Specification for Implementing Per-Status Limits in Signup Status

Derek Wright (http://drupal.org/user/46549)

Introduction

The Signup Status module (http://drupal.org/project/signup_status) enhances the Signup module ((http://drupal.org/project/signup) by providing an additional feature-rich field to associate with each signup. A common use-case would be to let potential event attendees indicate if they're coming or not ('yes', 'no', 'maybe', etc). However, other sites use the signup status for things like a waiting list, or even tracking payments.

The Signup module has the notion of a total limit on the number of signups for each node. A global limit poses a few problems once Signup status module is enabled:

1) Not every signup status should "count" towards the global limit. For example, if someone signs up to say "no", that shouldn't be included in the total number of signups as far as the limit is concerned.

2) Some sites need to be able to specify separate limits for each possible status. For example, if an event holds 30 people, you might want to have a limit of 30 for an "Approved" status, but leave the "Wait-list" status unlimited.

Having both kinds of signup limits can lead to a lot of confusion in the user interface, and unexpected behavior.

Furthermore, the limits per signup status is currently removed from the Drupal 6 version of Signup Status since it was deemed too complicated (and not working properly) during the initial port to D6.

This document aims to provide a road-map for implementing a fully-functional Signup Status Limits solution for Drupal 6 that will overcome these and other problems.

Two kinds of limits: global and per-status

One of the fundamental problems here is that different sites (and even different events on the same site) will want different limit behavior depending on the specific situation. Therefore, we cannot simply pick one or the other and implement that, we need to support both kinds of limits. Futhermore, we need to support both kinds of limits on the same site. So, it is not simply a question of turning on a new sub-module and having that enforce a single limit style on the entire site.

Global limits and Signup Status

If a site only needs the global signup limits, but has signup_status enabled they still need the ability to specify which limits should impact the total signup count and limit. Because this is a property of each status code itself, this can be configured when adding or modifying the status codes, and can be stored along with the other metadata about the status code in the {signup_status_codes} table. The D6 version of signup_status maintains a checkbox when adding or editing status codes for this, and stores the value in the {signup_status_codes}.mod_signup_count column.

Unfortunately, this checkbox currently has no effect whatsoever, because there's no way for the Signup module to know about this. What is needed here is a hook invoked by the Signup module to allow other modules (in this case, signup_status) to alter the total signup count for a given node (see http://drupal.org/node/359411 for more) or to specify if a given signup should count towards the total or not.

The signup total is used in a few places in the Signup module:

1) When a new signup is added, the total is incremented. If a limit is defined and the total reaches the limit, signups are closed on the node.

2) When a signup is canceled, the total is decremented. If a limit is defined and if the total falls below the limit, signups are re-opened on the node.

3) If a node is edited and the limit is changed, the new limit is tested against the current signup total and signups are opened or closed as appropriate.

4) When viewing the Signup administrative page for a single node, the current total and limit is displayed.

5) When viewing the site-wide Signup administration page, the totals and limits for each node are displayed in a table.

6) The signup total is loaded into the $node object during node_load() and potentially used by other modules.

Given all of this (and especially because of #5) I believe the only viable solution here is that the {signup_log} table needs a new column called something like {signup_log}.count_towards_limit which indicates that a given record in the {signup_log} table (a signup) should count towards the global signup limit for that node. Otherwise, there's no performant way to invoke a hook to alter the totals, especially across multiple nodes.

By default, the new count_towards_limit field would be TRUE. However, inside signup_sign_up_user() we would invoke a new hook called something like hook_signup_data_alter(), which would get the fully-loaded $signup object and give other modules a chance to modify the values before they are saved into the database. This would allow signup_status to properly fill in the $signup->count_towards_limit field depending on the status of the signup. At the end of signup_sign_up_user(), only if $signup->count_towards_limit was still TRUE would the signup total be incremented and the limit checked, which will solve #1.

Once the proper value is saved in {signup_log}.count_towards_limit, all of the other uses of the signup total mentioned above can be handled correctly. #2 is trivial, since only if count_towards_limit is TRUE should the fact that it was canceled impact the total and check the limits. #3 would only use the signups where count_towards_limit is TRUE when deciding if a change to the node's global limit should open or close signups. #4 would be solved by only counting records where count_towards_limit is TRUE for the purposes of displaying the total vs. the limit. Similarly, #5 can be solved by adding a WHERE clause to the LEFT JOIN on {signup_log} inside signup_admin_form_sql() in admin.signup_administration.inc to only count signups where count_towards_limit is TRUE.

#6 brings up the interesting question of what to load into the $node object. I think the best solution here is to actually load two values, the total number of signups, regardless of count_towards_limit, and the total number that count towards the limit. That way, other modules can use either value depending on what they need. We could consider displaying both numbers in various spots in the administrative interface, e.g. #4 and #5 above.

```
---------------------------------------------------------------------
Per-status limits: Signup Status Limit Module (signup_status_limit)
---------------------------------------------------------------------
```

For sites that optionally need to define limits per status, they need something quite a bit more complicated than the above. Also, not everyone using signup_status needs such limits. Therefore, all the code for per-status limits should live in a new signup_status_limit sub-module.

```
------------------------------
Schema
------------------------------
```

The signup_status_limit module would use a new database table called {signup_status_limit_node} for recording the per-node limits (if any) for each node. This table should have the following schema definition:

```
  $schema['signup_status_limit_node'] = array(
    'description' => t('Signup status per-node limits.'),
    'fields' => array(
      'nid' => array(
        'description' => 'Foreign key: {node}.nid of the node with the
limits',
        'type' => 'int',
        'unsigned' => TRUE,
        'not null' => TRUE,
        'default' => 0,
      ),
      'cid' => array(
        'description' => 'Foreign key: {signup_status_codes}.cid of the
status',
        'type' => 'int',
        'unsigned' => TRUE,
        'not null' => TRUE,
        'default' => 0,
      ),
      'limit' => array(
        'description' => 'The limit for this node and status',
        'type' => 'int',
        'unsigned' => TRUE,
        'not null' => TRUE,
        'default' => 0,
      ),
      'primary key' => array('nid, cid'),
    ),
  );
```

A very similar table existed in D5, so in hook_enable(), we could call db_table_exists('signup_status_node_limits'), and if that table is found, we could populate the new table with the values of the old table and then drop it, which would provide a seamless upgrade path for D5 sites.

UI: Node signup settings

Once the signup_status_limit module is enabled, the UI for the "Signup settings" fieldset when editing a node should change. Instead of a single text field for the "Signup limit", there needs to be a set of radio buttons called something like "Signup limit type" which lets you choose:

```
(*) None
( ) Limit on total signups
( ) Separate limits for each signup status
```

The existing "Signup limit" field would be renamed "Limit on total signups", and a new table would be added for the per-status limits for that node with a separate text field for each status code. Ideally, some jQuery would be used to conditionally hide/show the appropriate form sub-element depending on the currently selected value of the radio buttons.

If the "Separate limits" type is used, and any status codes have a limit defined, the appropriate records would be written to the {signup_status_limit_node} table for that node. If the radio is set to None, the {signup}.close_signup_limit field would be set to 0, and if it is set to "Limit on total signups", the value in the "Limit on total signups" field would be written to {signup}.close_signup_limit. If either "None" or "Limit on total signups" is used, all the records from {signup_status_limit_node} for the given node ID (nid) would need to be removed.

UI: Node signup administration

When visiting the "Administer" sub-tab on the "Signups" tab on a signup-enabled node (e.g. node/N/signups/admin), there is a "Signup summary" fieldset that show the current status, total, (global) limit. This is a separate form (id: signup_node_admin_summary_form) and can therefore be altered easily by signup_status_limit. If there are records in the {signup_status_limit_node} table for this node (meaning that per-status limits are in effect), the form would be altered to remove the global total and limit elements, and instead a table with rows for each status would be presented, showing the total number of signups in each status, and the current limits.

Alternate UI for nodes

It could be confusing to event administrators that they can specify the per-status limits in two places (currently true in the Signup module itself for the global signup limit for each node). Furthermore, it might be a nice thing to preserve the old per-status limits when the "limit type" is set to a value that does not use per-status limits. If this was desired, we'd want a seperate DB table like {signup_status_limit_node_settings} that just had a nid column and a column to record which kind of limits should be enforced for that node. The radio buttons on the node editing form described above would be there, but instead of a way to specify the limits per status, there would be a link to the node/N/signups/admin tab where the alternate "Signup summary" UI would be presented. In this case, setting the limit type to "None" or "Total signups" would not clear the records in the {signup_status_limit_node} table. With this approach, the {signup_status_limit_node} table might want to be renamed to something like {signup_status_limit_node_limits} to be more self-documenting and prevent confusion with {signup_status_limit_node_settings}.

UI: Site signup administration

The site-wide signup administration page at admin/content/signup is currently a hard-coded table listing with form elements to change the global limit and signup status (open vs. closed) of every signup enabled node on the site. Modifying this interface to handle both global and per-node signup limits is one of the worst parts of this entire proposal. It is a form (id: signup_admin_form) and can be modified, but due to the custom theming on the form to render it as a sortable table, the options are somewhat limited. Moreover, there's no good way to alter the query used to generate this table. There are a few options:

1) Alter the form but leave it as a hard-coded table. In this approach, I would recommend that for any node where per-status limits are enabled, remove the text area for the "Limit" column, and replace it with a link to the node/N/signups/admin page with the text "Edit per-status limits" or something. This would be relatively easy, but might not be flexible enough or powerful enough, especially for sites that make heavy use of per-status limits. On the other hand, an overview table like this is probably not the best place to be trying to do fine-grained operations with the limits.

2) Convert this table into a view. The signup_status_limit module could provide its own default disabled view for the same location that was better suited for per-status limits. The installation instructions for the signup_status_limit module would encourage sites to disabled the default view from the Signup module for admin/content/signup, and to enable the default provided by the signup_status_limit module instead.

Once this table was implemented as a view, if the Views Bulk Operations (http://drupal.org/project/views_bulk_operations) module (sometimes refered to as simply "VBO") was enabled, the interface could be simplified a lot by making better use of signup-related node actions. For example, Signup module could support an action to open or close signups on selected nodes, and signup_status_limit to support an action to modify the per-status signup limits on selected nodes. This could be a "configurable" action, meaning that when it was chosen, the administrator would be redirected to a form that would allow them to specify the new limit values to use for each signup status currently configured on the site. This would allow a signup administrator to quickly change the per-status limits of many nodes at once.

Implementation details

Whenever the limit type on a node was configured to use per-status limits, internally, the global signup limit would be set to 0 (unlimited). This way, all of the features to automatically open/close signups would be disabled, and signup_status_limits could enforce its own limits.

Whenever a signup-enabled node is loaded, signup_status_limit would inject a $node->signup_status_limit nested array, keyed by status code id (cid). Each sub-array would be an associative array with the keys 'limit' and 'total', which would allow other modules (and signup_status_limit itself) to work with the accurate data.

When the signup form is presented to a user, for each status that is presented on the signup form, the current total for that status would be compared against the limit, and if the limit had already been reached, that status would be removed from the options. If no available status values would be presented, the entire signup form would be replaced with a message that the [node-type] was full and not accepting new signups. As far as the Signup module itself was concerned, signups would still technically be open, but in practice, no one would be able to signup until the limits or totals for a visible status were changed.

Because signups would official stay open for any node with per-status limits, there's no real need for the signup_status_limit module to interact with the Signup module itself regarding opening and closing signups. It would simply alter the signup form as appropriate such that whenever someone tries to signup, they only are presented with choices that are valid based on the current limits. Of course, there would need to be a custom validation handler injected into the signup form to ensure that by the time the user tries to submit the signup form, the limits and totals are enforced at that time, since it could be a long time between when the signup form is rendered on a page and someone actually tries to submit it.