# [Mastering WP\_Query: Related Functions](http://code.tutsplus.com/tutorials/mastering-wp_query-related-functions--cms-22951)

掌握WP\_Query：相关的函数

Hi, and welcome to the second part of the series called "Mastering WP\_Query". In the first part, we started off with the series by simply introducing what the WP\_Queryclass is. And in this second part, we're going to learn about the functions related toWP\_Query.

大家好，欢迎来到“掌握WP\_Query”第二部分。在第一部分中，我们学习了WP\_Query类，在这部分中，我们将要学习WP\_Query的相关函数。

## Harnessing the Power of WP\_Query With Functions, Actions and Filters

掌握wp\_query的强大功能，从函数（Functions）、行动器（Actions）、过滤器（Filters）着手。

I'm not an expert on programming, but I can see that the WP\_Query class is one of the best examples of the MVC pattern. It's extremely powerful, completely extensible, and very easy to use once you get to know the basics.

我不是编程的专家，但是我知道WP\_Query类是MVC模式最好的例子。当你掌握了基本知识后，你会了解到wp\_query功能很强大、容易扩展和使用。

In addition to its properties, methods and parameters (which we're going to go through in the future), WordPress offers extra functions and hooks (meaning actions and filters) to work with the WP\_Query class. In this tutorial, we're going to get to know the functions and in the next part, we'll see actions and filters related to WP\_Query. If you see that I forgot something, feel free to notify me by shooting a comment.

除本篇文章之外，我们在将来可能要用到的方法和参数，WordPresss都提供了另外的函数和挂钩（hooks），也就是行动器（Actions）、过滤器（Filters），用来与WP\_Query类共同工作。本教程的下一部分中将会进行介绍与WP\_Query相关的行动器（Actions）、过滤器（Filters）。如果其中有什么问题，可以邮件我。

## WP\_Query Related Functions

WP\_Query 相关的函数

There are 13 WordPress functions that work with the WP\_Query class that can help you harness the power of WP\_Query. They don't need any introduction, so here we go:

下面介绍13个Wordpress与WP\_Query相关的函数，帮助你掌握WP\_Query的强大功能。不需要过多的辞藻，让我们开始：

### Getting Public Query Variables: get\_query\_var()

获取公用的查询参数：get\_query\_var()

Our first function's name speaks for itself, really: It takes the global $wp\_query object and retrieves a public query variable from it. It has two parameters: The first one is the variable to return its value, and the second one is a default value to return if the variable is not set:

从函数名的字面义就可以知道：从全局变量$wp\_query对象中，检索公共变量。有两个参数：第一个返回值，第二个在第一个没有设置的情况下返回默认值。

<?php

$paged = get\_query\_var( 'paged', 1 );

?>

### Affecting the Main Loop: query\_posts()

关联主循环：query\_post()

To be honest, this is a bad function. You really don't have to use it at all, but for the sake of telling "the right way", we're going to go over it and tell you it's "the wrong way".

如实的讲，这是一个很差劲的函数。可以完全不用使用它，但是为了讲明白“正确的方式”使用它，我们将会学习它，并且告诉你什么是“差劲的功能”。

query\_posts() is a function that alters the main query by putting the main query aside and running a new query—and you'll need to use wp\_reset\_query() to clean up after it.

Query\_posts()是用来与主查询进行替换的，把主查询放在一边，并开始一个新的查询——如之前提到的，需要在查询完后使用wp\_reset\_query()。

<?php

query\_posts( 'category\_name=news' );

//codes here

wp\_reset\_query();

?>

This function is one of the most misused functions in the core of WordPress. You shouldn't use it to create secondary queries—you can use the WP\_Query class or the get\_posts() function (which we'll talk about later in this tutorial). You shouldn't use it to alter the main query either—you should use the pre\_get\_posts action (which we'll talk about later as well). [Even the Codex discourages us](https://codex.wordpress.org/Function_Reference/query_posts) from using it and shows alternatives.

这是wordpress中最少用到的功能。完全可以不使用它去新建一个查询——可以直接使用WP\_Query类中的get\_posts()函数（后面将要介绍到）。不用使用它去替换主查询，另外可以使用pre\_get\_posts 行动器（action）来完成相应的操作（后面我们将会介绍到）。甚至，在[wordpress官方文档](https://codex.wordpress.org/Function_Reference/query_posts)中，也不推荐我们去使用它。

Bottom line: Don't use it.

总之，另用它。

### Getting a Single Post: get\_post()

取一篇文章：get\_post()

Another self-explanatory function is get\_post(), which you use to get a single post. It has three optional parameters:

另一个自明其义的函数是get\_post()，可以用来取一篇文章。有3个可选参数：

* The first one is the post ID (or the ID of the current post by default).
* The second one is the type of the result you'll get: either OBJECT, ARRAY\_A (an associative array) or ARRAY\_N (a numeric array).
* The third one is the choice of filtering the result. The default is 'raw' so it won't be filtered unless you set it to 'edit', 'display', 'attribute' or 'js'.
* 第一个参数是文章ID（或者是默认的当前文章）
* 第二个参数是获得文章的类型：对象OBJECT、关联数组ARRAY\_A或者是数值数组ARRAY\_N。
* 第三个参数是可选的过滤结果。如果没有设置默认的是“行”row排列，或者你可以设置成为“编辑”edit，“显示”display，“归属”attribute或者“js”进行排列结果。

<?php

$first\_post = get\_post( 1, ARRAY\_A );

$post\_title = $first\_post[ 'post\_title' ];

?>

### Saving Queries to Arrays: get\_posts()

保存查询结果到数组中：get\_posts()

The get\_posts() function allows us to run queries and save them as arrays to use in various places. It requires the same arguments with WP\_Query so you can customize the query as you like. (We'll get to review WP\_Query's parameters in the future, so stay tuned!) It's the best and most efficient way to create post lists—but not loops.

get\_posts()函数可以运行查询并保存到数组中，当需要在不同的地方使用的时候，就可以拿出来使用。需要与WP\_Query相同的参数，所以定制你自己喜欢的查询吧。（我们将会在之后重新回顾WP\_Query的参数，所以请继续看教程）get\_posts()函数是最好也是最高效生成文章列表的非loop循环的功能。

<?php

$args = array(

    'category\_name' => 'news',

    'order' => 'ASC',

    'orderby' => 'post\_title',

    'posts\_per\_page' => -1

);

// Return an array of all posts in the "news" category.

$all\_posts\_list = get\_posts( $args );

?>

Although this function can be used to run "secondary queries" without a problem, the Codex recommends using WP\_Query when creating multiple loops and using get\_posts() when fetching a post list. [Peter R. Knight explains the difference between get\_posts() and WP\_Query](http://www.peterrknight.com/differences-between-using-get_post-and-wp_query/) in simple terms: The main difference is that WP\_Query makes more database queries (post data, meta data, author data and comment data) while get\_posts() makes only one query (post data).

虽然这个函数可以运行“第二个查询”而不出现任何问题，官方文档指导推荐使用WP\_Query去创建多重loop循环而使用get\_posts()去生成文章列表。Peter R. Knight 简单的解释了get\_posts()和WP\_Query之间的区别：最主要的区别是WP\_Query会导致更多的数据库查询（文章数据，摘要数据，作者和评论数据），而get\_posts()仅仅查询了文章数据。

### Getting Pages: get\_pages()

This odd function has the purpose of fetching a list of pages, although it has a post\_type parameter that can also let you select another post type (provided that the post type is hierarchical, otherwise it returns false).

获取页面：get\_pages()

这个奇葩的函数是用来获取页面的列表的，不仅可以传递post\_type参数，还可以让你选择另一篇文章类型 (前提是这个文章类型是分层的，否则返回false)。

<?php

$args = array(

    'sort\_order' => 'ASC',

    'sort\_column' => 'post\_title',

    'hierarchical' => 1,

    'exclude' => '',

    'include' => '',

    'meta\_key' => '',

    'meta\_value' => '',

    'authors' => '',

    'child\_of' => 0,

    'parent' => -1,

    'exclude\_tree' => '',

    'number' => '',

    'offset' => 0,

    'post\_type' => 'page',

    'post\_status' => 'publish'

);

$pages = get\_pages( $args );

?>

It accepts arguments that are very similar to WP\_Query's arguments, but they're slightly different:

这个函数接收到的参数很类似WP\_Query的参数，有一点点不同：

* sort\_order: Whether to sort the pages in an ascending (asc) or descending (desc) order.
* sort\_column: How to sort the pages. Accepts post\_title, menu\_order, post\_date, post\_modified, ID, post\_author, and post\_name.
* hierarchical: Whether to list pages hierarchically (1) or not (0).
* exclude: A comma separated list or an array of page IDs to exclude from the listing.
* include: A comma separated list or an array of page IDs to include from the listing and exclude everything else.
* meta\_key: When used with the meta\_value argument, it only includes pages that have the defined meta key and value.
* meta\_value: When used with the meta\_key argument, it only includes pages that have the defined meta key and value.
* authors: A comma separated list of author IDs.
* child\_of: An ID of a page to fetch only its children and grandchildren in the listing.
* parent: List the pages that have the provided page ID as parent. In order for this argument to work, the hierarchical argument must be set to 0.
* exclude\_tree: A comma separated list or an array of page IDs to exclude with its children.
* number: Number of pages to fetch.
* offset: Number of pages to skip from the top.
* post\_type: The post type to query. Naturally, it defaults to page.
* post\_status: A comma separated list of post status types to include.
* sort\_order: 是否在一个提升序排列 (asc)或降序排列(desc)。
* sort\_column:如何分类页面。接受post\_title、menu\_order post\_date post\_modified,ID,post\_author和post\_name参数。
* hierarchical：是否分层（1）显示页面，不分层（0）
* exclude:用逗号分隔的列表、页面ID数组，用来排除get\_post生成的列表结果
* include: 用逗号分隔的列表、页面ID数组，用来包含get\_post生成的特定页面列表
* meta\_key: 当使用了meta\_key参数时，get\_post的结果只会显示有meta\_key的页面。
* meta\_value: 当使用了meta\_value参数时，get\_post的结果只会显示有meta\_value的页面。
* authors: 用逗号分隔的列表作者ID列表。
* child\_of: 仅获取指定ID的子页面和孙页面。
* parent: 仅获取指定ID为父页面的页面，并且需要设置hierarchical为0，才能正常的返回结果。
* exclude\_tree：用逗号分隔的列表、页面ID数组，用来排除get\_post获取子页面列表结果
* number: 获取页面的数量
* offset: 跳过从顶部的页面数。
* post\_type: 查询的类型，默认是page。（页面）
* post\_status: 用逗号分隔的文章状态列表，使get\_post获取到指定文章状态的列表。

### Checking Whether the Query Returns Posts: have\_posts()

判断查询是否返回了文章：have\_posts()

Without accepting any parameters, this function simply returns TRUE if the query returns any records and FALSE if not.

不需要任何参数，如果是查询到了任何结果，这个函数返回true，否则的话false.

<?php

if ( have\_posts() ) {

    // Success.

} else {

    // Failure.

}

?>

### Working the Loop: the\_post()

[The Codex says](http://codex.wordpress.org/Function_Reference/the_post) that it "iterates the post index in the Loop". It does several things:

1. It fetches the next record from the query.
2. It sets up the $post data.
3. It sets the in\_the\_loop parameter to TRUE.

运行loop循环：the\_post()

官方文件：the\_post()重置了loop循环中的文章索引，它主要做了以下几件事：

1. 获取下一个查询结果。
2. 建立$post数据
3. 设置in\_the\_loop参数为TRUE.

<?php

if ( have\_posts() ) {

    while ( have\_posts() ) {

        the\_post();

        the\_title();

        the\_content();

    }

}

?>

### Setting Up $post: setup\_postdata()

建立$post数据：setup\_postdata()

This function also speaks for itself: It sets up the global post data. Let's see what the Codex says about this one:

同样，从字面义上可以理解这个函数：它建立了全局的文章数据。让我们来看看官方文档是怎么说的吧：

setup\_postdata() fills the global variables $id, $authordata, $currentday, $currentmonth, $page, $pages, $multipage, $more, $numpages, which help many Template Tags work in the current post context. It does not assign the global $post variable, but seems to expect that its argument is a reference to it.

setup\_postdata()给$id, $authordata, $currentday, $currentmonth, $page, $pages, $multipage, $more, $numpages进行了赋值，从而使许多模板标签能够使用在当前的文章内容中。它并没有直接给参数$post进行赋值，但是从某种角度来说，是$post参数的一个参考。

<?php

global $post;

setup\_postdata( $post );

?>

### Clearing the Current Loop: rewind\_posts()

Yet another function whose name states its functionality: This function simply "rewinds" the Loop so that you can run it again later.

清除当前的loop循环：rewind\_posts()

另一个可以从字面上知道其功能的函数是rewind\_posts()，完成了“倒回”循环的功能，所以在接下来可以重新运行它。

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <?php    // After the loop.  rewind\_posts();    ?> |

### Resetting $post: wp\_reset\_postdata()

This function resets the global $post variable back to the first post in the main query. It's better to use this one after a secondary query.

重置$post：wp\_reset\_postdata()

此函数重置了主查询中的全局变量$post，推荐在第二次查询之前使用它。

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <?php    // After a secondary query.  wp\_reset\_postdata();    ?> |

### Checking Whether Current Query Is the Main Query: is\_main\_query()

This one is a [Conditional Tag](http://codex.wordpress.org/Conditional_Tags) that returns TRUE if the current query is the main query and FALSE if it's not. Simple, right?

判断是否当前查询为主查询：is\_main\_query()

这是一个条件标签，如果当前的查询是主查询则返回true，否则返回false。简单吧？

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13 | <?php    if ( is\_main\_query() ) {        // Success.    } else {        // Failure.    }    ?> |

### Checking Whether We're in the Loop: in\_the\_loop()

Another Conditional Tag is in\_the\_loop() that simply returns TRUE or FALSE if your code is running inside the Loop or not.

判断是否在loop循环中：in\_the\_loop()

另一个条件标签是in\_the\_loop()，如果在loop循环在，则返回true，否则返回false.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13 | <?php    if ( in\_the\_loop() ) {        // Success.    } else {        // Failure.    }    ?> |

## End of Part Two

There you go—now you know (probably) all the functions related to WP\_Query! Stay tuned for the next part where we'll learn about WP\_Query related actions and filters.

Do you have any comments or anything to add to this part? Feel free to share your thoughts by commenting below. And if you liked the article, don't forget to share it with your friends!

第二部分完

现在我们了解到了与wp\_query相关的（几乎）所有函数！下面的教程中，我们会介绍WP\_Query相关的行动器（actions）和过滤器（filters）。

如果对本篇内容有任何的疑问，可以在下方留言！如果你喜欢的话，可以分享到你的朋友圈里！

译者：痴空见观 链接：wayearn.com