**AsenZor**

Guía del Desarrollador

**Edición nº1**

**Por:** Jesús Abraham Zerpa Maldonado

**PRELIMINARES**

**Reconocimientos**

Primero que nada me gustaría comenzar agradeciendo a Dios ya que sin el nada de esto hubiera sido posible, ha sido el quien ha guiado todos mis pasos desde el inicio de este proyecto, es increíble saber sentir su fuerza mas aun en estos últimos meses, despertares intelectuales que aun cambiado mi forma de pensar hacia una cada vez mas madura, echos como conseguir un trabajo en mi área de especialidad, con la oportunidad de aprender muchas mas cosas fundamentales para mis proyectos profesionales abriéndome camino no solo en la programación sin ahora también al marketing y en el momento en que mas lo estaba necesitando, se puede decir que ahora veo las cosas con mas claridad.

En segundo lugar agradecer a mi madre con la que siempre estaré agradecido no solo por darme la vida sino también por toda su lucha para sacarme adelante y hacerme el hombre que hoy en día soy, si de alguien puedo decir que saque lo “echado para adelante” fue de ella, gracias por apoyarme en todas mis locuras jajaja. En tercer lugar a los demás miembros de mi familia mi prima Lili (Carmen) quien aun en sus arranques por sus finanzas y la situación país ha sido mi escucha para mis ideas de negocio, a mi difunta tia Migue con quien aprendí a ser alguien intelectual y a definir mis creencias, ademas de haber sido mi madrina y socia de mi primer intento de legalizar una empresa.

En cuarto lugar agradecer a esos profesores que han creído en mi, el Profesor Pedro Ocanto, El profesor Nairo, el profesor Julio Acosta.

En quinto lugar a mis amigos y compañeros de trabajo quienes en sana competencia, Criticas constructivas, formación de sociedades y dinámica de equipo me han enseñado a darme cuenta de muchos de mis errores y de buscar la manera de corregirlos

**Sobre el autor**

**Jesús Zerpa** Desarrollador full stack, es el creador de los web frameworks FlashFront (para FrontEnd) y AsenZor (para BackEnd) fundador de ZerpaTechnology, web stratup para creación de productos tecnológicos con relación principalmente enfocada al software y un de los fundadores de CIDGUN fundación sin animo de lucro para la innovación tecnológica de estudiantes universitarios en el núcleo de UNEXPO Charallave, a la fecha cursando la carrera de ingeniería mecatrónica.

**Sobre este libro**

Este libro a sido publicado a modo que muchas personas puedan empezar a aprender utilizar este framewok que he creado y de tal manera compartirme sus comentarios del mismo, algunas posibles mejoras que vean pertinentes, alguna duda que tengan sobre como usar python y crear sus sitios web de una forma facil o compleja depende del alcance de su proyecto y que este sirva a alguien mas o sea AsenZor no solo la herramienta de la creación de su sitio web sino también su plataforma de negocios.

**INTRODUCCIÓN**

Con el nacimiento de la Internet cada vez mas se a ido pensando en cuales serian las mejores formas compartir un contenido, esta técnicas han ido evolucionando al paso del tiempo, se a necesitado estandarizar una estructura de representación de datos con lo cual nació el HTML un lenguaje de estructurado el cual hoy dia sigue siendo la base de las paginas web que podemos apreciar al buscar en nuestros Navegadores web ya sea firefox, Chrome, Safari, etc. En el pasado no existían las hojas de estilos, esos archivos con extinción .css los cuales nos permiten dar la forma y colores a los objetos en HTML ademas de que este HTML se hacia manualmente y crear cada cosa se hacia un proceso tedioso.

Fue hasta la llegada de una tecnología llamada CGI (Puerta de Enlace de uso Común, sus siglas derivan del ingles) la ha permitido que los lenguajes de programación puedan manipular estos HTML y entregarlos al servidor quien enviá estos códigos trabajados con lo cual se crea el uso de contenido dinámico, ya las cosas no son tan repetitivas, sin embargo la creación de un sitio web muchas veces puede resultar en crear algo mas halla de una simple pagina por lo cual ha sido necesario la creación de diversos frameworks los archivos de código que organizan el flujo de desarrollo, estos ha sido creados para sus respectivos lenguajes entre ellos están Codigniter, Laravel, Wordpress para PHP un lenguaje que fue uno de los primeros en salir al mercado del web y a logrado posicionarse en proyectos muy importantes, con una facilidad de embebido en HTML natural cosa que en los otros lenguajes solo se logra gracias a los framework PHP lo puede hacer sin necesidad de estos y ya viene preparado para casi todos los servidores, pero este posee muchas limitaciones que para este momento lo están convirtiendo en un lenguaje del pasado, entre ellas destacan su dificultad para la orientación a objetos y su sintaxis cosa que no lo hace mucho de mi agrado.

Luego esta Ruby on Rails para ruby framework que junto a Django para python forman parte de esta llamada tercera generación de frameworks, asi como estos hay muchos lenguajes que han estado desde hace mas tiempo en la web como Perl, Java y Javascript este ultimo en el lado del cliente es decir para ser utilizado por los Navegadores Web pero ahora con el uso de Node este también puede ser usado para el Servidor. Existen quizás mas lenguajes que estos, en nuestro caso nos interesa mas python ya que a pesar de hasta hace poco haber desconocido todas sus bondades para la creacion de sitios web, nos ofrece la mejor sintaxis con la cual podemos programar, su poder en orientacion a objetos contra otros lenguajes es incomparables y este paradigma ha ido ganando mucha importancia en estos últimos tiempos por ser la mejor forma de organizar y mantener nuestro codigo.

**INTRODUCCIÓN A ASENZOR**

Este libro es sobre AsenZor, un framework de desarrollo Web que busca ser un entorno de desarrollo para trabajo colaborativo con persona que usen diferentes lenguajes de programación de una forma fácil y divertida, este framework tiene su nucleo echo en python por lo cual es necesario que sepas algo de python, sin embargo su uso es tan sencillo que hasta tu sin ser un programador en python podrías ponerlo a funcionar, esta es la razon que diferencia a AsenZor de otros frameworks escritos en python como Django o Web2py que solo se enfocan en el lenguaje, otra cosa que debes saber es que AsenZor no usa los patrones de MVC para acceder a su contenido sino por el contrario usa Z-API el cual una manera muy útil de trabajar con las aplicaciones web a través de las url algo muy al estilo de Facebook y Google le proveen a los desarrolladores pero con unas características propias dadas por mi.

**INSTALACIÓN**

AsenZor pretende ser un framwework fácil de usar y esto quiere decir que hasta en el proceso de su instalación se buscan de hacerlo de la forma mas sencilla posible, si has trabajado con codeigniter sabrás o worpress sabrás que para instalar estos debes colocar la carpeta del framework en la carpeta publica del servidor en nuestro caso htdocs para XAMPP y luego se debe editar el archivo de configuración para establecer los parámetros para la base de datos, en el caso de AsenZor este ultimo paso nos lo podemos ahorrar ya que por defecto el framework trabaja con su propio motor de bases de datos no SQL el ZerpaDataBase (zdb), esto no quiere decir que no podamos trabajar con la bases de datos en motores SQL, simplemente preferimos hacerlo de esta manera porque consideramos que es la forma mas natural para trabajar los modelos del framework ya que este motor esta escrito en python por lo cual se presume que su respuesta sea mas rápida, ademas podamos de aprovechar otras características que diremos mas tarde.

Que necesitamos:

* Un servidor http para esto instalaremos XAMPP el cual nos da un paquete de servidor apache, mysql y FTP se puede descargar desde su pagina oficial:<https://www.apachefriends.org/es/download.html>
* Tener Python instalado (si esta utilizando Linux o Mac estos lo traen ya por defecto) en el caso de window lo pueden descargar desde su web oficial: <https://www.python.org/downloads/>

Para su instalación podemos descargar hacerlo simplemente descargando el archivo .zip y descomprimirlo dentro de htdoc o mediante git si lo tenemos instalado en nuestro computador.

Usando git:

git clone <https://github.com/ZerpaTechnology/AsenZor.git>

También te invitamos a darte un paso por nuestro sitio web: [http://zerpatechnology.com.ve](http://zerpatechnology.com.ve/) donde podrás ver mas de nuestros productos

**FUNCIONAMIENTO DE ASENZOR**

**MODELO MVC**

El modelo MVC es uno de las principales estructuras de desarrollo en los frameworks la cual nos permite separar el trabajo que realiza cada uno de los integrantes del proyecto.

**El modelo:**  Es donde se crea la lógica para el manejo de la base de datos, no esta diseñado específicamente para la creación de la misma ya que en estas normalmente se diseñan por lo general en un programa gestor de tablas SQL y luego el código que nos entrega el programa es cargado en nuestro motor de base de datos que por lo general son: mySQL o SQLite. En el modelo se programa como es el funcionamiento de la inserción, modificación, borrado y consulta de datos en estas bases de datos. Esta área la trabaja el **backend**

**La vista:** Es donde se escribe el HTML de la pagina web, de esta área se encarga principalmente el **frontend** junto a los diseñadoresya que esta sera la interfaz gráfica que el usuario final podrá apreciar, por lo general estos frameworks nos brindan la posibilidad de embeber(incrustar) código del lado del servidor para dinamizar la creación de estas vistas cosa que hace el **backend.**

**El controlador:** Es donde se hace la lógica de como las vistas son servidas y cuales son los datos que desde los modelos se pueden mostrar en las vistas, de esta área se ocupa el **backend.**

Ademas de que para el acceso a los contenidos de se hace atrave de la url: <https://hostDominio/aplicacion/controlador/accion/valor>

**MODELO Z-API**

En la introducción del libro dijimos que Django era un framework de tercera generación y cabe mencionar que AsenZor también o tal vez podría considerarse posterior debido a su funcionamiento que hereda características del MVC en cuanto a sus aplicaciones sin embargo la misma es distribuida en administrador y usuario.

**lanzador:** Todas las aplicaciones son cargadas por el controlador principal del framework el cual llamaremos lanzador de aplicaciones desde allí se proporciona todos los recursos globales que nos da el framework, siendo las carpetas de estilos dentro de las aplicaciones para uso local de las mismas.

**apps:** Es donde estan todas las aplicaciones en fase de producción es decir las que se pueden mostrar al usuario. Estas se descomponen en:

* **admin(**Administrador**)** donde se manejan:
  + los **modelos y bases de datos**
  + las librerías,
  + el registro de operaciones,
  + la configuración de la aplicación
  + los idiomas
  + las **vistas** del dashboard(panel de control) junto con sus estilos.
* **user(**usuario**)** donde se manejan :
  + El **controlador**
  + las **vistas** del usuario junto con sus estilos

**Projects:** Es donde están todas las aplicaciones que aun se encuentran en fase de desarrollo, es decir todavía están en prueba y solo son accesibles mediante el modo debug

El uso de esta estructura permite un facil manejo de las aplicaciones como si de un modulo de codeginiter se tratase solo hace falta colocar la carpeta de la aplicación en el directorio **apps** y esta quedara configurada si es que considera que esta para producción de lo contrario colocarla en **project**  esto también ofrece otro enfoque y es que pueden tomarse como temas al estilo de worpress, todas las apps derivan de las nuevas apps **ejemplo** dentro de project asi que tenga cuidado de no eliminar la carpeta ni modificarla.

Esta forma de estructurar el trabajo nos permite tener mas claridad a la hora de programar y desde el lanzador se pueden revisar todas las url que van a ir al controlador de la aplicación lo que nos permite aplicar los criterios para las url de Z-API los cuales son:

<https://hostDominio/codigo-a-utilzar>

donde codigo-a-utilizar puede ser:

variable1=valor1**;** variable2=valor2

se pueden pasar parámetros por pares clave valor a la aplicación de la forma variable=valor y si es mas de uno los mismos son separados por punto y coma (;) que indica que el comando a finalizado

variable1=valor1,valor2,valor3

También se pueden pasar múltiples valores a una variable la cual generara una lista valores dentro del la variable, estos valores van separados por comas(,)

**metodo:**parametro1,parametro2,parametro3

se puede ejecutar un método que este permitido de la forma nombreDelMetodo seguido de dos puntos (:) y luego los parámetros separados por comas (,).

a**==**v **metodo:** parametro1,parametro2,parametro3**]** igual que

a**!=**v **metodo:** parametro1,parametro2,parametro3**]** diferente de

a**>=**v **metodo:** parametro1,parametro2,parametro3**]** mayor o igual que

a**<=**v **metodo:** parametro1,parametro2,parametro3**]** menor o igual que

a**>**v **metodo:** parametro1,parametro2,parametro3**]** mayor que

a**<**v **metodo:** parametro1,parametro2,parametro3**]** menor que

a **in** v **metodo:** parametro1,parametro2,parametro3**]** a esta en b

a **~in** v **metodo**: parametro1,parametro2,parametro3**]** a no esta en b

a **~in** v **&** b **in** c **metodo**: parametro1,parametro2,parametro3**]** a no esta en b y b esta en c

a **~in** v **|** b **in** c **metodo**: parametro1,parametro2,parametro3**]** a no esta en b o b esta en c

También es posible condicionar eventos de la forma variable1 condicion variable2 codigo para indicar el final del codigo de la funcion de utiliza una corchete de cierre (**]**) no se debe colocar corchete de apertura ya que se sobrentiende con el espacio que hay despues de la ultima variable se asume que la primera ves que se use este bloque lleva por delante un **if** (si) las demas veces un **elif** (si no) y si no lleva una condicion entonces se tomara como un **else** ( de lo contrario) el cual idica su fin con llave de cierre (**}**) el siguiente codigo muestra esto en Z-API y como seria en python

Z-API

a not in v m: p1,p2,p3**]** a in v m2: p1,p3**]** a**==**b m3: p4**;** f=3**}**

PYTHON

if a not in v:

m(p1,p2,p3)

elif a in v:

m2(p1,p3)

elif a==b:

m3(p4)

else:

f=3

**while** a**>=**v **metodo**: parametro1,parametro2,parametro3**}}**

También se pueden crear bucles anteponiendo la palabra reservada **while** la condición y esta marca su fin con llaves de cierre dobles (**}}**)

**for** a **in** v **metodo**: parametro1,parametro2,parametro3**]]**

El bucle **for** se comporta igual al de python y finaliza con corchetes de cierre dobles (**]]**)

**objeto.(objetoMadre , objetoPadre)**  creación de objeto con Herencia múltiple

**objeto.atributo=54** creando atributos para el objeto

**objeto.** **metodo** Asignación de un metodo aun objeto

**objeto.**nuevoMetodo= **metodo** Asignación de un metodo renombrandolo aun objeto

**objeto.** **metodo(** parametro1,parametro2,parametro3) ejecutando metodos del objeto

También es soportado uso de la programación orientada a objetos el uso del punto después de los dos puntos indica que el método no se debe ejecutar.

el código que se puede pasar por la url solo pude ser de máximo 200 caracteres.

Otra estructura de la url seria

<https://hostDominio/nombre-de-la-sesion/codigo-a-utilzar>

donde nombre-de-la-session indica que algo es pasado por un usuario especifico sin la necesidad del uso de los tokens de acceso, para poder usar este modo es necesario que el servidor almacene de forma temporal la direccion ip de la maquina que se esta conectando ademas de un websocket de javascript

**EL NUCLEO**

Para entender un poco mas a fondo como trabaja AsenZor es necesario saber como es que python puede servir HTML a través de CGI, lo que veremos es una representacion muy simple de una vista .py del motor de plantillas que tiene AsenZor.

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

print "Content-type: text/html\n\n"

print “<!DOCTYPE html>"

print “<html>"

print “<head>"

A=”Mi pagina en python”

B=”soy contendio de python”

print “<title>",A ,“<title>"

print “</head>"

print “<body>"

print B

print “</body>"

print “</html>"

Con lo anterior podemos ver un codigo facil de entender, en la primera linea se hace un comentario indicando la direccion del interprete de python para que cgi pueda saber quien ejecuta el codgio, la segunda linea es un comentario que le dice a python que utilizaremos caracteres utf-8 con lo cual podremos escribir con acentos sin tener problemas con el interprete, despues la cuarta linea imprime una cadena de encabezado que le dice a el tipo de archivo que el navegador va a representar, en nuestro caso un html, las lineas siguientes toman todo lo que imprimamos como codigo html, con lo cual podemos ir jugando tal es el caso de las variables A y B esto no permite ir dinamizando el contenido que podemos colocar en nuestra pagina HTM

Para correr el script basta con crear un archivo .htacess dentro de la carpeta publica de nuestro servidor o donde tengamos colocados nuestros archivos cgi, yo estoy usando xampp y mi carpeta publica es htdocs, en el .htaccess colocamos el siguiente codigo.

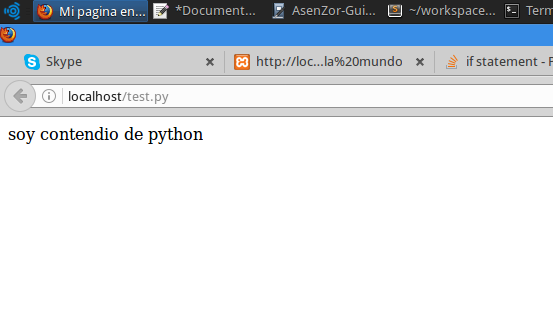
Options +ExecCGI

AddHandler cgi-script .py

con esto le estamos diciendo al servidor que concidere a todos los archivos con extención .py como scripts cgi con lo cual solo bastataria dar permisos de ejecucion a nuestro script de python que acabamos de crear el cual llamaremos index.py, entonces luego de esto si introducimos en el navegador la url:

<https://localhost/index.py>

nos debería ejecutar el script de python sirviéndonos el HTML de las impresiones de consola echas

**LAS VISTAS**

Ahora si se fijaron bien en el código python este imprime el html con el cual podemos ir jugando pero para quienes maquetamos en html esta forma de escribirlo es muy incomoda por lo cual se nos presenta la necesidad de hacerlo de otra forma y es alli donde aparece el codigo embebido que nos dan los frameworks a través de sus motores de plantillas. Veamos un ejemplo de codigo embebido al la forma de PHP:

<!DOCTYPE html>

<html>

<?php include("head.php"); ?>

<body>

<?php include("header.php"); ?>

<section>

</section>

<?php include("footer.php"); ?>

</body>

</html>

Esta forma es mucho mas cómoda ya que mediante de unas etiquetas de apertura y cierre se pueden introducir dentro codigo del lenguaje para PHP la entiqueta de apertura es <?php y la de cierre es ?>

Ahora veamos como seria con python en nuestro framework AsenZor:

<!DOCTYPE html>

<html>

{{incluir("head")}}

<body>

{{incluir("header")}}

<section>

{{}}

</section>

{{incluir("footer")}}

</body>

</html>

En este caso usamos como etiquetas de apertura y cierre llaves dobles ( {{ y }} ) para introducir código python dentro, como python funciona con la tabulación esta solo se puede apreciar en el código que se escriba dentro de estas llaves, para los casos en el que sea necesario hacer esto al interactuar con el html no es necesario hacer la tabulación ya que el motor de plantillas de AsenZor la agrega automáticamente, pero esto hace que tengamos que indicarle al motor de plantillas donde termina la tabulación y para esto empleamos la palabra reservada pass, para mostrar el funcionamiento de un código con estas especificaciones podemos hacerlo de la siguiente manera:

<!DOCTYPE html>

<html>

{{incluir("head")}}

<body>

{{incluir("header")}}

<section>

{{variable=[1,2,3,4,5,6]}}

{{for elem in variable:}}

<h1>{{print elem}}</h1>

{{pass}}

</section>

{{incluir("footer")}}

</body>

</html>

En Django las etiquetas son {% %} en lugar de {{ }} pero el funcionamiento seria casi igual, otra cosa que podemos mencionar esta función incluir que sirve para como su nombre lo indica incluir los widgets que hallamos creado. Para esto veamos como funciona la estructura de las vistas.

Dependiendo del tipo de acceso podemos trabajar las vistas en la carpeta admin/ para el panel de control o en user/ para la interfaz publica de usuario alli tendremos:

**user/**

* **vistas/** (Aquí iran los archivos html bases para los templates, es decir las plantillas y su directorio, ademas del directorio de los widgets)
  + index.html(archivo .html base por defecto)
  + **templates/**(Aquí se crean automaticamente las plantillas desde los archivos base .html)
    - index.py (plantilla generada apartir de index.html)
  + **widgets/**(Aqui se crean y generan automáticamente los widgets, es decir comparten el mismo directorio los .html bases y los widgets .py)
    - head.html (archivo .html base)
    - head.py (archivo .py generado apartir del .html)

En AsenZor se destacan mucho la diferencia existente entre los templates y los widgets el primero es donde se crea la estructura de la pagina y se agrega contenido dinamicamente es por eso que desde ellos se pueden incluir los widgets y estos últimos son objetos html maquetados para poderlos utilizar en cualquier template de modo de ahorranos el tener que crear cada objeto por template esto hace que nuestro trabajo pueda fluir mucho mas rápido, es decir si no nos sirve un pieza simplemente la cambiamos.

Algunas cosas que podemos considerar widgets pueden ser el head, el header, los sidebar, el slider o el hero, el footer, entre otros. La forma que explicamos anteriormente del embebido con llaves dobles se hace es dentro de estos archivos .html y con la función incluir en el caso del los widgets los archivos .py se generan automáticamente desde el motor de plantillas junto con el template al cargar la pagina en sus directorios correspondientes.

**LOS CONTROLADORES**

Como ya sabemos del modelo MVC las vistas se cargan a través de los controladores en AsenZor existe un controlador por defecto para todas las vistas llamado default.py dentro de **user/controles/**