**Archivos de configuración de Canis**

Canis está concebido para realizar aplicaciones te tamaño pequeño a mediano. Fue creado para reducir el tiempo de desarrollo de las aplicaciones ya que tiene incluido una gama de funcionalidades e integraciones con muchas de las mejores tecnologías de desarrollo web.

Hay 4 archivos muy importantes que debemos conocer antes de empezar:

**El fichero binding.xml**

Es el “controlador” del framework, se supone que los vamos construyendo y editando a medida que desarrollamos la aplicación y vamos creando las vistas. En él se especifican todos los detalles que Canis necesita para poder renderizar una vista. No es más que una gran lista de definiciones de “vistas” y los elementos que utilizan, veamos el siguiente archivo bindings como ejemplo:

*<?xml version="1.0"?>*

*<views>*

*<view* ***name="inicio"******url="inicio.php"******class="InicioDelegate"******template=”home”*** */>*

*<view name="registro" url="\_user/registro.php" class="RegistroDelegate">*

***<getter destination="locations">getLocations</getter>***

*<getter destination="roles">getRoles</getter>*

*</view>*

*<view* ***name="account"******url="inicio.php"*** */>*

*</views>*

En este caso hemos definido que nuestra página web está compuesta de 3 vistas: “Inicio” y “registro” y “account”. Además le hemos especificado al frameword la ubicación (url) de estas vistas. Estos dos parámetros son obligatorios para cualquier vista que definamos en este archivo. Los otros parámetros son opcionales pero muy importantes si queremos explotar todas las funcionalidades del framework que de seguro si las conocemos bien, nos harán la vida mucho más fácil.

Para aprender un poco más sobre este archivo puedes leer el artículo “Trabajando con binding.xml”.

**El fichero roles.xml**

Es mucho más simple que binding.xml, se encarga de definir los privilegios de lectura y escritura para cada rol y en cada vista, cada vez que creemos una nueva vista en binding.xml debemos venir a este fichero y darle privilegios según el rol, de lo contrario Canis nos arrojara un error de seguridad al momento de intentar renderizar la vista. Veamos un ejemplo de este archivo:

<?xml version="1.0"?>

<roles>

**<role name='invalid' >**

<view read='true' **edit='true'** delete='true' add='true'> ***inicio*** </view>

<view read='true' edit='true' delete='true' add='true'> *registro* </view>

</role>

<role name='admin' >

<view read='true' **edit='true'** delete='true' add='true'> ***inicio*** </view>

<view read='true' edit='true' delete='true' add='true'> *registro* </view>

<view read='true' edit='true' delete='true' add='true'> *registro* </view>

</role>

</roles>

En el archivo anterior hemos definido dos roles “invalid” y “admin”. El primero de estos roles es obligatorio de definir, ya que es el rol que los usuarios traen por defecto cuando no están registrados en la página.

Para ver una explicación mas avanzada sobre este archivo puedes proceder al artículo “Trabajando con roles.xml”.

**El archivo globals.php**

Este es el archivo de configuración de todo el framework, es muy importante configurarlo antes de empezar a desarrollar la aplicación puesto que muchos de sus parámetros son utilizados en procesos iniciales de Canis, aquí se especifican variables como el nombre de la BD, si se desea trabajar en debug\_mode, si se desean activar los logs de errores, etc. Estas son algunas de las variables:

|  |  |
| --- | --- |
|  |  |
| dbServer | La ruta del servidor de la base de datos, si estás trabajando en un ambiente local como WAMP o XAMP puedes colocar “localhost” |
| dbName | Nombre de la base de datos. |
| dbUser | Usuario de la base de datos |
| dbPassword | Contraseña del usuario de la base de datos |
| LOGIN\_VIEW | Es la vista por defecto a la que el framework te llevara cuando el usuario trate de entrar a un area restringida. |
| PRIVATE\_VIEW | Es la vista por defecto a la que el framework te llevara cuando el usuario se autentique exitosamente dentro del sistema. |
| DEFAULT\_VIEW | Es la vista que cualquier usuario ve cuando entra a tu página por primera vez, se puede pensar en esta vista como un “index” de la pagina. |
| DEFAULT\_PANEL | En lugar de definir una DEFAULT\_VIEW podemos definir un DEFAULT\_PANEL, la diferencia que tienen ambos es que en el primero de ellos se incluirán todos los elementos del template en el cual la vista está inmersa. En el segundo caso solo se renderizara el contenido de la vista y no los elementos del template. |
| debugMode | Este modo habilita funcionalidades importantes para los desarrolladores, se arrojan errores mas explícitos durante la ejecución de la aplicación, no se utiliza memoria cache, etc. |

Estas son solo algunas de las variables que se pueden configurar. Para ver una explicación mas avanzada sobre este archivo puedes proceder al artículo “Trabajando con globals.php”.

**El archivo schema.yml**

Canis utiliza un framework ORM llamado Doctrine para manejar la capa de datos o “modelo” de la aplicación. Doctrine a su vez, utiliza un archivo llamado schema.yml para describir la estructura de datos que va a utilizar en la aplicación. ¿Esto que quiere decir? Bueno que ahora en lugar de crear tablas en lenguaje SQL vamos a utilizar este archivo yml y describir los “objetos” que vamos a utilizar. El siguiente es un ejemplo de archivo schema.yml:

*detect\_relations: true*

*User:*

*columns:*

*username: string*

*password: string*

*contact\_id: integer*

*Contact:*

*columns:*

*first\_name: string*

*last\_name: string*

*phone: string*

*email: string*

*address: string*

En el archivo anterior estamos definiendo dos objetos, “User” y “Contact”. En el primero de ellos decimos que tiene 3 columnas o propiedades llamadas “username”, “passwords” y “contact\_id”. Además, por cuestiones de nomenclatura ya Doctrine sabe que contact\_id es una clave foranea de la tabla Contact.

Bueno para conocer un poco más sobre este archivo te recomiendo leer el artículo “Trabajando con Schema.yml”.