TYPO3.Flow - Feature #43104

Add Confirm- Radio- CheckBoxDialog to CommandController(CLI)

2012-11-19 16:25 - Rafael Kähm

Status:	Rejected	**Start date:**	2012-11-19
Priority:	Could have	**Due date:**	
Assignee:	Bastian Waidelich	**% Done:**	30%
Category:	Cli	**Estimated time:**	0.00 hour
Target version:			
PHP Version:	5.3	**Complexity:**	medium
Has patch:	No		

Description

For interactive confirmation or multiple choice by Commands (CLI).

for example: the same code is on https://gist.github.com/4111197
sorry for doccomments

Settings as assoc-array is not comfortably for developers, please comment or speak your ideas about settings!

```
<?php
namespace ..... ;

/*                                                                        *
 * This script belongs to the TYPO3 Flow package '.....'.                 *
 *                                                                        *
 *                                                                        */

use TYPO3\Flow\Annotations as Flow;

/**
 * ConfirmationDialog command controller for the Train2Web.Core package
 * @todo ANSI FOREGROUNDS and BLINK & co.
 * @Flow\Scope("singleton")
 */
class ConfirmationDialogCommandController extends \TYPO3\Flow\Cli\CommandController {

        /**
         * Constants for anwer pattern
         */
        const PATTERN_USER_INPUT_YES = '/^(y|yes|j|ja|si)$/i';
        const PATTERN_USER_INPUT_NO = '/^(n|no|nein)$/i';
        const PATTERN_USER_INPUT_CANCEL = '/^(c|cancel|abort|abbrechen|abortire|abortar)$/i';

        /**
     * A simpliest example for confirm dialog
         *
         * if (user puts yes) {do something} else { do something else }
         * also if user puts n, no, c, cancel then is it else
         *
     * @return void
     */
    public function exampleCommand() {
            if ($this->confirmDialog()) {
                echo "I will continue.".PHP_EOL;
            } else {
                echo "I will not continue.".PHP_EOL;
            }
    }

        /**
     * A simple example confirm dialog with cancel
```

```
         *
         * Please note that switch case provide == comparison and not === <br>
         * also if you use == instead of === you can't differentiate between no and cancel!!!
         *
     * @return void
     */
    public function example2Command() {
            $usersChoice = $this->confirmDialog();
            if ($usersChoice) {
                echo 'You typed "yes"!'.PHP_EOL;
            } else if ($usersChoice === NULL) {
                echo 'You typed "cancel"!'.PHP_EOL;
            } else if ($usersChoice === FALSE) {
                echo 'You typed "no"!'.PHP_EOL;
            }
    }

        /**
     * An example confirm dialog with $optionalSettings[], answer depends messages and callbacks
         *
         * Please note that $optionalSettings['*'] - array has priority
         *
     * @return void
     */
    public function demoCommand() {
            $settings = array(
                'confirmationMessage'       =>
'Please put "y" or "n" or "c" or nothing to use "%1$s" and press enter: ',
                'standardAnswer'            => 'yes',
                'messageByYes'              =>
'You typed "%2$s", i will run one callback for "AGREEMENT" but only if it is declared!',
                'messageByNo'               =>
'You typed "%2$s", i will run one callback for "REJECTION" but only if it is declared!',
                'messageByCancel'           =>
'You typed "%2$s", i will run one callback for "RESET" but only if it is declared!',
                'messageByWrongInput'       => 'Wrong input; "%2$s"!!!'.PHP_EOL.
'You will be asked for so long, until correct entry is detected.',
                'callBackByYes'             => array(array($this, 'callBackForYes'), array()),
                'callBackByNo'              => array(array($this, 'callBackForNo'), array()),
                'callBackByCancel'          => array(array($this, 'callBackForCancel'), array()),
                'callBackByWrongInput'      => array(array($this, 'callBackForWrongInput'), array
())
            );
            $this->confirmDialog(
'This message will be overwritten if $optionalSettings[\'confirmationMessage\'] is set.', 'NO',
$settings);
    }

        /**
     * An example <b>DEMO for callbacks returnvalue</b> based on <b>confirmationdialog:demo</b>
         *
         *
         * Please note that $optionalSettings['*'] - array has priority
         *
     * @return void
     */
    public function demo2Command() {
            $settings = array(
                'confirmationMessage'       =>
'Please put "y" or "n" or "c" or nothing to use "%1$s" and press enter: ',
                'standardAnswer'            => 'yes',
                'messageByYes'              =>
'You typed "%2$s", i will run one callback for "AGREEMENT" but only if it is declared!',
                'messageByNo'               =>
'You typed "%2$s", i will run one callback for "REJECTION" but only if it is declared!',
                'messageByCancel'           =>
'You typed "%2$s", i will run one callback for "RESET" but only if it is declared!',
```

```
                'callBackByYes'             => array(array($this, 'callBackForYes'), array()),
                'callBackByNo'              => array(array($this, 'callBackForNo'), array()),
                'callBackByCancel'          => array(array($this, 'callBackForCancel'), array()),
                'callBackByWrongInput'      => array(array($this, 'callBackForWrongInput2'), array
())
            );
            $this->confirmDialog(
'This message will be overwritten if $optionalSettings[\'confirmationMessage\'] is set.', 'NO',
$settings);
    }

    /**
     * Shows confirmation message and reads users answer from command line.
     *
     * <b>WARNING: don't set anything if you do not want to use it, it will cause one error. <br>
     * ALL PARAMETERS ARE OPTIONAL. <br>
     * ATTENTION: all settings in array $optionalSettings have priority.</b><br>
     *
     * <b>$optionalSettings</b> = array (<br>
     * <b>'confirmationMessage'</b>     => 'You can use %1$s as placeholder for your "standardAnsw
er".',<br>
     * <b>'standardAnswer'</b>          => $standardAnswer,<br>
     * <b>'messageByYes'</b>            => 'You can use %2$s as placeholder for users input string
.',<br>
     * <b>'messageByNo'</b>             => 'Please see "messageByYes".',<br>
     * <b>'messageByCancel'</b>         => 'Please see "messageByYes".',<br>
     * <b>'messageByWrongInput'</b>     => 'Please see "messageByYes".',<br>
     * <b>'callBackByYes'</b>           => array (array($object, 'functionName'), array($param1, $
param2, 'otherParam')), // if your callback return one value then confirmDialog return this instea
d of standard return of confirmDialog<br>
     * <b>'callBackByNo'</b>            => '', // can also be emppty, confirmDialog will print mes
sageBy{yes|no|cancel} and then returns standard value for this answer.<br>
     * <b>'callBackByCancel'</b>        => 'please see "callBackByYes" and "callBackByNo"',<br>
     * <b>'callBackByWrongInput'</b>    => 'please see "callBackByYes" and "callBackByNo"', // if
return is void then : will ask for so long, until correct entry is detected.<br>
     * <b>'infiniteLoopByWrongInput'</b> => TRUE, // if FALSE and callBackByWrongInput is void the
n throws \InvalidArgumentException with code 1352747275<br>
     * <b>'vsprintfVars'</b>            => array($your, $own, $vars) <br>
     * );
     *
     * @param string $confirmationMessage = <b>'Would you like to perform this command? [yes|no|ca
ncel] [%1$s] : '</b>
     * @param string $standardAnswer = <b>'no'</b>
     * @param array $optionalSettings <b>have priority for $confirmationMessage and $standardAnswe
r</b>
     * @return boolean|NULL|yourOwnType this function will return your own type if your callback r
eturns one value and if not then callbackBy{<br>Yes returns <b>TRUE</b><br>No returns <b>FALSE</b>
<br>Cancel returns <b>NULL</b><br>WrongInput throws <b>\InvalidArgumentException</b><br>}.
     * @throws \InvalidArgumentException (code: 1353073679) if this function will be called with w
rong settings
     */
    protected function confirmDialog($confirmationMessage =
'Would you like to perform this command? [yes|no|cancel] [%1$s] : ', $standardAnswer = 'no', array
 $optionalSettings = NULL){
        // search patterns

        //| prepare and set $optionalSettings
        //|
        //|// set needed DEFAULTSETTINGS -> can be overwritten with array $optionalSettings
        $defaultSettings = array(
            'confirmationMessage'           => $confirmationMessage,
            'standardAnswer'                => $standardAnswer,
            // by infiniteLoopByWrongInput == TRUE prints messageByWrongInput
            'messageByWrongInput'           => 'Wrong input: %2$s',
            'callBackByWrongInput'          => '',
```

```
// set to TRUE to prevent \InvalidArgumentException by wrong input -> will cause infinite loop to
the same confirm dialog. Prints "Wrong input: %3$s" and confirmationMessage again
            'infiniteLoopByWrongInput'      => TRUE,
            'vsprintfVars'                  => array (
                                                   $standardAnswer,
                                                   ''
// placeholder for current users input, can be used in 'wrongInputMessage'
                                               )
        );

        //|//| overwrite DEFAULTSETTINGS with users SETTINGS if nacessary ($optionalSettings)
        if ($optionalSettings === NULL) {
            $settings = $defaultSettings;
        } else {
            $settings = array_merge($defaultSettings, $optionalSettings);

//|//|//| set 'vsprintfVars' properly -> this will force priority for entire settings array
            if (!empty($optionalSettings['standardAnswer'])) {
                $settings['vsprintfVars'][0] = $optionalSettings['standardAnswer'];
            }
        }
        // analyse developers set of $optionalSettings['standardAnswer']
        $developersStandardAnswerIsYes = preg_match(self::PATTERN_USER_INPUT_YES, $settings[
'standardAnswer']);
        $developersStandardAnswerIsNo = preg_match(self::PATTERN_USER_INPUT_NO, $settings[
'standardAnswer']);
        $developersStandardAnswerIsCancel = preg_match(self::PATTERN_USER_INPUT_CANCEL, $settings[
'standardAnswer']);

        // throw an exception, if this function will be called with wrong parameter
        //
        // developers standardAnswer must be set correctly and can not be empty
        $developersStandardAnswerIsCorrect = $developersStandardAnswerIsYes ||
$developersStandardAnswerIsNo || $developersStandardAnswerIsCancel;
        if (!$developersStandardAnswerIsCorrect) {
            throw new \TYPO3\Flow\Exception('Developer has set $settings[\'standardAnswer\'] as '.
$settings['standardAnswer']. ' please set this in your code properly', 1352736688);
        }
        $messages = array();
        foreach ($settings as $key => $value) {

// Texts validation: will trying to render defined mesage by fail developer can see it immediately
            if (preg_match('/^message/', $key) && !empty($value)) {
                $messages[$key] = \vsprintf($settings[$key], $settings['vsprintfVars']);
            }
            // CallBack: throw an Exceptions immediately if something is wrong with callBacks
            if (preg_match('/^callBack/', $key) && !empty($value)
                && !method_exists($value[0][0], $value[0][1])) {

                throw new \InvalidArgumentException(
                        'function "'.$value[0][1].'" for "'.$key.
'" not exists or can not be called.'.PHP_EOL.
                        'See: call_user_func_array()'.PHP_EOL.
                        'http://php.net/manual/en/function.call-user-func-array.php',
                        1353073679
                );
            }
        }

        // render output for question
        $output = vsprintf($settings['confirmationMessage'], $settings['vsprintfVars']);
        echo $output.' ';

        $usersInput = rtrim(fgets(STDIN));
        // set 'vsprintfVars' properly
        $settings['vsprintfVars'][1] = $usersInput;
        if (empty($usersInput)) {
```

```
            $settings['vsprintfVars'][1] = $settings['standardAnswer'];
        }
        // prepare users answer
        $userAnsweredWithYes = preg_match(self::PATTERN_USER_INPUT_YES, $usersInput)
                || empty($usersInput) && $developersStandardAnswerIsYes;
        $userAnsweredWithNo = preg_match(self::PATTERN_USER_INPUT_NO, $usersInput)
                || empty($usersInput) && $developersStandardAnswerIsNo;
        $userAnsweredWithCancel = preg_match(self::PATTERN_USER_INPUT_CANCEL, $usersInput)
                || empty($usersInput) && $developersStandardAnswerIsCancel;

        $usersInputIsCorrect = $userAnsweredWithYes || $userAnsweredWithNo ||
$userAnsweredWithCancel || empty($usersInput);

        if (empty($usersInput)) {
            $usersInput = $settings['standardAnswer'];
        }
        // evaluate
        if ($usersInputIsCorrect) {
            // for YES
            if ($userAnsweredWithYes) {
                if (!empty($settings['messageByYes'])) {
                    echo vsprintf($settings['messageByYes'], $settings['vsprintfVars']).PHP_EOL;
                }
                if (!empty($settings['callBackByYes'])) {
                    $returnValueByYes = call_user_func_array($settings['callBackByYes'][0],
$settings['callBackByYes'][1]);
                }
                if (isset($returnValueByYes)) {
                    return $returnValueByYes;
                }
                return true;
            }
            // for NO
            if ($userAnsweredWithNo) {
                if (!empty($settings['messageByNo'])) {
                    echo vsprintf($settings['messageByNo'], $settings['vsprintfVars']).PHP_EOL;
                }
                if (!empty($settings['callBackByNo'])) {
                    $returnValueByNo = call_user_func_array($settings['callBackByNo'][0],
$settings['callBackByNo'][1]);
                }
                if (isset($returnValueByNo)) {
                    return $returnValueByNo;
                }
                return false;
            }
            // for CANCEL
            if ($userAnsweredWithCancel) {
                if (!empty($settings['messageByCancel'])) {
                    echo vsprintf($settings['messageByCancel'], $settings['vsprintfVars']).PHP_EOL
;
                }
                if (!empty($settings['callBackByCancel'])) {
                    $returnValueByCancel = call_user_func_array($settings['callBackByCancel'][0],
$settings['callBackByCancel'][1]);
                }
                if (isset($returnValueByCancel)) {
                    return $returnValueByCancel;
                }
                return NULL;
            }
        } else {
            // for wrong input
            if (!empty($settings['messageByWrongInput'])) {
                echo vsprintf($settings['messageByWrongInput'], $settings['vsprintfVars']).PHP_EOL
;
            }
```

```
            if (!empty($settings['callBackByWrongInput'])) {
                $returnValueByWrongInput = call_user_func_array($settings['callBackByWrongInput'][
0], $settings['callBackByCancel'][1]);
            }
            if (isset($returnValueByWrongInput)) {
                return $returnValueByWrongInput;
            }
            // throws  Exception, if developer will no infinite loops
            if (!$settings['infiniteLoopByWrongInput']) {
                throw new \InvalidArgumentException(
'User has made incorrect input and you have not caught this exception in your confirmation dialog.
  TIPP: You can set $optionalSettings[\'infiniteLoopByWrongInput\']', 1352747275);
            }
        }

// will give rise to infinite looping by incorrect user's input if $optionalSettings['infiniteLoop
ByWrongInput'] == TRUE
        return $this->confirmDialog($confirmationMessage, $standardAnswer, $settings);
    }

    private function callBackForYes() {
        echo 'You called '.__FUNCTION__.'().'.PHP_EOL;
    }
    private function callBackForNo() {
        echo 'You called '.__FUNCTION__.'().'.PHP_EOL;
    }
    private function callBackForCancel() {
        echo 'You called '.__FUNCTION__.'().'.PHP_EOL;
    }
    private function callBackForWrongInput() {
        echo 'You called '.__FUNCTION__.'().'.PHP_EOL;
    }
    /**
     * as you can see this function has return value
     *
     * confirmDialog() will return calbacks return value
     */
    private function callBackForWrongInput2() {
        echo 'You called '.__FUNCTION__.'().'.PHP_EOL;
        echo 'Wrong input is not allowed, I will cancel this command.'.PHP_EOL;
        return 'blockish';
    }

}

?>
```

History

#1 - 2012-11-19 16:44 - Adrian Föder

- Assignee deleted (Adrian Föder)

- Has patch changed from Yes to No

Rafael, thanks for your submission.
The functionality generally is nice (i.e. the idea); I would say it could be an AbstractInteractiveCommandController which users/developers will be allowed to use.
Please update your code to this, maintaining the Coding Guidelines (http://flow.typo3.org/documentation/codingguidelines.html), and push such a patch to the review system (http://wiki.typo3.org/Contribution_Walkthrough_with_CommandLine).

However, feel free to work on the gist until the code would be ready for use.

#2 - 2012-11-19 16:44 - Adrian Föder

- Priority changed from Should have to Could have

#3 - 2012-11-20 09:53 - Karsten Dambekalns

- Target version deleted (2.0 beta 1)

#4 - 2012-12-10 15:34 - Bastian Waidelich

- Category changed from Command to Cli

- Status changed from New to Needs Feedback

- Assignee set to Bastian Waidelich

Thanks for this!
IMO this is quite specific and a bit too much magic to be implemented in the core.
Does anything speak against adding this to a separate package which someone could add a dependency on when relying on this kind of feature?

If you agree I'd close this issue.

#5 - 2012-12-18 15:06 - Bastian Waidelich

- Status changed from Needs Feedback to Rejected

Closing due to lack of feedback.
I hope this can be useful for people looking for a similar feature (google should still point them here) but – as written above – I'd consider this too specific to be added to the core.
Feel free to reopen and/or comment if you don't agree