Codeinwp / visualizer Public

Code Issues 36 Pull requests 1 Actions Projects Security ...

master

visualizer / classes / Visualizer / Source / Csv.php / Jump to

contactashish13 Better error messages for invalid CSVs History

2 contributors

213 lines (191 sloc) | 6.4 KB

```php
<?php

// +----------------------------------------------------------------------+
// | Copyright 2013  Madpixels  (email : visualizer@madpixels.net)        |
// +----------------------------------------------------------------------+
// | 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                                                    |
// +----------------------------------------------------------------------+
// | Author: Eugene Manuilov <eugene@manuilov.org>                        |
// +----------------------------------------------------------------------+
/**
 * Source manager for local CSV files.
 *
 * @category Visualizer
 * @package Source
 *
 * @since 1.0.0
 */
```

```
class Visualizer_Source_Csv extends Visualizer_Source {

	/**
	 * The path to the file with data.
	 *
	 * @since 1.0.0
	 *
	 * @access protected
	 * @var string
	 */
	protected $_filename;

	/**
	 * Constructor.
	 *
	 * @since 1.0.0
	 *
	 * @access public
	 * @param string $filename The path to the file.
	 */
	public function __construct( $filename = null ) {
		$this->_filename = trim( $filename );
	}

	/**
	 * Fetches series information.
	 *
	 * @since 1.0.0
	 *
	 * @access private
	 * @param resource $handle The file handle resource.
	 */
	private function _fetchSeries( &$handle ) {
		// read column titles
		$labels = fgetcsv( $handle, 0, VISUALIZER_CSV_DELIMITER, VISUALIZER_CSV_ENCLOSURE
		$types = null;

		if ( false !== strpos( $this->_filename, 'tqx=out:csv' ) ) {
			$attributes = $this->_fetchSeriesForGoogleQueryLanguage( $labels );
			if ( ! $attributes['abort'] ) {
				$labels = $attributes['labels'];
				$types = $attributes['types'];
			}
		}

		if ( is_null( $types ) ) {
			// read series types
			$types = fgetcsv( $handle, 0, VISUALIZER_CSV_DELIMITER, VISUALIZER_CSV_ENC
		}

		$labels = array_filter( $labels );
		$types = array_filter( $types );

		if ( ! $labels || ! $types ) {
			$this->_error = esc_html__( 'File should have a heading row (1st row) and
			return false;
		}

		$types = array_map( 'trim', $types );
		if ( ! self::_validateTypes( $types ) ) {
			$this->_error = esc_html__( 'Invalid data types detected in the data type
			return false;
		}

		for ( $i = 0, $len = count( $labels ); $i < $len; $i++ ) {
			$default_type = $i === 0 ? 'string' : 'number';

			$labels[ $i ] = $this->toUTF8( $labels[ $i ] );

			$this->_series[] = array(
				'label' => $labels[ $i ],
				'type'  => isset( $types[ $i ] ) ? $types[ $i ] : $default_type,
			);
		}

		return true;
	}

	/**
	 * Returns file handle to fetch data from.
	 *
	 * @since 1.4.2
	 *
	 * @access protected
	 * @param string $filename Optional file name to get handle. If omitted, $_filename is use
	 * @return resource File handle resource on success, otherwise FALSE.
	 */
	protected function _get_file_handle( $filename = false ) {
		// set line endings auto detect mode
		ini_set( 'auto_detect_line_endings', true );
		// open file and return handle
		return fopen( $filename ? $filename : $this->_filename, 'rb' );
	}

	/**
	 * Fetches information from source, parses it and builds series and data arrays.
	 *
	 * @since 1.0.0
	 *
	 * @access public
	 * @return boolean TRUE on success, otherwise FALSE.
	 */
	public function fetch() {
		// if filename is empty return false
		if ( empty( $this->_filename ) ) {
			$this->_error = esc_html__( 'No file provided. Please try again.', 'visual
			return false;
		}

		// read file and fill arrays
		$handle = $this->_get_file_handle();
		if ( $handle ) {
			// fetch series
			if ( ! $this->_fetchSeries( $handle ) ) {
				return false;
			}

			// fetch data
			// phpcs:ignore WordPress.CodeAnalysis.AssignmentInCondition.FoundInWhileC
			while ( ( $data = fgetcsv( $handle, 0, VISUALIZER_CSV_DELIMITER, VISUALIZE
				$this->_data[] = $this->_normalizeData( $data );
			}

			// close file handle
			fclose( $handle );
		}

		return true;
	}

	/**
	 * Returns source name.
	 *
	 * @since 1.0.0
	 *
	 * @access public
	 * @return string The name of source.
	 */
	public function getSourceName() {
		return __CLASS__;
	}

	/**
	 * Adds support for QueryLanguage https://developers.google.com/chart/interactive/docs/que
	 * where the user can provide something like /gviz/tq?tq=select%20A%2C%20B%20&tqx=out:csv
	 * to get the subset of data specified by the query
	 * this will conflate the heading and the type into one value viz. for heading XXX and typ
```

```php
	 * so we need to split them apart logically
	 * also the $types variable now contains the first row because the header and the type got
	 */
	private function _fetchSeriesForGoogleQueryLanguage( $labels, $types = array() ) {
		$new_labels = array();
		$new_types = array();
		$abort = false;
		foreach ( $labels as $label ) {
			// get the index of the last space
			$index = strrpos( $label, ' ' );
			if ( $index === false ) {
				// no space here? something has gone wrong; abort the entire proce
				$abort = true;
				break;
			}
			$type = trim( substr( $label, $index + 1 ) );
			if ( ! self::_validateTypes( array( $type ) ) ) {
				// some other data type? abort the entire process.
				$abort = true;
				break;
			}
			$label = substr( $label, 0, $index );
			$new_labels[] = $label;
			$new_types[] = $type;
		}
		if ( ! $abort ) {
			$labels = $new_labels;
			$types = $new_types;
		}

		return array(
			'abort' => $abort,
			'labels'    => $labels,
			'types' => $types,
		);
	}
}
```