Workflow Designer

custom filesource

15.03.2025

PDFMake, PDFGenerator and Flexx.Reports are implemented in this way!

If you create a similar module you could create a interfacefile to attach this files to Mails, Store as Documents or simple use the file in every task, which could handle files in any way.

Create a files in **modules/Workflow2/extends/interfaceFiles/** with the filename **<individual>.inc.php**.

This file must include a Class, which extends from **\Workflow\InterfaceFiles**.
You must implement the following structure to be recognized by Workflow Designer:

```
<?php
namespace Workflow\Plugins\InterfaceFiles;

class [ModuleName] extends \Workflow\InterfaceFiles {
    protected $title = '[ModuleName]';
    protected $key = '[SanitizedModuleName]';

    public function __construct() {
        if(!$this->isModuleActive()) {
            return;
        }
    }

    /**
     * @var String $moduleName The module of the current Workflow
     * @return array -  array(
     *                   'filekeyA' => 'File title of the first file from
this module',
     *                    ...
     *                  )
     *
     *      The file title will be shown in the task configurations.
     *      The filekey will be given to your _getFile and should
uniquely identify the file you should generate
     */
    protected function _getAvailableFiles($moduleName) {
        $return = array();
        if(!$this->isModuleActive()) {
            return $return;
        }
```



```
        /*
          This function must simple return an array, like this:
             array(
                'filekeyA' => 'File title of this file'
             )

        */

        return $return;
    }

    /**
     * @var String $key - The fileKey you set in the _getAvailableFiles
function
     * @var String $moduleName - The module from the given RecordID,
which should be used to generate the file
     * @var Int $crmid - The ID of the Record, which should be used to
generate the file
     * @return array - An Array with the following Structure
     *                  array(
     *                      'path' => "<temporarily path to the generated
file>",
     *                      'type' => "<mime type of the generated
file>",
     *                      'name' => "<filename of the generated file>"
     *                  );
     * The filename will probably overwritten by some tasks,
     * but if not, it will be used to give this file to the user.
     */
    protected function _getFile($key, $moduleName, $crmid) {
        if(!$this->isModuleActive()) {
            return false;
        }

        return array(
            'path' => "<temporarily path to the generated file>",
            'type' => "<mime type of the generated file>",
            'name' => "<filename of the generated file>"
        );
    }

    public function isModuleActive() {
        return getTabid('[moduleName]') &&
```



```
vtlib_isModuleActive('[moduleName]');
    }
}

// This will register this class in the Interface Registry
\Workflow\InterfaceFiles::register('[SanitizedModuleName]',
'\Workflow\Plugins\InterfaceFiles\[ModuleName]');
```

