



```
<?php
/*
Plugin Name: WP Comment Notes
Plugin URI: http://andrewnorcross.com/plugins/
Description: Add custom notes before or after the comment form.
Version: 1.0.0
Author: Andrew Norcross
Author URI: http://andrewnorcross.com

[ ] Copyright 2013 Andrew Norcross

[ ] This program is free software; you can redistribute it and/or modify
[ ] it under the terms of the GNU General Public License, version 2, as
[ ] published by the Free Software Foundation.

[ ] This program is distributed in the hope that it will be useful,
[ ] but WITHOUT ANY WARRANTY; without even the implied warranty of
[ ] MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
[ ] GNU General Public License for more details.

[ ] You should have received a copy of the GNU General Public License
[ ] along with this program; if not, write to the Free Software
[ ] Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
*/

if( !defined( 'WPCMN_VER' ) )
[ ] define( 'WPCMN_VER', '1.0.0' );

// Start up the engine
class WP_Comment_Notes
{

[ ] /**
[ ]  * Static property to hold our singleton instance
[ ]  */
```

```

    */
    static $instance = false;

    /**
     * This is our constructor
     *
     * @return void
     */
    private function __construct() {
        // back end
        add_action( 'plugins_loaded', array( $this, 'textdomain' ) );
        add_action( 'admin_enqueue_scripts', array( $this, 'admin_scripts' ) );
        add_action( 'do_meta_boxes', array( $this, 'create_metaboxes' ), 10, 2 );
        add_action( 'save_post', array( $this, 'save_custom_meta' ), 1 );

        // front end
        add_action( 'wp_enqueue_scripts', array( $this, 'front_scripts' ), 10 );
        add_filter( 'comment_form_defaults', array( $this, 'custom_notes_filter' ) );
    }

    /**
     * If an instance exists, this returns it. If not, it creates one and
     * returns it.
     *
     * @return WP_Comment_Notes
     */

    public static function getInstance() {
        if ( !self::$instance )
            self::$instance = new self;
        return self::$instance;
    }

    /**
     * load textdomain
     *
     * @return void
     */

    public function textdomain() {

```

```

load_plugin_textdomain( 'wpcmn', false, dirname( plugin_basename( __FILE__ ) ) . '/languages/' );

}

/**
 * Admin styles
 *
 * @return void
 */

public function admin_scripts() {

    $types = $this->get_post_types();

    $screen = get_current_screen();

    if ( in_array( $screen->post_type , $types ) ) :

        wp_enqueue_style( 'wpcmn-admin', plugins_url('lib/css/admin.css', __FILE__), array(), WPCMN_VER, 'all' );

    endif;

}

/**
 * call metabox
 *
 * @return void
 */

public function create_metaboxes( $page, $context ) {

    $types = $this->get_post_types();

    if ( in_array( $page, $types ) )
        add_meta_box( 'wp-comment-notes', __( 'Comment Notes', 'wpcmn' ), array( $this, 'wpcmn_notes_meta' ),
            $page, 'advanced', 'high' );

}

```

```

/**
 * display meta fields for notes meta
 *
 * @return void
 */

public function wpcmn_notes_meta( $post ) {

    // Use nonce for verification
    wp_nonce_field( 'wpcmn_meta_nonce', 'wpcmn_meta_nonce' );

    $post_id = $post->ID;

    // get postmeta, and our initial settings
    $notes = get_post_meta( $post_id, '_wpcmn_notes', true );

    $before_text = isset( $notes['before-text'] ) ? $notes['before-text'] : '';
    $before_type = isset( $notes['before-type'] ) ? $notes['before-type'] : 'wpcmn-notes-standard';
    $after_text = isset( $notes['after-text'] ) ? $notes['after-text'] : '';
    $after_type = isset( $notes['after-type'] ) ? $notes['after-type'] : 'wpcmn-notes-standard';

    echo '<script
type="text/javascript">jQuery(document).ready(function($){$("#comment_status").click(function(){$(".wpcmn-
notes-table tr").toggle();})});</script>';

    $disabled_display = comments_open( $post_id ) ? ' style="display:none;" : '';
    $enabled_display = ! comments_open( $post_id ) ? ' style="display:none;" : '';

    echo '<table class="form-table wpcmn-notes-table">';

    echo '<tr class="wpcmn-notes-disabled" . $disabled_display . '>';
    echo '<th>' . __( 'Enable comments in order to use Comment Notes', 'wpcmn' ) . '</th>';
    echo '</tr>';

    echo '<tr class="wpcmn-notes-title" . $enabled_display . '>';
    echo '<td colspan="2"><h5>' . __( 'Before Notes Area', 'wpcmn' ) . '</h5></td>';
    echo '</tr>';

    echo '<tr class="wpcmn-notes-data wpcmn-notes-before-text" . $enabled_display . '>';

```

```

        echo '<th>'.__( 'Message Text', 'wpcmn' ) . '</th>';
        echo '<td>';
        echo '<textarea class="widefat" name="wpcmn-notes[before-text]" id="wpcmn-before">'.esc_attr(
        $before_text ) . '</textarea>';
        echo '<p class="description">'.__( 'Note: This will not appear to users who are logged in.', 'wpcmn' ) . '</p>';
        echo '</td>';
        echo '</tr>';

        echo '<tr class="wpcmn-notes-data wpcmn-notes-before-type" . $enabled_display . '>';
        echo '<th>'.__( 'Message Type', 'wpcmn' ) . '</th>';
        echo '<td>';
        echo '<select id="wpcmn-before-type" name="wpcmn-notes[before-type]">';
        echo '<option value="wpcmn-notes-standard" . selected( $before_type, 'wpcmn-notes-standard', false ) . '>'.
        __( 'Standard', 'wpcmn' ) . '</option>';
        echo '<option value="wpcmn-notes-warning" . selected( $before_type, 'wpcmn-notes-warning', false ) . '>'.
        'Warning', 'wpcmn' ) . '</option>';
        echo '<option value="wpcmn-notes-alert" . selected( $before_type, 'wpcmn-notes-alert', false ) . '>'. __( 'Alert',
        'wpcmn' ) . '</option>';
        do_action( 'wpcmn_before_types', $before_type );
        echo '</select>';
        echo '</td>';
        echo '</tr>';

        echo '<tr class="wpcmn-notes-title" . $enabled_display . '>';
        echo '<td colspan="2"><h5>'.__( 'After Notes Area', 'wpcmn' ) . '</h5></td>';
        echo '</tr>';

        echo '<tr class="wpcmn-notes-data wpcmn-notes-after-text" . $enabled_display . '>';
        echo '<th>'.__( 'Message Text', 'wpcmn' ) . '</th>';
        echo '<td>';
        echo '<textarea class="widefat" name="wpcmn-notes[after-text]" id="wpcmn-after">'.esc_attr( $after_text
        '</textarea>';
        echo '</td>';
        echo '</tr>';

        echo '<tr class="wpcmn-notes-data wpcmn-notes-after-type" . $enabled_display . '>';
        echo '<th>'.__( 'Message Type', 'wpcmn' ) . '</th>';
        echo '<td>';
        echo '<select id="wpcmn-after-type" name="wpcmn-notes[after-type]">';
        echo '<option value="wpcmn-notes-standard" . selected( $after_type, 'wpcmn-notes-standard', false ) . '>'.

```

```

'Standard', 'wpcmn' ) . '</option>';

    echo '<option value="wpcmn-notes-warning" . selected( $after_type, 'wpcmn-notes-warning', false ) . '>' . _
'Warning', 'wpcmn' ) . '</option>';

    echo '<option value="wpcmn-notes-alert"' . selected( $after_type, 'wpcmn-notes-alert', false ) . '>' . __( 'Aler
'wpcmn' ) . '</option>';

    do_action( 'wpcmn_after_types', $after_type );

    echo '</select>';

    echo '</td>';

    echo '</tr>';

echo '</table>';

}

/**
 * save post metadata
 *
 * @return void
 */

public function save_custom_meta( $post_id ) {

    // make sure we aren't using autosave
    if ( defined( 'DOING_AUTOSAVE' ) && DOING_AUTOSAVE )
        return $post_id;

    // do our nonce check. ALWAYS A NONCE CHECK
    if ( ! isset( $_POST['wpcmn_meta_nonce'] ) || ! wp_verify_nonce( $_POST['wpcmn_meta_nonce'],
'wpcmn_meta_nonce' ) )
        return $post_id;

    $types = $this->get_post_types();

    if ( ! in_array ( $_POST['post_type'], $types ) )
        return $post_id;

    // and make sure the user has the ability to do shit
    if ( 'page' == $_POST['post_type'] ) {

        if ( ! current_user_can( 'edit_page', $post_id ) ) {

```

```

    return $post_id;
}

} else {

    if ( ! current_user_can( 'edit_post', $post_id ) ) {
        return $post_id;
    }

}

// all clear. get data via $_POST and store it
$notes = ! empty( $_POST['wpcmn-notes'] ) ? $_POST['wpcmn-notes'] : false;

// update side meta data
if ( $notes ) {

    $allowed_html = array(
        'a' => array(
            'href' => array(),
            'title' => array(),
            'class' => array(),
            'id' => array()
        ),
        'br' => array(),
        'em' => array(),
        'strong' => array(),
        'span' => array(
            'class' => array(),
            'id' => array()
        )
    );

    update_post_meta( $post_id, '_wpcmn_notes', wp_kses( $notes, $allowed_html ) );

    do_action( 'wpcmn_notes_save', $post_id, $notes );

} else {
    delete_post_meta( $post_id, '_wpcmn_notes' );
}

```

```

    }

    /**
     * call front-end CSS
     *
     * @return void
     */

    public function front_scripts() {

        // check for killswitch first
        $killswitch = apply_filters( 'wpcmn_killswitch', false );

        if ( $killswitch )
            return false;

        $types = $this->get_post_types();

        if ( is_singular( $types ) )
            wp_enqueue_style( 'wpcmn-notes', plugins_url( 'lib/css/wpcmn-notes.css', __FILE__ ), array(), WPCMN_VER, 'all' );

    }

    /**
     * The actual filter for adding the notes.
     *
     * @return array
     */

    public function custom_notes_filter( $fields ) {

        global $post;

        // get the possible meta fields
        $notes = get_post_meta( $post->ID, '_wpcmn_notes', true );

        if ( empty( $notes ) )
            return $fields;
    }

```



```

if ( isset( $notes['before-text'] ) ) :
    // grab the variables
    $text = $notes['before-text'];
    $class = isset( $notes['before-type'] ) ? $notes['before-type'] : 'wpcmn-notes-standard';
    // build the string

    $before = '<p class="wpcmn-notes wpcmn-notes-before" . esc_attr( $class ) . "'>' . $text . '</p>';

    // output
    $fields['comment_notes_before'] = $before;

endif;

if ( isset( $notes['after-text'] ) ) :
    // grab the variables
    $text = $notes['after-text'];
    $class = isset( $notes['after-type'] ) ? $notes['after-type'] : 'wpcmn-notes-standard';
    // build the string

    $after = '<p class="wpcmn-notes wpcmn-notes-after" . esc_attr( $class ) . "'>' . $text . '</p>';

    // output
    $fields['comment_notes_after'] = $after;

endif;

return $fields;
}

/**
 * Return the list of post types that support Comment Notes
 *
 * @return array
 */
public function get_post_types() {

    $types = get_post_types( array( 'public' => true, 'show_ui' => true ) );

```

```
foreach( $types as $type ) {
    if( ! post_type_supports( $type, 'comments' ) ) {
        unset( $types[ $type ] );
    }
}

return apply_filters( 'wpcmn_type_support', $types );
}

// end class
}

// Instantiate our class
$wp_comment_notes = WP_Comment_Notes::getInstance();
```

#1

Vcanson 26 2023 15:28:49

Vcanson 26 2023 15:34:10