# Custom Meta Boxes

## What is a Meta Box?

When a user edits a post, the edit screen is composed of several default boxes; Editor, Publish, Categories, Tags, etc. These boxes are meta boxes. Plugins an add custom meta boxes to an edit screen of any post type.

A meta box is a feature in WordPress (and other CMS platforms) that allows developers to add custom fields, controls, and information to the post editing screen.

The primary goal of a meta box is to extend the default capabilities of WordPress. By default, WordPress provides fields like title, content, categories, tags, etc. Meta boxes let developers add extra fields and control tailored to specific needs.

Meta boxes usually appear on the right-hand side of the post editing screen in WordPress, below the main content area. They are typically draggable and collapsible, making them flexible in terms of screen real estate

Developers can populate meta boxes with various types of fields such as text boxes, checkboxes, radio buttons, dropdowns, date pickers, and even file uploaders. These fields can capture additional information related to a post, such as custom metadata, SEO settings, related post links, or any other specific data relevant to the content being edited.

Meta boxes are highly customizable. Developers can create their own meta boxes using PHP and WordPress functions. They define what fields to include, how they should behave (validation, default values, etc.), and where the meta box should appear (on which post types or screens).

## Why use Meta Boxes?

Meta boxes are handy, flexible, modular edit screen elements that can be used to collect information related to the post being edited. Your custom meta box will be on the same screen as all the other post related information; so a clear relationship is established.

Meta boxes are easily hidden from users that do not need to see them, and displayed to those that do. Meta boxes can be user-arranged on the edit screen. The users are free to arrange the edit screen in a way that suits them, giving users control over their editing environment.

## Adding Meta Boxes

To create a meta box use the ***add\_meta\_box()*** function and plug it’s execution to the ***add\_meta\_boxes*** action hook.

**function** wporg\_add\_custom\_box()

{

    $screens = ['post', 'wporg\_cpt'];

**foreach** ($screens **as** $screen) {

        add\_meta\_box(

            'wporg\_box\_id',           // Unique ID

            'Custom Meta Box Title',  // Box title

            'wporg\_custom\_box\_html',  // Content callback, must be of type callable

            $screen                   // Post type

        );

    }

}

add\_action('add\_meta\_boxes', 'wporg\_add\_custom\_box');

The ***wprog\_custom\_box\_html*** function will hold the HTML for the meta box.

The following example is adding form elements, labels, and other HTML elements.

**function** wporg\_custom\_box\_html($post)

{

    ?>

    <label **for**="wporg\_field">Description **for** this field</label>

    <select name="wporg\_field" id="wporg\_field" **class**="postbox">

        <option value="">Select something...</option>

        <option value="something">Something</option>

        <option value="else">Else</option>

    </select>

    <?php

}

The example shown here only contains one form field, a drop-down list. You may create as many as needed in any particular meta box. If you have a lot of fields to display, consider using multiple meta boxes, grouping similar fields together in each meta box. This helps keep the page more organized and visually appealing.

## Getting Values

To retrieve saved user data and make use of it, you need to get it from wherever you saved it initially. If it was stored in the **postmeta** table, you may get the data with ***get\_post\_meta().***

The following example enhances the previous form elements with pre-populated data based on saved meta box values.

**function** wporg\_custom\_box\_html($post)

{

    $value = get\_post\_meta($post->ID, '\_wporg\_meta\_key', true);

    ?>

    <label **for**="wporg\_field">Description **for** this field</label>

    <select name="wporg\_field" id="wporg\_field" **class**="postbox">

        <option value="">Select something...</option>

        <option value="something" <?php selected($value, 'something'); ?>>Something</option>

        <option value="else" <?php selected($value, 'else'); ?>>Else</option>

    </select>

    <?php

}

## Saving Values

When a post type is saved or updated, several actions fire, any of which might be appropriate to hook into in order to save the entered values. In this example we use the **save\_post** action hook but other hooks may be more appropriate for certain situations. Be aware that **save\_post** may fire more than once for a single update event. Structure your approach to saving data accordingly.

You may save the entered data anywhere you want, even outside WordPress. Since you are probably dealing with data related to the post, the **postmeta**  table is often a good place to store data.

The following example will save the ***wprog\_field*** field value in the ­***\_wprog\_meta\_key*** meta key, which is hidden.

**function** wporg\_save\_postdata($post\_id)

{

**if** (array\_key\_exists('wporg\_field', $\_POST)) {

        update\_post\_meta(

            $post\_id,

            '\_wporg\_meta\_key',

            $\_POST['wporg\_field']

        );

    }

}

add\_action('save\_post', 'wporg\_save\_postdata');

## OOP

Adding meta boxes using OOP is easy and saves you from having to worry about naming collisions in the global namespace.

To save memory and allow easier implementation, the following example uses an abstract Class with static methods.

**abstract** **class** WPOrg\_Meta\_Box

{

**public** **static** **function** add()

    {

        $screens = ['post', 'wporg\_cpt'];

**foreach** ($screens **as** $screen) {

            add\_meta\_box(

                'wporg\_box\_id',          // Unique ID

                'Custom Meta Box Title', // Box title

                [self::**class**, 'html'],   // Content callback, must be of type callable

                $screen                  // Post type

            );

        }

    }

**public** **static** **function** save($post\_id)

    {

**if** (array\_key\_exists('wporg\_field', $\_POST)) {

            update\_post\_meta(

                $post\_id,

                '\_wporg\_meta\_key',

                $\_POST['wporg\_field']

            );

        }

    }

**public** **static** **function** html($post)

    {

        $value = get\_post\_meta($post->ID, '\_wporg\_meta\_key', true);

        ?>

        <label **for**="wporg\_field">Description **for** this field</label>

        <select name="wporg\_field" id="wporg\_field" **class**="postbox">

            <option value="">Select something...</option>

            <option value="something" <?php selected($value, 'something'); ?>>Something</option>

            <option value="else" <?php selected($value, 'else'); ?>>Else</option>

        </select>

        <?php

    }

}

add\_action('add\_meta\_boxes', ['WPOrg\_Meta\_Box', 'add']);

add\_action('save\_post', ['WPOrg\_Meta\_Box', 'save']);

## Example of custom meta box for movies

function add\_movies\_metabox(){

    add\_meta\_box(

        'movies\_meta\_box',

        'Movies',

        'movies\_meta\_box\_html',

        'post'

    );

}

add\_action('add\_meta\_boxes','add\_movies\_metabox');

function movies\_meta\_box\_html($post){

    $terms = get\_terms(array('taxonomy' => 'movie', 'hide\_empty' => false));

    $post\_terms = get\_the\_terms($post->ID,'movie');

    $post\_term\_ids = $post\_terms ? wp\_list\_pluck($post\_terms,'term\_id') : array();

    //Display checkboxes for each terms

    foreach($terms as $term){

        echo '<label>';

        echo '<input type="checkbox" name="movie[]" value="' . esc\_attr($term->term\_id) . '" ' . checked(in\_array($term->term\_id, $post\_term\_ids), true, false) . '>';

        echo esc\_html($term->name);

        echo '</label><br>';

    }

}

function save\_movies\_meta\_box($post\_id){

    if(isset($\_POST['movie'])){

        $movie\_terms = array\_map('intval', $\_POST['movie']); // sanitize as integers

        wp\_set\_post\_terms($post\_id,$movie\_terms,'movie');

    }else{

        wp\_set\_post\_terms($post\_id,[],'movie');

    }

}

add\_action('save\_post','save\_movies\_meta\_box');