

Interested in functions, hooks, classes, or methods? Check out the new [WordPress Code Reference!](#)

pt-br:O Loop

Languages: [English](#) • [Loop 日本語](#) [Português do Brasil](#) • [Русский](#) • [中文\(繁體\)](#) • [\(Add your language\)](#)

ARTIGO PARCIALMENTE TRADUZIDO OU QUE PRECISA DE TRADUÇÃO

Este documento está parcialmente traduzido ou precisa ser traduzido. Toda a tradução é feita por voluntários e você pode ser um deles.

[PARTICIPAR](#) • [ARTIGOS PARA TRADUZIR](#) • [FÓRUM DE SUPORTE](#) • [TODOS OS ARTIGOS](#)

O Loop é usado pelo WordPress para exibir cada uma de suas postagens. Usando o Loop, o WordPress processa cada uma das postagens a serem exibidas na página atual e formata-as de acordo com critérios específicos. Qualquer código [HTML](#) ou [PHP](#) colocado no Loop será repetido em cada postagem. Quando a documentação do WordPress diz "Essa tag deve estar dentro do Loop", como no caso específico de [Tags de Modelo](#) ou plugins, a tag será repetida para cada postagem.

Por exemplo, dentre as informações que o Loop mostra por padrão: o Título ([the_title\(\)](#)), a Hora([the_time\(\)](#)) e Categorias ([the_category\(\)](#)) para cada post. Outras informações sobre cada post podem ser mostradas com as [Tags de Modelo](#) apropriadas ou (para usuários avançados) acessando a variável `$post`, que recebe as informações do post atual enquanto o Loop está sendo executado.

Para uma visão introdutória do Loop, veja [O Loop em ação](#).

Utilizando o Loop

O Loop deve ser colocado na `index.php` e em qualquer outro arquivo do tema utilizado para mostrar as informações de posts.

Certifique-se de incluir a chamada para o header no topo dos arquivos do seu [Tema](#). Se você está utilizando o Loop em um design próprio (e o seu design não é um Tema), defina `WP_USE_THEMES` como falso.

```
<?php define('WP_USE_THEMES', false); get_header(); ?>
```

O Loop começa aqui:

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
```

e termina aqui:

```
<?php endwhile; else: ?>
<p><?php _e('Sorry, no posts matched your criteria.'); ?></p>
<?php endif; ?>
```

Contents

- [1 Utilizando o Loop](#)
- [2 Exemplos de Loop](#)
 - [2.1 Diferenciar os estilos dos posts de algumas categorias](#)
 - [2.2 Excluir posts de uma ou mais categorias](#)
- [3 Múltiplos Loops](#)
 - [3.1 Loop Examples](#)
 - [3.1.1 Multiple Loops Example 1](#)
 - [3.1.2 Multiple Loops Example 2](#)
 - [3.2 Multiple Loops in Action](#)
- [4 Nested Loops](#)
- [5 Sources](#)
- [6 More Loop Resources](#)

Entre o começo e o fim do Loop, são mostradas as Template Tags, que mostram as informações de cada post, tais como título, categorias, autor, data, etc. Para uma lista do que pode ser mostrado dentro do Loop, consulte as [Template Tags](#).

É provável que seu tema já tenha o loop em todas as páginas, o que é chamado de Loop principal. Se você quiser personalizar o Loop principal, utilize o [WP_Query\(\)](#). Para saber qual Loop você deve personalizar, dependendo da parte do seu site que você pretende modificar, consulte a [Hierarquia de Modelos WordPress](#).

Se, por outro lado, você quiser criar outros loops adicionais, utilize uma das formas abaixo (Mais informações em [Múltiplos Loops](#)):

- `get_posts()`

Exemplos de Loop

Diferenciar os estilos dos posts de algumas categorias

Este exemplo mostra cada post com seu Título (que é usado como link para o [link permanente](#) do post), suas Categorias e Conteúdo. É um exemplo simples, básico; provavelmente seu Tema mostrará mais informações, de forma a tornar mais fácil definir seu estilo com [CSS](#).

Mas, para ser um pouco mais instrutivo, este exemplo também permite diferenciar o estilo dos posts na Categoria com ID '3'. Para conseguir fazer isso, é utilizada a [Template Tag in_category\(\)](#).

A tag `<!-- -->` é uma tag de comentário HTML; se você usar este exemplo, essas tags não serão mostradas no navegador. Elas servem apenas para comentar o código abaixo.

```
<!-- Começa o Loop. -->
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>

<!-- O código a seguir testa se o post atual pertence à categoria 3 -->
<!-- Se pertence, a classe css da DIV será definida como "post-cat-three". -->
<!-- Se não, a classe da DIV será definida como "post". -->
<?php if ( in_category('3') ) { ?>
    <div class="post-cat-three">
<?php } else { ?>
    <div class="post">
<?php } ?>

<!-- Mostra o título como um link para o link permanente do post -->
<h2><a href="<?php the_permalink() ?>" rel="bookmark" title="Permanent Link to <?php the_title_attribute(); ?>"><?php the_title(); ?></a></h2>

<!-- Mostra a data e um link para outros posts do mesmo autor. -->
<small><?php the_time('F jS, Y') ?> by <?php the_author_posts_link() ?></small>

<!-- Mostra o conteúdo do post em uma DIV -->
<div class="entry">
    <?php the_content(); ?>
</div>

<!-- Mostra uma lista de categorias do post separadas por vírgula -->
<p class="postmetadata">Posted in <?php the_category(', '); ?></p>
</div> <!-- Fecha a primeira DIV -->

<!-- Termina o Loop (mas repare no "else" - veja próxima linha) -->
<?php endwhile; else: ?>

<!-- O primeiro IF testou para ver se havia posts a serem mostrados -->
<!-- Este ELSE diz ao WordPress o que fazer se não houver nenhum -->
<p>Sorry, no posts matched your criteria.</p>

<!-- Término verdadeiro do Loop -->
<?php endif; ?>
```

Observação: Sempre que você quiser utilizar código em [HTML](#), você deve estar fora das tags `<?php ?>`. Código [PHP](#) (mesmo coisas tão simples como parênteses `: }`) devem estar contidas em tags `<?php ?>`. Você pode iniciar e interromper código em PHP de forma a introduzir código HTML, mesmo dentro de comandos `if` e `else`, como é mostrado no exemplo acima.

Excluir posts de uma ou mais categorias

Este exemplo pode ser usado para excluir uma determinada categoria ou categorias do Loop. Neste caso, as categorias com IDs 3 e 8 são excluídas. Este exemplo é diferente do [exemplo acima](#) porque ele faz uma modificação na própria `query` que define o que será mostrada pelo Loop.

```

<?php query_posts($query_string . '&cat=-3,-8'); ?>
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>

<div class="post">

<!-- Mostra o título como um link para o link permanente do post --&gt;
&lt;h2&gt;&lt;a href="&lt;?php the_permalink() ?&gt;" rel="bookmark" title="Permanent Link to &lt;?php the_title_attribute(); ?&gt;"&gt;&lt;?php the_title(); ?&gt;&lt;/a&gt;&lt;/h2&gt;

<!-- Mostra a data e um link para outros posts do mesmo autor. --&gt;
&lt;small&gt;&lt;?php the_time('F jS, Y') ?&gt; by &lt;?php the_author_posts_link() ?&gt;&lt;/small&gt;

&lt;div class="entry"&gt;
    &lt;?php the_content(); ?&gt;
&lt;/div&gt;

&lt;p class="postmetadata"&gt;Posted in &lt;?php the_category(', ') ?&gt;&lt;/p&gt;
&lt;/div&gt; &lt!-- closes the first div box --&gt;

&lt;?php endwhile; else: ?&gt;
&lt;p&gt;Sorry, no posts matched your criteria.&lt;/p&gt;
&lt;?php endif; ?&gt;
</pre>

```

Observação: Se você usar este exemplo na página principal, você deve usar um [Template](#) diferente para a sua [página de categorias](#), porque senão o WordPress excluirá todos os posts das categorias 3 e 8, mesmo quando estiver mostrando o Arquivo de Categorias! Entretanto, se você quiser utilizar o mesmo arquivo template, você pode evitar isso usando a função [is_home\(\)](#) para garantir que os posts das categorias 3 e 8 só serão excluídos na página inicial:

```

...
<?php if ( is_home() ) {
    query_posts($query_string . '&cat=-3,-8');
}
?>
...

```

Há outras [Tags Condicionais](#) que podem ser usadas para controlar o que é mostrado dependendo de uma determinada condição ser verdadeira ou não com respeito a uma página que está sendo mostrada.

Múltiplos Loops

Esta seção explica o uso avançado do Loop. É um pouco técnica, mas não deixe isso te assustar. Começaremos em um nível mais fácil e evoluiremos a partir daí. Com um pouco de bom senso, paciência e entusiasmo, você também poderá fazer Loops múltiplos.

Em primeiro lugar, "por que alguém iria querer usar múltiplos loops?" Geralmente, a resposta é que você poderia querer fazer algo diferente para um determinado grupo de posts, mas mostrar os dois grupos na mesma página. "Algo diferente" quer dizer qualquer coisa; você está limitado apenas pelas suas habilidades em PHP e sua imaginação.

Mostraremos exemplos daqui a pouco, mas primeiro você deve conhecer o básico. Veja o loop básico. Ele consiste de:

```

<?php if (have_posts()) : ?>
    <?php while (have_posts()) : the_post(); ?>
        <!-- faça coisas... -->
    <?php endwhile; ?>
<?php endif; ?>

```

Em bom português (pessoas familiarizadas com PHP podem pular esta explicação), o código acima deve ser lido da seguinte maneira: se vamos mostrar posts, vá buscá-los, um a um. Para cada post da lista, mostre-o de acordo com `<!-- faça coisas... -->`. Quando chegar ao último post, pare. As linhas do `faça coisas` dependem do tema utilizado.

A little aside on `<tt>Do stuff`: in this example it is simply a placeholder for a bunch of code that determines how to format and display each post on a page. This code can change depending on how you want your WordPress to look. If you look at the Kubrick theme's index.php the `do stuff` section would be everything below:

```

<?php while (have_posts()) : the_post(); ?>

```

To above:

```
<?php comments_popup_link('No Comments »', '1 Comment »', '% Comments »'); ?>
```

An explanation for the coders out there: The `have_posts()` and `the_post()` are convenience wrappers around the global `$wp_query` object, which is where all of the action is. The `$wp_query` is called in the blog header and fed query arguments coming in through GET and PATH_INFO. The `$wp_query` takes the arguments and builds and executes a DB query that results in an array of posts. This array is stored in the object and also returned back to the blog header where it is stuffed into the global `$posts` array (for backward compatibility with old post loops).

Once WordPress has finished loading the blog header and is descending into the template, we arrive at our post Loop. The `have_posts()` simply calls into `$wp_query->have_posts()` which checks a loop counter to see if there are any posts left in the post array. And `the_post()` calls `$wp_query->the_post()` which advances the loop counter and sets up the global `$post` variable as well as all of the global post data. Once we have exhausted the loop, `have_posts()` will return false and we are done.

Loop Examples

Below are three examples of using multiple loops. The key to using multiple loops is that `$wp_query` can only be called once. In order to get around this it is possible to re-use the query by calling `rewind_posts()` or by creating a new query object. This is covered in example 1. In example 2, using a variable to store the results of a query is covered. Finally, 'multiple loops in action' brings a bunch of ideas together to document one way of using multiple loops to promote posts of a certain category on your blog's homepage.

Multiple Loops Example 1

In order to loop through the same query a second time, call `rewind_posts()`. This will reset the loop counter and allow you to do another loop.

```
<?php rewind_posts(); ?>  
<?php while (have_posts()) : the_post(); ?>  
    <!-- Do stuff... -->  
<?php endwhile; ?>
```

If you are finished with the posts in the original query, and you want to use a different query, you can reuse the `$wp_query` object by calling `query_posts()` and then looping back through. The `query_posts()` will perform a new query, build a new posts array, and reset the loop counter.

```
// Get the last 10 posts in the special_cat category.  
<?php query_posts('category_name=special_cat&posts_per_page=10'); ?>  
  
<?php while (have_posts()) : the_post(); ?>  
    <!-- Do special_cat stuff... -->  
<?php endwhile; ?>
```

If you need to keep the original query around, you can create a new query object.

```
<?php $my_query = new WP_Query('category_name=special_cat&posts_per_page=10'); ?>  
  
<?php while ($my_query->have_posts()) : $my_query->the_post(); ?>  
    <!-- Do special_cat stuff... -->  
<?php endwhile; ?>
```

The query object `my_query` is used because you cannot use the global `have_posts()` and `the_post()` since they both use `$wp_query`. Instead, call into your new `$my_query` object.

Multiple Loops Example 2

Another version of using multiple Loops takes another tack for getting around the inability to use `have_posts()` and `the_post()`. To solve this, you need to store the original query in a variable, then re-assign it with the other Loop. This way, you can use all the standard functions that rely on all the globals.

For example:

```
// going off on my own here
<?php $temp_query = $wp_query; ?>
<!-- Do stuff... -->

<?php query_posts('category_name=special_cat&posts_per_page=10'); ?>

<?php while (have_posts()) : the_post(); ?>
    <!-- Do special_cat stuff... -->
<?php endwhile; ?>

// now back to our regularly scheduled programming
<?php $wp_query = $temp_query; ?>
```

Note: In PHP 5, objects are referenced with the "=" operator instead of copied like in PHP 4. To make Example 2 work in PHP 5 you need to use the following code:

```
// going off on my own here
<?php $temp_query = clone $wp_query; ?>
<!-- Do stuff... -->

<?php query_posts('category_name=special_cat&posts_per_page=10'); ?>

<?php while (have_posts()) : the_post(); ?>
    <!-- Do special_cat stuff... -->
<?php endwhile; ?>
<?php endif; ?>

// now back to our regularly scheduled programming
<?php $wp_query = clone $temp_query; ?>
```

However, this second example does not work in WordPress 2.1.

Multiple Loops in Action

The best way to understand how to use multiple loops is to actually show an example of its use. Perhaps the most common use of multiple loops is to show two (or more) lists of posts on one page. This is often done when a webmaster wants to feature not only the very latest post written, but also posts from a certain category.

Leaving all formatting and CSS issues aside, let us assume we want to have two lists of posts. One which would list the most recent posts (the standard 10 posts most recently added), and another which would contain only one post from the category 'featured'. Posts in the 'featured' category should be shown first, followed by the second listing of posts (the standard). The catch is that no post should appear in both categories.

Step 1. Get only one post from the 'featured' category.

```
<?php $my_query = new WP_Query('category_name=featured&posts_per_page=1');
while ($my_query->have_posts()) : $my_query->the_post();
$do_not_duplicate = $post->ID; ?>
    <!-- Do stuff... -->
<?php endwhile; ?>
```

In English the above code would read:

Set \$my_query equal to the result of querying all posts where the category is named featured and by the way, get me one post only. Also, set the variable \$do_not_duplicate equal to the ID number of the single post returned. Recall that the Do stuff line represents all the formatting options associated for the post retrieved.

Note that we will need the value of \$do_not_duplicate in the next step to ensure that the same post doesn't appear in both lists.

Step 2. The second loop, get the X latest posts (except one).

The following code gets X recent posts (as defined in WordPress preferences) save the one already displayed from the first loop and displays them according to Do stuff.

```
<?php if (have_posts()) : while (have_posts()) : the_post();
if( $post->ID == $do_not_duplicate ) continue;?>
    <!-- Do stuff... -->
<?php endwhile; endif; ?>
```

In English the above code would read:

Get all posts, where a post equals \$do_not_duplicate then just do nothing (continue), otherwise display all the other the posts according to Do stuff. Also, update the cache so the tagging and keyword plugins play nice. Recall, \$do_not_duplicate variable contains the ID of the post already displayed.

The End Result

Here is what the final piece of code looks like without any formatting:

```
<?php $my_query = new WP_Query('category_name=featured&posts_per_page=1');
while ($my_query->have_posts()) : $my_query->the_post();
$do_not_duplicate = $post->ID;?>
<!-- Do stuff... -->
<?php endwhile; ?>
<!-- Do other stuff... -->
<?php if (have_posts()) : while (have_posts()) : the_post();
if( $post->ID == $do_not_duplicate ) continue; ?>
<!-- Do stuff... -->
<?php endwhile; endif; ?>
```

The end result would be a page with two lists. The first list contains only one post -- the most recent post from the 'feature' category. The second list will contain X recent posts (as defined in WordPress preferences) except the post that is already shown in the first list. So, once the feature post is replaced with a new one, the previous feature will show up in standard post list section below (depending on how many posts you choose to display and on the post frequency). This technique (or similar) has been used by many in conjunction with knowledge of the [Template Hierarchy](#) to create a different look for home.php and index.php. See associated resources at the bottom of this page.

Note for Multiple Posts in the First Category

If posts_per_page=2 or more, you will need to alter the code a bit. The variable \$do_not_duplicate needs to be changed into an array as opposed to a single value. Otherwise, the first loop will finish and the variable \$do_not_duplicate will equal only the id of the latest post. This will result in duplicated posts in the second loop. To fix the problem replace

```
<?php $my_query = new WP_Query('category_name=featured&posts_per_page=1');
while ($my_query->have_posts()) : $my_query->the_post();
$do_not_duplicate = $post->ID;?>
```

with

```
<?php $my_query = new WP_Query('category_name=featured&posts_per_page=2');
while ($my_query->have_posts()) : $my_query->the_post();
$do_not_duplicate[] = $post->ID ?>
```

Note that "posts_per_page" can be any number. This changes \$do_not_duplicate into an array. Then replace

```
<?php if (have_posts()) : while (have_posts()) : the_post(); if( $post->ID ==
$do_not_duplicate ) continue; ?>
```

with

```
<?php if (have_posts()) : while (have_posts()) : the_post();
if (in_array($post->ID, $do_not_duplicate)) continue;
?>
```

Where you continue the pattern for whatever posts_per_page is set equal to (2 in this case).

Alternatively you can pass the entire \$do_not_duplicate array to \$wp_query and only entries that match your criteria will be returned:

```
<?php query_posts(array('post__not_in'=>$do_not_duplicate));
if (have_posts()) : while (have_posts()) : the_post();
?>
```

Note that instead a string, the query parameter was an associative array, with `post__not_in` option.

Nested Loops

Nesting loops means that you are running a second loop before finishing the first one. This can be useful to display a post list with a [shortcode](#) for example.

```
$my_query = new WP_Query( "cat=3" );
if ( $my_query->have_posts() ) {
    while ( $my_query->have_posts() ) {
        $my_query->the_post();
        the_content();
    }
}
wp_reset_postdata();
```

It is necessary to reset the main loop data after a nested loop so that some global variables hold the correct values again.

Sources

The section on multiple loops is a combination of [Ryan Boren](#) and [Alex King's discussion](#) about the Loop on the [Hackers Mailing List](#) as well as the tutorial written at [MaxPower](#). The nested loops example was inspired by [another discussion](#) on the mailing list and a post by [Nicolas Kuttler](#).

More Loop Resources

To learn more about the WordPress Loop, and the various template tags that work only within the Loop, here are more resources.

- [The Loop in Action](#)
- [Template Tags](#)
- [Using the Loop in Template Files](#)
- [Getting Into The Loop](#) - (slides) an introduction to how plugins and themes can modify the Loop



Codex WordPress Brasil

A documentação do WordPress em Português do Brasil.
Todas as comunidades lusófonas também são bem-vindas! Adicione {{Codex-pt}} em seus artigos.

Category:

- [Codex-pt](#)

About
News
Hosting
Privacy
Showcase
Themes
Plugins
Patterns
Learn
Documentation

[Developers](#)

[WordPress.tv ↗](#)

[Get Involved](#)

[Events](#)

[Donate ↗](#)

[Swag Store ↗](#)

[WordPress.com ↗](#)

[Matt ↗](#)

[bbPress ↗](#)

[BuddyPress ↗](#)



CODE IS POETRY