```
<?php
/**
 * @version    4.2.0
 * @package    JEM
 * @copyright  (C) 2013-2023 joomlaeventmanager.net
 * @copyright  (C) 2005-2009 Christoph Lukes
 * @license    https://www.gnu.org/licenses/gpl-3.0 GNU/GPL
 */

defined('_JEXEC') or die;

use Joomla\CMS\Factory;
use Joomla\CMS\Filesystem\Path;
use Joomla\CMS\Filesystem\File;
use Joomla\CMS\Language\Text;
use Joomla\CMS\MVC\Model\AdminModel;
use Joomla\CMS\Component\ComponentHelper;
use Joomla\CMS\Client\ClientHelper;
use Joomla\CMS\Plugin\PluginHelper;

/**
 * Source Model
 */
class JemModelSource extends AdminModel
{
    /**
     * Cache for the template information.
     *
     * @var                    object
     */
    private $_template = null;

    /**
     * Method to auto-populate the model state.
     *
     * @Note Calling getState in this method will result in recursion.
     */
    protected function populateState()
    {
        $app = Factory::getApplication('administrator');

        // Load the User state.
        $id = $app->getUserState('com_jem.edit.source.id');

        // Parse the template id out of the compound reference.
        $temp = $id ? (base64_decode($id)) : $id;
        $fileName = $temp;

        if(!empty($fileName))
        {
                  $this->setState('filename', $fileName);
```

```
                // Save the syntax for later use
                $app->setUserState('editor.source.syntax', File::getExt($fileName));
        }
        // Load the parameters.
        $params  = ComponentHelper::getParams('com_jem');
        $this->setState('params', $params);
    }

    /**
     * Method to get the record form.
     *
     * @param  array   $data     Data for the form.
     * @param  boolean $loadData True if the form is to load its own data (default case), false if not.
     * @return JForm   A JForm object on success, false on failure
     */
    public function getForm($data = array(), $loadData = true)
    {
        // Initialise variables.
        $app = Factory::getApplication();

        // Codemirror or Editor None should be enabled
        $db = Factory::getContainer()->get('DatabaseDriver');
        $query = $db->getQuery(true);
        $query->select('COUNT(*)');
        $query->from('#__extensions as a');
        $query->where('((a.name ='.$db->quote('plg_editors_codemirror').' AND a.enabled = 1) OR
(a.name ='.$db->quote('plg_editors_none').' AND a.enabled = 1))');
        $db->setQuery($query);
        $state = $db->loadResult();
        if ((int)$state < 1 ) {
            $app-
>enqueueMessage(Text::_('COM_JEM_CSSMANAGER_ERROR_EDITOR_DISABLED'),
'warning');
        }

        // Get the form.
        $form = $this->loadForm('com_jem.source', 'source', array('control' => 'jform', 'load_data' =>
$loadData));
        if (empty($form)) {
            return false;
        }

        return $form;
    }

    /**
     * Method to get the data that should be injected in the form.
     *
     * @return mixed The data for the form.
     */
    protected function loadFormData()
    {
        // Check the session for previously entered form data.
        $data = Factory::getApplication()->getUserState('com_jem.edit.source.data', array());

        if (empty($data)) {
            $data = $this->getSource();
        }

        return $data;
    }

    /**
     * Method to get a single record.
     *
     * @return mixed Object on success, false on failure.
     */
    public function getSource()
    {
        $fileName = $this->getState('filename');

        $custom   = $fileName ? stripos($fileName, 'custom#:') : false;

        # custom file?
        if ($custom !== false) {
            $file = str_replace('custom#:', '', $fileName);
            $filePath = Path::clean(JPATH_ROOT . '/media/com_jem/css/custom/' . $file);
        } else {
            $file = $fileName;
            $filePath = Path::clean(JPATH_ROOT . '/media/com_jem/css/' . $file);
        }

        $item = new stdClass;
        if(file_exists($filePath)){
                            if ($file) {
                        $item->custom   = $custom !== false;
                        $item->filename = $file;
                        $item->source   = file_get_contents($filePath);
                } else {
                        $item->custom   = false;
                        $item->filename = false;
                        $item->source   = false;
                }
        }else{
            $this-
>setError(Text::_('COM_JEM_CSSMANAGER_ERROR_SOURCE_FILE_NOT_FOUND'));
        }

        return $item;
    }

    /**
     * Method to store the source file contents.
     *
     * @param  array   The souce data to save.
     *
     * @return boolean True on success, false otherwise and internal error set.
     */
    public function save($data)
    {
        $dispatcher = JemFactory::getDispatcher();
        $fileName   = $this->getState('filename');
        $custom     = $fileName  ? stripos($fileName, 'custom#:') : false;

        # custom file?
        if ($custom !== false) {
            $file = str_replace('custom#:', '', $fileName);
            $filePath = Path::clean(JPATH_ROOT . '/media/com_jem/css/custom/' . $file);
        } else {
            $file = $fileName;
            $filePath = Path::clean(JPATH_ROOT . '/media/com_jem/css/' . $file);
        }

        // Include the extension plugins for the save events.
        PluginHelper::importPlugin('extension');

        // Set FTP credentials, if given.
        ClientHelper::setCredentialsFromRequest('ftp');
        $ftp = ClientHelper::getCredentials('ftp');

        // Trigger the onExtensionBeforeSave event.
        $result = $dispatcher->triggerEvent('onExtensionBeforeSave', array('com_jem.source', $data,
false));
        if (in_array(false, $result, true)) {
            $this-
>setError(Text::sprintf('COM_JEM_CSSMANAGER_ERROR_FAILED_TO_SAVE_FILENAME'
, $file));
            return false;
        }

        // Try to make the template file writeable.
        if (!$ftp['enabled'] && Path::isOwner($filePath) && !Path::setPermissions($filePath, '0644'))
{
            $this-
>setError(Text::_('COM_JEM_CSSMANAGER_ERROR_SOURCE_FILE_NOT_WRITABLE'));
            return false;
        }

        $return = File::write($filePath, $data['source']);
        // But report save error with higher priority
        if (!$return) {
            $this-
>setError(Text::sprintf('COM_JEM_CSSMANAGER_ERROR_FAILED_TO_SAVE_FILENAME'
, $file));
            return false;
        }
        // Try to make the custom template file read-only again.
        if (!$ftp['enabled'] && Path::isOwner($filePath) && !Path::setPermissions($filePath, '0444'))
{
            $this-
>setError(Text::_('COM_JEM_CSSMANAGER_ERROR_SOURCE_FILE_NOT_UNWRITABLE'
));
            return false;
        }

        return true;
    }
}
```