Quality Management Plugin

%$SHORTDESCRIPTION%

On this page:

Description

QMPlugin provides a way to manage the quality of content created in a wiki. While Foswiki itself already provides efficient means to control access, content sometimes needs to be reviewed and approved while evolving as part of a quality management process. This process is encoded in a kind of workflow where contributors play different roles, e.g. being an idea creator, elaborator, maintainer, reviewer, approver, or interested party. The actual workflow varies in most real world scenarios and can be customized to exactly your needs. Of course multiple workflows may exist for different kind of content being managed in different ways.

Similar plugins have been implemented before: WorkflowPlugin, ApprovalPlugin. QMPlugin is a completely new and clean implementation. Key differences are:

- strictly object oriented
- parallel approvals: multiple persons may have to sign off to a certain degree
- consistent access control: purely rely on Foswiki's access control mechanism to grant access to documents; no unapproved content is disclosed by any means, such as other plugins accessing or indexing the content
- approval rights: maintain approval rights separate to pure change rights to a topic, so people can approve content yet have no other rights to change it

- roles: a role concept eases workflow definitions a lot
- visual workflows: workflows are visualized using GraphvizPlugin
- workflow history: improved storage format of the workflow history of a topic; backwards compatible with WorkflowPlugin
- commands: implement an open command handling mechanism that can be customized and extended by other plugins (commands are executed during transitions)
- fork-merge: approved documents may be forked into a copy while working on a new revision; the previously approved content is still accessible while the forked version is work in progress; the newly approved revision may then be merged back onto its origin where it was initially forked from
- conditions: additional checks may be put in place to limit or extend the set of edges that may be transitioned at a certain point in the workflow (e.g. when estimated costs are below a certain threshold can a product be acquired directly without extra approval by a responsible person)

Terminology

The key concepts are:

net
: a net consists of nodes and directed edges connecting them; a net may be traversed by switching from one node to the next as long as they are connected by an edge and all conditions are met

node
: identifies a set of properties at this position in the net

edge
: connects two nodes in the net: a source node with a target node (we only consider directed graphs here)

workflow
: a kind of net that defines a progression of steps that comprise a process involving one or more persons

role
: one or more persons or groups that have specific responsibilities while acting on the workflow process

state
: a document can be in a certain state by refering to a current node in the workflow; the state inherits properties from the current node

transition
: a document may change its state by traversing one of the outgoung edges of the current node

activation
: an edge is active when a document is in the state of the source node and all of the edge's constraints are met

acl
: access control lists describe the set of persons or roles that are allowed to carry out a specific action

parallel approval
: a document state change needs to be approved by multiple persons; the number of required approvals is specified by a sign-off percentage

A node in a workflow has got a set of properties which then affect the document when it reaches this state:

- `id`: (required) unique identifier among all nodes of a workflow (e.g. draft, approved, submitted, …)
- `title`: (optional) display title (e.g. Draft, Approved, Waiting for Approval, …)
- `message`: (optional) verbose description (e.g. "This document is being worked on", "This document is waiting for approval.", "This document is approved.", …)
- `allowEdit`: (optional) acl controling edit rights on the document
- `allowView`: (optional) acl controling view rights

The first node in the list of nodes is the "default node" where the process starts. Any document that is under workflow control but has no state assigned to it yet is in this state by definition. A node `id` may be marked with an asterisk, e.g. `released*`. This node is the "approval node" depicting those nodes that specify a kind of final state of the document. Alternatively, you may just use the string "approved" (or any other custom id specified in the preference setting `QMPLUGIN_APPROVAL`) to define an approval node id.

An edge in a workflow defines the kind of actions a person may perform from a source to a target node:

- `from`: (required) id of the source node (or incoming node) of the edge
- `action`: (optional) id of the action performed on the source node, defaults to `to`
- `title: (optional) display title of the action to be performed, defaults to =action`
- `to`: (required) id of the target node (or outgoing node)
- `allowed`: (optional) acl controling the set of persons that are allowed to transition the state of the document
- `enabled`: (optional) a boolean constraint implemented as a TML expression to control if the edge's is allowed to be traversed
- `notify`: (optional) list of roles that are notified when the edge is transitioned
- `command`: (optional) list of commands that are executed when this edge is traversed; see the list below for a description of known actions and their parameters
- `signOff`: (optional) minimum percentage of participants required to transition the state, default to 0%; as long as the sign-off is not reached yet will the document remain in the current state; only when all participants (specified in the "allowed" property) performed the same action on the node will the transition be carried out and the document reaches the target state

A role in a workflow bundles the following properties:

- `id`: (required) unique identifier among all roles of a workflow (e.g. Creator, Approver, Interested Party, …)
- `members`: (required) list of persons, wiki groups or other roles
- `description`: (optional) a text to describe what this role is about
- `notify`: (optional) list of email addresses to be notified when required; if undefined will email addresses be extracted fom the wiki account of the listed persons; notification will be suppressed if defined as `nobody`.

Note that all properties except `id` may contain TML expressions that are expanded in the context of the controlled document. For example the following defines a "Creator" role by extracting the the author of the initial revision of a document. A role `id` may be marked with an asterisk, e.g. `ResponsiblePersons*`. This role is the "admin role" that has got special rights with regards to the workflow. I.e. admins may cancel an ongoing parallel approval. Alternatively, the admin role is named "Admin". Or any other custom id can be specified in the preference setting `QMPLUGIN_ADMIN`.

```
| *ID* | *Members* | *Description* | *Notify* |
| Creator | %IF{"istopic '%WEB%.%TOPIC%'" then="$percntQUERY{\"info.author\" rev=\"1\"}$per-
cnt" else="%USERNAME%"}% | %TRANSLATE{"Person that created this topic."}% | |
| Admin* | JohnDoe | %TRANSLATE{"Persons with admin rights on the workflow."}% |  |
```

Transition Commands

An edge may be annotated with actions in the "Command" column of the edges table. These actions are executed whenever the edge is transitioned. For example a fork and/or copy operation is executed, a preference setting is set. There is a fixed set of available commands, that can be extended by other plugins using the registerCommandHandler() API.

ID	Parameters	Description	Example
`copy`	`topic="<targetTopic>"`, `web="<targetWeb>"`	copy a topic to a different location	`copy(web="Public")`, `copy(topic="SomeOth` `copy(topic="%TOPIC%Published")`
`deletemeta`	`<name>`	delete named meta data of the current topic	`deletemeta(CustomMeta)`
`fork`	`suffix="Copy"`	create a copy of the current topic and append `suffix` to its name; then redirect to it	
`merge`		reverts a previous `fork` operation	

ID	Parameters	Description	Example
`pref`	`topic="<targetTopic>"`, `"<prefName>"`, `name="<prefName>"`, `value="<..."`	set a preference value in a target topic (default current topic), a preference is removed when assigning the empty string	`pref("token" value="1")`, `pref("token" value="%CALCULATE{"$EVAL(%counter{default`
`formfield`	`topic="<targetTopic>"`, `"<fieldName>"`, `name="<fieldName>"`, `value="<..."`	set a formfield in a target topic (default current topic)	`formfield("TopicType" value="%FORMFIELD` `pic="QMPluginPublished")`
`trash`	`"<sourceTopic>`, `topic="<sourceTopic>"`	move a source topic (default current topic) to the trash	`trash(topic="%TOPIC%Published")`

Note that multiple commands may be triggered listing them comma-separated in the "Command" column of the edges table.

Usage

Before you can establish a workflow for a document you need to define the workflow. A minimal workflow defines a set of nodes and edges only:

```
---++ Nodes
| *ID*      |
| created   |
| inprocess |
| submitted |
| approved*  |
| rejected  |

---++ Edges
| *From*    | *To*      |
| created   | submitted |
| inprocess | submitted |
| submitted | approved  |
| submitted | rejected  |
| approved  | inprocess |
| rejected  | inprocess |
```

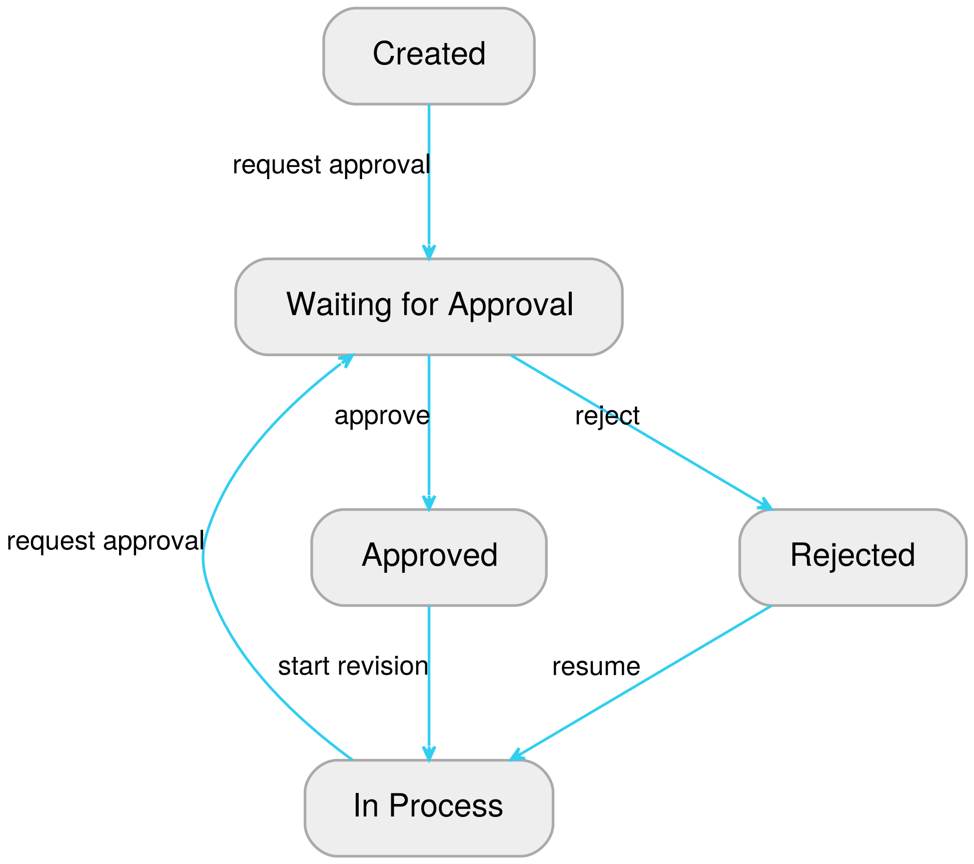
In this example no roles are defined, nor any access control on nodes or edges. A full-featured workflow can be outlined in the following example:

```
---++ Roles
| *ID*     | *Members*                          | *Description* | *Notify* |
| Creator  | %IF{"istopic '%WEB%.%TOPIC%'" then="$percntQUERY{\"info.author\" rev=\"1\"}$per-
cnt" else="%USERNAME%"}% | %TRANSLATE{"The person that created this topic."}% | |
| Approver | %FORMFIELD{"ResponsiblePersons"}%  | %TRANSLATE{"People to approve this topic."}
% |  |
| Admin*   | AdminGroup                         | %TRANSLATE{"People with admin rights on the
workflow."}% |  |

---++ Nodes
| *ID*       | *Title*              | *Allow Edit* | *Allow View*      | *Message* |
| created    | Created              | Creator      | Creator           | %TRANSLATE{"This do-
cument has just been created."}% |
| inprogress | In Progress          | Creator      | Creator           | %TRANSLATE{"This do-
cument is being worked on."}% |
| submitted  | Waiting for Approval | Nobody       | Creator, Approver | %BLUE%%TRANSLA-
TE{"This document is waiting for approval."}%%ENDCOLOR% |
| approved*  | Approved             | Nobody       |                   | %GREEN%%TRANSLA-
TE{"This document is approved."}%%ENDCOLOR% |
| rejected   | Rejected             | Creator      | Creator, Approver | %RED%%TRANSLA-
TE{"Approval for this document has been rejected."}%%ENDCOLOR% |

---++ Edges
| *From*     | *Action*           | *Icon*             | *Title*          | *To*       | *Allo-
wed* | *Notify* | *Enabled* | *Trigger* | *Command* | *Sign Off* |
| created    | request approval  | fa-question-circle | Request approval | submitted  | Crea-
tor   | Approver | | | | |
| inprogress | request approval  | fa-question-circle | Request approval | submitted  | Crea-
tor   | Approver | | | | |
| submitted  | approve           | fa-thumbs-o-up     | Approve          | approved   | Appro-
ver  | | | | merge | 100% |
| submitted  | reject            | fa-thumbs-o-down   | Reject           | rejected   | Appro-
ver  | Creator | | | | |
| approved   | start revision    | fa-pencil          | Start revision   | inprogress | Appro-
ver, Admin | Creator | | | fork | |
| rejected   | resume            | fa-pencil          | Resume           | inprogress | Crea-
tor   | | | | | |
```

A workflow consists of at least two tables to define nodes and edges. A third table may be present optionally defining roles. Note that the properties `ID`, `From`, `Action` and `To` are used internally and must stay constant, whereas the `Title` and `Message` properties may be used more freely such as rendering a multi-lingual display label that varies based on the user's language settings, given MultiLingualPlugin is installed. The resulting net can be visualized using the `%QMGRAPH%` macro in case you also installed the GraphvizPlugin.



Once this workflow definition is stored in a topic, say `SimpleApprovalWorkflow`, it may be added to a topic to start the workflow:

```
%QMSTATE{workflow="SimpleApprovalWorkflow"}%
```

Custom properties

Nodes and edges may have custom properties besides those outlined above. These are defined in additional columns to the roles, nodes or edges table:

Nodes

ID	Class
created	foswikiInfoMessage
inpro-cess	foswikiWarningMessa-ge
submit-ted	foswikiWarningMessa-ge

ID	Class
approved*	foswikiSuccessMessage
rejected	foswikiErrorMessage

These custom properties - in this example "Class" - may be used when rendering a QMSTATE:

```
%QMSTATE{format="<div class='$class'>$message</div>"}%
```

In this example messages will be styled using a css class to control the way messages are displayed.

QMSTATE

This macro renders information about a topic in a specific workflow state.

Parameter	Description	Default
`"..."` or `topic`	topic to render the workflow state for	current topic
`rev`	version of the topic	`rev` urlparam if present, otherwise latest revision
`ignoreerror`	boolean switch to disable any possible error message that might occur	`off`
`format`	format string	value of `template`
`template`	template being used to render the workflow state; this will only be used when no manual `format` string has been specified	`qm::state`
`workflow`	workflow definition	current topic's workflow
`reviewheader`	header string to be prepended to all reviews	`qm::review::header`
`reviewformat`	format string to render a review in the list of available reviews	`qm::review::format`
`reviewfooter`	footer string appended to the reviews being rendered	`qm::review::footer`
`reviewlimit`	maximum number of reviews being rendered	0

Parameter	Description	Default
`reviewreverse`	boolean flag indicating the sorting direction of all reviews	0
`reviewseparator`	separator string put between formatted reviews	
`reviewskip`	used for paging through a list of reviews offset into that list	

The `format` string as well as the `template` content may use the following variables:

- `$action`: last action that lead to the current state
- `$actions`: list of possible actions of outgoing edges from the given state
- `$edges`: list of possible transitions from the given state
- `$adminRole`: id of the admin role
- `$approval`: id of node flag as being the approval node
- `$approvalRev`: most recently approved revision
- `$approvalTime`: time in epoch seconds of the most recent approval
- `$approvalDuration`: time since since the last approval
- `$author`: user that performed the last action
- `$comment`: comment provided by performing the last action
- `$comments`: all comments of people participating in a parallel approval
- `$date`: date of the recent change of the state
- `$datetime`: date-time of the recent change
- `$defaultNode`: starting node in the workflow net
- `$duration`: time since the last change in seconds
- `$emails`: list of emails to notify when this state has been reached
- `$epoch`: epoch seconds of the recent change
- `$hasComments`: boolean flag indicating the existence of comments
- `$hasPending`: boolean flag indicating the existence of pending reviews in a parallel approval
- `$id` or `$state`: current state id
- `$isAdmin`: boolean indicating whether the current user is an admin
- `$notify`: list of people to notify when this state has been reached
- `$numActions`: number of next actions that the current user can perform on the current state
- `$numEdges`: number of possible transitions that the current user can follow from the current state
- `$numReviews`: number of reviews of the current state; in general this is one, but could be more on parallel reviews
- `$numComments`: number of reviews that have comments
- `$origin`: if the current state is a fork, then this property points back to the original topic from which this one has been forked from.
- `$pendingAction`: action that has been already performed by other reviewers of a parallel approval
- `$pendingReviewers`: list of users that still require to review the state change in a parallel approval
- `$possibleReviewers`: list of all users that may perform an action on any outgoing edge from the current state on
- `$prevTitle`: title of the node that lead to the current workflow state
- `$previousState`, `$previousNode`: id of the node that lead to the current workflow state

- `$rev`: revision of the topic that the current workflow state is attached to
- `$reviewFrom`: state that has been reviewed
- `$reviewAction`: action of the last review
- `$reviewTo`: target state of the last review
- `$reviewers`: list of users that already reviewed the state change in a parallel approval
- `$reviews`: list of reviews that led to this state; this is composed by iterating over all reviews using the following `review...` format strings
- `$roles`: list of available roles in a workflow definition
- `$signOff`: percentage ratio of number of users that already reviewed the current state change
- `$topic`: topic name
- `$web`: web name
- `$workflow`: topic of the workflow definition

There are a couple of helper functions that might be called in addition to these variables.

QMBUTTON

This macro renders a button to pop up the workflow dialog.

Parameter	Description	Default
`"..."` or `text`	button label	`Change State`
`topic`	topic to render the button for	current topic
`rev`	version of the topic	`rev` urlparam if present, otherwise latest revision
`workflow`	workflow definition	current topic's workflow
`icon`	button icon	`add`
`format`	format string to render the button	
`template`	template being used when no manual `format` string has been specified	`qm::button`
`class`	css class to add to the button, in addition to `foswikiDialogLink`, `qmChangeStateButton`; if the current state does not allow any followup action for the current user will the button be disabled using the `jqButtonDisabled` class	

The `format` string as well as the `template` content may use the following variables:

- `$class`: css class
- `$icon`: button icon, see VarJQICON
- `$rev`: revision of the topic that the current workflow state is attached to
- `$text`: button label
- `$topic`: topic name
- `$web`: web name
- `$workflow`: topic of the workflow definition

QMNODE

This macro renders information about one specific node in a workflow.

Parameter	Description	Default
`"..."` or `id`		current topic's node or the default node of the given workflow if undefined otherwise
`topic`	topic that has got a workflow assigned to it	current topic
`rev`	version of the topic	`rev` urlparam if present, otherwise latest revision
`format`	format string	`$id, $title, $message`
`ignore-error`	boolean switch to disable any possible error message that might occur	`off`
`workflow`	workflow definition	current topic's workflow

The `format` string as well as the `template` content may use the following variables:

- `$allowEdit`, `$allowView`: list of roles that are allowed to edit/view a document in this node state
- `$isParallel`: true when there are next actions that must be signed off by multiple users
- `$message`: message string part of this node
- `$rev`: revision of the topic that the current workflow state is attached to
- `$state`, `$id`: the id of the current node
- `$text`: button label
- `$title`: title of this node
- `$topic`: topic name
- `$web`: web name

There are a couple of helper functions that might be called in addition to these variables.

QMEDGE

This macro renders information about a specific edge in a workflow.

Parameter	Description	Default
`"..."` or `topic`	topic that has got a workflow assigned to it	current topic
`rev`	version of the topic	`rev` urlparam if present, otherwise latest revision
`from`	source node of the edge	
`action`	action carried out while traversing the edge	
`to`	target node of the edge	
`ignoreerror`	boolean switch to disable any possible error message that might occur	`off`
`format`	format string	`$from, $action, $to`
`workflow`	workflow definition	current topic's workflow

If `from`, `action` or `to` are left unspecified will the state of the current topic be used extracting information about the edge that has been traversed to come to the current state.

The `format` string as well as the `template` content may use the following variables:

- `$action`: action carried out while traversing the edge
- `$from`: source node id
- `$rev`: revision of the topic that the current workflow state is attached to
- `$text`: button label
- `$topic`: topic name
- `$to`: target node id
- `$web`: web name

There are a couple of helper functions that might be called in addition to these variables.

QMROLE

This macro renders information about a specific role in a workflow.

Parameter	Description	Default
`"..."` or `id`	id of the role to render information of	
`topic`	topic that has got a workflow assigned to it	current topic

Parameter	Description	Default
`rev`	version of the topic	`rev` urlparam if present, otherwise latest revision
`format`	format string	`$members`
`ignoreerror`	boolean switch to disable any possible error message that might occur	`off`
`workflow`	workflow definition	current topic's workflow

The `format` string as well as the `template` content may use the following variables:

- `$id`: id of the role
- `$members`: list of role members
- `$isMember`, `$isMember(…)`: returns `1` if the current user (or the one given in brackets) is a member of this role, or `0` otherwise
- `$description`: text in the description column of a role definition
- `$notify`: list of people to notify when an action is performed using this role; if left empty the system email information will be used
- `$rev`: revision of the topic that the current workflow state is attached to
- `$topic`: topic name
- `$web`: web name

There are a couple of helper functions that might be called in addition to these variables.

QMHISTORY

This macro renders the history of the workflow process. It iterates in a loop through the versions of the given topic and renders their state information. Each state information per revision is formatted using the `format` string.

Parameter	Description	Default
`"..."` or `topic`		current topic
`rev`	maximum version of the topic to render the history for	`rev` urlparam if present, otherwise latest revision
`title`	title string normally part of the header	value of the `qm::history::header::title` template
`header`	header string prepended to the output if any thing was found	value of the `qm::history::header` template

Parameter	Description	Default
`format`	format string for each entry in the history	value of the `qm::history::format` template
`separator`	separator between formatted items	
`footer`	footer string appended to the output if anything was found	valoe of the `qm::history::footer` template
`sort`, `order`	sorting of found records; can be any property of a state: id, author, comment, date, etc	`date`
`reverse`	boolean flag to inverse the results	`off`
`skip`	number of records to skip in the loop	`0`
`limit`	maximum number of records to return; return all recurds if left unspecified; both properties `limit` and `skip` can be used to implement pagination	
`workflow`	workflow definition	current topic's workflow
`filter_action`, `filter_author`, `filter_comment`, `filter_message`, `filter_reviewer`, `filter_state`	regular expression that records must match	
`from_date`	only return records later than this date	
`to_date`	only return records up to this date	
`ignoreerror`	boolean switch to disable any possible error message that might occur	`off`

Each entry in the history displays the state the topic was in in this revision. The `format` string may thus use the same variables as in `%QMSTATE`. In addition `$index`, expands to the index in the list of history entries.

The `header`, `format` and `footer` strings may also have:

- `$count`: the total number of states found in the history filters applied
- `$title`: the value of the `title` parameter

There are a couple of helper functions that might be called in addition to these variables.

QMGRAPH

With the help of the GraphvizPlugin this macro renders the flow chart representing the given workflow.

Parameter	Description	Default
`"..."` or `topic`		current topic
`ignoreerror`	boolean switch to disable any possible error message that might occur	`off`
`workflow`	workflow definition	current topic's workflow
`template`	template being used when to render the dot graph	value of the `qm::graph` template

Helper Functions

The format strings of `%QMEDGE`, `%QMNODE`, `%QMROLE`, `%QMSTATE` and `%QHISTORY` might also use the following helper functions.

- `$emails(<list of user(s)>)`: get all emails of all given user strings
- `$formatDateTime(<epoch>)`: format datetime of epoch seconds
- `$formatTime(<epoch>, <format>)`: format date of epoch seconds
- `$edgeTitle(<from>, <action>[, <to>])`: get the title of an edge
- `$nodeTitle(<node-id>)`: get the title of a node
- `$userName(<list of user(s)>)`: get usernames of all given user strings
- `$wikiName(<list of user(s)>)`: get wikinames of all given user strings
- `$wikiUserName(<list of user(s)>)`: get wikiusernames of all given user strings

DataForms Integration

`qmworkflow` formfield

Adding a workflow to a topic is done best using a DataForm with a `qmworkflow` formfield. This lets you select which workflow should be used for the topic.

```
| *Name:*   | *Type:*    | *Size:*  | *Values:*                 | *Description:*           |
*Attributes:*  | *Default:* |
| Workflow  | qmworkflow | 30       | web="..." TopicType="..." | workflow for this topic
|                |            |
...
```

A `qmworkflow` formfield is a type of `topic` formfield (see MoreFormfieldsPlugin). It is a reference to a topic of type `WorkflowDefinition`. A different `TopicType` may be specified using appropriate parameter in the `Values` column. The optional `web` parameter specifies the web where to search for workflow definitions.

If you don't want your users to change a predefinied workflow use the `Default` column and set the `hidden` flag in the `Attributes` column. Note if you are using WikiWorkbenchContrib you should also enable the `create` flag so that the topic gets its workflow as part of the topic creator dialog.

```
| *Name:*   | *Type:*    | *Size:*  | *Values:* | *Description:*   | *Attributes:*  |
*Default:*     |
| Workflow  | qmworkflow | 30       |           | |                | c,h            | <web-topic> |
...
```

`qmstate` formfield

While in most cases the state of a topic is changed as part of the `%QMSTATE` user interface, you may also choose to transition a topic as part of the edit-save cycle. This can be achieved with a DataForm attached to the topic that has got a formfield of type `qmstate`:

```
| *Name:*| *Type:* | *Size:* | *Values:*      | *Description:*         | *Attributes:* |
*Default:* |
| State   | qmstate | 1       | workflow="..." | current workflow state  |
|            |
...
```

This will add a special radiobox interface to the editor that displays optional next actions on the currently being edited topic. When no outgoing edges are activated there will be no option to chose from either. Only *one* formfield of type `qmstate` is allowed per topic. The `Valuse` column is used to parametrize the formfield, i.e. to specify the workflow to be used for this topic.

WikiWorkbench Integration

The Foswiki:Extensions.ClassificationPlugin comes with a set of predefinied TopicTypes that eases creating topics of type WorkflowDefinition as well as ControlledTopics (topics that have a workflow attached to it) as well as a set of simple workflows to start with. Getting started with workflows is achieved by deploying the TopicTypes:

- WorkflowDefinition
- ControlledTopic
- ClassifiedControlledTopic

Perl API

See QMPluginPerlAPI

Installation Instructions

%$INSTALL_INSTRUCTIONS%

Dependencies

%$DEPENDENCIES%

Change History
