CBACK Forum v4

Handbuch für Entwickler

Stand: 03 / 2020
Autor: Christian Knerr
Web: https://www.cback.de

Inhaltsübersicht

Generelle Informationen

Herzlich Willkommen im CF4 Entwicklerhandbuch!

Wir begrüßen Sie ganz herzlich im Entwicklerhandbuch des CBACK Forums Version 4 (kurz **CF4** genannt), und freuen uns, dass Sie Interesse haben unser Softwareprodukt mit eigenen Funktionen, Designs oder Sprachen zu erweitern. Dieses Handbuch soll Ihnen beim Einstieg in die Welt der CBACK API helfen: Sie erfahren nicht nur grundlegendes über die Entwicklung eigener Plugins, sondern finden in diesem Handbuch auch hilfreiche Informationen für Template-Designer oder Sprachdatei-Übersetzer.

Zusätzlich wurden in unserer Software auch alle Funktionen, Variablen und Klassen mit PHPDoc ausgestattet. Informationen zu Funktionsparametern sowie definierten Umgebungsvariablen erfahren Sie also immer zusätzlich im Quelltext der Software selbst.

Wir sind gespannt, welche zusätzlichen Funktionen Sie mit unserer vielseitigen Software umsetzen und wünschen Ihnen viel Spaß bei der Entwicklung!

Nützliche Links und Ressourcen

Das CF4 basiert auf der Scriptsprache PHP und nutzt MySQL als Datenbanksystem. Zusätzlich nutzt es einige Drittanbietermodule, für die es gesonderte und sehr ausführliche Dokumentationen gibt. Nachfolgend finden Sie hilfreiche Links zu diesen Dokumentationen.

- **PHP**
  Unter https://secure.php.net/ finden Sie die offizielle Webseite der Programmiersprache PHP, welche die Basis unseres Systems bildet. In der dortigen Dokumentation finden Sie alle PHP eigenen Funktionen genau und mit vielen Beispielen erklärt.

- **Smarty**
  Die Grundlage für unser Templatesystem bildet Smarty, welches durch unsere $UI Klasse erweitert wird. Smarty besitzt viele praktische Funktionen, die direkt in Templatedateien nutzbar sind. Die Dokumentation zu dieser Template Engine finden Sie unter: http://www.smarty.net/

- **LESS**
  Um die Möglichkeit zu bieten, Dinge wie Farbwerte in Variablen zu verwalten, und einen effektiveren und kürzeren CSS Code zu schreiben und Template-Anpassungen deutlich einfacher zu machen verwenden wir im CF4 LESS als Basis für unseren generierten Style-Code. Diese „Kurzsprache" für CSS benötigt vor Nutzung eine kurze Kompilierung bzw. Interpretierung der LESS Dateien und wandelt diese in eine CSS Datei um. Wenn Sie mit LESS Sprachkonstrukten nicht arbeiten möchten, dann können Sie auch weiterhin regulären CSS Code in den LESS Dateien verwenden. Hilfreiche Tipps zu LESS Funktionen, zur Nutzung der vorgefertigten Variablen sowie einige einfache LESS Compiler finden Sie unter: http://lesscss.org/ - Hier sollte insbesondere die Liste von GUI Basierten Compilerprogrammen für Einsteiger sehr nützlich sein: http://lesscss.org/usage/ - guis-for-less. Viele IDEs (z.B. phpStorm) ermöglichen es auch, einen LESS Compiler als FileWatcher zu definieren. So wird die benötigte CSS Datei bei Änderungen an einer .less Datei automatisch beim Speichern erzeugt und Ihre Arbeit wird nicht durch einen zusätzlichen Zwischenschritt ausgebremst.

- **UIKit**
  Unser Template besitzt als Basis im Frontend das Themekit „UIKit", welches viele nützliche Elemente wie Buttons, Responsive-Verhalten und Hilfsklassen bereits direkt mitliefert. Als Template-Designer kann so mittels UIKit in der Basis sehr viel im CF4 Design umgesetzt werden ohne sich z.B. mit wechselnden Bildschirmgrößen oder Browser-Kompatibilitäten Gedanken machen zu müssen. Alle Informationen zu UIKit Konstrukten im Template finden Sie unter: https://getuikit.com/v2/docs/core.html. Das CF4 liefert außerdem Systemweit die UIKit Zusatzkomponenten „Grid", „Lightbox", „Autocomplete", „Datepicker", „Slideset", „Parallax", „Accordion", „Notify", „Sticky", „Progressbar" sowie „Tooltip" mit, die viele häufig benötigte Dinge im Bereich der Anzeige vereinfachen. So ist z.B. für das Generieren eines Parallax Effektes oder einer Lightbox kein extra-Plugin notwendig.

- **jQuery**
  Auf JavaScript / AJAX Seite liefert das CF4 die sehr populäre JS-Library jQuery mit. Sie können jQuery Funktionen überall im System verwenden. Hilfe zu jQuery Befehlen finden Sie unter: http://jquery.com/.

Coding Guidelines / Wichtige Grundregeln

Wir möchten, dass das Entwickeln im CF4 viel Spaß macht und verzichten daher auf eine zu strenge Regulierung der Programmierweise. Dennoch müssen aus Gründen von System-Performance, Kompatibilität und Sicherheit einige Grundregeln beim Entwickeln von Plugins oder Templates befolgt werden. Ansonsten wird eine Erweiterung nicht zuverlässig im CBACK Forum funktionieren, das System instabil oder unsicher machen oder gar bei Aktualisierungen der Software zu Problemen führen. Nachfolgend finden Sie also einige wichtige Regeln, die bei

der Entwicklung von Plugins, Templates oder Sprachdateien immer befolgt werden müssen:

- **Der Begriff „cback“ ist reserviert!**
  Verwenden Sie bitte weder in Variablen, Klassen, eigenen Hookpoints, Datei- oder Ordnernamen oder Plugin-Titeln den Begriff „cback“. Dieser ist für uns selbst reserviert und kennzeichnet nicht nur offizielle Plugins, sondern wird auch für offizielle Hookpoints genutzt. Eine Kollision mit eventuellen anderen Plugins ist damit ausgeschlossen, da wir diesen Namensraum stets komplett selbst „überwachen“ können.

- **Nutzen Sie Ihren Autornamen als Pluginordner- sowie Templatedatei-Prefix!**
  Während unser Cache-System zwar jedes installierte Template in getrennten Ordnern verwaltet, werden zwischengespeicherte Template-Dateien für eine höhere Systemperformance nicht erneut aufgetrennt. Um die Kollision von Templatedateien bei mehreren Styles zu vermeiden ist es daher wichtig, dass Sie für all Ihre Plugin-Template-Dateien Ihren Autornamen oder ein Kürzel davon als Prefix für den Dateinamen verwenden. Nennen Sie die Templatedatei Ihres Plugins also nicht beispielsweise „index.htm“ sondern beispielsweise „cback_index.htm“. Um auch Kollisionen mit mehreren Ihrer eigenen Plugins zu vermeiden sollte ggf. auch der Name des Plugins selbst mit in den Dateinamen aufgenommen werden, beispielsweise „cback_portal_index.htm“ oder „meinautorname_tollesplugin_topicliste.htm“
  Das selbe gilt für die Namen von Plugin-Ordnern selbst: Wenn es beispielsweise mehrere Plugins mit dem Ordnernamen „like_plugin“ gibt, dann würden diese sich nicht parallel installieren lassen. Nennen Sie Ihren Pluginordner also beispielsweise „autorname_facebook_like“, um Kollisionen mit gleichnamigen Plugins zu vermeiden.

- **Nutzen Sie bitte immer unsere eigenen API Funktionen statt ihre Eigenen!**
  Das CF4 und die CBACK API stellen viele eigene Systemfunktionen bereit, beispielsweise das Sitzungssystem, die Benutzerverwaltung, die Verbindung zur Datenbank, das Template-System, das SEO System, Verwalten und Absichern von Werteübergaben, etc. – Diese Funktionen befinden sich in unseren Basisklassen, beispielsweise „$Core“, „$DB“, „$User“, „$UI“, „$SEO“, etc. Bitte nutzen Sie für Ihre eigenen Plugins die bereitgestellten Funktionen unserer API, um die Sicherheit, Performance und Kompatibilität des Systems zu gewährleisten. Nutzen Sie also zum Beispiel nicht eine

eigene Methode, um eine Verbindung zur Datenbank aufzubauen, sondern nutzen Sie dafür immer die Befehle von $DB->... Oder verwenden Sie kein „echo“ Befehl zur Ausgabe an den Browser oder ein eigenes Template-System, sondern lassen Sie den Output immer über unsere $UI-> Komponente laufen.

- **Nutzen Sie keine „global“ Befehle!**
  Um zu vermeiden, dass Variablen an anderer Stelle überschrieben werden und zur Verbesserung der Sicherheit sollten Sie die Nutzung des „global“ Befehls vermeiden.
  Innerhalb Ihrer eigenen Klasse können Sie über unseren Controller:: an Standardobjekte wie $DB, $Core, etc. gelangen. Nutzen Sie hierzu beispielsweise den Befehl: $Core = Controller::getClass('Core'); Näheres zum Controller erfahren Sie später in diesem Handbuch. Für eigene Funktionsvariablen sollten lokale Variablen per Übergabevariable in Ihre eigene Funktion übertragen werden anstatt dies über den global $meineVar Befehl zu lösen.

- **Nutzen Sie für den $lang Alias immer einen Pointer, um Speicher zu sparen!**
  Wenn Sie in einer Funktion Werte aus den Sprachdateien benötigen, können Sie eine lokale Variable $lang als Alias innerhalb Ihrer Funktion definieren, damit Sie nicht immer Controller::$lang['...'] schreiben müssen. Nutzen Sie hierzu aber bitte die Pointer-Zuweisung (mit &$), um das Duplizieren des Arrays zu vermeiden. So wird PHP Speicher gespart. Die Referenzierung eines $lang Alias sieht innerhalb einer (Klassen)funktion also so aus: $lang = &Controller::$lang; - Wenn Sie eine eigene Sprachdatei nachladen müssen, verwenden Sie bitte die Funktionen $User->reload_lang_file('...') für enthaltene Sprachdateien bzw. $User->reload_plugin_lang('...') für eine Sprachdatei aus Ihrem Plugin (dazu später mehr). Diese Funktionen verwalten nicht nur automatisch das globale Language-Array im Controller sondern weisen auch alle aktuellen Sprachvariablen an das Template zu.

Abschließend noch einige kurze Zusatzregeln:

- Beachten Sie stets die Lizenzbestimmungen unserer Software
- Liefern Sie keine Original-Dateien des CF4 mit Ihren Plugins mit
- Sie dürfen keinen Code bzw. Funktionen des CF4 auf andere Systeme portieren oder vorhandenen Quellcode des CF4 duplizieren
- Sie dürfen auch ein von Ihnen angepasstes Forum nicht weiterverkaufen oder als „gemoddet“ bzw. „geforkt“ weitervertreiben.

- Alle Dateien müssen im Zeichensatz UTF-8 (ohne BOM) erstellt und bearbeitet werden.
- Ihre Plugins dürfen nie vorhandene Dateien des Forensystems editieren, sondern müssen ausschließlich unser Hooksystem nutzen. Nur so ist gewährleistet, dass zusätzliche Templates oder das Aktualisieren des CBACK Forums keine Probleme verursachen.
- Nutzen Sie bitte Template- und Sprachdateien für Ihre eigenen Module: So ist es möglich, PHP und HTML Code eindeutig zu trennen sowie ggf. Übersetzungen für Plugins anzubieten.
- Wenn Sie selbst mit Ihrem Plugin zusätzliche DB Felder anlegen, beachten Sie bitte auch diese in der Deinstallationsroutine wieder zu entfernen, sodass keine Datenreste im System verbleiben, falls ein Plugin deinstalliert wird.
- Sie können innerhalb Ihrer eigenen Plugins sogar weitere Hooks mit unserer Hookschnittstelle hinzufügen. Bitte nutzen Sie jedoch für das Funktionsfeld „sid“ immer Ihren eigenen Autornamen und niemals den Begriff „cback“. Die maximallänge der SID ist 40 Zeichen. Es ist beschränkt auf Buchstaben und Zahlen.
- Während der Entwicklung sollten Sie die Konstante DEBUG in der Datei includes/config.php auf „true“ setzen. So werden Ihnen alle Fehlermeldungen gezeigt und es ist gewährleistet, dass Ihr Plugin sauber funktioniert.
- Sofern Sie den Ordner „setup/“ innerhalb einer Entwicklungsumgebung behalten möchten und die Hinweismeldung, dass der Ordner noch vorhanden ist unterdrücken möchten, dann fügen Sie einfach die Codezeile *define('NO_SETUP_CHECK', true);* in der includes/config.php hinzu, sodass diese Meldung unterdrückt wird. Dies sollte jedoch nicht auf Produktivumgebungen oder öffentlichen Installationen ausgeführt werden!
- Denken Sie bitte immer daran, Variablen in Ihrem Plugin zu initialisieren, um PHP-Notices zu vermeiden!
- Sie dürfen das CF4 nicht selbst weiterverkaufen. Es ist jedoch erlaubt Ihre eigenen Plugins bzw. Templates zu verkaufen, sofern diese keine Originaldateien des CBACK Forums mitliefern.
- Bitte programmieren Sie nach gültigem PHP7 Kompatiblen Standard – abwärtskompatibel zu PHP5.6+, um die Systemanforderungen und Kompatibilitäten des CF4 einzuhalten.
- Vermeiden Sie Laufcode und schreiben Sie die Funktionen für Ihr Plugin am Besten in eigene Klassen.
- Zum Verschachteln des Codes nutzen Sie bitte den Tabulator und nicht mehrere Leerzeichen. Klammersetzung darf nach Ihren Vorlieben erfolgen.
- Überschreiben Sie keine CF4 Systemklassen!

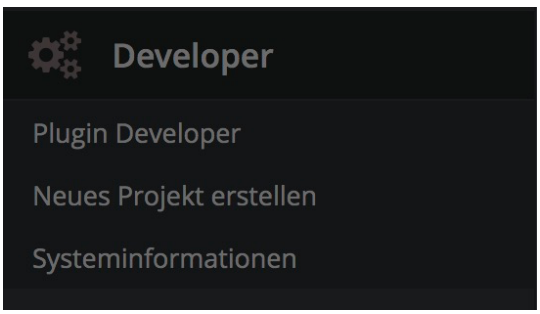
- Nutzen Sie bitte möglichst keine Inline-Styles sondern greifen Sie auf CSS Code zurück. Für die Farbe des Benutzernamens beispielsweise kann die CSS-Hilfsklasse .cf-gcolor-{farbcode aus der User oder GroupDB} verwendet werden. So ist es möglich systemweit die Darstellung diverser Elemente global per CSS zu beeinflussen, was Template-Designern deutlich mehr Anpassungsmöglichkeiten gibt und zusätzlich die Kompatibilität zu hellen und gleichzeitig dunklen Themes gewährleistet.
- Links zu Index, Foren, Posts, User & Topics bitte nur über die $SEO Klasse erzeugen! So werden unabhängig vom gewählten Link-Modus alle Links im gesamten Forum immer korrekt umgeschrieben bzw. dargestellt.
- Sie können den schließenden PHP-Tag (?>) am Ende von PHP-Dateien weglassen. Dies vermeidet Fehler durch eventuell von manchen Programmen angefügten Zeilenumbrüchen. Vor dem Beginnenden PHP Tag (<?php) sollten keine Umbrüche oder Leerzeilen sein. Bitte verwenden Sie aus Gründen der Kompatibilität niemals den Short-Tag (<?).

Plugin Entwicklung

Ein neues Plugin mit dem Plugin Developer anlegen

> **HINWEIS:**
> Bevor Sie mit der Programmierung eines Plugins beginnen, sollten Sie die Konstante „DEBUG“ in der Datei includes/config.php auf „true“ stellen. So werden alle Fehlermeldungen ausgegeben, die bei der Entwicklung entstehen können. Probleme in späteren Systemen oder Hinweise im error_log wegen uninitialisierter Variablen werden somit direkt vermieden.

Was ist der Plugin Developer?
Der Plugin Developer ist ein kleines Tool, welches Sie im Administrationsbereich des CF4 in der Kategorie „Developer“ vorfinden:



Dieses Tool zeigt Ihnen nicht nur Hilfestellungen an, wie Sie gezielt Hookpoints im CF4 finden, sondern unterstützt sie auch beim Einfügen von PHP Code an die entsprechenden Hookstellen. Außerdem legt dieses Tool bei jedem neuen Plugin-Projekt automatisch die nötigen Projekt- und Installationsdateien an. Nachdem Ihr Plugin fertiggestellt ist können Sie außerdem automatisiert eine Datei erzeugen lassen, welche den nötigen PHP Code für die Installation der Hookpoints für Ihr Plugins beinhaltet, welchen Sie dann in der package_setup.php verwenden können. Sie sollten Plugins also immer mit Hilfe des Plugin-Developers entwickeln, da dieser Ihnen bereits viele Schritte für ein validen CF4 Plugin abnimmt.

Umgang mit den angelegten Plugindateien – Wo finde ich was?
Der Plugin-Developer legt einen von Ihnen bestimmten neuen Ordner mit Ihrem Plugin im Ordner modules/ des CF4 an. Dieser enthält zu Beginn folgende Dateien:

- package_info.php
  Hier können Sie einige Informationen zu Ihrem Plugin und zu dem Autor einfügen.

- package_setup.php
  In dieser Datei ist der Code für die Installation, Deinstallation, Admin-Navigation, Administrationsseite sowie Updateroutine des Plugins vorhanden. Innerhalb der Datei wird jede Klassenfunktion durch PHPDoc näher beschrieben.

- package_icon.png
  Diese Grafik können Sie mit einem Bildbearbeitungs-Programm anpassen. Es ist ein Icon, welches für Ihr Plugin im ACP angezeigt wird. Bitte ersetzen Sie die Grafik ausschließlich mit einer Icondatei mit den gleichen Ausmaßen, da es ansonsten u.U. zu Darstellungsfehlern im ACP kommen kann. Es empfiehlt sich die vom Plugin Developer angelegte Grafikdatei einfach zu editieren, ohne die Dimensionen der Grafikdatei zu verändern.

Eigene Templatedateien oder Sprachdateien mit dem Plugin mitliefern
Näheres hierzu erfahren Sie in den Unterpunkten „eine Sprachdatei für ein Plugin einbinden“ sowie „eine Templatedatei aus einem Plugin einbinden“. Wenn

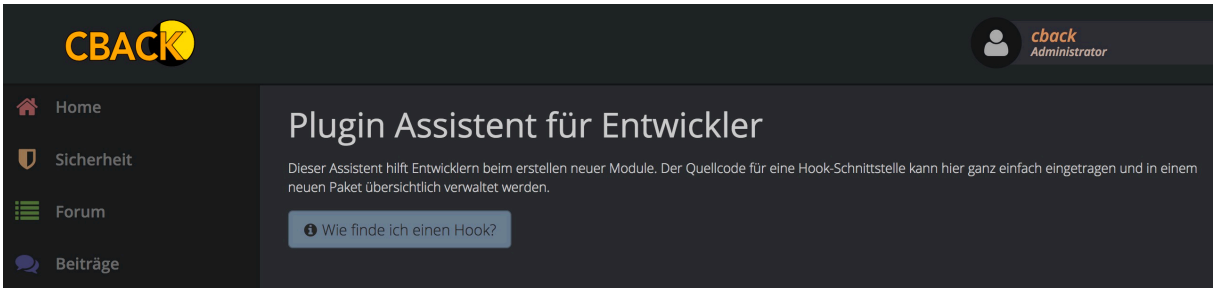
Ihr Plugin Sprachdateien und Templatedateien verwendet sollten Sie folgende Ordner anlegen:

- einen Ordner „tpl“, welcher alle Templatedateien des Plugins beinhalten wird. Bitte benennen Sie die Templatedateien mit Ihrem Autor-Prefix und Pluginnamen, um eventuelle Konflikte zu vermeiden (siehe Coding Guidelines).

- einen Ordner „lang“, welcher die Sprachdateien (.php) beinhaltet. Es muss mindestens eine Datei „default.php“ geben, welche die Standard/Fallbacksprache des Plugins enthält, falls eine bestimmte Übersetzung des Plugins nicht vorhanden ist. Zusätzliche Übersetzungen können Sie mit den Sprachpaket-Kürzeln hinzufügen, beispielsweise „de.php“ für die deutsche Sprache, „en.php“ für die englische Sprache, etc. – Übrigens ist es nicht notwendig, eine bestimmte Übersetzung, die für die default.php verwendet wird zu duplizieren. Sofern die default.php also beispielsweise die englische Sprache als Standard beinhaltet, dann benötigen Sie keine zusätzliche „en.php“ mit dem selben Inhalt.

- Es empfiehlt sich, zusätzliche Plugin-Funktionen in eine eigene Klasse auszulagern. Ob Sie die Klasse in einen eigenen Unterordner „classes“ packen oder die zusätzlichen PHP Dateien direkt in den Plugin-Ordner legen bleibt Ihnen überlassen.

Wie finde ich Hooks?
Der Plugin-Developer stellt Ihnen im ACP direkt eine Hilfe zum Auffinden von Hookpoints in der Software bereit. Klicken Sie hierzu im Plugin-Developer einfach auf den Button „Wie finde ich einen Hook?“.



Aufruf einer eigenen Seite für mein Plugin / die Datei xt.php
Plugins befinden sich immer im Ordner „modules“. Dies bedeutet, dass Sie nicht selbst Root-Dateien für das CBACK Forum erzeugen müssen, sondern Ihr Plugin

immer über einen Hookpoint in der Software aufgerufen wird. Dies sorgt dafür, dass sich Plugins immer vollautomatisch installieren bzw. deinstallieren lassen und der Anwender selbst nie Dateien kopieren muss.

Nun kann es natürlich einmal vorkommen, dass Ihr Plugin eine ganz eigene Seite im Forum benötigt, da das Plugin eine Extrafunktion (z.B. ein Portal, eine Fotogalerie, eine Datenbankseite, etc.) im Forum hinzufügt und nicht nur eine bereits vorhandene Funktion erweitert.

Um eine solche Plugin-Seite aufzurufen haben wir im CF4 die Datei xt.php im Root-Folder bereitgestellt, die einen Plugin-Hook enthält:

```
eval($Core->hook_execute('xtension_route', 'cback'));
```

Sie können die Aufruffunktionen Ihres Plugins also an den Hookpoint xtension_route / cback einhängen. Nutzen Sie den Aufruf-Parameter $app und eine eindeutige Bezeichnung für den Aufruf Ihres Plugins. Beispiel: Sie haben ein Kalender-Plugin entwickelt und möchten den Handler für Ihren Kalender aufrufen. Ihr Pluginordner heißt: **autorname_calendar**.

Nutzen Sie in diesem Fall also am Besten den Aufruf der Datei wiefolgt:
xt.php?app=autorname_calendar

Angenommen Ihr Plugin stellt die nötigen Funktionen der Seite über eine Klasse bereit, dann würde der Code für den Hook xtension_route / cback wiefolgt aussehen:

```
if( $app == 'autorname_calendar' )
{
    include_once(PATH.'modules/autorname_calendar/myCalendar.class.'.EXT);
    $AutornameCalendar = new AutorCalendar();
    $AutornameCalendar->router();
}
```

Die hier gezeigte Beispielfunktion Ihrer Pluginklasse Namens **$Autorname-Calendar->router();** kann weitere Funktionen zum schalten ihrer Pluginseiten beinhalten. Beispielsweise über einen zusätzlichen Parameter „submode“ für Funktionen wie Termin hinzufügen, Jahresansicht, etc. Dies könnte in Ihrer Pluginklasse also wie folgt aussehen:

```php
class AutorCalendar
{
    public function router()
    {
        // CF4 Core Klasse in einen Alias packen
        $Core = Controller::getClass('Core');

        // Parameter "submode" von der URL Übergabe lesen
        $submode = $Core->get(GET, 'submode');

        // Schalten diverser Funktionen gemäß dem Submode
        switch($submode)
        {
            default:
                // Kein Submode, Kalender Startseite zeigen
                /* Code hier */
                break;

            case 'new':
                // Neuen Termin eintragen
                /* Code hier */
                break;

            case 'dayview':
                // Tagansicht zeigen
                /* Code hier */
                break;

            /** weitere Schalter hier **/
        }
    }


    /** weitere Klassenfunktionen hier **/
}
```

ACHTUNG:
Es gibt auch Hook-Points dieser Art in anderen Root-Dateien des CBACK Forums, die zumindest technisch betrachtet wie in diesem Schema beschrieben für eigene Pluginseiten genutzt werden *könnten*. (Beispielsweise ein Hook direkt in der **index.php**)

Um Konflikte mit dem Rewriter / URL Umschreiber oder Folgefunktionen zu vermeiden sollte allerdings für Plugins immer nur die Datei xt.php als Aufruf-Wrapper für eigene Pluginseiten verwendet werden!

Nutzen Sie die anderen Root-Hooks bitte nur, wenn Sie eine bereits vorhandene Systemfunktion des CF4 in Ausnahmefällen unbedingt mit Ihrer Pluginfunktion **ersetzen** möchten. – Beispielsweise wenn Sie eine eigene Variante der Teamseite programmieren und den Aufruf der CF4 Teamseite komplett „umlenken“ möchten. Weisen Sie in Ihrem Plugin bitte darauf hin, wenn Sie Systemfunktionen umlenken, da eventuell andere Plugins die in solchen Bereichen tätig werden dann nicht mehr funktionieren.

Korrekte Datei-Header für verschiedene Situationen

Diverse Dateien im CF4 benötigen spezifische Header zur Validität. Diese werden in diesem Kapitel aufgelistet.

Eigenständige Aufrufdateien
Normalerweise ist es nicht notwendig, dass Sie eigenständige Aufrufdateien für das CF4 (Root-Dateien) selbst erstellen müssen. (Siehe vorheriges Kapitel xt.php). Zur Vollständigkeit geben wir das nötige Schema einer solchen Datei jedoch trotzdem nachfolgend an:

```php
<?php
/**
 * xt.php
 *
 * @author Christian Knerr (cback)
 * @package CBACK_Forum
 * @version 4.0.0
 * @license CF4 EULA
 * @copyright (c) CBACK Software - www.cback.net
 */

// Init
define('CBACK', true);
define('PATH', './');
define('EXT', substr(strrchr(__FILE__, '.'), 1));

/** @noinspection PhpIncludeInspection - load the startup file */
require(PATH . 'includes/cback/init.' . EXT);
```

- Eine Aufrufdatei beginnt mit einem Header-Kommentar, der den Dateinamen, Autor, Dateiversion, Lizenz sowie Copyright Information enthält. Die Angabe @package ist optional.

- Einige notwendige Basiskonstanten müssen definiert werden. Dies ist die Schutzkonstante „CBACK“, die Variable für den Pfad der Datei (in diesem Fall ‚./‘ für den Root Ordner. In einem Unterordner wäre der Pfad beispielsweise ‚../../‘).

- Die Definition der Konstante ‚EXT‘, die in der Regel immer den Wert ‚php‘ enthält, jedoch auch die Änderung der globalen Dateiändung des CF4 für exotische Hoster ermöglicht (nicht empfohlen!).

- Das Einbinden der Initialisierungs-Datei ‚init.php‘ per require – dies startet z.B. die CF4 Session und erzeugt die grundlegenden Objekte und Umgebungsvariablen des Systems.

Eingebundene Dateien / Include Dateien / Klassendateien

Eingebundene Dateien, Include Dateien / Klassendateien sehen beispielsweise so aus:

```php
<?php
/**
 * cback_topic_tags.class.php
 *
 * ...
 *
 * @author    Christian Knerr (cback)
 * @version   1.0.0
 * @since     30.12.16 - 15:10
 * @copyright ©2016 CBACK Software - www.cback.net
 */

if ( !defined('CBACK') )
{
    exit;
}

class PluginCbackTopicTags
{
```

- Header Kommentar mit Informationen zur Datei / Autor / Copyright / Lizenz.

- Die Prüfung, ob die Schutzkonstante ‚CBACK' gesetzt ist. (if (!defined(...). Eigentlich ist bei Klassen- oder Funktionsdateien eine solche Prüfung nicht notwendig, der der enthaltene PHP Code nicht extern ausgelöst werden kann, doch sollte sie als „Failsafe" trotzdem immer überprüft werden. Dies vermeidet, dass eingebundene Dateien direkt im Browser aufgerufen werden können.

Sprachdateien

```php
<?php
/**
 * lang_index.php
 *
 * @author Douzeper, Madman-Maniac, Mark Eichmann (3rd party translators)
 * @package CBACK_Forum
 * @version 4.0.0
 * @license CF4 EULA
 * @copyright (c) 3rd party translators
 */

/* ************************************************************
 * DIESEN CODE BITTE NIEMALS ENTFERNEN ODER ÄNDERN!
 * PLEASE NEVER REMOVE OR CHANGE THIS CODE!
 * ************************************************************/
if ( !defined('CBACK') ) { exit; }
if ( empty($lang) || !is_array($lang) ) { $lang = array(); }


/* Sprachstrings für Forenindex */
$lang = array_merge($lang, array(
    'index_forum'           => 'Forum',
    /* .... */
    )
);
```

Das CF4 verwaltet Sprachdateien mittels der Variable $lang des Objektes Controller:: und mittels der Nachlade-Routinen $User->reload_lang_file und $User->reload_plugin_lang. Um Konflikte zu vermeiden werden alle Sprachvariablen per Array Merge miteinander verbunden. Zusätzlich sollte am Anfang jeder Sprachdatei geprüft werden, ob das Array $lang bereits gesetzt ist oder nicht. Dies vermeidet Fehler aufgrund von entweder ungültig initialisierten Variablen oder falschen Datentypen.

Eine Sprachdatei ist also wie folgt aufgebaut:

- Header Kommentar mit Informationen zur Sprache / Autor / Version / etc.

- Prüfung der Schutzkonstante CBACK, da Sprachdateien immer nur Einbinddateien sein können (sehr wichtig!)

- Prüfung, ob die Variable $lang bereits gesetzt ist, nicht leer ist und vom Typ Array ist. Ansonsten wird die Variable neu initialisiert.

- Die zusätzlichen Sprach-Strings werden per array_merge an ein eventuell bereits vorhandenes $lang Array angefügt. Im Screenshot ist der gesamte Aufbau einer Sprachdatei mit einem Beispiel Sprachstring gezeigt.

Die Schaltzentrale des CF4: Das Objekt Controller::

Das zentrale Objekt Controller:: beinhaltet einige grundlegende Funktionen zur Steuerung und Verwaltung aller grundlegenden Systemklassen und der Sprachdateien des CF4. Mit diesem Objekt ist es z.B. auch möglich Systemklassen in jeder Funktion zuzugreifen ohne den Befehl „global“ verwenden zu müssen. Außerdem verhindert der Controller, dass bereits registrierte Systemklassen nachträglich überschrieben werden. Zusätzlich wird sichergestellt, dass gewisse Zusatzklassen nur einmalig geladen werden – sofern mehrere Plugins die selben Klassen benötigen ist somit sichergestellt, dass nicht mehrfach PHP Speicher verbraucht wird und damit die Performance des Systems auch bei vielen Plugins hoch gehalten wird. Alle grundlegenden Arbeiten mit den Basisklassen des CF4 sollten daher immer über den Controller:: ausgeführt werden und nicht durch manuelles nachladen von Systemklassen (solche Klassendateien werden übrigens auch automatisch via Autoload bezogen und müssen nicht extra eingebunden werden).

In diesem Teil des Handbuches werden einige Funktionen des Controllers näher beschrieben.

Eine Klasse aus dem Controller laden

Um eine bereits geladene Klasse aus dem Controller zu laden und in einen Alias (z.B. in einer Klasse oder Funktion von Ihnen) zuzuweisen können Sie die Funktion Controller::getClass() verwenden:

```
public function generate_board_index($no_display = false)
{
    $Core       = Controller::getClass('Core');
    $User       = Controller::getClass('User');
    $DB         = Controller::getClass('DB');
    $Cache      = Controller::getClass('Cache');
    $UI         = Controller::getClass('UI');
    $BBCode     = Controller::getClass('BBCode');
    $lang       = &Controller::$lang;
```

Auch die Sprachvariablen können Sie so beziehen. Bitte beachten Sie jedoch, dass ein Alias für Sprachvariablen per &Controller::$lang zugewiesen wird, sodass das Sprach-Array nicht dupliziert wird und damit der Speicherverbrauch steigt. (siehe auch: Coding Guidelines).

Prüfen, ob eine Systemklasse bereits registriert ist und sie ggf. nachladen

Die Systemklassen $Core, $DB, $Cache und $User sind global im gesamten System verfügbar und sind damit immer im Controller:: registriert. Andere Klassen jedoch (z.B. $BBCode) müssen ggf. jedoch nachgeladen werden. Der Controller bietet hier die Möglichkeit zu prüfen, ob ein Objekt bereits registriert wurde. Diese geschieht mit dem Befehl:

Controller::isRegistered({Objektname})

Diese Funktion antwortet mit true (bereits registriert) oder false (nicht registriert). Sofern ein Objekt noch nicht registriert ist können Sie dieses dann nachladen und im Controller registrieren. Die Registrierung des Objektes im Controller erlaubt es dann anderem Programmcode auf die bereits geladene und initialisierte Klasse zuzugreifen – egal wo man sich während des Durchlaufs durch ein Script befindet. Dies sieht beispielsweise so aus:

```
if(!Controller::isRegistered('BBCode'))
{
    $BBCode = new \classes\board\BBCode();
    Controller::registerClass('BBCode', $BBCode);
}
else
{
    $BBCode = Controller::getClass('BBCode');
}
```

In diesem Beispiel wird zunächst geprüft, ob das BBCode Objekt bereits registriert ist (Codezeile 1). Wenn das Objekt noch nicht registriert ist, dann wird es erzeugt. ($BBCode = new \classes\board\BBCode();) – Aufgrund des CF4 Autoloaders ist es also **nicht** notwendig die entsprechende Klassendatei per include oder require nachzuladen! Systemklassen des CF4 werden immer durch den zugehörigen Namespace identifiziert und automatisch eingebunden.

Wurde das Objekt erzeugt wird es im Controller angemeldet, sodass man es später erneut nutzen kann und es nicht am Ende der Funktion verfällt (Controller::registerClass({Klassenname}, {Objektvariable});)

Der else-Zweig in diesem Beispiel wird aufgerufen, falls die BBCode Klasse bereits anderweitig geladen und initialisiert wurde. In diesem Fall wird lediglich ein Alias zum bereits Vorhandenen Objekt per Controller::getClass der Variable $BBCode zugewiesen. Im weiteren Verlauf dieses Script-Beispiels ist die BBCode Klasse also in beiden Szenarien immer wie üblich mittels $BBCode->... nutzbar.

Werteübergaben: Nutzen Sie die $Core->get() Funktion!

Alle Werteübergaben im CF4 sollten ausschließlich über die Klassenfunktion $Core->get() erfolgen – außer die Dateiverarbeitung mittels $_FILES wenn dies erforderlich ist.

Der Vorteil von $Core->get() gegenüber dem üblichen $_POST, $_GET, $_SERVER, $_COOKIE liegt darin, dass in diesem Fall das CF4 erste Absicherungsmaßnahmen beim entgegennehmen des Wertes ausführt und ggf. auch noch zusätzliche Funktionen auf eine Übergabe anwendet – beispielsweise die Validierung von UTF8 Zeichensatz etc.

In der Datei **\classes\core\Core.class.php** finden Sie im PHPDoc zu dieser Funktion weitere Angaben über die möglichen Parameter.

Hier jedoch zur besseren Übersicht noch ein Vergleich zum gewohnten PHP Vorgehen für Werteübergaben und wie diese mit dem CF4 zu realisieren wären:

PHP Standard	CF4 Wrapper $Core->get()
`$test = $_GET['test'];`	`$test = $Core->get(GET, 'test');`
`if(isset($_GET['test'])) { … }`	`if($Core->get(GET, 'test', true)) { … }`

`$meinfeld = trim(htmlspecial-chars($_POST['var']));`	`$meinfeld = trim(htmlspecial-chars($Core->get(POST, 'var')));`
`$num = intval($_GET['id'])`	`$num = intval($Core->get(GET, 'id'));`
`if(empty($_POST['test'])){ … }`	`$var = $Core->get(POST, 'test');` `if(empty($var)){ … }`

Arbeiten mit der Datenbank ($DB)

Das Objekt $DB ist systemweit vorhanden und hat bereits eine Verbindung zur zugehörigen Foren-Datenbank aufgebaut.

DB Queries ausführen
Queries werden in 3-4 Schritten ausgeführt. Zuerst wird die MySQL Syntax übergeben. Hierbei ist zu beachten, dass alle Positionen in denen Werte stehen sollen mit einem Token (Doppelpunkt+Positionsnummer) durchnummeriert werden also **:1**, **:2**, **:3** usw. Diese werden dann später beim Ausführen mitgesendet und vom Datenbanksystem automatischabgesichert und entsprechend formatiert. Bitte beachten Sie, dass Feldnamen immer in `` eingeschlossen sein müssen. Also `feld` ist korrekt feld ist nicht korrekt. Wenn eine Rückgabe erwartet wird kann man diese mit $DB->fetch_assoc() abgreifen. Zum beenden einer MySQL Transaktion muss jede Datenbankaktion mit $DB->free() abgeschlossen werden. Dieser Befehl gibt interne Resourcen frei, die dann wieder zur Verfügung stehen. Man kann auch mehrere Queries ineinander verschachteln, nur ist hier ebenfalls zu beachten, dass Abfragen immer nacheinander mit $DB->free(); terminiert werden, da es ansonsten zu Fehlern in der internen Verwaltung der Resourcenzeiger kommen kann. Wichtig ist auch, dass Sie bei vorhandenen Tabellen bitte IMMER die in der **includes/cback/constants.php** genutzten Tabellenkonstanten verwenden, beispielsweise USERS für die Benutzertabelle, CONFIG für die Konfigurationstabelle etc. Bei Ihren Plugins ist es möglich den genutzten Tabellen-Prefix des installierten CF4 mit der Konstante PREFIX abzugreifen. Alle DB Tabellen die Ihr Plugin selbst mitliefert müssen mit diesem PREFIX angelegt werden, um eventuelle Konflikte bei mehreren CF4 Installationen in einer einzigen DB (mit unterschiedlichen Tabellen-Prefixen) zu vermeiden.

Eine Datenbank-Aktion die keine Rückgabe erfordert und zwei Werte übermittelt sieht beispielsweise so aus:

```php
$val1 = 'Test';
$val2 = 'Test2';

$DB->set_sql('UPDATE ' . USERS . ' SET `user_nickname`=:1 WHERE `user_nickname`=:2');
$DB->execute((string)$val1, (string)$val2);
$DB->free();
```

Achten Sie bitte darauf wie wir die Werte übergeben: Wir markieren die Position der Werte bei $DB->set_sql() mit den Tokens **:1** und **:2** und übergeben die beiden Variablen mit dem Inhalt dann bei $DB->execute() in der selben Reihenfolge. Zusätzlich werden aus Sicherheitsgründen die Variablentypen vor der Variable angegeben: *(int)* für Integer, *(string)* für Strings. Bitte senden Sie derartige Übergaben, egal ob als Variable oder als fester Wert **immer** im execute mit. Einzige Ausnahme von dieser Regel wären MySQL *IN* Befehle, die mit einem implode, welches vorher PHP Seitig abgesichert werden muss (nicht leer, nur die erwarteten Werte, etc.) direkt bei $DB->set_sql() erzeugt werden dürfen.

Übergaben eines festen, unveränderten Wertes sehen beispielsweise so aus:

```php
$DB->set_sql('SELECT * FROM ' . USERS . ' WHERE `user_nickname`=:1');
$DB->execute('cback');
$result = $DB->fetch_assoc();
$DB->free();
```

Wenn mehrere Rückgaben erwartet werden, dann können Sie den Empfang der Rückgabewerte in einer while Schleife durchführen. Wenn Sie die Werte später an das Template senden möchten legen Sie bitte ein separates Wertearray an in denen alle Rückgaben gespeichert werden. Dieser Standard erlaubt:

- die Nutzung von foreach und section Befehlen in $UI (Smarty)
- die nachträgliche Manipulation des Wertearrays nach der Schleife durch einen Hookpoint

Hier ein entsprechendes Beispiel:

```php
// Initialisierung des Arrays für die Werte
$meine_user = array();

// DB Query ausführen
$DB->set_sql('SELECT * FROM ' . USERS . ' ORDER BY `user_nickname` ASC');
$DB->execute();
while($res = $DB->fetch_assoc())
{
    // Werte aus Result in das Ergebnis-Array transferieren
    $meine_user[] = $res;
}
$DB->free();
```

Extrafunktion des DB Wrappers: INSERT und UPDATE von Array erzeugen
Der DB Wrapper bietet unter anderem zusätzliche Funktionen, um ein MySQL Query für einen INSERT oder UPDATE Befehl automatisch auf Basis eines Wertearrays zu erzeugen. So sind sehr lange, unübersichtliche MySQL Queries vermieden und bestehende Arrays können ganz einfach für das Eintragen oder Aktualisieren von Werten in der Datenbank genutzt werden. Zusätzlich werden die Werte innerhalb dieses Arrays automatisch vor dem Einsetzen in die Datenbank abgesichert. Achten Sie bitte auf die Datentypen!

Diese Funktion ist mittels $DB->syntax_built() zu nutzen.

Beispiel eines INSERT Queries:

```
$werte = array(
    'feld1'         => 'wert1',
    'feld2'         => 10,
    'feld3'         => 'wert2'
);

$DB->set_sql('INSERT INTO ' . MYTABLE . $DB->syntax_built(SQL_INSERT, $werte));
$DB->execute();
$DB->free();
```

Beispiel eines UPDATE Queries:

```
$updatewert = 2;

$werte = array(
    'feld1'         => 'wert1',
    'feld2'         => 10,
    'feld3'         => 'wert2'
);

$DB->set_sql('UPDATE ' . MYTABLE . $DB->syntax_built(SQL_UPDATE, $werte) . ' WHERE `feld4`=:1');
$DB->execute((int)$updatewert);
$DB->free();
```

Natürlich können Sie die Arrays auch mehrstufig füllen:

```
$werte = array();
$werte['feld1'] = 'wert1';
$werte['feld2'] = 10;
$werte['feld3'] = 'wert2';
```

Wenn Sie einen bestimmten MySQL Befehl in dem Array durchrouten möchten, ohne, dass unsere automatischen Sicherungsfunktionen den Befehl beeinflussen sollen können Sie dies mit einem Tilde-Zeichen im Array-Key bewerkstelligen. Mit dieser Methode sind beispielsweise Zählererhöhungen eines Feldes möglich:

```
$werte['~counter'] = '`counter`=`counter`+1';
```

Erzeugen einer Pagination (Blätterfunktion) inklusive LIMIT für die Datenbank
Das CF4 besitzt eine Datenbankfunktion zum automatischen Erzeugen einer Blätterfunktion (Pagination) für eine Seite inklusive der automatischen Verwaltung der entsprechenden LIMITS für den Datenbank-Query.

Wenn eine Pagination auf einer Seite gewünscht ist, dann können Sie im Template jederzeit den Include *{include file='pagination.htm'}* an die gewünschte Stelle einfügen. Die Blätterfunktion – wenn vorhanden – wird dann automatisch an dieser Stelle in Ihrem Template ausgegeben. Sie können innerhalb des Templates diesen include Befehl auch mehrfach verwenden, beispielsweise wenn die Blätterfunktion ganz oben und ganz unten auf einer entsprechenden Werteseite auftauchen soll.

Beispiel einer automatisch erzeugten Blätterfunktion im PHP Code:

```
$limit = $DB->generate_pagination($anzahl_werte, $anzahl_pro_seite, 'testdatei.' . EXT);
$DB->set_sql('SELECT * FROM ' . TESTTABELLE . $limit);
$DB->execute();
while( $row = $DB->fetch_assoc() )
{
    // Code...
}
$DB->free();
```

Erklärung dieses Vorgehens im Einzelnen:

- zunächst wird eine Variable $limit erzeugt, die durch den Befehl $DB->generate_pagination() sowohl die Pagination erzeugt als auch den LIMIT Befehl für die DB in die entsprechende Variable zurückschreibt. Die Parameter der Funktion $DB->generate_pagination() sind in diesem Beispiel $anzahl_werte (wie viele Werte insgesamt vorhanden sind, muss zuvor z.B. per count Befehl ermittelt werden), $anzahl_pro_seite (wie viele Werte sollen pro Seite angezeigt werden?), 'testdatei.'.EXT (in diesem Fall heißt die Datei, mittels der geblättert wird bzw. die für die Ausgabe zuständig ist so).
- im SQL Befehl $DB->set_sql() wird dann $limit an den Query angehangen, damit die automatisch erzeugte *LIMIT* Syntax dem Query hinzugefügt werden kann.

Nähere Informationen zu diesem Befehl und dessen Anwendung finden Sie im PHPDoc der Datei **\classes\core\DB.class.php**

> **ACHTUNG:**
> Es gibt auch eine Funktion Namens **$DB->generate_seo_pagination()** mit zusätzlichen Parametern, die ebenfalls im PHPDoc der Datei **\classes\core\DB.class.php** beschrieben sind. Diese Funktion kann genutzt werden, wenn Sie eine Blätterfunktion für Bereiche mit Rewriting benötigen. Beispielsweise das Auflisten von Topics oder Benutzern. Gute Beispiele zur Nutzung dieser Funktion finden Sie z.B. in der **\classes\board\Forum.class.php** bei der Ausgabe der Topics.

Umgang mit dem Template-System ($UI)

Das Template System des CF4 basiert auf Smarty. Entsprechend ist in den Templatedateien selbst eine gewohnte Smarty-Syntax nutzbar. Auch PHP Seitig können die meisten Smarty Befehle direkt per $UI-> genutzt werden. Bitte verwenden Sie zur Ausgabe von Daten immer unser Templatesystem und **nicht** etwa einfache echo bzw. print PHP Befehle. Dies sorgt für einen störungsfreien Ablauf unserer Seite und eventuell gesendeter header und Komprimierungstechniken.

Dennoch gibt es zu beachten, dass die Ausgabe von Plugin-Templates sowie der Seite selbst beim CF4 leicht anders stattfindet. Verwenden Sie also bitte **nicht** den direkten Smarty Befehl $UI->display() zur Ausgabe einer Seite, sondern verwenden Sie die CF4 eigenen Funktionen, die nachfolgend beschrieben werden.

Auf einen Blick: Ein Template im CF4 ausgeben
Zunächst können Sie benötigte Template-Variablen mit $UI->assign() an das Template senden. Bitte benutzen Sie hier ggf. einen Prefix, um das Überschreiben von eventuelle gleichnamigen Variablen innerhalb einer Seite durch Ihr Plugin zu vermeiden. Alle zugewiesenen Variablen sind dann im Template mit gewohnter Smarty-Syntax nutzbar (auch in Schleifen).

Beispiel:

```
$UI->assign('EINZELWERT', $wert);

$UI->assign(array(
        'WERT1'         => $wert1,
        'WERT2'         => 'Hallo Welt',
        'WERT3'         => 10
    )
);
```

Die Ausgabe einer Template-Datei kann dann mittels $UI->queue() in die Warteschlange der Ausgabe hinzugefügt werden. Dies ist ein wichtiger Unterschied zum regulären Vorgehen mit Smarty: Die CF4 Seite wird erst ganz am Ende jedes Scriptdurchlaufes erzeugt. Alle per $UI->queue() in die Warteschlange gestellten Seiten werden dann in ihrer Reihenfolge ausgegeben, sobald der „Seitenabschluss“ ausgelöst wird.

```
$UI->queue('hello_world.htm');
```

Die eigentliche Ausgabe der Seite wird dann in der Regel automatisch ausgelöst, wenn Sie den CF4 Footer mit $Core->show_footer() ausgeben. Sollten Sie die Ausgabe der Seite einmal manuell auslösen wollen, dann verwenden Sie bitte $UI->ui_output_page(). Achten Sie in diesem Fall jedoch darauf, dass wirklich kein CF4 Footer getriggert wird, da Sie ansonsten u.U. die Seite doppelt oder in der falschen Reihenfolge ausgeben.

Inhalt einer Templatedatei in eine Variable schreiben
Manchmal ist es notwendig, den Inhalt einer Template-Datei direkt nur in eine Variable zu schreiben, beispielsweise um diese an einem anderen Template-Hook im CF4 weiterzuverwenden. Nehmen Sie in diesem Fall bitte nicht die gewohnte Smarty Funktion „fetch()“ sondern $UI->quick_fetch(). Diese Funktion unterstützt auch den Sekundärparameter für Plugin-Template-Ordner (siehe folgender Abschnitt).

Template aus einem Plugin(ordner) beziehen
Beim CF3 war es notwendig, den Pfad des Templates kurzzeitig auf das externe Plugin umzuschalten und nach Bezug des Templates aus einem Pluginordner wieder auf das eigentliche CF Template zurückzuschalten. Dieses Vorgehen war recht umständlich, weshalb die UI Komponente des CF4 dies automatisch übernehmen kann.

Um ein Template aus einem Pluginordner auszugeben können Sie bei den Befehlen $UI->queue() sowie $UI->quick_fetch() als zweiten Parameter den Ordnernamen Ihres Plugins im Ordner **modules/** angeben. Die benötigten Umschaltungen werden dann vom System automatisch vollzogen wenn sie benötigt werden.

In unserem Demo-Plugin „cback_topic_tags“ haben wir beispielsweise im modules/ Ordner „cback_topic_tags“ des Plugins selbst einen Unterordner Namens „tpl“ für die Templates angelegt. Um ein Template daraus zu beziehen sieht der benötigte Befehl also so aus:

```
$UI->queue('plugin_cbacktopictags_output.htm', 'cback_topic_tags/tpl');
```

Beachten Sie, dass hier aus der Perspektive des Ordners „modules“ der Pfad zum tpl Ordner in unserem Plugin cback_topic_tags angegeben wurde. Die Parameter mit $UI->quick_fetch() würde identisch aussehen.

Extrafunktion: $UI->noparallax()
Das CF4 verwendet im Standard-Template „MIRA“ im Header einen Parallax Effekt beim Scrollen. Sofern auf der von Ihnen genutzten Seite dieser Effekt nicht gewünscht ist, können Sie einfach mit dem Befehl $UI->noparallax() in Ihrem PHP Code vor der Ausgabe der Seite das Parallax-Verhalten abschalten.

Extrafunktion: $UI->add_header_code()
Die Funktion $UI->add_header_code() ermöglicht das einfache Hinzufügen von Code in den großen Mittelbereich des Standardtemplates „MIRA“ bzw. in den Headeraum jedes CF4 Templates. Dies wird beispielsweise in der Topicansicht genutzt, um den Topic-Titel dem Header hinzuzufügen oder bei der Forenübersicht für den Titel und die Beschreibung des Forums.

Diese Funktion empfängt zwei Parameter: $content sowie $autotag. Benötigt wird nur der erste Funktionsparameter. Wenn lediglich eine automatisch erzeugte h1 Überschrift gewünscht ist sieht der Befehl beispielsweise so aus:

```
$UI->add_header_code('Hello World');
```

Ist hingegen ein völlig eigener HTML Code als Ausgabe gewünscht, dann sieht der Befehl so aus:

```
$UI->add_header_code('<b>eigener</b> HTML Code hier', false);
```

Extrafunktion: $UI->add_footer_js()
Um eine eigene JavaScript Datei dem Footer der Seite hinzuzufügen, rufen Sie bitte die Funktion $UI->add_footer_js({relativer Pfad zu Ihrer JS Datei ausgehend vom Root}); auf. Dabei wird automatisch vom System eine Versionskennung an den Pfad angehangen *(?v=kennung)*, sodass z.B. nach Software Updates der Browser Cache eines Seitenbesuchers ein Signal erhält, dass er die Datei ggf. neu abrufen muss. Sofern Sie Ihre eigene JavaScript Datei jedoch mit eigenen Parametern aufrufen müssen, und diese Versionskennung hinderlich wäre, dann setzen Sie einfach den zweiten Funktionsparameter auf true: $UI->add_footer_js({relativer Pfad zu Ihrer JS Datei ausgehend vom Root}**, true**); - In diesem Fall wird das automatische Hinzufügen der Versionskennung unterbunden und der angegebene Pfad wird 1:1 ausgegeben. **WICHTIG:** Dieser zweite Funktionsparameter ist erst ab **CF >= 4.7.0** verfügbar!

Extrafunktion: $UI->add_onload_event()
Um eine eigene onload Aktion dem JavaScript der Seite hinzuzufügen, rufen Sie bitte die Funktion $UI->add_onload_event({JavaScript Code}); auf. Diese Funktion fügt ihren übergebenen JavaScript Code allen eventuellen onload Events einer Seite hinzu und arbeitet sie dann beim Aufruf der Seite ab. Dies vermeidet konflikte durch eventuell überschriebene onload Ereignisse bei mehreren Plugins gleichzeitig.

Verlinkungen und Rewriting – Nutzen Sie die $SEO Klasse!

Das CF4 unterstützt es, Links zu den Hauptbereichen wie Benutzerprofilen, Foren, Themen, etc. per mod_rewrite umzuschreiben. Im Administrations-Bereich ist es möglich, diese Funktion auf Wunsch zu aktivieren oder zu deaktivieren. Aus diesem Grund ist es wichtig, dass Sie alle Links in Ihren Plugins, die potentiell umgeschrieben werden können, über unsere $SEO Klasse laufen lassen. Zusätzlich bietet diese Klasse auch noch weitere Suchmaschinenrelevante Funktionen, die im Folgenden beschrieben werden.

Welche Bereiche lassen sich umschreiben?
Das CF4 unterstützt SEO Links für diese Bereiche des Forums sowohl auf PHP Seite als auch direkt als Smarty Template Tag:

- **Foren-Index / Home-Seite (index.php)**
  Links zu dieser Seite werden nicht als index.php sondern als Aufruf-URL des Forums umgeschrieben, um Duplicate Content zu vermeiden.

- **Foren Übersicht**
  Der Inhalt von Foren (Foren-Übersichten) in denen ggf. Subforen und die Themen gelistet werden. Als zusätzlicher Parameter ist hier noch die Seite (Parameter: „page“) nutzbar, um z.B. in Foren mit vielen Topics zu blättern.

- **Themen / Topics**
  Links zu bestimmten Themen im Forum können umgeschrieben werden. Als zusätzlicher Parameter ist hier ebenfalls der Parameter „page“ nutzbar, um in Themen mit vielen Beiträgen zu blättern. Zusätzlich ist als Extra-Angabe bei „page“ nicht nur die Seitenzahl nutzbar sondern auch die Angabe „fst_unread“, um zum ersten ungelesenen Beitrag in einem Topic zu springen.

- **Posts / Permalinks**
  Einzelne Beiträge / Postlinks können ebenfalls umgeschrieben werden. Dies wird z.B. zur Adressierung des letzten Beitrags in einem Topic auf Foren-Übersichten oder Topic-Übersichten verwendet. Die Nutzung dieses Permalinks ist kein Duplicate Content, da in jedem Topic automatisch im Quellcode der Canonical Link zum entsprechenden Topic generiert wird.

- **Benutzerprofile**
  Die Links zu Benutzerprofilen können ebenfalls umgeschrieben werden.

- **Links (Linkforen)**
  Gewisse Foren können auch als Link mit einem Linkzähler fungieren. Auch diese Art von Links können umgeschrieben werden.

Nutzung der Rewrite Funktion direkt im Template und in PHP Dateien
Ein Vorteil des Rewritings im CF4 ist es, im gesamten System die Funktion zum Umschreiben von Links nicht nur PHP Seitig, sondern auch direkt innerhalb eines Templates (.htm Datei) zu verwenden. Sofern Sie also irgendwo die Angaben zu ID und Name für einen Link verfügbar haben können Sie also überall direkt einen umgeschriebenen Link erzeugen lassen. Es ist ebenfalls möglich, einen Link nur mit einer vorhandenen ID zu generieren. In diesem Fall ergänzt das CF4 die fehlenden Angaben automatisch. Es ist jedoch empfohlen in jeder Liste den benötigten Namen mit zu erfassen, um Datenbank-Queries zu sparen (insbesondere bei Auflistungen mit vielen Links spart man so sehr viele unnötige Queries).

Rewriting im Template sieht beispielsweise so aus:

```
<a href="{rewrite mode='topic' id=$TOPICS[topics].topic_id title=$TOPICS[topics].topic_title page='fst_unread'}"
```

In diesem Beispiel wird ein Link zu einem Topic erzeugt (mode='topic'). Parameter id= gibt die ID des Topics aus einer Template-Variable an. Der Parameter title= beinhaltet den Topic-Titel aus einer Template-Variable. In diesem Beispiel soll zur ersten ungelesenen Seite gesprungen werden, daher ist der Parameter page='fst_unread' zu finden.

```
{rewrite mode='post' id=$TOPICS[topics].topic_lst_post_id title=$TOPICS[topics].topic_lst_subject}
```

In diesem Beispiel wird der Link zu einem spezifischen Post erzeugt (mode='post'). Die id= gibt die ID des Posts an, title= den Betreff des Postings. Eine Seitenzahl (parameter page=) ist in diesem Fall nicht erforderlich.

```
{rewrite mode='user' id=2 title='cback'}
```

Zur besseren Veranschaulichung hier noch ein Beispiel mit dem Umschreiben eines Benutzerprofils ohne die Wert aus Smarty Variablen, also ein statischer Link zu dem Benutzer mit dem Nicknamen „cback“ und der Profil-ID „2“.

Rewriting in PHP Dateien sieht so aus:
Natürlich kann das Rewriting auch in PHP Dateien genutzt werden. Die Parameter sind hier die selben. Zunächst werden benötigte Parameter in ein temporäres Array abgelegt. Dieses kann dann der Funktion $SEO->rewrite() übergeben werden. Die Rückgabe dieser Funktion ist der fertig umgeschriebene Link. Die URL zum Forum muss nicht extra angefügt werden, die Funktion erzeugt diese automatisch.

```
$seo_params = array('mode'=>'forum', 'id'=>$forum_infos['forum_id'], 'title'=>$forum_infos['forum_name']);
if(isset($DB->pag_current_pg_number) && intval($DB->pag_current_pg_number) > 1)
{
    $seo_params['page'] = (int)$DB->pag_current_pg_number;
}
$canonical_seo_url = $SEO->rewrite($seo_params);
```

In diesem Beispiel zu sehen: In der ersten Zeile wird ein Array $seo_params angelegt, welches die Werte für den modus (in diesem Fall Foren-URL), die ID sowie den Namen des Forums enthält.

Anschließend wird hier per if-Abfrage geprüft, ob der Link sich auf der ersten Seite des Zielforums befindet, oder ob zusätzlich der parameter „page“ erforderlich ist. Sofern die Seitenzahl größer ist als 1 (bei Seite 1 wird kein extra page Indikator benötigt) wird dem Array $seo_params der Parameter ‚page‘ und die entsprechende Seitenzahl hinzugefügt.

Anschließend wird in die Variable $canonical_seo_url der erzeugte umgeschriebene Link geschrieben.

> **HINWEIS:**
> Der Umgang mit den SEO Funktionen kann zunächst etwas verwirrend wirken. Eine nähere Erklärung finden Sie jedoch auch in der Datei \classes\seo\SEO.class.php im PHPDoc der Funktion rewrite().
> Zusätzlich empfiehlt es sich, Beispiele für das Rewriting direkt im CF4 anzusehen, um den Umgang mit SEO URLs und Weiterleitungen besser zu verstehen. Suchen Sie hierfür beispielsweise in den Dateien **\classes\board\Forum.class.php** sowie **\classes\board\Post.class.php** nach $SEO->rewrite(oder für Beispiele im Template nach {rewrite mode= in Dateien wie z.B. **index.htm**, **forumview.htm**, **topic.htm**, **search_result_topics.htm**.

Einen canonical Link für eine bestimmte Seite zuweisen

Hin und wieder ist es klug, einen canonical Link für eine bestimmte Seite einzusetzen, um Duplicate Content zu vermeiden. Das CF4 tut dies Beispielsweise in Topics, um der Suchmaschine immer nur einheitliche Topiclinks zu liefern, auch wenn die einzelnen Postings / Permalinks nach wie vor einzeln adressierbar sind.

Wenn Sie für eine Seite in Ihrem Plugin einen canonical Link benötigen, dann können Sie diesen generieren und mit der Funktion $SEO->set_canonical_url({Link}) zuweisen. Der canonical Meta-Tag wird dann automatisch beim Ausgeben der Seite erzeugt.

```
$canonical_seo_url = $SEO->rewrite($seo_params);
$SEO->set_canonical_url($canonical_seo_url);
```

OpenGraph Tags / zusätzliche Meta-Tags hinzufügen

Seiten wie Pinterest, Twitter und Facebook haben manchmal zusätzliche Meta-Tags, um diverse Informationen an die Seite zu liefern, die beim Teilen von Links genutzt werden können. Das CF4 liefert im Standard bei Foren und Topics solche Meta-Tags für Twitter und Facebook automatisch mit. Doch falls Sie per Plugin weitere solcher Tags benötigen, können Sie diese über die Funktion $SEO->set_og_tag() hinzufügen. In der Datei **\classes\seo\SEO.class.php** finden Sie im PHPDoc dieser Funktion nähere Erklärungen zu den Parametern.

Beispiel:

Hinzufügen von MetaTags für Facebook und Twitter für ein Unterforum:

```
$SEO->set_og_tag('twitter:title', $forum_infos['forum_name'], 'name');
$SEO->set_og_tag('twitter:description', $forum_infos['forum_desc'], 'name');
$SEO->set_og_tag('og:title', $forum_infos['forum_name'], 'property');
$SEO->set_og_tag('og:description', $forum_infos['forum_desc'], 'property');
```

Die hinzugefügten Tags werden beim generieren der Seite automatisch hinzugefügt.

Mode-Parameter des SEO Systems im Überblick

Wie im Kapitel „Nutzung der Rewrite Funktion…“ bereits gezeigt gibt es verschiedene Modi, um Links im Forum umzuschreiben. Hier finden Sie eine kurze übersicht über die möglichen Modi und deren Zusatzparameter. Beachten Sie bitte: Die Angabe „page“ ist immer optional. Auf Seite 1 wird dieser Parameter nicht benötigt, da die Seite 1 immer die Hauptadresse des Links ist. Beim Modus „to-

pic“ können Sie zusätzlich als Seitenangabe page='fst_unread' anstatt der numerischen Seitenzahl verwenden, wenn Sie automatisch auf den ersten ungelesenen Beitrag springen möchten.

> **ACHTUNG:**
> In der Funktion $SEO->rewrite (PHP) bzw. {rewrite ...} (Template) ist lediglich die Angabe ‚**mode**‘ sowie ‚**id**‘ erforderlich. Der Titel des Links kann, wenn er fehlt, automatisch vom System nachgetragen werden. Es wird jedoch **dringend** empfohlen der Funktion immer direkt auch den Namen des Links mitzuliefern (kann ja i.d.R. beim Abfragen der DB direkt mit bezogen werden). Dies spart zusätzliche Queries, was dafür sorgt, dass umgeschriebene Links absolut ohne zusätzliche Abfragen auf die DB erzeugt werden können. **Aus Performancegründen ist es daher dringend empfohlen, dass Sie immer auch den title-Parameter der Funktion mitliefern!**

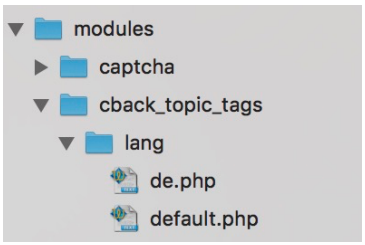
mode=	id=	title=	page=	Funktion
index	nicht vorhanden	nicht vorhanden	nicht vorhanden	Erzeugt einen Link zur Startseite des Forums. Zusätzliche Parameter sind hier nicht erforderlich.
root	nicht vorhanden	nicht vorhanden	nicht vorhanden	Dieser Parameter ist ab CF >= 4.7.0 verfügbar und gibt (z.B. zum Verlinken von Bildpfaden) nur das Wurzelverzeichnis des Forums zurück. Für solche Pfade sicherer als der Parameter „index“, da dieser z.B. von Portal Plugins umgeschrieben werden könnte.
forum	id des Forums	name des Forums	Seite für Blätterfunktion (nur wenn die Seite > 1 ist!)	Schreibt einen Link zu einem Forum um, auf dem ggf. Subforen sowie die Topicliste zu sehen ist.
category	id der Kategorie	name der Kategorie	nicht vorhanden	Diese Rewrite Variante ist aufgrund des neuen Foren-Index-Schemas von „MIRA“ nicht mehr genutzt (Kategorien sind nicht getrennt abrufbar). Dieser Modus ist nur dazu gedacht ggf. bei der Nutzung in Plugins zu funktionieren. Es wird jedoch nicht empfohlen, Kategorien extra zu verlinken.
link	id des Linkforums	name des Linkforums	nicht vorhanden	Bei Foren die als Link mit Linkzähler fungieren wird dieser Link generiert.
topic	id des Topics	titel des Topics	Seite für Blätterfunktion (nur wenn die Seite > 1 ist). Alternativ die Angabe ‚fst_unread‘, um zum ersten ungelesenen Beitrag zu springen.	Dies generiert den Link zu einem bestimmten Topic.
post	id des Postings	Betreff des Postings (**WICHTIG:** Nicht Topic Titel sondern post_subject!)	nicht vorhanden	Generiert einen Permalink auf ein bestimmtes Posting.
user	id des Benutzers	Nickname des Benutzers	nicht vorhanden	Erzeugt den Link auf ein Nutzerprofil.

Eine Sprachdatei für ein Plugin einbinden

Wie bereits im Kapitel **„Eigene Templatedateien oder Sprachdateien mit dem Plugin mitliefern“** angesprochen ist es möglich, dass beim CF4 automatisch die Sprachdateien eines Plugins – wenn vorhanden – nachgeladen werden können. Das aufwändige eigene Prüfen auf das vorhanden sein eines Sprachpaketes entfällt damit.

Um diese Funktion zu nutzen legen Sie im Ordner Ihres Plugins (z.B. modules/mein_plugin/) einen Unterordner namens „lang“ an. Dieser Ordner muss mindestens eine Datei Namens default.php beinhalten (aufgebaut nach dem Schema einer Datei für Sprachdateien), welche die Standardsprache des Plugins beinhaltet. Für jede zusätzliche Übersetzung des Plugins kann dann eine Datei mit dem Kürzel des Sprachpaketes erzeugt werden (z.B. en.php oder de.php). Die Sprache, die in default.php genutzt wird muss nicht zusätzlich mit einem kürzel angelegt werden.

Dies sieht dann beispielsweise so aus:



In diesem Beispiel wird im Plugin „**cback_topic_tags**“ die Sprache Deutsch mitgeliefert. Die Englische Sprache befindet sich in der Datei „**default.php**“ weshalb keine zusätzliche Datei „**en.php**“ notwendig ist: Das CF4 springt automatisch auf die Datei „**default.php**“ wenn keine Zusatzdatei zu einem Sprachpaket vorhanden ist.

Um die Sprachdateien automatisch in Ihr Plugin einzubinden schreiben Sie im PHP Code Ihres Plugins, wo Sie die Sprachen benötigen, einfach die Funktion $User->reload_plugin_lang({pluginordnername});. Die Sprachstrings Ihres Plugins werden automatisch dem Controller::$lang hinzugefügt. In unserem Beispiel-Plugin sieht das so aus:

```
// Load Plugin Language
$User->reload_plugin_lang('cback_topic_tags');
```

Eine existierende CF4 Sprachdatei nachladen

Manchmal kann es sein, dass Ihr Plugin ein zusätzliches Sprachpaket aus dem vorhandenen CF4 System benötigt, in dem sich weitere Sprachdateien befinden. Die enthaltenen Sprachpakete des CF4 sind im Ordner „**lang**“ zu finden. Jedes Sprachpaket hat einen eigenen Unterordner mit seinem Sprachkürzel, z.B. „**de**“ für die deutsche Sprache, „**en**“ für die englische Sprache.

Die Sprach-Strings aus der Datei „**lang_global.php**“ sind im gesamten System vorhanden. Sprachstrings für z.B. Topicfunktionen oder Posting befinden sich in eigenen Unterdateien, z.B. „**lang_topic.php**“ oder „**lang_post.php**“.

Wenn Ihr Plugin eine solche Zusatzsprache nachladen muss an einer Stelle, an der sie bislang noch nicht vorhanden ist, dann können Sie dies mit dem Befehl $User->reload_lang_file({paket}); durchführen.

Die Angabe von ‚lang_‘ und der Erweiterung ‚.php‘ ist nicht notwendig.

Um das Sprachpaket lang_search.php nachzuladen genügt also einfach:

```
$User->reload_lang_file('search');
```

Eine Templatedatei aus einem Plugin einbinden

Bitte beachten Sie hierzu das Kapitel zur $UI Klasse, die praktische Beispiele für Templates aus Plugin-Ordnern mittels $UI->queue() sowie $UI->quick_fetch() enthält.

Eigene JavaScript Dateien oder onload Ereignisse nachladen

Bitte beachten Sie hierzu das Kapitel „Extrafunktion: $UI->add_onload_event()“.

Eigene CSS Dateien oder Styleinformationen hinzufügen

Das CF4 Style ist so aufgebaut, dass auf inline-Styles verzichtet wird und jedes Element der Seite mittels CSS Klassen gestaltbar ist. Dies ermöglicht es, das Design viel besser anzupassen und Änderungen automatisch global auf der Seite umzusetzen. Wie bereits in diesem Handbuch angesprochen werden sogar dynamische CSS Klassen für die Usernamensfarben bzw. Gruppenfarben eingeführt.

Wenn Sie mit Ihrem eigenen Plugin zusätzlichen CSS Code oder zusätzliche CSS-Dateien hinzufügen möchten, dann Nutzen Sie dafür am Besten einen Hookpoint in der Datei **\classes\core\Core.class.php**:

```
eval($this->hook_execute('class_core_header_1', 'cback'));
```

Hängen Sie ihren eigenen Code bzw. HTML Syntax zur CSS Datei bitte per .= (anfügen) an die dortige lokale Variable $head_hook an. Bitte nutzen Sie nicht die direkte Zuweisung, um keinen Code von weiteren Plugins zu überschreiben sondern Ihren eigenen lediglich hinzuzufügen!

RICHTIG:
```
$head_hook .= 'mein HTML Code vor </head> hier';
```

FALSCH:
```
$head_hook = 'mein HTML Code vor </head> hier';
```

Super-Moderatoren Zugriff auf Ihr Plugin-ACP ermöglichen

Das CF4 ermöglicht es im ACP, einem Super-Moderator den Zugriff auf bestimmte Module des ACPs zu gewährleisten. Dies kann ebenfalls für Ihre Plugins genutzt werden. **Bitte denken Sie in diesem Fall aber daran, den Wert für die SMod-Access-Datenbank nicht nur bei der Plugin Installation in der Datenbank hinzuzufügen, sondern diesen Wert auch wieder bei der Deinstallation Ihres Plugins zu löschen!**

Das Hinzufügen Ihres Plugins geschieht per **INSERT DB Query** während der Plugin-Installation in die Tabelle mit der Konstante **SMOD_ACCESS**.

Es werden folgende Feldwerte benötigt:

- `smod_access_name`
  String Name Ihres Plugins oder des Plugin-Moduls. Alternativ kann dies auch eine globale ACP Sprachvariable sein. Wichtig: Zusätziche Sprach-Strings Ihres Plugins werden aus Sicherheitsgründen im ACP nicht berücksichtigt. Verwenden Sie stattdessen also einfach Ihren Plugin-Namen.

- `smod_access_file`
  Bei Plugins lautet dieser Wert immer „**admin_plugins**“. Auch wenn dieser Wert fest ist geben Sie ihn bitte bei der Deinstallation des Plugins zusätzlich an, um eventuelle fehlerhafte Löschungen von anderen ACP Bereichen mit gleichen Parameternamen zu vermeiden.

- `smod_access_mode`
  Tragen Sie hier den Wert ein, der beim Aufruf Ihres ACP Plugins in der URL als Parameter ?p= zu sehen ist. Normalerweise ist dies der Ordnername Ihres Plugins.
  Wenn das ACP Modul Ihres Plugins z.B. **admin_plugins.php?p=cback_topic_tags** lautet, dann tragen Sie in der Datenbanktabelle einfach „**cback_topic_tags**“ ein.

- `smod_access_on`
  Bitte setzen Sie diesen Wert bei der Installation **IMMER** auf 0! Dieser Wert wird später vom System verwaltet, ob ein Super-Moderator das Modul sehen darf oder nicht.

Benachrichtigungszentrale: Push Benachrichtigungen an Nutzer

ACHTUNG:
Bitte erstellen Sie **keine** Plugins oder Maßnahmen, welche die Benachrichtigungszentrale des CBACK Forums für einen Nutzer **generell** deaktiviert, da diese Funktion mitunter auch für wichtige Systemmeldungen genutzt werden kann. Bieten Sie stattdessen in Ihrem Plugin eine eigene Einstellung an, die lediglich Benachrichtigungen des jeweiligen Plugins selbst verhindert. Einfach umzusetzen wäre dies z.B. über den Hookpoint der „Extras“ Einstellungen des UserCP. Schauen Sie sich am besten einmal das Plugin „**CBACK UserMentions**“ an, in dem die Nutzung der CF Benachrichtigungszentrale mit ein/aus Schalter nur für dieses Plugin korrekt umgesetzt ist. Siehe dazu auch den Punkt „Plugin Einstellungen in den Punkt UserCP Extras hinzufügen“.

HINWEIS:
Die Funktionen der CBACK Forum Benachrichtigungszentrale sind nur bei CF Versionen **>= 4.5.0** verfügbar!

Das CBACK Forum stellt seit der Version 4.5.0 oder höher eine zentrale Benachrichtigungszentrale zur Verfügung, mit der Plugins Hinweismeldungen ganz einfach an Nutzer pushen können. Die Verwendung der Benachrichtigungszentrale ist sehr simpel: Zum Aussenden einer Benachrichtigung an einen Benutzer verwendet man lediglich die Funktion:

```
$User->push_notification({UserID Empfänger}, {Pluginname und/oder Zusatzdaten}, {Angezeigter Nachrichtentext, kann auch Links / HTML Code beinhalten});
```

Wie man sieht wird als erster Funktionsparameter einfach die Nutzer-ID des Empfängers angegeben. Die Nutzer ID muss hierbei höher sein als der Gastnutzer (Wert: 1 bzw. Konstante: ANONYMOUS) und muss einen bestehenden Foren-Account abbilden.

Der zweite Funktionsparameter ist üblicherweise der Pluginname. Es können jedoch – beispielsweise um später auf Vorgänge des Plugins zu reagieren, gewisse Benachrichtigungen wieder von einem Nutzer löschen zu müssen – weitere Daten dort mit eingetragen werden. In der Datenbank ist dieses Feld ein VARCHAR(255) und bietet daher genügend Platz für eigene Steuerdaten.

Im Beispiel des Plugins „**CBACK UserMentions**“ - welches zur Analyse aus der CBACK Community heruntergeladen werden kann - wird beispielsweise die Beitrags ID mit in den String für den zweiten Funktionsparameter aufgenommen:

Im folgenden Codebeispiel die Stelle mit: *('cback_umention_pid'.(int)$post_id)*

```
$User->push_notification((int)$v['user_id'], ('cback_umention_pid'.(int)$post_id), $rep);
```

In diesem Fall wird im Datensatz der Benachrichtigung die Post-ID der Benachrichtigung zusätzlich gespeichert, sodass man dann, falls beispielsweise ein Nutzer einen Beitrag mit einem erwähnten Nutzer bearbeitet und diesen entfernt, eine bereits gesendete Benachrichtigung wieder storniert werden kann. Somit wird vermieden, dass ein Plugin einen Nutzer mit Benachrichtigungen „zuspammt“ oder, dass ungültige Benachrichtigungen nachträglich wieder entfernt werden können. In der Datenbank würden die Einträge später so aussehen:

Für Post mit der ID 6:	cback_umention_pid6
Für Post mit der ID 10:	cback_umention_pid10
Für Post mit der ID 382:	cback_umention_pid382
	usw.

Innerhalb des Plugins muss das Entfernen einer bereits gesendeten Benachrichtigung dann mit einem DB Query ausgeführt werden. (Siehe dazu später im Beispiel).

Der Dritte und letzte Funktionsparameter beinhaltet die für den Nutzer angezeigte Benachrichtigung. In dieser Nachricht kann auch HTML Code (z.B. für einen Link zum Beitrag) verwendet werden. Es empfiehlt sich, dass man für Benachrichtigungstexte in seinem Plugin eine zentrale / globale Sprachdatei anlegt, die alle Übersetzungen beinhaltet. Durch Abruf der vom Nutzer gewählten Standardsprache können so Benachrichtigungen in der korrekten Sprache des Zielnutzers zugestellt werden. (Siehe folgender Abschnitt).

Benachrichtigung in der korrekten Sprache des Zielnutzers senden
Beim Plugin „CBACK UserMentions“ wird im Sprachordner **lang/** des Plugins eine globale Datei Namens **global_notify_strings.php** angelegt. In dieser Datei befindet sich ein Array mit den übersetzten Werten jeder verfügbaren Sprache des Plugins. Zusätzlich ein Fallback-Key Namens ‚default‘ der dann verwendet wird, wenn das Plugin keine Übersetzung zu einem anderweitig installierten Sprachpaket zur Verfügung stellt. **WICHTIG:** Verwenden Sie für Ihre Sprachvariablen einen eindeutigen Namen (im Beispiel „$cback_usermentions_notification_language“) und niemals den für CF4 Sprachdateien reservierten Namen „$lang“:

```php
<?php
/**
* global_notify_strings.php
* Global Notification Message Strings for all Languages Supported
*
* @author Christian Knerr
* @package cback_usermentions
* @version 1.0.0
* @copyright (C)2019 CBACK Software - www.cback.net
*/
if ( !defined('CBACK') ) { exit; }

$cback_usermentions_notification_language = array(
    'default'           => '<b>%s</b> mentioned you in a <a href="%s">post</a>.',
    'de'                => '<b>%s</b> hat Dich in einem <a href="%s">Beitrag</a> erwähnt.',
    'en'                => '<b>%s</b> mentioned you in a <a href="%s">post</a>.'
);

```

Im Plugin selbst wird an der benötigten Stelle vor dem Senden von Benachrichtigungen mit *$User->push_notification* diese Sprachdatei einfach per include_once eingebunden:

```php
$cback_usermentions_notification_language = array();
include_once(PATH.'modules/cback_usermentions/lang/global_notify_strings.'.EXT);
```

Die Sprache bzw. das Sprachpaket-Kürzel des Zielnutzers kann vorher aus der Nutzertabelle abgerufen werden. Das entsprechende Feld lautet `user_def_lang`:

```
$DB->set_sql('SELECT `user_id`, `user_nickname`, `user_color`, `user_def_lang`, `user_cback_umention_off` FROM ' . USERS . ' WHERE `user_id`>:1 AND `user_nickname`=:2');
$DB->execute(ANONYMOUS, (string)$v);
$res = $DB->fetch_assoc();
$DB->free();
```

Beachten Sie, dass wir in diesem Query auch gleich validieren, dass der Nutzer existiert und nicht der Gastnutzer ist (>ANONYMOUS). Zusätzlich legt das Beispiel Plugin mit dem vom Plugin angelegten DB Feld *`user_cback_umention_off`* in der Usrtabelle des Forums auch noch eine Steueroption an, mit der ein Nutzer die Benachrichtigungen des Plugins abschalten kann. Damit kann Nutzern ein Opt-Out von einem Plugin angeboten werden, ohne, dass das gesamte CF4 Benachrichtigungssystem lahmgelegt werden müsste.

An der verarbeitenden Stelle prüfen wir dann einfach, ob das Sprachpaket des Nutzers in den globalen Benachrichtigungsstrings vorhanden ist. Wenn nicht springen wir auf den ‚default' Eintrag:

```
$rep = isset($cback_usermentions_notification_language[$v['user_def_lang']])? $cback_usermentions_notification_language[$v['user_def_lang']] : $cback_usermentions_notification_language['default'];
```

^ Übersetzung in Nutzersprache vorhanden? ^ vorhanden & wird benutzt. **ansonsten** ^ nicht vorhanden, default

Eine bereits gesendete Benachrichtigung löschen
Wie bereits erwähnt haben wir beim CBACK UserMention Plugin im Kennungs-Parameter für die Benachrichtigung die Post-ID hinzugefügt. Somit ist es neben der Zielnutzer-ID genau möglich mit *('cback_umention_pid'.(int)$post_id)* eine exakte Meldung erneut anzusteuern bzw. in diesem Fall zu löschen. Auf Wunsch sogar global für alle Nutzer, welche diese Benachrichtigung eventuell erhalten haben. (z.B.: wenn besagter Beitrag komplett gelöscht oder editiert wurde).

Beispiel 1:
Alle Meldungen, die von dem CBACK UserMention Plugin zu einer bestimmten Post-ID gesendet wurden, werden global für alle Nutzer entfernt:

```
$DB->set_sql('DELETE FROM ' . NOTIFY . ' WHERE `notify_plugin`=:1');
$DB->execute(('cback_umention_pid'.(int)$res['post_id']));
$DB->free();
```

Beispiel 2:
Eine vom CBACK UserMention Plugin gesendete Meldung zu einer bestimmten Post-ID wird nur für eine **bestimmte** Nutzer ID entfernt:

```
$DB->set_sql('DELETE FROM ' . NOTIFY . ' WHERE `notify_plugin`=:1 AND `notify_user`=:2');
$DB->execute(('cback_umention_pid'.(int)$pid), (int)$user_id);
$DB->free();
```

Dies kann anfänglich ein wenig kompliziert wirken; wir empfehlen jedoch durch Analyse des CBACK UserMention Plugins die Funktion von Benachrichtigungen in der Praxis zu betrachten. Dann wird man mit dem Notification System des CF4 sehr schnell Plugins mit praktischen Benachrichtigungsfunktionen erstellen können. Übrigens: In diesem Plugin wird auch gezeigt, wie man den OPT-OUT auf die Seite „UserCP“ => „Extras“ hinzufügen kann ohne hierfür eigene komplizierte UserCP Seiten programmieren zu müssen. Wenn alle Plugins mit Benachrichtigung diese Stelle nutzen, dann macht es das für den Nutzer sogar sehr übersichtlich alle Plugins mit Benachrichtigungen an zentraler Stelle ein- bzw. auszuschalten.

Die Verwaltung von Benachrichtigungen (z.B. als gelesen markieren, vom Nutzer löschen lassen, etc.) wird übrigens vom CF4 automatisch übernommen und durchgeführt. Ein Plugin muss also für diese Fälle keinen zusätzlichen Code vorsehen.

Plugin Einstellungen in den Punkt „UserCP Extras“ hinzufügen

> **HINWEIS:**
> Während der Pluginentwicklung setzen Sie bitte den Wert des Datensatzes „**extra_settings_active**“ der Datenbank Tabelle *{prefix}_dynamic_values* von **0** auf **1**, sodass der entsprechende Punkt im UserCP angezeigt wird. Bei der regulären Installation eines Pluginpaketes nimmt das CBACK Forum die Anpassung dieses Wertes seit CF v4.7.0 **automatisiert** vor, wenn der Hookpoint dieses Einstellungsmenüs durch irgend ein Plugin angesprochen wird. Für CF4 Versionen älter als 4.7.0 können Sie zum aktivieren des UserCP „Extras“ Punktes in der Installationsroutine Ihres Plugins den Wert auch manuell auf 1 setzen mit folgendem Query:
>
> ```
> $DB->set_sql('UPDATE ' . DYNAMIC_VALUES . ' SET `value`=1 WHERE `name`=\'extra_settings_active\'');
> $DB->execute();
> $DB->free();
> ```

Manchmal kann es sein, dass ein Plugin eigene Einstellungen zur Nutzertabelle hinzufügt, das Plugin allerdings so wenige Einstellungen bietet, dass das Anlegen einer eigenen Seite im UserCP übertrieben und unübersichtlich wäre. Zu diesem Zweck haben wir mit dem UserCP Punkt „Extras“ im CF4 eine zentrale Position eingerichtet, in der Plugins mit wenigen Einstellungen zentral ihre eigene Optionen hinzufügen können. Als Beispiel dafür kann zum Beispiel das Plugin „CBACK UserMentions“ aus der CBACK Community benutzt werden. Dieses nutzt die

UserCP „Extras“ funktion für seine Einstellung, ob Benachrichtigungen von diesem Plugin vom Nutzer gewünscht werden oder nicht. Im folgenden Handbuch-Beispiel verwenden wir ebenfalls den Code dieses Plugins.

Schritt 1: Die Funktion zur Ausgabe und Speicherung in Ihrer Plugin Klasse hinzufügen

Zur Verwendung einer eigenen Einstellung im UserCP „Extras“ Menüpunkt des UserCPs benötigen Sie in einer Klasse Ihres Plugins zwei Methoden, welche Sie später per Hook in den „Extras“ Menüpunkt einhängen können. Die erste Methode dient zur Ausgabe der Einstellungsoption(en), die zweite Methode zur Speicherung der Daten.

Im Beispiel des CBACK UserMentions Plugins wird für das Anzeigen des Einstellungspunktes zunächst in der Funktion ein entsprechender HTML Code vorbereitet und mit **return** zurückgeliefert. Bei mehreren Einstellungspunkten empfiehlt es sich, ggf. eine eigene Plugin-Templatedatei anzulegen und diese dann mit *return $UI->fetch(....* zu compilieren und zurückzugeben. Die Methode im Beispielplugin sieht so aus:

```php
/**
* Show UserCP Settings (Extras Panel)
*/
public function usercp_extras_show()
{
    $lang = &Controller::$lang;
    $User = Controller::getClass('User');

    $is_off_stat = false;
    if($User->get_user_value('user_cback_umention_off')==1)
    {
        $is_off_stat = true;
    }

    return '<div class="uk-margin-small"></div><div class="cf-box1 cf-box-border cf-box-padding">
    <b>'.$lang['cback_usermentions_plugin']['usercp_head'].'</b><br />'.$lang['cback_usermentions_plugin']['usercp_info'].'
    <br /><br />
    <input type="radio" name="cback_usermention_off" value="1"'.((!$is_off_stat)? '' : ' checked="checked"').' /> ' . $lang['cback_usermentions_plugin']['usercp_no'] . '<br />
    <input type="radio" name="cback_usermention_off" value="0"'.(($is_off_stat)? '' : ' checked="checked"').' /> ' . $lang['cback_usermentions_plugin']['usercp_yes'] . '<br />
    </div><div class="uk-margin-small"></div>';
}
```

Als nächstes benötigen wir noch eine Methode, welche nach dem Absenden des UserCP „Extras“ Formulares den Wert aus dem zuvor angelegten HTML Feld ausliest und in der Datenbank speichert. In unserem Beispiel-Plugin handelt es sich um eine vom Plugin hinzugefügtes Feld in der Nutzertabelle selbst, natürlich können Sie mit eigenen Queries jedoch auch Werte in anderen Tabellen Ihres Plugins speichern und verändern:

```php
/**
* Store UserCP Settings (Extras Panel)
*/
public function usercp_extras_parse()
{
    $User = Controller::getClass('User');
    $Core = Controller::getClass('Core');

    $newval = intval($Core->get(POST, 'cback_usermention_off'));
    $newval = ($newval <= 0)? 0 : $newval;
    $newval = ($newval >= 1)? 1 : $newval;

    $User->set_user_value(false, (int)$newval, 'user_cback_umention_off');
}
```

Schritt 2: Die beiden Methoden zur Eingabe und Ausgabe per Hook „einhängen“
Damit das „Extras“ Formular im UserCP die beiden Plugin Funktionen korrekt verwendet, müssen diese am besten via ACP Plugin Developer an die entsprechenden Hooks „eingehangen“ werden. Üblicherweise wird hier zunächst (falls nicht bereits vorhanden) die Klassendatei Ihres Plugins einmalig eingebunden und jeweils die zuvor angelegten Klassen aufgerufen.

Zur Ausgabe sieht dies so aus:

```
1  $ex_html_output .= $CBACKUsermentions->usercp_extras_show();
```

WICHTIG: Achten Sie darauf, dass die Ausgabe mit .= statt nur = an die Variable $ex_html_output ausgegeben wird, sodass die Werte hinzugefügt und nicht ersetzt werden. Ansonsten sind andere Plugins an dieser Stelle u.U. nicht mehr sichtbar.

Die Hook-Adressierung für diesen Code lautet: **HookKey:** class_usercp_extras_4, **Hook-SID:** cback

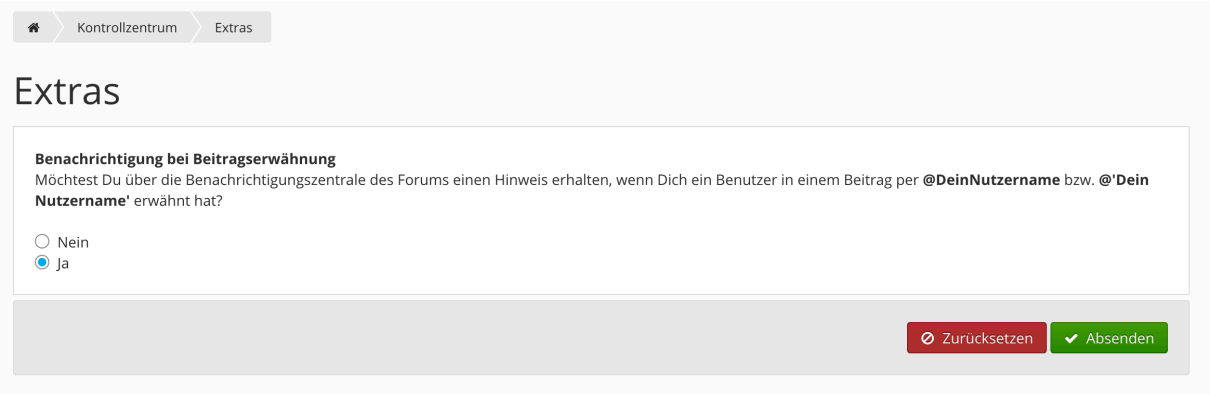
Zur Speicherung der Daten sieht der Hook Code so aus:

```
1  $CBACKUsermentions->usercp_extras_parse();
```

Er befindet sich an der Hook-Adressierung: **HookKey:** class_usercp_extras_2, **Hook-SID:** cback und wird automatisch nur dann getriggert, wenn im UserCP das Formular tatsächlich vom Nutzer abgesendet wurde.

Im UserCP sieht man, dass der neue Punkt „Extras“ hinzugekommen ist:

Wird dieser aufgerufen sehen wir die mit unserem Plugin hinzugefügte Option:



CFProtect: Schützen Sie Ihre Plugins vor CSRF & Phishing

Das CBACK Forum stellt ein mit mehreren Layern geschütztes System (Token → Referrer → Schutzcookie → Zeitverhalten) für sensible Formular- oder Linkübertragungen bereit, mit dem es möglich ist, Ihre Nutzer besser vor Phishing, manipulierten Webseiten oder sogenannten Cross Site Resource Forgery (CSRF) Angriffen zu schützen. Außerdem verhindert das System zur Analyse des Zeitverhaltens auch gängige automatische Botscripte, die beispielsweise versuchen, Werbethemen zu posten oder Spamaccounts zu registrieren.

Mit CBACK Forum in der Version **4.7.0 oder höher** wurde dieses System deutlich verbessert, erweitert und konsequent überall im CBACK Forum angewendet. Zusätzlich haben wir das System seit dieser Version so gestaltet, dass auch Plugin Autoren sehr leicht von dem davon bereitgestellten Sicherheitsgewinn profitieren können. In diesem Abschnitt zeigen wir Ihnen, wie gängige Plugin Datenübertragungen mit GET, POST oder AJAX Requests mit dem CFProtect System abgesichert werden können. Sofern Sie diese Praktiken anwenden müssen Sie Ihre Plugin Kompatibilität auf CF >= v4.7.0 kennzeichnen, da frühere Versionen des CBACK Forums die hier genannten automatischen Funktionen nicht bereitstellen.

Sensible $_POST Übertragungen mit CFProtect absichern

$_POST Übertragungen finden normalerweise über die Konstellation statt, dass irgendwo in einem Template ein *<form>* Tag mit *method="post"* existiert und später im PHP-Bereich mittels *if ($Core->get(POST, 'feldname', true))* geprüft wird, ob das Formular abgesendet wurde. Innerhalb dieser Logik ist das Nachrüsten von CFProtect sehr leicht möglich.

Zunächst wird im Template direkt unter dem öffnenden *<form>* Tag die Templatevariable *{$CFPROTECT_FORMTOKEN}* hinzugefügt, um dort das automatische Hinzufügen der zusätzlichen Sicherheits- und Verifikationsparameter durch das CFProtect System zu ermöglichen. Hier ein Beispiel, wie es im Login-Template des CF4 gelöst ist:

```
<form action="{$U_NAV.login}" method="post" name="login" class="uk-form uk-form-stacked uk-form-horizontal">
    {$CFPROTECT_FORMTOKEN}
    <input type="hidden" name="redirect" value="{$F_REDIRECT}" />
```

Auf PHP Seite wird einfach direkt dort (so früh wie möglich), wo das gesendete Formular verarbeitet wird die Methode *$Core->cfprotect_check();* aufgerufen:

```
// Do Login
if ( $Core->get( source: POST, name: 'login_send', icheck: true) )
{
    // Token Check
    $Core->cfprotect_check();
```

Damit wird das vom Plugin abgesendete $_POST Formular automatisch durch CFProtect gesichert. Ein weiterer Code oder das Handeln von Fehlermeldungen ist nicht erforderlich. Der Rest der Programmierung erfolgt ganz wie gewohnt.

Sensible $_GET Daten / Links mit CFProtect absichern

Sensible $_GET Übertragungen oder Links (z.B. Löschen-Links) können ähnlich einfach mit CFProtect abgesichert werden. Zunächst wird auf Template-Seite einem Hyperlink oder dem action-Parameter eines $_GET Forms mit der Ziel-URL die Template Variable *{$CFPROTECT_GETTOKEN}* hinzugefügt. Bitte beachten Sie, dass vor dieser Variable noch das korrekte Query-String Bindezeichen hinzugefügt werden muss (also je nach Link "?" oder "&").

Beispiel 1:

```
<a href="search.{$EXT}?mode=markread&amp;all=true&amp;{$CFPROTECT_GETTOKEN}">
```

Der hier gezeigte Link hat bereits eigene Parameter. Es wird also der hier markierte Teil *&{$CFPROTECT_GETTOKEN}* hinzugefügt.

Beispiel 2:

```
<a href="modules/helloworld/deleteme.php?{$CFPROTECT_GETTOKEN}">
```

Dieses wohl seltener vorkommende Beispiel ist ein Link ohne eigene Parameter im Query-String. Entsprechend wird der markierte Teil *?{$CFPROTECT_GETTOKEN}* hinzugefügt.

Auf PHP Seite wird dann lediglich beim Prüfen des Aufrufs die Methode *$Core->cfprotect_getcheck();* verwendet, damit das CFProtect System die Schutzparameter überprüfen kann. Wie beispielsweise hier:

```
$Core->cfprotect_getcheck();

$lid = intval($Core->get( source: GET,  name: 'lid'));
if ( $lid != 1 && $lid != 2 )
```

Damit ist die Absicherung von $_GET Übergaben bzw. sensiblen Links bereits erledigt. Das CFProtect System kümmert sich automatisch um die notwendigen Datenverifikationen.

Sensible AJAX Requests mit CFProtect absichern

Mit dem CFProtect System ist es ebenfalls möglich, AJAX Übertragungen abzusichern und zu validieren. An dieser Stelle empfehlen wir gleichzeitig, eigene Plugin-AJAX Funktionen **mit einem sehr eindeutigen Mode-Namen** - um Kollisionen zu vermeiden - (z.B. *"meinPlugin_funktionsname"*) über den Hookpoint der CF4 eigenen **ajax.php** laufen zu lassen, da diese Datei automatisch noch viele weitere Absicherungen und Verifikationen für XHR / AJAX Requests ausführt.

Der entsprechende Hookpoint des AJAX Systems lautet:

```
eval($Core->hook_execute('ajax_php', 'cback'));
```

Im JavaScript Bereich wird das URL Ziel Ihres AJAX Requests mit der Funktion:

```
append_get_hash(hs, tm, '&')
```

erweitert. Wie auch im $_GET Beispiel zuvor bestimmen Sie durch den dritten Parameter dieser Funktion (in diesem Fall mit einem '&'), ob bereits eigene QueryString Übergaben an der URL vorhanden sind und entsprechend weitere Parameter mit "&" angehangen werden, oder ob die URL lediglich auf eine fixe Datei zeigt, noch keine Parameter hat und die Verknüpfung also zunächst mit "?" beginnen soll. Die ersten beiden hier gezeigten Parameter **hs** (Hashcode) sowie **tm** (Timestamp) wurden zuvor an die Parent-Funktion übergeben. Die entsprechenden Werte können beim Aufruf dieser „zündenden" JavaScript Funktion im HTML Bereich (Template) mit den Templatevariablen **{$CFPROTECT_SHORTTOKEN}** für den Hash sowie **{$CFPROTECT_TIME}** für den Timestamp an die JavaScript Funktion übertragen werden. Die Werte hs und tm sind nicht global im

JavaScript verfügbar, da dies eventuelles auslesen dieser Werte für Angreifer zusätzlich erschwert. Diese Parameter müssen also vor der Verwendung immer selbst übergeben werden.

In einem Codebeispiel kann das bilden einer AJAX Aufruf URL mit der append_get_hash Funktion übrigens beispielsweise so aussehen:

```
var url = PHP_PATH + 'ajax.' + PHP_EXTENSION + '?mode=topicrating&t=' + id + '&ra=' + value + append_get_hash(hs, tm, begin: '&');
$.ajax({ url: url, timeout:14000, async:true}).done(function(data){ ajax_indicator( start_transaction: false); $('#'+'ratet'+id).html(d
```

Auf PHP Seite verwenden Sie zur Verifikation einfach wieder die gettcheck Methode *$Core->cfprotect_getcheck();* um die übergebenen Daten zu validieren.

Um dies noch einmal verständlicher zu veranschaulichen hier ein kurzes Beispiel des gesamten Vorgehens:

```
// HTML Bereich:
// Aufruf einer eigenen JavaScript Funktion inklusive Übergabe der Hash & Time-Parameter

<a href="javascript:void()" onclick="meineAjaxFunktion('machetwas', '{$CFPROTECT_SHORTTOKEN}', {$CFPROTECT_TIME});">meine AJAX Funktion auslösen</a>

// Beachte den String Einschluss von {$CFPROTECT_SHORTTOKEN} mit Hochkommas ('')!


// JavaScript Bereich:
// Deine eigentliche JavaScript Funktion für den AJAX Request. Beachte die Übergabe von hs (Hash) und tm (Time) bei den Funktionsparametern

function meineAjaxFunktion(mode, hs, tm)
{
    // hier dein js code

    // AJAX URL inlusive append_get_hash Anhängsel für CFProtect erzeugen
    var meine_AJAX_URL = 'test.php?mode=' + mode + append_get_hash(hs, tm, '&');

    // weiterer js code
}




// PHP Seite:
// Validieren des Tokens

$Core = Controller::getClass('Core');

if ( $Core->get(GET, 'mode', true) && $Core->get(GET, 'mode') == 'machetwas') )
{
    $Core->cfprotect_getcheck();


    // weiterer verarbeitungscode wie gewohnt hier
}
```

Zusatzhinweis zur Verwendung der Prüfmethoden (PHP Seite)
Wie in den Abschnitten zuvor unter $_POST und $_GET gezeigt, kann die Überprüfung der übermittelten Daten ganz einfach durch das Verwenden der beiden $Core Methoden *$Core->cfprotect_check();* bzw. *$Core->cfprotect_getcheck();* stattfinden.

Im Normalfall ist diese Variante auch schon ausreichend: Das CF4 prüft in diesem Fall automatisch die Validität der Daten, sichert diese ab und gibt im Fehlerfall automatisch eine entsprechende Fehlermeldung aus und hält die weitere Ausführung des Scriptes an.

Zusätzlich wird bei $_POST Übergaben hier auch die Zeitanalyse gegen Botscripte automatisch ausgeführt.

In seltenen Fällen kann es jedoch auch vorkommen, dass Plugin Autoren die Validität einfach nur prüfen möchten (ohne Zeitanalyse) und ggf. selbst eine Fehlermeldung ausgeben möchten oder notwendigerweise noch gewisse Dinge in Ihrem Code ausführen, bevor sie selbst die Ausführung des Scripts beenden möchten.

Wenn so eine Situation vorliegt, können die beiden Prüfmethoden einfach mit einem gesetzten true-Parameter aufgerufen werden, also sprich mit *$Core->cfprotect_check(**true**);* bzw. *$Core->cfprotect_getcheck(**true**);* . In diesem Fall gibt die Funktion lediglich ein true oder false zurück basierend auf dem Prüfungsergebnis, reagiert jedoch selbst **nicht** auf eine fehlgeschlagene Prüfung und gibt auch selbständig keine Fehlermeldungen aus.

Ein Plugin Autor müsste in diesem Fall also selbst auf einen Fehler reagieren, indem er beispielsweise eine eigene Fehlermeldung ausgibt. Ein mögliches Codebeispiel hierzu sieht wie folgt aus:

```
if ( $Core->get(POST, 'submit' true) ) {                                    // Formular wurde abgeschickt

    if ( !$Core->cfprotect_check(true) ) {                                  // Reagiere selbst auf einen Fehler des CFProtect Systems
        $Core->showmessage('', 'Hier ist was passiert!', RED, true);        // Eigene Fehlermeldung ausgeben
    } else {
        // Kein Problem mit dem abgeschickten Formular, also eigenen Code ausführen
    }

}
```

Plugin Installation, Deinstallation, Update & Versionscheck

Sämtliche Routinen zur Installation, Deinstallation, Aktualisierung und Versionscheck eines Plugins sind in der Datei **package_setup.php** im Ordner Ihres Plugins zu finden. Der Plugin Developer fügt beim Anlegen eines Projektes dort genaue Kommentare hinzu, die Ihnen den Umgang mit diesen Funktionen erleichtern.

Sie können mit dem Plugin Developer nach der Entwicklung Ihres Plugins auch automatisch eine Setup-Syntax für Ihre Hookpoints generieren lassen. Den erzeugten Code können Sie dann einfach in die Installationsroutine Ihres Plugins hineinkopieren.

Schauen Sie sich unser Demo-Plugin an!

In der CBACK Community finden Sie im Bereich der Plugin-Downloads für das CF4 das Plugin mit dem Namen „CBACK Topic Tags“. Dieses Plugin fügt automatische Such-Tags unter jedes Topic an, wurde jedoch so programmiert und dokumentiert, dass es die Funktionsweisen eines Plugin-Templates, der Nutzung von Hooks, der Nutzung von Installation, Deinstallation & Versionscheck sowie die Nutzung eigener Sprachdateien genauestens zeigt.

Laden Sie sich dieses Plugin herunter und schauen Sie sich gerne im Code des Plugins um. Wir denken, dass mit diesem praktischen Beispiel ein Einstieg in die Welt der Pluginentwicklung des CF4 sehr leicht möglich ist.

Ein Plugin für die Veröffentlichung vorbereiten

Wenn Sie Ihr Plugin veröffentlichen möchten, dann können Sie den **INHALT** Ihres Pluginordners mit einem Packprogramm Ihrer Wahl (z.B. 7Zip unter Windows oder Keka unter Mac) in ein **.tar.gz** Archiv packen. Somit ist Ihr fertiges Plugin als automatisch installierbares Paket im CF4 nutzbar.

Wichtig zu beachten ist hierbei folgendes:

- Bitte stellen Sie sicher, dass Sie die Installations- und Deinstallations-Routine in der Datei package_setup.php fertiggestellt haben! Alle zusätzlichen DB Felder, die Sie installieren sollten beim deinstallieren Ihres Plugins wieder entfernt werden. Angelegte Hookpoints Ihres Plugins werden automatisch entfernt.

- Bitte packen Sie nicht einfach Ihren Plugin-Ordner in das Archiv, sondern lediglich den INHALT des Ordners.

- Bitte nennen Sie die .tar.gz Datei genau so, wie Ihr Plugin-Ordner heißt. Wenn Ihr Pluginordner also z.B. cback_topic_tags heißt, dann nennen Sie das Pluginpaket bitte cback_topic_tags.tar.gz

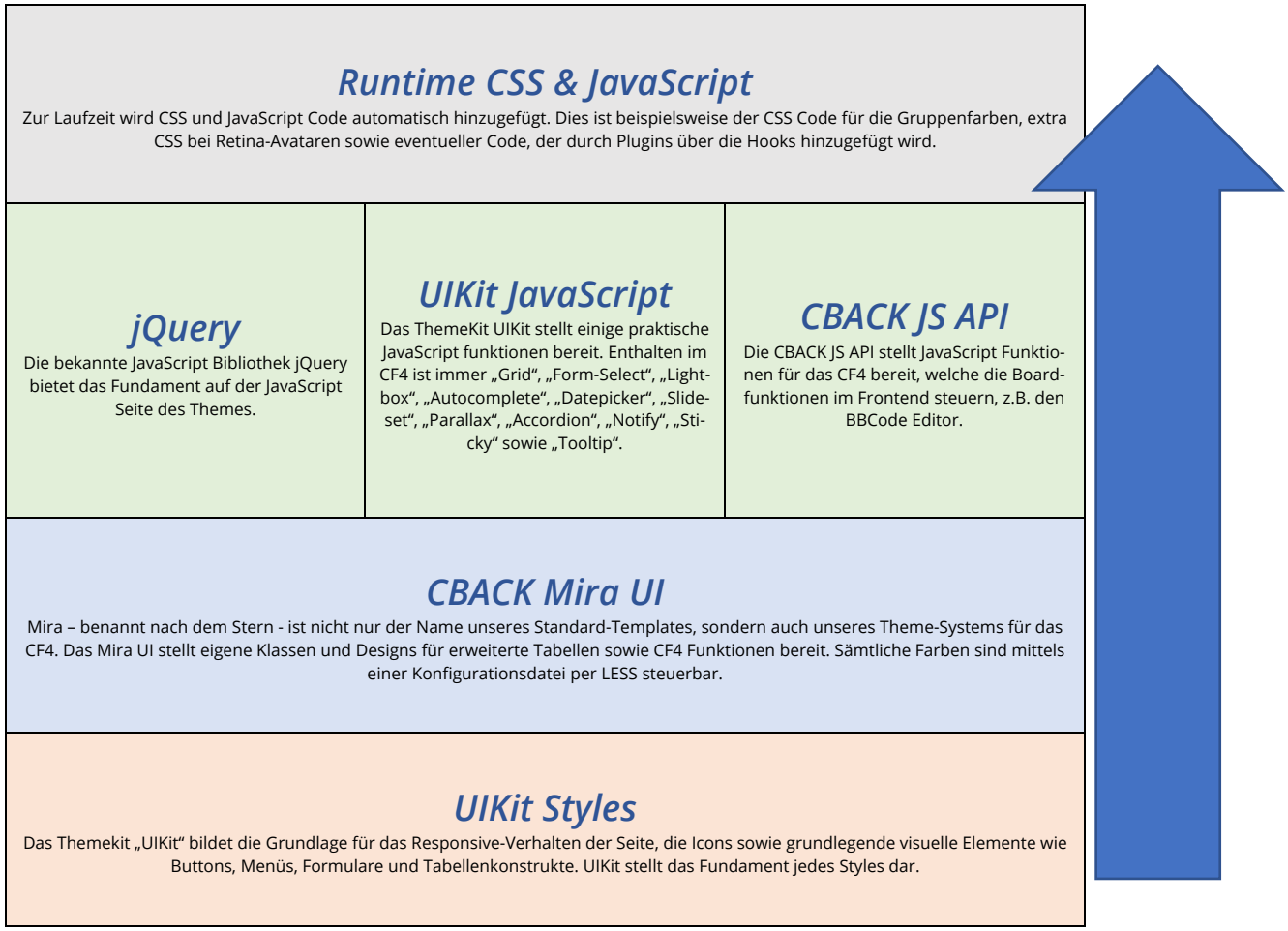
- Ein installiertes Plugin legt eine Datei Namens „installed.txt“ in den Pluginordner. Bitte stellen Sie sicher, dass Sie Ihr Pluginpaket **NICHT** mit dieser Datei packen, da sonst das CF4 fälschlicherweise davon ausgeht, dass das Plugin bereits installiert ist.

Styles & Templates entwickeln

Ein neues Stylepaket erstellen

Das Standard Template des CF4 „MIRA“ ist ein guter Ausgangspunkt, um ein neues Templatepaket zu erstellen. Dank vieler LESS Konfigurationsvariablen und einem geordneten HTML5 / CSS Code ist es möglich auf Basis von „MIRA“ sehr vielfältige eigene Styles zu entwickeln.

Der Aufbau eines CF4 Templates
Um den Aufbau eines CF4 Templates besser zu verstehen hier eine kurze Übersicht, wie das Stylesystem aufgebaut ist:



„Mira“-Style duplizieren

Bevor Sie mit dem Ändern des Templates anpassen empfiehlt es sich, dass Sie zunächst ein eigenes Stylepaket anlegen, mit dem Sie dann frei arbeiten können. Duplizieren Sie hierzu den Ordner „mira“ im Ordner „templates“ und nennen Sie ihn so, wie Ihr Style später heißen soll. Beispielsweise „mystyle“. Bitte verwenden Sie für den Ordner nur Kleinbuchstaben und Zahlen.

Als nächstes müssen Sie das Style im System manuell anmelden. Dies ist nur für die Style-Entwicklung erforderlich: Wenn Sie später Ihren Style als Stylepaket verpacken wird sich Ihr neues Template bei jedem Anwender automatisch installieren. Gehen Sie in Ihre Datenbank (z.B. mit einem Tool wie phpMyAdmin oder dem MySQLDumper) und führen Sie folgenden Query aus, um Ihr neues Style im System anzumelden:

```
INSERT INTO `IhrPrefix_themes` (`name`, `pathname`) VALUES ('Ihr Stylename', 'styleOrdnername');
```

Bitte ersetzen Sie “IhrPrefix” mit ihrem Tabellen-Prefix. Der Wert „styleOrdnername“ wäre in diesem Fall der Ordnername Ihres neuen Styles, den Sie zuvor verwendet haben. Bei dem Wert „Ihr Stylename“ kann im Klartext der Name des Styles eingefügt werden, wie er später dem Anwender angezeigt werden soll.

Der neue Style ist nun im System angemeldet und könnte im Profil ausgewählt werden.

Als nächstes passen Sie bitte in Ihrem duplizierten Styleordner die Datei „**package_info.php**“ an. Tragen Sie hier bitte die entsprechenden Informationen für Ihr Stylepaket ein, die später im ACP sichtbar sind:

```php
<?php
/**
* package_info.php
*
* Package info for the CF4 default template "MIRA"
*
* @author Christian Knerr (cback)
* @package CBACK_Forum
* @version 4.7.0
* @license CF4 EULA
* @copyright (c) CBACK Software - www.cback.net
*/

if ( !defined('CBACK') )
{
    exit;
}

$style_name     = 'Mira';                   // Name of the Template in Listing
$style_w3c      = 'HTML5 / CSS3';           // Code Standard
$style_author   = 'Christian Knerr';        // Who created this Template?
$style_web      = 'https://www.cback.de';   // Author Homepage
$style_license  = 'CBACK Forum EULA';       // Template License
$style_software = 'CF4';                    // Product ID
$style_madefor  = '4.7.0';                  // Style Made For CF4 Version
```

Dies wären im Einzelnen:

- $style_name
  Klartext-Name Ihres Styles, wie er später bei der Installation Ihres Stylepaketes in der Datenbank eingetragen wird. Dies ist normalerweise der Name, den Sie zuvor beim MySQL Query unter „Ihr Stylename“ verwendet haben.

- $style_w3c
  Hier können Sie – falls abweichend – den Standard Ihres Templates eintragen. Normalerweise ist der Standard beim CF4 HTML5 Code sowie CSS3.

- $style_author
  Ihr Name als Style-Designer. Wird im ACP bei den Style-Informationen angezeigt.

- $style_web
  Webadresse des Style-Autors (falls vorhanden)

- $style_license
  Die Lizenz Ihres Styles, z.B. „GPL“ oder „Apache“

- $style_software
  ACHTUNG: Diesen Wert bitte nicht ändern! Andere Werte als CF4 sorgen dafür, dass sich das Stylepaket nicht in einem CF4 Forum installieren lässt, da es als nicht kompatibel gewertet wird!

- $style_madefor
  Für welche Version des CBACK Forums wurde das Stylepaket erstellt? Dieser Parameter ist seit CF4 Version 4.7.0 verfügbar und im ACP neben dem Style sichtbar. So ist es bei Software Updates für die Anwender leichter ersichtlich, ob sie einen bestimmten Style noch aktualisieren müssen oder nicht bzw. besser nachvollziehbar, ob alle Styles ebenfalls auf der neuesten Version verfügbar sind.

Das war es an Vorbereitung! Sie können nun Ihren neuen Style in Ihrem Forum auswählen und mit der Arbeit am Template beginnen. Sofern Sie zum erzeugen einer CSS Datei aus den LESS Dateien noch einen Compiler/Interpreter brauchen, schauen Sie am besten unter http://lesscss.org/usage/ - guis-for-less vorbei.

Umgang mit LESS Dateien und UIKit Themes

Im Ordner „**less**“ Ihres Templates finden Sie das gesamte CSS Konstrukt für das CBACK Forum. Mittels dieser Dateien können Sie im Grunde die gesamte Darstellung des Styles und die Farben beeinflussen.

Sie finden folgendes in den Unterschiedlichen Dateien:

- **global.less**
  in dieser Datei werden alle benötigten Stylekomponenten miteinander verknüpft. Dies ist auch die einzige Datei, die Sie mittels LESS Compiler erzeugen müssen. Alle anderen .less Dateien werden automatisch hinzugefügt.

- **miraUI.less**
  in dieser Datei befindet sich die Konfiguration für das UIKit Layoutgitter, welches die Farben und Größen von visuellen Elementen wie Buttons, Hinweisboxen, etc. bestimmt. Sie können sich eine solche Konfigurationsdatei sehr einfach mit dem UIKit Customizer unter https://getui-kit.com/v2/docs/customizer.html zusammenbauen. Laden Sie sich die fertige Datei herunter und kopieren Sie deren Inhalt in die Datei „**miraUI.less**“ Ihres Styles. Bitte beachten Sie auch die Hinweisbox unter dieser Dateiliste!

- **config.less**
  Diese Datei ist sozusagen das „Herz“ des CF4 Layouts und beinhaltet alle Konfigurationsvariablen für das Style und alle CF4 Funktionen selbst. Beispielsweise die Kopf-Menüleiste, das Header-Hintergrundbild, die genutzte Systemschriftart, Topic-Farben, etc. – Das Ändern der Konfigurationsvariablen in dieser Datei kann das farbliche Aussehen der gesamten Seite verändern ohne, dass Sie auch nur einen CSS Code ändern müssen. Wenn Sie beispielsweise den Farbwert von „@cf_navbar_bg_color“ ändern, dann wird die im Standard blaufarbige Kopfzeile inklusive aller Hover-Farben verändert. Es ist so möglich mit kleinsten Anpassungen sehr große Effekte im Template zu erzielen.

- **style.less**
  In dieser Datei befinden sich alle strukturellen Layoutinformationen für das CBACK Forum Template. Einschließlich der Darstellung von BBCodes, Tabellen, dem Foren-Index, etc.

- **responsive.less**
  In dieser Datei befinden sich alle zusätzlichen Styles für die verschiedenen Viewports der Seite. Es gibt für jeden Viewport einen eigenen Bereich, in dem ggf. eigene Styleabstufungen für Mobilgeräte eingetragen werden können. Zusätzlich finden Sie hier auch die CSS-Regeln für hochauflösende Displays (HiDPI / Retina).

> **ACHTUNG:**
> Das Standard-Template „Mira“ des CF4 basiert auf dem UIKit Layoutgitter „gradient“. Sollten Sie als Basis für Ihr Template eine Variante wie „almost-flat“ oder „default“ bevorzugen, dann ändern Sie bitte in der Datei „**global.less**“ alle Vorkommen von „gradient“ in das entsprechende Layoutgitter „almost-flat“ bzw. „default“ und verwenden Sie als Konfigurationsdatei (**miraUI.less**) die entsprechende Konfiguration. Konfigurationen für das UIKit Layoutgitter können Sie z.B. mit dem Customizer unter https://getuikit.com/v2/docs/customizer.html generieren.

> **HINWEIS:**
> Bitte beachten Sie, dass das Style des CF4 selbst eine fertig erzeugte CSS Datei aus dem Ordner **css/** des Styles bezieht (**global.min.css**). Sie müssen diesen **css/** Ordner Ihres Styles also als Ziel für Ihre aus den .less Dateien interpretierten CSS Code verwenden. Ansonsten sind Änderungen in .less Dateien nicht direkt auf der Webseite sichtbar!

Umgang mit den Responsive-Viewports

Das CF4 stellt 4 verschiedene Stufen zur Darstellung der Hauptseite / des Viewports bereit:

- **XL**
  Große Monitore / Fernseher

- **L**
  Reguläre Monitore (meist von Laptops / Netbooks) oder größere Tablet-Bildschirme.

- **M**
  Meistens Tablets, die eine kleinere Bildschirmauflösung besitzen. Bei Touch wird u.U. bereits hier das verkleinerte Menüband aktiv. Dieses ist in der Datei header.htm zu finden.

- **S**
  Smartphone-Ansicht. Hierbei wird auch das Responsive-Menü / Touch Menü aktiviert. Dieses ist in der Datei header.htm zu finden.

Ziehen Sie Ihren Browser Testweise kleiner, um die Auswirkungen verschiedener Viewports auf die Anordnung von Seitenelementen zu betrachten. Bitte beachten Sie, dass für die Darstellung von Responsive-Elementen, Grid-Layouts, etc. bereits vorhandene Systemfunktionen der UIKit Basis nutzbar sind. Schauen Sie sich die UIKit Dokumentation unter https://getuikit.com/v2/docs/core.html an, um alles über die möglichen Template-CSS Befehle zu erfahren.

> **HINWEIS:**
> Wenn Sie in der Datei „**includes/config.php**“ die Konstante „DEBUG“ auf „true“ setzen, dann wird Ihnen unterhalb Ihrer CF4 Seite nicht nur alles Technische zur PHP Umgebung angezeigt, sondern auch der aktuelle Template-Viewport. So ist es möglich, dass Sie verschiedene Varianten der Seitendarstellung durch zusammenziehen Ihres Browsers direkt testen können und zusätzlich am unteren Seitenrand immer sehen, welcher Viewport gerade aktiv ist, sodass Sie die Darstellung der einzelnen Viewports sehr einfach beeinflussen / nachbessern können und nicht lange suchen müssen, welche Responsive-Regel gerade aktiv ist.

HTML Code bzw. Layout beeinflussen

Im Ordner Ihres Styles finden Sie verschiedene .htm Dateien, die für die gleichnamigen Bereiche im Forum zuständig sind. Beispielsweise „**topic.htm**“ für die Topicansicht, „**forumview.htm**“ für die Forenansicht, „**header.htm**“ für den Seitenkopf, etc. – Sie können diese HTML Dateien nach Belieben anpassen und verändern. Als Variablen / Schleifen / Funktionen sind alle Funktionen der Template-Engine Smarty nutzbar. Eine Dokumentation zu Smarty finden Sie unter: http://www.smarty.net/.

Bitte beachten Sie, falls Sie umgeschriebene Links benötigen, auch dieses Handbuch im Bereich der $SEO Klasse. Es gibt eine Template-Funktion {rewrite ...} zum direkten Umschreiben von Links im CF4 direkt im Template ohne Verändern des PHP Codes!

Hinweis zu eigenen JS Bibliotheken oder Zusatzdateien im Style

Bitte packen Sie alle zusätzlichen Dateien, die Ihr Style benötigt, direkt in Ihren Styleordner, sodass ein Anwender später nicht manuell Dateien in seinem CF4 System umkopieren muss und alle Funktionen immer direkt im Stylepaket vorliegen (und sich dieses entsprechend vollautomatisch installieren lässt).

Auch wenn wir im CF4 um Speicherplatz zu sparen beispielsweise jQuery aus unserem „**shared/**“ Ordner im Hauptverzeichnis des CF4 beziehen legen Sie eventuell für Ihr Style benötigte Zusatzscripte in Ihrem Styleordner ab. Sie können diese Komponten in der Datei header.htm bzw. footer.htm dann verlinken. Sie dürfen beliebige Unterordner in Ihrem Style erzeugen, wenn Sie diese benötigen. Bitte lassen Sie lediglich die vorhandenen Unterordner („**images**“, „**css**“, „**less**“) des Styles unverändert und löschen Sie diese nicht. Natürlich können Sie jedoch auch unter „**images**“ weitere eigene Unterordner anlegen, falls benötigt.

Ein Template für die Veröffentlichung vorbereiten

Wenn Sie Ihr Template fertiggestellt haben, dann machen Sie bitte noch einen Screenshot Ihres Styles für die ACP-Übersichtsseite. Öffnen Sie dann in Ihrem Template-Ordner die Datei „**template_preview.png**“ im Unterordner „**images**“ und fügen Sie dort Ihren Screenshot ein. Bitte behalten Sie die Breite der Bilddatei bei, um eventuelle Fehldarstellungen im ACP zu vermeiden. Die Höhe der Bilddatei kann nach Bedarf erweitert oder reduziert werden, damit ein Screenshot Ihres gesamten Styles darin hineinpasst.

Nun ist Ihr Template bereit für die Veröffentlichung (die nötigen Paket-Infos haben Sie ja bereits beim Duplizieren des mira/ Ordners eingetragen). Packen Sie den Inhalt (!) Ihres Templateordners mit einem Packprogramm als .tar.gz Paket (z.B: mit 7Zip unter Windows oder Keka unter Mac). Dieses Paket kann dann in jedem CF4 Forum automatisch installiert werden. Bitte nennen Sie die erzeugte .tar.gz Datei so, wie der Ordner Ihres Styles heißt.

Beispiel:
Der Ordner Ihres Styles heißt „**meinstyle**“. Nennen Sie die erzeugte .tar.gz Datei also bitte „**meinstyle.tar.gz**“.

Sprachdateien erstellen

Ein neues Sprachpaket anlegen

Um ein neues Sprachpaket anzulegen empfiehlt es sich, eines der vorhandenen CF4 Sprachpakete zu duplizieren. Nehmen Sie – ausgehend von Ihrer Bevorzugten Sprache – also den Ordner „**de**“ oder „**en**“ im Ordner „**lang**“ und duplizieren Sie diesen für Ihr neues Sprachpaket.

Diesen Ordner nennen Sie mit dem Kürzel der neuen Sprache, die Sie übersetzen möchten, beispielsweise „**fr**“ für französisch, „**it**“ für italienisch, etc.

Als nächstes bearbeiten Sie bitte die Datei „**package_info.php**“ in Ihrem neu erzeugten Sprachordner.

```php
<?php
/**
 * package_info.php
 *
 * @author Christian Knerr (cback)
 * @package CBACK_Forum
 * @version 4.0.0
 * @license CF4 EULA
 * @copyright (c) CBACK Software - www.cback.net
 */

if ( !defined('CBACK') ) { exit; }

$lang_name      = 'Deutsch';
$lang_author    = 'Christian Knerr';
$lang_web       = 'https://www.cback.de';
$lang_version   = '4.0.0';
$lang_license   = 'CBACK Forum EULA';
$lang_software  = 'CF4';
```

Die Werte sind im Einzelnen:

- $lang_name
  Vollständig ausgeschriebener Name Ihres Sprachpaketes, wie er dem Anwender später im Forum angezeigt wird.

- $lang_author
  Ihr Name / Name des Übersetzers

- $lang_web
  Die Webadresse der Autor-Webseite (falls vorhanden)

- $lang_version
  Die Version Ihres Sprachpaketes. Dies ist unabhängig vom CF4 selbst.

- $lang_license
  Die Lizenz für Ihr Sprachpaket, z.B. „GPL“ oder „Apache“.

- $lang_software
  Bitte lassen Sie dieses Feld unverändert, da sich Sprachpakete, die nicht als CF4 gekennzeichnet sind, nicht im Forum installieren lassen!

Nun müssen Sie das Sprachpaket erstmalig in Ihrem Forum anmelden. Dieser Schritt ist nur jetzt notwendig, wo das Paket noch nicht als .tar.gz vorliegt. Beim Endanwender wird sich das fertige Sprachpaket dann automatisch installieren.

Gehen Sie in Ihre Datenbank (z.B. mit einem Tool wie phpMyAdmin oder dem MySQLDumper) und führen Sie folgenden Query aus, um Ihr neues Sprachpaket im System anzumelden:

```
INSERT INTO `IhrPrefix_languages` (`name`, `pathname`) VALUES ('Sprachname', 'Ordnername');
```

Bitte ersetzen Sie “IhrPrefix” mit ihrem Tabellen-Prefix. Der Wert „Ordnername“ wäre in diesem Fall der Ordnername Ihres neuen Sprachpaketes, z.B. „**fr**“, den Sie zuvor verwendet haben. Bei dem Wert „Sprachname“ kann im Klartext der Name des Paketes eingefügt werden, wie er später dem Anwender angezeigt werden soll. Also beispielsweise „Français“.

Der neue Sprache ist nun im System angemeldet und kann im Profil ausgewählt werden.

Tipp: CF3 Sprachpakete auf CF4 upgraden

Um unseren Übersetzern die Arbeit mit bereits für das CF3 erstellten Paketen einfacher zu machen haben wir bei der Entwicklung des CF4 bewusst die grundlegende Struktur und der Aufbau der Sprachdateien beibehalten. Sofern Sie eine vorhandene Übersetzung auf das CF4 aktualisieren möchten empfehlen wir folgendes Vorgehen:

- Laden Sie sich das CF3 sowie das CF4 Paket aus dem Kundenmanager herunter.

- Nutzen Sie ein Vergleichsprogramm (Beispielsweise „Beyond Compare“), um jede einzelne Sprachdatei von Beiden Paketen miteinander abzugleichen. So finden Sie heraus, welche Sprach-Strings vom CF3 auf das CF4

verändert, hinzugefügt oder gelöscht wurden.

- Notieren Sie sich die Sprach-Strings und passen Sie diese entsprechend in Ihrer bereits vorhandenen CF3 Übersetzung an.

- Entfernen Sie den schließenden PHP-Tag (?>) am Ende jeder Sprachdatei, um das Paket auf den CF4 Codestandard anzupassen. Ändern Sie auch ggf. den Header-Kommentar in jeder Datei.

- Nutzen Sie als package_info.php Datei bitte eine aktuelle Datei aus dem CF4 Sprachpaket und passen Sie diese an.

Mit diesem Vorgehen ist es leichter, bereits übersetzte Pakete vom CF3 auf das CF4 zu aktualisieren, ohne, dass identische Sprach-Strings aus dem CF3, die in Beiden Systemen gleich geblieben sind (z.B.: „Neues Topic schreiben“ – „Antworten“ – „Zitat“, „Kontrollzentrum“) noch einmal zu übersetzen.

> **ACHTUNG:**
> Bitte vergleichen Sie nicht nur die Array-Keys sondern immer die gesamte Dateizeile! Viele Übersetzungen haben sich aufgrund neuer oder veränderter Funktionen im CF4 inhaltlich geändert, haben jedoch noch den selben Variablennamen.

Der richtige Zeichensatz zum Übersetzen

Nicht jeder Editor ist für die Übersetzung von Sprachpaketen geeignet. Bitte verwenden Sie ausschließlich einen geeigneten Code-Editor (z.B. phpStorm, Notepad++, Coda, TextWrangler, ...) und stellen Sie diesen auf den Zeichensatz UTF-8 ohne BOM. (Meist dann lediglich als UTF-8 bezeichnet).

Andere Zeichencodierungen in der Datei oder gar die Editierung der PHP Datei mit einem Rich-Text-Editor (z.B.: Editor, WordPad, Word, ...) kann Fehler in Ihrem Forum verursachen oder gar die Funktion Ihres Forums vollständig blockieren. („Weiße Seite“).

Sprachpaket zur Veröffentlichung vorbereiten

Wie auch bei einem Plugin oder Template müssen Sie einfach den Inhalt Ihres Sprachordners in ein .tar.gz Archiv packen, beispielsweise mit dem Packer „7Zip“ unter Windows oder „Keka“ unter Mac. Bitte komprimieren Sie den **INHALT** des Ordners, nicht den Ordner selbst. Benennen Sie die erzeugte tar.gz Datei dann so, wie Ihr Sprachordner heißt. Beispielsweise würde der Sprachordner „**fr**“ dann als „**fr.tar.gz**“ erscheinen. Diese .tar.gz Datei kann dann in jedem CF4 Forum automatisch als Sprachpaket installiert werden.

Zusätzliche Informationen

Als freigeschalteter Kunde mit einer Lizenz des CF4 haben Sie Zugang zum Forum „Developer Talk“ in der CBACK Community. Dort können Sie nicht nur weitere Informationen zur Plugin-Entwicklung finden, sondern sich auch jederzeit mit anderen Pluginentwicklern austauschen.