Workflow Action and Notification Customization

One of the ideas behind translate5 workflows is, that specific actions in the application are triggered automatically by a well defined list of triggering conditions.

The mapping between actions and triggers is predefined but can be set by the admin of a translate5 installation. Currently there is no GUI for that, so the settings has to be done directly in the database in the LEK_workflow_action table.

Using the LEK_workflow_action table

Example of inserting a new workflow action

```
INSERT INTO `LEK_workflow_action` (`workflow`,`trigger`,`inStep`,`byRole`,`userState`,`actionClass`,`action`,
`parameters`,`position`)
VALUES ('default', 'doCronDaily', null, null, null, 'editor_Workflow_Actions', 'finishOverduedTasks', null, 0);
```

Example Content in the LEK_workflow_action table

```
MariaDB [icorrectT5]> select * from LEK_workflow_action;
+----+----------+-------------------------------------------------+-----------+--------+-----------
+-----------------------------+--------------------------+------------+----------+
| id | workflow | trigger                                         | inStep    | byRole | userState |
actionClass                 | action                   | parameters | position |
+----+----------+-------------------------------------------------+-----------+--------+-----------
+-----------------------------+--------------------------+------------+----------+
|  1 | default  | handleAllFinishOfARole                          | lectoring | lector | finished  |
editor_Workflow_Actions     | segmentsSetUntouchedState |            |        0 |
|  2 | default  | handleAllFinishOfARole                          | lectoring | lector | finished  |
editor_Workflow_Actions     | taskSetRealDeliveryDate   |            |        1 |
|  3 | default  | handleAllFinishOfARole                          | NULL      | NULL   | finished  |
editor_Workflow_Notification | notifyAllFinishOfARole    |            |        2 |
|  4 | default  | handleUnfinish                                  | NULL      | lector | NULL      |
editor_Workflow_Actions     | segmentsSetInitialState   |            |        0 |
|  5 | default  | handleBeforeImport                              | NULL      | NULL   | NULL      |
editor_Workflow_Actions     | autoAssociateTaskPm       |            |        0 |
|  6 | default  | handleImport                                    | NULL      | NULL   | NULL      |
editor_Workflow_Notification | notifyNewTaskForPm        |            |        0 |
|  7 | default  | handleImport                                    | NULL      | NULL   | NULL      |
editor_Workflow_Actions     | autoAssociateEditorUsers  |            |        1 |
|  8 | default  | handleUserAssociationAdded                      | NULL      | NULL   | NULL      |
editor_Workflow_Notification | notifyNewTaskAssigned     |            |        0 |
| 45 | default  | handleDirect::notifyAllUsersAboutTaskAssociation | NULL      | NULL   | NULL      |
editor_Workflow_Notification | notifyAllAssociatedUsers  |            |        0 |
| 51 | default  | doCronDaily                                     | NULL      | NULL   | NULL      |
editor_Workflow_Actions     | deleteOldEndedTasks       | NULL       |        0 |
+----+----------+-------------------------------------------------+-----------+--------+-----------
+-----------------------------+--------------------------+------------+----------+
```

Meanings of the fields in the action table

An action is either just a function (called action) which does something or a notification which sends a notification (by default an email) to some users.

For examples of usage see the classe: editor_Workflow_Notification and editor_Workflow_Actions.

Field	Description / idea behind
id	DB auto incremented ID

workflow	defines to which workflow the action belongs. So only the actions for the workflow configured in the task are triggered. Currently the class inheritance hierarchy is reflected here. So for a Workflow class "foo" extending "default" all actions with both values are considered. **This behavior will change in the future** to enable the possibilty to disable extended default actions by child classes.
trigger	The named event trigger to react on. See list of available triggers below.
inStep	In step filter: the action is executed only if the tasks workflow is in the configured step. NULL means no filter.
byRole	By role filter: the action is executed only if the initiator of the trigger is in the configured role. NULL means no filter.
userState	userState filter: the action is executed only if the initiator of the trigger has the configured job status. NULL means no filter.
actionClass	the class where the to be called action is contained. Must inherit from "editor_Workflow_Actions_Abstract" class.
action	the action to be called
parameter	optional additional parameters, which are passed to the executed action. For example Mail notifications can be configured
position	Order of execution for entries with the same trigger configuration.

External Workflow Trigger

The TaskController provides a callable action to trigger workflow events via URL - if configured in the workflow action table.

The value for the field trigger must contain the prefix: "handleDirect::" the rest of the string is the name of the trigger passed as POST parameter.

See also Task REST Api and the examples.

Example:

By default the notifyAllAssociatedUsers action is configured:

DirectTrigger configuration

```
| 45 | default  | handleDirect::notifyAllUsersAboutTaskAssociation | NULL      | NULL   | NULL      |
editor_Workflow_Notification | notifyAllAssociatedUsers  |            |        0 |
```

To trigger that action the following POST request must be done. This can be either done via frontend or via external API call. The only precondition is, that the calling user is a PM user.

Calling the workflow trigger via HTTP

What	Value
Method	POST
URL	"/editor/task/ID/workflow" where the ID is the tasks DB id.
Parameter "trigger"	the trigger to be triggered, in the example above "notifyAllUsersAboutTaskAssociation"

Workflow Trigger in the application

- TODO List all in the core available workflow trigger
- TODO Explain that more Actions can be added in the core code

Workflow Actions and Notifications

- TODO List all in the core available workflow actions and notifications and explain them (from editor_Workflow_Actions)
- TODO Explain that more Actions can be added by custom Plugins or in the core code

Default activated Actions

- TODO To be done

Daily called (Cronjob) Actions

In order to use the periodically triggered actions (cronjobs) the following configuration has to be done:

- Setup a cronjob or scheduled task which calls the URL https://YOURINSTALLATION/editor/cron/daily/ (under linux for example with wget)
- The IP Adress of the caller must configured in the Zf_configuration table for security reasons
    - Set the caller IP Adress in the configuration runtimeOptions.cronIP

After setting up the cron job, the workflow trigger "doCronDaily" is fired on each call of the above URL.
By default no actions are bound to the "doCronDaily" trigger, so no daily actions are activated.

Useful actions for this trigger are:

- `editor_Workflow_Actions::finishOverduedTasks`
- `editor_Workflow_Actions::deleteOldEndedTasks`