



The Handbook

Codex

[http://codex.wordpress.org/Using Custom Fields](http://codex.wordpress.org/Using_Custom_Fields)

Version Date

8 August 2005

Using Custom Fields

WordPress has the ability to allow post authors to assign custom fields to a post. This arbitrary extra information is known as meta-data. This meta-data can include bits of information such as:

- **Mood: Happy**
- **Currently Reading: Cinderella**
- **Listening To: Rock Around the Clock**
- **Weather: Hot and humid**

With some extra coding, it is possible to achieve more complex actions, such as using the **metadata** to store an expiration date for a post.

Meta-data is handled with **key/value** pairs. The **key** is the name of the meta-data element. The **value** is the information that will appear in the meta-data list on each individual post that the information is associated with.

Keys can be used more than once per post. For example, if you were reading two different books (perhaps a technical book at work and a fiction at home), you could create a "reading" key and use it twice on the same post, once for each book.

Here is an example of what this information might look like on your post:

Currently Reading: *Calvin and Hobbes*

Today's Mood: *Jolly and Happy*

Usage

Based upon our example above, let's put this into action. We'll add two custom fields, one called "Currently Reading" and the other "Today's Mood". The following instructions will demonstrate how to add this information to a post using Custom Fields.

1. From the [Write Post](#) panel, choose **Advanced Editing**. If you are using the **Simple Editing** screen, look for a button with **Advanced Editing »** next to the **Publish** button. Click the button to go to the advanced editing screen.
2. After you have written your post, scroll down to the bottom of the **Advanced Editing** screen and look for an area titled **Custom Fields**.
3. To create a new **Custom Field** called "Currently Reading", enter the text "Currently Reading" (without the quotes) in the text entry field titled **Key**.
4. The newly created **Key** should now be assigned a **Value**, which in our case is the name of the book currently being read, "Calvin and Hobbes". Type "Calvin and

Hobbes" in the *Value* field, again without the quotes.

5. Click **Add Custom Field** button to save this custom information for that post.

To add your "Today's Mood", repeat the process and add "Today's Mood" to the **key** and a description of your mood in the **value** text boxes and click **SAVE** to save this information with the post.

On your next post, you can add a new book and mood to your meta-data. In the **Custom Fields** section, the **Key** will now feature a pull down list with the previously entered Custom Fields. Choose "Currently Reading" and then enter the new book you are reading in the **value**. Click **Add Custom Field** and then repeat the process to add "Today's Mood".

You only need to create a new "KEY" **once**, after which you can assign a value to that key for every post, if you so desire. You can also assign more than one *Value* to a key, for a post. This will come in handy for people who read more than one book at a time.

Displaying Custom Fields

With a Custom Field added to the post, it's time to display your books and mood to the world. To display the Custom Fields for each post, use the `the_meta()` template tag. The tag must be put within [The Loop](#) in order to work. Many people add `the_meta()` template tag to the end of their post or in their [Post Meta Data Section](#). Here is a basic example of using the tag:

```
<?php the_meta(); ?>
```

It might look like this in the source code:

```
<ul class='post-meta'>
<li><span class='post-meta-key'>Curently Reading:</span> Calvin and Hobbes</li>
<li><span class='post-meta-key'>Today's Mood:</span> Jolly and Happy</li>
</ul>
```

The template tag automatically puts the entire meta-data into a CSS style called `post-meta`. The **key** is in a `span` called `post-meta-key` so you can style it in your style sheet. All of this is showcased in an unordered list.

To customize the look of the post-meta list, change the characteristics in your style sheet. For instance, let's add some style to our example from the top. The style sheet elements would look like this:

```
.post-meta {font-variant: small-caps; color: maroon; }
.post-meta-key {color: green; font-weight: bold; font-size: 110%; }
```

- **CURRENTLY READING:** CALVIN AND HOBBS
- **TODAY'S MOOD:** JOLLY AND HAPPY

There are also a [few plugins](#) that add some nice features to the job of displaying meta tags. A search for [Custom Field plugins at Google](#) (<http://www.google.com/search?hl=en&q=custom+fields+plugin+wordpress&>

btnG=Google+Search) should help you find even more.

Advanced Techniques for Custom Fields

The following are more advanced techniques for getting and customizing meta-data and custom fields.

Getting Custom Fields

To fetch meta values use the `get_post_meta()` function:

```
get_post_meta($post_id, $key, $single);
```

- `$post_id` is the ID of the post you want the meta values for. Use `$post->ID` to get a post's ID.
- `$key` is a string containing the name of the meta value you want.
- `$single` can either be `true` or `false`. If set to `true` then the function will return a single result, as a **string**. If `false`, or not set, then the function returns an **array** of the custom fields.

Implementation Details

The PostMeta information is stored in a new table, `$wpdb->postmeta`. This table has four fields:

```
meta_id: A unique id for each entry
post_id: The ID of the post for this metadata
meta_key: The name of the 'key'
meta_value: The value associated with the key
```

The values from this table are pulled into a structured multi-dimensional array called `$post_meta_cache`, just after the `$posts` array is fetched in `wp-blog-header.php`. This variable will only contain values for the list of posts fetched for the current page build. The structure of the array will look something like this:

```
[
  postid1 => [
    [
      key1 => [val1, val2, ...],
      key2 => [val1, val2, ...],
      ...
    ],
    postid2 => [ ... ],
    ...
  ]
```

So, if you wanted to fetch the "reading" values from post number 256, you use this PHP code:

```
// Fetch an array of values for what I'm reading:
$readinglist = $post_meta_cache[256]['reading'];
```

Don't forget that `$readinglist` will be an array, not a single value.

PostMeta Functions

Internal Functions

These functions are intended for use inside the wp-loop, and all return arrays.

`get_post_custom()`

Get all key/value data for the current post.

`get_post_custom_keys()`

Get a list of all key names for the current post.

`get_post_custom_values($key)`

Get the list of values for a particular key on the current post.

`get_post_meta($post_id, $key, $single = false)`

In WP 1.5 and beyond, this function returns the meta information without cache problems. The function requires the post id, the key, and if `$single` is set to TRUE, it returns only the first result (NOT as an array) for PHP use.

Template Functions

At the time of this writing, there is only one template function.

[`the_meta\(\)`](#)

Echoes an unordered list containing the current post's meta-data with a class for the UL as *post-meta* and the LI as *post-meta-key*.

We expect that independent developers will come up with many interesting uses for post meta-data in the form of plugins. The `the_meta()` template function is just an extremely basic example.

At this time, you can only add and delete entries. The ability to modify existing entries will be implemented later.