
        <input type="email" name="email">
        <div>
            <? echo form_error("email")?>
        </div>
    </div>
    <div>
        <label for="">Contraseña</label>
        <input type="password" name="password">
        <div>
            <? echo form_error("password")?>
        </div>
    </div>
    <input type="submit" value="Registrarse">
</form>
```

En este formulario vemos la utilización de elementos input, pero aplica para cualquier elemento de formulario (select, textarea, etc)

Form_error es un helper en codeigniter, el cual veremos luego.



Lógica de validación en el controlador

En el controlador es donde aplicamos la lógica de validación, por ejemplo:

```
public function save(){  
    //var_dump($_POST); exit;  
    try{  
        $this->form_validation->set_rules("nombre","Nombre","required");  
        $this->form_validation->set_rules("apellido","Apellido","required|min_length[3]");  
        $this->form_validation->set_rules("email","Email","required",  
            array("required"=>"El campo %s es obligatorio"));  
        $this->form_validation->set_rules("password","Contraseña","required");  
  
        if($this->form_validation->run()==FALSE){  
            //Error en la validacion  
        }else{  
            //Formulario valido  
        }  
    }  
    }catch(Exception $e){  
    }  
}
```

Para la validación entonces se utiliza:

```
$this->form_validation->set_rules("nombre","Nombre","required");
```

El método set_rules nos permite establecer los criterios de validación.

El primer parámetro indica el name del elemento a validar, es decir debe indicar el mismo nombre que se indica en el name del elemento. Para el ejemplo anterior sería el siguiente input:

```
<label for="">Nombre</label>  
<input type="text" name="nombre">
```



El segundo parámetro indica el texto que se va a mostrar cuando visualicemos el error por pantalla.

El tercer parámetro indica la validación a realizar, las mismas son:

- required
- min_length
- max_length
- max
- min
- email
- pattern: Permite la utilización de expresiones regulares

En el caso de aplicar un min_length lo vemos en la segunda regla, indicamos entre [] la cantidad de caracteres a validar:

```
$this->form_validation->set_rules("apellido", "Apellido", "required|min_length[3]");
```

El mensaje por default está en inglés, el mismo puede ser personalizado enviando un cuarto parámetro:

```
$this->form_validation->set_rules("email", "Email", "required",  
    array("required"=>"El campo %s es obligatorio"));
```

Este cuarto parámetro es un array asociativo, en la clave indica el tipo validación y en su value el mensaje a mostrar. %s será reemplazado por lo indicado en el segundo parámetro (label)



Una vez aplicada la lógica de validación debemos consultar al método run el cual nos indica si el formulario es valido o no. Por ejemplo:

```
if($this->form_validation->run()==FALSE){  
    //Error en la validacion  
}else{  
    //Formulario valido  
}
```

Si el método run() retorna TRUE el formulario es válido, en caso de retorna FALSE el mismo no es válido.

De esta manera validamos el formulario, para luego por ejemplo en caso de ser valido insertar en base de datos.

En caso de que el mismo tenga un error, podemos mostrar los mismos en el HTML a través del helper **form_error**

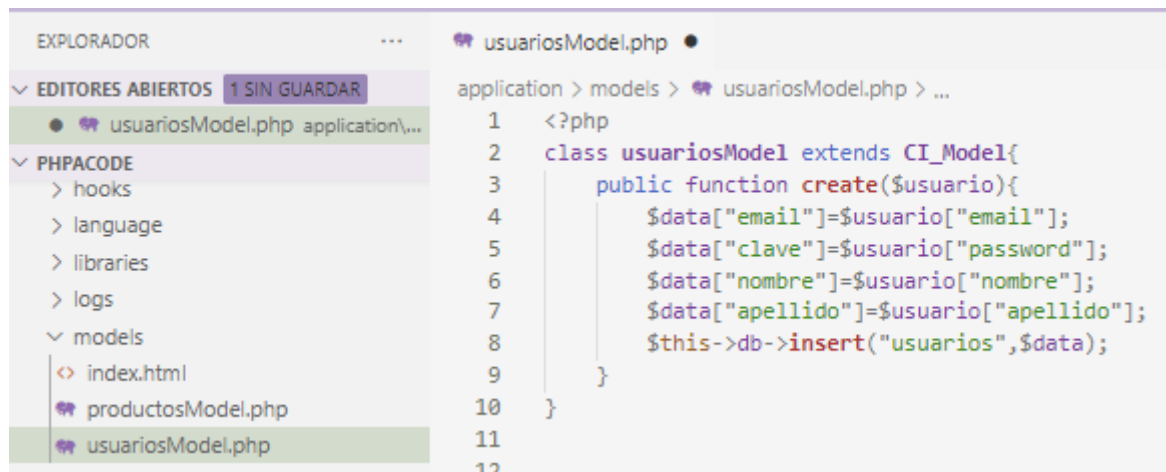
```
<div>  
    <label for="">Nombre</label>  
    <input type="text" name="nombre">  
    <div>  
        <? echo form_error("nombre")?>  
    </div>  
</div>
```

En el ejemplo vemos su utilización, form_error recibe como parámetro el nombre del campo. Entonces en caso de que ese campo (validado en el controller) tenga un error se visualizara el mensaje en pantalla.



Insertar usuario en base de datos

En caso de que el formulario sea valido debemos insertar el mismo en base de datos. Lo primero que vamos a crear es un método en el modelo usuarios:



```
1 <?php
2 class usuariosModel extends CI_Model{
3     public function create($usuario){
4         $data["email"]=$usuario["email"];
5         $data["clave"]=$usuario["password"];
6         $data["nombre"]=$usuario["nombre"];
7         $data["apellido"]=$usuario["apellido"];
8         $this->db->insert("usuarios",$data);
9     }
10 }
11
12
```

Una vez generado el método llamamos al mismo desde el controlador:

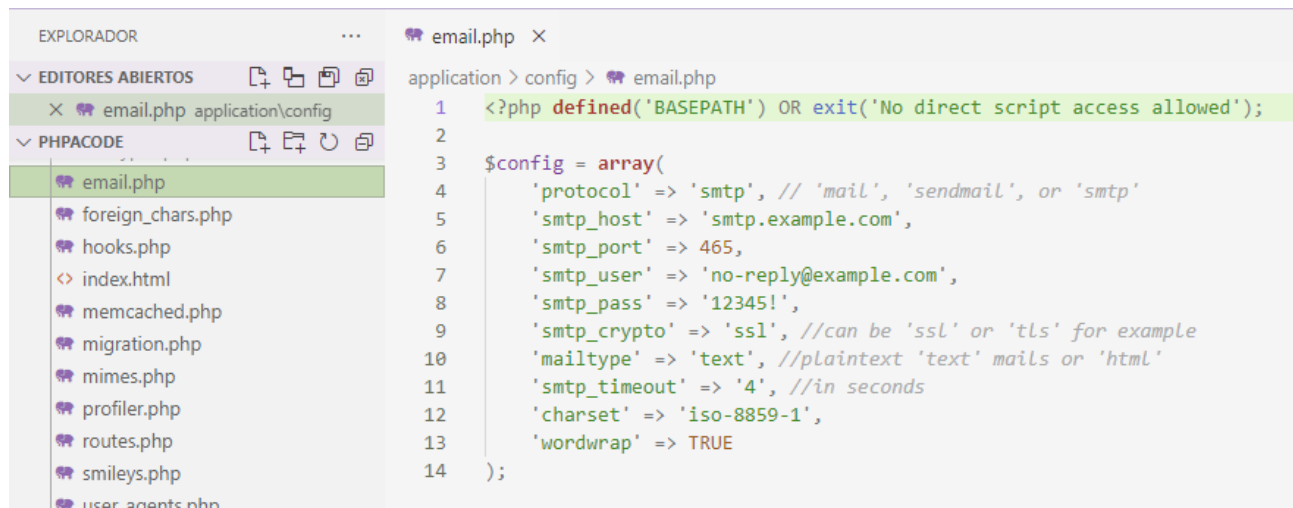
```
public function save(){
    //var_dump($_POST); exit;
    try{
        $this->form_validation->set_rules("nombre", "Nombre", "required");
        $this->form_validation->set_rules("apellido", "Apellido", "required|min_length[3]");
        $this->form_validation->set_rules("email", "Email", "required",
            array("required"=>"El campo %s es obligatorio"));
        $this->form_validation->set_rules("password", "Contraseña", "required");

        if($this->form_validation->run()==FALSE){
            //Error en formulario
        }else{
            $this->load->model("usuariosModel");
            $this->usuariosModel->create($_POST);
        }
    }catch(Exception $e){
    }
}
```



Enviar emails

Debemos agregar un archivo **email.php** dentro del directorio **application -> config** con la siguiente configuración:



```
1 <?php defined('BASEPATH') OR exit('No direct script access allowed');
2
3 $config = array(
4     'protocol' => 'smtp', // 'mail', 'sendmail', or 'smtp'
5     'smtp_host' => 'smtp.example.com',
6     'smtp_port' => 465,
7     'smtp_user' => 'no-reply@example.com',
8     'smtp_pass' => '12345!',
9     'smtp_crypto' => 'ssl', //can be 'ssl' or 'tls' for example
10    'mailtype' => 'text', //plaintext 'text' mails or 'html'
11    'smtp_timeout' => '4', //in seconds
12    'charset' => 'iso-8859-1',
13    'wordwrap' => TRUE
14 );
```



Luego en el controller que queramos agregamos lo siguiente

```
$this->load->config('email');
$this->load->library('email');
$from = $this->config->item('smtp_user');
$to = $_POST["email"];
$subject = 'subject'+$_POST['nombre'];
$message = 'message';
// $message = $this->load->view("sdasd", array(), true);

$this->email->from($from);
$this->email->to($to);
$this->email->subject($subject);
$this->email->message($message);

if ($this->email->send()) {
    |         echo 'Your Email has successfully been sent.';
} else {
    |         show_error($this->email->print_debugger());
}
```

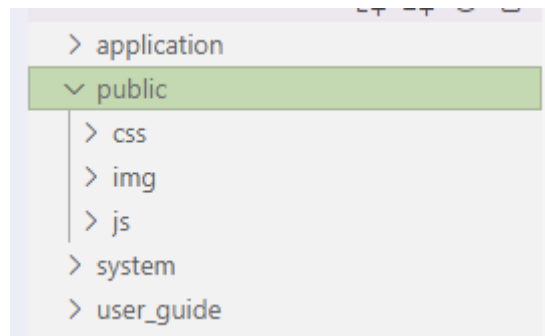
Acá debemos configurar los datos del email destino:

- from: indica el from (origen) a mostrar en el email enviado
- to: Direccion email destino
- subject: Indica el asunto del email
- message: Mensaje principal o body del email



Incluir hojas de estilos e imágenes

Debemos crear un directorio public en la raíz del proyecto, en el cual incluimos las hojas de estilo, imágenes y javascript



En caso de incluir css o js lo hacemos desde el head del html utilizando el helper `base_url()`

```
<head>
  <meta charset="utf-8">
  <title><?php echo $title?></title>
  <script src="<?php echo base_url()?>public/js/script.js"></script>
  <link rel="stylesheet" href="<?php echo base_url()?>public/css/styles.css">
</head>
```

De la misma manera incluiremos las imágenes:

```

```




Codeigniter – Base de datos

Insertar

Para insertar un registro en la base de datos desde nuestro modelo, debemos llamar al objeto db y al método **insert()**. Este método recibirá dos parámetros, el nombre de la tabla y un array asociativo en donde los índices serán el nombre de las tablas y los valores son los datos a guardar:

```
$data = array(  
    'titulo' => 'Título del informe',  
    'descripcion' => 'Descripción del informe',  
    'prioridad' => 3  
);  
$this->db->insert('informes', $data);
```

Actualizar

El método update sirve para modificar un registro existente.

Con el método **where** establecemos la condición para la modificación.

```
$data = array(  
    'titulo' => 'Título del informe',  
    'descripcion' => 'Descripción del informe',  
    'prioridad' => 3  
);  
$this->db->where('id', 1);  
$this->db->update('informes', $data);
```

El primer parámetro del método update es la tabla que queremos modificar, el segundo parámetro es el array de datos a modificar.



Ultimo id insertado

Cuando insertamos un registro nuevo, mediante este método podemos obtener el id en el que se insertó.

```
$id = $this->db->insert_id();|
```

Eliminar

Llamamos el método **delete** en el cual pasamos como parámetro la tabla en donde queremos eliminar el registro.

Con el **where** establecemos la condición para realizar la eliminación.

```
$this->db->where('id', 1);  
$this->db->delete('informes');|
```

Join

```
$this->db->select('*');  
$this->db->from('publicaciones');  
$this->db->join('comentarios', 'comentarios.publicacion_id = publicaciones.id');  
$consulta = $this->db->get();  
$resultado = $consulta->result();
```

A la sentencia vista anteriormente le agregamos la sentencia **join**.

El primer parámetro indica la tabla con la que queremos realizar el join, el segundo parámetro indica las condiciones del **ON** del join. En caso de querer realizar un left join debemos pasar un tercer parámetro que indique 'left'.

Consulta simple

```
$consulta = $this->db->query("SELECT * FROM informes");  
$resultado = $consulta->result();|
```

Mediante el método query podemos realizar cualquier tipo de consulta, desde un select hasta un update o insert.



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

**Centro de
e-Learning**

p. 19

Centro de e-Learning SCEU UTN - BA.

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148

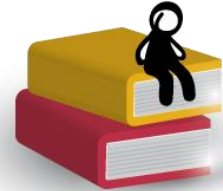
www.sceu.frba.utn.edu.ar/e-learning



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

**Centro de
e-Learning**

p. 20



Bibliografía utilizada y sugerida

<http://www.desarrolloweb.com/articulos/que-es-mvc.html>

<http://www.desarrolloweb.com/manuales/manual-codeigniter.html>

<https://www.codeigniter.com/download>

<http://fernando-gaitan.com.ar/codeigniter-parte-6-modelos/>

http://codeigniter.com/userguide3/libraries/form_validation.html#setting-rules-using-an-array

Centro de e-Learning SCEU UTN - BA.

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148

www.sceu.frba.utn.edu.ar/e-learning



Lo que vimos:

En esta unidad profundizamos en el conocimiento de codeigniter 3, formularios, envíos de email e inclusión de css, js e imágenes





Lo que viene:

En la próxima unidad comenzaremos el camino de javascript, lenguaje de los mas utilizados para front-end en la actualidad

