* [Welcome to CodeIgniter](http://docs.google.com/general/welcome.html)
* [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Downloading CodeIgniter](http://docs.google.com/installation/downloads.html)
  + [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Upgrading From a Previous Version](http://docs.google.com/installation/upgrading.html)
  + [Troubleshooting](http://docs.google.com/installation/troubleshooting.html)
* [CodeIgniter Overview](http://docs.google.com/overview/index.html)
  + [Getting Started](http://docs.google.com/overview/getting_started.html)
  + [CodeIgniter at a Glance](http://docs.google.com/overview/at_a_glance.html)
  + [Supported Features](http://docs.google.com/overview/features.html)
  + [Application Flow Chart](http://docs.google.com/overview/appflow.html)
  + [Model-View-Controller](http://docs.google.com/overview/mvc.html)
  + [Architectural Goals](http://docs.google.com/overview/goals.html)
* [Tutorial](http://docs.google.com/tutorial/index.html)
  + [Static pages](http://docs.google.com/tutorial/static_pages.html)
  + [News section](http://docs.google.com/tutorial/news_section.html)
  + [Create news items](http://docs.google.com/tutorial/create_news_items.html)
  + [Conclusion](http://docs.google.com/tutorial/conclusion.html)
* [Contributing to CodeIgniter](http://docs.google.com/contributing/index.html)
  + [Writing CodeIgniter Documentation](http://docs.google.com/documentation/index.html)
  + [Developer’s Certificate of Origin 1.1](http://docs.google.com/DCO.html)
* [General Topics](http://docs.google.com/general/index.html)
  + [CodeIgniter URLs](http://docs.google.com/general/urls.html)
  + [Controllers](http://docs.google.com/general/controllers.html)
  + [Reserved Names](http://docs.google.com/general/reserved_names.html)
  + [Views](http://docs.google.com/general/views.html)
  + [Models](http://docs.google.com/general/models.html)
  + [Helpers](http://docs.google.com/general/helpers.html)
  + [Using CodeIgniter Libraries](http://docs.google.com/general/libraries.html)
  + [Creating Libraries](http://docs.google.com/general/creating_libraries.html)
  + [Using CodeIgniter Drivers](http://docs.google.com/general/drivers.html)
  + [Creating Drivers](http://docs.google.com/general/creating_drivers.html)
  + [Creating Core System Classes](http://docs.google.com/general/core_classes.html)
  + [Creating Ancillary Classes](http://docs.google.com/general/ancillary_classes.html)
  + [Hooks - Extending the Framework Core](http://docs.google.com/general/hooks.html)
  + [Auto-loading Resources](http://docs.google.com/general/autoloader.html)
  + [Common Functions](http://docs.google.com/general/common_functions.html)
  + [Compatibility Functions](http://docs.google.com/general/compatibility_functions.html)
  + [URI Routing](http://docs.google.com/general/routing.html)
  + [Error Handling](http://docs.google.com/general/errors.html)
  + [Caching](http://docs.google.com/general/caching.html)
  + [Profiling Your Application](http://docs.google.com/general/profiling.html)
  + [Running via the CLI](http://docs.google.com/general/cli.html)
  + [Managing your Applications](http://docs.google.com/general/managing_apps.html)
  + [Handling Multiple Environments](http://docs.google.com/general/environments.html)
  + [Alternate PHP Syntax for View Files](http://docs.google.com/general/alternative_php.html)
  + [Security](http://docs.google.com/general/security.html)
  + [PHP Style Guide](http://docs.google.com/general/styleguide.html)
* [Libraries](http://docs.google.com/index.html)
  + [Benchmarking Class](http://docs.google.com/benchmark.html)
  + [Caching Driver](http://docs.google.com/caching.html)
  + [Calendaring Class](http://docs.google.com/calendar.html)
  + [Shopping Cart Class](http://docs.google.com/cart.html)
  + [Config Class](http://docs.google.com/config.html)
  + [Email Class](http://docs.google.com/email.html)
  + [Encrypt Class](http://docs.google.com/encrypt.html)
  + [Encryption Library](http://docs.google.com/encryption.html)
  + [File Uploading Class](http://docs.google.com/file_uploading.html)
  + [Form Validation](http://docs.google.com/form_validation.html)
  + [FTP Class](http://docs.google.com/ftp.html)
  + [Image Manipulation Class](http://docs.google.com/image_lib.html)
  + [Input Class](http://docs.google.com/input.html)
  + [Javascript Class](http://docs.google.com/javascript.html)
  + [Language Class](http://docs.google.com/language.html)
  + Loader Class
  + [Migrations Class](http://docs.google.com/migration.html)
  + [Output Class](http://docs.google.com/output.html)
  + [Pagination Class](http://docs.google.com/pagination.html)
  + [Template Parser Class](http://docs.google.com/parser.html)
  + [Security Class](http://docs.google.com/security.html)
  + [Session Library](http://docs.google.com/sessions.html)
  + [HTML Table Class](http://docs.google.com/table.html)
  + [Trackback Class](http://docs.google.com/trackback.html)
  + [Typography Class](http://docs.google.com/typography.html)
  + [Unit Testing Class](http://docs.google.com/unit_testing.html)
  + [URI Class](http://docs.google.com/uri.html)
  + [User Agent Class](http://docs.google.com/user_agent.html)
  + [XML-RPC and XML-RPC Server Classes](http://docs.google.com/xmlrpc.html)
  + [Zip Encoding Class](http://docs.google.com/zip.html)
* [Database Reference](http://docs.google.com/database/index.html)
  + [Quick Start: Usage Examples](http://docs.google.com/database/examples.html)
  + [Database Configuration](http://docs.google.com/database/configuration.html)
  + [Connecting to a Database](http://docs.google.com/database/connecting.html)
  + [Running Queries](http://docs.google.com/database/queries.html)
  + [Generating Query Results](http://docs.google.com/database/results.html)
  + [Query Helper Functions](http://docs.google.com/database/helpers.html)
  + [Query Builder Class](http://docs.google.com/database/query_builder.html)
  + [Transactions](http://docs.google.com/database/transactions.html)
  + [Getting MetaData](http://docs.google.com/database/metadata.html)
  + [Custom Function Calls](http://docs.google.com/database/call_function.html)
  + [Query Caching](http://docs.google.com/database/caching.html)
  + [Database Manipulation with Database Forge](http://docs.google.com/database/forge.html)
  + [Database Utilities Class](http://docs.google.com/database/utilities.html)
  + [Database Driver Reference](http://docs.google.com/database/db_driver_reference.html)
* [Helpers](http://docs.google.com/helpers/index.html)
  + [Array Helper](http://docs.google.com/helpers/array_helper.html)
  + [CAPTCHA Helper](http://docs.google.com/helpers/captcha_helper.html)
  + [Cookie Helper](http://docs.google.com/helpers/cookie_helper.html)
  + [Date Helper](http://docs.google.com/helpers/date_helper.html)
  + [Directory Helper](http://docs.google.com/helpers/directory_helper.html)
  + [Download Helper](http://docs.google.com/helpers/download_helper.html)
  + [Email Helper](http://docs.google.com/helpers/email_helper.html)
  + [File Helper](http://docs.google.com/helpers/file_helper.html)
  + [Form Helper](http://docs.google.com/helpers/form_helper.html)
  + [HTML Helper](http://docs.google.com/helpers/html_helper.html)
  + [Inflector Helper](http://docs.google.com/helpers/inflector_helper.html)
  + [Language Helper](http://docs.google.com/helpers/language_helper.html)
  + [Number Helper](http://docs.google.com/helpers/number_helper.html)
  + [Path Helper](http://docs.google.com/helpers/path_helper.html)
  + [Security Helper](http://docs.google.com/helpers/security_helper.html)
  + [Smiley Helper](http://docs.google.com/helpers/smiley_helper.html)
  + [String Helper](http://docs.google.com/helpers/string_helper.html)
  + [Text Helper](http://docs.google.com/helpers/text_helper.html)
  + [Typography Helper](http://docs.google.com/helpers/typography_helper.html)
  + [URL Helper](http://docs.google.com/helpers/url_helper.html)
  + [XML Helper](http://docs.google.com/helpers/xml_helper.html)

Toggle Table of Contents

[CodeIgniter](http://docs.google.com/index.html)

* [Welcome to CodeIgniter](http://docs.google.com/general/welcome.html)
* [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Downloading CodeIgniter](http://docs.google.com/installation/downloads.html)
  + [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Upgrading From a Previous Version](http://docs.google.com/installation/upgrading.html)
  + [Troubleshooting](http://docs.google.com/installation/troubleshooting.html)
* [CodeIgniter Overview](http://docs.google.com/overview/index.html)
  + [Getting Started](http://docs.google.com/overview/getting_started.html)
  + [CodeIgniter at a Glance](http://docs.google.com/overview/at_a_glance.html)
  + [Supported Features](http://docs.google.com/overview/features.html)
  + [Application Flow Chart](http://docs.google.com/overview/appflow.html)
  + [Model-View-Controller](http://docs.google.com/overview/mvc.html)
  + [Architectural Goals](http://docs.google.com/overview/goals.html)
* [Tutorial](http://docs.google.com/tutorial/index.html)
  + [Static pages](http://docs.google.com/tutorial/static_pages.html)
  + [News section](http://docs.google.com/tutorial/news_section.html)
  + [Create news items](http://docs.google.com/tutorial/create_news_items.html)
  + [Conclusion](http://docs.google.com/tutorial/conclusion.html)
* [Contributing to CodeIgniter](http://docs.google.com/contributing/index.html)
  + [Writing CodeIgniter Documentation](http://docs.google.com/documentation/index.html)
  + [Developer’s Certificate of Origin 1.1](http://docs.google.com/DCO.html)
* [General Topics](http://docs.google.com/general/index.html)
  + [CodeIgniter URLs](http://docs.google.com/general/urls.html)
  + [Controllers](http://docs.google.com/general/controllers.html)
  + [Reserved Names](http://docs.google.com/general/reserved_names.html)
  + [Views](http://docs.google.com/general/views.html)
  + [Models](http://docs.google.com/general/models.html)
  + [Helpers](http://docs.google.com/general/helpers.html)
  + [Using CodeIgniter Libraries](http://docs.google.com/general/libraries.html)
  + [Creating Libraries](http://docs.google.com/general/creating_libraries.html)
  + [Using CodeIgniter Drivers](http://docs.google.com/general/drivers.html)
  + [Creating Drivers](http://docs.google.com/general/creating_drivers.html)
  + [Creating Core System Classes](http://docs.google.com/general/core_classes.html)
  + [Creating Ancillary Classes](http://docs.google.com/general/ancillary_classes.html)
  + [Hooks - Extending the Framework Core](http://docs.google.com/general/hooks.html)
  + [Auto-loading Resources](http://docs.google.com/general/autoloader.html)
  + [Common Functions](http://docs.google.com/general/common_functions.html)
  + [Compatibility Functions](http://docs.google.com/general/compatibility_functions.html)
  + [URI Routing](http://docs.google.com/general/routing.html)
  + [Error Handling](http://docs.google.com/general/errors.html)
  + [Caching](http://docs.google.com/general/caching.html)
  + [Profiling Your Application](http://docs.google.com/general/profiling.html)
  + [Running via the CLI](http://docs.google.com/general/cli.html)
  + [Managing your Applications](http://docs.google.com/general/managing_apps.html)
  + [Handling Multiple Environments](http://docs.google.com/general/environments.html)
  + [Alternate PHP Syntax for View Files](http://docs.google.com/general/alternative_php.html)
  + [Security](http://docs.google.com/general/security.html)
  + [PHP Style Guide](http://docs.google.com/general/styleguide.html)
* [Libraries](http://docs.google.com/index.html)
  + [Benchmarking Class](http://docs.google.com/benchmark.html)
  + [Caching Driver](http://docs.google.com/caching.html)
  + [Calendaring Class](http://docs.google.com/calendar.html)
  + [Shopping Cart Class](http://docs.google.com/cart.html)
  + [Config Class](http://docs.google.com/config.html)
  + [Email Class](http://docs.google.com/email.html)
  + [Encrypt Class](http://docs.google.com/encrypt.html)
  + [Encryption Library](http://docs.google.com/encryption.html)
  + [File Uploading Class](http://docs.google.com/file_uploading.html)
  + [Form Validation](http://docs.google.com/form_validation.html)
  + [FTP Class](http://docs.google.com/ftp.html)
  + [Image Manipulation Class](http://docs.google.com/image_lib.html)
  + [Input Class](http://docs.google.com/input.html)
  + [Javascript Class](http://docs.google.com/javascript.html)
  + [Language Class](http://docs.google.com/language.html)
  + Loader Class
  + [Migrations Class](http://docs.google.com/migration.html)
  + [Output Class](http://docs.google.com/output.html)
  + [Pagination Class](http://docs.google.com/pagination.html)
  + [Template Parser Class](http://docs.google.com/parser.html)
  + [Security Class](http://docs.google.com/security.html)
  + [Session Library](http://docs.google.com/sessions.html)
  + [HTML Table Class](http://docs.google.com/table.html)
  + [Trackback Class](http://docs.google.com/trackback.html)
  + [Typography Class](http://docs.google.com/typography.html)
  + [Unit Testing Class](http://docs.google.com/unit_testing.html)
  + [URI Class](http://docs.google.com/uri.html)
  + [User Agent Class](http://docs.google.com/user_agent.html)
  + [XML-RPC and XML-RPC Server Classes](http://docs.google.com/xmlrpc.html)
  + [Zip Encoding Class](http://docs.google.com/zip.html)
* [Database Reference](http://docs.google.com/database/index.html)
  + [Quick Start: Usage Examples](http://docs.google.com/database/examples.html)
  + [Database Configuration](http://docs.google.com/database/configuration.html)
  + [Connecting to a Database](http://docs.google.com/database/connecting.html)
  + [Running Queries](http://docs.google.com/database/queries.html)
  + [Generating Query Results](http://docs.google.com/database/results.html)
  + [Query Helper Functions](http://docs.google.com/database/helpers.html)
  + [Query Builder Class](http://docs.google.com/database/query_builder.html)
  + [Transactions](http://docs.google.com/database/transactions.html)
  + [Getting MetaData](http://docs.google.com/database/metadata.html)
  + [Custom Function Calls](http://docs.google.com/database/call_function.html)
  + [Query Caching](http://docs.google.com/database/caching.html)
  + [Database Manipulation with Database Forge](http://docs.google.com/database/forge.html)
  + [Database Utilities Class](http://docs.google.com/database/utilities.html)
  + [Database Driver Reference](http://docs.google.com/database/db_driver_reference.html)
* [Helpers](http://docs.google.com/helpers/index.html)
  + [Array Helper](http://docs.google.com/helpers/array_helper.html)
  + [CAPTCHA Helper](http://docs.google.com/helpers/captcha_helper.html)
  + [Cookie Helper](http://docs.google.com/helpers/cookie_helper.html)
  + [Date Helper](http://docs.google.com/helpers/date_helper.html)
  + [Directory Helper](http://docs.google.com/helpers/directory_helper.html)
  + [Download Helper](http://docs.google.com/helpers/download_helper.html)
  + [Email Helper](http://docs.google.com/helpers/email_helper.html)
  + [File Helper](http://docs.google.com/helpers/file_helper.html)
  + [Form Helper](http://docs.google.com/helpers/form_helper.html)
  + [HTML Helper](http://docs.google.com/helpers/html_helper.html)
  + [Inflector Helper](http://docs.google.com/helpers/inflector_helper.html)
  + [Language Helper](http://docs.google.com/helpers/language_helper.html)
  + [Number Helper](http://docs.google.com/helpers/number_helper.html)
  + [Path Helper](http://docs.google.com/helpers/path_helper.html)
  + [Security Helper](http://docs.google.com/helpers/security_helper.html)
  + [Smiley Helper](http://docs.google.com/helpers/smiley_helper.html)
  + [String Helper](http://docs.google.com/helpers/string_helper.html)
  + [Text Helper](http://docs.google.com/helpers/text_helper.html)
  + [Typography Helper](http://docs.google.com/helpers/typography_helper.html)
  + [URL Helper](http://docs.google.com/helpers/url_helper.html)
  + [XML Helper](http://docs.google.com/helpers/xml_helper.html)

[CodeIgniter](http://docs.google.com/index.html)

* [Docs](http://docs.google.com/index.html) »
* [Libraries](http://docs.google.com/index.html) »
* Loader Class

classic layout

Loader Class[**¶**](#gjdgxs)

Loader, as the name suggests, is used to load elements. These elements can be libraries (classes) [*View files*](http://docs.google.com/general/views.html), [*Drivers*](http://docs.google.com/general/drivers.html), [*Helpers*](http://docs.google.com/general/helpers.html), [*Models*](http://docs.google.com/general/models.html), or your own files.

Note

This class is initialized automatically by the system so there is no need to do it manually.

* [Application “Packages”](#30j0zll)
  + [Package view files](#1fob9te)
* [Class Reference](#3znysh7)

[**Application “Packages”**](#2et92p0)[**¶**](#30j0zll)

An application package allows for the easy distribution of complete sets of resources in a single directory, complete with its own libraries, models, helpers, config, and language files. It is recommended that these packages be placed in the application/third\_party directory. Below is a sample map of an package directory.

The following is an example of a directory for an application package named “Foo Bar”.

/application/third\_party/foo\_bar  
  
config/  
helpers/  
language/  
libraries/  
models/

Whatever the purpose of the “Foo Bar” application package, it has its own config files, helpers, language files, libraries, and models. To use these resources in your controllers, you first need to tell the Loader that you are going to be loading resources from a package, by adding the package path via the add\_package\_path() method.

[**Package view files**](#tyjcwt)[**¶**](#1fob9te)

By Default, package view files paths are set when add\_package\_path() is called. View paths are looped through, and once a match is encountered that view is loaded.

In this instance, it is possible for view naming collisions within packages to occur, and possibly the incorrect package being loaded. To ensure against this, set an optional second parameter of FALSE when calling add\_package\_path().

$this->load->add\_package\_path(APPPATH.'my\_app', FALSE);  
$this->load->view('my\_app\_index'); // Loads  
$this->load->view('welcome\_message'); // Will not load the default welcome\_message b/c the second param to add\_package\_path is FALSE  
  
// Reset things  
$this->load->remove\_package\_path(APPPATH.'my\_app');  
  
// Again without the second parameter:  
$this->load->add\_package\_path(APPPATH.'my\_app');  
$this->load->view('my\_app\_index'); // Loads  
$this->load->view('welcome\_message'); // Loads

[**Class Reference**](#3dy6vkm)[**¶**](#3znysh7)

*class* CI\_Loader[¶](#1t3h5sf) library(*$library*[, *$params = NULL*[, *$object\_name = NULL*]])[¶](#4d34og8)

| Parameters: | * **$library** (*mixed*) – Library name as a string or an array with multiple libraries * **$params** (*array*) – Optional array of parameters to pass to the loaded library’s constructor * **$object\_name** (*string*) – Optional object name to assign the library to |
| --- | --- |
| Returns: | CI\_Loader instance (method chaining) |
| Return type: | CI\_Loader |

This method is used to load core classes.

Note

We use the terms “class” and “library” interchangeably.

For example, if you would like to send email with CodeIgniter, the first step is to load the email class within your controller:

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

Once loaded, the library will be ready for use, using $this->email.

Library files can be stored in subdirectories within the main “libraries” directory, or within your personal *application/libraries* directory. To load a file located in a subdirectory, simply include the path, relative to the “libraries” directory. For example, if you have file located at:

libraries/flavors/Chocolate.php

You will load it using:

$this->load->library('flavors/chocolate');

You may nest the file in as many subdirectories as you want.

Additionally, multiple libraries can be loaded at the same time by passing an array of libraries to the load method.

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

**Setting options**

The second (optional) parameter allows you to optionally pass configuration setting. You will typically pass these as an array:

$config = array (  
 'mailtype' => 'html',  
 'charset' => 'utf-8,  
 'priority' => '1'  
);  
  
$this->load->library('email', $config);

Config options can usually also be set via a config file. Each library is explained in detail in its own page, so please read the information regarding each one you would like to use.

Please take note, when multiple libraries are supplied in an array for the first parameter, each will receive the same parameter information.

**Assigning a Library to a different object name**

If the third (optional) parameter is blank, the library will usually be assigned to an object with the same name as the library. For example, if the library is named Calendar, it will be assigned to a variable named $this->calendar.

If you prefer to set your own class names you can pass its value to the third parameter:

$this->load->library('calendar', NULL, 'my\_calendar');  
  
// Calendar class is now accessed using:  
$this->my\_calendar

Please take note, when multiple libraries are supplied in an array for the first parameter, this parameter is discarded.

driver(*$library*[, *$params = NULL*[, *$object\_name*]])[¶](#2s8eyo1)

| Parameters: | * **$library** (*mixed*) – Library name as a string or an array with multiple libraries * **$params** (*array*) – Optional array of parameters to pass to the loaded library’s constructor * **$object\_name** (*string*) – Optional object name to assign the library to |
| --- | --- |
| Returns: | CI\_Loader instance (method chaining) |
| Return type: | CI\_Loader |

This method is used to load driver libraries, acts very much like the library() method.

As an example, if you would like to use sessions with CodeIgniter, the first step is to load the session driver within your controller:

$this->load->driver('session');

Once loaded, the library will be ready for use, using $this->session.

Driver files must be stored in a subdirectory within the main “libraries” directory, or within your personal *application/libraries* directory. The subdirectory must match the parent class name. Read the [*Drivers*](http://docs.google.com/general/drivers.html) description for details.

Additionally, multiple driver libraries can be loaded at the same time by passing an array of drivers to the load method.

$this->load->driver(array('session', 'cache'));

**Setting options**

The second (optional) parameter allows you to optionally pass configuration settings. You will typically pass these as an array:

$config = array(  
 'sess\_driver' => 'cookie',  
 'sess\_encrypt\_cookie' => true,  
 'encryption\_key' => 'mysecretkey'  
);  
  
$this->load->driver('session', $config);

Config options can usually also be set via a config file. Each library is explained in detail in its own page, so please read the information regarding each one you would like to use.

**Assigning a Driver to a different object name**

If the third (optional) parameter is blank, the library will be assigned to an object with the same name as the parent class. For example, if the library is named Session, it will be assigned to a variable named $this->session.

If you prefer to set your own class names you can pass its value to the third parameter:

$this->load->library('session', '', 'my\_session');  
  
// Session class is now accessed using:  
$this->my\_session

view(*$view*[, *$vars = array()*[, *return = FALSE*]])[¶](#17dp8vu)

| Parameters: | * **$view** (*string*) – View name * **$vars** (*array*) – An associative array of variables * **$return** (*bool*) – Whether to return the loaded view |
| --- | --- |
| Returns: | View content string if $return is set to TRUE, otherwise CI\_Loader instance (method chaining) |
| Return type: | mixed |

This method is used to load your View files. If you haven’t read the [*Views*](http://docs.google.com/general/views.html) section of the user guide it is recommended that you do since it shows you how this method is typically used.

The first parameter is required. It is the name of the view file you would like to load.

Note

The .php file extension does not need to be specified unless you use something other than .php.

The second **optional** parameter can take an associative array or an object as input, which it runs through the PHP [extract()](http://php.net/extract) function to convert to variables that can be used in your view files. Again, read the [*Views*](http://docs.google.com/general/views.html) page to learn how this might be useful.

The third **optional** parameter lets you change the behavior of the method so that it returns data as a string rather than sending it to your browser. This can be useful if you want to process the data in some way. If you set the parameter to TRUE (boolean) it will return data. The default behavior is FALSE, which sends it to your browser. Remember to assign it to a variable if you want the data returned:

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

vars(*$vars*[, *$val = ''*])[¶](#3rdcrjn)

| Parameters: | * **$vars** (*mixed*) – An array of variables or a single variable name * **$val** (*mixed*) – Optional variable value |
| --- | --- |
| Returns: | CI\_Loader instance (method chaining) |
| Return type: | CI\_Loader |

This method takes an associative array as input and generates variables using the PHP [extract()](http://php.net/extract) function. This method produces the same result as using the second parameter of the $this->load->view() method above. The reason you might want to use this method independently is if you would like to set some global variables in the constructor of your controller and have them become available in any view file loaded from any method. You can have multiple calls to this method. The data get cached and merged into one array for conversion to variables.

get\_var(*$key*)[¶](#26in1rg)

| Parameters: | * **$key** (*string*) – Variable name key |
| --- | --- |
| Returns: | Value if key is found, NULL if not |
| Return type: | mixed |

This method checks the associative array of variables available to your views. This is useful if for any reason a var is set in a library or another controller method using $this->load->vars().

get\_vars()[¶](#lnxbz9)

| Returns: | An array of all assigned view variables |
| --- | --- |
| Return type: | array |

This method retrieves all variables available to your views.

clear\_vars()[¶](#35nkun2)

| Returns: | CI\_Loader instance (method chaining) |
| --- | --- |
| Return type: | CI\_Loader |

Clears cached view variables.

model(*$model*[, *$name = ''*[, *$db\_conn = FALSE*]])[¶](#1ksv4uv)

| Parameters: | * **$model** (*mixed*) – Model name or an array containing multiple models * **$name** (*string*) – Optional object name to assign the model to * **$db\_conn** (*string*) – Optional database configuration group to load |
| --- | --- |
| Returns: | CI\_Loader instance (method chaining) |
| Return type: | CI\_Loader |

$this->load->model('model\_name');

If your model is located in a subdirectory, include the relative path from your models directory. For example, if you have a model located at *application/models/blog/Queries.php* you’ll load it using:

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

If you would like your model assigned to a different object name you can specify it via the second parameter of the loading method:

$this->load->model('model\_name', 'fubar');  
$this->fubar->method();

database([*$params = ''*[, *$return = FALSE*[, *$query\_builder = NULL*]]])[¶](#44sinio)

| Parameters: | * **$params** (*mixed*) – Database group name or configuration options * **$return** (*bool*) – Whether to return the loaded database object * **$query\_builder** (*bool*) – Whether to load the Query Builder |
| --- | --- |
| Returns: | Loaded CI\_DB instance or FALSE on failure if $return is set to TRUE, otherwise CI\_Loader instance (method chaining) |
| Return type: | mixed |

This method lets you load the database class. The two parameters are **optional**. Please see the [*database*](http://docs.google.com/database/index.html) section for more info.

dbforge([*$db = NULL*[, *$return = FALSE*]])[¶](#2jxsxqh)

| Parameters: | * **$db** (*object*) – Database object * **$return** (*bool*) – Whether to return the Database Forge instance |
| --- | --- |
| Returns: | Loaded CI\_DB\_forge instance if $return is set to TRUE, otherwise CI\_Loader instance (method chaining) |
| Return type: | mixed |

Loads the [*Database Forge*](http://docs.google.com/database/forge.html) class, please refer to that manual for more info.

dbutil([*$db = NULL*[, *$return = FALSE*]])[¶](#z337ya)

| Parameters: | * **$db** (*object*) – Database object * **$return** (*bool*) – Whether to return the Database Utilities instance |
| --- | --- |
| Returns: | Loaded CI\_DB\_utility instance if $return is set to TRUE, otherwise CI\_Loader instance (method chaining) |
| Return type: | mixed |

Loads the [*Database Utilities*](http://docs.google.com/database/utilities.html) class, please refer to that manual for more info.

helper(*$helpers*)[¶](#3j2qqm3)

| Parameters: | * **$helpers** (*mixed*) – Helper name as a string or an array containing multiple helpers |
| --- | --- |
| Returns: | CI\_Loader instance (method chaining) |
| Return type: | CI\_Loader |

This method loads helper files, where file\_name is the name of the file, without the \_helper.php extension.

file(*$path*[, *$return = FALSE*])[¶](#1y810tw)

| Parameters: | * **$path** (*string*) – File path * **$return** (*bool*) – Whether to return the loaded file |
| --- | --- |
| Returns: | File contents if $return is set to TRUE, otherwise CI\_Loader instance (method chaining) |
| Return type: | mixed |

This is a generic file loading method. Supply the filepath and name in the first parameter and it will open and read the file. By default the data is sent to your browser, just like a View file, but if you set the second parameter to boolean TRUE it will instead return the data as a string.

language(*$files*[, *$lang = ''*])[¶](#4i7ojhp)

| Parameters: | * **$files** (*mixed*) – Language file name or an array of multiple language files * **$lang** (*string*) – Language name |
| --- | --- |
| Returns: | CI\_Loader instance (method chaining) |
| Return type: | CI\_Loader |

This method is an alias of the [*language loading method*](http://docs.google.com/language.html): $this->lang->load().

config(*$file*[, *$use\_sections = FALSE*[, *$fail\_gracefully = FALSE*]])[¶](#2xcytpi)

| Parameters: | * **$file** (*string*) – Configuration file name * **$use\_sections** (*bool*) – Whether configuration values should be loaded into their own section * **$fail\_gracefully** (*bool*) – Whether to just return FALSE in case of failure |
| --- | --- |
| Returns: | TRUE on success, FALSE on failure |
| Return type: | bool |

This method is an alias of the [*config file loading method*](http://docs.google.com/config.html): $this->config->load()

is\_loaded(*$class*)[¶](#1ci93xb)

| Parameters: | * **$class** (*string*) – Class name |
| --- | --- |
| Returns: | Singleton property name if found, FALSE if not |
| Return type: | mixed |

Allows you to check if a class has already been loaded or not.

Note

The word “class” here refers to libraries and drivers.

If the requested class has been loaded, the method returns its assigned name in the CI Super-object and FALSE if it’s not:

$this->load->library('form\_validation');  
$this->load->is\_loaded('Form\_validation'); // returns 'form\_validation'  
  
$this->load->is\_loaded('Nonexistent\_library'); // returns FALSE

Important

If you have more than one instance of a class (assigned to different properties), then the first one will be returned.

$this->load->library('form\_validation', $config, 'fv');  
$this->load->library('form\_validation');  
  
$this->load->is\_loaded('Form\_validation'); // returns 'fv'

add\_package\_path(*$path*[, *$view\_cascade = TRUE*])[¶](#3whwml4)

| Parameters: | * **$path** (*string*) – Path to add * **$view\_cascade** (*bool*) – Whether to use cascading views |
| --- | --- |
| Returns: | CI\_Loader instance (method chaining) |
| Return type: | CI\_Loader |

Adding a package path instructs the Loader class to prepend a given path for subsequent requests for resources. As an example, the “Foo Bar” application package above has a library named Foo\_bar.php. In our controller, we’d do the following:

$this->load->add\_package\_path(APPPATH.'third\_party/foo\_bar/')  
 ->library('foo\_bar');

remove\_package\_path([*$path = ''*])[¶](#2bn6wsx)

| Parameters: | * **$path** (*string*) – Path to remove |
| --- | --- |
| Returns: | CI\_Loader instance (method chaining) |
| Return type: | CI\_Loader |

When your controller is finished using resources from an application package, and particularly if you have other application packages you want to work with, you may wish to remove the package path so the Loader no longer looks in that directory for resources. To remove the last path added, simply call the method with no parameters.

Or to remove a specific package path, specify the same path previously given to add\_package\_path() for a package.:

$this->load->remove\_package\_path(APPPATH.'third\_party/foo\_bar/');

get\_package\_paths([*$include\_base = TRUE*])[¶](#qsh70q)

| Parameters: | * **$include\_base** (*bool*) – Whether to include BASEPATH |
| --- | --- |
| Returns: | An array of package paths |
| Return type: | array |

Returns all currently available package paths.

[Next](http://docs.google.com/migration.html)   [Previous](http://docs.google.com/language.html)

© Copyright 2014 - 2016, British Columbia Institute of Technology. Last updated on Oct 28, 2016.

Built with [Sphinx](http://sphinx-doc.org/) using a [theme](https://github.com/snide/sphinx_rtd_theme) provided by [Read the Docs](https://readthedocs.org).