Entity-Based Discussion Forum for Drupal 7

Rasmus Werling

Bachelor's Thesis

Degree Programme in Business Information Technology

Raseborg 2011

BACHELOR'S THESIS

Author: Rasmus Werling
Degree Programme: Business Information Technology
Supervisors: Klaus Hansen

Title: Entity-Based Discussion Forum for Drupal 7

---

15 November 2011 Number of pages 43 + 5 Appendices 0

Summary

This thesis describes the planning and creation of an entity-based discussion forum module for the content management system Drupal 7.

The main focus is on how the new entities in Drupal 7 are taken into use in a discussion forum module created from scratch. Also the database structure, the different implemented features and the integration with the Views module are explained.

The major differences between Drupal 6 and Drupal 7 are dealt with as well: how everything circled around nodes earlier and how Drupal's different components have now been converted into their own entities.

Last but not least the usage of Drupal's different APIs are described as well as all the custom functions developed for the module.

Language: English Key words: Drupal, PHP, entity, module, forum

---

EXAMENSARBETE
Författare: Rasmus Werling
Utbildningsprogram och ort: Informationsbehandling, Raseborg
Handledare: Klaus Hansen

Titel: Entity-Based Discussion Forum for Drupal 7 /
Entitetsbaserat diskussionsforum för Drupal 7

15.11.2011 Sidantal 43 + 5 Bilagor 0

Sammanfattning

Detta arbete behandlar planerandet och förverkligandet av en entitetsbaserad diskussionsforumsmodul för innehållshanteringssystemet Drupal 7.

Huvudsakligt fokus ligger på hur de nya entiteterna i Drupal 7 används för att skapa en diskussionsforumsmodul från början. Också databasstrukturen, de olika implementerade funktionerna och integrationen med Views-modulen förklaras.

I arbetet behandlas också de största skillnaderna mellan Drupal 6 och Drupal 7, nämligen hur allt cirkulerade kring noder tidigare och hur alla olika komponenter i Drupal nu är skilda entiteter.

Sist men inte minst förklaras användningen av de olika Drupal API och alla skräddarsydda funktioner som utvecklats för denna modul.

Språk: Engelska Nyckelord: Drupal, PHP, entitet, modul, forum

---

Table of contents

1 Preface 1
2 Customer 2
3 Drupal 3
3.1 What are entities? 4
3.1.1 "Everything is a node" 4
3.1.2 Entities 5
3.1.4 Fields 7
4 Ready solutions 8
4.1 Core Forum 8
4.2 Advanced Forum 8
4.3 Artesian Forum 8
5 Developing Brain Forum 10
5.1 What the customer wanted 10
5.2 Coding and documentation standards 10
5.3 Features 13
5.3.1 Reply 13
5.3.2 Quote 14
5.3.3 BBCode support 15
5.3.4 Thread categorization 16
5.3.6 Sheriff function 18
5.3.7 Editing of own posts 18
5.4 User permissions 18
5.5 Database structure 20
5.6 Entities 21
5.7 Loading an entity 22
5.8 Views integration 23
5.8.1 Default views 24
5.9 Managing posts and threads 24
5.9.1 Adding posts 25
5.9.2 Editing posts 26
5.9.3 Post submission 27
5.9.4 Deleting posts 27
5.10 Implemented hooks 28
5.10.1 hook_entity_info() 28
5.10.2 hook_entity_property_info_alter() 29

5.10.3 hook_help() 29
5.10.4 hook_menu() 29
5.10.5 hook_permission() 29
5.10.6 hook_theme() 29
5.10.7 hook_views_pre_render() 29
5.10.8 hook_views_api() 29
5.10.9 hook_views_pre_render() 30
5.10.10 hook_field_attach_form() 30
5.10.11 hook_{ENTITY_ID}_insert() 30
5.10.12 hook_{ENTITY_ID}_update() 30
5.10.13 hook_{ENTITY_ID}_delete() 31
5.11 Custom functions 31
5.11.1 brain_forum_thread_get_pids() 31
5.11.2 brain_forum_thread_get_last_published_pid() 32
5.11.3 brain_forum_thread_update_last_post_info() 32
5.11.4 brain_forum_thread_reset_last_post_info() 33
6 Analysis 33
6.1 Customer's feedback 33
6.2 Employer's feedback 33
6.3 Personal evaluation 34
7 Conclusion 35
Sources 36
Summary in Swedish

1 Preface

This thesis is written as part of a project done by Soprano Brain Alliance Ltd (SBA), a Finnish software development company specializing in web-based solutions made with Zend Framework and Drupal. The company was founded in 2003 by Jukka Hassinen and has the most certified PHP programmers in Finland today (Zend Technologies Ltd, n.d.). It is also the only official partner of Zend Technologies in Finland. (Soprano Brain Alliance Oy, n.d. a; n.d. b).

Now led by Taneli Tikka, CEO, the company strives for growth and professional development. Mr. Tikka was nominated as the ICT person of the year 2010 and is known for his many involvements in and achievements with various Internet companies. As of January 2011 he started as the Chief Executive Officer of SBA. (Soprano Brain Alliance Oy, 2011; Tietotekniikan Liitto ry, n.d.).

The main focus area of this work is on how Drupal's new entities are used in Brain Forum, a discussion forum module developed for a customer. The idea behind the module as well as the database structure, the main features and the custom functions are presented. Also the use of Drupal's different APIs and hooks is described.

Brain Forum is a module for the seventh version of Drupal created by me during the summer of 2011. The module provides a typical discussion forum with common basic functionalities. What makes this module stand out amongst other discussion forum modules written for Drupal is that it makes use of Drupal's entities, which were presented with the launch of Drupal 7 in January 2011. The other modules (see section 4 Ready solutions) are built using the "everything is a node" principle, which was the only way to realize it in earlier versions of Drupal.

The greatest challenge with the project was to find out the best way of using Drupal's entities to create a sustainable and performance-wise well written module that would not be dependent on using nodes and comments, and which could possibly be shared with others as well. There were no other discussion forum modules to take example of. That is really the reason why SBA decided to develop one.

The target group of this document is anyone who is familiar with module development for Drupal 6 or 7 and has a basic knowledge of database structures and object-oriented programming.

2 Customer

The large Finnish media company Aller Media Oy is the publisher of two widely read weekly magazines (7 päivää and Katso) and five specialized monthly magazines (Elle, FIT, Koti ja keittiö, Miss Mix and TOPModel). The target groups of these magazines vary a lot and therefore the company's customer crowd covers a wide range of people. As the company focuses a lot of attention on online visibility, it has branded websites for each of the above mentioned magazines. (Aller Media Oy, n.d.).

Aller Media currently runs their websites using an outdated Java-based platform. Their vision is now to be able to run all their websites on a single platform to enhance the consistency of the website structure and the ability to control all the content on the different sites. The company has chosen to work with SBA to fulfill this vision.

After various discussions it was concluded that the best way to go forward with using Drupal is to build all sites individually but with a certain consistency. The main emphasis when building the sites is kept on reusable features. In practice this means that the site structure is the same on every site, the system configurations are the same, and the content types including their field instances are identical. As shown in Figure 1 each site may have its own special features but will use as many of the shared features as possible.

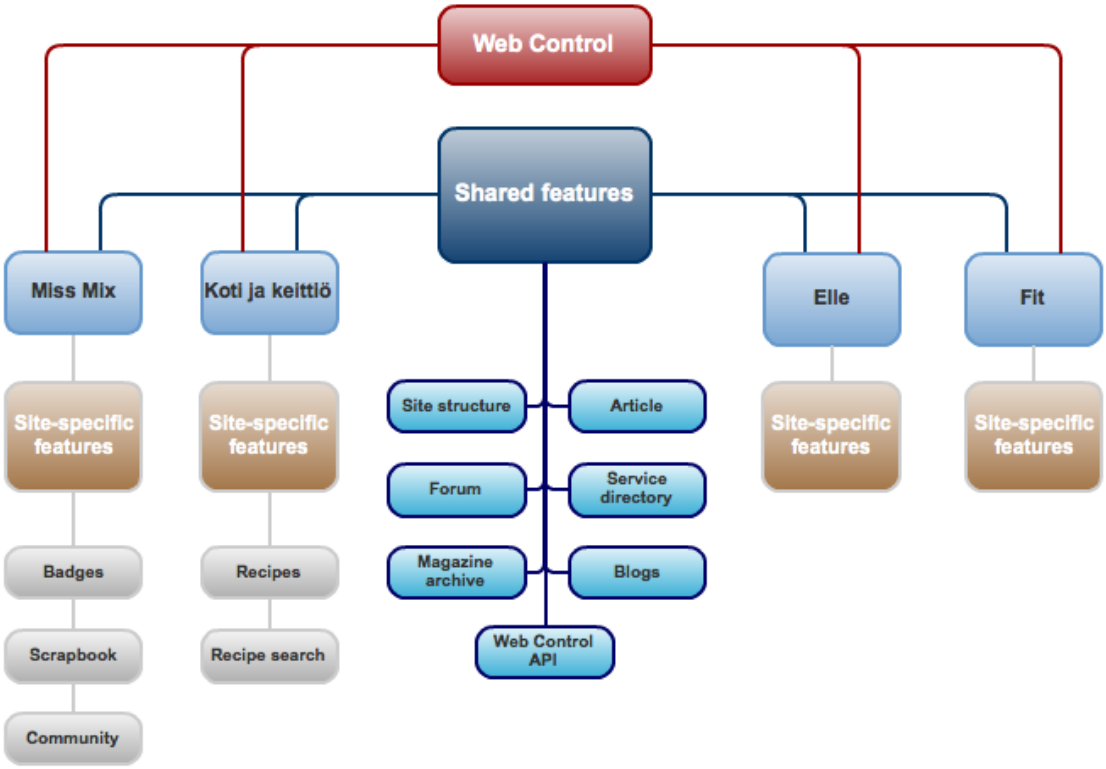


Figure 1. Chart of the structure of Aller Media's Drupal Platform.

The centralized content moderation will be handled via a completely different Drupal site called Web Control. This site includes a set of custom moderator modules specifically built for Aller Media. The modules collect the content to be moderated from all other sites. The site will also provide an API to use when rendering e.g. a "Report inappropriate content" button on each site.

3 Drupal

The open-source software Drupal, which started off as a simple message board in 1999, is one of the most widely used content management systems (CMS) today (Drupal, n.d. e). WordPress and Joomla! are commonly considered its largest competitors (The State Of Drupal, n.d.). What makes Drupal stand out of the group is the fact that it is backed up by a strong online community with over 630,000 users (Drupal, n.d. a). The users contribute, communicate, debate, brainstorm, and most importantly provide support for others.

Drupal's modular system allows for any developer to contribute to the software with his or her own modules and themes, which enriches the flexibility of the software. There are over 8 600 modules (full projects not including sandbox projects) and themes freely available for download on the Drupal Community's website (Drupal, n.d. b).

The latest version of Drupal is Drupal 7, which was published in January 2011. Drupal 6 came out in 2008 and many websites are still using this version since upgrading to the newest version requires some more effort than doing a normal minor update. (Drupal, n.d. d).

3.1 What are entities?

Many improvements have been made since Drupal 6. On the user interface side some of the more significant changes are the Overlay and Toolbar modules. On the technical side the largest change is the transition from nodes, comments, users, etc. to the newly introduced entities.

3.1.1 "Everything is a node"

In Drupal 6 the node was the main abstraction layer on which everything was built. Even though nodes were originally meant to contain text-based content, they were so central in Drupal 6 that there are contributed modules that transform comments and user profiles into nodes to enable the ability to control and modify them even more. Even image galleries are created using nodes on many sites still running Drupal 6.

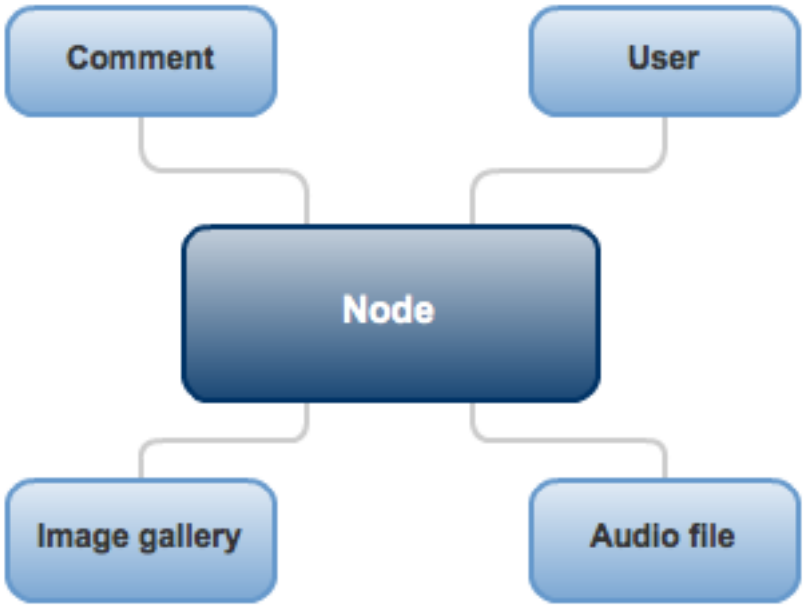


Figure 2. The node was always in the center in Drupal 6.

At the same time as nodes were used for almost everything, developers and users knew that nodes are quite heavy to work with performance-wise and that nodes contain a lot of data that is fully unnecessary in many use cases. What if there was a way to have something that is similar to a node but which could be customized for specific formats of content?

This is where Drupal 7's new entities come in. The introduction of entities changes the whole mentality of Drupal developers. Now everything does not circle around nodes anymore, but around entities.

3.1.2 Entities

The Drupal entities are not to be confused with the entities in entity-relation models (ERM), which are commonly used in software engineering to present and visualize the architectural structure of a database and the relations between its entities. (Chapple, n.d.).

A Drupal entity could be called the equivalent of a node in Drupal 6, with the difference that modules may define their own entity types with specific properties and database tables. Now nodes, comments, users and taxonomy vocabularies are different entity types and they all have the same set of properties that may be modified. This provides a huge amount of consistency to Drupal that has been wanted for a long time. (Drupal, 2011a).

The new concept consists of entity types, bundles and entities. A good way to explain the different components is to take the Taxonomy module as an example. The taxonomy is an entity type. Each vocabulary is a bundle of the entity type. By default a vocabulary bundle contains the fields "Title" and "Description". Each term in a vocabulary is an individual entity. For each entity specific values for each field in its bundle may be set.

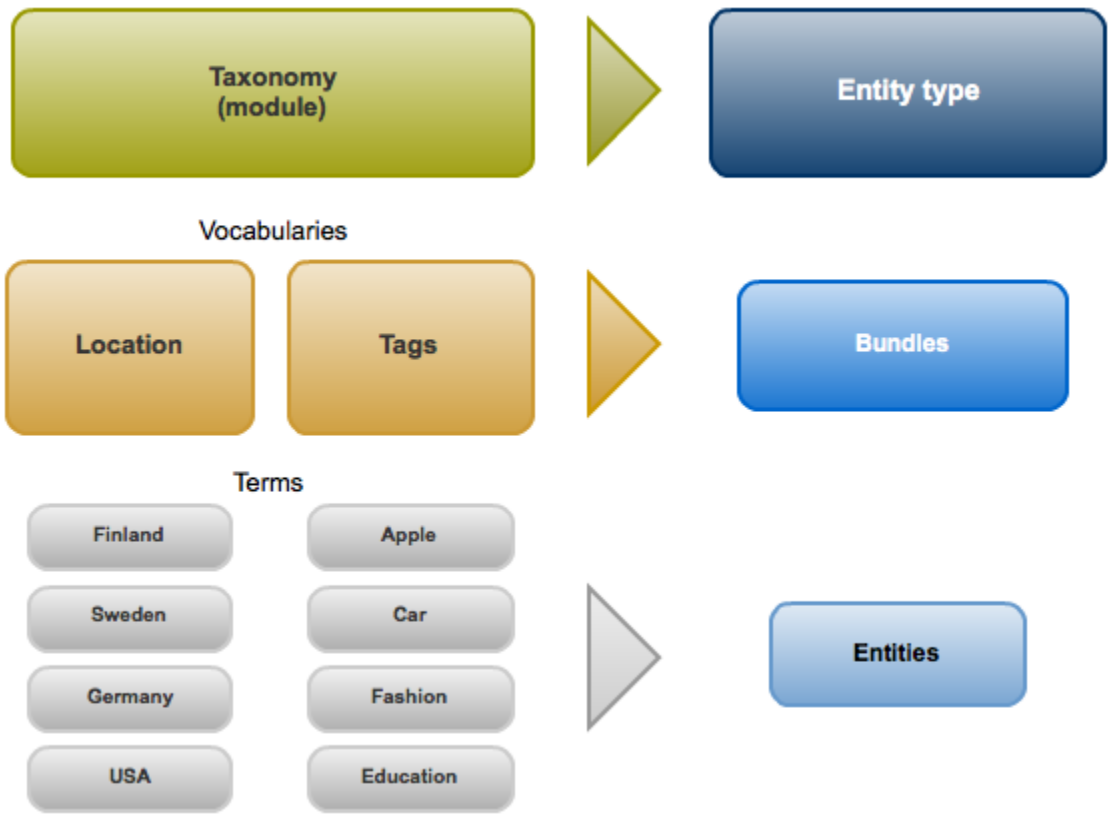


Figure 3. Entity concept breakdown.

The Drupal core only provides one CRUD (Create, Read, Update, Delete) operation: the entity_load() function. The rest of the operations were not ready in time to be included in the core for the launch of Drupal 7. As a result of this the contributed Entity API module provides the rest of the CRUD operations. (Ziegler, 2009).

3.1.4 Fields

Thanks to the Field API fields may be attached to any entity type unless it is specifically defined as not fieldable. This is a big step forward from older versions of Drupal in which fields could only be added to nodes.

The Field API concept consists of field types, widgets and instances. A field type specifies what kind of data is stored, e.g. a string, a number, a file or a Boolean value. Depending on which field type is chosen there are certain widgets available to choose from. A widget is a form element to add and edit the field data with. Examples of widgets are text field, text area, check box, radio button, select list, etc. When attaching a field to a bundle an instance of the field is created. If the same field is added to two different bundles, there are two instances of that field. The field type cannot be changed after a field is created, but the widget may be chosen separately for each instance of the field. (Drupal, 2011d).

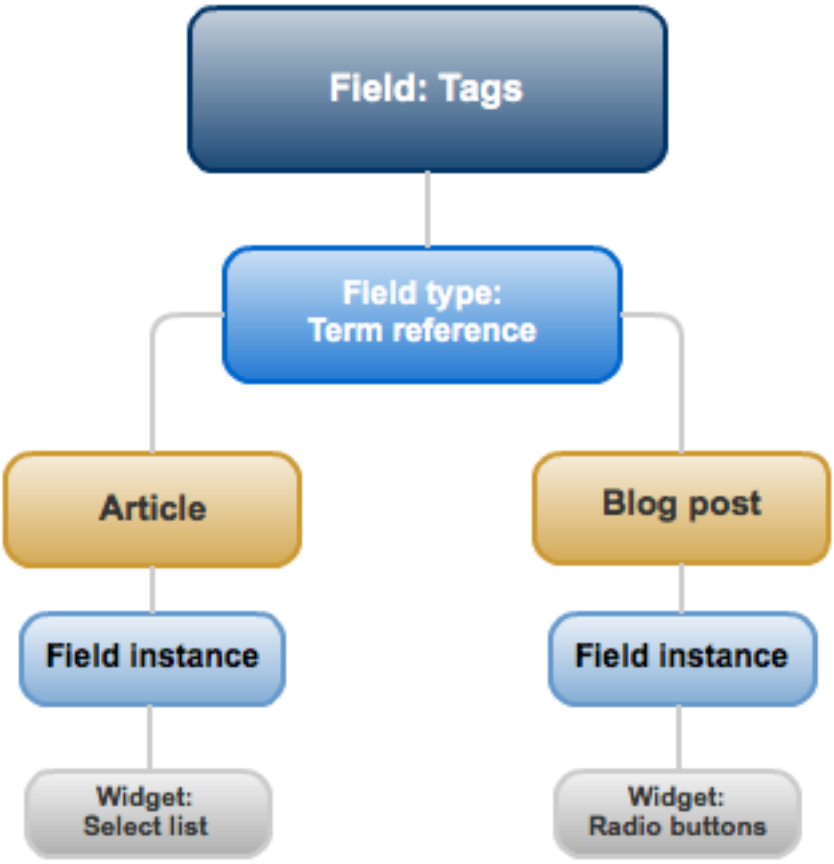


Figure 4. Field API breakdown chart.

4 Ready solutions

4.1 Core Forum

The Drupal core includes the Forum module, which provides very basic discussion forum functionalities. As almost everything else in Drupal 6, the module is based on nodes and additionally their comments. Creating a new node of a certain node type starts a thread, and replying to a thread is basically adding a new comment to the node. The Taxonomy module handles the categorization of threads.

The core Forum module lacks many features found in other "full" online forums. There is no tracking read or unread posts, no "hot" threads indicating which topics are popular, no search engine other than what the Drupal core can provide. In conclusion, there are no other typical discussion forum features other than the possibility to create and reply to threads.

4.2 Advanced Forum

There are a few contributed discussion forum modules available for Drupal of which the most used is Advanced Forum developed by Michelle Cox. Advanced Forum extends the core Forum module with additional functionalities and is also based on nodes, comments and a taxonomy vocabulary. The module was originally created for Drupal 6 and there is now a ported beta version of it available also for Drupal 7. (Cox, 2007; n.d.).

Advanced Forum provides some features found in common stand-alone online discussion forums such as tracking of read and unread posts per user, folder or thread icons, and somewhat better search capabilities than the core Forum has to offer. Together with the Author Pane module it also provides more information about the users who post on the forum, such as online or offline status and number of posts per user. (Cox, 2010).

4.3 Artesian Forum

Artesian Forum is a module that Michelle Cox is planning to develop during 2011 (Cox, 2011b). Even though Advanced Forum was ported to a Drupal 7 version she realized that the use of entities would be best practice. Using nodes and comments is a waste of performance now that there is a more suitable option available.

There is a preliminary plan of how the database structure of an entity-based discussion forum might look and how the entities would be taken into use. The plan and discussion around the topic can be found in the Artesian Forum group on groups.drupal.org.

The idea is to create different entity types for forum categories, threads and posts. Each entity would have its own table in the database. While the post entity's table would contain all required information about individual posts, the thread and forum entities' tables would mostly consist of denormalized data to speed up the database queries, and to enhance the overall performance of the module. (Cox, 2011d).

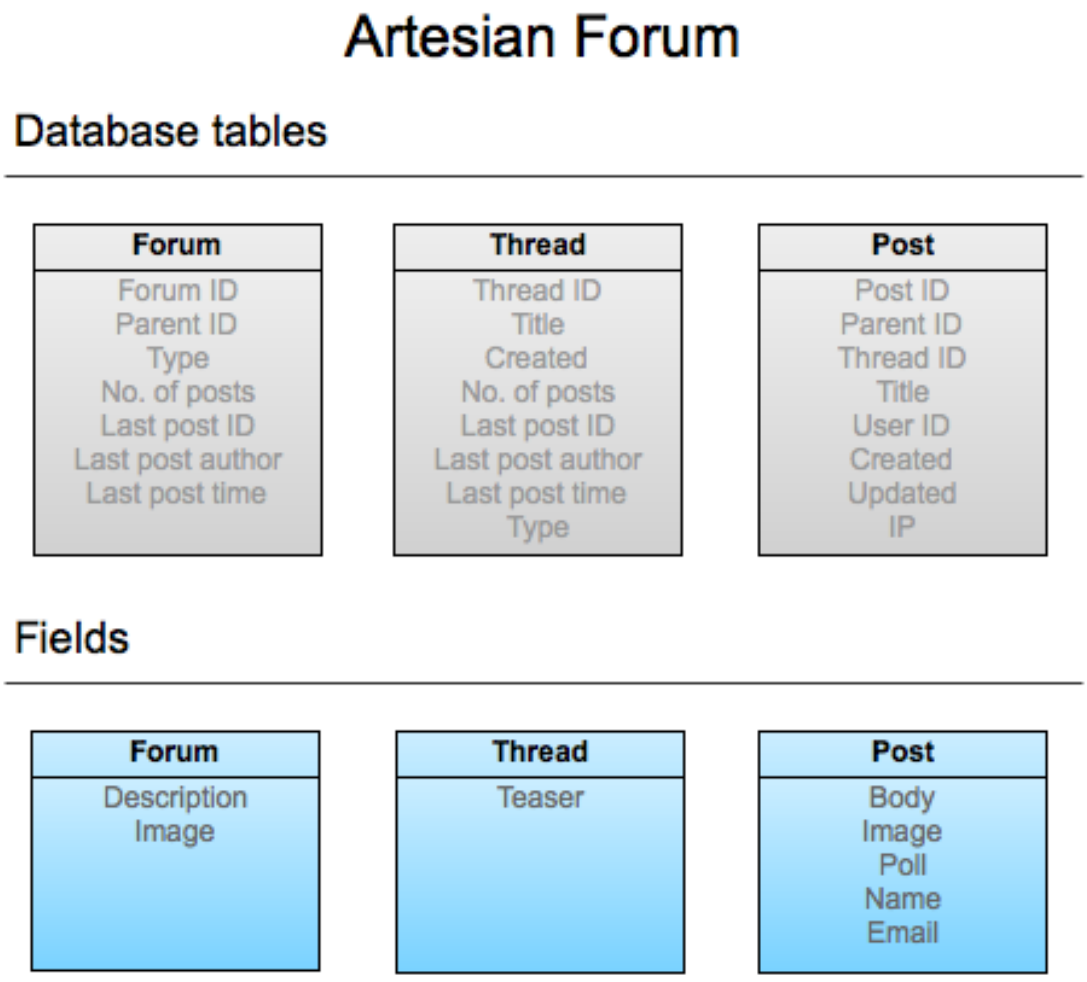


Figure 5. Planned structure of Artesian Forum. (Cox, 2011b).

Michelle Cox first posted the plan of creating Artesian Forum in late May 2011, and has since then announced that the starting of the project is delayed until at least mid September. She also states that the module should not be expected to be released before 2012 since she is not a full-time developer and it is a large project to deal with, especially

since she feels that she is not very familiar with the whole entity concept for the time being. (Cox, 2011a; 2011c).

5 Developing Brain Forum

5.1 What the customer wanted

The original order from the customer was that he wanted an online discussion forum for his website. When the layout for it was provided, it did not was far from a common online discussion forum, but it was rather like a simple news article listing with user comments enabled for each article. This functionality could have been done with Drupal's core modules and Views in a very short time using the mentality "everything is a node", which has been the main way to think for developers before the launch of Drupal 7.

Now that Drupal has its own entities the way of thinking and the mentality of developing Drupal modules have changed. The main abstraction layer is not the node anymore; it is the entity. This is the reason why the customer was convinced that building a new module from scratch would be the best way to go forward. If the forum was built using nodes and comments it would have been quicker, but the performance would have taken a large hit. Additionally developing an entity-based forum module enables having more advanced features, it gives more flexibility and is easier to maintain. If the module is released on drupal.org, which depends on the customer, a lot of contributions can be expected from the community as many users look forward to a forum based on entities. In this way further development and testing of the module can be divided between many people.

5.2 Coding and documentation standards

Documentation of code has a significant role in an open source community. Hence, the Drupal Community has defined certain standards to be followed always when developing something for Drupal. The accepted documentation standard in Drupal is based on the Doxygen documentation system used in various programming languages. Using the Doxygen system also enriches the developing experience when using IDE (Integrated Development Environment) software, such as NetBeans. (Drupal, 2011c; Heesch, 2011; Nourine, 2005).

An example of the documentation standard is that before each function there needs to be a comment block describing briefly what the function does, what parameters it accepts and what the return value will be. If formatted correctly this information can be interpreted by an IDE and shown in a tooltip when the function is going to be used in the code.

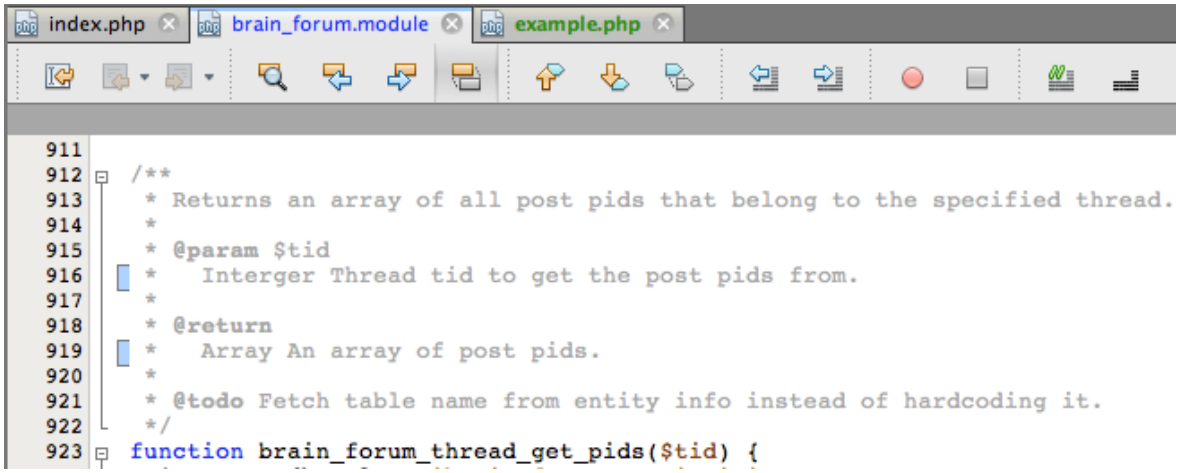


Figure 6. Screenshot of NetBeans showing a Doxygen documented function.

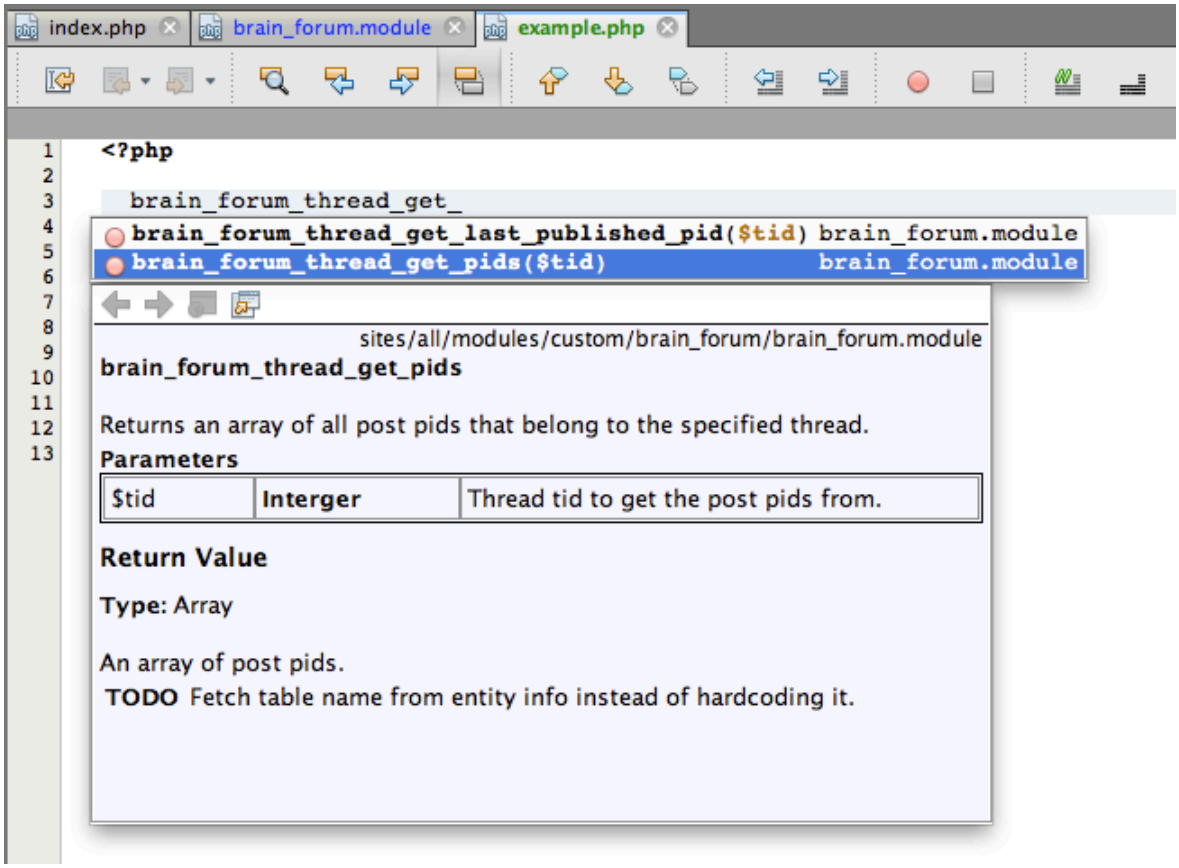


Figure 7. Screenshot of NetBeans showing a tooltip with well-documented functions.

Another important matter is how code is written. There are several different coding standards and it would be very confusing if each module in Drupal were written following a different standard. The Drupal coding standard is "loosely based on the PEAR Coding standards" (Drupal, 2011b). It is quite strict and covers a lot, but it certainly makes reading modules developed by others much easier.

The coding standard defines, amongst many other things, how line breaks should be used, how if- and switch-statements should be built, how functions should be called and even what indentation and line length should be used. Following all these into the smallest detail results in readable and good-looking code. (Drupal, 2011b).

Code 1. Example of how a switch-statement should be structured. (Drupal, 2011b).

```
switch (condition) {
  case 1:
    action1;
    break;

  case 2:
    action2;
    break;

  default:
    defaultaction;
}
```

The code of Brain Forum follows Drupal's instructions in order to enable other developers to contribute, should the module some day be released in the community. During the development process the module was regularly checked with the Coder module to ensure well-written code according to the right standards.

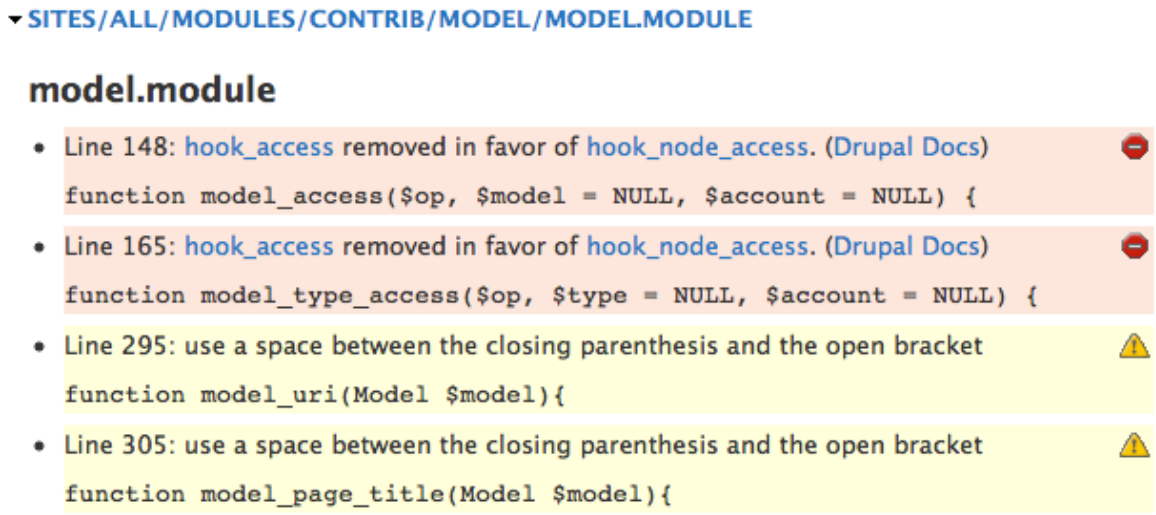


Figure 8. Screenshot of output from the Coder module.

The Coder module is a handy tool for any Drupal module or theme developer. It runs a code review for any given module or theme and shows a log of code snippets with incorrect syntax and gives tips on how they can be fixed to meet the standards. (Power, 2006).

5.3 Features

The customer has a list of features he wishes to have on the discussion forum. Due to the time frame of the project not all features have been implemented yet. This section includes a brief description of what each feature does and how they are technically built and also how the different Drupal APIs are utilized.

5.3.1 Reply

The ability for users to reply to a thread is crucial for a discussion forum to be able to function. The reply form is shown on the thread page below the list of posts. If there are many posts and the thread is broken into multiple pages navigated with a pager, the reply form is shown only on the last page of the thread.

As the threads are shown through a view (see section 5.8 Views integration), the appropriate way to tell Drupal to show the reply form is by hooking into a view display hook. In this case hook_views_pre_render() is used.

After ensuring that the correct view is being loaded (the thread view), that the result of the view is not empty, that thread ID is available and that the user has sufficient privileges to add a reply, Brain Forum adds the reply form as an attachment after the view.

Code 2. Function attaching reply from after view.

```
function brain_forum_views_pre_render(&$view) {
...
  // Attach reply form to view if viewing last page of the view.
  $last_page = $GLOBALS['pager_total'][0]-1;
  if (pager_find_page() == $last_page && user_access('create forum
posts')) {
    $view->attachment_after =
drupal_render(brain_forum_post_add_page($first_post));
  }
...
}
```

In order for users to be able to add replies, a "Reply" link is added to each post as a Views field. The field is first registered in hook_views_data_alter() and it is given its own handler defined in its own include file.

The handler creates an options form with one text field, in which the link title may be defined. The link is built using Drupal's l() function. Should the thread have so many posts that it is paged, the link is pointed directly to the last page of the thread, since that is where the reply form is rendered. Before the link is returned by the handler, user_access() is called to check that the user has permission to create forum posts.

5.3.2 Quote

Quoting other users' posts while replying to a thread is a feature found on almost every discussion forum online. There are several ways of how to implement such a feature. In this module the ID of the post about to be quoted is fetched from the URL. The post in question is loaded and the text is fetched from the message field, wrapped with BBCode quote tags and set as default value for the message area of the reply form. As a security measure Brain Forum checks that the quoted post is published and belongs to the same thread as the reply is being added to.

Code 3. Function for quoting other users' posts.

```
// Get post pid of the post to quote.
$quoted_pid = $_GET['quote'];

// Load the post object to quote.
$quoted_post = post_load($quoted_pid);

// Allow quoting only if the quoted post is published and belongs to the
same thread.
if ($quoted_post->status && ($quoted_post->tid == $form_state['post']-
>tid)) {
  // Use field_get_items() to get the message field, as it is created
through the Field API.
  $quoted_post_message = field_get_items('post', $quoted_post,
'forum_post_message');

  // Wrap the text in quote-tags and populate the message field.
  $form['forum_post_message'][LANGUAGE_NONE][0]['#default_value'] =
'[quote]' . $quoted_post_message[0]['value'] . '[/quote]';
}
```

As in the reply feature, the quote link is also defined as a Views field and has its own include file for the views field handler. The handler is very similar to the reply link's handler, but it adds the post ID as a parameter to the URL the link is pointing to.

Code 4. Custom Views handler for Quote-field.

```
// Build the quote link anchored to the reply form.
// If there is only one page, don't include query '?page=x' in the link.
$return = ($last_page == -0) ? l($text, $current_path, array('query' =>
array('quote' => $pid), 'fragment' => 'brain-forum-post-add-form')) :
l($text, $current_path, array('query' => array('page' => $last_page,
'quote' => $pid), 'fragment' => 'brain-forum-post-add-form'));

// Return link only if user has permission to create posts
if (user_access('create forum posts') && user_access('use text format
bbcode') && filter_format_exists('bbcode')) {
  return $return;
}
```

5.3.3 BBCode support

Due to security risks many discussion forums limit their users' input by stripping away HTML tags from forum posts. Instead Bulletin Board Code (BBCode) is often supported to provide users with a way of formatting their text.

BBCode tags are very similar to HTML tags. They differ with the characters used for wrapping the tags. HTML tags are wrapped with more than (<) and less than (>) characters, whereas in BBCode square brackets ([and]) are used instead. (BBCode.org, n.d.).

Table 1. Comparison of HTML and BBCode tags.

HTML code	BBCode code	Output
bold bold	[b]bold[/b]	**bold**
emphasized emphasized	[i]emphasized[/i]	*emphasized*
 example.com	[url=http://example.com] example.com[/url]	example.com

The risk of allowing users to input HTML tags directly is that they could post harmful code, e.g. inline styles, which could break the page layout or executable JavaScript, which could harm the website itself or its users.

Drupal provides configurable text formats that parse any user input and filter unwanted tags. There is a contributed module for parsing BBCode and converting it to normal HTML when displayed (Naudefj, 2003).

5.3.4 Thread categorization

As threads in a discussion forum may vary in topics, they are usually categorized. This makes it easier for users to browse through a specific topic they are interested in or are seeking information about.

Michelle Cox plan is to build an entity with the purpose of categorizing threads, which is a good long-term solution. Due to the time limit of this project, Brain Forum does not register another entity for categorization. Instead, it uses the Taxonomy module included in Drupal core.

Upon installation the module creates a taxonomy vocabulary named "Forum categories", to which the site administrator can add terms (categories). A taxonomy term reference field is also created using the Field API, and an instance of the field is attached to the thread entity. This is to allow users to choose a category from a select list when starting a new thread.

Code 5. Upon installation a taxonomy term reference field is created and attached to the thread entity.

```
// Create a taxonomy field for forum threads.
$field = array(
  'field_name' => 'forum_category',
  'type' => 'taxonomy_term_reference',
  'settings' => array(
    'allowed_values' => array(
      array(
        'vocabulary' => $vocabulary->machine_name,
        'parent' => 0,
      ),
    ),
  ),
  'cardinality' => 1,
  'translatable' => TRUE,
);
field_create_field($field);

// Attach forum catergory field to forum thread entity.
$instance = array(
  'entity_type' => 'thread',
  'field_name' => 'forum_category',
  'bundle' => 'thread',
  'label' => st('Category'),
  'required' => TRUE,
);
field_create_instance($instance);
```

It is important to remember to include functionality that deletes all fields and their instances that have been created by the module itself. This is done in hook_uninstall() using Field API's functions field_delete_field() and field_delete_instance(). These functions mark the fields and instances for deletion in the database but do not actually erase them from it. The final erasing is done by calling field_purge_batch(), which goes through all the fields and instances in the database and deletes the ones that are marked for deletion.

Code 6. Upon uninstallation all fields created by Brain Forum and their instances are removed.

```
foreach ($instances as $instance_name => $instance) {
  field_delete_instance($instance);
}
field_purge_batch(1000);
```

5.3.6 Sheriff function

A planned but not yet implemented feature is for users to be able to apply for so called sheriff privileges, i.e. forum moderator status. This will allow sheriff users, who are active on the discussion forum, to keep an eye on the threads and look for spam or otherwise inappropriate content, and moderate the content of posts.

The idea is to make the users feel more needed and valuable in order to bring more traffic to the forum, and to keep existing users there, but also to obtain more resources for keeping the forum clean of unwanted content.

5.3.7 Editing of own posts

Another planned feature that will be implemented in the future is to enable users to edit their newly added post or thread during a certain amount of time after the creation of it. This is a somewhat common feature amongst online discussion forums, and it is very useful for users who notice a misspelling in their text right after posting it. This feature would allow users to edit their post instead of adding a new post explaining what they actually meant to say.

The time limit, during which users are allowed to edit their post, will be defined in the forum's configuration page.

5.4 User permissions

In order to control what features are available for certain user roles, different kinds of user permissions may be defined. Most modules define separate permissions for administering the module, for adding content, editing content and, of course, for deleting content. This way the site administrator can allow guests to add content (for example comments) while not having to worry about them being able to either accidentally or intentionally delete anything.

Using the hook_permission() function the Brain Forum defines which permissions should be available for this specific module. This function only adds the permissions to the Permissions page; it does not check whether or not the user has sufficient permissions to take certain actions. (Drupal, n.d. g).

With the permissions defined and set up, the current user's permissions can be checked using the user_access() function. This function takes a string as a parameter, checks if the current user's role has the specific permission assigned to it, and returns a Boolean value representing the result. This enables developers to run the function and ensure a specific user's sufficient permission-level before allowing any action to be taken by the module. (Drupal, n.d. h).

Brain Forum defines separate permissions for creating, editing and deleting forum posts and threads, for moving entire forum threads, for viewing an entire thread, for viewing a list of threads, and also for accessing the module's administration pages.

Code 7. Permissions are defined using hook_permission().

```
function brain_forum_permission() {
  return array(
...
    'create forum threads' => array(
      'title' => t('Create forum threads'),
      'description' => t('Start a new forum thread.'),
    ),
...
  );
}
```

5.5 Database structure

The database structure of the Brain Forum is greatly based on the plan earlier mentioned by Michelle Cox. There are, however, some modifications due to limitation of time and also for serving the purpose of the module better.

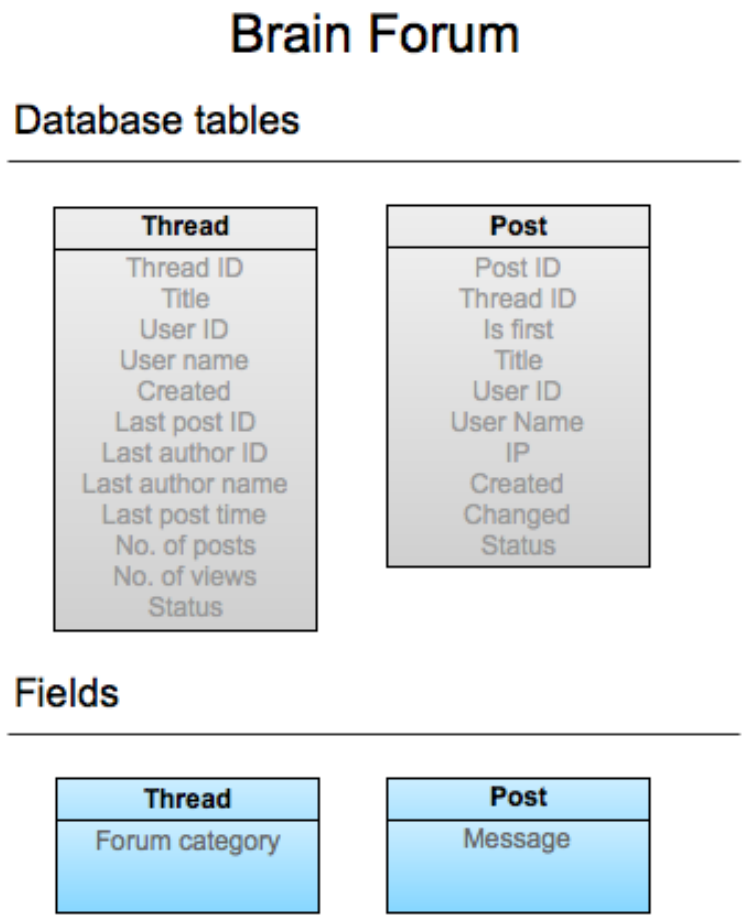


Figure 9. Structure of Brain Forum.

As seen in Figure 9 the module creates two database tables, one for the forum posts and one for the threads. There is no shared base table like the node table for nodes since the module also registers two separate entities with one bundle in each entity (see section 5.6 Entities).

While the post table contains critical data about each individual post, the role of the thread table is to work as a kind of cache table, i.e. its purpose is to cut down on database queries. This is why the table includes data like the number of posts in the thread and information about the last post in the thread. It is common for information about the latest post in a

thread to be shown next to the thread title in thread listings on various discussion forums online.

The titles of threads are stored in both tables. This is also to improve the performance of the forum, since the title is shown both in the thread listing and in the thread itself on top of the posts. If the title was stored in only one of the tables, a query would have to be made for both tables each time the thread listing or a single thread is loaded.

Using this kind of database structure it is important to keep in mind that the so called cache table, in this case the thread table, has to be updated each time data is changed. Such events are when a published post is added or deleted, and when changing the status of the last post in a thread, i.e. publishing or unpublishing it.

As the message field is attached to the post using Field API, the message text of each post is stored in a separate table and from there referenced back to the forum post table.

5.6 Entities

The module registers two entities: the thread entity and the post entity. Following Michelle Cox' plan the post entity contains information about a post, while the thread entity consists of denormalized data about a thread.

The forum entity is omitted since categorization using a Taxonomy module vocabulary is sufficient for the purpose of the module. Michelle Cox' idea of creating a separate entity for forums (or forum containers) is fully usable but with the time frame of this project the forum entity seems like a component that may be left out now and added in the future if necessary.

The entities are registered using the hook_entity_info() function provided by the System module. Using this hook all the properties of an entity are declared. The function returns an associative array with all the necessary information about the entity. (Drupal, n.d. f).

Code 8. Entities are registered using hook_entity_info().

```
function brain_forum_entity_info() {
  $info['post'] = array(
    // Define module for this entity for Views integration to work.
    'module' => 'brain_forum',
    'label' => t('Forum post'),
    'controller class' => 'EntityAPIController',
    'base table' => 'brain_forum_post',
    'fieldable' => TRUE,
    'bundles' => array(
      'post' => array(
        'label' => 'Post',
        'admin' => array(
          'path' => 'admin/structure/brain_forum/manage/post',
          'access arguments' => array('access forum settings'),
...
}
```

In some cases the data type that is going to be saved in the database needs to be more specific than what is allowed to be specified in hook_schema(). Using hook_entity_property_info_alter() the entities' field mappings may be altered. In this case the mappings for some fields are altered to specify that the data stored in them is a date. Some fields are given more descriptive labels and descriptions, which will be visible, for instance, in the Views interface. (Drupal Contrib API, n.d. b).

Code 9. Entity field mappings are altered using hook_entity_info_property_alter().

```
function brain_forum_entity_property_info_alter(&$info) {
...
  // Alter forum post field mappings.
  $post['created']['type'] = 'date';
  $post['created']['label'] = 'Created date';
  $post['created']['description'] = t('The date the post was created.');
  $post['changed']['type'] = 'date';
  $post['changed']['label'] = 'Updated date';
  $post['changed']['description'] = t('The date the post was last
updated.');
...
}
```

5.7 Loading an entity

To load an entity Drupal core provides a function called entity_load(). The function takes the entity type and an array of entity IDs as parameters and returns the entity objects in an array. If no IDs are set, the function returns all entities of the specified type. (Drupal, n.d. c).

Brain forum has its own functions used for loading either a single post or multiple posts; post_load() and post_load_multiple(). Functions for loading threads exist as well, prefixed with "thread" instead of "post".

The post_load_multiple() function accepts an array of entity IDs as a parameter and takes advantage of the core's entity_load() function. The difference here is that the entity type (post) is predefined in the function and does not have to be set as a parameter. The return value is the same as in entity_load().

What post_load() does is that it takes only one integer as a parameter (the ID of the entity to be loaded), puts it in an array and then calls post_load_multiple(). It resets the returned array using PHP's reset() function before returning it further. This way the return value is not an array with a single index but rather the entity object. (The PHP Group, 2011).

Code 10. Custom functions for loading post entities.

```
function post_load($pid = NULL, $reset = FALSE) {
  $pids = (isset($pid) ? array($pid) : array());
  $post = post_load_multiple($pids, $reset);
  return $post ? reset($post) : FALSE;
}

function post_load_multiple($pids = array(), $conditions = array(),
$reset = FALSE) {
  return entity_load('post', $pids, $conditions, $reset);
}
```

5.8 Views integration

There are different ways of presenting data in Drupal. Brain Forum takes advantage of the contributed Views module since it was concluded that using the Views module would not only save time, but also provide site builders with a variety of different options of what data to output and how to format it.

Regarding Views integration, the Entity API module definitely is extremely helpful. By adding a line to the array returned in hook_entity_info() stating which module the defined entity belongs to (see Code 8 above), Entity API allows for almost fully automatic integration between the entity and the Views module. (Drupal, 2011e).

Now all the fields defined in hook_schema() are available as fields in Views. Should any relationships be needed, however, they have to be defined separately using hook_views_data_alter(). In this hook any other necessary data alterations can be included as well. For example, the handler of different fields can be changed, which is most probably needed to be done in any entity-based module with Views integration.

Before any Views data alterations or additions can be added the module has to call hook_views_api() to register itself as a module that uses Views. After this the module can include any files containing Views hooks. (Drupal Contrib API, n.d. c).

5.8.1 Default views

To make using of the module easier for the end user (in this case the end user being a site builder or a developer), Brain forum creates two default views upon installation; the forum_thread view and the forum_listing view. The first one displays an entire thread with all its posts and the latter displays a list of all threads. Including the file brain_forum.views_default.inc, which contains the exported code of both views, creates the views upon installation of the module.

The forum listing view lists the title, author, start date and number of views and replies of each published thread in the specified forum category, or, if no category is specified, the threads from all categories are shown. If available, also the date of when the last reply was added to the thread, is shown. All the data is fetched directly from the brain_forum_thread database table so no queries to the post table are necessary. Reducing the number of database queries is an important factor when considering performance.

The forum thread view displays the title of the thread and then lists all posts in it. The outputted fields are the name and image (if available) of the post author, the created date and the message of the post. Additionally, if the user has sufficient permissions, reply and quote links are shown. Should the user be an administrator or moderator also an edit link is displayed.

5.9 Managing posts and threads

A forum thread is a container for forum posts. When replying to a thread in Brain Forum a reference to the thread is added to the post about to be created. If a reference is not found, it is assumed that a whole new thread is being created. With this logic the first post in a thread always represents the entire thread.

Both the post and the thread entities are defined to be fieldable so any fields may be added to them through the Field API user interface.

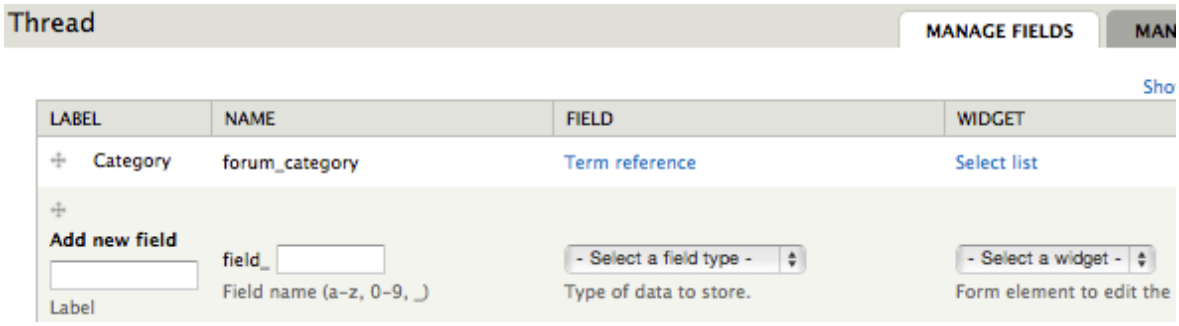


Figure 10. Screenshot of the "Manage fields" page of Brain Forum's thread entity.

5.9.1 Adding posts

The form for adding posts is built in the normal way forms are built in Drupal, using the Form API. The form, as shown to the end user, consists of a username and message. If the user is starting a new thread, also the forum category and title fields are shown. Titles may be added only for threads, not for individual posts. This is the reason why the title field is hidden when replying to an existing thread. Additionally, some other data is saved in hidden fields, such as the thread ID (if available) and the status of the post (published or unpublished).

The username field is shown as an empty text field to anonymous users, and as a link to the user's profile if the user is logged in. Also all the fields attached to the entities through Field API are shown using the field_attach_form() function.

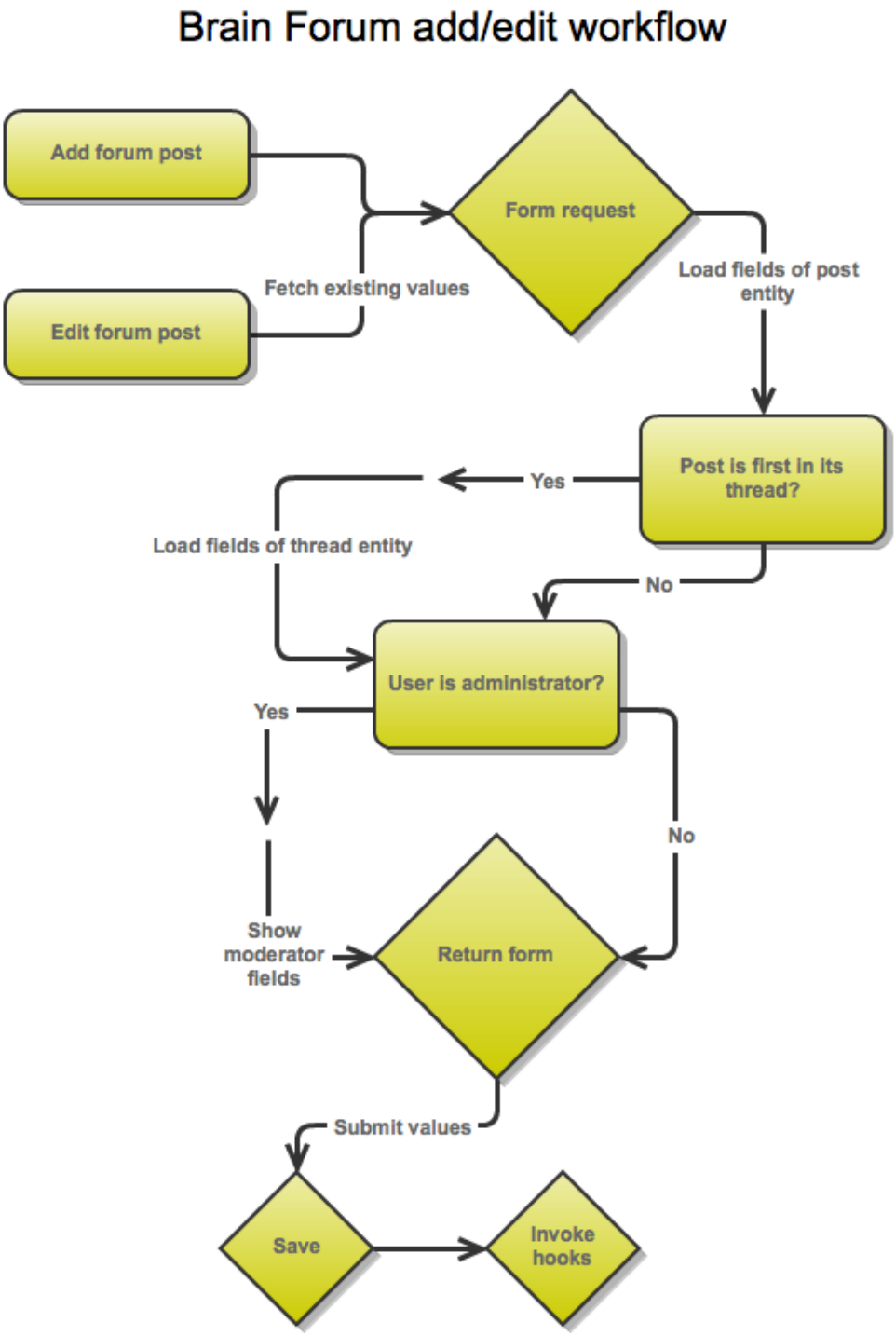


Figure 11. Flowchart of the Brain Forum workflow.

5.9.2 Editing posts

Editing a forum post is similar to adding a post with the difference that the form is prefilled with the existing values of the post being edited. The same form as for adding posts is used but this time the existing post entity is loaded, and the default value for each field is fetched from the entity object.

5.9.3 Post submission

Upon adding or editing a post the submitted values are validated before they are saved in the database. To validate the data in fields added using the Field API the field_attach_form_validate() function provided by the API is used.

After validation the data is sent to a submission function, which builds the entity objects, saves all necessary data in them as properties, and eventually saves them using entity_save() provided by the Entity API. (Drupal Contrib API, n.d. a).

Before saving the entities some additional information is saved in the objects. Taking the value of the REQUEST_TIME constant, defined during Drupal's bootstrap process, sets the created and changed time stamps. The reason why the constant is used instead of PHP's time() function is performance. Since the time is already saved once, it is unnecessary to call yet another function to get it. For convenience the user's IP address is saved in the post entity using Drupal's ip_address() function.

When the entities have been saved, it is time to redirect the user to the thread he or she created or replied to. In order to do this the path of the page display in the default view forum_thread is needed. The path is fetched by loading the view using views_get_view() and then retrieving the value from the returned object. The path is suffixed with the thread ID and the user is redirected by defining the path in redirect property of the form_state array. The path of the view could, of course, be hardcoded but it is best practice to dynamically fetch values that might change.

The last thing happening at a post submission is that the module hooks into the insert and update actions to do some modifications to certain values in the database (see section 5.10 Implemented hooks).

5.9.4 Deleting posts

As in several other Drupal modules the deletion form for posts in Brain Forum is built in using Drupal's confirm_form() function. The form is very simple; it asks the user to confirm whether or not he or she is sure of wanting to delete the post or thread, and it provides a confirm button and a cancel link.

Once the deletion action is confirmed, there is a separate submission function that handles the actual deletion. If the post being deleted is the first post in its thread, it means that the

whole thread should be deleted. In this case the IDs of all posts in the thread are gathered using the custom function brain_forum_thread_get_pids() (see section 5.11 Custom functions). The IDs are passed in an array onto Entity API's function entity_delete_multiple(). Finally a message is shown to the user confirming a successful deletion.

Code 11. Deletion of posts and threads.

```
function brain_forum_post_delete_form_submit($form, &$form_state) {
...
  if ($post->is_first) {
...
    $pids = brain_forum_thread_get_pids($post->tid);
  }
...
  entity_delete_multiple('post', $pids);

  drupal_set_message($message);
  watchdog('Brain Forum', $log_message, array('%title' => $post->title,
'%pid' => $post->pid));

  $form_state['redirect'] = '<front>';
}
```

Also upon deletion the module uses a hook to make necessary modifications to the database (see section 5.10 Implemented hooks).

5.10 Implemented hooks

Brain Forum implements a number of different hooks provided by the Drupal core and contributed modules to interact with them when certain actions are taken. The hooks are used, for example, to register entities, define user permissions and menu paths, to provide default views, and to make some modifications to the database upon adding, editing or deleting posts. The most important implemented hooks as well as the purpose of their usage are presented briefly in this section.

5.10.1 hook_entity_info()

This hook is implemented to register the entities the module uses: the post entity and the thread entity. Without this hook the module would be useless as it is entirely based on entities.

5.10.2 hook_entity_property_info_alter()

This hook is implemented in order to alter the type, label or description of fields in the database.

5.10.3 hook_help()

This hook is implemented to provide users with information about the module, e.g. what it does and where the configuration pages can be found.

5.10.4 hook_menu()

This hook is implemented to register all the menu paths the module is using. These are the paths to the entities' administrator user interface and to the add, edit and delete pages for posts.

5.10.5 hook_permission()

This hook is implemented to provide various user permissions that may be configured in the User Permissions page in Drupal administrator section.

5.10.6 hook_theme()

This hook is implemented to provide predefined templates for the default views that the module is using to present the thread lists and individual threads.

5.10.7 hook_views_pre_render()

This hook is implemented to alter the output of the default view forum_thread provided by the module.

5.10.8 hook_views_api()

This hook is implemented in order for the module's integration with Views to work correctly. Without this hook files in the module's "Views" folder would not be included, which would result in the Views integration not working.

5.10.9 hook_views_pre_render()

This hook is implemented in order to attach the reply form at the end for the thread view. As this hook is invoked each time a thread is viewed, it is also used for incrementing the number of views in a thread. This value is saved in the thread table in the database.

5.10.10 hook_field_attach_form()

This hook is implemented in order for the quote function to work, and to ensure that users are using the BBCode text format if available. The hook is invoked when displaying fields attached to the entities through Field API, i.e. after calling the field_attach_form() function.

5.10.11 hook_{ENTITY_ID}_insert()

This hook is implemented in order to take certain action each time an entity is added. When a published reply is added to a forum thread the information about the last post in the thread has to be updated in the database. In this hook a custom function is called in order to do that. Also the number of posts in the thread is incremented by one.

Code 12. Brain Forum hooks into insertion of post entities.

```
function brain_forum_post_insert($post) {
  if ($post->status && !$post->is_first) {
    // Load the thread associated with the inserted post.
    $thread = thread_load($post->tid);

    // Save information about last post in this thread.
    brain_forum_thread_update_last_post_info($thread, $post);

    // Increment number of posts by 1.
    $thread->posts++;

    entity_save('thread', $thread);
  }
}
```

5.10.12 hook_{ENTITY_ID}_update()

This hook is implemented in order to take certain action each time an existing entity is modified. Upon submitting an existing post (i.e. editing a post) Brain Forum uses this hook to check if the published status of the post is changed. If so, the number of posts in the thread is either incremented or decremented by one depending on the new status of the post. Should the changed post be the latest post in the thread, information about the last published post in the thread is updated in the database.

5.10.13 hook_{ENTITY_ID}_delete()

This hook is implemented in order to take certain action each time an entity is deleted. When deleting the first post in a thread and thus all posts in it, also the thread entity itself is deleted in this hook. If only a single published post in a thread is deleted, the number of posts in the thread is decremented by one.

Code 13. Brain Forum hooks into deletion of post entities.

```
function brain_forum_post_delete($post) {
  if ($post->is_first) {
    // Delete thread.
    entity_delete('thread', $post->tid);
  }
  else {
    // Load the thread associated with the deleted post.
    $thread = thread_load($post->tid);
    if ($thread->posts > 0 && $post->status) {
      // Decrement number of posts by 1.
      $thread->posts--;
    }
    // Update information of last post in thread.
    if ($thread->last_pid == $post->pid) {
      $pid = brain_forum_thread_get_last_published_pid($post->tid);
      $last_post = post_load($pid);
      brain_forum_thread_update_last_post_info($thread, $last_post);
    }

    entity_save('thread', $thread);
  }
}
```

5.11 Custom functions

Some custom functions are included in the module in order to simplify tasks that need to be performed often. The functions are documented following Drupal documentation standards enabling other developers to understand the code and develop the functions further.

5.11.1 brain_forum_thread_get_pids()

This function accepts one parameter of type integer: a thread ID. It does a database query to fetch the IDs of all the post that are associated with the specified thread ID. The return value is an indexed array of the posts' IDs. The function is used, for example, when an entire thread is deleted.

Code 14. Custom function to get all post's IDs from a thread.

```
function brain_forum_thread_get_pids($tid) {
  $query = db_select('brain_forum_post', 'p');
  $query->fields('p', array('pid'))
    ->condition('tid', $tid, '=');

  $result = $query->execute();
  while ($record = $result->fetchAssoc()) {
    $pids[] = $record['pid'];
  }

  return $pids;
}
```

5.11.2 brain_forum_thread_get_last_published_pid()

As with the function described in section 5.11.1, this function also accepts one parameter that is the thread ID. The function does a database query against the brain_forum_post table and fetches the ID of the newest published post in the specified thread. The return value is the post ID as an integer. The function is used, for example, to be able to load the object of the last published post in a thread in order to update the thread table.

Code 15. Custom function to get the ID of the last published post in a thread.

```
function brain_forum_thread_get_last_published_pid($tid) {
  $query = db_select('brain_forum_post', 'p');
  $query->fields('p', array('pid'))
    ->condition('tid', $tid, '=')
    ->condition('status', 1, '=')
    ->orderBy('pid', 'DESC')
    ->range(0, 1);
...
}
```

5.11.3 brain_forum_thread_update_last_post_info()

This function accepts two parameters, a thread entity object and a post entity object. The post object passed to the function is the post from which information is fetched and saved in the thread object. The function simply ensures that both parameters are objects and then assigns the values from the post to the thread. There is no return value as the thread object is passed by reference, ergo it is modified within the function itself. The function is used whenever a published post is added to a thread or when the last post in a thread is either removed or unpublished.

Code 16. Custom function to update information about the last post in a thread.

```
function brain_forum_thread_update_last_post_info(&$thread, $post) {
  if (is_object($thread) && is_object($post)) {
    $thread->last_pid = $post->pid;
    $thread->last_uid = $post->uid;
    $thread->last_name = $post->name;
    $thread->last_time = $post->created;
  }
}
```

5.11.4 brain_forum_thread_reset_last_post_info()

This function accepts only one parameter that is a thread entity object. The object is passed by reference, thus the function is not returning any value. The function ensures the passed parameter to be an object and then resets all the values related to the last post in the thread. The function is used, for example, when the only reply to a thread is either removed or unpublished.

Code 17. Custom function to reset information about the last post in a thread.

```
function brain_forum_thread_reset_last_post_info(&$thread) {
  if (is_object($thread)) {
    $thread->last_pid = 0;
    $thread->last_uid = 0;
    $thread->last_name = '';
    $thread->last_time = 0;
  }
}
```

6 Analysis

6.1 Customer's feedback

The customer Aller Media is satisfied with the solution and has given positive feedback regarding the implemented functionalities and the user-friendliness of the module. Aller looks forward to having the rest of the planned functionalities developed and added to the module, and also to use the module on other sites that are going to be built for them in the near future. (Österlund, 2011).

6.2 Employer's feedback

I have received very positive and constructive feedback from my employer Soprano Brain Alliance Oy, mainly from Drupal Architect Santeri Lindgren who has been leading the whole project. According to him I have shown good technical understanding and dealt very

well with all the challenges. He also states that I have worked systematically "to get the client the best possible functionality with very high quality", and that I never compromised on quality in order to deliver the solution faster. (Lindgren, 2011).

6.3 Personal evaluation

Personally I think the project, i.e. writing the module, was quite successful. The starting point was that I had not written any Drupal module before and Drupal 7 was altogether new for me, as it was for the whole team. I had only read through some online tutorials about developing modules for Drupal 6, but the knowledge gained from those did not offer much help in this project.

The main source of information used was Drupal's website at www.drupal.org, where an enormous amount of documentation about Drupal's functions and hooks can be found. I also received assistance from my colleagues, even though they were not familiar with Drupal 7 from earlier on either.

Certain parts could have been done in a more efficient or practical way. For example, the module could register only one entity, and define posts and threads as different bundles of that entity. This seems to be the way many other contributed entity-based modules are written. Entities are still very new to the community and everyone is trying to work out the best way of using them in module development.

At the beginning of the project I analyzed the source code of various Drupal modules that use entities to see how they were structured and how the entities were taken into use. Nearly all the modules I looked at have the same functions for loading entities, but other than that they were quite different from each other. This led to a lot of testing and trying different structures and doing a great deal of debugging. Finding out how to take advantage of entities in the right way I found challenging but at the same time very valuable.

The decision to use Views for presenting the thread listing and the individual threads was not part of the original plan. I wanted to create a module that would not depend on Views, mostly because of the performance hit it would take, but also because of the fact that Views is not a part of Drupal's core. After e-mailing with Michelle Cox about the idea I changed my mind when she mentioned that it seemed like inventing the wheel again. When considering this afterwards, I have come to the conclusion that using Views was a good decision for many reasons: it allowed me to finish the project faster, I learned how to

integrate a module with Views, and it gives users of the module the possibility to modify the output through the Views interface.

I look forward to developing Brain Forum further with the vision to one day have a "full-blown" Drupal 7 discussion forum module that could be compared to phpBB or similar online forums.

7 Conclusion

With the introduction to entities and the Field API Drupal 7 has made developing modules for it more sustainable than ever before. Now there is a smart way of handling different types of content whether it is text, users, images or other files. Replacing the "everything is a node" principle entities have brought a new way of thinking to the community.

This has been the ground for developing Brain Forum, an entity-based discussion forum module providing basic forum functionalities. Much assistance was offered by Michelle Cox' plan of creating Artesian Forum, and should she need any support in her project, the developers at Soprano Brain Alliance will surely assist her, should Brain Forum not be released for the community.

The long-term goal is to develop the module further to resemble a full-blown discussion forum software. This has long been needed in the Drupal Community, and now it is the first time that it is possible to create such a module without having to use nodes everywhere.

All in all, developing the first version of Brain Forum was successful. Positive and constructive feedback has been received from both the customer and the employer himself.

Sources

Aller Media Oy. (n.d.) *Lyhyesti*.
http://www.aller.fi/yritys/lyhyesti/ (fetched 29.9.2011).

BBCode.org. (n.d.) *BBCode*.
http://www.bbcode.org (fetched 29.9.2011).

Drupal. (2011a) *An Introduction to Entities*.
http://drupal.org/node/1261744 (fetched 29.9.2011).

Drupal. (2011b) *Coding standards*.
http://drupal.org/coding-standards (fetched 29.9.2011).

Drupal. (2011c) *Doxygen and comment formatting conventions*.
http://drupal.org/node/1354 (fetched 29.9.2011).

Drupal. (2011d) *Field API glossary*.
http://drupal.org/node/443540 (fetched 29.9.2011).

Drupal. (2011e) *Views integration*.
http://drupal.org/node/1208874 (fetched 29.9.2011).

Drupal. (n.d. a) *About Drupal*.
http://drupal.org/about (fetched 29.9.2011).

Drupal. (n.d. b) *Download & Extend*.
http://drupal.org/project/modules (fetched 29.9.2011).

Drupal. (n.d. c) *entity_load*.
http://api.drupal.org/api/drupal/includes--common.inc/function/entity_load/7
(fetched 29.9.2011).

Drupal. (n.d. d) *Friendly and powerful: Drupal 7*.
http://drupal.org/drupal-7.0 (fetched 29.9.2011).

Drupal. (n.d. e) *History*.
http://drupal.org/about/history (fetched 29.9.2011).

Drupal. (n.d. f) *hook_entity_info*.
http://api.drupal.org/api/drupal/modules--system--
system.api.php/function/hook_entity_info/7 (fetched 29.9.2011).

Drupal. (n.d. g) *hook_permission*.
http://api.drupal.org/api/drupal/modules--system--
system.api.php/function/hook_permission/7 (fetched 29.9.2011).

Drupal. (n.d. h) *user_access*.
http://api.drupal.org/api/drupal/modules--user--user.module/function/user_access (fetched 29.9.2011).

Drupal Contrib API. (n.d. a) *entity_save*.
http://drupalcontrib.org/api/drupal/contributions--entity--entity.module/function/entity_save/7 (fetched 29.9.2011).

Drupal Contrib API. (n.d. b) *hook_entity_property_info_alter*.
http://drupalcontrib.org/api/drupal/contributions--entity--entity.api.php/function/hook_entity_property_info_alter/7 (fetched 29.9.2011).

Drupal Contrib API. (n.d. c) *hook_views_api*.
http://drupalcontrib.org/api/drupal/contributions--views--docs--views.api.php/function/hook_views_api/7 (fetched 29.9.2011).

Chapple, Mike. (n.d.) *Entity-Relationship Diagram*.
http://databases.about.com/cs/specificproducts/g/er.htm (fetched 29.9.2011).

Cox, Michelle. (2011a) *Artesian Forum*. Drupal.
http://drupal.org/project/artesian (fetched 29.9.2011).

Cox, Michelle. (2011b) *Artesian Forum*. Groups.Drupal.
http://groups.drupal.org/artesian-forum (fetched 29.9.2011).

Cox, Michelle. (2011c) *Artesian Forum helper list*. Drupal.
http://drupal.org/node/1266772 (fetched 29.9.2011).

Cox, Michelle (2011d) *Initial brainstorming (From last May)*. Groups.Drupal.
http://groups.drupal.org/node/145589 (fetched 29.9.2011).

Cox, Michelle. (2010) *Document Advanced Forum's features*. Drupal.
http://drupal.org/node/841056 (fetched 29.9.2011).

Cox, Michelle. (2007) *Advanced Forum*. Drupal.
http://drupal.org/project/advanced_forum (fetched 29.9.2011).

Cox, Michelle (n.d.) *Michelle*. Drupal.
http://drupal.org/user/23570 (fetched 29.9.2011).

Heesch, Dimitri. (2011) *Generate documentation from source code*.
http://www.stack.nl/~dimitri/doxygen/index.html (fetched 29.9.2011).

Lindgren, Santeri. (2011). E-mail (fetched 2.11.2011).

Naudefj. (2003) *Bbcode*. Drupal.
http://drupal.org/project/bbcode (fetched 29.9.2011).

Nourie, Dana. (2005) *Getting Started with an Integrated Development Environment (IDE)*.
http://java.sun.com/developer/technicalArticles/tools/intro.html (fetched 29.9.2011).

Power, Stella. (2006) *Coder*. Drupal.
http://drupal.org/project/coder (fetched 29.9.2011).

Soprano Brain Alliance Oy. (2011) *Taneli Tikka Soprano Brain Alliancen toimitusjohtajaksi*.
http://www.brainalliance.com/taneli_tikka_soprano_brain_alliancen_toimitusjohtajaksi (fetched 29.9.2011).

Soprano Brain Alliance Oy. (n.d. a) *Soprano Brain Alliance briefly in English*.
http://www.brainalliance.com/briefly-english (fetched 29.9.2011).

Soprano Brain Alliance Oy. (n.d. b) *Yhtiö*.
http://www.brainalliance.com/yhtio (fetched 29.9.2011).

The PHP Group. (2011) *reset*.
http://php.net/manual/en/function.reset.php (fetched 29.9.2011).

The State of Drupal (n.d.)
http://london2011.drupal.org/scheduleitem/keynote-state-drupal (fetched 29.9.2011).

Tietotekniikan liitto ry. (n.d.) *Vuoden 2010 vaikuttaja ja tuote*.
http://www.ttlry.fi/vuoden-2010-vaikuttaja-ja-tuote (fetched 29.9.2011).

Zend Technologies Ltd. (n.d.) *PHP Yellow Pages*.
http://www.zend.com/en/store/education/certification/yellow-pages.php#list-cid=75&firstname=&lastname=&orderby=name&sid=XX&company=&photo_first=&certtype=ANYPHP&ClientCandidateID= (fetched 29.9.2011).

Ziegler, Wolfgang. (2009) *Entity API*. Drupal.
http://drupal.org/project/entity (fetched 29.9.2011).

Österlund, Riku. (2011). E-mail (fetched 31.10.2011).

1 Inledning

Detta är en sammanfattning av examensarbetet "Entity-based discussion forum for Drupal 7" av Rasmus Werling.

Arbetet är skrivet som en del av ett projekt utfört av Soprano Brain Alliance, ett finskt webbutvecklingsföretag specialiserat på lösningar skapade med Zend Framework och Drupal.

Huvudsakligt fokus ligger på hur de nya entiteterna i Drupal 7 använts i utvecklingen av Brain Forum, en entitetsbaserad diskussionsforumsmodul skapad på kundens beställning. Grundidén bakom modulen, databasstrukturen, de olika implementerade funktionaliteterna och användningen av Drupals olika API beskrivs.

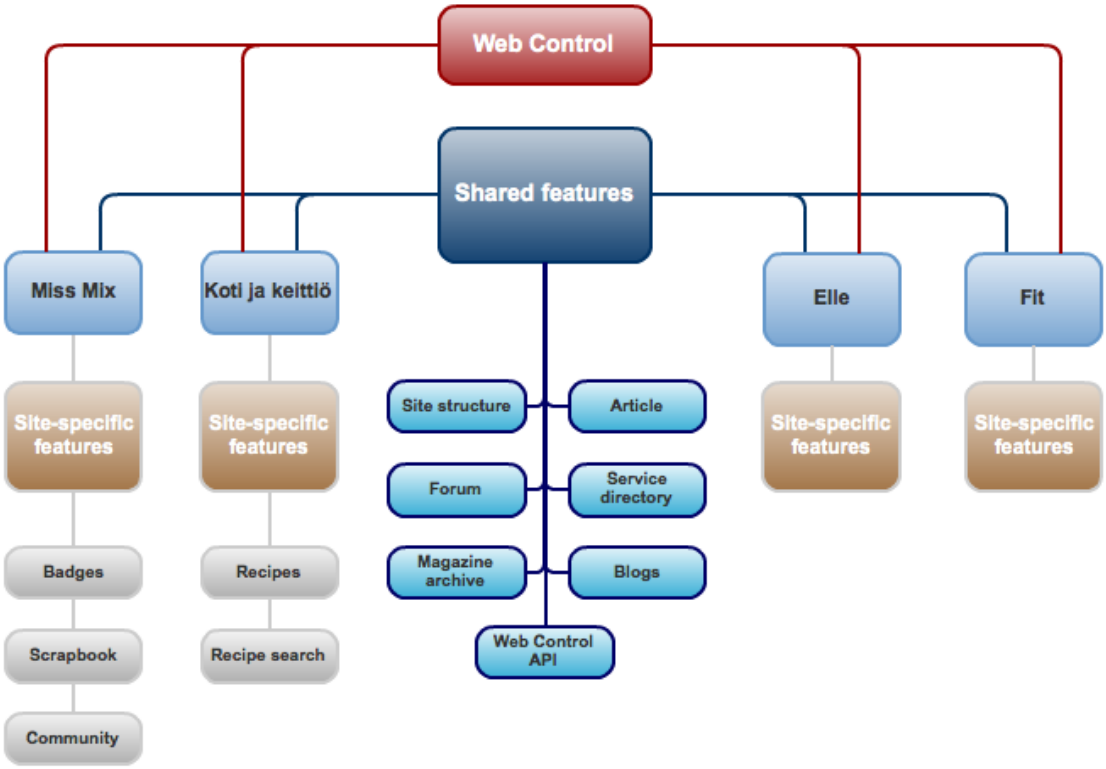
Målgruppen för examensarbetet är främst utvecklare som från tidigare är bekanta med modulutveckling för Drupal 6 eller Drupal 7 samt har grundläggande kunskap inom objektorienterad programmering.

2 Kunden

Det finska mediaföretaget Aller Media Oy äger två stora veckotidningar (7 päivää och Katso) samt fem månadstidningar (Elle, FIT, Koti ja keittiö, Miss Mix och TOPModel). Företagets olika kundsegment har tillsammans en mycket stor omfattning. Eftersom företaget satsar stort på synlighet på nätet, har varje tidning en egen webbplats.

Aller Media beslöt sig för att samarbeta med Soprano Brain Alliance för att slippa sin gamla Java-baserade plattform och för att utveckla en ny plattform med hjälp av Drupal. Visionen är att kunna ha konsekvent uppbyggnad på alla webbplatser och färdigt paketerade funktionaliteter som lätt kan implementeras på de olika sidorna samt även möjliggöra en enkel upprätthållning och uppdatering.

Kunden önskar även ha ett centraliserat modereringsverktyg som kommunicerar med alla de andra webbplatserna.



Figur 12. Översikt över Aller Media's Drupal Platform.

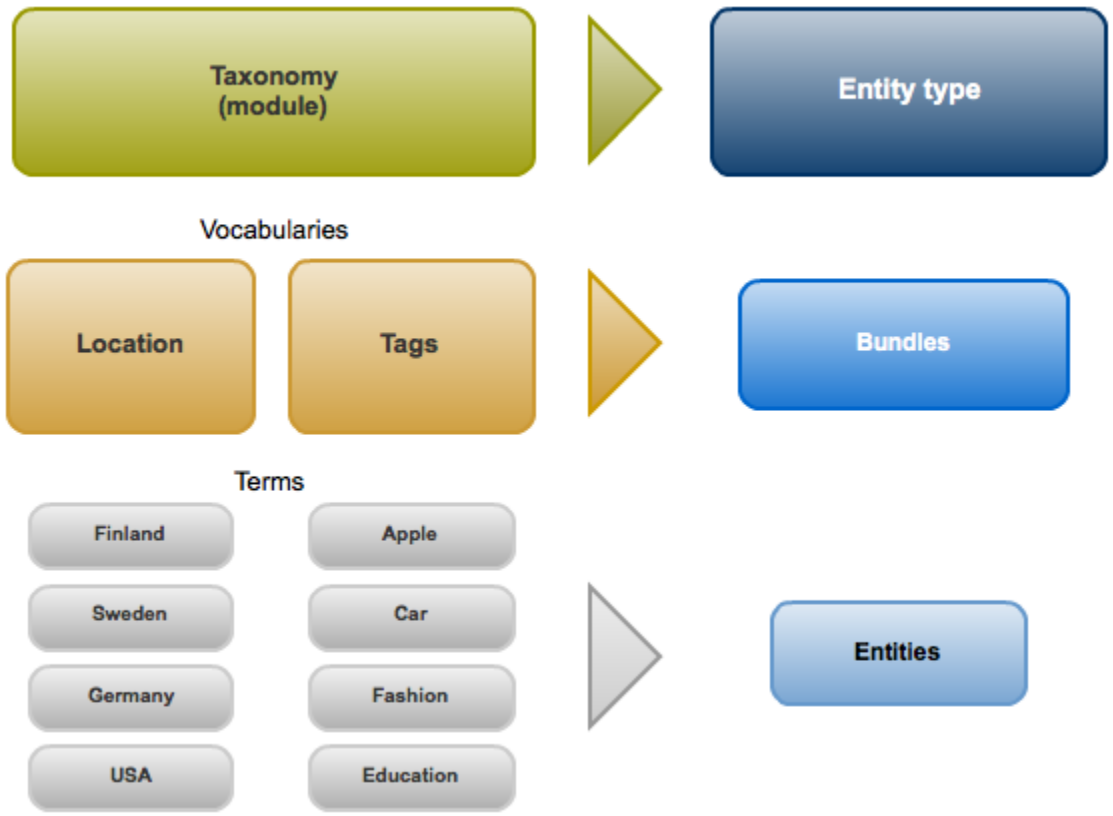
3 Drupal och entiteter

Drupal är ett välkänt innehållshanteringsverktyg gjort med öppen källkod. Bakom verktyget ligger Drupal Community där över 630 000 användare och utvecklare samlas för att bolla med idéer och hjälpa varandra. Det finns över 8600 färdiga moduler att ladda ner samt massvis med teman som alla kan använda gratis. Den senaste versionen av verktyget är Drupal 7 som lanserades i januari 2011.

Drupal 7 introducerade de nya entiteterna som anses vara lösningen till mentaliteten att allt skall cirkulera kring noder. Noden var ursprungligen menad att innehålla textbaserat innehåll, men eftersom det inte verkade finnas någon bättre komponent tillgänglig användes noder för att spara alla sorts data, även bilder och andra sorts filer.

Nu är de olika komponenterna i Drupal indelade i skilda entiteter. Det finns en entitet bland annat för noder, användare, taxonomier och kommentarer. Denna förändring påverkar alla utvecklare och moduler eftersom hela tankesättet över hur data skall lagars har ändrats.

Det nya konceptet omfattar entitetstyper, buntar (bundles) och entiteter. Ett bra sätt att förklara dessa är med hjälp av Taxonomy-modulen. Taxonomy är entitetstypen. Varje ordlista (vocabulary) är en bunt av entitetstypen. Varje term i en ordlista är en individuell entitet.



Figur 2. Konceptet om entiteter.

4 Utveckling av Brain Forum

Det finns några färdiga diskussionsforumsmoduler för Drupal såsom Forum och Advanced Forum, men de baserar sig helt och hållet på noder och kommenterar. Det är detta man vill ändra med att ha entiteter. Därför har Michelle Cox, utvecklaren för dessa två moduler, skissat en plan över hur man skulle kunna förverkliga en modul som tar nytta av entiteterna istället för att använda noder. Modulen hon planerar göra kallar hon för Artesian Forum.

I utvecklandet av Brain Forum användes många delar av Michelles plan. Vissa ändringar gjordes angående vilka entiteter som skulle registreras och hur databastabellerna skulle se ut.

4.1 Funktioner

De viktigaste funktionerna som modulen har är att skapa nya trådar, att kunna svara på dem och att kunna citera andras svar. Eftersom allt innehåll visas med hjälp av Views-modulen skrevs en hel del skräddarsydda Views hanterare (handlers) för att få funktionaliteterna att fungera. Brain Forum stöder även BBCode-modulen som finns färdigt att ladda ner för Drupal.

Kategoriseringen av diskussioner sköts av Taxonomy-modulen. Vid installation skapar Brain Forum en ordlista för kategorier dit en administratör kan lägga till termer. I framtiden kommer detta möjligtvis att ersättas med en skild entitet för forumbehållare.

Vissa av de funktioner som kunden önskat har inte ännu implementerats eftersom tidsramen för projektet var mycket snäv. En av de önskade funktionerna är att användare skall kunna redigera sitt eget inlägg ett visst antal minuter efter att inlägget skapats. Då skulle administratören kunna specificera i en inmatningsruta hur många minuter det får gå innan användaren inte längre kan redigera inlägget. Med denna funktion skulle man slippa onödiga inlägg där folk förklarar att de hittat stavfel i sitt tidigare inlägg.

4.2 Entiteter

Modulen registrerar två entiteter; ett för trådar (threads) och ett för inlägg (posts). För att skapa dem används hook_entity_info() som kommer med System-modulen. Entiteterna laddas genom stödfunktioner som i sin tur använder Drupal's entity_load_multiple() funktion. Båda entiteterna lagrar data i sina egna tabeller i databasen.

4.3 Databasstruktur

Brain Forum skapar vid installation liksom två entiteter också två tabeller i databasen: brain_forum_post och brain_forum_thread. Som namnen säger lagrar den förstnämnda tabellen data om inläggen och den senare data om trådarna.

Den största skillnaden mellan tabellerna är att post-tabellen innehåller information om varje individuellt inlägg, medan thread-tabellen fungerar mer som en cache-tabell. Där sparas information om en hel tråd, till exempel trådens skapare, hur många svar tråden har och vem som lagt till det senaste inlägget i tråden, all sådan information som oftast visas då man listar alla trådar som finns. Orsaken till att man har en egen tabell för detta är att

det förenklar sparandet av information men det drar också ner på de olika databasförfrågningar som behöver köras varje gång listan med trådar renderas.

4.4 Views-integration

Eftersom all information i Brain Forum visas via vyer (views) skapar Brain Forum två färdiga vyer vid installation; en som listar trådar och en som visar alla inlägg i en specifik tråd. En hel del av krokar (hooks) fick användas för att modifiera olika data så att det fungerar ihop med Views.

5 Analys och reflektion

Både kunden och utvecklarens arbetsgivare har varit väldigt nöjda med resultatet och ser fram emot vidare utveckling av modulen. Utvecklaren har fått ta emot positiv feedback av alla som testat modulen, speciellt angående användarvänligheten.

Utvecklaren av Brain Forum är själv också nöjd med projektet. Med tanke på att utgångspunkten var att han inte använt Drupal 7 tidigare och inte utvecklat en hel modul för Drupal tidigare så är han mycket nöjd med slutresultatet. Vissa problem kunde dock ha lösts med mer praktiska och effektiva lösningar. Utvecklaren har lärt mig mycket nytt och kommer att fortsätta utveckla modulen för att få det att blir mer likt ett fullständigt diskussionsforum.

Den huvudsakliga informationskällan var drupal.org och stöd gavs också av arbetskompisar även om de inte heller var bekanta med Drupal 7 från tidigare.

De nya entiteterna i Drupal 7 och den nya Field API möjliggör utvecklandet av mycket mer hållbara moduler än någonsin tidigare. Nu finns det ett effektivt sätt att behandla data i alla olika former, vare sig det är text eller binärt.

Detta har varit grunden för utvecklandet av Brain Forum. Mycket hjälp fick utvecklaren av Michelle Cox som planerat hur ett entitetsbaserat diskussionsforum kunde vara strukturerat. Målet på långsikt är utveckla modulen vidare till ett fullständigt diskussionsforum, vilket länge har varit efterlängtat i Drupal Communityt.