

Conception of

Workflow-ng

Next Generation Workflows for Drupal

by Wolfgang Ziegler

Vienna, Austria

Introduction

Workflow-ng ist the next generation module package for building workflows with Drupal¹. It allows building configurable state machines, which can be supplied programmatically or through the admin interface. So workflow-ng will be a tool for module developers as well as for site admins.

Workflow-ng doesn't work only for content nodes, it will be coded on top of a "drupal entity" - so it will start with support for nodes, comments and users. Furthermore it doesn't urge you to introduce new states for your entities, because it does interpret each saved entity as a new possible state. This allows one to reuse existing information, e.g. reuse the existing „published“ and „moderated“ fields of nodes.

Also workflow-ng won't allow one building state-machines only. One will also be able to react on various "events" with configurable actions, which allows site-admins or modules to adapt any default behaviour. E.g. this suits very well for e-mail notifications. Send a thank you message to the author of a certain node type? Just configure the action and workflow-ng will do it for you.

Base-Architecture

Workflow-ng works event driven, so it can work in every imaginable situation.

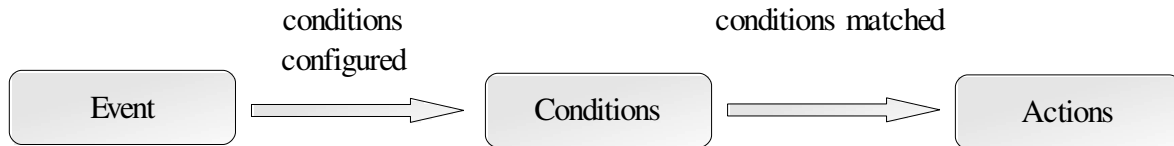
Possible events would be

- a node update/creation/deletion
- a user update/creation/deletion
- a user login
- a buddy invitation (buddylist² module)
- ..

Modules may supply a set of conditions with an associated set of actions to each event. The workflow-ng module will check if all conditions are matched, and if the actions will be executed.

1 <http://drupal.org/>

2 <http://drupal.org/project/buddylist>



That's the basic architecture on which top everything will be built.

Events

Each drupal module may initiate a new event, however workflow-ng will come with a set of default events for common situations. An event is not only characterized by its name, but also through a list of variables that will be passed to the conditions and to potential fired actions. Each variable has to be a “drupal base type” and gets a name. Examples of an event with variables would be:

- a node update, with variables <Node (type: node), Author (type: user)>
- comment creation, with variables <Comment (type: comment), Node (type: node), Author (type:user)>
- a buddylist invitation, with variables <User (type: user), Invited Buddy (type:user)>

Conditions

A set of conditions is only matched, if every single condition of the set matches. Each condition itself consists of a non empty set of sub-conditions, whereas a condition matches if at least one sub-condition is matched.

Example:

```
Condition 1
| Sub-Condition 1a
| Sub-Condition 1b

Condition 2
| Sub-Condition 2a

Condition 3
| Sub- Condition 3a
| Sub- Condition 3b
| Sub- Condition 3c
```

Sub-Conditions are evaluated on the arguments supplied by the calling event, whereas any imaginable comparison is possible due to the fact that modules can even use a code for writing conditions. However there will be a syntax, which allows one to specify conditions like the “node creation date has to be after 30 December 2006” or the “node has to be published”. This can be used to implement an easy usable UI for building conditions.

Actions

Actions will be executed on the supplied arguments. Again, each module is able to supply various actions with different functionality. Optionally they may be configurable through the admin interface, consider a “send an e-mail” action with configurable subject and message body.

Furthermore the great token³ module written by Jeff Eaton could allow the “send an e-mail” action, to provide various text substitutions for all supplied arguments.

Architecture of the state machine

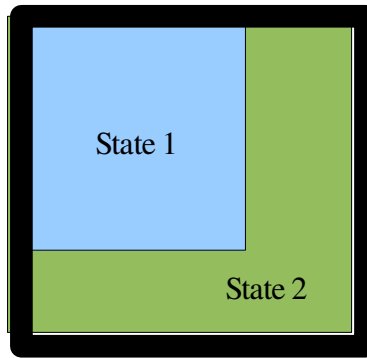
Automatical state definition

The basic architecture somehow misses any states though they are there: It just considers each event as a transition between two states. So one is able to set various actions for specific events.

But there is no possibility for a module operating on a base entity to determine easily in which state this entity is. For this reason the workflow-ng module provides a mechanism to label these states. This can be easily done with the help of the base architecture and a special action, which actually sets the label for the state in case that the conditions have been matched. Actually this mechanism does split the whole volume of all possible states up into some parts of equal labelled states:

³ <http://drupal.org/project/token>

All states



With this architecture it would be possible to use each existing CCK⁴ field (and combinations of them) for initiating a state transition, which would one allow to build sophisticated workflows with flexible forms.

Furthermore it's planned to integrate these labelled states in other drupal tools, ideas which come here to my mind are:

- a views⁵ integration
- a node-access module, which sets node-access dependant on the active state
- a CCK integration, which allows setting field access/visibility dependant on the labelled state of a node

State machines for modules

Workflow-ng will also provide a state machine API for modules, which can be used in conjunction with every “drupal entity”. So this can be used to define new state variables on nodes, comments or users and to initiate the transition between them.

Note that these state variables are only a part of the whole state of the base entity. E.g. if one creates a new state variable on a node, the node's state changes also when the state variable keeps its value, but the title is changed.

4 <http://drupal.org/project/cck>

5 <http://drupal.org/project/views>

Map existing workflows to workflow-ng

The existing workflow⁶ module allows one to define a set of possible transitions per user role for each state. It also provides an UI for setting the state, when a node is edited. The difference of this UI from an usual select is, that it displays only the for the editing user reachable states.

For providing this functionality in workflow-ng a new CCK field will be provided, that uses the state machine API of workflow-ng. However workflow-ng won't limit reachable values of the state variables through the API or through configured automatic occurring transitions, because this would provide no new functionality. Furthermore it would limit the API and complicate the design.

Workflow-ng will just limit the selectable values of the state variables through state transitions caused by the CCK field, so that a proper select field is displayed like the current available workflow module does.

With this existing workflows can be easily mapped to workflow-ng by creating the appropriate CCK-field while automatic occurring state transitions are still available.

Implementation Plan Draft

This is a rough implementation plan for workflow-ng.

Workflow-ng will be a module package consisting of the states, action_ng and workflow_ng modules. The initial implementation will be Drupal 5.x compatible.

states module:

The states module will provide an API for other modules, which allows to build simple state machines on drupal entities — just a simple variable with a name and a set of possible values. The states module will take over the storage and provide a useful integration to other drupal tools like views and the token module.

The states module will also provide the mechanism to label states: Labelled

⁶ <http://drupal.org/project/workflow>

states will be just another state with another name. So labelled states can benefit from the drupal integration provided by this module.

workflow-ng module:

The workflow-ng module will cover the handling of events. It will also provide a mechanism to create and associate conditions with events and a set of actions.

If an event occurs it also evaluates the conditions and — if necessary — executes the actions.

actions-ng module:

This module provides the API for the actions, which may be used with workflow-ng. It also collects a list of available actions and takes over the configuration of actions, if necessary.

workflow-ng_admin module:

This module provides an admin interface, which allows admins to configure conditions and actions for all known events.

workflow-ng_cck module:

The workflow-ng_cck module creates a new CCK field type, which controls a state machine provided by the states module. Ideally this is done for state machines provided by the states module. So this field can be used for controlling new states as well as for controlling states from other modules.

Use Cases

This are just some use cases, which came to my mind and are quite obvious. The main cause for writing workflow-ng is increased flexibility, so that admins can configure more of their site behaviour through the admin interface without writing any code.

Here are some examples:

- Build a review and publishing system

With workflow-ng it's easy to create a simple review and publishing system. The author creates a node, let's call it article. When he has done his article, he sets the node's state to "ready". Now a reviewer can review the article and set the node state to "published". As a result general access to the node will be granted. If the article is not ok the reviewer can just set another state "needs work".

Workflow-ng would also allow adding other features, e.g. an e-mail notification for reviewers, so that they are informed when there is an article waiting for review.

- Build a review and grading system for multiple courses

In conjunction with the organic groups module⁷ it will be possible to create a sophisticated grading system for multiple student courses. Each course has to be its own organic group, which students can join. Now students can submit their work to the course. Then their work will be reviewed by reviewers and as soon as enough reviews are submitted the course instructor gets informed. So the course instructor can mark the the student based on the existing reviews done by the reviewers.

- Force users to finish their profile

It's planned that the pageroute⁸ module will use the states module for tracking, if users have finished their nodeprofile(s)⁹. A simple condition on the user login event could check if the user has finished the profile, and if not fire an action that redirects him to the profile form and tells him to finish his profile.

- Use the state-machine API

Other modules can use the API provided by workflow-ng. They can define state machines with the API provided by the states module and configure conditions and actions by using the workflow-ng module API. They would not only benefit from additions like the views integration offered by the states module, but also save some extra work as the states module takes over the database storage for the provided state machine.

7 <http://drupal.org/project/og>

8 <http://drupal.org/project/pageroute>

9 <http://drupal.org/project/nodeprofile>

- Use the workflow-ng events API

By using the workflow-ng events API modules can economise extra implementations for sending out notification e-mails or messages. E.g. the buddylist module contains code for several optional and configurable mail notifications, which could all be replaced by simple introductions of workflow-ng events. Then users could configure their e-mail notifications using the workflow-ng admin interface or the buddylist module could supply default notifications using workflow-ng too. Furthermore it would be easy to replace e-mail notifications by private messages. This is an often requested feature. Just implement a simple private message action and users are able to use them instead of e-mail notifications.

So you would have less code, but earn much more flexibility.