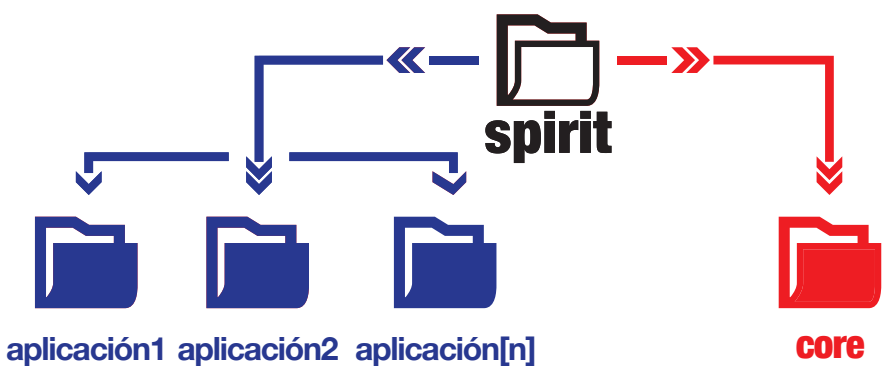
KumbiaPHP Framework

Versión 1.0 **Spirit**

Nueva estructura de directorios

KumbiaPHP Framework

KumbiaPHP Framework en su versión 1.0 incorpora un cambio en la estructura de directorios con la intención de desacoplar el core del framework de nuestra aplicación (app) y ofrecer mayor independencia de las aplicaciones, como se puede apreciar en el gráfico existen dos grandes directorios **app** y **core**.



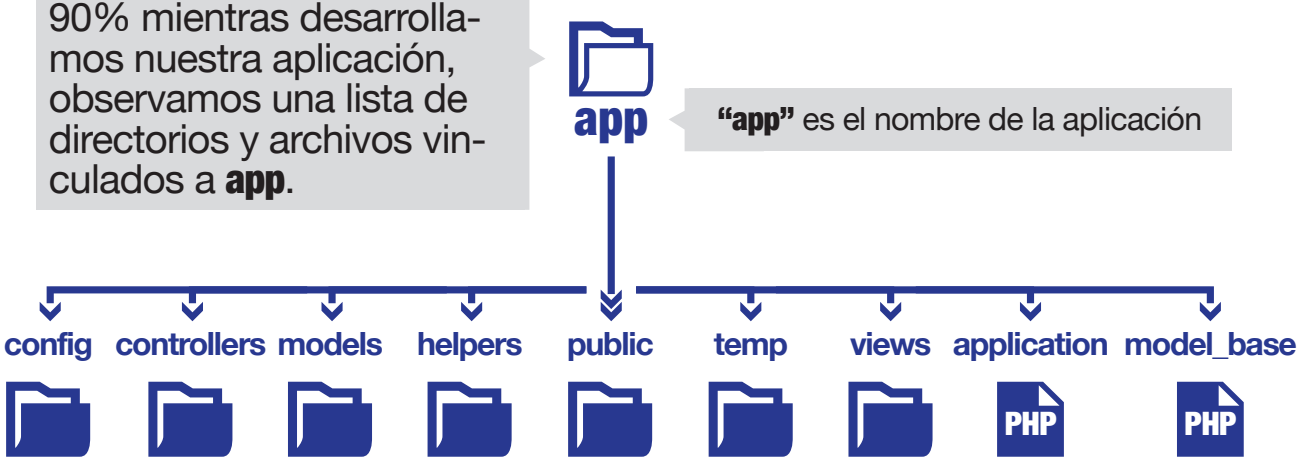
KumbiaPHP Framework

Ventajas de la nueva estructura de directorio

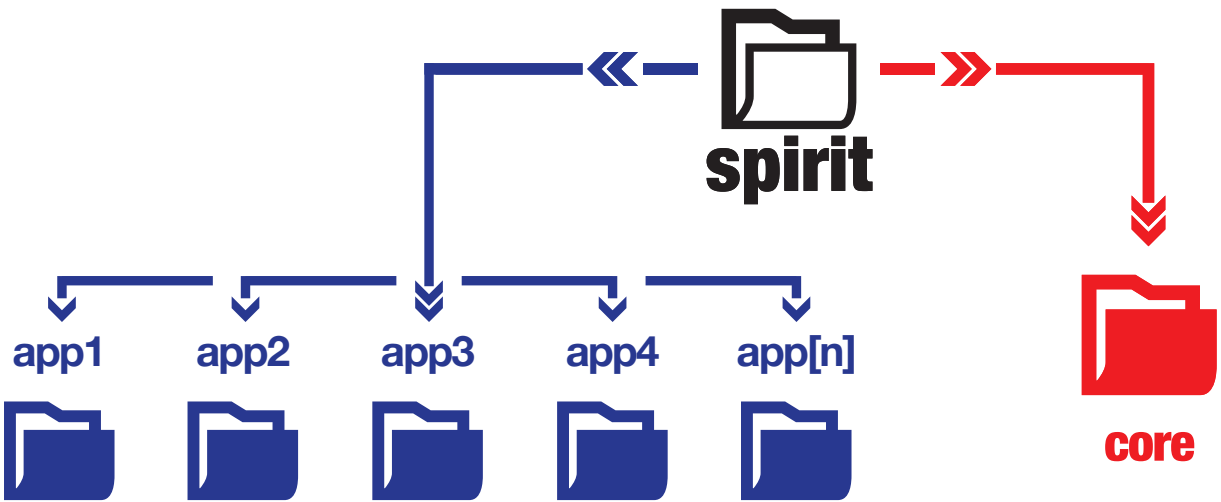
- Mayor velocidad :-)
- Cada aplicación tiene su propio frontcontroller.
- Independencia total de nuestra aplicación respecto al core del framework.
- Cada aplicación tendrá sus propios directorios (public, temp, helpers, etc). En versiones anteriores si se tenia 40 aplicaciones significaba que todo iba al mismo public (css, img, js, etc).
- En cada actualización del framework, sólo se ha de pasar la carpeta de tu aplicación (“app”) a la nueva versión de kumbiaPHP Framework y ya tendremos la última versión del Framework.

KumbiaPHP Framework

Este será el directorio sobre el cual trabajamos el 90% mientras desarrollamos nuestra aplicación, observamos una lista de directorios y archivos vinculados a **app**.



KumbiaPHP Framework



Pueden existir tantas carpetas como aplicaciones se necesiten.

KumbiaPHP Framework

estructura aplicación

config

Archivos de configuración del framework: **config.ini, routes.ini, databases.ini, boot.ini**

controllers

Están agrupados los controladores (controllers) y/o módulos. Por defecto se encuentra el controller: **page_controller.php**

models

Están agrupados los modelos (models)

helpers

Contiene los helpers de usuarios, funciones, librerías, etc, que necesite la aplicación. Normalmente desarrolladas por los usuarios

public

Son los archivos imágenes, css, javascript y files, que utilizara la aplicación

temp

Este directorio contiene las carpetas y archivos creados cuando Kumbia PHP está cacheando un template, view o un partial y cuando se realiza operaciones de logger (logs). **Necesita permisos de escritura.**

views

Están agrupadas las vistas de los controllers. Por defecto están los dir: **template/**, **pages/**, **partials/** y **errors/**

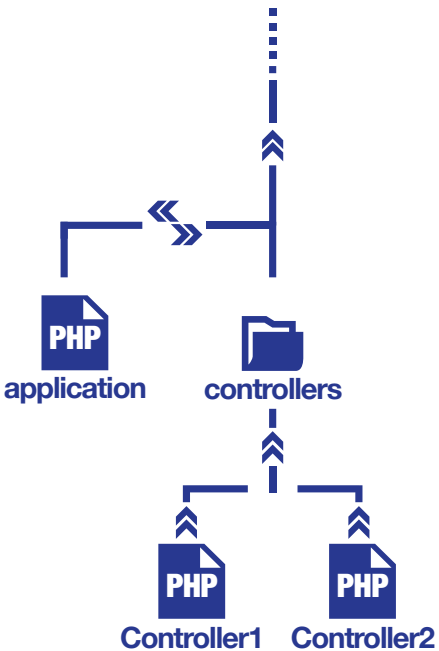
KumbiaPHP Framework

application

model_base

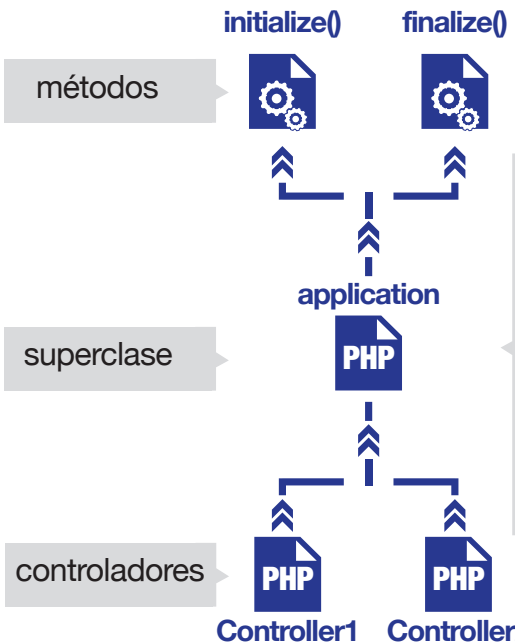
Vamos a profundizar más en estos dos importantisimos y a la vez grandes desconocidos, que nos permitiran agilizar muchisimo el proceso de creación de nuestra aplicación si se usan correctamente.

KumbiaPHP Framework



Cada vez es mas usado en las aplicaciones creadas en PHP la Programación Orientada a Objetos (POO), KumbiaPHP Framework fomenta el uso de la misma, debemos hacer especial hincapié en el archivo **application.php** el cual fue concebido como una superclase (ApplicationController) padre de todos los controladores.

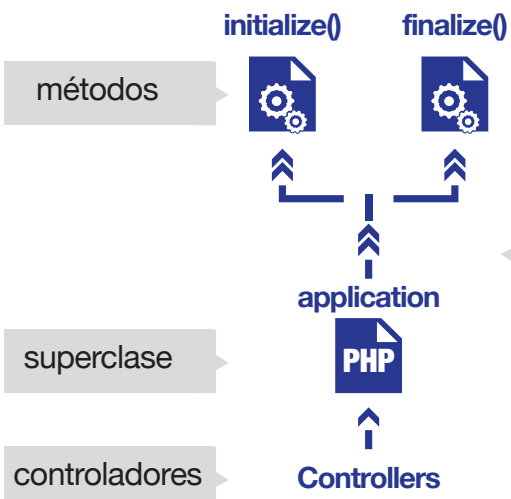
KumbiaPHP Framework



Existe una estrecha relación entre apllication.php y los controladores disponibles en el directorio controllers.

Esta relación se basa principalmente en que las variables y metodos creados en el ApplicationController (application.php), estarán disponibles para ser usados en cualquier controlador que tengamos en nuestra aplicación.

KumbiaPHP Framework

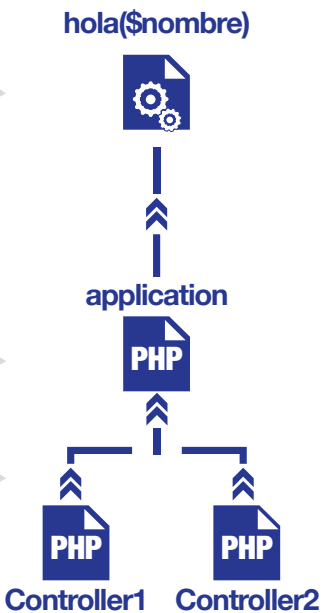


En esta class se encuentran dos métodos principales, dichos métodos se comportan como un filtro.

El método **Initialize()** se ejecuta justo antes de llamar al controller

El método **finalize()** se ejecuta después de haber llamado al controller.

KumbiaPHP Framework



Veamos un ejemplo funcional de lo que podemos llegar a hacer controlando esta superclase; Agregamos el método **hola($nombre)** con un parámetro.

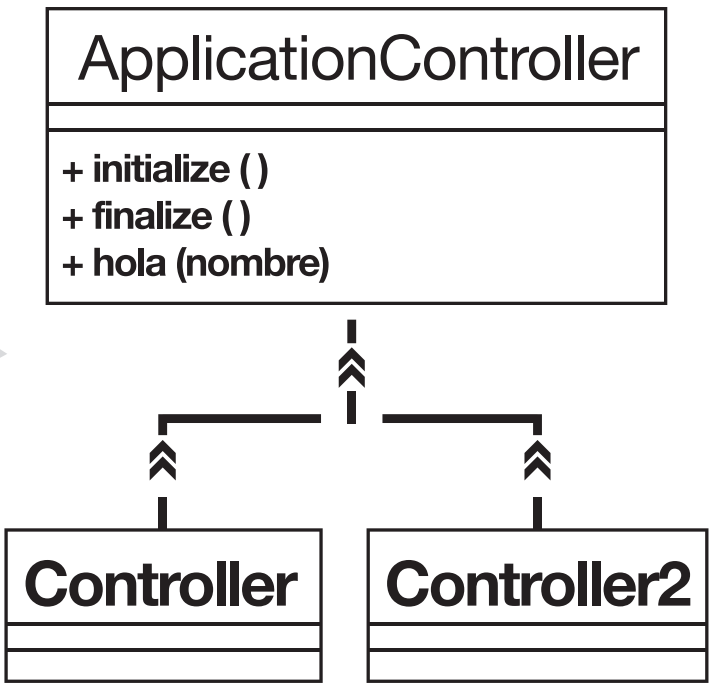
El método **hola($nombre)** estara disponible inmediatamente para todos los controladores.

Podríamos acceder directamente al método **hola($nombre);** de la siguiente manera:

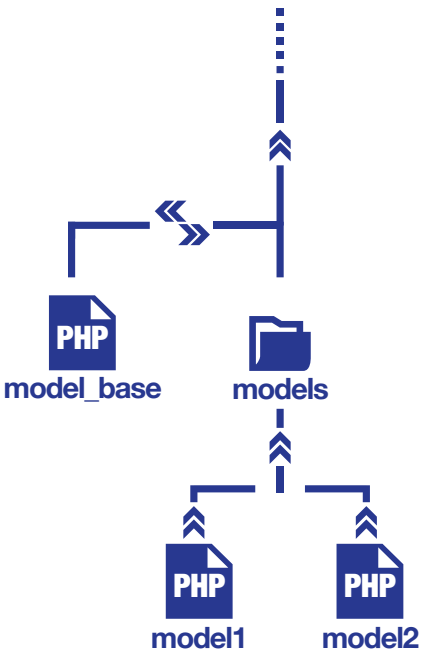
http://dominio.com/controller/hola/jose

http://dominio.com/controller2/hola/jose

Así, la forma en la que se relacionan los controladores con el **ApplicationController** quedaría resumida en el siguiente esquema:



KumbiaPHP Framework



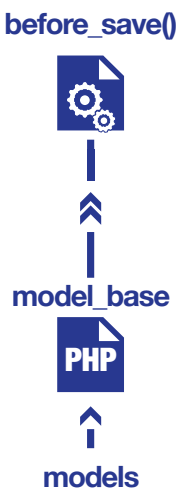
De la misma manera que se relacionan los controladores con application.php, existe otra relación muy importante entre el archivo **model_base.php** y los modelos de objeto **ActiveRecord** ubicados en: **models/*.php** | **models/dir/*.php**

Los modelos aqui alojados representan las tablas de nuestra base de datos.

métodos

superclase **ActiveRecord**

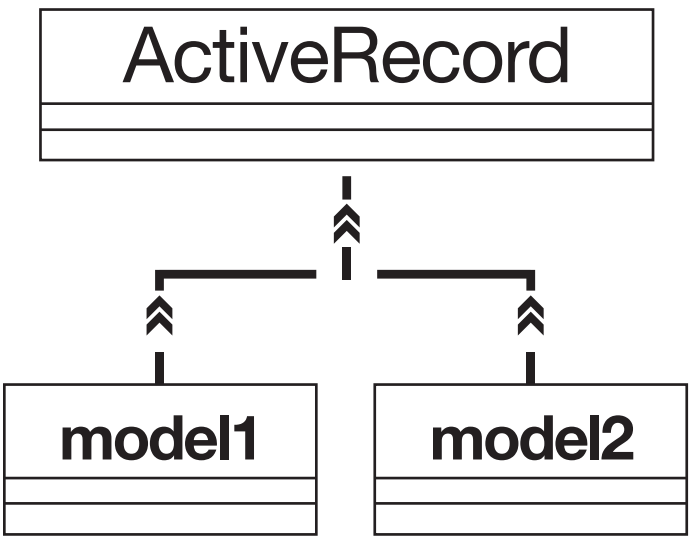
modelos



En la clase ActiveRecord existen una gran cantidad de CallBack, en los que podemos apoyarnos para construir la parte lógica de nuestra aplicación, además los podremos extender con metodos propios.

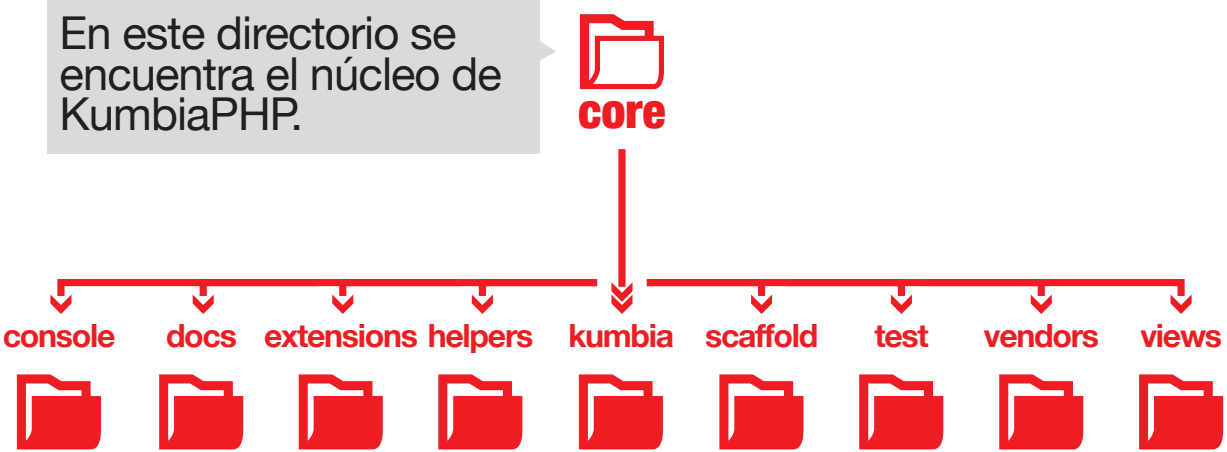
En este ejemplo hemos usado el método **before_save(),** nos permite comprobar lo que necesitemos antes de guardar el registro en la base de datos.

Así, la forma en la que se relacionan los modelos con el **ActiveRecord** quedaría resumida como sigue:



KumbiaPHP Framework

En este directorio se encuentra el núcleo de KumbiaPHP.



KumbiaPHP Framework

estructura core

Script de consola del framework

Aquí están los archivos de licencia y README.

Extensiones de KumbiaPHP, que extiende el core.

Están los Helpers de las vistas.

Core de KumbiaPHP (dispatcher, router, front-controller).

Plantilla para los scaffolds.

Pruebas del framework.

Librerías externas al framework (libchart, fpdf, excel, etc.)

Plantillas de vistas para las excepciones y otras.