* [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](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](#gjdgxs)
  + [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](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](#gjdgxs)
  + [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) »
* Template Parser Class

classic layout

Template Parser Class[**¶**](#30j0zll)

The Template Parser Class can perform simple text substitution for pseudo-variables contained within your view files. It can parse simple variables or variable tag pairs.

If you’ve never used a template engine, pseudo-variable names are enclosed in braces, like this:

<html>  
 <head>  
 <title>{blog\_title}</title>  
 </head>  
 <body>  
 <h3>{blog\_heading}</h3>  
  
 {blog\_entries}  
 <h5>{title}</h5>  
 <p>{body}</p>  
 {/blog\_entries}  
  
 </body>  
</html>

These variables are not actual PHP variables, but rather plain text representations that allow you to eliminate PHP from your templates (view files).

Note

CodeIgniter does **not** require you to use this class since using pure PHP in your view pages lets them run a little faster. However, some developers prefer to use a template engine if they work with designers who they feel would find some confusion working with PHP.

Important

The Template Parser Class is **not** a full-blown template parsing solution. We’ve kept it very lean on purpose in order to maintain maximum performance.

* [Using the Template Parser Class](#1fob9te)
  + [Initializing the Class](#3znysh7)
  + [Parsing templates](#2et92p0)
  + [Variable Pairs](#tyjcwt)
  + [Usage Notes](#3dy6vkm)
  + [View Fragments](#1t3h5sf)
* [Class Reference](#4d34og8)

[**Using the Template Parser Class**](#2s8eyo1)[**¶**](#1fob9te)

[**Initializing the Class**](#17dp8vu)[**¶**](#3znysh7)

Like most other classes in CodeIgniter, the Parser class is initialized in your controller using the $this->load->library() method:

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

Once loaded, the Parser library object will be available using: $this->parser

[**Parsing templates**](#3rdcrjn)[**¶**](#2et92p0)

You can use the parse() method to parse (or render) simple templates, like this:

$data = array(  
 'blog\_title' => 'My Blog Title',  
 'blog\_heading' => 'My Blog Heading'  
);  
  
$this->parser->parse('blog\_template', $data);

The first parameter contains the name of the [view file](http://docs.google.com/general/views.html) (in this example the file would be called blog\_template.php), and the second parameter contains an associative array of data to be replaced in the template. In the above example, the template would contain two variables: {blog\_title} and {blog\_heading}

There is no need to “echo” or do something with the data returned by $this->parser->parse(). It is automatically passed to the output class to be sent to the browser. However, if you do want the data returned instead of sent to the output class you can pass TRUE (boolean) as the third parameter:

$string = $this->parser->parse('blog\_template', $data, TRUE);

[**Variable Pairs**](#26in1rg)[**¶**](#tyjcwt)

The above example code allows simple variables to be replaced. What if you would like an entire block of variables to be repeated, with each iteration containing new values? Consider the template example we showed at the top of the page:

<html>  
 <head>  
 <title>{blog\_title}</title>  
 </head>  
 <body>  
 <h3>{blog\_heading}</h3>  
  
 {blog\_entries}  
 <h5>{title}</h5>  
 <p>{body}</p>  
 {/blog\_entries}  
  
 </body>  
</html>

In the above code you’ll notice a pair of variables: {blog\_entries} data… {/blog\_entries}. In a case like this, the entire chunk of data between these pairs would be repeated multiple times, corresponding to the number of rows in the “blog\_entries” element of the parameters array.

Parsing variable pairs is done using the identical code shown above to parse single variables, except, you will add a multi-dimensional array corresponding to your variable pair data. Consider this example:

$this->load->library('parser');  
  
$data = array(  
 'blog\_title' => 'My Blog Title',  
 'blog\_heading' => 'My Blog Heading',  
 'blog\_entries' => array(  
 array('title' => 'Title 1', 'body' => 'Body 1'),  
 array('title' => 'Title 2', 'body' => 'Body 2'),  
 array('title' => 'Title 3', 'body' => 'Body 3'),  
 array('title' => 'Title 4', 'body' => 'Body 4'),  
 array('title' => 'Title 5', 'body' => 'Body 5')  
 )  
);  
  
$this->parser->parse('blog\_template', $data);

If your “pair” data is coming from a database result, which is already a multi-dimensional array, you can simply use the database result\_array() method:

$query = $this->db->query("SELECT \* FROM blog");  
  
$this->load->library('parser');  
  
$data = array(  
 'blog\_title' => 'My Blog Title',  
 'blog\_heading' => 'My Blog Heading',  
 'blog\_entries' => $query->result\_array()  
);  
  
$this->parser->parse('blog\_template', $data);

[**Usage Notes**](#lnxbz9)[**¶**](#3dy6vkm)

If you include substitution parameters that are not referenced in your template, they are ignored:

$template = 'Hello, {firstname} {lastname}';  
$data = array(  
 'title' => 'Mr',  
 'firstname' => 'John',  
 'lastname' => 'Doe'  
);  
$this->parser->parse\_string($template, $data);  
  
// Result: Hello, John Doe

If you do not include a substitution parameter that is referenced in your template, the original pseudo-variable is shown in the result:

$template = 'Hello, {firstname} {initials} {lastname}';  
$data = array(  
 'title' => 'Mr',  
 'firstname' => 'John',  
 'lastname' => 'Doe'  
);  
$this->parser->parse\_string($template, $data);  
  
// Result: Hello, John {initials} Doe

If you provide a string substitution parameter when an array is expected, i.e. for a variable pair, the substitution is done for the opening variable pair tag, but the closing variable pair tag is not rendered properly:

$template = 'Hello, {firstname} {lastname} ({degrees}{degree} {/degrees})';  
$data = array(  
 'degrees' => 'Mr',  
 'firstname' => 'John',  
 'lastname' => 'Doe',  
 'titles' => array(  
 array('degree' => 'BSc'),  
 array('degree' => 'PhD')  
 )  
);  
$this->parser->parse\_string($template, $data);  
  
// Result: Hello, John Doe (Mr{degree} {/degrees})

If you name one of your individual substitution parameters the same as one used inside a variable pair, the results may not be as expected:

$template = 'Hello, {firstname} {lastname} ({degrees}{degree} {/degrees})';  
$data = array(  
 'degree' => 'Mr',  
 'firstname' => 'John',  
 'lastname' => 'Doe',  
 'degrees' => array(  
 array('degree' => 'BSc'),  
 array('degree' => 'PhD')  
 )  
);  
$this->parser->parse\_string($template, $data);  
  
// Result: Hello, John Doe (Mr Mr )

[**View Fragments**](#35nkun2)[**¶**](#1t3h5sf)

You do not have to use variable pairs to get the effect of iteration in your views. It is possible to use a view fragment for what would be inside a variable pair, and to control the iteration in your controller instead of in the view.

An example with the iteration controlled in the view:

$template = '<ul>{menuitems}  
 <li><a href="{link}">{title}</a></li>  
{/menuitems}</ul>';  
  
$data = array(  
 'menuitems' => array(  
 array('title' => 'First Link', 'link' => '/first'),  
 array('title' => 'Second Link', 'link' => '/second'),  
 )  
);  
$this->parser->parse\_string($template, $data);

Result:

<ul>  
 <li><a href="/first">First Link</a></li>  
 <li><a href="/second">Second Link</a></li>  
</ul>

An example with the iteration controlled in the controller, using a view fragment:

$temp = '';  
$template1 = '<li><a href="{link}">{title}</a></li>';  
$data1 = array(  
 array('title' => 'First Link', 'link' => '/first'),  
 array('title' => 'Second Link', 'link' => '/second'),  
);  
  
foreach ($data1 as $menuitem)  
{  
 $temp .= $this->parser->parse\_string($template1, $menuitem, TRUE);  
}  
  
$template = '<ul>{menuitems}</ul>';  
$data = array(  
 'menuitems' => $temp  
);  
$this->parser->parse\_string($template, $data);

Result:

<ul>  
 <li><a href="/first">First Link</a></li>  
 <li><a href="/second">Second Link</a></li>  
</ul>

[**Class Reference**](#1ksv4uv)[**¶**](#4d34og8)

*class* CI\_Parser[¶](#44sinio) parse(*$template*, *$data*[, *$return = FALSE*])[¶](#2jxsxqh)

|  |  |
| --- | --- |
| Parameters: | * **$template** (*string*) – Path to view file * **$data** (*array*) – Variable data * **$return** (*bool*) – Whether to only return the parsed template |
| Returns: | Parsed template string |
| Return type: | string |

Parses a template from the provided path and variables.

parse\_string(*$template*, *$data*[, *$return = FALSE*])[¶](#z337ya)

|  |  |
| --- | --- |
| Parameters: | * **$template** (*string*) – Path to view file * **$data** (*array*) – Variable data * **$return** (*bool*) – Whether to only return the parsed template |
| Returns: | Parsed template string |
| Return type: | string |

This method works exactly like parse(), only it accepts the template as a string instead of loading a view file.

set\_delimiters([*$l = '{'*[, *$r = '}'*]])[¶](#3j2qqm3)

|  |  |
| --- | --- |
| Parameters: | * **$l** (*string*) – Left delimiter * **$r** (*string*) – Right delimiter |
| Return type: | void |

Sets the delimiters (opening and closing) for a pseudo-variable “tag” in a template.

[Next](http://docs.google.com/security.html)   [Previous](http://docs.google.com/pagination.html)

© Copyright 2014 - 2018, British Columbia Institute of Technology. Last updated on Jun 12, 2018.

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).