# Curso de Programación en PHP

# Nivel I

Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología - Oro Verde - v2.1

## Capítulo 6: Seguridad en PHP

**Filtrando datos de entrada y salida**

**Formularios**

**Seguridad en base de datos**

**Inclusión de archivos**

**Registros globales**

**Configurando el archivo php.ini**

## Clase 11: Introducción a la seguridad Web con PHP

Funciones para filtrar la entrada de datos

### ctype_alpha($texto)

Verifica si todos los caracteres en la cadena entregada $texto, son alfabéticos.

### ctype_digit($texto)

Verifica si todos los caracteres en la cadena entregada $texto, son numéricos.

### ctype_alnum($texto)

Chequea si todos los caracteres en la cadena entregada $texto, son alfanuméricos.

### ctype_graph($texto)

Verifica si todos los caracteres en la cadena entregada $texto, generan una salida visible.

# Ejemplo para filtrar la entrada de datos

```
<form method="POST">
    Usuario: <input type="text" name="username" />
    Contraseña: <input type="text" name="password" />
    Color Favorito:
        <select name="color">
            <option value=”rojo”>Rojo</option>
            <option value=”azul”>Azul</option>
            <option value=”amarillo”>Amarillo</option>
        </select>
    <input type="submit" />
</form>
```

# Ejemplo para filtrar la entrada de datos

```
<?php

$clean = array();

if (ctype_alpha($_POST['username'])) {
    $clean['username'] = $_POST['username'];
}
if (ctype_alnum($_POST['password'])) {
    $clean['password'] = $_POST['password'];
}
$colours = array('rojo', 'azul', 'amarillo');
if (in_array($_POST['color'], $colours)) {
    $clean['color'] = $_POST['color'];
}
```

## Clase 11: Introducción a la seguridad Web con PHP

Funciones para filtrar la salida de datos

**htmlentities()**
Convierte todos los caracteres a su entidad HTML aplicable.

**htmlspecialchars()**
Convierte caracteres especiales a entidades HTML.

# Ejemplo para filtrar la salida de datos

**htmlentities**

```
<?php

$str = "<b>Texto en negrita</b>";

echo htmlentities($str);

?>
```

**Html enviado al navegador**

```
<b>Texto en negrita</b>

&lt;b&gt;Texto en negrita&lt;/b&gt;
```

**En pantalla....**

**Texto en negrita**

Texto en negrita

# Ejemplo para filtrar la salida de datos

**htmlspecialchars**

```
<?php

$str="<a href='test'>Test</a>";
echo $str;
$nuevo = htmlspecialchars($str, ENT_QUOTES);
echo $nuevo;

?>
```

**Html enviado al navegador**

```
<a href='test'>Test</a>

&lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;
```

**En pantalla....**

**Test ← enlace**

```
<a href='test'>Test</a>
```

## Capítulo 6: Seguridad en PHP

~~Filtrando datos de entrada y salida~~

**Formularios**

**Seguridad en base de datos**

**Inclusión de archivos**

**Registros globales**

**Configurando el archivo php.ini**

## Falsificación de formularios

Formulario situado http://ejemplo.org/formulario.html.

formulario.html

```
<form action="/process.php" method="POST">
    <select name="color">
        <option value="red">red</option>
        <option value="green">green</option>
        <option value="blue">blue</option>
        </select>
    <input type="submit" />
</form>
```

## **Falsificación de formularios** (continuación)

La misma versión del formulario modificada:

formulario.html

```
<form action="http://example.org/process.php" method="POST">
    <input type="text" name="color" />
    <input type="submit" />
</form>
```

## Falsificación de solicitudes HTTP

La utilidad telnet se puede utilizar para realizar algunas pruebas. En el ejemplo siguiente realiza una sencilla solicitud GET para http://www.php.net/:

```
HTTP/1.1 200 OK
Date: Fri, 18 Jun 2010 03:59:35 GMT
Server: Apache/1.3.41 (Unix) PHP/5.2.12RC4-dev
X-Powered-By: PHP/5.2.12RC4-dev
Last-Modified: Fri, 18 Jun 2010 03:20:22 GMT
Content-language: en
Set-Cookie: COUNTRY=ARG%2C190.228.241.199; expires=Fri, 25-Jun-2010 03:59:35 GMT; path=/;
domain=.php.net
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html;charset=utf-8

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

## Clase 11: Introducción a la seguridad Web con PHP

**Falsificación de solicitudes HTTP** (continuación)

Se puede escribir su propio cliente en lugar de introducir manualmente las solicitudes mediante telnet. Ejemplo con PHP:

```
<?php
$http_response = '';

$fp = fsockopen('www.php.net', 80);
fputs($fp, "GET / HTTP/1.1\r\n");
fputs($fp, "Host: www.php.net\r\n\r\n");

while (!feof($fp))
{
   $http_response .= fgets($fp, 128);
}
fclose($fp);
echo nl2br(htmlentities($http_response));
?>
```

## Cross-Site Scripting (XSS)

Se refiere a vulnerabilidades que abarcaban cualquier ataque que permita ejecutar código de "scripting", como JavaScript en un sitio web.

# Cross-Site Scripting (XSS)

Ejemplo de un simple tablón de mensajes en PHP.

```
<form>
      <input type="text" name="message" />
      <input type="submit" />
</form>

<?php

if (isset($_GET['message']))
{
   $fp = fopen('./messages.txt', 'a');
   fwrite($fp, "{$_GET['message']}");
   fclose($fp);
   ......

  echo $_GET['message'];
}

?>
```

**Algún usuario ingresa:**

```
<script languaje=”text/javascript”>
      javascript:while(1)
      alert("Hola");
</script>
```

## **Cross-Site Scripting (XSS)** (continuación)

Ejemplo de como mitigar el XSS en un simple tablón de mensajes.

```
<form>
	<input type="text" name="message"><br />
	<input type="submit">
</form>

<?php
if (isset($_GET['message']))
{
   $message = htmlentities($_GET['message']);

   $fp = fopen('./messages.txt', 'a');
   fwrite($fp, "$message<br />");
   fclose($fp);
}
  echo $_GET['message'];
?>
```

## Cross-site request forgery o falsificación de petición en sitios cruzados

Ejemplo de una respuesta a una solicitud:

```
<html>
    <img src="http://example.org/image.png" />
</html>
```

```
GET /image.png HTTP/1.1
Host: example.org
User-Agent: Mozilla/5.0 Gecko
Accept: text/xml, image/png, image/jpeg, image/gif, */*
```

**http://stocks.example.org/buy.php?symbol=SCOX&quantity=1000**

# Cross-site request forgery o falsificación de petición en sitios cruzados (continuación)

Ejemplo de un simple tablón de mensajes en PHP mejorado para mitigar ataques de tipo CSRF.

```
<?php
	$token=md5(rand(15000));
?>
<form method="POST">
	<input type="hidden" name="token" value="<?php echo $token; ?>" />
	<input type="text" name="message"><br />
	<input type="submit">
</form>

<?php
	session_start();
	$_SESSION['token'] = $token;
?>
```

# Cross-site request forgery o falsificación de petición en sitios cruzados (continuación)

Ejemplo de un simple tablón de mensajes en PHP mejorado para mitigar ataques de tipo CSRF.

```
<?php
session_start();
if (isset($_POST['message']))
{
if (isset($_SESSION['token']) && ($_POST['token'] == $_SESSION['token']))
    {
        $message = htmlentities($_POST['message']);
        $fp = fopen('./messages.txt', 'a');
        fwrite($fp, "$message<br />");
        fclose($fp);
    }
}

?>
...
```

## Capítulo 6: Seguridad en PHP

~~Filtrando datos de entrada y salida~~

~~Formularios~~

**Seguridad en base de datos**

**Inclusión de archivos**

**Registros globales**

**Configurando el archivo php.ini**

# Exposición de las credenciales de Acceso

Ejemplo de un script con las credenciales de acceso.

db.inc

```
<?php

$motor = 'mysql';
$host = '127.0.0.1';
$port = '3306';
$username = 'root';
$password = '';
$db = 'phpn1_db_clase11';

?>
```

## **Exposición de las credenciales de Acceso** (continuación)

Ejemplo de un script para mitigar acceso a las credenciales de acceso.

db.php

```
<?php

$motor = 'mysql';
$host = '127.0.0.1';
$port = '3306';
$username = 'root';
$password = '';
$db = 'phpn1_db_clase11';

?>
```

# SQL Injection

Ejemplo de inyección de SQL en un formulario de login.

```
<?php
if (isset($_POST['bt_entrar'])) {

	$sql = “SELECT * FROM usuarios WHERE username = '”.$_POST['usuario'];
	$sql .= “' AND password = '”.$_POST['password'].”'”;

	// Nos conectamos a la base y realizamos la consulta
}
?>

<form action='login.php' method='post'>
	Usuario: <input type='text' name='usuario' />
	Contraseña: <input type='password' name='password' />
<input type='submit' name='bt_entrar' value='Entrar' />
</form>
```

# **SQL Injection** (continuación)

## Ejemplo de inyección de SQL en un formulario de login.

```
// Un usuario se logea con el siguiente nombre de usuario
// usuario' OR 1 = 1; --

// Teniendo en cuenta la consulta que se genera del lado del servidor:
        $sql = “SELECT * FROM usuarios WHERE username = '”.$_POST['usuario'];
        $sql .= “' AND password = '”.$_POST['password'].”'”;

// La consulta quedaría de la siguiente forma:
        SELECT * FROM usuarios WHERE username = 'usuario' OR 1 = 1; -- ' password = ''
```

Los **comentarios** en el lenguaje SQL se hacen con el símbolo **--** (doble guión medio).

# **SQL Injection** (continuación)

## Ejemplo de como mitigar la inyección de SQL en un formulario de login. (funciones MySQL)

```
<?php
if (isset($_POST['bt_entrar'])) {
	$usuario =mysql_real_escape_string($_POST['usuario'],$conn);
	$password =mysql_real_escape_string($_POST['password'],$conn);

	$sql = “SELECT * FROM usuarios WHERE username = '”.$usuario;
	$sql .= “' AND password = '”.$password.”'”;

	// Nos conectamos a la base y realizamos la consulta
}
?>
<form action='login.php' method='post'>
	Usuario: <input type='text' name='usuario' />
	Contraseña: <input type='password' name='password' />
	<input type='submit' name='bt_entrar' value='Entrar' />
</form>
```

# **SQL Injection** (continuación)

## Ejemplo de como mitigar la inyección de SQL en un formulario de login. (PHP Data Object)

```
<?php
if (isset($_POST['bt_entrar'])) {
	$sql = “SELECT * FROM usuarios WHERE username = ? AND password = ?”;
	// Nos conectamos a la base de datos

	$stmt = $dbh->prepare($sql);
	$stmt->bindValue(1, $_POST['usuario'], PDO::PARAM_STR);
	$stmt->bindValue(2, $_POST['password'], PDO::PARAM_STR);
	$stmt->execute();
}
?>
<form action='login.php' method='post'>
	Usuario: <input type='text' name='usuario' />
	Contraseña: <input type='password' name='password' />
	<input type='submit' name='bt_entrar' value='Entrar' />
</form>
```

## Capítulo 6: Seguridad en PHP

~~Filtrando datos de entrada y salida~~

~~Formularios~~

~~Seguridad en base de datos~~

**Inclusión de archivos**

**Registros globales**

**Configurando el archivo php.ini**

# Inclusión de archivos en PHP

Ejemplo inclusión dinámica de archivos en PHP.
Los archivos a incluir son enviados a través de la url.

http://www.ejemplo.com.ar/listarUsuario.php?nombreArchivo=config.php

```
<?php

    include "$_GET['nombreArchivo']";

?>
```

## Inclusión de archivos en PHP

Ejemplo para mitigar **code injection** en la inclusión dinámica de archivos en PHP.

```
<?php
    $clean = array();
    $nombreArchivo = array('usuarios.php', 'personas.php');
    if (in_array($_GET['nomberArchivo'], $nombreArchivo)) {
        $clean['nombreArchivo'] = $_GET['nombreArchivo'];
    } else {
        $clean['nombreArchivo'] = 'login.php';
    }

include "$clean['nombreArchivo']";

?>
```

## Capítulo 6: Seguridad en PHP

~~Filtrando datos de entrada y salida~~

~~Formularios~~

~~Seguridad en base de datos~~

~~Inclusión de archivos~~

**Registros globales**

**Configurando el archivo php.ini**

## Register Globals

Permite asumir que ***todas las variables son globales***, sin importar la clasificación de las mismas (de sesión, get, post..)

PHP **3** tuvo inconvenientes de seguridad.

PHP **4** y **5** permitieron configurar el uso de las mismas.

PHP **5.3** las advierte como deprecada.

PHP **5.4** no tiene disponible el uso de *register globals*.

## Clase 11: Introducción a la seguridad Web con PHP

**Register Globals** (continuación)

Ejemplo de acceso a variables de un sistema Web utilizando register globals.

autentico.php

```
....
If (isset($usuario)){
      Dejo que siga mi aplicación....
}
```

http://www.miaplicacion.com.ar/autentico.php?usuario=tue

autentico.php

```
....
If (isset($_SESSION['usuario'])){
      Dejo que siga mi aplicación....
}
```

## Clase 11: Introducción a la seguridad Web con PHP

**Register Globals** (continuación)

Deshabilitar los registros globales de nuestro interprete PHP.

Opción register_globals:

**register_globals = Off**

En el archivo **php.ini** correspondiente al interprete de PHP.

## Capítulo 6: Seguridad en PHP

~~Filtrando datos de entrada y salida~~

~~Formularios~~

~~Seguridad en base de datos~~

~~Inclusión de archivos~~

~~Registros globales~~

**Configurando el archivo php.ini**

# Archivo php.ini

Las capacidades de nuestro lenguaje se encuentran reflejadas en el archivo de configuración php.ini.

**php.ini-development**
La instalación nos deja a nuestra disposición un archivo de configuración preparado para desarrollar aplicaciones.

**php.ini-production**
La instalación también nos deja a nuestra disposición un archivo de configuración preparado para correr aplicaciones en ambientes de producción.

## Clase 11: Introducción a la seguridad Web con PHP

**Archivo php.ini** (continuación)

Ver las capacidades de nuestro lenguaje desde un script PHP.

```
<?php

    phpinfo();

    print_r(ini_get_all());

?>
```

## Clase 11: Introducción a la seguridad Web con PHP

**Archivo php.ini** (continuación)

Obtener y modificar las capacidades de nuestro lenguaje desde un script PHP.

```
<?php

    echo ini_get('display_errors');

    ini_set('display_errors', 'Off');

?>
```

## Clase 11: Introducción a la seguridad Web con PHP

**Archivo php.ini** (continuación)

Modificar la capacidad de registro (log) de nuestro lenguaje desde un script PHP.

```
<?php

    ini_set('log_errors', 'Off');

?>
```

## Clase 11: Introducción a la seguridad Web con PHP

**Archivo php.ini** (continuación)

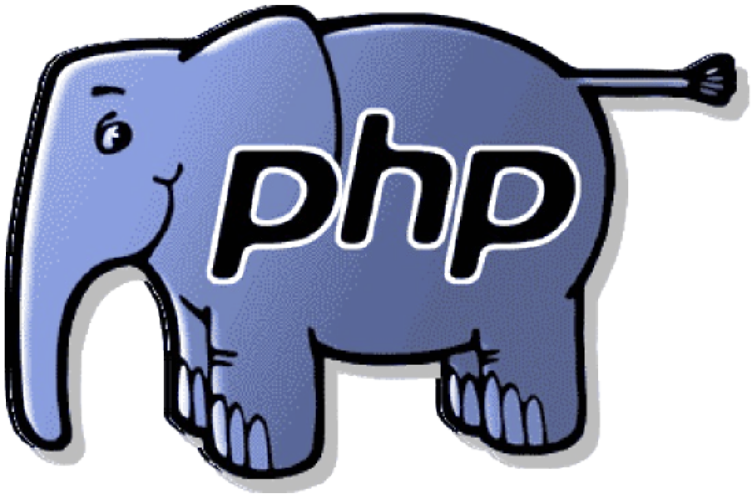
Modificar la capacidad de tiempo máximo de ejecución de un script PHP.

```
<?php

    ini_set('max_execution_time', 300);

?>
```

**¿Dudas?**




**¿Consultas?**

# Información de contacto

**Web:**

http://www.gugler.com.ar

http://campusvirtual.gugler.com.ar

http://www.facebook.com/gugler.com.ar

http://www.twitter.com/cgugler

**Mail:**

contacto@gugler.com.ar

academica@gugler.com.ar

administracion@gugler.com.ar

Versión 2.1