UTN.BA FACULTAD REGIONAL BUENOS AIRES
SECRETARÍA DE EXTENSIÓN UNIVERSITARIA FRBA UTN

Centro de e-Learning

# EXPERTO UNIVERISTARIO EN MySQL Y PHP NIVEL AVANZADO

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

# Módulo II

# CONTINUAMOS CON PHP AVANZADO

# Frameworks PHP.

# CodeIgniter.

# Presentación de la Unidad:

**Veremos cual es el concepto del patrón de diseño MVC (Modelo-Vista-Controlador) y como el utilizar algún Framework basado en este patrón puede contribuir a trabajar de manera más ordenada separando tareas.**

**Veremos la instalación de CodeIgniter y veremos un ejemplo programado sobre la estructura de este framework.**

# Objetivos:

- Comprender el concepto de Patrón de diseño MVC (Modelo-Vista-Controlador).
- Aprender a instalar y configurar CodeIgniter.
- Aprender a utilizar el Framework y las distintas facilidades que nos ofrece.

# Temario:

# CONSIGNAS PARA EL APRENDIZAJE COLABORATIVO

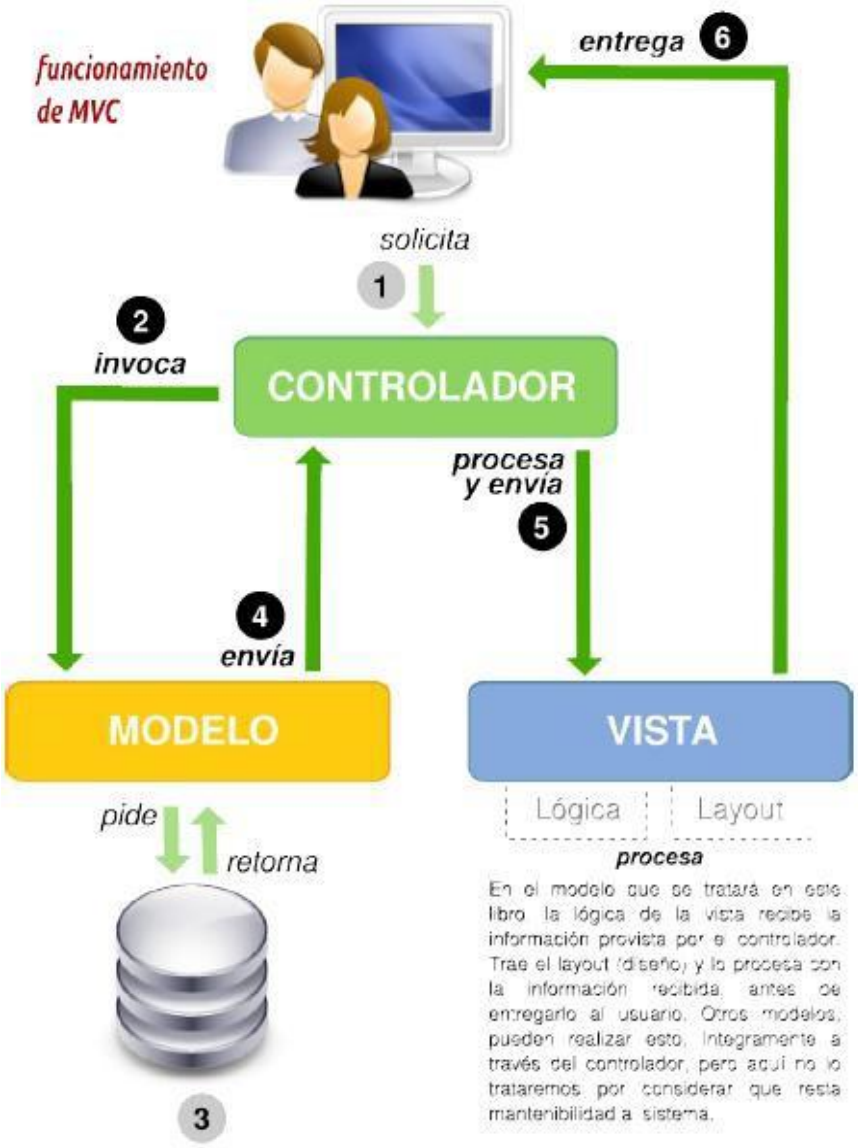
En esta Unidad los participantes se encontrarán con diferentes tipos de consignas que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

1. Los foros asociados a cada una de las unidades.
2. La Web 2.0.
3. Los contextos de desempeño de los participantes.

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

**Modelo Vista Controlador**

Modelo Vista Controlador es un patrón de arquitectura de software que se utiliza mucho en aplicaciones web, separándolas en tres capas (modelo, controlador y vista).




**Modelo:**

Todo el código que tiene que ver con el acceso a base de datos. En el modelo mantendremos encapsulada la complejidad de nuestra base de datos y simplemente crearemos funciones para recibir, insertar, actualizar o borrar información de

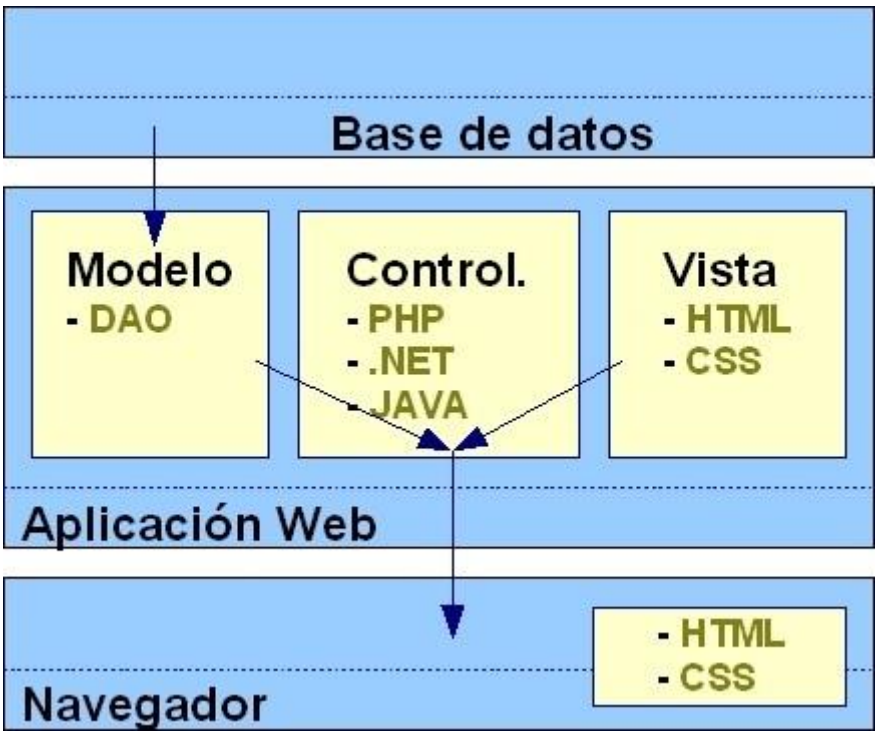
nuestras tablas. Al mantenerse todas las llamadas a la base de datos en un mismo código, desde otras partes del programa podremos invocar las funciones que necesitemos del modelo y éste se encargará de procesarlas. En el modelo nos podrán preocupar cosas como el tipo de base de datos con la que trabajamos, o las tablas y sus relaciones, pero desde las otras partes del programa simplemente llamaremos a las funciones del modelo sin importarnos qué tiene que hace éste para conseguir realizar las acciones invocadas.

**Vista:**

La vista codifica y mantiene la presentación final de nuestra aplicación de cara al usuario. Es decir, en la vista colocaremos todo el código HTML, CSS, Javascript, etc. que se tiene que generar para producir la página tal cual queremos que la vea el usuario. En la práctica la vista no sólo sirve para producir páginas web, sino también cualquier otra salida que queramos enviar al usuario, en formatos o lenguajes distintos, como pueden ser feeds RSS, archivos JSON, XML, etc.

**Controlador:**

El controlador podríamos decir que es la parte más importante, porque hace de enlace entre el modelo, la vista y cualquier otro recurso que se tenga que procesar en el servidor para generar la página web. En resumen, en el controlador guardamos la lógica de nuestras páginas y realizamos todas las acciones que sean necesarias para generarlas, ayudados del modelo o la vista.



**Frameworks MVC**

| Lenguaje | Licencia | Nombre |
|---|---|---|
| Ruby | MIT | Ruby on Rails |
| Ruby | MIT | Merb |
| Ruby | MIT | Ramaze |
| Java / J2ee | Apache | Grails |
| Java / J2ee | GPL | Interface Java Objects |
| Java / J2ee | LGPL | Framework Dinámica |
| Java / J2ee | Apache | Struts |
| Java / J2ee | Apache | Beehive |
| Java / J2ee | Apache | Spring |
| Java / J2ee | Apache | Tapestry |
| Java / J2ee | Apache | Aurora |
| Java / J2ee | Apache | JavaServerFaces |
| JavaScript | GPLv3 | ExtJS 4 |
| Perl | GPL | Mojolicious |
| Perl | GPL | Catalyst |
| Perl | GPL | CGI::Application |
| Perl | GPL | Gantry Framework |
| Perl | GPL | Jifty |
| Perl | GPL | Maypole |
| Perl | GPL | OpenInteract2 |
| Perl | Comercial | PageKit |
| Perl | GPL | Cyclone 3 |
| Perl | GPL | CGI::Builder |
| PHP | GPL | Self Framework ( php5, MVC, ORM, Templates, I18N, Multiples DB) |
| PHP | LGPL | ZanPHP |
| PHP | LGPL | Tlalokes |
| PHP | GPL | SiaMVC |
| PHP | LGPL | Agavi |
| PHP | BSD | Zend Framework |
| PHP | MIT | CakePHP |
| PHP | GNU/GPL | KumbiaPHP |

| | | |
|---|---|---|
| PHP | MIT | Symfony |
| PHP | MIT | QCodo |
| PHP | GNU/GPL | CodeIgniter |
| PHP | Otra | Kohana |
| PHP | MPL 1.1 | PHP4ECore |
| PHP | BSD | PRADO |
| PHP | GNU | FlavorPHP |
| PHP | Apache 2.0 | Yupp PHP Framework |
| PHP | BSD | Yii PHP Framework |
| PHP | GPL | Logick PHP Framework |
| PHP | GPL | Osezno PHP Framework |
| Python | ZPL | Zope3 |
| Python | Varias | Turbogears |
| Python | GPL | Web2py |
| Python | BSD | Pylons |
| Python | BSD | Django |
| .NET | Castle Project | MonoRail |
| .NET | Castle Project | MonoRail |
| .NET | Apache | Spring .NET |
| .NET | Apache | Maverick .NET |
| .NET | MS-PL | ASP.NET MVC |

**Características generales de CodeIgniter**

De esta lista de frameworks MVC que acabamos de observar vamos a tomar CodeIgniter. Veremos paso a paso la instalación del Framework y como armar una aplicación sencilla sobre el mismo.

Por estar basados todos en la misma arquitectura (MVC) una vez que conocen y comprenden el funcionamiento de uno de ellos es mas sencillo aprender cualquiera de los otros ya que todos están basados en el mismo concepto.

Algunos de los puntos más interesantes sobre este framework, sobre todo en comparación con otros productos similares, son los siguientes:

**Versatilidad**: Quizás la característica principal de CodeIgniter, en comparación con otros frameworks PHP. CodeIgniter es capaz de trabajar la mayoría de los entornos o servidores, incluso en sistemas de alojamiento compartido, donde sólo tenemos un acceso por FTP para enviar los archivos al servidor y donde no tenemos acceso a su configuración.

**Compatibilidad**: CodeIgniter es compatible con la versión PHP 4, lo que hace que se pueda utilizar en cualquier servidor, incluso en algunos antiguos. Por supuesto, funciona correctamente también en PHP 5.

> Nota: Desde la versión 2 de CodeIgniter ya solo es compatible con la versión 5 de PHP. Para los que todavía usen PHP 4 pueden descargar una versión antigua del framework, como CodeIgniter V 1.7.3, que todavía era compatible. Estas versiones están en la página de descargas de CodeIgniter.

**Facilidad de instalación**: No es necesario más que una cuenta de FTP para subir CodeIgniter al servidor y su configuración se realiza con apenas la edición de un archivo, donde debemos escribir cosas como el acceso a la base de datos. Durante la configuración no necesitaremos acceso a herramientas como la línea de comandos, que no suelen estar disponibles en todos los alojamientos.

**Flexibilidad**: CodeIgniter es bastante menos rígido que otros frameworks. Define una manera de trabajar específica, pero en muchos de los casos podemos seguirla o no y sus reglas de codificación muchas veces nos las podemos saltar para trabajar como más a gusto nos encontremos. Algunos módulos como el uso de plantillas son totalmente opcionales. Esto ayuda muchas veces también a que la curva de aprendizaje sea más sencilla al principio.

**Ligereza**: El núcleo de CodeIgniter es bastante ligero, lo que permite que el servidor no se sobrecargue interpretando o ejecutando grandes porciones de código. La mayoría de los módulos o clases que ofrece se pueden cargar de manera opcional, sólo cuando se van a utilizar realmente.

**Documentación tutorializada**: La documentación de CodeIgniter es fácil de seguir y de asimilar, porque está escrita en modo de tutorial. Esto no facilita mucho la referencia rápida, cuando ya sabemos acerca del framework y queremos consultar sobre una función o un método en concreto, pero para iniciarnos sin duda se agradece mucho.

Sin duda, lo más destacable de CodeIgniter es su accesibilidad, ya que podemos utilizarlo en la mayor gama de entornos. Esta es la razón por la que hemos elegido este framework PHP. En siguientes artículos ítems iremos contando diferentes aspectos de este framework y lo utilizaremos para crear una primera aplicación web. Para continuar vamos a ver como hacer la instalación y configuración de CodeIgniter.

**Pasos para la instalación de Code Igniter**

1-Bajar la ultima versión de Code Igniter de http://codeigniter.com/downloads/

2-Descomprimir el paquete

3-Subirlo al servidor (al ser en modo local copiar la carpeta de Code Igniter en c://xammp/htdocs)

4-Configurar la URL base de la aplicación: para ello hay que modificar en el archivo de configuración application/config/config.php la línea:
$config[‘base_url’]=”http://localhost/CodeIgniter_3.0.6”

5-Configurar la base de datos: para ello editar el archivo application/config/database.php y modificar estos cuatro parámetros:
$db['default']['hostname'] = "localhost";
$db['default']['username'] = "root";
$db['default']['password'] = "";
$db['default']['database'] = "aplicacionarticulos";

6-Establecer el controlador por defecto: editando el archivo application/config/routes.php modificando la variable: $route[‘default_controller’]=’productos’

En el archivo: application/config/autoload.php
Se pueden configurar librerias, helpers, modelos, etc que queremos que CodeIgniter cargue automáticamente modificando las variables:
$autoload[‘libraries’] = array(‘database’); por ej.
$autoload[‘model’] = array(‘Articulo_model’,’Usuario_Model’); por ej.
$autoload[‘helper’] = array(‘url’, ’file’, ‘cookie’);

Los modelos o los helpers si no están agregados en este archivo debemos cargarlos en los controladores que vayan a usarlos mediante $this->load->helper(‘url’); o $this-

>load->model('articulo_model');

**Ejemplos:**

Para comprender el funcionamiento de una aplicación en CodeIgniter veremos una serie de ejemplos los cuales irán subiendo la complejidad y se irán incorporando conceptos.

Una vez que tenemos hecha la instalación y hemos modificado los archivos de configuración estamos en condiciones de ir probando los distintos ejemplos.

Para ello debemos saber que nuestros ejemplos constarán de tres tipos de scripts:

- Controladores
- Modelos
- Vistas

Y que cada uno de estos scripts deben estar cada uno en la carpeta correspondiente dentro de la estructura de archivos que quedó conformada una vez que finalizamos la instalación de CodeIgniter, asi:

Los controladores irán en la carpeta
C:\xampp\htdocs\CodeIgniter_3.0.6\application\controllers
Las vistas irán en la carpeta
C:\xampp\htdocs\CodeIgniter_3.0.6\application\views
Los modelos irán en la carpeta
C:\xampp\htdocs\CodeIgniter_3.0.6\application\models

Como bien dijimos en estos 5 ejemplos iremos subiendo la complejidad e iremos incorporando conceptos nuevos en cada uno de ellos.

He aquí la composición de cada uno de los ejemplos, para ir probándolos debemos copiar cada uno de los archivos en la carpeta correspondiente e ir modificando el archivo routes.php según corresponda al ejercicio que queremos probar.

Por ejemplo en el caso numero 1 debemos copiar el archivo productos.php en la carpeta C:\xampp\htdocs\CodeIgniter_3.0.6\application\controllers y modificar el archivo routes.php la línea correspondiente (según se explica en el punto 6 de la instalación y configuración de CodeIgniter). Y así con cada uno de los distintos ejemplos respetando, las carpetas mencionadas anteriormente y la modificación del archivo routes.php

**Ej 1**-controlador basico para explicar funcionamiento de code igniter
productos.php (Controlador)

**Ej 2**-mostrar ej basico de vista
micontrolador.php (Controlador)
mivista.php (Vista)

**Ej 3**-como pasar datos mediante variables a la vista que se carga desde el controlador
micontrolador2.php (Cotrolador)
mivista2.php (Vista)

**Ej 4**-introduccion de modelos y acceso desde el modelo mediante Active Record Class
articulos.php (Controlador)
home.php (Vista)
muestra_articulo.php (Vista)
articulo_model.php (Modelo)

**Ej 5**-como armar plantillas para tratar todo un sitio con el mismo diseño mediante anidacion de vistas
articulos_plantilla.php (Controlador)
plantilla_articulo.php (Vista)
listado_articulo.php (Vista)
cuerpo_articulo.php (Vista)
articulo_model.php (Modelo)

**Ej 5b**-Idem 5 con otra manera de anidar las vistas
articulos_plantilla2.php (Controlador)
cabecera.php (Vista)
detalle_articulo.php / contenido_articulos.php (Vista)
pie.php (Vista)
articulo_model.php (Modelo)

**Ej 6**-Formulario de login
php.php (C)
login.php (V)
usuarios_model.php (M)

**Ej 7**-formulario de login y manejo de variables de sesion
php2.php (C)
login2.php (V)
escritorio.php (V)
usuarios_model.php (M)

**Página Code Igniter**

http://codeigniter.com/