# [Mastering WP\_Query: Actions and Filters](http://code.tutsplus.com/tutorials/mastering-wp_query-actions-and-filters--cms-23112)

# 掌握WP\_Query：行动器和过滤器

Welcome to third part of our series called "Mastering WP\_Query". In the previous part, we went through 13 WordPress functions that are related to the WP\_Query class. In this part, we will be reviewing WordPress hooks (filters and actions) that you can use in conjunction with WP\_Query.

So, without further ado, let's begin!

欢迎来到掌握WP\_Query系列的第三部分。在先前的部分，我们学习了wordpress的13个与WP\_Query类相关的函数。在这个部分，我们将学习到与WP\_Query相关的WordPress的钩子（行动器和过滤器）。

## WP\_Query-Related Filters

In the "Filters Reference" of the Codex, [there are 16 WordPress filters](http://codex.wordpress.org/Plugin_API/Filter_Reference#WP_Query_Filters) listed under the title "WP\_Query Filters". In this section, we're going to talk about them. It will be a fairly quick section because most of the filters are about different "clauses" of a database query made by the WP\_Query class.

We'll get back to them, right after going through the filters that are not related to the clauses.

WP\_Query相关的过滤器

在过滤器“官方参考”中，有16个与WP\_Query相关的WordPress过滤器。在这个部分中，我们将要学习它们。这是个很简短的部分，因为所有的过滤器都是WP\_Query类不同的数据库查询操作。

我们将会在之后学习数据库的一些查询操作，。

### Filtering the Number of Found Posts for the Query:found\_posts

This filter allows you to change the number of found items without the limit that comes from WP\_Query's posts\_per\_page argument.

过滤器1：过滤Query查询找到的文章数量found\_posts

这个过滤器可以无限制的改变由WP\_Query的posts\_per\_page变量而找到的项目数量。

This filter is particularly useful for creating custom pagination. Let's see [what the Codex says](http://codex.wordpress.org/Plugin_API/Filter_Reference/found_posts):

*For instance, if you are declaring a custom offset value in your queries, WordPress will NOT deduct the offset from the the $wp\_query->found\_posts parameter (for example, if you have 45 usable posts after an offset of 10, WordPress will ignore the offset and still give found\_posts a value of 55).*

Makes sense.

这个过滤器可以用来创立用户自定义的页码。让我们看看官方文档怎么说：

例如，如果你想在查询中重新自定义值，通过$wp\_query->found\_posts 参数，WordPress不会修改这个定义值（就比如说，你有45篇可用的文章当你设置这个参考值为10篇，WordPress会忽略参考值，仍然会给你找到55篇文章。）

多读几遍，理解一下吧。

### Filtering the Query to Run for Retrieving the Found Posts: found\_posts\_query

The number of found items is calculated with the SELECT FOUND\_ROWS() command in MySQL. This filter lets you change this command to something else, in case you need to calculate the number of found items in a different way.

过滤器2：当Query查询去检索寻找文章的时候，过滤Query查询found\_posts\_querys

查询结果是MySQL的 SELECT FOUND\_ROWS() 命令计算出来的。这个过滤器可以去改变这个命令，当需要用一种不同的方式去计算查询项目的数量时候，这个过滤器非常的有用。

### Filtering the Entire SQL Query: posts\_request

If you want to change the completed SQL query, this is the filter you should use. It basically overrides the whole SQL query composed by the WP\_Query class.

过滤器3：过滤整个SQL的Query查询posts\_request

如果需要改变整个SQL的query查询，这个过滤器是非常好的帮手，它可以很简单的重写整个由WP\_Query类生成的SQL的query查询。

### Filtering the Array the Query Returns: posts\_results

If you want to alter the PHP array that WP\_Query generates with the SQL query, you can use this filter. There's [a neat example over on the Codex](http://codex.wordpress.org/Plugin_API/Filter_Reference/posts_results) so you can see how it works.

(Please note that this filter handles the raw array that's generated from the SQL query.)

过滤器3：过滤由Query查询返回的结果posts\_results

如果需要改变由WP\_Query的SQL语言产生的PHP的数组（这组数据：后称文章列表），可以使用posts\_results这个过滤器。这里有一个[官方的例子](http://codex.wordpress.org/Plugin_API/Filter_Reference/posts_results)，可以看看它是怎么工作的。

### Filter the Array of Retrieved Posts: the\_posts

Unlike posts\_results, the the\_posts filter waits for the array to be internally processed, meaning that the array would be checked against unpublished and sticky posts. Use this one if you don't want unpublished or posts in your array.

过滤器4：检索文章列表的时候进行过滤the\_posts

不像posts\_results，the\_posts过滤器等到检索文章列表，即将操作检索的时候，进行过滤——意思是这个文章列表会被再次与未发表的和置顶的文章进行核对。使用这个过滤器的情景：当你不想在你的文章列表中出现未发表的文章的时候。

### Filtering the Field List (and the SELECT Clause) of the Query: posts\_fields

The SELECT clause of an SQL query determines which database fields will be selected from the resulting database rows, and this filter helps you filter it.

过滤器5：过滤Query查询的（SELECT子句）的查询范围posts\_fields

SQL中的SELECT子句会决定数据中的哪个表或者范围被选择，可以使用posts\_fields过滤器来实现。

### Filtering the LIMIT Clause of the Query: post\_limits

The LIMIT clause of an SQL query sets the limitations to the query, and this filter helps you filter it.

过滤器6：过滤查询的LIMIT子句post\_limits

SQL查询的LIMIT子句将限制查询，该过滤器可以帮助你过滤。

### Filtering the DISTINCT Clause of the Query: posts\_distinct

The DISTINCT clause of an SQL query states that the query should return only different results, and this filter helps you filter it. Naturally, WP\_Query does not return distinct results but when you use this filter with a function that returns "DISTINCT", the query will be adjusted to return only different results.

过滤器7：过滤查询的DISTINCT子句posts\_distinct

SQL查询的DISTINCT子句指出查询应该只返回不同的结果，而这种过滤器可以帮助你过滤。

当然，WP\_Query不返回不同的结果，但是当你使用此过滤器，它会返回像“DISTINCT”功能的结果，查询将被调整为只返回不同的结果。

### Filtering the WHERE Clause of the Query: posts\_where

The WHERE clause of an SQL query is used to filter MySQL's SELECT, INSERT,UPDATE or DELETE statements, and this filter helps you filter this filter. While theWP\_Query class does all the work needed to filter the results, you can take it a step further by using this filter.

过滤器8：过滤查询的WHERE子句posts\_where

SQL查询中的WHERE子句是用来过滤MySQL的SELECT,INSERT,UPDATE或者DELETE状态的，这个过滤器可以实现过滤WHERE子句。当WP\_Query类实现了所有所需要的过滤结果工作，依旧可以在WP\_Query之前使用这个过滤器来实现过滤功能。

### Filtering the WHERE Clause of the Query After the Paging Is Calculated: posts\_where\_paged

This filter is an iteration to the posts\_where filter, which you can use with paging queries.

过滤器9：在计算分页之后，过滤查询的WHERE子句posts\_where\_paged

这个过滤器其实就是posts\_where过滤器的一个反复过程，可以用在分页查询中。

### Filtering the WHERE Clause of a Search Query: posts\_search

Another iteration to the posts\_where filter is this filter, which you can use to alter theWHERE clause of a query that's used to get search results in WordPress.

过滤器10：过滤搜索查询的WHERE子句posts\_search

另一种迭代的posts\_where过滤器的方法——posts\_search过滤器，可以用其来改变（用来获取WordPress的搜索结果）查询的WHERE子句。

### Filtering the JOIN Clause of the Query: posts\_join

The JOIN clause of an SQL query allows you to work your SQL command within multiple database tables, and this filter helps you filter it. This is one of the advanced parts of MySQL, so I don't recommend using this filter unless you know very well how MySQL JOINs work.

过滤器11：过滤查询的JOIN子句posts\_join

查询的JOIN子句允许SQL语句在多个数据库表内的联合工作，该过滤器可以帮助你过滤JOIN子句。JOIN是MySQL的比较牛B的地方，除非你知道MySQL的JOIN工作原理，不然的话，我不建议使用这种过滤器，

### Filtering the JOIN Clause of the Query After the Paging Is Calculated: posts\_join\_paged

Just as posts\_where\_paged is an iteration of posts\_where, this is an iteration to theposts\_join filter that works on paging queries.

过滤器12：当分页被计算，过滤查询的JOIN子句posts\_join\_paged

就像posts\_where\_paged过滤器去迭代posts\_where过滤器一样，posts\_join\_paged过滤器是当在进行分页查询时迭代psots\_join过滤器。

### Filtering the ORDER BY Clause of the Query: posts\_orderby

The ORDER BY clause of an SQL query arranges the ordering of the query, and this filter helps you filter the order.

过滤器13：过滤查询的OREDER BY子句posts\_orderby

ORDER BY子句是SQL查询用来对查询结果进行排序的，这个过滤器是用来过滤排序的。

### Filtering the GROUP BY Clause of the Query: posts\_groupby

The GROUP BY clause of an SQL query makes the query return "grouped" results by a database field, and this filter helps you filter how to group the results.

过滤器14：过滤查询的GROUP BY子句posts\_groupby

Group BY子句使SQL查询返回一个数据库字段的“分组”结果，该过滤器可以帮助你过滤如何分组的结果。

### Filtering All Clauses of the Query: posts\_clauses

If you want to deal with all clauses at the same time, there's a filter for that, too:posts\_clauses. This filter covers the WHERE, GROUP BY, JOIN, ORDER BY, DISTINCT,SELECT, and LIMITS clauses.

过滤器15：过滤所有的查询子句posts\_clauses

如果想要在同一时间去处理所有的子句，这里有一个过滤器可以实现posts\_clases。这个过滤器涵盖了WHERE,GROUP,BY,JOIN,ORDER BY,DISTINCT,SELECT和LIMITS子句。

## WP\_Query-Related Actions

Now we've reviewed the filters related to WP\_Query, let's move on to the other kind of hooks: actions.

WP\_Query相关的行动器

现在，我们已经学习了与WP\_Query相关的过滤器，让我们继续学习另一种挂钩：行动器。

### Interfere With the Query Before It's Run: pre\_get\_posts

Before the query is parsed, you can interact with it (e.g. inject additional query variables) by using this action. Let's see [a quick example from Tom McFarlin](https://tommcfarlin.com/exclude-categories-from-wordpress-loop/) to learn how to exclude a category from the main loop:

行动器1：在运行查询之前去插入pre\_get\_posts

在查询解释之前，可以通过这个行动器与其进行一些交互（比如：插入一个附加的查询变量）。让我们看一个[简单的例子](https://tommcfarlin.com/exclude-categories-from-wordpress-loop/)，学习如何从主循环中排除分类。

<?php

function tutsplus\_exclude\_category( $wp\_query ) {

    /\*

     \*类别添加到排除的类别数组中，在这种情况下，虽然只有一个

     \*

     \*/

    $excluded = array( '-1' );

    /\*

     \*请注意，这是一个不同的或许更简单的编写方式：     \*

     \* $wp\_query->set( 'category\_\_not\_in', $excluded );

     \*/

    set\_query\_var( 'category\_\_not\_in', $excluded );

}

add\_action( 'pre\_get\_posts', 'tutsplus\_exclude\_category' );

?>

### Handle the Parsing of the Query: parse\_query

Unlike pre\_get\_posts, which intervenes with the query before the query variables are set, the parse\_query action handles the process after the query variables are set. Thus, you should use this action if you want to check the current variables and take action in accordance with them.

行动器2：查询在解析的时候，进行干预parse\_query

不像pre\_get\_posts那样，这个行动器会在查询设置查询变量之前进行干预，parse\_query行动器会控制查询变量设置之后的进程。因此，当需要去检查当前的变量，并需要一些附加的操作的时候，使用这个行动器

### Modify the Post Object: the\_post

The term the\_action is a little bit confusing because it's the name of an action hook, a WP\_Query-related function and also a method of the WP\_Query class.

行动器3：修改文章对象the\_post

the\_actions这个标签有一点点的让人搞不清楚头脑，因为这个名字其实是一个行动器钩子，是一个WP\_Query相关函数，也是一个WP\_Query类的方法。

This action, as the Codex says, lets us modify the post object immediately after being queried and set up. Using this action, you can change the output directly. Let's see a quick example:

这个行动器，按照文档的说法，可以在查询和获取结果之后，立即修改文章对象。使用这个行动器，可以轻松的改变输出结果。来看一个例子吧：

<?php

function tutsplus\_featured\_badge( $post ) {

    if ( is\_single() && in\_category( 'featured' ) ) {

        echo '<div class="featured-badge">' . \_\_( 'FEATURED', 'tutsplus' ) . '</div>';

    }

}

add\_action( 'the\_post', 'tutsplus\_featured\_badge' );

?>

## End of Part Three

Actions and filters are always a fun topic to talk and write about. (I know for a fact that my two series about [WordPress actions](http://code.tutsplus.com/series/fifty-actions-of-wordpress--cms-708) and [WordPress filters](http://code.tutsplus.com/series/50-filters-of-wordpress--cms-694) were fun to write and got a great reaction from our reader base.) I hope you enjoyed this part as much as I enjoyed writing it.

Want to add anything to the article? Shoot a comment in the comments section below. And if you liked the article, don't forget to share it with your friends!

See you in the next part, where we'll be talking about the properties and methods of the WP\_Query class!

第三部分完

过滤器和行动器一直是一个很有趣的大有可谈的部分（我可以从我的两个系列WordPress行动器和WordPress过滤器读者的反馈看出来）。希望大家能像我去享受谈这个部分一样，去喜欢这个部分的文章。

你想说些什么吗？请在文章下方留言。如果你喜欢这篇文章，请分享吧！

在下一个部分，我们会谈到WP\_Query类的属性和方法。