TYPO3.Flow - Feature # 43104

Status:	Rejected	**Priority:**	Could have
Author:	Rafael Kähm	**Category:**	Cli
Created:	2012-11-19	**Assigned To:**	Bastian Waidelich
Updated:	2012-12-18	**Due date:**	
PHP Version:	5.3		
Has patch:	No		
Complexity:	medium		
Subject:	Add Confirm- Radio- CheckBoxDialog to CommandController(CLI)		

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!

```
1<?php
2namespace ..... ;
3
4/*                                                                    *
5 * This script belongs to the TYPO3 Flow package '.....'.       *
6 *                                                                    *
7 *                                                                    */
8
9use TYPO3\Flow\Annotations as Flow;
10
11/**
12 * ConfirmationDialog command controller for the Train2Web.Core package
13 * @todo ANSI FOREGROUNDS and BLINK & co.
14 * @Flow\Scope("singleton")
15 */
16class ConfirmationDialogCommandController extends \TYPO3\Flow\Cli\CommandController {
17
18      /**
19       * Constants for anwer pattern
20       */
21      const PATTERN_USER_INPUT_YES = '/^(y|yes|j|ja|si)$/i';
22      const PATTERN_USER_INPUT_NO = '/^(n|no|nein)$/i';
23      const PATTERN_USER_INPUT_CANCEL = '/^(c|cancel|abort|abbrechen|abortire|abortar)$/i';
24
25      /**
26    * A simpliest example for confirm dialog
27       *
28       * if (user puts yes) {do something} else { do something else }
29       * also if user puts n, no, c, cancel then is it else
30       *
31    * @return void
32    */
33   public function exampleCommand() {
```

```
34          if ($this->confirmDialog()) {
35              echo "I will continue.".PHP_EOL;
36          } else {
37              echo "I will not continue.".PHP_EOL;
38          }
39      }
40
41          /**
42       * A simple example confirm dialog with cancel
43           *
44           * Please note that switch case provide == comparison and not === <br>
45           * also if you use == instead of === you can't differentiate between no and cancel!!!
46           *
47       * @return void
48       */
49      public function example2Command() {
50          $usersChoice = $this->confirmDialog();
51          if ($usersChoice) {
52              echo 'You typed "yes"!'.PHP_EOL;
53          } else if ($usersChoice === NULL) {
54              echo 'You typed "cancel"!'.PHP_EOL;
55          } else if ($usersChoice === FALSE) {
56              echo 'You typed "no"!'.PHP_EOL;
57          }
58      }
59
60          /**
61       * An example confirm dialog with $optionalSettings[], answer depends messages and callbacks
62           *
63           * Please note that $optionalSettings['*'] - array has priority
64           *
65       * @return void
66       */
67      public function demoCommand() {
68          $settings = array(
69              'confirmationMessage'       => 'Please put "y" or "n" or "c" or nothing to use "%1$s" and press enter: ',
70              'standardAnswer'            => 'yes',
71              'messageByYes'              => 'You typed "%2$s", i will run one callback for "AGREEMENT" but only if it is
declared!',
72              'messageByNo'               => 'You typed "%2$s", i will run one callback for "REJECTION" but only if it is declared!',
73              'messageByCancel'           => 'You typed "%2$s", i will run one callback for "RESET" but only if it is declared!',
74              'messageByWrongInput'       => 'Wrong input; "%2$s"!!!'.PHP_EOL.'You will be asked for so long, until correct
entry is detected.',
75              'callBackByYes'             => array(array($this, 'callBackForYes'), array()),
76              'callBackByNo'              => array(array($this, 'callBackForNo'), array()),
77              'callBackByCancel'          => array(array($this, 'callBackForCancel'), array()),
78              'callBackByWrongInput'      => array(array($this, 'callBackForWrongInput'), array())
79          );
80          $this->confirmDialog('This message will be overwritten if $optionalSettings[\'confirmationMessage\'] is set.', 'NO',
$settings);
81      }
82
83          /**
```

```
 84     * An example <b>DEMO for callbacks returnvalue</b> based on <b>confirmationdialog:demo</b>
 85         *
 86         *
 87         * Please note that $optionalSettings['*'] - array has priority
 88         *
 89     * @return void
 90     */
 91    public function demo2Command() {
 92            $settings = array(
 93                'confirmationMessage'       => 'Please put "y" or "n" or "c" or nothing to use "%1$s" and press enter: ',
 94                'standardAnswer'            => 'yes',
 95                'messageByYes'              => 'You typed "%2$s", i will run one callback for "AGREEMENT" but only if it is
declared!',
 96                'messageByNo'               => 'You typed "%2$s", i will run one callback for "REJECTION" but only if it is declared!',
 97                'messageByCancel'           => 'You typed "%2$s", i will run one callback for "RESET" but only if it is declared!',
 98
 99                'callBackByYes'             => array(array($this, 'callBackForYes'), array()),
100                 'callBackByNo'              => array(array($this, 'callBackForNo'), array()),
101                 'callBackByCancel'          => array(array($this, 'callBackForCancel'), array()),
102                 'callBackByWrongInput'      => array(array($this, 'callBackForWrongInput2'), array())
103             );
104             $this->confirmDialog('This message will be overwritten if $optionalSettings[\'confirmationMessage\'] is set.', 'NO',
$settings);
105     }
106
107     /**
108      * Shows confirmation message and reads users answer from command line.
109      *
110      * <b>WARNING: don't set anything if you do not want to use it, it will cause one error. <br>
111      * ALL PARAMETERS ARE OPTIONAL. <br>
112      * ATTENTION: all settings in array $optionalSettings have priority.</b><br>
113      *
114      * <b>$optionalSettings</b> = array (<br>
115      * <b>'confirmationMessage'</b>     => 'You can use %1$s as placeholder for your "standardAnswer".',<br>
116      * <b>'standardAnswer'</b>          => $standardAnswer,<br>
117      * <b>'messageByYes'</b>            => 'You can use %2$s as placeholder for users input string.',<br>
118      * <b>'messageByNo'</b>             => 'Please see "messageByYes".',<br>
119      * <b>'messageByCancel'</b>         => 'Please see "messageByYes".',<br>
120      * <b>'messageByWrongInput'</b>     => 'Please see "messageByYes".',<br>
121      * <b>'callBackByYes'</b>           => array (array($object, 'functionName'), array($param1, $param2, 'otherParam')), // if
your callback return one value then confirmDialog return this instead of standard return of confirmDialog<br>
122      * <b>'callBackByNo'</b>            => '', // can also be emppty, confirmDialog will print messageBy{yes|no|cancel} and then
returns standard value for this answer.<br>
123      * <b>'callBackByCancel'</b>        => 'please see "callBackByYes" and "callBackByNo"',<br>
124      * <b>'callBackByWrongInput'</b>    => 'please see "callBackByYes" and "callBackByNo"', // if return is void then : will
ask for so long, until correct entry is detected.<br>
125      * <b>'infiniteLoopByWrongInput'</b> => TRUE, // if FALSE and callBackByWrongInput is void then throws
\InvalidArgumentException with code 1352747275<br>
126      * <b>'vsprintfVars'</b>            => array($your, $own, $vars) <br>
127      * );
128      *
129      * @param string $confirmationMessage = <b>'Would you like to perform this command? [yes|no|cancel] [%1$s] : '</b>
130      * @param string $standardAnswer = <b>'no'</b>
```

```
131     * @param array $optionalSettings <b>have priority for $confirmationMessage and $standardAnswer</b>
132     * @return boolean|NULL|yourOwnType this function will return your own type if your callback returns one value and if not then callbackBy{<br>Yes returns <b>TRUE</b><br>No returns <b>FALSE</b><br>Cancel returns <b>NULL</b><br>WrongInput throws <b>\InvalidArgumentException</b><br>}.
133     * @throws \InvalidArgumentException (code: 1353073679) if this function will be called with wrong settings
134     */
135    protected function confirmDialog($confirmationMessage = 'Would you like to perform this command? [yes|no|cancel] [%1$s] : ', $standardAnswer = 'no', array $optionalSettings = NULL){
136        // search patterns
137
138        //| prepare and set $optionalSettings
139        //|
140        //|// set needed DEFAULTSETTINGS -> can be overwritten with array $optionalSettings
141        $defaultSettings = array(
142            'confirmationMessage'           => $confirmationMessage,
143            'standardAnswer'                => $standardAnswer,
144            // by infiniteLoopByWrongInput == TRUE prints messageByWrongInput
145            'messageByWrongInput'           => 'Wrong input: %2$s',
146            'callBackByWrongInput'          => '',
147            // 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
148            'infiniteLoopByWrongInput'      => TRUE,
149            'vsprintfVars'                  => array (
150                                               $standardAnswer,
151                                               '' // placeholder for current users input, can be used in 'wrongInputMessage'
152                                              )
153        );
154
155        //|//| overwrite DEFAULTSETTINGS with users SETTINGS if nacessary ($optionalSettings)
156        if ($optionalSettings === NULL) {
157            $settings = $defaultSettings;
158        } else {
159            $settings = array_merge($defaultSettings, $optionalSettings);
160        //|//|//| set 'vsprintfVars' properly -> this will force priority for entire settings array
161            if (!empty($optionalSettings['standardAnswer'])) {
162                $settings['vsprintfVars'][0] = $optionalSettings['standardAnswer'];
163            }
164        }
165        // analyse developers set of $optionalSettings['standardAnswer']
166        $developersStandardAnswerIsYes = preg_match(self::PATTERN_USER_INPUT_YES, $settings['standardAnswer']);
167        $developersStandardAnswerIsNo = preg_match(self::PATTERN_USER_INPUT_NO, $settings['standardAnswer']);
168        $developersStandardAnswerIsCancel = preg_match(self::PATTERN_USER_INPUT_CANCEL, $settings['standardAnswer']);
169
170        // throw an exception, if this function will be called with wrong parameter
171        //
172        // developers standardAnswer must be set correctly and can not be empty
173        $developersStandardAnswerIsCorrect = $developersStandardAnswerIsYes || $developersStandardAnswerIsNo || $developersStandardAnswerIsCancel;
174        if (!$developersStandardAnswerIsCorrect) {
175            throw new \TYPO3\Flow\Exception('Developer has set $settings[\'standardAnswer\'] as '.$settings['standardAnswer']. ' please set this in your code properly', 1352736688);
176        }
```

```
177     $messages = array();
178     foreach ($settings as $key => $value) {
179         // Texts validation: will trying to render defined mesage by fail developer can see it immediately
180         if (preg_match('/^message/', $key) && !empty($value)) {
181             $messages[$key] = \vsprintf($settings[$key], $settings['vsprintfVars']);
182         }
183         // CallBack: throw an Exceptions immediately if something is wrong with callBacks
184         if (preg_match('/^callBack/', $key) && !empty($value)
185             && !method_exists($value[0][0], $value[0][1])) {
186
187             throw new \InvalidArgumentException(
188                 'function "'.$value[0][1].'" for "'.$key.'" not exists or can not be called.'.PHP_EOL.
189                 'See: call_user_func_array()'.PHP_EOL.
190                 'http://php.net/manual/en/function.call-user-func-array.php',
191                 1353073679
192             );
193         }
194     }
195
196     // render output for question
197     $output = vsprintf($settings['confirmationMessage'], $settings['vsprintfVars']);
198     echo $output.' ';
199
200     $usersInput = rtrim(fgets(STDIN));
201     // set 'vsprintfVars' properly
202     $settings['vsprintfVars'][1] = $usersInput;
203     if (empty($usersInput)) {
204         $settings['vsprintfVars'][1] = $settings['standardAnswer'];
205     }
206     // prepare users answer
207     $userAnsweredWithYes = preg_match(self::PATTERN_USER_INPUT_YES, $usersInput)
208             || empty($usersInput) && $developersStandardAnswerIsYes;
209     $userAnsweredWithNo = preg_match(self::PATTERN_USER_INPUT_NO, $usersInput)
210             || empty($usersInput) && $developersStandardAnswerIsNo;
211     $userAnsweredWithCancel = preg_match(self::PATTERN_USER_INPUT_CANCEL, $usersInput)
212             || empty($usersInput) && $developersStandardAnswerIsCancel;
213
214     $usersInputIsCorrect = $userAnsweredWithYes || $userAnsweredWithNo || $userAnsweredWithCancel || empty(
$usersInput);
215
216     if (empty($usersInput)) {
217         $usersInput = $settings['standardAnswer'];
218     }
219     // evaluate
220     if ($usersInputIsCorrect) {
221         // for YES
222         if ($userAnsweredWithYes) {
223             if (!empty($settings['messageByYes'])) {
224                 echo vsprintf($settings['messageByYes'], $settings['vsprintfVars']).PHP_EOL;
225             }
226             if (!empty($settings['callBackByYes'])) {
227                 $returnValueByYes = call_user_func_array($settings['callBackByYes'][0], $settings['callBackByYes'][1]);
228             }
```

```
229             if (isset($returnValueByYes)) {
230                 return $returnValueByYes;
231             }
232             return true;
233         }
234         // for NO
235         if ($userAnsweredWithNo) {
236             if (!empty($settings['messageByNo'])) {
237                 echo vsprintf($settings['messageByNo'], $settings['vsprintfVars']).PHP_EOL;
238             }
239             if (!empty($settings['callBackByNo'])) {
240                 $returnValueByNo = call_user_func_array($settings['callBackByNo'][0], $settings['callBackByNo'][1]);
241             }
242             if (isset($returnValueByNo)) {
243                 return $returnValueByNo;
244             }
245             return false;
246         }
247         // for CANCEL
248         if ($userAnsweredWithCancel) {
249             if (!empty($settings['messageByCancel'])) {
250                 echo vsprintf($settings['messageByCancel'], $settings['vsprintfVars']).PHP_EOL;
251             }
252             if (!empty($settings['callBackByCancel'])) {
253                 $returnValueByCancel = call_user_func_array($settings['callBackByCancel'][0], $settings['callBackByCancel'][
1]);
254             }
255             if (isset($returnValueByCancel)) {
256                 return $returnValueByCancel;
257             }
258             return NULL;
259         }
260     } else {
261         // for wrong input
262         if (!empty($settings['messageByWrongInput'])) {
263             echo vsprintf($settings['messageByWrongInput'], $settings['vsprintfVars']).PHP_EOL;
264         }
265         if (!empty($settings['callBackByWrongInput'])) {
266             $returnValueByWrongInput = call_user_func_array($settings['callBackByWrongInput'][0], $settings['
callBackByCancel'][1]);
267         }
268         if (isset($returnValueByWrongInput)) {
269             return $returnValueByWrongInput;
270         }
271         // throws  Exception, if developer will no infinite loops
272         if (!$settings['infiniteLoopByWrongInput']) {
273             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);
274         }
275     }
276     // will give rise to infinite looping by incorrect user's input if $optionalSettings['infiniteLoopByWrongInput'] == TRUE
277     return $this->confirmDialog($confirmationMessage, $standardAnswer, $settings);
278 }
```

```
279
280    private function callBackForYes() {
281        echo 'You called '.__FUNCTION__.'().'.PHP_EOL;
282    }
283    private function callBackForNo() {
284        echo 'You called '.__FUNCTION__.'().'.PHP_EOL;
285    }
286    private function callBackForCancel() {
287        echo 'You called '.__FUNCTION__.'().'.PHP_EOL;
288    }
289    private function callBackForWrongInput() {
290        echo 'You called '.__FUNCTION__.'().'.PHP_EOL;
291    }
292    /**
293     * as you can see this function has return value
294     *
295     * confirmDialog() will return calbacks return value
296     */
297    private function callBackForWrongInput2() {
298        echo 'You called '.__FUNCTION__.'().'.PHP_EOL;
299        echo 'Wrong input is not allowed, I will cancel this command.'.PHP_EOL;
300        return 'blockish';
301    }
302
303}
304
305?>
```

History

#1 - 2012-11-19 16:44 - Adrian Föder

- Assigned To 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

- Assigned To 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