Developer-Dokumentation

Modul-Entwicklung

Buran v1.0

September 2008

Was ist ein Modul genau?

Die grundsätzliche Aufgabe eines Moduls ist es, die Funktionalität einer Buran-Installation auf irgend eine Weise zu erweitern. Dies kann auf mehrere Arten geschehen.

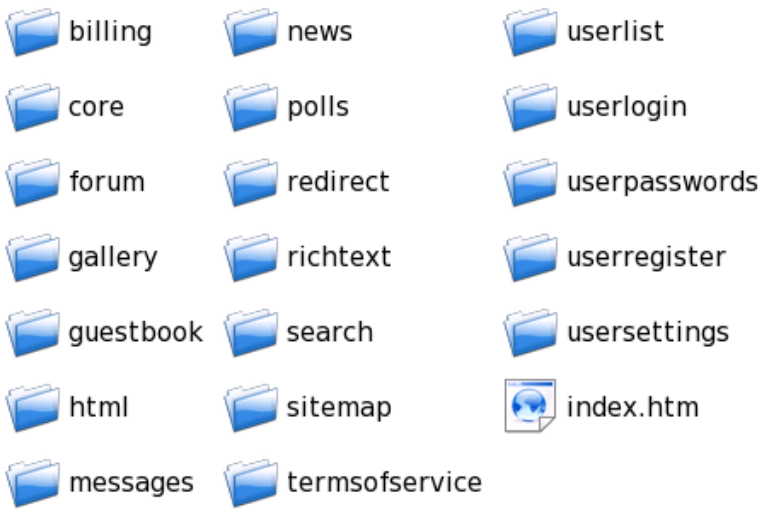
Die meisten Module werden auf einer Seite im Seitenbaum platziert und liefern/verwalten dann deren Inhalt. Beispiele hierfür sind die Module “forum”, “news” oder “polls”.

Andere Module wiederum können auf mehreren Seiten platziert werden, z.B. “richtext”, “redirect” und “html”.

Wiederum andere Module können auf überhaupt keinen Seiten platziert werden und wirken nur im Hintergrund, ohne dem Webseitenbesucher direkt Inhalt zu liefern. Dies ist allerdings eher selten anzutreffen.

Module können ausserdem Blöcke zur Verfügung stellen, die auf der Webseite angezeigt werden (z.B. “polls”). Sie haben auch die Möglichkeit, auf bestimmte Ereignissen in einem Buran-System, z.B. die Erstellung eines neuen Benutzerkontos, zu reagieren.

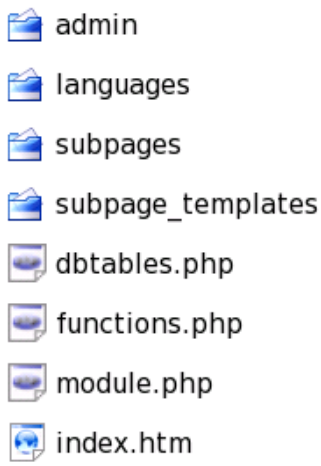
Ein Modul besteht aus einem eigenen Unterverzeichnis im Verzeichnis “modules” und den Dateien darin. Das Verzeichnis muss den gleichen Namen tragen wie das Modul.



Die Verzeichnisse und Dateien, die ein Modul ausmachen, werden im nächsten Abschnitt einzeln vorgestellt und beschrieben.

Die Verzeichnisse & Dateien eines Moduls

Als Beispiel und zur besseren Übersicht hier eine Abbildung der Dateien im Verzeichnis “modules/forum”:



dbtables.php

Diese Datei enthält für jede Datenbanktabelle des Moduls einen Array, der die betreffende Tabelle beschreibt und dem bereits existierenden Array $tables angehängt wird. Beispielsweise etwa so:

```php
// {db_prefix}forum_categories

$table = array();

$table['name'] = '{db_prefix}forum_categories';
$table['is_multilingual'] = true;
$table['fields'] = array();

$table['fields']['id'] = array(
        'name' => 'id',
        'type' => 'serial',
        'primary_key' => true
);

$table['fields']['displayname'] = array(
        'name' => 'displayname',
        'type' => 'varchar(100)'
);

$table['fields']['position'] = array(
        'name' => 'position',
        'type' => 'integer'
);

$tables[] = $table;
```

Ein Array, der eine Tabelle beschreibt, kann/muss über folgende Einträge

verfügen:

name

Der Name der Tabelle. Muss dem Format {db_prefix}*modulname_tabellenname* entsprechen!

is_multilingual

true oder false – regelt die Mehrsprachigkeit der Tabelle. Kann auch weggelassen werden; wird dann als “false” behandelt.
Mehr zum Thema Mehrsprachigkeit folgt später in dieser Dokumentation.

fields

Dies ist wiederum ein Array, der die einzelnen Felder der Datenbanktabelle beschreibt. Jedes im fields-Array beschriebene Feld muss seinen Namen als Key haben und kann folgende Felder enthalten:

name

Der Name des Feldes – muss gleich lauten wie der Array-Key, unter dem dieses Feld definiert wird. Für Beispiele, siehe Screenshot oben.

type

Der Datentyp des Feldes. Folgende Werte sind hier möglich:

- serial – *ein Integer, der mit jedem neuen Datenbankeintrag automatisch erhöht wird. Entspricht also einem MySQL-Integer-Feld mit auto_increment.*
- integer – *ein normaler Integer.*
- numeric – *ein grosser Integer.*
- text – *Text.*
- varchar(***x***) – *eine beliebige Zeichenfolge mit der maximalen Länge **x**.*
- double(***x***) – *eine Kommazahl, wobei **x** die Anzahl*

Kommastellen angibt.

primary_key

Dieser true oder false-Wert legt fest, ob es sich bei dem Feld um den Primary Key der Datenbanktabelle haltet. Kann weggelassen werden, wird dann einfach als “false” behandelt. Muss als “true” angegeben werden, wenn der type des Felds als “serial” angegeben ist.

Unmittelbar an den Anfang dieser Datei muss aus Sicherheitsgründen folgende Schutzzeile gesetzt werden:

```
<?php if(!defined('IN_BURAN')){die('Hacking attempt');}
```

functions.php

Diese Datei beinhaltet Modul-eigene Funktionen, die im Code mehrfach verwendet werden. Kann auch weggelassen werden, wenn sie nicht benötigt wird.

Diese Datei muss, wo sie benötigt wird, per *include_once()* oder *require_once()* manuell eingebunden werden.

Um Überlappungen zu verhindern, empfiehlt es sich, Modul-eigene Funktionen überlegt zu benennen. Im Optimalfall sollte der Namen jeder Funktion mit dem Namen des Moduls beginnen.

Unmittelbar an den Anfang dieser Datei muss aus Sicherheitsgründen folgende Schutzzeile gesetzt werden:

```
<?php if(!defined('IN_BURAN')){die('Hacking attempt');}
```

module.php

Diese Datei enthält mehrere Funktionen, die das Modul mit dem Buran-Kernsystem verknüpfen. Unbedingt vorhanden sein müssen dabei buran_*modulname*_info(), buran_*modulname*_install() und buran_*modulname*_uninstall(), wobei *modulname* durch den Namen des Moduls

zu ersetzen ist.

buran_*modulename*_info() reicht einen Array zurück, der folgende Einträge enthält:

> **name** – der Name des Moduls. Muss gleich lauten wie der Name des Verzeichnisses, in dem sich das Modul befindet.
>
> **Version** – die dreistellige Version des Moduls. Beispiel: 1.0.0
> Die letzte Ziffer muss geändert werden bei neuen Versionen, die nur kleine Veränderungen (z.B. Bugfixes) mit sich bringen, die mittlere Ziffer bei neuer Funktionalität und die erste Ziffer bei starken grundlegenden Veränderungen.

buran_*modulename*_install() ist die Funktion, die bei der Installation des Moduls gerufen wird. Was in dieser Funktion genau geschieht, ist von Modul zu Modul etwas unterschiedlich. Auf jeden Fall muss aber die Funktion buran_module_install() gerufen werden und je nach Erfolg true oder false zurückgereicht werden. Folgendes würde als grundlegende Install-Funktion für ein Modul namens “mymodule” also schon ausreichen:

```
function buran_mymodule_install(){

        $result = true;

        if(!buran_module_install('mymodule', 1)){
                $result = false;
        }

        return $result;

}
```

Die Funktion buran_module_install() wird vom Buran-Kernsystem zur Verfügung gestellt und ist in erster Linie dafür verantwortlich, alle in der Datei dbtables.php festgelegten Datenbanktabellen einzurichten und das Modul im Buran-Kernsystem als aktiv zu registrieren.

Die Funktion nimmt zwei Parameter an, *$module* und *$placement*.

buran_module_install(*$module*, *$placement* **)**

$module ist der Name des zu installierenden Moduls.

$placement ist ein numerischer Wert, der angibt, wo das Modul platziert werden kann (2 = auf mehreren Seiten, 1 = auf nur einer Seite, 0 = auf gar keinen Seiten).

Die dritte Funktion, die in der module.php-Datei auf keinen Fall fehlen darf, ist buran_*modulname*_uninstall(). Diese Funktion wird beim Deinstallieren des Moduls gerufen. Ihre Aufgabe ist es, die Funktion buran_module_uninstall() zu rufen und je nach Erfolg true oder false zurückzureichen. Etwa so:

```
function buran_mymodule_uninstall(){

        $result = true;

        if(!buran_module_uninstall('mymodule')){
                $result = false;
        }

        return $result;

}
```

Die Funktion buran_module_uninstall() nimmt nur ein Argument entgegen, den Namen des zu uninstallierenden Moduls. Ihre Aufgabe ist es, alle Datenbanktabellen, Datenbankeinträge, Cache-Dateien, usw. des Moduls zu löschen und das Modul damit inaktiv zu setzen.

Die module.php-Datei kann je nach Modul auch noch weitere Funktionen enthalten, die das Modul mit bestimmten Teilbereichen des Buran-Systems verknüpfen (z.B. Suchsystem, RSS-Feed, usw). Diese Funktionen werden aber, da sie nicht obligatorisch sind, nicht hier, sondern später, in der Dokumentation zum jeweiligen Teilbereich, erläutert.

Unmittelbar an den Anfang der module.php-Datei muss aus Sicherheitsgründen folgende Schutzzeile gesetzt werden:

```
<?php if(!defined('IN_BURAN')){die('Hacking attempt');}
```

index.htm

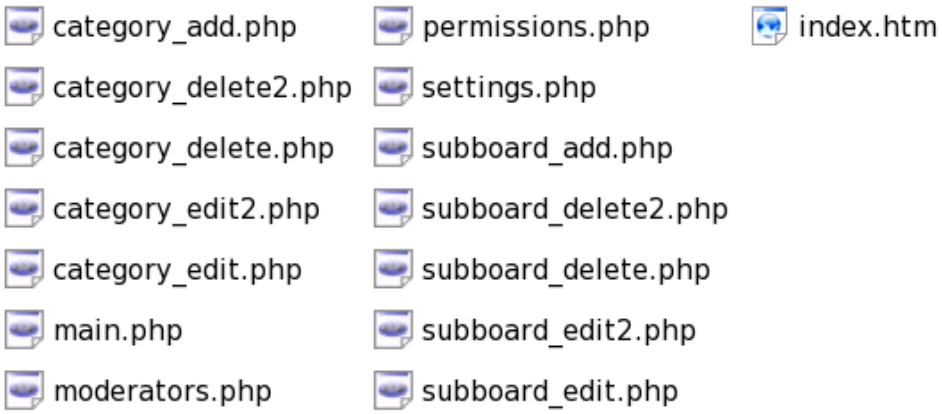
Diese Datei enthält lediglich eine grundlegende HTML-Struktur ohne Inhalt. Sie soll vermeiden, dass bei schlecht konfigurierten Servern einem Besucher der Inhalt des Verzeichnisses angezeigt werden kann.

Sehr viele Verzeichnisse in Buran enthalten solche index.htm-Dateien, sie werden deshalb in Zukunft nicht mehr näher erläutert.

admin/

Jedes Modul, das auf einer oder mehreren Seiten im Seitenbaum platziert werden kann, muss über Admin-Kontrollen verfügen – also das, was im Admin-Panel erscheint, wenn man den Inhalt einer Seite, auf der sich das Modul befindet, bearbeitet. Diese Admin-Kontrollen bestehen aus einer oder mehreren PHP-Dateien im “admin/”-Unterverzeichnis des Moduls, von denen eine “main.php” heissen muss.

Als Beispiel hier der Inhalt des Verzeichnisses “modules/forum/admin/”:



Der Admin-Panel bindet bei der Seitenbearbeitung die benötigten Dateien per include() ein. Die Datei, die standardmässig beim Bearbeiten einer Seite angezeigt wird, ist “main.php”. Über den URL-Parameter “subpage”, der sowohl per GET als auch per POST übergeben werden kann, kann die Auswahl der einzubindenden Datei aber auch manuell festgelegt werden. Lautet er “settings” wird die Datei namens “settings.php” eingebunden (sofern vorhanden), lautet er “main”, wird “main.php” eingebunden, usw.

In den Dateien des “admin/”-Verzeichnisses kann HTML und PHP beliebig vermischt werden. Diese Praktik geniesst zwar keinen sehr guten Ruf, doch an dieser Stelle ist ihr Einsatz aus mehreren Gründen tolerabel.

Zum einen ist der für die Admin-Kontrolle benötigte HTML-Code zumeist sehr simpel aufgebaut und enthält keine komplexen Layoutdaten, kein wirkliches “Design”. Schwierigkeiten mit der HTML/PHP-Vermischung bei etwaigen Redesigns, Überlappungen der Zuständigkeitsbereiche von Designer und Developer oder ein grosser Verlust an Übersichtlichkeit im Code – also das, was was man üblicherweise mit Templates zu vermeiden sucht – sind dadurch praktisch ausgeschlossen.

Zum anderen würde ein Templating-System für die Admin-Kontrollen den Umfang und die Komplexität des Admin-Systems erhöhen, wodurch ein höheres Potenzial für Bugs oder Sicherheitslücken entstünde.

Bei der Verwendung von Links im Code dieser Dateien muss berücksichtigt werden:

- Links innerhalb des Admin-Panels müssen mit “index.php?page=***X***&subpage=***Y***” beginnen, wobei ***X*** mit dem Inhalt der Variable $CONF['page'] zu ersetzen ist und ***Y*** mit der Datei, auf die zugegriffen werden soll (ohne .php-Endung).
- Entsprechend müssen auch alle Formulare den Action-Parameter “index.php” haben und die “page” und “subpage”-Werte übermitteln (üblicherweise unter Verwendung von versteckten Input-Elementen).

Im HTML-Code dieser Dateien kann von folgenden CSS-Klassen gebraucht gemacht werden:

Name	Beschreibung
content_table	Eine normale Tabelle mit Inhalt
content_td	Eine Zelle in einer normalen Tabelle
headerbar_td	Eine Zelle in der Kopfzeile einer normalen Tabelle
separator_td	Eine Zelle, die Abstand zwischen zwei anderen Zellen schafft
breadcrumb	Die Breadcrumb-Navigation oben auf der Seite
statusmessage	Statusbotschaften, die nicht zum festen Inhalt einer Seite gehören (z.B. die “gespeichert”-Meldungen unter dem Editor im Richtext-Modul)
input_textarea	Eine normale Textarea
input_select	Eine normales `<input type="select">`-Element
input_text	Ein normales `<input type="text">`-Element
input_button	Ein normaler Button
input_checkbox	Ein normales `<input type="checkbox">`-Element

Jede Datei im Verzeichnis “admin/” muss aus Sicherheitsgründen an erster Stelle folgenden Schutzcode enthalten:

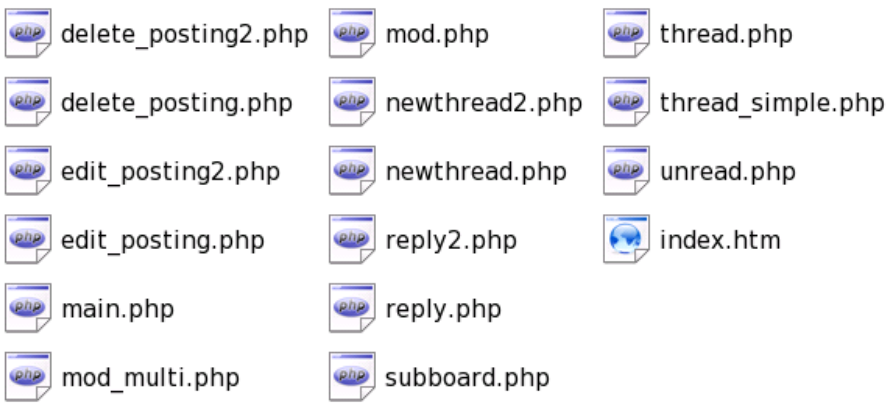
```
<?php
if(!defined('IN_BURAN')){
        die('Hacking attempt');
}
if(!buran_user_isadmin()){
        die('ERROR: You can not use the admin
panel before you are logged in as an
admin.');
}
?>
```

languages/

Dieses Verzeichnis beinhaltet die Sprachpakete der Module. Diese werden nicht hier, sondern im Kapitel “Sprachpakete” näher erklärt.

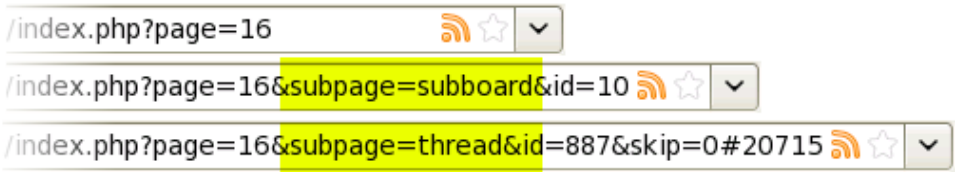
subpages/

In diesem Verzeichnis befinden sich eine oder mehrere PHP-Dateien, von denen eine “main.php” heissen muss. Die “main.php”-Datei wird vom Kernsystem ausgeführt, wenn im Front-End des Buran-Systems eine Seite abgerufen wird, auf der sich das Modul befindet.



Die Auswahl der auszuführenden Datei kann aber über den URL-Parameter

“subpage” gesteuert werden. Dieser kann sowohl per GET als auch per POST übergeben werden.



Mit der obersten URL wird “main.php” geladen, mit der mittleren “subboard.php” und mit der untersten “thread.php”.

Die .php-Dateien des subpage-Verzeichnisses dürfen keinen HTML-Code beinhalten. Ebenso dürfen keine echo-Statements, u.Ä. Verwendet werden. Diese .php-Dateien geben keinen Output zurück, sie zeigen nichts an. In Ihnen werden lediglich Variablen vorbereitet, die dann automatisch in die Templates im “subpage_templates/”-Verzeichnis eingefüllt werden.

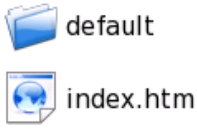
Unmittelbar an den Anfang jeder Datei in diesem Verzeichnis muss aus Sicherheitsgründen folgende Schutzzeile gesetzt werden:

```
<?php if(!defined('IN_BURAN')){die('Hacking attempt');}
```

subpage_templates

Da auf einer Buran-Seite mehrere Themes gleichzeitig aktiv sein können, die sich unter Umständen unterschiedlichen HTML-Codes bedienen um den Inhalt des Moduls anzuzeigen, existiert für jedes Theme ein eigener Satz Templates in einem eigenen Unterverzeichnis von “subpage_templates/”.

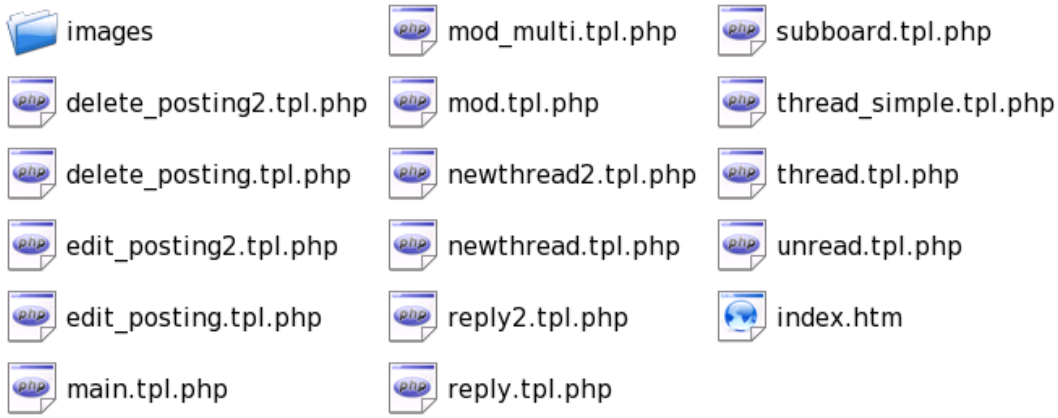
Wenn ein Besucher die Seite z.B. mit einem Theme namens “mytheme” abruft, werden die Templates im Verzeichnis “mytheme” verwendet, sofern vorhanden. Standardmässig in den Modulen mitgeliefert wird jedoch nur der Template-Satz für das “default”-Theme.



Auf den Template-Satz “default” wird auch zurückgegriffen, wenn für ein

Theme kein eigener Template-Satz gefunden wird. Deshalb muss der “default”-Template-Satz in jedem Modul vorhanden sein und darf nicht gelöscht werden.

Werfen wir nun einen Blick auf den Template-Satz “default” des “forum”-Moduls:



Wir sehen hier ein Unterverzeichnis namens “images” – darin befinden sich die im Layout des Moduls verwendeten Grafiken. Dieses Verzeichnis kann auch weggelassen werden, wenn ein Modul keine eigenen Grafiken verwendet.

Was aber viel wichtiger ist: Wir sehen hier für jede .php-Datei im “subpages/”-Verzeichnis eine gleichnamige “.tpl.php”-Datei.

Diese Dateien bestehen inhaltlich grösstenteils aus HTML-Code. In ihnen können alle Variablen wiedergegeben werden, die in der dazugehörigen “subpages/”-Scriptdatei zur Verfügung standen. Nehmen wir an, in einer Scriptdatei stünde folgendes:

```
$foo = "Hello world!";
```

Um den Inhalt einer Variable im dazugehörigen Template wiederzugeben, muss an der gewünschten Stelle im HTML-Code der Name der Variable zwischen zwei Paaren von geschwungenen Klammern eingetragen werden. Z.B:

```
<strong>{{foo}}</strong>
```

Dies würde schlussendlich auf der Webseite dann natürlich etwa so aussehen:

Hello world!

Es ist möglich, PHP in den Templates zu verwenden. Dies darf aber ausschliesslich für Kontroll-Strukturen, also if()-Überprüfungen, foreach()-Schlaufen, usw. verwendet werden. Im Template darf sich kein Teil der Programmlogik befinden, also keine Datenbankabfragen, keine komplexen Funktionsaufrufe, usw. – dies hat sich alles gänzlich in den "subpages/"-Dateien zu befinden!

Die Verwendung von PHP ist hier etwas unorthodox; viele andere CMS stellen in ihren Templates eigene Kontrollstrukturen in Form von eigenen Tags, Kommentaren, etc. zur Verfügung.

Bei der Trennung zwischen Logik und Layout geht es aber um genau dies – eine Trennung zwischen Logik und Layout. Nicht etwa eine Trennung von PHP und HTML. Es spricht also nichts grundsätzliches gegen die Verwendung von PHP in den Templates – solange es sinnvoll eingesetzt wird, sprich, solange sich seine Verwendung der oben beschriebenen Einschränkung unterwirft. Die "selbstgebackenen" Lösungen, die hierfür von vielen anderen Systemen verwendet werden, erhöhen nur den Entwicklungsaufwand und die Komplexität des Systems, ohne Features oder klare Vorteile zu liefern.

Beim verfassen des HTML-Codes dieser Dateien gibt es einige Besonderheiten zu berücksichtigen.

Sogenannte "interne Links" – also Links zu anderen Orten innerhalb des Front-Ends der Webseite, die mit "index.php" beginnen – müssen mit einem sogenannten "URL-Tag" angegeben werden, in folgendem Format:

```
<!--URL index.php?page=5-->
```

Anstatt

```
<a href="index.php?page=5&subpage=main">Link</a>
```

wird also folgendes notiert:

```
<a href="≤!--URL index.php?page=5&subpage=main-->">Link</a>
```

Der Action-Parameter aller Formulare muss folgendermassen lauten:

```
action="≤!--URL index.php?page={{CONF['page']}}-->"
```

Zusätzlich müssen alle Formulare die “page” und “subpage”-Werte übermitteln (üblicherweise unter Verwendung von versteckten Input-Elementen).

Ein Beispiel zu einem korrekt notierten Formular:

```
<form accept-charset="utf-8" method="post"
action="≤!--URL index.php?page={{CONF['page']}}-->">

<input type="hidden" name="page" value="{{CONF['page']}}">
<input type="hidden" name="subpage" value="edit_posting2">

<!-- Rest des Formulars -->

</form>
```

Folgende CSS-Klassen sind in allen Themes vorhanden und eignen sich daher zur Verwendung im HTML-Code der Templates:

Name	Beschreibung
content_table	Eine normale Tabelle mit Inhalt
content_td	Eine Zelle in einer normalen Tabelle
headerbar_td	Eine Zelle in der Kopfzeile einer normalen Tabelle
separator_td	Eine Zelle, die Abstand zwischen zwei anderen Zellen schafft
breadcrumb	Die Breadcrumb-Navigation oben auf der Seite
input_textarea	Eine normale Textarea
input_select	Eine normales `<input type="select">`-Element
input_text	Ein normales `<input type="text">`-Element
input_button	Ein normaler Button
input_checkbox	Ein normales `<input type="checkbox">`-Element

Unmittelbar an den Anfang jeder Datei in diesem Verzeichnis muss aus

Sicherheitsgründen folgende Schutzzeile gesetzt werden:

```
<?php if(!defined('IN_BURAN')){die('Hacking attempt');}
```

Sprachpakete

Alle Texte eines Moduls müssen sich in den Sprachdateien im “languages/”-Verzeichnis des Moduls befinden, damit sie einfach in zusätzliche Sprachen übersetzt werden können.

Diese Sprachdateien sind benannt nach dem Muster lang_***X***.php, wobei ***X*** das Kürzel einer Sprache ist. Die englische Sprachdatei eines Moduls heisst lang_en.php.

Mindestens diese englische Sprachdatei muss in jedem Modul vorhanden sein. Sie wird geladen, wenn eine andere Sprache nicht gefunden werden kann.

Der Code dieser Dateien sieht etwa folgendermassen aus:

```
$LANG['forum_thread_name'] = 'Thread';
$LANG['forum_number_of_postings'] = 'Postings';
$LANG['forum_read'] = 'read';
$LANG['forum_times'] = 'times';
$LANG['forum_overview'] = 'Back to forum overview';
$LANG['forum_new_thread'] = 'New thread';
$LANG['forum_write_a_posting'] = 'Write a posting';
$LANG['forum_write_a_reply'] = 'Write a reply';
$LANG['forum_edit_a_posting'] = 'Edit a posting';
$LANG['forum_emoticons'] = 'Emoticons';
$LANG['forum_bbcode'] = 'BBCode';
$LANG['forum_enter_posting_below'] = 'Enter your posting below:';
$LANG['forum_this_is_posting_preview'] = 'Preview of posting:';
$LANG['forum_announcement'] = 'Announcement';
$LANG['forum_sticky'] = 'Sticky';
$LANG['forum_normal_thread'] = 'Normal thread';
$LANG['forum_button_save'] = 'Save';
```

Wie man hier sehen kann, werden die Strings des Moduls als Einträge in einen Array namens $LANG gespeichert. Die Array Keys müssen dabei mit dem Namen des Moduls beginnen.

Dieser $LANG-Array steht im Code von Subpage-Scripts und Subpage-Templates zur Verfügung. Er kann also im Code des Moduls verwendet werden, und wie andere Variablen mit der üblichen Notationsweise in Templates direkt wiedergegeben werden:

```
<input type="submit" value="{{LANG['forum_button_proceed']}}"
class="input_button">
```

Unmittelbar an den Anfang jeder Sprachdatei muss aus Sicherheitsgründen folgende Schutzzeile gesetzt werden:

```
<?php if(!defined('IN_BURAN')){die('Hacking attempt');}
```

Bei der Entwicklung zu beachten

Lizenz:

Buran wird unter der GPL (General Public License) unter Verfügung gestellt. Jedes Buran-Modul muss ebenfalls unter diese Lizenz gestellt werden. Es ist nicht möglich, ein Buran-Modul zu entwickeln, das nicht unter der GPL steht.

UTF-8-Encoding:

Der gesamte Buran Source-Code ist UTF-8-encodet. Achten Sie also unbedingt darauf, dass Ihr Text-Editor oder IDE beim Arbeiten an Buran-Modulen stehts ebenfalls so eingestellt ist.

Code-Konventionen:

- $_GET und $_POST dürfen im Code eines Moduls nicht verwendet werden. Statt dessen muss die Variable $PARAMS verwendet werden, die den Inhalt von beiden enthält.
- Bei Datenbankzugriffen im Code dürfen die normalen PHP-Datenbank-Funktionen (z.B. mysql_query(), mysql_fetch_array(), usw.) nicht verwendet werden. Statt dessen müssen die Buran-eigenen Datenbankfunktionen verwendet werden. Mehr dazu im Kapitel “Datenbankzugriffe”.
- Anstatt die() oder exit() sollte stets buran_die() verwendet werden.
- Alle Formulare müssen über folgenden Parameter verfügen: `accept-charset="utf-8"`

Buran-eigene Variablen und Defines

Folgende Variablen werden vom Buran-Kernsystem zur Verfügung gestellt und können überall im Code eines Moduls verwendet werden:

$CONF

Dieser Array enthält Informationen zur aktuell geladenen Seite und deren Inhalt, die Konfigurationswerte (Settings) des Kernsystems sowie die Settings des Moduls (sofern vorhanden).
Mindestens folgende Einträge sind stets vorhanden:

Name	Beschreibung
$CONF['module']	Der Name des momentan geladenen Moduls
$CONF['page']	Die ID der momentan geladenen Seite
$CONF['subpage']	Der Name der momentan geladenen Subpage
$CONF['standardlanguage']	Die ID der Standardsprache der Webseite
$CONF['standardlanguagepack']	Der Name des Standard-Sprachpakets der Webseite (z.B. “en”, “de1”, “de2”, usw.)
$CONF['standardtheme']	Der Name des Standardthemes der Webseite
$CONF['standardpage']	Die ID der ersten Seite, die beim Besuchen der Webseite erscheinen soll
$CONF['cache']	Numerischer Wert, der angibt, ob Caching auf der Webseite aktiviert ist. 1 = Ja 0 = Nein
$CONF['userregistration']	Umgang mit Benutzerregistrationen: 0 = Neue Benutzerkonten sind sofort aktiv 1 = Freischaltung nach überprüfen der E-Mail-Adresse 2 = Manuelle Freischaltung durch Admin 3 = Deaktiviert – keine Benutzerregistration möglich
$CONF['adminmail']	Die E-Mail-Adresse des/eines Administrators
$CONF['floodprotect']	Numerischer Wert. Gibt an, wieviele Sekunden zwischen einzelnen Postings eines Benutzers vergehen müssen. 0 = Deaktiviert
$CONF['favicon']	URL des für die Webseite zu verwenden Favicons
$CONF['rss_on']	Ist der RSS-Newsfeed aktiv? 1 = Ja 0 = Nein
$CONF['doubleregister']	Gibt an, ob für eine E-Mail-Adresse mehrere Benutzerkonten eingerichtet werden dürfen: 1 = Ja 0 = Nein
$CONF['flags_on']	Kann von Benutzer-Landesflaggen Gebrauch gemacht werden?

	1 = Ja 0 = Nein
$CONF['emoticons_on']	Kann von Emoticons Gebrauch gemacht werden? 1 = Ja 0 = Nein
$CONF['website_title']	Titel der Webseite
$CONF['website_active']	Ist die Webseite der Öffentlichkeit zugängig? 1 = Ja 0 = Nein
$CONF['website_active_admin_logon']	Falls die Webseite der Öffentlichkeit verschlossen ist: Können Administratoren sich trotzdem einloggen, um die Webseite zu betrachten? 1 = Ja 0 = Nein
$CONF['shorturls']	Soll von suchmaschinenfreundlichen Kurz-URLs Gebrauch gemacht werden? 1 = Ja 0 = Nein
$CONF['set_language_by_url']	Wie soll, sofern die Webseite mehrsprachig ist, die Sprache ausgewählt werden? 0 = Wird in den Einstellung jedes Benutzerkontos einzeln gespeichert 1 = Die Sprache ist Teil der URL und wird darüber festgelegt, nicht über die Benutzerkonten
$CONF['default_blockarea']	Die Blockarea, in der ein Block platziert werden sollte, wenn keine bestimmte Area angegeben wurde
$CONF['content_type']	Welcher Header dem Browser zurückgesendet werden soll. Entweder “html” oder “xhtml”.
$CONF['buranview']	Welches Layout des verwendeten Themes zur Anzeige der momentan geladenen Seite verwendet werden soll. (Ein Theme kann mehrere Layouts enthalten.)
$CONF['buranversion']	Version des Buran-Kernsystems.
$CONF['page_title']	Der Titel der momentan geladenen Seite.
$CONF['page_browsertitle']	Normalerweise leer, wird aber von der Funktion buran_forceTitle() benötigt (siehe Kapitel “Diverse nützliche Funktionen”.

$LANG

Dieser Array enthält die Sprachdaten des Kernsystems sowie die Sprachdaten des Moduls. (Siehe Kapitel “Sprachpakete”.)

$PARAMS

Dieser Array enthält alle $_POST und $_GET-Daten, mit denen die Seite aufgerufen wurde.

$_SESSION

Der $_SESSION-Array wird vom Kernsystem automatisch mit Daten über den momentanen Benutzer gefüllt. Er enthält mindestens folgende Einträge:

Name	Beschreibung
$_SESSION['user_id']	ID des Benutzerkontos
$_SESSION['user_name']	Benutzername
$_SESSION['user_group']	Benutzergruppe
$_SESSION['user_ip']	IP-Adresse
$_SESSION['user_flag']	Landesflagge (sofern Flaggen auf der Webseite aktiviert sind)
$_SESSION['user_posts']	Posting-Zähler
$_SESSION['user_lastlogin']	Datum des letzten Einloggens (als UNIX-Timestamp)
$_SESSION['user_regdate']	Registrierungsdatum (als UNIX-Timestamp)
$_SESSION['user_theme']	Name des vom Benutzer verwendeten Themes
$_SESSION['user_language']	ID der vom Benutzer standardmässig verwendeten Sprache
$_SESSION['buran_languagepack']	Sprachkürzel des vom Benutzer standardmässig verwendeten Sprachpakets (z.B. “en”, “de1”, usw.)
$_SESSION['php_locale']	Zu verwendendes PHP-Locale
$_SESSION['fckeditor_locale']	Zu verwendendes FCKEditor-Sprachpaket
$_SESSION['email']	E-Mail-Adresse
$_SESSION['realname']	Echter Name des Benutzers
$_SESSION['avatar']	URL zum Avatar
$_SESSION['title']	Titel (z.B. “Administrator”)
$_SESSION['sig']	Forum-Signatur

Zusätzlich wird der Inhalt aller zusätzlichen Benutzerprofilfelder in den $_SESSION-Array abgefüllt. (Siehe Kapitel: “Benutzerprofile”.)

Es werden ausserdem folgende Defines vom Buran-Kernsystem zur Verfügung gestellt, die überall im PHP-Code eines Moduls verwendet werden können:

Name	Beschreibung
IN_BURAN	Ist vorhanden und lautet true, wenn das Modul ordnungsgemäss über das Buran-Kernsystem abgerufen wurden. Wenn dieses Define nicht existiert, liegt ein unzulässiger Zugriffsversuch vor.
db_host	Datenbank-Host (zumeist localhost)
db_username	Datenbank-Benutzername
db_password	Datenbank-Passwort
db_dbname	Name der verwendeten Datenbank
db_type	Datenbank-Typ (in der aktuellen Version von Buran immer “mysql”)

db_prefix	Der Datenbank-Prefix, der bei der Installation gesetzt wurde
base_url	URL zum Front-End der Webseite. Endet immer mit einem Schrägstrich. Z.B. http://www.myburansite.com/cms/
base_path	Serverinterner Pfad zu dem Verzeichnis, in dem die Buran-Installation gespeichert ist. Endet immer mit einem Schrägstrich. Z.B. /var/www/html/cms/
base_domain	Domain, auf der sich die Webseite befindet (wird hauptsächlich beim setzen von Cookies verwendet). Bei der URL http://www.myburansite.com/cms/ wäre dies “myburansite.com”.
base_directory	Der in der URL enthaltene Verzeichnispfad zur Buran-Installation. Muss mit einem Schrägstrich beginnen und enden. Bei der URL http://www.myburansite.com/cms/ wäre dies “/cms/”. Befände sich die Buran-Installation direkt im obersten Verzeichnis einer Domain, wäre es “/”.
img_processor	Welches Tool zur Bearbeitung von Grafiken verwendet wird – entweder “gd” oder “imagemagick”
img_convertpath	Pfad zur “convert”-Binary von ImageMagick.

In den Templates eines Moduls kann von diesen Defines allerdings nur “base_url” verwendet werden – einerseits aus sicherheitstechnischen Gründen, andererseits weil die Verwendung der anderen Defines in Templates gar keinen Sinn ergeben würde.

Um base_url in einem Template einzusetzen, wird die selbe Notation wie bei einer Variable verwendet. Beispielsweise also so:

```
<img src="{{base_url}}modules/forum/subpage_templates/default/images/icon.png">
```

Datenbankzugriffe

Buran verfügt über eigene Datenbankfunktionen, die für sämtliche Datenbankzugriffe verwendet werden müssen. Die Verwendung von PHP-Standard-Funktionen wie z.B. mysql_query() ist tabu! Dies aus mehreren Gründen:

- Die Datenbankzugriffe in Buran eng mit den Mehrsprachigkeitsfeatures verknüpft sind und diese Funktionalität würde durch die Verwendung von Standard-Datenbankfunktionen zerstört.
- Die Datenbankfunktionen von Buran verfügen über eingebaute Schutzmechanismen um Hackerangriffe wie SQL-Injections, etc. abzuwehren. Diese fehlen bei den Standardfunktionen.
- In zukünftigen Buran-Versionen sollen nebst MySQL auch andere Datenbanksysteme unterstützt werden, wie z.B. PostgreSQL. Dazu werden eigene, plattform-agnostische Funktionen benötigt, und es gibt Sinn, diese von Anfang an einzusetzen, anstatt später alles umzuschreiben.

Die wichtigste Funktion ist buran_db_query(). In ihrer Funktionalität ist sie das direkte Gegenstück zur PHP-Funktion mysql_query(), angewendet wird sie aber etwas anders.

> **buran_db_query(** *$query* **)**
>
> ***$query*** ist die Datenbank-Query, die ausgeführt werden soll. Vor den Namen der DB-Tabelle, auf die zugegriffen werden soll, muss das Kürzel “{db_prefix}” gestellt werden.
>
> Anders als in mysql_query() dürfen in dieser Funktion keine Variablen direkt eingesetzt werden. Anstelle jeder Variable muss in der Query das Kürzel “{?}” gesetzt werden. Die Variable selbst wird dann als zusätzlicher Parameter der Funktion angegeben.
>
> Hier ein Anwendungsbeispiel zur Veranschaulichung. Was man in einem regulären PHP-Skript etwa folgendermassen erreichen würde...

```
$q1 = mysql_query("SELECT id FROM news_articles WHERE title='$title'
AND category='$category'");
```

..wird in Buran folgendermassen umgesetzt:

```
$q1 = buran_db_query("SELECT id FROM {db_prefix}news_articles WHERE
title='{?}' AND category='{?}'", $title, $category);
```

Alternativ ist es auch möglich, alle in der Query verwendeten Werte in einem Array anzugeben, also folgendermassen:

```
$values = array();

$values[] = $title;
$values[] = $category;

$q1 = buran_db_query("SELECT id FROM {db_prefix}news_articles WHERE
title='{?}' AND category='{?}'", $values);
```

Die meisten anderen Datenbankfunktionen werden genau gleich eingesetzt wie ihre Standardversionen:

buran_db_fetchassoc()
Verwendung analog zu mysql_fetch_assoc().

buran_db_fetchrow()
Verwendung analog zu mysql_fetch_row().

buran_db_fetcharray()
Verwendung analog zu mysql_fetch_array().

buran_db_numrows()
Verwendung analog zu mysql_num_rows().

buran_db_insertid()
Verwendung analog zu mysql_insert_id().

buran_db_error()
Verwendung analog zu mysql_error().

buran_db_free()
Verwendung analog zu mysql_free_result().

Modul mit mehrsprachigem Inhalt

Über die Sprachpakete der Module und den $LANG-Array können die Strings aller Module theoretisch auf beliebig vielen Sprachen angezeigt werden. Die Mehrsprachigkeitsunterstützung von Buran geht aber noch weiter – auch Inhalte können in beliebig vielen Sprachvarianten angelegt und entsprechend auch angezeigt werden.

Technisch gesehen funktioniert die Mehrsprachigkeit so, dass von den Datenbanktabellen eines Moduls mehrere Kopien angelegt werden können – eine für jede in der Buran-Installation vorhandene Sprachvariante. Auf welche Tabelle dann bei Verwendung des Moduls zugegriffen wird, hängt von der Spracheinstellung des individuellen Seitenbenutzers ab.

Um eine Datenbanktabelle eines Moduls mehrsprachig zu machen, muss ihrer Beschreibung in der dbtables.php-Datei folgender Eintrag hinzugefügt werden:

```
$table['is_multilingual'] = true;
```

Das sieht dann z.B. folgendermassen aus (Beispiel aus der dbtables.php-Datei des “news”-Moduls):

```
// {db_prefix}news_categories

$table = array();

$table['name'] = '{db_prefix}news_categories';
$table['charset'] = 'utf8';
$table['is_multilingual'] = true;
$table['fields'] = array();

$table['fields']['id'] = array(
```

Dabei gilt es, darauf zu achten, dass auch wirklich alle benötigten Datenbanktabellen mehrsprachig sind. Ein weiteres Beispiel aus dem news-Modul: Dort gibt es u.A. eine Tabelle namens news_articles und eine namens news_counted. Beide werden dem System bei der Installation als mehrsprachig bekannt gemacht.

Bei news_articles ist sofort klar, wieso: Diese Tabelle beinhaltet News-Artikel,

die dem Benutzer schlussendlich angezeigt werden. Hier ist Inhalt vorhanden, der schlichtweg mehrsprachig sein muss.

news_counted hingegen wird nur verwendet, um den Lesezähler jedes Artikels zu steuern. Wenn ein Besucher einen Artikel liest, wird der Zähler des Artikels erhöht und für die nächste Stunde ein Eintrag mit der ID des Artikels und der ID des Benutzers in news_counted angelegt. Solange dieser Eintrag besteht, wird der Lesezähler nicht erhöht, egal wie oft der Benutzer den Artikel erneut liest. So wird vermieden, dass ein Benutzer den Lesezähler übermässig und unrealistisch in die Höhe treiben kann.

Dies heisst aber auch, dass news_counted nur aus technischen Gründen besteht und eigentlich keinen dem Benutzer direkt anzuzeigenden und dadurch mehrsprachigen Inhalt besitzt. Warum wird die Tabelle dann als mehrsprachig registriert?

Der Grund ist einfach: In news_counted sind ja auch die IDs von Artikeln gespeichert, und Artikel sind mehrsprachig. Die verschiedenen Sprachvarianten von news_articles haben unterschiedliche Einträge mit unterschiedlichen IDs, d.h. nicht jede ID ist in jeder Sprachvariante der Webseite gültig. Dadurch, dass news_counted ebenfalls mehrsprachig ist, wird wirklich nur mit den IDs gearbeitet, die in der aktuell verwendeten Sprachvariante gültig sind.

Die Lektion davon ist, dass nicht nur jede Tabelle mit mehrsprachigem Inhalt, sondern auch jede Tabelle, die sich auf eine Tabelle mit mehrsprachigem Inhalt bezieht, mehrsprachig sein muss. Bei den meisten Modulen läuft dies darauf hinaus, dass es am einfachsten ist, alle Tabellen als mehrsprachig zu deklarieren.

Bei der Deinstallation eines Moduls über die funktion buran_module_uninstall() werden automatisch alle Sprachversionen der Datenbanktabellen des Moduls gelöscht.

Mit Modulen arbeiten

Überprüfen, ob ein Modul installiert ist

Manchmal ist es nötig zu überprüfen ob ein bestimmtes Modul installiert ist. Nehmen wir z.B. an, wir entwickeln gerade ein Modul, dass die Kontaktdaten jedes Benutzers anzeigen soll. Wenn das “Personal messages”-Modul installiert ist, soll es auch möglich sein, dem Benutzer direkt über einen Link eine Message zu schicken. Sollte das “Personal messages”-Modul aber nicht installiert sein, soll an dieser Stelle einfach nichts angezeigt werden.

Um genau eine solche Überprüfung vorzunehmen, gibt es die Funktion buran_module_insinstalled().

> **buran_module_isinstalled(** *$module* **)**
>
> ***$module*** ist der Name des Moduls, das gesucht werden soll.
>
> Wenn das Modul registriert ist, gibt die Funktion *true* zurück; andernfalls *false*.

Überprüfen, auf welcher Seite ein Modul platziert ist

Was aber tun, wenn wir z.B. gerne den Link zu einem bestimmten Modul liefern möchten, aber nicht wissen, auf welcher Seite es installiert ist (und uns verständlicherweise auch nicht darauf verlassen können, dass es sich in jeder Buran-Installation auf derselben Seite befindet)?

Hierzu bedient man sich der Funktion buran_module_getpage().

> **buran_module_getpage(** *$module* **)**
>
> ***$module*** ist der Name des Moduls, das gesucht werden soll.
>
> Die Funktion reicht die ID der Seite, auf der das Modul sich befindet, zurück. Wenn das Modul auf keiner Seite gefunden werden kann, wird *false* zurückgereicht.

Mit Texten & Strings arbeiten

Buran verfügt über eine ganze Reihe von Funktionen zur Formatierung von Text, hier nach Themenbereich aufgeführt.

Formatieren eines Benutzernamen:

buran_text_username(*$string* **)**

Wenn $string der Name eines registrierten Benutzers und das userlist-Modul auf der Webseite installiert ist, wird ein der HTML-Code für den Link zum Profil des Benutzers zurückgereicht.
Wenn nicht, wird $string unverändert zurückgereicht.

Formatieren eines Datums:

buran_text_timestamp(*$string* **)**

$string ist ein UNIX-Timestamp. Zurückgereicht wird ein String mit dem Datum im korrekten Format für die Sprache des Benutzers.

Formatieren einer internen URL:

buran_text_url(*$string* **)**

Wenn $string eine interne URL ist – d.h. eine URL, die mit “index.php” beginnt und somit auf eine Seite innerhalb der Buran-Installation verweist – wird eine absolute Version der URL zurückgereicht. Wenn suchmaschinenfreundliche Kurz-URLs in der Buran-Installation aktiviert sind, wird die URL in ihr suchmaschinenfreundliches Gegenstück umgewandelt.

Wenn es sich bei $string nicht um eine interne URL handelt, wird $string unverändert zurückgereicht.

Jede interne URL muss durch diese Funktion geschickt werden, bevor sie angezeigt oder in Weiterleitungen verwendet wird.

Interne URLs, die in Templates in Buran-URL-Tags angegeben werden, also im Format

```
<!--URL index.php?foo=bar-->
```

werden automatisch an diese Funktion weitergereicht, bevor sie dem Webseitenbesucher angezeigt werden.

Formatieren einer Überschrift oder eines Titels:

buran_text_title(*$string* **)**

Vorgesehen zur Formatierung von Überschriften. Die Funktion entfernt HTML-Tags und PHP-Befehle sowie Begriffe, die im Admin-Panel als verboten festgelegt wurden, aus dem String und reicht ihn dann zurück.

Erstellen eines Vorschautextes oder Textausschnittes:

buran_text_bbexcerpt(*$string, $length=500, $emoticons=true, $bbcode=true***)**

Aus ***$string*** wird ein Vorschautext erstellt, der dann zurückgereicht wird. Im Admin-Panel als verboten festgelegte Begriffe werden herausgefiltert.

Diese Funktion sollte bei evtl. bbCode-haltigen Texten ohne HTML-Texte, z.B. Kommentaren oder Forumpostings, verwendet werden.

Alle Parameter ausser $string sind fakultativ.

$length ist die maximale Zeichenlänge des Vorschautextes (Standardwert 500).

$emoticons und ***$bbcode*** sind Boolean-Werte, die festlegen, ob Emoticons und bbCode-Tags im Text in Grafiken bzw. HTML umgewandelt werden sollen (der Standardwert lautet bei beiden *true*).

buran_text_htmlexcerpt(*$string, $length=500* **)**

Aus ***$string*** wird ein Vorschautext erstellt, der dann zurückgereicht wird. ***$length*** ist die maximale Zeichenlänge des Vorschautextes (als Parameter fakultativ, Standardwert 500).

Sollte angewendet werden, wenn es sich bei $string um HTML-haltigen Text handelt.

Formatieren eines ganzen Textes:

buran_text_bbtext(*$string, $emoticons=true, $bbcode=true***)**

Formatiert den bbCode-haltigen Text ***$string***. Im Admin-Panel als verboten festgelegte Begriffe, HTML-Tags und PHP-Befehle werden herausgefiltert.

$emoticons und ***$bbcode*** sind Booleansche Werte, die festlegen, ob Emoticons und bbCode-Tags im Text in Grafiken bzw. HTML umgewandelt werden sollen (der Standardwert lautet bei beiden *true*).

buran_text_htmltext(*$string* **)**

Formatiert HTML-Text. Im Admin-Panel als verboten festgelegte Begriffe werden herausgefiltert.

In ***$string*** enthaltene Links werden automatisch durch buran_text_url() bearbeitet, bevor sie angezeigt werden.

Formatieren von Text zur Anzeige in einem Formular:

buran_sanitise_toform(*$string* **)**

Wandelt ***$string*** für die Anzeige in einem HTML-Formular um, indem HTML-Sondercharaktere ersetzt werden, und reicht ihn dann zurück.

buran_sanitise_fromform(*$string* **)**

Kehrt die Wirkung von buran_sanitise_toform() wieder um.

Sicherheit

Überprüfen, ob ein Pfad sich innerhalb der Buran-Installation befindet:

buran_sanitise_path($string **)**

Wenn $string kein valider Pfad innerhalb der Buran-Installation ist, wird die Ausführung des Scripts sofort unterbrochen.

Überprüfen, ob eine URL sich innerhalb der Buran-Installation befindet:

buran_sanitise_url($string **)**

Wenn $string keine valide URL innerhalb der Buran-Installation ist, wird die Ausführung des Scripts sofort unterbrochen.

Rich Text Editor

Zum Arbeiten mit Rich Text-Inhalt, also HTML-haltigen Texten, ist das Open-Source-Projekt **FCKeditor** in Buran integriert.
Ausführliche Informationen zum FCKeditor sind auf der offiziellen Projektwebseite zu unter www.fckeditor.net zu finden.

Der FCKeditor kann unter Verwendung folgender Funktion eingesetzt werden:

> **buran_fckeditor_display(** *$name, $value='', $height=400* **)**
>
> Die Funktion reicht den HTML-Code des Editors zurück.
>
> Der Parameter ***$name*** ist der Name, unter dem der Inhalt des Editors nach Abschicken des Formulars im $PARAMS-Array zu finden ist.
>
> ***$value*** ist der im Editor anzuzeigende bzw. zu bearbeitende Text und ***$height*** die gewünschte Höhe des Editors in Pixel. $value und $height sind als Parameter nicht obligatorisch.

Breadcrumb-Navigation

Standardmässig wird auf jeder Seite einer Buran-betriebenen Webseite eine Breadcrumb-Navigation angezeigt, die den Besucher wissen lässt, wo er sich gerade befindet.

Diese Breadcrumb-Navigation mag nicht auf jeder erdenklichen Buran-betriebenen Webseite notwendig sein, aber es handelt sich dennoch um ein nützliches Feature, welches oft Verwendung findet.

Wenn sie in einem Projekt nicht benötigt wird, kann sie immer noch aus dem Template auskommentiert werden. Wäre sie aber nicht überall vorhanden, müsste sie in gewissen Modulen für jedes Projekt, in dem sie benötigt wird, von Hand eingesetzt werden – was unvorteilhaft wäre.

Daher sollte die Breadcrumb-Navigation grundsätzlich von jedem Modul, das öffentlich verbreitet wird, unterstützt werden.

Entwicklungstechnisch benötigt dies keinen grossen Aufwand. Im Script jeder Subpage muss dazu folgender Code vorhanden sein:

```
// Get the breadcrumb navigation
$breadcrumbs = buran_page_breadcrumb($CONF['page']);
```

Und an oberster Stelle jedes Subpage-Templates sollte folgendes stehen:

```
<div class="breadcrumb">
	<?php foreach($breadcrumbs as $breadcrumb){ ?>
		<a href="{{breadcrumb['url']}}">{{breadcrumb['title']}}</a>
		 / 
	<?php } ?>
	<a href="<!--URL index.php?page={{CONF['page']}}-->">{{CONF['page_title']}}</a>
</div>
<p />
```

Dadurch wird überall eine Breadcrumb-Navigation zur aktuellen Seite angezeigt.

Auf den einzelnen Subpages des Moduls sollte dies dann noch um die aktuelle Subpage ergänzt werden. Ein Beispiel aus der Datei “subboard.tpl.php” im Forum-Modul:

```
<div class="breadcrumb">
        <?php foreach($breadcrumbs as $breadcrumb){ ?>
                <a href="{{breadcrumb['url']}}">{{breadcrumb['title']}}</a>
                 / 
        <?php } ?>
        <a href="<!--URL index.php?page={{CONF['page']}}-->">{{CONF['page_title']}}</a>
         / 
         <a href="<!--URL index.php?page={{CONF['page']}}&subpage=subboard&id={{subboard_id}}-->">{{subboard_displayname}}</a>
</div>
<p />
```

Der hier rot markierte zusätzliche Code sorgt in diesem Fall dafür, dass nach dem Breadcrumb-Link für die Forum-Hauptseite noch ein Link für das aktuell betrachtete Subboard angezeigt wird.

Wenn der Besucher dann einen bestimmten Thread aus dem Subboard öffnet, wird das Template thread.tpl.php verwendet, in dem dann zusätzlich noch ein Link für den aktuellen Thread angegeben ist:

```
<div class="breadcrumb">
        <?php foreach($breadcrumbs as $breadcrumb){ ?>
                <a href="{{breadcrumb['url']}}">{{breadcrumb['title']}}</a>
                 / 
        <?php } ?>
        <a href="<!--URL index.php?page={{CONF['page']}}-->">{{CONF['page_title']}}</a>
         / 
         <a href="<!--URL index.php?page={{CONF['page']}}&subpage=subboard&id={{subboard_id}}-->">{{subboard_displayname}}</a>
         / 
         <a href="<!--URL index.php?page={{CONF['page']}}&subpage=thread&id={{thread_id}}-->">{{thread_displayname}}</a>
</div>

<p />
```

Dank diesem einfachen System ist es selten mehr als eine Sache von wenigen

Minuten, eine Breadcrumb-Navigation in ein Modul einzubauen.

Mit hochgeladenen Dateien arbeiten

Ähnlich wie bereits bei den Datenbankfunktionen verfügt Buran über eigene Funktionen zum arbeiten mit hochgeladenen Dateien, die anstelle der PHP-Standard-Funktionen verwendet werden sollten.

Dateien hochladen

Um eine Datei hochzuladen, bedient man sich der Funktion buran_upload_upload().

Nehmen wir an, wir haben eine Datei, die mit dem Namen “userfile” über ein Formular an den Server geschickt wurde, also etwa über folgendes Input-Element:

```
<input type="file" name="userfile">
```

Es gilt nun, diese Datei endgültig auf dem Server zu speichern. Normalerweise würden wir uns hier der PHP-Funktion move_uploaded_file() bedienen. In einem Buran-Script lassen wir diese Funktion jedoch links liegen und verwenden statt dessen buran_upload_upload().

> **buran_upload_upload(** *$file, $allow_overwrite=false, $uploadgroup=1, $path='/', $handle_zip=false, $limit_to_mimetypes=false* **)**
>
> ***$file*** ist der Eintrag der hochzuladenden Datei im $_FILES-Array – in diesem Fall $_FILES['userfile']. Dies ist der einzig obligatorische Parameter dieser Funktion, alle anderen sind fakultativ.
>
> Die hochgeladene Datei wird in das Verzeichnis verschoben, das in ***$path*** angegeben ist – wichtig ist hierbei, dass der Ausgangspunkt von $path das Verzeichnis “uploads” ist. Der Standardwert von $path, '/', ist also “uploads/” selbst. Es ist also nicht möglich, Dateien hochzuladen und ausserhalb des “uploads/”-Verzeichnisses zu speichern.
>
> Wenn ***$allow_overwrite*** auf “true” gesetzt ist, werden im Zielverzeichnis bereits existierende Dateien mit demselben Namen wie

die neu hochgeladene vom Neuankömmling überschrieben. Wenn dies der Fall ist und $allow_overwrite auf “false” steht, wird der Upload hingegen abgebrochen.

$uploadgroup gibt die ID der Uploadgruppe an, zu der die neu hochgeladene Datei gehören soll – standardmässig die Gruppe mit der ID 1, die in jeder Buran-Installation vorhanden ist und nicht gelöscht werden kann.

Wenn ***$handle_zip*** auf “true” gesetzt ist und es sich bei der hochzuladenden Datei um ein ZIP-Archiv handelt, wird das Archiv geöffnet und jede darin enthaltene Datei einzeln als hochgeladene Datei gespeichert.

$limit_to_mimetypes kann ein Array mit zum Upload zulässigen Mimetypes sein. Dateien, die nicht einem dieser Mimetypes entsprechen, werden nicht hochgeladen.
Dieser Parameter kann auch weggelassen werden – in diesem Falle ist sein Wert “false”.

Dateien umbenennen

buran_upload_rename(*$upload_id, $new_name* **)**

$upload_id ist die ID der hochgeladenen Datei und ***$new_name*** der gewünschte neue Dateiname.

Diese Funktion ändert den Namen der Datei und passt alle dazugehörigen Datenbankeinträge an.

Dateien löschen

buran_upload_delete(*$upload_id* **)**

$upload_id ist die ID der hochgeladenen Datei, die gelöscht werden soll. Alle zur Datei gehörigen Datenbankeinträge werden ebenfalls gelöscht.

URL einer hochgeladenen Datei herausfinden

buran_upload_geturl(*$upload_id* **)**

$upload_id ist die ID der hochgeladenen Datei, deren URL in Erfahrung gebracht werden soll.

Die URL wird von der Funktion zurückgereicht. Falls die Datei nicht gefunden werden konnte, wird ein leerer String zurückgereicht.

Alle hochgeladenen Dateien einer bestimmten Gruppe finden

buran_upload_ofgroup(*$uploadgroup_id* **)**

$uploadgroup_id ist die ID der Upload-Gruppe, deren Dateien gefunden werden sollen.

Zurückgereicht wird ein Array, der die IDs der Dateien enthält. Wenn keine Dateien gefunden wurden, ist der Array leer.

Upload-Gruppen hinzufügen

buran_uploadgroup_add(*$displayname, $module, $editable=0, $permitted=''* **)**

$displayname ist der Name der Uploadgruppe.

$module ist der Name des Moduls, das die Uploadgruppe erstellt (gleich wie der Name des Verzeichnisses, in dem sich das Modul befindet, z.B. “forum” oder “news”).

Wenn ***$editable*** auf 1 gesetzt wird, kann die Uploadgruppe von Administratorn über den Upload-Manager manuell bearbeitet werden, beim Standardwert von 0 nicht. Dieser Parameter ist freiwillig.

$permitted ist ein Array mit den IDs aller Benutzergruppen, die die Dateien in dieser Uploadgruppe sehen dürfen. Dieser Parameter ist

freiwillig.

Upload-Gruppen verändern

buran_uploadgroup_update(*$uploadgroup_id, $displayname, $module, $editable, $permitted* **)**

$uploadgroup_id ist die ID der Uploadgruppe, die bearbeitet werden soll.

$displayname ist der neue Name der Uploadgruppe.

$module ist der Name des Moduls, das die Uploadgruppe erstellt (gleich wie der Name des Verzeichnisses, in dem sich das Modul befindet, z.B. “forum” oder “news”).

Wenn ***$editable*** auf 1 gesetzt wird, kann die Uploadgruppe von Administratorn über den Upload-Manager manuell bearbeitet werden, beim Standardwert von 0 nicht. Dieser Parameter ist Fakultativ.

$permitted ist ein Array mit den IDs aller Benutzergruppen, die die Dateien in dieser Uploadgruppe sehen dürfen. Dieser Parameter ist freiwillig.

Upload-Gruppen löschen

buran_uploadgroup_remove(*$uploadgroup_id, $deletefiles=false, $movetarget=1* **)**

$uploadgroup_id ist die ID der Uploadgruppe, die gelöscht werden soll.

Wenn ***$deletefiles*** auf “true” gesetzt ist, werden die Dateien der Gruppe ebenfalls gelöscht. Andernfalls werden sie in eine andere Uploadgruppe verschoben, deren ID in ***$movetarget*** angegeben ist (standardmässig die in allen Buran-Installationen vorhandene Uploadgruppe mit der ID 1).

Alle Upload-Gruppen eines Moduls finden

buran_uploadgroup_ofmodule($module **)**

$module ist der Name des Moduls, dessen Uploadgruppen gefunden werden sollen. Zurückgereicht wird ein Array mit den IDs der gefundenen Gruppen.

Wenn keine Uploadgruppen gefunden wurden, wird ein leerer Array zurückgereicht

Herausfinden, ob der aktuelle Benutzer Zugriff auf eine bestimmte Dateigruppe hat

buran_uploadgroup_ispermitted(*$uploadgroup_id* **)**

$uploadgroup_id ist die ID der Uploadgruppe, die überprüft werden soll.

Wenn der aktuelle Benutzer auf die Dateien in dieser Gruppe zugreifen darf, reicht diese Funktion 1 zurück, andernfalls 0.

Dateiauswahl im Admin-Panel

In den Admin-Kontrollen mancher Module trifft man ein Buran-eigenes Formularelement an, das der Auswahl hochgeladener Dateien dient.

Den Formularelement, den “file selector”, in ein Formular einzusetzen, ist nicht sonderlich aufwändig – es muss lediglich an der betreffenden Stelle im Code die Funktion buran_admin_fileselector() eingesetzt werden:

buran_admin_fileselector(*$LANG, $name, $upload_id=0* **)**

$LANG ist der unveränderte $LANG-Array, in dem sich die Sprachdaten des Moduls befinden.

$name ist der Name des Formular-Elements, unter dem der ausgewählte Wert nach Abschicken des Formulars im $PARAMS-Array abgerufen werden kann.

$upload_id ist die ID des Uploads, der standardmässig als ausgewählt angezeigt werden soll. Dieser Parameter ist fakultativ.

Wenn also an der betreffenden Stelle im Formular folgender Code eingesetzt wird...

```
<?php buran_admin_fileselector($LANG, 'preview_image'); ?>
```

...kann auf der nächsten Seite der gewählte Wert über

```
$PARAMS['preview_image']
```

abgerufen werden. Der Wert ist dabei die ID der ausgewählten Datei, oder 0, falls keine Datei ausgewählt wurde.

Mit hochgeladenen Grafiken arbeiten

Durch Buran hochgeladene Grafiken vergrössern oder verkleinern

buran_image_resizeUpload(*$upload_id, $newsize=150* **)**

$upload_id ist die ID der hochgeladenen Grafikdatei, von der eine Version mit neuen Dimensionen erzeugt werden soll.

$newsize gibt die gewünschte neue Grösse der Grafik in Pixel an. Der Wert wird dabei auf die längste Seite des Bildes bezogen, d.h. er wird je nach geometrischem Format des Bildes entweder auf Länge oder Breite bezogen. Wenn hier also z.B. 150 angegeben wird (der Standardwert), übersteigt die neue Version des Bildes auf keiner Seite die Länge von 150 Pixeln.

Das ursprüngliche Bild wird durch buran_image_resize() nicht verändert; statt dessen wird eine neue Version des Bildes erzeugt, die im “temp”-Verzeichnis gespeichert wird und deren Namen von der Funktion zurückgereicht wird.

Nicht durch Buran hochgeladene Grafiken vergrössern oder verkleinern

buran_image_resizeImage(*$original_location, $resized_location, $resized_x, $resized_y, $mimetype=*'' **)**

$original_location ist der Pfad zu der Grafikdatei auf dem Server, die verkleinert oder vergrössert werden soll.

$resized_location ist der Pfad, an dem sich die veränderte Version befinden soll.

$resized_x ist die neue Breite des Bildes in Pixeln.

$resized_y ist die neue höhe des Bildes in Pixeln.

$mimetype ist der Mimetype des zu bearbeitenden Bilds – kann aber auch weggelassen werden und wird dann automatisch ermittelt.

CAPTCHAs verwenden

CAPTCHA *(**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part)* sind die im Internet oft anzutreffenden Spam-Schutzvorrichtungen bei Formularen, bei denen in der Regel Text von einem Bild abgelesen und in eine Textbox eingegeben werden muss. Da die Bilder für Spambots nicht ohne weiteres lesbar sind, scheitern diese bei der Absendung des Formulars.

Buran bietet Funktionen zur Erzeugung und Verwendung einfacher CAPTCHAs. Im Script der Subpage, die das zu schützende Formular liefert, sollte etwa folgendes eingefügt werden:

```
// Generate a CAPTCHA
$captcha = buran_captcha_create();
```

Und in das Template der betreffenden Subpage an der Stelle, an der das CAPTCHA angezeigt werden soll, etwa folgendes:

```
<strong>Spamschutz</strong>
<br />
<img src="{{captcha['url']}}" alt="captcha" />
<br />
<input type="text" name="captcha" class="input_text" value="" style="width: 200px;" />
<input type="hidden" name="captcha_id" value="{{captcha['id']}}">
```

Damit wird das CAPTCHA erzeugt und angezeigt. Um das CAPTCHA bei der Absendung des Formulars dann zu validieren, muss im Script der Subpage, die

die Formulardaten entgegennimmt, noch etwa folgendes stehen:

```
// Handle CAPTCHAs
if(buran_captcha_check($PARAMS['captcha_id'],$_SESSION['user_id'],$PARAMS['captcha']) ==
false){
        buran_die('Falsches CAPTCHA');
}
```

Damit wird die Ausführung des Scripts mit einer Fehlermeldung abgebrochen, wenn das CAPTCHA nicht korrekt eingegeben wurde.

Es ist noch anzumerken, dass die in diesem Beispiel verwendeten Sprachstrings, “Spamschutz” und “Falsches CAPTCHA”, nur zur Veranschaulichung direkt angegeben wurden – in tatsächlich verwendetem Code sollten an dieser Stelle die betreffenden $LANG-Variablen stehen.

Benutzer & Benutzergruppen

Daten eines Benutzerkontos abrufen:

buran_user_info(*$userid* **)**

$userid ist die ID des Benutzers, über den Informationen eingeholt werden soll.

Zurückgereicht wird ein Array mit folgenden Angaben:

'**user_id**' – die ID des Benuters.

'**user_ip**' – die letzte IP-Adresse des Benutzers.

'**user_name**' – der Name des Benutzers.

'**user_group**' – die Benutzergruppe.

'**user_flag**' – Flagge des Benutzers

'**user_posts**' – Posting-Zähler des Benutzers

'**user_regdate**' – Registrierungsdatum des Benutzers als UNIX-Timestamp

'**user_lastlogin**' – letztes Einloggen des Benutzers auf der Webseite als UNIX-Timestamp

'**email**' – E-Mail-Adresse des Benutzers

'**user_theme**' – Theme, mit dem der Benutzer die Webseite betrachtet

'**user_language**' – Sprache, mit der der Benutzer die Webseite betrachtet

'**buran_languagepack**' – Buran-Languagepack, das für diesen Benutzer standardmässig verwendet werden soll

'**php_locale**' – PHP-Locale, das für diesen Benutzer standardmässig verwendet werden soll

'**fckeditor_locale**' – FCKeditor-Locale, das für diesen Benutzer standardmässig verwendet werden soll

'**sig**' – Signatur des Benutzers

'**avatar**' – URL der Avatar-Grafik des Benutzers

'**title**' – Benutzertitel des Benutzers

'**realname**' – echter Name des Benutzers

Ausserdem beinhaltet der Array den Wert aller zusätzlichen Benutzerprofil-Felder für diesen Benutzer. (Benutzerprofile werden im nächsten Kapitel näher erklärt.)

Wenn nur einzelne Informationen eingeholt werden müssen, wie z.B. ein Benutzername, mag die Verwendung von buran_user_info() auf den ersten Blick übertrieben erschienen. Die Funktion ist jedoch eng mit dem Cache-System verbunden. Oftmals muss zum Einholen der Informationen im Array dadurch nicht einmal auf die Datenbank zugegriffen werden; dies macht die Funktion äusserst effizient.

Überprüfen, ob der aktuelle Benutzer registriert ist:

buran_user_isregistered()

Wenn der momentane Benutzer registriert ist, wird *true* zurückgegeben. Ist er ein Gast, wird *false* zurückgegeben.

Überprüfen, ob ein Benutzer über Administratorrechte verfügt:

buran_user_isadmin(*$userid=false* **)**

Wenn der Benutzer mit der ID **$userid** über Administratorrechte verfügt wird *true* zurückgegeben, andernfalls *false*.

$userid ist als Parameter freiwillig; wenn nichts angegeben wird, wird die Überprüfung für den aktuellen Benutzer durchgeführt.

Überprüfen, ob ein Benutzer verbannt ist:

buran_user_isbanned(*$userid=false* **)**

Wenn der Benutzer mit der ID **$userid** verbannt ist wird *true* zurückgegeben, andernfalls *false*.

$userid ist als Parameter freiwillig; wenn nichts angegeben wird, wird die Überprüfung für den aktuellen Benutzer durchgeführt.

Den Benutzer zum Einloggen zwingen, sofern er noch nicht eingeloggt ist:

buran_user_forcelogin(*$message='none'* **)**

Wenn der aktuelle Benutzer nicht eingeloggt ist, wird die Ausführung des Scriptes unterbrochen und ein Login-Formular angezeigt.

$message ist dabei der Text, der über dem Login-Formular angezeigt werden soll. Wenn $message nicht angegeben wird oder “none” lautet, wird das Formular ohne Text angezeigt.

Diese Funktion setzt voraus, dass das “userlogin”-Modul installiert ist. Sollte dem nicht so sein, wird dem Benutzer anstelle des Formulars lediglich der Text von $message als Fehlermeldung angezeigt.

Posting-Zähler eines Benutzers erhöhen:

buran_user_postincrease(*$userid* **)**

Erhöht den Posting-Zähler des Benutzers mit der ID ***$userid*** um 1.

Posting-Zähler eines Benutzers vermindern:

buran_user_postdecrease(*$userid* **)**

Senkt den Posting-Zähler des Benutzers mit der ID ***$userid*** um 1.

Benutzerkonto hinzufügen:

buran_user_add(*$insert_values* **)**

$insert_values ist ein Array mit Angaben zum neu zu erstellenden Benutzerkonto.

Mindestens folgende Werte müssen darin enthalten sein:

```
'username'          =>   Benutzername
'password'          =>   Passwort
```

Folgende Werte können vorhanden sein, müssen aber nicht:

```
'usergroup'         =>   ID der Benutzergruppe
'flag'              =>   Landesflagge (sofern aktiviert)
'posts'             =>   Anzahl Postings auf der Webseite
'standardtheme'     =>   Zu verwendendes Theme
'standardlanguage'  =>   Zu verwendende Sprache
'active'            =>   1 = Benutzerkonto aktiv
                         0 = Benutzerkonto inaktiv
'email'             =>   E-Mail-Adresse
'realname'          =>   Echter Name
'avatar'            =>   URL zur Avatar-Grafik
'title'             =>   Titel (z.B. “Administrator”)
```

```
'sig'          =>   Forum-Signatur
```

Ausserdem können die Werte zusätzlich erstellter Profilfelder, die sich oftmals von Webseite zu Webseite unterscheiden, ebenfalls in diesem Array angegeben werden. (Siehe Kapitel “Benutzerprofile”.)

Benutzerkonto bearbeiten:

buran_user_update(*$userid, $insert_values* **)**

$userid ist die ID des zu bearbeitenden Benutzerkontos.

$insert_values ist ein Array, der die zu ändernden Werte enthält. Folgende Werte können darin enthalten sein:

```
'username'          =>   Benutzername
'password'          =>   Passwort
'usergroup'         =>   ID der Benutzergruppe
'flag'              =>   Landesflagge (sofern aktiviert)
'posts'             =>   Anzahl Postings auf der Webseite
'standardtheme'     =>   Zu verwendendes Theme
'standardlanguage'  =>   Zu verwendende Sprache
'active'            =>   1 = Benutzerkonto aktiv
                         0 = Benutzerkonto inaktiv
'email'             =>   E-Mail-Adresse
'realname'          =>   Echter Name
'avatar'            =>   URL zur Avatar-Grafik
'title'             =>   Titel (z.B. “Administrator”)
'sig'               =>   Forum-Signatur
```

Ausserdem können die Werte zusätzlich erstellter Profilfelder, die sich oftmals von Webseite zu Webseite unterscheiden, ebenfalls in diesem Array angegeben werden. (Siehe Kapitel “Benutzerprofile”.)

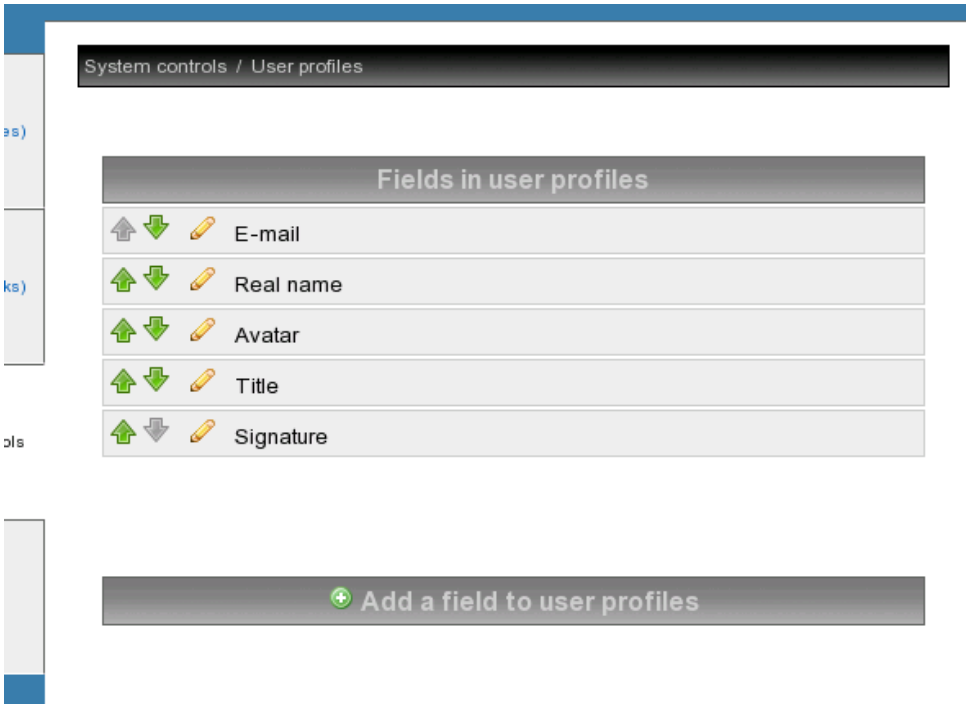
Benutzerkonto löschen:

buran_user_delete(*$userid* **)**

$userid = Die ID des zu löschenden Benutzerkontos.

Benutzerprofile

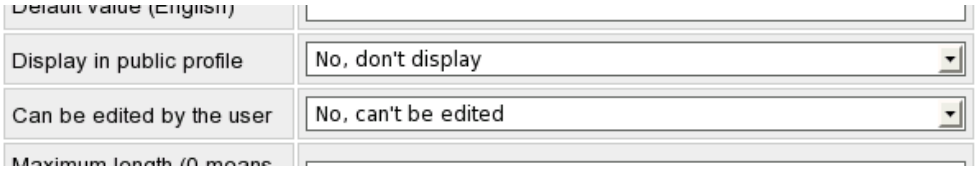
Was genau ein Benutzerprofil auf einer Buran-betriebenen Webseite beinhaltet, ist nicht in Stein gemeisselt. Wie bereits in der allgemeinen Übersicht zu Buran am Anfang dieser Dokumentation erklärt gibt es zwar gewisse Grundfelder, die immer vorhanden sind, doch im Gegensatz zu den meisten anderen CMS bietet Buran die notwendige Funktionalität, die Benutzerprofile um beliebig viele zusätzliche Felder frei zu erweitern.



Dabei gibt es einige Benutzerprofilfelder, die in jeder Buran-Installation vorhanden sind und zwar bearbeitet, nicht aber gelöscht werden können, weil sie für den stabilen Betrieb von Buran unentbehrlich sind.

```
email           =>      E-Mail-Adresse
realname        =>      Echter Name des Benutzers
avatar          =>      URL zum Avatar des Benutzers
title           =>      Titel des Benutzers (z.B. “Administrator”)
sig             =>      Forum-Signatur des Benutzers
```

Wenn diese Felder auf einer Webseite nicht gebraucht werden, können sie einfach deaktiviert werden. Sie werden dann nie angezeigt.



Die Daten der zusätzlichen Profilfelder fliessen automatisch in den $_SESSION-Array und den Output von buran_user_info() ein. Sie können ausserdem mit den üblichen Benutzerfunktionen für jeden Benutzer einzeln bearbeitet werden.

Permissions & Settings

Oft kommt es vor, dass in einem Modul Benutzerberechtigungen oder Einstellungen verwendet werden, die im Admin-Panel eingestellt werden können. Ein gutes Beispiel dafür ist das Galerie-Modul, wo für jedes Album festgelegt werden kann, welche Benutzergruppen Bilder kommentieren oder hochladen dürfen, und wo in den Einstellungen z.B. festgelegt wird, wieviele Bilder in einem Album pro Seite angezeigt werden sollen, etc.

Da derartige Funktionalität sehr häufig gebraucht wird ist sie bereits im Buran-Kernsystem integriert und kann in jedem Modul eingesetzt werden. Dies bietet mehrere Vorteile:

- Settings & Permissions werden in jedem Modul gleich behandelt, ohne dass der Autor jedes neuen Moduls einen eigenen neuen Lösungsansatz finden muss – das macht den Code verständlicher und spart Arbeit.
- Das Settings & Permissions-System, das von Buran bereitgestellt wird, ist direkt mit dem Cache-System integriert und ist dadurch sehr effizient – es werden nur sehr selten Datenbankzugriffe nötig. Das spart Ressourcen.

Grundsätzliches Arbeiten mit Settings

Ein Konfigurationswert oder “Setting” kann mit der Funktion buran_settings_newvalue() gespeichert werden:

> **buran_settings_newvalue(** *$module, $name, $value, $limit_to_language=false* **)**
>
> ***$module*** ist der Name des Moduls, zu dem das Setting gehört.
>
> ***$name*** ist der Name des Settings.
>
> ***$value*** ist der Wert des Settings.
>
> ***$limit_to_language*** ist ein freiwilliger Parameter, der bei Webseiten mit mehrsprachigem Inhalt wirksam ist. Lautet er true, so kann das Setting in

jeder Sprachversion der Webseite einzeln eingestellt werden und über verschiedene Werte verfügen. Lautet er false, so hat das Setting in allen Sprachvarianten der Webseite denselben Wert.

Wenn es bereits einen entsprechenden Konfigurationswert gibt, wird er mit dem neuen Wert überschrieben, ansonsten wird er neu eingerichtet.

Der so gespeicherte Konfigurationswert wird beim Zugriff auf das Modul automatisch in den $CONF-Array geladen und kann daraus abgerufen werden. Wenn z.B. im Gallery-Modul ein Wert folgendermassen festgelegt würde...

```
buran_settings_newvalue('gallery', 'testvalue', 'Dies ist ein Test', false);
```

...können wir im Code jeder Subpage des Gallery-Moduls den Wert jederzeit über

```
$CONF['testvalue']
```

abrufen. Wir können ihn auch in jedem Subpage-Template des Moduls wiedergeben:

```
{{CONF['testvalue']}}
```

(Dies würde auf der Webseite schlussendlich als “Dies ist ein Test” angezeigt.)

Die Settings eines Moduls müssen bereits während der Installation des Moduls eingerichtet werden. Dies geschieht, indem buran_settings_newvalue() in der buran_*modulname*_install()-Funktion verwendet wird – nach Abruf der buran_module_install()-Funktion, die das Modul im Kernsystem registriert.

Bei der Deinstallation eines Moduls über die Funktion buran_module_uninstall()

werden alle Settings des Moduls automatisch wieder gelöscht.

Wenn die $CONF-Daten eines Moduls neu geladen werden müssen, kann dies folgendermassen getan werden:

```
$CONF = buran_settings_reload($CONF);
```

Grundsätzliches Arbeiten mit Permissions

Das Arbeiten mit Permissions verläuft ähnlich wie das Arbeiten mit Settings. Bei einer Permission handelt es sich ebenfalls um eine Art Konfigurationswert. Der Unterschied zu den Settings liegt darin, dass bei den Permissions für jede Benutzergruppe ein anderer Wert möglich ist und dass die Permissions nicht von alleine geladen und zur Verfügung gestellt werden (wie das bei den Settings durch den $CONF-Array der Fall ist).

Eingerichtet wird eine Permission mit der Funktion buran_permissions_newvalue():

> **buran_permissions_newvalue(** *$module, $name, $value, $usergroup, $limit_to_language=false* **)**
>
> ***$module*** ist der Name des Moduls, zu dem die Permission gehört.
>
> ***$name*** ist der Name der Permission.
>
> ***$value*** ist der Wert der Permission (0 oder 1).
>
> ***$usergroup*** ist die ID der Benutzergruppe, auf die sich die Permission bezieht.
>
> ***$limit_to_language*** ist ein freiwilliger Parameter, der bei Webseiten mit mehrsprachigem Inhalt wirksam ist. Lautet er true, so kann die Permission in jeder Sprachversion der Webseite einzeln eingestellt werden und über verschiedene Werte verfügen. Lautet er false, so hat die Permission in allen Sprachvarianten der Webseite denselben Wert für diese Benutzergruppe.

Wie bei den Settings findet buran_permissions_newvalue() bei der Installation des Moduls und überall dort, wo eine Änderung an einer Permission möglich ist, Verwendung.

Ein Unterschied besteht aber darin, dass bei der Einrichtung von Permissions während der Installation eines Moduls der Wert für alle grundlegenden 5 Benutzergruppen separat gesetzt werden muss (also für Gäste, Administratoren, Globale Moderatoren, Registrierte Benutzer und verbannte Benutzer). Um während einer Modulinstallation eine Permission vollständig einzurichten, muss buran_permissions_newvalue() also fünfmal verwendet werden, wie z.B. auch in folgendem Beispiel aus dem “forum”-Modul:

```php
function buran_forum_install(){
	if(!buran_user_isadmin()){die('ERROR: Admin permissions required!');}

	$result = true;

	if(!buran_module_install('forum', 1)){$result = false;}

	// Set permissions for Guests
	if(!buran_permissions_newvalue('forum','subread1',1,1,true)){$result = false;}
	if(!buran_permissions_newvalue('forum','subwrite1',0,1,true)){$result = false;}
	if(!buran_permissions_newvalue('forum','submod1',0,1,true)){$result = false;}
	// Set permissions for Administrators
	if(!buran_permissions_newvalue('forum','subread1',1,2,true)){$result = false;}
	if(!buran_permissions_newvalue('forum','subwrite1',1,2,true)){$result = false;}
	if(!buran_permissions_newvalue('forum','submod1',1,2,true)){$result = false;}
	// Set permissions for Global Moderators
	if(!buran_permissions_newvalue('forum','subread1',1,3,true)){$result = false;}
	if(!buran_permissions_newvalue('forum','subwrite1',1,3,true)){$result = false;}
	if(!buran_permissions_newvalue('forum','submod1',1,3,true)){$result = false;}
	// Set permissions for Registered Users
	if(!buran_permissions_newvalue('forum','subread1',1,4,true)){$result = false;}
	if(!buran_permissions_newvalue('forum','subwrite1',1,4,true)){$result = false;}
	if(!buran_permissions_newvalue('forum','submod1',0,4,true)){$result = false;}
	// Set permissions for Completely Banned
	if(!buran_permissions_newvalue('forum','subread1',0,5,true)){$result = false;}
	if(!buran_permissions_newvalue('forum','subwrite1',0,5,true)){$result = false;}
	if(!buran_permissions_newvalue('forum','submod1',0,5,true)){$result = false;}
```

Bei der Deinstallation eines Moduls über die funktion buran_module_uninstall() werden alle Permissions des Moduls automatisch gelöscht.

Anders als bei den Settings, gibt es dennoch eine Möglichkeit, Permissions zu löschen, nämlich über die Funktion buran_permissions_delete().

> **buran_permissions_delete(** *$module, $name* **)**
>
> ***$module*** ist der Name des Moduls, zu dem die zu löschende Permission gehört.
>
> ***$name*** ist der Name der zu löschenden Permission.

Diese Funktion wird beispielsweise im Forum-Modul eingesetzt. Dort wird für jedes vorhandene Subboard ein eigener Satz Permissions angelegt. Wenn ein Subboard gelöscht wird, werden die so überflüssig gewordenen Permissions mittels buran_permissions_delete() gelöscht.

Abgerufen werden können Permissions durch folgende Funktionen:

> **buran_permissions_load(** *$module, $usergroup=false* **)**

$module ist der Name des Moduls.

$usergroup kann die ID einer bestimmten Benutzergruppe sein, für die die Permissions geladen werden sollen, oder false. Wenn $usergroup false ist, werden die Permissions einfach für die Benutzergruppe des momentanen Benutzers geladen.

Die Permissions werden als Array von der Funktion zurückgereicht.

buran_permissions_getallowed(*$module, $name* **)**

$module ist der Name des Moduls.

$name ist der Name einer einzelnen Permission.

Die Funktion reicht einen Array zurück, der die IDs aller Benutzergruppen enthält, bei denen die Permission “$name” positiv ist.

buran_permissions_getdisallowed(*$module, $name* **)**

$module ist der Name des Moduls.

$name ist der Name einer einzelnen Permission.

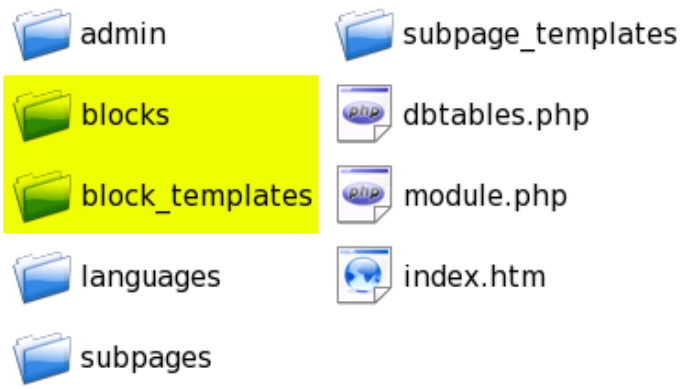
Die Funktion reicht einen Array zurück, der die IDs aller Benutzergruppen enthält, bei denen die Permission “$name” negativ ist.

Mit Blöcken arbeiten

Module können eigene Blöcke auf einer Seite erstellen, bearbeiten oder löschen.

Um eigene Blöcke zu erstellen, muss ein Modul dem Kernsystem sogenannte “Blocktypen” zur Verfügung stellen. Diese befinden sich in eigenen Unterverzeichnissen des Modulverzeichnisses.

Als Beispiel hier das Verzeichnis “modules/polls”:

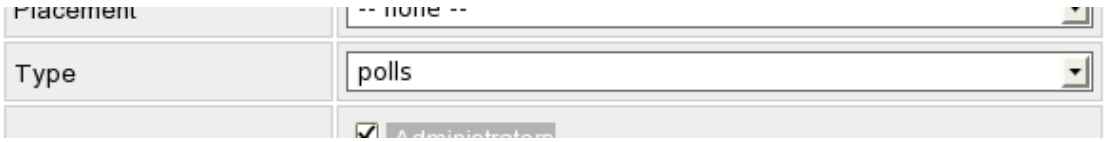


Die Aufteilung zwischen “blocks” und “block_templates” verhält sich ähnlich wie die Aufteilung zwischen “subpages” und “subpage_templates”: Im Verzeichnis “blocks” befinden sich die Scriptdateien, die im Blocks ausgeführt werden, und im Verzeichnis “block_templates” die Layoutdaten, die die Darstellung des Blockes kontrollieren. Auch hier kann es für jedes Theme einen eigenen Template-Satz geben, wobei der Template-Satz “default” immer vorhanden sein muss und verwendet wird, wenn für ein anderes Theme kein eigener Template-Satz gefunden wird.

Jedes Script im “blocks”-Verzeichnis bildet einen eigenen Blocktypen. Das Polls-Modul hat z.B. nur einen Blocktypen namens “polls”, deshalb befindet sich im “blocks”-Verzeichnis dieses Moduls nur eine Datei namens “polls.php” und im Template-Verzeichnis das dazugehörige Layout “polls.tpl.php”.

Alle Blocktypen eines Moduls werden bei der Modulinstallation automatisch ins

Kernsystem eingelesen und sind somit automatisch auch in der Auswahl beim Hinzufügen neuer Blöcke verfügbar.



Für jeden Blocktypen können im “blocks/”-Verzeichnis eines Moduls ausserdem noch folgende Dateien angelegt werden:

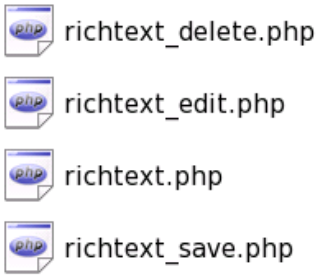
- *block*_edit.php
  Wird eingebunden, wenn ein Block dieses Types im Admin-Panel bearbeitet wird

- *block*_save.php
  Wird eingebunden, wenn ein Block dieses Types im Admin-Panel nach dem Bearbeiten gespeichert wird

- *block*_delete.php
  Wird eingebunden, wenn ein Block dieses Types im Admin-Panel gelöscht wird

block ist dabei natürlich mit dem Namen des betreffenden Block-Types zu ersetzen.

Beispiel aus dem “richtext”-Modul:



Über das zur Verfügung stellen von eigenen Blocktypen hinaus kann ein Modul die Blöcke einer Seite auch durch direktes Bearbeiten beeinflussen. Die dazu benötigten Funktionen werden vom Buran-Kernsystem zur Verfügung gestellt.

Einen Block hinzufügen

buran_block_add(*$insert_values* **)**

$insert_values ist ein Array, der die Daten des neuen Blocks enthält. Dabei müssen mindestens folgende Einträge vorhanden sein:

`'module'`	`=>`	Name des Moduls, dem der Block gehört
`'type'`	`=>`	Name des zu verwendenden Blocktyps

Folgende Einträge können ausserdem vorhanden sein, müssen aber nicht:

`'blockname'`	`=>`	Name des Blocks
`'area'`	`=>`	Block-Area, in welcher der Block platziert werden soll
`'usergroups'`	`=>`	Array mit der ID aller Benutzergruppen, die diesen Block sehen dürfen – oder der String “all” für Alle
`'pages'`	`=>`	Array mit der ID aller Seiten, auf denen dieser Block angezeigt werden soll – oder der String “all” für Alle

Die Funktion reicht die ID des neu erstellten Blocks zurück.

Einen Block verändern

buran_block_update(*$block_id, $update_values* **)**

$block_id ist die ID des Blocks, der bearbeitet werden soll.

$update_values ist ein Array, der die zu verändernden Daten des Blocks enthält. Dabei können folgende Werte angegeben werden:

```
'blockname'    =>   Name des Blocks
'area'         =>   Block-Area, in welcher der Block
                    platziert werden soll
'usergroups'   =>   Array mit der ID aller
                    Benutzergruppen, die diesen Block
                    sehen dürfen – oder der String “all”
                    für Alle
'pages'        =>   Array mit der ID aller Seiten, auf
                    denen dieser Block angezeigt werden
                    soll – oder der String “all”
                    für Alle
```

Die Funktion reicht bei Erfolg “true” zurück.

Einen Block löschen

buran_block_delete(*$id* **)**

$id ist die ID des Blocks, der gelöscht werden soll.

Einen Block in der Reihenfolge nach oben verschieben

buran_block_moveup(*$id* **)**

$id ist die ID des Blocks, der verschoben werden soll.

Einen Block in der Reihenfolge nach unten verschieben

buran_block_movedown(*$id* **)**

$id ist die ID des Blocks, der verschoben werden soll.

Alle Blöcke eines Moduls finden

buran_block_ofmodule(*$module* **)**

$module ist der Name des Moduls, dessen Blöcke gefunden werden sollen.

Zurückgereicht wird ein Array mit den IDs der Blöcke dieses Moduls.

Blöcke eines Moduls bei Deinstallation löschen

```
foreach(buran_block_ofmodule($modulname) as $block_id){
     buran_block_delete($block_id);
}
```

Modul mit Suchsystem integrieren

Buran verfügt über ein zentrales Suchsystem, das in jedes Modul integriert werden kann.

Dabei ist alles, was als theoretisch als Suchresultat angezeigt werden könnte, mit Titel, Beschreibung, durchsuchbarem Text und URL in einer Datenbanktabelle gespeichert, die dann von der Buran-Suchfunktion abgefragt werden kann.

Im Falle des News-Moduls bedeutet “jedes erdenkliche Suchresultat” z.B. jeden Artikel mitsamt seinen Kommentaren, im Falle des Galerie-Moduls jedes Album mitsamt seinen Kommentaren und Bildern, usw.

Um ein Modul mit dem Suchsystem zu verknüpfen, muss als erstes an jeder Stelle im Code des Moduls, an der ein mögliches Suchresultat hinzugefügt, gelöscht oder verändert wird, folgende Funktion gerufen werden:

> **buran_search_index(** *$module, $what_to_index, $id* **)**
>
> ***$module*** ist der Name des Moduls, dem das Suchresultat gehört.
>
> ***$what_to_index*** ist ein beliebiger String, der die Art des neuen, veränderten oder gelöschten Eintrags beschreibt. Im “news”-Modul könnte dies z.B. “article” sein, im “gallery”-Modul “album”, usw.
>
> ***$id*** ist die ID des neuen, veränderten oder gelöschten Eintrags.

Ausserdem muss in der module.php-Datei des betreffenden Moduls eine neue Funktion mit Namen buran_*modulname*_index() angelegt werden. Diese Funktion wird von buran_search_index() als Hilfsfunktion gerufen.

> **buran_*modulename*_index(** *$what_to_index, $id* **)**
>
> ***$what_to_index*** und ***$id*** sind dabei dieselben Werte, mit denen buran_search_index() im Code des Moduls gerufen wird – sie beschreiben also auch hier einen neuen, veränderten oder gelöschten Eintrag des Moduls.
>
> Diese Funktion hat folgende Aufgabe:

1. Die interne URL (beginnend mit “index.php?page=”, usw.) rekonstruieren, unter der der durch $what_to_index und $id beschriebene Eintrag im Front-End erreichbar ist
2. Überprüfen, ob der durch $what_to_index und $id beschriebene Eintrag existiert und der Öffentlichkeit zugänglich ist – d.h., ob er in den Suchresultaten vorkommen kann und soll
3. Falls ja, muss buran_search_add() gerufen werden.
4. Falls nein, muss buran_search_delete() gerufen werden.

Die Funktionen buran_search_add() und buran_search_delete() dienen dazu, den Suchindex auf dem aktuellen Stand zu halten, und nehmen folgende Parameter entgegen:

buran_search_add(*$module, $title, $searchtext, $url, $preview = '', $usergroups = array(), $limitlanguage = false* **)**

$module ist der Name des Moduls.

$title ist der Titel, unter dem der Eintrag in den Suchresultaten erscheint.

$searchtext ist der gesamte zu indexierende Text im HTML-Format. Im “news”-Modul wäre dies z.B. der Text eines Artikels sowie der Text all seiner Kommentare, im “gallery”-Modul wären es die Titel aller Bilder in einem Album sowie die Texte aller Kommentare des Albums, usw.

$url ist die interne URL, die im 1. Schritt von buran_*modulename*_index() rekonstruiert wird – siehe oben.

$preview ist der Vorschautext, mit dem der Eintrag in den Suchresultaten erscheint.

$usergroups ist ein Array mit den IDs aller Benutzergruppen, deren Mitglieder dieses Suchresultat sehen dürfen.

$limitlanguage gibt an, ob der Sucheintrag in allen Sprachvarianten der Webseite vorkommen soll (false) oder nur in der momentan verwendeten (true).

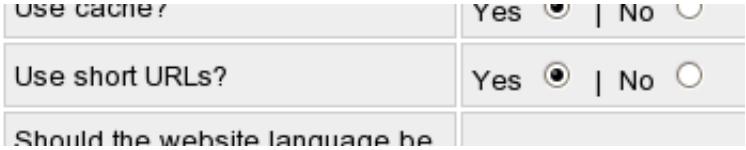
buran_search_delete(*$url* **)**

$url ist die interne URL, die im 1. Schritt von buran_*modulename*_index() rekonstruiert wird – siehe oben.

Bei der Deinstallation eines Moduls über die funktion buran_module_uninstall() werden alle möglichen Suchresultate des Moduls automatisch gelöscht.

Module mit Kurz-URLs

Unterstützung für sogenannte “short URLs”, also “Kurz-URLs”, ist ein Feature, das inzwischen von so vielen Web-Applikationen und Systemen unterstützt wird, dass man es eigentlich fast schon zur Standardfunktionalität eines CMS rechnen kann. Buran bildet hier keine Ausnahme; die Unterstützung für Kurz-URLs ist im Kernsystem integriert und kann in jedem Modul ohne grösseren Aufwand implementiert werden.



Wenn die Option “Kurz-URLs verwenden” bei den Einstellungen im Admin-Panel einer Buran-Installation aktiviert ist, wird in jeder internen URL der Name der Zielseite direkt angegeben. Aus

http://www.webseite.com/index.php?page=16&subpage=subboard&id=7

wird also z.B.:

http://www.webseite.com/forum/&subpage=subboard&id=7

Dies stellt bereits eine Verbesserung in der Suchmaschinenfreundlichkeit der URL dar. Wir stellen allerdings fest, dass die URL-Parameter in ihrer ursprünglichen Form verblieben sind. Aber keine Angst, dies muss nicht so sein!

Der Grund dafür ist, dass jedes Modul selbst für die Verwaltung seiner URL-Parameter im Kurz-URL-System verantwortlich ist, und das im obigen Beispiel verwendete Forum-Modul Kurz-URLs zum aktuellen Zeitpunkt noch nicht unterstützt. Täte es dies, könnte die URL aus dem Beispiel bei aktivierten Kurz-URLs vielleicht etwa so aussehen:

http://www.myburanwebsite.com/forum/test-subboard/irgendein-thread/

Ein Modul vollständig mit dem Kurz-URL-System zu integrieren, ist, wie schon erwähnt, nicht aufwändig.

Zunächst muss an jeder Stelle im Code des Moduls, an der Einträge hinzugefügt, verändert oder gelöscht werden, die im Front-End erreichbar sind (d.h., die über eine Kurz-URL verfügen können), folgende Funktion gerufen werden:

> **buran_shorturl_refresh(** *$module, $what_to_index, $id* **)**
>
> ***$module*** ist der Name des Moduls, dem die Kurz-URL gehört.
>
> ***$what_to_index*** ist ein beliebiger String, der die Art des neuen, veränderten oder gelöschten Eintrags beschreibt. Im “news”-Modul könnte dies z.B. “article” sein, im “gallery”-Modul “album”, usw.
>
> ***$id*** ist die ID des neuen, veränderten oder gelöschten Eintrags.

Ausserdem muss in der module.php-Datei des betreffenden Moduls eine neue Funktion mit Namen buran_*modulname*_shorturl() angelegt werden. Diese Funktion wird von buran_shorturl_refresh() als Hilfsfunktion gerufen.

> **buran_*modulename*_shorturl(** *$what_to_index, $id* **)**
>
> ***$what_to_index*** und ***$id*** sind dabei dieselben Werte, mit denen buran_shorturl_refresh() im Code des Moduls gerufen wird – sie beschreiben also auch hier einen neuen, veränderten oder gelöschten Eintrag des Moduls.
>
> Diese Funktion hat folgende Aufgabe:
>
> 1. Überprüfen, ob der durch $what_to_index und $id beschriebene Eintrag existiert
> 2. Falls ja, muss für jede URL, die auf diesen Eintrag verweist, buran_shorturl_add() gerufen werden.
> 3. Falls nein, muss für jede URL, die auf diesen Eintrag verweist,

buran_shorturl_delete() gerufen werden.

Die Formulierung “jede URL, die auf diesen Eintrag verweist” bezieht sich auf die Tatsache, dass durch einen Eintrag mehrere URLs entstehen können. Ein Beispiel – durch nur einen News-Artikel entstehen 6 neue URLs:

- URL zum Artikel
- URL zum “Neuen Kommentar hinzufügen”-Formular
- URL der “Neuen Kommentar speichern”-Seite
- URL zum “Kommentar bearbeiten”-Formular
- URL der “Bearbeiteten Kommentar speichern”-Seite
- URL zum “Kommentar löschen”-Bestätigungsformular
- URL der “Kommentar löschen”-Seite

Die buran_shorturl_add() und buran_shorturl_delete()-Funktionen dienen dem Zweck, die Kurz-URL-Liste des Buran-Kernsystems immer auf dem aktuellen Stand zu halten. Sie nehmen folgende Parameter entgegen:

buran_shorturl_add(*$module, $short_params, $long_params, $optional_params=''* **)**

$module ist der Name des Moduls.

$short_params enthält die suchmaschinenfreundliche Kurz-URL-Version der URL-Parameter in $long_params. Im Beispiel aus der Einleitung dieses Kapitels wäre dies:

```
test-subboard/irgendein-thread/
```

$long_params enthält die URL-Parameter der internen URL, die in die Kurz-URL-Liste des Kernsystems eingetragen werden soll (ohne den “page”-Parameter). Im Beispiel aus der Einleitung dieses Kapitels wäre dies:

```
&subpage=subboard&id=7
```

Die Reihenfolge der Parameter im String ist dabei egal.

$optional_params enthält die nicht-obligatorischen URL-Parameter der

internen URL – Parameter, die manchmal vorhanden sind und manchmal auch nicht (z.B. “skip”-Parameter).

```
skip
```

buran_shorturl_delete(*$module, $long_params* **)**

$module ist der Name des Moduls.

$long_params enthält die URL-Parameter der internen URL, die in die Kurz-URL-Liste des Kernsystems eingetragen werden soll (ohne den “page”-Parameter). Im Beispiel aus der Einleitung dieses Kapitels wäre dies:

```
&subpage=subboard&id=7
```

Die Reihenfolge der Parameter im String ist dabei egal.

Bei der Erstellung der Kurz-URL-Parameters kann die Funktion buran_shorturl_propersyntax() nützlich sein; sie stellt sicher, dass jeder ihr weitergereichten String in seiner Formatierung absolut URL-gerecht ist; sie verwandelt also z.B. “Dies ist ein Artikel” in “dies-ist-ein-artikel”.

buran_shorturl_propersyntax(*$string* **)**

Die Funktion reicht eine URL-freundliche Version von ***$string*** zurück.

Ein kurzes Code-Beispiel aus dem “news”-Modul:

```
$article_title = buran_shorturl_propersyntax($article_r1['title']);
$category_title = buran_shorturl_propersyntax($category_r1['title']);

$long_params = 'subpage=read&id='.$id;
$short_params = $category_title.'/'.$article_title;
buran_shorturl_add('news', $short_params, $long_params);

$long_params = 'subpage=comment_add&id='.$id;
$short_params = $category_title.'/'.$article_title.'/comment-add';
buran_shorturl_add('news', $short_params, $long_params);

$long_params = 'subpage=comment_add2&id='.$id;
$short_params = $category_title.'/'.$article_title.'/comment-add2';
buran_shorturl_add('news', $short_params, $long_params);

$long_params = 'subpage=comment_edit&id='.$id;
$short_params = $category_title.'/'.$article_title.'/comment-edit';
buran_shorturl_add('news', $short_params, $long_params);

$long_params = 'subpage=comment_edit2&id='.$id;
$short_params = $category_title.'/'.$article_title.'/comment-edit2';
buran_shorturl_add('news', $short_params, $long_params);

$long_params = 'subpage=comment_delete&id='.$id;
$short_params = $category_title.'/'.$article_title.'/comment-delete';
```

RSS-Feed

Um ein Modul dem RSS-Feed hinzuzufügen, muss in der buran_*modulname*_install()-Funktion die Funktion buran_rss_connectModule() gerufen werden.

> **buran_rss_connectModule(** *$module* **)**
>
> ***$module*** ist der Name des Moduls, das dem RSS-Feed hinzugefügt werden soll.

Ausserdem muss an jeder Stelle im Code des Moduls, an der Inhalt, der im RSS-Feed erscheinen soll, hinzugefügt oder bearbeitet wird, die Funktion buran_rss_add() gerufen werden:

> **buran_rss_add(** *$title, $link, $description, $module, $id* **)**
>
> ***$title*** ist der Titel des RSS-Eintrags.
>
> ***$link*** ist der Link des RSS-Eintrags.
>
> ***$description*** ist der Inhalt des RSS-Eintrags.
>
> ***$module*** ist der Name des Moduls.
>
> ***$id*** ist die ID dieses RSS-Eintrags. Sie kann frei festgelegt werden. Wichtig ist, dass in keinem Modul mehrere RSS-Einträge mit derselben ID vorkommen.
> Diese ID wird in erster Linie bei der Löschung von Einträgen durch die Funktion buran_rss_remove() benötigt (siehe unten).

Und an jeder Stelle, an der Inhalt, der im RSS-Feed vorkommt, gelöscht wird, muss buran_rss_remove() gerufen werden:

> **buran_rss_remove(** *$module, $id* **)**
>
> ***$module*** ist der Name des Moduls.
>
> ***$id*** ist die ID, die für diesen Eintrag beim Rufen von buran_rss_add() angegeben wurde.

Events

Module haben die Möglichkeit, auf gewisse Ereignisse im Buran-Kernsystem jederzeit zu reagieren, selbst wenn gerade nicht direkt auf sie zugegriffen geladen sind.

Diese Ereignisse sind:

Name	Beschreibung
newActiveUser	Ein neues, aktives Benutzerkonto wird erstellt
newInactiveUser	Ein neues, inaktives Benutzerkonto wird erstellt
pageLoads	Eine Seite wird geladen
userBecomesInactive	Ein aktives Benutzerkonto wird auf inaktiv geschaltet
userBecomesActive	Ein inaktives Benutzerkonto wird auf aktiv geschaltet
userDeleted	Ein Benutzerkonto wird gelöscht
userLoggedIn	Ein Benutzer hat sich eingeloggt
userLoggedOut	Ein Benutzer hat sich ausgeloggt
userMovedToOtherGroup	Ein Benutzer wurde in eine andere Benutzergruppe verschoben
usergroupDeleted	Eine Benutzergruppe wurde gelöscht
usergroupNew	Eine neue Benutzergruppe wurde hinzugefügt

Damit ein Modul auf ein solches Event reagieren kann, muss in der buran_*modulname*_install()-Funktion des Moduls die Funktion buran_event_connectModule() gerufen werden.

> **buran_event_connectModule(** *$module, $event* **)**
>
> ***$module*** ist der Name des Moduls.
>
> ***$event*** ist der Name des Events, auf welches das Modul reagieren können soll.

Ausserdem muss in der module.php eine eigene Funktion namens buran_*modulname_eventname*() erstellt werden, wobei *eventname* mit dem Namen des Events zu ersetzen ist. Diese Funktion wird ausgeführt, wenn das Event statt findet. Als Parameter nimmt sie die ID des Benutzerkontos/der Benutzergruppe/der Seite/usw. entgegen, auf das/die sich das Event bezieht.

Ein Beispiel aus dem “billing”-Modul:

```php
function buran_billing_userDeleted($userid){

        if(!function_exists('billing_account_remove')){
                include_once(base_path.'modules/billing/functions.php');
        }

        if(billing_user_check($userid) == false){
                billing_account_remove($userid);
        }

        return true;

}
```

Ausserdem hat jedes Modul die Möglichkeit, selbst Events auszulösen. Dazu muss die Funktion buran_event_run() gerufen werden.

buran_event_run(*$event, $id=false* **)**

$event ist der Name des Events, das ausgelöst werden soll. Dies kann sowohl ein bereits existierendes Event sein (siehe Tabelle oben), als auch ein ganz neues.

$id ist die ID, auf die sich das Event bezieht. Wenn das Event keine ID mitliefert, kann der Parameter aber auch weggelassen werden und lautet dann einfach false.

Static caching

Auf gewissen Subpages mancher Module ist der Inhalt kaum oder gar nicht interaktiv und verändert sich tendenziell eher selten. So ist dies z.B. im “richtext”-Modul der Fall.

In solchen Modulen kann es sinnvoll sein, von der “static caching”-Funktionalität des Buran-Kernsystems Gebrauch zu machen. “Static caching” bedeutet, dass der Output einer Subpage beim Aufruf gespeichert wird, und in Zukunft direkt aus dem Cache geladen wird. Sämtliche Datenbankzugriffe, Scriptabläufe, usw. fallen beim zukünftigen Aufrufen der Subpage also weg, wodurch Ressourcen geschont werden und die Scalability der Webseite verbessert wird.

Um eine Subpage statisch zu Cachen, muss an erster Stelle in ihrem Code folgende Funktion gerufen werden:

```
buran_cache_static();
```

Die Cache-Version dieser Subpage ist entweder solange aktiv bis der Cache der Webseite geleert wird, oder bis sie anderswo im Code des Moduls manuell gelöscht wird.

Das bedeutet: An jeder Stelle im Code an der Inhalt verändert oder gelöscht wird, der im Front-End statisch gecachet werden kann, müssen allfällig vorhandene und nun veraltete gecachete Versionen des Inhalts manuell gelöscht werden. Dies geschieht mittels der Funktion buran_cache_clear().

buran_cache_clear(*$restrict_to = array()* **)**

$restrict_to ist ein Array, der genau festlegt, welche Cache-Dateien gelöscht werden sollen. Wenn er nicht angegeben wird oder leer ist, wird der gesamte Cache der Webseite gelöscht.
Der Array kann folgende Einträge enthalten:

```
'module'       =>   Nur Cache-Dateien dieses
                    Moduls werden gelöscht.
```

'subpage'	=>	Nur Cache-Dateien einer Subpage mit diesem Namen werden gelöscht.
'usergroup'	=>	Nur Cache-Dateien für Benutzer dieser Gruppe werden gelöscht.
'userid'	=>	Nur Cache-Dateien für den Benutzer mit dieser ID werden gelöscht.

Diverse nützliche Funktionen

buran_emoticons( )

Reicht einen Array mit allen im System vorhandenen Emoticons zurück. Der Array ist folgendermassen aufgebaut:

```
$emoticons = Array(
      0 => Array(
            'original' => ':)',
            'replacement' => 'smile.gif'
      )
      1 => Array(
            'original' => ':D',
            'replacement' => 'biggrin.gif'
      )
      // usw...
)
```

buran_flags( )

Reicht einen Array mit allen im System vorhandenen Flaggen zurück. Der Array ist folgendermassen aufgebaut:

```
$flags = Array(
      0 => Array(
            'title' => 'Blank',
            'filename' => 'blank.gif'
      )
      1 => Array(
            'title' => 'Switzerland',
            'filename' => 'ch.gif'
      )
      // usw...
)
```

buran_floodprotect(*$message* **)**

Falls seit dem letzten Posting des momentanen Benutzers noch nicht genug Zeit verflossen ist, um ein neues anzulegen (Zeitspanne kann im Admin-Panel eingestellt werden), wird die Ausführung des Scripts unterbrochen und ***$message*** als Fehlermeldung angezeigt.

buran_die(*$message*='' **)**

Unterbricht die Ausführung des Scripts und zeigt *$message* als Fehlermeldung an. Der Parameter kann auch weggelassen werden und ist dann einfach leer.

Sollte im Code anstelle der PHP-Standardfunktion die() und deren Alias exit() verwendet werden.

buran_pagination(*$no_of_items, $items_per_page, $currently_skipped* **)**

Diese Funktion hilft beim Erstellen einer Pagination (Seitenzahlennavigation), die z.B. verwendet wird, wenn aus einer grossen Auswahl von Datenbankeinträgen immer nur eine bestimmte Anzahl gleichzeitig angezeigt werden soll (Seite 1 enthält die Einträge 0-10, Seite 2 die Einträge 11-20, Seite 3 die Einträge 21-30, usw).

$no_of_items ist die gesamte Anzahl an Einträgen.

$items_per_page ist die Anzahl an Einträgen, die pro “Seite” angezeigt werden soll.

$currently_skipped ist die Anzahl an “übersprungenen” Einträgen, also die Zahl, bei der die momentan angezeigte Auswahl an Einträgen beginnt (wenn wir uns auf Seite 3 befinden wäre dies z.B. 20).

Von der Funktion zurückgereicht wird ein Array, der folgendermassen aufgebaut ist:

```
$pagination = array(
      [0] = array(
            'number' => 1,    // Die Zahl der Seite
```

```
            'skip' => 0,      // Zu überspringende Einträge
        );
        [1] = array(
            'number' => 1,    // Die Zahl der Seite
            'skip' => 10,     // Zu überspringende Einträge
        );
        [2] = array(
            'number' => 1,    // Die Zahl der Seite
            'skip' => 20,     // Zu überspringende Einträge
        );
        [3] = array(
            'number' => 1,    // Die Zahl der Seite
            'skip' => 30,     // Zu überspringende Einträge
        );
);
```

Dieser Array kann dann z.B. folgendermassen in einem Template zu einer Pagination verarbeitet werden:

```
<select name="skip">
<?php foreach ($pagination as $page) {
      <option value="{{page['skip']}}">{{page['number']}}</option>
<?php } ?>
</select>
```

buran_reversepagination(*$no_of_items, $items_per_page, $currently_skipped* **)**

Diese Funktion verhält sich genau gleich wie buran_pagination(), wird aber bei Paginations verwendet, die rückwärts verlaufen sollen (also nicht bei 1, sondern bei der höchsten Seitenzahl beginnend).

buran_loadlanguage(*$module* **)**

Lädt per include() das Sprachpaket des Moduls ***$module***.
Wenn die Sprache des Benutzers nicht gefunden werden kann, wird die Standardsprache der Webseite geladen. Wenn diese ebenfalls nicht vorhanden ist, wird das englische Sprachpaket geladen, das in jedem Modul vorhanden sein muss.

buran_jslang(*$LANG* **)**

Diese Funktion reicht einen String mit HTML-Code zurück, in dem eine Javascript-Funktion namens buranLangData() definiert wird. Diese Javascript-Funktion reicht den Inhalt des ***$LANG***-Arrays als Javascript-Array zurück.

So können die Sprachdaten im $LANG-Array nicht nur im PHP-Code, sondern auch im Javascript-Code des Moduls verwendet werden.

buran_notfound(*$message* = '' **)**

Diese Funktion sollte gerufen werden, wenn Inhalt nicht gefunden werden kann. Die Ausführung des Scripts wird damit unterbrochen und ***$message*** wird als Fehlermeldung angezeigt.

Der Parameter kann auch weggelassen werden und ist dann einfach leer.

buran_admin_linkselector(*$LANG, $name, $target=0* **)**

Diese Funktion steht nur im Code der Admin-Kontrollen eines Moduls zur Verfügung und setzt an der Stelle, an der sie gerufen wird, die Buran-eigene Auswahlkomponente für interne Links ein.

$LANG ist der $LANG-Array mit den Sprachdaten des Moduls.

$name ist der Name, unter dem der durch diese Komponente ausgewählte Wert nach Absenden eines Formulars im $PARAMS-Array erscheinen soll.

$target ist die ID der Seite im Buran-Seitenbaum, die in der Auswahlkomponente bereits ausgewählt sein soll. 0 bedeutet “keine”. Dieser Parameter kann auch weggelassen werden und hat dann standardmässig den Wert 0.

buran_forceTitle(*$title* **)**

Mit dieser Funktion kann manuell angegeben werden, was beim Betrachten der Seite im Browser als Fenstertitel angezeigt werden soll.

$title ist der dabei anzuzeigende String.

Funktioniert nur im Front-End!