CodeIgniter Rocks

Versión 3.1.9

CodeIgniter was created by EllisLab and is now a project of the British Columbia Institute of Technology

Manual en Español

Barcelona, 16 de Junio del 2018

Acerca de...

Hola amigo/a, el no saber inglés me impide leer directamente el manual de CodeIgniter, mi solución ha sido utilizar el traductor de Google, una vez finalizada la traducción, he creído conveniente pasar el manual a .pdf por si puede servir a otro/a.

Quiero dar las gracias al equipo de CodeIgniter por su generosidad al crearlo y mantenerlo, también agradecer a Google por su traductor.

Lector/a le pido indulgencia, lo he repasado, pero igual he cometido errores de transcripción, ante la duda, consulte el Manual Original; me he permitido algunas licencias: me han parecido más legibles los párrafos sin el punto final; faltan acentos en fragmentos de código, en las palabras que forman parte de un URI y pueden aparecer algunas otras cosillas reñidas con la correcta Ortografía y Gramática.

Gracias por su paciencia, saludos desde: Mundo/Europa/España/Cataluña/Barcelona (World/Europe/Spain/Catalonia/Barcelona)

Carlos Valencia Ruiz

Contenido

Bienvenido a CodeIgniter ***33***

Instalación ***34***

Descargar CodeIgniter *34*

Instrucciones de instalación *34*

Solución de problemas *35*

Introducción ***36***

Empezar con CodeIgniter *36*

CodeIgniter de un vistazo *37*

Funcionalidades de CodeIgniter *39*

Diagrama de flujo de aplicaciones *40*

Modelo-Vista-Controlador *41*

Objetivos de Diseño y Arquitectura *42*

Tutorial ***43***

Páginas estáticas *43*

Agregar lógica al controlador *44*

Enrutamiento *45*

Sección de noticias *46*

Configurar su modelo *46*

Mostrar las noticias *47*

Enrutamiento ... 49
Crear noticias ... 50
Crear un formulario ... 50
El modelo ... 51
Enrutamiento ... 52
Conclusión ... 52
Temas generales ... 54
Las URLs de CodeIgniter ... 54
Segmentos de URI ... 54
Eliminar el archivo index.php ... 54
Agregar un sufijo a la URL ... 55
Habilitar la Query string ... 55
Controllers - Controladores ... 57
Qué es un controlador? ... 57
Métodos ... 58
Pasar segmentos de URI a sus métodos ... 59
Definir un controlador predeterminado ... 59
Remapear las llamadas a métodos ... 59
Procesar la salida ... 61
Métodos privados ... 61
Organizar sus controladores en subdirectorios ... 62
Constructores de clase ... 62
Nombres de métodos reservados ... 63
Nombres Reservados ... 64
Nombres de controlador ... 64
Nombres de funciones ... 64
Nombres de variables ... 64
Nombres de constantes ... 65
Views -Vistas ... 66
Crear una vista ... 66
Cargar una vista ... 66
Cargar múltiples vistas ... 67
Almacenar vistas en subdirectorios ... 67
Agregar datos dinámicos a la vista ... 68

Crear bucles ... 69

Devolver vistas como datos ... 70

Models -Modelos ... 71

Qué es un modelo? ... 71

Anatomía de un modelo ... 72

Cargar un modelo ... 72

Cargar un modelo automáticamente ... 73

Conectar a su Base de datos ... 73

Helpers - Ayudantes ... 75

Cargar un Helper ... 75

Cargando múltiples helpers ... 75

Carga automática de helpers ... 76

Usar un helper ... 76

"Extender" los Helpers ... 76

Establecer su propio prefijo ... 77

Usar las Bibliotecas de CodeIgniter ... 78

Crear Bibliotecas ... 79

Almacenar bibliotecas ... 79

Convenciones de nombres ... 79

El archivo de clase ... 79

Usar su clase ... 80

Pasar parámetros al inicializar su clase ... 80

Utilizar recursos CodeIgniter dentro de su biblioteca ... 81

Reemplazar bibliotecas nativas con sus versiones ... 82

Extender bibliotecas nativas ... 83

Cargar su subclase ... 84

Establecer su propio prefijo ... 84

Drivers de CodeIgniter ... 85

Crear drivers ... 86

Directorio de drivers y estructura de archivos ... 86

Crear clases del núcleo ... 87

Lista de las clases del núcleo ... 87

Reemplazar las clases del núcleo ... 87

Extender las clases del núcleo ... 88

Establecer su propio prefijo ... 89
Crear clases auxiliares ... 90
get_instance() ... 90
Hooks - ampliar el núcleo del Framework ... 92
Habilitar hooks ... 92
Definir un hook ... 92
Varias llamadas al mismo hook ... 93
Puntos de hook ... 93
Carga automática de recursos ... 95
Funciones Comunes ... 96
config_item() ... 96
function_usable() ... 96
get_mimes() ... 97
html_escape() ... 97
is_cli() ... 97
is_https() ... 97
is_php() ... 98
is_really_writable() ... 98
remove_invisible_characters() ... 99
set_status_header() ... 99
Funciones de compatibilidad ... 100
Hashing de contraseña ... 100
Constantes ... 100
password_get_info() ... 100
password_hash() ... 100
password_needs_rehash() ... 101
password_verify() ... 101
Hash (Message Digest) ... 102
hash_equals() ... 102
hash_pbkdf2() ... 102
Multibyte String ... 103
mb_strlen() ... 103
mb_strpos() ... 104
mb_substr() ... 104
Funciones estándar ... 104
array_column() ... 105
hex2bin() ... 105
Enrutamiento URI ... 106
Estableciendo sus propias reglas de enrutamiento ... 106

Comodines ... 106
Expresiones regulares ... 107
Callbacks ... 108
Usar verbos HTTP en las rutas ... 108
Rutas Reservadas ... 109
Manejo de errores ... 110
log_message() ... 110
show_404() ... 111
show_error() ... 112
Almacenamiento en caché de la página web ... 113
Cómo funciona el almacenamiento en caché? ... 113
Habilitar el almacenamiento en caché ... 113
Eliminar cachés ... 114
Perfilar la aplicación ... 115
Iniciar la clase ... 115
Activar el perfilador ... 115
Establecer puntos de referencia ... 115
Habilitar y deshabilitar las secciones del perfilador ... 115
CLI - CodeIgniter desde la línea de comandos ... 117
Que es la CLI? ... 117
Por qué ejecutar a través de la línea de comandos? ... 117
Probémoslo: ¡Hola mundo! ... 117
Administrar sus aplicaciones ... 119
Cambiar el nombre del directorio de la aplicación ... 119
Reubicación de su directorio de aplicaciones ... 119
Ejecución de múltiples aplicaciones con una instalación CodeIgniter ... 119
Manejar múltiples entornos ... 121
La constante ENVIRONMENT ... 121
Efectos sobre el comportamiento del Framewok predeterminado ... 121
Reportes de Error ... 121
Archivos de configuración ... 121
Sintaxis alternativa de PHP para archivos de Vistas ... 122
Soporte automático de Short Tags ... 122
"echo" alternativo ... 122
Estructuras de control alternativas ... 122

Seguridad ... 124
Seguridad de URI ... 124
register_globals ... 124
display_errors ... 124
magic_quotes_runtime ... 125
Buenas prácticas ... 125
XSS Filtering ... 125
CSRF protection ... 125
Contraseñas - Passwords ... 125
Validar los datos de entrada ... 126
Escape todos los datos antes del insertarlos en la Base de datos ... 126
Ocultar sus archivos ... 127
Guía de estilo de PHP ... 128
Formato de archivo ... 128
Etiqueta de cierre de PHP ... 129
Nombres de archivos ... 129
Denominación de clase y método ... 129
Nombres de variables ... 130
Comentarios ... 131
Constantes ... 132
TRUE, FALSE y NULL ... 132
Operadores Lógicos ... 132
Comparar valores de retorno y Typecasting ... 133
Código de depuración ... 134
Espacio en blanco en archivos ... 134
Compatibilidad ... 134
Un archivo por clase ... 134
Espacio en blanco ... 134
Saltos de línea ... 134
Indentación de código ... 135
Espaciado entre paréntesis ... 136
Texto localizado ... 137
Métodos privados y variables ... 137
Errores de PHP ... 137
Etiquetas cortas abiertas ... 138
Una sentencia por línea ... 138

Strings ... 139
Consultas SQL ... 139
Argumentos de función predeterminados ... 140
Bibliotecas ... ***141***
Benchmarking Class - clase para Análisis comparativo ... 141
Usar Benchmark Class ... 141
Perfilar sus puntos de evaluación ... 142
Visualizar del tiempo total de ejecución ... 142
Mostrar el consumo de memoria ... 143
class CI_Benchmark ... 143
class CI_Benchmark ... **143**
elapsed_time() ... **143**
mark() ... **144**
memory_usage() ... **144**
Caching Driver - Driver de Caché ... 145
Ejemplo de uso ... 145
class CI_Cache ... 145
class CI_Cache ... **145**
cache_info() ... **145**
clean() ... **146**
decrement() ... **146**
delete() ... **147**
get() ... **147**
get_metadata() ... **147**
increment() ... **148**
is_supported() ... **148**
save() ... **149**
Drivers ... 149
Caché alternativo de PHP (APC) ... 149
Caché basado en archivos ... 150
Caché Memcached ... 150
Caché WinCache ... 150
Caché Redis ... 150
Dummy Cache - Caché simulada ... 151
Calendaring Class - Clase calendario ... 152
Iniciar Calendaring Class ... 152
Mostrar un calendario ... 152
Pasar datos a sus celdas de calendario ... 152

Configurar las preferencias de visualización ... 153
Mostrar enlaces del mes siguiente/anterior ... 154
Crear una plantilla de calendario ... 154
class CI_Calendar ... 156
class CI_Calendar ... 156
adjust_date() ... 156
default_template() ... 156
generate() ... 156
get_day_names() ... 157
get_month_name() ... 157
get_total_days() ... 157
initialize() ... 158
parse_template() ... 158
Config Class - Clase de configuración ... 159
Anatomía de un archivo de configuración ... 159
Cargando un archivo de configuración ... 159
1 - Carga manual ... 160
2 - Carga automática ... 160
Recuperar elementos de configuración ... 160
Establecer un elemento de configuración ... 161
Entornos ... 161
class CI_Config ... 162
class CI_Config ... 162
$config ... 162
$is_loaded ... 162
base_url() ... 162
item() ... 162
load() ... 163
set_item() ... 163
site_url() ... 163
slash_item() ... 164
system_url() ... 164
Correo electrónico ... 165
Enviar correo ... 165
Establecer preferencias ... 165
Preferencias en un archivo de configuración ... 166
Preferencias de correo electrónico ... 166
Word Wrapping ... 167
class CI_Email ... 167
class CI_Email ... 167

attach() ... **167**
attachment_cid() ... **168**
bcc() ... **169**
cc() ... **169**
clear() ... **170**
from() ... **170**
message() ... **171**
print_debugger() ... **171**
reply_to() ... **172**
send() ... **172**
set_alt_message() ... **173**
set_header() ... **173**
subject() ... **174**
to() ... **174**
Encryption Library ... *176*
Iniciar Encryption Library ... *176*
Comportamiento por defecto ... *176*
Configurar la clave de cifrado ... *177*
Cifrados y modos de encriptación admitidos ... *178*
Cifrados portables ... *178*
Cifrados específicos del controlador ... *179*
Modos de encriptación ... *180*
Longitud del mensaje ... *181*
Configurar la biblioteca ... *181*
Cifrado y descifrado de datos ... *182*
Cómo funciona ... *182*
Parámetros personalizados ... *183*
Algoritmos de autenticación HMAC admitidos ... *184*
class CI_Encryption ... *185*
class CI_Encryption ... **185**
create_key() ... **185**
decrypt() ... **185**
encrypt() ... **186**
hkdf() ... **186**
initialize() ... **187**
File Uploading Class ... *188*
El proceso ... *188*
Creando el formulario de carga ... *188*
La página de éxito ... *188*
El controlador ... *189*

El directorio Upload ... *190*
Intentelo ! ... *190*
Iniciar File Upload Class ... *190*
Establecer preferencias ... *190*
Preferencias disponibles ... *191*
Establecer preferencias en un archivo de configuración ... *192*
class CI_Upload ... *193*
class CI_Upload ... **193**
data() ... **193**
display_errors() ... **194**
do_upload() ... **195**
initialize() ... **195**
Form Validation ... *196*
Introducción ... *196*
Tutorial de validación de formularios ... *196*
El formulario ... *197*
La página de éxito ... *197*
El controlador ... *198*
Intentelo! ... *198*
Explicación ... *198*
Establecer reglas de validación ... *199*
Establecer reglas usando un array ... *200*
Reglas en cascada ... *201*
Preparar los datos ... *202*
Volver a llenar el formulario ... *202*
Callbacks: sus propias métodos de validación ... *204*
Callable: use cualquier cosa como una regla ... *205*
Configuración de mensajes de error ... *206*
Traducir los nombres de campo ... *207*
Cambiar los delimitadores de error ... *208*
Mostrar errores individualmente ... *208*
Validar un array (que no sea $_POST) ... *209*
Guardar reglas de validación en un archivo de configuración ... *210*
Cómo guardar sus reglas ... *210*
Crear un grupo de reglas ... *211*
Llamar a un grupo de reglas específicas ... *212*
Asociar un método de controlador con un grupo de reglas ... *212*

Usar arrays como nombres de campo ... 213
Referencia de reglas ... 215
Referencia de preparaciones ... 216
class CI_Form_validation ... 217
class CI_Form_validation ... 217
error() ... 217
error_array() ... 217
error_string() ... 217
has_rule() ... 218
reset_validation() ... 218
run() ... 218
set_data() ... 219
set_error_delimiters() ... 219
set_message() ... 219
set_rules() ... 220
Helper Reference ... 220
FTP Class ... 221
Iniciar FTP Class ... 221
Ejemplos de uso ... 221
class CI_FTP ... 222
class CI_FTP ... 222
changedir() ... 222
chmod() ... 223
close() ... 223
connect() ... 223
Preferencias de FTP en un archivo de configuración ... 224
Opciones de conexión disponibles ... 224
delete_dir() ... 224
delete_file() ... 225
download() ... 225
list_files() ... 226
mirror() ... 226
mkdir() ... 227
move() ... 227
rename() ... 228
upload() ... 228
Image Manipulation Class ... 230
Iniciar Image Manipulation Class ... 230
Procesar una imagen ... 230
Métodos de procesamiento ... 231
Preferencias ... 232

Leyenda de disponibilidad: ... 232
Preferencias en un archivo de configuración ... 234
Marca de agua ... 234
Dos tipos de marca de agua ... 234
Poner marca de agua a una imagen ... 234
Preferencias de marca de agua ... 235
Preferencias para tipo text ... 236
Preferencias para tipo overlay ... 237
class CI_Image_lib ... 237
class CI_Image_lib ... 237
clear() ... 237
crop() ... 238
display_errors() ... 238
initialize() ... 239
resize() ... 239
rotate() ... 240
watermark() ... 241
Input Class ... 242
Filtrado de seguridad ... 242
Filtrado de XSS ... 242
Usar datos POST, GET, COOKIE o SERVER ... 243
Usando php://input stream ... 243
class CI_Input ... 244
class CI_Input ... 244
$raw_input_stream ... 244
cookie() ... 244
get() ... 245
get_post() ... 245
get_request_header() ... 246
input_stream() ... 246
ip_address() ... 247
is_ajax_request() ... 247
is_cli_request() ... 247
method() ... 248
post() ... 248
post_get() ... 249
request_headers() ... 249
server() ... 250
set_cookie() ... 250
user_agent() ... 252
valid_ip() ... 252

Language Class ... 253
Manejo de múltiples idiomas ... 253
Ejemplo de cambio de idioma ... 254
Internacionalización ... 254
Crear archivos de idioma ... 254
Cargar un archivo de idioma ... 255
Obtener una línea de texto ... 255
Usar líneas de idioma como etiquetas de formulario ... 255
Auto carga de idiomas ... 256
class CI_Lang ... 256
class CI_Lang ... 256
line() ... 256
load() ... 256
Loader Class ... 257
Paquetes de aplicación ... 257
Archivos de vista del paquete ... 257
class CI_Loader ... 258
class CI_Loader ... 258
add_package_path() ... 258
clear_vars() ... 258
config() ... 259
database() ... 259
dbforge() ... 259
dbutil() ... 260
driver() ... 260
file() ... 261
get_package_paths() ... 262
get_var() ... 262
get_vars() ... 262
helper() ... 262
is_loaded() ... 263
language() ... 264
library() ... 264
Opciones de configuración ... 265
Asignar una biblioteca a un nombre de objeto diferente ... 265
model() ... 266
remove_package_path() ... 266
vars() ... 267
view() ... 267
Migrations Class ... 269

Nombres de archivos de migración ... 269
Crear una migración ... 269
Preferencias de migración ... 271
class CI_Migration ... 271
class CI_Migration ... 271
current ... 271
error_string() ... 271
find_migrations() ... 272
latest() ... 272
version() ... 272
Output Class ... 273
class CI_Output ... 273
class CI_Output ... 273
$parse_exec_vars ... 273
append_output() ... 273
cache() ... 274
_display() ... 274
enable_profiler ... 275
get_content_type ... 275
get_header() ... 276
get_output() ... 276
set_content_type ... 277
set_header ... 277
set_output() ... 278
set_profiler_sections() ... 278
set_status_header() ... 279
Pagination Class ... 280
Establecer preferencias en un archivo de configuración ... 280
Personalizar la paginación ... 281
Agregar marcado de delimitación ... 282
El enlace "First" - "Primera" ... 282
El enlace "Last" - "Última" ... 282
El enlace "Next" - "Siguiente" ... 283
El enlace "Previous" - "Anterior" ... 283
El enlace "Current Page" - "Página actual" ... 283
El enlace "Digit" - "Dígito" ... 283
Ocultar páginas ... 284
Agregar atributos a los anclajes ... 284
Desactivar el atributo "rel" ... 284
class CI_Pagination ... 284

class CI_Pagination ... **284**
create_links() ... **284**
initialize() ... **285**
Template Parser Class ... *286*
Iniciar Template Parser Class ... *286*
Plantillas de análisis ... *287*
Pares de variables ... *287*
Notas de uso ... *288*
Fragmentos de vistas ... *290*
class CI_Parser ... *291*
class CI_Parser ... **291**
parse() ... **291**
parse_string() ... **291**
set_delimiters() ... **292**
Security Class ... *293*
XSS Filtering - Filtrado de XSS ... *293*
Cross-site request forgery (CSRF) - Falsificación de solicitudes entre sitios (CSRF) ... *293*
class CI_Security ... *294*
class CI_Security ... **294**
entity_decode ... **294**
get_csrf_hash() ... **295**
get_csrf_token_name() ... **295**
get_random_bytes() ... **295**
sanitize_filename() ... **295**
xss_clean() ... **296**
Session Library ... *297*
Iniciar Session Library ... *297*
Cómo funcionan las sesiones? ... *297*
Una nota sobre concurrencia ... *298*
Qué son los datos de sesión? ... *298*
Recuperar datos de sesión ... *299*
Agregar datos de sesión ... *300*
Eliminar datos de sesión ... *300*
Flashdata ... *301*
Tempdata ... *302*
Destruyendo una sesión ... *304*
Acceder a los metadatos de la sesión ... *304*
Preferencias de sesión ... *304*

Controladores de sesión ... 306
Controlador file ... 306
Bonus Tip ... 307
Controlador database ... 307
Controlador Redis ... 309
Controlador Memcached ... 309
Bonus Tip ... 310
Controladores personalizados ... 310
class CI_Session ... 312
class CI_Session ... 312
all_userdata() ... 312
flashdata() ... 312
__get() ... 313
get_flash_keys() ... 313
get_temp_keys() ... 313
&get_userdata() ... 313
has_userdata() ... 313
keep_flashdata() ... 314
mark_as_flash() ... 314
mark_as_temp() ... 314
sess_destroy() ... 314
sess_regenerate() ... 315
__set() ... 315
set_flashdata() ... 316
set_tempdata() ... 316
set_userdata() ... 316
tempdata() ... 316
unmark_flash() ... 316
unmark_temp() ... 317
unset_userdata() ... 317
userdata() ... 317
HTML Table Class ... 318
Iniciar HTML Table Class ... 318
Cambiando el aspecto de su tabla ... 319
class CI_Table ... 320
class CI_Table ... 320
$function ... 320
add_row() ... 321
clear() ... 321
generate() ... 322
make_columns() ... 322
set_caption() ... 323

set_empty() ... **323**
set_heading() ... **324**
set_template() ... **324**
Trackback Class ... *325*
Iniciar Trackback Class ... *325*
Envío de referencias ... *325*
Recibir Trackbacks ... *326*
su Ping URL ... *326*
Crear una tabla de seguimiento ... *326*
Procesando un Trackback ... *327*
Notas: ... *328*
class CI_Trackback ... *328*
class CI_Trackback ... **328**
$data ... **328**
$convert_ascii ... **328**
convert_ascii() ... **329**
convert_xml() ... **329**
data() ... **329**
display_errors() ... **329**
extract_urls() ... **330**
get_id() ... **330**
limit_characters() ... **330**
process() ... **331**
receive() ... **331**
send() ... **331**
send_error() ... **332**
send_success() ... **332**
set_error() ... **332**
validate_url() ... **333**
Typography Class ... *334*
Iniciar Typography Class ... *334*
class CI_Typography ... *334*
class CI_Typography ... **334**
$protect_braced_quotes ... **334**
auto_typography() ... **334**
format_characters() ... **335**
nl2br_except_pre() ... **336**
Unit Testing Class ... *337*
Iniciar Unit Testing Class ... *337*
Ejecutar pruebas ... *337*
Generando Informes ... *338*

Modo estricto ... 338
Habilitar / deshabilitar las pruebas unitarias ... 339
Pantalla de prueba de la unidad ... 339
Personalizar las pruebas mostradas ... 339
Creando una plantilla ... 339
class CI_Unit_test ... 340
class CI_Unit_test ... 340
active() ... 340
report() ... 340
result() ... 340
run() ... 341
set_template() ... 341
set_test_items() ... 341
use_strict() ... 342
URI Class ... 343
class CI_URI ... 343
class CI_URI ... 343
assoc_to_uri() ... 343
rsegment() ... 343
rsegment_array() ... 344
ruri_string() ... 344
ruri_to_assoc() ... 344
segment() ... 345
segment_array() ... 345
slash_rsegment() ... 346
slash_segment() ... 346
total_rsegments() ... 347
total_segments() ... 347
uri_string() ... 347
uri_to_assoc() ... 348
User Agent Class ... 349
Iniciar User Agent Class ... 349
Definiciones de agente de usuario ... 349
class CI_User_agent ... 350
class CI_User_agent ... 350
accept_charset() ... 350
accept_lang() ... 350
agent_string() ... 351
browser() ... 351
charsets() ... 351
languages() ... 352
mobile() ... 352

is_browser() ... **352**
is_mobile() ... **353**
is_referral() ... **353**
is_robot() ... **353**
parse() ... **354**
platform() ... **354**
referrer() ... **354**
robot() ... **355**
version() ... **355**
XML-RPC and XML-RPC Server Classes ... *356*
Qué es XML-RPC? ... *356*
Iniciar XML-RPC y XML-RPCS ... *356*
Envío de solicitudes XML-RPC ... *356*
Explicación ... *357*
Anatomía de una solicitud ... *357*
Crear un servidor XML-RPC ... *358*
Procesamiento de solicitudes del servidor ... *358*
Notas: ... *360*
Formatear una respuesta ... *360*
Envío de una respuesta de error ... *361*
Creando su propio cliente y servidor ... *361*
El cliente ... *361*
El servidor ... *362*
Intentelo ! ... *362*
Uso de arrays asociativos en un parámetro de solicitud ... *363*
Tipos de datos ... *363*
class CI_Xmlrpc ... *364*
class CI_Xmlrpc ... **364**
display_error() ... **364**
display_response() ... **364**
initialize() ... **364**
method() ... **365**
request() ... **365**
send_error_message() ... **365**
send_request() ... **366**
server() ... **366**
timeout() ... **367**
Zip Encoding Class ... *368*
Iniciar Zip Encoding Class ... *368*
Ejemplo de uso ... *368*

class CI_Zip ... 368

class CI_Zip ... 368
$compression_level ... 368
add_data() ... 369
add_dir() ... 370
archive() ... 370
clear_data() ... 370
download() ... 371
get_zip() ... 371
read_dir() ... 372
read_file() ... 373

Bases de datos ... 374

Inicio rápido: código de ejemplo ... 374

Iniciar la clase Base de datos ... 374
Consulta estándar con resultados múltiples (versión objeto) ... 374
Consulta estándar con resultados múltiples (versión array) ... 374
Consulta estándar con resultado único ... 375
Consulta estándar con resultado único (versión array) ... 375
Inserción estándar ... 375
Consulta con Query Builder ... 375
Insertar con Query Builder ... 376

Configuración de la Base de datos ... 377

Query Builder ... 379
Explicación de valores de configuración ... 380

Conectar con su Base de datos ... 382

Conexión automática ... 382
Conexión manual ... 382
Parámetros disponibles ... 382
Conexión manual a una Base de datos ... 382
Conexión a múltiples Bases de datos ... 383
Reconectar / mantener la conexión activa ... 384
Cerrar manualmente la conexión ... 384

Queries - Consultas ... 385

Consultas regulares ... 385
Consultas simplificadas ... 385
Trabajar con prefijos de Base de datos manualmente ... 385
Protección de identificadores ... 386

Escapar consultas ... 386
Enlazado de consultas ... 387
Manejar errores ... 387
Arrays de resultados ... 388
Filas de resultados ... 389
Objetos de resultado personalizados ... 390
Métodos de ayuda de resultados ... 392
class CI_DB_result ... 394
class CI_DB_result ... 394
custom_result_object() ... 394
custom_row_object() ... 394
data_seek() ... 394
field_data() ... 395
first_row() ... 395
free_result() ... 395
last_row() ... 395
list_fields() ... 396
next_row() ... 396
num_fields() ... 396
num_rows() ... 396
previous_row() ... 397
result() ... 397
result_array() ... 397
result_object() ... 397
row() ... 398
row_array() ... 398
row_object() ... 398
set_row() ... 399
unbuffered_row() ... 399
Métodos Query Helper ... 400
Información de la ejecución de una consulta ... 400
Información sobre su Base de datos ... 400
Haciendo tus consultas más fáciles ... 401
Query Builder Class ... 403
Seleccionar datos ... 403
Buscando datos específicos ... 407
Buscando datos similares ... 409
Ordenar los resultados ... 411
Limitar o contar los resultados ... 412
Agrupar consultas ... 413

Insertar datos ... 414
Actualización de datos ... 416
Eliminar datos ... 420
Method chaining ... 421
Caché en Query Builder ... 421
Un ejemplo de almacenamiento en caché ... 422
Restablecer el generador de consultas ... 422
class CI_DB_query_builder ... 423
class CI_DB_query_builder ... 423
count_all_results() ... 423
dbprefix() ... 424
delete() ... 424
distinct() ... 424
empty_table() ... 425
flush_cache() ... 425
from() ... 425
get() ... 426
get_compiled_delete() ... 426
get_compiled_insert() ... 426
get_compiled_select() ... 427
get_compiled_update() ... 427
get_where() ... 427
group_by() ... 428
group_end() ... 428
group_start() ... 428
having() ... 428
insert() ... 429
insert_batch() ... 429
join() ... 430
like() ... 430
limit() ... 431
not_group_start() ... 431
not_like() ... 431
offset() ... 432
order_by() ... 432
or_group_start() ... 432
or_having() ... 432
or_like() ... 433
or_not_group_start() ... 433
or_not_like() ... 434
or_where() ... 434
or_where_in() ... 434
or_where_not_in() ... 435

replace() ... **435**
reset_query() ... **436**
select() ... **436**
select_avg() ... **436**
select_max() ... **437**
select_min() ... **437**
select_sum() ... **437**
set() ... **438**
set_dbprefix() ... **438**
set_insert_batch() ... **438**
set_update_batch() ... **439**
start_cache() ... **439**
stop_cache() ... **439**
truncate() ... **439**
update() ... **440**
update_batch() ... **440**
where() ... **441**
where_in() ... **441**
where_not_in() ... **442**
Transacciones ... *443*
Enfoque de CodeIgniter a las transacciones ... *443*
Ejecutando Transacciones ... *443*
Modo estricto ... *443*
Gestión de errores ... *444*
Deshabilitar transacciones ... *444*
Modo de prueba ... *444*
Ejecutando Transacciones Manualmente ... *444*
Obtener Metadatos ... *446*
Metadatos de Tablas ... *446*
Enumera las tablas en su Base de datos ... *446*
Determinar si existe una tabla ... *446*
Campo metadata ... *446*
Listar los campos en una tabla ... *446*
Determinar si un campo está presente en una tabla ... *447*
Recuperar metadatos de campo ... *447*
Llamadas a funciones personalizadas ... *449*
$this->db->call_function(); ... *449*
Clase de caché para la Base de datos ... *450*
Habilitar el almacenamiento en caché ... *450*

Cómo funciona el almacenamiento en caché? ... 450
Caching mejorará el rendimiento de su sitio? ... 451
Cómo se almacenan los archivos de caché? ... 451
Administrar sus archivos de caché ... 451
Referencia de funciones ... 452
Database Forge Class - Administrar la Base de Datos ... 453
Iniciar la clase Forge ... 453
Crear y eliminar Bases de datos ... 453
Crear y borrar tablas ... 454
Agregar campos ... 454
Pasando strings como campos ... 455
Crear un campo ID ... 455
Agregar índices ... 456
Crear una tabla ... 456
Borrar una tabla ... 457
Renombrar una tabla ... 457
Modificar tablas ... 457
Agregar una columna a una tabla ... 457
Borrar una columna de la tabla ... 457
Modificar una columna de la tabla ... 458
class CI_DB_forge ... 458
class CI_DB_forge ... 458
add_column() ... 458
add_field() ... 459
add_key() ... 459
create_database() ... 459
create_table() ... 460
drop_column() ... 460
drop_database() ... 460
drop_table() ... 461
modify_column() ... 461
rename_table() ... 461
Clase de utilidades para Base de datos ... 463
Iniciar la Clase de Utilidad ... 463
Uso de las Utilidades ... 463
Obtener los nombres de las Bases de datos ... 463
Determinar si existe una Base de datos ... 464
Optimizar una tabla ... 464

Reparar una tabla ... *464*
Optimizar la Base de datos ... *465*
Exportar un resultado de consulta como un archivo CSV ... *465*
Exportar un resultado de consulta como un documento XML ... *466*
Copia de seguridad de la Base de datos ... *466*
Notas de la Copia de seguridad ... *466*
Establecer preferencias de copia de seguridad ... *467*
Descripción de preferencias de la Copia de seguridad ... *467*
class CI_DB_utility ... *468*
class CI_DB_utility ... **468**
backup() ... **468**
csv_from_result() ... **468**
database_exists() ... **469**
list_databases() ... **469**
optimize_database() ... **469**
optimize_table() ... **469**
repair_table() ... **470**
xml_from_result() ... **470**
Referencia del controlador DB ... *471*
class CI_DB_driver ... *471*
class CI_DB_driver ... **471**
affected_rows() ... **471**
cache_delete() ... **471**
cache_delete_all() ... **472**
cache_off() ... **472**
cache_on() ... **472**
cache_set_path() ... **472**
call_function() ... **472**
close() ... **473**
compile_binds() ... **473**
count_all() ... **473**
db_connect() ... **474**
db_pconnect() ... **474**
db_select() ... **474**
db_set_charset() ... **475**
display_error() ... **475**
elapsed_time() ... **475**
escape() ... **476**
escape_identifiers() ... **476**
escape_like_str() ... **476**
escape_str() ... **477**
field_data() ... **477**

field_exists() ... **477**
initialize() ... **478**
insert_string() ... **478**
is_write_type() ... **478**
last_query() ... **479**
list_fields() ... **479**
list_tables() ... **479**
platform() ... **479**
primary() ... **480**
protect_identifiers() ... **480**
query() ... **481**
reconnect() ... **481**
simple_query() ... **481**
table_exists() ... **482**
total_queries() ... **482**
trans_complete() ... **482**
trans_off() ... **482**
trans_start() ... **483**
trans_status() ... **483**
trans_strict() ... **483**
update_string() ... **483**
version() ... **484**

Helpers - Ayudantes ... ***485***

Array Helper ... *485*

Cargar Array Helper ... *485*

Funciones ... *485*

element() ... **485**
elements() ... **486**
random_element() ... **487**

CAPTCHA Helper ... *488*

Cargar CAPTCHA Helper ... *488*

Usar CAPTCHA Helper ... *488*

Agregar una Base de datos ... *489*

Funciones ... *490*

create_captcha() ... **490**

Cookie Helper ... *492*

Cargar Cookie Helper ... *492*

Funciones ... *492*

delete_cookie() ... **492**
get_cookie() ... **493**
set_cookie() ... **493**

Date Helper ... 494
Cargar Date Helper ... 494
Funciones ... 494
date_range() ... 494
days_in_month() ... 495
gmt_to_local() ... 495
human_to_unix() ... 496
local_to_gmt() ... 496
mdate() ... 497
mysql_to_unix() ... 497
nice_date() ... 498
now() ... 498
standard_date() ... 498
timespan() ... 498
timezones() ... 499
timezone_menu() ... 500
unix_to_human() ... 500
Referencia de zona horaria ... 501
Directory Helper ... 503
Cargar Directory Helper ... 503
Funciones ... 503
directory_map() ... 503
Download Helper ... 505
Cargar Download Helper ... 505
Funciones ... 505
force_download() ... 505
File Helper ... 506
Cargar File Helper ... 506
Funciones ... 506
delete_files() ... 506
get_dir_file_info() ... 507
get_file_info() ... 507
get_filenames() ... 507
get_mime_by_extension() ... 508
octal_permissions() ... 508
read_file() ... 509
symbolic_permissions() ... 509
write_file() ... 509
Form Helper ... 511
Cargar Form Helper ... 511

Escapar valores de campo ... 511
Funciones ... 511
form_button() ... 511
form_checkbox() ... 512
form_close() ... 514
form_dropdown() ... 514
form_error() ... 516
form_fieldset() ... 516
form_fieldset_close() ... 517
form_hidden() ... 518
form_input() ... 519
form_label() ... 521
form_multiselect() ... 521
form_open() ... 522
form_open_multipart() ... 523
form_password() ... 524
form_prep() ... 524
form_radio() ... 524
form_reset() ... 525
form_submit() ... 525
form_textarea() ... 526
form_upload() ... 526
set_checkbox() ... 526
set_radio() ... 527
set_select() ... 528
set_value() ... 528
validation_errors() ... 529
HTML Helper ... 530
Cargar HTML Helper ... 530
Funciones ... 530
br() ... 530
doctype() ... 530
heading() ... 532
img() ... 532
link_tag() ... 533
meta() ... 534
nbs() ... 536
ol() ... 536
ul() ... 536
Inflector Helper ... 540
Cargar Inflector Helper ... 540
Funciones ... 540
camelize() ... 540

humanize() ... **540**
is_countable() ... **541**
plural() ... **541**
singular() ... **541**
underscore() ... **542**
Language Helper ... *543*
Cargar Language Helper ... *543*
Funciones ... *543*
lang() ... **543**
Number Helper ... *544*
Cargar Number Helper ... *544*
Funciones ... *544*
byte_format() ... **544**
Path Helper ... *545*
Cargar Path Helper ... *545*
Funciones ... *545*
set_realpath() ... **545**
Security Helper ... *547*
Cargar Security Helper ... *547*
Funciones ... *547*
do_hash() ... **547**
encode_php_tags() ... **547**
sanitize_filename() ... **548**
strip_image_tags() ... **548**
xss_clean() ... **548**
String Helper ... *550*
Cargar String Helper ... *550*
Funciones ... *550*
alternator() ... **550**
increment_string() ... **551**
quotes_to_entities() ... **551**
random_string() ... **552**
reduce_double_slashes() ... **552**
reduce_multiples() ... **553**
repeater() ... **553**
strip_quotes() ... **554**
strip_slashes() ... **554**
trim_slashes() ... **555**
Text Helper ... *556*

Cargar Text Helper ... 556
Funciones ... 556
ascii_to_entities() ... 556
character_limiter() ... 556
convert_accented_characters() ... 557
ellipsize() ... 558
highlight_code() ... 558
highlight_phrase() ... 559
word_censor() ... 560
word_limiter() ... 560
word_wrap() ... 561
Typography Helper ... 562
Cargar Typography Helper ... 562
Funciones ... 562
auto_typography() ... 562
entity_decode() ... 562
nl2br_except_pre() ... 563
URL Helper ... 564
Cargar URL Helper ... 564
Funciones ... 564
anchor() ... 564
anchor_popup() ... 565
auto_link() ... 566
base_url() ... 567
current_url() ... 567
index_page() ... 568
mailto() ... 568
prep_url() ... 569
redirect() ... 569
safe_mailto() ... 570
site_url() ... 570
uri_string() ... 571
url_title() ... 572
XML Helper ... 573
Cargar XML Helper ... 573
Funciones ... 573
xml_convert() ... 573

Bienvenido a CodeIgniter

CodeIgniter es un Framework, un marco de desarrollo de aplicaciones, un conjunto de herramientas, para crear sitios web utilizando PHP

Su objetivo es permitirle desarrollar proyectos mucho más rápido de lo que podría hacerlo si estuviera escribiendo código desde cero, proporciona un amplio conjunto de bibliotecas para las tareas más comunes, así como una interfaz simple y una estructura lógica para acceder a estas bibliotecas

CodeIgniter le permite centrarse creativamente en su proyecto al minimizar la cantidad de código necesario para una tarea determinada

CodeIgniter es adecuado para usted si:

- Quiere un framework ligero
- Necesita un rendimiento excepcional
- Quiere compatibilidad con cuentas de hosting estándar que ejecutan una variedad de versiones y configuraciones de PHP
- Necesita un framework que requiera una configuración mínima
- Quiere un framework que no requiera el uso de la línea de comando
- Desea un framework que no requiera reglas de codificación restrictivas
- No está interesado en bibliotecas monolíticas de gran tamaño como PEAR
- No quiere verse forzado a aprender un lenguaje de plantillas (aunque, si lo desea, dispones de un analizador de plantillas propio)
- Busca soluciones simples, lejos de la complejidad
- Quiere una documentación clara y completa

Instalación

Descargar CodeIgniter

La última versión e instrucciones de actualización desde versiones anteriores las puede encontrar en la web de CodeIgniter:

```
https://codeigniter.com/
```

Instrucciones de instalación

CodeIgniter se instala en cuatro pasos:

1. Descomprima el paquete
2. Suba las carpetas y los archivos de CodeIgniter al servidor, normalmente, el archivo index.php estará en su raíz
3. Abra el archivo **application/config/config.php** con un editor de texto y configure la URL base, si tiene la intención de utilizar el cifrado o las sesiones, configure su clave de cifrado
4. Si va a utilizar una base de datos, abra el archivo **application/config/database.php** con un editor de texto y establezca la configuración de la base de datos

Si quiere aumentar la seguridad, puede ocultar la ubicación de los archivos de CodeIgniter renombrando las carpetas **system** y **application**, si lo hace, tiene que abrir el archivo **index.php** principal y establecer las variables **$system_path** y **$application_folder** con los nuevos nombres

Para mejorar la seguridad, tanto el sistema como las carpetas de las aplicaciones deben colocarse por encima de la raíz web para evitar que sean accesibles directamente con un navegador

De forma predeterminada, se incluye un archivo .**htaccess** en cada carpeta para evitar el acceso directo, pero es mejor sacar las carpetas del acceso público por si cambia la configuración del servidor web o no soporta los archivos .**htaccess**

Si quiere mantener sus **views** (vistas) públicas, también es posible mover la carpeta **views** fuera de la carpeta de la aplicación

Después de moverlos, abra el archivo **index.php** principal y establezca las variables **$system_path**, **$application_folder** y **$view_folder**, preferiblemente con la ruta completa

Ejemplo

```
/www/mi_usuario/system
```

Una medida adicional en producción es desactivar el informe de errores PHP y cualquier otra funcionalidad utilizada en el desarrollo, en CodeIgniter puede hacerlo mediante la constante **ENVIRONMENT**

Solución de problemas

Si, independientemente de lo que coloque en su URL, solo se está cargando la página predeterminada, es posible que el servidor no soporte la variable **REQUEST_URI** necesaria para utilizar URLs compatible con los motores de búsqueda

Primero, abra el archivo **application/config/config.php** y busque la variable **$config['uri_protocol']**, vera una serie de ajustes alternativos para probar, si sigue sin funcionar, debe forzar a CodeIgniter a agregar un signo de interrogación a las URLs, para hacerlo, abra el archivo **application/config/config.php,** busque la variable **$config['index_page']** y cambie:

```
$config['index_page'] = "index.php";
```

por:

```
$config['index_page'] = "index.php?";
```

Introducción

Empezar con CodeIgniter

Toda aplicación de software requiere algún esfuerzo para aprender, hemos hecho todo lo posible para minimizar la curva de aprendizaje y hacer que el proceso sea lo más agradable posible

El primer paso es instalar CodeIgniter, a continuación, lea cada uno de los temas de esta sección en orden, cada tema se basa en el anterior, e incluye ejemplos de código que se recomienda probar

Una vez que comprenda los conceptos básicos, estará listo para explorar las Bibliotecas (Libraries) y Ayudantes (Helpers) para aprender a utilizarlas

Si tiene preguntas o problemas, siéntase libre de aprovechar los foros de la comunidad:

http://forum.codeigniter.com/

y de consultar la Wiki:

https://github.com/bcit-ci/CodeIgniter/wiki

podrá ver ejemplos de código publicados por otros usuarios

CodeIgniter de un vistazo

CodeIgniter es un Framework para Aplicaciones

CodeIgniter es un conjunto de herramientas para personas que crean aplicaciones web utilizando PHP, su objetivo es permitirle desarrollar proyectos mucho más rápido de lo que podría hacerlo si estuvieras escribiendo código desde cero

Le proporciona un amplio conjunto de bibliotecas para las tareas más comunes, así como una interfaz simple y una estructura lógica para acceder a esas bibliotecas, CodeIgniter le permite centrarse creativamente en su proyecto al minimizar la cantidad de código necesario para una tarea determinada

CodeIgniter es gratis

CodeIgniter está licenciado bajo la licencia de **MIT** para que pueda usarlo como quiera, para obtener más información, consulte el acuerdo de licencia en : **https://www.codeigniter.com/user_guide/license.html**

CodeIgniter es ligero

Verdaderamente ligero, el sistema central requiere solo unas pocas bibliotecas muy pequeñas, esto contrasta con otros frameworks que requieren significativamente más recursos

Las bibliotecas se cargan dinámicamente bajo petición, en función de sus necesidades para un proceso determinado, por lo que el sistema base es muy ligero y bastante rápido

CodeIgniter es rápido

Realmente rápido, le retamos a que encuentres un framework que tenga un rendimiento mejor que CodeIgniter

CodeIgniter utiliza M-V-C

CodeIgniter utiliza el enfoque Modelo-Vista-Controlador, que permite la separación entre la lógica y la presentación, esto es particularmente bueno para proyectos en los que los diseñadores trabajan con sus archivos de plantilla, ya que el código que contienen estos archivos se minimizará, describimos **MVC** con más detalle más adelante

CodeIgniter genera URL limpias

Las URL generadas por CodeIgniter son limpias y amigables para los motores de búsqueda, en lugar de utilizar el enfoque estándar de **"query string"** característico de sistemas dinámicos, CodeIgniter utiliza un enfoque basado en segmentos:

```
ejemplo.com/noticias/articulo/345
```

Nota

De forma predeterminada, el archivo **index.php** se incluye en la URL, pero se puede eliminar utilizando un simple archivo **.htaccess**

CodeIgniter trae soluciones

CodeIgniter incluye una gama completa de bibliotecas que permiten las tareas de desarrollo web más comunes, como acceder a una Base de datos, enviar correos electrónicos, validar datos de formularios, mantener sesiones, trabajar con datos XML-RPC, manipular imágenes y mucho más

CodeIgniter es Extensible

El sistema se puede ampliar fácilmente mediante el uso de sus propias bibliotecas, ayudantes, extensiones de clase y hooks

CodeIgniter no requiere un motor de plantillas

Aunque CodeIgniter viene con un analizador de plantillas que puede usar opcionalmente, no le obliga a usar uno, los motores de plantillas simplemente no pueden igualar el rendimiento del PHP nativo, y la sintaxis que se debe aprender para usar un motor de plantillas generalmente es solo un poco más fácil que aprender los principios básicos de PHP

Considere este bloque de código PHP:

```
<ul>
  <?php foreach ($agenda as $nombre):?>
    <li><?=$nombre?></li>
  <?php endforeach; ?>
</ul>
```

Compárelo con el pseudo-código utilizado por un motor de plantillas:

```
<ul>
  {foreach from=$agenda item="nombre"}
    <li>{$nombre}</li>
  {/foreach}
</ul>
```

El ejemplo del motor de plantillas es un poco más limpio, pero tiene el precio del rendimiento, ya que el pseudocódigo para ejecutarse debe convertirse en PHP., como uno de nuestros objetivos es el máximo rendimiento, optamos por no requerir el uso de un motor de plantillas

CodeIgniter está completamente documentado

Los programadores adoran codificar y detestan escribir documentación, no somos diferentes, por supuesto, pero como la documentación es tan importante como el código en sí, nos comprometemos a hacerlo, nuestro código fuente es extremadamente limpio y bien comentado

CodeIgniter tiene una comunidad amigable de usuarios

Puede comprobar la creciente comunidad de usuarios participando activamente en nuestros Foros de la comunidad en:

http://forum.codeigniter.com/

Funcionalidades de CodeIgniter

Las funcionalidades en sí misma es una forma muy pobre de juzgar una aplicación, ya que no te dicen nada acerca de la experiencia del usuario, ni si está diseñada intuitiva e inteligentemente, las funcionalidades no revelan nada sobre la calidad del código, el rendimiento, la atención al detalle o las prácticas de seguridad

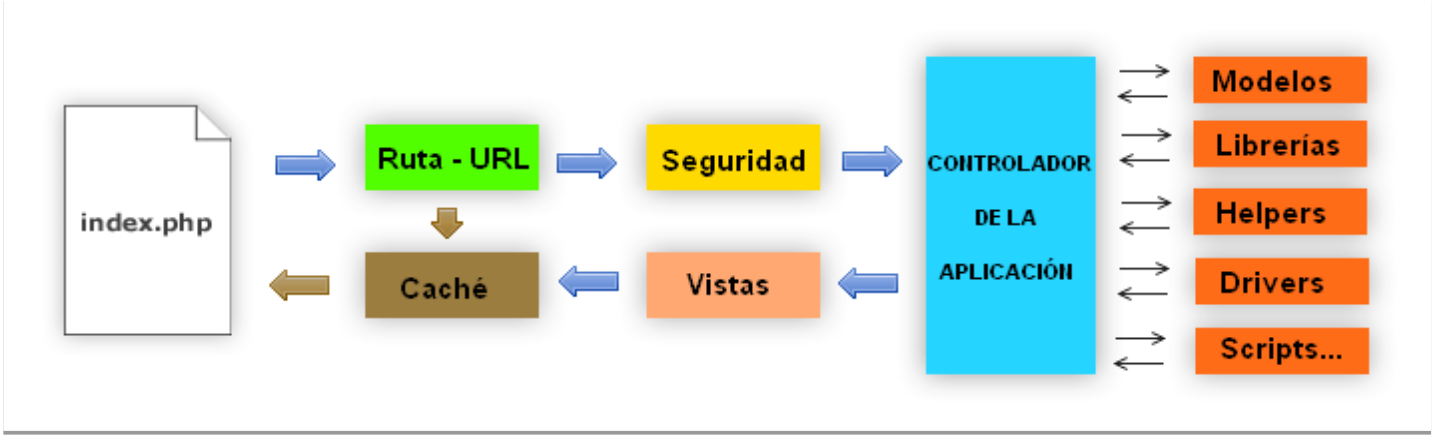
La única forma de juzgar realmente una aplicación es probarla y conocer su código, instalar CodeIgniter es muy sencillo, así que le recomendamos que lo haga

Aquí hay una lista de las características principales de CodeIgniter:

- Sistema basado en Modelo-Vista-Controlador
- Extremadamente ligero
- Clases de bases de datos completas con soporte para varias plataformas
- Soporte de la base de datos de Query Builder
- Validación de datos y formularios
- Seguridad y filtrado XSS
- Gestión de sesiones
- Clase de envío de correo electrónico. Admite archivos adjuntos, correo electrónico HTML/de texto, múltiples protocolos (sendmail, SMTP y Mail) y más
- Biblioteca de manipulación de imágenes (recorte, redimensión, rotación, etc.). Admite GD, ImageMagick y NetPBM
- Clase para subir archivos
- Clase de FTP
- Localización
- Paginación
- Cifrado de datos
- Benchmarking - Evaluación de rendimiento
- Caché de página completa
- Registro de errores
- Perfilado de aplicaciones
- Clase de calendario
- User Agent Class
- Clase de codificación zip
- Clase de motor de plantilla
- Clase Trackback
- Biblioteca XML-RPC
- Clase par pruebas Unit
- URL amigables para motores de búsqueda
- Enrutamiento URI flexible
- Soporte para hooks y extensiones de clase
- Una extensa biblioteca de funciones Helper (Ayudantes)

Diagrama de flujo de aplicaciones

El siguiente gráfico ilustra cómo fluyen los datos en el sistema:



Flujo de aplicación CodeIgniter

1. **Index.php** sirve como controlador frontal, iniciando los recursos base necesarios para ejecutar CodeIgniter
2. El enrutador examina la solicitud HTTP para determinar qué se debe hacer con ella
3. Si existe un archivo de caché, se envía directamente al navegador, evitando la ejecución normal del sistema
4. Seguridad, antes de que se cargue el controlador de la aplicación, la solicitud HTTP y los datos enviados por el usuario se filtran para mayor seguridad
5. El controlador carga el modelo, las bibliotecas centrales, los helpers, drivers, hooks, lenguajes y cualquier otro recurso necesario para procesar la solicitud específica
6. La Vista finalizada se procesa y luego se envía al navegador web para que se pueda ver, si el almacenamiento en caché está habilitado, la vista se almacena primero en la memoria caché para que se pueda atender en solicitudes posteriores

Modelo-Vista-Controlador

CodeIgniter se basa en el patrón de desarrollo Model-View-Controller, MVC es un enfoque de software que separa la lógica de la aplicación de la presentación, en la práctica, permite que sus páginas web contengan un mínimo de código, ya que la presentación es independiente de las secuencias de comandos de PHP

- El modelo representa las estructuras de datos, normalmente, las clases de modelo contendrán funciones que le ayudarán a recuperar, insertar y actualizar información en su Base de datos
- La Vista es la información que se presenta al usuario, una Vista normalmente será una página web, pero en CodeIgniter, una vista también puede ser un fragmento de página como un encabezado o pie de página, también puede ser una página RSS o cualquier otro tipo de "página"
- El controlador sirve como intermediario entre el modelo, la vista y cualquier otro recurso necesario para procesar la solicitud HTTP y generar una página web

CodeIgniter tiene un enfoque bastante flexible para MVC ya que los modelos no son necesarios, si no necesita la separación adicional, o encuentra que el mantenimiento de modelos requiere más complejidad de la que desea, puede ignorarlos y crear su aplicación usando solo Controladores y Vistas

CodeIgniter también le permite incorporar sus propias secuencias de comandos existentes, o incluso desarrollar bibliotecas principales para el sistema, permitiéndole trabajar de la manera que le resulte más útil y provechosa

Objetivos de Diseño y Arquitectura

Nuestro objetivo para CodeIgniter es el máximo rendimiento, capacidad y flexibilidad en el paquete más pequeño y liviano posible

Para cumplir con este objetivo, nos comprometemos con la evaluación comparativa, la reorganización y la simplificación en cada paso del proceso de desarrollo, rechazando todo lo que no supere el objetivo establecido

Desde un punto de vista técnico y arquitectónico, CodeIgniter se creó con los siguientes objetivos:

- **Instanciación dinámica**: En CodeIgniter, los componentes se cargan y las rutinas se ejecutan solo cuando se solicita, en lugar de hacerlo de manera global, el sistema no hace suposiciones con respecto a lo que puede ser necesario más allá de los recursos básicos mínimos, por lo que el sistema es muy ligero por defecto, los eventos, según la solicitud HTTP, y los controladores y vistas que ha diseñado determinan qué se invoca
- **Bajo acoplamiento**: El acoplamiento es el grado en que los componentes de un sistema dependen el uno del otro, cuanto menos componentes dependan unos de otros, más reutilizable y flexible se vuelve el sistema, nuestro objetivo era un sistema muy débilmente acoplado
- **Singularidad del componente**: Singularidad es el grado en que los componentes tienen un propósito estrechamente enfocado, en CodeIgniter, cada clase y sus funciones son altamente autónomas para permitir la máxima utilidad

CodeIgniter es un sistema dinámicamente instanciado, débilmente acoplado con alta singularidad en sus componentes, se esfuerza por ser simple, flexible y con un alto rendimiento en un paquete de tamaño reducido

Tutorial

Páginas estáticas

Nota
Este tutorial supone que ha descargado CodeIgniter e instalado el framework en su entorno de desarrollo

Lo primero que debe hacer es configurar un controlador para manejar páginas estáticas, un controlador es simplemente una clase que ayuda a delegar el trabajo, es el núcleo de su aplicación web

Por ejemplo, cuando se realiza una llamada a:

```
http://ejemplo.com/noticias/nuevas/10
```

Podemos imaginar que hay un controlador llamado "**noticias**", el método que se llama en noticias sería "**nuevas**"

El trabajo del método noticias podría consistir en obtener **10** noticias y presentarlas en la página, muy a menudo en MVC, verá patrones de URL que coinciden con:

```
http://ejemplo.com/[clase-controlador]/[método-controlador]/[argumentos]
```

A medida que los esquemas de URL se vuelven más complejos, esto puede cambiar, pero por ahora, esto es todo lo que necesitaremos saber

Cree un archivo en **application/controllers/Paginas.php** con el siguiente código:

```
<?php
class Paginas extends CI_Controller {
  public function ver($pagina = 'inicio')
  {
  }
}
```

Ha creado una clase llamada **Paginas**, con un método llamado **ver** que recibe un argumento llamado **$pagina**, la clase **Paginas** amplía la clase **CI_Controller,** esto significa que la nueva clase **Paginas** puede acceder a los métodos y variables definidos en la clase **CI_Controller** (**system/core/Controller.php**)

El controlador se convierte en el centro de cada solicitud a su aplicación web, en discusiones muy técnicas de CodeIgniter, puede ser referido como **el súper objeto**, al igual que cualquier clase PHP, se refiere a ella dentro de sus controladores como **$this**, referirse a **$this** es como cargará bibliotecas, vistas y, en general utilizará el framework

Ahora que ha creado su primer método, es hora de crear algunas plantillas básicas, crearemos dos "**views**" (plantillas de página) que actúen como encabezado y pie de página
Cree el encabezado en **application/views/plantillas/Cabecera.php** y agregue el siguiente código:

```
<html>
  <head>
    <title>Tutorial de CodeIgniter</title>
  </head>
  <body>
    <h1><?php echo $título; ?></h1>
```

La cabecera contiene el código HTML básico que querrá mostrar antes de cargar la vista principal, junto con una cabecera, también mostrará la variable **$título**, que definiremos más adelante en el controlador

Ahora, cree un pie de página en **application/views/plantillas/Pie.php** que incluya el siguiente código:

```
    <em>&copy; 2015</em>
  </body>
</html>
```

Agregar lógica al controlador

Anteriormente ha creado un controlador con un método **ver()**, el método acepta un parámetro, que es el nombre de la página que se cargará, las plantillas de página estáticas se ubican en el directorio **application/views/paginas/**

En ese directorio, cree dos archivos llamados **Inicio.php** y **Acercade.php** en ellos, escriba algo de texto, cualquier cosa que quiera, y guárdelos, si no le gusta ser original, escriba: "Hola Mundo !"

Para cargar esas páginas, modifique el método **ver()** del controlador en **application/controllers/Paginas.php** para verificar si realmente existe la página solicitada:

```
public function ver($pagina = 'inicio')
{
  if (!file_exists(APPPATH.'views/paginas/'.$pagina.'.php'))
  {
    // Cáspita ... no tenemos una pagina para esto!
    show_404();
  }

  $datos['título'] = ucfirst($pagina); // Capitaliza la primera letra

  $this->load->view('plantillas/cabecera', $datos);
  $this->load->view('paginas/'.$pagina, $datos);
  $this->load->view('plantillas/pie', $datos);
}
```

Ahora, si la página existe, se carga, incluidos el encabezado y el pie de página, y se muestra al usuario, si la página no existe, se muestra un error "404 página no encontrada"

La primera línea de este método verifica si la página realmente existe, la función de PHP **file_exists ()** se usa para verificar si el archivo está donde se espera que esté, **show_404 ()** es una función incorporada de CodeIgniter que muestra la página de error predeterminada

En la plantilla de la cabecera, la variable **$título** se usó para personalizar el título de la página, el valor del título se define en este método, pero en lugar de asignar el valor a una variable, se asigna en el array **$datos**

Lo último que debe hacerse es cargar las vistas en el orden en que se mostrarán, el segundo parámetro en el método **ver()** se usa para pasar valores a la vista, cada valor en el array **$datos** se asigna a una variable con el nombre de su clave, entonces el valor de **$datos ['título']** en el controlador es equivalente a **$título** en la vista

Enrutamiento

El controlador ahora está funcionando !

Dirija su navegador a [**url-de-su-sitio**] **index.php/paginas/ver** para ver su página, cuando visite **index.php/paginas/ver/acercade**, verá la página **Acercade**, una vez más, incluyendo la cabecera y el pie de página

Al usar reglas de enrutamiento personalizadas, se tiene la capacidad de asignar cualquier URI a cualquier controlador y método, y liberarse de la convención normal:

```
http://ejemplo.com/[clase-controlador]/[método-controlador]/[argumentos]
```

Vamos a hacerlo, abra el archivo de enrutamiento ubicado en **application/config/routes.php** y agregue las siguientes dos líneas, elimine el resto del código y establezca los elementos del array **$route** así:

```
$route['default_controller'] = 'paginas/ver';
$route['(:any)'] = 'paginas/ver/$1';
```

CodeIgniter lee sus reglas de enrutamiento de arriba hacia abajo y encamina la solicitud a la primera regla que coincide, cada regla es una expresión regular, lado izquierdo, asignada a un controlador y nombre de método separado por barras, lado derecho, cuando aparece una solicitud, CodeIgniter busca la primera coincidencia y llama al controlador y el método apropiados, posiblemente con argumentos

Aquí, la segunda regla en el array **$route** coincide con cualquier solicitud usando la cadena comodín **(:any)** y pasa el parámetro al método **ver()** de la clase **Paginas**

Ahora visite **index.php/acercade**

Se enruta correctamente al método **ver()** en el controlador de páginas ?

Olé...olé... Híjole... Mirá vos... !!!

Puede encontrar más información sobre el enrutamiento en la documentación de enrutamiento de URI

Sección de noticias

En la última sección, vimos algunos conceptos básicos del framework al escribir una clase que incluye páginas estáticas, limpiamos el URI agregando reglas de enrutamiento personalizadas, ahora es el momento de introducir contenido dinámico y comenzar a usar una Base de datos

Configurar su modelo

En lugar de escribir las operaciones de la Base de datos directamente en el controlador, las consultas se deben colocar en un modelo, para que se puedan reutilizar fácilmente más adelante, los modelos son el lugar donde recupera, inserta y actualiza la información en su Base de datos u otros almacenamientos de datos, ellos representan sus datos

Abra el directorio **application/models/** y cree un nuevo archivo llamado **Noticias_model.php** y agregue el siguiente código

Asegúrese de haber configurado su Base de datos correctamente como se describe en la sección **Configuración de la Base de Datos**

```
<?php
class Noticias_model extends CI_Model {

  public function __construct()
  {
     $this->load->database();
  }
}
```

Este código es similar al código del controlador que utilizó anteriormente, crea un nuevo modelo al extender **CI_Model** y carga la biblioteca de Base de datos, esto hará que la clase **database** esté disponible a través del objeto **$this->db**

Antes de consultar la Base de datos, se debe crear un esquema de Base de datos, conéctese a su Base de datos y ejecute los siguientes comandos SQL (MySQL), agregue también algunos registros

Nota: slug (babosa) es el término inglés para las "url amigables"

```
CREATE TABLE noticias (
  id int(11) NOT NULL AUTO_INCREMENT,
  título varchar(128) NOT NULL,
  slug varchar(128) NOT NULL,
  texto text NOT NULL,
  PRIMARY KEY (id),
  KEY slug (slug)
);
```

Ahora que la Base de datos y el modelo se han configurado, necesitará un método para obtener todas las publicaciones de la Base de datos, para ello, se utiliza la capa de abstracción de la Base de datos que se incluye con CodeIgniter **Query Builder**, esto hace posible escribir sus 'consultas' una vez y hacer que funcionen **en todos los sistemas de bases de datos compatibles**

Agregue el siguiente código a su modelo:

```
public function get_noticias($slug = FALSE)
{
    if ($slug === FALSE)
    {
        $query = $this->db->get('noticias');
        return $query->result_array();
    }

    $query = $this->db->get_where('noticias', array('slug' => $slug));
    return $query->row_array();
}
```

Con este código puede realizar dos consultas diferentes, puede obtener todos los registros de noticias u obtener una noticia por su identificador, es posible que haya notado que la variable **$slug** no se saneo antes de ejecutar la consulta; **Query Builder** hace esto por usted

Mostrar las noticias

Ahora que se han escrito las consultas, el modelo debería vincularse a las vistas que se encargan de mostrar las noticias al usuario, esto puede hacerse en nuestro controlador **Paginas** creado anteriormente, pero, en aras de la claridad, se define un nuevo controlador **Noticias**

Cree el nuevo controlador en **application/controllers/Noticias.php**:

```
<?php
class Noticias extends CI_Controller {

  public function __construct()
  {
     parent::__construct();
     $this->load->model('noticias_model');
     $this->load->helper('url_helper');
  }

  public function index()
  {
     $datos['noticias'] = $this->noticias_model->get_noticias();
  }
  public function ver($slug = NULL)
  {
     $datos['noticias_item'] = $this->noticias_model->get_noticias($slug);
  }
}
```

Al mirar el código, puede verse cierta similitud con los archivos que creamos anteriormente, primero, el método **__construct ()** llama al constructor de su clase padre, **CI_Controller**, y carga el modelo, por lo que se puede usar en

todos los métodos del controlador, también carga el ayudante **URL Helper** con su conjunto de funciones, porque usaremos una de ellas en una vista posterior

A continuación, hay dos métodos para ver todas las noticias y una para una noticia específica, puede ver que la variable **$slug** se pasa al método del modelo en el segundo método, el modelo usa esta variable para identificar la noticia que se devuelve

Ahora el controlador recupera los datos a través de nuestro modelo, pero aún no se muestra nada, el siguiente paso es pasar esta información a la vista, pero primero actualice el método **index()** en el controlador:

```
public function index()
{
   $datos['noticias'] = $this->noticias_model->get_noticias();
   $datos['título'] = 'Noticias';

   $this->load->view('plantillas/cabecera', $datos);
   $this->load->view('noticias/index', $datos);
   $this->load->view('plantillas/pie');
}
```

El código anterior obtiene todos los registros de noticias del modelo y los asigna a una variable, el valor para el título también se asigna al elemento **$datos['título']** y todos los datos se pasan a las vistas, ahora necesita crear una vista para presentar las noticias

Cree **application/views/ noticias/index.php** y agregue el siguiente código:

```
<h2><?php echo $título; ?></h2>

<?php foreach ($noticias as $noticia_item):?>

  <h3><?php echo $noticia_item['título']; ?></h3>
    <div class = "inicio">
       <?php echo $noticia_item['texto']; ?>
    </div>
  <p><a href="<?php echo site_url('noticias/'.$noticia_item['slug']); ?> "> Ver artículo</a></p>

<?php endforeach; ?>
```

Aquí, cada noticia se coloca en bucle y se muestra al usuario, puede ver que escribimos nuestra plantilla en PHP mezclado con HTML, si prefiere utilizar un lenguaje de plantilla, puede usar la clase **Template Parser** de CodeIgniter u otro analizador de plantillas

La página de resumen de noticias ahora está lista, pero todavía no hay una página para mostrar noticias individuales, el modelo creado anteriormente está hecho de tal manera que puede usarse fácilmente para esta funcionalidad, solo necesita agregar un código al controlador y crear una nueva vista

Regrese al controlador **Noticias** y actualice **ver()** con lo siguiente:

```
public function ver($slug = NULL)
{
   $datos['noticia_item'] = $this->noticias_model->get_noticias($slug);

   if (empty($datos['noticia_item']))
   {
      show_404();
   }
   $datos['título'] = $datos['noticia_item']['título'];

   $this->load->view('plantillas/cabecera', $datos);
   $this->load->view('noticias/ver', $datos);
   $this->load->view('plantillas/pie');
}
```

En lugar de llamar al método **get_noticias()** sin parámetro, se le pasa la variable **$slug**, por lo que devolverá la noticia específica, lo único que queda por hacer es crear la vista correspondiente en **application/views/noticias/Ver.php**

Ponga el siguiente código en ese archivo:

```
<?php
echo '<h2>'.$noticia_item['título'].'</h2>';
echo $noticia_item['texto'];
```

Enrutamiento

Debido a la regla de enrutamiento comodín creada anteriormente, necesita una ruta adicional para ver el controlador que acaba de crear

Modifique su archivo de enrutamiento (**application/ config/routes.php**) para que se vea de la manera siguiente, esto asegura que las solicitudes lleguen al controlador **Noticias** en lugar de ir directamente al controlador **Paginas**, la primera línea enruta el URI al método **ver()** del controlador **Noticias**:

```
$route['noticias/(:any)'] = 'noticias/ver/$1';
$route['noticias'] = 'noticias';
$route['(:any)'] = 'paginas/ver/$1';
$route['default_controller'] = 'paginas/ver';
```

Dirija su navegador a la raíz de su documento, seguido de **index.php/noticias** y mire la página de noticias

Crear noticias

Ahora sabe cómo puede leer datos de una Base de datos usando CodeIgniter, pero aún no ha escrito ninguna información en la Base de datos, en esta sección expandirá su controlador de noticias y modelo creado anteriormente para incluir esta funcionalidad

Crear un formulario

Para ingresar datos en la Base de datos, debe crear un formulario donde pueda ingresar la información que se almacenará, esto significa que necesitará un formulario con dos campos, uno para el título y otro para el texto, obtendrá el identificador de nuestro título en el modelo

Cree la nueva vista en **application/views/noticias/crear.php**:

```
<h2><?php echo $título; ?></h2>

<?php echo validation_errors(); ?>

<?php echo form_open('noticias/crear'); ?>

  <label for="título">Título</label>
  <input type="input" name="título" /><br/>

  <label for="texto">Texto</label>
  <textarea name="texto"></textarea><br/>

  <input type="submit" name="submit" value="Crear una nueva noticia" />

</form>
```

Aquí hay solo dos cosas que probablemente no le resulten familiares: la función **form_open()** y la función **validation_errors()**

La primera función la proporciona el ayudante **form helper** y representa al elemento de formulario y agrega funcionalidad adicional, como un campo oculto para evitar el **CSRF** (del inglés Cross-Site Request Forgery o falsificación de solicitud de sitios cruzados), este último se usa para informar errores relacionados con la validación de formulario

Regrese a su controlador de noticias, aquí va a hacer dos cosas, comprobar si el formulario se envió y si los datos enviados pasaron las reglas de validación, utilizará la biblioteca **form validation** para hacerlo

```
public function crear()
{
   $this->load->helper('form');
   $this->load->library('form_validation');

   $datos['título'] = 'Crear una nueva noticia';
```

```
    $this->form_validation->set_rules('título','Título','required');
    $this->form_validation->set_rules('texto','Texto','required');

    if ($this->form_validation->run() === FALSE)
    {
        $this->load->view('plantillas/cabecera', $datos);
        $this->load->view('noticias/crear');
        $this->load->view('plantillas/pie');
    }
    else
    {
        $this->noticias_model->set_noticias();
        $this->load->view('noticias/resultado');
    }
}
```

El código anterior agrega mucha funcionalidad, las primeras líneas cargan el ayudante **form helper** y la biblioteca **form validation**, después de eso, se establecen las reglas para la validación del formulario, el método **set_rules()** toma tres argumentos; el nombre del campo de entrada, el nombre que se utilizará en los mensajes de error y la regla, en este caso, los campos de título y texto son obligatorios

CodeIgniter tiene una poderosa biblioteca de validación de formularios como se demostró anteriormente, puede leer más acerca de esta biblioteca en la sección correspondiente

Más abajo, puede ver una condición que verifica si la validación del formulario se ejecutó correctamente, si no, se muestra el formulario, si se envió y pasó todas las reglas, se llama al modelo, después de esto, se carga una vista para mostrar un mensaje de éxito, cree una vista en **application/views/noticias/resultado.php** y escriba un mensaje de éxito

El modelo

Lo único que queda es escribir un método que escriba los datos en la Base de datos, utilizará la clase **Query Builder** para insertar la información y usar la biblioteca **Input library** para obtener los datos publicados

Abra el modelo creado anteriormente y agregue lo siguiente:

```
public function set_noticias()
{
    $this->load->helper('url');

    $slug = url_title($this->input->post('título'), 'dash', TRUE);

    $datos = array(
      'título' => $this->input->post('título'),
      'slug' => $slug,
      'texto' => $this->input->post('texto')
    );
```

```
    return $this->db->insert('noticias', $datos);
}
```

Este nuevo método se encarga de insertar la noticia en la Base de datos, la tercera línea contiene una nueva función, **url_title ()**, esta función, proporcionada por el ayudante **URL helper**, divide la cadena que se le pasa, reemplaza todos los espacios por guiones (-) y se asegura de que todo esté en minúsculas, esto le deja con una identificador limpio, perfecto para crear URI

Continuemos con la preparación del registro que se insertará más adelante, dentro del array **$datos**, cada elemento se corresponde con una columna en la tabla de la Base de datos creada anteriormente, es posible que observe un nuevo método, el método **post()** de la biblioteca **input libary**, este método se asegura que los datos se desinfectan, protegiéndole de los ataques de terceros, la biblioteca **input libary** se carga de manera predeterminada, por último, insertamos nuestro array **$datos** en la base de datos

Enrutamiento

Antes de que pueda comenzar a agregar noticias en su aplicación CodeIgniter, debe agregar una regla adicional en el archivo **config/routes.php**, asegúrese de que su archivo contenga lo siguiente, esto asegura que CodeIgniter vea '**crear**' como un método en lugar del identificador de una noticia:

```
$route['noticias/crear'] = 'noticias/crear';
$route['noticias/(:any)'] = 'noticias/ver/$1';
$route['noticias'] = 'noticias';
$route['(:any)'] = 'paginas/ver/$1';
$route['default_controller'] = 'paginas/ver';
```

Ahora dirija su navegador a su entorno de desarrollo local donde instaló CodeIgniter y agregue **index.php/noticias/crear** a la URL

¡ Felicidades, acaba de crear su primera aplicación CodeIgniter !

Agregue algunas noticias y revise las diferentes páginas que ha creado

Conclusión

Este tutorial no cubre todo lo puede esperar de un framework, pero lo introdujo a los temas más importantes de enrutamiento, escritura de controladores y modelos, esperamos que este tutorial le haya dado una idea de algunos de los patrones de diseño básicos de CodeIgniter, los cuales puede ampliar

Ahora que ha completado este tutorial, le recomendamos que consulte el resto de la documentación, CodeIgniter a menudo se elogia debido a su completa documentación, use esto en su ventaja y lea a fondo las secciones "**Introducción**" y "**Temas generales**", debe leer las bibliotecas y ayudantes cuando sea necesario

Todos los programadores de PHP intermedios deberían poder utilizar CodeIgniter en pocos días

Si todavía tiene preguntas sobre el marco o su propio código CodeIgniter, puede:

Mirar en nuestros foros: **http://forum.codeigniter.com/**

Visite nuestra IRC chatroom: **https://github.com/bcit-ci/CodeIgniter/wiki/IRC**

Explore la Wiki: **https://github.com/bcit-ci/CodeIgniter/wiki/**

Temas generales

Las URLs de CodeIgniter

De forma predeterminada, las URL en CodeIgniter están diseñadas para ser motor de búsqueda y también amigables para el ser humano

En lugar de utilizar el enfoque estándar de "**query string**" para las URL que es sinónimo de sistemas dinámicos, CodeIgniter utiliza un enfoque basado en segmentos:

```
ejemplo.com/noticias/articulo/mi_articulo
```

Nota
Las URL tipo "**query string**" pueden habilitarse opcionalmente, como se describe más adelante

Segmentos de URI

Siguiendo el enfoque Modelo-Vista-Controlador, los segmentos en la URL normalmente se presentan:

```
ejemplo.com/clase/método/ID
```

1. El primer segmento representa la clase de controlador que debe invocarse:
   ejemplo.com/**clase**/método/ID
2. El segundo segmento representa el método o función, que debería llamarse:
   ejemplo.com/clase/**método**/ID
3. El tercero y cualquier segmento adicional representan cualquier variable que se pasará al controlador:
   ejemplo.com/clase/método/**ID**

La biblioteca de **URI** y el ayudante **URL Helper** contienen funciones que facilitan el trabajo con los datos de URI, además, sus URL se pueden reasignar utilizando la función de enrutamiento **URI Routing** para obtener más flexibilidad

Eliminar el archivo index.php

De forma predeterminada, el archivo **index.php** se incluirá en sus URL:

```
ejemplo.com/index.php/noticias/articulo/mi_articulo
```

Si su servidor Apache tiene habilitado **mod_rewrite**, puede eliminar fácilmente este archivo utilizando un archivo **.htaccess** con algunas reglas simples

Aquí hay un ejemplo de dicho archivo, utilizando el método "**negativo**" en el que todo se redirige excepto los elementos especificados:

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php/$1 [L]
```

En el ejemplo anterior, cualquier solicitud HTTP que no sea la de directorios existentes y archivos existentes se trata como una solicitud para su archivo **index.php**

Nota

Estas reglas específicas pueden no funcionar en todas las configuraciones de servidor

Asegúrese de excluir también de la regla anterior cualquier recurso que pueda necesitar para que sea accesible desde el mundo exterior

Agregar un sufijo a la URL

En su archivo **config/config.php** puede especificar un sufijo que se agregará a todas las URL generadas por CodeIgniter

Por ejemplo, si tiene esta URL:

```
ejemplo.com/index.php/productos/ver/zapatos
```

Opcionalmente, puede agregar un sufijo, como .html, haciendo que la página parezca ser de cierto tipo:

```
ejemplo.com/index.php/productos/ver/zapatos.html
```

Habilitar la Query string

En algunos casos, es posible que prefiera utilizar las URL con cadena de consulta (**Query string**):

```
index.php?c=productos&m=ver&id=345
```

CodeIgniter admite opcionalmente esta capacidad, que se puede habilitar en su archivo **application/config.php**, si abre su archivo de configuración, verá estos elementos:

```
$config['enable_query_strings'] = FALSE;
$config['controller_trigger'] = 'c';
$config['function_trigger'] = 'm';
```

Si cambia "**enable_query_strings**" a TRUE, esta característica se activará, se podrá acceder a sus controladores y funciones utilizando el "**trigger**" que haya configurado para invocar sus controladores y métodos :

```
index.php?c=controlador&m=método
```

Nota

Si está utilizando una cadena de consulta, (**Query string**), tendrá que crear sus propias URL, en lugar de utilizar los ayudantes de URL, y otros ayudantes que generan URL, como algunos de formularios, ya que están diseñados para trabajar con URL basadas en segmentos

Controllers - Controladores

Los controladores son el corazón de su aplicación, ya que determinan cómo se manejan las solicitudes HTTP

Qué es un controlador?

Un controlador es simplemente un archivo de clase cuyo nombre se puede asociar con un URI

Considere este URI:

```
ejemplo.com/index.php/blog/
```

En el ejemplo anterior, CodeIgniter intentaría encontrar un controlador llamado **Blog.php** y cargarlo, cuando el nombre de un controlador coincide con el primer segmento de un URI, se cargará

Probémoslo: ¡Hola mundo!

Vamos a crear un controlador simple para que pueda verlo en acción

Usando su editor de texto, cree un archivo llamado **Blog.php** y escriba el siguiente código:

```
<?php
class Blog extends CI_Controller {

  public function index()
  {
     echo 'Hola Mundo!';
  }
}
```

A continuación, guarde el archivo en el directorio: **application/controllers/**

Importante

El archivo debe llamarse "**Blog.php**" con la "**B**" mayúscula

Ahora visite su sitio usando una URL similar a esta:

```
ejemplo.com/index.php/blog/
```

Si lo hizo bien, debería ver: Hola Mundo !

Importante

Los nombres de clase deben comenzar con una letra mayúscula

Esto es válido:

```
<?php
class Blog extends CI_Controller {
}
```

Esto no es válido:

```
<?php
class blog extends CI_Controller {
}
```

Además, siempre asegúrese de que su clase controlador "**extends**" la clase del controlador padre para que pueda heredar todos sus métodos

Métodos

En el ejemplo anterior, el nombre del método es **index()**, el método "**index**" siempre se carga de manera predeterminada si el segundo segmento del URI está vacío

Otra forma de mostrar su mensaje "Hola Mundo" sería esta:

```
ejemplo.com/index.php/blog/index/
```

El segundo segmento del URI determina qué método se llama en el controlador

Vamos a intentarlo, agregue un nuevo método a su controlador:

```
<?php
class Blog extends CI_Controller {

  public function index()
  {
     echo 'Hola Mundo!';
  }

  public function comentarios()
  {
     echo 'Mire esto!';
  }
}
```

Ahora cargue la siguiente URL para ver el método de comentarios:

```
ejemplo.com/index.php/blog/comentarios/
```

Debería ver su nuevo mensaje

Pasar segmentos de URI a sus métodos

Si su URI contiene más de dos segmentos, a partir del tercer segmento, se pasarán a su método como parámetros

Por ejemplo, digamos que tiene una URI como esta:

```
ejemplo.com/index.php/productos/zapatos/sandalias/123
```

Los segmentos URI 3 y 4 ("sandalias" y "123") se pasarán a su método como parámetros:

```
<?php
class Products extends CI_Controller {

  public function zapatos($tipo, $id)
  {
     echo $tipo;
     echo $id;
  }
}
```

Importante

Si está utilizando URI Routing, los segmentos transferidos a su método serán redirigidos

Definir un controlador predeterminado

Se puede indicar a CodeIgniter que cargue un controlador predeterminado cuando un URI no está presente, como será el caso cuando solo se solicite la URL raíz de su sitio

Para especificar un controlador predeterminado, abra su archivo **application/config/routes.php** y configure esta variable:

```
$route['default_controller'] = 'blog';
```

Donde '**blog**' es el nombre de la clase de controlador que desea utilizar, si ahora carga su archivo **index.php** principal sin especificar ningún segmento URI, verá su mensaje "Hola Mundo" de forma predeterminada
Para obtener más información, consulte la sección "**Rutas reservadas**" en la documentación de **URI Routing**

Remapear las llamadas a métodos

Como se indicó anteriormente, el segundo segmento del URI típicamente determina qué método se llama en el controlador

CodeIgniter le permite anular este comportamiento mediante el uso del método **_remap()**:

```
public function _remap()
{
  // Algo de código ...
}
```

Importante

Si su controlador contiene un método llamado **_remap ()** siempre se llamará independientemente de lo que contenga su URI, anula el comportamiento normal en el que el URI determina a qué método se llama, lo que le permite definir sus propias reglas de enrutamiento de métodos

La llamada al método reemplazado, generalmente el segundo segmento del URI, se pasará como un parámetro al método **_remap ()**:

```
public function _remap($metodo)
{
   if ($metodo === 'algun_metodo')
   {
      $this->$metodo();
   }
   else
   {
      $this->default_metodo();
   }
}
```

Cualquier segmento adicional después del nombre del método se pasa a **_remap ()** como un segundo parámetro opcional

Este array se puede usar en combinación con **call_user_func_array()** de PHP para emular el comportamiento predeterminado de CodeIgniter:

```
public function _remap($metodo, $parametros = array())
{
   $metodo = 'process_'.$metodo;
   if (method_exists($this, $metodo))
   {
      return call_user_func_array(array($this, $metodo), $parametros);
   }
   show_404();
}
```

Procesar la salida

CodeIgniter tiene una clase de salida que se encarga de enviar automáticamente sus datos renderizados al navegador web, se puede encontrar más información sobre esto en las páginas **Views** y **Output Class**, sin embargo, en algunos casos, es posible que desee pos procesar los datos finalizados de alguna manera y enviarlos al navegador usted mismo, CodeIgniter le permite agregar un método llamado **_output()** a su controlador que recibirá los datos de salida finalizados

Importante

Si su controlador contiene un método llamado **_output ()** siempre será llamado por la clase de salida en lugar de repetir los datos finalizados directamente, el primer parámetro del método contendrá la salida finalizada

Ejemplo

```
public function _output($salida)
{
   echo $salida;
}
```

Nota

Tenga en cuenta que su método **_output()** recibirá los datos en su estado finalizado, los datos de referencia y de uso de la memoria se procesarán, se escribirán los archivos de caché, si tiene habilitado el almacenamiento en caché, y se enviarán los encabezados, si usa esa característica, antes de transferirlos al método **_output ()**. Para tener la salida de su controlador en la memoria caché, su método **_output ()** puede usar:

```
if ($this->output->cache_expiration > 0)
{
   $this->output->_write_cache($salida);
}
```

Si está utilizando esta característica, el temporizador de ejecución de la página y las estadísticas de uso de la memoria pueden no ser perfectamente exactos, ya que no tendrán en cuenta ningún procesamiento adicional que realice, para una forma alternativa de controlar la salida antes de que se realice el procesamiento final, consulte los métodos disponibles en **Output Library**

Métodos privados

En algunos casos, es posible que desee ciertos métodos ocultos para el acceso público, para lograr esto, simplemente declare el método como privado o protegido y no será atendido a través de una solicitud de URL

Por ejemplo, si tuviera un método como este:

```
private function _utility()
{
   // Algo de código ...
}
```

Intentar acceder a través de la URL, así, no funcionará:

```
ejemplo.com/index.php/blog/_utility/
```

Nota

La prefijación de los nombres de los métodos con un guión bajo también evitará que se llamen, esta es una característica heredada que se deja para compatibilidad hacia atrás

Organizar sus controladores en subdirectorios

Si está creando una aplicación grande, es posible que desee organizar jerárquicamente o estructurar sus controladores en subdirectorios, CodeIgniter le permite hacerlo

Simplemente cree subdirectorios dentro de **application/controllers** y coloque sus clases de controlador dentro de ellos

Nota

Al utilizar esta función, el primer segmento de su URI debe especificar la carpeta, por ejemplo, supongamos que tiene un controlador ubicado aquí:

```
application/controllers/productos/Zapatos.php
```

Para llamar al controlador anterior, su URI se verá más o menos así:

```
ejemplo.com/index.php/productos/zapatos/show/123
```

Cada uno de sus subdirectorios puede contener un controlador predeterminado al que se llamará si el URL contiene solo el subdirectorio, simplemente coloque un controlador que coincida con el nombre de su '**default_controller**' como se especifica en su archivo **application/config/routes.php**

CodeIgniter también le permite reasignar sus URI utilizando **URI Routing**

Constructores de clase

Si tiene la intención de utilizar un constructor en cualquiera de sus Controladores, **DEBE** colocar la siguiente línea de código:

```
parent::__construct();
```

La razón por la que esta línea es necesaria es porque su constructor local anulará al de su clase padre, por lo que debemos llamarlo manualmente:

```
<?php
class Blog extends CI_Controller {

  public function __construct()
  {
     parent::__construct();
     // Su propio código de constructor
  }
}
```

Los constructores son útiles si necesita establecer algunos valores predeterminados o ejecutar un proceso predeterminado cuando se crea una instancia de su clase, los constructores no pueden devolver un valor, pero pueden hacer un trabajo predeterminado

Nombres de métodos reservados

Dado que las clases de su controlador ampliarán el controlador principal de la aplicación, debe tener cuidado de no nombrar sus métodos de manera idéntica a los utilizados por esa clase, de lo contrario, sus funciones locales los anularán, vea los **nombres reservados** para una lista completa

Importante

Tampoco debe tener nunca un método con el mismo nombre que su nombre de clase, si lo hace, y no hay un método **__construct ()** en la misma clase, entonces su por ejemplo, método **Index::index()** se ejecutará como un constructor de clase, esta es una característica de compatibilidad con versiones anteriores a PHP4

¡Eso es!

Eso, en pocas palabras, es todo lo que hay que saber sobre los controladores

Nombres Reservados

CodeIgniter usa una serie de funciones y nombres en su operación, debido a esto, el desarrollador no puede usar algunos nombres, la siguiente es la lista de nombres reservados que no se pueden usar:

Nombres de controlador

Como sus clases de controlador extenderán al controlador principal de la aplicación, tiene que ser cuidadoso de no nombrar a sus funciones del mismo modo que las usadas por otras clases, sino sus clases las anularán, la siguiente es la lista de nombres reservados

No use ninguno de estos nombres para llamar a su controlador:

- CI_Controller
- Default
- index

Nombres de funciones

- is_php()
- is_really_writable()
- load_class()
- is_loaded()
- get_config()
- config_item()
- show_error()
- show_404()
- log_message()
- set_status_header()
- get_mimes()
- html_escape()
- remove_invisible_characters()
- is_https()
- function_usable()
- get_instance()
- _error_handler()
- _exception_handler()
- _stringify_attributes()

Nombres de variables

- $config
- $db
- $lang

Nombres de constantes

ENVIRONMENT
FCPATH
SELF
BASEPATH
APPPATH
VIEWPATH
CI_VERSION
MB_ENABLED
ICONV_ENABLED
UTF8_ENABLED
FILE_READ_MODE
FILE_WRITE_MODE
DIR_READ_MODE
DIR_WRITE_MODE
FOPEN_READ
FOPEN_READ_WRITE
FOPEN_WRITE_CREATE_DESTRUCTIVE
FOPEN_READ_WRITE_CREATE_DESTRUCTIVE
FOPEN_WRITE_CREATE
FOPEN_READ_WRITE_CREATE
FOPEN_WRITE_CREATE_STRICT
FOPEN_READ_WRITE_CREATE_STRICT
SHOW_DEBUG_BACKTRACE
EXIT_SUCCESS
EXIT_ERROR
EXIT_CONFIG
EXIT_UNKNOWN_FILE
EXIT_UNKNOWN_CLASS
EXIT_UNKNOWN_METHOD
EXIT_USER_INPUT
EXIT_DATABASE
EXIT__AUTO_MIN
EXIT__AUTO_MAX

Views -Vistas

Una vista es simplemente una página web, o un fragmento de página, como un encabezado, pie de página, barra lateral, etc., de hecho, las vistas pueden integrarse flexiblemente en otras vistas, dentro de otras vistas, etc., etc., si necesita este tipo de jerarquía

Las vistas nunca se llaman directamente, deben ser cargadas por un controlador, recuerde que en un marco MVC, el controlador actúa como el policía de tráfico, por lo que es responsable de buscar una vista en particular, si no ha leído la página **Controladores**, debe hacerlo antes de continuar

Usando el controlador de ejemplo que creó en la página de **controladores**, agreguemos una vista

Crear una vista

Usando su editor de texto, cree un archivo llamado **Blogview.php** y pónga en él:

```
<html>
<head>
   <title>Mi Blog</title>
</head>
<body>
   <h1>Bienvenido a mi Blog!</h1>
</body>
```

Luego guarde el archivo en su directorio **application/views/**

Cargar una vista

Para cargar un archivo de vista particular, usará el siguiente método:

```
$this->load->view('nombre');
```

Donde nombre es el nombre de tu archivo de vista

Nota

No se necesitar especificar la extensión de archivo .php, a menos que use otra distinta a .php

Ahora, abra el archivo del controlador que creó anteriormente llamado **Blog.php** y reemplace la instrucción **echo** con el método de carga de la vista:

```
<?php
class Blog extends CI_Controller {

  public function index()
  {
     $this->load->view('blogview');
  }
}
```

Si visita su sitio usando la URL que hizo anteriormente, debería ver su nueva vista, con una URL fue similar a esto:

```
ejemplo.com/index.php/blog/
```

Cargar múltiples vistas

CodeIgniter manejará inteligentemente múltiples llamadas a **$this->load->view()** desde dentro de un controlador, si hay más de una llamada, se juntaran, por ejemplo, es posible que desee tener una vista de encabezado, una vista de menú, una vista de contenido y una vista de pie de página

Eso podría verse más o menos así:

```
<?php

class Page extends CI_Controller {

  public function index()
  {
    $datos['pagina_title'] = 'Su título';
    $this->load->view('cabecera');
    $this->load->view('menu');
    $this->load->view('contenido', $datos);
    $this->load->view('pie');
  }
}
```

En el ejemplo anterior, estamos usando **"datos agregados dinámicamente"**, que verá a continuación

Almacenar vistas en subdirectorios

Sus archivos de vista también se pueden almacenar dentro de subdirectorios si prefiere ese tipo de organización, al hacerlo, deberá incluir el nombre del directorio que carga la vista:

```
$this->load->view('nombre_del_directorio/nombre_de_archivo');
```

Agregar datos dinámicos a la vista

Los datos se pasan del controlador a la vista por medio de una array o un objeto en el segundo parámetro del método de carga de la vista

Aquí hay un ejemplo usando una array:

```
$datos = array(
  'título'   => 'Mi título',
  'cabecera' => 'Mi cabecera',
  'mensage'  => 'Mi mensage'
);

$this->load->view('blogview', $datos);
```

Y aquí hay un ejemplo usando un objeto:

```
$datos = new Alguna_clase();
$this->load->view('blogview', $datos);
```

Nota

Si usa un objeto, las variables de clase se convertirán en elementos del array

Probémoslo con su archivo de controlador, ábralo y agregue este código:

```
<?php
class Blog extends CI_Controller {

  public function index()
  {
     $datos['título'] = "Mi título real";
     $datos['cabecera'] = "Mi cabecera real";

     $this->load->view('blogview', $datos);
  }
}
```

Ahora abra su archivo de vista y cambie el texto para variables que corresponden a las claves del array en sus datos:

```
<html>
<head>
   <title><?php echo $título;?></title>
</head>
<body>
   <h1><?php echo $cabecera;?></h1>
</body>
</html>
```

Cargue la página en la URL que ha estado usando y debería ver las variables reemplazadas

Crear bucles

El array de datos que pasa a sus archivos de vista no está limitado a variables simples, puede pasar arrays multidimensionales, que pueden enlazarse para generar múltiples filas, por ejemplo, si extrae datos de su Base de datos, generalmente tendrá la forma de una array multidimensional

Aquí hay un ejemplo simple, agregue esto a su controlador:

```
<?php
class Blog extends CI_Controller {

  public function index()
  {
     $datos['lista_completa'] = array('Limpiar la casa', 'Llamar a mamá', 'Hacer cosas');

     $datos['título'] = "Mi título real";
     $datos['cabecera'] = "Mi cabecera real";

     $this->load->view('blogview', $datos);
  }
}
```

Ahora abra su archivo de vista y cree un bucle:

```
<html>
<head>
  <title><?php echo $título;?></title>
</head>
<body>
  <h1><?php echo $cabecera;?></h1>
  <h3>Mi lista completa</h3>
    <ul>
      <?php foreach ($lista_completa as $item):?>
        <li><?php echo $item;?></li>
      <?php endforeach;?>
    </ul>
</body>
</html>
```

Nota

Notará que en el ejemplo anterior estamos usando la sintaxis alternativa de PHP, si no está familiarizado con esto, puede leer sobre ella el apartado "**Sintaxis alternativa de PHP para archivos de vistas**"

Devolver vistas como datos

Hay un tercer parámetro opcional que le permite cambiar el comportamiento del método para que devuelva datos como un string en lugar de enviarla a su navegador, esto puede ser útil si desea procesar los datos de alguna manera, si

establece el parámetro a TRUE, devolverá los datos, el comportamiento predeterminado es FALSE, que lo envía al navegador

Recuerde asignarlo a una variable si desea que se devuelvan los datos:

```
$string = $this->load->view('mi_archivo', '', TRUE);
```

Models -Modelos

Los modelos están disponibles opcionalmente para aquellos que quieran usar un enfoque MVC más tradicional

Qué es un modelo?

Los modelos son clases PHP que están diseñadas para trabajar con información en su Base de datos, por ejemplo, supongamos que usa CodeIgniter para administrar un blog, es posible que tenga una clase de modelo que contenga funciones para insertar, actualizar y recuperar los datos de su blog

Aquí hay un ejemplo de cómo podría ser una clase de modelo así:

```
class Blog_model extends CI_Model {

  public $título;
  public $contenido;
  public $fecha;

  public function get_ultimas_diez_entradas()
  {
     $query = $this->db->get('entradas', 10);
     return $query->result();
  }
  public function insertar_entrada()
  {
     $this->título    = $_POST['título']; //por favor leer la nota de abajo
     $this->contenido = $_POST['contenido'];
     $this->fecha     = time();

     $this->db->insert('entradas', $this);
  }

  public function update_entry()
  {
     $this->título    = $_POST['título'];
     $this->contenido = $_POST['contenido'];
     $this->fecha     = time();

     $this->db->update('entradas', $this, array('id' => $_POST['id']));
  }
}
```

Nota

Los métodos en el ejemplo anterior usan los métodos de Base de datos del **Query Builder**

Nota

En aras de la simplicidad, en este ejemplo estamos usando **$_POST** directamente, en general, esto es una mala práctica, y un enfoque más común sería usar la biblioteca de entrada **Input Library: $this->input->post('título')**

Anatomía de un modelo

Las clases de modelo se almacenan en el directorio **application/models/**, se pueden anidar dentro de subdirectorios si desea este tipo de organización

El prototipo básico para una clase modelo es este:

```
class Nombre_modelo extends CI_Model {


}
```

Donde **Nombre_modelo** es el nombre de su clase, los nombres de clase deben tener la primera letra en mayúscula y el resto del nombre en minúscula y asegúrese de que su clase '**extends**' la clase de modelo base

El nombre del archivo debe coincidir con el nombre de la clase, por ejemplo, si esta es su clase:

```
class Usuario_model extends CI_Model {


}
```

Su archivo será este:

```
application/models/Usuario_model.php
```

Cargar un modelo

Sus modelos generalmente se cargarán y llamán desde los métodos de su **controlador**, para cargar un modelo, usará el siguiente método:

```
$this->load->model('nombre_modelo');
```

Si su modelo está ubicado en un subdirectorio, incluya la ruta relativa desde su directorio de modelos, por ejemplo, si tiene un modelo ubicado en **application/models/blog/Consultas.php**, lo cargará usando:

```
$this->load->model('blog/consultas');
```

Una vez cargado, accederá a los métodos de su modelo utilizando un objeto con el mismo nombre que su clase:

```
$this->load->model('nombre_modelo');


$this->nombre_modelo->método();
```

Si desea que su modelo se asigne a un nombre de objeto diferente, puede especificarlo mediante el segundo parámetro del método de carga:

```
$this->load->model('nombre_modelo', 'foobar');

$this->foobar->método();
```

Aquí hay un ejemplo de un controlador que carga un modelo y luego sirve una vista:

```
class Blog_controller extends CI_Controller {

  public function blog()
  {
     $this->load->model('blog');

     $datos['consulta'] = $this->blog->get_ultimas_diez_entradas();

     $this->load->view('blog', $datos);
  }
}
```

Cargar un modelo automáticamente

Si encuentra que necesita tener un modelo globalmente en su aplicación, puede indicarle a CodeIgniter que realice la carga automáticamente durante la inicialización del sistema, esto lo puede hacer abriendo el archivo **application/config/autoload.php** y agregando el modelo al array de autocarga

Conectar a su Base de datos

Cuando se carga un modelo, **NO** se conecta automáticamente a su Base de datos, las siguientes opciones para conectarse están disponibles para usted:

- Puede conectarse utilizando los métodos de Base de datos estándar que se describen aquí, ya sea desde su clase de controlador o su clase de modelo
- Puede indicarle al método de carga del modelo que se conecte automáticamente pasando TRUE a través del tercer parámetro, y se usarán las configuraciones de conectividad, como se define en el archivo de configuración de su Base de datos:

```
$this->load->model('nombre_modelo', '', TRUE);
```

- Puede pasar manualmente la configuración de conectividad de la Base de datos a través del tercer parámetro:

```
$config['hostname'] = 'localhost';
$config['username'] = 'mi_nombre_de_usuario';
$config['password'] = 'mi_contraseña';
$config['database'] = 'mi_base_de_datos';
$config['dbdriver'] = 'mysqli';
```

```
$config['dbprefix'] = '';
$config['pconnect'] = FALSE;
$config['db_debug'] = TRUE;

$this->load->model('nombre_modelo', '', $config);
```

Helpers - Ayudantes

Los ayudantes (Helpers), como su nombre sugiere, lo ayudan con las tareas, cada archivo helper es simplemente una colección de funciones en una categoría particular, hay **URL Helpers**, que ayudan a crear enlaces, hay **Form Helpers** que lo ayudan a crear elementos de formulario, **Text Helpers** que realizan varias rutinas de formateo de texto, **Cookie Helpers** que configuran y leen cookies, **File Helpers** que lo ayudan a manejar archivos, etc.

A diferencia de la mayoría de los otros sistemas en CodeIgniter, los Helpers no están escritos en un formato orientado a objetos, son simples funciones de procedimiento, cada función auxiliar realiza una tarea específica, sin dependencia de otras funciones

CodeIgniter no carga archivos auxiliares de forma predeterminada, por lo que el primer paso para usar un Helper es cargarlo, una vez cargado, se vuelve disponible globalmente en su controlador y vistas

Los helpers normalmente se almacenan en el directorio **system/helpers**, o en **application/helpers**, CodeIgniter buscará primero en el directorio **application/helpers**, si el directorio no existe o el asistente específico no está ubicado allí, CodeIgniter buscará en el directorio **system/helpers/**

Cargar un Helper

Cargar un helper es bastante simple usando el siguiente método:

```
$this->load->helper('nombre');
```

Donde '**nombre**' es el nombre de archivo del helper, sin la extensión de archivo .php o la parte "**helper**"

Por ejemplo, para cargar el archivo URL Helper, que se llama **url_helper.php**, haría esto:

```
$this->load->helper('url');
```

Un helper puede cargarse en cualquier lugar dentro de los métodos de su controlador, o incluso dentro de sus archivos de vistas, aunque no es una buena práctica, siempre que lo cargue antes de usarlo, puede cargar sus helpers en el constructor de su controlador para que estén disponibles automáticamente en cualquier método, o puede cargar un helper en un método específico que lo necesite

Nota

El método de carga del helper de arriba no devuelve un valor, por lo tanto, no intente asignarlo a una variable, solo úselo como se muestra

Cargando múltiples helpers

Si necesita cargar más de un helper, puede especificarlos en un array, como este:

```
$this->load->helper(
  array('helper1', 'helper2', 'helper3')
);
```

Carga automática de helpers

Si encuentra que necesita disponer de un helper globalmente en toda la aplicación, puede indicarle a CodeIgniter que lo cargue automáticamente durante la inicialización del sistema, esto se hace abriendo el archivo **application/config/autoload.php** y agregando el helper al array de autocarga

Usar un helper

Una vez que haya cargado el helper que contiene la función que quiere usar, lo llamará como lo haría con una función PHP estándar

Por ejemplo, para crear un enlace usando la función **anchor()** en uno de sus archivos de vista, haría esto:

```
<?php echo anchor('blog/comentarios', 'Clic aquí');?>
```

Donde "**Clic aquí**" es el nombre del enlace, y "**blog/comentarios**" es el URI al **controlador/método** que desea enlazar

"Extender" los Helpers

Para "**extender**" los Helpers, cree un archivo en el directorio **application/helpers/** con un nombre idéntico al helper existente, pero con el prefijo **MY_** (este elemento es configurable. Ver más abajo)

Si todo lo que necesita hacer es agregar alguna funcionalidad a un Helper existente, tal vez agregar una función o dos, o cambiar la forma en que funciona un helper en particular, entonces es excesivo reemplazar el asistente completo con su versión, en este caso, es mejor simplemente "**extender**" el Helper

Nota

El término "extender" se usa de manera vaga porque las funciones de Helper son procedimentales y discretas y no se pueden extender en el sentido programático tradicional, internamente, esto le da la capacidad de agregar o reemplazar las funciones que proporciona un Helper

Por ejemplo, para extender el **Array Helper** nativo, creae un archivo llamado a**pplication/helpers/MY_array_helper.php** y agrege o anule funciones:

```
// any_in_array() no está en Helper Array, por lo que define una nueva función

function any_in_array($needle, $haystack)
{
   $needle = is_array($needle) ? $needle : array($needle);
```

```
    foreach ($needle as $item)
    {
       if (in_array($item, $haystack))
        {
           return TRUE;
        }
    }

    return FALSE;
}

// random_element() está incluido en Helper Array, por lo que anula la función nativa
function random_element($array)
{
    shuffle($array);
    return array_pop($array);
}
```

Establecer su propio prefijo

El prefijo de nombre de archivo para "**extender**" Helpers es el mismo usado para extender bibliotecas y clases principales

Para configurar su propio prefijo, abra su archivo **application/config/config.php** y busque este elemento:

```
$config['subclass_prefix'] = 'MY_';
```

Tenga en cuenta que todas las bibliotecas nativas de CodeIgniter llevan el prefijo **CI_** por lo que **NO** lo use como su prefijo

¿Y Ahora Qué?

En el índice, encontrará una lista de todos los helper disponibles, examine cada uno para ver lo que hacen

Usar las Bibliotecas de CodeIgniter

Todas las bibliotecas disponibles se encuentran en su directorio **system/libraries/**, en la mayoría de los casos, utilizar una de estas clases implica iniciarla dentro de un controlador utilizando el siguiente método:

```
$this->load->library('nombre_de_clase');
```

Donde **'nombre_de_clase'** es el nombre de la clase que desea invocar

Por ejemplo, para cargar la **Form Validation Library**, debe hacer esto:

```
$this->load->library('form_validation');
```

Una vez iniciada, puede usarla como se indica en la página de la guía del usuario correspondiente a esa clase, además, se pueden cargar múltiples bibliotecas al mismo tiempo pasando un array de bibliotecas al método de carga:

```
$this->load->library(array('email', 'table'));
```

Crear Bibliotecas

Cuando usamos el término **"Bibliotecas"**, normalmente nos referimos a las clases que se encuentran en el directorio **system/libraries**, y que se describen en la **Referencia de Bibliotecas** (Libraries) de esta guía del usuario, sin embargo, en este caso, describiremos cómo puede crear sus propias bibliotecas dentro del directorio **application/libraries** para mantener la separación entre sus recursos locales y los recursos globales del framework

Como una ventaja adicional, CodeIgniter permite que sus bibliotecas amplíen las clases nativas si necesita agregar alguna funcionalidad a una biblioteca existente, o incluso puede reemplazar las bibliotecas nativas colocando versiones con nombres idénticos en su directorio **application/libraries**

En resumen:

- Puede crear bibliotecas completamente nuevas
- Puede extender las bibliotecas nativas
- Puede reemplazar las bibliotecas nativas

A continuación se explican estos tres conceptos en detalle

Nota

Las clases de Base de datos '**database**' no se pueden ampliar ni reemplazar con sus propias clases., las otras clases pueden ser reemplazadas/extendidas

Almacenar bibliotecas

Sus clases de biblioteca deben ubicarse dentro de su directorio **application/libraries**, ya que es aquí donde CodeIgniter las buscará cuando se inicialicen

Convenciones de nombres

- Los nombres de archivo deben estar en mayúscula, por ejemplo: Mi_clase.php
- Las declaraciones de clase deben estar en mayúscula, por ejemplo: clase Mi_clase
- Los nombres de clases y archivos deben coincidir

El archivo de clase

Las clases deben tener este prototipo básico:

```
<?php

defined('BASEPATH') OR exit('No se permite el acceso directo al script');

class Alguna_clase {

  public function algun_metodo()
  {
  }
}
```

Nota

Se utiliza el nombre Alguna_clase puramente como un ejemplo

Usar su clase

Desde cualquiera de sus métodos de Controlador puede iniciar su clase utilizando el estándar:

```
$this->load->library('aguna_clase');
```

Donde **alguna_clase** es el nombre del archivo, **sin la extensión** de archivo **".php"**, puede enviar el nombre del archivo en mayúscula o minúscula, a CodeIgniter no le importa

Una vez cargado, puede acceder a su clase con la versión en minúscula:

```
// Las instancias de los objetos siempre estarán en minúsculas
$this->alguna_clase->algun_metodo();
```

Pasar parámetros al inicializar su clase

En el método de carga de la biblioteca, puede pasar dinámicamente datos como un array mediante el segundo parámetro, el cual se pasará al constructor de su clase:

```
$params = array('tipo'=>'grande', 'color'=>'rojo');

$this->load->library('alguna_clase', $params);
```

Si usa esta característica, debe configurar su constructor de clase para esperar datos:

```
<?php

defined('BASEPATH') OR exit('No se permite el acceso directo al script');

class Alguna_clase{
```

```
    public function __construct($params)
    {
        // Hacer algo con $params
    }
}
```

También puede pasar parámetros almacenados en un archivo de configuración, cree un archivo de configuración con el mismo nombre que el nombre de archivo de la clase y guárdelo en el directorio **application/config/,** tenga en cuenta que si pasa dinámicamente los parámetros como se describe anteriormente, la opción del archivo de configuración no estará disponible

Utilizar recursos CodeIgniter dentro de su biblioteca

Para acceder a los recursos nativos de CodeIgniter dentro de su biblioteca, use el método **get_instance()**, este método devuelve el **súper objeto de CodeIgniter**

Normalmente, desde sus métodos de controlador, llamará a cualquiera de los métodos de CodeIgniter disponibles utilizando la construcción **$this**:

```
$this->load->helper('url');
$this->load->library('session');
$this->config->item('base_url');
// etc.
```

$this, sin embargo, solo funciona directamente dentro de sus controladores, sus modelos o sus vistas

Si desea utilizar las clases de CodeIgniter desde sus propias clases personalizadas, puede hacerlo de la siguiente manera:

Primero, asigne el objeto CodeIgniter a una variable:

```
$CI = &get_instance();
```

Una vez que haya asignado el objeto a una variable, usará esa variable en lugar de **$this**:

```
$CI =&get_instance();


$CI->load->helper('url');
$CI->load->library('session');
$CI->config->item('base_url');
// etc.
```

Nota

Notará que la función **get_instance()** anterior se pasa por referencia:

```
$CI = &get_instance();
```

Esto es muy importante, asignar por referencia le permite usar el objeto CodeIgniter original en lugar de crear una copia del mismo

Sin embargo, dado que una biblioteca es una clase, sería mejor si aprovecha al máximo los principios de POO

Por lo tanto, para poder utilizar el **súper objeto** de CodeIgniter en todos los métodos de su clase, se recomienda que **lo asigne a una propiedad** :

```
class Example_library {

  protected $CI;

  // Usar el constructor, cuando no se puede llamar directamente
  // en la definición de la propiedad
  public function __construct()
  {
     // Asignar el súper objeto de CodeIgniter
     $this->CI =&get_instance();
  }

  public function foo()
  {
     $this->CI->load->helper('url');
     redirect();
  }

  public function bar()
  {
     echo $this->CI->config->item('base_url');
  }

}
```

Reemplazar bibliotecas nativas con sus versiones

Puede reemplazar las bibliotecas nativas nombrando sus archivos de clase de manera idéntica a una biblioteca nativa, Codeigniter la usa en lugar de la nativa, para usar esta característica, debe nombrar el archivo y la declaración de clase exactamente igual que la biblioteca nativa

Por ejemplo, para reemplazar la biblioteca de correo electrónico nativa, creará un archivo llamado **application/libraries/Email.php** y declarará su clase con:

```
clase CI_Email {

}
```

Tenga en cuenta que la mayoría de las clases nativas tienen el prefijo **CI_**

Para cargar su biblioteca, utilizará el método de carga estándar:

```
$this->load->library('email');
```

Nota

En este momento, las clases de la Base de datos '**database**' no se pueden reemplazar con sus propias versiones

Extender bibliotecas nativas

Si todo lo que necesita hacer es agregar alguna funcionalidad a una biblioteca existente, tal vez agregar un método o dos, entonces es excesivo reemplazar toda la biblioteca con su versión, en este caso, es mejor simplemente extender la clase, extender una clase es casi idéntico a reemplazar una clase con un par de excepciones:

- La declaración de clase debe extender la clase principal
- Su nuevo nombre de clase y nombre de archivo debe ir precedido de **MY_** (este elemento es configurable. Ver a continuación).

Por ejemplo, para extender la clase nativa de correo electrónico '**Email**' , creará un archivo llamado **application/libraries/MY_Email.php** y declarará su clase así:

```
clase MY_Email extends CI_Email {

}
```

Si necesita usar un constructor en su clase, asegúrese de extender el constructor padre:

```
class MY_Email extends CI_Email {

  public function __construct($config = array())
  {
     parent::__construct($config);
     // Su código de constructor
  }
}
```

Nota

No todas las bibliotecas tienen los mismos parámetros, o ninguno, en su constructor, primero consulte la biblioteca que está ampliando para ver cómo debe implementarse

Cargar su subclase

Para cargar su subclase, usará la sintaxis estándar normalmente utilizada, **NO incluya su prefijo**

Por ejemplo, para cargar el ejemplo anterior, que amplía la clase de correo electrónico, use:

```
$this->load->library('email');
```

Una vez cargada, use la variable de clase como lo haría normalmente para la clase que está ampliando, en el caso de la clase **email**, en todas las llamadas usará:

```
$this->email->algun_metodo();
```

Establecer su propio prefijo

Para configurar su propio prefijo de subclase, abra su archivo **application/config/config.php** y busque este elemento:

```
$config['subclass_prefix'] = 'MY_';
```

Tenga en cuenta que todas las bibliotecas nativas de CodeIgniter llevan el prefijo **CI_** por lo que **NO lo use** como su prefijo

Drivers de CodeIgniter

Los drivers son un tipo especial de biblioteca que tiene una clase padre y cualquier cantidad de clases hijas, las clases hijas tienen acceso a la clase padre, pero no a sus hermanas, los drivers proporcionan una sintaxis elegante en sus controladores de bibliotecas que se benefician o requieren dividirse en clases discretas

Los drivers se encuentran en el directorio **system/libraries/**, en su propio subdirectorio, que se llama de forma idéntica a la clase de biblioteca padre, también dentro de ese directorio hay un subdirectorio llamado **drivers**, que contiene todos los posibles archivos de clases hijas

Para usar un driver, lo inicializará dentro de un controlador utilizando el siguiente método de inicialización:

```
$this->load->driver('nombre_de_clase');
```

Donde **nombre_de_clase** es el nombre de la clase del driver que desea invocar

Por ejemplo, para cargar un driver llamado **'algun_padre '**, haría esto:

```
$this->load->driver('algun_padre');
```

Los métodos de esa clase se pueden invocar con:

```
$this->algun_padre->algun_metodo();
```

Las clases secundarias, los drivers mismos, pueden llamarse directamente a través de la clase principal, sin iniciarlas:

```
$this->algun_padre->hija_una->algun_metodo();

$this->algun_padre->hija_dos->otro_metodo();
```

Crear drivers

Directorio de drivers y estructura de archivos

Ejemplo de directorio de controladores y estructura de archivos:

```
/application/libraries/Nombre_de_driver
   Nombre_de_driver.php
      drivers
         Nombre_de_driver_subclase_1.php
         Nombre_de_driver_subclase_2.php
         Nombre_de_driver_subclase_3.php
```

Nota

Para mantener la compatibilidad en los sistemas de archivos sensibles a mayúsculas y minúsculas, el directorio **nombre_de_driver** debe nombrarse en el formato devuelto por **ucfirst()**

Nota

La arquitectura de la biblioteca **Driver** es tal que las subclases no se extienden y, por lo tanto, **no heredan propiedades** o métodos del driver principal

Crear clases del núcleo

Cada vez que se ejecuta CodeIgniter, hay varias clases base que se inicializan automáticamente como parte del núcleo central del framework, sin embargo, es posible intercambiar cualquiera de las clases principales del sistema con sus propias versiones o incluso extender las versiones del núcleo

La mayoría de los usuarios nunca tendrán la necesidad de hacer esto, pero existe la opción de reemplazarlos o ampliarlos para aquellos que deseen alterar significativamente el núcleo de CodeIgniter

Nota
Jugar con una clase del núcleo central tiene muchas implicaciones, así que asegúrese de saber lo que hace antes de intentarlo

Lista de las clases del núcleo

La siguiente es la lista de los archivos del núcleo del sistema que se invocan cada vez que se ejecuta CodeIgniter:

- Benchmark
- Config
- Controller
- Exceptions
- Hooks
- Input
- Language
- Loader
- Log
- Output
- Router
- Security
- URI
- Utf8

Reemplazar las clases del núcleo

Para usar una de sus propias clases de sistema en lugar de una predeterminada, coloque su versión dentro de su directorio **application/core/**:

```
application/core/alguna_clase.php
```

Si este directorio no existe, puede crearlo, cualquier archivo nombrado del mismo modo que uno de la lista anterior, se usará en lugar del original, tenga en cuenta que su clase debe usar **CI** como prefijo
Por ejemplo, si su archivo se llama **Input.php**, la clase se denominará:

```
class CI_Input {
}
```

Extender las clases del núcleo

Si todo lo que necesita hacer es agregar alguna funcionalidad a una biblioteca existente, tal vez agregar un método o dos, entonces es excesivo reemplazar toda la biblioteca con su versión, en este caso, es mejor simplemente extender la clase, extender una clase es casi idéntico a reemplazar una clase con un par de excepciones:

La declaración de clase debe extender la clase principal

Su nuevo nombre de clase y nombre de archivo debe ir precedido de **MY_** (este elemento es configurable, consulte a continuación)

Por ejemplo, para extender la clase **Input class** nativa, creará un archivo llamado **application/core/MY_Input.php** y declarará su clase con:

```
class MY_Input extends CI_Input {
}
```

Nota

Si necesita usar un constructor en su clase, asegúrese de extender al constructor padre:

```
class MY_Input extends CI_Input {

  public function __construct()
  {
     parent::__construct();
     // Su código de constructor
  }
}
```

Sugerencia: Se utilizarán todas las funciones de su clase que tengan un nombre idéntico a los métodos de la clase principal en lugar de las nativas, esto se conoce como "anulación de método", esto le permite alterar sustancialmente el núcleo de CodeIgniter

Si está ampliando la clase **Controller** del núcleo, asegúrese de extender su nueva clase en los constructores de su controlador de aplicación:

```
class Welcome extends MY_Controller {

  public function index()
  {
     $this->load->view('welcome_message');
  }
}
```

Establecer su propio prefijo

Para configurar su propio prefijo de subclase, abra su archivo **application/config/config.php** y busque este elemento:

```
$config['subclass_prefix'] = 'MY_';
```

Tenga en cuenta que todas las bibliotecas nativas de CodeIgniter llevan el prefijo **CI_** por lo que **NO lo use como prefijo**

Crear clases auxiliares

En algunos casos, es posible que desee desarrollar clases que existan aparte de sus controladores, pero que tengan la capacidad de utilizar todos los recursos de CodeIgniter, esto es posible fácilmente

get_instance()

object get_instance()

Valores devueltos

Referencia a la instancia de su controlador

Tipo de devolución

CI_Controller

Cualquier clase que instancia en los métodos de su controlador puede acceder a los recursos nativos de CodeIgniter usando la función **get_instance()**, esta función devuelve el objeto CodeIgniter principal

Normalmente, para llamar a cualquiera de los métodos disponibles, CodeIgniter requiere que use **$this**:

```
$this->load->helper('url');
$this->load->library('session');
$this->config->item('base_url');
// etc.
```

$this, sin embargo, solo funciona dentro de sus controladores, sus modelos o sus vistas, si desea utilizar las clases de CodeIgniter desde sus propias clases personalizadas, puede hacerlo de la siguiente manera:

Primero, asigne el objeto CodeIgniter a una variable:

```
$CI = &get_instance();
```

Una vez que haya asignado el objeto a una variable, usará esa variable en lugar de **$this**:

```
$CI = &get_instance();

$CI->load->helper('url');
$CI->load->library('session');
$CI->config->item('base_url');
// etc.
```

Si va a usar **get_instance()** dentro de otra clase, entonces sería mejor si lo asigna a una propiedad, de esta forma, no necesitará llamar a **get_instance()** en cada método

Ejemplo

```
class Example {

  protected $CI;
```

```
    // Usar el constructor, cuando no se puede llamar directamente en la definición de la propiedad
    public function __construct()
    {
        // Asignar el súper objeto de CodeIgniter
        $this->CI =& get_instance();
    }
    public function foo()
    {
        $this->CI->load->helper('url');
        redirect();
    }

    public function bar()
    {
        $this->CI->config->item('base_url');
    }
}
```

En el ejemplo anterior, ambos métodos **foo()** y **bar()** funcionarán después de crear una instancia de la clase Example, sin la necesidad de llamar a **get_instance()** en cada uno de ellos

Hooks - ampliar el núcleo del Framework

Los **Hooks** de CodeIgniter proporcionan un medio para acceder y modificar el funcionamiento interno del framework sin alterar los archivos del núcleo

Cuando CodeIgniter se ejecuta, sigue un proceso de ejecución específico, ver el diagrama en el cápitulo **Diagrama de flujo de aplicaciones**, sin embargo, puede querer que desee que se lleve a cabo alguna acción en una etapa particular del proceso de ejecución

Por ejemplo, es posible que desee ejecutar un script justo antes de que sus controladores se carguen, o inmediatamente después, o puede que desee activar uno de sus propios scripts en alguna otro momento

Habilitar hooks

La función de Hooks se puede habilitar/deshabilitar globalmente configurando el siguiente elemento en el archivo **application/config/config.php**:

```
$config['enable_hooks'] = TRUE;
```

Definir un hook

Los Hooks se definen en el archivo **application/config/hooks.php**

Cada hook se especifica como un array con este prototipo:

```
$hook['pre_controller'] = array(
   'class'    => 'Mi_clase',
   'function' => 'Mi_metodo',
   'filename' => 'Mi_clase.php',
   'filepath' => 'hooks',
   'params'   => array('cerveza', 'tintorro', 'pocholate')
);
```

Nota

El índice del array se corresponde con el nombre del punto de enlace particular que desea usar, en el ejemplo anterior, el punto de enlace es **pre_controller**, una lista de puntos de enlace se encuentra a continuación

Los siguientes elementos se deben definir en su array asociativo de hooks:

- **class**: Nombre de la clase que desea invocar. Si prefiere usar una función de procedimiento en lugar de una clase, deje este elemento en blanco
- **function**: El nombre de la función, o método, que desea llamar
- **filename**: Nombre del archivo que contiene su clase/función

- **filepath**: El nombre del directorio que contiene su script. Nota: Su script debe estar ubicado en un directorio **DENTRO** de su directorio **application/**, por lo que la ruta del archivo es relativa a ese directorio. Por ejemplo, si su script se encuentra en **application/hooks/**, usará **'hooks'** como su ruta de archivo, si su script se encuentra en **application/hooks/utilities/** usará **'hooks/utilities'** como su ruta de archivo, sin barra al final
- **params**: Cualquier parámetro que quiera pasar a su script, este elemento es opcional

También puede usar funciones lambda/anoymous (o cierres) como hooks, con una sintaxis más simple:

```
$hook['post_controller'] = function()
{
   /* Hacer algo aquí */
};
```

Varias llamadas al mismo hook

Si desea utilizar el mismo hook con más de un script, simplemente haga que su declaración de array sea multidimensional, como esta:

```
$hook['pre_controller'] = array(
  'class'    => 'Mi_clase',
  'function' => 'Mi_metodo',
  'filename' => 'Mi_clase.php',
  'filepath' => 'hooks',
  'params'   => array('cerveza', 'tintorro', 'pocholate')
);

$hook['pre_controller'][] = array(
  'class'    => 'Mi_otra_clase',
  'function' => 'Mi_otro_metodo',
  'filename' => 'Mi_otra_clase.php',
  'filepath' => 'hooks',
  'params'   => array('amarillo', 'rojo', 'violeta')
);
```

Observe los corchetes después de cada índice del array:

```
$hook['pre_controller'][]
```

Esto le permite tener el mismo hook con varias secuencias de comandos, el orden en que define su array será el orden de ejecución

Puntos de hook

La siguiente es una lista de hooks disponibles:

- **pre_system**: Llamado muy pronto durante la ejecución del sistema, solo la clase **benchmark** y **hooks** se cargaron en este punto, no se ha enrutado ni abierto otros procesos

- **pre_controller**: Se llama inmediatamente antes de llamar a cualquiera de sus controladores, se han realizado todas las clases base, enrutamiento y comprobaciones de seguridad
- **post_controller_constructor**: Se llama inmediatamente después de que se crea una instancia de su controlador, pero antes de que ocurra cualquier llamada a un método
- **post_controller**: Llamado inmediatamente después de que su controlador se haya ejecutado por completo
- **display_override**: Reemplaza el método **_display()**, utilizado para enviar la página finalizada al navegador web al final de la ejecución del sistema, esto le permite usar su propia metodología de visualización, tenga en cuenta que necesitará hacer referencia al superobjeto de **CI** con **$this->CI =& get_instance()** y luego los datos finalizados estarán disponibles llamando a **$this->CI->output->get_output()**
- **cache_override**: Le permite llamar a su propio método en lugar del método **_display_cache()** en la **Output Library**, esto le permite usar su propio mecanismo de visualización de caché
- **post_system**: Llamado después de que la página final renderizada se envía al navegador, al final de la ejecución del sistema y después de que los datos finalizados se envían al navegador

Carga automática de recursos

CodeIgniter viene con una función de "**carga automática**" que permite que las bibliotecas, ayudantes y modelos se inicialicen automáticamente cada vez que se ejecuta el sistema, si necesita ciertos recursos globalmente en su aplicación, debería considerar cargarlos automáticamente para mayor comodidad

Los siguientes elementos se pueden cargar automáticamente:

- **Clases** que se encuentran en el directorio **libraries/**
- **Helpers** que se encuentran en el directorio **helpers/**
- **Archivos de configuración** personalizados que se encuentran en el directorio **config/**
- **Archivos de idioma** que se encuentran en el directorio **system/language/**
- **Modelos** que se encuentran en el directorio **models/**

Para cargar recursos automáticamente, abra el archivo **application/config/autoload.php** y agregue el elemento que desea cargar en el array de autocarga, encontrará instrucciones en el archivo correspondientes a cada tipo de elemento

Nota
No incluya la extensión de archivo (.php) cuando agregue elementos al array de autocarga

Además, si desea que CodeIgniter use el autocargador de **Composer**, simplemente configure **config['composer_autoload']** a TRUE o personalice la ruta en **application/config/config.php**

Funciones Comunes

CodeIgniter utiliza algunas funciones para su funcionamiento que están definidas globalmente, y están disponibles para usted en cualquier momento, estas no requieren cargar ninguna biblioteca o helper

config_item()

mixed config_item(string $key)

Descripción

La biblioteca de configuración es la forma preferida de acceder a la información de configuración, sin embargo, **config_item()** se puede usar para recuperar claves individuales, ver la documentación de **Config Library** para más información

Parámetros

key
Clave del elemento de configuración

Valores devueltos

Valor de la clave de configuración o NULL si no se encuentra

function_usable()

bool function_usable(string $function_name)

Descripción

Devuelve TRUE si existe una función y es utilizable, FALSE de lo contrario

Esta función ejecuta una comprobación **function_exists()** y si la extensión **Suhosin** <**http://www.hardened-php.net/suhosin/**> está cargada, comprueba si no desactiva la función que se está comprobando

Es útil si desea verificar la disponibilidad de funciones como **eval()** y **exec()**, que son peligrosas y pueden estar deshabilitadas en servidores con políticas de seguridad altamente restrictivas

Parámetros

function_name
Nombre de la función

Valores devueltos

TRUE si la función se puede usar, FALSE si no

Nota

Esta función se introdujo porque Suhosin terminó la ejecución del script, pero resultó ser un error, una solución ha estado disponible durante algún tiempo (versión 0.9.34), pero desafortunadamente aún no se ha lanzado

get_mimes()

array get_mimes()

Descripción

Esta función devuelve una referencia a la array **MIME** desde **application/config/mimes.php**

Valores devueltos

Array asociativo de tipos de archivos

html_escape()

mixed html_escape(mixed $var)

Descripción

Esta función actúa como un alias para la función **htmlspecialchars()** nativa de PHP, con la ventaja de poder aceptar un array de strings, es útil para evitar **Cross Site Scripting** (XSS)

Parámetros

var

Variable para escapar, string o array

Valores devueltos

String HTML escapada

is_cli()

bool is_cli()

Descripción

Devuelve TRUE si la aplicación se ejecuta a través de la línea de comandos y FALSE si no

Nota
Esta función comprueba si el valor PHP_SAPI es 'cli' o si se define la constante STDIN

is_https()

bool is_https()

Descripción

Devuelve TRUE si se utiliza una conexión segura **HTTPS** y FALSE en cualquier otro caso, incluidas las solicitudes que no son HTTP

Devuelve

TRUE si actualmente usa HTTP-con-SSL y FALSE si no lo usa

is_php()

bool is_php(string $version)

Descripción

Determina si la versión de PHP utilizada es mayor que el número de versión proporcionado

Parámetros

version

Número de versión

Valores devueltos

Devuelve TRUE si la versión de PHP instalada es igual o mayor que el número de versión proporcionado, FALSE si la versión de PHP instalada es menor que el número proporcionado

Ejemplo

```
if (is_php('5.3'))
{
   $str = quoted_printable_encode($str);
}
```

is_really_writable()

bool is_really_writable(string $file)

Descripción

Esta función determina si un archivo es realmente modificable al intentar escribirlo primero, por lo general, solo se recomienda en plataformas donde esta información puede no ser confiable

Parámetros

file

Ruta del archivo

Valores devueltos

TRUE si la ruta es escribible, FALSE si no, **is_writable ()** devuelve TRUE en los servidores de Windows cuando realmente no puede escribir en el archivo ya que el sistema operativo informa a PHP como FALSE solamente si el atributo de solo lectura está marcado

Ejemplo

```
if (is_really_writable('file.txt'))
{
   echo "Puedo escribir lo que quiera";
}
else
{
   echo "No se puede escribir en el archivo";
}
```

> **Nota**
>
> Vea también el error PHP # 54709 para más información

remove_invisible_characters()

string remove_invisible_characters(string $str [, bool $url_encoded = TRUE])

Descripción

Esta función evita insertar caracteres NULL entre caracteres ASCII, como **Java\0script**

Parámetros

str

String de entrada

url_encoded

TRUE elimina también los caracteres codificados en URL

Valores devueltos

String desinfectada

Ejemplo

```
remove_invisible_characters('Java\\0script');  // Retorna 'Javascript'
```

set_status_header()

void set_status_header(int $code [, string $text = ''])

Descripción

Le permite configurar manualmente un encabezado de estado del servidor

Parámetros

code

Código de estado de respuesta HTTP

text

Un mensaje personalizado para configurar con el código de estado

valores devueltos

void

Ejemplo

```
set_status_header(401); // Establece el encabezado como: no autorizado
```

Vea: **http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html** para una lista completa de encabezados

Funciones de compatibilidad

CodeIgniter proporciona un conjunto de funciones de compatibilidad que le permiten usar funciones que de otra manera estarían disponibles de forma nativa en PHP, pero solo en versiones superiores o dependiendo de cierta extensión

Al ser implementaciones personalizadas, estas funciones también tendrán un conjunto de dependencias propias, pero siguen siendo útiles si su configuración de PHP no las ofrece de forma nativa

Nota

Al igual que las funciones comunes, las funciones de compatibilidad siempre están disponibles, siempre que se cumplan sus dependencias

Hashing de contraseña

Este conjunto de funciones de compatibilidad ofrece un **"backport"** de la extensión de hashing de contraseña estándar de PHP que de otra manera solo está disponible desde PHP 5.5

Dependencias: PHP 5.3.7 Soporte de **CRYPT_BLOWFISH** para **crypt()**

Constantes

PASSWORD_BCRYPT
PASSWORD_DEFAULT

password_get_info()

array password_get_info(string $hash)

Descripción

Información sobre la contraseña hash, puede obtener más información consultando el manual de PHP para **password_get_info()**

Parámetros

hash
Hash de la contraseña

Valores devueltos

Información sobre la contraseña hash

password_hash()

string password_hash(string $password, int $algo [, array $options = array()])

Descripción

Consulte el manual de PHP para **password_hash()**

Parámetros

password
: Contraseña en texto plano

algo
: Algoritmo hashing

options
: Opciones de hashing

Valores devueltos

Contraseña hash o FALSE en caso de error

> ***Nota***
>
> A menos que proporcione su propia, y válida, salt, esta función tiene una dependencia adicional de una fuente CSPRNG (Generador de números pseudo-aleatorios) disponible
>
> Cada una de las siguientes opciones satisfaría que:
> mcrypt_create_iv() con MCRYPT_DEV_URANDOM - openssl_random_pseudo_bytes() - /dev/arandom - /dev/urandom

password_needs_rehash()

bool password_needs_rehash(string $hash, int $algo [, array $options])

Descripción

Consulte el manual de PHP para **password_needs_rehash()**

Parámetros

hash
: Hash de la contraseña

algo
: Algoritmo Hashing

options
: Opciones de hashing

Valores devueltos

TRUE si el hash se debe volver a generar para coincidir con el algoritmo y las opciones dados, FALSE en caso contrario

password_verify()

bool password_verify(string $password, string $hash)

Descripción

Comprueba que la contraseña coincida con un hash, consulte el manual de PHP para **password_verify()**

Parámetros

password
: Contraseña de texto plano

hash

Hash de la contraseña

Valores devueltos

TRUE si la contraseña coincide con el hash, FALSE si no

Hash (Message Digest)

Esta capa de compatibilidad contiene backports para las funciones **hash_equals()** y **hash_pbkdf2()**, que de lo contrario requieren PHP 5.6 y / o PHP 5.5 respectivamente

Dependencias : Ninguna

hash_equals()

string hash_equals(string $known_string, string $user_string)

Descripción

Consulte el manual de PHP para **hash_equals()**

Parámetros

known_string

Cadena conocida

user_string

Cadena proporcionada por el usuario

Valores devueltos

TRUE si las strings coinciden, de lo contrario FALSE

hash_pbkdf2()

string hash_pbkdf2(string $algo, string $password, string $salt, int $iterations [, int $length = 0 [, bool $raw_output = FALSE]])

Descripción

Consulte el manual de PHP para **hash_pbkdf2()**

Parámetros

algo

Algoritmo Hashing

password

Contraseña

salt

Salt hash

iterations

Número de iteraciones a realizar durante la derivación

length

Longitud del string de salida

raw_output

Si se deben devolver datos binarios en bruto

Valores devueltos

Clave derivado de contraseña o FALSE en caso de error

Multibyte String

Este conjunto de funciones de compatibilidad ofrece soporte limitado para la extensión de strings multibyte de PHP, debido a las soluciones alternativas limitadas, solo unas pocas funciones están disponibles

> ***Nota***
>
> Cuando se omite un parámetro de conjunto de caracteres, se usará **$config['charset']**

Dependencias: extensión **iconv** - **http://php.net/iconv**

> ***Importante***
>
> Esta dependencia es opcional y estas funciones siempre serán declaradas, si iconv no está disponible, VOLVERÁN a sus versiones no mbstring

> ***Importante***
>
> Cuando se proporciona un conjunto de caracteres, debe estar respaldado por iconv y en un formato que reconozca

> ***Nota***
>
> Para su propia comprobación de dependencia en la extensión mbstring real, utilice la constante MB_ENABLED

mb_strlen()

string mb_strlen(string $str [, string $encoding = NULL])

Descripción

Consulte el manual de PHP para **mb_strlen()**

Parámetros

str

String de entrada

encoding

Conjunto de caracteres

Valores devueltos

Número de caracteres en la string de entrada o FALSE si falla

mb_strpos()

mixed mb_strpos(string $haystack, string $needle [, int $offset = 0 [, string $encoding = NULL]])

Descripción

Consulte el manual de PHP para **mb_strpos()**

Parámetros

haystack
: Cadena para buscar en

needle
: Parte de la string para buscar

offset
: Compensación de búsqueda

encoding
: Conjunto de caracteres

Valores devueltos

Posición del carácter numérico donde se encontró **$needle** o FALSE si no se encontró

mb_substr()

string mb_substr(string $str, int $start [, int $length = NULL [, string $encoding = NULL]])

Descripción

Consulte el manual de PHP para **mb_substr()**

Parámetros

str
: String de entrada

start
: Posición del primer personaje

length
: Número máximo de caracteres

encoding
: Conjunto de caracteres

Valores devueltos

Porción de **$str** especificada por **$start** y **$length** o FALSE cuando falla

Funciones estándar

Este conjunto de funciones de compatibilidad ofrece soporte para algunas funciones estándar en PHP que de otra manera requieren una versión PHP más reciente

Dependencias: Ninguna

array_column()

```
array array_column(array $array, mixed $column_key [, mixed $index_key = NULL])
```

Descripción

Consulte el manual de PHP para **array_column()**

Parámetros

array

Un array multidimensional o un array de objetos desde el que extraer una columna de valores

column_key

La columna de valores a devolver

index_key

La columna a usar como los índices/claves para el array devuelto

Valores devueltos

Devuelve un array de valores que representa una sola columna del array de entrada

hex2bin()

```
string hex2bin(array $data)
```

Descripción

Consulte el manual de PHP para **hex2bin()**

Parámetros

data

La representación hexadecimal de datos

Valores devueltos

Devuelve la representación binaria de los datos dados o FALSE en caso de error

Enrutamiento URI

Normalmente, existe una relación de uno a uno entre una string de URL y su clase/método de controlador correspondiente

Los segmentos en un URI normalmente siguen este patrón:

```
ejemplo.com/clase/funcion/id/
```

Sin embargo, en algunos casos, es posible que desee reasignar esta relación para poder llamar a una clase/método diferente en lugar de la correspondiente a la URL

Por ejemplo, supongamos que quiere que sus URL tengan este prototipo:

```
ejemplo.com/producto/1/
ejemplo.com/producto/2/
ejemplo.com/producto/3/
ejemplo.com/producto/4/
```

Normalmente, el segundo segmento de la URL está reservado para el nombre del método, pero en el ejemplo anterior, en cambio, tiene una identificación del producto, para superar esto, CodeIgniter le permite reasignar el controlador de URI

Estableciendo sus propias reglas de enrutamiento

Las reglas de enrutamiento se definen en su archivo **application/config/routes.php**, en él, verá una array llamado **$route** que le permite especificar sus propios criterios de enrutamiento, las rutas se pueden especificar utilizando comodines o expresiones regulares

Comodines

Una ruta típica de comodín puede verse más o menos así:

```
$route['producto/:num'] = "catalogo/buscar_producto";
```

En una ruta, la clave del array contiene el URI a coincidir, mientras que el valor del array contiene el destino al que debe redirigirse, en el ejemplo anterior, si la palabra literal **"producto"** se encuentra en el primer segmento de la URL, y un número en el segundo, en su lugar se usa la clase **"catálogo"** y el método "**buscar_producto"**

Puede hacer coincidir valores literales o puede usar dos tipos de comodines:

(**:num**) coincidirá con un segmento que contiene solo números

(**:any**) coincidirá con un segmento que contenga cualquier carácter, excepto '/', que es el delimitador de segmento

Nota

- Los comodines son en realidad alias para expresiones regulares, **(:any)** se corresponde con la expresión **[^/]+** y **(:num)** se corresponde con **[0-9]+**, respectivamente
- Las rutas se ejecutarán en el orden en que están definidas, las rutas más altas siempre tendrán prioridad sobre las más bajas
- **Las reglas de ruta no son filtros!** Establecer una regla por ejemplo **'catalogo/producto/(:num)'** no evitará que el controlador **Catalogo** y el método **producto** sean llamados con un valor no numérico si esa es una ruta válida

Ejemplos

Una URL que contiene la palabra **"diarios"** en el primer segmento, se remapeará a la clase **"blogs"**:

```
$route['diarios'] = "blogs";
```

Una URL que contiene los segmentos **blog/jose** se reasignará a la clase **"blogs"** y al método **"usuarios"**, el ID se establecerá como **"34"**:

```
$route['blog/jose'] = "blogs/usuarios/34";
```

Una URL con **"producto"** como primer segmento, y cualquier otra en el segundo se reasignará a la clase **"catálogo"** y al método **"buscar_producto"**:

```
$route['product/(:any)'] = 'catalog/buscar_producto';
```

Una URL con **"producto"** como primer segmento y cualquier cosa en el segundo, se reasignará a la clase **"catalogo"** y al método **"buscar_producto"**:

```
$route['producto/(:any)'] = "catalogo/buscar_producto";
```

Una URL con **"producto"** como primer segmento y un número en el segundo, se reasignará a la clase **"catalogo"** y al método **"producto_buscar_por_id"** pasándole una variable al método:

```
$route['producto/(:num)'] = "catalogo/producto_buscar_por_id/$1";
```

Importante

No use barras inclinadas / al final

Expresiones regulares

Si lo prefiere, puede usar expresiones regulares para definir sus reglas de enrutamiento, se permite cualquier expresión regular válida, al igual que las referencias anteriores

Nota

Si usa **back-references**, debe usar la sintaxis dólar en lugar de la sintaxis doble barra invertida

Una ruta típica de RegEx podría verse más o menos así:

```
$route['productos/([a-z]+)/(\d+)'] = "$1/id_$2";
```

En el ejemplo anterior, un URI similar a **productos/camisetas/123** llamaría en su lugar a la clase controlador **"camisetas"** y al método **"id_123"**

Con expresiones regulares, también puede atrapar varios segmentos a la vez, por ejemplo, si un usuario accede a un área protegida con contraseña de su aplicación web y desea poder redirigirlos a la misma página después de iniciar sesión, puede encontrar útil este ejemplo:

```
$route['login/(.+)'] = 'auth/login/$1';
```

Nota

En el ejemplo anterior, si el marcador de posición de **$1** contiene una barra inclinada, aún se dividirá en varios parámetros cuando se pase a **Auth::login()**

Para aquellos de ustedes que no conocen las expresiones regulares y quieren aprender más sobre ellas, **http://www.regular-expressions.info/** podría ser un buen punto de partida

Nota

También puede mezclar y combinar comodines con expresiones regulares

Callbacks

También puede usar callbacks en lugar de las reglas de enrutamiento normales para procesar las referencias anteriores:

Ejemplo

```
$route['productos/([a-zA-Z]+)/edit/(\d+)'] = function($tipo_de_producto, $id)
{
   return 'catalogo/editar_producto/'.strtolower($tipo_de_producto).'/'.$id;
};
```

Usar verbos HTTP en las rutas

Es posible usar verbos HTTP , método de solicitud, para definir sus reglas de enrutamiento, es particularmente útil cuando se construyen aplicaciones **RESTful**, puede usar verbos HTTP estándar (GET, PUT, POST, DELETE, PATCH) o uno personalizado como, por ejemplo PURGE, las reglas del verbo HTTP no distinguen entre mayúsculas y minúsculas

Todo lo que necesita hacer es agregar el verbo como una array a su ruta:

Ejemplos

```
$route['productos']['PUT'] = 'producto/insertar';
```

En el ejemplo anterior, una solicitud **PUT** para la URI **"productos"** llamaría al método de controlador **Producto::insertar()**

```
$route['productos/(:num)']['DELETE'] = 'producto/eliminar/$1';
```

Una solicitud **DELETE** a la URL **"productos"** ya que primero se asignará el segmento y un número en el segundo al método **Producto::eliminar()**, pasando el valor numérico como primer parámetro

Usar verbos HTTP es, por supuesto, opcional

Rutas Reservadas

Hay tres rutas reservadas:

```
$route['default_controller'] = 'welcome';
```

Esta ruta apunta a la acción que se debe ejecutar si el URI no contiene datos, que será el caso cuando las personas carguen su URL raíz

La configuración acepta un valor de **controlador/método** e **index()** sería el método predeterminado si no especifica uno, en el ejemplo anterior, se llamará a **Welcome::index()**

Nota

NO puede usar un **directorio** como parte de esa configuración !

Se recomienda que siempre tenga una ruta predeterminada, de lo contrario aparecerá una página 404 de forma predeterminada

```
$route['404_override'] = '';
```

Esta ruta indica qué controlador debe cargarse si no se encuentra el controlador solicitado, anula la página de error 404 predeterminada, las mismas reglas para directorio que con **'default_controller'** se aplican aquí también

No afectará a la función **show_404()**, que continua cargando el archivo predeterminado **error_404.php** que se encuentra en: **application/views/errors/error_404.php**

```
$route['translate_uri_dashes'] = FALSE;
```

Como es evidente por el valor booleano, esta no es exactamente una ruta, esta opción le permite reemplazar de forma automática guiones (**'-'**) con guiones bajos en el controlador y en los segmentos del URI de método, de este modo puede guardar entradas de ruta adicionales si necesita hacerlo, esto es obligatorio, porque el guión no es un nombre de clase o método válido y podría causar un error fatal si intenta usarlo

Manejo de errores

CodeIgniter le permite generar informes de errores en sus aplicaciones utilizando las funciones que se describen a continuación, además, tiene una clase de registro de errores que permite que los errores y los mensajes de depuración se guarden como archivos de texto

Nota

Por defecto, CodeIgniter muestra todos los errores de PHP, es posible que desee cambiar este comportamiento una vez que se complete su desarrollo

Encontrará la función **error_reporting()** ubicada en la parte superior de su archivo principal **index.php**

La desactivación de informes de errores **NO** evitará que se escriban archivos de registro si hay errores

A diferencia de la mayoría de los sistemas en CodeIgniter, las funciones de error son simples interfaces de procedimientos que están disponibles globalmente en toda la aplicación, este enfoque permite que los mensajes de error se activen sin tener que preocuparse por el alcance de clase/método

CodeIgniter también devuelve un código de estado cada vez que una parte del núcleo llama a **exit()**, este código de estado de salida es independiente del código de estado HTTP, y sirve como aviso para otros procesos que pueden estar observando si el script se completó con éxito o, en caso negativo, qué tipo de problema encontró que provocó que abortara

Estos valores se definen en **application/config/constants.php**, si bien los códigos de estado de salida son más útiles en la configuración de la **CLI**, devolver el código adecuado ayuda al software del servidor a realizar un seguimiento de sus scripts y el estado de su aplicación

Las siguientes funciones le permiten generar errores

log_message()

void log_message(string $level, string $message)

Descripción

Esta función le permite escribir mensajes en sus archivos de registro, debe proporcionar uno de los tres "niveles" en el primer parámetro, indicando qué tipo de mensaje es (**error**, **debug** o **info**), con el mensaje mismo en el segundo parámetro

Parámetros

level
: Nivel de registro: 'error', 'debug' o 'info'

message
: Mensaje para iniciar sesión

valores devueltos

void

Ejemplo

```
if ($alguna_variable == '')
{
   log_message('error', 'alguna_variable no contenía valor.');
}
else
{
   log_message('debug', 'alguna_variable está bien establecida');
}

log_message('info', 'El propósito de alguna_variable es proveer algún valor.');
```

Hay tres tipos de mensajes:

1. **Error Messages**: Estos son errores reales, como errores de PHP o de usuario
2. **Debug Messages**: Estos son mensajes que ayudan a la depuración, por ejemplo, si se ha inicializado una clase, puede registrarla como información de depuración
3. **Informational Messages**:Estos son los mensajes de menor prioridad, simplemente dan información con respecto a algún proceso

> ***Nota***
>
> Para que el archivo de registro se escriba realmente, el directorio **logs/** debe ser de escritura, además, debe establecer la clave **$config['log_threshold']** en **application/config/config.php**, puede, por ejemplo, solo querer que se registren los mensajes de error, y no los otros dos tipos, si lo configura en cero, el registro se desactivará

show_404()

void show_404(string $page = '', bool $log_error = TRUE)

Descripción

Esta función mostrará el mensaje de error 404 que se le proporcionó utilizando la plantilla de error adecuada para su ejecución:

```
application/views/errors/html/error_404.php

//o

application/views/errors/cli/error_404.php
```

Parámetros

page

String URI, la función espera que la string que se le pasa sea la ruta del archivo a la página que no se encuentra, el código de estado de salida se establecerá en **EXIT_UNKNOWN_FILE**, tenga en cuenta que CodeIgniter muestra automáticamente el mensaje 404 si no se encuentran los controladores

log_error

Si debe registrarse el error, CodeIgniter registra automáticamente cualquier llamada a **show_404()**, establecer el segundo parámetro opcional en FALSE omitirá el registro

valores devueltos

void

show_error()

void show_error(mixed $message, int $status_code, string $heading = 'An Error Was Encountered')

Descripción

Esta función mostrará el mensaje de error proporcionado utilizando la plantilla de error apropiada para su ejecución:

```
application/views/errors/html/error_general.php

//o

application/views/errors/cli/error_general.php
```

Parámetros

message

Mensaje de error

status_code

Código de estado de respuesta HTTP, el parámetro opcional **$status_code** determina qué código de estado HTTP se debe enviar con el error, si **$status_code** es menor que 100, el código de estado HTTP se establecerá en 500, y el código de estado de salida se establecerá en **status_code + EXIT__AUTO_MIN**, si ese valor es mayor que **EXIT__AUTO_MAX**, o si **$status_code** es 100 o superior, el código de estado de salida se establecerá en **EXIT_ERROR**, puede consultar en **application/config/constants.php** para más detalles

heading

Encabezado de la página de error

valores devueltos

void

Almacenamiento en caché de la página web

CodeIgniter le permite almacenar en caché sus páginas para lograr el máximo rendimiento

Aunque CodeIgniter es bastante rápido, la cantidad de información dinámica que muestre en sus páginas se correlacionará directamente con los recursos del servidor, la memoria y los ciclos de procesamiento utilizados, lo que afectará la velocidad de carga de su página

Si almacena en caché sus páginas, ya que se guardan en su estado completamente procesado, puede lograr un rendimiento cercano al de las páginas web estáticas

Cómo funciona el almacenamiento en caché?

El almacenamiento en caché se puede habilitar por página, y puede establecer el tiempo que una página debe permanecer en la memoria caché antes de actualizarse, cuando se carga una página por primera vez, el archivo de caché se escribirá en el directorio **application/cache**, en las cargas de página posteriores, el archivo de caché se recuperará y se enviará al navegador del usuario que realiza la solicitud, si ha expirado, se eliminará y actualizará antes de enviarse al navegador

Habilitar el almacenamiento en caché

Para habilitar el almacenamiento en caché, coloque la siguiente etiqueta en cualquiera de sus métodos de controlador:

```
$this->output->cache($n);
```

Donde **$n** es la cantidad de minutos que desea que la página permanezca en la memoria caché entre las actualizaciones

La etiqueta anterior puede ir en cualquier lugar dentro de un método, no se ve afectado por el orden en que aparece, así que colóquelo donde le parezca más lógico, una vez que la etiqueta esté en su lugar, sus páginas comenzarán a almacenarse en caché

Importante

- Debido a la forma en que CodeIgniter almacena el contenido para la salida, el almacenamiento en caché solo funcionará si está generando una pantalla para su controlador con una vista
- Si cambia las opciones de configuración que pueden afectar su resultado, debe eliminar manualmente sus archivos de caché

Nota

Antes de que se puedan escribir los archivos de caché, debe establecer los permisos de archivo en su directorio **application/cache/** de manera que sea escribible

Eliminar cachés

Si ya no desea almacenar en caché un archivo, puede eliminar la etiqueta de almacenamiento en caché y ya no se actualizará cuando caduque

Nota
La eliminación de la etiqueta no eliminará la caché de inmediato, tendrá que caducar normalmente

Si necesita eliminar manualmente la memoria caché, puede usar el método **delete_cache()**:

```
// Borra el caché de las URIs actuales
$this->output->delete_cache();

// Borra el caché en /foo/bar
$this->output->delete_cache('/foo/bar');
```

Perfilar la aplicación

La clase **Profiler** mostrará resultados de referencia, consultas que haya ejecutado y datos de **$_POST** al final de sus páginas, esta información puede ser útil durante el desarrollo para ayudar con la depuración y la optimización

Iniciar la clase

Importante

Esta clase **NO necesita inicializarse**, la biblioteca **Output Library** la carga automáticamente si el perfilado está habilitado como se muestra a continuación

Activar el perfilador

Para habilitar el generador de perfiles, coloque la siguiente línea en cualquier lugar dentro de sus métodos de Controlador:

```
$this->output->enable_profiler(TRUE);
```

Al habilitarse, se generará un informe y que se insertará en la parte final de sus páginas

Para desactivar el generador de perfiles, usará:

```
$this->output->enable_profiler(FALSE);
```

Establecer puntos de referencia

Para que el Analizador compile y muestre sus datos de referencia, debe nombrar sus puntos usando una sintaxis específica

Lea la información sobre cómo establecer los puntos de referencia en la sección **Benchmark Class**

Habilitar y deshabilitar las secciones del perfilador

Cada sección de los datos del perfilador puede habilitarse o deshabilitarse configurando la variable de configuración correspondiente en TRUE o FALSE

Esto se puede hacer de dos maneras, primero, puede establecer los valores predeterminados de la aplicación con el archivo de configuración **application/config/profiler.php**

Ejemplo

```
$config['config'] = FALSE;

$config['queries'] = FALSE;
```

En sus controladores, puede anular los valores predeterminados y los valores de archivo de configuración llamando al método **set_profiler_sections()** de la **Output Library**:

```
$secciones = array(
   'config'  => TRUE,
   'queries' => TRUE
);

$this->output->set_profiler_sections($secciones);
```

Las secciones disponibles y la clave de array utilizada para acceder a ellas se describen en la siguiente tabla:

Clave	Descripción	Por defecto
benchmarks	Tiempo transcurrido en los puntos Benchmark y tiempo total de ejecución	TRUE
config	Variables de configuración de CodeIgniter	TRUE
controller_info	La clase de controlador y el método solicitado	TRUE
get	Cualquier dato GET pasado en la solicitud	TRUE
http_headers	Los encabezados HTTP para la solicitud actual	TRUE
memory_usage	Cantidad de memoria consumida por la solicitud actual, en bytes	TRUE
post	Cualquier dato POST pasado en la solicitud	TRUE
queries	Listado de todas las consultas de Bases de datos ejecutadas, incluido el tiempo de ejecución	TRUE
uri_string	El URI de la solicitud actual	TRUE
session_data	Datos almacenados en la sesión actual	TRUE
query_toggle_count	Número de consultas después de las cuales el bloque de consulta se ocultará de forma predeterminada.	25

Nota

Al deshabilitar la configuración de **save_queries** en la configuración de su Base de datos también se deshabilitará el perfilado de las consultas de Base de datos y se dejará inútil la configuración de 'consultas', opcionalmente puede anular esta configuración con:

```
$this->db->save_queries = TRUE;
```

Sin esta configuración, no podrá ver las consultas o **last_query<database/helpers>**

CLI - CodeIgniter desde la línea de comandos

Además de llamar controladores de aplicaciones a través de la URL en un navegador, también se pueden cargar a través de la interfaz de línea de comandos (**CLI**)

Que es la CLI?

La interfaz de línea de comandos es un método basado en texto para interactuar con las computadoras, para obtener más información, consulte el artículo de Wikipedia

Por qué ejecutar a través de la línea de comandos?

- Hay muchas razones para ejecutar CodeIgniter desde la línea de comandos, pero no siempre son obvias
- Ejecute sus **cron-jobs** sin necesidad de usar **wget** o **curl**
- Haga que sus **cron-jobs** inaccesibles se carguen en la URL comprobando el valor de retorno de **is_cli()**
- Realiza "tareas" interactivas que pueden hacer cosas como establecer permisos, podar carpetas de caché, ejecutar copias de seguridad, etc.
- Integrar con otras aplicaciones en otros idiomas, por ejemplo, un script aleatorio de C ++ podría invocar un comando y ejecutar código en sus modelos

Probémoslo: ¡Hola mundo!

Vamos a crear un controlador simple para que pueda verlo en acción, usando su editor de texto, cree un archivo llámelo **Test.php** y coloque el siguiente código en él:

```
<?php
class Test extends CI_Controller {

  public function mensaje($to = 'Mundo')
  {
     echo "Hola {$to}!".PHP_EOL;
  }
}
```

Luego guarde el archivo en su directorio **application/controllers/**
Ahora normalmente visitarías el sitio usando una URL similar a esta:

```
example.com/index.php/test/mensaje/to
```

En cambio, vamos a abrir la terminal en **Mac/Linux** o ir a **Ejecutar> "cmd"** en **Windows** y navegar a nuestro proyecto CodeIgniter:

```
$cd /path/to/project;
$php index.php test mensaje
```

Si lo hizo bien, debería ver **Hola Mundo!** en pantalla

```
$php index.php test mensaje "Mortadelo y Filemón"
```

Aquí estamos pasando un argumento de la misma manera que los parámetros de URL. **"Mortadelo y Filemón"** se pasa como argumento y la salida es:

```
Hola Mortadelo y Filemón!
```

¡Eso es!

Eso, en pocas palabras, es todo lo que hay que saber sobre los controladores en la línea de comando, recuerde que esto es solo un controlador normal, por lo que el enrutamiento y **_remap()** funciona bien

Administrar sus aplicaciones

De forma predeterminada, se supone que solo tiene la intención de utilizar CodeIgniter para administrar una aplicación, que creará en su directorio **application/**, sin embargo, es posible tener múltiples conjuntos de aplicaciones que comparten una sola instalación de CodeIgniter, o incluso cambiar el nombre o reubicar el directorio de su aplicación

Cambiar el nombre del directorio de la aplicación

Si desea cambiar el nombre del directorio de su aplicación, puede hacerlo, abra su archivo **index.php** principal y configure su nombre usando la variable **$application_folder**:

```
$application_folder = 'application';
```

Reubicación de su directorio de aplicaciones

Es posible mover su directorio de aplicaciones a una ubicación diferente en su servidor que su raíz web

Para hacerlo, abra su **index.php** principal y establezca una ruta completa del servidor en la variable **$application_folder**:

```
$application_folder = '/path/a/su/application';
```

Ejecución de múltiples aplicaciones con una instalación CodeIgniter

Si desea compartir una instalación común de CodeIgniter para administrar varias aplicaciones diferentes, simplemente coloque todos los directorios ubicados dentro de su directorio de aplicaciones en sus propios subdirectorios

Por ejemplo, supongamos que desea crear dos aplicaciones, llamadas **"foo"** y **"bar"**

Puede estructurar los directorios de sus aplicaciones de esta manera:

```
applications/foo/
applications/foo/config/
applications/foo/controllers/
applications/foo/libraries/
applications/foo/models/
applications/foo/views/
applications/bar/
applications/bar/config/
applications/bar/controllers/
applications/bar/libraries/
applications/bar/models/
applications/bar/views/
```

Para seleccionar una aplicación en particular para su uso, debe abrir su archivo **index.php** principal y establecer la variable **$application_folder**

Por ejemplo, para seleccionar la aplicación **"foo"** para su uso, debería hacer esto:

```
$application_folder = 'applications/foo';
```

Nota

Cada una de sus aplicaciones necesitará su propio archivo **index.php** que llama a la aplicación deseada, el archivo **index.php** puede nombrarse como quiera

Manejar múltiples entornos

Los desarrolladores a menudo desean un comportamiento diferente del sistema según si una aplicación se está ejecutando en un entorno de desarrollo o producción, por ejemplo, la salida de error detallado es algo que sería útil al desarrollar una aplicación, pero también puede presentar un problema de seguridad cuando está "activo"

La constante ENVIRONMENT

Por defecto, CodeIgniter viene con la constante **ENVIRONMENT** configurada para usar el valor proporcionado en **$_SERVER ['CI_ENV']**, de lo contrario se predetermina a ' **development**'

En la parte superior de **index.php**, verá:

```
define('ENVIRONMENT', isset($_SERVER['CI_ENV']) ? $_SERVER['CI_ENV'] : 'development');
```

Esta variable del servidor se puede establecer en su archivo **.htaccess**, o en la configuración de Apache usando **SetEnv**, hay métodos alternativos disponibles para **nginx** y otros servidores, o puede eliminar esta lógica por completo y establecer la constante en función de la dirección IP del servidor

Además de afectar algunos comportamientos básicos del framework, consulte la siguiente sección, puede usar esta constante en su propio desarrollo para diferenciar entre el entorno en el que se está ejecutando

Efectos sobre el comportamiento del Framewok predeterminado

Hay algunos lugares en el sistema CodeIgniter donde se utiliza la constante **ENVIRONMENT**, esta sección describe cómo se ve afectado el comportamiento predeterminado del framework

Reportes de Error

Establecer la constante **ENVIRONMENT** a un valor de **'development'** hará que todos los errores de PHP cuan do ocurran sean renderizados al navegador, por el contrario, establecer la constante en **'production'** deshabilitará todos los resultados de error

Deshabilitar los informes de error en la producción es una buena práctica de seguridad

Archivos de configuración

Opcionalmente, puede hacer que CodeIgniter cargue archivos de configuración específicos del entorno, puede ser útil para administrar cosas como diferentes claves de API en múltiples entornos, se describe con más detalle en la documentación de la biblioteca **Config Class**

Sintaxis alternativa de PHP para archivos de Vistas

Si no utiliza el motor de plantillas de CodeIgniter, utilizará PHP puro en sus archivos de vistas, para minimizar el código PHP en estos archivos, y para facilitar la identificación de los bloques de código, se recomienda utilizar la sintaxis alternativa de PHP para las estructuras de control y las sentencias cortas de '**echo**', si no está familiarizado con esta sintaxis, ésta le permite eliminar las llaves de su código y eliminar las declaraciones **"echo"**

Soporte automático de Short Tags

Nota

Si encuentra que la sintaxis descrita en esta página no funciona en su servidor, es posible que las **"short tags"** estén deshabilitadas en su archivo PHP ini, CodeIgniter opcionalmente reescribirá las etiquetas cortas sobre la marcha, lo que le permite usar esa sintaxis incluso si su servidor no la admite, esta característica se puede habilitar en su archivo **config/config.php**

Tenga en cuenta que si utiliza esta función, si se encuentran errores de PHP en sus archivos de vista, el mensaje de error y el número de línea no se mostrarán con precisión, en cambio, todos los errores se mostrarán como errores **eval()**

"echo" alternativo

Normalmente para hacer '**echo**' o imprimir una variable, haría esto:

```
<?php echo $variable; ?>
```

Con la sintaxis alternativa, puede hacerlo de esta manera:

```
<?=$variable?>
```

Estructuras de control alternativas

Las estructuras de controles, como **if**, **for**, **foreach** y **while** también se pueden escribir en un formato simplificado

Aquí hay un ejemplo usando **foreach**:

```
<ul>
  <?php foreach ($todo as $item): ?>
    <li><?=$item?></li>
  <?php endforeach; ?>
</ul>
```

Tenga en cuenta que no hay llaves, en cambio, la llave final se reemplaza por **endforeach**, cada una de las estructuras de control enumeradas anteriormente tiene una sintaxis de cierre similar: **endif**, **endfor**, **endforeach** y **endwhile**

También observe que en lugar de usar un punto y coma **(;)** después de cada estructura (excepto la última), hay dos puntos **(:)**

Esto es importante!

Aquí hay otro ejemplo, usando **if / elseif / else**, observe los dos puntos:

```
<?php if ($username === 'quevedo'): ?>

  <h3>Hola Quevedo</h3>

<?php elseif ($username === 'cervantes'): ?>

  <h3>Hola Cervantes</h3>

<?php else: ?>

  <h3>Hola usuario deconocido</h3>

<?php endif; ?>
```

Seguridad

Esta página describe algunas de las "mejores prácticas" relacionadas con la seguridad web y detalla las características de seguridad interna de CodeIgniter

Seguridad de URI

CodeIgniter es bastante restrictivo con respecto a los caracteres que permite en las strings de URI para ayudar a minimizar la posibilidad de que se pasen datos maliciosos a la aplicación

Los URI solo pueden contener lo siguiente:

- Texto alfanumérico (solo caracteres latinos)
- Virgulilla: ~
- Signo de porcentaje: %
- Punto:.
- Dos puntos: :
- Guion bajo: _
- Guion: -
- Espacio

register_globals

Durante la inicialización del sistema, se destruyen todas las variables globales que se encuentran en **$_GET**, **$_POST**, **$_REQUEST** y **$_COOKIE**

La rutina de destrucción es la misma que **register_globals = off**

display_errors

En entornos de producción, generalmente es deseable **"deshabilitar"** el informe de errores de PHP al establecer el indicador interno **display_errors** en un valor de **0**, esto desactiva los errores nativos de PHP para que no se presenten como salida, lo que podría contener información confidencial

Establecer la constante **ENVIRONMENT** de CodeIgniter en **index.php** a un valor de **'production'** desactivará estos errores, en el modo de desarrollo, se recomienda usar un valor de **'development'**, se puede encontrar más información sobre la diferenciación entre entornos en **Manejo de múltiples entornos**

magic_quotes_runtime

La directiva **magic_quotes_runtime** se desactiva durante la inicialización del sistema, para que no tenga que quitar barras al recuperar datos de su Base de datos

Buenas prácticas

Antes de aceptar datos en su aplicación, ya sean datos **POST** de un envío de formulario, datos **COOKIE**, datos **URI**, datos **XML-RPC** o incluso datos del array **SERVER**, le recomendamos que siga este enfoque de tres pasos:

1. Valide los datos para asegurarse de que se ajustan correctamente al tipo, longitud, tamaño, etc.
2. Filtrar los datos como si estuvieran contaminados
3. Escapar los datos antes de enviarlos a su Base de datos o enviarlos a un navegador

CodeIgniter proporciona las siguientes funciones y sugerencias para ayudarlo en este proceso:

XSS Filtering

CodeIgniter viene con un filtro **Cross Site Scripting**,este filtro busca técnicas comúnmente utilizadas para incrustar JavaScript malicioso en sus datos, u otros tipos de código que intentan apropiarse de las cookies u otras cosas maliciosas, el filtro XSS se describe en la **Security Library**

Nota

El filtrado XSS solo debe realizarse en la salida, el filtrado de los datos de entrada puede modificar los datos de formas no deseadas, incluida la eliminación de caracteres especiales de las contraseñas, lo que reduce la seguridad en lugar de mejorarla

CSRF protection

CSRF significa Falsificación de solicitudes entre sitios, que es el proceso de un atacante engañando a su víctima para que envíe una solicitud sin saberlo

CodeIgniter proporciona una protección CSRF lista para usar, que se activará automáticamente para cada solicitud HTTP no GET, pero también necesita que cree sus formularios de envío de cierta manera, esto se explica en la documentación de la **Security Library**

Contraseñas - Passwords

Es fundamental que maneje las contraseñas en su aplicación correctamente

Desafortunadamente, muchos desarrolladores no saben cómo hacerlo, y la web está llena de consejos obsoletos o incorrectos, lo que no ayuda, nos gustaría darle una lista de lo que se debe y lo que no se debe hacer para ayudarle con esto

Por favor lea lo siguiente:

- **NO** almacene contraseñas en formato de texto plano, aplique **siempre hash** a sus contraseñas

- **NO** use Base64 o una codificación similar para almacenar contraseñas, esto es casi como almacenarlos en texto plano, aplique **siempre hash**, no codifique

  La codificación y el cifrado son procesos bidireccionales, las contraseñas son secretos que solo deben ser conocidos por el propietario y, por lo tanto, deben funcionar solo en una dirección, Hashing hace eso, es muy difícil averiguar la contraseña desde un hashing, pero hay decodificación y descifrado

- **NO** use algoritmos de hashing débiles o rotos como MD5 o SHA1, estos algoritmos son antiguos, comprobados como defectuosos, y no están diseñados para el uso de contraseñas

- **NO** invente sus propios algoritmos
  Solo use algoritmos de **hashing de contraseñas** fuertes como **BCrypt**, que se usa en las funciones propias de Hashing de contraseñas de PHP, úselos, incluso si no está ejecutando PHP 5.5+, CodeIgniter los proporciona por usted

- **NO** muestre ni envíe nunca una contraseña en formato de texto plano!

  Incluso para el propietario de la contraseña, si necesita una característica de "Olvidó su contraseña", solo genere aleatoriamente una contraseña nueva, única, esto también es importante, y envíela en su lugar

- **NO** ponga límites innecesarios en las contraseñas de sus usuarios

  Si usa un algoritmo hash distinto de **BCrypt** (que tiene un límite de 72 caracteres), debe establecer un límite relativamente alto en la longitud de las contraseñas para mitigar los ataques DoS, por ejemplo, 1024 caracteres

  Sin embargo, aparte de eso, no tiene sentido forzar una regla de que una contraseña solo puede contener un número de caracteres o que no puede contener un cierto conjunto de caracteres especiales, esto no solo reduce la seguridad en lugar de mejorarla, sino que literalmente no hay razón para hacerlo, no se aplican limitaciones técnicas ni limitaciones de almacenamiento (prácticas) una vez que las ha procesado, ¡ninguna!

Validar los datos de entrada

CodeIgniter dispone de la **Form Validation Library** lo ayuda a validar, filtrar y preparar sus datos

Incluso si eso no funciona para su caso de uso, asegúrese de validar y desinfectar siempre todos los datos de entrada, por ejemplo, si espera un string numérico para una variable de entrada, puede verificarlo con **is_numeric()** o **ctype_digit()**, siempre trate de reducir su comprobaciones a un cierto patrón

Tenga en cuenta que esto incluye no solo las variables **$_POST** y **$ _GET**, sino también las cookies, la string de agente de usuario y básicamente todos los datos que no son creados directamente por su propio código

Escape todos los datos antes del insertarlos en la Base de datos

Nunca inserte información en su Base de datos sin primero escaparlos, para más información consulte la sección **database queries** que trata las consultas de la Base de datos

Ocultar sus archivos

Otra buena práctica de seguridad es dejar su **index.php** y sus "activos" , por ejemplo, .js, css y archivos de imagen, en el directorio raíz de su servidor (comúnmente llamado **"htdocs/"**), estos son los únicos archivos que se necesitan para acceder desde la web, no debe permitir que los visitantes vean algo más y puedan acceder a datos confidenciales, ejecutar scripts, etc.

Si no puede hacerlo, puede intentar usar un archivo **.htaccess** para restringir el acceso a esos recursos

CodeIgniter tendrá un archivo **index.html** en todos sus directorios en un intento de ocultar algunos de estos datos, pero tenga en cuenta que esto no es suficiente para evitar un ataque serio

Guía de estilo de PHP

Seguidamente se describen los **estilos** de codificación a seguir cuando se contribuye al desarrollo de CodeIgniter

No es necesario utilizar estos estilos en su propia aplicación CodeIgniter, aunque sí se recomiendan

Formato de archivo

Los archivos deberían guardarse con codificación Unicode **(UTF-8)**, **no** debe usarse el **BOM**, a diferencia de UTF-16 y UTF-32, no hay un bit de orden para indicar en un archivo codificado con UTF-8, y el BOM puede tener efectos colaterales negativos en PHP en el envío de la salida, evitando que la aplicación sea capaz de establecer sus encabezados

Se deben usar las terminaciones de línea de Unix **(LF)**

Aquí se explica cómo aplicar estas configuraciones en algunos de los editores de texto más comunes, las instrucciones para su editor de texto pueden variar; revise la documentación de su editor

Notepad++

1. Menú Codificación, seleccione "Codificar en UTF-8 sin BOM"
2. Menú Configuración, abra "Preferencias.."
3. Selecione "Archivo nuevo"
4. En "Formato" marque "UNIX"
5. En "Codificación" marque "UTF-8 sin BOM" y "Aplicar a los archivos ANSI abiertos"

TextMate

1. Abra las Preferencias de la Aplicación
2. Hacer clic en Avanzado, y luego en la pestaña "Guardar"
3. En "Codificación de Archivo", seleccione "UTF-8 (recomendado)"
4. En "Terminación de Línea", seleccione "LF (recomendado)"
5. Opcional : Marque "También usar para los archivos existentes" si desea modificar las terminaciones de línea de archivos que abre para las nuevas preferencias.

BBEdit

1. Abra las Preferencias de la Aplicación
2. Seleccione "Codificaciones del Texto" en la izquierda.
3. En "Codificación del texto para nuevos documentos", seleccione "Unicode (UTF-8, sin BOM)"
4. Opcional: En "Si no se puede determinar la codificación del archivo, usar", seleccione "Unicode (UTF-8, sin BOM)"
5. Selecciones "Archivos de Texto" en la izquierda.
6. En "Saltos de Línea por Defecto", seleccione "Mac OS X y Unix (LF)"

Etiqueta de cierre de PHP

La etiqueta de cierre de PHP **?>** es opcional para el analizador de PHP, sin embargo, si se usa, cualquier espacio en blanco que siga a la etiqueta de cierre, ya sea que lo haya introducido el desarrollador, el usuario o una aplicación de FTP, puede generar resultados no deseados, errores de PHP o, en caso de que se supriman, páginas en blanco, por esta razón, todos los archivos PHP **DEBEN OMITIR** la etiqueta de cierre de PHP y terminar con una única línea vacía en su lugar

Nombres de archivos

Los archivos de clase deben nombrarse de la manera **Ucfirst-like**, mientras que cualquier otro nombre de archivo configuraciones, vistas, scripts genéricos, etc., debe estar en minúsculas:

```
INCORRECTO

algunalibrería.php
algunaLibrería.php
ALGUNALIBRERíA.php
Alguna_Librería.php

Aplicacion_config.php
Aplicacion_Config.php
aplicacionConfig.php

CORRECTO

Algunalibrería.php
Alguna_librería.php

aplicacionconfig.php
aplicacion_config.php
```

Además, los nombres de los archivos de clase deben coincidir con el nombre de la clase, por ejemplo, si tiene una clase llamada **Miclase**, el nombre de archivo debe ser: **Miclase.php**

Denominación de clase y método

Los nombres de clase siempre deben comenzar con una letra mayúscula, las palabras múltiples deben separarse con un guion bajo y no CamelCase:

```
INCORRECTO

class superclass
class SuperClass
CORRECTO

class Super_class {

  public function __construct()
  {

  }
}
```

Los métodos de clase deben estar completamente en minúscula y su nombre deber indicar claramente su función, incluyendo preferiblemente un verbo, trate de evitar los nombres excesivamente lagos y prolijos, las palabras múltiples se deben separar con un guión bajo:

```
INCORRECTO

function fileproperties()     // no es descriptivo y necesita un guión de subrayado de separación
function fileProperties()     // no es descriptivo y usa CamelCase
function getfileproperties()  // Mejor! Pero todavía le falta el guión de subrayado de separación
function getFileProperties()  // usa CamelCase
function get_the_file_properties_from_the_file()   // demasiada palabrería

CORRECTO

function get_file_properties()  // descriptivo, guión de subrayado de separación y todas la letras son
minúsculas
```

Nombres de variables

Las pautas para el nombramiento de variables son muy similares a las utilizadas para los métodos de clase, las variables deben contener solo letras minúsculas, usar separadores de subrayado y ser razonablemente nombradas para indicar su propósito y contenido, las variables cortas, sin palabras, solo se deben usar como iteradores en los bucles **for()**:

```
INCORRECTO

$j = 'foo';     // las variables de una sola letra se deberían usar solamente en bucles for()
$Str            // contiene letras mayúsculas
$buferDeTexto   // usa CamelCase y podría acortarse sin perder sentido semántico
$idgrupo        // varias palabras, necesita un separador de guión de subrayado
$nombre_de_la_ultima_ciudad   // demasiado largo

CORRECTO

for ($j = 0; $j < 10; $j++)
$str
$buffer
$id_grupo
$ultima_ciudad
```

Comentarios

En general, el código debe ser comentado prolíficamente, no solo ayuda a describir el flujo y la intención del código para los programadores menos experimentados, sino que puede resultar muy valiosa cuando regresa a su propio código meses después, no hay un formato requerido para comentarios, pero se recomiendan los siguientes

Los comentarios del estilo **DocBlock** preceden a la clase, el método y las declaraciones de propiedades para que puedan ser recogidos por los IDEs:

```
/**
 * Super Clase
 *
 * @package     Nombre del paquete
 * @subpackage  Subpaquete
 * @category    Categoría
 * @author      Nombre del autor
 * @link        http://miweb.com
 */
class Super_clase {


/**
 * Codifica una string para usarla en XML
 *
 * @param    string  $str  Cadena de entrada
 * @return   string
 */
function xml_encode($str)


/**
 * Datos para modificar la clase
 *
 * @var array
 */
public $datos = array();
```

Use comentarios de una sola línea dentro del código, dejando una línea en blanco entre los bloques de comentarios grandes y el código:

```
$parts = explode("\n", $str);  // dividir la string por nuevas líneas

// Un comentario más largo que necesita dar más detalles sobre lo que ocurre y por qué puede usar múltiples
// comentarios de una sola línea. Intente mantener el ancho razonable, alrededor de 70 caracteres es el más
// fácil de leer. No dude en conectarse a recursos externos permanentes que puede proporcionar mayor
detalle:
// http://miweb.com/information_acerca_de_algo/en_particular/

$parts = $this->foo($parts);
```

Constantes

Las constantes siguen las mismas pautas que las variables, excepto que las constantes siempre deben estar en mayúsculas, utilice siempre constantes de CodeIgniter cuando corresponda, es decir, **SLASH**, **LD**, **RD**, **PATH_CACHE**, etc.:

```
INCORRECTO

myConstante  // falta el guión de subrayado y no está completamente en mayúsculas
N            // no hay constantes de una sola letra
S_C_VER      // no es descriptivo
$str = str_replace('{foo}', 'bar', $str); // debería usar las constantes LD y RD

CORRECTO

MY_CONSTANTE
NEWLINE
SUPER_CLASS_VERSION
$str = str_replace(LD.'foo'.RD, 'bar', $str);
```

TRUE, FALSE y NULL

Las palabras clave TRUE, FALSE, y NULL siempre deben estar en mayúsculas:

```
INCORRECTO

if ( $foo == true )
$bar = false;
function foo( $bar = null )
CORRECTO

if ( $foo == TRUE )
$bar = FALSE;
function foo( $bar = NULL )
```

Operadores Lógicos

Se desaconseja el uso de **||** "O" el operador de comparación, ya que en pantallas de baja resolución puede confundirse con el número **11**, por ejemplo, es preferible utilizar **&&** en lugar de **AND** pero ambos son aceptables, y un espacio siempre debe precederlo y seguirlo:

```
INCORRECTO

if ( $foo || $bar )   // confusión con 11
if ( $foo AND $bar )  // ok, pero no se recomienda para las aplicaciones comunes
if ( !$foo )           // falta espacio separador
if (! is_array($foo) ) // falta espacio separador
CORRECTO

if ( $foo OR $bar )
if ( $foo && $bar )   // recomendado
if ( ! $foo )
if ( ! is_array($foo) )
```

Comparar valores de retorno y Typecasting

Algunas funciones de PHP devuelven FALSE en caso de error, pero también pueden tener un valor de retorno válido de **""** o **0**, que se evaluaría como FALSE en comparaciones sueltas, sea explícito al comparar el tipo de variable al usar estos valores de retorno en condicionales para garantizar que el valor de retorno sea el que espera, y no un valor que tenga una evaluación de tipo suelto equivalente

Use la misma severidad al devolver y verificar sus propias variables, use === y !== según sea necesario:

```
INCORRECTO

// Si 'foo' está al principio de la string, strpos devolverá un 0,
// dando como resultado esta evaluación condicional como TRUE
if (strpos($str, 'foo') == FALSE)

CORRECTO

if (strpos($str, 'foo') === FALSE)

INCORRECTO

function build_string($str = "")
{
  if ($str == "") // oh-oh! ¿Qué ocurre si se pasa como argumento FALSE o el entero 0 ?
  {

  }
}

CORRECTO

function build_string($str = "")
{
  if ($str === "")
  {

  }
}
```

Vea acerca del **typecasting**, lo que puede ser muy útil, el **typecasting** tiene un efecto ligeramente distinto que puede ser deseable, al convertir una variable a un string, por ejemplo, las variables NULL y FALSE se convierten en strings vacías, 0, y otros números, se convierten en strings de dígitos, y el booleano **TRUE** se convierte en **"1"**:

```
$str = (string)$str;  // trata a $str como un string
```

Código de depuración

No deje el código de depuración en su código, incluso cuando esté comentado, cosas como **var_dump()**, **print_r()**, **die()/exit()** no deben incluirse en su código a menos que sirva para un propósito específico que no sea la depuración

Espacio en blanco en archivos

Ningún espacio en blanco puede preceder a la etiqueta PHP de apertura o seguir la etiqueta PHP de cierre, la salida se almacena en el búfer, por lo que el espacio en blanco en sus archivos puede provocar que la salida comience antes de que CodeIgniter genere su contenido, lo que genera errores y la incapacidad de CodeIgniter para enviar encabezados adecuados

Compatibilidad

CodeIgniter recomienda utilizar PHP 5.6 o posterior, pero debe ser compatible con PHP 5.3.7, su código debe contar con este requisito

Además, no use funciones de PHP que requieren la instalación de bibliotecas no predeterminadas a menos que su código contenga un método alternativo cuando la función no esté disponible

Un archivo por clase

Use archivos separados para cada clase, a menos que las clases estén estrechamente relacionadas, un ejemplo de un archivo CodeIgniter que contiene múltiples clases es el archivo de la **Xmlrpc Class**

Espacio en blanco

Use tabuladores para espacios en blanco en su código, no espacios, esto puede parecer algo pequeño, pero el uso de tabuladores en lugar de espacios en blanco permite al desarrollador que busca en su código tener una sangría en los niveles que prefiera y personalizar cualquier aplicación que use. Como beneficio adicional, resultan archivos, un poco, más compactos, que almacenan un carácter de tabulación frente a, por ejemplo, cuatro caracteres de espacio

Saltos de línea

Los archivos se deben guardar con saltos de línea Unix, esto es más un problema para los desarrolladores que trabajan en Windows, pero en cualquier caso asegúrese de que su editor de texto esté configurado para guardar archivos con saltos de línea Unix **(LF)**

Indentación de código

Use el estilo de indentación de **Allman**, con la excepción de las declaraciones de clase, las llaves se colocan siempre en una línea con ellas mismas y se indentan al mismo nivel que la instrucción de control que las "posee":

INCORRECTO

```
function foo ($bar) {
        // ...
}

foreach ($arr as $key => $val) {
        // ...
}
if ($foo == $bar) {
        // ...
} else {
        // ...
}

for ($i = 0; $i < 10; $i++)
        {
        for ($j = 0; $j < 10; $j++)
                {
                // ...
                }
        }

try {
        // ...
}
catch() {
        // ...
}
```

CORRECTO

```
function foo($bar)
{
 // ...
}

foreach ($arr as $key => $val)
{
 // ...
}

if ($foo == $bar)
{
 // ...
}
else
```

```
{
 // ...
}

for ($i = 0; $i < 10; $i++)
{
 for ($j = 0; $j < 10; $j++)
 {
    // ...
 }
}

try
{
 // ...
}
catch()
{
 // ...
}
```

Espaciado entre paréntesis

En general, los paréntesis y corchetes no deben usar espacios adicionales, la excepción es que un espacio siempre debe seguir las estructuras de control de PHP que aceptan argumentos con paréntesis (**declare**,**do-while**, **elseif**, **for**, **foreach**, **if**, **switch**, **while**), para ayudar a distinguirlos de las funciones y aumentar la legibilidad:

```
INCORRECTO

$arr[ $foo ] = 'foo';

CORRECTO:

$arr[$foo] = 'foo'; // no hay espacios alrededor de la clave del array

INCORRECTO

function foo ( $bar )
{

}
```

```
CORRECTO

// no hay espacios alrededor de los paréntesis en declaraciones de funciones
function foo($bar)
{

}
+
INCORRECTO

foreach( $query->result() as $row )
foreach ($ query-> result () as $ row)

CORRECTO

// un solo espacio siguiendo la estructura de control de PHP, pero no en
// paréntesis interiores
foreach ($query->result() as $row)
```

Texto localizado

Las bibliotecas de CodeIgniter deben aprovechar los archivos de idioma correspondientes siempre que sea posible:

```
INCORRECTO

return "Selección inválida";

CORRECTO

return $this->lang->line('seleccion_invalida');
```

Métodos privados y variables

Los métodos y variables a los que solo se accede internamente, como las funciones de utilidad y auxiliares que usan los métodos públicos para la abstracción de código, deben ir precedidos de un guión bajo:

```
public function convert_text()
private function _convert_text()
```

Errores de PHP

El código debe funcionar sin errores y no depender que las advertencias y avisos estén ocultos para cumplir con este requisito, por ejemplo, nunca acceda a una variable que no haya configurado usted mismo, tal como claves del array **$_POST,** sin verificar si están establecidas con **isset()**

Asegúrese de que su entorno de desarrollo tenga habilitados los informes de errores para **TODOS** los usuarios, y que **display_errors** esté habilitado en el entorno de PHP

Puede verificar esta configuración con:

```
if (ini_get('display_errors') == 1)
{
   exit "Enabled";
}
```

En algunos servidores donde **display_errors** está deshabilitado y no tiene la capacidad de cambiar esto en **php.ini**, a menudo puede habilitarlo con:

```
ini_set('display_errors', 1);
```

> ***Nota***
>
> Establecer la configuración **display_errors** con **ini_set()** en tiempo de ejecución no es idéntico a tenerlo habilitado en el entorno PHP, a saber, no tendrá ningún efecto si el script tiene errores fatales

Etiquetas cortas abiertas

Utilice siempre etiquetas de apertura PHP completas, en caso que el servidor no tenga habilitada **short_open_tag**:

```
INCORRECTO

<? echo $foo; ?>

<?=$foo?>

CORRECTO

<?php echo $foo; ?>
```

> ***Nota***
>
> PHP 5.4 siempre tendrá la etiqueta **<? =** disponible

Una sentencia por línea

Nunca combine sentencias en una línea :

```
INCORRECTO

$foo = 'esto'; $bar = 'eso'; $bat = str_replace ($foo, $bar, $saco);
```

```
CORRECTO

$foo = 'esto';
$bar = 'eso';
$bat = str_replace($foo, $bar, $saco);
```

Strings

Delimite siempre las strings con comillas simples a menos que necesite variables analizadas, y en los casos en que necesite variables analizadas, use llaves para evitar el análisis avaricioso de tokens, también puede usar strings de comillas dobles si la string contiene comillas simples, por lo que no tendrá que usar caracteres de escape:

```
INCORRECTO

"My String"       // no hay análisis de variables, por lo tanto no use comillas dobles
"My string $foo"  // se necesitan llaves
'SELECT foo FROM bar WHERE baz = \'bag\'' // feo feo

CORRECTO

'My String'
"My string {$foo}"
"SELECT foo FROM bar WHERE baz = 'bag'"
```

Consultas SQL

Las palabras clave SQL se escriben siempre en mayúscula: **SELECT**, **INSERT**, **UPDATE**, **WHERE**, **AS**, **JOIN**, **ON**, **IN**, etc.

Divida las consultas largas en varias líneas para mayor legibilidad, preferiblemente para cada cláusula:

```
INCORRECTO

// las palabras clave están en minúsculas y las consultas son demasiado largas
// para una línea simple (... indica continuación de línea)

$query = $this->db->query("select foo, bar, baz, foofoo, foobar as raboof, foobaz from
exp_pre_email_addresses
...where foo != 'oof' and baz != 'zab' order by foobaz limit 5, 100");

CORRECTO

$query = $this->db->query("SELECT foo, bar, baz, foofoo, foobar AS raboof, foobaz
                        FROM exp_pre_email_addresses
                        WHERE foo != 'oof'
                        AND baz != 'zab'
```

```
                        ORDER BY foobaz
                        LIMIT 5, 100");
```

Argumentos de función predeterminados

Cuando corresponda, proporcione los valores predeterminados de los argumentos de la función, lo que ayuda a prevenir los errores de PHP con llamadas erróneas y proporciona valores que pueden evitar algunas líneas de código:

```
function foo($bar = '', $baz = FALSE)
```

Bibliotecas

Benchmarking Class - clase para Análisis comparativo

CodeIgniter tiene una clase de Benchmarking que siempre está activa, lo que permite calcular la diferencia de tiempo entre dos puntos marcados

Nota

Esta clase se inicia automáticamente por el sistema y no es necesario hacerlo manualmente

El punto de referencia siempre se inicia en el momento en que se invoca el framework, y finaliza con la clase **output** justo antes de enviar la vista final al navegador, lo que permite mostrar una medición de tiempo exacta de la ejecución del sistema completo

Usar Benchmark Class

La clase Benchmark se puede usar dentro de sus controladores, vistas o modelos

El proceso de uso es este:

1. Marcar un punto de inicio
2. Marcar un punto de fin
3. Ejecutar la función "**elapsed time**" para ver el resultado

Un ejemplo con código real:

```
$this->benchmark->mark('codigo_inicio');
  // Algún código aquí
$this->benchmark->mark('codigo_fin');

echo $this->benchmark->elapsed_time('codigo_inicio', 'codigo_fin');
```

Nota

Las palabras **"codigo_inicio"** y **"codigo_fin"** son arbitrarias, son simplemente palabras usadas para establecer dos marcadores. puede usar las palabras que desee y puede establecer múltiples conjuntos de marcadores

Considere este ejemplo:

```
$this->benchmark->mark('perro');

  // Algunos códigos pasan aquí

$this->benchmark->mark('gato');

  // Más código por aquí

$this->benchmark->mark('pajaro');

echo $this->benchmark->elapsed_time('perro', 'gato');
echo $this->benchmark->elapsed_time('gato', 'pajaro');
echo $this->benchmark->elapsed_time('perro', 'pajaro');
```

Perfilar sus puntos de evaluación

Si desea que sus datos de evaluación estén disponibles para el **Profiler**, todos sus puntos marcados deben configurarse en pares, y cada nombre de punto de marca debe iniciarse con **_start** y finalizar con **_end**. Cada par de puntos debe ser nombrado de manera idéntica:

```
$this->benchmark->mark('mi_marca_start');

  // Algún código aquí...

$this->benchmark->mark('mi_marca_end');

$this->benchmark->mark('otra_marca_start');

  // Más código aquí...

$this->benchmark->mark('otra_marca_end');
```

Por favor, lea la sección **Perfilando su aplicación** para más información

Visualizar del tiempo total de ejecución

Si desea visualizar el tiempo total transcurrido desde el momento en que CodeIgniter comienza hasta el momento en que se envía el resultado final al navegador, simplemente colóquelo en una de sus plantillas de visualización:

```
<?php echo $this->benchmark->elapsed_time();?>
```

Notará que es la misma función utilizada en los ejemplos anteriores para calcular el tiempo entre dos puntos, excepto que no está usando ningún parámetro, cuando los parámetros están ausentes, CodeIgniter no detiene el punto de

referencia hasta justo antes de que se envíe el resultado final al navegador, no importa dónde use la llamada a la función, el temporizador continuará ejecutándose hasta el final

Una forma alternativa de mostrar su tiempo transcurrido en sus archivos de vista es usar esta pseudovariable, si prefiere no usar el PHP puro:

```
{elapsed_time}
```

Nota

Si desea comparar cualquier elemento dentro de las funciones de su controlador, debe establecer sus propios puntos de **start/end**

Mostrar el consumo de memoria

Si su instalación de PHP está configurada con **–enable-memory-limit**, puede visualizar la cantidad de memoria consumida por todo el sistema usando el siguiente código en uno de sus archivos de vista:

```
<?php echo $this->benchmark->memory_usage();?>
```

Nota

Esta función solo se puede usar en sus archivos de vista, el consumo reflejará la memoria total utilizada por toda la aplicación

Una forma alternativa de mostrar el uso de la memoria en sus archivos de vista es usar esta pseudovariable, si prefiere no usar el PHP puro:

```
{memory_usage}
```

class CI_Benchmark

class CI_Benchmark

class CI_Benchmark

elapsed_time()

string elapsed_time([string $point1 = '' [, string $point2 = '' [, int $decimals = 4]]])

Descripción

Calcula y devuelve la diferencia de tiempo entre dos puntos benchmark, si el primer parámetro está vacío, esta función devuelve la pseudovariable **{elapsed_time}**, esto permite que se muestre el tiempo total de ejecución en una plantilla. La clase **Output Class** intercambiará el valor real de esta variable

Parámetros

point1

Un punto marcado particular

point2

Un punto marcado particular

decimals

Número de decimales para precisión

Valores devueltos

Tiempo transcurrido

mark()

void mark(string $name)

Descripción

Establece un marcador **benchmark**

Parámetros

name

El nombre que desea asignar a su marcador

Valores devueltos

void

memory_usage()

string memory_usage()

Descripción

Devuelve el marcador **{memory_usage}**, esto permite colocarlo en cualquier lugar de una plantilla sin que la memoria se calcule hasta el final, **Output Class** intercambiará el valor real de esta variable

Valores devueltos

String con el valor de **{memory_usage}**

Caching Driver - Driver de Caché

CodeIgniter permite algunas de las formas más populares de almacenamiento en caché rápido y dinámico, excepto el almacenamiento en caché basado en archivos que requieren requisitos específicos del servidor, si no se cumplen los requisitos se produce una excepción fatal

Ejemplo de uso

El siguiente ejemplo carga el controlador de caché, especifica **APC** (Alternative PHP Cache) como el controlador que utilizará y volverá al caché basado en archivos si **APC** no está disponible en el entorno de alojamiento

```
$this->load->driver('cache', array('adapter'=>'apc', 'backup'=>'file'));

if ( ! $foo = $this->cache->get('foo'))
{
   echo 'Guardar en el caché!<br />';
   $foo = 'foobarbaz!';

   // Guardar en la memoria caché durante 5 minutos
   $this->cache->save('foo', $foo, 300);
}

echo $foo;
```

También puede prefijar nombres de elementos de caché a través de la configuración **key_prefix**, es útil para evitar colisiones cuando se ejecutan varias aplicaciones en el mismo entorno:

```
$this->load->driver('cache', array('adapter'=>'apc', 'backup'=>'file', 'key_prefix'=>'my_'));

$this->cache->get('foo'); // Obtendremos la entrada de caché llamada 'my_foo'
```

class CI_Cache

class CI_Cache

class CI_Cache

cache_info()

mixed cache_info()

Descripción

Este método devolverá información de todo el caché

Valores devueltos

Información de toda la Base de datos de caché

Ejemplo

```
var_dump($this->cache->cache_info());
```

> ***Nota***
>
> La información devuelta y la estructura de los datos dependen del adaptador que se está utilizando

clean()

bool clean()

Descripción

Este método '**limpia**' todo el caché, si la eliminación de los archivos de caché falla, el método devolverá FALSE

Valores devueltos

TRUE en caso de éxito, FALSE en caso de error

Ejemplo

```
$this->cache->clean();
```

decrement()

mixed decrement(string $id [, int $offset = 1])

Descripción

Realiza la disminución atómica de un valor almacenado

Parámetros

id
: ID de caché

offset
: Intervalo/valor para reducir

Valores devueltos

Nuevo valor si tiene éxito, FALSE si falla

Ejemplo

```
// 'iterator' tiene un valor de 6

$this->cache->decrement('iterator'); // 'iterator' ahora es 5

$this->cache->decrement('iterator', 2); // 'iterator' ahora es 3
```

delete()

bool delete(string $id)

Descripción

Este método elimina un elemento específico del caché

Parámetros

id

Nombre del elemento en caché

Valores devueltos

TRUE si es eliminado, FALSE si falla

Ejemplo

```
$this->cache->delete('cache_item_id');
```

get()

mixed get(string $id)

Descripción

Este método intenta recuperar un elemento de la memoria caché

Parámetros

id

Nombre del elemento en caché

Valores devueltos

Valor del elemento o FALSE si no se encuentra

Ejemplo

```
$foo = $this->cache->get('my_cache_item');
```

get_metadata()

mixed get_metadata(string $id)

Descripción

Este método devolverá información detallada sobre un elemento específico en la memoria caché

Parámetros

id

Nombre del elemento de caché

Valores devueltos

Metadatos del elemento en caché

Ejemplo

```
var_dump($this->cache->get_metadata('my_cached_item'));
```

> ***Nota***
>
> La información devuelta y la estructura de los datos dependen del adaptador que se está utilizando

increment()

mixed increment(string $id [, int $offset = 1])

Descripción

Realiza el incremento atómico de un valor almacenado

Parámetros

id

ID de caché

offset

Intervalo/valor para agregar

Valores devueltos

Nuevo valor si tiene éxito, FALSE si falla

Ejemplo

```
// 'iterator' tiene un valor de 2

$this->cache->increment('iterator'); // 'iterator' ahora es 3

$this->cache->increment('iterator', 3); // 'iterator' ahora es 6
```

is_supported()

bool is_supported(string $driver)

Descripción

Averiguar si el controlador de caché **$driver** es soportado, este método se llama automáticamente al acceder a los controladores a través de **$this->cache->get()**, sin embargo, si se utiliza los controladores individuales, asegúrese de llamar a este método para asegurarse de que el controlador sea compatible en el entorno de alojamiento

Parámetros

driver

El nombre del controlador de almacenamiento en caché

Valores devueltos

TRUE si es soportado, FALSE si no

Ejemplo

```
if ($this->cache->apc->is_supported())
{
   if ($datos = $this->cache->apc->get('my_cache'))
   {
      // hacer cosas.
   }
}
```

save()

string save(string $id, mixed $data [, int $ttl = 60 [, bool $raw = FALSE]])

Descripción

Este método guardará un elemento en el caché, si el almacenamiento falla, el método devolverá FALSE

Parámetros

id
: Nombre del elemento de caché

data
: Los datos para guardar

ttl
: Time To Live, en segundos (por defecto 60)

raw
: Si se debe almacenar el valor bruto

Valores devueltos

TRUE si tiene éxito, FALSE si falla

Ejemplo

```
$this->cache->save('cache_item_id', 'datos_al_cache');
```

> ***Nota***
>
> El parámetro **$raw** solo es utilizado por **APC** y **Memcache**, para permitir el uso de **increment()** y **decrement()**

Drivers

Caché alternativo de PHP (APC)

Se puede acceder a todos los métodos enumerados anteriormente sin pasar un adaptador específico:

```
$this->load->driver('cache');
$this->cache->apc->save('foo', 'bar', 10);
```

Para obtener más información sobre **APC**, consulte **http://php.net/apc**

Caché basado en archivos

A diferencia del almacenamiento en caché de la **Output Class**, el controlador de caché basado en archivos permite almacenar en caché algunos fragmentos de archivos de vista, úselo con cuidado y asegúrese de medir el tiempo en su aplicación, ya que puede llegar un momento en que la E/S de disco anule las ganancias positivas de almacenar en caché

Se puede acceder a todos los métodos enumerados anteriormente sin pasar un adaptador específico:

```
$this->load->driver('cache');
$this->cache->file->save('foo', 'bar', 10);
```

Caché Memcached

Se pueden especificar múltiples servidores **Memcached** en el archivo de configuración **memcached.php**, ubicado en el directorio **application/config/**

Se puede acceder a todos los métodos enumerados anteriormente sin pasar un adaptador específico de la siguiente manera:

```
$this->load->driver('cache');
$this->cache->memcached->save('foo', 'bar', 10);
```

Para obtener más información sobre **Memcached**, consulte **http://php.net/memcached**

Caché WinCache

En Windows, también puede utilizar el controlador **WinCache**

Se puede acceder a todos los métodos enumerados anteriormente sin pasar un adaptador específico de la siguiente manera:

```
$this->load->driver('cache');
$this->cache->wincache->save('foo', 'bar', 10);
```

Para obtener más información sobre **WinCache**, consulte **http://php.net/wincache**

Caché Redis

Redis es un almacén de valores-clave en memoria que puede operar en el modo de caché LRU, para usarlo, necesita el servidor **Redis** y la extensión **PHP phpredis**: PHP **https://github.com/phpredis/phpredis**

Las opciones de configuración para conectarse al servidor **Redis** deben almacenarse en el archivo **application/config/redis.php**. Las opciones disponibles son:

```
$config['socket_type'] = 'tcp'; //`tcp` o `unix`
$config['socket']      = '/var/run/redis.sock'; // en caso de tipo de socket `unix`
$config['host']        = '127.0.0.1';
$config['password']    = NULL;
$config['port']        = 6379;
$config['timeout']     = 0;
```

Se puede acceder a todos los métodos enumerados anteriormente sin pasar un adaptador específico de la siguiente manera:

```
$this->load->driver('cache');
$this->cache->wincache->save('foo', 'bar', 10);
```

Para obtener más información sobre **Redis**, consulte **http://redis.io**

Dummy Cache - Caché simulada

Este es un backend de almacenamiento en caché que siempre 'se perderá', no almacena datos, pero le permite mantener su código de almacenamiento en caché en su lugar en entornos que no son compatibles con la memoria caché elegida

Calendaring Class - Clase calendario

La clase Calendario le permite crear dinámicamente calendarios, sus calendarios se pueden formatear mediante el uso de una plantilla de calendario, lo que permite un control del 100% de cada aspecto de su diseño, además, puede pasar datos a sus celdas de calendario

Iniciar Calendaring Class

Calendaring Class se inicia usando **$this->load->library**:

```
$this->load->library('calendar');
```

Una vez cargada, el objeto esta disponible usando:

```
$this->calendar
```

Mostrar un calendario

Aquí hay un ejemplo muy simple que muestra cómo puede mostrar un calendario:

```
$this->load->library('calendar');
echo $this->calendar->generate();
```

El código anterior genera un calendario para el mes/año actual según la hora del servidor, para mostrar un calendario para un mes y año específico, pase esa información al método **generate()** del calendario:

```
$this- load->library('calendar');
echo $this->calendar->generate(2018, 6);
```

El código anterior generará un calendario que muestra el mes de junio del año 2018, el primer parámetro especifica el año, el segundo especifica el mes

Pasar datos a sus celdas de calendario

Agregar datos a las celdas de su calendario implica crear un array asociativo en el que las claves corresponden a los días que desea rellenar y el valor del array contiene los datos, el array se pasa al tercer parámetro del método **generate()**:

Ejemplo

```
$this->load->library('calendar');

$datos = array(
  3 => 'http://ejemplo.com/noticias/dia/2018/06/03/',
  7 => 'http://ejemplo.com/noticias/dia/2018/06/07/',
```

```
  13 =>'http://ejemplo.com/noticias/dia/2018/06/16/',
  26 =>'http://ejemplo.com/noticias/dia/2018/06/26/'
);

echo $this->calendar->generate(2018, 6, $datos);
```

Con el ejemplo anterior, los días 3, 7, 16 y 26 se convertirán en enlaces que apuntan a las URL que se proporcionan

Nota

Por defecto, se supone que su array contiene enlaces, en la sección que explica la plantilla del calendario, verá cómo puede personalizar los datos transferidos a sus celdas para que pueda pasar diferentes tipos de información

Configurar las preferencias de visualización

Hay siete preferencias que puede configurar para controlar varios aspectos del calendario, las preferencias se establecen pasando un array en el segundo parámetro del método de carga:

Ejemplo

```
$prefs = array(
  'start_day' => 'monday',
  'month_type' =>'long',
  'day_type' =>  'short'
);

$this->load->library('calendar', $prefs);

echo $this->calendar->generate();
```

El código anterior comenzaría el calendario el lunes, usa el encabezado del mes **"largo"** y los nombres de día **"corto"**

Detalle de las preferencias:

Preferencia	Predeterminado	Opciones	Descripcion
template	No	No	Un string o array que contiene su plantilla de calendario
local_time	time ()	No	Una marca de tiempo Unix correspondiente a la hora actual
start_day	sunday	Cualquier día de la semana (sunday, monday, tuesday, etc.)	El día de la semana en que debe comenzar el calendario
month_type	long	long,short	Qué versión del nombre de mes se usará en el encabezado. long = January, short = Jan

day_type	abr	long, short, abr	Qué versión de los nombres del día de la semana se usará en los encabezados de columna long = Sunday, short = Sun, abr = Su
show_next_prev	FALSE	TRUE/FALSE (boolean)	Determina si mostrar enlaces que le permiten alternar a los meses siguientes / anteriores
next_prev_url	Controlador /método	Una URL	Establece la ruta base utilizada en los enlaces del calendario siguiente/anterior
show_other_days	FALSE	TRUE/FALSE (boolean)	Determina si mostrar los días de otros meses que comparten la primera o última semana del mes

Mostrar enlaces del mes siguiente/anterior

Para permitir que su calendario incremente/disminuya dinámicamente a través de los enlaces siguiente/anterior, debe configurar su código de forma similar a este:

Ejemplo

```
$prefs = array(
  'show_next_prev'  => TRUE,
  'next_prev_url'   => 'http://ejemplo.com/index.php/calendar/show/'
);

$this->load->library('calendar', $prefs);

echo $this->calendar->generate($this->uri->segment(3), $this->uri->segment(4));
```

Notará algunas cosas sobre el ejemplo anterior:

- Debe establecer "**show_next_prev**" en TRUE
- Debe proporcionar la URL al controlador que contiene su calendario en la preferencia "**next_prev_url**", si no lo hace, se establecerá en el controlador/método actual
- Debe proporcionar el "**año**" y "**mes**" a l método **generate()** a través de los segmentos de URI donde aparecen. Nota: la clase de calendario agrega automáticamente el año/mes a la URL base que proporcionó

Crear una plantilla de calendario

Al crear una plantilla de calendario, tiene un 100% de control sobre el diseño de su calendario, usando el método de string, cada componente de su calendario se colocará dentro de un par de pseudovariables como se muestra aquí:

```
$prefs['template'] = '

{table_open}<table border="0" cellpadding="0" cellspacing="0">{/table_open}
```

```
{heading_row_start}<tr>{/heading_row_start}

{heading_previous_cell}<th><a href="{previous_url}">&lt;&lt;</a></th> {/heading_previous_cell}
{heading_title_cell}<th colspan="{colspan}">{heading}</th> {/heading_title_cell}{heading_next_cell}<th><a
ref="{next_url}">&gt;&gt;</a></th> {/heading_next_cell}{heading_row_end}</tr>{/heading_row_end}

{week_row_start}<tr>{/week_row_start}
{week_day_cell}<td>{week_day}</td>{/week_day_cell}
{week_row_end}</tr>{/week_row_end}

{cal_row_start}<tr>{/cal_row_start}
{cal_cell_start}<td>{/cal_cell_start}
{cal_cell_start_today}<td>{/cal_cell_start_today}
{cal_cell_start_other}<td class="other-month">{/cal_cell_start_other}

{cal_cell_content}<a href="{content}">{day}</a>{/cal_cell_content}
{cal_cell_content_today}<div class="highlight"><a href="{content}">{day}</a></div>
{/cal_cell_content_today}

{cal_cell_no_content}{day}{/cal_cell_no_content}
{cal_cell_no_content_today}<div class="highlight">{day}</div> {/cal_cell_no_content_today}

{cal_cell_blank} {/cal_cell_blank}

{cal_cell_other}{day}{/cal_cel_other}

{cal_cell_end}</td>{/cal_cell_end}
{cal_cell_end_today}</td>{/cal_cell_end_today}
{cal_cell_end_other}</td>{/cal_cell_end_other}
{cal_row_end}</tr>{/cal_row_end}
{table_close}</table>{/table_close}';

$this->load->library('calendar', $prefs);

echo $this->calendar->generate();
```

Usando el método de array, pasará pares **clave=>valor**, puede pasar tantos o tan pocos valores como desee, las claves omitidas usarán los valores predeterminados heredados en la clase de calendario:

Ejemplo

```
$prefs['template'] = array(
  'table_open'           => '<table class="calendar">',
  'cal_cell_start'       => '<td class="day">',
  'cal_cell_start_today' => '<td class="today">'
);
$this->load->library('calendar', $prefs);
echo $this->calendar->generate();
```

class CI_Calendar

class CI_Calendar

class CI_Calendar

adjust_date()

array adjust_date(int $month, int $year)

Descripción

Este método asegura que tiene un mes/año válido, por ejemplo, si envía 13 como mes, el año aumentará y el mes se convertirá en enero

Parámetros

month
Mes

year
Año

Valores devueltos

Un array asociativa que contiene mes y año

```
print_r($this->calendar->adjust_date(13, 2017));
```

Resultado:

```
Array
(
    [month] => '01'
    [year]  => '2018'
)
```

default_template()

array default_template()

Descripción

Establece la plantilla predeterminada, este método se usa cuando no ha creado su propia plantilla

Valores devueltos

Un array con los valores de plantilla

generate()

string generate([int $year = '' [, int $month = '' [, array $data = array()]]])

Descripción

Genera el calendario

Parámetros

year

Año

month

Mes

data

Datos que se mostrarán en las celdas del calendario

Valores devueltos

Calendario con formato HTML

get_day_names()

```
array get_day_names(string $day_type = '')
```

Descripción

Devuelve un array con los nombres de días (Sunday, Monday, etc.) según el tipo proporcionado, opciones: largo, corto, abreviado, si no se proporciona **$day_type** ,o si se proporciona un tipo no válido, devolverá el estilo "**abreviado**"

Parámetros

day_type

'long', 'short' o 'abr'

Valores devueltos

Array con los nombres de días

get_month_name()

```
string get_month_name(int $month)
```

Descripción

Genera un texto con el nombre del mes correspondiente al número de mes proporcionado

Parámetros

month

Mes

Valores devueltos

Nombre del mes

get_total_days()

```
int get_total_days(int $month, int $year)
```

Descripción

El número de días en un mes dado

Nota

Este método es un alias para la función Date Helper **days_in_month()**

Parámetros

month

Mes

year

Año

Valores devueltos

El número de días en el mes especificado

Ejemplo

```
echo $this->calendar->get_total_days(2, 2016); // Presenta: 29
```

initialize()

```
object initialize([ array $config = array()])
```

Descripción

Inicia las preferencias del calendario

Parámetros

config

Array asociativo con las preferencias de visualización

Valores devueltos

Instancia CI_Calendar (method chaining)

parse_template()

```
object parse_template()
```

Descripción

Recolecta los datos dentro de la plantilla **{pseudovariables}** utilizada para mostrar el calendario

Valores devueltos

Instancia CI_Calendar (method chaining)

Config Class - Clase de configuración

Config Class proporciona un medio para recuperar las preferencias de configuración, estas preferencias pueden provenir del archivo de configuración predeterminado, **application/config/config.php** , o de sus propios archivos de configuración personalizados

Nota
Esta clase se inicializa automáticamente por el sistema y no es necesario hacerlo manualmente

Anatomía de un archivo de configuración

Por defecto, CodeIgniter tiene un archivo de configuración principal, ubicado en **application/config/config.php,** si abre el archivo con su editor de texto, verá que los elementos de configuración se almacenan en un array llamado **$config**

Puede agregar sus propios elementos de configuración en este archivo, o si prefiere mantener los elementos de configuración separados, suponiendo que necesite incluso elementos de configuración, simplemente cree su propio archivo y guárdelo en el directorio **config**

Nota
Si crea sus propios archivos de configuración, use el mismo formato que en el original, almacene sus elementos en un array llamado **$config**, CodeIgniter administrará de forma inteligente estos archivos para que no haya conflictos aunque el array tenga el mismo nombre, suponiendo que un índice de array no se llame igual que otro

Cargando un archivo de configuración

Nota
• Si crea sus propios archivos de configuración, use el mismo formato que en el original, almacene sus elementos en un array llamado **$config**, CodeIgniter administrará de forma inteligente estos archivos para que no haya conflictos aunque el array tenga el mismo nombre, suponiendo que un índice de array no se llame igual que otro • CodeIgniter carga automáticamente el archivo de configuración principal, **application/config/config.php**, por lo que solo tendrá que cargar un archivo de configuración si ha creado el suyo propio

Hay dos formas de cargar un archivo de configuración:

1 - Carga manual

Para cargar uno de sus archivos de configuración personalizados, usará la siguiente función dentro del controlador que lo necesita:

```
$this->config->load('nombre_de_archivo');
```

Donde **nombre_de_archivo** es el nombre de su archivo de configuración, sin la extensión de archivo **.php**

Si necesita cargar múltiples archivos de configuración normalmente, se fusionarán en un array de configuración maestro, no obstante, pueden producirse colisiones de nombres si ha nombrado de forma idéntica índices de array en diferentes archivos de configuración, para evitar colisiones, puede establecer el segundo parámetro en TRUE y cada archivo de configuración se almacenará en un índice de array correspondiente al nombre del archivo de configuración:

Ejemplo

```
// Almacenado en un array con este prototipo:
// $this->config['opciones_del_blog'] = $config

$this->config->load('opciones_del_blog', TRUE);
```

Consulte la sección **Recuperar elementos de configuración** para aprender cómo recuperar elementos de configuración establecidos de esta manera

El tercer parámetro le permite suprimir errores en caso de que no exista un archivo de configuración:

```
$this->config->load('opciones_del_blog', FALSE, TRUE);
```

2 - Carga automática

Si necesita un archivo de configuración particular de forma global, puede hacer que el sistema lo cargue automáticamente, para hacerlo, abra el archivo **autoload.php**, ubicado en **application/config/autoload.php** y agregue su archivo de configuración como se indica en el archivo

Recuperar elementos de configuración

Para recuperar un elemento de su archivo de configuración, use lo siguiente:

```
$this->config->item('item_name');
```

Donde **item_name** es el índice del array **$config** que desea recuperar

Por ejemplo, para buscar su elección de idioma, hará esto:

```
$lang = $this->config->item('language');
```

La función devuelve NULL si el elemento que está intentando recuperar no existe

Si utiliza el segundo parámetro del método **$this->config->load** para asignar sus elementos de configuración a un índice específico, puede recuperarlo indicando el nombre del índice en el segundo parámetro de **$this->config->item()**

Ejemplo

```
// Carga un archivo de configuración llamado opciones_del_blog.php y lo asigna a un
// índice llamado "opciones_del_blog"
$this->config->load('opciones_del_blog', TRUE);

// Recuperar un elemento de configuración denominado nombre_de_sitio contenido en el
// array opciones_del_blog
$site_name = $this->config->item('nombre_de_sitio', 'opciones_del_blog');

// Una forma alternativa de especificar el mismo elemento:
$opciones_del_blog = $this->config->item('opciones_del_blog');
$nombre_de_sitio = $opciones_del_blog['nombre_de_sitio'];
```

Establecer un elemento de configuración

Si desea establecer dinámicamente un elemento de configuración o cambiar uno existente, puede hacerlo usando:

```
$this->config->set_item('item_name', 'item_value');
```

Donde **item_name** es el índice del array **$config** que desea cambiar, e **item_value** es su valor

Entornos

Puede cargar diferentes archivos de configuración dependiendo del entorno actual, la constante **ENVIRONMENT** se define en **index.php** y se describe en detalle en la sección **Manejar múltiples entornos**

Para crear un archivo de configuración específico del entorno, cree o copie un archivo de configuración en **application/config/{ENVIRONMENT}/{FILENAME}.php**

Por ejemplo, para crear un **config.php** de solo producción, haría:

1. Cree el directorio **application/config/production/**
2. Copie su config.php existente en el directorio de arriba
3. Edite **application/config/production/config.php** para que contenga su configuración de producción

Cuando establezca la constante **ENVIRONMENT** en **'production'**, se cargarán las configuraciones para su nuevo **config.php** de solo producción

Puede colocar los siguientes archivos de configuración en carpetas específicas del entorno:

- Archivos de configuración de CodeIgniter predeterminados
- Sus propios archivos de configuración personalizados

Nota

CodeIgniter siempre carga primero el archivo de configuración global ,es decir, el de **application/config/**, luego intenta cargar los archivos de configuración para el entorno actual, esto significa que no está obligado a ubicar todos sus archivos de configuración en una carpeta de entorno, solo los archivos que cambian por entorno

Además, no tiene que copiar todos los elementos de configuración en el archivo de configuración del entorno, solo los elementos de configuración que desea cambiar para su entorno, los elementos de configuración declarados en las carpetas de su entorno siempre sobrescriben aquellos en sus archivos de configuración global

class CI_Config

class CI_Config

class CI_Config

$config

$config

Array de todos los valores de configuración cargados

$is_loaded

$is_loaded

Array de todos los archivos de configuración cargados

base_url()

string base_url()

Descripción

Este método recupera la URL de su sitio, más una ruta opcional, como una hoja de estilo o una imagen, a este método se accede normalmente a través de las funciones correspondientes en el **URL Helper**

Valores devueltos

URL base

item()

mixed item(string $item [, string $index = ''])

Descripción

Obtiene un elemento de archivo de configuración

Parámetros

item

Nombre del elemento de configuración

index

Nombre del índice

Valores devueltos

Valor de elemento de configuración o NULL si no se encuentra

load()

bool load([string $file = '' [, bool $use_sections = FALSE [, bool $fail_gracefully = FALSE]]])

Descripción

Carga un archivo de configuración

Parámetros

file

Nombre del archivo de configuración

use_sections

Si los valores de configuración deben cargarse en su propia sección (índice de la array de configuración principal)

fail_gracefully

Si se devuelve FALSE o se muestra un mensaje de error

Valores devueltos

TRUE en el éxito, FALSE si falla

set_item()

void set_item(string $item, string $value)

Descripción

Establece un elemento de archivo de configuración para el valor especificado

Parámetros

item

Nombre del elemento de configuración

value

Valor del elemento de configuración

valores devueltos

void

site_url()

string site_url()

Descripción

Este método recupera la URL de su sitio, junto con el valor de **"index"** que ha especificado en el archivo de configuración, a este método se accede normalmente a través de las funciones correspondientes en el **URL Helper**

Valores devueltos

URL del sitio

slash_item()

mixed slash_item(string $item)

Descripción

Este método es idéntico al método **item()** excepto que agrega una barra inclinada al final del elemento

Parámetros

item

Nombre del elemento

Valores devueltos

Valor del elemento de configuración con una barra inclinada o NULL si no se encuentra

system_url()

string system_url()

Descripción

Este método recupera la URL de su sistema/directorio CodeIgniter

> ***Nota***
>
> Este método está **DESACONSEJADO** porque fomenta el uso de prácticas de codificación inseguras, su sistema/directorio no debe ser de acceso público

Valores devueltos

URL que apunta a su sistema/directorio de CI

Correo electrónico

Enviar correo

Enviar correos electrónicos no es solo simple, sino que puede configurarlo sobre la marcha o establecer sus preferencias en un archivo de configuración

Aquí se muestra cómo puede enviar un correo electrónico. Nota: Se supone que está enviando el correo electrónico desde uno de sus controladores

Ejemplo

```
$this->load->library('email');

$this->email->from('tu@ejemplo.com', 'Tu nombre');
$this->email->to('alguien@ejemplo.com ');
$this->email->cc('otro@otro-ejemplo.com ');
$this->email->bcc('ellos@su-ejemplo.com ');

$this->email->subject('Email de prueba');
$this->email->message('Prueba de la clase Email.');

$this->email->send();
```

Establecer preferencias

Hay 21 preferencias diferentes disponibles para personalizar cómo se envían sus mensajes de correo electrónico, puede configurarlos manualmente como se describe aquí o automáticamente a través de las preferencias almacenadas en su archivo de configuración

Puede establecer las preferencias al pasar un array de valores de preferencia al método de inicialización del correo electrónico

Aquí puede ver como se pueden establecer algunas preferencias en su archivo de configuración:

Ejemplo

```
$config['protocol'] = 'sendmail';
$config['mailpath'] = '/ usr / sbin / sendmail';
$config['charset']  = 'iso-8859-1';
$config['wordwrap'] = TRUE;

$this->email->initialize($config);
```

Nota

La mayoría de las preferencias tienen valores predeterminados que se usarán si no los establece

Preferencias en un archivo de configuración

Si prefiere no establecer preferencias utilizando el método anterior, puede colocarlas en un archivo de configuración, cree un nuevo archivo llamado **email.php**, agregue el array **$config** en ese archivo, luego guarde el archivo en **config/email.php** y se usará automáticamente, si guarda sus preferencias en un archivo de configuración **NO** necesita usar el método **$this->email->initialize()**

Preferencias de correo electrónico

La siguiente es una lista de todas las preferencias que se pueden establecer al enviar un correo electrónico:

Preferencia	Predeterminado	Opciones	Descripción
useragent	CodeIgniter	No	El "agente de usuario"
protocol	mail	mail, sendmail o smtp	El protocolo de envío de correo
mailpath	/usr/sbin/sendmail	No	La ruta del servidor a Sendmail
smtp_host	No	No	Dirección del Servidor SMTP
smtp_host	No	No	Dirección SMTP del Servidor
smtp_user	No	No	Nombre de Usuario SMTP
smtp_pass	No	No	Contraseña SMTP
smtp_port	25	No	Puerto SMTP
smtp_timeout	5	No	SMTP Timeout (en segundos)
smtp_keepalive	FALSE	TRUE o FALSE (boolean)	Habilita las conexiones SMTP persistentes
smtp_crypto	No	tls o ssl	SMTP Encryption
wordwrap	TRUE	TRUE o FALSE (boolean)	Habilita el word-wrap
wrapchars	76		Número de caracteres para el salto de línea
mailtype	text	text o html	Tipo de correo. Si envía un correo electrónico en HTML, debe enviarlo como una página web completa. Asegúrese de que no tiene ningún enlace relativo o ruta de imagen relativa; de lo contrario, no funcionará
charset	$config['charset']		Juego de caracteres (utf-8, iso-8859-1, etc.)
validate	FALSE	TRUE o FALSE (boolean)	Si validar la dirección de correo electrónico
priority	3	1, 2, 3, 4, 5	Prioridad del correo electrónico:. 1 = la más alta. 5 = la más baja. 3 = normal
crlf	\n	"\r\n" o "\n" o "\r"	Carácter de nueva línea. (Use "\r\n" para cumplir con RFC 822)

newline	\n	"\r\n" o "\n" o "\r"	Carácter de nueva línea. (Use "\r\n" para cumplir con RFC 822)
bcc_batch_mode	FALSE	TRUE o FALSE (boolean)	Habilita el modo BCC Batch
bcc_batch_size	200	No	Número de correos electrónicos en cada lote de BCC
dsn	FALSE	TRUE o FALSE (boolean)	Habilitar notificar mensaje del servidor

Word Wrapping

Si tiene habilitado el **word wrapping**, salto de línea automático, se recomienda para cumplir con RFC 822, y tiene un enlace muy largo en su correo electrónico, puede ocurrir que el salto de línea automático lo corte, causando que se vuelva imposible de cliquear por la persona que lo recibe, CodeIgniter le permite anular manualmente el salto de línea automático dentro de parte de su mensaje:

```
El texto de su correo lectrónico que
normalmente salta de línea.

{unwrap}http://ejemplo.com/un_enlace_largo_que_no_debe_cortarse.html{/unwrap}

Más texto que normalmente
tiene su salto de línea.
```

Coloque el elemento que no quiere que se corte entre: **{unwrap} {/unwrap}**

class CI_Email

class CI_Email

class CI_Email

attach()

object attach(string $filename [, string $disposition = '' [, string $newname = NULL [, string $mime = '']]])

Descripción

Le permite enviar un archivo adjunto, coloque la ruta/nombre del archivo en el primer parámetro, para archivos adjuntos múltiples, use el método varias veces

Parámetros

filename

Nombre de archivo

disposition

'disposición' del archivo adjunto, la mayoría de los clientes de correo electrónico toman su propia decisión independientemente de la especificación MIME utilizada

vea: **https://www.iana.org/assignments/cont-disp/cont-disp.xhtml**

newname

Nombre de archivo personalizado para usar en el correo electrónico

mime

Tipo MIME a usar (útil para datos almacenados en búfer)

Valores devueltos

Instancia de CI_Email (method chaining)

Ejemplo

```
$this->email->attach('/path/to/photo1.jpg');
$this->email->attach('/path/to/photo2.jpg');
$this->email->attach('/path/to/photo3.jpg');
```

Para usar la disposición predeterminada, archivo adjunto, deje en blanco el segundo parámetro, de lo contrario, use una disposición personalizada:

```
$this->email->attach('imagen.jpg', 'inline');
```

También puede usar una URL:

```
$this->email->attach('http://ejemplo.com/nombre_de_archivo.pdf');
```

Si desea usar un nombre de archivo personalizado, puede usar el tercer parámetro:

```
$this->email->attach('nombre_de_archivo.pdf', 'attachment', 'report.pdf');
```

Si necesita utilizar un string de búfer en lugar de un archivo físico real, puede usar el primer parámetro como búfer, el tercer parámetro como nombre de archivo y el cuarto parámetro como tipo MIME:

```
$this->email->attach($buffer, 'attachment', 'report.pdf', 'application/pdf');
```

attachment_cid()

string attachment_cid(string $filename)

Descripción

Establece y devuelve **Content-ID** de un archivo adjunto, que le permite insertar un archivo adjunto en línea, imagen, en HTML, el primer parámetro debe ser el nombre del archivo ya adjunto

Parámetros

filename

Archivo adjunto existente

Valores devueltos

Adjunto Content-ID o FALSE si no se encuentra

Ejemplo

```
$nom_archivo = '/img/foto1.jpg';
$this->email->attach($nom_archivo);
foreach ($lista as $direccion)
{
   $this->email->to($direccion);
   $cid = $this->email->attachment_cid($nom_archivo);
   $this->email->message('<img src="cid:'. $cid .'" alt="foto1" />');
   $this->email->send();
}
```

Nota

Content-ID debe ser único en cada correo electrónico, debe volver a generarlo para que sea único

bcc()

object bcc(mixed $bcc [, int $limit = ''])

Descripción

Establece las direcciones de correo electrónico de **BCC**, al igual que el método **to()** puede ser un solo correo, una lista delimitada por comas o un array

Parámetros

bcc

Cadena delimitada por comas o un array de direcciones de correo electrónico

limit

Número máximo de correos para enviar por lote, si se establece **$limit** se habilitará el modo **"batch"**, modo por lotes, que enviará los correos en lotes sin que cada lote exceda el valor especificado en **$limit**

Valores devueltos

Instancia de CI_Email (method chaining)

cc()

object cc(mixed $cc)

Descripción

Establece las direcciones de correo electrónico **CC**, al igual que en **"to"**, puede ser un solo correo electrónico, una lista delimitada por comas o un array

Parámetros

cc

Cadena delimitada por comas o un array de direcciones de correo electrónico

Valores devueltos

Instancia de CI_Email (method chaining)

clear()

object clear([bool $clear_attachments = FALSE])

Descripción

Inicializa todas las variables de correo electrónico a un estado vacío, este método está destinado a utilizarse si ejecuta el método de envío de correo electrónico en bucle, lo que permite que los datos se restablezcan entre ciclos

Parámetros

clear_attachments

Si borrar o no los archivos adjuntos

Valores devueltos

Instancia de CI_Email (method chaining)

Ejemplo

```
foreach ($lista as $nombre => $direccion)
{
   $this->email->clear();

   $this->email->to($direccion);
   $this->email->from('su@ejemplo.com ');
   $this->email->subject('Aquí está su información'. $nombre);
   $this->email->message('Hola'. $nombre. 'Aquí está la información que solicitó.');
   $this->email->send();
}
```

Si establece el parámetro en TRUE, también se borrarán los archivos adjuntos:

```
$this->email->clear(TRUE);
```

from()

object from(string $from [, string $name = '' [, string $return_path = NULL]])

Descripción

Establece la dirección de correo electrónico, el nombre de la persona que lo envía y la dirección de devolución

Parámetros

from

Dirección de correo electrónico

name

Nombre para mostrar

return_path

Dirección de correo electrónico en caso de devolución, si el correo no se ha entregado

Valores devueltos

Instancia de CI_Email (method chaining)

Ejemplo

Establece la dirección de correo electrónico y el nombre de la persona que lo envía:

```
$this->email->from('mi@ejemplo.com', 'Mi nombre');
```

También puede establecer un Return-Path para ayudar a redirigir el correo no entregado:

```
$this->email->from('mi@ejemplo.com', 'Mi nombre', 'email_devuelto@ejemplo.com');
```

> ***Nota***
>
> Return-Path no se puede usar si ha configurado '**smtp**' como su protocolo

message()

object message(string $body)

Descripción

Establece el cuerpo del mensaje del correo electrónico

Parámetros

body

Cuerpo del mensaje del correo

Valores devueltos

Instancia de CI_Email (method chaining)

Ejemplo

```
$this->email->message('Este es mi mensaje');
```

print_debugger()

string print_debugger([array $include = array('headers', 'subject', 'body')])

Descripción

Devuelve un string que contiene los mensajes del servidor, los encabezados y el mensaje del correo electrónico, útil para la depuración

Parámetros

include

Opcional, la parte del mensaje a imprimir, las opciones son: **'headers'** (encabezados), **'subject'** (asunto), **'body'** (cuerpo)

Valores devueltos

Datos de depuración formateados

Ejemplo

```
// Necesita pasar FALSE en el método send() para que los datos no se borren,
// en caso contrario print_debugger() no tendría nada que mostrar.
$this->email->send(FALSE);

// Solo imprimirá los encabezados de correo electrónico, excluyendo el mensaje,
// asunto y cuerpo
$this->email->print_debugger(array('headers'));
```

> ***Nota***
>
> De manera predeterminada, se imprimirán todos los datos en bruto

reply_to()

object reply_to(string $replyto [, string $name = ''])

Descripción

Establece la dirección de respuesta, si no se proporciona la información, se utiliza la información del método **from()**

Parámetros

replyto
: Dirección de correo para respuesta

name
: Nombre para mostrar para la dirección de correo de respuesta

Valores devueltos

Instancia de CI_Email (method chaining)

Ejemplo

```
$this->email->reply_to('mi@ejemplo.com', 'Mi nombre');
```

send()

bool send([bool $auto_clear = TRUE])

Descripción

El método de envío de correo electrónico

Parámetros

auto_clear
: Si borrar los datos del mensaje automáticamente

Valores devueltos

TRUE en el éxito, FALSE si falla

Ejemplo

```
if ( ! $this->email->send())
{
   // Generar error
}
```

Este método borrará automáticamente todos los parámetros si la solicitud tuvo éxito, para detener este comportamiento pase FALSE:

```
if ($this->email->send(FALSE))
{
   // Los parámetros no se borran
}
```

> ***Nota***
>
> Para utilizar el método **print_debugger()** debe evitar borrar los parámetros del correo electrónico

set_alt_message()

object set_alt_message(string $str)

Descripción

Establece el cuerpo alternativo del correo electrónico

Parámetros

str

Cuerpo alternativo del mensaje de correo electrónico, si envía un correo en formato HTML , le permite especificar un mensaje alternativo sin formato HTML que se agrega al string de encabezado para las personas que no aceptan correo en HTML, si no configura su propio mensaje, CodeIgniter extraerá el mensaje de su correo en HTML y quitará las etiquetas

Valores devueltos

Instancia de CI_Email (method chaining)

Ejemplo

```
$this->email->set_alt_message('Este es el mensaje alternativo si no acepta HTML');
```

set_header()

object set_header(string $header, string $value)

Descripción

Agrega encabezados adicionales al correo electrónico

Parámetros

header

Nombre del encabezado

value

Valor de encabezado

Valores devueltos

Instancia de CI_Email (method chaining)

Ejemplo

```
$this->email->set_header('Encabezado1', 'Valor1');
$this->email->set_header('Encabezado2', 'Valor2');
```

subject()

object subject(string $subject)

Descripción

Establece el asunto del correo electrónico

Parámetros

subject

Asunto del correo

Valores devueltos

Instancia de CI_Email (method chaining)

Ejemplo

```
$this->email->subject('Este es mi asunto');
```

to()

object to(mixed $to)

Descripción

Establece las direcciones de correo electrónico de los destinatarios, puede ser un solo correo, una lista delimitada por comas o un array

Parámetros

to

Cadena delimitada por comas o un array de direcciones de correo

Valores devueltos

Instancia de CI_Email (method chaining)

Ejemplo

```
$this->email->to('alguien@ejemplo.com');
$this->email->to('uno@ejemplo.com, dos@ejemplo.com, tres@ejemplo.com');
$this->email->to(array ('uno@ejemplo.com','dos@ejemplo.com','tres@ejemplo.com'));
```

Encryption Library

Importante

¡NO use esta o ninguna otra biblioteca de cifrado para el almacenamiento de contraseñas del usuario!

Las contraseñas deben guardarse en **hash** y debe hacerlo a través de la propia extensión **Hashing Password** de PHP: **http://php.net/password**

La biblioteca de cifrado proporciona cifrado de datos bidireccional, para hacerlo de una manera criptográficamente segura, utiliza extensiones PHP que no siempre están disponibles en todos los sistemas. Debe contar con una de las siguientes dependencias para usar esta biblioteca:

- **OpenSSL**
- **MCrypt** (con **MCRYPT_DEV_URANDOM** disponible)

Si no se cumple ninguna de las dependencias anteriores, simplemente no podemos ofrecerle una implementación lo suficientemente buena como para cumplir con los altos estándares requeridos para una criptografía adecuada

Iniciar Encryption Library

Encryption Library se inicia usando **$this->load->library()**:

```
$this->load->library('encryption');
```

Una vez cargada, el objeto esta disponible usando:

```
$this->encryption
```

Comportamiento por defecto

De forma predeterminada, la biblioteca de cifrado utiliza el cifrado AES-128 en modo CBC, utilizando la clave de cifrado configurada y la autenticación HMAC SHA512

Nota

AES-128 se elige tanto porque se ha demostrado que es sólido como por su amplia disponibilidad en diferentes API de lenguajes de programación y software criptográfico

Sin embargo, la clave de cifrado no se usa tal cual

Si está familiarizado con la criptografía, ya debe saber que un HMAC también requiere una clave secreta y usar la misma clave tanto para el cifrado como para la autenticación es una mala práctica, debido a eso, dos claves separadas se

derivan de su clave de cifrado ya configurada: una para el cifrado y otra para la autenticación. Esto se realiza a través de una técnica llamada **HMAC-based Key Derivation Function (HKDF)**

Configurar la clave de cifrado

Una clave de cifrado es una información que controla el proceso criptográfico y permite encriptar un string de texto sin formato y luego descifrarla, es el "ingrediente" secreto en todo el proceso que le permite ser el único que puede descifrar los datos que ha decidido ocultar, después de utilizar una clave para cifrar los datos, esa misma clave proporciona el único medio para descifrarlos, por lo que no solo debe elegir la clave cuidadosamente, sino que no debe perderla o perderá también el acceso a los datos cifrados

Debe tenerse en cuenta que para garantizar la máxima seguridad, dicha clave no solo debe ser lo más fuerte posible, sino que también debe cambiarse con frecuencia, sin embargo, este comportamiento rara vez es práctico o posible de implementar, y es por eso que CodeIgniter le brinda la capacidad de configurar una única clave que se utilizará (casi) cada vez

No hace falta decir que debe guardar su clave cuidadosamente, si alguien obtiene acceso a su clave, los datos se descifrarán fácilmente

Si su servidor no está totalmente bajo su control, es imposible garantizar la seguridad de las claves, por lo que es posible que desee pensar detenidamente antes de utilizarlo para cualquier cosa que requiera alta seguridad, como almacenar números de tarjetas de crédito

Su clave de cifrado debe ser tan larga como lo permita el algoritmo de encriptación en uso, para AES-128, eso es 128 bits o 16 bytes (caracteres) de largo, mas abajo puede encontrar una tabla que muestra las longitudes de claves admitidas en diferentes cifrados

La clave debe ser lo más aleatoria posible y no debe ser un string de texto normal, ni el resultado de una función de hash, etc., para crear una clave adecuada, debe usar el método **create_key()** de la **Encryption Library**

```
//A $key se le asignará una clave aleatoria de 16 bytes (128 bits)
$key = $this->encryption->create_key(16);
```

La clave puede almacenarse en su **application/config/config.php** o puede diseñar su propio mecanismo de almacenamiento y pasar la clave de forma dinámica al cifrar/descifrar

Para guardar su clave en su **application/config/config.php** abra el archivo y configure:

```
$config['encryption_key'] = 'SU CLAVE';
```

Notará que el método **create_key()** genera datos binarios, lo cual es difícil de manejar, es decir, copiar y pegar puede dañarlo, así que puede usar **bin2hex()**, **hex2bin()** o codificación **Base64** para trabajar con la clave de una manera más amigable:

Ejemplo

```
// Obtenga una representación codificada en hexadecimal de la clave:
$key = bin2hex($this->encryption->create_key(16));

// Ponga el mismo valor en su configuración con hex2bin(), para que aún se pase como binario a la
biblioteca:
$config['encryption_key'] = hex2bin(<su clave con codificación hexadecimal>);
```

Cifrados y modos de encriptación admitidos

Nota

Los términos 'cifrado' y 'algoritmo de cifrado' son intercambiables

Cifrados portábles

Como MCrypt y OpenSSL admiten diferentes conjuntos de algoritmos de cifrado y, a menudo, los implementan de diferentes maneras, nuestra biblioteca de cifrado está diseñada para usarlos de manera **portable**, o **en otras palabras**, le **permite intercambiarlos**, al menos para los cifrados admitidos por ambos

También se implementa de una manera que pretende hacer coincidir las implementaciones estándar en otros lenguajes de programación y bibliotecas

Aquí hay una lista de los llamados sistemas de cifrado **"portables"**, donde **"CodeIgniter name"** es el valor de string que tiene que pasar a la **Encryption library** para usar ese cifrado:

Nombre de cifrado	CodeIgniter name	Longitudes de clave (bits / bytes)	Modos admitidos
AES-128 / Rijndael-128	aes-128	128/16	CBC, CTR, CFB, CFB8, OFB, BCE
AES-192	aes-192	192/24	CBC, CTR, CFB, CFB8, OFB, BCE
AES-256	aes-256	256/32	CBC, CTR, CFB, CFB8, OFB, BCE
DES	des	56/7	CBC, CFB, CFB8, OFB, BCE
TripleDES	triples	56/7, 112/14, 168/21	CBC, CFB, CFB8, OFB
Blowfish	blowfish	128-448 / 16-56	CBC, CFB, OFB, BCE
CAST5 / CAST-128	cast5	88-128 / 11-16	CBC, CFB, OFB, BCE
RC4 / ARCFour	rc4	40-2048 / 5-256	Stream

Importante

Debido a la forma en que funciona MCrypt, si no proporciona una clave con la longitud adecuada, puede terminar usando un algoritmo diferente al configurado, ¡así que tenga mucho cuidado con eso!

Nota

- En caso de que no quede claro en la tabla anterior, Blowfish, CAST5 y RC4 admiten claves de longitud variable, es decir, cualquier número en los rangos mostrados es válido, aunque en términos de bit solo ocurre en incrementos de 8 bits

- A pesar de que CAST5 admite longitudes de clave inferiores a 128 bits (16 bytes), de hecho se rellenarán con ceros a la longitud máxima, como se especifica en RFC 2144

- Blowfish admite longitudes de clave tan pequeñas como 32 bits (4 bytes), pero nuestras pruebas han demostrado que solo las longitudes de 128 bits (16 bytes) o más son compatibles tanto con MCrypt como con OpenSSL, también es una mala práctica usar claves de longitud baja de todos modos

Cifrados específicos del controlador

Como se señaló anteriormente, MCrypt y OpenSSL admiten diferentes conjuntos de cifrado, por motivos de portabilidad y debido a que no los hemos probado correctamente, no le recomendamos que use los que son específicos del controlador, pero independientemente, aquí hay una lista de la mayoría de ellos:

Nombre de cifrado	Driver	Longitudes de clave del controlador (bits / bytes)	Modos admitidos
AES-128	OpenSSL	128/16	CBC, CTR, CFB, CFB8, OFB, BCE, XTS
AES-192	OpenSSL	192/24	CBC, CTR, CFB, CFB8, OFB, BCE, XTS
AES-256	OpenSSL	256/32	CBC, CTR, CFB, CFB8, OFB, BCE, XTS
Rijndael-128	MCrypt	128/16, 192/24, 256/32	CBC, CTR, CFB, CFB8, OFB, OFB8, BCE
Rijndael-192	MCrypt	128/16, 192/24, 256/32	CBC, CTR, CFB, CFB8, OFB, OFB8, BCE
Rijndael-256	MCrypt	128/16, 192/24, 256/32	CBC, CTR, CFB, CFB8, OFB, OFB8, BCE
GOST	MCrypt	256/32	CBC, CTR, CFB, CFB8, OFB, OFB8, BCE
Twofish	MCrypt	128/16, 192/24, 256/32	CBC, CTR, CFB, CFB8, OFB, OFB8, BCE
CAST-128	MCrypt	40-128 / 5-16	CBC, CTR, CFB, CFB8, OFB, OFB8, BCE
CAST-256	MCrypt	128/16, 192/24, 256/32	CBC, CTR, CFB, CFB8, OFB, OFB8, BCE
Loki97	MCrypt	128/16, 192/24, 256/32	CBC, CTR, CFB, CFB8, OFB, OFB8, BCE
SaferPlus	MCrypt	128/16, 192/24, 256/32	CBC, CTR, CFB, CFB8, OFB, OFB8, BCE
Serpent	MCrypt	128/16, 192/24, 256/32	CBC, CTR, CFB, CFB8, OFB, OFB8, BCE
XTEA	MCrypt	128/16	CBC, CTR, CFB, CFB8, OFB, OFB8, BCE
RC2	MCrypt	8-1024 / 1-128	CBC, CTR, CFB, CFB8, OFB, OFB8, BCE
RC2	OpenSSL	8-1024 / 1-128	CBC, CFB, OFB, BCE

Camellia-128	OpenSSL	128/16	CBC, CFB, CFB8, OFB, BCE
Camellia-192	OpenSSL	192/24	CBC, CFB, CFB8, OFB, BCE
Camellia-256	OpenSSL	256/32	CBC, CFB, CFB8, OFB, BCE
Seed	OpenSSL	128/16	CBC, CFB, OFB, BCE

Nota

- Si desea utilizar uno de esos sistemas de cifrado, deberá pasar su nombre en minúscula a la **Encryption library**
- Probablemente haya notado que la lista también incluye todos los AES (y Rijndael-128), esto se debe a que los controladores admiten diferentes modos para estos cifrados, además, es importante tener en cuenta que AES-128 y Rijndael-128 son en realidad el mismo cifrado, pero solo cuando se usan con una clave de 128 bits
- CAST-128 / CAST-5 también aparece en la lista de cifrados portables y específicos del controlador, esto se debe a que la implementación de OpenSSL no parece funcionar correctamente con tamaños de clave de 80 bits o menos
- RC2 aparece en la lista como compatible con MCrypt y OpenSSL, sin embargo, ambos controladores los implementan de manera diferente y no son intercambiables, debemos señalar que solo encontramos una fuente que indica que MCrypt no lo implementa correctamente

Modos de encriptación

Los diferentes modos de cifrado tienen diferentes características y sirven para diferentes propósitos, algunos son más fuertes que otros, algunos son más rápidos y otros ofrecen características adicionales. No vamos a profundizar en eso aquí, se lo dejamos a los expertos en criptografía

La siguiente tabla es para proporcionar una breve referencia informativa a nuestros usuarios más experimentados, si usted es un principiante, simplemente manténgase en el modo CBC; es ampliamente aceptado como fuerte y seguro para propósitos generales:

Nombre del modo	**Nombre de CodeIgniter**	**Soporte del controlador**	**Información adicional**
CBC	cbc	MCrypt, OpenSSL	Una opción predeterminada segura
CTR	ctr	MCrypt, OpenSSL	Considerado como teóricamente mejor que CBC, pero no tan ampliamente disponible
CFB	cfb	MCrypt, OpenSSL	N / A
CFB8	cfb8	MCrypt, OpenSSL	Igual que CFB, pero funciona en modo de 8 bits (no recomendado)
OFB	ofb	MCrypt, OpenSSL	N / A
OFB8	ofb8	MCrypt	Igual que OFB, pero funciona en modo de 8 bits (no recomendado).

ECB	ecb	MCrypt, OpenSSL	Ignores IV (no recomendado)
XTS	xts	OpenSSL	Generalmente se utiliza para cifrar datos de acceso aleatorio, como la memoria RAM o el almacenamiento en el disco duro
Stream	stream	MCrypt, OpenSSL	Esto no es realmente un modo, solo dice que se usa un cifrado de flujo. Requerido debido al proceso de inicialización del modo de cifrado general

Longitud del mensaje

Probablemente sea importante que sepa que un string cifrado generalmente es más larga que el string original de texto sin formato, dependiendo del cifrado, esto está influenciado por el algoritmo de cifrado en sí, el IV (vector de inicialización) antepuesto al texto de cifrado y el mensaje de autenticación HMAC que también está precedido, además, el mensaje cifrado también está codificado en Base64 para que sea seguro para el almacenamiento y la transmisión, independientemente de un posible conjunto de caracteres en uso

Tenga en cuenta esta información al seleccionar su mecanismo de almacenamiento de datos, las cookies, por ejemplo, solo pueden contener 4K de información

Configurar la biblioteca

Para uso, rendimiento, pero también por razones históricas relacionadas con nuestra antigua Encrypt Class, la **Encryption library** está diseñada para usar repetidamente el mismo controlador, cifrado , modo y clave

Como se indicó en la sección anterior "**Comportamiento predeterminado**", esto significa utilizar un controlador de detección automática, OpenSSL tiene una prioridad más alta, el cifrado AES-128 en modo CBC y su valor de **$config['encryption_key']**

Sin embargo, si desea cambiar eso, necesita usar el método **initialize()**, que acepta un array asociativo de parámetros, todos los cuales son opcionales:

Opción	Valores posibles
driver	'mcrypt', 'openssl'
cipher	Nombre de cifrado cifrado (consulte Cifrados y modos de cifrado admitidos)
mode	Modo de encriptación (ver modos de encriptación)
key	Clave de cifrado

Por ejemplo, si cambiara el algoritmo y el modo de cifrado a AES-256 en modo CTR, esto es lo que debe hacer:

```
$this->encryption->initialize(
  array(
   'cipher' => 'aes-256',
   'mode'   => 'ctr',
   'key'    => '<una string aleatoria de 32 caracteres>'
  )
);
```

Tenga en cuenta que solo mencionamos que desea cambiar el cifrador y el modo, pero también incluimos una clave en el ejemplo, como se indicó anteriormente, es importante que elija una clave con un tamaño adecuado para el algoritmo utilizado

También existe la posibilidad de cambiar el controlador, si por alguna razón tiene ambos, pero desea utilizar **MCrypt** en lugar de **OpenSSL**:

```
// Cambiar al controlador de MCrypt
$this->encryption->initialize(array('driver' => 'mcrypt'));

// Volver al controlador OpenSSL
$this->encryption->initialize(array('driver' => 'openssl'));
```

Cifrado y descifrado de datos

Cifrar y descifrar datos con la configuración de la biblioteca ya configurada es simple, tan simple como pasar el string a los métodos **encrypt()** y / o **decrypt()**:

```
$texto_plano = '¡Este es un mensaje de texto simple!';
$texto_cifrado = $this->encryption->encrypt($texto_plano);

// Salidas: ¡Este es un mensaje de texto simple!
echo $this->encryption->decrypt($texto_cifrado);
```

¡Y eso es! , la **Encryption library** hará todo lo necesario para que todo el proceso esté criptográficamente seguro desde el primer momento, no necesita preocuparse por eso

Importante

Ambos métodos devolverán **FALSE** en caso de error, mientras que para **encrypt()** esto solo puede significar una configuración incorrecta, siempre debe verificar el valor de retorno de **decrypt()** en el código de producción

Cómo funciona

Si quiere saber cómo funciona el proceso, esto es lo que sucede bajo el capó:

$this->encryption->encrypt($texto_plano)

1. Obtiene una clave de encriptación y una clave HMAC de su clave de cifrado configurada a través de HKDF, utilizando el algoritmo de resumen SHA-512
2. Genera un vector de inicialización aleatorio (IV)
3. Cifra los datos a través de AES-128 en modo CBC ,u otro modo y cifrado previamente configurado, utilizando la clave de cifrado derivada mencionada anteriormente y IV
4. Antepone dicho IV al texto de cifrado resultante

5 Base64-codifica el string resultante, para que pueda ser almacenado o transferido sin preocuparse por los juegos de caracteres
6 Crea un mensaje de autenticación SHA-512 HMAC utilizando la clave HMAC derivada para garantizar la integridad de los datos y anteponerla al string Base64

$this->encryption->decrypt($texto_cifrado)

1 Obtiene una clave de encriptación y una clave HMAC de su clave de cifrado configurada a través de HKDF, utilizando el algoritmo SHA-512. Debido a que su **encryption_key** de configuración es la misma, esto producirá el mismo resultado que en el método **encrypt()** anterior, de lo contrario, no podrá descifrarlo

2 Comprueba si el string es lo suficientemente largo, separa el HMAC y valida si es correcto, esto se hace de una manera que evite los ataques de temporización, devuelve FALSE si alguna de las comprobaciones falla

3 Base64-decodifica el string

4 Separa el IV del texto de cifrado y descifra dicho texto de cifrado utilizando ese IV y la clave de cifrado derivada

Parámetros personalizados

Supongamos que tiene que interactuar con otro sistema que está fuera de su control y utiliza otro método para cifrar datos, un método que seguramente no coincidirá con la secuencia descrita anteriormente y probablemente tampoco se usen todos los pasos

La Encryption library le permite cambiar la forma en que funcionan sus procesos de cifrado y descifrado, de modo que pueda adaptar fácilmente una solución personalizada para tales situaciones

Nota
Es posible usar la biblioteca de esta manera, sin establecer una clave de encriptación en su archivo de configuración

Todo lo que tiene que hacer es pasar un array asociativo con algunos parámetros al método **encrypt()** o **decrypt()**:

Ejemplo

```
// Supongamos que tenemos $texto_cifrado, $clave y $clave_hmac
// procedente de otra fuente

$mensaje = $this->encryption->decrypt(
 $texto_cifrado,
 array(
    'cipher'      => 'blowfish',
    'mode'        => 'cbc',
    'key'         => $clave,
    'hmac_digest' => 'sha256',
    'hmac_key'    => $clave_hmac
 )
);
```

En el ejemplo anterior, estamos descifrando un mensaje cifrado mediante el cifrado Blowfish en modo CBC y autenticado a través de un HMAC SHA-256

Importante

Tenga en cuenta que tanto '**clave**' como '**clave_hmac**' se utilizan en este ejemplo, al usar parámetros personalizados, las claves de cifrado y HMAC no se derivan como en el comportamiento predeterminado de la biblioteca

A continuación hay una lista de las opciones disponibles

Sin embargo, a menos que realmente lo necesite y sepa lo que está haciendo, le aconsejamos que no modifique el proceso de cifrado, ya que podría afectar la seguridad, por lo que, si lo hace, debe hacerlo con precaución

Opción	Predeterminado	Obligatorio / Opcional	Descripción
cipher	N/A	Si	Algoritmo de encriptación (consulte **Cifrados y modos de encriptación admitidos**)
mode	N/A	Si	Modo de encriptación (ver **modos de encriptación**)
key	N/A	Si	Clave de encriptación
hmac	TRUE	No	Si se debe usar HMAC. Booleano Si se establece en FALSE, entonces **hmac_digest** y **hmac_key** serán ignorados
hmac_digest	sha512	No	Algoritmo de resumen de mensaje HMAC (ver **Algoritmos de utenticación HMAC admitidos**)
hmac_key	N/A	Sí, a menos que hmac sea FALSE	clave HMAC
raw_data	FALSE	No	Si el texto de cifrado debe estar en bruto. Booleano Si se establece en TRUE, la codificación y decodificación de Base64 no se realizará y HMAC no será un string hexadecimal

Importante

encrypt() y **decrypt()** devolverá **FALSE** si no se proporciona un parámetro obligatorio o si el valor proporcionado es incorrecto, esto incluye **hmac_key**, a menos que hmac esté configurado en **FALSE**

Algoritmos de autenticación HMAC admitidos

Para la autenticación de mensajes HMAC, la biblioteca de cifrado admite el uso de la familia de algoritmos SHA-2:

Algoritmo	Longitud en bruto (bytes)	Longitud codificada en Hexadecimal(bytes)
sha512	64	128
sha384	48	96
sha256	32	64
sha224	28	56

La razón por la que no se incluyen otros algoritmos populares, como MD5 o SHA1, es que ya no se consideran lo suficientemente seguros y, por lo tanto, no queremos fomentar su uso, si realmente necesita usarlos, es fácil hacerlo a través de la función nativas de PHP **hash_hmac()**

Algoritmos más fuertes, por supuesto, se agregarán en el futuro a medida que aparezcan y estén ampliamente disponibles

class CI_Encryption

class CI_Encryption

```
class CI_Encryption
```

create_key()

```
string create_key(int $length)
```

Descripción

Crea una clave criptográfica obteniendo datos aleatorios de las fuentes del sistema operativo (es decir, /dev/urandom)

Parámetros

length

Longitud de salida

Valores devueltos

Una clave criptográfica pseudoaleatoria con la longitud especificada o FALSE en caso de error

decrypt()

```
string decrypt(string $data [, array $params = NULL])
```

Descripción

Descifra los datos de entrada y los devuelve en texto sin formato

Parámetros

data

Datos para descifrar

params

Parámetros opcionales

Valores devueltos

Datos descifrados o FALSE en caso de error

Ejemplo

```
echo $this->encryption->decrypt($ciphertext);
```

Consulte la sección **Uso de parámetros personalizados** para obtener información sobre los parámetros opcionales

encrypt()

string encrypt(string $data [, array $params = NULL])

Descripción

Encripta los datos de entrada y devuelve el texto cifrado

Parámetros

data

Datos para encriptar

params

Parámetros opcionales

Valores devueltos

Datos cifrados o FALSE en caso de error

Ejemplo

```
$ciphertext = $this->encryption->encrypt('Mi mensaje secreto');
```

Consulte la sección **Uso de parámetros personalizados** para obtener información sobre los parámetros opcionales

hkdf()

string hkdf(string $key [, string $digest = 'sha512' [, string $salt = NULL [, int $length = NULL [, string $info = '']]]])

Descripción

Deriva una clave de otra, presumiblemente más débil, este método se usa internamente para derivar un cifrado y una clave HMAC de su clave de cifrado configurada, está disponible públicamente debido a su propósito general, se describe en RFC 5869, sin embargo, a diferencia de la descripción en RFC 5869, esta implementación no es compatible con SHA1

Parámetros

key

Material clave de entrada

digest

Un algoritmo de resumen de familia SHA-2

salt

Salt opcional

length

Longitud de salida opcional

info

Información opcional de contexto / aplicación específica

Valores devueltos

Una clave pseudoaleatoria o FALSE en caso de error

Ejemplo

```
$hmac_key = $this->encryption->hkdf(
    $clave,
    'sha512',
    NULL,
    NULL,
    'authentication'
);

 // $hmac_key es una clave pseudoaleatoria con una longitud de 64 bytes
```

initialize()

object initialize(array $params)

Descripción

Inicializa, configura, la biblioteca para usar un controlador, cifrado, modo o clave diferente

Parámetros

params

Parámetros de configuración

Valores devueltos

Instancia CI_Encryption (method chaining)

Ejemplo

```
$this->encryption->initialize(
  array('mode' => 'ctr')
);
```

Consulte la sección Configuración de la biblioteca para obtener información detallada

File Uploading Class

File Uploading Class permite subir archivos, puede establecer varias preferencias, restringiendo el tipo y el tamaño de los archivos

El proceso

Subir un archivo implica el siguiente proceso:

1. Se muestra un formulario de carga, que permite a un usuario seleccionar un archivo y subirlo
2. Cuando el formulario se envía, el archivo se carga en el destino que especifique, en el camino, el archivo se valida para asegurarse de que se puede cargar según las preferencias establecidas
3. Una vez cargado, el usuario recibe un mensaje de éxito

Para demostrar este proceso aquí hay un breve tutorial, luego, encontrará información de referencia

Creando el formulario de carga

Usando un editor de texto, cree un formulario llamado **upload_form.php,** en él, coloque este código y guárdelo en su directorio **application/views/**:

```
<html>
<head>
<title>Formulario de Upload</title>
</head>
<body>
  <?php echo $error;?>
  <?php echo form_open_multipart('upload/do_upload');?>
  <input type="file" name="userfile" size="20" />
  <br /><br />
  <input type="submit" value="upload" />
</form>
```

Notará que estamos usando **form helper** para crear la etiqueta de apertura de formulario, las cargas de archivos requieren un formulario **"multipart"**, por lo que el helper crea la sintaxis adecuada para usted, también notará que tenemos una variable **error$,** esta se utiliza para poder mostrar los mensajes de error en caso de que el usuario haga algo mal

La página de éxito

Usando un editor de texto, cree un formulario llamado **upload_success.php**, en él, coloque este código y guárdelo en su directorio **application/views/**:

```
<html>
<head>
  <title>Formulario de Upload</title>
</head>
  <body>
    <h3>Su archivo se cargó correctamente!</h3>
      <ul>
        <?php foreach ($upload_data as $item => $value):?>
          <li><?php echo $item;?>: <?php echo $value;?></li>
        <?php endforeach; ?>
      </ul>
    <p><?php echo anchor('subir', 'Subir otro archivo!'); ?></p>
  </body>
</html>
```

El controlador

Usando un editor de texto, cree un controlador llamado **Upload.php**, en él, coloque este código y guárdelo en su directorio **application/controllers/**:

```
<?php

class Upload extends CI_Controller {

  public function __construct()
  {
     parent::__construct();
     $this->load->helper(array('form', 'url'));
  }

  public function index()
  {
     $this->load->view('upload_form', array('error' => ' ' ));
  }

  public function do_upload()
  {
     $config['upload_path']   = './uploads/';
     $config['allowed_types'] = 'gif|jpg|png';
     $config['max_size']      = 100;
     $config['max_width']     = 1024;
     $config['max_height']    = 768;

     $this->load->library('upload', $config);

     if ( ! $this->upload->do_upload('userfile'))
```

```
        {
                $error = array('error' => $this->upload->display_errors());

                $this->load->view('upload_form', $error);
        }
        else
        {
                $datos = array('upload_data' => $this->upload->data());

                $this->load->view('upload_success', $datos);
        }
    }
}
```

El directorio Upload

Necesitará un directorio de destino para sus imágenes cargadas, cree un directorio en la raíz de su instalación de CodeIgniter llamado **uploads** y establezca sus permisos de archivos en 777

Intentelo !

Para probar su formulario, visite su sitio usando una URL similar a esta:

```
ejemplo.com/index.php/upload/
```

Debería ver su formulario de carga, intente cargar un archivo de imagen, ya sea un jpg, gif o png, si la ruta en su controlador es correcta, debería funcionar

Iniciar File Upload Class

File Uploading Class se inicia usando **$this->load->library()**:

```
$this->load->library('upload');
```

Una vez cargada , el objeto esta disponible usando:

```
$this->upload
```

Establecer preferencias

Al igual que en otras bibliotecas, controlará mediante preferencias lo que se permite cargar, en el controlador que ha creado, configure las siguientes preferencias:

```
$config['upload_path']   = './uploads/';
$config['allowed_types'] = 'gif|jpg|png';
$config['max_size']      = '100';
```

```
$config['max_width']     = '1024';
$config['max_height']    = '768';

$this->load->library('upload', $config);

// Alternativamente puede establecer preferencias llamando al método initialize()
// Útil si carga automáticamente la clase:
$this->upload->initialize($config);
```

Las preferencias anteriores deben ser bastante autoexplicativas, a continuación hay una tabla que describe todas las preferencias disponibles

Preferencias disponibles

Las siguientes preferencias están disponibles, el valor predeterminado indica qué se usará si no especifica esa preferencia:

Preferencia	Predeterminado	Opciones	Descripción
upload_path	No	No	La ruta al directorio donde deben guardarse los archivos subidos, en el directorio debe poder escribirse y la ruta puede ser absoluta o relativa
allowed_types	No	No	Los tipos mime correspondientes a los tipos de archivos que permite subir, por lo general, la extensión de archivo se puede utilizar como tipo de mime, puede ser un array o un string separado por barras verticales
file_name	No	Nombre de archivo deseado	Si lo establece CodeIgniter cambiará el nombre del archivo cargado a este nombre, la extensión provista en el nombre del archivo también debe ser un tipo de archivo permitido, si no se proporciona una extensión en el archivo original, se usará el nombre
file_ext_tolower	FALSE	TRUE/FALSE (boolean)	Si se establece en TRUE, la extensión del archivo se verá forzada a minúsculas
overwrite	FALSE	TRUE/FALSE (boolean)	Si se establece en TRUE, si existe un archivo con el mismo nombre que el que está cargando, se sobrescribirá, si se establece en FALSE, se agregará un número al nombre del archivo si existe otro con el mismo nombre
max_size	0	No	El tamaño máximo (en kilobytes) que puede tener el archivo, establecer en cero para sin límite. Nota: la mayoría de las instalaciones de PHP tienen su propio límite, según se especifica en el archivo php.ini, por lo general, 2 MB (o 2048 KB)
max_width	0	No	El ancho máximo (en píxeles) que puede tener la imagen, establecer a cero para sin límite

max_height	0	No	La altura máxima (en píxeles) que puede tener la imagen, establecer a cero para sin límite
min_width	0	No	El ancho mínimo (en píxeles) que puede ser la imagen, establecer a cero para sin límite
min_height	0	No	La altura mínima (en píxeles) que puede tener la imagen, establecer a cero para sin límite
max_filename	0	No	La longitud máxima que puede tener un nombre de archivo, establecer a cero para sin límite
max_filename_increment	100	No	Cuando overwrite se establece en FALSE, establece el máximo incremento que CodeIgniter agregará al nombre del archivo
encrypt_name	FALSE	TRUE/FALSE (boolean)	Si se establece en TRUE, el nombre del archivo se convertirá en un string aleatorio encriptado, puede ser útil, por ejemplo, si quiere que la persona que suba el archivo no pueda discernir su nombre
remove_spaces	TRUE	TRUE/FALSE (boolean)	Si se establece en TRUE, los espacios en el nombre del archivo se convertirán en guiones bajos. **Se recomienda**
detect_mime	TRUE	TRUE/FALSE (boolean)	Si se establece en TRUE, se realizará una detección del tipo de archivo del lado del servidor para evitar los ataques de inyección de código. **NO desactive esta opción** a menos que no tenga otra opción, ya que eso podría causar un riesgo de seguridad
mod_mime_fix	TRUE	TRUE/FALSE (boolean)	Si se establece en TRUE, las extensiones de varios nombres de archivo se agregarán con un guión bajo para evitar la activación de **mod_mime** de Apache. **NO desactive esta opción** si su directorio de carga es público, ya que es un riesgo de seguridad

Establecer preferencias en un archivo de configuración

Si prefiere no establecer preferencias utilizando el método anterior, puede colocarlas en un archivo de configuración, simplemente cree un nuevo archivo llamado **upload.php**, agregue el array **$config** en ese archivo y luego guarde el archivo en: **config/upload.php** y se usará automáticamente. **NO** necesitará utilizar el método **$this->upload->initialize()** si guarda sus preferencias en un archivo de configuración

class CI_Upload

class CI_Upload

class CI_Upload

data()

mixed data([string $index = NULL])

Descripción

Este es un método que devuelve un array que contiene todos los datos relacionados con el archivo que se cargó

Parámetros

index

Elemento a devolver en lugar de la array completa

Valores devueltos

Información sobre el archivo cargado

Aquí está el prototipo del array:

```
Array
(
 [file_name]      => mypic.jpg
 [file_type]      => image/jpeg
 [file_path]      => /path/to/your/upload/
 [full_path]      => /path/to/your/upload/jpg.jpg
 [raw_name]       => mypic
 [orig_name]      => mypic.jpg
 [client_name]    => mypic.jpg
 [file_ext]       => .jpg
 [file_size]      => 22.2
 [is_image]       => 1
 [image_width]    => 800
 [image_height]   => 600
 [image_type]     => jpeg
 [image_size_str] => width="800" height="200"
)
```

Para obtener un elemento del array:

```
$this->upload->data('file_name');   // Devuelve: mypic.jpg
```

Aquí hay una tabla que explica los elementos del array anterior:

Elemento	Descripción
file_name	Nombre del archivo que se cargó, incluida su extensión
file_type	Tipo MIME del archivo MIME
file_path	Ruta absoluta del archivo en el servidor

full_path	Ruta absoluta del servidor, incluido el nombre del archivo
raw_name	Nombre del archivo, sin la extensión
orig_name	Nombre original del archivo. Solo es útil si usa la opción de nombre encriptado
client_name	Nombre de archivo proporcionado por el agente de usuario del cliente, posiblemente sanitizado
file_ext	Extensión del nombre de archivo, punto incluido
file_size	Tamaño del archivo en kilobytes
is_image	Si el archivo es o no una imagen. 1 = imagen. 0 = no
image_width	Ancho de la imagen
image_height	Alto de la imagen
image_type	Tipo de imagen, generalmente la extensión del nombre de archivo sin el punto
image_size_str	Un string que contiene el ancho y alto (útil para ponerlo en una etiqueta de imagen)

display_errors()

string display_errors([string $open = ' ' [, string $close = ' ']])

Descripción

Recupera cualquier mensaje de error si el método **do_upload()** devuelve falso, el método no se repite automáticamente, devuelve los datos para que pueda asignarlos como lo necesite

Parámetros

open
: Apertura de marcado, delimitador de inicio

close
: Marcado de cierre, delimitador de fin

Valores devueltos

Mensaje(s) de error formateado

Formateo de errores

Por defecto, el método anterior ajusta cualquier error dentro de las etiquetas , puede establecer sus propios delimitadores de esta manera:

```
$this->upload->display_errors('<p>', '</p>');
```

do_upload()

bool do_upload([string $field = 'userfile'])

Descripción

Realiza la carga según las preferencias que ha establecido

Parámetros

field

Nombre del campo de formulario

Valores devueltos

TRUE en el éxito, FALSE si falla

> ***Nota***
>
> De forma predeterminada, la rutina de carga espera que el archivo proceda de un campo de formulario denominado **userfile**, y el formulario debe ser del tipo **"multipart"**

```
<form method="post" action="some_action" enctype="multipart/form-data" />
```

Si desea establecer su propio nombre de campo, simplemente pase su valor al método **do_upload()**:

```
$nombre_campo = "algun_nombre_de_campo";
$this->upload->do_upload($nombre_campo);
```

initialize()

object initialize([array $config = array() [, bool $reset = TRUE]])

Descripción

Inicia las preferencias

Parámetros

config

Preferencias

reset

Si restablecer las preferencias (que no se proporcionan en **$config**) a sus valores predeterminados

Valores devueltos

Instancia CI_Upload (method chaining)

Form Validation

CodeIgniter proporciona una validación completa de formularios y una clase de preparación de datos que le ayuda a minimizar la cantidad de código que escribirá

Introducción

Antes de explicar el enfoque de CodeIgniter para la validación de datos, describamos el escenario ideal:

1. Se muestra un formulario
2. Usted lo completa y lo envía
3. Si envió algo no válido, o tal vez perdió un elemento requerido, el formulario se vuelve a mostrar con sus datos junto con un mensaje de error que describe el problema
4. Este proceso continúa hasta que haya enviado un formulario válido

En el extremo receptor, el script debe:

1. Verificar los datos requeridos
2. Verificar que los datos sean del tipo correcto y que cumplan con los criterios correctos, por ejemplo, si se envía un nombre de usuario, debe validarse para que contenga solo los caracteres permitidos, debe tener una longitud mínima y no exceder una longitud máxima, el nombre de usuario no puede ser el nombre de un usuario existente, o tal vez una palabra reservada. Etc.
3. Desinfectar los datos por seguridad
4. Pre formatear los datos si es necesario (Los datos deben ser recortados?, codificado HTML? Etc.)
5. Preparar los datos para la inserción en la Base de datos

Aunque no hay nada terriblemente complejo en el proceso anterior, generalmente requiere una gran cantidad de código, y para mostrar mensajes de error, varias estructuras de control generalmente se colocan dentro del formato HTML. La validación de formularios, aunque simple de crear, generalmente es muy complicada y tediosa de implementar

Tutorial de validación de formularios

Lo que sigue es un tutorial "práctico" para implementar la validación de formularios de CodeIgniter

Para implementar la validación de formularios necesitará tres cosas:

1. Un archivo de vista que contiene un formulario
2. Un archivo de vista que contiene un mensaje de "éxito" que se mostrará al momento del envío exitoso
3. Un método de controlador para recibir y procesar los datos enviados

Vamos a crear esas tres cosas, usando un formulario de registro de miembro como ejemplo

El formulario

Usando un editor de texto, crea un formulario llamado **myform.php**, en él, coloque este código y guárdelo en su directorio **application/views/**:

```
<html>
<head>
  <title>Mi formulario</title>
</head>
  <body>
    <?php echo validation_errors(); ?>
    <?php echo form_open('form'); ?>

     <h5>Usuario</h5>
     <input type="text" name="nom_usuario" value="" size="50" />

     <h5>Contraseña</h5>
     <input type="text" name="contraseña" value="" size="50" />

     <h5>Confirmar contraseña</h5>
     <input type="text" name="conf_contraseña" value="" size="50" />

     <h5>Dirección de Email</h5>
     <input type="text" name="email" value="" size="50" />

     <div><input type="submit" value="Enviar" /></div>
    </form>
  </body>
</html>
```

La página de éxito

Usando un editor de texto, crea un formulario llamado **form_completado.php**, en él, coloque este código y guárdelo en su directorio **application/views/**:

```
<html>
<head>
  <title>Mi formulario</title>
</head>
  <body>
    <h3>Su formulario se envió correctamente!</h3>
    <p><?php echo anchor('formulario', 'Inténtelo de nuevo!'); ?></p>
  </body>
</html>
```

El controlador

Usando un editor de texto, cree un controlador llamado **Form.php**, en él, coloque este código y guárdelo en su directorio **application/controllers/**:

```
<?php

class Form extends CI_Controller {

  public function index()
  {
     $this->load->helper(array('form', 'url'));

     $this->load->library('form_validation');

     if ($this->form_validation->run() == FALSE)
     {
        $this->load->view('myform');
     }
     else
     {
        $this->load->view('form_completado');
     }
  }
}
```

Intentelo!

Para probar su formulario, visite su sitio usando una URL similar a esta:

```
ejemplo.com/index.php/form/
```

Si envía el formulario, simplemente debe ver la recarga del formulario, eso es porque todavía no ha configurado ninguna regla de validación

Como no le ha indicado a la clase **Form Validation** que valide nada, devuelve FALSE de forma predeterminada. el método "**run()**" solo devuelve TRUE si ha aplicado correctamente sus reglas sin que ninguna de ellas haya fallado

Explicación

Notará varias cosas sobre las páginas anteriores

El formulario **myform.php** es un formulario web estándar con un par de excepciones:

1. Utiliza un **form helper** para crear la apertura de formulario, técnicamente, esto no es necesario, puede crear el formulario usando HTML estándar, sin embargo, el beneficio de usar el helper es que genera la URL de acción para usted, en función de la URL en su archivo de configuración, esto hace que su aplicación sea más portable en caso que cambie sus URL
2. En la parte superior del formulario, verás la siguiente llamada a la función:

```
<?php echo validation_errors(); ?>
```

Esta función devolverá cualquier mensaje de error enviado por el validador, si no hay mensajes, devuelve un string vacío

El controlador **Form.php** tiene un método: **index ()**, este método inicializa la clase de validación y carga **form helper** y **URL helper** utilizado por sus archivos de vista, también ejecuta la rutina de validación. En función de si la validación tiene éxito, presenta el formulario o la página de éxito

Establecer reglas de validación

CodeIgniter le permite establecer tantas reglas de validación como necesite para un campo determinado, ponerlas en orden en cascada e incluso le permite preparar y procesar previamente los datos de campo al mismo tiempo

Para establecer las reglas de validación, usará el método **set_rules()**:

```
$this->form_validation->set_rules();
```

El método anterior toma tres parámetros como entrada:

1. El nombre del campo: el nombre exacto que se le asignó al campo de formulario
2. Un nombre **"**humano" para este campo, que se insertará en el mensaje de error, por ejemplo, si su campo se llama "usuario", puede darle un nombre humano de "Nombre de usuario"
3. Las reglas de validación para ese campo de formulario
4. (opcional) Establezca mensajes de error personalizados en cualquier regla dada para el campo actual, si no se proporciona utilizará el predeterminado

Nota

Si desea que el nombre del campo se almacene en un archivo de idioma, consulte la sección **Traducir los nombres de campo**

Aquí hay un ejemplo. En su controlador **Form.php**, agregue este código justo debajo del método de inicialización de validación:

```
$this->form_validation->set_rules('nom_usuario', 'Nombre', 'required');
$this->form_validation->set_rules('contraseña', 'Contraseña', 'required');
$this->form_validation->set_rules('conf_contraseña', 'Confirmar contraseña', 'required');
$this->form_validation->set_rules('email', 'Email', 'required');
```

Su controlador ahora debería verse así:

```
<?php

class Form extends CI_Controller {

  public function index()
  {
     $this->load->helper(array('form', 'url'));

     $this->load->library('form_validation');

     $this->form_validation->set_rules('nom_usuario', 'Usuario', 'required');
     $this->form_validation->set_rules('contraseña', 'Contraseña', 'required',
       array('required' => 'Debe indicar %s.'));

     $this->form_validation->set_rules('conf_contraseña', 'Confirmar contraseña', 'required');
     $this->form_validation->set_rules('email', 'Email', 'required');

     if ($this->form_validation->run() == FALSE)
     {
        $this->load->view('myform');
     }
     else
     {
        $this->load->view('form_completado');
     }
  }
}
```

Ahora envíe el formulario con los campos en blanco y debería ver los mensajes de error, si envía el formulario con todos los campos rellenos, verá su página de éxito

Nota

Los campos de formulario aún no se vuelven a llenar con los datos cuando hay un error. En breve vamos a llegar a eso

Establecer reglas usando un array

Antes de continuar, debe tenerse en cuenta que al método de configuración de reglas se le puede pasar un array para establecer todas las reglas de una sola vez

Si usa este enfoque, debe nombrar sus claves de array como se indica:

```
$config = array(
  array(
    'field' => 'nom_usuario',
    'label' => 'Usuario',
    'rules' => 'required'
  ),
```

```
    array(
      'field' => 'contraseña',
      'label' => 'Contraseña',
      'rules' => 'required',
      'errors' => array(
                    'required' => 'Debe indicar %s.',
                  ),
    ),
   array(
      'field' => 'conf_contraseña',
      'label' => 'Confirmar contraseña',
      'rules' => 'required'
   ),
   array(
      'field' => 'email',
      'label' => 'Email',
      'rules' => 'required'
      )
 );

 $this->form_validation->set_rules($config);
```

Reglas en cascada

CodeIgniter le permite conectar varias reglas juntas, vamos a intentarlo, cambie sus reglas en el tercer parámetro del método de configuración de reglas, como este:

```
$this->form_validation->set_rules(
  'nom_usuario', 'Usuario',
  'required|min_length[5]|max_length[12]|is_unique[users.nom_usuario]',
  array(
     'required'  => 'No ha proporcionado %s.',
     'is_unique' => '%s ya existe.'
  )
);

$this->form_validation->set_rules('contraseña', 'Contraseña', 'required');
$this->form_validation->set_rules('conf_contraseña', 'Confrimar contraseña',
                                  'required|matches[contraseña]');
$this->form_validation->set_rules('email', 'Email',
                                  'required|valid_email|is_unique[users.email]');
```

El código anterior establece las siguientes reglas:

1. El campo Usuario no debe tener menos de 5 caracteres ni más de 12
2. El campo Contraseña debe coincidir con el campo de Confirmación de contraseña
3. El campo Email debe contener una dirección de correo electrónico válida

Dele una oportunidad! , envíe su formulario sin los datos adecuados y verá nuevos mensajes de error que corresponden a sus nuevas reglas, existen numerosas reglas disponibles sobre las que puede leer en la referencia de validación

Nota

También puede pasar un array de reglas a **set_rules()**, en lugar de a un string

Ejemplo:

```
$this->form_validation->set_rules('nom_usuario','Usuario',array('required','min_length[5]'));
```

Preparar los datos

Además del método de validación utilizado anteriormente, también puede preparar sus datos de varias maneras

Por ejemplo, puede configurar reglas como esta:

```
$this->form_validation->set_rules('nom_usuario', 'Usuario',
                                  'trim|required|min_length[5]|max_length[12]');
$this->form_validation->set_rules('contraseña', 'Contraseña', 'trim|required|min_length[8]');
$this->form_validation->set_rules('conf_contraseña', 'Confirmar contraseña',
                                     'trim|required|matches[contraseña]');
$this->form_validation->set_rules('email', 'Email', 'trim|required|valid_email');
```

En el ejemplo anterior, estamos "recortando" los campos, verificando la longitud cuando sea necesario y asegurándonos de que coincidan ambos campos de contraseña

Cualquier función PHP nativa que acepte un parámetro se puede usar como una regla, como **"htmlspecialchars()"**, **"trim ()"**, etc.

Nota

En general, querrá utilizar las funciones de preparación después de las reglas de validación, por lo que si hay un error, los datos originales se mostrarán en el formulario

Volver a llenar el formulario

Hasta ahora solo hemos estado lidiando con errores, es hora de volver a llenar el campo de formulario con los datos enviados

CodeIgniter ofrece varias funciones helper que le permiten hacer esto. La que usará más comúnmente es la contenida en **Form Helper**:

```
set_value('field_name')
```

Abra su archivo de vista **myform.php** y actualice el valor en cada campo usando la función **set_value()**:

No olvide incluir cada nombre de campo en las llamadas a la función php: **"set_value ()"**

```
<html>
<head>
  <title>Mi formulario</title>
</head>
  <body>
    <?php echo validation_errors(); ?>
    <?php echo form_open('form'); ?>

      <h5>Usuario</h5>
      <input type="text" name="nom_usuario" value="<?php echo set_value('nom_usuario'); ?>
                                                  " size="50" />
      <h5>Contraseña</h5>
      <input type="text" name="contraseña" value="<?php echo set_value('contraseña'); ?>
                                                 " size="50" />
      <h5>Confirmar contraseña</h5>
      <input type="text" name="conf_contraseña" value="<?php echo set_value('conf_contraseña'); ?>
                                                      " size="50" />
      <h5>Dirección de Email</h5>
      <input type="text" name="email" value="<?php echo set_value('email'); ?>" size="50" />

      <div><input type="submit" value="Enviar" /></div>
    </form>
  </body>
</html>
```

Ahora recargue su página y envíe el formulario para que se desencadene un error, sus campos de formulario ahora deben llenarse nuevamente

Nota

La sección de referencia de clase siguiente contiene métodos que le permiten volver a llenar menús **<select>,** botones de opción y casillas de verificación

Importante

Si utiliza un array como nombre de campo de formulario, debe proporcionarlo como un array a la función

Ejemplo:

```
<input type="text" name="colors[]" value="<?php echo set_value('colors[]'); ?>" size="50" />
```

Para más información, consulte la sección **Uso de arrays como nombres de campo** a continuación

Callbacks: sus propias métodos de validación

El sistema de validación admite callbacks a sus propios métodos de validación, esto le permite extender la clase de validación para satisfacer sus necesidades, por ejemplo, si necesita ejecutar una consulta de Base de datos para ver si el usuario está eligiendo un nombre de usuario único, puede crear un callback que lo haga, vamos a crear un ejemplo de esto

En su controlador, cambie la regla de **"nom_usuario"** a esto:

```
$this->form_validation->set_rules('nom_usuario', 'Usuario', 'callback_nom_usuario_check');
```

A continuación, agregue un nuevo método llamado **nom_usuario_check()** a su controlador, así es como debería verse ahora su controlador:

```
<?php

class Form extends CI_Controller {

  public function index()
  {
     $this->load->helper(array('form', 'url'));

     $this->load->library('form_validation');

     $this->form_validation->set_rules('nom_usuario', 'Usuario', 'callback_nom_usuario_check');
     $this->form_validation->set_rules('contraseña', 'Contraseña', 'required');
     $this->form_validation->set_rules('conf_contraseña', 'Confirmar contraseña', 'required');
     $this->form_validation->set_rules('email', 'Email', 'required|is_unique[users.email]');

     if ($this->form_validation->run() == FALSE)
     {
        $this->load->view('myform');
     }
     else
     {
        $this->load->view('form_completado');
     }
   }

  public function nom_usuario_check($str)
  {
     if ($str == 'test')
     {
        $this->form_validation->set_message('nom_usuario_check', 'El campo {field} no
                                              puede ser la palabra "test"');
        return FALSE;
     }
     else
     {
        return TRUE;
```

```
        }
    }
}
```

Recargue su formulario y envíelo con la palabra **"test"** como nombre de usuario, puede ver que los datos del campo de formulario se pasaron a su método callback para que usted los procese

Para invocar una callback simplemente ponga el nombre del método en una regla, con **"callback_"** como el prefijo de la regla, si necesita recibir un parámetro adicional en su método callback, simplemente agréguelo normalmente después del nombre del método entre corchetes, como en: **callback_foo[bar]**, luego se pasará como el segundo argumento de su método callback

Nota

También puede procesar los datos del formulario que se pasan a su callback y devolverlos, si su callback devuelve algo que no sea un booleano TRUE/FALSE, se asume que los datos son los datos de formulario recién procesados

Callable: use cualquier cosa como una regla

Si las callbacks no son lo suficientemente buenas para usted, por ejemplo, porque están limitadas a su controlador, no se desilusione, hay más de una forma de crear reglas personalizadas: cualquier cosa que sea **is_callable()** y devuelva **TRUE**

Considere el siguiente ejemplo:

```
$this->form_validation->set_rules(
  'nom_usuario', 'Usuario',
  array(
   'required',
   array($this->users_model, 'usuario_correcto')
  )
);
```

El código anterior usaría el método **usuario_correcto()** de su objeto **Users_model**

Esto es solo un ejemplo, por supuesto, y las callbacks no están limitadas a los modelos, puede usar cualquier objeto/método que acepte el valor del campo como su '**primer parámetro**'

También puedes usar una función anónima:

```
$this->form_validation->set_rules(
  'nom_usuario', 'Usuario',
  array(
    'required',
    function($value)
    {
       // Verificar $value
    }
 )
);
```

Por supuesto, dado que una regla callable por sí sola no es un string, tampoco es un nombre de regla, ese es un problema cuando desea establecer mensajes de error, para evitar ese problema, puede poner tales reglas como el segundo elemento de un array, siendo el primero el nombre de la regla:

```
$this->form_validation->set_rules(
  'nom_usuario', 'Usuario',
  array(
    'required',
    array('nom_usuario_callable', array($this->users_model, 'usuario_correcto'))
  )
);
```

Versión de función anónima:

```
$this->form_validation->set_rules(
  'nom_usuario', 'Usuario',
  array(
    'required',
    array(
       'nom_usuario_callable',
       function($str)
       {
          // Verifica si $str es válido y devuelve TRUE o FALSE
       }
    )
  )
);
```

Configuración de mensajes de error

Todos los mensajes de error nativos se encuentran en el siguiente archivo de idioma:
system/language/english/form_validation_lang.php

Para establecer su propio mensaje personalizado global para una regla, puede extender/escribir el archivo de idioma creando el suyo propio en **application/language/english/form_validation_lang.php** (lea más sobre esto en la documentación de **Language Class**), o use el siguiente método:

```
$this->form_validation->set_message('regla', 'Mensaje de error');
```

Si necesita establecer un mensaje de error personalizado para un campo en particular en alguna regla particular, use el método **set_rules()**:

```
$this->form_validation->set_rules('nombre_campo', 'Etiqueta_campo', 'regla1|regla2|regla3',
     array('regla2' => 'Mensaje de error en rule2 para este nombre_campo')
);
```

Donde la regla corresponde al nombre de una regla en particular, y el mensaje de error es el texto que desea mostrar

Si desea incluir el nombre **"humano"** de un campo, o el parámetro opcional que algunas reglas permiten (como **max_length**), puede agregar las etiquetas **{field}** y **{param}** a su mensaje, respectivamente:

```
$this->form_validation->set_message('min_length', '{field} debe tener al menos {param} caracteres.');
```

En un campo con el nombre humano **Usuario** y una regla de **min_length[5],** se mostrará un error: **"El Usuario debe tener al menos 5 caracteres"**

Nota

El antiguo método **sprintf()** y usar **%s** en los mensajes de error seguirá funcionando, sin embargo, anulará las etiquetas anteriores, debe usar uno u otro

En el ejemplo anterior de la callback, el mensaje de error se configuró al pasar el nombre del método (sin el prefijo **"callback_"**):

```
$this->form_validation->set_message('nom_usuario_check')
```

Traducir los nombres de campo

Si desea almacenar el nombre **"humano"** que pasó al método **set_rules()** en un archivo de idioma y, por lo tanto, hacer que el nombre se pueda traducir, debe hacer lo siguiente:

Primero, prefije su nombre **"humano"** con lang, como en este ejemplo:

```
$this->form_validation->set_rules('segundo_nombre', 'lang:segundo_nombre', 'required');
```

Luego, almacene el nombre en una de sus array de archivos de idioma (sin el prefijo):

```
$lang['segundo_nombre'] = 'Apellido';
```

Nota

El antiguo método **sprintf()** y usar **%s** en los mensajes de error seguirá funcionando, sin embargo, anulará las etiquetas anteriores, debe usar uno u otro

Si almacena su elemento de array en un archivo de idioma que CodeIgniter no carga automáticamente, deberá recordar cargarlo en su controlador utilizando:

```
$this->lang->load('nombre_de_archivo');
```

Consulte la referencia de **Language Class** para obtener más información sobre los archivos de idioma

Cambiar los delimitadores de error

De forma predeterminada, la clase Form Validation agrega una etiqueta de párrafo **(<p>)** alrededor de cada mensaje de error que se muestra, puede cambiar estos delimitadores globalmente, individualmente o cambiar los valores predeterminados con un archivo de configuración

1. Cambiar los delimitadores globalmente

Para cambiar globalmente los delimitadores de error, en su método de controlador, justo después de cargar la clase **Form Validation**, agregue esto:

```
$this->form_validation->set_error_delimiters('<div class="error">', '</div>');
```

En este ejemplo, hemos cambiado los delimitadores a la etiqueta **div**

2. Cambiar los delimitadores individualmente

A cada una de las dos funciones de generación de errores que se muestran en este tutorial le puede suministrar sus propios delimitadores de la siguiente manera:

```
<?php echo form_error('nombre_de_campo', '<div class="error">', '</div>'); ?

o:

<?php echo validation_errors('<div class="error">', '</div>'); ?>
```

3. Establecer delimitadores en un archivo de configuración

Puede agregar los delimitadores de error en **application/config/form_validation.php** de la siguiente manera:

```
$config['error_prefix'] = '<div class="error_prefix">';
$config['error_suffix'] = '</div>';
```

Mostrar errores individualmente

Si prefiere mostrar un mensaje de error al lado de cada campo de formulario, en lugar de hacerlo como una lista, puede usar la función **form_error()**

Intentelo !

Cambie su formulario para que se vea así:

```
<h5>Usuario</h5>
<?php echo form_error('nom_usuario'); ?>
<input type="text" name="nom_usuario" value="<?php echo set_value('nom_usuario'); ?>" size="50" />

<h5>Contraseña</h5>
<?php echo form_error('contraseña'); ?>
<input type="text" name="contraseña" value="<?php echo set_value('contraseña'); ?>" size="50" />

<h5>Confirme contraseña</h5>
<?php echo form_error('conf_contraseña'); ?>
<input type="text" name="conf_contraseña" value="<?php echo set_value('conf_contraseña'); ?>" size="50" />

<h5>Dirección de Email</h5>
<?php echo form_error('email'); ?>
<input type="text" name="email" value="<?php echo set_value('email'); ?>" size="50" />
```

Si no hay errores, no se mostrará nada, si hay un error, aparecerá el mensaje

Importante

Si utiliza un array como nombre de un campo de formulario, debe proporcionarla como un array a la función

Ejemplo:

```
<?php echo form_error('options[size]'); ?>
<input type="text" name="options[size]" value="<?php echo set_value("options[size]");?>" size="50" />
```

Para obtener más información, consulte la sección **Uso de matrices como nombres de campo** a continuación

Validar un array (que no sea $_POST)

En ocasiones, es posible que desee validar un array que no se origine a partir de los datos **$_POST**

En ese caso, puede especificar el array que se validará:

```
$datos = array(
  'nom_usuario'     => 'H.P. Lovecraft',
  'contraseña'      => 'Cthulhu',
  'conf_contraseña' => 'Cthulhu'
);
$this->form_validation->set_data($datos);
```

La creación de reglas de validación, la ejecución de la validación y la recuperación de mensajes de error funcionan igual ya sea que esté validando datos **$_POST** u otro array de su elección

Importante

- Debe llamar al método **set_data()** antes de definir cualquier regla de validación
- Si desea validar más de una array durante una sola ejecución, debe llamar al método **reset_validation()** antes de configurar reglas y validar el nuevo array

Para obtener más información, consulte la sección de referencia de clase a continuación

Guardar reglas de validación en un archivo de configuración

Una buena característica de la clase **Form Validation** es que le permite almacenar para toda su aplicación, todas sus reglas de validación en un archivo de configuración, puede organizar estas reglas en "**grupos**", estos grupos pueden cargarse automáticamente cuando se llama a un controlador/método, o puede llamar cada grupo manualmente según lo necesite

Cómo guardar sus reglas

Para almacenar sus reglas de validación, cree un archivo llamado **form_validation.php** en su directorio **application/config/**, en ese archivo, coloque un array llamado **$config** con sus reglas

Como se mostró anteriormente, el array de validación tiene este prototipo:

```
$config = array(
 array(
    'field' => 'nom_usuario',
    'label' => 'Usuario',
    'rules' => 'required'
 ),
 array(
    'field' => 'contraseña',
    'label' => 'Contraseña',
    'rules' => 'required'
 ),
 array(
    'field' => 'conf_contraseña',
    'label' => 'Confirmar contraseña',
    'rules' => 'required'
 ),
 array(
    'field' => 'email',
    'label' => 'Email',
    'rules' => 'required'
 )
);
```

Su archivo de reglas de validación se carga automáticamente y se usará cuando llame al método **run()**

Tenga en cuenta que el array **TIENE que llamarse $config**

Crear un grupo de reglas

Para organizar sus reglas en **"grupos"**, debe colocarlos en **"sub-arrays"**, considere el ejemplo siguiente, que muestra dos grupos de reglas, hemos llamado arbitrariamente estas dos reglas **"registro"** y **"email"**

Puede nombrar sus reglas como desee:

```
$config = array(
    'registro' => array(
        array(
            'field'  => 'nom_usuario',
            'label' => 'Usuario',
            'rules' => 'required'
        ),
        array(
            'field'  => 'contraseña',
            'label' => 'Contraseña',
            'rules' => 'required'
        ),
        array(
            'field'  => 'conf_contraseña',
            'label' => 'Confirmar contraseña',
            'rules' => 'required'
        ),
        array(
            'field'  => 'email',
            'label' => 'Email',
            'rules' => 'required'
        )
    ),
    'email' => array(
        array(
            'field'  => 'emailaddress',
            'label' => 'Dirección de Email',
            'rules' => 'required|valid_email'
        ),
        array(
            'field'  => 'nombre',
            'label' => 'Nombre',
            'rules' => 'required|alpha'
        ),
        array(
            'field'  => 'asunto',
            'label' => 'Asunto',
            'rules' => 'required'
```

```
        ),
        array(
            'field'  => 'mensaje',
            'label' => 'Mensaje',
            'rules' => 'required'
        )
    )
);
```

Llamar a un grupo de reglas específicas

Para llamar a un grupo específico, pasará su nombre al método **run()**

Por ejemplo, para llamar a la regla de registro, usted hará esto:

```
if ($this->form_validation->run('registro') == FALSE)
{
    $this->load->view('myform');
}
else
{
    $this->load->view('form_completado');
}
```

Asociar un método de controlador con un grupo de reglas

Un método alternativo, y más automático, para llamar a un grupo de reglas es nombrarlo de acuerdo con la clase/método del controlador con el que pretende usarlo, por ejemplo, supongamos que tiene un controlador llamado **Cliente** y un método llamado **registro**

Así es como se verá su clase:

```
<?php

class Cliente extends CI_Controller {

  public function registro()
  {
     $this->load->library('form_validation');
if ($this->form_validation->run() == FALSE)
     {
        $this->load->view('mi_form');
     }
```

```
        else
        {
            $this->load->view('form_completado');
        }
    }
}
```

En su archivo de configuración de validación, nombre **cliente/registro** a su grupo de reglas:

```
$config = array(
    'cliente/registro' => array(
        array(
            'field'  => 'nom_usuario',
            'label' => 'Usuario',
            'rules' => 'required'
        ),
        array(
            'field'  => 'contraseña',
            'label' => 'Contraseña',
            'rules' => 'required'
        ),
        array(
            'field'  => 'conf_contraseña',
            'label' => 'Confirmar contraseña',
            'rules' => 'required'
        ),
        array(
            'field'  => 'email',
            'label' => 'Email',
            'rules' => 'required'
        )
    )
);
```

Cuando un grupo de reglas se nombra de forma idéntica a una clase/método de controlador, se usará automáticamente cuando se invoque el método **run()** desde esa clase/método

Usar arrays como nombres de campo

La clase **Form Validation** admite el uso de arrays como nombres de campo
Considere este ejemplo:

```
<input type="text" name="opciones[]" value="" size="50" />
```

Si usa un array como nombre de campo, debe usar el nombre **EXACTO** del array en las **Helper Functions** que requieren el nombre del campo y como nombre de campo de la regla de validación

Por ejemplo, para establecer una regla para el campo de arriba usaría:

```
$this->form_validation->set_rules('opciones[]', 'Ociones', 'required');
```

O, para mostrar un error para el campo de arriba, usaría:

```
<?php echo form_error('opciones[]'); ?>
```

O para volver a llenar el campo usaría:

```
<input type="text" name="ociones[]" value="<?php echo set_value('opciones[]'); ?>" size="50" />
```

También puede usar arrays multidimensionales como nombres de campo

Por ejemplo:

```
<input type="text" name="opciones[size]" value="" size="50" />
```

O incluso:

```
<input type="text" name="deportes[nba][baloncesto]" value="" size="50" />
```

Al igual que con nuestro primer ejemplo, debe usar el nombre exacto del array en las funciones helper:

```
<?php echo form_error('deportes[nba][baloncesto]'); ?>
```

Si está usando casillas de verificación, u otros elementos, que tienen múltiples opciones, no olvide dejar un corchete vacío después de cada opción, de modo que todas las selecciones se agregarán al array **POST**:

```
<input type="checkbox" name="opciones[]" value="rojo" />
<input type="checkbox" name="opciones[]" value="azul" />
<input type="checkbox" name="opciones[]" value="verde" />
```

O si usa un array multidimensional:

```
<input type="checkbox" name="opciones[color][]" value="rojo" />
<input type="checkbox" name="opciones[color][]" value="azul" />
<input type="checkbox" name="opciones[color][]" value="verde" />
```

También incluirá corchetes cuando use una función helper:

```
<?php echo form_error('opciones[color][]'); ?>
```

Referencia de reglas

La siguiente es una lista de las reglas que están disponibles para usarse, y que devuelven FALSE si no se cumple :

Regla	Parámetro	Devuelve FALSE	Ejemplo
required	No	Si el elemento del formulario está vacío	
matches	Si	Si el elemento del formulario no coincide con el del parámetro	matches[form_item]
regex_match	Si	Si el elemento del formulario no coincide con la expresión regular	regex_match [/ regex /]
differs	Si	Si el elemento del formulario no difiere del que está en el parámetro	differs [form_item]
is_unique	Si	Si el elemento del formulario no es exclusivo de la tabla y el nombre del campo en el parámetro Nota: Esta regla para funcionar, requiere que **Query Builder** esté habilitado.	is_unique [table.field]
min_length	Si	Si el elemento del formulario es más corto que el valor del parámetro	min_length [3]
max_length	Si	Si el elemento del formulario es más largo que el valor del parámetro	max_length [12]
exact_length	Si	Si el elemento de formulario no es exactamente el valor del parámetro.	exact_length [8]
greater_than	Si	Si el elemento del formulario es menor o igual que el valor del parámetro o no es numérico	greater_than [8]
greater_than_equal_to	Si	Si el elemento del formulario es menor que el valor del parámetro, o no es numérico	greater_than_equal_to [8]
less_than	Si	Si el elemento del formulario es mayor o igual que el valor del parámetro o no es numérico	less_than [8]
less_than_equal_to	Si	Si el elemento de formulario es mayor que el valor del parámetro, o no es numérico	less_than_equal_to [8]
in_list	Si	Si el elemento del formulario no está dentro de una lista predeterminada	in_list [rojo, azul, verde]
alpha	No	Si el elemento del formulario contiene algo más que caracteres alfabéticos	
alpha_numeric	No	Si el elemento del formulario contiene algo más que caracteres alfanuméricos	
alpha_numeric_spaces	No	Si el elemento del formulario contiene algo que no sean caracteres alfanuméricos o espacios, debería usarse después del recorte para evitar espacios al principio o al final	
alpha_dash	No	Si el elemento del formulario contiene algo más que caracteres alfanuméricos, guiones bajos o guiones	
numeric	No	Si el elemento del formulario contiene algo más que caracteres numéricos	
integer	No	Si el elemento del formulario contiene algo que no sea un número entero	

decimal	No	Si el elemento del formulario contiene algo que no sea un número decimal	
is_natural	No	Si el elemento del formulario contiene algo que no sea un número natural: 0, 1, 2, 3, etc.	
is_natural_no_zero	No	Si el elemento del formulario contiene algo más que un número natural, pero no cero: 1, 2, 3, etc.	
valid_url	No	Si el elemento del formulario no contiene una URL válida	
valid_email	No	Si el elemento del formulario no contiene una dirección de correo electrónico válida	
valid_emails	No	Si el valor proporcionado en una lista separada por comas no es un correo electrónico válido	
valid_ip	Si	Si la dirección IP suministrada no es válida. Acepta un parámetro opcional de 'ipv4' o 'ipv6' para especificar un formato de IP	
valid_base64	No	Si la string suministrada contiene algo más que caracteres válidos de Base64	

Nota

- Estas reglas también se pueden llamar como métodos discretos. Por ejemplo:

```
$this->form_validation->required($string);
```

- También puede usar cualquier función nativa de PHP que permita hasta dos parámetros, donde al menos se requiere uno (para pasar los datos del campo).

Referencia de preparaciones

La siguiente es una lista de todos los métodos de preparación que están disponibles para su uso:

Nombre	Parámetro	Descripción
prep_for_form	No	**DESACONSEJADO**: Convierte caracteres especiales para que los datos HTML se puedan mostrar en un campo de formulario sin romperlo
prep_url	No	Agrega "http: //" a la URL si falta
strip_image_tags	No	Quita el HTML de las etiquetas de imagen. deja la URL sin formato
encode_php_tags	No	Convierte etiquetas PHP a entidades

Nota

También puede usar cualquier función PHP nativa que permita un parámetro, como **trim()**, **htmlspecialchars()**, **urldecode()**, etc.

class CI_Form_validation

class CI_Form_validation

class CI_Form_validation

error()

string error(string $field [, string $prefix = '' [, string $suffix = '']])

Descripción

Devuelve el mensaje de error para un campo específico, opcionalmente agregando un prefijo y/o sufijo, generalmente etiquetas HTML

Parámetros

field

Nombre del campo

prefix

Prefijo opcional

suffix

Sufijo opcional

Valores devueltos

Cadena de mensaje de error

error_array()

array error_array()

Descripción

Devuelve los mensajes de error como un array

Valores devueltos

Array de mensajes de error

error_string()

string error_string([string $prefix = '' [, string $suffix = '']])

Descripción

Devuelve todos los mensajes de error ,como se devuelve desde **error_array()**, formateados como un string y separados por un carácter de nueva línea

Parámetros

prefix

Prefijo del mensaje de error

suffix

Sufijo del mensaje de error

Valores devueltos

Mensajes de error como un string

has_rule()

```
bool has_rule(string $field)
```

Descripción

Verifica si hay una regla establecida para el campo especificado

Parámetros

field
Nombre del campo

Valores devueltos

TRUE si el campo tiene reglas establecidas, FALSE si no

reset_validation()

```
object reset_validation()
```

Descripción

Le permite restablecer la validación cuando valida más de un array, este método debe invocarse antes de validar cada nuevo array

Valores devueltos

Instancia CI_Form_validation (method chaining)

run()

```
bool run([ string $group = ''])
```

Descripción
Ejecuta las rutinas de validación, opcionalmente, puede pasar el nombre del grupo de validación a través del método, como se describe en la sección **Guardar conjuntos de reglas de validación en un archivo de configuración**

Parámetros

group
El nombre del grupo de validación para ejecutar

Valores devueltos

TRUE en caso de éxito, FALSE si falló la validación

set_data()

```
object set_data(array $data)
```

Descripción

Le permite establecer un array para la validación, en lugar de usar el array **$_POST** predeterminado

Parámetros

data

Array de datos de validación

Valores devueltos

Instancia CI_Form_validation (method chaining)

set_error_delimiters()

object set_error_delimiters([string $prefix = ' ' [, string $suffix = ' ']])

Descripción

Establece el prefijo y el sufijo por defecto para los mensajes de error

Parámetros

prefix

Prefijo del mensaje de error

suffix

Sufijo del mensaje de error

Valores devueltos

Instancia CI_Form_validation (method chaining)

set_message()

object set_message(string $lang [, string $val = ''])

Descripción

Le permite establecer mensajes de error personalizados, consulte **Configuración de mensajes de error**

Parámetros

lang

La regla para la cual es el mensaje

val

El mensaje

Valores devueltos

Instancia CI_Form_validation (method chaining)

set_rules()

object set_rules(string $field [, string $label = '' [, mixed $rules = ' ' [, array $errors = array()]]])

Descripción

Le permite establecer reglas de validación, como se describe en las secciones:

- **Establecer reglas de validación**
- **Guardar grupos de reglas de validación en un archivo de configuración**

Parámetros

field

Nombre del campo

label

Etiqueta de campo

rules

Reglas de validación, como una lista de strings separadas por una tubería "|", o como un array con reglas

errors

Una lista de mensajes de error personalizados

Valores devueltos

Instancia CI_Form_validation (method chaining)

Helper Reference

Consulte **Form Helper** para las siguientes funciones:

form_error()
validation_errors()
set_value()
set_select()
set_checkbox()
set_radio()

Tenga en cuenta que se trata de **funciones** de procedimiento, por lo que **no requieren** que las anteponga con : **$this->form_validation**

FTP Class

FTP Class permite que los archivos se transfieran a un servidor remoto, los archivos remotos también se pueden mover, cambiar de nombre y eliminar, la clase FTP también incluye una función de "**duplicación**" que permite recrear de manera remota un directorio local completo a través de FTP

Nota

Los protocolos FTP SFTP y SSL no son compatibles, solo FTP estándar

Iniciar FTP Class

FTP Class se inicia con **$this->load->library()**:

```
$this->load->library('ftp');
```

Una vez cargada, el objeto esta disponible con:

```
$this->ftp
```

Ejemplos de uso

En este ejemplo, se abre una conexión al servidor FTP, se lee y carga un archivo local en modo ASCII, los permisos de archivos están establecidos en 755

```
$this->load->library('ftp');

$config['hostname'] = 'ftp.ejemplo.com';
$config['username'] = 'su_nombre_de_usuario';
$config['password'] = 'su_contraseña';
$config['debug']    =  TRUE;

$this->ftp->connect($config);

$this->ftp->upload('/local/path/a/mi_archivo.html', '/publico_html/mi_archivo.html', 'ascii', 0775);

$this->ftp->close();
```

En este ejemplo, se recupera una lista de archivos del servidor:

```
$this->load->library('ftp');

$config['hostname'] = 'ftp.ejemplo.com';
$config['username'] = 'su_nombre_de_usuario';
$config['password'] = 'su_contraseña';
$config['debug']    =  TRUE;
```

```
$this->ftp->connect($config);

$lista = $this->ftp->list_files('/publico_html/');

print_r($lista);

$this->ftp->close();
```

En este ejemplo, un directorio local se duplica en el servidor:

```
$this->load->library('ftp');

$config['hostname'] = 'ftp.ejemplo.com';
$config['username'] = 'su_nombre_de_usuario';
$config['password'] = 'su_contraseña';
$config['debug']    =  TRUE;

$this->ftp->connect($config);

$this->ftp->mirror('/path/a/mi_carpeta/', '/publico_html/mi_carpeta/');

$this->ftp->close();
```

class CI_FTP

class CI_FTP

class CI_FTP

changedir()

bool changedir(string $path [, bool $suppress_debug = FALSE])

Descripción

Cambia el directorio de trabajo actual a la ruta especificada

Parámetros

path

Ruta del directorio

suppress_debug

Si se desactivan los mensajes de depuración para este comando, es útil en caso de que desee utilizar este método como alternativa a **is_dir()** para FTP

Valores devueltos

TRUE en el éxito, FALSE si falla

chmod()

bool chmod(string $path, int $perm)

Descripción

Le permite configurar permisos de archivos, proporcione la ruta al archivo o directorio en el que desea modificar los permisos

Parámetros

path

Ruta para modificar los permisos para

perm

Permisos (octal)

Valores devueltos

TRUE en el éxito, FALSE si falla

Ejemplo

```
// Chmod "jamón_serrano" a 755
$this->ftp->chmod('/publico_html/foo/jamón_serrano/', 0755);
```

close()

bool close()

Descripción

Cierra la conexión a su servidor, se recomienda que use este método cuando termine de cargar

Valores devueltos

TRUE en caso de éxito, FALSE en caso de error

connect()

bool connect([array $config = array()])

Descripción

Se conecta e inicia sesión en el servidor FTP, las preferencias de conexión se establecen pasando un array al método, o puede almacenarlas en un archivo de configuración

Parámetros

config

Valores de conexión

Valores devueltos

TRUE en el éxito, FALSE si falla

Aquí hay un ejemplo que muestra cómo configurar las preferencias de forma manual:

```
$this->load->library('ftp');

$config['hostname'] = 'ftp.ejemplo.com';
$config['username'] = 'su_nombre_de_usuario';
$config['password'] = 'su_contraseña';
$config['port']     = 21;
$config['passive']  = FALSE;
$config['debug']    = TRUE;

$this->ftp->connect($config);
```

Preferencias de FTP en un archivo de configuración

Si lo prefiere, puede almacenar sus preferencias de FTP en un archivo de configuración, simplemente cree un nuevo archivo llamado **ftp.php**, agregue el array **$config** en ese archivo, luego guarde el archivo en **application/config/ftp.php** y se usará automáticamente

Opciones de conexión disponibles

Opción	Predeterminado	Descripción
hostname	n/a	Nombre de host FTP (generalmente algo como: ftp.ejemplo.com)
username	n/a	Nombre de usuario FTP
password	n/a	Contraseña FTP
port	21	FTP Número de puerto del servidor FTP
debug	FALSE	TRUE/FALSE (boolean): Si habilita la depuración para mostrar los mensajes de error
passive	TRUE	TRUE/FALSE (boolean): Si se debe usar el modo pasivo

delete_dir()

bool delete_dir(string $filepath)

Descripción

Permite eliminar un directorio y todo lo que contiene, proporcione la ruta de origen al directorio con una barra inclinada

Parámetros

filepath

Ruta al directorio para eliminar

Valores devueltos

TRUE en el éxito, FALSE si falla

Importante

MUCHO CUIDADO con este método !

Se eliminará de forma recursiva todo dentro de la ruta de acceso, incluidas las subcarpetas y todos los archivos

Asegúrese de que su ruta sea correcta. Intente usar primero **list_files()** para verificar que la ruta sea la correcta

Ejemplo

```
$this->ftp->delete_dir('/publico_html/path/a/directorio/');
```

delete_file()

bool delete_file(string $filepath)

Descripción

Permite eliminar un archivo

Parámetros

filepath
: Ruta al archivo para eliminar

Valores devueltos

TRUE en el éxito, FALSE si falla

Ejemplo

```
// Borra el archivo gasoil.html del directorio 'combustibles'

$this->ftp->delete_file('/publico_html/combustibles/gasoil.html');
```

download()

bool download(string $rempath, string $locpath [, string $mode = 'auto'])

Descripción

Descarga un archivo de su servidor, debe proporcionar la ruta remota y la ruta local, y opcionalmente puede establecer el modo

Parámetros

rempath
: Ruta del archivo remoto

locpath
: Ruta del archivo local

mode
: Modo FTP, por defecto es 'auto' (las opciones son: 'auto', 'binary', 'ascii'), si utiliza el modo 'auto' basará el modo en la extensión de archivo del archivo de origen

Valores devueltos

TRUE en el éxito, FALSE si falla, devuelve FALSE si la descarga no se ejecuta correctamente, incluso si PHP no tiene permiso para escribir el archivo local

Ejemplo

```
$this->ftp->download('/publico_html/mi_archivo.html', '/local/path/a/mi_archivo.html', 'ascii');
```

list_files()

array list_files([string $path = '.'])

Descripción

Le permite recuperar una lista de archivos en su servidor devuelta como un array, debe proporcionar la ruta al directorio deseado

Parámetros

path
Ruta del directorio

Valores devueltos

Un array con la lista de archivos o FALSE en caso de error

Ejemplo

```
$list = $this->ftp->list_files('/publico_html/');
print_r($list);
```

mirror()

bool mirror(string $locpath, string $rempath)

Descripción

Lee de manera recursiva una carpeta local y todo lo que contiene, incluidas las subcarpetas, y crea una réplica mediante FTP en función de ella, cualquiera que sea la estructura del directorio de la ruta original del archivo, se recreará en el servidor, debe proporcionar una ruta de origen y una ruta de destino

Parámetros

locpath
Ruta local
rempath
Ruta remota

Valores devueltos

TRUE en el éxito, FALSE si falla

Ejemplo

```
$this->ftp->mirror('/path/a/mi_directorio/', '/publico_html/mi_directorio/');
```

mkdir()

bool mkdir(string $path [, int $permissions = NULL])

Descripción

Le permite crear un directorio en su servidor, proporcione la ruta que termina en el nombre de la carpeta que desea crear, con una barra al final, los permisos pueden establecerse pasando un valor octal en el segundo parámetro

Parámetros

path

Ruta al directorio para crear

permissions

Permisos (octal)

Valores devueltos

TRUE en el éxito, FALSE si falla

Ejemplo

```
// Crea una carpeta llamada 'chavo'
$this->ftp->mkdir('/publico_html/foo/chavo/', 0755);
```

move()

bool move(string $old_file, string $new_file)

Descripción

Le permite mover un archivo

Parámetros

old_file

Antiguo nombre de archivo

new_file

Nuevo nombre de archivo

Valores devueltos

TRUE en el éxito, FALSE si falla

Ejemplo

```
// Mueve gasoil.html del directorio "bueno" a "malo"
$this->ftp->move('/publico_html/bueno/gasoil.html', '/publico_html/malo/gasoil.html');
```

Nota

Si el nombre del archivo de destino es diferente, se cambiará el nombre del archivo

rename()

bool rename(string $old_file, string $new_file [, bool $move = FALSE])

Descripción

Le permite cambiar el nombre de un archivo, proporcione el nombre/ruta del archivo de origen y el nuevo nombre/ruta de archivo

Parámetros

old_file
Antiguo nombre de archivo

new_file
Nuevo nombre de archivo

move
Si se está realizando un movimiento

Valores devueltos

TRUE en el éxito, FALSE si falla

Ejemplo

```
// Cambia el nombre de verde.html a azul.html
$this->ftp->rename('/publico_html/foo/verde.html', '/publico_html/foo/azul.html');
```

upload()

bool upload(string $locpath, string $rempath [, string $mode = 'auto' [, int $permissions = NULL]])

Descripción

Carga un archivo a su servidor, debe proporcionar la ruta local y la ruta remota, y opcionalmente puede establecer el modo y los permisos

Parámetros

locpath
Ruta del archivo local

rempath
Ruta del archivo remoto

mode
Modo FTP, por defecto es 'auto' (las opciones son: 'auto', 'binario', 'ascii'), si se utiliza el modo 'auto', automático, basará el modo en la extensión de archivo del archivo de origen.

permissions
Permisos de archivos, si se establece, los permisos deben pasarse como un valor octal

Valores devueltos

TRUE en el éxito, FALSE si falla

Ejemplo

```
$this->ftp->upload('/local/path/a/mi_archivo.html','/publico_html/mi_archivo.html','ascii', 0775);
```

Image Manipulation Class

La clase de manipulación de imágenes de CodeIgniter le permite realizar las siguientes acciones:

- Cambio de tamaño de la imagen
- Creación de miniaturas
- Recorte de imagen
- Rotación de imagen
- Imagen de marca de agua

Se admiten las tres principales librerías de imágenes: GD/GD2, NetPBM e ImageMagick

Nota
La marca de agua solo está disponible con la biblioteca GD/GD2, además, aunque se admiten otras librerías, para que el script pueda calcular las propiedades de la imagen requiere GD . Sin embargo, el procesamiento de la imagen se realizará con la librería que especifique

Iniciar Image Manipulation Class

Image Manipulation Class se inicia con **$this->load->library()**:

```
$this->load->library('image_lib');
```

Una vez cargada, el objeto esta disponible con:

```
$this->image_lib
```

Procesar una imagen

Independientemente del tipo de procesamiento que desee realizar (cambio de tamaño, recorte, rotación o marca de agua), el proceso general es idéntico, establecerá algunas preferencias correspondientes a la acción que pretende realizar y luego llamará a una de las cuatro funciones de procesamiento disponibles

Por ejemplo, para crear una miniatura de imagen, hará esto:

```
$config['image_library']   = 'gd2';
$config['source_image']    = '/path/a/imagen/mi_imagen.jpg';
$config['create_thumb']    = TRUE;
$config['maintain_ratio']  = TRUE;
$config['width']           = 75;
$config['height']          = 50;

$this->load->library('image_lib', $config);

$this->image_lib->resize();
```

El código anterior le dice a l método **image_resize** que busque una imagen llamada **mi_imagen.jpg** ubicada en la carpeta indicada en **source_image**, luego crea una miniatura de **75 X 50** píxeles usando la **GD2 image_library,** dado que la opción **maintain_ratio** está habilitada, la miniatura se creará lo más aproximada al ancho y alto indicado, conservando la relación de aspecto original; la miniatura se llamará **mi_imagen_thumb.jpg** y se ubicará en el mismo nivel indicado en S**ource_image**

Nota

Para que la clase **Image Manipulation** pueda procesar, el directorio que contiene los archivos debe tener permisos de escritura

Nota

El procesamiento de imágenes puede requerir una cantidad considerable de memoria del servidor para algunas operaciones, si experimenta errores de memoria mientras procesa imágenes, puede que necesite limitar su tamaño máximo y/o ajustar los límites de memoria de PHP

Métodos de procesamiento

Hay cuatro métodos de procesamiento disponibles:

- **$this->image_lib->resize()**
- **$this->image_lib->crop()**
- **$this->image_lib->rotate()**
- **$this->image_lib->watermark()**

Estos métodos devuelven el valor booleano TRUE si tiene éxito y FALSE si fracasan, si fallan, puede recuperar el mensaje de error usando este método:

```
echo $this->image_lib->display_errors();
```

Una buena práctica es usar la función de procesamiento condicionalmente, mostrando un error si falla, como aquí:

```
if ( ! $this->image_lib->resize())
{
   echo $this->image_lib->display_errors();
}
```

Nota

Opcionalmente, puede especificar el formato HTML que se aplicará a los errores, al enviar las etiquetas de apertura/cierre en el método, así:

$this->image_lib->display_errors(' ', ' ');

Preferencias

Las preferencias que se describen a continuación le permiten adaptar el procesamiento de imágenes para que se ajuste a sus necesidades

Tenga en cuenta que no todas las preferencias están disponibles para cada método, por ejemplo, las preferencias del eje x/y solo están disponibles para recortar imágenes, del mismo modo, las preferencias de ancho y alto no tienen efecto sobre el recorte, la columna de "disponibilidad" indica qué métodos admiten una preferencia determinada

Leyenda de disponibilidad:

- **R - Redimensionamiento de la imagen -resize()**
- **C - Recorte de imagen - crop()**
- **X - Rotación de la imagen - rotate()**
- **W - Imagen de marca de agua - watermark()**

Preferencia	Predeterminado	Opciones	Descripción	Disponibilidad
image_library	GD2	GD, GD2, ImageMagick, NetPBM	Establece la biblioteca de imágenes que se utilizará	R, C, X, W
library_path	No	No	Establece la ruta del servidor a su biblioteca ImageMagick o NetPBM, si usa cualquiera de esas bibliotecas, debe proporcionar la ruta	R, C, X R, C,S, W
source_image	No	No	Establece el nombre/ruta de la imagen de origen, la ruta debe ser una ruta de servidor relativa o absoluta, no una URL	
dynamic_output	FALSE	TRUE/FALSE (boolean)	Determina si el nuevo archivo de imagen debe escribirse en el disco o generarse dinámicamente. Nota: Si elige la configuración dinámica, solo se puede mostrar una imagen a la vez, y no se puede colocar en la página. Simplemente genera la imagen sin formato de forma dinámica en su navegador, junto con los encabezados de imagen	R, C, X, W
file_permissions	0644	(integer)	Permisos del sistema de archivos para aplicar en el archivo de imagen resultante, escribiéndolo en el disco. **ADVERTENCIA**: Usa notación de entero en **octal**	R, C, X, W
quality	90%	1 - 100%	Establece la calidad de la imagen, cuanto mayor sea la calidad, mayor será el tamaño del archivo resultante	R, C, X, W

new_image	No	No	Establece el nombre/ruta de la imagen de destino. Utilizará esta preferencia al crear una copia de imagen, la ruta debe ser una ruta de servidor relativa o absoluta, no una URL	R, C, X, W
width	No	No	Establece el ancho al que le gustaría que se configurara la imagen	R, C
height	No	No	Establece la altura a la que desea que se ajuste la imagen	R, C
create_thumb	FALSE	TRUE/FALSE (boolean)	Indica al método de procesamiento de imágenes que crea una miniatura	R
thumb_marker	_thumb	No	Especifica el indicador de miniatura,se inserta justo antes de la extensión del archivo, por lo que mipic.jpg se convertiría en mipic_thumb.jpg	R
maintain_ratio	TRUE	TRUE/FALSE (booleano)	Especifica si se debe mantener la relación de aspecto original al redimensionar	R, C
master_dim	auto	auto, width, height	Especifica qué usar como eje maestro al redimensionar o crear miniaturas, por ejemplo, supongamos que desea cambiar el tamaño de una imagen a 100 X 75 píxeles, si el tamaño de la imagen fuente no permite el cambio de tamaño perfecto a esas dimensiones, esta configuración determina qué eje se debe utilizar como el valor fijo. "Auto" establece el eje automáticamente en función de si la imagen es más alta que ancha, o viceversa	R
rotation_angle	No	90, 180, 270, vrt, hor	Especifica el ángulo de rotación cuando se giran las imágenes, tenga en cuenta que PHP gira en sentido contrario a las agujas del reloj, por lo que una rotación de 90 grados a la derecha se debe especificar como 270	X
x_axis	No	No	Establece la coordenada X en píxeles para recortar la imagen, por ejemplo, una configuración de 30 recortará una imagen de 30 píxeles desde la izquierda	C

y_axis	No	No	Establece la coordenada Y en píxeles para recortar la imagen, por ejemplo, una configuración de 30 recortará una imagen de 30 píxeles desde la parte superior	C

Preferencias en un archivo de configuración

Si prefiere no establecer preferencias utilizando el método anterior, puede colocarlas en un archivo de configuración, cree un nuevo archivo llamado **image_lib.php**, agregue el array **$config** en ese archivo, luego guarde el archivo en **config/image_lib.php** y se usará automáticamente. **NO** necesita usar el método **$ this-> image_lib-> initialize ()** si guarda sus preferencias en un archivo de configuración

Marca de agua

La función de marca de agua requiere la biblioteca GD/GD2

Dos tipos de marca de agua

Hay dos tipos de marcas de agua que puede usar:

- **Text**: El mensaje de marca de agua se genera usando texto, ya sea con una fuente True Type que usted especifique, o utilizando la nativa que admite la librería GD, si usa la versión de True Type, su instalación de GD debe compilarse con compatibilidad con True Type (la mayoría lo son, pero no todas)
- **Overlay**: El mensaje de marca de agua se genera superponiendo una imagen (generalmente un PNG transparente o GIF) que contiene su marca de agua sobre la imagen original

Poner marca de agua a una imagen

Al igual que con los otros métodos (cambio de tamaño, recorte y rotación), el proceso general de marca de agua implica establecer las preferencias correspondientes a la acción que desea realizar y luego llamar al método de marca de agua

Aquí hay un ejemplo:

```
$config['source_image']     = '/path/a/imagen/mi_imagen.jpg';
$config['wm_text']          = 'Copyright 2018 - Bartolomé Esteban Murillo';
$config['wm_type']          = 'text';
$config['wm_font_path']     = './system/fonts/texb.ttf';
$config['wm_font_size']     = '16';
$config['wm_font_color']    = 'ffffff';
```

```
$config['wm_vrt_alignment'] = 'bottom';
$config['wm_hor_alignment'] = 'center';
$config['wm_padding']       = '20';

$this->image_lib->initialize($config);

$this->image_lib->watermark();
```

El ejemplo anterior utilizará una fuente True Type de 16 píxeles para crear el texto **"Copyright 2018 - Bartolomé Esteban Murillo"**, la marca de agua se colocará en la parte inferior/central de la imagen, a 20 píxeles de la parte inferior de la imagen

Nota

Para que la clase de imagen pueda procesar, el archivo de imagen debe tener permisos de "escritura" de archivos, por ejemplo, 777

Preferencias de marca de agua

Esta tabla muestra las preferencias que están disponibles para ambos tipos de marca de agua (**Text** y **Overlay**)

Preferencia	Predeterminado	Opciones	Descripción
wm_type	text	text, overlay	Establece el tipo de marca de agua que se debe usar
source_image	No	No	Establece el nombre/ruta de la imagen de origen, la ruta debe ser una ruta de servidor relativa o absoluta, no una URL
dynamic_output	FALSE	TRUE / FALSE (boolean)	Determina si el nuevo archivo de imagen debe escribirse en el disco o generarse dinámicamente. Nota: Si elige la configuración dinámica, solo se puede mostrar una imagen a la vez, y no se puede colocar en la página, simplemente genera la imagen sin formato de forma dinámica en su navegador, junto con los encabezados de imagen
quality	90%	1 - 100%	Establece la calidad de la imagen, cuanto mayor sea la calidad, mayor será el tamaño del archivo
wm_padding	No	Un número	Cantidad de relleno, establecida en píxeles, que se aplicará a la marca de agua para separarla del borde de las imágenes
wm_vrt_alignment	bottom	top, middle, bottom	Establece la alineación vertical para la imagen de marca de agua
wm_hor_alignment	center	left, center, right	Establece la alineación horizontal para la imagen de marca de agua

wm_hor_offset	No	No	Puede especificar un desplazamiento horizontal (en píxeles) para aplicarlo a la posición de la marca de agua, el desplazamiento normalmente mueve la marca de agua hacia la derecha, excepto si tiene su alineación establecida en "derecha", entonces su valor de desplazamiento moverá la marca de agua hacia la izquierda de la imagen
wm_vrt_offset	No	No	Puede especificar un desplazamiento vertical (en píxeles) para aplicarlo a la posición de la marca de agua, el desplazamiento normalmente mueve la marca de agua hacia abajo, excepto si tiene su alineación establecida en "abajo", entonces su valor de desplazamiento moverá la marca de agua hacia la parte superior de la imagen

Preferencias para tipo text

Esta tabla muestra las preferencias que están disponibles para el tipo **Text** de marca de agua.

Preferencia	Predeterminado	Opciones	Descripción
wm_text	No	No	El texto que desea mostrar como marca de agua, por lo general, este será un aviso de copyright
wm_font_path	No	No	La ruta del servidor a la fuente de True Type que desea utilizar, si no usa esta opción, se usará la fuente nativa de GD
wm_font_size	16	No	El tamaño del texto. Nota: Si no está utilizando la opción True Type anterior, el número se establece usando un rango de 1 a 5, de lo contrario, puede usar cualquier tamaño de píxel válido para la fuente que está usando
wm_font_color	ffffff	No	El color de la fuente, especificado en hexadecimal, se admiten tanto la versión abreviada completa de seis dígitos (es decir, 993300) como la breve de tres caracteres (es decir, fff)
wm_shadow_color	No	No	El color de la sombra paralela, especificado en hexadecimal, si lo deja vacio, no se usará una sombra paralela, se admiten tanto la versión abreviada completa de seis dígitos (es decir, 993300) como la breve de tres caracteres (es decir, fff).
wm_shadow_distance	3	No	La distancia (en píxeles) de la fuente que debería aparecer la sombra paralela

Preferencias para tipo overlay

Esta tabla muestra las preferencias que están disponibles para el tipo de marca de agua overlay:

Preferencia	Predeterminado	Opciones	Descripción
wm_overlay_path	No	No	La ruta del servidor a la imagen que desea usar como su marca de agua. Obligatorio solo si está utilizando el método de superposición
wm_opacity	50	1 - 100	Opacidad de la imagen. Puede especificar la opacidad (es decir, la transparencia) de su imagen de marca de agua. Esto permite que la marca de agua sea débil y no oscurezca por completo los detalles de la imagen original que se encuentra detrás. Una opacidad del 50% es típica
wm_x_transp	4	Un número	Si su imagen de marca de agua es una imagen PNG o GIF, puede especificar que el color de la imagen sea "transparente". Esta configuración (junto con la siguiente) le permitirá especificar ese color. Esto funciona al especificar el píxel de las coordenadas "X" e "Y" (medido desde la esquina superior izquierda) dentro de la imagen que corresponde a un píxel representativo del color que desea que sea transparente
wm_y_transp	4	Un número	Junto con la configuración anterior, esto le permite especificar la coordenada a un píxel representativo del color que desea que sea transparente

class CI_Image_lib

class CI_Image_lib

class CI_Image_lib

clear()

void clear()

Descripción

Restablece todos los valores utilizados al procesar una imagen, debe llamarlo si está procesando imágenes en un bucle

Ejemplo

```
$this->image_lib->clear();
```

valores devueltos

void

crop()

bool crop()

Descripción

Recorta una imagen, funciona de forma casi idéntica al método de cambio de tamaño, excepto que requiere que establezca preferencias para los ejes X e Y (en píxeles) especificando dónde recortar

Todas las preferencias enumeradas en la tabla de preferencias están disponibles para este método excepto estas: **rotation_angle**, **create_thumb** y **new_image**

Valores devueltos

TRUE en caso de éxito, FALSE en caso de error

Ejemplo

El método requiere que establezca preferencias para los ejes X e Y (en píxeles) especificando dónde recortar, como aquí:

```
$config['x_axis'] = 100;
$config['y_axis'] = 40;
```

Aquí hay un ejemplo que muestra cómo puedes recortar una imagen:

```
$config['image_library']= 'imagemagick';
$config['library_path'] = '/usr/X11R6/bin/';
$config['source_image'] = '/path/a/imagen/mipic.jpg';
$config['x_axis']       = 100;
$config['y_axis']       = 60;

$this->image_lib->initialize($config);

if ( ! $this->image_lib->crop())
{
 echo $this->image_lib->display_errors();
}
```

> ***Nota***
>
> Sin una interfaz visual, es difícil recortar imágenes, por lo que este método no es muy útil a menos que tenga la intención de construir dicha interfaz, eso es exactamente lo que hicimos utilizando para el módulo de galería de fotos en ExpressionEngine, el CMS que desarrollamos, agregamos una interfaz de usuario JavaScript que permite seleccionar el área de recorte

display_errors()

string display_errors([string $open = ' [, string $close = ' ']])

Descripción

Devuelve todos los errores detectados formateados como un string

Parámetros

open

Etiqueta de apertura del mensaje de error

close

Etiqueta de cierre del mensaje de error

Valores devueltos

Mensajes de error

Ejemplo

```
echo $this->image_lib->display_errors();
```

initialize()

bool initialize([array $props = array()])

Descripción

Inicializa la clase para procesar una imagen

Parámetros

props

Preferencias de procesamiento de imágenes

Valores devueltos

TRUE en caso de éxito, FALSE en el caso de configuraciones no válidas

resize()

bool resize()

Descripción

Cambia el tamaño de una imagen, permite cambiar el tamaño de la imagen original, crear una copia (con o sin cambio de tamaño) o crea una imagen en miniatura

A efectos prácticos, no hay diferencia entre crear una copia y crear una miniatura, excepto que tendrá un indicador de miniatura como parte del nombre (es decir, mipic**_thumb**.jpg)

Todas las preferencias enumeradas en la tabla de preferencias están disponibles para este método, excepto estas tres: **rotation_angle, x_axis** e **y_axis**

Valores devueltos

Devuelve: TRUE en caso de éxito, FALSE en caso de error

Ejemplos

Crear una miniatura

El método de cambio de tamaño creará un archivo en miniatura (y conserva el original) si establece esta preferencia en TRUE:

```
$config['create_thumb'] = TRUE;
```

Esta única preferencia determina si se crea o no una miniatura

Creando una Copia

El método de cambio de tamaño creará una copia del archivo de imagen, y conserva el original, si con esta preferencia establece una ruta y/o un nuevo nombre de archivo:

```
$config['new_image'] = '/path/a/nueva_imagen.jpg';
```

Notas con respecto a esta preferencia:

- Si solo se especifica el nuevo nombre de imagen, se colocará en el mismo directorio
- Si solo se especifica la ruta, la nueva imagen se colocará en el destino con el mismo nombre que el original
- Si se especifican tanto la ruta como el nombre de la imagen, se ubicará en su propio destino y se le asignará el nuevo nombre

Cambiar el tamaño de la imagen original

Si no se utiliza ninguna de las dos preferencias enumeradas anteriormente (**create_thumb** y **new_image**), el método **resize()** se centrará en la imagen original para su procesamiento

rotate()

bool rotate()

Descripción

Rota una imagen, el método requiere que el ángulo de rotación se establezca a través de su preferencia

Hay 5 opciones de rotación:

1. **90** - gira en sentido antihorario 90 grados
2. **180** - gira en sentido antihorario 180 grados
3. **270** - gira en sentido antihorario 270 grados
4. **hor** - voltea la imagen horizontalmente
5. **vrt** - voltea la imagen verticalmente

Valores devueltos

TRUE en caso de éxito, FALSE en caso de error

Ejemplos

El ángulo de rotación se establece través de su preferencia:

```
$config['rotation_angle'] = '90';
```

Aquí se muestra cómo puede rotar una imagen:

```
$config['image_library']  = 'netpbm';
$config['library_path']   = '/usr/bin/';
$config['source_image']   = '/path/a/imagen/mipic.jpg';
$config['rotation_angle'] = 'hor';

$this->image_lib->initialize($config);

if ( ! $this->image_lib->rotate())
{
    echo $this->image_lib->display_errors();
}
```

watermark()

bool watermark()

Descripción

Crea una marca de agua sobre una imagen, consulte la sección **Marca de agua** para obtener más información

Valores devueltos

Devuelve TRUE en caso de éxito, FALSE en caso de error

Input Class

La Clase Input sirve para dos propósitos:

- **Por seguridad, preprocesa los datos de entrada globales**
- **Proporciona algunos métodos de ayuda para obtener datos de entrada y preprocesarlos**

Nota

Esta clase se inicializa automáticamente por el sistema, por lo que no es necesario hacerlo manualmente

Filtrado de seguridad

El método de filtrado de seguridad se llama automáticamente cuando se invoca un nuevo controlador

Hace lo siguiente:

- Si **$config['allow_get_array']** es **FALSE** (el valor predeterminado es **TRUE**), destruye el array global **GET**
- Destruye todas las variables globales en caso de que **register_globals** esté activado
- Filtra las claves de los array **GET/POST/COOKIE**, permitiendo solo caracteres alfanuméricos (y algunos otros)
- Proporciona filtrado **XSS** (Cross-site Scripting Hacks), esto se puede habilitar globalmente, o previa solicitud
- Estandariza los caracteres de nueva línea a **PHP_EOL** (**\n** en sistemas operativos basados en UNIX, **\r\n** en Windows), esto es configurable

Filtrado de XSS

La clase Input tiene la capacidad de filtrar la entrada automáticamente para evitar ataques cross-site scripting, si desea que el filtro se ejecute automáticamente cada vez que encuentre datos **POST** o **COOKIE**, puede habilitarlo estableciendo lo siguiente:

```
$config['global_xss_filtering'] = TRUE;
```

en su archivo **application/config/config.php**

Consulte la documentación de **Security class** para obtener información sobre el uso del Filtrado de XSS en su aplicación

Importante

La configuración **'global_xss_filtering'** es **OBSOLETA** y se conserva únicamente para fines de compatibilidad con versiones anteriores. **El filtrado XSS debe realizarse en la salida**, no en la entrada

Usar datos POST, GET, COOKIE o SERVER

CodeIgniter viene con métodos de ayuda que le permiten obtener elementos **POST, GET, COOKIE o SERVER, l**a principal ventaja de utilizar los métodos proporcionados en lugar de ir directamente a un elemento **($_POST ['algo'])** es que los métodos verifican si el elemento está establecido y devolverán NULL si no lo está, esto le permite usar datos convenientemente sin tener que comprobar si el elemento existe

En otras palabras, normalmente haría esto:

```
$something = isset($_POST['algo']) ? $_POST['algo'] : NULL;
```

Con los métodos integrados en CodeIgniter puede simplemente hacer esto:

```
$something = $this->input->post('algo');
```

Los principales métodos son:

- **$this->input->post()**
- **$this->input->get()**
- **$this->input->cookie()**
- **$this->input->server()**

Usando php://input stream

Si desea utilizar **PUT, DELETE, PATCH** u otros métodos de solicitud exóticos, solo se puede acceder a ellos a través de un **input stream** especial, que solo se puede leer una vez, esto no es tan fácil como leer de, por ejemplo, el array **$_POST**, porque siempre existirá y puede intentar acceder a múltiples variables sin importar que solo tenga una oportunidad para todos los datos **POST**

CodeIgniter se encarga de eso, y usted puede leer los datos de la secuencia **php://input** en cualquier momento, simplemente usando la propiedad **$raw_input_stream**:

```
$this->input->raw_input_stream;
```

Además, si el flujo de entrada está codificado como en **$_POST**, puede acceder a sus valores llamando al método **input_stream()**:

```
$this->input->input_stream('clave');
```

De forma similar a otros métodos, como **get()** y **post()**, si no se encuentran los datos solicitados, devolverá NULL y también puede decidir si ejecutar los datos a través de **xss_clean()** pasando un valor booleano como segundo parámetro:

```
$this->input->input_stream('clave', TRUE); // XSS Limpio
$this->input->input_stream('clave', FALSE); // Sin filtrado XSS
```

Nota

Puede utilizar **method()** para saber si está leyendo datos **PUT, DELETE o PATCH**

class CI_Input

class CI_Input

class CI_Input

$raw_input_stream

$raw_input_stream

Descripción

Propiedad de solo lectura que devolverá los datos de entrada **php://** tal como están, la propiedad se puede leer varias veces

cookie()

mixed cookie([mixed $index = NULL [, bool $xss_clean = NULL]])

Descripción

Retorna el valor de **$_COOKIE ,** es idéntico a **post()** y **get()**, solo que busca datos de cookies

Parámetros

index
: Nombre del cookie

xss_clean
: Si debe aplicarse el filtrado XSS

Valores devueltos

$_COOKIE si no se han proporcionado los parámetros, de lo contrario el valor de COOKIE si se encuentra o NULL si no

Ejemplos

Este método es idéntico a **post() y get()**, solo que busca datos de cookies:

```
$this->input->cookie('algun_cookie');
$this->input->cookie('algun_cookie', TRUE); // con filtro XSS
```

Para devolver un array de múltiples valores de cookies, pase todas las claves requeridas como un array:

```
$this->input->cookie(array('algun_cookie1', 'algun_cookie2'));
```

> ***Nota***
>
> A diferencia de la función **get_cookie()** de **Cookie Helper**, este método **NO** antepone el valor configurado en **$config['cookie_prefix']**

get()

mixed get([mixed $index = NULL [, bool $xss_clean = NULL]])

Descripción

Retorna el valor de **$_GET**, para devolver un array con todos los elementos **GET** llame sin ningún parámetro, este método es idéntico a **post()**, solo que obtiene datos **GET**

Parámetros

index
: Nombre del elemento GET

xss_clean
: Si se debe aplicar el filtrado XSS

Valores devueltos

$_GET si no se han proporcionado los parámetros, de lo contrario el valor GET si se encuentra o NULL si no

Ejemplos

```
$this->input->get('algun_dato', TRUE);
```

Para devolver todos los elementos **GET** y pasarlos a través del filtro XSS, establezca el primer parámetro a NULL mientras configura el segundo parámetro como TRUE:

```
$this->input->get(NULL, TRUE); // devuelve todos los elementos de GET filtrados XSS
$this->input->get(NULL, FALSE); // devuelve todos los elementos GET sin filtro XSS
```

Para devolver un array de múltiples elementos **GET**, pase todas las claves requeridas como un array:

```
$this->input->get(array('elemento1', 'elemento2'));
```

Se aplica la misma regla para recuperar los parámetros con el filtrado XSS activado, establezca el segundo parámetro en TRUE:

```
$this->input->get(array('elemento1', 'elemento2'), TRUE);
```

get_post()

mixed get_post(string $index [, bool $xss_clean = NULL])

Descripción

Retorna el valor de **$_GET** y **$_POST ,** el método funciona más o menos igual que **post()** y **get()**, en este caso combinados, buscará en **$_GET** y **$_POST** los datos, buscando primero en **$_GET** y luego en **$_POST**

Parámetros

index

Nombre del elemento GET/POST

xss_clean

Si debe aplicarse el filtrado XSS

Valores devueltos

Valor GET/POST si se encuentra, NULL si no

Ejemplo

Este método funciona de la misma manera que **post_get()** solo que busca datos GET primero:

```
$this->input->get_post('algun_dato', TRUE);
```

> ***Nota***
>
> Este método solía actuar EXACTAMENTE como **post_get()**, pero su comportamiento ha cambiado en CodeIgniter 3.0

get_request_header()

string get_request_header(string $index [, bool $xss_clean = FALSE])

Descripción

Devuelve un único miembro del array de encabezados de solicitud o NULL si no se encuentra el encabezado buscado

Parámetros

index

Nombre del encabezado de solicitud HTTP

xss_clean

Si debe aplicarse el filtrado XSS

Valores devueltos

Un encabezado de solicitud HTTP o NULL si no se encuentra

Ejemplo

```
$this->input->get_request_header('alguna-header', TRUE);
```

input_stream()

mixed input_stream([mixed $index = NULL [, bool $xss_clean = NULL]])

Descripción

Este método es idéntico a **get()**, **post()** y **cookie()**, solo que obtiene los datos de la ruta de entrada **php://**

Parámetros

index

Nombre clave

xss_clean

Si debe aplicarse el filtrado XSS

Valores devueltos

Array de flujo de entrada si no se suministran los parámetros, de lo contrario el valor especificado si se encuentra o NULL si no

ip_address()

string ip_address()

Descripción

Devuelve la dirección IP para el usuario actual, si la dirección IP no es válida, el método devolverá '0.0.0.0'

Valores devueltos

Dirección IP del visitante o '0.0.0.0' si no es válido

Ejemplo

```
echo $this->input->ip_address();
```

Importante

Este método tiene en cuenta la configuración **$config['proxy_ips']** y devolverá la dirección HTTP_X_FORWARDED_FOR,
HTTP_CLIENT_IP, HTTP_X_CLIENT_IP o HTTP_X_CLUSTER_CLIENT_IP informada para las direcciones IP permitidas

is_ajax_request()

bool is_ajax_request()

Descripción

Indica si es una solicitud Ajax, comprueba si el encabezado del servidor **HTTP_X_REQUESTED_WITH** se ha establecido y devuelve booleano TRUE si esta o FALSE si no

Valores devueltos

TRUE si es una solicitud Ajax, FALSE si no

is_cli_request()

bool is_cli_request()

Nota

Este método es **OBSOLETO** y ahora es solo un alias para la función **is_cli()**

method()

string method([bool $upper = FALSE])

Descripción

Devuelve **$ _SERVER['REQUEST_METHOD']**, con la opción de establecerlo en mayúsculas o minúsculas

Parámetros

upper
Si se debe devolver el nombre del método de solicitud en mayúscula o minúscula

Valores devueltos

Método de solicitud HTTP

Ejemplo

```
echo $this->input->method(TRUE); // Resultado: POST
echo $this->input->method(FALSE); // Resultado: post
echo $this->input->method(); // Resultado: post
```

post()

mixed post([mixed $index = NULL [, bool $xss_clean = NULL]])

Descripción

Retorna el valor de **$_POST**, para devolver un array con todos los elementos POST llame sin ningún parámetro

Parámetros

index
Nombre del elemento POST
xss_clean
Si se debe aplicar el filtrado XSS

Valores devueltos

$_POST si no se han proporcionado los parámetros, de lo contrario el valor POST si se encuentra o NULL si no

Ejemplos

El primer parámetro contendrá el nombre del elemento POST que está buscando:

```
$this->input->post('algun_dato');
```

El método devuelve NULL si el elemento que está intentando recuperar no existe

El segundo parámetro opcional le permite ejecutar el filtro XSS, se habilita estableciendo el segundo parámetro en TRUE o configurando **$config['global_xss_filtering']** en TRUE:

```
$this->input->post('algun_dato', TRUE);
```

Para devolver todos los elementos POST y pasarlos a través del filtro XSS, establezca el primer parámetro a NULL mientras configura el segundo parámetro como booleano TRUE:

```
$this->input->post(NULL, TRUE); // devuelve todos los elementos de POST filtrados XSS
$this->input->post(NULL, FALSE); // devuelve todos los elementos POST sin filtro XSS
```

Para devolver un array de múltiples elementos POST, pase todas las claves requeridas como un array:

```
$this->input->post(array('elemento1', 'elemento2'));
```

Se aplica la misma regla, para recuperar los parámetros con el filtrado XSS activado, establezca el segundo parámetro en TRUE:

```
$this->input->post(array('elemento1', 'elemento2'), TRUE);
```

post_get()

mixed post_get(string $index [, bool $xss_clean = NULL])

Descripción

Retorna el valor de **$_POST** y **$_GET**, el método funciona más o menos igual que **post()** y **get()**, en este caso combinados, buscará en **$_POST** y **$_GET** los datos, buscando primero en **$_POST** y luego en **$_GET**

Parámetros

index
: Nombre del elemento POST/GET

xss_clean
: Si debe aplicarse el filtrado XSS

Valores devueltos

Valor POST/GET si se encuentra, NULL si no

Ejemplo

Busca en **$_POST** y **$_GET** los datos, buscando primero en **$_POST** y luego en **$_GET**:

```
$this->input->post_get('algun_dato', TRUE);
```

request_headers()

array request_headers([bool $xss_clean = FALSE])

Descripción

Devuelve un array de encabezados **HTTP**, útil si está corriendo en un entorno que no es **Apache** donde **apache_request_headers()** no está soportado

Parámetros

xss_clean
: Si debe aplicarse el filtrado XSS

Valores devueltos

Un array de encabezados de solicitud HTTP

Ejemplo

```
$headers = $this->input->request_headers();
```

server()

mixed server(mixed $index [, bool $xss_clean = NULL])

Descripción

Retorna el valor de **$ _SERVER,** es idéntico a los métodos **post()**, **get()** y **cookie()**, solo recupera los datos del servidor **$ _SERVER**

Parámetros

index
: Nombre del elemento

xss_clean
: Si debe aplicarse el filtrado XSS

Valores devueltos

El valor de **$ _SERVER** si se encuentra o NULL si no

Ejemplos

```
$this->input->server('algun_dato');
```

Para devolver un array de múltiples valores $ _SERVER, pase todas las claves requeridas como un array :

```
$this->input->server(array('SERVER_PROTOCOL', 'REQUEST_URI'));
```

set_cookie()

void set_cookie(mixed $name = '' [, string $value = '' [, int $expire = '' [, string $domain = '' [, string $path = '/' [, string $prefix = '' [, bool $secure = NULL [, bool $httponly = NULL]]]]]])

Descripción

Establece una cookie que contiene los valores que especifique

Parámetros

name
: Nombre de la cookie o una array de parámetros

value
: Valor de la cookie

expire
: Tiempo de expiración de la cookie en segundos

domain
: Dominio de cookies

path
: Ruta de las cookies

prefix
: Prefijo de nombre de cookie

secure

Si solo se transfiere la cookie a través de HTTPS

httponly

Si solo hacer que la cookie sea accesible para las solicitudes HTTP (sin JavaScript)

Hay dos formas de pasar información a este método para que se pueda establecer una cookie: **Método de array** y de **Parámetros discretos**:

Método de array

Usando este método, un array asociativo se pasa al primer parámetro:

```
$cookie = array(
  'name'   => 'Nombre de la Cookie',
  'value'  => 'El valor',
  'expire' => '86500',
  'domain' => '.algun-dominio.com',
  'path'   => '/',
  'prefix' => 'mi_prefijo_',
  'secure' => TRUE
);
$this->input->set_cookie($cookie);
```

Notas

Solo se requiere el nombre y el valor, para eliminar una cookie, configúrela con un espacio en blanco en caducidad

La caducidad se establece en segundos, que se agregará a la hora actual, no incluya el tiempo, sino solo el número de segundos a partir del momento en que desea que la cookie sea válida, si la caducidad se establece en cero, la cookie solo durará mientras el navegador esté abierto

Para las cookies de todo el sitio, independientemente de cómo se solicite su sitio, agregue su URL al dominio comenzando con un punto, como este: .**su_dominio.com**

La ruta generalmente no es necesaria ya que el método establece una ruta raíz

El prefijo solo es necesario si necesita evitar colisiones de nombres con otras cookies idénticas para su servidor

Los parámetros **httponly** y **secure**, cuando se omiten, se ajustan de manera predeterminada a sus configuraciones de **$config['cookie_httponly']** y **$config['cookie_secure']**

Parámetros discretos

Si lo prefiere, puede configurar la cookie pasando datos usando parámetros individuales:

```
$this->input->set_cookie($name, $value, $expire, $domain, $path, $prefix, $secure);
```

user_agent()

mixed user_agent([bool $xss_clean = NULL])

Descripción

Devuelve la cadena de agente de usuario (navegador web) que utiliza el usuario actual o NULL si no está disponible, consulte **User Agent Class** para conocer los métodos que extraen información de la cadena del agente de usuario

Parámetros

xss_clean

Si debe aplicarse el filtrado XSS

Valores devueltos

Cadena de agente de usuario o NULL si no está establecido

Ejemplo

```
echo $this->input->user_agent();
```

valid_ip()

bool valid_ip(string $ip [, string $which = ''])

Descripción

Toma una dirección IP como entrada y devuelve TRUE o FALSE (booleano) dependiendo de si es válida o no

Parámetros

ip

Dirección IP

which

Protocolo IP ('ipv4' o 'ipv6') que especifica un formato de IP, el valor predeterminado comprueba ambos formatos

Valores devueltos

TRUE si la dirección es válida, FALSE si no

Ejemplo

```
if ( ! $this->input->valid_ip($ip))
{
   echo 'No es válida';
}
else
{
   echo 'IP válida';
}
```

Language Class

Language Class proporciona funciones para recuperar archivos de idiomas y líneas de texto con fines de internacionalización

En la carpeta del sistema CodeIgniter, encontrará un subdirectorio de idioma que contiene un conjunto de archivos de idioma para el idioma inglés, los archivos en este directorio (**system/language/english/**) definen los mensajes regulares, mensajes de error y otros términos o expresiones generales, para las diferentes partes del framework

Puede crear o incorporar sus propios archivos de idioma, según sea necesario, a fin de proporcionar un error específico de la aplicación y otros mensajes, o para proporcionar traducciones de los mensajes principales a otros idiomas, estas traducciones o mensajes adicionales irían dentro de su directorio **application/language/**, con subdirectorios separados para cada idioma (por ejemplo, '**español**' o 'portugués')

CodeIgniter viene con un conjunto de archivos de idioma para el idioma "inglés"., se pueden encontrar traducciones adicionales aprobadas para diferentes idiomas en los repositorios de CodeIgniter

CodeIgniter carga primero el archivo de idioma de **system/language/** y luego busca en el directorio **application/language/**

Nota

Cada idioma debe ser almacenado en su propia carpeta. Por ejemplo, los archivos en español se encuentran en: **system/language/spanish**

Manejo de múltiples idiomas

Si desea admitir varios idiomas en su aplicación, debe proporcionar, para cada uno de ellos, un directorio en **application/language/** y también especificar el idioma predeterminado en **application/config/config.php**

El directorio **application/language/spanish/** contendría cualquier archivo de idioma adicional que necesite su aplicación, por ejemplo para mensajes de error

Cada uno de los otros directorios específicos de idioma contendría los archivos de lenguaje central que obtuvo de los repositorios de traducciones, o que usted mismo tradujo, así como cualquier otro adicional que necesite su aplicación

Debería almacenar el idioma que está utilizando actualmente, por ejemplo, en una variable de sesión

Archivos de idioma de muestra:

```
system/
    language/
          spanish/
                ...
            email_lang.php
            form_validation_lang.php
            ...
```

```
application/
        language/
            spanish/
               error_messages_lang.php
            portuguese/
                ...
                email_lang.php
                error_messages_lang.php
                form_validation_lang.php
                ...
```

Ejemplo de cambio de idioma

```
$idioma = $this->session->get_userdata('language');
$this->lang->load('error_messages', $idioma);
$oops = $this->lang->line('clave_mensaje');
```

Internacionalización

La clase de Idioma en CodeIgniter está diseñada para proporcionar una manera fácil y ligera de admitir múltiples idiomas en su aplicación, no pretende ser una implementación completa de lo que comúnmente se llama internacionalización y localización

Utilizamos el término **"idioma"** para referirnos a un idioma utilizando su nombre común, en lugar de utilizar cualquiera de los estándares internacionales, como "en", "en-US" o "en-CA-x-ca" para inglés y algunas de sus variantes

Nota

No hay nada que le impida usar esas abreviaturas en su aplicación !

Crear archivos de idioma

Los archivos de idioma deben nombrarse con **_lang.php** como extensión de nombre de archivo, por ejemplo, supongamos que quiere crear un archivo que contenga mensajes de error, usted podría nombrarlo: **error_lang.php**

Dentro del archivo, asignará cada línea de texto a un array llamado **$lang** con este prototipo:

```
$lang['language_key'] = 'El mensaje real que se mostrará';
```

Nota

Es una buena práctica usar un prefijo común para todos los mensajes en un archivo dado para evitar colisiones con elementos con nombres similares en otros archivos, por ejemplo, si está creando mensajes de error, puede prefijarlos con **error_**

```
$lang['error_falta_email']   = 'Debe indicar una dirección de email';
$lang['error_falta_url']     = 'Debe indicar una URL';
$lang['error_falta_usuario'] = 'Debe indicar un nombre de usuario';
```

Cargar un archivo de idioma

Para buscar una línea de un archivo en particular, primero debe cargar el archivo, la carga de un archivo de idioma se realiza con el siguiente código:

```
$this->lang->load('archivo', 'idioma');
```

Donde **archivo** es el nombre del archivo que desea cargar (sin la extensión del archivo), e **idioma** es el conjunto de idioma que lo contiene (por ejemplo, **spanish**), si falta el segundo parámetro, se usará el idioma predeterminado en el archivo **application/config/config.php**

También puede cargar varios archivos de idioma al mismo tiempo al pasar un array de archivos de idioma como primer parámetro:

```
$this->lang->load(array('archivo1', 'archivo2'));
```

Nota

El parámetro **idioma** solo puede consistir en letras

Obtener una línea de texto

Una vez que haya cargado el archivo de idioma deseado, puede acceder a cualquier línea de texto con esta función:

```
$this->lang->line('clave_idioma');
```

Donde **clave_idioma** es la clave de array correspondiente a la línea que desea mostrar

Opcionalmente puede pasar FALSE como el segundo parámetro de este método para deshabilitar el registro de errores, en caso de que no esté seguro de si existe la línea:

```
$this->lang->line('una_clave', FALSE);
```

Nota

Este método simplemente devuelve la línea, no hace '**echo**'

Usar líneas de idioma como etiquetas de formulario

Esta característica ha quedado obsoleta en la biblioteca de idiomas y se ha movido a la función **lang()** de **Language Helper**

Auto carga de idiomas

Si encuentra que necesita un idioma en particular a nivel global en su aplicación, puede indicarle a CodeIgniter que lo cargue automáticamente durante la inicialización del sistema, esto se hace abriendo el archivo **application/config/autoload.php** y agregando el idioma(s) al array de autocarga

class CI_Lang

class CI_Lang

class CI_Lang

line()

string line(string $line [, bool $log_errors = TRUE])

Descripción

Obtiene una línea de traducción de los archivos de idioma ya cargados, según la clave de la línea

Parámetros

line

Nombre de la clave de línea de idioma

log_errors

Si se registra un error si la línea no se encuentra

Valores devueltos

Cadena de línea de idioma o FALSE en caso de error

load()

mixed load(mixed $langfile [, string $idiom = '' [, bool $return = FALSE [, bool $add_suffix = TRUE [, string $alt_path = '']]]])

Descripción

Carga un archivo de idioma

Parámetros

langfile

Archivo de idioma para cargar o array con múltiples archivos

idiom

Nombre del idioma (por ejemplo "spanish")

return

Si se debe devolver el array cargado de traducciones

add_suffix

Si se agrega el sufijo '**_lang**' al nombre del archivo de idioma

alt_path

Una ruta alternativa para buscar el archivo de idioma

Valores devueltos

Un array de líneas de idioma si **$return** está establecido en TRUE, en caso contrario void

Loader Class

Loader, como su nombre lo sugiere, se usa para cargar elementos, estos elementos pueden ser bibliotecas (clases), archivos de vistas, controladores, helpers, modelos o sus propios archivos

Nota

Esta clase se inicializa automáticamente por el sistema, por lo que no es necesario hacerlo manualmente

Paquetes de aplicación

Un paquete de aplicación permite la fácil distribución de conjuntos completos de recursos en un solo directorio, con sus propias bliotecas, modelos, helpers, configuración y archivos de idioma, se recomienda que estos paquetes se coloquen en el directorio **application/third_party**, a continuación se muestra un esquema de muestra de un directorio de paquetes

El siguiente es un ejemplo de un directorio para un paquete de aplicación llamado **"Leonardo"**:

```
/application/third_party/leonardo

            config/
            helpers/
            language/
            libraries/
            models/
```

Sea cual sea el propósito del paquete de aplicación **"Leonardo"**, tiene sus propios archivos de configuración, ayudantes, archivos de idioma, bibliotecas y modelos, para usar estos recursos en sus controladores, primero debe decirle al cargador que va a cargar recursos desde un paquete, agregando la ruta del paquete a través del método **add_package_path()**

Archivos de vista del paquete

De manera predeterminada, las rutas de los archivos de vista de paquete se establecen cuando se llama a **add_package_path(), l**as rutas de vistas se recorren, y una vez que se encuentra una coincidencia, se carga esa vista

En este ejemplo, es posible ver colisiones de nombres, y posiblemente se cargue el paquete incorrecto, para garantizar que eso no ocurra, establezca el segundo parámetro opcional a FALSE al llamar a **add_package_path()**

```
$this->load->add_package_path(APPPATH.'mi_app', FALSE);
$this->load->view('mi_app_index'); // Carga

// No carga el welcome_message predeterminado, el segundo paramámetro de add_package_path() es FALSE
$this->load->view('welcome_message');
```

```
// Restablecer las cosas
$this->load->remove_package_path(APPPATH.'mi_app');

// De nuevo sin el segundo parámetro:
$this->load->add_package_path(APPPATH.'mi_app');
$this->load->view('mi_app_index'); // Carga
$this->load->view('welcome_message'); // Carga
```

class CI_Loader

class CI_Loader

class CI_Loader

add_package_path()

object add_package_path(string $path [, bool $view_cascade = TRUE])

Descripción

Agrega una ruta para que la clase **Loader** la anteponga en solicitudes posteriores de recursos, como ejemplo, el paquete de aplicación **"Leonardo"** anterior tiene una biblioteca llamada **Leonardo.php,** en nuestro controlador, haríamos lo siguiente:

```
$this->load->add_package_path(APPPATH.'third_party/leonardo/')->library('leonardo');
```

Parámetros

path
Ruta que se agrega

view_cascade
Si se deben usar vistas en cascada

Valores devueltos

Instancia de CI_Loader (method chaining)

clear_vars()

object clear_vars()

Descripción

Borra las variables de vista en caché

Valores devueltos

Una instancia de CI_Loader (method chaining)

config()

bool config(string $file [, bool $use_sections = FALSE [, bool $fail_gracefully = FALSE]])

Descripción

Este método es un alias del método de carga del archivo de configuración: **$this->config->load()**

Parámetros

file
: Nombre del archivo de configuración

use_sections
: Si los valores de configuración deben cargarse en su propia sección

fail_gracefully
: Si solo devuelve FALSE en caso de fallo

Valores devueltos

TRUE en el éxito, FALSE si falla

database()

```
mixed database([ mixed $params = '' [, bool $return = FALSE [, bool $query_builder = NULL]]])
```

Descripción

Este método le permite cargar la clase de la Base de datos, los dos parámetros son opcionales, por favor, mire la sección **Bases de datos** para más información

Parámetros

params
: Nombre del grupo de Base de datos u opciones de configuración

return
: Si se debe devolver el objeto de Base de datos cargado

query_builder
: Si se carga el Query Builder

Valores devueltos

Instancia de CI_DB cargada o FALSE en caso de error si **$return** está establecido a TRUE, de lo contrario la instancia de CI_Loader (method chaining)

dbforge()

```
mixed dbforge([ object $db = NULL [, bool $return = FALSE]])
```

Descripción

Carga la clase **Database Forge**

Parámetros

db
: Objeto de Base de datos

return
: Si se devuelve la instancia de **Database Forge**

Valores devueltos

Instancia de CI_DB_forge cargada si **$return** es TRUE, de lo contrario la instancia de CI_Loader (method chaining)

dbutil()

mixed dbutil([object $db = NULL [, bool $return = FALSE]])

Descripción

Carga la clase **Database Utilities**

Parámetros

db
Objeto de Base de datos

return
Si se devuelve la instancia de **Database Utilities**

Valores devueltos

Instancia de CI_DB_utility cargada si **$return** es TRUE, de lo contrario la instancia de CI_Loader (method chaining)

driver()

object driver(mixed $library [, array $params = NULL [, string $object_name]])

Descripción

Este método se usa para cargar las bibliotecas de los controladores, actúa de manera muy similar al método **library()**

Parámetros

library
Nombre de la biblioteca como un string o un array con múltiples bibliotecas

params
Array opcional de parámetros para pasar al constructor de la biblioteca cargada

object_name
Nombre de objeto opcional para asignar a la biblioteca

Valores devueltos

Instancia de CI_Loader (method chaining)

Ejemplo

Si desea utilizar sesiones con CodeIgniter, el primer paso es cargar el session driver dentro de su controlador:

```
$this->load->driver('session');
```

Una vez cargada, la biblioteca estará lista para usar, usando **$this->session**

Los archivos del controlador deben almacenarse en un subdirectorio dentro del directorio principal **"libraries"**, o dentro de su directorio personal **application/libraries,** el subdirectorio debe coincidir con el nombre de la clase principal
Lea la descripción de **Drivers** para más detalles

Además, se pueden cargar varias bibliotecas de controladores al mismo tiempo pasando un array de controladores al método de carga:

```
$this->load->driver(array('session', 'cache'));
```

Opciones de configuración

El segundo parámetro, opcional, le permite pasar opciones de configuración, por lo general, pasará estos como un array:

```
$config = array(
  'sess_driver'         => 'cookie',
  'sess_encrypt_cookie' => true,
  'encryption_key'      => 'clave_secreta'
);

$this->load->driver('session', $config);
```

Las opciones de configuración generalmente también se pueden establecer a través de un archivo de configuración, cada biblioteca lo explica en detalle en su propia sección, por lo tanto, lea la información relativa a cada una

Asignación de un controlador a un nombre de objeto diferente

Si el tercer parámetro, opcional, está en blanco, la biblioteca se asignará a un objeto con el mismo nombre que la clase padre, por ejemplo, si la biblioteca se llama **Session**, se asignará a una variable llamada **$this->session**

Si prefiere establecer sus propios nombres de clase, puede pasar su valor al tercer parámetro:

```
$this->load->library('session', '', 'mi_session');

// Ahora se accede a la clase de sesión usando:
$this->mi_session
```

file()

mixed file(string $path [, bool $return = FALSE])

Descripción

Este es un método genérico de carga de archivos, proporcione la ruta de archivo y el nombre en el primer parámetro y se abrirá y leerá el archivo, por defecto, los datos se envían a su navegador, al igual que un archivo de vista, pero si establece el segundo parámetro como booleano TRUE, en su lugar devolverá los datos como un string

Parámetros

path

Ruta del archivo

return

Si se debe devolver el archivo cargado

Valores devueltos

Contenidos del archivo si **$return** está establecido en TRUE, de lo contrario la instancia de CI_Loader (method chaining)

get_package_paths()

array get_package_paths([bool $include_base = TRUE])

Descripción

Devuelve todas las rutas de paquetes disponibles actualmente

Parámetros

include_base

Si se debe incluir BASEPATH

Valores devueltos

Array de rutas de paquetes

get_var()

mixed get_var(string $key)

Descripción

Este método verifica el array asociativo de variables disponibles para sus vistas, esto es útil si por alguna razón se establece una variable en una biblioteca u otro método de controlador usando **$this->load->vars()**

Parámetros

key

Clave de nombre de variable

Valores devueltos

Valor si se encuentra la clave, NULL si no

get_vars()

array get_vars()

Descripción

Este método recupera todas las variables disponibles para sus vistas

Valores devueltos

Un array de todas las variables de vista asignadas

helper()

object helper(mixed $helpers)

Descripción

Este método carga archivos auxiliares **Helper**

Parámetros

helpers

Nombre del helper como un string o un array que contiene múltiples helpers, el nombre es sin la extensión

Valores devueltos

Instancia de CI_Loader (method chaining)

is_loaded()

mixed is_loaded(string $class)

Descripción

Le permite verificar si una clase ya se ha cargado o no

> ***Nota***
>
> La palabra "clase" aquí se refiere a bibliotecas y controladores

Parámetros

class

Nombre de la clase

Valores devueltos

Nombre de propiedad Singleton si se encuentra, FALSE si no

Si la clase solicitada se ha cargado, el método devuelve su nombre asignado en el superobjeto de CI y FALSE si no es así:

```
$this->load->library('form_validation');
$this->load->is_loaded('Form_validation');      // devuelve 'form_validation'

$this->load->is_loaded('Librería_que_no_existe');  // devuelve FALSE
```

> ***Importante***
>
> Si tiene más de una instancia de una clase (asignada a diferentes propiedades), se devolverá la primera

```
$this->load->library('form_validation', $config, 'fv');
$this->load->library('form_validation');

$this->load->is_loaded('Form_validation');     // devuelve 'fv'
```

language()

object language(mixed $files [, string $lang = ''])

Descripción

Este método es un alias del método de carga del lenguaje: **$this->lang->load()**

Parámetros

files
Nombre del archivo de idioma o un array de múltiples archivos de idioma

lang
Nombre del idioma

Valores devueltos

Instancia de CI_Loader (method chaining)

library()

object library(mixed $library [, array $params = NULL [, string $object_name = NULL]])

Descripción

Este método se usa para cargar clases principales

Parámetros

library
Nombre de la biblioteca como un string o un array con múltiples bibliotecas

params
Array opcional de parámetros para pasar al constructor de la biblioteca cargada

object_name
Nombre de objeto opcional para asignar la biblioteca

Valores devueltos

Instancia de CI_Loader (method chaining)

Ejemplo

Nota

Usamos los términos "clase" y "bibliotecas" indistintamente

Si desea enviar un correo electrónico con CodeIgniter, el primer paso es cargar la clase **email** dentro de su controlador:

```
$this->load->library('email');
```

Una vez cargada, la biblioteca esta lista para usarse, usando:

```
$this->email
```

Los archivos de la biblioteca pueden almacenarse en subdirectorios dentro del directorio principal **"libraries"**, o dentro de su directorio **application/libraries,** para cargar un archivo ubicado en un subdirectorio, simplemente incluya la ruta, relativa al directorio **"libraries"**

Por ejemplo, si tiene un archivo ubicado en:

```
libraries/pintores/Goya.php
```

Lo cargará usando:

```
$this->load->library('pintores/goya');
```

Puede anidar el archivo en tantos subdirectorios como desee

Además, se pueden cargar múltiples bibliotecas al mismo tiempo pasando un array de bibliotecas al método de carga:

```
$this->load->library(array('email', 'table'));
```

Opciones de configuración

El segundo parámetro, opcional, le permite opcionalmente pasar la configuración, por lo general, pasará estos como un array:

```
$config = array (
  'mailtype' => 'html',
  'charset'  => 'utf-8',
  'priority' => '1'
);

$this->load->library('email', $config);
```

Las opciones de configuración generalmente también se pueden establecer a través de un archivo de configuración, cada biblioteca lo explica en detalle, por lo tanto, lea la información relativa a cada una

Tenga en cuenta que cuando se proporcionan varias bibliotecas en un array en el primer parámetro, cada una recibirá la misma información de parámetros

Asignar una biblioteca a un nombre de objeto diferente

Si el tercer parámetro, opcional, está en blanco, la biblioteca generalmente se asignará a un objeto con el mismo nombre que la biblioteca, por ejemplo, si la biblioteca se llama **Calendar**, se asignará a una variable llamada **$this->calendar**

Si prefiere establecer sus propios nombres de clase, puede pasar su valor en el tercer parámetro:

```
$this->load->library('calendar', NULL, 'mi_calendario');

// Ahora se accede a la clase de calendario usando:
$this->mi_calendario
```

Tenga en cuenta que cuando se suministran varias bibliotecas en un array en el primer parámetro, este parámetro se descarta

model()

object model(mixed $model [, string $name = '' [, string $db_conn = FALSE]])

Descripción

Este método se usa para cargar sus archivos de modelo

Parámetros

model

Nombre del modelo o un array que contiene varios modelos

name

Si desea que a su modelo se le asigne un nombre de objeto diferente, puede especificarlo mediante este parámetro:

```
$this->load->model('nombre_modelo', 'ieso');
$this->ieso->method();
```

db_conn

Opcional, configuración de la Base de datos a cargar

Valores devueltos

Instancia de CI_Loader (method chaining)

Ejemplo

```
$this->load->model('nombre_del_modelo');
```

Si su modelo está ubicado en un subdirectorio, incluya la ruta relativa desde su directorio de modelos, por ejemplo, si tiene un modelo ubicado en **application/models/blog/Consultas.php**, lo cargará usando:

```
$this->load->model('blog/consultas');
```

remove_package_path()

object remove_package_path([string $path = ''])

Descripción

Cuando su controlador haya terminado de usar los recursos de un paquete de aplicaciones, y particularmente si tiene otros paquetes de aplicaciones con los que desea trabajar, para que el cargador ya no busque recursos en ese directorio es posible que desee eliminar la ruta del paquete, para eliminar la última ruta agregada, simplemente llame al método sin parámetros, o para eliminar una ruta específica del paquete, especifique la misma ruta previamente dada a **add_package_path ()**:

```
$this->load->remove_package_path(APPPATH.'third_party/leonardo/');
```

Parámetros

path

Ruta para eliminar

Valores devueltos

Instancia de CI_Loader (method chaining)

vars()

object vars(mixed $vars [, mixed $val = ''])

Descripción

Este método toma un array asociativo como entrada y genera variables usando la función PHP **extract(),** este método produce el mismo resultado que usar el segundo parámetro del método **$this->load->view ()** anterior, la razón por la que utilizar este método de forma independiente es si desea establecer algunas variables globales en el constructor de su controlador y hacer que estén disponibles en cualquier archivo de vista cargado desde cualquier método, puede tener múltiples llamadas a este método, los datos se almacenan en caché y se fusionan en un array para la conversión a variables

Parámetros

vars
: Un array de variables o un solo nombre de variable

val
: Valor de variable opcional

Valores devueltos

Instancia de CI_Loader (method chaining)

view()

mixed view(string $view [, array $vars = array() [, bool return = FALSE]])

Descripción

Este método se usa para cargar sus archivos de vista. En la sección **Views** se muestra cómo se usa este método

Parámetros

view
: Es el nombre del archivo de vista que desea cargar, no necesita especificar la extensión **.php**, a menos que esté usando otra distinta

vars
: Parámetro opcional que puede tomar como entrada un array asociativo o un objeto, que se ejecuta mediante la función **extract()** de PHP para convertir a variables que se pueden usar en sus archivos de vista

return
: Si se devuelve la vista cargada, parámetro opcional que le permite cambiar el comportamiento del método para que devuelva los datos como un string en lugar de enviarlos al navegador, puede ser útil si quiere procesar los datos de alguna forma, si lo establece a TRUE , devolverá datos, por defecto es FALSE, que los envía al navegador. Recuerde asignarla a una variable si quiere que los datos sean devueltos:

```
$string = $this->load->view('mi_archivo', '', TRUE);
```

Valores devueltos

El string de contenido si **$return** está establecido en TRUE, de lo contrario la instancia de CI_Loader (method chaining)

Migrations Class

Las migraciones son una forma conveniente de modificar su Base de datos de forma estructurada y organizada, puede editar fragmentos de SQL a mano, pero entonces sería responsable de decirle a otros desarrolladores que deben ejecutarlos, también debería realizar un seguimiento de los cambios que se deben ejecutar contra las máquinas de producción la próxima vez que implemente

La migración de la tabla de Base de datos rastrea qué migraciones ya se han ejecutado, de modo que todo lo que tiene que hacer es actualizar sus archivos de aplicación y llamar a **$this->migration->current()** para determinar qué migraciones se deben ejecutar, la versión actual se encuentra en **application/config/migration.php**

Nombres de archivos de migración

Cada migración se ejecuta en orden numérico hacia adelante o hacia atrás según el método utilizado, dos estilos de numeración están disponibles:

- **Secuencial**: cada migración se numera en secuencia, comenzando con 001, cada número debe tener tres dígitos, y no debe haber espacios en la secuencia. (Este fue el esquema de numeración anterior a CodeIgniter 3.0)
- **Timestamp**: cada migración se numera utilizando la marca de tiempo cuando se creó la migración, en formato YYYYMMDDHHIISS (por ejemplo, 2018061618300537), esto ayuda a evitar conflictos de numeración cuando se trabaja en un entorno de equipo, y es el esquema preferido en CodeIgniter 3.0 y posterior

Puede seleccionarse el estilo usando la configuración **$config['migration_type']** en su archivo **application/config/migration.php**

Independientemente del estilo de numeración que elija, prefija sus archivos de migración con el número de migración seguido de un guión bajo y un nombre descriptivo para la migración

Ejemplo

```
001_add_blog.php (numeración secuencial)
2018061618300537_add_blog.php (numeración timestamp)
```

Crear una migración

Esta será la primera migración para un nuevo sitio que tiene un blog, todas las migraciones van en el directorio **application/migrations/** y tienen nombres como **2018061618300537_add_blog.php**

```
<?php

defined('BASEPATH') OR exit('No direct script access allowed');

class Migration_Add_blog extends CI_Migration {
```

```
  public function up()
  {
     $this->dbforge->add_field(array(
       'blog_id' => array(
          'type'           => 'INT',
          'constraint'     => 5,
          'unsigned'       => TRUE,
          'auto_increment' => TRUE
       ),
       'blog_nombre' => array(
         'type'       => 'VARCHAR',
         'constraint' => '100',
       ),
       'blog_detalle' => array(
          'type' => 'TEXT',
          'null' => TRUE,
       ),
     ));
     $this->dbforge->add_key('blog_id', TRUE);
     $this->dbforge->create_table('blog');
  }

  public function down()
  {
     $this->dbforge->drop_table('blog');
  }
}
```

Luego, en **application/config/migration.php** configure :

```
$config['migration_version'] = 2018061618300537;.
```

Ejemplo

En este ejemplo, se coloca un código simple en **application/controllers/Migrate.php** para actualizar el esquema:

```
?php

class Migrate extends CI_Controller{

   public function index()
   {
      $this->load->library('migration');

      if ($this->migration->current() === FALSE)
      {
         show_error($this->migration->error_string());
      }
```

```
    }
}
```

Preferencias de migración

La siguiente es una tabla de todas las opciones de configuración para migraciones:

Preferencias	Opciones	Predeterminado	Descripción
migration_enabled	FALSE	TRUE/FALSE	Habilita o deshabilita las migraciones
migration_path	APPPATH.'migrations/ '	No	La ruta de acceso a la carpeta de migraciones
migration_version	0	No	La versión actual que su Base de datos debería usar
migration_table	migrations	No	El nombre de la tabla para almacenar el número de versión del esquema
migration_auto_latest	FALSE	TRUE/FALSE	Habilita o deshabilita la ejecución automática de migraciones
migration_type	'timestamp'	'timestamp' / 'sequential'	El tipo de identificador numérico utilizado para nombrar los archivos de migración

class CI_Migration

class CI_Migration

class CI_Migration

current

mixed current()

Descripción

Migra hasta la versión actual, lo que está configurado para **$config['migration_version']** en **application/config/migration.php**

Valores devueltos

TRUE si no se encuentran migraciones, string de versión actual en caso de éxito, FALSE en caso de error

error_string()

string error_string()

Descripción

Devuelve un string con los errores que se detectaron al realizar una migración

Valores devueltos

Mensajes de error

find_migrations()

array find_migrations()

Descripción

Se devuelve un array de nombres de archivos de migración que se encuentran en la propiedad **migration_path**

Valores devueltos

Un array de archivos de migración

latest()

mixed latest()

Descripción

Esto funciona de forma muy similar a **current()**, pero en lugar de buscar en **$config['migration_version']** la clase **Migration** utilizará la migración más reciente que se encuentre en el sistema de archivos

Valores devueltos

String de versión actual en caso de éxito, FALSE en caso de error

version()

mixed version(mixed $target_version)

Descripción

La versión se puede utilizar para deshacer cambios o avanzar de forma programática a versiones específicas, funciona igual que **current()** pero ignora **$config['migration_version']**

Parámetros

target_version
: Versión de migración para procesar

Valores devueltos

TRUE si no se encuentran migraciones, string de versión actual en caso de éxito, FALSE en caso de error

Ejemplo

```
$this->migration->version(5);
```

Output Class

Output Class es una clase principal con una función principal: enviar la página web finalizada al navegador solicitante, también es responsable de almacenar en caché sus páginas web, si usa esa característica

Nota

Esta clase se inicializa automáticamente por el sistema, por lo que no es necesario hacerlo manualmente

En circunstancias normales, ni siquiera se dará cuenta de la clase Output, ya que funciona de forma transparente sin su intervención, por ejemplo, cuando usa la clase **Loader** para cargar un archivo de vista, se pasa automáticamente a la clase Output, que CodeIgniter llamará automáticamente al final de la ejecución del sistema, sin embargo, es posible intervenir manualmente con la salida si es necesario

class CI_Output

class CI_Output

class CI_Output

$parse_exec_vars

```
$parse_exec_vars = TRUE;
```

Habilita/deshabilita el análisis de las pseudovariables **{elapsed_time}** y **{memory_usage}**

CodeIgniter analizará esos tokens en su salida de forma predeterminada, para desactivar esto, establezca esta propiedad en FALSE en su controlador:

```
$this->output->parse_exec_vars = FALSE;
```

append_output()

```
object append_output(string $output)
```

Descripción

Añade datos en el string de salida

Parámetros

output

Datos de salida adicionales para agregar

Valores devueltos

Instancia de CI_Output (method chaining)

Ejemplo

```
$this->output->append_output($datos);
```

cache()

object cache(int $time)

Descripción

Almacena en caché la página actual durante la cantidad de minutos especificada, para obtener más información, consulte la sección de **almacenamiento en caché**

Parámetros

time

Tiempo de caducidad de caché en minutos

Valores devueltos

Instancia de CI_Output (method chaining)

_display()

void _display([string $output = ''])

Descripción

Envía datos de salida finalizados al navegador junto con los encabezados de servidor, también detiene los temporizadores de referencia

Parámetros

output

Anulación de datos de salida

> ***Nota***
>
> Este método se llama automáticamente al final de la ejecución del script, no tendrá que llamarlo manualmente a menos que esté cancelando la ejecución del script usando **exit()** o **die()** en su código

valores devueltos

void

Ejemplo

```
$response = array('status' => 'OK');

$this->output
 ->set_status_header(200)
 ->set_content_type('application/json', 'utf-8')
 ->set_output(json_encode($response, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE |
JSON_UNESCAPED_SLASHES))
 ->_display();
exit;
```

Nota

Llamar a este método manualmente sin abortar la ejecución del script generará una salida duplicada

enable_profiler

object enable_profiler([bool $val = TRUE])

Descripción

Le permite habilitar/deshabilitar el **Profiler**, que mostrará los puntos de referencia y otros datos en la parte inferior de sus páginas para fines de depuración y optimización

Parámetros

val

Si habilitar o deshabilitar el Profiler

Valores devueltos

Instancia de CI_Output (method chaining)

Ejemplo

Para habilitar el generador de perfiles, coloque la siguiente línea en cualquier lugar dentro de sus métodos de Controlador:

```
$this->output->enable_profiler(TRUE);
```

Cuando se habilite, se generará un informe y se insertará en la parte inferior de sus páginas

Para deshabilitar el generador de perfiles, debe usar:

```
$this->output->enable_profiler(FALSE);
```

get_content_type

string get_content_type()

Descripción

Devuelve el encabezado **HTTP Content-Type** que está actualmente en uso, excluyendo el valor del conjunto de caracteres

Valores devueltos

String de tipo de contenido

Ejemplo

```
$mime= $this->output->get_content_type();
```

Nota

Si no se establece, el valor de retorno predeterminado es '**text/html**'

get_header()

mixed get_header(string $header)

Descripción

Devuelve el valor del encabezado HTTP solicitado o NULL si el encabezado solicitado no está establecido

Parámetros

header

Nombre del encabezado HTTP

Valores devueltos

Encabezado de respuesta HTTP o NULL si no se encuentra

Ejemplo

```
$this->output->set_content_type('text/plain', 'UTF-8');
echo $this->output->get_header('content-type');
// Salidas: text/plain; charset=utf-8
```

Nota

- El nombre del encabezado se compara de una manera insensible a mayúsculas y minúsculas
- Los encabezados sin formato enviados a través de la función del encabezado nativo de PHP() también se detectan

get_output()

string get_output()

Descripción

Le permite recuperar manualmente cualquier salida que se haya enviado para su almacenamiento en la clase output, tenga en cuenta que los datos solo se podrán recuperar de este método si se han enviado previamente a la clase **output** mediante una de las funciones de CodeIgniter como **$this->load->view()**

Valores devueltos

String de salida

Ejemplo

```
$string = $this->output->get_output();
```

set_content_type

object set_content_type(string $mime_type [, string $charset = NULL])

Descripción

Le permite configurar el tipo de MIME de su página para que pueda servir datos JSON, JPEG, XML, etc. fácilmente

Parámetros

mime_type

String con el tipo MIME

charset

Conjunto de caracteres

Puede establecer el conjunto de caracteres del documento, usando este parámetro:

```
$this->output->set_content_type('css', 'utf-8');
```

Valores devueltos

Instancia de CI_Output (method chaining)

Ejemplo

```
$this->output->set_content_type('application/json')
  ->set_output(json_encode(array('foo' => 'bar')));

// También podría usar ".jpeg" que tendrá el punto eliminado antes
// de buscar en config/mimes.php
$this->output->set_content_type('jpeg')->set_output(file_get_contents('files/algo.jpg'));
```

Importante

Asegúrese de que cualquier string **no_mime** que pase a este método exista en **application/config/mimes.php** o no tendrá efecto

set_header

object set_header(string $header [, bool $replace = TRUE])

Descripción

Le permite configurar manualmente los encabezados del servidor, que la clase de salida enviará por usted cuando entregue la visualización renderizada final

Parámetros

header

Encabezado de respuesta HTTP

replace

Si se reemplaza el antiguo valor del encabezado, si ya está configurado

Valores devueltos

Instancia de CI_Output (method chaining)

Ejemplo

```
$this->output->set_header('HTTP/1.0 200 OK');
$this->output->set_header('HTTP/1.1 200 OK');
$this->output->set_header('Last-Modified: '.gmdate('D, d M Y H:i:s', $last_update).' GMT');
$this->output->set_header('Cache-Control: no-store, no-cache, must-revalidate');
$this->output->set_header('Cache-Control: post-check=0, pre-check=0');
$this->output->set_header('Pragma: no-cache');
```

set_output()

object set_output(string $output)

Descripción

Le permite establecer manualmente el stribng de salida final

Parámetros

output

Cadena para establecer el resultado de salida

Valores devueltos

Instancia de CI_Output (method chaining)

Ejemplo

```
$this->output->set_output($datos);
```

> ***Importante***
>
> Si configura su salida manualmente, debe ser lo último que haga en el método desde lo que lo llama, por ejemplo, si construye una página en uno de sus métodos de controlador, no configure la salida hasta el final

set_profiler_sections()

object set_profiler_sections(array $sections)

Descripción

Le permite habilitar/deshabilitar secciones específicas del **Profiler** cuando está habilitado, consulte la documentación de **Profiler** para obtener más información

Parámetros

sections

Secciones de Profiler

Valores devueltos

Instancia de CI_Output (method chaining)

set_status_header()

object set_status_header([int $code = 200 [, string $text = '']])

Descripción

Le permite configurar manualmente un encabezado de estado del servidor

Parámetros

code

Código de estado HTTP

text

Mensaje opcional

Valores devueltos

Instancia de CI_Output (method chaining)

Ejemplo

```
//Establece el encabezado como: no autorizado
$this->output->set_status_header(401);
```

Vea aquí para una lista completa de encabezados: **http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html**

Nota

Este método es un alias para la función común: **set_status_header()**

Pagination Class

Pagination Class , la clase de paginación de CodeIgniter, es muy fácil de usar, y es 100% personalizable, ya sea dinámicamente o mediante preferencias almacenadas

Si no está familiarizado con el término "paginación", se refiere a los enlaces que le permiten navegar de página en página, de esta manera:

```
« First  < 1 2 3 4 5 >  Last »
```

Ejemplo

Aquí hay un ejemplo simple que muestra cómo crear paginación en uno de sus métodos de controlador:

```
$this->load->library('pagination');

$config['base_url']   = 'http://ejemplo.com/index.php/test/pagina/';
$config['total_rows'] = 200;
$config['per_page']   = 20;

$this->pagination->initialize($config);

echo $this->pagination->create_links();
```

Notas

El array **$config** contiene sus variables de configuración, se pasa al método **$this->pagination->initialize()** como se muestra arriba

Aunque hay unos veinte elementos que puede configurar, como mínimo necesita los tres que se muestran, aquí hay una descripción de lo que representan esos elementos:

- **base_url** : Esta es la URL completa de la clase/método del controlador que contiene su paginación, en el ejemplo anterior, apunta a un controlador llamado **"Test"** y un método llamado **"pagina",** tenga en cuenta que puede redirigir su URI si necesita una estructura diferente
- **total_rows:** Este número representa las filas totales en el conjunto de resultados para el que está creando la paginación, normalmente, este número será el total de las filas devueltas a su consulta de Base de datos
- **per_page:** La cantidad de elementos que pretende mostrar por página, en el ejemplo anterior, mostraría 20 elementos por página

El método **create_links()** devuelve una string vacía cuando no hay paginación para mostrar

Establecer preferencias en un archivo de configuración

Si prefiere no establecer preferencias utilizando el método anterior, puede colocarlas en un archivo de configuración

Simplemente cree un nuevo archivo llamado **pagination.php**, agregue el array **$config** en ese archivo, luego guarde el

archivo en **application/config/pagination.php** y se usará automáticamente

NO necesita usar **$this->pagination->initialize()** si guarda sus preferencias en un archivo de configuración

Personalizar la paginación

Seguidamente una lista de todas las preferencias que puede pasar al método de inicialización para adaptar la pantalla:

$config['uri_segment'] = 3;
El método de paginación determina automáticamente qué segmento de su URI contiene el número de página, si necesita algo diferente, puede especificarlo

$config['num_links'] = 2;
La cantidad de enlaces de "dígitos" que desea antes y después del número de página seleccionado, por ejemplo, el número 2 colocará dos dígitos en cada lado, como en los enlaces de ejemplo en la parte superior de esta página

$config['use_page_numbers'] = TRUE;
De forma predeterminada, el segmento URI usará el índice de inicio para los elementos que está paginando, si prefiere mostrar el número de página real, configúrelo como TRUE

$config['page_query_string'] = TRUE;
De forma predeterminada, la biblioteca de paginación supone que está utilizando Segmentos de URI y construye sus enlaces de la siguiente manera:

```
http://ejemplo.com/index.php/test/pagina/20
```

Si tiene **$config['enable_query_strings']** establecido en TRUE, sus enlaces se volverán a escribir automáticamente utilizando **Query Strings,** esta opción también se puede establecer explícitamente, usando **$config['page_query_string']** establecido en TRUE, el enlace de paginación se convertirá en:

```
http://ejemplo.com/index.php?c=test&m=pagina&per_page=20
```

$config['query_string_segment'] = 'su_string'
Tenga en cuenta que **"per_page"** es el string de consulta predeterminado que se pasa, sin embargo puede redefinirlo
$config['reuse_query_string'] = FALSE;

Por defecto, se ignorarán los argumentos de la Cadena de consulta (nada que ver con otras opciones de string de consulta), establecer esta configuración en TRUE agregará argumentos de string de consulta existentes a la URL después del segmento URI y antes del sufijo:

```
http://ejemplo.com/index.php/test/pagina/20?query=search%term
```

Esto te ayuda a mezclar segmentos de URI normales y consultar los argumentos de string, que hasta 3.0 no era posible

$config['prefix'] = '';
Un prefijo personalizado agregado a la ruta, el valor del prefijo estará justo antes del segmento

$config['suffix'] = '';
Un sufijo personalizado agregado a la ruta, el valor del sufijo será justo después del segmento

$config['use_global_url_suffix'] = FALSE;

Cuando se establece en TRUE, anulará el valor de **$config['suffix']** y, en su lugar, lo configurará en el archivo de **application/config/config.php** en **$config ['url_suffix']**

Agregar marcado de delimitación

Si desea rodear la paginación completa con algún marcado, puede hacerlo con estas dos preferencias:

$config['full_tag_open'] = '<p>';

La etiqueta de apertura colocada en la parte izquierda de todo el resultado

$config['full_tag_close'] = '</ p>';

La etiqueta de cierre se ubica en la parte derecha de todo el resultado

El enlace "First" - "Primera"

$config['first_link'] = 'first';

Texto que se muestra en el enlace izquierdo **"first",** si no desea que se muestre , establezca su valor a FALSE

Nota
Este valor también se puede traducir a través de un archivo de idioma

$config['first_tag_open'] = ' ';

Etiqueta de apertura del enlace "**first**"

$config['first_tag_close'] = '</ div>';

Etiqueta de cierre del enlace "**first**"

$config['first_url'] = '';

Una URL alternativa para usar en el enlace "**first**"

El enlace "Last" - "Última"

$config['last_link'] = 'Last';

Texto que muestra en el enlace derecho "**last**", si no desea que se muestre, establezca su valor a FALSE

Nota
Este valor también se puede traducir a través de un archivo de idioma

$config['last_tag_open'] = ' ';

Etiqueta de apertura del enlace "**last**"

$config['last_tag_close'] = '</ div>';

Etiqueta de cierre del enlace "**last**" enlace

El enlace "Next" - "Siguiente"

$config['next_link'] = '& gt;';

El texto que desea mostrar en el enlace "**next**", si no desea que se muestre, establezca su valor en FALSE

Nota
Este valor también se puede traducir a través de un archivo de idioma

$config['next_tag_open'] = ' ';

Etiqueta de apertura del enlace "**next**"

$config['next_tag_close'] = '</ div>';

La etiqueta de cierre del enlace "**next**"

El enlace "Previous" - "Anterior"

$config ['prev_link'] = '<';

El texto que desea mostrar en el enlace "**previous**", si no desea que se muestre, establezca su valor en FALSE

Nota
Este valor también se puede traducir a través de un archivo de idioma

$config['prev_tag_open'] = ' ';

La etiqueta de apertura del enlace "**previous**"

$config['prev_tag_close'] = '</ div>';

La etiqueta de cierre del enlace "**previous**"

El enlace "Current Page" - "Página actual"

$config['cur_tag_open'] = '<b>';

Etiqueta de apertura del enlace "**curren**t"

$config['cur_tag_close'] = '</ b>';

Etiqueta de cierre del enlace "**current**"

El enlace "Digit" - "Dígito"

$config['num_tag_open'] = ' ';

Etiqueta de apertura del enlace "**digit**"

$config['num_tag_close'] = '</ div>';

Etiqueta de cierre del enlace "**digit**"

Ocultar páginas

Si quiere que no se listen páginas específicas (por ejemplo, quiere solamente los enlaces **"siguiente"** y "**anterior**", puede suprimir su presentación agregando:

```
$config['display_pages'] = FALSE;
```

Agregar atributos a los anclajes

Si desea agregar un atributo adicional para agregar a cada enlace representado por la clase pagination, puede configurarlos como pares de clave/valor en la configuración de " **attributes** ":

```
// Produce: class = "mi_clase"
$config['attributes'] = array('clase' => 'mi_clase');
```

> ***Nota***
>
> El antiguo método de configuración de clases a través de **"anchor_class"** está en desuso

Desactivar el atributo "rel"

Por defecto, el atributo rel se genera dinámicamente y se anexa a los anclajes apropiados, si por alguna razón quiere desactivarlo, puede pasar FALSE como un atributo regular

```
$config['attributes']['rel'] = FALSE;
```

class CI_Pagination

class CI_Pagination

class CI_Pagination

create_links()

string create_links()

Descripción

Devuelve una barra de **"paginación"** que contiene los enlaces generados o una string vacía si solo hay una página

Valores devueltos

Paginación con formato HTML

initialize()

object initialize([array $params = array()])

Descripción

Inicializa la clase de paginación con sus opciones preferidas

Parámetros

params

Parámetros de configuración

Valores devueltos

Instancia CI_Pagination (method chaining)

Template Parser Class

Template Parser Class puede realizar la sustitución de texto simple para las pseudovariables contenidas en los archivos de vista, puede analizar variables simples o pares de etiquetas variables

Si nunca ha utilizado un motor de plantillas, los nombres de las pseudovariables se incluyen entre llaves, así:

```
<html>
 <head>
    <title>{blog_título}</title>
 </head>
 <body>
    <h3>{blog_cabecera}</h3>

 {blog_entradas}
    <h5>{título}</h5>
    <p>{cuerpo}</p>
 {/blog_entradas}

 </body>
</html>
```

Estas variables no son variables de PHP reales, sino representaciones de texto sin formato que le permiten eliminar el PHP de sus plantillas (archivos de vista)

Nota

CodeIgniter no requiere que use esta clase, ya que el uso de **PHP puro** en sus páginas de vista **se ejecuta más rápido,** sin embargo, algunos desarrolladores prefieren usar un motor de plantillas si trabajan con diseñadores que no conocen PHP

Importante

La clase **Template Parser** no es una solución de análisis de plantillas en toda regla, lo hemos mantenido muy ligero a propósito para mantener el máximo rendimiento

Iniciar Template Parser Class

Template Parser Class se inicia con **$this->load->library()**:

```
$this->load->library('parser');
```

Una vez cargado, el objeto esta disponible usando:

```
$this->parser
```

Plantillas de análisis

Puede usar el método **parse()** para analizar o representar plantillas simples, como esta:

```
$datos = array(
  'blog_título'   => 'Mi título del blog',
  'blog_cabecera' => 'Encabezado de mi blog'
);

$this->parser->parse('blog_plantilla', $datos);
```

El primer parámetro contiene el nombre del archivo de vista, en este ejemplo, el archivo se llamará **blog_plantilla.php**, y el segundo parámetro contiene un array asociativo de datos para ser reemplazado en la plantilla, la plantilla contendría dos variables: **{blog_título}** y **{blog_cabecera}**

No hay necesidad de hacer **"echo"** o hacer algo con los datos devueltos por **$this->parser->parse() ,** se pasan automáticamente a la clase de salida para enviarse al navegador, sin embargo, si desea que se devuelvan los datos en lugar de enviarlos a la clase de salida, puede pasar TRUE como tercer parámetro:

```
$string = $this->parser->parse('blog_plantilla', $datos, TRUE);
```

Pares de variables

El código anterior permite reemplazar variables simples. ¿Qué sucede si desea que un bloque completo de variables se repita, con cada iteración que contiene nuevos valores? Considere el ejemplo de plantilla que mostramos en la parte superior de la página:

```
<html>
 <head>
    <title>{blog_título}</title>
 </head>
 <body>
    <h3>{blog_cabecera}</h3>

 {blog_entradas}
    <h5>{título}</h5>
    <p>{cuerpo}</p>
 {/blog_entradas}

 </body>
</html>
```

En el código, verá un par de variables: **{blog_entradas}** datos ... **{/ blog_entradas}**. En un caso como este, el fragmento completo de datos entre estos pares se repetiría varias veces, lo que corresponde al número de filas en el elemento **"blog_entradas"** del array de parámetros

Los pares de variables de análisis se realizan utilizando el código idéntico que se muestra arriba para analizar las variables individuales, excepto que agregará un array multidimensional correspondiente a los datos de los pares de variables

Considere este ejemplo:

```
$this->load->library('parser');

$datos = array(
  'blog_título'   => 'Mi título del blog',
  'blog_cabecera' => 'Encabezado de mi blog',
  'blog_entradas' => array(
       array('título' => 'Título 1', 'cuerpo' => 'Cuerpo 1'),
       array('título' => 'Título 2', 'cuerpo' => 'Cuerpo 2'),
       array('título' => 'Título 3', 'cuerpo' => 'Cuerpo 3'),
       array('título' => 'Título 4', 'cuerpo' => 'Cuerpo 4'),
       array('título' => 'Título 5', 'cuerpo' => 'Cuerpo 5')
   )
);

$this->parser->parse('blog_plantilla', $datos);
```

Si sus **"parejas"** de datos provienen de un resultado de Base de datos, que ya es un array multidimensional, puede simplemente usar el método **result_array()**:

```
$consulta = $this->db->query("SELECT * FROM blog");

$this->load->library('parser');

$datos = array(
  'blog_título'   => 'Mi título del blog',
  'blog_cabecera' => 'Encabezado de mi blog',
  'blog_entradas' => $consulta->result_array()
);

$this->parser->parse('blog_plantilla', $datos);
```

Notas de uso

Si incluye parámetros de sustitución que no están referenciados en su plantilla, se ignoran:

```
$plantilla = 'Hola, {nombre} {apellido}';
$datos = array(
  'title'    => 'Mr',
  'nombre'   => 'Salvador',
  'apellido' => 'Dalí'
);

$this->parser->parse_string($plantilla, $datos);
// Resultado: Hola, Salvador Dalí
```

Si no incluye un parámetro de sustitución al que se hace referencia en su plantilla, la pseudovariable original se muestra en el resultado:

```
$plantilla = 'Hola, {iniciales} {nombre} {apellido}';
$datos = array(
  'title'    => 'Mr',
  'nombre'   => 'Salvador',
  'apellido' => 'Dalí'
);
$this->parser->parse_string($plantilla, $datos);

// Resultado: Hola, {iniciales} Salvador Dalí
```

Si proporciona un parámetro de sustitución de string cuando se espera un array, es decir, para un par de variables, la sustitución se realiza para la etiqueta de par de variables de apertura, pero la etiqueta de par de variables de cierre no se representa correctamente:

```
$plantilla = 'Hola, {nombre} {apellido} ({degrees}{degree} {/degrees})';
$datos = array(
  'degrees'  => 'Mr',
  'nombre'   => 'Salvador',
  'apellido' => 'Dalí',
  'titles'   => array(
      array('degree' => 'BSc'),
      array('degree' => 'PhD')
   )
);

$this->parser->parse_string($plantilla, $datos);

// Resultado: Hola, Salvador Dalí (Mr{degree} {/degrees})
```

Si nombra uno de sus parámetros de sustitución individuales el mismo que el utilizado dentro de un par de variables, los resultados pueden no ser los esperados:

```
$plantilla = 'Hola, {nombre} {apellido} ({degrees}{degree} {/degrees})';
$datos = array(
  'degree'   => 'Mr',
  'nombre'   => 'Salvador',
  'apellido' => 'Dalí',
'degrees'  => array(
      array('degree' => 'BSc'),
      array('degree' => 'PhD')
   )
);
$this->parser->parse_string($plantilla, $datos);

// Resultado: Hola, Salvador Dalí (Mr Mr )
```

Fragmentos de vistas

No tiene que usar pares de variables para obtener el efecto de la iteración en sus vistas, es posible usar un fragmento de vista para lo que estaría dentro de un par variable, y para controlar la iteración en su controlador en lugar de en la vista

Un ejemplo con la iteración controlada en la vista:

```
$plantilla = '<ul>{menuitems}
 <li><a href="{link}">{título}</a></li>
{/menuitems}</ul>';

$datos = array(
 'menuitems' => array(
       array('título' => 'Primer link', 'link' => '/primero'),
       array('título' => 'Segundo link', 'link' => '/segundo'),
 )
);
$this->parser->parse_string($plantilla, $datos);
```

Resultado:

```
<ul>
 <li><a href="/primero">Primer link</a></li>
 <li><a href="/segundo">Segundo link</a></li>
</ul>
```

Un ejemplo con la iteración controlada en el controlador, usando un fragmento de vista:

```
$temp = '';
$plantilla1 = '<li><a href="{link}">{título}</a></li>';

$datos1 = array(
  array('título' => 'Primer link', 'link' => '/primero'),
  array('título' => 'Segundo link', 'link' => '/segundo'),
);

foreach ($datos1 as $menuitem)
{
   $temp .= $this->parser->parse_string($plantilla1, $menuitem, TRUE);
}

$plantilla = '<ul>{menuitems}</ul>';

$datos     = array(
  'menuitems' => $temp
);

$this->parser->parse_string($plantilla, $datos);
```

Resultado:

```
<ul>
 <li><a href="/primero">Primer link</a></li>
 <li><a href="/segundo">Segundo link</a></li>
</ul>
```

class CI_Parser

class CI_Parser

class CI_Parser

parse()

string parse(string $template, array $data [, bool $return = FALSE])

Descripción

Analiza una plantilla en la ruta y variables proporcionadas

Parámetros

template
: Ruta para del archivo de vista

data
: Variables con los datos

return
: Si solo devolverá la plantilla analizada

Valores devueltos

String de plantilla analizada

parse_string()

string parse_string(string $template, array $data [, bool $return = FALSE])

Descripción

Este método funciona exactamente como **parse()**, solo que acepta la plantilla como un string en lugar de cargar un archivo de vista

Parámetros

template
: String con la plantilla

data
: Variables con los datos

return
: Si solo devolverá la plantilla analizada

Valores devueltos

String de plantilla analizada

set_delimiters()

```
void set_delimiters([ string $l = '{ '[, string $r = '}']])
```

Descripción

Establece los delimitadores, apertura y cierre, para una **"etiqueta"** pseudovariable en una plantilla

Parámetros

l

Delimitador izquierdo

r

Delimitador derecho

valores devueltos

void

Security Class

Security Class contiene métodos que lo ayudan a crear una aplicación segura, procesando datos de entrada para seguridad

XSS Filtering - Filtrado de XSS

CodeIgniter viene con un filtro de prevención de Cross Site Scripting, que busca las técnicas más utilizadas en JavaScript u otros tipos de código que intentan apropiarse de las cookies o hacer otras cosas maliciosas, si se encuentra algo no permitido, lo transforma en seguro al convertir los datos en entidades de caracteres

Para filtrar datos a través del filtro XSS, use el método **xss_clean()**:

```
$datos = $this->security->xss_clean($datos);
```

Un segundo parámetro opcional, **is_image**, permite que el método se use para comprobar imágenes en busca de posibles ataques XSS, útil para la incrementar la seguridad en la carga de archivos, cuando este segundo parámetro se establece en TRUE, en lugar de devolver un string modificado, el método devuelve TRUE si la imagen es segura y FALSE si contiene información potencialmente maliciosa que un navegador puede intentar ejecutar:

```
if ($this->security->xss_clean($archivo_de_imagen, TRUE) === FALSE)
{
   // Atención: $archivo_de_imagen falló la prueba XSS
}
```

Importante
Si desea filtrar valores de atributos HTML, use **html_escape()** en su lugar

Cross-site request forgery (CSRF) - Falsificación de solicitudes entre sitios (CSRF)

Puede habilitar la protección CSRF modificando su archivo **application/config/config.php** de la siguiente manera:

```
$config['csrf_protection'] = TRUE;
```

Si usa **Form Helper**, **form_open()** insertará automáticamente un campo **csrf oculto** en sus formularios, de lo contrario, puede usar **get_csrf_token_name()** y **get_csrf_hash()**

```
$csrf = array(
  'name' => $this->security->get_csrf_token_name(),
  'hash' => $this->security->get_csrf_hash()
);
...

<input type="hidden" name="<?=$csrf['name'];?>" value="<?=$csrf['hash'];?>" />
```

Los tokens pueden ser regenerados en cada envío, por defecto, o mantenidos iguales durante toda la vida útil de la cookie CSRF, la regeneración predeterminada de los tokens proporciona una seguridad más estricta, pero puede generar problemas de usabilidad ya que otros tokens se vuelven inválidos (navegación hacia atrás/adelante, múltiples pestañas/ventanas, acciones asíncronas, etc.). Puede modificar este comportamiento editando el siguiente parámetro de configuración:

```
$config['csrf_regenerate'] = TRUE;
```

Los URI seleccionados pueden incluirse en la lista blanca de la protección csrf, por ejemplo, puntos finales API que esperan contenido POSTed externamente, puede agregar estos URI editando el parámetro de configuración **'csrf_exclude_uris'**:

```
$config['csrf_exclude_uris'] = array('api/persona/agregar');
```

Las expresiones regulares también son compatibles (no distingue entre mayúsculas y minúsculas):

```
$config['csrf_exclude_uris'] = array(
  'api/registro/[0-9] +',
  'api/título/[a-z] +'
);
```

class CI_Security

class CI_Security

class CI_Security

entity_decode

string entity_decode(string $str [, string $charset = NULL])

Descripción

Este método funciona de forma muy parecida a la función **html_entity_decode()** nativa de PHP en el modo **ENT_COMPAT**, solo trata de detectar entidades HTML que no terminan en un punto y coma porque algunos navegadores lo permiten, si el parámetro **$charset** se deja vacío, entonces se usará el valor configurado en **$config['charset']**

Parámetros

str
: String de entrada

charset
: Conjunto de caracteres de la string de entrada

Valores devueltos

Cadena decodificada

get_csrf_hash()

string get_csrf_hash()

Descripción

Devuelve el valor hash CSRF, útil en combinación con **get_csrf_token_name()** para crear formularios manualmente o enviar solicitudes AJAX POST válidas

Valores devueltos

Hash CSRF

get_csrf_token_name()

```
string get_csrf_token_name()
```

Descripción

Devuelve el nombre del token CSRF, el valor **$config['csrf_token_name']**

Valores devueltos

Nombre del token CSRF

get_random_bytes()

```
string get_random_bytes(int $length)
```

Descripción

Usado para generar tokens CSRF y XSS, método conveniente para obtener bytes aleatorios adecuados a través de **mcrypt_create_iv( )**, **/dev/urandom** o **openssl_random_pseudo_bytes ( ),** en ese orden, si uno de ellos está disponible

Parámetros

length
: Longitud de salida

Valores devueltos

Una secuencia binaria de bytes aleatorios o FALSE en caso de error

> ***Nota***
>
> No se garantiza que la salida sea criptográficamente segura

sanitize_filename()

```
string sanitize_filename(string $str [, bool $relative_path = FALSE])
```

Descripción

Intenta desinfectar los nombres de los archivos para evitar intentos de cruce de directorios y otras amenazas de seguridad, es útil para los archivos que se suministran a través de la entrada del usuario

Parámetros

str
: Nombre de archivo / ruta

relative_path

Si desea conservar cualquier directorio en la ruta del archivo

Valores devueltos

Nombre/ruta del archivo desinfectado

Ejemplo

```
$filename = $this->security->sanitize_filename($this->input->post('nombre_archivo'));
```

Si es aceptable que la entrada del usuario incluya rutas relativas, por ejemplo a**rchivo/en/alguna/carpeta/aprobada.txt**, puede establecer el segundo parámetro opcional **$relative_path** en TRUE:

```
$filename = $this->security->sanitize_filename($this->input->post('nombre_archivo'), TRUE);
```

xss_clean()

mixed xss_clean(mixed $str [, bool $is_image = FALSE])

Descripción

Intenta eliminar exploits XSS de los datos de entrada y devuelve el string limpio, si el segundo parámetro opcional se establece en verdadero, devolverá TRUE si la imagen es segura de utilizar y FALSE si se detectan datos maliciosos en ella

Parámetros

str

Cadena de entrada o un array de strings

is_image

Opcional, permite que el método se use para comprobar imágenes

Valores devueltos

Datos XSS-clean

Importante

Este método no es adecuado para filtrar valores de atributos HTML **Use html_escape()** para ello

Session Library

Session Library le permite mantener el "estado" de un usuario y realizar un seguimiento de su actividad mientras navega por su sitio

CodeIgniter viene con algunos controladores de almacenamiento de sesión:

- **files** - archivos (predeterminado: el sistema de archivos del sistema)
- **database** - Base de datos
- **redis**
- **memcached**

Además, puede crear sus propios controladores de sesión personalizados basados en otros tipos de almacenamiento, al mismo tiempo que aprovecha las características de la **clase Session**

Iniciar Session Library

Por lo general, las sesiones se ejecutarán de forma global con cada página que se carga, por lo que la clase de sesión debe inicializarse en los constructores de su controlador o el sistema puede cargarla automáticamente, en su mayor parte, la clase de sesión se ejecutará desatendida en segundo plano, por lo que la inicialización de la clase hará que lea, cree y actualice sesiones cuando sea necesario

Session Class se inicia con **$this->load->library()**:

```
$this->load->library('session');
```

Una vez cargada, el objeto esta disponible usando:

```
$this->session
```

Importante

Debido a que la clase **Loader** es instanciada por el controlador base de CodeIgniter, asegúrese de llamar a **parent :: __ construct ()** antes de intentar cargar una biblioteca desde dentro de un constructor de controlador

Cómo funcionan las sesiones?

Cuando se carga una página, la clase de sesión verificará si el navegador del usuario envía una cookie de sesión válida, si no existe una cookie de sesión, o si no coincide con una almacenada en el servidor o ha expirado, se crea y guarda una nueva sesión

Si existe una sesión válida, su información se actualiza, con cada actualización, la ID de sesión puede regenerarse si está configurada para hacerlo

Es importante que comprenda que una vez iniciada, la clase **Session** se ejecuta automáticamente, no hay nada que deba hacer para que ocurra el comportamiento anterior, como verá a continuación, puede trabajar con datos de sesión, pero el proceso de leer, escribir y actualizar una sesión es automático

Nota

En **CLI**, la biblioteca Session se detendrá automáticamente, ya que este es un concepto basado completamente en el protocolo HTTP

Una nota sobre concurrencia

A menos que esté desarrollando un sitio web con gran uso de AJAX, puede omitir esta sección, sin embargo, si lo está y tiene problemas de rendimiento, esta nota es exactamente lo que está buscando

Las sesiones en versiones anteriores de CodeIgniter no implementaban el bloqueo, lo que significaba que dos solicitudes HTTP que utilizaban la misma sesión podían ejecutarse exactamente al mismo tiempo, para usar un término técnico más apropiado, las solicitudes no eran bloqueadas

Sin embargo, las solicitudes de no bloqueo en el contexto de las sesiones significan inseguridad, porque las modificaciones de los datos de la sesión, o la regeneración de ID de sesión, en una solicitud puede interferir con la ejecución de una segunda solicitud concurrente, este detalle fue la raíz de muchos problemas y la razón principal por la cual CodeIgniter 3.0 tiene una biblioteca de sesiones completamente nueva

Por qué le estamos diciendo esto?, debido a que es probable que después de tratar de encontrar el motivo de sus problemas de rendimiento, pueda concluir que el bloqueo es el problema y, por lo tanto, analizar cómo eliminar los bloqueos

NO LO HAGA - Eliminar los bloqueos es incorrecto y le causará más problemas !!!

Bloquear no es el problema, es una solución, su problema es que todavía tiene la sesión abierta, mientras que ya la ha procesado y, por lo tanto, ya no la necesita, entonces, lo que necesita es **cerrar la sesión** para la solicitud actual después de que **ya no la necesite**

Para abreviar, llame a **session_write_close()** una vez que ya no necesite las variables de la sesión

Qué son los datos de sesión?

Los datos de sesión son simplemente un array asociado con un ID de sesión particular (cookie)

Si ha utilizado sesiones en PHP anteriormente, debe estar familiarizado con **$_SESSION** superglobal de PHP, si no, lea **http://php.net/manual/en/reserved.variables.session.php**

CodeIgniter brinda acceso a sus datos de sesión a través de los mismos medios, ya que utiliza el mecanismo de los manejadores de sesión provisto por PHP, el uso de datos de sesión es tan simple como manipular, leer, establecer y deshacer valores, el array **$_SESSION**

Además, CodeIgniter también proporciona dos tipos especiales de datos de sesión que se explican a continuación: **flashdata** y **tempdata**

Nota

En versiones anteriores, los datos de sesión regulares en CodeIgniter se denominaban '**userdata**', tenga esto en cuenta si ese término se usa en otro lugar del manual. La mayor parte está escrita para explicar cómo funcionan los métodos personalizados de **'userdata'**

Recuperar datos de sesión

Cualquier información del conjunto de sesiones está disponible a través del superglobal **$_SESSION**:

```
$_SESSION ['item']
```

O a través del **magic getter**:

```
$this->session->item
```

Y para compatibilidad con versiones anteriores, a través del método **userdata()**:

```
$this->session->userdata('item');
```

Donde **item** es la clave de array correspondiente al elemento que desea buscar.

Por ejemplo, para asignar un elemento **'nombre'** previamente almacenado en la variable **$nom**, lo hará:

```
$nom = $_SESSION['nombre'];

// o:
$nom = $this->session->nombre

// o:

$nom = $this->session->userdata('nombre');
```

Nota

El método **userdata()** devuelve NULL si el elemento al que intenta acceder no existe

Si desea recuperar todos los datos de usuario existentes, simplemente puede omitir la clave del elemento, **magic getter** solo funciona para las propiedades:

```
$_SESSION

// o:

$this->session->userdata();
```

Agregar datos de sesión

Digamos que un usuario en particular inicia sesión en su sitio, una vez autenticado, puede agregar su nombre de usuario y correo electrónico a la sesión, haciendo que esos datos estén disponibles globalmente para usted sin tener que ejecutar una consulta de la Base de datos cuando la necesite

Simplemente puede asignar datos al array **$_SESSION**, como con cualquier otra variable, o como una propiedad de **$this->session**, alternativamente, el viejo método de asignarlo como **"userdata"** también está disponible, puede pasar un array que contiene sus datos nuevos al método **set_userdata()**:

```
$this->session->set_userdata($array);
```

Donde **$array** es un array asociativo que contiene sus nuevos datos

Aquí hay un ejemplo:

```
$array = array (
  'username'  => 'Antonio Gaudí',
  'email'     => 'antonio_gaudi@sagrada-familia.com',
  'logged_in' => TRUE
);
$this->session->set_userdata($array);
```

Si desea agregar un valor a la vez, **set_userdata()** también admite esta sintaxis:

```
$this->session->set_userdata('algun_nombre', 'algun_valor');
```

Si desea verificar que existe un valor de sesión, puede verificarlo con **isset()**:

```
// devuelve FALSE si el elemento 'algun_nombre' no existe o es NULL, TRUE de lo contrario:
isset($_ SESSION['algun_nombre'])
```

o puede llamar a **has_userdata()**:

```
$this->session->has_userdata('algun_nombre');
```

Eliminar datos de sesión

Al igual que con cualquier otra variable, puede deshacer un valor en **$_SESSION** a través de **unset()**:

```
unset($_SESSION['algun_nombre']);
```

Valores múltiples:

```
unset (
  $_SESSION['algun_nombre'],
  $_SESSION ['otro_nombre']
);
```

Además, al igual que **set_userdata()** se puede usar para agregar información a una sesión, **unset_userdata()** se puede usar para eliminarlo, pasando la clave de sesión, por ejemplo, si desea eliminar **'algun_nombre'** de su array de datos de sesión:

```
$this->session->unset_userdata('algun_nombre');
```

Este método también acepta un array de claves de elementos para eliminar:

```
$array_items = array('nombre de usuario', 'correo electrónico');

$this-> session->unset_userdata($array_items);
```

Nota

En versiones anteriores, el método **unset_userdata()** solía aceptar un array asociativo de pares clave => 'valor ficticio', esto ya no es compatible

Flashdata

CodeIgniter admite **"flashdata"**, o datos de sesión que solo están disponibles para la siguiente solicitud, y luego se borran de forma automática, esto puede ser muy útil, especialmente para mensajes informativos, de error o de estado de una sola vez, por ejemplo:

"Registro 2 eliminado"

Se debe tener en cuenta que las variables de flashdata son variables de sesión regulares, solo se marcan de una manera específica bajo la clave **'__ci_vars'** (no lo toque, ha sido advertido)

Para marcar un elemento existente como "**flashdata**":

```
$this->session->mark_as_flash('item');
```

Si desea marcar varios elementos como flashdata, simplemente pase las claves como un array:

```
$this->session->mark_as_flash(array('item', 'item2'));
```

Para agregar flashdata:

```
$_SESSION['item'] = 'valor';
$this->session->mark_as_flash('item');
```

O alternativamente, usando el método **set_flashdata()**:

```
$this->session->set_flashdata('item', 'valor');
```

También puede pasar un array a **set_flashdata()**, de la misma manera que **set_userdata()**

Leer variables de flashdata es lo mismo que leer datos de sesión regulares a través de **$_SESSION**:

```
$_SESSION['item']
```

Importante

El método **userdata() NO** devolverá elementos flashdata

Sin embargo, si quiere estar seguro de que está leyendo **"flashdata"** (y no de otro tipo), también puede usar el método **flashdata()**:

```
$this->session->flashdata('item');
```

O para obtener un array con todos los flashdata, simplemente omita el parámetro clave:

```
$this->session->flashdata();
```

Nota

El método **flashdata()** devuelve NULL si el elemento no se puede encontrar

Si encuentra que necesita preservar una variable de flashdata a través de una solicitud adicional, puede hacerlo utilizando el método **keep_flashdata()**, puede pasar un solo elemento o un array de elementos:

```
$this->session->keep_flashdata('item');
$this->session->keep_flashdata(array('item1', 'item2', 'item3'));
```

Tempdata

CodeIgniter también admite **"tempdata"**, o datos de sesión con un tiempo de caducidad específico, una vez que el valor expira, o la sesión expira o se elimina, el valor se elimina automáticamente

De manera similar a flashdata, las variables de tempdata son variables de sesión regulares que están marcados de una manera específica bajo la clave **'__ci_vars'** (nuevamente, no lo toque)

Para marcar un elemento existente como **"tempdata"**, pase su clave y el tiempo de caducidad (**ien segundos!)** al método **mark_as_temp()**:

```
// 'item' será borrado después de 300 segundos
$this->session->mark_as_temp('item', 300);
```

Puede marcar varios elementos como tempdata de dos maneras, dependiendo de si desea que todos tengan el mismo tiempo de caducidad o no:

```
// Ambos 'item' y 'item2' expirarán después de 300 segundos
$this->session->mark_as_temp(array('item', 'item2'), 300);

// 'item' se borrará después de 300 segundos, mientras que 'elemento 2' lo hará después de solo 240
segundos
$this->session->mark_as_temp(array('item' => 300, 'item2' => 240 ));
```

Para agregar tempdata:

```
$_SESSION['item'] = 'valor';
$this->session->mark_as_temp('item', 300); // Caduca en 5 minutos
```

O alternativamente, usando el método **set_tempdata()**:

```
$this->session->set_tempdata('item', 'valor', 300);
```

También puede pasar un array a **set_tempdata()**:

```
$tempdata = array('newuser' => TRUE, 'mensaje' => 'Gracias por unirse!');
$this->session->set_tempdata($tempdata, NULL, $expire);
```

Nota

Si la caducidad se omite o se establece en 0, se utilizará el valor predeterminado de tiempo de vida de 300 segundos (o 5 minutos)

Para leer una variable tempdata, nuevamente puede acceder a ella a través del array superglobal **$_SESSION**:

```
$ _SESSION ['item']
```

Importante

El método **userdata() NO** devolverá elementos de tempdata

O si quiere estar seguro de que está leyendo **"tempdata"**, y no de otro tipo, también puede usar el método **tempdata()**:

```
$this->session->tempdata('item');
```

Y, por supuesto, si desea recuperar todos los tempda existentes:

```
$this->session->tempdata();
```

Nota

El método **tempdata()** devuelve **NULL** si el elemento no se puede encontrar

Si necesita eliminar un valor de tempdata antes de que caduque, puede desvincularlo directamente del array **$_SESSION**:

```
unset($_SESSION['item']);
```

Sin embargo, esto no eliminará el marcador que hace que este elemento específico sea tempdata, se invalidará en la próxima solicitud HTTP, por lo que si tiene intención de volver a utilizar esa misma clave en la misma solicitud, tendrá que usar **unset_tempdata ()**:

```
$this->session->unset_tempdata('item');
```

Destruyendo una sesión

Para borrar la sesión actual, por ejemplo, durante el cierre de sesión, puede simplemente usar la función **session_destroy()** de PHP o el método **sess_destroy()**, ambos funcionarán exactamente de la misma manera:

```
session_destroy();
// o
$this->session->sess_destroy();
```

Nota

Esta debe ser la última operación relacionada con la sesión que realice durante la misma solicitud, todos los datos de sesión, incluidos flashdata y tempdata, se destruirán de forma permanente y las funciones quedarán inutilizables durante la misma solicitud después de destruir la sesión

Acceder a los metadatos de la sesión

En versiones anteriores de CodeIgniter, el array de datos de sesión incluía cuatro elementos de forma predeterminada: **'session_id'**, **'ip_address'**, **'user_agent'**, **'last_activity'**

Esto se debió a los detalles de cómo funcionaban las sesiones, pero ahora ya no es necesario con nuestra nueva implementación, sin embargo, puede suceder que su aplicación confíe en estos valores, así que aquí hay métodos alternativos para acceder a ellos:

- **session_id**: session_id()
- **ip_address**: $_SERVER['REMOTE_ADDR']
- **user_agent**: $this->input->user_agent() (no utilizado por las sesiones)
- **last_activity**: Depende del almacenamiento, no hay manera directa. ¡Lo siento!

Preferencias de sesión

CodeIgniter generalmente hará que todo salga bien, sin embargo, las sesiones son un componente muy sensible de cualquier aplicación, por lo que se debe realizar una configuración cuidadosa, tómese su tiempo para considerar todas las opciones y sus efectos

Encontrará las siguientes preferencias relacionadas con la sesión en su archivo **application/config/config.php**:

Preferencia	Opciones	Predeterminado	Descripción
sess_driver	files	files/database/redis/memcached/custom	El controlador de almacenamiento de sesión para usar
sess_cookie_name	ci_session	[A-Za-z_-] characters only	El nombre utilizado para la cookie de sesión

sess_expiration	7200 (2 hours)	Time in seconds (integer)	La cantidad de segundos que desea que dure la sesión, si desea una sesión que no expira, hasta que el navegador esté cerrado, establezca el valor en cero: 0
sess_save_path	NULL	No	Especifica la ubicación de almacenamiento, depende del controlador que se utilice
sess_match_ip	FALSE	TRUE/FALSE (boolean)	Si validar la dirección IP del usuario al leer la cookie de sesión, tenga en cuenta que algunos ISP cambian dinámicamente la dirección IP, por lo que si desea una sesión que no expira, probablemente establezca esto en FALSE
sess_time_to_update	300	Time in seconds (integer)	Esta opción controla la frecuencia con la que la clase de sesión se regenerará y creará una nueva ID de sesión, establecerlo en 0 desactiva la regeneración de ID de sesión.
sess_regenerate_destroy	FALSE	TRUE/FALSE (boolean)	Si se destruyen los datos de sesión asociados con el ID de sesión anterior al regenerar automáticamente el ID de sesión, cuando se establece en FALSE, los datos se borran más tarde por el recolector de basura

Nota

Como último recurso, la biblioteca de sesión intentará recuperar las configuraciones de INI relacionadas con la sesión de PHP, así como las configuraciones de CodeIgniter heredadas, como **'sess_expire_on_close'** cuando ninguna de las anteriores esté configurada, sin embargo, nunca debe confiar en este comportamiento, ya que puede causar resultados inesperados o modificarse en el futuro, por favor, configure todo correctamente

Además de los valores anteriores, la cookie y los controladores nativos aplican los siguientes valores de configuración compartidos por las clases **Input** y **Security**:

Preferencia	Descripción	Predeterminado
cookie_domain	''	El dominio para el cual la sesión es aplicable
cookie_path	/	La ruta a la que se aplica la sesión
cookie_secure	FALSE	Si se debe crear la cookie de sesión solo en conexiones encriptadas (HTTPS)

Nota

La configuración **'cookie_httponly'** no tiene ningún efecto en las sesiones, en cambio, el parámetro **HttpOnly** siempre está habilitado, por razones de seguridad, además, la configuración **'cookie_prefix'** se ignora por completo

Controladores de sesión

Como ya se mencionó, la biblioteca de sesión viene con cuatro controladores o motores de almacenamiento, que puede usar:

- **files** - archivos
- **database** - Base de datos
- **redis**
- **memcached**

De forma predeterminada, **Files Driver** se utilizará cuando se inicialice una sesión, porque es la opción más segura y se espera que funcione en todas partes, prácticamente todos los entornos tienen un sistema de archivos

Sin embargo, cualquier otro controlador se puede seleccionar a través de la variable **$config['sess_driver']** en su archivo **application/config/config.php**, si elige hacerlo, tenga en mente, sin embargo, que cada driver tiene diferentes advertencias, así que asegúrese de familiarizarse con ellos, abajo, antes de tomar esa decisión

Además, también puede crear y utilizar controladores personalizados, si los que se proporcionan de forma predeterminada no satisfacen su caso de uso

Nota

En las versiones anteriores de CodeIgniter, la única opción era un "controlador de cookies" diferente y recibimos comentarios negativos al no proporcionar esa opción, si bien escuchamos los comentarios de la comunidad, queremos advertirle que se eliminó porque no es seguro y le aconsejamos que **NO** intente **replicarlo** a través de un controlador personalizado

Controlador file

El controlador **'files'** usa su sistema de archivos para almacenar datos de sesión

Se puede decir con seguridad que funciona exactamente como la implementación de la sesión predeterminada de PHP, pero en caso de que sea un detalle importante para usted, tenga en cuenta que, de hecho, no es el mismo código y tiene algunas limitaciones y ventajas

Para ser más específico, no es compatible con los formatos de nivel y modo de directorio de PHP utilizados en **session.save_path** , y tiene la mayoría de las opciones codificadas por seguridad, en cambio, solo se admiten rutas absolutas para **$config['sess_save_path']**

Otra cosa importante que debe saber es asegurarse de no utilizar un directorio público o compartido para almacenar sus archivos de sesión, asegúrese de que solo usted tenga acceso para ver los contenidos de su directorio **sess_save_path,** de lo contrario, cualquiera que pueda hacer eso, también puede robar cualquiera de las sesiones actuales, ataque conocido como de **"session fixation"**

En los sistemas operativos tipo UNIX, esto generalmente se logra configurando los permisos del modo 0700 en ese directorio a través del comando chmod, que permite que solo el propietario del directorio realice operaciones de lectura y escritura en él, pero tenga cuidado porque el usuario del sistema que ejecuta el script generalmente no es el suyo, sino algo así como **'www-data'**, por lo que solo configurar esos permisos probablemente romperá su aplicación

En cambio, debería hacer algo como esto, dependiendo de su entorno:

```
mkdir /<ruta al directorio de la aplicación>/sessions/
chmod 0700 /<ruta de acceso al directorio de la aplicación>/sessions/
chown www-data /<ruta al directorio de su aplicación>/sessions/
```

Bonus Tip

Alguno de ustedes probablemente optará por elegir otro controlador de sesión porque el almacenamiento de archivos suele ser más lento, esto solo es verdad a medias

Una prueba muy básica probablemente le engañe y crea que una Base de datos SQL es más rápida, pero en el 99% de los casos, esto solo es cierto, mientras que solo tiene algunas sesiones activas, a medida que aumentan las sesiones, aumenta la carga del servidor, que es el momento que importa, el sistema de archivos superará sistemáticamente a casi todas las configuraciones de Bases de datos relacionales

Además, si el rendimiento es su única preocupación, es posible que desee considerar el uso de **tmpfs**, advertencia: es un recurso externo, que puede hacer que sus sesiones sean extremadamente rápidas

Controlador database

El controlador **'database'** utiliza una Base de datos relacional como MySQL o PostgreSQL para almacenar sesiones, esta es una opción popular entre muchos usuarios, ya que permite al desarrollador acceder fácilmente a los datos de la sesión dentro de una aplicación, es solo otra tabla en su Base de datos

Sin embargo, hay algunas condiciones que deben cumplirse:

- Solo se puede usar su conexión de Base de datos predeterminada, o la que acceda como **$this->db** desde sus controladores
- Debe tener Query Builder habilitado
- NO puedes usar una conexión persistente
- NO puede usar una conexión con la configuración de caché activada

Para utilizar el controlador de sesión '**database**', debe crear la tabla y luego establecerla como su valor de **$config['sess_save_path']**

Por ejemplo, si desea usar **'ci_sessions'** como el nombre de su tabla, debería hacer esto:

```
$config['sess_driver'] = 'database';
$config['sess_save_path'] = 'ci_sessions';
```

Nota

Si ha actualizado desde una versión anterior de CodeIgniter y no tiene configurada **'sess_save_path'**, entonces la biblioteca de sesión buscará la antigua configuración **'sess_table_name'** y la usará en su lugar, no confíe en este comportamiento, ya que se eliminará en el futuro

Y luego, por supuesto, cree la tabla en la Base de datos ...,

Para MySQL:

```
CREATE TABLE IF NOT EXISTS 'ci_sessions' (
  'id' varchar(128) NOT NULL,
  'ip_address' varchar(45) NOT NULL,
  'timestamp' int(10) unsigned DEFAULT 0 NOT NULL,
  'data' blob NOT NULL,
  KEY 'ci_sessions_timestamp' ('timestamp')
);
```

Para PostgreSQL:

```
CREATE TABLE "ci_sessions" (
  "id" varchar(128) NOT NULL,
  "ip_address" varchar(45) NOT NULL,
  "timestamp" bigint DEFAULT 0 NOT NULL,
  "data" text DEFAULT '' NOT NULL
);

CREATE INDEX "ci_sessions_timestamp" ON "ci_sessions" ("timestamp");
```

También necesitará agregar una **CLAVE PRIMARIA** dependiendo de su configuración **'sess_match_ip', l**os siguientes ejemplos funcionan tanto en MySQL como en PostgreSQL:

```
// Cuando sess_match_ip = TRUE
ALTER TABLE ci_sessions ADD PRIMARY KEY (id, ip_address);

// Cuando sess_match_ip = FALSE
ALTER TABLE ci_sessions ADD PRIMARY KEY (id);

// Para eliminar una clave primaria creada previamente (usar al cambiar la configuración)
ALTER TABLE ci_sessions DROP PRIMARY KEY;
```

Importante

Solo las Bases de datos MySQL y PostgreSQL son oficialmente compatibles, debido a la falta de mecanismos de bloqueo en otros gestores, el uso de sesiones sin bloqueo puede causar todo tipo de problemas, especialmente con el uso intensivo de AJAX, y no admitiremos dichos casos. Utilice **session_write_close()** después de que haya terminado
de procesar los datos de la sesión si tiene problemas de rendimiento

Controlador Redis

Nota

Dado que Redis no tiene un mecanismo de bloqueo, los bloqueos para este controlador se emulan mediante un valor separado que se mantiene durante un máximo de 300 segundos

Redis es un motor de almacenamiento que normalmente se utiliza para el almacenamiento en caché y es popular debido a su alto rendimiento, que también es probablemente su razón para usar el controlador de sesión **'redis'**

La desventaja es que no es tan omnipresente como las Bases de datos relacionales y requiere instalar la extensión de PHP phpredis en su sistema, y no viene incluida con PHP, lo más probable es que solo use el controlador **'redis'** solo si ya está familiarizado con **Redis** y lo está usando para otros fines

Al igual que con los controladores **'files'** y **'database'**, también debe configurar la ubicación de almacenamiento para sus sesiones a través de la configuración **$config['sess_save_path'],** el formato aquí es un poco diferente y complicado al mismo tiempo, lo explica mejor el archivo README de la extensión phpredis, así que simplemente lo vincularemos a él:

```
https://github.com/phpredis/phpredis#php-session-handler
```

Advertencia

La biblioteca de sesión de CodeIgniter NO usa la sesión real **'redis'**. Save_handler. Tome nota solamente del formato de ruta en el enlace de arriba

Sin embargo, para el caso más común, un simple host:port debería ser suficiente:

```
$config['sess_driver'] = 'redis';
$config['sess_save_path'] = 'tcp: // localhost: 6379';
```

Controlador Memcached

Nota

Dado que Memcache no tiene un mecanismo de bloqueo, los bloqueos para este controlador se emulan mediante un valor separado que se mantiene un máximo de 300 segundos

El controlador **'memcached'** es muy similar al **'redis'** en todas sus propiedades, excepto tal vez por la disponibilidad, porque la extensión Memcached de PHP se distribuye a través de PECL y algunas distribuciones de Linux lo tienen disponible como un paquete fácil de instalar

Aparte de eso, y sin ningún prejuicio intencional hacia Redis, no hay mucho más que decir sobre Memcached, también es un producto popular que generalmente se usa para el almacenamiento en caché y es famoso por su velocidad

Sin embargo, vale la pena señalar que la única garantía dada por Memcached es que el valor de configuración X caduque después de Y segundos dará como resultado que se elimine después de que hayan transcurrido Y segundos, pero no

necesariamente que no caduque antes de ese momento, esto ocurre muy raramente, pero se debe considerar ya que puede provocar la pérdida de sesiones

El formato **$config['sess_save_path']** es bastante sencillo aquí, solo necesita el par **host:port**

```
$config['sess_driver'] = 'memcached';
$config['sess_save_path'] = 'localhost:11211';
```

Bonus Tip

La configuración Multi-server con un parámetro opcional como el tercer valor separado por dos puntos (:peso) también es compatible, pero debemos tener en cuenta que no hemos probado si eso es confiable

Si desea experimentar con esta característica, **bajo su propio riesgo**, simplemente separe las múltiples rutas de acceso del servidor con comas:

```
// localhost tendrá mayor prioridad (5) aquí,
// en comparación con 192.0.2.1 con un peso de 1.
$config['sess_save_path'] = 'localhost:11211:5,192.0.2.1:11211:1';
```

Controladores personalizados

También puede crear sus propios controladores de sesión personalizados, sin embargo, tenga en cuenta que, por lo general, esta no es una tarea fácil, ya que se necesita mucho conocimiento para hacerlo correctamente

Necesita saber no solo cómo funcionan las sesiones en general, sino también cómo funcionan específicamente en PHP, cómo funciona el mecanismo de almacenamiento subyacente, cómo manejar la concurrencia, evitar interbloqueos, pero NO a través de la falta de bloqueos, y por último, pero no menos importante: cómo para manejar los posibles problemas de seguridad, lo cual está lejos de ser trivial

Para resumir, si no sabes cómo hacerlo en PHP sin formato, tampoco deberías tratar de hacerlo en CodeIgniter. Ha sido advertido

Si solo desea agregar alguna funcionalidad adicional a sus sesiones, simplemente amplíe la clase de sesión base, que es mucho más fácil, lea **Cómo crear bibliotecas** para aprender cómo hacerlo

Ahora, al grano, hay tres reglas generales que debe seguir al crear un controlador de sesión para CodeIgniter:

- Coloque el archivo de su controlador en **application/libraries/session/drivers/** y siga las convenciones de nomenclatura utilizadas por la clase **Session**

  Por ejemplo, si creara un controlador **'ficticio'**, tendría un nombre de clase **Session_ficticio_driver**, que está declarado en **application/libraries/session/drivers/ Session_ficticio_driver.php**

- Extienda la clase **CI_Session_driver**, esta es solo una clase básica con algunos métodos internos de ayuda, también se puede ampliar como cualquier otra biblioteca, si realmente necesita hacerlo, pero no vamos a explicar cómo ... si está familiarizado con el funcionamiento de las extensiones/creación de clases en CodeIgniter, entonces ya sabe cómo hacerlo, si no, bueno, no debería hacerlo en primer lugar

- Implemente la interfaz **SessionHandlerInterface** ver **http://php.net/sessionhandlerinterface**

> ***Nota***
>
> Puede observar que **SessionHandlerInterface** es proporcionado por PHP desde la versión 5.4.0. CodeIgniter declarará automáticamente la misma interfaz si está ejecutando una versión anterior de PHP

El enlace explicará por qué y cómo

Por lo tanto, en función de nuestro ejemplo de controlador **'ficticio'** anterior, terminaría con algo como esto:

```
// application/libraries/Session/drivers/Session_ficticio_driver.php:

class CI_Session_ficticio_driver extends CI_Session_driver implements SessionHandlerInterface
{

  public function __construct(&$params)
  {
     // NO OLVIDE esto
     parent::__construct($params);
     // Configuración y otras inicializaciones
  }

  public function open($save_path, $name)
  {
     // Inicializar el mecanismo de almacenamiento (conexión)
  }

  public function read($session_id)
  {
     // Leer los datos de la sesión (si existen), gestionar bloqueos
  }

  public function write($session_id, $session_data)
  {
     // Crear / actualizar datos de sesión (podría no existir!)
  }
```

```
  public function close()
  {
     // Liverar bloqueos , conexiones / flujos / etc.//
  }

  public function destroy($session_id)
  {
     // Llama al método close () y destruye los datos de la sesión actual (el orden puede variar)
  }

  public function gc($maxlifetime)
  {
     // Borrar datos para las sesiones caducadas
  }

}
```

Si ha hecho todo correctamente, ahora puede establecer el valor de configuración de **sess_driver** en **'ficticio'** y usar su propio controlador

Felicitaciones !

class CI_Session

class CI_Session

class CI_Session

all_userdata()

array all_userdata()

> ***Nota***
>
> Este método está **ABANDONADO**. Use **userdata()** sin parámetros en su lugar

flashdata()

mixed flashdata([mixed $key = NULL])

> ***Nota***
>
> Este es un método heredado que se mantiene solo por compatibilidad con versiones anteriores de aplicaciones anteriores. Debe acceder directamente a **$_SESSION** en su lugar

__get()

mixed __get(string $key)

Descripción

Un método mágico que le permite usar **$this->session->item** en lugar de **$_SESSION['item']**, si eso es lo que prefiere, también devolverá el ID de sesión llamando a **session_id()** si intenta acceder a **$this->session->session_id**

Parámetros

key
Clave de elemento de sesión

Valores devueltos

El elemento de datos de la sesión solicitada, o NULL si no existe

get_flash_keys()

array get_flash_keys()

Descripción

Obtiene una lista de todos los elementos **$_SESSION** que se han marcado como **'flashdata'**

Valores devueltos

Array que contiene las claves de todos los elementos **'flashdata'**

get_temp_keys()

array get_temp_keys()

Descripción

Obtiene una lista de todos los elementos **$_SESSION** que se han marcado como **'tempdata'**

Valores devueltos

Un array que contiene las claves de todos los elementos **'tempdata'**

&get_userdata()

array &get_userdata()

> ***Nota***
> Este es un método heredado que se mantiene solo por compatibilidad con versiones anteriores de aplicaciones anteriores

has_userdata()

bool has_userdata(string $key)

> ***Nota***
> Este es un método heredado que se mantiene solo por compatibilidad con versiones anteriores de aplicaciones anteriores, es solo un alias para **isset($_SESSION[$ key])** , úselo en su lugar

keep_flashdata()

bool keep_flashdata(mixed $key)

Nota

Este es un método heredado que se mantiene solo por compatibilidad con versiones anteriores, es un alias para el método **mark_as_flash()**

mark_as_flash()

bool mark_as_flash(mixed $key)

Descripción

Marca una clave de elemento **$_SESSION** (o varias) como **'flashdata'**

Parámetros

key

Clave para marcar como flashdata, o un array de múltiples claves

Valores devueltos

TRUE en el éxito, FALSE si falla

mark_as_temp()

bool mark_as_temp(mixed $key [, int $ttl = 300])

Descripción

Marca una clave de elemento **$_SESSION** (o varias) como **'tempdata'**

Parámetros

key

Clave para marcar como tempdata, o un array de múltiples claves

ttl

Valor de tiempo de vida, en segundos, para los tempdatos

Valores devueltos

TRUE en el éxito, FALSE si falla

sess_destroy()

void sess_destroy()

Nota

Este método es solo un alias para la función **session_destroy()** nativa de PHP

Descripción

Destruye la sesión actual

Nota

Esta debe ser la última función relacionada con la sesión a la que llama, todos los datos de la sesión se perderán después de hacer eso

valores devueltos

void

sess_regenerate()

```
void sess_regenerate([ bool $destroy = FALSE])
```

Nota

Este método es solo un alias para la función nativa **session_regenerate_id()** de PHP

Descripción

Vuelve a generar la ID de la sesión, destruyendo opcionalmente los datos de la sesión actual

Parámetros

destroy

Si se destruyen los datos de la sesión

valores devueltos

void

_set()

```
void  __set(string $key, mixed $value)
```

Descripción

Un método mágico que le permite asignar elementos a **$_SESSION** accediendo a ellos como **$this->session->propiedad de sesión**

Parámetros

key

Clave de elemento de sesión

value

Valor para asignar a la clave del elemento de sesión

valores devueltos

void

Ejemplo

```
$this->session->foo ='milú';

// Resultado:
// $_SESSION['foo'] = 'milú';
```

set_flashdata()

```
void set_flashdata(mixed $data [, mixed $value = NULL])
```

> **Nota**
>
> Este es un método heredado que se mantiene solo por compatibilidad con versiones anteriores

set_tempdata()

```
void set_tempdata(mixed $data [, mixed $value = NULL, int $ttl = 300])
```

> **Nota**
>
> Este es un método heredado que se mantiene solo por compatibilidad con versiones anteriores de aplicaciones anteriores

set_userdata()

```
void  set_userdata(mixed $data [, mixed $value = NULL])
```

> **Nota**
>
> Este es un método heredado que se mantiene solo por compatibilidad con versiones anteriores de aplicaciones anteriores

tempdata()

```
mixed tempdata([ mixed $key = NULL])
```

> **Nota**
>
> Este es un método heredado que se mantiene solo por compatibilidad con versiones anteriores de aplicaciones anteriores, debe acceder directamente a **$_SESSION** en su lugar

unmark_flash()

```
void unmark_flash(mixed $key)
```

Descripción

Desmarca una clave de elemento **$_SESSION** (o varias) como **'flashdata'**

Parámetros

key

Clave que se debe anular como flashdata, o un array de múltiples claves

valores devueltos

void

unmark_temp()

```
void unmark_temp(mixed $key)
```

Descripción

Desmarca una clave de elemento **$_SESSION** (o varias) como **'tempdata'**

Parámetros

key

Clave que se debe eliminar como tempdata, o un array de múltiples claves

valores devueltos

void

unset_userdata()

```
void unset_userdata(mixed $key)
```

> ***Nota***
>
> Este es un método heredado que se mantiene solo por compatibilidad con versiones anteriores de aplicaciones anteriores. Es solo un alias para **unset($_SESSION[$key])** , úselo en su lugar

userdata()

```
mixed userdata([ mixed $key = NULL])
```

> ***Nota***
>
> Este es un método heredado que se mantiene solo por compatibilidad con versiones anteriores de aplicaciones anteriores, debe acceder directamente a **$_SESSION** en su lugar

HTML Table Class

HTML Table Class proporciona métodos que le permiten generar automáticamente tablas HTML a partir de arrays o conjuntos de resultados de Bases de datos

Iniciar HTML Table Class

HTML Table Class se inicia con **$this->load->library()**:

```
$this->load->library('table');
```

Una vez cargada, el objeto esta disponible usando:

```
$this->table
```

Ejemplos

Aquí hay un ejemplo que muestra cómo puede crear una tabla a partir de un array multidimensional, tenga en cuenta que el primer índice de array se convertirá en el encabezado de tabla, o puede establecer sus propios encabezados utilizando el método **set_heading()** descrito a continuación:

```
$this->load->library('table');

$datos = array(
  array('Pintor', 'Obra', 'Fecha'),
  array('Diego Velázquez', 'Vieja friendo huevos','1619'),
  array('Salvador Dalí', 'Relojes blandos','1931'),
  array('Francisco de Zurbarán', 'San Francisco de pie','1633')
);

echo $this->table->generate($datos);
```

Aquí hay un ejemplo de una tabla creada a partir de un resultado de consulta de Base de datos, la clase de tabla genera automáticamente los encabezados en función de los nombres de tabla, o puede establecer sus propios encabezados utilizando el método **set_heading()** descrito en la referencia de clase:

```
$this->load->library('table');

$consulta = $this->db->query('SELECT * FROM mi_tabla');

echo$this->table->generate($consulta);
```

Aquí hay un ejemplo que muestra cómo puede crear una tabla usando parámetros discretos:

```
$this->load->library('table');

$this->table->set_heading('Pintor', 'Obra', 'Fecha');
```

```
$this-> table-> add_row('Diego Velázquez', 'Vieja friendo huevos','1619');
$this-> table-> add_row('Salvador Dalí', 'Relojes blandos','1931');
$this-> table-> add_row('Francisco de Zurbarán', 'San Francisco de pie','1633');

echo $this->table->generate();
```

Aquí está el mismo ejemplo, excepto que en lugar de parámetros individuales, se usan matrices:

```
$this->load->library('table');

$this->table->set_heading(array('Pintor', 'Obra', 'Fecha'));

$this->table->add_row(array('Diego Velázquez', 'Vieja friendo huevos','1619'));
$this->table->add_row(array('Salvador Dalí', 'Relojes blandos','1931'));
$this->table->add_row(array('Francisco de Zurbarán', 'San Francisco de pie','1633'));

echo $this->table->generate ();
```

Cambiando el aspecto de su tabla

La clase de tabla le permite establecer una plantilla de tabla con la que puede especificar el diseño, aquí está el prototipo de la plantilla:

```
$template = array(
  'table_open'         => '<table border="0" cellpadding="4" cellspacing="0">',

  'thead_open'         => '<thead>',
  'thead_close'        => '</thead>',

  'heading_row_start'  => '<tr>',
  'heading_row_end'    => '</tr>',
  'heading_cell_start' => '<th>',
  'heading_cell_end'   => '</th>',

  'tbody_open'         => '<tbody>',
  'tbody_close'        => '</tbody>',

  'row_start'          => '<tr>',
  'row_end'            => '</tr>',
  'cell_start'         => '<td>',
  'cell_end'           => '</td>',

  'row_alt_start'      => '<tr>',
  'row_alt_end'        => '</tr>',
  'cell_alt_start'     => '<td>',
  'cell_alt_end'       => '</td>',
```

```
  'table_close'        => '</table>'
);

$this->table->set_template($template);
```

Nota

Notará que hay dos conjuntos de bloques de **'fila'** en la plantilla, esto le permite crear colores alternativos de fila o elementos de diseño que se alternan con cada iteración de los datos de la fila

NO es necesario que envíe una plantilla completa, si solo necesita cambiar partes del diseño, simplemente puede enviar esos elementos

En este ejemplo, solo se está cambiando la etiqueta de apertura de la tabla:

```
$plantilla = array(
  'table_open' => '<table border="1" cellpadding="2" cellspacing="1" class="mytable">'
);

$this->table->set_template($plantilla);
```

También puede establecer valores predeterminados en un archivo de configuración

class CI_Table

class CI_Table

class CI_Table

$function

$function = NULL

Le permite especificar una función PHP nativa o un objeto de array de función válida para aplicar a todos los datos de celda

Ejemplo

```
$this->load->library('table');

$this->table->set_heading('Pintor', 'Obra', 'Fecha');
$this->table->add_row('Salvador Dalí', '<strong>Relojes blandos</strong>','1931');

$this->table->function = 'htmlspecialchars';
echo $this->table->generate();
```

En el ejemplo anterior, todos los datos de la celda se ejecutarán a través de la función **htmlspecialchars()** de PHP, lo que da como resultado:

```
<td>Salvador Dalí</td><td>&lt;strong&gt;Relojes blandos&lt;/strong&gt;</td><td>1931</td>
```

add_row()

object add_row([mixed $args = array() [, ...]])

Descripción

Le permite agregar una fila a su tabla, puede enviar un array o varias strings

Parámetros

args

Una array o varias strings que contienen los valores de la fila

Valores devueltos

Instancia de CI_Table (method chaining)

Ejemplo

```
$this->table->add_row('Gaudí', 'Jujol', 'Domènech i Montaner');
$this->table->add_row(array('Gaudí', 'Jujol', 'Domènech i Montaner'));
```

Si desea establecer los atributos de etiqueta de una celda individual, puede usar un array asociativo para esa celda, los datos de clave asociativos definen los datos de la celda, cualquier otro par **clave =>valor** se agrega como **clave = atributos** a la etiqueta:

```
$celda = array('data' => 'Gaudí', 'class' => 'highlight', 'colspan' => 2);
$this->table->add_row($celda, 'Jujol', 'Domènech i Montaner');

// Produce :
// <td class='highlight' colspan='2'>Gaudí</td><td>Jujol</td><td>Domènech i Montaner</td>
```

clear()

object clear()

Descripción

Le permite borrar el encabezado de la tabla y los datos de la fila, si necesita mostrar varias tablas con datos diferentes, debe llamar a este método después de que se haya generado cada tabla para borrar la información de la tabla anterior

Valores devueltos

Instancia CI_Table (method chaining)

Ejemplo

```
$this->load->library('table');

$this->table->set_heading('Obra','Arquitecto', 'Fecha');
$this->table->add_row('Sagrada Familia,'Gaudí', '1883');
$this->table->add_row('Hospital de San Pablo', 'Domènech i Montaner', '1902');
```

```
$this->table->add_row('Trencadís', 'Jujol', '1906');

echo $this->table->generate();

$this->table->clear();

$this->table->set_heading('Obra', 'Escritor', 'Publicación');
$this->table->add_row('El Buscón', 'Quevedo', '1626');
$this->table->add_row('El Quijote', 'Cervantes', '1605');
$this->table->add_row('El proceso','Kafka', '1925');

echo $this->table->generate();
```

generate()

string generate([mixed $table_data = NULL])

Descripción

Devuelve un string que contiene la tabla generada, acepta un parámetro opcional que puede ser un array o un objeto de resultado de Base de datos

Parámetros

table_data
Datos para llenar las filas de la tabla

Valores devueltos

Tabla HTML

make_columns()

array make_columns([array $array = array() [, int $col_limit = 0]])

Descripción

Este método toma un array unidimensional como entrada y crea una array multidimensional con una profundidad igual al número de columnas deseado, esto permite que una única array con muchos elementos se muestre en una tabla que tiene un conteo de columnas fijo

Parámetros

array
Un array que contiene datos de múltiples filas
col_limit
Número de columnas en la tabla

Valores devueltos

Un array de columnas de tabla HTML

Ejemplo

```
$lista = array('uno', 'dos', 'tres', 'cuatro', 'cinco', 'seis', 'siete', 'ocho', 'nueve', 'diez', 'once',
'doce');

$nueva_lista = $this->table->make_columns($lista, 3);

$this->table->generate($nueva_lista);

// Genera una tabla con este prototipo
<table border="0" cellpadding="4" cellspacing="0">
 <tr>
   <td>uno</td><td>dos</td><td>tres</td>
 </tr><tr>
   <td>cuatro</td><td>cinco</td><td>seis</td>
 </tr><tr>
   <td>siete</td><td>ocho</td><td>nueve</td>
 </tr><tr>
   <td>diez</td><td>once</td><td>doce</td></tr>
</table>
```

set_caption()

object set_caption(string $caption)

Descripción

Permite agregar un título a la tabla

Parámetros

caption

Título de la tabla

Valores devueltos

Instancia de CI_Table (method chaining)

Ejemplo

```
$this->table->set_caption('Pintores');
```

set_empty()

object set_empty(mixed $value)

Descripción

Le permite establecer un valor predeterminado para usar en cualquier celda de la tabla que esté vacía

Parámetros

value

Valor para poner en celdas vacías

Valores devueltos

Instancia de CI_Table (method chaining)

Ejemplo

Establecer un espacio sin ruptura:

```
$this->table->set_empty(" ");
```

set_heading()

object set_heading([mixed $args = array() [, ...]])

Descripción

Le permite configurar el encabezado de la tabla, puede enviar un array o varias strings

Parámetros

args

Una array o varias strings que contienen los títulos de columna de la tabla

Valores devueltos

Instancia de CI_Table (method chaining)

Ejemplo

```
$this->table->set_heading('Pintor', 'Obra', 'Fecha');

$this->table->set_heading(array('Pintor', 'Obra', 'Fecha'));
```

set_template()

bool set_template(array $template)

Descripción

Le permite configurar su plantilla, puede enviar una plantilla completa o parcial

Parámetros

template

Un array asociativa que contiene valores de plantilla

Valores devueltos

TRUE en el éxito, FALSE si falla

Ejemplo

```
$plantilla = array(
 'table_open'  => '<table border="1" cellpadding="2" cellspacing="1" class="mytable">'
);

$this->table->set_template($plantilla);
```

Trackback Class

Trackback Class proporciona funciones que le permiten enviar y recibir datos de Trackback

Iniciar Trackback Class

Trackback Class se inicia con **$this->load->library()**:

```
$this->load->library('trackback');
```

Una vez cargada, el objeto esta disponible usando:

```
$this->trackback
```

Envío de referencias

Se puede enviar un Trackback desde cualquiera de las funciones de su controlador utilizando un código similar a este:

```
$this->load->library('trackback');

$tb_data = array(
  'ping_url'  => 'http://ejemplo.com/trackback/456',
  'url'       => 'http://www.mi-ejemplo.com/blog/entrada/123',
  'title'     => 'Título de mi entrada',
  'excerpt'   => 'Contenido de la entrada.',
  'blog_name' => 'El nombre de mi blog',
  'charset'   => 'utf-8'
);

if ( ! $this->trackback->send($tb_data))
{
   echo $this->trackback->display_errors();
}
else
{
   echo 'Se envió el Trackback!';
}
```

Descripción del array de datos:

- **ping_url** - URL del sitio al que envía el Trackback, puede enviar referencias a múltiples URL separando cada URL con una coma
- **url** - URL de SU sitio donde se puede ver la entrada del weblog
- **title** - Título de su entrada de blog
- **excerpt** - Contenido de su entrada de blog

- **blog_name** - Nombre de su weblog
- **charset** - Codificación de caracteres con la que se escribe en su weblog, si se omite, se usará UTF-8

Nota
La clase Trackback enviará automáticamente solo los primeros 500 caracteres de su entrada, también quitará todo el HTML

El método de envío Trackback devuelve TRUE/FALSE en caso de éxito o error, si falla, puede recuperar el mensaje de error usando:

```
$this->trackback->display_errors();
```

Recibir Trackbacks

Para poder recibir Trackbacks debe crear un weblog, **si** todavía **no tiene un blog**, **no tiene sentido continuar**

Recibir Trackbacks es un poco más complejo que enviarlos, solamente porque necesitará una tabla de Base de datos donde almacenarlos y necesitará validar los datos del trackback entrante, se le recomienda implementar un proceso de validación exhaustivo para evitar el correo basura y los datos duplicados, también puede desear limitar la cantidad de Trackbacks que permite desde una IP en particular en un lapso de tiempo determinado para reducir aún más el correo basura, el proceso de recibir un Trackback es bastante simple, la validación es lo que se lleva el mayor esfuerzo

su Ping URL

Para aceptar Trackbacks debe mostrar una URL de Trackback al lado de cada una de las entradas de su weblog, esta será la URL que las personas utilizarán para enviarle Trackbacks, nos referiremos a esto como su **'Ping URL'**

Su Ping URL debe apuntar a una función de controlador donde se encuentra su código de recepción de Trackback, y la URL debe contener el número de ID para cada entrada en particular, de modo que cuando se reciba la Trackback podrá asociarla con una entrada en particular

Por ejemplo, si su clase de controlador se llama **Trackback**, y la función de recepción se llama recibir, sus Ping URL se verán así:

```
http://ejemplo.com/index.php/trackback/recibir/entrada_id
```

Donde **entrada_id** representa el número de identificación individual para cada una de sus entradas.

Crear una tabla de seguimiento

Para poder recibir Trackbacks, debe crear una tabla donde almacenarlos. Aquí hay un prototipo básico para dicha tabla:

```
CREATE TABLE trackbacks (
  tb_id int(10) unsigned NOT NULL auto_increment,
  entry_id int(10) unsigned NOT NULL default 0,
  url varchar(200) NOT NULL,
```

```
  title varchar(100) NOT NULL,
  excerpt text NOT NULL,
  blog_name varchar(100) NOT NULL,
  tb_date int(10) NOT NULL,
  ip_address varchar(45) NOT NULL,
  PRIMARY KEY `tb_id` (`tb_id`),
  KEY `entry_id` (`entry_id`)
);
```

La especificación de Trackback solo requiere que se envíen cuatro piezas de información en un Trackback, url, título, extracto, nombre de blog, pero para que los datos sean más útiles, hemos agregado algunos campos más en el esquema de la tabla anterior, fecha, dirección IP , etc.

Procesando un Trackback

Aquí hay un ejemplo que muestra cómo recibirá y procesará un Trackback, el siguiente código está destinado a utilizarse dentro de la función del controlador donde espera recibir Trackbacks.:

```
$this->load->library('trackback');
$this->load->database();

if ($this->uri->segment(3) == FALSE)
{
   $this->trackback->send_error('No se puede determinar el ID de la entrada');
}

if ( ! $this->trackback->receive())
{
   $this->trackback->send_error('El Trackback no contiene datos válidos');
}

$datos = array(
  'tb_id'      => '',
  'entry_id'   => $this->uri->segment(3),
  'url'        => $this->trackback->data('url'),
  'title'      => $this->trackback->data('title'),
  'excerpt'    => $this->trackback->data('excerpt'),
  'blog_name'  => $this->trackback->data('blog_name'),
  'tb_date'    => time(),
  'ip_address' => $this->input->ip_address()
);

$sql = $this->db->insert_string('trackbacks', $datos);
$this->db->query($sql);

$this->trackback->send_success();
```

Notas:

El número de identificación de entrada se espera en el tercer segmento de su URL, esto se basa en el ejemplo de URI que dimos anteriormente:

```
http://ejemplo.com/index.php/trackback/recibir/entrada_id
```

Observe que **entrada_id** está en el tercer segmento de URI, y se puede recuperar usando:

```
$this->uri->segment(3);
```

En nuestro código de recepción de Trackback anterior, si falta el tercer segmento, emitiremos un error, sin una identificación de entrada válida, no hay razón para continuar

El método **$this->trackback->receive()** examina los datos entrantes y se asegura de que contenga los cuatro datos necesarios, **url, title, excerpt, blog_name**, devuelve TRUE en caso de éxito y FALSE en caso de error, si falla, emitirá un mensaje de error

Los datos de Trackback entrantes se pueden recuperar usando este método:

```
$this->trackback->data('item')
```

Donde item representa uno de los cuatro datos: **url, title, excerpt,** o **blog_name**

Si los datos de Trackback se reciben con éxito, emitirá un mensaje de éxito utilizando:

```
$this->trackback->send_success();
```

Nota
El código anterior no contiene validación de datos, que se le recomienda agregarla

class CI_Trackback

class CI_Trackback

class CI_Trackback

$data

$data = array('url' => '', 'title' => '', 'excerpt' => '', 'blog_name' => '', 'charset' => '')

Array de datos para el Trackback

$convert_ascii

$convert_ascii = TRUE

Si se deben convertir los caracteres ASCII y MS Word a entidades HTML

convert_ascii()

string convert_ascii(string $str)

Descripción

Convierte texto ASCII y caracteres especiales de MS Word en entidades HTML

Parámetros

str

String de entrada

Valores devueltos

String convertida

convert_xml()

string convert_xml(string $str)

Descripción

Convierte caracteres XML reservados a entidades

Parámetros

str

String de entrada

Valores devueltos

String convertida

data()

string data(string $item)

Descripción

Devuelve un solo elemento del array de datos de respuesta

Parámetros

item

Clave de datos

Valores devueltos

Valor de datos o un string vacío si no se encuentra

display_errors()

string display_errors([string $open = ' ' [, string $close = ' ']])

Descripción

Devuelve mensajes de error formateados en HTML o un string vacío si no hay errores

Parámetros

open

Etiqueta de apertura

close

Etiqueta de cierre

Valores devueltos

Mensajes de error en formato HTML

extract_urls()

array extract_urls(string $urls)

Descripción

Este método permite el envío de múltiples trackbacks, toma un string de URL, separadas por coma o espacio, y coloca cada URL en un array

Parámetros

urls

Lista de URL separadas por comas

Valores devueltos

Array de URL

get_id()

string get_id(string $url)

Descripción

Busca y devuelve un ID de trackback URL

Parámetros

url

URL de referencia

Valores devueltos

ID de URL o FALSE en caso de error

limit_characters()

string limit_characters(string $str [, int $n = 500 [, string $end_char = '…']])

Descripción

Limita el string en función del recuento de caracteres, preservará palabras completas

Parámetros

str

String de entrada

n

Número máximo de caracteres

end_char

Carácter para poner al final del string

Valores devueltos

String recortado

process()

bool process(string $url, string $data)

Descripción

Abre una conexión de socket y pasa los datos al servidor

Parámetros

url

URL objetivo

data

Datos Raw POST

Valores devueltos

TRUE en el éxito, FALSE si falla

receive()

bool receive()

Descripción

Este método valida los datos de trackback entrantes, si los datos son válidos, se incorporan para que pueda insertarse en una Base de datos con **$this->data**

Valores devueltos

TRUE en caso de éxito, FALSE en caso de error

send()

bool send(array $tb_data)

Descripción

Enviar trackback

Parámetros

tb_data

Datos de referencia

Valores devueltos

TRUE en el éxito, FALSE si falla

send_error()

```
void send_error([ string $message = 'Incomplete information'])
```

Descripción

Responde a una petición trackback con un mensaje de error

Parámetros

message

Mensaje de error

> ***Nota***
>
> Este método terminará la ejecución del script

valores devueltos

void

send_success()

```
void send_success()
```

Descripción

Responde a una petición trackback con un mensaje de éxito

> ***Nota***
>
> Este método terminará la ejecución del script

valores devueltos

void

set_error()

```
void set_error(string $msg)
```

Descripción

Establece un registro como un mensaje de error

Parámetros

msg

Mensaje de error

valores devueltos

void

validate_url()

```
void validate_url(string &$url)
```

Descripción

Simplemente agrega el prefijo **http://** si aún no está presente en la URL

Parámetros

url

URL de referencia

valores devueltos

void

Typography Class

Typography Class proporciona métodos que lo ayudan a formatear el texto

Iniciar Typography Class

Typography Class se inicia con **$this->load->library()**:

```
$this->load->library('typography');
```

Una vez cargada, el objeto esta disponible usando:

```
$this->typography
```

class CI_Typography

class CI_Typography

class CI_Typography

$protect_braced_quotes

$protect_braced_quotes = FALSE

Al utilizar la Typography library junto con la **Template Parser library**, a menudo puede ser conveniente proteger las comillas simples y dobles entre llaves, para habilitar esto, establezca la propiedad de clase **protect_braced_quotes** en TRUE

Ejemplo

```
$this->load->library('typography');
$this->typography->protect_braced_quotes = TRUE;
```

auto_typography()

string auto_typography(string $str [, bool $reduce_linebreaks = FALSE])

Descripción

Formatea texto para que sea HTML semántica y tipográficamente correctos, toma como entrada un string y lo devuelve con el siguiente formato:

- Rodea párrafos dentro de <p> </ p> ,busca saltos de línea dobles para identificar los párrafos
- Los saltos de línea individuales se convierten en
, excepto los que aparecen dentro de las etiquetas <pre>
- Los elementos a nivel de bloque, como las etiquetas , no están incluidos en los párrafos, pero sí su texto contenido, si contiene párrafos
- Las comillas se convierten correctamente en comillas tipográficas enfrentadas, excepto aquellas que aparecen dentro de las etiquetas
- Los apóstrofes se convierten en apóstrofes tipográficos

- Los guiones dobles, sea como -- estos o como estos--otros, se convierten en em—dashes
- Tres puntos correlativos sea precediendo o siguiendo a una palabras, se convierten en (…)
- Los espacios dobles siguiendo sentencias se convierten en   que imitan el doble espaciado
- Los espacios dobles que siguen las oraciones se convierten en non-breaking spaces para imitar el doble espacio

Parámetros

str

String de entrada

reduce_linebreaks

Opcional, si es TRUE reduce más de dos linebreaks (salto de línea '\n') consecutivos a dos

Valores devueltos

Cadena tipográfica HTML-safe

Ejemplo

```
$string = $this->typography->auto_typography($string);
```

> ***Nota***
>
> El formateo tipográfico puede ser intensivo en el uso del procesador, particularmente si hay mucho contenido a formatear, si elige esta función puede querer considerar cachear sus páginas

format_characters()

string format_characters(string $str)

Descripción

Este método es similar a **auto_typography()**, excepto que solo hace conversión de caracteres:

- Las comillas se convierten correctamente en comillas tipográficas enfrentadas, excepto aquellas que aparecen dentro de las etiquetas
- Las cotizaciones se convierten en entidades de cotización rizada que se enfrentan correctamente, excepto aquellas que aparecen dentro de las etiquetas
- Los apóstrofes se convierten en apóstrofes tipográficos
- Los guiones dobles, sea como -- estos o como estos--otros, se convierten en em—dashes
- Tres puntos correlativos sea precediendo o siguiendo a una palabras, se convierten en (…)
- Los espacios dobles siguiendo sentencias se convierten en   que imitan el doble espaciado

Parámetros

str

String de entrada

Valores devueltos

String formateada

Ejemplo

```
$string = $this->typography->format_characters($string);
```

nl2br_except_pre()

string nl2br_except_pre(string $str)

Descripción

Convierte las nuevas líneas en etiquetas **
** a menos que aparezcan dentro de las etiquetas **<pre>**, esta función es idéntica a la función **nl2br()** nativa de PHP, excepto que ignora las etiquetas **<pre>**

Parámetros

str

String de entrada

Valores devueltos

String formateada

Ejemplo

```
$string = $this->typography->nl2br_except_pre($string);
```

Unit Testing Class

Las pruebas de unidad son un enfoque de la ingeniería del software en el que se escriben pruebas para cada función en su aplicación, si no está familiarizado con este concepto, podría hacer una pequeña investigación googleando el tema

La clase Unit Testing de CodeIgniter es bastante simple, y consta de una función de evaluación y dos funciones de resultado, no pretende ser un conjunto de pruebas completo, sino un mecanismo simple para evaluar su código y determinar si se está produciendo el tipo de datos y el resultado correctos

Iniciar Unit Testing Class

Unit Testing Class se inicia con **$this->load->library()**:

```
$this->load->library('unit_test');
```

Una vez cargada, el objeto esta disponible usando:

```
$this->unit
```

Ejecutar pruebas

Ejecutar una prueba implica proporcionar una prueba y un resultado esperado de la siguiente manera:

```
$this->unit->run('prueba', 'resultado_esperado', 'nombre_prueba', 'notas');
```

Donde prueba es el resultado del código que desea probar, **resultado_esperado** es el tipo de dato que espera, **nombre_prueba** es un nombre opcional que le da a su prueba y notas son notas opcionales

Ejemplo

```
$prueba = 1 + 1;

$resultado_esperado = 2;

$nombre_prueba = 'Sumar uno más uno';

$this->unit->run($prueba, $resultado_esperado, $nombre_prueba);
```

El **resultado_esperado** que proporciona puede ser una coincidencia literal o una coincidencia de tipo de datos, ejemplo con un literal:

```
$this->unit->run('Foo', 'Foo');
```

Aquí un ejemplo de coincidencia de tipo de dato:

```
$this->unit->run('Foo', 'is_string');
```

Observe el uso de **"is_string"** en el segundo parámetro, esto le dice al método que evalúe si su prueba está

produciendo un string como resultado

Aquí hay una lista de tipos de comparación permitidos:

- is_object
- is_string
- is_bool
- is_true
- is_false
- is_int
- is_numeric
- is_float
- is_double
- is_array
- is_null
- is_resource

Generando Informes

Puede mostrar los resultados después de cada prueba, o puede ejecutar varias pruebas y generar un informe al final, para mostrar un informe directamente, simplemente use '**echo**' o utilice la devolución de **run()**:

```
echo $this->unit->run($prueba, $resultado_esperado);
```

Para ejecutar un informe completo de todas las pruebas, use esto:

```
echo $this->unit->report();
```

El informe se formateará en una tabla HTML para su visualización, si prefiere los datos sin procesar, puede recuperar un array usando:

```
echo $this->unit->result();
```

Modo estricto

Por defecto, la clase **unit_test** evalúa las coincidencias literales vagamente, considere este ejemplo:

```
$this->unit->run(1, TRUE);
```

La prueba se evalúa como entero, pero el resultado esperado es booleano, PHP, sin embargo, debido a su tipado de datos, evaluará el código anterior como TRUE usando la prueba de igualdad normal:

```
if (1 == TRUE) echo 'Esto se evalúa como verdadero';
```

Si lo prefiere, puede poner la clase **unit_test** en modo estricto, que comparará el tipo de datos y el valor:

```
if (1 === TRUE) echo 'Esto se evalúa como FALSE';
```

Puede habilitar el modo estricto así:

```
$this->unit->use_strict(TRUE);
```

Habilitar / deshabilitar las pruebas unitarias

Si quisiera dejar algunas pruebas en sus scripts, pero no ejecutarlas a menos que lo necesite, puede desactivar las pruebas unitarias usando:

```
$this->unit->active(FALSE);
```

Pantalla de prueba de la unidad

Al visualizar los resultados de la prueba de unidad, se muestran los siguientes elementos por defecto:

- **test_name** - Nombre de prueba
- **test_datatype** - Tipo de dato de prueba
- **res_datatype** - Tipo de datos esperado
- **result** - Resultado
- **file** - Nombre de archivo
- **line** - Número de línea
- **notes** - Cualquier nota que ingresó para la prueba

Puede personalizar cuáles de estos elementos se muestran usando **$this->unit->set_test_items()**, por ejemplo, si solo desea el nombre de la prueba y el resultado que se muestra

Personalizar las pruebas mostradas

```
$this->unit->set_test_items(array('test_name', 'result'));
```

Creando una plantilla

Si desea que los resultados de sus pruebas tengan un formato diferente, puede configurar su propia plantilla por defecto, aquí hay un ejemplo de una plantilla simple, tenga en cuenta las pseudovariables requeridas:

```
$str = '
<table border="0" cellpadding="4" cellspacing="1">
{rows}
 <tr>
    <td>{item}</td>
    <td>{result}</td>
 </tr>
{/rows}
</table>';

$this->unit->set_template($str);
```

Nota

Su plantilla debe declararse antes de ejecutar el proceso de prueba

class CI_Unit_test

class CI_Unit_test

```
class CI_Unit_test
```

active()

```
void active([ bool $state = TRUE])
```

Descripción

Habilita/deshabilita la prueba unitaria

Parámetros

state

Si se deben habilitar las pruebas

valores devueltos

void

report()

```
string report([ array $result = array()])
```

Descripción

Genera un informe sobre las pruebas ya completadas

Parámetros

result

Array que contiene los resultados de las pruebas

Valores devueltos

Informe de prueba

result()

```
array result([ array $results = array()])
```

Descripción

Devuelve datos de resultados de pruebas sin procesar

Parámetros

results

Lista de resultados de las pruebas

Valores devueltos

Array de datos de resultados sin procesar

run()

string run(mixed $test [, mixed $expected = TRUE [, string $test_name = 'undefined' [, string $notes = '']]])

Descripción

Ejecuta pruebas unitarias

Parámetros

test

Datos de prueba

expected

Resultado esperado

test_name

Nombre de la prueba

notes

Cualquier nota que se adjunte a la prueba

Valores devueltos

Informe de prueba

set_template()

void set_template(string $template)

Descripción

Establece la plantilla para mostrar los resultados de las pruebas

Parámetros

template

Plantilla de resultados de prueba

valores devueltos

void

set_test_items()

void set_test_items(array $items)

Descripción

Establece una lista de elementos que deberían ser visibles en las pruebas. Las opciones válidas son:

- test_name
- test_datatype
- res_datatype
- result
- file
- line
- notes

Parámetros

items

Array con la lista de elementos de prueba visibles

valores devueltos

void

use_strict()

```
void use_strict([ bool $state = TRUE])
```

Descripción

Habilita/deshabilita la comparación de tipos estrictos en las pruebas

Parámetros

state

Indicador de estado estricto

valores devueltos

void

URI Class

URI Class proporciona métodos que le ayudan a recuperar información de sus strings de URI, si utiliza el enrutamiento URI, también puede recuperar información sobre los segmentos redirigidos

Nota

Esta clase se inicializa automáticamente por el sistema, por lo que no es necesario hacerlo manualmente

class CI_URI

class CI_URI

class CI_URI

assoc_to_uri()

string assoc_to_uri(array $array)

Descripción

Toma un array asociativo como entrada y genera un string URI a partir de el, las claves de array se incluirán en el string

Parámetros

array

Array de entrada de pares clave/valor

Valores devueltos

Cadena de URI

Ejemplo

```
$array = array('producto' => 'zapato', 'material' => 'gamuza', 'color' => 'azul');
$str = $this->uri->assoc_to_uri($array);

// Crea: producto/zapato/material/gamuza/color/azul
```

rsegment()

mixed rsegment(int $n [, mixed $no_result = NULL])

Descripción

Este método es idéntico a **segment()**, excepto que le permite recuperar un segmento específico de su URI reencaminado en caso de que esté utilizando la característica de enrutamiento URI Routing de CodeIgniter

Parámetros

n

Número de índice del segmento

no_result

Qué devolver si no se encuentra el segmento buscado

Valores devueltos

Valor del segmento indicado o el valor **$no_result** si no se encuentra

rsegment_array()

```
array rsegment_array()
```

Descripción

Este método es idéntico a **segment_array()**, excepto que devuelve el array de segmentos en su URI redirigido en caso de que esté utilizando la característica de enrutamiento URI Routing de CodeIgniter

Valores devueltos

Array de segmentos de URI enrutados

ruri_string()

```
string ruri_string()
```

Descripción

Este método es idéntico a **uri_string()**, excepto que devuelve el URI redirigido en caso de que esté utilizando la característica de enrutamiento URI Routing de CodeIgniter

Valores devueltos

String URI enrutada

ruri_to_assoc()

```
array ruri_to_assoc([ int $n = 3 [, array $default = array()]])
```

Descripción

Este método es idéntico a **uri_to_assoc(),** excepto que crea un array asociativo utilizando el URI redirigido en caso de que esté utilizando la característica de enrutamiento URI Routing de CodeIgniter

Parámetros

n

Número de índice del segmento

default

Valores predeterminados

Valores devueltos

Array asociativo de segmentos URI

segment()

mixed segment(int $n [, mixed $no_result = NULL])

Descripción

Le permite recuperar un segmento específico, donde **n** es el número de segmento que desea recuperar, los segmentos están numerados de izquierda a derecha

Por ejemplo, si su URL completa es esta:

```
http://ejemplo.com/index.php/noticias/nacional/gobierno/suben_las_pensiones
```

Los números de segmento serán estos:

1. noticias
2. nacional
3. gobierno
4. suben_las_pensiones

Parámetros

n

Número de índice del segmento

no_result

Opcional, valor a devolver si no se encuentra el segmento buscado, se establece por defecto a NULL y permite establecer el valor de retorno del método cuando falta el segmento de URI solicitado

Por ejemplo, esto le indicaría al método que devuelva el número cero en caso de fallo:

```
$product_id = $this->uri->segment(3, 0);
```

Ayuda a evitar tener que escribir código como este:

```
if ($this->uri->segment(3) === FALSE)
{
   $product_id = 0;
}
else
{
   $product_id = $this->uri->segment(3);
}
```

Valores devueltos

Valor de segmento o el valor de **$no_result** si no se encuentra

segment_array()

array segment_array()

Descripción

Devuelve un array que contiene los segmentos de URI

Valores devueltos

Array de segmentos URI

Ejemplo

```
$segs = $this->uri->segment_array();

foreach ($segs as $segment)
{
   echo $segment;
   echo '<br />';
}
```

slash_rsegment()

string slash_rsegment(int $n [, string $where = 'trailing'])

Descripción

Este método es idéntico a **slash_segment()**, excepto que le permite agregar barras un segmento específico de su URI redirigido en caso de que esté utilizando la característica URI Routing de CodeIgniter

Parámetros

n

Número de índice del segmento

where

Dónde agregar la barra ('trailing' o 'leading')

Valores devueltos

Valor del segmento con una barra inclinada al comienzo/final, o una barra inclinada si no se encuentra

slash_segment()

string slash_segment(int $n [, string $where = 'trailing'])

Descripción

Este método es casi idéntico a **segment()**, excepto que agrega las barras al comienzo y/o final según el valor del segundo parámetro. Si no se usa este parámetro, se agrega una barra al final

Parámetros

n

Número de índice del segmento

where

Dónde agregar la barra ('trailing' o 'leading')

Valores devueltos

Valor del segmento con una barra inclinada al comienzo/final, o una barra inclinada si no se encuentra

Ejemplo

```
$this->uri->slash_segment(3);
$this->uri->slash_segment(3, 'leading');
$this->uri->slash_segment(3, 'both');
```

Valores devueltos :

1. segmento/
2. /segmento
3. /segmento/

total_rsegments()

int total_rsegments()

Descripción

Este método es idéntico a **total_segments()**, excepto que devuelve el número total de segmentos en su URI redirigido en caso de que esté utilizando la característica de enrutamiento URI Routing de CodeIgniter

Valores devueltos

Recuento de segmentos de URI enrutados

total_segments()

int total_segments()

Descripción

Devuelve la cantidad total de segmentos

Valores devueltos

Recuento de segmentos de URI

uri_string()

string uri_string()

Descripción

Devuelve un string con el URI completo

Valores devueltos

String URI

Ejemplo

Si esta es la URL completa:

```
http://ejemplo.com/index.php/noticias/local/345
```

El método devolvera esto:

```
noticias/local/345
```

uri_to_assoc()

array uri_to_assoc([int $n = 3 [, array $default = array()]])

Descripción

Este método le permite convertir segmentos URI en un array asociativo de pares clave/valor

Considere este URI:

```
index.php/usuario/buscar/nombre/carmen/pais/spain/genero/mujer
```

Con este método, puedes convertir el URI en una array asociativa con este prototipo:

```
[array]
(
   'nombre' => 'carmen'
   'país'   => 'spain'
   'genero' => 'mujer'
)
```

El primer parámetro le permite establecer un desplazamiento, que por defecto es **3** ya que su URI normalmente contendrá un par de controlador/método en los segmentos primero y segundo:

```
$array = $this->uri->uri_to_assoc(3);
echo $array['nombre'];
```

El segundo parámetro le permite establecer nombres de clave predeterminados, de modo que el array devuelto contendrá siempre los índices esperados, incluso si faltan en el URI:

```
$default = array('nombre', 'genero', 'país', 'tipo', 'ordenar');
$array = $this->uri->uri_to_assoc(3, $default);
```

Si el URI no contiene un valor valor predeterminado, un índice de array se establecerá a ese nombre, con valor NULL. Por último, si no se encuentra un valor correspondiente para una clave dada, si hay un número impar de segmentos de URI, el valor se establecerá en NULL

Parámetros

n

Número de índice del segmento

default

Valores predeterminados

Valores devueltos

Serie de segmentos URI asociativos

User Agent Class

User Agent Class proporciona métodos que ayudan a identificar información sobre el navegador, dispositivo móvil o robot que visita su sitio, además, puede obtener información de referencia, así como el idioma y la información de conjunto de caracteres admitidos

Iniciar User Agent Class

User Agent Class se inicia con **$this->load->library()**:

```
$this->load->library('user_agent');
```

Una vez cargada, el objeto esta disponible usando:

```
$this-> agent
```

Definiciones de agente de usuario

Las definiciones de nombre del User Agent se encuentran en un archivo de configuración ubicado en: **application/config/user_agents.php**, puede agregar elementos a los diversos arrays de agente de usuario si es necesario

Ejemplo

Cuando se inicializa la clase, intentará determinar si el agente de usuario que explora su sitio es un navegador web, un dispositivo móvil o un robot, también recopilará la información de la plataforma si está disponible:

```
$this->load->library('user_agent');

if ($this->agent->is_browser())
{
   $agent = $this->agent->browser().' '.$this->agent->version();
}
elseif ($this->agent->is_robot())
{
   $agent = $this->agent->robot();
}
elseif ($this->agent->is_mobile())
{
   $agent = $this->agent->mobile();
}
else
{
   $agent = 'Agente de usuario no identificado';
}
```

```
echo $agent;

echo $this->agent->platform(); // Información de la plataforma (Windows, Linux, Mac, etc.)
```

class CI_User_agent

class CI_User_agent

class CI_User_agent

accept_charset()

bool accept_charset([string $charset = 'utf-8'])

Descripción

Permite determinar si el agente de usuario acepta un juego de caracteres en particular

Parámetros

charset

Conjunto de caracteres

Valores devueltos

TRUE si el conjunto de caracteres es aceptado, FALSE si no

Ejemplo

```
if ($this->agent->accept_charset('utf-8'))
{
   echo 'Su navegador soporta UTF-8!';
}
```

Nota

Este método no suele ser muy confiable ya que algunos navegadores no proporcionan información de conjunto de caracteres, e incluso entre los que sí lo hacen, no siempre es preciso

accept_lang()

bool accept_lang([string $lang = 'en'])

Descripción

Determina si el agente de usuario acepta un idioma en particular

Parámetros

lang

Clave de idioma

Valores devueltos

TRUE si se acepta el idioma proporcionado, FALSE si no

Ejemplo

```
if ($this->agent->accept_lang('en'))
{
   echo 'Acepta el inglés!';
}
```

> ***Nota***
>
> Este método no suele ser muy confiable, ya que algunos navegadores no proporcionan información de idioma, e incluso entre los que sí lo hacen, no siempre es preciso

agent_string()

string agent_string()

Descripción

Obtener el string de agente de usuario completo

Normalmente, será algo como esto:

```
Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en-US; rv:1.8.0.4) Gecko/20060613 Camino/1.0.2
```

Valores devueltos

String completa de agente de usuario o un string vacío

browser()

string browser()

Descripción

Obtener el nombre del navegador web que visita su sitio

Valores devueltos

Navegador detectado o un string vacío

charsets()

array charsets()

Descripción

Obtener una lista del conjuntos de caracteres aceptados por el agente de usuario

Valores devueltos

Un array con la lista de conjuntos de caracteres aceptados

languages()

array languages()

Descripción

Obtiene los idiomas admitidos por el agente de usuario

Valores devueltos

Una array con la lista de idiomas aceptados

mobile()

string mobile()

Descripción

Averiguar el nombre del dispositivo móvil que visita su sitio

Valores devueltos

Marca del dispositivo móvil detectado o una string vacía

is_browser()

bool is_browser([string $key = NULL])

Descripción

Devuelve TRUE/FALSE si el agente de usuario es un navegador web conocido

Parámetros

key

Opcional, Nombre del navegador

Valores devueltos

TRUE si el agente de usuario es un navegador (especificado), FALSE si no

Ejemplo

```
if ($this->agent->is_browser('Safari'))
{
   echo 'Está usando Safari.';
}
elseif ($this->agent->is_browser())
{
   echo 'Está usando un navegador.';
}
```

Nota

La string **"Safari"** en este ejemplo es una clave de array en la lista de definiciones de navegador. Puede encontrar esta lista en **application/config/user_agents.php** si desea agregar nuevos navegadores o cambiar las opciones

is_mobile()

bool is_mobile([string $key = NULL])

Descripción

Devuelve TRUE/FALSE si el agente de usuario es un dispositivo móvil conocido

Parámetros

key

Opcional, nombre del dispositivo móvil

Valores devueltos

TRUE si el agente de usuario es un dispositivo móvil (especificado), FALSE si no

Ejemplo

```
if ($this->agent->is_mobile('iphone'))
{
   $this->load->view('iphone/home');
}
elseif ($this->agent->is_mobile())
{
   $this->load->view('mobile/home');
}
else
{
   $this->load->view('web/home');
}
```

is_referral()

bool is_referral()

Descripción

Averiguar si el agente de usuario fue derivado desde otro sitio

Valores devueltos

TRUE si el agente de usuario es una referencia, FALSE si no

is_robot()

bool is_robot([string $key = NULL])

Descripción

Devuelve TRUE/FALSE si el agente de usuario es un robot conocido

Parámetros

key

Opcional, nombre del robot

Valores devueltos

TRUE si el agente de usuario es un robot (especificado), FALSE si no

> ***Nota***
>
> La biblioteca de agente de usuario solo contiene las definiciones de robot más comunes, no es una lista completa de robots, hay cientos de ellos, por lo que buscarlos no sería muy eficiente. Si encuentra que algunos bots que comúnmente visitan su sitio no se encuentran en la lista, puede agregarlos a su **archivo application/config/user_agents.php**

parse()

void parse(string $string)

Descripción

Analiza un string personalizado de agente de usuario, es diferente de la información facilitada para el visitante actual

Parámetros

string

Un string personalizada de agente de usuario

valores devueltos

void

platform()

string platform()

Descripción

Averiguar la plataforma que visualiza su sitio, Linux, Windows, OS X, etc.

Valores devueltos

Sistema operativo detectado o una string vacía

referrer()

string referrer()

Descripción

El referente, si el agente de usuario fue referido desde otro sitio

Normalmente lo probará de la siguiente manera:

```
if ($this->agent->is_referral())
{
   echo $this->agent->referrer();
}
```

Valores devueltos

Referrer detectado o una string vacía

robot()

string robot()

Descripción

Obtener el nombre del robot que visita su sitio

Valores devueltos

Nombre del robot detectado o una string vacía

version()

string version()

Descripción

Obtener el número de versión del navegador web que visita su sitio

Valores devueltos

Versión del navegador detectado o una string vacía

XML-RPC and XML-RPC Server Classes

Las clases XML-RPC de CodeIgniter le permiten enviar solicitudes a otro servidor o configurar su propio servidor XML-RPC para recibir solicitudes

Qué es XML-RPC?

Simplemente, es una forma en que dos computadoras se comuniquen en Internet usando XML, una computadora, que llamaremos cliente, envía una solicitud XML-RPC a la otra, que llamaremos servidor, una vez que el servidor recibe y procesa la solicitud, enviará una respuesta al cliente

Por ejemplo, al utilizar el API MetaWeblog, un cliente XML-RPC, generalmente una herramienta de autoedición, enviará una solicitud a un servidor XML-RPC que se ejecute en su sitio, esta solicitud podría ser una nueva entrada del weblog que se envía para su publicación, o podría ser una solicitud de una entrada existente para su edición, cuando el servidor XML-RPC recibe esta solicitud, la examinará para determinar qué clase/método debe invocarse para procesar la solicitud, una vez procesado, el servidor devolverá un mensaje de respuesta

Para especificaciones detalladas, puede visitar el sitio XML-RPC

Iniciar XML-RPC y XML-RPCS

XML-RPC y XML-RPCS se inicia con **$this->load->library()**:

```
$this->load->library('xmlrpc');
```

Una vez cargada, el objeto esta disponible usando:

```
$this->xmlrpc
```

Para cargar la clase del Servidor XML-RPC, usará:

```
$this->load->library('xmlrpc');
$this->load->library('xmlrpcs');
```

Una vez cargada, el objeto esta disponible usando:

```
$this->xmlrpcs
```

Nota

Al usar la clase del Servidor XML-RPC, debe cargar **AMBAS** clases XML-RPC y la clase del Servidor XML-RPC

Envío de solicitudes XML-RPC

Para enviar una solicitud a un servidor XML-RPC, debe especificar la siguiente información:

- La URL del servidor

- El método en el servidor al que desea llamar
- Los datos de solicitud (explicados a continuación)

Este es un ejemplo básico que envía un ping simple de Weblogs.com para Ping-o-Matic:

```
$this->load->library('xmlrpc');

$this->xmlrpc->server('http://rpc.pingomatic.com/', 80);
$this->xmlrpc->method('weblogUpdates.ping');

$request = array('My Photoblog', 'http://www.my-site.com/photoblog/');
$this->xmlrpc->request($request);

if ( ! $this->xmlrpc->send_request())
{
   echo $this->xmlrpc->display_error();
}
```

Explicación

El código anterior inicializa la clase XML-RPC, establece el URL y el método del servidor para que se llame (**weblogUpdates.ping**), la solicitud, en este caso, el título y la URL de su sitio, se coloca en una array para el transporte y se compila utilizando la función **request()**, por último, se envía la solicitud completa, si el método **send_request()** devuelve falso, mostraremos un mensaje de error enviado desde el servidor XML-RPC

Anatomía de una solicitud

Una solicitud XML-RPC es simplemente la información que está enviando al servidor XML-RPC, cada parte de los datos en una solicitud se conoce como un parámetro de solicitud, el ejemplo anterior tiene dos Parámetros el título de su sitio y la URL, cuando el servidor XML-RPC recibe su solicitud, buscará los parámetros que requiera

Los parámetros de solicitud se deben colocar en un array para el transporte, y cada parámetro puede ser uno de los siete tipos de datos ,strings, numbers, dates, etc., si sus parámetros son algo más que strings, deberá incluir el tipo de datos en el array de solicitud

Aquí hay un ejemplo de un array simple con tres parámetros:

```
$request = array('Mortadelo', 'Filemón', 'www.latia-site.com');
$this->xmlrpc->request($request);
```

Si usa tipos de datos que no sean strings, o si tiene varios tipos de datos diferentes, colocará cada parámetro en su propio array, con el tipo de datos en la segunda posición:

```
$solicitud = array(
  array('Mortadelo', 'string'),
  array('Filemón', 'string'),
  array(FALSE, 'boolean'),
```

```
    array(12345, 'int')
);
$this->xmlrpc->request($solicitud);
```

Tiene una lista completa de tipos de datos en la sección **Tipos de datos**

Crear un servidor XML-RPC

Un servidor XML-RPC actúa como un tipo de policía de tráfico, esperando las solicitudes entrantes y redirigiéndolas a las funciones apropiadas para el procesamiento

Crear su propio servidor XML-RPC implica inicializar la clase de Servidor XML-RPC en su controlador donde espera que aparezca la solicitud entrante, luego configurar un array con instrucciones de asignación para que las solicitudes entrantes puedan enviarse a la clase y método apropiados para su tratamiento

Ejemplo

```
$this->load->library('xmlrpc');
$this->load->library('xmlrpcs');

$config['functions']['post_nuevo'] = array('function' => 'Mi_blog.entrada_nueva');
$config['functions']['post_actualizar'] = array('function' => 'Mi_blog.entrada_actualizar');
$config['object'] = $this;

$this->xmlrpcs->initialize($config);
$this->xmlrpcs->serve();
```

El ejemplo anterior contiene un array que especifica dos solicitudes de método que el servidor permite, los métodos permitidos están en el lado izquierdo del array, cuando se reciben cualquiera de estos, se asignarán a la clase y método a la derecha

La clave **'object'** es una clave especial a la que se le pasa un objeto de clase instanciado, que es necesario cuando el método que se está mapeando no forma parte del súper objeto CodeIgniter

En otras palabras, si un Cliente XML-RPC envía una solicitud para el método **post_nuevo**, su servidor cargará la clase **Mi_blog** y llamará a la función **entrada_nueva**, si la solicitud es para el método **post_actualizar**, su servidor cargará la clase **Mi_blog** y llamará al método **entrada_actualizar()**

Los nombres de los métodos en el ejemplo anterior son arbitrarios, usted decide qué se deben llamar en su servidor, o si usa un API estandarizadas, como la API de Blogger o MetaWeblog, usará sus nombres de funciones

Hay dos claves de configuración adicionales que puede utilizar al inicializar la clase de servidor: **debug** que puede establecerse en TRUE para habilitar la depuración, y **xss_clean** que puede establecerse en FALSE para evitar el envío de datos a través del método **xss_clean()** de la **Security library**

Procesamiento de solicitudes del servidor

Cuando el Servidor XML-RPC recibe una solicitud y carga la clase/método para su procesamiento, pasará un objeto a ese método que contiene los datos enviados por el cliente

Usando el ejemplo anterior, si se solicita el método **post_nuevo**, el servidor esperará que exista una clase con este prototipo:

```
class My_blog extends CI_Controller {

  public function post_nuevo($request)
  {

  }
}
```

La variable **$request** es un objeto compilado por el Servidor, que contiene los datos enviados por el Cliente XML-RPC. Al usar este objeto, tendrá acceso a los parámetros de solicitud que le permitirán procesar la solicitud. Cuando haya terminado, enviará una respuesta al cliente

A continuación se muestra un ejemplo del mundo real, utilizando la API de Blogger, uno de los métodos en la API de Blogger es **getUserInfo()**, con este método, un Cliente XML-RPC puede enviar al Servidor un nombre de usuario y contraseña, a cambio el Servidor devuelve información sobre ese usuario en particular, apodo, ID de usuario, dirección de correo electrónico, etc.

Así es como se vería la función de procesamiento:

```
class My_blog extends CI_Controller {

  public function getUserInfo($petición)
  {
    $nombre = 'gunterkunkel';
    $contraseña = 'excepcionalpersona';

    $this->load->library('xmlrpc');

    $parametros = $petición->output_parameters();

    if ($parametros[1] != $nombre && $parametros[2] != $contraseña)
    {
       return $this->xmlrpc->send_error_message('100', 'Acceso erroneo');
    }

    $respuesta = array(
      array(
         'nickname' => array('gunterkunkel', 'string'),
         'userid'   => array('99', 'string'),
         'url'      => array('http://suweb.com', 'string'),
         'email'    => array('loskunkel@suweb.com', 'string'),
         'nombre'   => array('Gunter', 'string'),
         'apellido' => array('Kunkel', 'string')
       ),
         'struct'
```

```
    );

    return $this->xmlrpc->send_response($respuesta);
  }
}
```

Notas:

El método **output_parameters()** recupera un array indexado correspondiente a los parámetros de solicitud enviados por el cliente, en el ejemplo anterior, los parámetros de salida serán username y password

Si **nombre** y **contraseña** enviados por el cliente no son válidos, se devuelve un mensaje de error utilizando **send_error_message()**

Si la operación fue exitosa, se enviará al cliente un array de respuesta que contiene la información del usuario

Formatear una respuesta

De forma similar a las solicitudes, las respuestas deben formatearse como un array, sin embargo, a diferencia de las solicitudes, una respuesta es un array que contiene un solo elemento, este elemento puede ser un array con varios arrays adicionales, pero solo puede haber un índice de array principal

En otras palabras, el prototipo básico es este:

```
$respuesta = array('Datos de respuesta', 'array');
```

Las respuestas, sin embargo, generalmente contienen varias piezas de información, para lograr esto, debemos colocar la respuesta en su propio array para que el array primario continúe conteniendo una sola pieza de datos

Aquí hay un ejemplo que muestra cómo se podría lograr esto:

```
$respuesta = array(
  array(
    'primer_nombre'   => array('Gunter', 'string'),
    'primer_apellido' => array('Kunkel', 'string'),
    'member_id'       => array(123435, 'int'),
    'todo_list'       => array(array('expedición', 'semillas', 'clasificar'), 'array'),
  ),
  'struct'
);
```

Tenga en cuenta que el array anterior tiene el formato de una estructura, este es el tipo de datos más común para las respuestas

Al igual que con las solicitudes, una respuesta puede ser uno de los siete tipos de datos enumerados en la sección **Tipos de datos**

Envío de una respuesta de error

Si necesita enviar al cliente una respuesta de error, usará lo siguiente:

```
return $this->xmlrpc->send_error_message('123', 'Datos solicitados no disponibles');
```

El primer parámetro es el número de error, mientras que el segundo parámetro es el mensaje de error

Creando su propio cliente y servidor

Para ayudarlo a comprender todo lo expuesto hasta ahora, creamos un par de controladores que actúen como cliente y servidor XML-RPC, utilizará el Cliente para enviar una solicitud al Servidor y recibir una respuesta

El cliente

Usando un editor de texto, cree un controlador llamado **Xmlrpc_client.php**, en él, coloque este código y guárdelo en el directorio **application/controllers/**

```
class Xmlrpc_client extends CI_Controller {

  public function index()
  {
     $this->load->helper('url');
     $server_url = site_url('xmlrpc_server');

     $this->load->library('xmlrpc');

     $this->xmlrpc->server($server_url, 80);
     $this->xmlrpc->method('Saludos');

     $pregunta = array('Cómo va?');
     $this->xmlrpc->request($pregunta);

     if ( ! $this->xmlrpc->send_request())
     {
       echo $this->xmlrpc->display_error();
     }
     else
     {
       echo '<pre>';

       print_r($this->xmlrpc->display_response());
       echo '</pre>';
    }
  }
}
?>
```

Nota

En el código anterior estamos usando un **"url helper"**. Puede encontrar más información en la sección de **Helpers**

El servidor

Usando un editor de texto, cree un controlador llamado **Xmlrpc_server.php**, en él, coloque este código y guárdelo en el directorio **application/controllers/**

```
<?php

class Xmlrpc_server extends CI_Controller {

  public function index()
  {
     $this->load->library('xmlrpc');
     $this->load->library('xmlrpcs');

     $config['functions']['Saludos'] = array('function' => 'Xmlrpc_server.process');

     $this->xmlrpcs->initialize($config);
     $this->xmlrpcs->serve();
  }

  public function process($pregunta)
  {
    $parametros = $pregunta->output_parameters();

    $respuesta = array(
      array(
       'you_said'  => $parametros[0],
       'i_respond' => 'Nada mal.'
       ),
       'struct'
    );

    return $this->xmlrpc->send_response($respuesta);
  }
}
```

Intentelo !

Ahora visite su sitio usando una URL similar a esta:

```
ejemplo.com/index.php/xmlrpc_client/
```

Ahora debería ver el mensaje que envió al servidor y su respuesta
El cliente que creó envía un mensaje (**"Cómo va?"**) al servidor, junto con una solicitud del método **"Saludos"**, el servidor recibe la solicitud y la asigna al método **process()**, donde se envía una respuesta

Uso de arrays asociativos en un parámetro de solicitud

Si desea utilizar un array asociativo en los parámetros de su método, necesitará usar un tipo de datos struct:

```
$pregunta = array(
  array(
    // Param 0
    array('nombre' => 'Ataulfo'),
    'struct'
   ),
   array(
      // Param 1
      array(
         'especie' => 'alcornoque',
         'fruto'   => 'bellota'
      ),
      'struct'
   )
);

$this->xmlrpc->request($pregunta);
```

Puede recuperar el array asociativo al procesar la solicitud en el Servidor:

```
$parametros = $pregunta->output_parameters();
$nombre = $parametros[0]['nombre'];
$especie = $parametros[1]['especie'];
$fruto = $parametros[1]['fruto'];
```

Tipos de datos

Según la especificación de XML-RPC, hay siete tipos de valores que puede enviar a través de XML-RPC:

- int o i4
- boolean
- string
- double
- dateTime.iso8601
- base64
- struct (contiene una array de valores)
- array (contiene una array de valores)

class CI_Xmlrpc

class CI_Xmlrpc

class CI_Xmlrpc

display_error()

string display_error()

Descripción

Devuelve un mensaje de error como un string si su solicitud falló por alguna razón

Valores devueltos

String de mensaje de error

Ejemplo

```
echo $this->xmlrpc->display_error();
```

display_response()

mixed display_response()

Descripción

Devuelve la respuesta del servidor remoto una vez que se recibe la solicitud, la respuesta normalmente será un array asociativo

Valores devueltos

Respuesta

Ejemplo

```
$this->xmlrpc->display_response();
```

initialize()

void initialize([array $config = array()])

Descripción

Inicializa la biblioteca XML-RPC

Parámetros

config

Array asociativo que contiene su configuración

valores devueltos

void

method()

void method(string $function)

Descripción

Establece el método que se solicitará desde el servidor XML-RPC

Parámetros

function

Nombre del método

valores devueltos

void

Ejemplo

```
$this->xmlrpc->method('method');
```

Donde **method** es el nombre del método

request()

void request(array $incoming)

Descripción

Toma un array de datos y construye la solicitud a enviarse al servidor XML-RPC

Parámetros

incoming

Datos solicitados

valores devueltos

void

Ejemplo

```
$pregunta = array(array('My Photoblog', 'string'), 'http://www.suite.com/photoblog/');
$this->xmlrpc->request($pregunta);
```

send_error_message()

object send_error_message(int $number, string $message)

Descripción

Permite enviar un mensaje de error desde su servidor al cliente

Parámetros

number

Número de error

mensaje

Mensaje de error

Valores devueltos

Instancia XML_RPC_Response

Ejemplo

```
return $this->xmlrpc->send_error_message(123, 'Datos no disponibles');
```

send_request()

bool send_request()

Descripción

Método de envío de solicitud, devuelve TRUE/FALSE basado en el éxito o error, lo que permite su uso condicional

Valores devueltos

TRUE en caso de éxito, FALSE en caso de error

server()

void server(string $url [, int $port = 80 [, string $proxy = FALSE [, int $proxy_port = 8080]]])

Descripción

Establece la URL y el número de puerto del servidor al que se enviará una solicitud

Parámetros

url
URL del servidor XML-RPC

port
Puerto del servidor

proxy
Proxy opcional

proxy_port
Puerto de escucha proxy

valores devueltos

void

Ejemplo

```
$this->xmlrpc->server('http://www.algunasite.com/pings.php', 80);
```

La autenticación HTTP básica también es compatible, simplemente agréguela a la URL del servidor:

```
$this->xmlrpc->server('http://user:pass@localhost/', 80);
```

timeout()

void timeout(int $seconds = 5)

Descripción

Establece un período de tiempo de espera, en segundos, después del cual se cancelará la solicitud

Parámetros

seconds

Tiempo de espera en segundos

valores devueltos

void

Ejemplo

```
$this->xmlrpc->timeout(6);
```

Este período de tiempo se usará tanto para una conexión inicial al servidor remoto, como para obtener una respuesta del mismo, asegúrese de configurar el tiempo de espera antes de llamar a **send_request()**

Zip Encoding Class

Zip Encoding Class permite crear archivos Zip. Los archivos pueden descargarse a su escritorio o guardarse en un directorio

Iniciar Zip Encoding Class

Zip Encoding Class se inicia con **$this->load->library()**:

```
$this->load->library('zip');
```

Una vez cargada, el objeto esta disponible usando:

```
$this->zip
```

Ejemplo de uso

Este ejemplo muestra cómo comprimir un archivo, guardarlo en un directorio en su servidor y descargarlo a su escritorio

```
$archivo = 'mis_datos.txt';
$datos = 'Un string con los datos!';

$this->zip->add_data($archivo, $datos);

// Escribe el archivo zip, llamado mi_copia.zip en un directorio de su servidor.
$this->zip->archive('/path/al/directorio/mi_copia.zip');

// Descarga el archivo mi_copia.zip a su escritorio.
$this->zip->download('mi_copia.zip');
```

class CI_Zip

class CI_Zip

class CI_Zip

$compression_level

$compression_level = 2

El nivel de compresión a usar.
Puede tener un rango de 0 a 9, siendo 9 la compresión más alta y 0 sin compresión:

```
$this->zip->compression_level = 0;
```

add_data()

void add_data(mixed $filepath [, array $data = NULL])

Descripción

Agrega datos al archivo Zip, puede funcionar tanto en modo archivos individuales como múltiples

Parámetros

filepath

Una única ruta de archivo o un array de archivo => datos

data

Contenido del archivo (ignorado si $filepath es un array)

valores devueltos

void

Ejemplo

Al agregar un solo archivo, el primer parámetro debe contener el nombre que le gustaría darle al archivo y el segundo debe contener el contenido del archivo:

```
$archivo = 'mis_datos1.txt';
$datos = 'Un string con los datos!';
$this->zip->add_data($archivo, $datos);

$archivo = 'mis_datos2.txt';
$datos = 'Un string con otros datos!';
$this->zip->add_data($archivo, $datos);
```

Al agregar múltiples archivos, el primer parámetro debe contener pares **archivo => contenido** y el segundo parámetro se ignora:

```
$datos = array(
  'mis_datos1.txt' => 'Un string con los datos!',
  'mis_datos2.txt' => 'Un string con otros datos!'
);
```

Si desea que sus datos comprimidos se organicen en subdirectorios, simplemente incluya la ruta como parte del nombre de archivo:

```
$archivo = 'cervantes/el_quijote.txt';
$datos = 'En un lugar de la mancha...';

$this->zip->add_data($archivo, $datos);
```

El ejemplo anterior coloca **el_quijote.txt** dentro de un directorio llamado cervantes

add_dir()

void add_dir(mixed $directory)

Descripción

Crear un directorio, por lo general, este método es innecesario ya que puede colocar sus datos en directorios cuando usa **$this->zip->add_data()**, pero si desea crear un directorio vacío, puede hacerlo

Parámetros

directory

Cadena de nombre de directorio o un array con múltiples directorios

valores devueltos

void

Ejemplo

```
$this->zip->add_dir('un_directorio'); // Crea un directorio llamado "un_directorio"
```

archive()

bool archive(string $filepath)

Descripción

Escribe el archivo Zip en un directorio en su servidor, envíe una ruta de servidor válida que termine con el nombre del archivo, asegúrese de que el directorio sea escribible (755 generalmente está bien)

Parámetros

filepath

Ruta al archivo zip de destino

Valores devueltos

TRUE en el éxito, FALSE si falla

Ejemplo

```
// Crea un archivo llamado mi_archivo.zip
$this->zip->archive('/path/a/directorio/mi_archivo.zip');
```

clear_data()

void clear_data()

Descripción

La clase Zip guarda en caché los datos zip para que no sea necesario volver a compilar el archivo Zip para cada método que utilice anteriormente, sin embargo, si necesita crear múltiples archivos Zip, cada uno con datos diferentes, puede borrar el caché entre llamadas

valores devueltos

void

Ejemplo

```
$nombre = 'obelix.txt';
$datos = 'Están locos estos romanos';

$this->zip->add_data($nombre, $datos);
$zip_file = $this->zip->get_zip();

$this->zip->clear_data();

$nombre = 'foto.jpg';

// Leer el contenido del archivo
$this->zip->read_file("/path/a/foto.jpg");
$this->zip->download('mifoto.zip');
```

download()

void download(string $filename = 'backup.zip')

Descripción

Hace que el archivo Zip se descargue del servidor, debe pasar el nombre al que desea que se llame el archivo comprimido

Parámetros

filename

Nombre de archivo

valores devueltos

void

Ejemplo

```
// El archivo se llamará "incontinencia_suma.zip"
this->zip->download('incontinencia_suma.zip');
```

> ***Nota***
>
> No mostrar ningún dato en el controlador que llama este método, ya que envía varios encabezados de servidor que hacen que la descarga suceda y el archivo se trata como binario

get_zip()

string get_zip()

Descripción

Devuelve los datos del archivo Zip, en general, no necesitará este método a menos que desee hacer algo único con los datos

Valores devueltos

Contenido del archivo Zip

Ejemplo

```
$nom = 'la_vida_del_buscón.txt';
$datos = '... nunca mejora su estado quien muda solamente de lugar, y no de vida y costumbres.';

$this->zip->add_data($nom, $datos);

$zip_file = $this->zip->get_zip();
```

read_dir()

bool read_dir(string $path [, bool $preserve_filepath = TRUE [, string $root_path = NULL]])

Descripción

Le permite comprimir un directorio, y sus contenidos, que ya existe en algún lugar de su servidor, proporcione una ruta al directorio y la clase zip leerá recursivamente y lo volverá a crear como un archivo Zip, todos los archivos contenidos dentro de la ruta de acceso suministrada serán codificados, al igual que cualquier subdirectorio que contenga

Parámetros

path
: Ruta al directorio

preserve_filepath
: Si se debe mantener la ruta original

root_path
: Parte de la ruta para excluir del directorio de archivos

Valores devueltos

TRUE en el éxito, FALSE si falla

Ejemplo

```
$path = '/path/a/su/directorio/';

$this->zip->read_dir($path);

// Descarga el archivo a su escritorio. lo llama "mi_copia.zip"
$this->zip->download('mi_copia.zip');
```

Por defecto, el archivo Zip colocará todos los directorios listados en el primer parámetro dentro del zip, si desea ignorar el árbol que precede al directorio de destino, puede pasar FALSE en el segundo parámetro:

```
$path = '/path/a/su/directorio/';

$this->zip->read_dir($path, FALSE);
```

Esto creará un ZIP con un directorio llamado **"directorio"** , con todos sus subdirectorios almacenados, pero no incluirá **/path/a/su/** como parte de la ruta

read_file()

bool read_file(string $path [, mixed $archive_filepath = FALSE])

Descripción

Le permite comprimir un archivo que ya existe en algún lugar de su servidor, proporcione una ruta de archivo y la clase zip lo leerá y lo agregará al archivo

Parámetros

path

Ruta al archivo

archive_filepath

Nuevo nombre de archivo/ruta (string) o (booleano) si se debe mantener la ruta de archivo original

Valores devueltos

TRUE en el éxito, FALSE si falla

Ejemplo

```
$path = '/path/a/foto.jpg';

$this->zip->read_file($path);

//Descargua el archivo a su escritorio. lo llama "mi_copia.zip"
$this->zip->download('mi_copia.zip');
```

Si desea que el archivo Zip mantenga la estructura de directorio, pase TRUE (booleano) en el segundo parámetro:

```
$path = '/path/a/foto.jpg';

$this->zip->read_file($path, TRUE);

//Descargua el archivo a su escritorio. lo llama "mi_copia.zip"
$this->zip->download('mi_copia.zip');
```

En el ejemplo anterior, **foto.jpg** se colocará en el directorio **/path/a/**

También puede especificar un nuevo nombre (ruta incluida) para el archivo agregado sobre la marcha:

```
$path = '/path/a/foto.jpg';
$nuevo_path = '/nuevo/path/alguna_foyo.jpg';

$this->zip->read_file($path, $nuevo_path);

// Descargar el archivo ZIP que contiene /nuevo/path/alguna_foyo.jpg
$this->zip->download('mi_archivo.zip');
```

Bases de datos

Inicio rápido: código de ejemplo

Aquí tiene un código de ejemplo que muestra cómo se usa la clase database, para obtener detalles completos, lea las páginas que describen cada función

Iniciar la clase Base de datos

El siguiente código carga e inicializa la clase database según su configuración:

```
$this->load->database();
```

Una vez cargada, la clase está lista para ser utilizada como se describe a continuación

Nota: Si todas sus páginas requieren acceso a la Base de datos, puede conectarse automáticamente, vea la página de **conexión** para más detalles

Consulta estándar con resultados múltiples (versión objeto)

```
$consulta = $this->db->query('SELECT nombre, apellidos, email FROM mi_tabla');

foreach ($consulta->result() as $fila)
{
   echo $fila->nombre;
   echo $fila->apellidos;
   echo $fila->email;
}

echo 'Total filas: ' . $query->num_rows();
```

La función **result()** anterior devuelve un array de objetos

Ejemplo: **$fila->nombre**

Consulta estándar con resultados múltiples (versión array)

```
$consulta = $this->db->query('SELECT nombre, apellidos, email FROM mi_tabla');

foreach ($consulta->result_array() as $fila)
{
   echo $fila['nombre'];
   echo $fila['apellidos'];
   echo $fila['email'];
}
```

La función **result_array()** anterior devuelve un array de índices de array estándar

Ejemplo

```
$fila['nombre']
```

Consulta estándar con resultado único

```
$consulta = $this->db->query('SELECT nombre FROM mi_tabla LIMIT 1');
$fila = $consulta->row();
echo $fila->nombre;
```

La función **row()** anterior devuelve un objeto

Ejemplo

```
$fila->nombre
```

Consulta estándar con resultado único (versión array)

```
$consulta = $this->db->query('SELECT nombre FROM mi_tabla LIMIT 1');
$fila = $consulta->row_array();
echo $fila['nombre'];
```

La función **row_array()** anterior devuelve un array

Ejemplo

```
$fila['nombre']
```

Inserción estándar

```
$sql = "INSERT INTO mi_tabla (nombre,apellidos) VALUES (".$this->db->escape($nombre).", "
  .$this->db->escape($apellidos).")";

$this->db->query($sql);
echo $this->db->affected_rows();
```

Consulta con Query Builder

Query Builder le proporciona un medio simplificado para recuperar datos:

```
$consulta = $this->db->get('mi_tabla');

foreach ($consulta->result() as $fila)
{
   echo $fila->nombre;
}
```

La función **get()** anterior recupera todos los resultados de la tabla suministrada, la clase **Query Builder** contiene un conjunto completo de métodos para trabajar con datos

Insertar con Query Builder

```
$datos = array(
   'nombre'    => $nombre,
   'apellidos' => $apellidos,
   'fecha'     => $fecha
);

// Produce: INSERT INTO mi_tabla ( nombre, apellidos,fecha) VALUES ('{$nombre}',
// '{$apellidos}', '{$fecha}')
$this->db->insert('mi_tabla', $datos);
```

Configuración de la Base de datos

CodeIgniter tiene un archivo de configuración que le permite almacenar los valores de conexión de su Base de datos, nombre de usuario, contraseña, nombre de la Base de datos, etc., el archivo de configuración se encuentra en **application/config/database.php,** también puede establecer valores de conexión a la Base de datos para entornos específicos colocando **database.php** en el directorio de configuración del entorno correspondiente

Los valores de configuración se almacena en un array multidimensional con este prototipo:

```
$db['default'] = array(
  'dsn'      => '',
  'hostname' => 'localhost',
  'username' => 'root',
  'password' => '',
  'database' => 'nombre_base_de_datos',
  'dbdriver' => 'mysqli',
  'dbprefix' => '',
  'pconnect' => TRUE,
  'db_debug' => TRUE,
  'cache_on' => FALSE,
  'cachedir' => '',
  'char_set' => 'utf8',
  'dbcollat' => 'utf8_general_ci',
  'swap_pre' => '',
  'encrypt'  => FALSE,
  'compress' => FALSE,
  'stricton' => FALSE,
  'failover' => array()
);
```

Algunos controladores de Base de datos (como PDO, PostgreSQL, Oracle, ODBC) pueden requerir un string DSN completo, si ese es el caso, debe usar los valores de configuración '**dsn**', como si estuviera usando la extensión PHP nativa del controlador, como esta:

```
// PDO
$db['default']['dsn'] = 'pgsql:host=localhost;port=5432;dbname=nombre_base_de_datos';

// Oracle
$db['default']['dsn'] = '//localhost/XE';
```

Nota

- Si no especifica un string DSN para un controlador que lo requiere, CodeIgniter intentará compilarla con el resto de la configuración proporcionada

- Si proporciona un string DSN y le faltan algunas configuraciones válidas ,por ejemplo, el conjunto de caracteres de la Base de datos, que están presentes en el resto de los campos de configuración, CodeIgniter los agregará

También puede especificar failovers para la situación cuando la conexión principal no se puede conectar por algún motivo, estas failovers se pueden especificar configurando la conmutación por error para una conexión como esta:

```
$db['default']['failover'] = array(
 array(
    'hostname' => 'localhost1',
    'username' => '',
    'password' => '',
    'database' => '',
    'dbdriver' => 'mysqli',
    'dbprefix' => '',
    'pconnect' => TRUE,
    'db_debug' => TRUE,
    'cache_on' => FALSE,
    'cachedir' => '',
    'char_set' => 'utf8',
    'dbcollat' => 'utf8_general_ci',
    'swap_pre' => '',
    'encrypt'  => FALSE,
    'compress' => FALSE,
    'stricton' => FALSE
 ),
 array(
    'hostname' => 'localhost2',
    'username' => '',
    'password' => '',
    'database' => '',
    'dbdriver' => 'mysqli',
    'dbprefix' => '',
    'pconnect' => TRUE,
    'db_debug' => TRUE,
    'cache_on' => FALSE,
    'cachedir' => '',
    'char_set' => 'utf8',
    'dbcollat' => 'utf8_general_ci',
    'swap_pre' => '',
    'encrypt'  => FALSE,
    'compress' => FALSE,
    'stricton' => FALSE
 )
);
```

Puede especificar tantos failovers como desee

La razón por la que utilizamos un array multidimensional en lugar de una más simple es permitir almacenar opcionalmente múltiples conjuntos de valores de conexión, si, por ejemplo, ejecuta múltiples entornos, desarrollo, producción, prueba, etc., en una sola instalación, puede configurar un grupo de conexión para cada uno, luego según sea necesario cambiar de grupo
Por ejemplo, para configurar un entorno de **"prueba"**, haría esto:

```
$db['prueba'] = array(
  'dsn'      => '',
  'hostname' => 'localhost',
  'username' => 'root',
  'password' => '',
  'database' => 'nombre_base_de_datos',
  'dbdriver' => 'mysqli',
  'dbprefix' => '',
  'pconnect' => TRUE,
  'db_debug' => TRUE,
  'cache_on' => FALSE,
  'cachedir' => '',
  'char_set' => 'utf8',
  'dbcollat' => 'utf8_general_ci',
  'swap_pre' => '',
  'compress' => FALSE,
  'encrypt'  => FALSE,
  'stricton' => FALSE,
  'failover' => array()
);
```

Luego, para decirle al sistema globalmente que use ese grupo, establecería esta variable ubicada en el archivo de configuración:

```
$active_group = 'prueba';
```

Nota

El nombre **'prueba'** es arbitrario, puede ser lo que quiera, de forma predeterminada, hemos utilizado la palabra **"default"** para la conexión principal, pero también se puede renombrar a algo más relevante para su proyecto

Query Builder

La Clase Query Builder está habilitada o deshabilitada globalmente estableciendo la variable **$query_builder** en el archivo de configuración de la Base de datos en TRUE/FALSE, la configuración predeterminada es TRUE, si no está utilizando la clase **query builder**, establecerlo en FALSE utilizará menos recursos cuando se inicialicen las clases de Base de datos

```
$query_builder = TRUE;
```

Nota

Algunas clases de CodeIgniter, como Sesiones, requieren que **Query Builder** esté habilitado para acceder a ciertas funcionalidades

Explicación de valores de configuración

Valor	Descripción
dsn	La string de conexión DSN (una secuencia de configuración todo en uno)
hostname	El nombre de host de su servidor de Base de datos, frecuentemente es '**localhost**'
username	El nombre de usuario utilizado para conectarse a la Base de datos
password	La contraseña utilizada para conectarse a la Base de datos
database	El nombre de la Base de datos a la que desea conectarse
dbdriver	El tipo de Base de datos. es decir: mysqli, postgre, odbc, etc., debe especificarse en minúsculas
dbprefix	Un prefijo de tabla opcional que se agregará al nombre de la tabla cuando se ejecutan consultas de **Query Builder**, esto permite que varias instalaciones de CodeIgniter compartan una Base de datos
pconnect	TRUE/FALSE (bool): si se debe usar una conexión persistente
db_debug	TRUE/FALSE (bool): si se deben mostrar los errores de la Base de datos
cache_on	TRUE/FALSE (bool): si está habilitado el almacenamiento en caché de consulta, consulte **Database Caching Class**
cachedir	La ruta absoluta del servidor a su directorio de caché de consulta de Base de datos
char_set	El juego de caracteres utilizado para comunicarse con la Base de datos
dbcollat	La intercalación de caracteres utilizada para comunicarse con la Base de datos ***Nota*** Solo se usa en los controladores 'mysql' y 'mysqli'
swap_pre	Prefijo de tabla predeterminado que debe intercambiarse con **dbprefix ,** es útil para aplicaciones distribuidas donde puede ejecutar consultas escritas manualmente, y necesita el prefijo para ser aún personalizable por el usuario final
schema	El esquema de la Base de datos, por defecto es '**public**'. Utilizado por los controladores PostgreSQL y ODBC.

encrypt	Si usar o no una conexión encriptada. - 'mysql' (en desuso), 'sqlsrv' y 'pdo/sqlsrv' aceptan TRUE/FALSE - ' mysqli' y 'pdo/mysql' aceptan un array con las siguientes opciones: 'Ssl_key' - Ruta al archivo de clave privada 'Ssl_cert'- Ruta al archivo de certificado de clave pública 'Ssl_ca' - Ruta al archivo de la autoridad del certificado 'Ssl_capath' - Ruta a un directorio que contiene certificados de CA confiables en formato PEM 'Ssl_cipher' - Lista de cifrados permitidos que se utilizarán para el cifrado, separados por dos puntos (':') 'Ssl_verify' - TRUE / FALSE; Si se debe verificar el certificado del servidor o no (solo 'mysqli')
compress	Si se usa compresión de cliente (solo MySQL)
stricton	TRUE/FALSE (bool): si se fuerzan las conexiones de "Modo estricto", es bueno para garantizar un SQL estricto al desarrollar una aplicación
port	El número de puerto de la Base de datos, para usar este valor, debe agregar una línea al array de configuración de la Base de datos. `$db['default']['port'] = 5432;`

Nota

Dependiendo de la plataforma de Base de datos que esté utilizando (MySQL, PostgreSQL, etc.), no se necesitarán todos los valores, por ejemplo, al usar SQLite no necesitará proporcionar un nombre de usuario o contraseña, y el nombre de la Base de datos será la ruta a su archivo de Base de datos, la información anterior supone que está utilizando MySQL

Conectar con su Base de datos

Hay dos formas de conectarse a una Base de datos:

Conexión automática

Conexión automática carga y crea una instancia de la clase Base de datos con cada carga de página, para habilitar la **"conexión automática"**, agregue la palabra **database** al array de la biblioteca, como se indica en el siguiente archivo:

```
application/config/autoload.php
```

Conexión manual

Si solo algunas de sus páginas requieren conectividad de Base de datos, puede conectarse manualmente a su Base de datos agregando esta línea de código en cualquier método donde sea necesaria, o en el constructor de su clase para hacer que la Base de datos esté disponible globalmente en esa clase:

```
$this->load->database();
```

Si el método no contiene información en el primer parámetro, se conectará al grupo especificado en el archivo de configuración de su Base de datos, comúnmente, este es el método preferido

Parámetros disponibles

Consulte el método **database()** de la **Loader Class**, parámetros:

1º Los valores de conexión de la Base de datos, pasados como un array o un string DSN

2º TRUE/FALSE , si se debe devolver el ID de conexión, consulte **Conexión a múltiples Bases de datos**

3º TRUE/FALSE , si se habilita la clase **Query Builder**, por defecto, establecido en TRUE

Conexión manual a una Base de datos

El primer parámetro de este método se puede usar opcionalmente para especificar un grupo de Base de datos particular de su archivo de configuración, o incluso puede enviar valores de conexión para una Base de datos que no está especificada en su archivo de configuración

Para elegir un grupo específico de su archivo de configuración puede hacer esto:

```
$this->load->database('nombre_grupo');
```

Donde **nombre_grupo** es el nombre del grupo de conexión de su archivo de configuración

Para conectarse manualmente a la Base de datos deseada, puede pasar un array de valores:

```
$config['hostname'] = 'localhost';
$config['username'] = 'mi_username';
$config['password'] = 'mi_password';
$config['database'] = 'mi_database';
$config['dbdriver'] = 'mysqli';
$config['dbprefix'] = '';
$config['pconnect'] = FALSE;
$config['db_debug'] = TRUE;
$config['cache_on'] = FALSE;
$config['cachedir'] = '';
$config['char_set'] = 'utf8';
$config['dbcollat'] = 'utf8_general_ci';
$this->load->database($config);
```

Para obtener información sobre cada uno de estos valores, consulte la página de **configuración**

Nota

Para el controlador PDO, debe usar la configuración **$config['dsn']** en lugar de '**hostname**' y '**database**':

```
$config['dsn'] = 'mysql:host=localhost;dbname=mydatabase';
```

O puede enviar los valores de su Base de datos como un Data Source Name (DSN), el DSN deben tener este prototipo:

```
$dsn = 'dbdriver://username:password@hostname/database';
$this->load->database($dsn);
```

Para anular los valores de configuración predeterminados cuando se conecta con un string DSN, agregue las variables de configuración como un query string

```
$dsn = 'dbdriver://username:password@hostname/database?char_set=utf8&dbcollat=utf8_general_ci
&cache_on=true&cachedir=/path/to/cache';

$this->load->database($dsn);
```

Conexión a múltiples Bases de datos

Si necesita conectarse a más de una Base de datos simultáneamente, puede hacerlo de la siguiente manera:

```
$DB1 = $this->load->database('grupo_uno', TRUE);
$DB2 = $this->load->database('grupo_dos', TRUE);
```

Nota: Cambie las palabras **"grupo_uno"** y **"grupo_dos"** a los nombres de grupo específicos a los que se está conectando, o puede pasar los valores de conexión como se indicó anteriormente

Al establecer el segundo parámetro en TRUE , el método devuelve el objeto de la Base de datos

Nota

Cuando se conecta de esta forma, usará su nombre de objeto para ejecutar comandos en lugar de la sintaxis utilizada en esta guía. En otras palabras, en lugar de ejecutar comandos con:

```
$this->db->query();
$this->db->result();
etc…
```

Usará:

```
$DB1->query();
$DB1->result();
etc…
```

Nota

No necesita crear configuraciones de Bases de datos separadas si solo necesita usar una Base de datos diferente en la misma conexión

Puede cambiar a una Base de datos diferente cuando lo necesite, de esta manera:

```
$this->db->db_select($nombre_de_database2);
```

Reconectar / mantener la conexión activa

Si excede el tiempo de espera de inactividad del servidor de Base de datos mientras realiza una carga pesada de PHP, por ejemplo, procesamiento de una imagen, debe considerar hacer **ping** al servidor utilizando el método **reconnect()** antes de enviar más consultas, lo que puede mantener la conexión viva o restablecerla:

```
$this->db->reconnect();
```

Cerrar manualmente la conexión

Si bien CodeIgniter se ocupa inteligentemente del cierre de las conexiones de su Base de datos, puede cerrar la conexión explícitamente:

```
$this->db->close();
```

Queries - Consultas

Consultas regulares

Para enviar una consulta, use el método de consulta:

```
$this->db->query('AQUÍ SU CONSULTA');
```

El método **query()** devuelve un objeto de resultado de Base de datos cuando se ejecutan consultas de tipo **"lectura"**, que puede usar para mostrar sus resultados, cuando se ejecutan consultas de tipo **"escritura"**, devuelve TRUE o FALSE dependiendo del éxito o el fracaso

Al recuperar datos, normalmente asignará la consulta a su propia variable, como esta:

```
$query = $this->db->query('AQUÍ SU CONSULTA');
```

Consultas simplificadas

El método **simple_query** es una versión simplificada del método **$this->db->query()**, **NO** devuelve un conjunto de resultados de la Base de datos, ni establece el temporizador de consulta, ni compila datos de enlace, ni almacena su consulta para la depuración, simplemente le permite enviar una consulta, la mayoría de los usuarios rara vez usarán esta función

Devuelve lo que retorna la función **"execute"** de los drivers de Base de datos, normalmente, esto es TRUE/FALSE en caso de éxito o fracaso en las consultas de tipo de escritura, como las instrucciones INSERT, DELETE o UPDATE **(que es lo que realmente debería usarse)** y un recurso/objeto en caso de éxito para las consultas con resultados extraíbles :

```
if ($this->db->simple_query('SU CONSULTA'))
{
   echo "Éxito!";
}
else
{
   echo "La consulta ha fallado!";
}
```

Nota

La función **pg_exec()** de PostgreSQL, por ejemplo, siempre devuelve un recurso en caso de éxito, incluso para las consultas de tipo escritura, así que téngalo en cuenta si está buscando un valor booleano

Trabajar con prefijos de Base de datos manualmente

Si ha configurado un prefijo de Base de datos y desea anteponerlo a un nombre de tabla para su uso en una consulta SQL nativa, por ejemplo, puede usar lo siguiente:

```
$this->db->dbprefix('nom_tabla'); // Salida: prefijo_nom_tabla
```

Si por algún motivo desea cambiar el prefijo programáticamente sin necesidad de crear una nueva conexión, puede utilizar este método:

```
$this->db->set_dbprefix('prefijo_nuevo');
$this->db->dbprefix('nom_tabla'); // Salida: prefijo_nuevo_nom_tabla
```

Protección de identificadores

En muchas Bases de datos, es aconsejable proteger los nombres de tablas y campos, por ejemplo, con barras invertidas en MySQL, las consultas de **Query Builder** están protegidas automáticamente, sin embargo, si necesita proteger manualmente un identificador, puede usar:

```
$this->db->protect_identifiers('nombre_tabla');
```

Importante

Aunque Query Builder hará todo lo posible para citar correctamente cualquier nombre de campo y tabla que usted le proporcione, tenga en cuenta que **NO** está diseñado para funcionar con la entrada arbitraria del usuario, **NO** lo alimente con datos de usuario no optimizados

Esta función también agregará un prefijo de tabla a su tabla, suponiendo que tiene un prefijo especificado en su archivo de configuración de Base de datos, para habilitar el prefijo, establezca a TRUE el segundo parámetro:

```
$this->db->protect_identifiers('nombre_tabla', TRUE);
```

Escapar consultas

Es una muy buena práctica de seguridad escapar sus datos antes de enviarlos a su Base de datos, CodeIgniter tiene tres métodos que le ayudan a hacerlo:

1. **$this->db->escape()**, este método determina el tipo de datos para que solo pueda escapar los datos de string, también agrega automáticamente comillas simples alrededor de los datos para que Ud. no tenga que hacerlo:

```
$sql = "INSERT INTO table(título) VALUES(".$this->db->escape($título).")";
```

2. **$this->db->escape_str()**, este método escapa los datos que se le pasan, independientemente del tipo, la mayoría de veces usará la función anterior en lugar de esta

   Use la función de esta manera:

```
$sql = "INSERT INTO table (título) VALUES('".$this->db->escape_str($título)."')";
```

3. **$this->db->escape_like_str()**, este método se debe usar cuando las strings se usarán en condiciones **LIKE** para que los comodines LIKE ('%', '_') en el string también se escapen correctamente:

```
$buscar = 'aumento 20%';
$sql = "SELECT id FROM table WHERE column LIKE '%" . $this->db->escape_like_str($buscar)
       ."%' ESCAPE '!'";
```

Importante

El método **escape_like_str()** usa **'!'** (Signo de exclamación) para escapar los caracteres especiales para las condiciones **LIKE**, debido a que este método escapa strings parciales que envolvería entre comillas usted mismo, **no puede agregar automáticamente ESCAPE '!' por usted**, por lo que tendrá que hacerlo manualmente

Enlazado de consultas

Los enlaces le permiten simplificar la sintaxis de su consulta al permitir que el sistema haga las consultas juntas por usted

Considere el siguiente ejemplo:

```
$sql = "SELECT * FROM alguna_tabla WHERE id = ? AND estado = ? AND autor = ?";
$this->db->query($sql, array(3, 'difunto', 'Kafka'));
```

Los signos de interrogación en la consulta se reemplazan automáticamente con los valores en el array en el segundo parámetro del método de consulta

La vinculación también funciona con arrays, que se transformarán en conjuntos **IN**:

```
$sql = "SELECT * FROM alguna_tabla WHERE id IN ? AND estado = ? AND autor = ?";
$this->db->query($sql, array(array(3, 6), 'difunto', 'Kafka'));
```

La consulta resultante será:

```
SELECT * FROM alguna_tabla WHERE id IN (3,6) AND estado = 'difunto' AND author = 'Kafka'
```

El beneficio secundario del enlazado es que los valores se escapan automáticamente, lo que genera consultas más seguras, no tiene que recordar escapar manualmente los datos; el motor lo hace automáticamente por usted

Manejar errores

$this->db->error();

Si necesita obtener el último error que ha ocurrido, el método **error()** le devolverá un array que contiene su código y mensaje

Aquí tiene un ejemplo rápido:

```
if ( ! $this->db->simple_query('SELECT 'campo_de_ejemplo' FROM 'tabla_de_ejemplo''))
{
   $error = $this->db->error(); // Tiene claves 'código' y 'mensaje'
}
```

Generar resultados de consulta

Hay varias formas de generar resultados de consulta:

- **Arrays de resultados**
- **Filas de resultados**
- **Objetos de resultados personalizados**
- **Métodos de ayuda de resultados**

Arrays de resultados

result()

Este método devuelve el resultado de la consulta como un array de objetos o un array vacío cuando falla, normalmente, usará este método en un ciclo foreach, como este:

```
$query = $this->db->query("SU CONSULTA");

foreach ($query->result() as $fila)
{
   echo $fila->título;
   echo $fila->cabecera;
   echo $fila->cuerpo;
}
```

Es un alias de **result_object()**

También puede pasar un string que representa una clase para instanciar para cada objeto de resultado (nota: esa clase debe estar cargada)

```
query = $this->db->query("SELECT * FROM usuarios;");
foreach ($query->result('Usuario') as $usuario)
{
   echo $usuario->nombre; // atributo
   echo $usuario->invierte_nombre(); // o métodos definidos en la clase 'Usuario'
}
```

result_array()

Este método devuelve el resultado de la consulta como un array puro, o un array vacío cuando no se produce ningún resultado, normalmente, lo usará en un ciclo foreach, como este:

```
$query = $this->db->query("SU CONSULTA");
foreach ($query->result_array() as $fila)
{
   echo $fila['título'];
   echo $fila['cabecera'];
   echo $fila['cuerpo'];
}
```

Filas de resultados

row()

Este método devuelve una sola fila de resultados, si su consulta tiene más de una fila, solo devuelve la primera fila, el resultado se devuelve como un objeto

Aquí tiene un ejemplo de uso:

```
$query = $this->db->query("SU CONSULTA");

$fila = $query->row();

if (isset($fila))
{
   echo $fila->título;
   echo $fila->cabecera;
   echo $fila->cuerpo;
}
```

Si desea que se devuelva una fila específica, puede enviar el número de fila como un dígito en el primer parámetro:

```
$fila = $query->row(5);
```

También puede agregar un segundo parámetro un string, que es el nombre de una clase para instanciar la fila con:

```
$query = $this->db->query("SELECT * FROM usuarios LIMIT 1;");
$fila = $query->row(0, 'Usuario');

echo $fila->nombre; // atributo
echo $fila->invierte_nombre(); // o métodos definidos en la clase 'Usuario'
```

row_array()

Idéntico al método **row()**, excepto que devuelve un array :

```
$query = $this->db->query("SU CONSULTA");

$fila = $query->row_array();

if (isset ($fila))
{
   echo $fila['título'];
   echo $fila['cabecera'];
   echo $fila['cuerpo'];
}
```

Si desea que se devuelva una fila específica, puede enviar el número de fila como un dígito en el primer parámetro:

```
$fila = $query->row_array(5);
```

Además, puede moverse hacia adelante/atrás/primero/último a través de sus resultados usando estas variaciones:

$fila = $query->first_row()
$fila = $query->last_row()
$fila = $query->next_row()
$fila = $query->previous_row()

Por defecto, devuelven un objeto a menos que coloque la palabra "**array**" en el parámetro:

$fila = $query->first_row('array')
$fila = $query->last_row('array')
$fila = $query->next_row('array')
$fila = $query->previous_row('array')

Nota

Todos los métodos anteriores cargarán todo el resultado en la memoria (prefetching), use **unbuffered_row()** para procesar grandes conjuntos de resultados

unbuffered_row ()

Este método devuelve una sola fila de resultados sin recuperar todo el resultado en la memoria como lo hace **row()**, si su consulta tiene más de una fila, devuelve la fila actual y mueve el puntero interno de datos hacia adelante:

```
$query = $this->db->query("SU CONSULTA");

while($fila = $query->unbuffered_row())
{
   echo $fila->título;
   echo $fila->cabecera;
   echo $fila->cuerpo;
}
```

Opcionalmente, puede pasar un **'objeto'** (predeterminado) o un **'array'** para especificar el tipo del valor devuelto:

```
$query->unbuffered_row(); // objeto
$query->unbuffered_row('object'); // objeto
$query->unbuffered_row('array'); // array asociativo
```

Objetos de resultado personalizados

Puede hacer que los resultados se devuelvan como una instancia de una clase personalizada en lugar de **stdClass** o **array**, como lo permiten los métodos **result()** y **result_array()**, esto requiere que la clase ya esté cargada en la memoria, el objeto tendrá todos los valores devueltos desde la Base de datos establecidos como propiedades, si estos han sido declarados y no son públicos, debe proporcionar un método **__set()** para permitir que se establezcan :

```
class Usuario {

  public $id;
  public $email;
  public $nom_usuario;

  protected $ultimo_login;

  public function ultimo_login($formato)
  {
    return $this->ultimo_login->format($formato);
  }

  public function __set($nombre, $valor)
  {
     if ($nombre === 'ultimo_login')
      {
         $this->ultimo_login = DateTime::createFromFormat('U', $valor);
      }
   }

   public function __get($nombre)
   {
      if (isset($this->$nombre))
      {
         return $this->$nombre;
      }
   }
}
```

Además de los dos métodos enumerados a continuación, los siguientes métodos también pueden tomar un nombre de clase para devolver los resultados como: **first_row()**, **last_row()**, **next_row()** y **previous_row()**

custom_result_object()

Devuelve el conjunto de resultados completo como un array de instancias de la clase solicitada, el único parámetro es el nombre de la clase para instanciar :

```
$query = $this->db->query("SU CONSULTA");
$filas = $query->custom_result_object('Usuario');

foreach ($rows as $fila)
{
   echo $fila->id;
   echo $fila->email;
   echo $fila->ultimo_login('Y-m-d');
}
```

custom_row_object ()

Devuelve una sola fila de los resultados de su consulta, el primer parámetro es el número de fila de los resultados, el segundo parámetro es el nombre de clase para instanciar :

```
$query = $this->db->query("SU CONSULTA");

$fila = $query->custom_row_object(0, 'Usuario');

if (isset($fila))
{
   echo $fila->email;                 // atributo
   echo $fila->ultimo_login('Y-m-d'); // método de la clase
}
```

También puede usar el método **row()** exactamente de la misma manera:

```
$fila = $query->custom_row_object(0, 'Usuario');
```

Métodos de ayuda de resultados

num_rows()

El número de filas devuelto por la consulta. Nota: en este ejemplo, **$query** es la variable a la que está asignado el objeto de resultado de la consulta:

```
$query = $this->db->query('SELECT * FROM mi_tabla');

echo $query->num_rows();
```

Nota

No todos los controladores de Base de datos tienen una forma nativa de obtener el número total de filas para un conjunto de resultados, cuando este es el caso, todos los datos son captados previamente y **count()** es llamado manualmente en el array resultante para lograr el mismo resultado

num_fields()

El número de **CAMPOS** (columnas) devueltos por la consulta, asegúrese de llamar al método utilizando su objeto de resultado de consulta:

```
$query = $this->db->query('SELECT * FROM mi_tabla');

echo $query->num_fields();
```

free_result()

Libera la memoria asociada con el resultado y elimina la ID del recurso resultante, normalmente PHP libera su memoria automáticamente al final de la ejecución del script, sin embargo, si está ejecutando una gran cantidad de consultas en una secuencia de comandos en particular, es posible que desee liberar el resultado después de cada resultado de la consulta se ha generado con el fin de reducir el consumo de memoria:

```
$query = $this->db->query('SELECT title FROM my_tabla');

foreach ($query->result() as $fila)
{
   echo $fila->título;
}

$query->free_result();  // El objeto de resultado $query ya no estará disponible

$query2 = $this->db->query('SELECT nombre FROM alguna_tabla');

$fila = $query2->row();
echo $fila->nombre;
$query2->free_result(); // El objeto de resultado $query2 ya no estará disponible
```

data_seek()

Este método establece el puntero interno para la siguiente fila de resultados que se va a buscar, solo es útil en combinación con **unbuffered_row()**

Acepta un valor entero positivo, que por defecto es 0 y devuelve TRUE en caso de éxito o FALSE en caso de error:

```
$query = $this->db->query('SELECT `nombre_de_campo` FROM `nombre_tabla`');
$query->data_seek(5); // Saltar las primeras 5 filas
$fila = $query->unbuffered_row();
```

Nota

No todos los controladores de Base de datos admiten esta característica y devolverán FALSE, en particular, no podrá usarlo con PDO

class CI_DB_result

class CI_DB_result

class CI_DB_result

custom_result_object()

array custom_result_object(string $class_name)

Descripción

Devuelve los resultados de la consulta como un array de filas, donde cada fila es una instancia de la clase especificada

Parámetros

class_name

Nombre de clase para las filas resultantes

Valores devueltos

Array que contiene las filas recuperadas

custom_row_object()

object custom_row_object(int $n, string $type)

Descripción

Devuelve la fila de resultados solicitados como una instancia de la clase solicitada

Parámetros

n

Índice de la fila de resultados a devolver

type

Nombre de la clase para la fila resultante

Valores devueltos

La fila solicitada o NULL si no existe

data_seek()

bool data_seek([int $n = 0])

Descripción

Mueve el puntero de la fila de resultados internos al desplazamiento deseado

Parámetros

n

Índice de la fila de resultados que se devolverá a continuación

Valores devueltos

TRUE en el éxito, FALSE si falla

field_data()

```
array field_data()
```

Descripción

Genera un array de objetos **stdClass** que contienen metadatos de campo

Valores devueltos

Array que contiene metadatos de campo

first_row()

```
mixed first_row([ string $type = 'object'])
```

Descripción

Devuelve la primera fila del conjunto de resultados

Parámetros

type
: Tipo del resultado solicitado array, objeto o nombre de clase

Valores devueltos

Primera fila del conjunto de resultados, o NULL si no existe

free_result()

```
void free_result()
```

Descripción

Libera un conjunto de resultados

Uso: ver **Métodos de ayuda de resultados**

valores devueltos

void

last_row()

```
mixed last_row([ string $type = 'object'])
```

Descripción

Devuelve la última fila del conjunto de resultados

Parámetros

type
: Tipo del resultado solicitado array, objeto o nombre de clase

Valores devueltos

Última fila del conjunto de resultados, o NULL si no existe

list_fields()

```
array list_fields()
```

Descripción

Devuelve una array que contiene los nombres de campo en el conjunto de resultados

Valores devueltos

Array de nombres de columna

next_row()

```
mixed next_row([ string $type = 'object'])
```

Descripción

Devuelve la siguiente fila del conjunto de resultados

Parámetros

type

Tipo del resultado solicitado array, objeto o nombre de clase

Valores devueltos

Fila siguiente del conjunto de resultados, o NULL si no existe

num_fields()

```
int num_fields()
```

Descripción

Devuelve el número de campos en el conjunto de resultados

Uso: ver **Métodos de ayuda de resultados**

Valores devueltos

Número de campos en el conjunto de resultados

num_rows()

```
int num_rows()
```

Descripción

Devuelve el número de filas en el conjunto de resultados

Uso: ver **Métodos de ayuda de resultados**

Valores devueltos

Número de filas en el conjunto de resultados

previous_row()

```
mixed previous_row([ string $type = 'object'])
```

Descripción

Devuelve la fila anterior del conjunto de resultados

Parámetros

type
Tipo del resultado solicitado array, objeto o nombre de clase

Valores devueltos

Fila anterior del conjunto de resultados, o NULL si no existe

result()

```
array result([ string $type = 'object'])
```

Descripción

Un contenedor para los métodos **result_array(), result_object() y custom_result_object()**

Uso: ver **Arrays de resultados**

Parámetros

type
Tipo de resultados solicitados array, objeto o nombre de clase

Valores devueltos

Array que contiene las filas recuperadas

result_array()

```
array result_array()
```

Descripción

Devuelve los resultados de la consulta como un array de filas, donde cada fila es en sí misma un array asociativo
Uso: ver **Arrays de resultados**

Valores devueltos

Array que contiene las filas recuperadas

result_object()

```
array result_object()
```

Descripción

Devuelve los resultados de la consulta como un array de filas, donde cada fila es un objeto de tipo **stdClass**

Uso: ver **Arrays de resultados**

Valores devueltos

Array que contiene las filas recuperadas

row()

```
mixed row([ int $n = 0 [, string $type = 'object']])
```

Descripción

Un contenedor para los métodos **row_array(), row_object() y custom_row_object()**
Uso: ver **Filas de resultados**

Parámetros

n
Índice de la fila de resultados de la consulta que se devolverá

type
Tipo del resultado solicitado array, objeto o nombre de clase

Valores devueltos

La fila solicitada o NULL si no existe

row_array()

```
array row_array([ int $n = 0])
```

Descripción

Devuelve la fila de resultado solicitada como un array asociativo
Uso: ver **Filas de resultados**

Parámetros

n
Índice de la fila de resultados de la consulta que se devolverá

Valores devueltos

La fila solicitada o NULL si no existe

row_object()

```
object row_object([ int $n = 0])
```

Descripción

Devuelve la fila de resultado solicitada como un objeto de tipo **stdClass**
Uso: ver **Filas de resultados**

Parámetros

n
Índice de la fila de resultados de la consulta que se devolverá

Valores devueltos

La fila solicitada o NULL si no existe

set_row()

```
void set_row(mixed $key [, mixed $value = NULL])
```

Descripción

Asigna un valor a una columna en particular

Parámetros

key

Nombre de la columna o array de pares clave/valor

value

Valor para asignar a la columna, **$key** es un solo nombre de campo

valores devueltos

void

unbuffered_row()

```
mixed unbuffered_row([ string $type = 'object'])
```

Descripción

Obtiene la siguiente fila de resultados y la devuelve en la forma solicitada

Uso: ver **Filas de resultados**

Parámetros

type

Tipo del resultado solicitado array, objeto o nombre de clase

Valores devueltos

Fila siguiente del conjunto de resultados o NULL si no existe

Métodos Query Helper

Información de la ejecución de una consulta

$this->db->insert_id()

El número de identificación de inserción al realizar inserciones de Base de datos

> ***Nota***
>
> Si usa el controlador PDO con PostgreSQL, o usa el controlador Interbase, esta función requiere un parámetro **$name**, que especifica la secuencia apropiada para verificar el id de inserción

$this->db->affected_rows()

Muestra el número de filas afectadas, al hacer consultas de tipo **"escritura"** (insertar, actualizar, etc.)

> ***Nota***
>
> En MySQL, **"DELETE FROM TABLE"** devuelve 0 filas afectadas, la clase **database** tiene un pequeño truco que le permite devolver el número correcto de filas afectadas, de forma predeterminada, este truco está habilitado, pero se puede desactivar en el archivo del controlador de la Base de datos

$this->db->last_query()

Devuelve la última consulta que se ejecutó, la string de consulta, no el resultado:

```
// Produce: SELECT * FROM alguna_tabla ....
$str = $this->db->last_query();
```

> ***Nota***
>
> Desactivar **save_queries** en la configuración de su Base de datos hará que esta función no sea útil

Información sobre su Base de datos

$this->db->count_all()

Le permite determinar el número de filas en una tabla en particular, envíe el nombre de la tabla en el primer parámetro:

```
echo $this->db->count_all('mi_tabla'); // Produce un entero, como 25
```

$this->db->platform()

Da salida a la plataforma de Base de datos que se está ejecutando (MySQL, MS SQL, Postgres, etc.):

```
echo $this->db->platform();
```

$this->db->version()

Muestra la versión de la Base de datos que se está ejecutando:

```
echo $this->db->version();
```

Haciendo tus consultas más fáciles

$this->db->insert_string()

Esta función simplifica el proceso de escritura de inserciones de Base de datos, devuelve un string de inserción de SQL correctamente formateada :

```
$datos = array('nombre' => $nombre, 'email' => $email, 'url' => $url);

$str = $this->db->insert_string('nombre_tabla', $datos);
```

El primer parámetro es el nombre de la tabla, el segundo es un array asociativo con los datos que se insertarán, el ejemplo anterior produce:

```
INSERT INTO nombre_tabla (nombre, email, url) VALUES ('Milú', 'milu@ejemplo.com', 'ejemplo.com')
```

Nota

Los valores se escapan automáticamente, lo que genera consultas más seguras

$this->db->update_string()

Esta función simplifica el proceso de escritura de actualizaciones de Bases de datos, devuelve un string de actualización de SQL correctamente formateado :

```
$datos = array ('nombre' => $nombre, 'email' => $email, 'url' => $url);

$where = "personaje_id = 1 AND tipo = 'animal'";

$str = $this->db->update_string('nombre_tabla', $datos, $where);
```

El primer parámetro es el nombre de la tabla, el segundo es un array asociativo con los datos que se actualizarán, y el tercer parámetro es la cláusula "**where**"

El ejemplo anterior produce:

```
UPDATE nombre_tabla SET nombre = 'Milú', email = 'milu@ejemplo.com', url = 'ejemplo.com' WHERE personaje_id
= 1 AND tipo = 'animal'
```

Nota

Los valores se escapan automáticamente, lo que genera consultas más seguras

Query Builder Class

CodeIgniter le da acceso a una clase de generador de consultas, este patrón permite que la información se recupere, inserte y actualice en su Base de datos con un mínimo de secuencias de comandos, en algunos casos, solo se necesitan una o dos líneas de código para realizar una acción de Base de datos. CodeIgniter no requiere que cada tabla de Base de datos sea su propio archivo de clase, en su lugar, proporciona una interfaz más simplificada

Más allá de la simplicidad, una de las principales ventajas de usar las características del **Query Builder** es que le permite crear aplicaciones independientes de la Base de datos, ya que cada adaptador de Base de datos genera la sintaxis de la consulta, también permite consultas más seguras, ya que el sistema escapa automáticamente los valores

Nota

Si tiene la intención de escribir sus propias consultas, puede deshabilitar esta clase en el archivo de configuración de la Base de datos, lo que permite que la biblioteca y el adaptador de la Base de datos central utilicen menos recursos

Seleccionar datos

Las siguientes funciones le permiten construir sentencias **SQL SELECT**

$this->db->get()

Ejecuta la consulta de selección y devuelve el resultado, puede usarse solo para recuperar todos los registros de una tabla:

```
$query = $this->db->get('mi_tabla');

// Produce: SELECT * FROM mi_tabla
```

Los parámetros segundo y tercero le permiten establecer una cláusula **LIMIT**:

```
$query = $this->db->get('mi_tabla', 10, 20);

// Ejecuta: SELECT * FROM mi_tabla LIMIT 20, 10
// (en MySQL. Otras bases de datos tienen una sintaxis ligeramente diferente)
```

Notará que la función anterior está asignada a una variable llamada **$query**, que se puede usar para mostrar los resultados:

```
$query = $this->db->get('mi_tabla');

foreach ($query->result() as $fila)
{
   echo $fila->título;
}
```

Lea **Generar resultados de consulta** para una discusión completa con respecto a la generación de resultados

$this->db->get_compiled_select()

Compila la consulta de selección como **$this->db->get()** pero no ejecuta la consulta, este método simplemente devuelve la consulta SQL como un string :

```
$sql = $this->db->get_compiled_select('mi_tabla');
echo $sql;  // Imprime el string: SELECT * FROM mi_tabla
```

El segundo parámetro le permite establecer si la consulta se borra, de manera predeterminada se borra, al igual que cuando se usa

$this->db->get() :

```
echo $this->db->limit(10,20)->get_compiled_select('mi_tabla', FALSE);

// Imprime el string: SELECT * FROM mi_tabla LIMIT 20, 10
// (en MySQL. Otras bases de datos tienen una sintaxis ligeramente diferente)

echo $this->db->select('título, contenido, fecha')->get_compiled_select();

// Imprime el string: SELECT título, contenido, fecha FROM mi_tabla LIMIT 20, 10
```

La clave para observar en el ejemplo anterior es que la segunda consulta no utilizó **$this->db->from()** y no pasó un nombre de tabla al primer parámetro, el motivo de este resultado es que la consulta no se ha ejecutado usando **$this->db->get()** que borra los valores o se han borrado directamente usando **$this->db->reset_query()**

$this->db->get_where()

Idéntico a la función anterior excepto que le permite agregar una cláusula **"where"** en el segundo parámetro, en lugar de usar la función **db->where()**:

```
$query = $this->db->get_where('mi_tabla', array('id' => $id), $limit, $offset);
```

Lea el método **where()** a continuación para obtener más información

Nota
get_where() se conocía anteriormente como getwhere(), que se ha eliminado

$this->db->select()

Le permite escribir la parte **SELECT** de su consulta:

```
$this->db->select('título, contenido, fecha');
$query = $this->db->get('mi_tabla');

// Ejecuta: SELECT título, contenido, fecha FROM mi_tabla
```

Nota

Si selecciona todo **(*)** de una tabla, no necesita usar esta forma, cuando se omiten, CodeIgniter asume que desea seleccionar todos los campos y automáticamente agrega **'SELECT *'**

$this->db->select() acepta un segundo parámetro opcional, si lo configura en FALSE, CodeIgniter no intentará proteger sus nombres de campo o de tabla, es útil si necesita una declaración de selección compuesta donde el escape automático de campos puede romperlos :

```
$this->db->select('(SELECT SUM(pagos.importe) FROM pagos WHERE pagos.factura_id=4) AS
   importe_pagado', FALSE);
$query = $this->db->get('mi_tabla');
```

$this->db->select_max()

Escribe "**SELECT MAX(campo)**" en su consulta, opcionalmente puede incluir un segundo parámetro para cambiar el nombre del campo resultante:

```
$this->db->select_max('edad');
$query = $this->db->get('miembros');

// Produce: SELECT MAX(edad) as edad FROM miembros

$this->db->select_max('edad', 'miembro_edad');
$query = $this->db->get('miembros');

// Produce: SELECT MAX(edad) as miembro_edad FROM miembros
```

$this->db->select_min()

Escribe "**SELECT MIN(campo)**" en su consulta, al igual que con **select_max()**, puede incluir opcionalmente un segundo parámetro para cambiar el nombre del campo resultante:

```
$this->db->select_min('edad');
$query = $this->db->get('miembros');

// Produce: SELECT MIN(edad) as edad FROM miembros
```

$this->db->select_avg()

Escribe "**SELECT AVG(campo)**" en su consulta, al igual que con **select_max()**, puede incluir opcionalmente un segundo parámetro para cambiar el nombre del campo resultante:

```
$this->db->select_avg('edad');
$query = $this->db->get('miembros');

// Produce: SELECT AVG(edad) as edad FROM miembros
```

$this->db->select_sum()

Escribe "**SELECT SUM(campo)**" en su consulta, al igual que con **select_max()**, puede incluir opcionalmente un segundo parámetro para cambiar el nombre del campo resultante:

```
$this->db->select_sum('edad');
$query = $this->db->get('miembros');

// Produce: SELECT SUM(edad) as edad FROM miembros
```

$this->db->from()

Le permite escribir la parte **FROM** de su consulta:

```
$this->db->select('título, contenido, fecha');
$this->db->from('mi_tabla');
$query = $this->db->get();

// Produce: SELECT título, contenido, fecha FROM mi_tabla
```

> ***Nota***
>
> Como se mostró anteriormente, la porción **FROM** de su consulta se puede especificar en la función **$this->db->get()**, por lo tanto, utilice el método que prefiera

$this->db->join()

Le permite escribir la parte **JOIN** de su consulta:

```
$this->db->select('*');
$this->db->from('blogs');
$this->db->join('opiniones', 'opiniones.id = blogs.id');
$query = $this->db->get();

// Produce:
// SELECT * FROM blogs JOIN opiniones ON opiniones.id = blogs.id
```

Se pueden realizar llamadas a múltiples funciones si necesita varias combinaciones en una consulta

Si necesita un tipo específico de **JOIN**, puede especificarlo a través del tercer parámetro de la función, las opciones son: **left**, **right**, **outer**, **inner**, **left outer** y **right outer**

```
$this->db->join('opiniones', 'opiniones.id = blogs.id', 'left');

// Produce: LEFT JOIN opiniones ON opiniones.id = blogs.id
```

Buscando datos específicos

$this->db->where()

Esta función le permite establecer cláusulas **WHERE** usando uno de los cuatro métodos:

Nota
Todos los valores pasados se escapan automáticamente, produciendo consultas más seguras

1. Método simple de clave / valor:

```
$this->db->where('nombre', $nombre); // Produce: WHERE nombre = 'Asterix'
```

Observe que el **signo igual** se agrega por usted

Si usa múltiples llamadas a función, éstas se unen con **AND** entre ellas:

```
$this->db->where('nombre', $nombre);
$this->db->where('tribu', $tribu);
$this->db->where('país', $país);

// WHERE nombre = 'Asterix' AND tribu = 'Galos' AND país = 'Francia'
```

2. Método personalizado de clave / valor:

Puede incluir un operador en el primer parámetro para controlar la comparación:

```
$this->db->where('nombre !=', $nombre);
$this->db->where('id <', $id);

// Produces: WHERE nombre != 'Asterix' AND id < 45
```

3. Método de array asociativa:

```
$array = array('nombre' => $nombre, 'tribu' => $tribu, 'país' => $país);
$this->db->where($array);

// Produces: WHERE nombre = 'Asterix' AND tribu = 'Galos' AND país = 'Francia'
```

Puede incluir sus propios operadores usando este método también:

```
$array = array('nombre !=' => $nombre, 'id <' => $id, 'fecha >' => $fecha);
$this->db->where($array);
```

4. String personalizado:

Puede escribir sus propias cláusulas de forma manual:

```
$where = "nombre='Asterix' AND tribu='Galos' OR país='Francia'";
$this->db->where($where);
```

$this->db->where() acepta un tercer parámetro opcional, si lo configura en FALSE, CodeIgniter no intentará proteger sus nombres de campo o tabla :

```
$this->db->where('MATCH (campo) AGAINST ("valor")', NULL, FALSE);
```

$this->db->or_where()

Esta función es idéntica a la anterior, excepto que varias instancias se unen mediante **OR**:

```
$this->db->where('nombre !=', $nombre);
$this->db->or_where('id >', $id);  // Produce: WHERE nombre != 'Asterix' OR id > 50
```

Nota

or_where() se conocía anteriormente como orwhere(), que se ha eliminado

$this->db->where_in()

Genera un **WHERE campo IN('item', 'item')** en la consulta SQL unida con **AND** si corresponde:

```
$nombres = array('Mortadelo', 'Filemón', 'Dr_Bacterio');
$this->db->where_in('nom_usuario', $nombres);

// Produce:  WHERE nom_usuario IN ('Mortadelo', 'Filemón', 'Dr_Bacterio')
```

$this->db->or_where_in()

Genera un **WHERE campo IN('item', 'item')** en la consulta unida con **OR** si es apropiado:

```
$nombres = array('Mortadelo', 'Filemón', 'Dr_Bacterio');
$this->db->or_where_in('nom_usuario', $nombres);

// Produce: OR nom_usuario IN ('Mortadelo', 'Filemón', 'Dr_Bacterio')
```

$this->db->where_not_in()

Genera un **WHERE campo NOT IN('item', 'item')** en la consulta unida con **AND** si corresponde:

```
$nombres = array('Mortadelo', 'Filemón', 'Dr_Bacterio');
$this->db->where_not_in('nom_usuario', $nombres);

// Produce: WHERE nom_usuario NOT IN ('Mortadelo', 'Filemón', 'Dr_Bacterio')
```

$this->db->or_where_not_in()

Genera un **WHERE NOT IN('item', 'item')** en la consulta unida con **OR** si corresponde:

```
$nombres = array('Mortadelo', 'Filemón', 'Dr_Bacterio');
$this->db->or_where_not_in('nom_usuario', $nombres);

// Produces: OR nom_usuario  NOT IN ('Mortadelo', 'Filemón', 'Dr_Bacterio')
```

Buscando datos similares

$this->db->like()

Este método le permite generar cláusulas **LIKE**, útiles para realizar búsquedas

Nota
Todos los valores pasados se escapan automáticamente

1. Método simple de clave / valor:

```
$this->db->like('cabecera','coincidencia');

// Produce: WHERE 'cabecera' LIKE '%coincidencia%' ESCAPE '!'
```

Si usa varias llamadas a métodos, se encadenan con **AND** entre ellas:

```
$this->db->like('cabecera', 'coincidencia1');
$this->db->like('opiniones', 'coincidencia2');

// WHERE 'cabecera' LIKE '%coincidencia1%' ESCAPE '!' AND 'opiniones' LIKE '%coincidencia2% ESCAPE '!'
```

Si desea controlar dónde se coloca el comodín (**%**), puede usar un tercer argumento opcional, sus opciones son **'before'** (antes), **'after'** (después) y **'both'** (ambos) que es el valor predeterminado :

```
$this->db->like('cabecera', 'coincidencia', 'before');
// Produce: WHERE 'cabecera' LIKE '%coincidencia' ESCAPE '!'

$this->db->like('cabecera', 'coincidencia', 'after');
// Produce: WHERE 'cabecera' LIKE 'coincidencia%' ESCAPE '!'

$this->db->like('cabecera', 'coincidencia', 'both');
// Produce: WHERE 'cabecera' LIKE '%coincidencia%' ESCAPE '!'
```

2. Método de array asociativa:

```
$array = array('cabecera' => $coincide, 'cuerpo1' => $coincide1, 'cuerpo2' => $coincide2);
$this->db->like($array);

// WHERE 'cabecera' LIKE '%coincide%' ESCAPE '!' AND  'cuerpo1' LIKE '%coincide1%' ESCAPE '!'
// AND  'cuerpo2' LIKE '%coincide2%' ESCAPE '!'
```

$this->db->or_like()

Este método es idéntico al anterior, excepto que varias instancias se unen mediante **OR**:

```
$this->db->like('cabecera', 'coincide1'); $this->db->or_like('cuerpo', $coincide2);

// WHERE 'cabecera' LIKE '%coincide1%' ESCAPE '!' OR  'cuerpo' LIKE '%coincide2%' ESCAPE '!'
```

Nota

or_like() se conocía anteriormente como orlike(), que se ha eliminado

$this->db->not_like()

Este método es idéntico a **like()**, excepto que genera sentencias **NOT LIKE**:

```
$this->db->not_like('cabecera', 'coincide');

// WHERE 'cabecera' NOT LIKE '%coincide% ESCAPE '!'
```

$ this-> db->or_not_like()

Este método es idéntico a **not_like()**, excepto que varias instancias se unen mediante **OR**:

```
$this->db->like('cabecera', 'coincide');
$this->db->or_not_like('cuerpo', 'coincide1');

// WHERE 'cabecera' LIKE '%coincide% OR 'cuerpo' NOT LIKE '%coincide1%' ESCAPE '!'
```

$this->db->group_by()

Le permite escribir la parte **GROUP BY** de su consulta:

```
$this->db->group_by("cabecera"); // Produce: GROUP BY título
```

También puede pasar un array de valores múltiples:

```
$this->db->group_by(array("informe", "fecha"));

// Produce: GROUP BY informe, fecha
```

Nota

group_by() anteriormente se conocía como groupby(), que se ha eliminado

$this->db->distinct()

Agrega la palabra clave **DISTINCT** a una consulta:

```
$this->db->distinct();
$this->db->get('tabla');

// Produce: SELECT DISTINCT * FROM tabla
```

$this->db->having()

Agrega una cláusula **HAVING** a una consulta, separando múltiples llamadas con **AND**

Hay 2 sintaxis posibles:

```
$this->db->having('user_id = 45');  // Produce: HAVING user_id = 45
$this->db->having('user_id',  45);  // Produce: HAVING user_id = 45
```

También puede pasar un array de valores múltiples:

```
$this->db->having(array('cabecera =' => 'Mi cabecera', 'id <' => $id));
// Produce: HAVING cabecera = 'Mi cabecera', id < 45
```

Si está utilizando una Base de datos por la que CodeIgniter escapa las consultas, puede evitar el escape de contenido al pasar un tercer argumento opcional y establecerlo en FALSE:

```
$this->db->having('user_id',  45);
// Produce: HAVING `user_id` = 45 en algunas bases de datos como MySQL

$this->db->having('user_id',  45, FALSE);
// Produce: HAVING user_id = 45
```

$this->db->or_having()

Idéntico a **having()**, solo que separa múltiples cláusulas con **OR**

Ordenar los resultados

$this->db->order_by()

Le permite establecer una cláusula **ORDER BY**, el primer parámetro contiene el nombre de la columna por la que desea ordenar, el segundo le permite establecer la dirección del resultado, las opciones son: **ASC**, **DESC** y **RANDOM** :

```
$this->db->order_by('nombre', 'DESC');

// Produce: ORDER BY 'nombre' DESC
```

También puede pasar su propio string en el primer parámetro:

```
$this->db->order_by('apellido DESC,nombre ASC');

// Produce: ORDER BY 'apellido' DESC,'nombre' ASC
```

O se pueden realizar llamadas a múltiples funciones si necesita múltiples campos:

```
$this->db->order_by('apellido','DESC');
$this->db->order_by('nombre','ASC');

// Produce: ORDER BY 'apellido' DESC,'nombre' ASC
```

Si elige la opción de dirección **RANDOM**, los primeros parámetros se ignorarán, a menos que especifique un valor de inicialización numérico :

```
$this->db->order_by('nombre','RANDOM'); // Produce: ORDER BY RAND()
$this->db->order_by(42,'RANDOM');       // Produce: ORDER BY RAND(42)
```

Nota

- order_by() anteriormente se conocía como orderby(), que se ha eliminado
- La ordenación aleatoria **no es** actualmente **compatible** con **Oracle** y, en su lugar, tendrá **ASC** por defecto

Limitar o contar los resultados

$this->db->limit()

Le permite limitar el número de filas que desea que devuelva la consulta:

```
$this->db->limit(10);  // Produce: LIMIT 10
```

El segundo parámetro le permite establecer un conjunto de resultado:

```
$this->db->limit(10, 20);
// Produce: LIMIT 20, 10 (en MySQL. Otras bases de datos tienen una sintaxis ligeramente
                          diferente)
```

$this->db->count_all_results()

Le permite determinar el número de filas en una consulta particular de Active Record, las consultas aceptarán restrictores de **Query Builder** como **where()**, **or_where()**, **like()**, **or_like()**, etc. :

```
echo $this->db->count_all_results('mi_tabla');  // Produces un entero, como 25
$this->db->like('nombre', 'coincide');
$this->db->from('mi_tabla');
echo $this->db->count_all_results(); // Produces un entero, como 17
```

Sin embargo, este método también restablece los valores de campo que haya pasado en **select()**, si necesita conservarlos, puede pasar FALSE como el segundo parámetro:

```
echo $this->db->count_all_results('mi_tabla', FALSE);
```

$this->db->count_all()

Le permite determinar el número de filas en una tabla en particular, envíe el nombre de la tabla en el primer parámetro :

```
echo $this->db->count_all('mi_tabla'); // Produce un entero, como 25
```

Agrupar consultas

La agrupación de consultas le permite crear grupos de cláusulas **WHERE** encerrándolas entre paréntesis, esto le permitirá crear consultas con cláusulas **WHERE** complejas, los grupos anidados son compatibles:

```
$this->db->select('*')->from('mi_tabla')
 ->group_start()
    ->where('a', 'a')
       ->or_group_start()
            ->where('b', 'b')
            ->where('c', 'c')
       ->group_end()
    ->group_end()
 ->where('d', 'd')
->get();

// Genera:
// SELECT * FROM ('mi_tabla') WHERE ( 'a' = 'a' OR ( 'b' = 'b' AND 'c' = 'c' ) ) AND 'd' = 'd'
```

Nota

Los grupos deben estar equilibrados, asegúrese de que **cada group_start()** coincida con un **group_end()**

$this->db->group_start()

Inicia un nuevo grupo agregando un paréntesis de apertura a la cláusula **WHERE** de la consulta

$this->db->or_group_start()

Inicia un nuevo grupo agregando un paréntesis de apertura a la cláusula **WHERE** de la consulta, prefijándolo con '**OR**'

$this->db->not_group_start()

Inicia un nuevo grupo agregando un paréntesis de apertura a la cláusula **WHERE** de la consulta, prefijándolo con '**NOT**'

$this->db->or_not_group_start()

Inicia un nuevo grupo agregando un paréntesis de apertura a la cláusula **WHERE** de la consulta, prefijándolo con '**OR NOT**'

$this->db->group_end()

Finaliza el grupo actual al agregar un paréntesis de cierre a la cláusula **WHERE** de la consulta

Insertar datos

$this->db->insert()

Genera un string de inserción basada en los datos que proporciona y ejecuta la consulta, puede pasar un array o un objeto a la función

Usando un array:

```
$datos = array (
  'apellido' => 'Mi apellido',
  'nombre'   => 'Mi nombre',
  'fecha'    => 'Mi fecha'
);

$this->db->insert('mi_tabla', $datos);
// Produce: INSERT INTO mi_tabla (apellido, nombre, fecha) VALUES
// ('Mi apellido', 'Mi nombre', 'Mi fecha')
```

El primer parámetro contendrá el nombre de la tabla, el segundo es un array asociativa de valores

Usando un objeto:

```
/ *
clase Mi_clase {
  public $apellido  = 'Mi apellido';
  public $contenido = 'Mi contenido';
  public $fecha     = 'Mi fecha';
}
* /

$objeto = new Mi_clase;
$this->db->insert('mi_tabla', $objeto);
// Produce: INSERT INTO mi_tabla (apellido, contenido, fecha) VALUES
// ('Mi apellido', 'Mi contenido', 'Mi fecha')
```

El primer parámetro contendrá el nombre de la tabla, el segundo es un objeto

Nota
Todos los valores se escapan automáticamente para generar consultas más seguras

$this->db->get_compiled_insert()

Compila la consulta de inserción como **$this->db->insert()** pero no ejecuta la consulta, este método simplemente devuelve la consulta SQL como un string:

```
$datos = array (
  'apellido' => 'Mi apellido',
  'nombre'   => 'Mi nombre',
  'fecha'    => 'Mi fecha'
);

$sql = $this->db->set($datos)->get_compiled_insert('mi_tabla');
echo $ sql;

// Produce un string:
// INSERT INTO mi_tabla ('apellido','nombre', 'fecha') VALUES ('Mi apellido', 'Mi nombre',
//                       'Mi fecha')
```

El segundo parámetro le permite establecer si la consulta del generador de consultas se restablecerá, de manera predeterminada será igual a **$this->db->insert()**:

```
echo $this->db->set('apellido', 'Mi apellido')->get_compiled_insert('mi_tabla', FALSE);
// Produce el string: INSERT INTO mi_tabla ('apellido') VALUES ('Mi apellido')

echo $this->db->set('contenido', 'Mi Contenido')->get_compiled_insert();
// Produce el string: INSERT INTO mi_tabla ('apellido', 'contenido') VALUES
//                                         ('Mi apellido', 'Mi Contenido')
```

La cuestión clave a tener en cuenta en el ejemplo anterior es que la segunda consulta no utilizó **$this->db->from()** ni pasó un nombre de tabla en el primer parámetro, la razón por la que esto funcionó es porque la consulta no se ha ejecutado utilizando **$this->db->insert()** que restablece valores o se restablecieron directamente usando **$this->db->reset_query()**

Nota
Este método no funciona para inserciones por lotes

$this->db->insert_batch()

Genera un string de inserción basada en los datos que proporciona y ejecuta la consulta, puede pasar un array o un objeto a la función:

Usando un array:

```
$datos = array (
 Array(
    'apellido' => 'Mi apellido',
    'nombre'   => 'Mi nombre',
    'fecha'    => 'Mi fecha'
 ),
 Array(
    'apellido' => 'Otro apellido',
    'nombre'   => 'Otro nombre',
    'fecha'    => 'Otra fecha'
 )
);

$this->db->insert_batch('mi_tabla', $datos);

// Produce: INSERT INTO mi_tabla (apellido, nombre, fecha) VALUES
// ('Mi apellido', 'Mi nombre', 'Mi fecha'), ('Otro apellido', 'Otro nombre', 'Otra fecha')
```

El primer parámetro contendrá el nombre de la tabla, el segundo es una array asociativa de valores

Nota

Todos los valores se escapan automáticamente para generar consultas más seguras

Actualización de datos

$this->db->replace()

Este método ejecuta una instrucción **REPLACE**, que es básicamente el estándar SQL para (opcional) **DELETE + INSERT**, utilizando las claves **PRIMARY** y **UNIQUE** como factor determinante, en nuestro caso, le ahorrará la necesidad de implementar lógicas complejas con diferentes combinaciones de llamadas **select()**, **update()**, **delete()** e **insert()** :

```
$datos = array (
  'apellido' => 'Mi apellido',
  'nombre'   => 'Mi nombre',
  'fecha'    => 'Mi fecha'
);

$this->db->replace('tabla', $datos);
// Ejecuta: REPLACE INTO mi_tabla(apellido, nombre, fecha) VALUES
//                              ('Mi apellido', 'Mi nombre', 'Mi fecha')
```

En el ejemplo anterior, si suponemos que el campo **apellido** es nuestra clave principal, si una fila contiene '**Mi apellido**' como el valor del **apellido**, esa fila se eliminará con nuestros nuevos datos de fila reemplazándola

También se permite el uso del método **set()** y todos los campos se escapan automáticamente, al igual que con **insert()**

$this->db->set()

Esta función le permite establecer valores para inserciones o actualizaciones, se puede usar en lugar de pasar un array de datos directamente a las funciones de inserción o actualización:

```
$this->db->set('nombre', $nombre);
$this->db->insert('mi_tabla');
// Produce: INSERT INTO mi_tabla('nombre') VALUES ('{$nombre}')
```

Si usa múltiples funciones llamadas, se ensamblarán correctamente en función de si está haciendo una inserción o una actualización:

```
$this->db->set('nombre', $nombre);
$this->db->set('apellido', $apellido);
$this->db->set('estado', $estado);
$this->db->insert('mi_tabla');
```

set() también aceptará un tercer parámetro opcional **$escape**, que evitará que los datos se escapen si se establece en FALSE, para ilustrar la diferencia, aquí se usa **set()** con y sin el parámetro de escape:

```
$this->db->set('campo', 'campo + 1', FALSE);
$this->db->where('id', 2);
$this->db->update('mi_tabla');
// Produce: UPDATE mi_tabla SET campo = campo + 1 WHERE id = 2

$this->db->set('campo', 'campo + 1');
$this->db->where('id', 2);
$this->db->update('mi_tabla');
// Produce: UPDATE 'mitabla' SET 'campo' = 'campo + 1' WHERE 'id' = 2
```

También puede pasar una array asociativa a esta función:

```
$array = array (
  'nombre'   => $nombre,
  'apellido' => $apellido,
  'estado'   => $estado
);

$this->db->set($array);
$this->db->insert('mi_tabla');
```

O un objeto:

```
/ *
clase Mi_clase {
  public $apellido = 'Mi apellido';
  public $contenido = 'Mi contenido';
  public $fecha = 'Mi fecha';
}
* /
```

```
$objeto = new Mi_clase;
$this->db->set($objeto);
$this->db->insert('mi_tabla');
$this->db->update();
```

Genera un string de actualización y ejecuta la consulta en función de los datos que proporciona, puede pasar un array o un objeto a la función

Usando un array:

```
$datos = array(
  'apellido' => $apellido,
  'nombre'   => $nombre,
  'fecha'    => $fecha
);

$this->db->where('id', $ id);
$this->db->update('mi_tabla', $datos);

// Produce:
// UPDATE mi_tabla
// SET título = '{$apellido}', nombre = '{$nombre}', fecha = '{$fecha}'
// WHERE id = $ id
```

O puede suministrar un objeto:

```
/ *
clase Mi_clase {
 public $apellido = 'Mi apellido';
 public $contenido = 'Mi contenido';
 public $fecha = 'Mi fecha';
}
* /

$objeto = new Mi_clase;
$this->db->where('id', $ id);
$this->db->update('mi_tabla', $objeto);

// Produce:
// UPDATE 'mi_tabla'
// SET 'apellido' = '{$apellido}', 'nombre' = '{$nombre}', 'fecha' = '{$fecha}'
// WHERE id = '$id'
```

Nota

Todos los valores se escapan automáticamente para generar consultas más seguras

Notará el uso de la función **$this->db->where()**, que le permite establecer la cláusula **WHERE**, opcionalmente puede pasar esta información directamente a la función de actualización como un string:

```
$this->db->update('mi_tabla', $fecha, "id = 4");
```

O como un array:

```
$this->db->update('mi_tabla', $fecha, array ('id' => $ id));
```

También puede usar el método **$this->db->set()** cuando realice actualizaciones

$this->db->update_batch()

Genera un string de actualización basada en los datos que proporciona y ejecuta la consulta, puede pasar un array o un objeto a la función

Usando un array:

```
$datos = array(
 Array(
    'apellido' => 'Mi apellido',
    'nombre'   => 'Mi nombre 2',
    'fecha'    => 'Mi fecha 2'
 ),
 Array(
    'apellido' => 'Otro apellido',
    'nombre'   => 'Otro nombre 2',
    'fecha'    => 'Otra fecha 2'
 )
);

$this->db->update_batch('mi_tabla', $datos, 'apellido');

// Produce:
// UPDATE 'mi_tabla' SET 'nombre' = CASE
// WHEN 'apellido' = 'Mi apellido' THEN 'Mi nombre 2'
// WHEN 'apellido' = 'Otro título' THEN 'Otro nombre 2'
// ELSE 'nombre' END, 'fecha' = CASE
// WHEN 'apellido' = 'Mi apellido' THEN 'Mi fecha 2'
// WHEN 'apellido' = 'Otro apellido' THEN 'Otra fecha 2'
// ELSE 'fecha' END
// WHERE 'apellido' IN ('Mi apellido','Otro apellido')
```

El primer parámetro contendrá el nombre de la tabla, el segundo es un array asociativo de valores, el tercer parámetro es la clave **where**.

Nota

- Todos los valores se escapan automáticamente para generar consultas más seguras
- **affected_rows()** no le dará los resultados adecuados con este método, debido a la naturaleza de cómo funciona, en cambio, **update_batch()** devuelve el número de filas afectadas

$this->db->get_compiled_update()

Este método funciona exactamente de la misma manera que **get_compiled_insert()** excepto que produce un string **UPDATE** en lugar de un string **INSERT**

Para obtener más información, consulte la documentación de **get_compiled_insert()**

Nota

Este método no funciona en las actualizaciones por lotes

Eliminar datos

Dispone de los método siguientes:

$this->db->delete()

Genera un string de borrado SQL **DELETE FROM** y ejecuta la consulta:

```
$this->db->delete('mi_tabla', array ('id' => $ id));

// Produce:  DELETE FROM mi_tabla WHERE id = $id
```

El primer parámetro es el nombre de la tabla, el segundo es la cláusula **where**, también puede usar las funciones **where()** o **or_where()** en lugar de pasar los datos al segundo parámetro del método:

```
$this->db->where('id', $ id);
$this->db->delete('mi_tabla');

// Produce:  DELETE FROM mi_tabla WHERE id = $id
```

Se puede pasar un array de nombres de tabla a **delete()** si desea eliminar datos de más de una tabla:

```
$tablas = array('tabla1', 'tabla2', 'tabla3');
$this->db->where('id', '5');
$this->db->delete($tablas);
```

Si desea eliminar todos los datos de una tabla, puede usar la función **empty_table()** o **truncate()**:

$this->db->empty_table()

Genera un string SQL de eliminación y ejecuta la consulta:

```
$this->db->empty_table('mi_tabla'); // Produce: DELETE FROM mi_tabla
```

$this->db->truncate()

Genera una string SQL truncada y ejecuta la consulta:

```
$this->db->from('mi_tabla');
$this->db->truncate();

// o

$this->db->truncate('mi_tabla');  // Produce:  TRUNCATE mi_tabla
```

Nota

Si el comando **TRUNCATE** no está disponible, **truncate()** se ejecutará como "**DELETE FROM tabla**"

$this->db->get_compiled_delete()

Esto funciona exactamente de la misma manera que **$this->db->get_compiled_insert()** excepto que produce un string **DELETE** en lugar de un string **INSERT**

Para obtener más información, consulte la documentación de **$this->db->get_compiled_insert()**

Method chaining

El method chaining le permite simplificar su sintaxis al conectar múltiples funciones. Considere este ejemplo:

```
$query = $this->db->select('título')
                ->where('id', $id)
                ->limit(10, 20)
                ->get('mi_tabla');
```

Caché en Query Builder

Aunque no es un almacenamiento en caché "verdadero", **Query Builder** le permite guardar (o "almacenar en caché") ciertas partes de sus consultas para su reutilización en un momento posterior de la ejecución del script, normalmente, cuando se completa una llamada de **Query Builder**, toda la información almacenada se restablece para la próxima llamada, con el almacenamiento en caché, puede evitar este restablecimiento y reutilizar la información fácilmente

Las llamadas en caché son acumulativas. Si realiza 2 llamadas **select()** en caché, y luego 2 llamadas **select()** no guardadas en caché, esto dará como resultado 4 llamadas de **select()**

Dispone de tres métodos de almacenamiento en caché:

$this->db->start_cache()

Este método debe llamarse para comenzar el almacenamiento en caché, todas las consultas de **Query Builder** del tipo correcto, consulte a continuación las consultas compatibles, se almacenan para su uso posterior

$this->db->stop_cache()

Se puede llamar a este método para detener el almacenamiento en caché

$this->db->flush_cache()

Este método borra todos los elementos del caché del generador de consultas

Un ejemplo de almacenamiento en caché

```
$this->db->start_cache();
$this->db->select('campo1');
$this->db->stop_cache();
$this->db->get('nombre_tabla');
// Genera: SELECT 'campo1' FROM ('nombre_tabla')

$this->db->select('campo2');
$this->db->get('nombre_tabla');
// Genera: SELECT 'campo1', 'campo2' FROM ('nombre_tabla')

$this->db->flush_cache();
$this->db->select('campo2');
$this->db->get('nombre_tabla');
// Genera: SELECT 'campo2' FROM ('nombre_tabla')
```

Nota

Las siguientes declaraciones se pueden almacenar en caché:

SELECT, FROM, JOIN, WHERE, LIKE, GROUP_BY, HAVING, ORDER_BY

Restablecer el generador de consultas

Para restablecer **Query Builder** dispone del método:

$this->db->reset_query()

Restablecer Query Builder le permite comenzar de nuevo con su consulta sin ejecutarla primero usando un método como **$this->db->get()** o **$this->db->insert()**, al igual que los métodos que ejecutan una consulta, esto no restablecerá los elementos que guardó en caché utilizando **Query Builder Caching**

Esto es útil en situaciones en las que esté utilizando **Query Builder** para generar SQL, por ejemplo,

$this->db->get_compiled_select() pero luego elige, por ejemplo, ejecutar la consulta:

```
// Tenga en cuenta que el segundo parámetro del método get_compiled_select es FALSE
$sql = $this->db->select(array('campo1','campo2'))
                                ->where('campo3',5)
                                ->get_compiled_select('mi_tabla', FALSE);

// ...
// Hacer algo loco con el código SQL ... como agregarlo a un script cron para
// ejecución posterior o algo ...
// ...

$datos = $this->db->get()->result_array();

// Ejecutaría y devolvería un array de resultados de la siguiente consulta:
// SELECT campo1, campo1 from mi_tabla where campo3 = 5;
```

Nota

Las llamadas dobles a **get_compiled_select()** mientras usa la funcionalidad **Caching** de **Query Builder** y **NO** restablecer sus consultas dará como resultado que la memoria caché se fusione dos veces, eso a su vez, es decir, si está almacenando en caché un **select()** selecciona el mismo campo dos veces

class CI_DB_query_builder

class CI_DB_query_builder

class CI_DB_query_builder

count_all_results()

int count_all_results([string $table = '' [, bool $reset = TRUE]])

Descripción

Genera un string de consulta específica de la plataforma que cuenta todos los registros devueltos por una consulta del generador de consultas

Parámetros

table

Nombre de la tabla

reset

Si restablecer valores para **SELECT**

Valores devueltos

Número de filas en el resultado de la consulta

dbprefix()

string dbprefix([string $table = ''])

Descripción

Antepone un prefijo de Base de datos, si existe uno en la configuración

Parámetros

table

El nombre de la tabla, a la que se anteponer el prefijo

Valores devueltos

El nombre de la tabla prefijada

delete()

mixed delete([mixed $table = '' [, string $where = '' [, int $limit = NULL [, bool $reset_data = TRUE]]]])

Descripción

Compila y ejecuta una consulta **DELETE**

Parámetros

table

La tabla(s) para borrar; string o array

where

La cláusula **WHERE**

limit

La cláusula **LIMIT**

reset_data

TRUE para restablecer la cláusula "**write**" de la consulta

Valores devueltos

Instancia CI_DB_query_builder (method chaining) o FALSE en caso de error

distinct()

object distinct([bool $val = TRUE])

Descripción

Agrega una cláusula **DISTINCT** a la parte **SELECT** de la consulta

Parámetros

val

Valor deseado de la bandera **"DISTINCT"**

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

empty_table()

bool empty_table([string $table = ''])

Descripción

Elimina todos los registros de una tabla a través de una declaración **DELETE**

Parámetros

table

Nombre de la tabla

Valores devueltos

TRUE en el éxito, FALSE en la falla

flush_cache()

object flush_cache()

Descripción

Vacía el caché del generador de consultas

Valores devueltos

Instancia de CI_DB_query_builder (method chaining)

from()

object from(mixed $from)

Descripción

Especifica la cláusula **FROM** de una consulta

Parámetros

from

Nombre(s) de la tabla; string o array

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

get()

object get([string $table = '' [, int $limit = NULL [, int $offset = NULL]]])

Descripción

Compila y ejecuta una instrucción **SELECT** basada en los métodos ya llamados de **Query Builder**

Parámetros

table

La tabla para consultar

limit

La cláusula **LIMIT**

offset

La cláusula **OFFSET**

Valores devueltos

Instancia CI_DB_result (method chaining)

get_compiled_delete()

```
string get_compiled_delete([ string $table = '' [, bool $reset = TRUE]])
```

Descripción

Compila una declaración **DELETE** y la devuelve como un string

Parámetros

table

Nombre de la tabla

reset

Si se restablecen los valores de QB actuales o no

Valores devueltos

La instrucción SQL compilada como una string

get_compiled_insert()

```
string get_compiled_insert([ string $table = '' [, bool $reset = TRUE]])
```

Descripción

Compila una instrucción **INSERT** y la devuelve como un string

Parámetros

table

Nombre de la tabla

reset

Si se restablecen los valores de QB actuales o no

Valores devueltos

La instrucción SQL compilada como una string

get_compiled_select()

```
string get_compiled_select([ string $table = '' [, bool $reset = TRUE]])
```

Descripción

Compila una instrucción **SELECT** y la devuelve como un string

Parámetros

table

Nombre de la tabla

reset

Si se restablecen los valores de QB actuales o no

Valores devueltos

La instrucción SQL compilada como una string

get_compiled_update()

```
string get_compiled_update([ string $table = '' [, bool $reset = TRUE]])
```

Descripción

Compila una instrucción **UPDATE** y la devuelve como un string

Parámetros

table

Nombre de la tabla

reset

Si se restablecen los valores de QB actuales o no

Valores devueltos

La instrucción SQL compilada como una string

get_where()

```
object get_where([ mixed $table = '' [, string $where = NULL [, int $limit = NULL [, int $offset = NULL]]]])
```

Descripción

Igual que **get()**, pero también permite que **WHERE** se agregue directamente

Parámetros

table

La (s) tabla (s) de donde obtener datos; string o array

where

La cláusula **WHERE**

limit

La cláusula **LIMIT**

offset

La cláusula **OFFSET**

Valores devueltos

Instancia CI_DB_result (method chaining)

group_by()

```
object group_by(mixed $by [, bool $escape = NULL])
```

Descripción

Agrega una cláusula **GROUP BY** a una consulta

Parámetros

by

Campo (s) para agrupar; string o array

escape

Si se deben escapar valores e identificadores

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

group_end()

```
object group_end()
```

Descripción

Termina una expresión de grupo

Valores devueltos

Instancia DB_query_builder

group_start()

```
object group_start()
```

Descripción

Inicia una expresión de grupo, usando **AND** para las condiciones dentro de él

Valores devueltos

Instancia de CI_DB_query_builder (method chaining)

having()

```
object having(mixed $key [, string $value = NULL [, string $escape = NULL]])
```

Descripción

Agrega una cláusula **HAVING** a una consulta, separando múltiples llamadas con **AND**

Parámetros

key

Identificador (string) o array asociativo de pares campo / valor

value

Valor buscado si **$key** es un identificador

escape

Si se deben escapar valores e identificadores

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

insert()

bool insert([string $table = '' [, array $set = NULL [, bool $escape = NULL]]])

Descripción

Compila y ejecuta una instrucción **INSERT**

Parámetros

table
: Nombre de la tabla

set
: Una array asociativa de pares campo / valor

escape
: Si se deben escapar valores e identificadores

Valores devueltos

TRUE en el éxito, FALSE en la falla

insert_batch()

mixed insert_batch(string $table [, array $set = NULL [, bool $escape = NULL [, int $batch_size = 100]]])

Descripción

Compila y ejecuta las instrucciones **INSERT** por lotes

Parámetros

table
: Nombre de la tabla

set
: Datos para insertar

escape
: Si se deben escapar valores e identificadores

batch_size
: Recuento de filas para insertar a la vez

Valores devueltos

Número de filas insertadas o FALSE en caso de error

Nota

Cuando se proporcionan **$batch_size**, se ejecutarán varias consultas **INSERT**, cada una de las cuales intenta insertar filas hasta **$batch_size**

join()

object join(string $table, string $cond [, string $type = '' [, bool $escape = NULL]])

Descripción

Agrega una cláusula **JOIN** a una consulta

Parámetros

table
: Nombre de tabla para unirse

cond
: La condición **JOIN ON**

type
: El tipo JOIN

escape
: Si se deben escapar valores e identificadores

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

like()

object like(string $field [, string $match = '' [, string $side = 'both' [, bool $escape = NULL]]])

Descripción

Agrega una cláusula **LIKE** a una consulta, separando múltiples llamadas con **AND**

Parámetros

field
: Nombre del campo

match
: Parte del texto para hacer coincidir

side
: En qué lado de la expresión colocar el comodín '%'

escape
: Si se deben escapar valores e identificadores

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

limit()

object limit(int $value [, int $offset = 0])

Descripción

Agrega cláusulas **LIMIT** y **OFFSET** a una consulta

Parámetros

value
: Número de filas para limitar los resultados

offset

Número de filas para saltar

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

not_group_start()

object not_group_start()

Descripción

Inicia una expresión grupal, usando **AND NOT** para las condiciones dentro de ella

Valores devueltos

Instancia de CI_DB_query_builder (method chaining)

not_like()

object not_like(string $field [, string $match = '' [, string $side = 'both' [, bool $escape = NULL]]])

Descripción

Agrega una cláusula **NOT LIKE** a una consulta, separando múltiples llamadas con **AND**

Parámetros

field

Nombre del campo

match

Parte del texto para hacer coincidir

side

En qué lado de la expresión colocar el comodín '%'

escape

Si se deben escapar valores e identificadores

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

offset()

object offset(int $offset)

Descripción

Agrega una cláusula **OFFSET** a una consulta

Parámetros

offset

Número de filas para saltar

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

order_by()

object order_by(string $orderby [, string $direction = '' [, bool $escape = NULL]])

Descripción

Agrega una cláusula **ORDER BY** a una consulta

Parámetros

orderby
Campo para ordenar

direction
La orden solicitada **ASC**, **DESC** o **RANDOM**

escape
Si se deben escapar valores e identificadores

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

or_group_start()

object or_group_start()

Descripción

Inicia una expresión de grupo, utilizando **OR** para las condiciones dentro de él

Valores devueltos

Instancia de CI_DB_query_builder (method chaining)

or_having()

object or_having(mixed $key [, string $value = NULL [, bool $escape = NULL]])

Descripción

Agrega una cláusula **HAVING** a una consulta, separando múltiples llamadas con **OR**

Parámetros

key
Identificador (string) o array asociativo de pares campo / valor

value
Valor buscado si **$key** es un identificador

escape
Si se deben escapar valores e identificadores

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

or_like()

object or_like(string $field [, string $match = '' [, string $side = 'both' [, bool $escape = NULL]]])

Descripción

Agrega una cláusula **LIKE** a una consulta, separando clases múltiples con **OR**

Parámetros

field
: Nombre del campo

match
: Parte del texto para hacer coincidir

side
: En qué lado de la expresión colocar el comodín '%'

escape
: Si se deben escapar valores e identificadores

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

or_not_group_start()

object or_not_group_start()

Descripción

Inicia una expresión de grupo, usando **O NOT** para las condiciones dentro de él

Valores devueltos

Instancia de CI_DB_query_builder (method chaining)

or_not_like()

object or_not_like(string $field [, string $match = '' [, string $side = 'both' [, bool $escape = NULL]]])

Descripción

Agrega una cláusula **NOT LIKE** a una consulta, separando múltiples llamadas con **OR**

Parámetros

field
: Nombre del campo

match
: Parte del texto para hacer coincidir

side
: En qué lado de la expresión colocar el comodín '%'

escape

Si se deben escapar valores e identificadores

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

or_where()

```
object or_where(mixed $key [, mixed $value = NULL [, bool $escape = NULL]])
```

Descripción

Genera la parte **WHERE** de la consulta, separa llamadas múltiples con **OR**

Parámetros

key

Nombre del campo a comparar, o array asociativo

value

Si es una sola clave, en comparación con este valor

escape

Si se deben escapar valores e identificadores

Valores devueltos

Instancia DB_query_builder

or_where_in()

```
object or_where_in([ string $key = NULL [, array $values = NULL [, bool $escape = NULL]]])
```

Descripción

Genera **WHERE campo IN('item','item')** enlazada con **OR** si corresponde

Parámetros

key

El campo para buscar

values

Los valores buscados

escape

Si se deben escapar valores e identificadores

Valores devueltos

Instancia DB_query_builder

or_where_not_in()

```
object or_where_not_in([string $key = NULL  [, array $values = NULL  [,  bool  $escape = NULL
]]])
```

Descripción

Genera un **WHERE campo NOT IN('item', 'item')** en la consulta unida con **OR** si corresponde

Parámetros

key

El campo para buscar

values

Los valores buscados

escape

Si se deben escapar valores e identificadores

Valores devueltos

Instancia DB_query_builder

replace()

```
bool replace([string $table = '' [, array $set = NULL]])
```

Descripción

Compila y ejecuta una instrucción **REPLACE**

Parámetros

table

Nombre de la tabla

set

Un array asociativa de pares campo / valor

Valores devueltos

TRUE en el éxito, FALSE en la falla

reset_query()

```
object reset_query()
```

Descripción

Restablece el estado actual del generador de consultas, útil cuando se quiere construir una consulta que se puede cancelar bajo ciertas condiciones

Valores devueltos

Instancia de CI_DB_query_builder (method chaining)

select()

```
object select([string $select = '*' [, bool $escape = NULL]])
```

Descripción

Agrega una cláusula **SELECT** a una consulta

Parámetros

select

La parte **SELECT** de una consulta

escape

Si se deben escapar valores e identificadores

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

select_avg()

object select_avg([string $select = '' [, string $alias = '']])

Descripción

Agrega una cláusula **SELECT AVG(campo)** a una consulta

Parámetros

select

Campo para calcular el promedio de

alias

Alias para el nombre del valor resultante

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

select_max()

object select_max([string $select = '' [, string $alias = '']])

Descripción

Agrega una cláusula **SELECT MAX (campo)** a una consulta

Parámetros

select

Campo para calcular el máximo de

alias

Alias para el nombre del valor resultante

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

select_min()

object select_min([string $select = '' [, string $alias = '']])

Descripción

Agrega una cláusula **SELECT MIN (campo)** a una consulta

Parámetros

select
Campo para calcular el mínimo de

alias
Alias para el nombre del valor resultante

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

select_sum()

```
object select_sum([ string $select = '' [, string $alias = '']])
```

Descripción

Agrega una cláusula **SELECT SUM(campo)** a una consulta

Parámetros

select
Campo para calcular la suma

alias
Alias para el nombre del valor resultante

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

set()

```
object set(mixed $key [, string $value = '' [, bool $escape = NULL]])
```

Descripción

Agrega pares de campo / valor que se pasarán más tarde a **insert()**, **update()** o **replace()**

Parámetros

key
Nombre de campo, o un array de pares de campo / valor

value
Valor de campo, si **$key** es un solo campo

escape
Si se deben escapar valores e identificadores

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

set_dbprefix()

string set_dbprefix([string $prefix = ''])

Descripción

Establece el prefijo de la Base de datos, sin tener que volver a conectar

Parámetros

prefix
: El nuevo prefijo a usar

Valores devueltos

El prefijo DB en uso

set_insert_batch()

object set_insert_batch(mixed $key [, string $value = '' [, bool $escape = NULL]])

Descripción

Agrega pares de campo / valor para ser insertados en una tabla más tarde a través de **insert_batch()**

Parámetros

key
: Nombre del campo o una array de pares de campo / valor

value
: Valor de campo, si **$key** es un solo campo

escape
: Si se deben escapar valores e identificadores

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

set_update_batch()

object set_update_batch(mixed $key [, string $value = ' '[, bool $escape = NULL]])

Descripción

Agrega pares de campo / valor que se actualizarán en una tabla más tarde mediante **update_batch()**

Parámetros

key
: Nombre del campo o una array de pares de campo / valor

value
: Valor de campo, si **$key** es un solo campo

escape
: Si se deben escapar valores e identificadores

Valores devueltos

Instancia CI_DB_query_builder (method chaining)

start_cache()

object start_cache()

Descripción

Inicia el caché del generador de consultas

Valores devueltos

Instancia de CI_DB_query_builder (method chaining)

stop_cache()

object stop_cache()

Descripción

Detiene el caché del generador de consultas

Valores devueltos

Instancia de CI_DB_query_builder (method chaining)

truncate()

bool truncate([string $table = ''])

Descripción

Ejecuta una declaración **TRUNCATE** en una tabla

Parámetros

table

Nombre de la tabla

Valores devueltos

TRUE en el éxito, FALSE en la falla

update()

bool update([string $table = '' [, array $set = NULL [, string $where = NULL [, int $limit = NULL]]]])

Descripción

Compila y ejecuta una instrucción **UPDATE**

Parámetros

table

Nombre de la tabla

set

Un array asociativo de pares campo / valor

where

La cláusula **WHERE**

limit

La cláusula **LIMIT**

Valores devueltos

TRUE en el éxito, FALSE en la falla

update_batch()

mixed update_batch(string $table [, array $set = NULL [, string $value = NULL [, int $batch_size = 100]]])

Descripción

Compila y ejecuta instrucciones **UPDATE** por lotes

Parámetros

table

Nombre de la tabla

set

Nombre de campo, o una array asociativa de pares de campo / valor

value

Valor del campo, si **$set** es un solo campo

batch_size

Recuento de condiciones para agrupar en una sola consulta

Valores devueltos

Número de filas actualizadas o FALSE en caso de error

Nota

Cuando se proporcionan pares de campo / valor de más de **$batch_size**, se ejecutarán múltiples consultas, cada una manejando pares de campo / valor de **$batch_size**

where()

object where(mixed $key [, mixed $value = NULL [, bool $escape = NULL]])

Descripción

Genera la parte **WHERE** de la consulta, separa llamadas múltiples con **AND**

Parámetros

key

Nombre del campo a comparar, o array asociativo

value

Si es una sola clave, en comparación con este valor

escape

Si se deben escapar valores e identificadores

Valores devueltos

Instancia DB_query_builder

where_in()

object where_in([string $key = NULL [, array $values = NULL [, bool $escape = NULL]]])

Descripción

Genera un **WHERE campo IN('item', 'item')** en la consulta SQL unida con **AND** si corresponde

Parámetros

key

Nombre del campo a examinar

values

Array de valores objetivo

escape

Si se deben escapar valores e identificadores

Valores devueltos

Instancia DB_query_builder

where_not_in()

object where_not_in([string $key = NULL [, array $values = NULL [, bool $escape = NULL]]])

Descripción

Genera **WHERE campo NOT IN('item', 'item')**, junto con **AND** si corresponde

Parámetros

key

El campo para buscar

values

Los valores buscados

escape

Si se deben escapar valores e identificadores

Valores devueltos

Instancia DB_query_builder

Transacciones

La abstracción de la Base de datos de CodeIgniter le permite usar transacciones con Bases de datos que admiten tipos de tablas seguras para transacciones. En MySQL, necesitará ejecutar tipos de tabla InnoDB o BDB en lugar de los más comunes MyISAM, la mayoría de las otras plataformas de Bases de datos admiten transacciones de forma nativa

Si no está familiarizado con las transacciones, le recomendamos que busque un buen recurso en línea para obtener información sobre ellas para su Base de datos en particular, la siguiente información supone que tiene un conocimiento básico de las transacciones

Enfoque de CodeIgniter a las transacciones

CodeIgniter utiliza un enfoque para las transacciones que es muy similar al proceso utilizado por la popular clase de Base de datos ADODB, elegimos ese enfoque porque simplifica enormemente el proceso de ejecución de transacciones, en la mayoría de los casos, todo lo que se requiere son dos líneas de código

Tradicionalmente, las transacciones han requerido una gran cantidad de trabajo para implementarse, ya que le exigen que realice un seguimiento de sus consultas y determine si realizarlas o deshacerlas en función del éxito o el fracaso de sus consultas, esto es particularmente engorroso con consultas anidadas, por el contrario, hemos implementado un sistema de transacción inteligente que hace todo esto por usted automáticamente, también puede administrar sus transacciones manualmente si lo desea, pero realmente no hay ningún beneficio

Ejecutando Transacciones

Para ejecutar sus consultas utilizando transacciones, usará las funciones **$this->db->trans_start()** y **$this->db->trans_complete()** de la siguiente manera:

```
$this->db->trans_start();
$this->db->query('UNA CONSULTA SQL ...');
$this->db->query('OTRA CONSULTA ...');
$this->db->query ('Y AÚN OTRA CONSULTA ...');
$this->db->trans_complete();
```

Puede ejecutar tantas consultas como desee entre las funciones de inicio / finalización y todas se confirmarán o retrotraerán según el éxito o el fracaso de una consulta dada

Modo estricto

Por defecto, CodeIgniter ejecuta todas las transacciones en modo estricto, cuando el modo estricto está habilitado, si está ejecutando varios grupos de transacciones, si un grupo falla, todos los grupos se revertirán, si el modo estricto está deshabilitado, cada grupo se trata de manera independiente, lo que significa que una falla de un grupo no afectará a ninguna otra

El modo estricto se puede desactivar de la siguiente manera:

```
$this->db->trans_strict(FALSE);
```

Gestión de errores

Si tiene habilitada la generación de informes de errores en su archivo **config/database.php**, verá un mensaje de error estándar si la confirmación no tuvo éxito, si la depuración está desactivada, puede administrar sus propios errores de esta manera:

```
$this->db->trans_start();
$this->db->query('UNA CONSULTA SQL ...');
$this->db->query('OTRA CONSULTA ...');
$this->db->trans_complete ();

if($this->db->trans_status() === FALSE)
{
   // genera un error ... o usaR la función log_message() para registrar su error
}
```

Deshabilitar transacciones

Si desea deshabilitar las transacciones, puede hacerlo usando **$this->db->trans_off()**:

```
$this->db->trans_off();

$this->db->trans_start ();
$this->db->query('UNA CONSULTA SQL ...');
$this->db->trans_complete();
```

Cuando las transacciones están deshabilitadas, sus consultas se confirmarán automáticamente, al igual que cuando se ejecutan consultas sin transacciones, prácticamente ignorando las llamadas a **trans_start()**, **trans_complete()**, etc.

Modo de prueba

Opcionalmente, puede poner el sistema de transacción en "**modo de prueba**", lo que hará que sus consultas se retrotraigan, incluso si las consultas producen un resultado válido, para usar el modo de prueba, simplemente configure el primer parámetro del método **$this->db->trans_start()** en TRUE:

```
$this->db->trans_start(TRUE); // La consulta se revertirá
$this->db->query('UNA CONSULTA SQL ...');
$this->db->trans_complete();
```

Ejecutando Transacciones Manualmente

Si desea ejecutar transacciones manualmente, puede hacerlo de la siguiente manera:

```
$this->db->trans_begin();

$this->db->query('UNA CONSULTA SQL ...');
$this->db->query('OTRA CONSULTA ...');
```

```
$this->db->query('Y AÚN OTRA CONSULTA ...');

if ($this->db->trans_status() === FALSE)
{
   $this->db->trans_rollback();
}
else
{
   $this->db->trans_commit();
}
```

Nota

Asegúrese de utilizar **$this->db->trans_begin()** al ejecutar transacciones manuales, **NO $this->db->trans_start()**

Obtener Metadatos

Metadatos de Tablas

Estas funciones le permiten obtener información de la tabla

Enumera las tablas en su Base de datos

$this->db->list_tables();

Devuelve una array que contiene los nombres de todas las tablas en la Base de datos a la que está conectado actualmente:

```
$tablas = $this->db->list_tables();

foreach ($tablas as $tabla)
{
   echo $tabla;
}
```

Determinar si existe una tabla

$this->db->table_exists();

A veces es útil saber si existe una tabla en particular antes de ejecutar una operación en ella, devuelve un valor booleano TRUE / FALSE:

```
if ($this->db->table_exists('nombre_tabla'))
{
   // algún código ...
}
```

Nota
Reemplace **nombre_tabla** con el nombre de la tabla que está buscando

Campo metadata

Listar los campos en una tabla

$this->db->list_fields()

Devuelve un array que contiene los nombres de los campos

Esta consulta se puede llamar de dos maneras:

1. Puede proporcionar el nombre de la tabla y llamarlo desde **$this->db->objeto**:

```
$campos = $this->db->list_fields('nombre_tabla');

foreach ($campos como $campo)
{
   echo $campo;
}
```

2. Puede recopilar los nombres de campo asociados con cualquier consulta que ejecute llamando a la función desde su objeto de resultado de consulta:

```
$query = $this->db->query('SELECT * FROM alguna_tabla');

foreach ($query->list_fields() as $campo)
{
   echo $campo;
}
```

Determinar si un campo está presente en una tabla

$this->db->field_exists()

Algunas veces es útil saber si un campo en particular existe antes de realizar una acción, devuelve un valor booleano TRUE / FALSE:

```
if ($this->db->field_exists('nombre_campo','nombre_tabla'))
{
   // algún código ...
}
```

Nota

Reemplace **nombre_campo** con el nombre de la columna que está buscando y reemplace **nombre_tabla** con el nombre de la tabla que está buscando

Recuperar metadatos de campo

$this->db->field_data()

Devuelve una array de objetos que contiene información de campo

A veces es útil recopilar los nombres de los campos u otros metadatos, como el tipo de columna, la longitud máxima, etc.

Nota

No todas las Bases de datos proporcionan metadatos

Ejemplo

```
$campos = $this->db->field_data('nombre_tabla');

foreach ($campos as $campo)
{
   echo $campo->name;
   echo $campo->type;
   echo $campo->max_length;
   echo $campo->primary_key;
}
```

Si ya ha ejecutado una consulta, puede usar el objeto resultado en lugar de proporcionar el nombre de la tabla:

```
$query = $this->db->query("SU CONSULTA");
$campos = $query->field_data();
```

La siguiente información está disponible desde esta función si la admite su Base de datos:

- **name** - nombre de la columna
- **max_length** - longitud máxima de la columna
- **primary_key** - si la columna es una clave principal
- **type** - el tipo de la columna

Llamadas a funciones personalizadas

$this->db->call_function();

Esta función le permite llamar a las funciones de la Base de datos PHP que no se incluyen de forma nativa en CodeIgniter, de una manera independiente de la plataforma, por ejemplo, supongamos que desea llamar a la función **mysql_get_client_info()**, que no es compatible nativamente con CodeIgniter

Podría hacerlo así:

```
$this->db->call_function('get_client_info');
```

Debe proporcionar el nombre de la función, **sin el prefijo mysql_**, en el primer parámetro, el prefijo se agrega automáticamente en función del controlador de Base de datos que se esté utilizando actualmente, esto le permite ejecutar la misma función en diferentes plataformas de Bases de datos, obviamente, no todas las llamadas a funciones son idénticas entre plataformas, por lo que existen límites a la utilidad de esta función en términos de portabilidad

Cualquier parámetro que necesite la función que está llamando se agregará al segundo parámetro :

```
$this->db->call_function('alguna_funcion', $parametro1, $paramétro2, etc.);
```

A menudo, deberá proporcionar una ID de conexión de Base de datos o una ID de resultado de la Base de datos, se puede acceder a la ID de conexión usando:

```
$this->db->conn_id;
```

Se puede acceder a la ID del resultado desde su objeto de resultado, como este:

```
$query = $this->db->query("ALGUNA CONSULTA");

$query->result_id;
```

Clase de caché para la Base de datos

La clase de caché de la Base de datos le permite almacenar sus consultas en caché como archivos de texto para reducir la carga de la Base de datos

Importante
Esta clase se inicializa automáticamente por el controlador de la Base de datos cuando se habilita el almacenamiento en caché. **NO cargue esta clase manualmente**.

Importante
No todas las funciones de resultado de consulta están disponibles cuando utiliza el almacenamiento en caché, por favor lea esta sección cuidadosamente

Habilitar el almacenamiento en caché

El almacenamiento en caché está habilitado en tres pasos:

1. Cree un directorio grabable en su servidor donde se pueden almacenar los archivos de caché
2. Establezca la ruta a su directorio de caché en su archivo: **application/config/database.php**
3. Habilite la función de almacenamiento en caché, ya sea de forma global configurando la preferencia en su archivo **application/config/database.php**, o manualmente como se describe a continuación

Una vez habilitado, el almacenamiento en caché se realizará automáticamente siempre que se cargue una página que contenga consultas a Bases de datos

Cómo funciona el almacenamiento en caché?

El sistema de caché de consultas de CodeIgniter ocurre dinámicamente cuando se visualizan sus páginas, cuando el almacenamiento en caché está habilitado, la primera vez que se carga una página web, el objeto de resultado de la consulta se serializará y se almacena en un archivo de texto en su servidor; la próxima vez que se cargue la página, se usará el archivo de caché en lugar de acceder a su Base de datos, el uso de la Base de datos se puede reducir efectivamente a cero para cualquier página que se haya guardado en caché

Solo las consultas de tipo de lectura ,**SELECT,** pueden almacenarse en caché, ya que son el único tipo de consultas que producen un resultado, las consultas de tipo de escritura, **INSERT, UPDATE**, etc., ya que no generan un resultado, no serán almacenadas en caché por el sistema

Los archivos de caché **NO caducan**, cualquier consulta que se haya almacenado en caché permanecerá en la memoria caché hasta que la elimine, el sistema de almacenamiento en caché le permite borrar cachés asociados con páginas individuales, o puede eliminar toda la colección de archivos de caché, normalmente, querrá utilizar las funciones de

limpieza que se describen a continuación para eliminar los archivos de caché después de que ocurran ciertos eventos, como cuando ha agregado información nueva a su Base de datos

Caching mejorará el rendimiento de su sitio?

Obtener una ganancia de rendimiento como resultado del almacenamiento en caché depende de muchos factores, si tiene una Base de datos altamente optimizada con muy poca carga, probablemente no verá un aumento de rendimiento, si su Base de datos está bajo uso intensivo, probablemente verá una respuesta mejorada, suponiendo que su sistema de archivos no esté excesivamente cargado

Recuerde que el almacenamiento en caché simplemente cambia la forma en que se recupera su información, al pasar de ser una operación de Base de datos a una de sistema de archivos

En algunos entornos de servidores en clúster, por ejemplo, el almacenamiento en caché puede ser perjudicial ya que las operaciones del sistema de archivos son muy intensas, en servidores únicos en entornos compartidos, el almacenamiento en caché probablemente sea beneficioso, lamentablemente, no hay una sola respuesta a la pregunta de si debe almacenar en caché su Base de datos, depende de la situación particular

Cómo se almacenan los archivos de caché?

CodeIgniter coloca el resultado de **CADA consulta** en su propio archivo de caché, los conjuntos de archivos de caché están organizados en subdirectorios correspondientes a los métodos de su controlador, para ser precisos, los subdirectorios se nombran de manera idéntica a los primeros dos segmentos de su URI, el nombre de la clase del controlador y el nombre de la función

Por ejemplo, supongamos que tiene un controlador llamado **blog** con un método llamado **comentarios** que contiene tres consultas, el sistema de almacenamiento en caché creará un directorio de caché llamado **blog + comentarios**, en la que escribirá tres archivos de caché

Si utiliza consultas dinámicas que cambian en función de la información en su URI, por ejemplo, cuando usa la paginación, cada instancia de la consulta generará su propio archivo de caché, es posible, por lo tanto, terminar con muchos más archivos de caché que consultas

Administrar sus archivos de caché

Como los archivos de caché no caducan, deberá crear rutinas de eliminación en su aplicación, por ejemplo, supongamos que tiene un blog que permite comentarios de los usuarios, siempre que se envíe un nuevo comentario, querrá eliminar los archivos de caché asociados con el método del controlador que sirve sus comentarios, encontrará dos funciones de eliminación que se describen a continuación que le ayudarán a borrar los datos

No todos los métodos de la Base de datos funcionan con el almacenamiento en caché

Por último, debemos señalar que el objeto de resultado que se almacena en caché es una versión simplificada del objeto de resultado completo, por ese motivo, algunas de las funciones de resultado de la consulta no están disponibles para su uso

Los siguientes métodos **NO están disponibles** cuando se utiliza un objeto de resultados en caché:

- **num_fields()**
- **field_names()**
- **field_data()**
- **free_result()**

Además, los dos recursos de la Base de datos, **result_id** y **conn_id**, no están disponibles cuando se almacenan en caché, ya que los recursos de resultados solo pertenecen a las operaciones en tiempo de ejecución

Referencia de funciones

$this->db->cache_on() / $this->db->cache_off()

Activa o desactiva manualmente el almacenamiento en caché, puede ser útil si desea evitar que ciertas consultas se almacenen en caché:

```
// Activar el almacenamiento en caché
$this->db->cache_on();
$query = $this->db->query("SELECT * FROM mi_tabla");

// Desactiva el almacenamiento en caché para esta consulta
$this->db->cache_off();
$query = $this->db->query("SELECT * FROM clientes WHERE cliente_id = '$usuario_actual'");

// Vuelva a activar el almacenamiento en caché
$this->db->cache_on();
$query = $this->db->query("SELECT * FROM otra_tabla");
```

$this->db->cache_delete()

Elimina los archivos de caché asociados con una página en particular, es útil si necesita borrar el almacenamiento en caché después de actualizar su Base de datos

El sistema de almacenamiento en caché guarda sus archivos de caché en carpetas que corresponden al URI de la página que está viendo, por ejemplo, si está viendo una página en: **ejemplo.com/index.php/blog/opiniones**, el sistema de caché colocará todos los archivos de caché asociados en un directorio llamado **blog + opiniones**

Para eliminar esos archivos de caché particulares, usará:

```
$this->db->cache_delete('blog', 'opiniones');
```

Si no usa ningún parámetro, el URI actual se usará para determinar qué debe eliminarse

$this->db->cache_delete_all()

Borra todos los archivos de caché existentes:

```
$this->db->cache_delete_all();
```

Database Forge Class - Administrar la Base de Datos

La clase de Database Forge contiene métodos que lo ayudan a administrar su Base de datos

Iniciar la clase Forge

Importante

Para inicializar la clase Forge, su controlador de Base de datos ya debe estar ejecutándose, ya que la clase **Forge** se basa en él

Cargue la clase Forge de la siguiente manera:

```
$this->load->dbforge()
```

También puede pasar otro objeto de Base de datos al cargador de DB **Forge**, en caso de que la Base de datos que desea administrar no sea la predeterminada:

```
$this->myforge = $this->load->dbforge($this->otra_db, TRUE);
```

En el ejemplo anterior, estamos pasando un objeto de Base de datos personalizado como primer parámetro y luego le pedimos que devuelva el objeto **dbforge**, en lugar de asignarlo directamente a **$this->dbforge**

Nota

Ambos parámetros se pueden usar individualmente, simplemente pase un valor vacío como el primero si desea omitirlo

Una vez inicializado, accederá a los métodos utilizando el objeto **$this->dbforge**:

```
$this->dbforge->algun_metodo();
```

Crear y eliminar Bases de datos

$this->dbforge->create_database('nombre_db')

Le permite crear la Base de datos especificada en el primer parámetro, devuelve TRUE/FALSE según el éxito o el fracaso:

```
if ($this->dbforge->create_database('mi_db'))
{
    echo 'Base de datos creada!';
}
```

$this->dbforge->drop_database('nombre_db')

Le permite borrar la Base de datos especificada en el primer parámetro, devuelve TRUE/FALSE según el éxito o el fracaso:

```
if ($this->dbforge->drop_database('mi_db'))
{
   echo 'Base de Datos borrada!';
}
```

Crear y borrar tablas

Hay varias cosas que puede hacer crear tablas. agregar campos, agregar claves a la tabla, modificar columnas, CodeIgniter proporciona un mecanismo para esto

Agregar campos

Los campos se crean a través de un array asociativo, dentro del array debe incluir una clave **'type'** que se relacione con el tipo de datos del campo, por ejemplo, **INT, VARCHAR, TEXT**, etc., muchos tipos de datos, por ejemplo **VARCHAR**, también requieren una clave '**CONSTRAINT**':

```
$campos = array(
  'usuarios' => array(
     'type'       => 'VARCHAR',
     'constraint' => '100',
   ),
);
// cambiara a "usuarios VARCHAR(100)" cuando se agrege el campo
```

Además, se pueden usar las siguientes claves/valores:

- **unsigned/true** : para generar "**UNSIGNED**" en la definición del campo
- **default/value** : para generar un valor predeterminado en la definición del campo
- **null/true** : para generar "**NULL**" en la definición de campo. Sin esto, el campo predeterminado será "**NOT NULL**"
- **auto_increment/true** : genera un indicador de auto_increment en el campo. Tenga en cuenta que el tipo de campo debe ser un tipo que lo admita, como un integer
- **unique/true** : para generar una clave única para la definición del campo

```
$campos = array(
 'blog_id' => array(
    'type'           => 'INT',
    'constraint'     => 5,
    'unsigned'       => TRUE,
    'auto_increment' => TRUE
 ),
```

```
  'blog_title' => array(
     'type'       => 'VARCHAR',
     'constraint' => '100',
     'unique'     => TRUE,
  ),
  'blog_author' => array(
     'type'       =>'VARCHAR',
     'constraint' => '100',
     'default'    => 'King of Town',
  ),
  'blog_description' => array(
     'type' => 'TEXT',
     'null' => TRUE,
  ),
 );
```

Después de que se hayan definido los campos, se pueden agregar usando **$this->dbforge->add_field($campos)**; seguido de una llamada al método **create_table()**

$this->dbforge->add_field()

El método de añadir campos aceptará el array anterior

Pasando strings como campos

Si sabe exactamente cómo quiere que se cree un campo, puede pasar el string con las definiciones de campo con **add_field()**:

```
$this->dbforge->add_field("label varchar (100) NOT NULL DEFAULT 'default label'");
```

Nota

- Pasar string sin formato como campos, no puede seguir con llamadas **add_key()** en esos campos
- Las llamadas múltiples a **add_field()** son acumulativas

Crear un campo ID

Hay una excepción especial para crear campos **id**., un campo de este tipo se asignará automáticamente como **INT(9) UTO_INCREMENT Primary Key**:

```
$this->dbforge->add_field('id');
// Produce: ID INT(9) NOT NULL AUTO_INCREMENT
```

Agregar índices

En general, querrá que su tabla tenga índices, esto se logra con **$this->dbforge->add_key('campo')**, un segundo parámetro opcional configurado en TRUE lo convertirá en un índice principal, tenga en cuenta que **add_key()** debe ir seguido de una llamada a **create_table()**:

Los índices no primarios de varias columnas se deben enviar como un array

El siguiente ejemplo es para MySQL:

```
$this->dbforge->add_key('blog_id', TRUE);
// Produce: PRIMARY KEY 'blog_id' ('blog_id')

$this->dbforge->add_key('blog_id', TRUE);
$this->dbforge->add_key('sitio_id', TRUE);
// Produce: PRIMARY KEY 'blog_id_sitio_id' ('blog_id', 'sitio_id')

$this->dbforge->add_key('blog_nombre');
// Produce: KEY 'blog_nombre' ('blog_nombre')

$this->dbforge->add_key(array('blog_nombre', 'blog_label'));
// Produce: KEY 'blog_nombre_blog_label' ('blog_nombre', 'blog_label')
```

Crear una tabla

Después de que se hayan declarado los campos y los índices, puede crear una nueva tabla con:

```
$this->dbforge->create_table('nombre_tabla');
// Produce: CREATE TABLE nombre_tabla
```

Un segundo parámetro opcional establecido en TRUE agrega una cláusula **IF NOT EXISTS** en la definición:

```
$this->dbforge->create_table('nombre_tabla', TRUE);
// Produce: CREATE TABLE IF NOT EXISTS nombre_tabla
```

También puede pasar atributos de tabla opcionales, como **ENGINE** de MySQL:

```
$atributos = array('ENGINE' =>'InnoDB');
$this->dbforge->create_table('nombre_tabla', FALSE, $atributos);
// Produce: CREATE TABLE 'nombre_tabla' (...) ENGINE = InnoDB DEFAULT
// CHARACTER SET utf8 COLLATE utf8_general_ci
```

Nota

A menos que especifique los atributos **CHARACTER SET** y / o **COLLATE**, **create_table()** siempre los agregará con los valores configurados de **char_set** y **dbcollat**, siempre que no estén vacíos (solo MySQL)

Borrar una tabla

Ejecuta una instrucción **DROP TABLE** y opcionalmente agregue una cláusula **IF EXISTS** :

```
$this->dbforge->drop_table('nombre_tabla');
// Produce: DROP TABLE nombre_tabla

$this->dbforge->drop_table('nombre_tabla',TRUE);
// Produce: DROP TABLE IF EXISTS nombre_tabla
```

Renombrar una tabla

Ejecuta **TABLE RENAME** :

```
$this->dbforge->rename_table('nombre_antiguo', 'nombre_nuevo');
// Produce: ALTER TABLE nombre_antiguo RENAME TO nombre_nuevo
```

Modificar tablas

Agregar una columna a una tabla

$this->dbforge->add_column()

El método **add_column()** se usa para modificar una tabla existente, se puede usar para un número ilimitado de campos adicionales:

```
$campos = array(
  'preferences' => array('type' => 'TEXT')
);
$this->dbforge->add_column('nombre_tabla', $campos);
// Produce: ALTER TABLE nombre_tabla ADD preferences TEXT
```

Si está utilizando MySQL o CUBIRD, puede aprovechar sus cláusulas **AFTER** y **FIRST** para ubicar la nueva columna:

```
// Colocará la nueva columna después de la columna 'otro_campo':
$campos = array(
  'preferences' => array('type' => 'TEXT', 'after' => 'otro_campo')
);

// Colocará la nueva columna al comienzo de la definición de la tabla:
$campos = array(
  'preferences' => array('type' => 'TEXT', 'first' => TRUE)
);
```

Borrar una columna de la tabla

$this->dbforge->drop_column()

Se usa para eliminar una columna de una tabla:

```
$this->dbforge->drop_column('nombre_tabla', 'columna_a_borrar');
```

Modificar una columna de la tabla

$this->dbforge->modify_column()

El uso de este método es idéntico a **add_column()**, excepto que altera una columna existente en lugar de agregar una nueva, para cambiar el nombre, puede agregar una clave "**name**" en el campo que define el array:

```
$campos = array(
  'old_name'=> array(
    'name' => 'nuevo_nombre',
    'type' => 'TEXT',
   ),
);
$this->dbforge->modify_column('nombre_tabla', $campos);
// Produce: ALTER TABLE nombre_tabla CHANGE antiguo_nombre nuevo_nombre TEXT
```

class CI_DB_forge

class CI_DB_forge

class CI_DB_forge

add_column()

bool add_column(string $table [, array $field = array() [, string $_after = NULL]])

Descripción

Agrega una columna a una tabla, se usa para modificar una tabla existente, acepta el mismo conjunto de campos que el anterior, y se puede usar para un número ilimitado de campos adicionales
Uso: ver **Agregar una columna a una tabla**

Parámetros

table
Nombre de tabla para agregar la columna
field
Definición de columna
_after
Columna para la cláusula AFTER (obsoleta)

Valores devueltos

TRUE en el éxito, FALSE si falla

add_field()

object add_field(array $field)

Descripción

Agrega un campo al conjunto que se usará para crear una tabla

Uso: ver **Agregar campos**

Parámetros

field
: Definición de campo para agregar

Valores devueltos

Instancia de CI_DB_forge (method chaining)

add_key()

```
object add_key(array $key [, bool $primary = FALSE])
```

Descripción

Agrega un índice al conjunto que se usará para crear una tabla

Uso: Ver **Agregar índice**

Parámetros

key
: Nombre de un campo clave

primary
: Establézcalo en TRUE si debe ser una clave principal o una normal, los índices no primarios de varias columnas se deben enviar como un array

Valores devueltos

Instancia de CI_DB_forge (method chaining)

create_database()

```
bool create_database(string $db_name)
```

Descripción

Crea una nueva Base de datos

Uso: vea **Crear y eliminar Bases de datos**

Parámetros

db_name
: Nombre de la Base de datos para crear

Valores devueltos

TRUE en el éxito, FALSE si falla

create_table()

```
bool create_table(string $table [, string $if_not_exists = FALSE [, array $attributes = array()]] )
```

Descripción

Crea una nueva tabla

Uso: ver **Crear una tabla**

Parámetros

table
: Nombre de la tabla para crear

if_not_exists
: Establézcalo en TRUE para agregar una cláusula **IF NOT EXISTS**

attributes
: Un array asociativo de atributos de tabla

Valores devueltos

TRUE en el éxito, FALSE si falla

drop_column()

bool drop_column(string $table, array $column_name)

Descripción

Borra una columna de una tabla

Uso: vea **Borrar una columna de la tabla**

Parámetros

table
: Nombre de la tabla

column_name
: El nombre de la columna a borrar

Valores devueltos

TRUE en el éxito, FALSE si falla

drop_database()

bool drop_database(string $db_name)

Descripción

Borra una Base de datos

Uso: vea **Crear y eliminar Bases de datos**

Parámetros

db_name
: Nombre de la Base de datos a borrar

Valores devueltos

TRUE en el éxito, FALSE si falla

drop_table()

bool drop_table(string $table_name [, string $if_exists = FALSE])

Descripción

Borra una tabla

Uso: vea **Borrar una tabla**

Parámetros

table

Nombre de la tabla a borrar

if_exists

Establézcalo en TRUE para agregar una cláusula **IF EXISTS**

Valores devueltos

TRUE en el éxito, FALSE si falla

modify_column()

bool modify_column(string $table, array $field)

Descripción

Modifica una columna de tabla

Uso: Ver **Modificar una columna de una tabla**

Parámetros

table

Nombre de la tabla

field

Definición de columna

Valores devueltos

TRUE en el éxito, FALSE si falla

rename_table()

bool rename_table(string $table_name, string $new_table_name)

Descripción

Cambia el nombre de una tabla

Uso: ver **Cambiar el nombre de una tabla**

Parámetros

table

Nombre actual de la tabla

new_table_name

Nuevo nombre de la tabla

Valores devueltos

TRUE en el éxito, FALSE si falla

Clase de utilidades para Base de datos

La clase de utilidad de Base de datos contiene métodos que lo ayudan a administrar su Base de datos

Iniciar la Clase de Utilidad

Importante

Para inicializar la clase su controlador de Base de datos ya debe estar ejecutándose, ya que la clase **dbutil** se basa en él

Cargue la clase de utilidad de la siguiente manera:

```
$this->load->dbutil();
```

También puede pasar otro objeto de Base de datos al cargador de **Utility DB**, en caso de que la Base de datos que desea administrar no sea la predeterminada:

```
$this->miutil = $this->load->dbutil($this->otra_db, TRUE);
```

En el ejemplo anterior, estamos pasando un objeto de Base de datos personalizado como primer parámetro y luego le pedimos que devuelva el objeto **dbutil**, en lugar de asignarlo directamente a **$this->dbutil**

Nota

Ambos parámetros se pueden usar individualmente, simplemente pase un valor vacío como el primero si desea omitirlo

Una vez iniciado, accederá a los métodos utilizando el objeto **$this->dbutil**:

```
$this->dbutil->algun_metodo();
```

Uso de las Utilidades

Obtener los nombres de las Bases de datos

Devuelve un array con los nombres de las Bases de datos:

```
$dbs = $this->dbutil->list_databases();

foreach ($dbs as $db)
{
   echo $db;
}
```

Determinar si existe una Base de datos

A veces es útil saber si existe una Base de datos en particular, devuelve un booleano TRUE/FALSE :

```
if ($this->dbutil->database_exists('nombre_db'))
{
   // algún código ...
}
```

Nota

Reemplace **nombre_db** con el nombre de la Base de datos que está buscando, este método es sensible a mayúsculas y minúsculas

Optimizar una tabla

Le permite optimizar una tabla usando el nombre de tabla especificado en el primer parámetro, devuelve TRUE / FALSE según el éxito o el fracaso:

```
if ($this->dbutil->optimize_table('nombre_db'))
{
   echo 'Tabla optimizada!';
}
```

Nota

No todas las plataformas de Bases de datos admiten la optimización de tablas, es principalmente para usar con MySQL

Reparar una tabla

Le permite reparar una tabla usando el nombre de tabla especificado en el primer parámetro, devuelve TRUE / FALSE según el éxito o el fracaso:

```
if ($this->dbutil->repair_table('nombre_tabla'))
{
   echo 'Tabla reparada !';
}
```

Nota

No todas las plataformas de Bases de datos admiten reparaciones de tablas

Optimizar la Base de datos

Permite optimizar la Base de datos a la que está conectada actualmente su clase de Base de datos, devuelve un array que contiene los mensajes de estado de la BD o FALSE cuando falla:

```
$resultado = $this->dbutil->optimize_database();

if ($resultado! == FALSE)
{
   print_r($resultado);
}
```

Nota

No todas las plataformas de Bases de datos admiten la optimización de la Base de datos, es principalmente para usar con MySQL

Exportar un resultado de consulta como un archivo CSV

Le permite generar un archivo CSV a partir de un resultado de consulta, el primer parámetro del método debe contener el objeto resultado de su consulta:

```
$this->load->dbutil();

$query = $this->db->query("SELECT * FROM mi_tabla");

echo $this->dbutil->csv_from_result($query);
```

Los parámetros segundo, tercero y cuarto le permiten establecer el separador, nueva línea y el carácter de delimitación, respectivamente, por defecto, la coma "," se utiliza como separador, "n" se usa como nueva línea y la comilla doble """ como el delimitador:

```
$separador = ",";
$nueva_linea = "\r\n";
$delimitador = '"';

echo $this->dbutil->csv_from_result($query, $separador, $nueva_linea, $delimitador);
```

Importante

Este método **NO** escribe el archivo CSV, simplemente crea el diseño CSV, si necesita escribir el archivo, use **File Helper**

Exportar un resultado de consulta como un documento XML

Le permite generar un archivo XML a partir de un resultado de consulta, el primer parámetro espera un objeto de resultado de consulta, el segundo puede contener un array opcional de parámetros de configuración:

```
$this->load->dbutil();

$query = $this->db->query("SELECT * FROM mi_tabla");


$config = array(
  'root'    => 'root',
  'element' => 'element',
  'newline' => "\n",
  'tab'     => "\t"
);

echo $this->dbutil->xml_from_result($query, $config);
```

Importante

Este método **NO** escribe el archivo XML por usted, simplemente crea el diseño XML, si necesita escribir el archivo, use **File Helper**

Copia de seguridad de la Base de datos

Notas de la Copia de seguridad

Le permite hacer una copia de seguridad de su Base de datos completa o tablas individuales, los datos de la copia de seguridad se pueden comprimir en formato Zip o Gzip

Nota

Esta función solo está disponible para las Bases de datos MySQL e Interbase/Firebird

Nota

Para Bases de datos Interbase/Firebird, el nombre del archivo de respaldo es el único parámetro:

```
$this->dbutil->backup('nombre_archivo_backup_db');
```

Nota

Debido al tiempo de ejecución limitado y la memoria disponible para PHP, es posible que no se puedan realizar copias de seguridad de Bases de datos muy grandes, si su Base de datos es muy grande, es posible que deba realizar una copia de seguridad directamente desde su servidor SQL a través de la línea de comandos o hacer que el administrador del servidor lo haga por usted si no tiene privilegios de administrador

Ejemplo

```
// Cargar la clase de utilidad DB
$this->load->dbutil();

// Hacer una copia de seguridad de toda su Base de datos y asignarla a una variable
$backup = $this->dbutil->backup();

// Cargue el archivo auxiliar y escriba el archivo en su servidor
$this->load->helper('file');
write_file('/path/a/mi_backup.gz', $backup);

// Cargue el asistente de descarga y envíe el archivo a su escritorio
$this->load->helper('download');
force_download('mi_backup.gz', $backup);
```

Establecer preferencias de copia de seguridad

Las preferencias de copia de seguridad se establecen al enviar un array de valores al primer parámetro del método **backup()** :

```
 $preferencias = array(
  'tables'     => array('tabla1', 'tabla2'),// Array de tablas para el backup.
  'ignore'     => array(), // Lista de tablas para omitir del backup.
  'format'     => 'txt',    // gzip, zip, txt
  'filename'   => 'mi_backup.sql',   // Nombre de archivo - NECESARIO SÓLO CON ARCHIVOS ZIP
  'add_drop'   => TRUE,  // Si agregar DROP TABLE al archivo backup.
  'add_insert' => TRUE,     // Si agregar INSERT al archivo backup.
  'newline'    => "\n"      // Carácter de nueva línea usado en el archivo de backup.
);

$this->dbutil->backup($preferencias);
```

Descripción de preferencias de la Copia de seguridad

Preferencia	Predeterminado	Opciones	Descripción
tables	array vacio	No	Un array de tablas de la que desea hacer una copia de seguridad, si se deja en blanco, todas las tablas se exportarán
ignore	array vacío	No	Un array de tablas que desea que ignore la rutina de copia de seguridad

format	gzip	gzip,zip,txt	El formato de archivo del archivo de exportación
filename	fecha y hora actual	No	El nombre del archivo respaldado, el nombre es necesario solo si está usando compresión zip
add_drop	TRUE	TRUE/FALSE	Si se incluyen las sentencias **DROP TABLE** en su archivo de exportación SQL
add_insert	TRUE	TRUE/FALSE	Si se incluyen las instrucciones **INSERT** en su archivo de exportación SQL
newline	"\n"	"\n", "\r", "\r\n"	Tipo de nueva línea para usar en su archivo de exportación SQL
foreign_key_checks	TRUE	TRUE/FALSE	Si la salida debe mantener activadas las comprobaciones de claves externas

class CI_DB_utility

class CI_DB_utility

class CI_DB_utility

backup()

string backup([array $params = array()])

Descripción

Realiza una copia de seguridad de la Base de datos, según las preferencias del usuario

Parámetros

params

Un array asociativo de opciones

Valores devueltos

String raw/(g) zipped de la consulta SQL

csv_from_result()

string csv_from_result(object $query [, string $delim = ', ' [, string $newline = "n" [, string $enclosure = '"']]])

Descripción

Transforma un objeto de resultado de Base de datos en un documento CSV

Parámetros

query

Un objeto de resultado de Base de datos

delim

Separador de campo

newline

Carácter de nueva línea

enclosure

Delimitador

Valores devueltos

Un string con el archivo CSV generado

database_exists()

```
bool database_exists(string $database_name)
```

Descripción

Verifica la existencia de una Base de datos

Parámetros

database_name

Nombre de la Base de datos

Valores devueltos

TRUE si la Base de datos existe, FALSE si no

list_databases()

```
array list_databases()
```

Descripción

Recupera una lista de todos los nombres de las Bases de datos

Valores devueltos

Array de nombres de las Bases de datos encontradas

optimize_database()

```
array optimize_database()
```

Descripción

Optimiza la Base de datos

Valores devueltos

Array de mensajes de optimización o FALSE en caso de error

optimize_table()

```
array optimize_table(string $table_name)
```

Descripción

Optimiza una tabla de la Base de datos

Parámetros

table_name

Nombre de la tabla para optimizar

Valores devueltos

Array de mensajes de optimización o FALSE en caso de error

repair_table()

```
array repair_table(string $table_name)
```

Descripción

Repara una tabla de Base de datos

Parámetros

table_name
: Nombre de la tabla para reparar

Valores devueltos

Matriz de mensajes de reparación o FALSE en caso de error

xml_from_result()

```
string xml_from_result(object $query [, array $params = array()])
```

Descripción

Transforma un objeto de resultado de Base de datos en un documento XML

Parámetros

query
: Un objeto de resultado de Base de datos

params
: Un array asociativo de preferencias

Valores devueltos

Un string con el documento XML generado

Referencia del controlador DB

Esta es la clase de implementación base DB independiente de la plataforma, esta clase no se llamará directamente, más bien, la clase de adaptador para la Base de datos específica se extenderá y creará una instancia

La manera de cómo hacerlo se ha dividido en varios artículos, este artículo está destinado a ser una referencia para ellos

Importante
No todos los métodos son compatibles con todos los controladores de Base de datos, algunos de ellos pueden fallar, y devolver FALSE, si el controlador subyacente no los admite

class CI_DB_driver

class CI_DB_driver

class CI_DB_driver

affected_rows()

int affected_rows()

Descripción

Devuelve el número de filas modificadas por la última consulta ejecutada, útil para verificar cuántas filas se crearon, actualizaron o eliminaron

Valores devueltos

Número de filas afectadas

cache_delete()

bool cache_delete([string $segment_one = '' [, string $segment_two = '']])

Descripción

Elimine los archivos de caché asociados con un URI particular

Parámetros

segment_one
: Primer segmento de URI

segment_two
: Segundo segmento de URI

Valores devueltos

TRUE en el éxito, FALSE si falla

cache_delete_all()

bool cache_delete_all()

Descripción

Eliminar todos los archivos de caché

Valores devueltos

TRUE en caso de éxito, FALSE en caso de error

cache_off()

bool cache_off()

Descripción

Deshabilita el almacenamiento en memoria caché de resultados

Valores devueltos

TRUE si el almacenamiento en caché está activado, FALSE si no

cache_on()

bool cache_on()

Descripción

Habilita el almacenamiento en memoria caché de resultados

Valores devueltos

TRUE si el almacenamiento en caché está activado, FALSE si no

cache_set_path()

void cache_set_path([string $path = ''])

Descripción

Establece la ruta del directorio que se utilizará para el almacenamiento en caché

Parámetros

path

Ruta al directorio de caché

valores devueltos

void

call_function()

string call_function(string $function)

Descripción

Ejecuta una función PHP nativa, utilizando un contenedor independiente de la plataforma

Parámetros

function

Nombre de la función

Valores devueltos

El resultado de la función

close()

```
void close()
```

Descripción

Cierra la conexión a la Base de datos

valores devueltos

void

compile_binds()

```
string compile_binds(string $sql, array $binds)
```

Descripción

Compila una consulta SQL con los valores de enlace pasados

Parámetros

sql

La instrucción SQL

binds

Un array de datos vinculantes

Valores devueltos

La declaración de SQL actualizada

count_all()

```
int count_all([ string $table = ''])
```

Descripción

Devuelve el número total de filas en una tabla, o 0 si no se proporcionó ninguna tabla

Parámetros

table

Nombre de la tabla

Valores devueltos

Recuento de filas para la tabla especificada

db_connect()

mixed db_connect(bool $persistent = TRUE)

Descripción

Establece una conexión con la Base de datos

Parámetros

persistent

Si se debe establecer una conexión persistente o una conexión regular

Valores devueltos

Base de datos recurso / objeto o FALSE en caso de error

> ***Nota***
>
> El valor devuelto depende del controlador subyacente en uso, por ejemplo, se devolverá una instancia de mysqli con el controlador 'mysqli'

db_pconnect()

mixed db_pconnect()

Descripción

Establece una conexión persistente con la Base de datos

Valores devueltos

Devuelve: recurso / objeto de conexión de Base de datos o FALSE en caso de error

> ***Nota***
>
> Este método es solo un alias para **db_connect(TRUE)**

db_select()

bool db_select([string $database = ''])

Descripción

Selecciona / cambia la Base de datos actual

Parámetros

database

Nombre de la Base de datos

Valores devueltos

TRUE en el éxito, FALSE si falla

db_set_charset()

bool db_set_charset(string $charset)

Descripción

Establecer conjunto de caracteres del cliente

Parámetros

charset

Nombre del conjunto de caracteres

Valores devueltos

TRUE en el éxito, FALSE si falla

display_error()

string display_error([string $error = '' [, string $swap = '' [, bool $native = FALSE]]])

Descripción

Muestra un mensaje de error y detiene la ejecución del script

Parámetros

error

El mensaje de error

swap

Cualquier valor de "intercambio"

native

Si se localiza el mensaje

Valores devueltos

Muestra la pantalla de error de DB, el mensaje se muestra utilizando la plantilla **application/views/errors/error_db.php**

elapsed_time()

string elapsed_time([int $decimals = 6])

Descripción

Calcula el tiempo transcurrido de la consulta agregada

Parámetros

decimales

La cantidad de decimales

Valores devueltos

El tiempo transcurrido de la consulta agregada, en microsegundos

escape()

mixed escape(mixed $str)

Descripción

Escapa los datos de entrada en función del tipo, incluidos booleanos y NULL

Parámetros

str
: El valor para escapar, o un array múltiple

Valores devueltos

Los valores escapados

escape_identifiers()

mixed escape_identifiers(mixed $item)

Descripción

Escape de identificadores de SQL, como columna, tabla y nombres

Parámetros

item
: El elemento o conjunto de elementos para escapar

Valores devueltos

El (los) elemento(s) de entrada escapados

escape_like_str()

mixed escape_like_str(mixed $str)

Descripción

Escape LIKE strings

Parámetros

str
: Un valor de string o array múltiple

Valores devueltos

Lo(s) string(s) escapado(s)

Similar a **escape_str()**, pero también escapará a los caracteres comodín % y _, para que no causen falsos positivos en condiciones **LIKE**

Importante

El método **escape_like_str()** usa '!' (Signo de exclamación) para escapar los caracteres especiales para las condiciones **LIKE**, debido a que este método escapa strings parciales que envolvería entre comillas usted mismo, no puede agregar automáticamente la condición **ESCAPE** '!' por usted, por lo que tendrá que hacerlo manualmente

escape_str()

mixed escape_str(mixed $str [, bool $like = FALSE])

Descripción

Escapa de los valores de string

Parámetros

str

Un valor de string o array múltiple

like

Si la string se usará o no en una condición LIKE

Valores devueltos

Lo(s) string(s) escapado(s)

Advertencia

Los strings devueltos **NO incluyen comillas** a su alrededor

field_data()

array field_data(string $table)

Descripción

Obtiene una lista que contiene datos de campo sobre una tabla

Parámetros

table

El nombre de la tabla

Valores devueltos

Matriz de elementos de datos de campo o FALSE en caso de error

field_exists()

bool field_exists(string $field_name, string $table_name)

Descripción

Determina si existe un campo en particular

Parámetros

table_name

El nombre de la tabla

field_name

El nombre del campo

Valores devueltos

TRUE si ese campo existe en esa tabla, FALSE si no

initialize()

bool initialize()

Descripción

Inicializa la configuración de la Base de datos, establece una conexión con la Base de datos

Valores devueltos

TRUE en caso de éxito, FALSE en caso de error

insert_string()

string insert_string(string $table, array $data)

Descripción

Genera un string de instrucción **INSERT**

Parámetros

table

La tabla objetivo

data

Un array asociativo de pares clave / valor

Valores devueltos

La instrucción SQL INSERT, como un string

is_write_type()

bool is_write_type(string $sql)

Descripción

Determina si una consulta es de tipo "**escritura**" (como **INSERT, UPDATE, DELETE**) o "**lectura**" (es decir, **SELECT**)

Parámetros

sql

La instrucción SQL

Valores devueltos

TRUE si la declaración SQL es de "**tipo de escritura**", FALSE si no

last_query()

```
string last_query()
```

Descripción

Devuelve la última consulta que se ejecutó

Valores devueltos

La última consulta ejecutada

list_fields()

```
array list_fields(string $table)
```

Descripción

Obtiene una lista de los nombres de campo en una tabla

Parámetros

table
: El nombre de la tabla

Valores devueltos

Array de nombres de campo o FALSE en caso de error

list_tables()

```
array list_tables([ bool $constrain_by_prefix = FALSE])
```

Descripción

Obtiene una lista de las tablas en la Base de datos actual

Parámetros

constrain_by_prefix
: TRUE para unir los nombres de las tablas con el **dbprefix** configurado

Valores devueltos

Array de nombres de tabla o FALSE en caso de error

platform()

```
string platform()
```

Descripción

El nombre de la plataforma en uso (mysql, mssql, etc ...)

Valores devueltos

Nombre de la plataforma

primary()

string primary(string $table)

Descripción

Recupera la clave principal de una tabla

Parámetros

table
Nombre de la tabla

Valores devueltos

El nombre de la clave principal, FALSE si no hay

> ***Nota***
>
> Si la plataforma de la Base de datos no es compatible con la detección de clave primaria, se puede suponer que el nombre de la primera columna es la clave principal

protect_identifiers()

string protect_identifiers(string $item [, bool $prefix_single = FALSE [, bool protect_identifiers = NULL [, bool $field_exists = TRUE]]])

Descripción

Toma un nombre de columna o tabla, opcionalmente con un alias, y aplica el **dbprefix** configurado

Parámetros

item
El ítem para trabajar con
prefix_single
Si se aplica el **dbprefix** incluso si el elemento de entrada es un identificador único
protect_identifiers
Si se deben citar identificadores
field_exists
Si el elemento suministrado contiene un nombre de campo o no

Valores devueltos

El item modificado

Es necesaria cierta lógica para tratar los nombres de las columnas que incluyen la ruta

Considere una consulta como esta:

```
SELECT * FROM hostname.database.table.column AS c FROM hostname.database.table
```

O una consulta con aliasing:

```
SELECT m.member_id, m.member_name FROM members AS m
```

Dado que el nombre de la columna puede incluir hasta cuatro segmentos, host, DB, table, column, o también tiene un prefijo de alias, necesitamos hacer un poco de trabajo para resolverlo e insertar el prefijo de la tabla, si existe, en el posición correcta, y escape solo los identificadores correctos

Este método es utilizado ampliamente por la clase **Query Builder**

query()

mixed query(string $sql [, array $binds = FALSE [, bool $return_object = NULL]])

Descripción

Ejecuta una consulta SQL

Parámetros

sql

La instrucción SQL para ejecutar

binds

Un array de datos vinculantes

return_object

Si se devuelve un objeto resultado o no

Valores devueltos

TRUE luego de la ejecución exitosa de una consulta de "**tipo escritura**", FALSE al fallar, un objeto CI_DB_result (method chaining) para consultas de "**tipo lectura**", acepta un string SQL como entrada y devuelve un objeto resultante al ejecutar con éxito una consulta de tipo "**lectura**"

reconnect()

bool reconnect()

Descripción

Mantiene / restablece la conexión a la Base de datos si no se han enviado consultas durante un período de tiempo que exceda el tiempo de espera inactivo del servidor

Valores devueltos

TRUE en caso de éxito, FALSE en caso de error

simple_query()

mixed simple_query(string $sql)

Descripción

Una versión simplificada del método **query()**, apropiado para usar cuando no necesita obtener un objeto de resultado o simplemente enviar una consulta a la Base de datos y no preocuparse por el resultado

Parámetros

sql

La instrucción SQL para ejecutar

Valores devueltos

Cualquiera que sea la función de "**consulta**" del controlador subyacente

table_exists()

```
bool table_exists(string $table_name)
```

Descripción

Determina si existe una tabla en particular

Parámetros

table_name
: El nombre de la tabla

Valores devueltos

TRUE si esa tabla existe, FALSE si no

total_queries()

```
int total_queries()
```

Descripción

Devuelve el número total de consultas que se han ejecutado hasta el momento

Valores devueltos

El número total de consultas ejecutadas

trans_complete()

```
bool trans_complete()
```

Descripción

Completa la transacción

Valores devueltos

TRUE en caso de éxito, FALSE en caso de error

trans_off()

```
void trans_off()
```

Descripción

Desactiva las transacciones en tiempo de ejecución

valores devueltos

void

trans_start()

```
bool trans_start([ bool $test_mode = FALSE])
```

Descripción

Comienza una transacción

Parámetros

test_mode

Indicador de modo de prueba

Valores devueltos

TRUE en el éxito, FALSE si falla

trans_status()

```
bool trans_status()
```

Descripción

Le permite recuperar el indicador de estado de la transacción para determinar si ha fallado

Valores devueltos

TRUE si la transacción fue exitosa, FALSE si falló

trans_strict()

```
void trans_strict([ bool $mode = TRUE])
```

Descripción

Activar / desactivar el modo "**estricto**" de transacción, cuando el modo estricto está habilitado, si está ejecutando varios grupos de transacciones y un grupo falla, todos los grupos subsiguientes se revertirán, si el modo estricto está des habilitado, cada grupo se trata de forma autónoma, lo que significa que un fallo de un grupo no afectará a ninguna otra

Parámetros

mode

Indicador de modo estricto

valores devueltos

void

update_string()

```
string update_string(string $table, array $data, mixed $where)
```

Descripción

Genera un string de instrucción **UPDATE**

Parámetros

table

La tabla objetivo

data

Una array asociativo de pares clave / valor

where

Las condiciones de la declaración **WHERE**

Valores devueltos

La instrucción SQL **UPDATE**, como un string

version()

string version()

Descripción

Número de versión de la Base de datos

Valores devueltos

La versión de la Base de datos que se está utilizando

Helpers - Ayudantes

Array Helper

Array Helper ayuda a trabajar con arrays

Cargar Array Helper

Array Helper se carga así:

```
$this->load->helper('array');
```

Funciones

element()

mixed element(string $item, array $array [, bool $default = NULL])

Descripción

Le permite buscar un elemento de un array, la función prueba si el índice del array está establecido y si tiene el valor

Parámetros

item
Elemento a recuperar del array

array
Array de entrada

default
Qué devolver si el array no es válido

Valores devueltos

Si existe el valor, se devuelve, si no existe devuelve NULL o lo que se haya especificado como predeterminado en el tercer parámetro

Ejemplo

```
$array = array(
  'color' => 'rojo',
  'forma' => 'octogono',
  'peso'  => ''
);

echo element('color', $array); // presenta: "rojo"
echo element('peso', $array, 'nolose'); // presenta: "nolose"
```

elements()

mixed elements(string $items, array $array [, bool $default = NULL])

Descripción

Le permite obtener una cantidad de elementos de un array, la función prueba si cada uno de los índices del array está establecido

Parámetros

item

Elemento a recuperar del array

array

Array de entrada

default

Qué devolver si el array no es válido

Valores devueltos

Si existe el valor, se devuelve, si no existe devuelve NULL o lo que se haya especificado como predeterminado en el tercer parámetro

Ejemplo

```
$array = array(
  'forma'    => 'octogono',
  'color'    => 'rojo',
  'radio'    => '10',
  'diámetro' => '20'
);

$mis_figuras = elements(array('forma','color','altura'), $array);
```

Devolverá el siguiente array:

```
Array(
  'forma'  => 'octogono',
  'color'  => 'rojo',
  'altura' => NULL
);
```

Puede establecer el tercer parámetro a cualquier valor predeterminado que desee :

```
$mis_figuras = elements(array ('forma','color','altura'), $array, 'nolose');
```

Devolverá:

```
Array(
  'forma'  => 'octogono',
  'color'  => 'rojo',
  'altura' => 'nolose'
);
```

Es útil al enviar el array **$_POST** a uno de sus Modelos, evita que los usuarios envíen datos **POST** adicionales para ser ingresados en sus tablas :

```
$this->load->model('post_model');

$this->post_model->update(
  elements(array('id', 'título', 'contenido'), $_POST)
);
```

Esto asegura que solo los campos **id**, **título** y **contenido** se envíen para actualizarse

random_element()

mixed random_element(array $array)

Descripción

Toma un array como entrada y devuelve un elemento aleatorio de el

Parámetros

array

Array de entrada

Valores devueltos

Un elemento aleatorio del array

Ejemplo

```
$citas = array (
  "Una vez terminado el juego, el rey y el peón vuelven a la misma caja",
  "Cuando los ricos se hacen la guerra, son los pobres los que mueren",
  "El mundo de los olores es el mundo de los recuerdos",
  "Dejar hacer pudiendo evitar, es aprobar",
  "Actuamos en este mundo como si tuviéramos uno de repuesto",
  "La vejez es la perdida de la curiosidad"
);

echo random_element($citas);
```

CAPTCHA Helper

CAPTCHA Helper ayuda a crear imágenes CAPTCHA

Cargar CAPTCHA Helper

CAPTCHA Helper se carga así:

```
$this->load->helper('captcha');
```

Usar CAPTCHA Helper

Una vez cargado, puede generar un CAPTCHA como este:

```
$vals = array(
  'word'          => 'Palabra aleatoria',
  'img_path'      => './captcha/',
  'img_url'       => 'http://ejemplo.com/captcha/',
  'font_path'     => './path/to/fuentes/texb.ttf',
  'img_width'     => '150',
  'img_height'    => 30,
  'expiration'    => 7200,
  'word_length'   => 8,
  'font_size'     => 16,
  'img_id'        => 'Imageid',
  'pool'          => '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ',

  // Fondo y borde blanco, texto negro y marco rojo
  'colors' => array(
     'background' => array(255, 255, 255),
     'border'     => array(255, 255, 255),
     'text'       => array(0, 0, 0),
     'grid'       => array(255, 40, 40)
     )
   );

$cap = create_captcha($vals);
echo $cap['image'];
```

- La función de captura requiere la biblioteca de imágenes GD
- Solo se requieren **img_path** e **img_url**
- Si no se proporciona una palabra, la función generará un string ASCII aleatorio. Puede crear su propia biblioteca de palabras a partir de la cual pueda dibujar al azar
- Si no especifica una ruta a una fuente TRUE TYPE, se usará la fuente nativa de GD

- En el directorio "**captcha**" debe poder escribirse
- La caducidad, en segundos, indica cuánto tiempo permanecerá una imagen en el directorio captcha antes de que se elimine, el valor predeterminado es dos horas
- La longitud de la palabra por defecto es 8, la agrupación predeterminada es : '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
- **font_size** tiene como valor predeterminado 16, la fuente GD nativa tiene un límite de tamaño, especifique una fuente "**true type**" para tamaños más grandes
- **img_id** se establecerá como el "**id**" de la imagen del captcha
- Si falta alguno de los valores de color, será reemplazado por el predeterminado

Agregar una Base de datos

Para que la función de captcha evite que alguien la envíe, deberá agregar la información devuelta desde **create_captcha()** a su Base de datos, luego, cuando el usuario envíe los datos del formulario, deberá verificar que los datos estén en la Base de datos y que no hayan expirado

Aquí tiene un prototipo de tabla:

```
CREATE TABLE captcha (
  captcha_id bigint(13) unsigned NOT NULL auto_increment,
  captcha_time int(10) unsigned NOT NULL,
  ip_address varchar(45) NOT NULL,
  word varchar(20) NOT NULL,
  PRIMARY KEY `captcha_id` (`captcha_id`),
  KEY `word` (`word`)
);
```

Aquí tiene un ejemplo de uso con una Base de datos, en la página donde se mostrará el **CAPTCHA**, tendrá algo como esto:

```
$this->load->helper('captcha');
$vals = array(
  'img_path' => './captcha/',
  'img_url'  => 'http://ejemplo.com/captcha/'
);

$cap = create_captcha($vals);
$datos = array(
  'captcha_time' => $cap['time'],
  'ip_address'   => $this->input->ip_address(),
  'word'         => $cap['word']
);

$query = $this->db->insert_string('captcha', $datos);
```

```
$this->db->query($query);

echo 'Escriba la siguiente palabra:';
echo $cap['image'];
echo '<input type="text" name="captcha" value="" />';
```

Luego, en la página que acepta el envío, tendrá algo como esto:

```
/ Primero, borrar los captchas anteriores
$expiration = time() - 7200; // Límite de dos horas
$this->db->where('captcha_time < ', $expiration) ->delete('captcha');

// Entonces ve si existe un captcha:
$sql = 'SELECT COUNT(*) AS count FROM captcha WHERE word = ? AND ip_address = ?
        AND captcha_time > ?';

$binds = array($_POST['captcha'], $this->input->ip_address(), $expiration);
$query = $this->db->query($sql, $binds);
$row = $query->row();

if ($row->count == 0)
{
   echo 'Debe enviar la palabra que aparece en la imagen.';
}
```

Funciones

create_captcha()

array create_captcha([array $data = '' [, string $img_path = '' [, string $img_url = '' [, string $font_path = '']]]])

Descripción

Toma una serie de información para generar un **CAPTCHA** como entrada y crea la imagen según sus especificaciones, devolviendo un array de datos asociativos sobre la imagen

Parámetros

data
: Array de datos para el **CAPTCHA**

img_path
: Ruta para crear la imagen (**DESACONSEJADO,** proporciónelo en el array **$data)**

img_url
: URL a la carpeta de imágenes **CAPTCHA** (**DESACONSEJADO,** proporciónelo en el array **$data)**

font_path
: Ruta del servidor a la fuente (**DESACONSEJADO,** proporciónelo en el array **$data)**

Valores devueltos

array('word' => $word, 'time' => $now, 'image' => $img)

```
array(
  'image' => IMAGE TAG
  'time'  => TIMESTAMP (in microtime)
  'word'  => CAPTCHA WORD
);
```

image: es la de la etiqueta:

```
<img src="http://ejemplo.com/captcha/12345.jpg" width="140" height="50" />
```

time : es la micro timestamp usada como el nombre de la imagen sin la extensión de archivo, será un número como este: 1139612155.3422

word: Es la palabra que aparece en la imagen del captcha, que si no se proporciona a la función, será un string aleatorio

Nota

El uso de los parámetros **$img_path**, **$img_url** y **$font_path** está **DESACONSEJADO**, proporciónelos en el conjunto **$data** en su lugar

Cookie Helper

Cookie Helper contiene funciones que ayudan a trabajar con cookies

Cargar Cookie Helper

Cookie Helper se carga así:

```
$this->load->helper('cookie');
```

Funciones

delete_cookie()

void delete_cookie(string $name [, string $domain = '' [, string $path = '/' [, string $prefix = '']]])

Descripción

Permite eliminar una cookie, a menos que haya establecido una ruta personalizada u otros valores, solo se necesita el nombre de la cookie

Parámetros

name
: Nombre de la cookie

domain
: Dominio de cookies (generalmente: .sudominio.com)

path
: Ruta de las cookies

prefix
: Prefijo de nombre de cookie

valores devueltos

void

Ejemplo

```
delete_cookie('nombre');
```

Esta función es idéntica a **set_cookie()**, excepto que no tiene el valor y los parámetros de caducidad, puede enviar un array de valores en el primer parámetro o puede establecer parámetros discretos:

```
delete_cookie($nombre, $dominio, $ruta, $prefijo);
```

get_cookie()

mixed get_cookie(string $index [, bool $xss_clean = NULL])

Descripción

Esta función le proporciona una sintaxis más amigable para obtener cookies del navegador, consulte la **Input Library** para una descripción detallada de su uso, ya que esta función actúa de similar a **CI_Input::cookie()**, excepto que antepone **config['cookie_prefix']** si se encuentra configurado en **application/config/config.php**

Parámetros

index
: Nombre de la cookie

xss_clean
: Si se debe aplicar el filtrado XSS al valor devuelto

Valores devueltos

El valor de la cookie o NULL si no se encuentra

set_cookie()

void set_cookie(mixed $name [, string $value = '' [, int $expire = '' [, string $domain = '' [, string $path = '/' [, string $prefix = '' [, bool $secure = NULL [, bool $httponly = NULL]]]]]]])

Descripción

Esta función auxiliar le brinda una sintaxis más amigable para configurar las cookies del navegador, consulte la **Input Library** para obtener una descripción de su uso, ya que esta función es un alias de **CI_Input::set_cookie()**

Parámetros

name
: Nombre de la cookie o array asociativo de todos los parámetros disponibles para esta función

value
: Valor de la cookie

expire
: Número de segundos hasta la caducidad

domain
: Dominio de cookies (generalmente: .sudominio.com)

path
: Ruta de las cookies

prefix
: Prefijo de nombre de cookie

secure
: Si solo enviar la cookie a través de HTTPS

httponly
: Si ocultar la cookie de JavaScript

valores devueltos

void

Date Helper

Date Helper le ayuda a trabajar con las fechas

Cargar Date Helper

Date Helper se carga así:

```
$this->load->helper('date');
```

Funciones

date_range()

array date_range([int $unix_start = '' [, int $mixed = '' [, bool $is_unix = TRUE [, string $format = 'Y-m-d']]]])

Descripción

Devuelve una lista de fechas dentro de un período especificado

Parámetros

unix_start
Marca de tiempo UNIX de la fecha de inicio del rango

mixed
Marca de tiempo UNIX de la fecha de finalización del rango o intervalo en días

is_unix
Establecido en FALSE si **$mixed** no es una marca de tiempo

format
Formato de fecha de salida, igual que en **date()**

Valores devueltos

Una variedad de fechas

Ejemplo

```
$rango = date_range('2012-01-01', '2012-01-15');
echo "Primeros 15 días de 2012:";
foreach ($rango as $fecha)
{
   echo $fecha."\n";
}
```

days_in_month()

int days_in_month([int $month = 0[, int $year = '']])

Descripción

Devuelve la cantidad de días en un mes / año determinado, tiene en cuenta si es año bisiesto

Parámetros

month

Un mes numérico

year

Un año numérico, si está vacío, se usará el año actual

Valores devueltos

Recuento de días en el mes especificado

Ejemplo

```
echo days_in_month(06, 2018);
```

> ***Nota***
>
> Esta función es un alias de la función nativa **cal_days_in_month()**, si está disponible

gmt_to_local()

int gmt_to_local([int $time = '' [, string $timezone = 'UTC' [, bool $dst = FALSE]]])

Descripción

Toma como entrada una marca de tiempo UNIX (referenciada a GMT), y la convierte a una marca de tiempo localizada según la zona horaria y el horario de verano enviados

Parámetros

time

Indicación de fecha y hora UNIX

timezone

Zona horaria

dst

Si el DST está activo

Valores devueltos

Marca de tiempo UNIX

Ejemplo

```
$timestamp = 1140153693;
$timezone  = 'UM8';
$daylight_saving = TRUE;
echo gmt_to_local($timestamp, $timezone, $daylight_saving);
```

Nota

Para obtener una lista de zonas horarias, consulte la referencia en la parte inferior de esta página

human_to_unix()

int human_to_unix([int $datestr = ''])

Descripción

Lo opuesto a la función **unix_to_time()**, toma un tiempo "**humano**" como entrada y lo devuelve como Unix, esta función es útil si acepta fechas formateadas como "**humanas**" enviadas desde un formulario. Devuelve FALSE (booleano) si la cadena de la fecha pasada no está formateada como se indicó anteriormente

Parámetros

datestr

Cadena de fecha

Valores devueltos

Marca de tiempo UNIX o FALSE en caso de error

Ejemplo

```
$ahora = time();
$humano = unix_to_human($ahora);
$unix = human_to_unix($humano);
```

local_to_gmt()

int local_to_gmt([time $time = ''])

Descripción

Toma una marca de tiempo UNIX como entrada y la devuelve como GMT

Parámetros

time

Indicación de fecha y hora UNIX

Valores devueltos

Marca de tiempo UNIX

Ejemplo

```
$gmt = local_to_gmt(time());
```

mdate()

string mdate([string $datestr = '' [, int $time = '']])

Descripción

Esta función es idéntica **date()** de PHP, excepto que le permite usar códigos de fecha de al estilo de MySQL, donde cada letra de código va precedida de un signo de porcentaje, por ejemplo: %Y %m %d etc.

Parámetros

datestr

Cadena de fecha

time

Indicación de fecha y hora UNIX

Valores devueltos

Fecha con formato de MySQL

El beneficio de hacer las fechas de esta forma es que no hay que preocuparse de escapar los caracteres que no son códigos de fecha, como tendría que hacerlo normalmente con la función **date()**

Ejemplo

```
$fecha_string = 'Año: %Y Mes: %m Día: %d - %h:%i %a';
$time = time();
echo mdate($fecha_string, $time);
```

Si no se incluye una marca de tiempo en el segundo parámetro, se usará la fecha/hora actual

mysql_to_unix()

int mysql_to_unix([string $time = ''])

Descripción

Toma una marca de tiempo MySQL como entrada y la devuelve como una marca de tiempo UNIX

Parámetros

time

Marca de tiempo MySQL

Valores devueltos

Marca de tiempo UNIX

Ejemplo

```
$unix = mysql_to_unix('20061124092345');
```

nice_date()

Nota

Esta función está **DESACONSEJADA**, use la clase nativa de PHP **DateTime** en su lugar

now()

int now([string $timezone = NULL])

Descripción

Devuelve la hora actual como una marca de tiempo UNIX, referenciada a la hora local de su servidor o cualquier zona horaria compatible con PHP, según el valor "**time reference**" del archivo de configuración, si no tiene la intención de establecer su referencia de hora maestra a cualquier otra zona horaria compatible con PHP (que normalmente hará si ejecuta un sitio que permite a cada usuario establecer sus propias configuraciones de zona horaria), no hay beneficio de usar esta función respecto de la función **time()** de PHP

Parámetros

timezone

Zona horaria, si no se proporciona una zona horaria, devolverá **time()** en función de la configuración de referencia de hora

Valores devueltos

Marca de tiempo UNIX

```
echo now('Australia/Victoria');
```

standard_date()

Nota

Función **DESACONSEJADA**. Use la función nativa date() combinada con las constantes de formato de DateTime:

```
echo date(DATE_RFC822, time());
```

timespan()

string timespan([int $seconds = 1 [, string $time = '' [, int $units = '']]])

Descripción

Formatea una marca de tiempo UNIX para que parezca similar a esto:

```
1 año, 10 meses, 2 semanas, 5 días, 10 horas, 16 minutos
```

El propósito más común para esta función es mostrar cuánto tiempo ha transcurrido desde algún punto en el tiempo en el pasado hasta ahora

Parámetros

seconds

Número de segundos, tiene que contener una marca de tiempo UNIX

time

Marca de tiempo UNIX, tiene que contener una marca de tiempo que sea mayor que la primera, si está vacío, se usará la hora actual

units

Número de unidades de tiempo para mostrar, parámetro opcional, limita el número de unidades de tiempo para mostrar

Valores devueltos

Diferencia horaria formateada

Ejemplo

```
$fecha_post = '1079621429';
$ahora = time();
$unidades = 2;
echo timespan($fecha_post, $ahora, $unidades);
```

> ***Nota***
>
> El texto generado por esta función se encuentra en el siguiente archivo de idioma:
> **language/<su_idioma>/date_lang.php**

timezones()

int timezones([string $tz = ''])

Descripción

Toma una referencia de zona horaria (para obtener una lista de zonas horarias válidas, consulte la "**Referencia de zona horaria**" a continuación) y devuelve el número de horas de compensación UTC

Parámetros

tz

Una zona horaria numérica

Valores devueltos

Diferencia horaria de UTC

Ejemplo

```
echo timezones('UM5');
```

Esta función es útil cuando se utiliza con **timezone_menu()**

timezone_menu()

string timezone_menu([string $default = 'UTC' [, string $class = '' [, string $name = 'timezones' [, mixed $attributes = '']]]])

Descripción

Genera un menú desplegable de zonas horarias, este menú es útil si ejecuta un sitio en el que los usuarios pueden establecer su valor de zona horaria local, consulte la **referencia de la zona horaria** para ver los valores de este menú

Parámetros

default

Zona horaria , le permite establecer el estado "**selected**" del menú. Por ejemplo, para configurar la hora del Pacífico como predeterminada, hará esto:

```
echo timezone_menu('UM8');
```

class

Nombre de la clase CSS para el menú

name

Nombre del menú

attributes

Establecer uno o más atributos HTML en la etiqueta de selección generada

Valores devueltos

Menú desplegable HTML con husos horarios

unix_to_human()

string unix_to_human([int $time = '' [, bool $seconds = FALSE [, string $fmt = 'us']]])

Descripción

Toma una marca de tiempo UNIX como entrada y la devuelve en un formato humanamente legible con este prototipo:

```
YYYY-MM-DD HH:MM:SS AM/PM
```

Puede ser útil si necesita mostrar una fecha en un campo de formulario para enviar

Parámetros

time

Indicación de fecha y hora UNIX

seconds

Si se muestran los segundos

fmt

Formato (us o euro)

Valores devueltos

Fecha formateada

Ejemplo

La hora se puede formatear con o sin segundos, y se puede configurar en formato europeo o estadounidense, si solo se envía la marca de tiempo, devolverá el tiempo sin segundos formateado para los EE. UU:

```
$now = time();
echo unix_to_human($now); // hora de EE. UU. sin segundos
echo unix_to_human($now, TRUE, 'us'); // Tiempo de EE. UU. con segundos
echo unix_to_human($now, TRUE, 'eu'); // Tiempo Europeo con segundos
```

Referencia de zona horaria

La siguiente tabla indica cada zona horaria y su ubicación

Tenga en cuenta que algunas de las listas de ubicaciones se han resumido para mayor claridad y formato

TimeZone	Ubicación
UM12	(UTC - 12:00) Baker/Howland Island
UM11	(UTC - 11:00) Samoa Time Zone, Niue
UM10	(UTC - 10:00) Hawaii-Aleutian Standard Time, Cook Islands
UM95	(UTC - 09:30) Marquesas Islands
UM9	(UTC - 09:00) Alaska Standard Time, Gambier Islands
UM8	(UTC - 08:00) Pacific Standard Time, Clipperton Island
UM7	(UTC - 07:00) Mountain Standard Time
UM6	(UTC - 06:00) Central Standard Time
UM5	(UTC - 05:00) Eastern Standard Time, Western Caribbean
UM45	(UTC - 04:30) Venezuelan Standard Time
UM4	(UTC - 04:00) Atlantic Standard Time, Eastern Caribbean
UM35	(UTC - 03:30) Newfoundland Standard Time
UM3	(UTC - 03:00) Argentina, Brazil, French Guiana, Uruguay
UM2	(UTC - 02:00) South Georgia/South Sandwich Islands
UM1	(UTC -1:00) Azores, Cape Verde Islands
UTC	(UTC) Greenwich Mean Time, Western European Time
UP1	(UTC +1:00) Central European Time, West Africa Time
UP2	(UTC +2:00) Central Africa Time, Eastern European Time
UP3	(UTC +3:00) Moscow Time, East Africa Time
UP35	(UTC +3:30) Iran Standard Time
UP4	(UTC +4:00) Azerbaijan Standard Time, Samara Time
UP45	(UTC +4:30) Afghanistan
UP5	(UTC +5:00) Pakistan Standard Time, Yekaterinburg Time
UP55	(UTC +5:30) Indian Standard Time, Sri Lanka Time

UP575	(UTC +5:45) Nepal Time
UP6	(UTC +6:00) Bangladesh Standard Time, Bhutan Time, Omsk Time
UP65	(UTC +6:30) Cocos Islands, Myanmar
UP7	(UTC +7:00) Krasnoyarsk Time, Cambodia, Laos, Thailand, Vietnam
UP8	(UTC +8:00) Australian Western Standard Time, Beijing Time
UP875	(UTC +8:45) Australian Central Western Standard Time
UP9	(UTC +9:00) Japan Standard Time, Korea Standard Time, Yakutsk
UP95	(UTC +9:30) Australian Central Standard Time
UP10	(UTC +10:00) Australian Eastern Standard Time, Vladivostok Time
UP105	(UTC +10:30) Lord Howe Island
UP11	(UTC +11:00) Srednekolymsk Time, Solomon Islands, Vanuatu
UP115	(UTC +11:30) Norfolk Island
UP12	(UTC +12:00) Fiji, Gilbert Islands, Kamchatka, New Zealand
UP1275	(UTC +12:45) Chatham Islands Standard Time
UP13	(UTC +13:00) Phoenix Islands Time, Tonga
UP14	(UTC +14:00) Line Islands

Directory Helper

Directory Helper ayuda a trabajar con directorios

Cargar Directory Helper

Directory Helper se carga así:

```
$this->load->helper('directory');
```

Funciones

directory_map()

array directory_map(string $source_dir [, int $directory_depth = 0 [, bool $hidden = FALSE]])

Descripción

Lee la ruta del directorio especificado y crea un array con él y todo su contenido

Parámetros

source_dir
: Ruta al directorio fuente

directory_depth
: Profundidad de directorios para recorrer (0 = totalmente recursivo, 1 = directorio actual, etc.)

hidden
: Si se incluyen directorios ocultos

Valores devueltos

Un array de archivos

Ejemplo

```
$map = directory_map('./mydirectory/');
```

Nota

Las rutas son casi siempre relativas a su archivo **index.php** principal

Los subdirectorios contenidos dentro del directorio también se incluyen, si desea controlar la profundidad de recursión, puede hacerlo utilizando el segundo parámetro, una profundidad de 1 solo recorre el directorio de nivel superior :

```
$map = directory_map('./mydirectory/', 1);
```

Por defecto, los archivos ocultos no se incluyen en el array devuelto, para anular este comportamiento, puede establecer el tercer parámetro como TRUE:

```
$map = directory_map('./mydirectory/', FALSE, TRUE);
```

Cada nombre de directorio será un índice de array, mientras que sus archivos contenidos serán indexados numéricamente

Aquí hay un ejemplo de un array típico:

```
Array (
 [libraries] => Array
 (
    [0] => benchmark.html
    [1] => config.html
    ["database/"] => Array
       (
          [0] => query_builder.html
          [1] => binds.html
          [2] => configuration.html
          [3] => connecting.html
          [4] => examples.html
          [5] => fields.html
          [6] => index.html
          [7] => queries.html
       )
    [2] => email.html
    [3] => file_uploading.html
    [4] => image_lib.html
    [5] => input.html
    [6] => language.html
    [7] => loader.html
    [8] => pagination.html
    [9] => uri.html
 )
)
```

Download Helper

Download Helper le permite descargar datos a su escritorio

Cargar Download Helper

Download Helper se carga así:

```
$this->load->helper('download');
```

Funciones

force_download()

void force_download([string $filename = '' [, mixed $data = '' [, bool $set_mime = FALSE]]])

Descripción

Genera encabezados de servidor que fuerzan la descarga de datos a su escritorio, útil con descargas de archivos

Parámetros

filename
: Nombre para el archivo descargado

data
: Contenidos del archivo, el archivo de datos, si tiene valor NULL y **$filename** es una ruta de archivo legible existente, su contenido se leerá en su lugar

set_mime
: Si se trata de enviar el tipo MIME real , si tiene valor TRUE, se enviará el tipo de archivo MIME real, basado en la extensión del nombre de archivo, de modo que si su navegador tiene un controlador para ese tipo, puede usarlo

valores devueltos

void

Ejemplo

```
$datos = 'Aquí hay algo de texto!';
$nombre = 'mitext.txt';
force_download($nombre, $datos);
```

Si desea descargar un archivo existente de su servidor, debe hacer lo siguiente:

```
// El contenido de foto.jpg se leerá automáticamente
force_download('/path/a/foto.jpg', NULL);
```

File Helper

File Helper ayuda a trabajar con archivos

Cargar File Helper

File Helper se carga así:

```
$this->load->helper('file');
```

Funciones

delete_files()

bool delete_files(string $path [, bool $del_dir = FALSE [, bool $htdocs = FALSE]])

Descripción

Elimina TODOS los archivos contenidos en la ruta suministrada

Parámetros

path
: Ruta del directorio

del_dir
: Si también eliminar directorios, si se establece en TRUE, también se eliminarán los directorios contenidos en la ruta raíz suministrada

htdocs
: Si se salta la eliminación de **.htaccess** y los archivos de página de índice

Valores devueltos

TRUE en caso de éxito, FALSE en caso de error

Ejemplo

```
delete_files('./path/al/directorio/');

delete_files('./path/al/directorio/', TRUE);
```

Nota

Los archivos deben ser de escritura o propiedad del sistema para ser eliminados

get_dir_file_info()

array get_dir_file_info(string $source_dir, bool $top_level_only)

Descripción

Lee el directorio especificado y crea un array que contiene el nombre de archivo, tamaño, fecha y permisos, los subdirectorios contenidos en la ruta especificada solo se leen si se fuerzan al enviar el segundo parámetro a FALSE, ya que puede tratarse de una operación lenta

Parámetros

source_dir

Ruta del directorio

top_level_only

Si se debe mirar solo el directorio especificado, excluyendo los subdirectorios

Valores devueltos

Un array que contiene información sobre los contenidos del directorio suministrado

Ejemplo

```
$models_info = get_dir_file_info(APPPATH.'models/');
```

get_file_info()

array get_file_info(string $file [, array $returned_values = array('name', 'server_path', 'size', 'date')])

Descripción

Obtener información de un archivo, nombre, ruta, tamaño, fecha de modificación y permisos

Parámetros

file

Ruta del archivo

returned_values

Qué información devolver, las opciones válidas son : name, size, date, readable, writeable, executable y fileperms

Valores devueltos

Un array con la información requerida o FALSE si falla

get_filenames()

array get_filenames(string $source_dir [, bool $include_path = FALSE])

Descripción

Toma una ruta de servidor como entrada y devuelve un array que contiene los nombres de todos los archivos que contiene, la ruta del archivo se puede agregar opcionalmente a los nombres de archivo estableciendo el segundo parámetro en TRUE

Parámetros

source_dir

Ruta del directorio

include_path

Si se debe incluir la ruta como parte de los nombres de archivo

Valores devueltos

Un array de nombres de archivos

Ejemplo

```
$controllers = get_filenames(APPPATH.'controllers/');
```

get_mime_by_extension()

string get_mime_by_extension(string $filename)

Descripción

Traduce la extensión de nombre de archivo en un tipo MIME basado en **config/mimes.php**

Parámetros:

filename

Nombre de archivo

Valores devueltos

Cadena de tipo MIME o FALSE en si no puede determinar el tipo o leer el archivo de configuración MIME

Ejemplo

```
$nomarch = 'algun_archivo.png';
echo $nomarch.' tiene un tipo mime de '.get_mime_by_extension($nomarch);
```

Nota

Esta no es una forma precisa de determinar los tipos MIME de archivos, y está aquí estrictamente por conveniencia

No debe usarse con fines de seguridad

octal_permissions()

string octal_permissions(int $perms)

Descripción

Toma permisos numéricos, como los devueltos por **fileperms()**, y devuelve una notación octal de tres caracteres de permisos de archivos

Parámetros

perms

Permisos

Valores devueltos

String de permisos octal

Ejemplo

```
echo octal_permissions(fileperms('./index.php')); // 644
```

read_file()

> ***Nota***
>
> Esta función está **DESACONSEJADA**. Use la función nativa de PHP : **file_get_contents()** en su lugar

symbolic_permissions()

string symbolic_permissions(int $perms)

Descripción

Toma permisos numéricos, como los devueltos por **fileperms()**, y devuelve la notación simbólica estándar de los permisos de archivos

Parámetros

perms
: Permisos

Valores devueltos

String de permisos simbólicos

Ejemplo

```
echo symbolic_permissions(fileperms('./index.php'));  // -rw-r--r--
```

write_file()

bool write_file(string $path, string $data [, string $mode = 'wb'])

Descripción

Escribe datos en el archivo especificado en la ruta, si el archivo no existe, la función lo crea

Parámetros

path
: Ruta del archivo

data
: Datos para escribir en el archivo

mode
: Modo **fopen()**

Valores devueltos

TRUE si la escritura fue exitosa, FALSE en caso de error

Ejemplo

```
$datos = 'Algunos datos de archivo';
if ( ! write_file('./path/a/archivo.php', $datos))
{
   echo 'No se puede escribir el archivo';
}
else
{
   echo 'Archivo escrito!';
}
```

Opcionalmente, puede establecer el modo de escritura a través del tercer parámetro:

```
write_file('./path/to/file.php', $datos, 'r+');
```

El modo predeterminado es **'wb'**, por favor, consulte la guía de usuario de PHP para conocer las opciones de modo

Nota

La ruta es relativa al archivo **index.php** de su sitio principal, **NO a su controlador** ni a sus archivos, CodeIgniter utiliza un controlador frontal para que las rutas sean siempre relativas al índice principal del sitio

Nota

Esta función adquiere un bloqueo exclusivo en el archivo mientras escribe en él

Form Helper

Form Helper ayuda a trabajar con formularios

Cargar Form Helper

Form Helper se carga así:

```
$this->load->helper('form');
```

Escapar valores de campo

Es posible que necesite usar HTML y caracteres entrecomillados en su formulario, para hacer eso de manera segura, necesitará usar la función **html_escape()**

Considere el siguiente ejemplo:

```
$string = 'Aquí hay una string que contiene el texto "entrecomillado".';

<input type="text" name="mi_campo" value="<?php echo $string; ?>" />
```

Como el string anterior contiene un conjunto entrecomillado, hará que el formulario se rompa, la función **html_escape()** convierte caracteres especiales HTML para que se puedan usar de forma segura:

```
<input type="text" name="mi_campo" value="<?php echo html_escape($string); ?>" />
```

Nota

Si usa alguna de las funciones auxiliares de formulario enumeradas en esta página, los valores de formulario se escaparán automáticamente, por lo que no es necesario llamar a esta función, úselo solo si está creando sus propios elementos de formulario

Funciones

form_button()

string form_button([string $data = '' [, string $content = '' [, mixed $extra = '']]])

Descripción

Le permite generar un elemento de botón estándar, puede pasar el nombre y el contenido del botón en el primer y segundo parámetro

Parámetros

data

Nombre del botón

contenido

Etiqueta de botón

extra

Atributos adicionales que se agregarán a la etiqueta, ya sea como un array o un string literal

Valores devueltos

Una etiqueta de botón HTML

Ejemplo

```
echo form_button('name','content');
// Would produce: <button name="name" type="button">Content</button>
```

O puede pasar un array asociativo que contiene todos los datos que desea que contenga su formulario:

```
$datos = array(
  'name'    => 'button',
  'id'      => 'button',
  'value'   => 'true',
  'type'    => 'reset',
  'content' => 'Reset'
);

echo form_button($datos);
// Would produce: <button name="button" id="button" value="true" type="reset">Reset</button>
```

Si desea que su formulario contenga algunos datos adicionales, como JavaScript, puede pasarlo como un string en el tercer parámetro:

```
$js = 'onClick="alguna_function()"';
echo form_button('mi_button', 'Pulsame', $js);
```

form_checkbox()

string form_checkbox([array $data = '' [, string $value = '' [, bool $checked = FALSE [, mixed $extra = '']]])

Descripción

Le permite generar un campo de casilla de verificación

Parámetros

data

Datos de atributos de campo

value

Valor del campo

checked

Si marcar la casilla como marcada

extra

Atributos adicionales que se agregarán a la etiqueta, ya sea como un array o un string literal

Valores devueltos

Una etiqueta de entrada HTML checkbox

Ejemplo

```
echo form_checkbox('noticias', 'accept', TRUE);

// Produce:  <input type="checkbox" name="noticias" value="accept" checked="checked" />
```

El tercer parámetro contiene un booleano TRUE / FALSE para determinar si la casilla debe ser marcada o no.

De forma similar a las otras funciones de formulario en este asistente, también puede pasar un array de atributos a la función:

```
$datos = array(
  'name'    => 'noticias',
  'id'      => 'id_noticias',
  'value'   => 'accept',
  'checked' => TRUE,
  'style'   => 'margin:10px'
);

echo form_checkbox($datos);
// Produce: <input type="checkbox" name="noticias" id="id_noticias" value="accept" checked="checked"
style="margin:10px" />
```

Además, como con otras funciones, si desea que la etiqueta contenga datos adicionales como JavaScript, puede pasarla como un string en el cuarto parámetro:

```
$js = 'onClick="alguna_function()"';
echo form_checkbox('noticias', 'accept', TRUE, $js);
```

O puede pasarlo como una array:

```
$js = array('onClick' => 'alguna_function();');
echo form_checkbox('noticias', 'accept', TRUE, $js);
```

form_close()

string form_close([string $extra = ''])

Descripción

Produce una etiqueta de cierre **</ form>**, la única ventaja de usar esta función es que le permite pasar datos que se agregarán debajo de la etiqueta

Parámetros

extra

Cualquier cosa para agregar después de la etiqueta de cierre

Valores devueltos

Una etiqueta de cierre de formulario HTML

Ejemplo

```
$string = '</div></div>';
echo form_close($string);

// Produce:  </form> </div></div>
```

form_dropdown()

string form_dropdown([string $name = '' [, array $options = array() [, array $selected = array() [, mixed $extra = '']]]])

Descripción

Le permite crear un campo desplegable estándar, el primer parámetro contendrá el nombre del campo, el segundo contendrá un array asociativo de opciones y el tercero el valor que desea seleccionar, también puede pasar un array de elementos múltiples a través del tercer parámetro, y CodeIgniter creará una selección múltiple para usted

Parámetros

name

Nombre del campo

options

Un array asociativo de opciones para ser listadas

selected

Lista de campos para marcar con el atributo seleccionado

extra

Atributos adicionales que se agregarán a la etiqueta, ya sea como un array o un string literal

Valores devueltos

Una lista desplegable HTML de campo de selección

```
$opciones = array(
  'pequeña' => 'Camisa pequeña',
  'mediana' => 'Camisa mediana',
  'grande'  => 'Camisa grande',
  'extra'   => 'Camisa extragrande',
);

$camisas_en_venta = array('pequeña', 'grande');
echo form_dropdown('camisas', $opciones, 'grande');

/*
 Produciría:

 <select name="camisas">
    <option value="pequeña">Camisa pequeña</option>
    <option value="mediana">Camisa mediana</option>
    <option value="grande" selected="selected">Camisa grande</option>
    <option value="extra">Camisa extragrande</option>
 </select>
*/

echo form_dropdown('camisas', $opciones, $camisas_en_venta);

/*
 Produciría:

 <select name="camisas" multiple="multiple">
    <option value="pequeña" selected="selected">Camisa pequeña</option>
    <option value="mediana">Camisa mediana</option>
    <option value="grande" selected="selected">Camisa grande</option>
    <option value="extra">Camisa extragrande</option>
 </select>
*/
```

Si desea que el **<select>** de apertura contenga datos adicionales, como un atributo id o JavaScript, puede pasarlo como un string en el cuarto parámetro:

```
$js = 'id="camisas" onChange="alguna_funcion();"';
echo form_dropdown('camisas', $opciones, 'grande', $js);
```

O puede pasarlo como un array:

```
$js = array(
  'id'       => 'camisas',
  'onChange' => 'alguna_funcion();'
);
echo form_dropdown('camisas', $opciones, 'grande', $js);
```

Si el array pasado como **$opciones** es un array multidimensional, **form_dropdown()** producirá un **<optgroup>** con la clave de array como etiqueta

form_error()

string form_error([string $field = '' [, string $prefix = '' [, string $suffix = '']]])

Descripción

Devuelve un mensaje de error de la **Form Validation Library**, asociado con el nombre de campo especificado, opcionalmente puede especificar las etiquetas de apertura y cierre para colocar el mensaje de error

Parámetros

field
Nombre del campo

prefix
Error al abrir la etiqueta

suffix
Error al cerrar la etiqueta

Valores devueltos

Mensaje (s) de error de validación de formulario con formato HTML

Ejemplo

```
// Suponiendo que el valor del campo 'username' fuera incorrecto:
echo form_error('mi_campo', '<div class="error">', '</div>');

// Produce: <div class="error">Mensaje de error asociado con el campo "username".</div>
```

form_fieldset()

string form_fieldset([string $legend_text = '' [, array $attributes = array()]])

Descripción

Le permite generar campos **fieldset / legend**

Parámetros

legend_text
Texto para poner en la etiqueta **<legend>**

attributes
Atributos que se establecerán en la etiqueta **<fieldset>**

Valores devueltos

Una etiqueta de apertura de conjunto de campos HTML

Ejemplo

```
echo form_fieldset('Dirección');
echo "<p>aquí el contenido del fieldset</p>\n";
```

```
echo form_fieldset_close();

/*
 Produce:

    <fieldset>
       <legend>Dirección</legend>
          <p>aquí el contenido del formulario</p>
    </fieldset>
*/
```

De forma similar a otras funciones, puede enviar un array asociativo en el segundo parámetro si prefiere establecer atributos adicionales:

```
$attributos = array(
  'id'    => 'inf_contacto',
  'class' => 'inf_contacto'
);

echo form_fieldset('Información de contacto', $attributos);
echo "<p>aquí el contenido del fieldset</p>\n";
echo form_fieldset_close();

/*
 Produce:

 <fieldset id="inf_contacto" class="inf_contacto">
    <legend>Información de contacto</legend>
    <p>aquí el contendo del form</p>
 </fieldset>
*/
```

form_fieldset_close()

string form_fieldset_close([string $extra = ''])

Descripción

Produce una etiqueta de cierre **</ fieldset>**, la única ventaja de usar esta función es que le permite pasar datos que se agregarán debajo de la etiqueta

Parámetros

extra

Cualquier cosa para agregar después de la etiqueta de cierre

Valores devueltos

Una etiqueta de cierre de campo de HTML

Ejemplo

```
$string = '</div></div>';
echo form_fieldset_close($string);
// Produce: </fieldset></div></div>
```

form_hidden()

string form_hidden(string $name [, string $value = ''])

Descripción

Le permite generar campos de entrada ocultos

Parámetros

name

Nombre del campo

value

Valor del campo

Valores devueltos

Una etiqueta de campo de entrada oculto HTML

Ejemplo

Puede enviar una string de nombre / valor para crear un campo:

```
form_hidden('nom_usuario', 'jujol');

// Produce: <input type="hidden" name="nom_usuario" value="jujol" />
```

... o puede enviar un array asociativo para crear múltiples campos:

```
$datos = array(
  'nombre' => 'Salvador Dalí',
  'email'  => 'salvador@example.com',
  'url'    => 'http://ejemplo.com'
);

echo form_hidden($datos);

/*
 Produciría:
 <input type="hidden" name="nombre" value="Salvador Dalí" />
 <input type="hidden" name="email" value="salvador@example.com" />
 <input type="hidden" name="url" value="http://example.com" />
*/
```

También puede pasar un array asociativo al valor de campo:

```
$datos = array(
  'nombre' => 'Antonio Gaudí',
  'email'  => 'antonio@example.com',
  'url'    => 'http://example.com'
);

echo form_hidden('mi_array', $datos);

/*
 Produciría:

 <input type="hidden" name="mi_array[nambre]" value="Antonio Gaudí" />
 <input type="hidden" name="mi_array[email]" value="antonio@example.com" />
 <input type="hidden" name="mi_array[url]" value="http://example.com" />
*/
*/
```

Si desea crear campos de entrada ocultos con atributos adicionales:

```
$datos = array(
  'type'  => 'hidden',
  'name'  => 'email',
  'id'    => 'mail_oculto',
  'value' => 'laura@example.com',
  'class' => 'mail_oculto'
);

echo form_input($datos);

/*
 Produce:

 <input type="hidden" name="email" value="laura@example.com" id="mail_oculto" class="mail_oculto"
  />
*/
```

form_input()

string form_input([array $data = '' [, string $value = '' [, mixed $extra = '']]])

Descripción

Le permite generar un campo de entrada de texto estándar

Parámetros

data
: Datos de atributos de campo

value
: Valor del campo

extra

Atributos adicionales que se agregarán a la etiqueta, ya sea como un array o un string literal

Valores devueltos

Una etiqueta de campo de entrada de texto HTML

Ejemplo

Como mínimo tiene que pasar el nombre del campo en el primer parámetro y el valor en el segundo:

```
echo form_input('nom_usuario', 'antonio_gaudí');
```

O puede pasar un array asociativo que contiene todos los datos que desea que contenga su formulario:

```
$datos = array(
  'name'      => 'nom_usuario',
  'id'        => 'nom_usuario',
  'value'     => 'antonio_gaudí',
  'maxlength' => '100',
  'size'      => '50',
  'style'     => 'width:50%'
);

echo form_input($datos);

/*
Produce:

<input type="text" name="nom_usuario" value="antonio_gaudí" id="nom_usuario" maxlength="100"
 size="50" style="width:50%"  />
*/
```

Si desea que su formulario contenga algunos datos adicionales, como JavaScript, puede pasarlo como un string en el tercer parámetro:

```
$js = 'onClick="alguna_function()"';
echo form_input('nom_usuario', 'antonio_gaudí', $js);
```

O puede pasarlo como un array:

```
$js = array('onClick' => 'some_function();');
echo form_input('nom_usuario', 'antonio_gaudí', $js);
```

form_label()

string form_label([string $label_text = '' [, string $id = '' [, mixed $attributes = array()]]])

Descripción

Le permite generar una **<label>**

Parámetros

label_text

Texto para poner en la etiqueta **<label>**

id

ID del elemento de formulario para el que estamos haciendo una etiqueta

attributes

Atributos de HTML

Valores devueltos

Una etiqueta de etiqueta de campo HTML

Ejemplo

```
echo form_label('Cuál es su nombre?', 'nom_usuario');
// Produce:  <label for="nom_usuario">Cuál es su nombre?</label>
```

De forma similar a otras funciones, puede enviar un array asociativo en el tercer parámetro si prefiere establecer atributos adicionales:

```
$atributos = array(
  'class' => 'micustomclass',
  'style' => 'color: #000;'
);

echo form_label('Cuál es su nombre?', 'nom_usuario', $atributos);
// Produce:  <label for="nom_usuario" class="micustomclass" style="color: #000;">Cuál es su
//           nombre?</label>
```

form_multiselect()

string form_multiselect([string $name = '' [, array $options = array() [, array $selected = array() [, mixed $extra = '']]]])

Descripción

Le permite crear un campo multiselect estándar, el primer parámetro contendrá el nombre del campo, el segundo parámetro contendrá un array asociativo de opciones y el tercer parámetro contendrá el valor o valores que desea seleccionar

El uso de parámetros es idéntico al uso de **form_dropdown()** anterior, excepto, por supuesto, que el nombre del campo necesitará usar la sintaxis del array **POST**, por ejemplo **foo[ ]**

Parámetros

name

Nombre del campo

options

Un array asociativo de opciones para ser listadas

selected

Lista de campos para marcar con el atributo seleccionado

extra

Atributos adicionales que se agregarán a la etiqueta, ya sea como un array o un string literal

Valores devueltos

Una etiqueta de campo de selección múltiple HTML desplegable

form_open()

string form_open([string $action = '' [, array $attributes = '' [, array $hidden = array()]]])

Descripción

Crea una etiqueta de apertura de formulario con una URL base creada a partir de sus preferencias de configuración, opcionalmente, le permite agregar atributos de formulario y campos de entrada ocultos, y siempre agregará el atributo accept-charset basado en el valor del conjunto de caracteres en su archivo de configuración

El beneficio principal al utilizar esta etiqueta en lugar de la codificación estricta de su propio HTML es que permite que su sitio sea más portáble si cambian sus URL

Parámetros

action

Cadena URI de acción / destino de formulario

attributes

Atributos de HTML

hidden

Un array de definiciones de campos ocultos

Valores devueltos

Una etiqueta de apertura de formulario HTML

Ejemplo

```
echo form_open('email/envia');
```

El ejemplo anterior crea un formulario que apunta a su URL base más los segmentos "**email/envia**", como este:

```
<form method="post" accept-charset="utf-8" action="http://ejemplo.com/index.php/email/envia">
```

Agregar atributos

Los atributos se pueden agregar pasando un array asociativo al segundo parámetro, como este:

```
$attributos = array('class' => 'email', 'id' => 'mi_form');
echo form_open('email/envia', $attributos);
```

Alternativamente, puede especificar el segundo parámetro como un string:

```
echo form_open('email/envia', 'class="email" id="mi_form"');
```

Los ejemplos anteriores crean un formulario similar a este:

```
<form method="post" accept-charset="utf-8" action="http://ejemplo.com/index.php/email/envio" class="email"
id="mi_form">
```

Agregar campos de entrada ocultos

Los campos ocultos se pueden agregar pasando un array asociativo al tercer parámetro, así:

```
$oculto = array('usuario_nombre' => 'Carmen', 'miembro_id' => '234');
echo form_open('email/enviar', '', $oculto);
```

Puede omitir el segundo parámetro pasándole cualquier valor falso

El ejemplo anterior crearía un form similar a este:

```
<form method="post" accept-charset="utf-8" action="http://ejemplo.com/index.php/email/enviar">
        <input type="hidden" name="usuario_nombre" value="Carmen" />
        <input type="hidden" name="miembro_id" value="234" />
```

form_open_multipart()

string form_open_multipart([string $action = '' [, array $attributes = array() [, array $hidden = array()]]])

Descripción

Esta función es idéntica a la anterior **form_open()**, excepto que agrega un atributo **multipart**, necesario si desea usar el formulario para cargar archivos

Parámetros

action
: Cadena URI de acción / destino de formulario

attributes
: Atributos de HTML

hidden
: Un array de definiciones de campos ocultos

Valores devueltos

Una etiqueta de apertura de formulario multipart HTML

form_password()

string form_password([array $data = '' [, string $value = '' [, mixed $extra = '']]])

Descripción

Esta función es idéntica a la función form_input() anterior, excepto que utiliza el tipo de entrada "**password**"

Parámetros

data

Datos de atributos de campo

value

Valor del campo

extra

Atributos adicionales que se agregarán a la etiqueta, ya sea como un array o un string literal

Valores devueltos

Una etiqueta de campo de entrada de contraseña HTML

form_prep()

> ***Nota***
>
> Esta función esta **DESACONSEJADA** y es solo un alias para la función común **html_escape()** , usela en su lugar

form_radio()

string form_radio([array $data = '' [, string $value = '' [, bool $checked = FALSE [, mixed $extra = '']]]])

Descripción

Esta función es idéntica en todos los aspectos a la función **form_checkbox()** anterior excepto que usa el tipo de entrada "**radio**"

Parámetros

data

Datos de atributos de campo

value

Valor del campo

checked

Si marcar el botón de radio como marcado

extra

Atributos adicionales que se agregarán a la etiqueta, ya sea como un array o un string literal

Valores devueltos

Una etiqueta de entrada de radio HTML

form_reset()

string form_reset([string $data = '' [, string $value = '' [, mixed $extra = '']]])

Descripción

Le permite generar un botón de reinicio estándar, el uso es idéntico a **form_submit()**

Parámetros

data
Nombre del botón

value
Valor del botón

extra
Atributos adicionales que se agregarán a la etiqueta, ya sea como un array o un string literal

Valores devueltos

Una etiqueta de botón de reinicio de entrada HTML

form_submit()

string form_submit([string $data = '' [, string $value = '' [, mixed $extra = '']]])

Descripción

Le permite generar un botón de envío estándar

Parámetros

data
Nombre del botón

value
Valor del botón

extra
Atributos adicionales que se agregarán a la etiqueta, ya sea como un array o un string literal

Valores devueltos

Una etiqueta de envío de entrada HTML

Ejemplo

```
echo form_submit('mi_submit', 'Submit Post!');

// Produce:  <input type="submit" name="mi_submit" value="Submit Post!" />
```

De forma similar a otras funciones, puede enviar un array asociativo en el primer parámetro si prefiere establecer sus propios atributos, el tercer parámetro le permite agregar datos adicionales a su formulario, como JavaScript

form_textarea()

string form_textarea([array $data = '' [, string $value = '' [, mixed $extra = '']]])

Descripción

Esta función es idéntica a la función **form_input()** anterior, excepto que genera un tipo "**textarea**"

Nota

En lugar de los atributos **maxlength** y **size** en el ejemplo anterior, en su lugar se especificarán **rows** y **cols**

Parámetros

data

Datos de atributos de campo

value

Valor del campo

extra

Atributos adicionales que se agregarán a la etiqueta, ya sea como un array o un string literal

Valores devueltos

Una etiqueta HTML textarea

form_upload()

string form_upload([array $data = '' [, string $value = '' [, mixed $extra = '']]])

Descripción

Esta función es idéntica a la función **form_input()** anterior, excepto que utiliza el tipo de entrada "**file**", lo que permite su uso para cargar archivos

Parámetros

data

Datos de atributos de campo

value

Valor del campo

extra

Atributos adicionales que se agregarán a la etiqueta, ya sea como un array o un string literal

Valores devueltos

Una etiqueta de campo de entrada de carga de archivo HTML

set_checkbox()

string set_checkbox(string $field [, string $value = '' [, string $default = FALSE]])

Descripción

Le permite mostrar una casilla de verificación en el estado en que fue enviada, el primer parámetro debe contener el nombre de la casilla de verificación, el segundo parámetro debe contener su valor y el tercer parámetro, opcional, le permite establecer un elemento como predeterminado, use TRUE / FALSE

Parámetros

field

Nombre del campo

value

Valor para verificar

default

Si el valor también es uno predeterminado

Valores devueltos

Atributo '**checked**' o una string vacía

Ejemplo

```
<input type="checkbox" name="mi_check" value="1" <?php echo set_checkbox('mi_check', '1'); ?> />
<input type="checkbox" name="mi_check" value="2" <?php echo set_checkbox('mi_check', '2'); ?> />
```

set_radio()

string set_radio(string $field [, string $value = '' [, string $default = FALSE]])

Descripción

Le permite mostrar botones de opción en el estado en que fueron enviados, esta función es idéntica a la función **set_checkbox()** anterior

Parámetros

field

Nombre del campo

value

Valor para verificar

default

Si el valor también es uno predeterminado

Valores devueltos

Atributo '**checked**' o una string vacía

Ejemplo

```
<input type="radio" name="mi_radio" value="1" <?php echo  set_radio('mi_radio', '1', TRUE); ?> />
<input type="radio" name="mi_radio" value="2" <?php echo  set_radio('mi_radio', '2'); ?> />
```

Nota

Si está utilizando la clase **Form Validation**, siempre debe especificar una regla para su campo, incluso si está vacía, para que funcionen las funciones **set_*()**, esto se debe a que si se define un objeto de **Form Validation**, el control para el conjunto **set_*()** se transfiere a un método de la clase en lugar de a la función auxiliar genérica

set_select()

string set_select(string $field [, string $value = '' [, string $default = FALSE]])

Descripción

Si usa un menú **<select>**, esta función le permite visualizar el elemento del menú que se seleccionó, el primer parámetro debe contener el nombre del menú de selección, el segundo parámetro debe contener el valor de cada elemento y el tercer parámetro, opcional, le permite establecer un elemento como predeterminado, use TRUE / FALSE

Parámetros

field

Nombre del campo

value

Valor para verificar

default

Si el valor también es uno predeterminado

Valores devueltos

Atributo '**seleccionado**' o una string vacía

Ejemplo

```
<select name="mi_select">
  <option value="uno" <?php echo  set_select('mi_select', 'uno', TRUE); ?> >Uno</option>
  <option value="dos" <?php echo  set_select('mi_select', 'dos'); ?> >Dos</option>
  <option value="tres" <?php echo  set_select('mi_select', 'tres'); ?> >Tres</option>
</select>
```

set_value()

string set_value(string $field [, string $default = '' [, bool $html_escape = TRUE]])

Descripción

Le permite establecer el valor de un formulario de entrada o área de texto, debe proporcionar el nombre del campo a través del primer parámetro de la función, el segundo parámetro, opcional, le permite establecer un valor predeterminado para el formulario, el tercer parámetro, opcional, le permite desactivar el escapado de HTML del valor, en caso de que necesite usar esta función en combinación con, por ejemplo, **form_input()** y evitar el doble escape

Parámetros

field

Nombre del campo

default

Valor predeterminado

html_escape

Si se debe desactivar el escape de HTML del valor

Valores devueltos

Valor de campo

Ejemplo

```
<input type="text" name="cantidad" value="<?php echo set_value('cantidad', '0'); ?>" size="50" />
```

El formulario anterior mostrará "0" cuando se cargue por primera vez

Nota

Si ha cargado la **Form Validation Library** y ha establecido una regla de validación para el nombre de campo en uso de este ayudante, reenviará la llamada al propio método **set_value()** de la **Form Validation Library**, de lo contrario, esta función se basa en **$_POST** para el valor del campo

validation_errors()

string validation_errors([string $prefix = '' [, string $suffix = '']])

Descripción

De forma similar a la función **form_error()**, devuelve todos los mensajes de error de validación producidos por la **Form Validation Library**, con etiquetas de apertura y cierre opcionales alrededor de cada uno de los mensajes

Parámetros

prefix

Error al abrir la etiqueta

sufijo

Error al cerrar la etiqueta

Valores devueltos

Mensaje(s) de error de validación de formulario con formato HTML

Ejemplo

```
echo validation_errors('<span class="error">', '</span>');

/*
 Produce, por ejemplo:

 <span class="error">El campo "email" no coniene una dirección válida!</span>
 <span class="error">El campo "contraseña" no coincide con el campo "repetir_contraseña"!</span>
*/
```

HTML Helper

HTML Helper ayuda a trabajar con HTML

Cargar HTML Helper

HTML Helper se carga así:

```
$this->load->helper('html');
```

Funciones

br()

> ***Nota***
>
> Esta función está **DESACONSEJADA**. Utilice la función nativa de PHP: **str_repeat()** en combinación con **
** en su lugar

doctype()

string doctype([string $type = 'xhtml1-strict'])

Descripción

Genera declaraciones de tipo de documento o DTD. XHTML 1.0 Strict se usa de manera predeterminada, pero hay muchos tipos de **DocType** disponibles

Parámetros

type

Doctype name

Valores devueltos

Etiqueta HTML DocType

Ejemplo

```
echo doctype(); // <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

echo doctype('html4-trans'); // <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

La siguiente es una lista de opciones de ' **DOCTYPE** ', tipo de documento, son configurables y extraídos de: **application/config/doctypes.php**

Document type	Opción	Resultado
XHTML 1.1	xhtml11	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
XHTML 1.0 Strict	xhtml1-strict	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
XHTML 1.0 Transitional	xhtml1-trans	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
XHTML 1.0 Frameset	xhtml1-frame	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
XHTML Basic 1.1	xhtml-basic11	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.1//EN" "http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd">
HTML 5	html5	<!DOCTYPE html>
HTML 4 Strict	html4-strict	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
HTML 4 Transitional	html4-trans	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
HTML 4 Frameset	html4-frame	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
MathML 1.01	mathml1	<!DOCTYPE math SYSTEM "http://www.w3.org/Math/DTD/mathml1/mathml.dtd">
MathML 2.0	mathml2	<!DOCTYPE math PUBLIC "-//W3C//DTD MathML 2.0//EN" "http://www.w3.org/Math/DTD/mathml2/mathml2.dtd">
SVG 1.0	svg10	<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
SVG 1.1 Full	svg11	<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
SVG 1.1 Basic	svg11-basic	<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1 Basic//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-basic.dtd">
SVG 1.1 Tiny	svg11-tiny	<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1 Tiny//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-tiny.dtd">
XHTML+MathML+SVG (XHTML host)	xhtml-math-svg-xh	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 plus MathML 2.0 plus SVG 1.1//EN" "http://www.w3.org/2002/04/xhtml-math-svg/xhtml-math-svg.dtd">
XHTML+MathML+SVG (SVG host)	xhtml-math-svg-sh	<!DOCTYPE svg:svg PUBLIC "-//W3C//DTD XHTML 1.1 plus MathML 2.0 plus SVG 1.1//EN" "http://www.w3.org/2002/04/xhtml-math-svg/xhtml-math-svg.dtd">
XHTML+RDFa 1.0	xhtml-rdfa-1	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN" "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
XHTML+RDFa 1.1	xhtml-rdfa-2	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.1//EN" "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-2.dtd">

heading()

string heading([string $data = '' [, string $h = '1' [, mixed $attributes = '']]])

Descripción

Le permite crear etiquetas de encabezado HTML, el primer parámetro contendrá los datos, el segundo el tamaño del encabezado

Parámetros

data

Contenido

h

Nivel de la etiqueta H, H1, H2,...

attributes

Atributos de HTML

Valores devueltos

Etiqueta de encabezado HTML

Ejemplo

```
echo heading('Bienvenido!', 3);
// Produce: <h3>Bienvenido!</ H3>
```

Además, para agregar atributos a la etiqueta de encabezado, como clases HTML, ids o estilos en línea, un tercer parámetro acepta un string o un array:

```
echo heading('Bienvenido!', 3, 'class="carmen"');
echo heading('Cómo está?', 4, array('id' => 'pregunta', 'class' => 'julia'));
```

El código anterior produce:

```
<h3 class="carmen">Bienvenido!<h3>
<h4 id="question" class="julia">Cómo está?</h4>
```

img()

string img([string $src = '' [, bool $index_page = FALSE [, array $attributes = '']]])

Descripción

Le permite crear etiquetas HTML **<img />**, el primer parámetro contiene la fuente de la imagen

Parámetros

src

Datos fuente de la imagen

index_page

Si se trata **$src** como un string de URI enrutada

attributes

Atributos de HTML

Valores devueltos

Etiqueta de imagen HTML

Ejemplo

```
echo img('images/picture.jpg');
// Produce: <img src="http://site.com/images/picture.jpg" />
```

Hay un segundo parámetro opcional que es un valor TRUE / FALSE que especifica si el **src** debe tener la página especificada por **$config['index_page']** agregada a la dirección que crea, presumiblemente, esto sería si estuviera usando un controlador de medios:

```
echo img('images/picture.jpg', TRUE);
// Produce: <img src="http://site.com/index.php/images/picture.jpg" alt="" />
```

Además, un array asociativo se puede pasar a la función **img()** para un control completo sobre todos los atributos y valores, si no se proporciona un atributo **alt**, CodeIgniter generará un string vacío:

```
$imagen_propiedades = array(
  'src'   => 'images/picture.jpg',
  'alt'   => 'Un pequeño gatito, un dulce micifú',
  'class' => 'post_images',
  'width' => '200',
  'height'=> '200',
  'title' => 'Algo nuevo,super, super original...',
  'rel'   => 'lightbox'
);

img($imagen_propiedades);

// <img src="http://site.com/index.php/images/picture.jpg" alt=" Un pequeño gatito, un dulce
//  micifú"
// class="post_images" width="200" height="200"
// title=" Algo nuevo,super, super original..." rel="lightbox" />
```

link_tag()

string link_tag([string $href = '' [, string $rel = 'stylesheet' [, string $type = 'text/css' [, string $title = '' [, string $media = '' [, bool $index_page = FALSE]]]]])

Descripción

Le permite crear etiquetas HTML **<link />**, esto es útil para enlaces de hojas de estilo, así como otros enlaces

Parámetros

href
: Qué estamos vinculando

rel
: Tipo de relación, parámetro opcional

type
: Tipo del documento relacionado, parámetro opcional

title
: Título del enlace, parámetro opcional

media
: Tipo de medios, parámetro opcional

index_page
: Si se trata **$src** como una string de URI enrutada, parámetro opcional

Valores devueltos

Etiqueta de enlace HTML

Ejemplo

index_page es un valor booleano que especifica si **href** debe tener la página especificada por **$config['index_page']** agregada a la dirección que crea:

```
echo link_tag('css/mystyles.css');
// gives <link href="http://site.com/css/mystyles.css" rel="stylesheet" type="text/css" />

echo link_tag('favicon.ico', 'shortcut icon', 'image/ico');
// <link href="http://site.com/favicon.ico" rel="shortcut icon" type="image/ico" />

echo link_tag('feed', 'alternate', 'application/rss+xml', 'My RSS Feed');
// <link href="http://site.com/feed" rel="alternate" type="application/rss+xml"
// title="My RSS Feed" />
```

Además, un array asociativo se puede pasar a la función **link()** para un control completo sobre todos los atributos y valores:

```
$link = array(
  'href'  => 'css/printer.css',
  'rel'   => 'stylesheet',
  'type'  => 'text/css',
  'media' => 'print'
);

echo link_tag($link);
// <link href="http://site.com/css/printer.css" rel="stylesheet" type="text/css" media="print" />
```

meta()

string meta([string $name = '' [, string $content = '' [, string $type = 'name' [, string $newline = "n"]]]])

Descripción

Generar meta etiquetas, puede pasar strings a la función, o arrays simples, o multidimensionales

Parámetros

name
: Meta nombre

content
: Meta contenido

type
: Meta type

newline

Carácter Newline

Valores devueltos

HTML meta tag

Ejemplo

```
echo meta('description', 'Mi gran sitio');
// Produce:  <meta name="description" content="Mi gran sitio" />

echo meta('Content-type', 'text/html; charset=utf-8', 'equiv');
// Tenga en cuenta el tercer parámetro. Puede ser "equiv" or "name"
// Produce:  <meta http-equiv="Content-type" content="text/html; charset=utf-8" />

echo meta(array('name' => 'robots', 'content' => 'no-cache'));
// Produce:  <meta name="robots" content="no-cache" />

$meta = array(
 array(
    'name'    => 'robots',
    'content' => 'no-cache'
 ),
 array(
    'name'    => 'description',
    'content' => 'Mi gran sitio'
 ),
 array(
    'name'    => 'keywords',
    'content' => 'dalí, gaudí, mortadelo, asterix,kafka, Lovecraft'
 ),
 array(
    'name'    => 'robots',
    'content' => 'no-cache'
 ),
 array(
    'name'    => 'Content-type',
    'content' => 'text/html; charset=utf-8', 'type' => 'equiv'
 )
);

echo meta($meta);
// Produce:
//
// <meta name="robots" content="no-cache" />
// <meta name="description" content=" Mi gran sitio" />
// <meta name="keywords" content=" dalí, gaudí, mortadelo, asterix,kafka, Lovecraft" />
// <meta name="robots" content="no-cache" />
// <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
```

nbs()

> ***Nota***
>
> Esta función está **DESACONSEJADA**. Utilice la función nativa de PHP: **str_repeat()** en combinación con ** ** en su lugar

ol()

string ol(array $list, array $attributes = '')

Descripción

Idéntico a **ul()**, solo que produce la etiqueta **<ol>** para las listas ordenadas en lugar de **<ul>**

Parámetros

list
: Entradas de la lista

attributes
: Atributos de HTML

Valores devueltos

Lista ordenada con formato HTML

ul()

string ul(array $list [, array $attributes = ''])

Descripción

Le permite generar listas HTML sin ordenar a partir de arrays simples o multidimensionales

Parámetros

list
: Entradas de la lista

attributes
: Atributos de HTML

Valores devueltos

Lista desordenada con formato HTML

Ejemplo

```
$lista = array(
  'verde',
  'rojo',
  'violeta'
);

$atributos = array(
  'class' => 'objetos',
  'id'    => 'mi_lista'
);

echo ul($lista, $atributos);
```

El código anterior produce:

```
<ul class="objetos" id="mi_lista">
  <li>verde</li>
  <li>rojo</li>
  <li>violeta</li>
</ul>
```

Aquí hay un ejemplo más complejo, usando un array multidimensional:

```
$atributos = array(
  'class' => 'objetos',
  'id'    => 'mi_lista'
);


$lista = array(
  'colores'  => array(
   'verde',
   'rojo',
   'violeta'
 ),
 'formas'  => array(
    'triángulos',
    'cuadrados',
    'círculos' => array(
       'elipse',
       'óvalo',
       'esfera'
    )
 ),
 'huesos'  => array(
    'cráneo',
    'oreja' => array(
       'oído medio' => array(
          'martillo',
```

```
            'yunque',
            'estribo'
        ),
        'mano',
        'brazo',
        'vértebras'
    )
 )
);

echo ul($lista, $atributos);
```

El código anterior producirá esto:

```
<ul class="objetos" id="mi_lista">
 <li>colores
    <ul>
       <li>verde</li>
       <li>rojo</li>
       <li>violeta</li>
    </ul>
 </li>
 <li>formas
    <ul>
       <li>triángulos</li>
       <li>cuadrados</li>
       <li>círculos
          <ul>
                <li>elipse</li>
                <li>óvalo</li>
                <li>esfera</li>
          </ul>
    </li>
    </ul>
    </li>
    <li>huesos
       <ul>
          <li>cráneo</li>
          <li>oreja
             <ul>
                <li>oído medio
                   <ul>
                      <li>martillo</li>
                      <li>yunque</li>
```

```
                    <li>estribo</li>
                  </ul>
              </li>
                 <li>mano</li>
                 <li>brazo</li>
                 <li>vértebras</li>
           </ul>
        </li>
     </ul>
  </li>
</ul>
```

Inflector Helper

Inflector Helper le permiten cambiar las palabras en inglés a plural, singular, camel case, etc.

Cargar Inflector Helper

Inflector Helper se carga así:

```
$this->load->helper('inflector');
```

Funciones

camelize()

string camelize(string $str)

Descripción

Cambia un string de palabras separadas por espacios o guiones bajos a camel case

Parámetros

str
String de entrada

Valores devueltos

String camelizada

Ejemplo

```
echo camelize('todo_mi_cariño_para_ti'); // Presenta: ' todoMiCariñoParaTi'
```

humanize()

string humanize(string $str [, string $separator = '_'])

Descripción

Toma varias palabras separadas por guiones bajos y agrega espacios entre ellas, cada palabra está en mayúscula

Parámetros

str
String de entrada

separator
Separador de entrada

Valores devueltos

String humanizada

Ejemplo

```
echo humanize('todo_mi_cariño_para_ti'); // Presenta: 'Todo Mi Cariño Para Ti'
```

Para usar guiones en lugar de guiones bajos:

```
echo humanize('todo-mi-cariño-para-ti', '-'); // Presenta: 'Todo Mi Cariño Para Ti'
```

is_countable()

bool is_countable(string $word)

Descripción

Comprueba si la palabra dada tiene una versión plural

Parámetros

word
String de entrada

Valores devueltos

TRUE si la palabra es contable o FALSE si no

Ejemplo

```
is_countable('equipment'); // Devuelve: FALSE
```

plural()

string plural(string $str)

Descripción

Cambia una palabra singular a plural

Parámetros

str
String de entrada

Valores devueltos

Una palabra plural

Ejemplo

```
echo plural('conejo'); // Presenta: 'conejos'
```

singular()

string singular(string $str)

Descripción

Cambia una palabra plural a singular

Parámetros

str

String de entrada

Valores devueltos

Una palabra singular

Ejemplo

```
echo singular('conejos'); // Presenta: 'conejo'
```

underscore()

string underscore(string $str)

Descripción

Toma múltiples palabras separadas por espacios y las subraya

Parámetros

str

String de entrada

Valores devueltos

String que contiene guiones bajos en lugar de espacios

Ejemplo

```
echo underscore('todo mi cariño para ti'); // Presenta: 'todo_mi_cariño_para_ti'
```

Language Helper

Language Helper ayuda a trabajar con archivos de idioma

Cargar Language Helper

Language Helper se carga así:

```
$this->load->helper('language');
```

Funciones

lang()

string lang(string $line [, string $for = '' [, array $attributes = array()]])

Descripción

Esta función devuelve una línea de texto de un archivo de idioma cargado con sintaxis simplificada, puede ser más deseable para los archivos de vista que **CI_Lang::line()**

Parámetros

line
Clave de línea de idioma

for
Atributo HTML "**for**" ,**ID** del elemento para el que estamos creando la etiqueta

attributes
Cualquier atributo HTML adicional

Valores devueltos

La línea del idioma, en una etiqueta '**label**' de HTML, si el parámetro **$for** no está vacío

Ejemplo

```
echo lang('language_key');
// Presenta: la línea de idioma


echo lang('language_key', 'form_item_id', array('class' => 'mi_clase'));
// Presenta: <label for="form_item_id" class="mi_clase">línea de idioma</label>
```

Number Helper

Number Helper ayuda a trabajar con datos numéricos

Cargar Number Helper

Number Helper se carga así:

```
$this->load->helper('number');
```

Funciones

byte_format()

string byte_format(mixed $num [, int $precision = 1])

Descripción

Formatea los números como bytes, según el tamaño, y agrega el sufijo apropiado

Parámetros

num

Número de bytes

precision

Precisión del punto flotante

Valores devueltos

String con los datos formateados

Ejemplos

```
echo byte_format(456); // Devuelve 456 Bytes
echo byte_format(4567); // Devuelve 4.5 KB
echo byte_format(45678); // Devuelve 44.6 KB
echo byte_format(456789); // Devuelve 447.8 KB
echo byte_format(3456789); // Devuelve 3.3 MB
echo byte_format(12345678912345); // Devuelve 1.8 GB
echo byte_format(123456789123456789); // Devuelve 11,228.3 TB
```

Un segundo parámetro opcional le permite establecer la precisión del resultado:

```
echo byte_format(45678, 2); // Devuelve 44.61 KB
```

Nota

El texto generado por esta función se encuentra en el siguiente archivo de idioma:
language/<your_lang>/number_lang.php

Path Helper

Path Helper le permiten trabajar con rutas de archivos en el servidor

Cargar Path Helper

Path Helper se carga así:

```
$this->load->helper('path');
```

Funciones

set_realpath()

string set_realpath(string $path [, bool $check_existance = FALSE])

Descripción

Esta función devolverá una ruta de servidor sin enlaces simbólicos o estructuras de directorio relativas. Un segundo argumento opcional provocará que se active un error si la ruta no se puede resolver

Parámetros

path
: Ruta

check_existance
: Si se debe verificar si la ruta realmente existe

Valores devueltos

Un path absoluto

Ejemplo

```
$file = '/etc/php5/apache2/php.ini';
echo set_realpath($file); // Imprime '/etc/php5/apache2/php.ini'

$non_existent_file = '/path/to/non-exist-file.txt';

// Muestra un error, ya que el path no se puede resolver
echo set_realpath($non_existent_file, TRUE);

// Imprime: '/path/to/non-exist-file.txt'
echo set_realpath($non_existent_file, FALSE);

$directory = '/etc/php5';
echo set_realpath($directory);  // Imprime '/etc/php5/'
```

```
$non_existent_directory = '/path/to/nowhere';

// Muestra un error, ya que el path no se puede resolver
echo set_realpath($non_existent_directory, TRUE);

// Imprime '/path/to/nowhere'
echo set_realpath($non_existent_directory, FALSE);
```

Security Helper

Security Helper contiene funciones relacionadas con la seguridad

Cargar Security Helper

Security Helper se carga así:

```
$this->load->helper('security');
```

Funciones

do_hash()

> ***Nota***
>
> Esta función está **DESACONSEJADA**. Use la función nativa **de PHP: hash()** en su lugar

encode_php_tags()

string encode_php_tags(string $str)

Descripción

Esta es una función de seguridad que convierte etiquetas PHP a entidades

> ***Nota***
>
> **xss_clean()** lo hace automáticamente, si lo usa

Parámetros

str

String de entrada

Valores devueltos

String con formato seguro

Ejemplo

```
$string = encode_php_tags($string);
```

sanitize_filename()

string sanitize_filename(string $filename)

Descripción

Proporciona protección contra el recorrido del directorio

Esta función es un alias para **CI_Security::sanitize_filename()**, para obtener más información, consulte la documentación de la **Security Library**

Parámetros

filename
Nombre de archivo

Valores devueltos

Nombre de archivo desinfectado

strip_image_tags()

string strip_image_tags(string $str)

Descripción

Esta es una función de seguridad que quitará etiquetas de imagen de un string. Deja la URL de la imagen como texto sin formato

Esta función es un alias para **CI_Security::strip_image_tags()**, para obtener más información, consulte la documentación de la **Security Library**

Parámetros

str
String de entrada

Valores devueltos

La string de entrada sin etiquetas de imagen

Ejemplo

```
$string = strip_image_tags($string);
```

xss_clean()

string xss_clean(string $str [, bool $is_image = FALSE])

Descripción

Proporciona filtrado de Cross Site Script Hack

Esta función es un alias para **CI_Input::xss_clean()**, para obtener más información, consulte la documentación de la **Input Library**

Parámetros

str

Datos de entrada

is_image

Si se trata de una imagen

Valores devueltos

Cadena XSS-clean

String Helper

String Helper ayuda a trabajar con strings

Importante

Tenga en cuenta que estas funciones **NO** están diseñadas ni son adecuadas para ningún tipo de lógica relacionada con la seguridad

Cargar String Helper

String Helper se carga así:

```
$this->load->helper('string');
```

Funciones

alternator()

mixed alternator(mixed $args)

Descripción

Permite que dos o más elementos alternen entre sí al ciclar dentro de un bucle

Parámetros

args

Cantidad variable de argumentos

Valores devueltos

Cadena(s) alternadas

Ejemplo

```
for ($i = 0; $i < 10; $i++)
{
   echo alternator('string uno', 'string dos');
}
```

Puede agregar tantos parámetros como desee y con cada iteración de su ciclo, se devolverá el siguiente elemento :

```
for ($i = 0; $i < 10; $i++)
{
   echo alternator('uno', 'dos', 'tres', 'cuatro', 'cinco');
}
```

> **Nota**
>
> Para usar múltiples llamadas separadas a esta función, simplemente llame a la función sin argumentos para reinicializar

increment_string()

string increment_string(string $str [, string $separator = '_' [, int $first = 1]])

Descripción

Incrementa un string añadiéndole un número o aumentando el número, útil para crear "copias" o un archivo o duplicar el contenido de la base de datos que tiene valores únicos o slugs

Parámetros

str
: String de entrada

separator
: Separador para agregar al número

first
: Número inicial

Valores devueltos

Un string incrementado

Ejemplo

```
echo increment_string('archivo', '_');    // "archivo_1"
echo increment_string('archivo', '-', 2); // "archivo-2"
echo increment_string('archivo_4');       // "archivo_5"
```

quotes_to_entities()

string quotes_to_entities(string $str)

Descripción

Convierte las comillas simples y dobles en una cadena a las entidades HTML correspondientes

Parámetros

str
: String de entrada

Valores devueltos

Cadena con comillas convertidas a entidades HTML

Ejemplo

```
$string = "Joe's \"dinner\"";
$string = quotes_to_entities($string); //resultado: "Joe&#39;s &quot;dinner&quot;"
```

random_string()

string random_string([string $type = 'alnum' [, int $len = 8]])

Descripción

Genera un string aleatorio según el tipo y la longitud que especifique, útil para crear contraseñas o generar hashes aleatorios

Parámetros

type

Tipo de aleatorización

Las siguientes opciones están disponibles:

- **alpha**: Un string con letras mayúsculas y minúsculas solamente
- **alnum**: Un string alfanumérico con mayúsculas y minúsculas
- **basic**: Un número aleatorio basado en **mt_rand()**
- **numeric**: Un string numérico
- **nozero**: Un string numérico sin ceros
- **md5**: Un número aleatorio encriptado basado en **md5()** (longitud fija de 32)
- **sha1**: Un número aleatorio encriptado basado en **sha1()** (longitud fija de 40)

len

Longitud del string de salida

Valores devueltos

Un string aleatorio

Ejemplo

```
echo random_string('alnum', 16);
```

reduce_double_slashes()

string reduce_double_slashes(string $str)

Descripción

Convierte barras diagonales dobles en un string de una sola barra, excepto las que se encuentran en los prefijos de protocolo URL (por ejemplo, http: //)

Parámetros

str

String de entrada

Valores devueltos

Un string con barras normalizadas

Ejemplo

```
$string = "http://ejemplo.com//index.php";

echo reduce_double_slashes($string);
// resulado: "http://ejemplo.com/index.php"
```

reduce_multiples()

string reduce_multiples(string $str [, string $character = '' [, bool $trim = FALSE]])

Descripción

Reduce múltiples instancias de un carácter particular que ocurren uno detrás de otro

Parámetros

str

Texto para buscar

character

Carácter para reducir

trim

Si se establece en TRUE, eliminará las apariciones del carácter especificado al principio y al final del string

Valores devueltos

String reducida

Ejemplo

```
$string = "Carmen, María,, Raquel, Helena";
$string = reduce_multiples($string,","); // Produce: "Carmen, María, Raquel, Helena"
```

Eliminar las apariciones del carácter al principio y al final del string :

```
$string = ",Carmen, María,, Raquel, Helena,";
$string = reduce_multiples($string, ", ", TRUE); //Produce: "Carmen, María, Raquel, Helena"
```

repeater()

> ***Nota***
>
> Esta función esta **DESACONSEJADA**, use la función nativa de PHP: **str_repeat()** en su lugar

strip_quotes()

string strip_quotes(string $str)

Descripción

Elimina comillas simples y dobles de un string

Parámetros

str

String de entrada

Valores devueltos

Cadena sin las comillas

Ejemplo

```
$string = "Gallina's \"ponedoras\"";
$string = strip_quotes($string); // Produce: "Gallinas ponedoras"
```

strip_slashes()

mixed strip_slashes(mixed $data)

Descripción

Elimina cualquier barra de un array de strings

Parámetros

data

Cadena de entrada o un array de strings

Valores devueltos

String(s) sin las barras

Ejemplo

```
$str = array(
 'pregunta' => 'Tu nombre es S\'cooby-Doo?',
 'respuesta'   => 'No, mi nombre es R\'antanplan.'
);

$str = strip_slashes($str);


Lo anterior devolverá la siguiente array:

array(
 'pregunta' => "Tu nombre es S'cooby-Doo?",
 'respuesta'   => "No, mi nombre es R'antanplan."
);
```

Nota

Por razones históricas, esta función también aceptará y manejará las entradas de string, sin embargo, esto lo hace solo como un alias de **stripslashes()**

trim_slashes()

Nota

Esta función está **DESACONSEJADA**, use la función nativa de PHP: **trim()** en su lugar: **trim($str, '/');**

Text Helper

Text Helper ayuda a trabajar con texto

Cargar Text Helper

Text Helper se carga así:

```
$this->load->helper('text');
```

Funciones

ascii_to_entities()

string ascii_to_entities (string $str)

Descripción

Convierte valores ASCII en entidades de caracteres, incluidos los caracteres ASCII y MS Word altos que pueden causar problemas cuando se utilizan en una página web, de modo que se pueden mostrar de forma coherente independientemente de la configuración del navegador o almacenarse de forma fiable en una base de datos

Existe cierta dependencia de los juegos de caracteres soportados por su servidor, por lo que puede no ser 100% confiable en todos los casos, pero en su mayor parte debe identificar correctamente los caracteres fuera del rango normal (como los caracteres acentuados)

Parámetros

str

String de entrada

Valores devueltos

Un string con valores ASCII convertidos en entidades

Ejemplo

```
$string = ascii_to_entities($string);
```

character_limiter()

string character_limiter(string $str [, int $n = 500 [, string $end_char = '…']])

Descripción

Trunca un string a la cantidad de caracteres especificados, mantiene la integridad de las palabras, por lo que el recuento de caracteres puede ser un poco más o menos de lo que especifique

Parámetros

str

String de entrada

n

Número de caracteres

end_char

Carácter final (usualmente una elipsis)

Valores devueltos

Cadena de caracteres limitados

Ejemplo

```
$string = "Aquí hay una bonita string de texto que consta de doce palabras.";
$string = character_limiter($string, 20);
// Devuelve:  Aquí hay una bonita..
```

El tercer parámetro es un sufijo opcional agregado al string; si no se declara, se usa una elipse

> ***Nota***
>
> Si necesita truncar a un número exacto de caracteres, consulte la función **ellipsize()** a continuación

convert_accented_characters()

string convert_accented_characters(string $str)

Descripción

Transcribe los caracteres ASCII altos a equivalentes ASCII bajos, útil cuando los caracteres no ingleses deben usarse donde solo se usan con seguridad los caracteres ASCII estándar, por ejemplo, en las URL

Parámetros

str

String de entrada

Valores devueltos

Un string con caracteres acentuados convertidos

Ejemplo

```
$string = convert_accented_characters($string);
```

> ***Nota***
>
> Esta función utiliza un archivo de configuración complementario **application/config/foreign_chars.php** para definir el conjunto de la transliteración

ellipsize()

string ellipsize(string $str, int $max_length [, mixed $position = 1 [, string $ellipsis = '…']])

Descripción

Esta función quitará las etiquetas de una string, la dividirá en una longitud máxima definida e insertará puntos suspensivos

Parámetros

str

String de entrada

max_length

Límite de longitud de string

position

Posición para dividir en (int o float), el tercer parámetro es donde en la string los puntos suspensivos deberían aparecer de 0 a 1, de izquierda a derecha, por ejemplo. un valor de 1 colocará los puntos suspensivos a la derecha de la string, 5 en el medio y 0 a la izquierda

ellipsis

Qué usar como puntos suspensivos, por defecto inserta '**…**'

Valores devueltos

String elíptica

Ejemplo

```
$str = 'en_un_lugar_de_la_mancha_de_cuyo_nombre.jpg';
echo ellipsize($str, 30, .5);

// Produce: en_un_lugar_de_…cuyo_nombre.jpg
```

highlight_code()

string highlight_code(string $str)

Descripción

Colorea una string de código (PHP, HTML, etc.)

Parámetros

str

String de entrada

Valores devueltos

String con código resaltado a través de HTML

Ejemplo

```
$string = highlight_code($string);
```

La función utiliza la función **highlight_string()** de PHP, por lo que los colores utilizados son los especificados en su archivo **php.ini**

highlight_phrase()

string highlight_phrase(string $str, string $phrase [, string $tag_open = '<mark>' [, string $tag_close = '</mark>']])

Descripción

Resaltará una frase dentro de una string de texto

Parámetros

str
: String de entrada

phrase
: Frase a resaltar

tag_open
: Etiqueta de apertura utilizada para resaltar

tag_close
: Etiqueta de cierre para resaltar

Valores devueltos

Cadena con una frase resaltada en HTML

Ejemplo

```
$string = "En un lugar de la Mancha de cuyo nombre";
echo highlight_phrase($string, "Mancha", '<span style="color:#990000;">', '</span>');
```

El código anterior imprime:

```
En un lugar de la<span style="color:#990000;">Mancha</span>de cuyo nombre
```

Nota

Esta función solía usar la etiqueta **** de forma predeterminada. Los navegadores antiguos pueden no ser compatibles con la nueva etiqueta de marca HTML5, por lo que se recomienda que inserte el siguiente código CSS en su hoja de estilos si necesita admitir dichos navegadores:

```
mark {
 background: #ff0;
 color: #000;
};
```

word_censor()

string word_censor(string $str, array $censored [, string $replacement = ''])

Descripción

Le permite censurar palabras dentro de un string de texto, el primer parámetro contendrá el string original

Parámetros

str

String de entrada

censored

Un array de palabras que usted no permite

replacement

Opcional, puede contener un valor de reemplazo para las palabras, si no se especifica, se reemplazan con signos de libra: ####

Valores devueltos

String censurada

Ejemplo

```
$párrafo ='En un lugar de la Mancha de cuyo nombre';
$censurado = array('Monegros', 'Mancha', 'Barcelona', 'Madrid');
$string = word_censor($párrafo, $censurado, 'Galaxia');

echo $string; // Produce: En un lugar de la Galaxia de cuyo nombre
```

word_limiter()

string word_limiter(string $str [, int $limit = 100 [, string $end_char = '…']])

Descripción

Trunca una string a la cantidad de palabras especificadas

Parámetros

str

String de entrada

limit

Límite

end_char

Carácter final, por defecto puntos suspensivos ...

Valores devueltos

String truncada

Ejemplo

```
$string = "En un lugar de la Mancha de cuyo nombre";
$string = word_limiter($string, 4);

echo $string; // Produce: En un lugar de…
```

word_wrap()

string word_wrap(string $str [, int $charlim = 76])

Descripción

Divide un texto en la porciones del número de caracteres especificado, ajustado a palabras completas

Parámetros

str

String de entrada

charlim

Límite de caracteres

Valores devueltos

String dividido ajustado al número indicado y respetando palabras completas

Ejemplo

```
$string = "En un lugar de la Mancha de cuyo nombre";
echo word_wrap($string, 12);

// Produce:
// En un lugar
// de la Mancha
// de cuyo
// nombre
```

Typography Helper

Typography Helper ayuda a formatear el texto de forma semánticamente relevante

Cargar Typography Helper

Typography Helper se carga así:

```
$this->load->helper('typography');
```

Funciones

auto_typography()

string auto_typography(string $str [, bool $reduce_linebreaks = FALSE])

Descripción

Formatea texto para que sea HTML semántica y tipográficamente correcto, esta función es un alias para **CI_Typography::auto_typography(),** Puede obtener más información, consultando la **Typography Library**

Parámetros

str

String de entrada

reduce_linebreaks

Si se deben reducir las instancias múltiples de líneas dobles nuevas a dos

Valores devueltos

String tipográfica con formato HTML

Ejemplo

```
$string = auto_typography($string);
```

> ***Nota***
>
> El formateo tipográfico puede usar intensamente al procesador, particularmente si tiene un mucho contenido para formatear, si elige usar esta función, debería considerar cachear sus páginas

entity_decode()

string entity_decode(string $str, string $charset = NULL)

Descripción

Esta función es un alias para **CI_Security::entity_decode()**, para obtener más información, consulte la documentación de la **Security Library**

Parámetros

str

String de entrada

charset

Conjunto de caracteres

Valores devueltos

String entidades HTML decodificadas

nl2br_except_pre()

string nl2br_except_pre(string $str)

Descripción

Convierte caracteres de nueva línea a etiquetas **
** a menos que éstas aparezcan dentro de etiquetas **<pre>**, esta función es idéntica a la función nativa **nl2br()** PHP, excepto que ignora las etiquetas **<pre>**

Parámetros

str

String de entrada

Valores devueltos

String con saltos de línea con formato HTML

Ejemplo

```
$string = nl2br_except_pre($string);
```

URL Helper

URL Helper ayuda a trabajar con URL

Cargar URL Helper

URL Helper se carga así:

```
$this->load->helper('url');
```

Funciones

anchor()

string anchor(string $uri = '', string $title = '', mixed $attributes = '')

Descripción

Crea un enlace de anclaje HTML estándar basado en la URL de su sitio

Parámetros

uri
: String URI, puede contener cualquier segmento que desee adjuntar a la URL, al igual que con la función **site_url()** anterior, los segmentos pueden ser un string o un array

title
: Título de anclaje, si lo deja en blanco, se usará la URL

attributes
: Atributos de HTML, un string simple o un array asociativo con los atributos que le gustaría agregar al enlace

Valores devueltos

Hipervínculo HTML (etiqueta de anclaje)

> ***Nota***
>
> Si está creando enlaces que son internos a su aplicación, no incluya la URL base (http://...), esto se agregará automáticamente a partir de la información especificada en su archivo de configuración, incluya solo los segmentos de URI que desee adjuntar a la URL

Ejemplo

```
echo anchor('news/local/123', 'La noticia', 'title="Título de la noticia"');
// Imprime: <a href="http://ejemplo.com/index.php/news/local/123"
// title="Título de la noticia">La noticia</a>
```

```
echo anchor('news/local/123', 'La noticia', array('title' => 'La mejor noticia!'));
// Imprime: <a href="http://ejemplo.com/index.php/news/local/123"
// title="La mejor noticia!">La noticia</a>

echo anchor('', 'Clic aquí');
// Imprime: <a href="http://ejemplo.com">Clic aquí</a>
```

anchor_popup()

string anchor_popup(string $uri = '', string $title = '', mixed $attributes = FALSE)

Descripción

Casi idéntica a la función **anchor()** excepto que abre la URL en una nueva ventana, puede especificar atributos de ventana de JavaScript en el tercer parámetro para controlar cómo se abre la ventana, si el tercer parámetro no está configurado, simplemente abrirá una nueva ventana con la configuración del navegador

Parámetros

uri
: String URI

title
: Título de anclaje

attributes
: Atributos de HTML

Valores devueltos

Hipervínculo emergente

Ejemplo

```
$atts = array(
  'width'       => 800,
  'height'      => 600,
  'scrollbars'  => 'yes',
  'status'      => 'yes',
  'resizable'   => 'yes',
  'screenx'     => 0,
  'screeny'     => 0,
  'window_name' => '_blank'
);

echo anchor_popup('news/local/123', 'Click aquí!', $atts);
```

Nota

- Los atributos anteriores son los valores predeterminados de la función, por lo que solo necesita establecer los que son diferentes de lo que necesita. Si desea que la función use todos sus valores predeterminados simplemente pase un array vacío en el tercer parámetro:

```
echo anchor_popup('news/local/123', 'Click aquí!', array());
```

- **window_name** no es realmente un atributo, sino un argumento para el método JavaScript **window.open()** <http://www.w3schools.com/jsref/met_win_open.asp>, que acepta un nombre de ventana o un destino de ventana

- Cualquier otro atributo distinto de los enumerados anteriormente se analizará como un atributo HTML para la etiqueta de anclaje

auto_link()

string auto_link(string $str, string $type = 'both', bool $popup = FALSE)

Descripción

Convierte automáticamente las URL y las direcciones de correo electrónico contenidas en un string en enlaces

Parámetros

str
: String de entrada

type
: Tipo de enlace ('**email**', '**url**' o '**both**'), determina si las URL y los correos electrónicos se convierten o solo uno o el otro, si el parámetro no está especificado el comportamiento predeterminado es '**both**', los enlaces de correos electrónico están codificados como **safe_mailto()**

popup
: Si se deben crear enlaces emergentes, determina si los enlaces se muestran en una nueva ventana

Valores devueltos

String de enlace

Ejemplo

```
$string = auto_link($string);
```

Convierte solo URLs:

```
$string = auto_link($string, 'url');
```

Convierte solo direcciones de correo electrónico:

```
$string = auto_link($string, 'email');
```

El tercer parámetro determina si los enlaces se muestran en una nueva ventana :

```
$string = auto_link($string, 'both', TRUE);
```

base_url()

string base_url(string $uri = '', string $protocol = NULL)

Descripción

Devuelve la URL base de su sitio, como se especifica en su archivo de configuración, esta función devuelve lo mismo que **site_url()**, sin que se agregue **index_page** o **url_suffix**, también como en **site_url()**, puede suministrar segmentos como un string o un array

Esta función es un alias para **CI_Config::base_url()**, para obtener más información, consulte la **Config Library**

Parámetros

uri
String URI
protocol
Protocolo, p. ej. 'Http' o 'https'

Valores devueltos

URL base

Ejemplo

```
echo base_url();
```

Con string:

```
echo base_url("blog/post/123");
```

El ejemplo anterior devolvería algo así como: **http://ejemplo.com/blog/post/123**

Esto es útil porque a diferencia de **site_url()**, puede suministrar un string a un archivo, como una imagen o una hoja de estilo :

```
echo base_url("images/icons/edit.png");
```

Esto daría algo como: **http://ejemplo.com/images/icons/edit.png**

current_url()

string current_url()

Descripción

Devuelve la URL completa, incluidos los segmentos, de la página que se está viendo actualmente

Nota

Llamar a esta función es lo mismo que hacer esto: **site_url(uri_string());**

Valores devueltos

La URL actual

index_page()

mixed index_page()

Descripción

Devuelve su sitio **index_page**, como se especifica en su archivo de configuración

Valores devueltos

Valor de 'index_page'

Ejemplo

```
echo index_page();
```

mailto()

string mailto(string $email, string $title = '', mixed $attributes = '')

Descripción

Crea un enlace de correo electrónico HTML estándar "**mail to**"

Parámetros

email
Dirección de correo electrónico

title
Título de anclaje

attributes
Atributos de HTML

Valores devueltos

Un hipervínculo "**mail to**"

Ejemplo

```
echo mailto('me@my-site.com', 'Click Here to Contact Me');
```

Al igual que con **anchor()**, puede establecer atributos utilizando el tercer parámetro:

```
$attributes = array('title' => 'Mail me');
echo mailto('me@my-site.com', 'Contact Me', $attributes);
```

prep_url()

string prep_url(string $str = '')

Descripción

Esta función agregará **http://** en caso de que falte un prefijo de protocolo de una URL

Parámetros

str

String URL

Valores devueltos

String de URL con prefijo de protocolo

Ejemplo

Pase la string de URL a la función de esta forma:

```
$url = prep_url('example.com');
```

redirect()

void redirect(string $uri = '', string $method = 'auto', string $code = NULL)

Descripción

Hace una "redirección de encabezado" a la URI especificada, si especifica la URL completa del sitio, se construye ese enlace, pero para enlaces locales, simplemente proveer los segmentos URI al controlador que quiere direccionar para crear el enlace, la función construye la URL basándose en los valores del archivo de configuración

Parámetros

uri

String URI

method

Método de re direccionamiento ('auto', 'location' o 'refresh'), el valor predeterminado es 'auto', que intentará elegir de forma inteligente el método en función del entorno del servidor, 'location' location es más rápida pero menos confiable en los servidores IIS

code

Código de respuesta HTTP (generalmente 302 o 303)

valores devueltos

void

Ejemplo

```
if ($logged_in == FALSE)
{
   redirect('/login/form/');
}
```

```
// with 301 redirect
redirect('/article/13', 'location', 301);
```

Nota

- Para que esta función funcione, debe usarse antes de que se genere algo en el navegador, ya que utiliza los encabezados del servidor
- Para obtener un control más detallado de los encabezados, debe utilizar el método **set_header()** de la **Output Library**
- Para los usuarios de IIS: si oculta el encabezado HTTP del servidor, el método automático no detectará IIS; en ese caso, se recomienda utilizar explícitamente el método de actualización
- Cuando se usa el método de ubicación, se seleccionará automáticamente un código de estado HTTP de 303 cuando se acceda a la página por medio de POST y se use HTTP/1.1

Importante

Esta función terminará la ejecución del script

safe_mailto()

string safe_mailto(string $email, string $title = '', mixed $attributes = '')

Descripción

Idéntico a la función **mailto()**, excepto que escribe una versión ofuscada de la etiqueta mailto utilizando números ordinales escritos con JavaScript para evitar que los bots de correo basura capturen la dirección de correo electrónico

Parámetros

email
: Dirección de correo electrónico

title
: Título de anclaje

attributes
: Atributos de HTML

Valores devueltos

Un hipervínculo de "**mail to**" ofuscado

site_url()

string site_url([string $uri = '' [, string $protocol = NULL]])

Descripción

Devuelve la URL de su sitio, como se especifica en su archivo de configuración, el archivo **index.php** (o el que haya establecido como el **index_page** en su archivo de configuración) se agregará a la URL, al igual que cualquier segmento URI que pase a la función, más el **url_suffix** como se establece en su archivo de configuración

Se le recomienda utilizar esta función cuando necesite generar una URL local para que sus páginas se vuelvan más portables en caso de que su URL cambie, los segmentos se pueden pasar opcionalmente a la función como un string o un array, esta función es un alias para **CI_Config :: site_url ()**, para obtener más información, consulte la documentación de **Config Library**

Parámetros

uri
String URI

protocol
Protocolo, p. ej. 'Http' o 'https'

Valores devueltos

Sitio URL

Ejemplo

Con string:

```
echo site_url('news/local/123');

// Devuelve algo así como: http://ejemplo.com/index.php/news/local/123
```

Con array :

```
$segmentos = array('news', 'local', '123');
echo site_url($segmentos);
```

uri_string()

string uri_string()

Descripción

Devuelve los segmentos de URI de cualquier página que contenga esta función, esta función es un alias para **CI_Config::uri_string()**, para obtener más información, consulte la documentación de la **Config Library**

Valores devueltos

String URI

Ejemplo

Si la URL fue esta:

```
http://ejemplo.com/blog/opiniones/123
```

La función devolvería:

```
blog/opiniones/123
```

url_title()

string url_title(string $str, string $separator = '-', bool $lowercase = FALSE)

Descripción

Toma un string como entrada y crea un string URL amigable para los humanos, es útil, por ejemplo, si tiene un blog en el que le gustaría usar el título de sus entradas en la URL

Parámetros

str

String de entrada

separator

Separador de palabras

lowercase

Si se debe transformar el string de salida a minúsculas

Valores devueltos

String con formato de URL

Ejemplo

```
$título = "Más vale prevenir que curar";
$url_título = url_title($título);

// Produce: Más-vale-prevenir-que-curar

$título = "Más/vale/prevenir/?%que#curar";
$url_título = url_title($título);

// Produce: Másvaleprevenirquecurar
```

El segundo parámetro determina el delimitador de palabra, por defecto, se usan guiones, las opciones preferidas son: - (guión) o _ (guión bajo):

```
$título = "Más vale prevenir que curar";
$url_título = url_title($título, 'underscore');

// Produce: Más_vale_prevenir_que_curar
```

Nota

El uso anterior de 'guion' y 'subrayado' como el segundo parámetro está **DESACONSEJADO**

El tercer parámetro determina si los caracteres minúsculos son forzados o no, por defecto no lo son:

```
$título = "Qué pasa con CSS?";
$url_title = url_title($título, 'underscore', TRUE);

// Produce: qué_pasa_con_css
```

XML Helper

XML Helper ayuda a trabajar con datos XML

Cargar XML Helper

XML Helper se carga así:

```
$this->load->helper('xml');
```

Funciones

xml_convert()

string xml_convert(string $str [, bool $protect_all = FALSE])

Descripción

Toma un string como entrada y convierte los siguientes caracteres XML en entidades

Ampersands:	**&**
Menos que y más que caracteres:	**<>**
Comillas simples y dobles:	**'"**
Guiones:	**-**

Esta función ignora los Ampersands si son parte de entidades de caracteres numerados existentes, e.g. e.g. {

Parámetros

str

La string de texto para convertir

protect_all

Si se debe proteger todo el contenido que se parece a una entidad potencial en lugar de solo las entidades numeradas, p. &foo;

Valores devueltos

String convertida XML

Ejemplo

```
$string = '<p>Aquí hay un párrafo & una entidad (&#123;).</p>';
$string = xml_convert($string);
echo $string;
```

Produce:

```
&lt;p&gt;Aquí hay un párrafo &amp; una entidad ({).&lt;/p&gt;
```