* [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
  + [Javascript Class](http://docs.google.com/javascript.html)
  + [Language Class](http://docs.google.com/language.html)
  + [Loader Class](http://docs.google.com/loader.html)
  + [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
  + [Javascript Class](http://docs.google.com/javascript.html)
  + [Language Class](http://docs.google.com/language.html)
  + [Loader Class](http://docs.google.com/loader.html)
  + [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) »
* Input Class

classic layout

Input Class[**¶**](#gjdgxs)

The Input Class serves two purposes:

1. It pre-processes global input data for security.
2. It provides some helper methods for fetching input data and pre-processing it.

Note

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

* [Input Filtering](#30j0zll)
  + [Security Filtering](#1fob9te)
  + [XSS Filtering](#3znysh7)
* [Accessing form data](#2et92p0)
  + [Using POST, GET, COOKIE, or SERVER Data](#tyjcwt)
  + [Using the php://input stream](#3dy6vkm)
* [Class Reference](#1t3h5sf)

[**Input Filtering**](#4d34og8)[**¶**](#30j0zll)

[**Security Filtering**](#2s8eyo1)[**¶**](#1fob9te)

The security filtering method is called automatically when a new [*controller*](http://docs.google.com/general/controllers.html) is invoked. It does the following:

* If $config['allow\_get\_array'] is FALSE (default is TRUE), destroys the global GET array.
* Destroys all global variables in the event register\_globals is turned on.
* Filters the GET/POST/COOKIE array keys, permitting only alpha-numeric (and a few other) characters.
* Provides XSS (Cross-site Scripting Hacks) filtering. This can be enabled globally, or upon request.
* Standardizes newline characters to PHP\_EOL (\n in UNIX-based OSes, \r\n under Windows). This is configurable.

[**XSS Filtering**](#17dp8vu)[**¶**](#3znysh7)

The Input class has the ability to filter input automatically to prevent cross-site scripting attacks. If you want the filter to run automatically every time it encounters POST or COOKIE data you can enable it by opening your *application/config/config.php* file and setting this:

$config['global\_xss\_filtering'] = TRUE;

Please refer to the [*Security class*](http://docs.google.com/security.html) documentation for information on using XSS Filtering in your application.

Important

The ‘global\_xss\_filtering’ setting is DEPRECATED and kept solely for backwards-compatibility purposes. XSS escaping should be performed on *output*, not *input*!

[**Accessing form data**](#3rdcrjn)[**¶**](#2et92p0)

[**Using POST, GET, COOKIE, or SERVER Data**](#26in1rg)[**¶**](#tyjcwt)

CodeIgniter comes with helper methods that let you fetch POST, GET, COOKIE or SERVER items. The main advantage of using the provided methods rather than fetching an item directly ($\_POST['something']) is that the methods will check to see if the item is set and return NULL if not. This lets you conveniently use data without having to test whether an item exists first. In other words, normally you might do something like this:

$something = isset($\_POST['something']) ? $\_POST['something'] : NULL;

With CodeIgniter’s built in methods you can simply do this:

$something = $this->input->post('something');

The main methods are:

* $this->input->post()
* $this->input->get()
* $this->input->cookie()
* $this->input->server()

[**Using the php://input stream**](#lnxbz9)[**¶**](#3dy6vkm)

If you want to utilize the PUT, DELETE, PATCH or other exotic request methods, they can only be accessed via a special input stream, that can only be read once. This isn’t as easy as just reading from e.g. the $\_POST array, because it will always exist and you can try and access multiple variables without caring that you might only have one shot at all of the POST data.

CodeIgniter will take care of that for you, and you can read the data from the **php://input** stream at any time, just by using the $raw\_input\_stream property:

$this->input->raw\_input\_stream;

Additionally if the input stream is form-encoded like $\_POST you can access its values by calling the input\_stream() method:

$this->input->input\_stream('key');

Similar to other methods such as get() and post(), if the requested data is not found, it will return NULL and you can also decide whether to run the data through xss\_clean() by passing a boolean value as the second parameter:

$this->input->input\_stream('key', TRUE); // XSS Clean  
$this->input->input\_stream('key', FALSE); // No XSS filter

Note

You can utilize method() in order to know if you’re reading PUT, DELETE or PATCH data.

[**Class Reference**](#35nkun2)[**¶**](#1t3h5sf)

*class* CI\_Input[¶](#1ksv4uv) $raw\_input\_stream

Read only property that will return php://input data as is.

The property can be read multiple times.

post([*$index = NULL*[, *$xss\_clean = NULL*]])[¶](#44sinio)

| Parameters: | * **$index** (*mixed*) – POST parameter name * **$xss\_clean** (*bool*) – Whether to apply XSS filtering |
| --- | --- |
| Returns: | $\_POST if no parameters supplied, otherwise the POST value if found or NULL if not |
| Return type: | mixed |

The first parameter will contain the name of the POST item you are looking for:

$this->input->post('some\_data');

The method returns NULL if the item you are attempting to retrieve does not exist.

The second optional parameter lets you run the data through the XSS filter. It’s enabled by setting the second parameter to boolean TRUE or by setting your $config['global\_xss\_filtering'] to TRUE.

$this->input->post('some\_data', TRUE);

To return an array of all POST items call without any parameters.

To return all POST items and pass them through the XSS filter set the first parameter NULL while setting the second parameter to boolean TRUE.

$this->input->post(NULL, TRUE); // returns all POST items with XSS filter  
$this->input->post(NULL, FALSE); // returns all POST items without XSS filter

To return an array of multiple POST parameters, pass all the required keys as an array.

$this->input->post(array('field1', 'field2'));

Same rule applied here, to retrive the parameters with XSS filtering enabled, set the second parameter to boolean TRUE.

$this->input->post(array('field1', 'field2'), TRUE);

get([*$index = NULL*[, *$xss\_clean = NULL*]])[¶](#2jxsxqh)

| Parameters: | * **$index** (*mixed*) – GET parameter name * **$xss\_clean** (*bool*) – Whether to apply XSS filtering |
| --- | --- |
| Returns: | $\_GET if no parameters supplied, otherwise the GET value if found or NULL if not |
| Return type: | mixed |

This method is identical to post(), only it fetches GET data.

$this->input->get('some\_data', TRUE);

To return an array of all GET items call without any parameters.

To return all GET items and pass them through the XSS filter set the first parameter NULL while setting the second parameter to boolean TRUE.

$this->input->get(NULL, TRUE); // returns all GET items with XSS filter  
$this->input->get(NULL, FALSE); // returns all GET items without XSS filtering

To return an array of multiple GET parameters, pass all the required keys as an array.

$this->input->get(array('field1', 'field2'));

Same rule applied here, to retrive the parameters with XSS filtering enabled, set the second parameter to boolean TRUE.

$this->input->get(array('field1', 'field2'), TRUE);

post\_get(*$index*[, *$xss\_clean = NULL*])[¶](#z337ya)

| Parameters: | * **$index** (*string*) – POST/GET parameter name * **$xss\_clean** (*bool*) – Whether to apply XSS filtering |
| --- | --- |
| Returns: | POST/GET value if found, NULL if not |
| Return type: | mixed |

This method works pretty much the same way as post() and get(), only combined. It will search through both POST and GET streams for data, looking in POST first, and then in GET:

$this->input->post\_get('some\_data', TRUE);

get\_post(*$index*[, *$xss\_clean = NULL*])[¶](#3j2qqm3)

| Parameters: | * **$index** (*string*) – GET/POST parameter name * **$xss\_clean** (*bool*) – Whether to apply XSS filtering |
| --- | --- |
| Returns: | GET/POST value if found, NULL if not |
| Return type: | mixed |

This method works the same way as post\_get() only it looks for GET data first.

$this->input->get\_post(‘some\_data’, TRUE);

Note

This method used to act EXACTLY like post\_get(), but it’s behavior has changed in CodeIgniter 3.0.

cookie([*$index = NULL*[, *$xss\_clean = NULL*]])[¶](#1y810tw)

| Parameters: | * **$index** (*mixed*) – COOKIE name * **$xss\_clean** (*bool*) – Whether to apply XSS filtering |
| --- | --- |
| Returns: | $\_COOKIE if no parameters supplied, otherwise the COOKIE value if found or NULL if not |
| Return type: | mixed |

This method is identical to post() and get(), only it fetches cookie data:

$this->input->cookie('some\_cookie');  
$this->input->cookie('some\_cookie, TRUE); // with XSS filter

To return an array of multiple cookie values, pass all the required keys as an array.

$this->input->cookie(array('some\_cookie', 'some\_cookie2'));

Note

Unlike the [*Cookie Helper*](http://docs.google.com/helpers/cookie_helper.html) function [get\_cookie()](http://docs.google.com/helpers/cookie_helper.html#get_cookie), this method does NOT prepend your configured $config['cookie\_prefix'] value.

server(*$index*[, *$xss\_clean = NULL*])[¶](#4i7ojhp)

| Parameters: | * **$index** (*mixed*) – Value name * **$xss\_clean** (*bool*) – Whether to apply XSS filtering |
| --- | --- |
| Returns: | $\_SERVER item value if found, NULL if not |
| Return type: | mixed |

This method is identical to the post(), get() and cookie() methods, only it fetches server data ($\_SERVER):

$this->input->server('some\_data');

To return an array of multiple $\_SERVER values, pass all the required keys as an array.

$this->input->server(array('SERVER\_PROTOCOL', 'REQUEST\_URI'));

input\_stream([*$index = NULL*[, *$xss\_clean = NULL*]])[¶](#2xcytpi)

| Parameters: | * **$index** (*mixed*) – Key name * **$xss\_clean** (*bool*) – Whether to apply XSS filtering |
| --- | --- |
| Returns: | Input stream array if no parameters supplied, otherwise the specified value if found or NULL if not |
| Return type: | mixed |

This method is identical to get(), post() and cookie(), only it fetches the *php://input* stream data.

set\_cookie(*$name = ''*[, *$value = ''*[, *$expire = ''*[, *$domain = ''*[, *$path = '/'*[, *$prefix = ''*[, *$secure = FALSE*[, *$httponly = FALSE*]]]]]]])[¶](#1ci93xb)

| Parameters: | * **$name** (*mixed*) – Cookie name or an array of parameters * **$value** (*string*) – Cookie value * **$expire** (*int*) – Cookie expiration time in seconds * **$domain** (*string*) – Cookie domain * **$path** (*string*) – Cookie path * **$prefix** (*string*) – Cookie name prefix * **$secure** (*bool*) – Whether to only transfer the cookie through HTTPS * **$httponly** (*bool*) – Whether to only make the cookie accessible for HTTP requests (no JavaScript) |
| --- | --- |
| Return type: | void |

Sets a cookie containing the values you specify. There are two ways to pass information to this method so that a cookie can be set: Array Method, and Discrete Parameters:

**Array Method**

Using this method, an associative array is passed to the first parameter:

$cookie = array(  
 'name' => 'The Cookie Name',  
 'value' => 'The Value',  
 'expire' => '86500',  
 'domain' => '.some-domain.com',  
 'path' => '/',  
 'prefix' => 'myprefix\_',  
 'secure' => TRUE  
);  
  
$this->input->set\_cookie($cookie);

**Notes**

Only the name and value are required. To delete a cookie set it with the expiration blank.

The expiration is set in **seconds**, which will be added to the current time. Do not include the time, but rather only the number of seconds from *now* that you wish the cookie to be valid. If the expiration is set to zero the cookie will only last as long as the browser is open.

For site-wide cookies regardless of how your site is requested, add your URL to the **domain** starting with a period, like this: .your-domain.com

The path is usually not needed since the method sets a root path.

The prefix is only needed if you need to avoid name collisions with other identically named cookies for your server.

The secure boolean is only needed if you want to make it a secure cookie by setting it to TRUE.

**Discrete Parameters**

If you prefer, you can set the cookie by passing data using individual parameters:

$this->input->set\_cookie($name, $value, $expire, $domain, $path, $prefix, $secure);

ip\_address()[¶](#3whwml4)

| Returns: | Visitor’s IP address or ‘0.0.0.0’ if not valid |
| --- | --- |
| Return type: | string |

Returns the IP address for the current user. If the IP address is not valid, the method will return ‘0.0.0.0’:

echo $this->input->ip\_address();

Important

This method takes into account the $config['proxy\_ips'] setting and will return the reported HTTP\_X\_FORWARDED\_FOR, HTTP\_CLIENT\_IP, HTTP\_X\_CLIENT\_IP or HTTP\_X\_CLUSTER\_CLIENT\_IP address for the allowed IP addresses.

valid\_ip(*$ip*[, *$which = ''*])[¶](#2bn6wsx)

| Parameters: | * **$ip** (*string*) – IP address * **$which** (*string*) – IP protocol (‘ipv4’ or ‘ipv6’) |
| --- | --- |
| Returns: | TRUE if the address is valid, FALSE if not |
| Return type: | bool |

Takes an IP address as input and returns TRUE or FALSE (boolean) depending on whether it is valid or not.

Note

The $this->input->ip\_address() method above automatically validates the IP address.

if ( ! $this->input->valid\_ip($ip))  
{  
 echo 'Not Valid';  
}  
else  
{  
 echo 'Valid';  
}

Accepts an optional second string parameter of ‘ipv4’ or ‘ipv6’ to specify an IP format. The default checks for both formats.

user\_agent([*$xss\_clean = NULL*])[¶](#qsh70q)

| Returns: | User agent string or NULL if not set |
| --- | --- |
| Parameters: | * **$xss\_clean** (*bool*) – Whether to apply XSS filtering |
| Return type: | mixed |

Returns the user agent string (web browser) being used by the current user, or NULL if it’s not available.

echo $this->input->user\_agent();

See the [*User Agent Class*](http://docs.google.com/user_agent.html) for methods which extract information from the user agent string.

request\_headers([*$xss\_clean = FALSE*])[¶](#3as4poj)

| Parameters: | * **$xss\_clean** (*bool*) – Whether to apply XSS filtering |
| --- | --- |
| Returns: | An array of HTTP request headers |
| Return type: | array |

Returns an array of HTTP request headers. Useful if running in a non-Apache environment where [apache\_request\_headers()](http://php.net/apache_request_headers) will not be supported.

$headers = $this->input->request\_headers();

get\_request\_header(*$index*[, *$xss\_clean = FALSE*])[¶](#1pxezwc)

| Parameters: | * **$index** (*string*) – HTTP request header name * **$xss\_clean** (*bool*) – Whether to apply XSS filtering |
| --- | --- |
| Returns: | An HTTP request header or NULL if not found |
| Return type: | string |

Returns a single member of the request headers array or NULL if the searched header is not found.

$this->input->get\_request\_header('some-header', TRUE);

is\_ajax\_request()[¶](#49x2ik5)

| Returns: | TRUE if it is an Ajax request, FALSE if not |
| --- | --- |
| Return type: | bool |

Checks to see if the HTTP\_X\_REQUESTED\_WITH server header has been set, and returns boolean TRUE if it is or FALSE if not.

is\_cli\_request()[¶](#2p2csry)

| Returns: | TRUE if it is a CLI request, FALSE if not |
| --- | --- |
| Return type: | bool |

Checks to see if the application was run from the command-line interface.

Note

This method checks both the PHP SAPI name currently in use and if the STDIN constant is defined, which is usually a failsafe way to see if PHP is being run via the command line.

$this->input->is\_cli\_request()

Note

This method is DEPRECATED and is now just an alias for the is\_cli() function.

method([*$upper = FALSE*])[¶](#147n2zr)

| Parameters: | * **$upper** (*bool*) – Whether to return the request method name in upper or lower case |
| --- | --- |
| Returns: | HTTP request method |
| Return type: | string |

Returns the $\_SERVER['REQUEST\_METHOD'], with the option to set it in uppercase or lowercase.

echo $this->input->method(TRUE); // Outputs: POST  
echo $this->input->method(FALSE); // Outputs: post  
echo $this->input->method(); // Outputs: post

[Next](http://docs.google.com/javascript.html)   [Previous](http://docs.google.com/image_lib.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).