

Hem

SSES courseweb 3: Course content and access

Visa

Redigera



JOHAN FALK
ONS, 2011-06-01 15:20

BLOGG: [SSES COURSEWEB](#)

This is a part of the experimental open technical documentation for the SSES courseweb project. This part gives explains the node access functionality on the site and gives some more details about the course-related content.

RELATERAT

- > [Technical documentation for "SSES courseweb"](#)
- > [SSES courseweb 2: Site architecture - node types and sections](#)
- > [SSES courseweb 5: Automatic e-mail notifications](#)
- > [SSES courseweb 4: Profile management](#)

One of the more complex aspects of the site is the access control for courses, and their related content. The end result is summarized like this:

- > Administrators and registrars can see/edit/delete everything.
- > Teachers can view and edit the courses they are teachers for.
- > Teachers can create assignments and blog posts relating to the courses they are teachers for. They can view/edit/delete the resulting nodes.
- > Students can view courses *where they are active* (which for example doesn't mean "in waiting list").
- > Students can view assignments and blog posts relating to courses where they are active.
- > Students can create submissions to assignments in courses where they are active. They may add any number of students *from the same course* as group members, and all group members are allowed to view and delete (but not edit) the submission.
- > Teachers can view and edit submissions relating to courses where they are teachers, but not delete the submissions.

All these access rules are managed by a combination of Content Access, Node Access User Reference, Node Access Node Reference and Rules – plus some small custom form alters:

- > Administrator and registrar access is set using normal Drupal permissions. (Note that registrars don't have the administer nodes permission though – this hides the vertical tabs for registrars, and simplifies the node edit forms considerably.)
- > Content Access is used to revoke view access for all relevant roles.
- > Teachers are added to courses using a multi-value user reference field. This field is enhanced with Node Access User Reference to allow referenced users to view and edit the node.
- > Students that are *active* on the course (for example not "in waiting list") are added to a user reference field using a special workflow – see below. This field, too, utilizes Node Access User Reference to grant view access to the students.
- > All course-related content (except submissions) has a node reference field pointing to the

FÖRFATTARE



JOHAN FALK
DRUPAL TRAINER / DEVELOPER

MER FRÅN JOHAN FALK

- > [Learn Drupal coding with NodeOne part 6: More arrays](#)
- > [Learn Drupal coding with NodeOne part 5: Arrays](#)
- > [Technical documentation for "SSES courseweb"](#)



Redigera

DRUPALBOKEN



Redigera

LEARN DRUPAL



> course. This field is enhanced with Node Access Node Reference to allow everyone with view/edit access for the related node to view/edit the node. The node reference field is auto-populated and hidden using Contextual Administration.

- > Since not all students should be able to view all submissions, view permissions are not forwarded by the node reference from submissions to courses. Instead the user reference field “Group members”, pointing to students, is used in combination with Node Access User Reference to allow referenced students to view and delete the submission.
- > The referencable students in the “Group members” field is restricted to the active students on the course, using a custom after_build function on the submission node form. The same function also sets the acting student as selected by default.
- > Teachers are granted view and edit access to the submissions by another user reference field. This is automatically populated by a rule set (“Actions when assignment submissions are saved”) on node save. It might have been possible to use Node Access Node Reference to pass on edit access for the course as view access for the submissions, but a [semi-solved bug](#) kind of prevents this – and it is also possible that teacher edit access for the courses will be removed in the future.

Some additional notes

- > On node edit, node and user reference fields used for access are removed using #access = FALSE. This is put in hook_form_alter in the custom modules *and cannot be put in hook_form_FORM_ID_alter* since it would prevent CCK from building up the form element in the first place.
- > Assignment submissions have two references that should be automatically populated – course reference *and* assignment reference. The assignment reference is populated using Contextual Administration, but the course reference is set by the rule set “Actions when assignment submissions are saved”.
- > The links for adding new content are deliberately placed as sidebar links, not tabs.

ADDING STUDENTS TO COURSES

Student participation at courses are managed by the node type Course Status (cw_status). The most crucial part of these nodes are the *profile reference*, the *course reference*, and the *current status* (being two node reference fields and one text select field, respectively).

The workflow for adding students to courses is this:

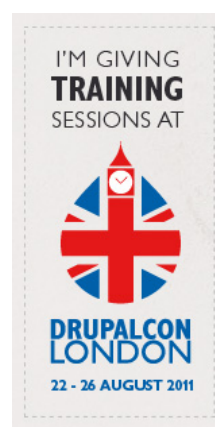
- > A registrar visits the page “List all profiles”, and finds one or many students to add to a course. (Filtering may be done on import node, among other options.)
- > This list is bulk operations view (vbo_list_profiles), so the registrar have checkboxes to select the profiles that should be added. When this is done, he/she hits the button “Add profile to course”.
- > This brings a configuration for for the VBO action, where a course is selected before continuing.
- > The action creates new status nodes for the selected profiles, relating to the selected course. The initial status level is *added*, one of the non-active status levels.
- > Next, the registrar visits the course page (suggestably by the jump menu in the header), and visits the sub-page “Statuses” found in the administration menu for the course.
- > The resulting page lists all status nodes relating to the relevant course, with student names, e-mail addresses and the current status. Filtering can be done by school, current status or previous status. (The sidebar contains a summary of statuses for this course, making it easier to see how many seats are left.)
- > This list, too, is a bulk operations view (statuses_by_course) and the registrar can select any statuses and hit the button “Change status”. Before the action is carried out, a configuration form is displayed where the new status is selected.
- > Finally, when the status level is changed a rule set is triggered (“Actions when saving course statuses”), checking if the status on the course has changed from non-active to active – if so, a chain of actions are triggered. This chain includes assuring that the profile has a user account (and creating one if necessary), updating the student references in the related course to include the student user account, and sending out a welcome e-mail to the student.

WITH NODEONE

1. [Drupal 7 introduction](#), 15 screencasts
2. [Learn Views - Taming the Beast](#), 27 screencasts
3. [Learn Flag](#), 8 screencasts
4. [Learn Page manager!](#), 13 screencasts
5. [Drupal learning curve for configurators](#)
6. [Learn Drupal coding](#) (experimental screencast series, ongoing)
7. [Rules for Drupal 6](#), 14 screencasts
8. [Theming Drupal 6](#), 8 blog posts
9. [49 Drupal 6 modules you should know](#), 7 blog posts

We love sharing Drupal power.

Do you like these learning series? Check out our pre-conference training at [DrupalCon London](#)!



Redigera



- > The same rule set also checks if the status level has changed from active to non-active, and if so removes the student reference from the course (but does not remove any user account).
- > The same rule set also logs information about the status change in the status node.

Some additional notes

- > The logging of status levels in the status nodes depend on two separate CCK fields: One select text field containing the actual name of the status levels, allowing Views exposed filtering on previous status; and one multivalue text field containing the new status level and the time stamp of the change. These are populated quite easily by the action “Add a value to a multiple-value field” in Rules Bonus Pack.
- > In order to get the *new* status level, Rules actually has to bypass tokens – since the token values are not yet updated on the node_pre_save event (see [this issue for details](#)). This, too, is done with an action in Rules Bonus Pack.
- > The status levels are keyed by numbers, in order to easily separate active states from non-active states. If new states are added, make sure that (1) the numbers are positive for active states and negative for non-active, and (2) that both the “Current state” field and the “Status history” fields are changed consequently.

Some notes about the “Add profile to course” action

The “Add profile to course” action is, like many other things on the site, a rule set. The rule set takes three arguments – profile to add, course to add to, and new status. Anyone who has tried using rule sets with multiple arguments in VBO (Drupal 6) knows that this results in textareas where you are supposed to write PHP code – not a situation you would like to have on a live site.

The solution for this was a quite remarkable hack, which included form_altering the second textarea to a select list – fetching all courses through a view (cw_course_list, display attachment_1). Since VBO still expected executable PHP as input, the key values in this list is actually (gulp) on the form “return 123;” – where 123 should be interpreted as the NID for the relevant course. Some other approaches were tried, and much thought went into assuring that this was actually safe. The conclusion was that this *is* actually safe (but extremely ugly), since Drupal’s form API won’t accept any PHP input that wasn’t already defined as an option in the select list to start with.

Exposing this ugly hack in the open is scary. From the code comments in the custom module (where the rest of the function is found):

```
// Note that we're keying this array with PHP snippets. This is most
// likely the most ugly hack I have every done, but VBO expects to send
// off PHP snippets to the rule set. Also note that the user cannot hack
// the form and enter arbitrary PHP script – the entered value will be
// compared against the original options by Drupal FAPI. But still.
// I take the blame for this. Really. @Itangalo
```



Du bevakar detta inlägg, klicka för att sluta bevaka

[Publicera på Drupal Planet](#)

Kommentarer

Skriv ny kommentar

Ditt namn:

[Johan Falk](#)

Kommentar: *

Webbadresser och e-postadresser görs automatiskt till länkar.

Lägger till typografiska justeringar.

Tillåtna HTML-taggar: <a> <cite> <code> <dl> <dt> <dd>

Rader och stycken bryts automatiskt.

Du kan skriva kod med <code>...</code> (generiska) eller <?php ... ?> (markerade PHP) taggar

[Mer information om formateringsmöjligheter](#)

Spara

Förhandsvisa



DRUPAL

Om Drupal
Att välja Drupal
Drupal och öppen
källkod
Drupal Community
Vem använder Drupal?

TJÄNSTER

Förstudie
Design
Utveckling
Drift
Utbildning
Konsultation

OM NODEONE

Människorna
Kontakt
Jobb och praktik
Filmer och
extramaterial