Erstellen eines Softwarepakets für den i-MSCP Software Installers

Diese Anleitung wird anhand eines existierenden Paketes (WordPress 3.3.1 DE) erstellt! Jeder der ein Paket selber erstellt, sollte sich vorher darüber informieren, ob die Webapplikation unter GNU GPL steht.

Im offiziellen Softwaredepot werde ausschließlich nur solche Pakete angeboten! Siehe dazu auch: Anforderungen an selbsterstellte Pakete

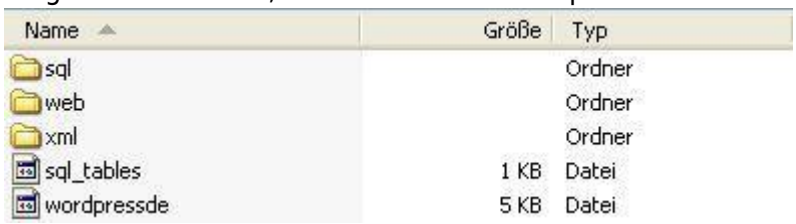
Des Weiteren sollte man das Paket später auf einem Linux-System packen!

Dinge auf die man dringend achten muss:

- Verzeichnisstruktur
- Dateibezeichnungen
- Inhalt des Installationsscripts (Paket Installations Skript (default))
- SQL-DUMP (SQL Dump erstellen)
- Inhalt der sql_tables
- Inhalt der install.xml und uninstall.xml (Paket XML Dateien)
- Packen des Paketes
- Paket Kurzanleitung

1. Verzeichnisstruktur

Der Inhalt des Paketes muss eine bestimmte Verzeichnisstruktur besitzen! Dieser muss unbedingt eingehalten werden, da dies auch beim Import des Installers überprüft wird.



Wie man sieht, benötigt man folgende Verzeichnisse:

- sql
- web
- xml

Verzeichnis "sql"

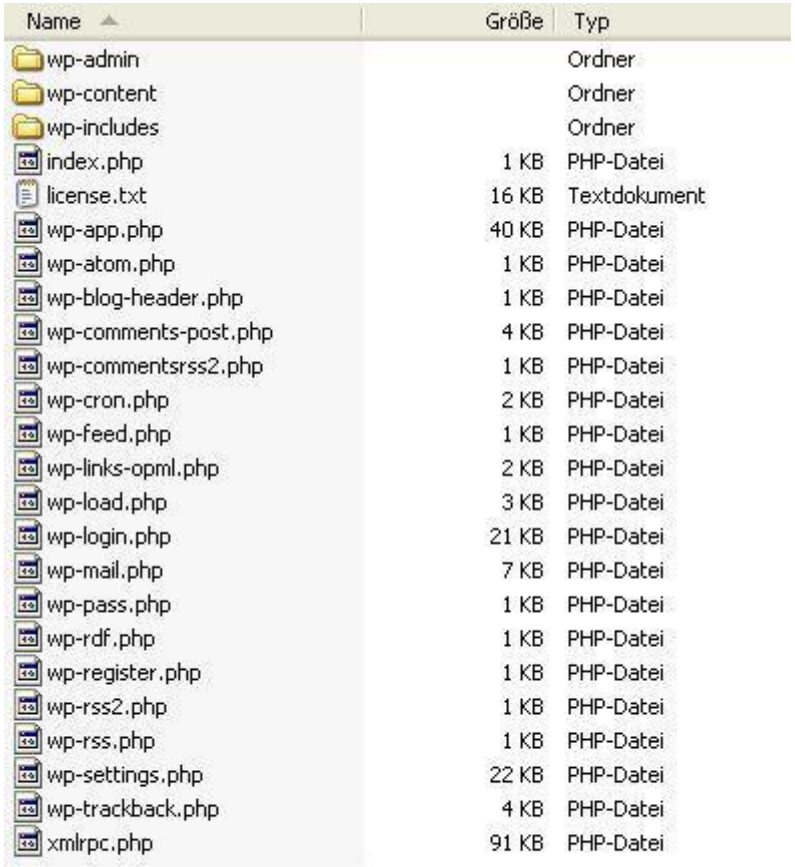
Unter dem Verzeichnis “sql” wird der SQL-Dump hinterlegt, der benötigt wird, wenn eine Webapplikation mit einer SQL-Datenbank arbeitet.

Verzeichnis "web"

Unter “web” werden von der Webapplikation die benötigten Dateien hinterlegt. Dabei ist darauf zu achten, dass man kein weiteres Unterverzeichnis mit den erforderlichen Dateien erstellt wird. Hier 2 Beispiele wie der Inhalt des Verzeichnisses “web” aussehen muß, und wie nicht:

Name	Größe	Typ
wordpress		Ordner

FALSCH



Name	Größe	Typ
wp-admin		Ordner
wp-content		Ordner
wp-includes		Ordner
index.php	1 KB	PHP-Datei
license.txt	16 KB	Textdokument
wp-app.php	40 KB	PHP-Datei
wp-atom.php	1 KB	PHP-Datei
wp-blog-header.php	1 KB	PHP-Datei
wp-comments-post.php	4 KB	PHP-Datei
wp-commentsrss2.php	1 KB	PHP-Datei
wp-cron.php	2 KB	PHP-Datei
wp-feed.php	1 KB	PHP-Datei
wp-links-opml.php	2 KB	PHP-Datei
wp-load.php	3 KB	PHP-Datei
wp-login.php	21 KB	PHP-Datei
wp-mail.php	7 KB	PHP-Datei
wp-pass.php	1 KB	PHP-Datei
wp-rdf.php	1 KB	PHP-Datei
wp-register.php	1 KB	PHP-Datei
wp-rss2.php	1 KB	PHP-Datei
wp-rss.php	1 KB	PHP-Datei
wp-settings.php	22 KB	PHP-Datei
wp-trackback.php	4 KB	PHP-Datei
xmlrpc.php	91 KB	PHP-Datei

RICHTIG

Wenn ihr die Dateien der Webapplikation in das Verzeichnis kopiert, achtet auch folgende Dinge:

- Die Konfigurationsdatei(en) welche später vom Paket Installations Skript erstellt werden müssen gelöscht werden
- Der Inhalt von Verzeichnissen, wo die Webapplikation Cache-Dateien oder TEMP-Dateien ablegt muss geleert werden

Verzeichnis "xml"

Das Verzeichnis “xml” beinhaltet nur die 2 XML-Dateien:

- install.xml
- uninstall.xml

Wie der Inhalt der XML-Dateien auszusehen hat kann man unter dem folgenden Link lesen: Paket XML Dateien

2. Dateibezeichnungen

Wie schon unter Punkt 1 erwähnt, müssen bestimmte Dateien vorhanden sein, und auch richtig geschrieben werden. Bitte keine Sonderzeichen und Umlaute! Bindestriche ("-") sollten auch vermieden werden. Nutzt dazu den Unterstrich ("_"). Im Verzeichnis "sql" soll immer der Dump für die entsprechende Webapplikation vorhanden sein! Der Dump muss immer den Namen:

sql.sql

besitzen (er darf zwar auch anders lauten, aber dies muss man dann im Installskript anpassen. Hat die Applikation keine Datenbank, braucht man auch keine Datei dort rein legen. Das Verzeichnis muss aber bestehen!

Im Verzeichnis "xml" müssen 2 Dateien vorhanden sein! Die Dateien müssen

install.xml

und

uninstall.xml

benannt werden.

Im Root-Verzeichnis des Paketes benötigt man 2 Dateien! Einmal liegt dort eine Paket Installations Skript (variable zu benennen, muss aber in der install.xml beschrieben werden) und eine Text-Datei ("sql_tables"). Der Name "sql_tables" ist auch Pflicht! Hierbei kann auch eine leere Datei existieren.

Das Paket Installations Skript wird auf jeden Fall benötigt und soll **nicht** ausführbar sein.

3. Inhalt des Paket Installations Skript

Die Datei wird im Prinzip dafür genutzt um bei der Installation verschiedene Aktionen auszuführen, die man mit PHP nicht so einfach machen kann. Sollte eine Webapplikation keine weiteren Aktionen benötigen kann man diese Vorlage nutzen: Paket Installations Skript (default). Durch das Paket Installations Skript erhält man die Möglichkeit auf alle Funktionen der **imscp_common_code.pl** zugreifen zu können.

Variablen

In dem Paket Installations Skript stehen einige Variablen zur Verfügung:

- **$sw_software_db** = Integer (0/1: Datenbank benötigt)
- **$sw_software_prefix** = Präfix für Tabellen

- **$sw_database** = Datenbankname
- **$sw_database_user** = Datenbank Benutzername
- **$sw_database_pass_clear** = Datenbank Passwort im Klartext
- **$sw_database_pass_md5** = Datenbank Passwort MD5 verschlüsselt
- **$sw_install_username** = Username für späteren Login
- **$sw_install_pass_clear** = Passwort für späteren Login im Klartext
- **$sw_install_pass_md5** = Passwort für späteren Login MD5 verschlüsselt
- **$sw_install_email** = Emailadresse für den späteren Login
- **$domain_name** = Domain Name (dort wo die Applikation installiert wird)
- **$sw_dest_path** = Pfad wo die Applikation installiert werden soll (absoluter Pfad: /var/www/virtual.....)
- **$sw_path** = Pfad ab “htdocs” (/htdocs/abc/def)
- **$url_path** = Hier wird ein Pfad erzeugt ohne htdocs und was dahinter liegt (z.B.: http://meineDomain.tld/abc/def)
- **$db_tables** = Datei mit dem Inhalt der Tabellen die einen Präfix brauchen

Weiterhin kann man auch eigene Variablen definieren. Diese sollten aber erst nach dem folgenden Abschnitt ersstelle werden:

```
#
# This is the place for dynamic vars
#
```

Jetzt stellen sich bestimmt einige die Frage, wie man das nutzt. Ich versuche es mal anhand von Beispielen des Paketes WordPress 2.8.4 zu erläutern.... WordPress benötigt unter anderem eine Konfigurationsdatei um die Connection zur Datenbank zu bekommen. Diese Datei nennt sich “wp-config.php” und liegt im Root- Verzeichnis der Webapplikation:

/var/www/virtual/mydomain.tld/htdocs/

Nun muss diese Datei bei der Installation auch erstellt werden. Dazu nutzt man die Paket Installations Skript. In der Sektion “dynamic vars” erstellt man dazu einen neuen Eintrag:

my $config_file = “wp-config.php”;

Hiermit definiert man die Variable “**$config_file**” mit dem Wert “**wp-config.php**” belegt. Als nächstes muss die Konfigurationsdatei auch erstellt werden. In der Sektion “**($processing_type eq “install”)**” wird dies nun mit folgenden Eintrag erledigt:

```
my $configfile_entry = "<?php
define('DB_NAME', '$sw_database');
define('DB_USER', '$sw_database_user');
define('DB_PASSWORD', '$sw_database_pass_clear');
define('DB_HOST', 'localhost');
define('DB_CHARSET', 'utf8');
define('DB_COLLATE', '');

$securekeys

\$table_prefix = '$sw_software_prefix';
```

```
define('WPLANG', 'de_DE');

define('WP_DEBUG', false);

if ( !defined('ABSPATH') )
define('ABSPATH', dirname(__FILE__) . '/');

require_once(ABSPATH . 'wp-settings.php');
?>";
open(CONFIGFILE,'>'.$sw_dest_path.'/'.$config_file);
print CONFIGFILE "$configfile_entry";
close CONFIGFILE;
```

Folgendes passiert jetzt hier... Mit "**my $configfile_entry =**" wird eine Variable definiert, die genau mit den Daten befüllt welche später mit

```
open(CONFIGFILE,'>'.$sw_dest_path.'/'.$config_file);
print CONFIGFILE "$configfile_entry";
close CONFIGFILE;
```

in die Datei "**wp-config.php**" unter dem absoluten Pfad (**$sw_dest_path**) geschrieben wird! Hier sieht man nun das veschiedene Variablen genutzt werden (**$sw_database**, **$sw_database_user**, **$sw_database_pass_clear**, **$sw_software_prefix**, **$sw_dest_path**). In den neueren Versionen von Wordpress, werden zusätzlich 4 neue Zeilen eingefügt:

```
define('AUTH_KEY',         '5HW4~W458+Q^wM>k:ahRu4SGXa2;HZK]-
yBAK60|!LtpcndIWBkFd&p2m}ts}rQG');
define('SECURE_AUTH_KEY', 'GuGW};v5Cn2Pg-)-vwn|7kaS.su+[)M-
}|5ePz4)yzI<M8*(EFYtFXFu-[K7~@L>');
define('LOGGED_IN_KEY',   'CJ.TMOTUzsL(wz-1jn|+%3i|)IQ#`A-mUV
f1[C;PTd9HI<z1u[Tw=xH*=90-@yP');
define('NONCE_KEY',       '_5}laFW-
_[DoXDsT5uGs#%=#y;t]xY2WGE::1(M3s_v._2Miz|{}il16Ty -Q%7?');
```

Damit nicht immer die gleichen Codes in den Installationen vorliegen, kann man auch externe Quellen nutzen um solche Variablen dynamisch zu befüllen. Dazu wird in diesem Fall eine ApPI von Wordpress.org genutzt.
http://api.wordpress.org/secret-key/1.1/
Vor der Zeile "**my $configfile_entry =**"erstellt man folgende Zeile:

```
my $securekeys = `$main::cfg{'CMD_PHP'} -n ./get_secure_keys.php`;
```

Dadurch wird die externe API aufgerufen und die Variable "**$securekeys**" mit der Ausgabe der API befüllt. Die befüllte Variable "**$securekeys**" wird dann unter **my $configfile_entry =** eingetragen. Es gibt viele Möglichkeiten solche Variablen dynamisch zu erstellen. Genauere Informationen findest Du hier: Externe Skripte erstellen

Solltet Ihr mal Konfiguartionsdateien haben wo Einträge wie

```
var \$sitename = '$domain_name';
var \$dbtype = 'mysqli';
```

```
var \$host = 'localhost';
```

eingetragen werden müssen, dann achtet darauf, dass ihr Zeichen wie "**@, $, %, [**" escaped (einen \ vor das entsprechende Zeichnen, damit es seine Wertigkeit verliert). Wenn Ihr das nicht macht wird die Installation 100%ig fehlschlagen!

Sektion install ($processing_type eq "install")

In diesem Bereich wird die Webapplikation komplett installiert. Hier werden sämtliche Aktionen Eingetragen die das Skript durchführen soll um die Webapplikation lauffähig zu machen. Inklusive des angegebenen Usernamen und Passwort für den Login.
Wordpress nutzt eine Datenbank. Um die Daten aus dem SQL-Dump in die Datenbank zu bekommen wird folgende Zeile genutzt:

```
#Dump import
sys_command("mysql -u".$sw_database_user." -p".$sw_database_pass_clear."
".$sw_database." < ./sql/sql.sql");
```

Als nächstes müssen die Tabellen mit dem "Präfix" versehen werden. Bei dieser Aktion kommt nun auch die Datei "sql_tables" zum Tragen:

```
#Rename SQL-Tables
open(SQL_TABLES,'<'.$db_tables) or die("Unable to open file: ".$db_tables);
my @sql_table_data = <SQL_TABLES>;
chomp (@sql_table_data);
close(SQL_TABLES);
foreach my $sql_table_data_line (@sql_table_data) {
    $sql = "
        RENAME TABLE
            `".$sql_table_data_line."`
        TO
        `".$sw_software_prefix.$sql_table_data_line."`
        ;
    ";
    doSQL($sql);
}
```

Nutzt bitte diese Zeilen, weil damit funktioniert es auf jeden Fall.

Als letztes kommen noch ein paar Updates verschiedener Tabellen. Dabei wird unser Username, Passwort, Emailadresse, URL aktualisiert. Woher ich die Stellen kenne liegt dann wohl auf der Hand. Ich installiere jede Software. Dabei installiere ich immer mir der Emailadresse "noreply@i-mscp.net" und durchsuche den DUMP später nach diesem Wert. Genauere Informationen zum SQL-DUMP findet Ihr hier: SQL Dump erstellen

```
my $setdatetime = `date '+%Y-%m-%d %H:%M:%S'`;
my $loginpasswordhash = `$main::cfg{'CMD_PHP'} -n ./make_password.php
$sw_install_pass_clear '$sw_dest_path'`;
```

```
#Update Datebase with variables
$sql = "
    UPDATE
        `".$new_table_comments."`
    SET
        `comment_date` = '".$setdatetime."',
        `comment_date_gmt` = '".$setdatetime."'
    WHERE
        `comment_ID` = '1'
    ;
";
doSQL($sql);

$sql = "
    UPDATE
        `".$new_table_options."`
    SET
        `option_value` = 'http://".$domain_name."'
    WHERE
        `option_id` = '3'
    ;
";
doSQL($sql);

$sql = "
    UPDATE
        `".$new_table_options."`
    SET
        `option_value` = '".$domain_name."'
    WHERE
        `option_id` = '4'
    ;
";
doSQL($sql);

$sql = "
    UPDATE
        `".$new_table_options."`
    SET
        `option_value` = '".$emailadress."'
    WHERE
        `option_id` = '7'
    ;
";
doSQL($sql);

$sql = "
    UPDATE
        `".$new_table_options."`
    SET
        `option_value` = 'http://".$domain_name."'
```

```
    WHERE
        `option_id` = '39'
    ;
";
doSQL($sql);

$sql = "
    UPDATE
        `".$new_table_posts."`
    SET
        `post_date` = '".$setdatetime."',
        `post_date_gmt` = '".$setdatetime."',
        `post_modified` = '".$setdatetime."',
        `post_modified_gmt` = '".$setdatetime."',
        `guid` = 'http://".$domain_name."/?p=1'
    WHERE
        `ID` = '1'
    ;
";
doSQL($sql);

$sql = "
    UPDATE
        `".$new_table_posts."`
    SET
        `post_date` = '".$setdatetime."',
        `post_date_gmt` = '".$setdatetime."',
        `post_content` = 'Dies ist ein Beispiel einer statischen Seite. Du
kannst sie bearbeiten und beispielsweise Infos über dich oder das Weblog
eingeben, damit die Leser wissen, woher du kommst und was du machst.\n\nDu
kannst entweder beliebig viele Hauptseiten (wie diese hier) oder
Unterseiten, die sich in der Hierachiestruktur den Hauptseiten unterordnen,
anlegen. Du kannst sie auch alle innerhalb von WordPress ändern und
verwalten.\n\nAls stolzer Besitzer eines neuen WordPress-Blogs, solltest du
zur Übersichtsseite, dem <a
href=\"http://".$domain_name."/wp-admin/\">Dashboard</a> gehen, diese Seite
löschen und damit loslegen, eigene Inhalte zu erstellen. Viel Spaß!',
        `post_modified` = '".$setdatetime."',
        `post_modified_gmt` = '".$setdatetime."',
        `guid` = 'http://".$domain_name."/?p=2'
    WHERE
        `ID` = '2'
    ;
";
doSQL($sql);

$sql = "
    UPDATE
        `".$new_table_users."`
    SET
```

```
        `user_login` = '".$sw_install_username."',
        `user_pass` = '".$loginpasswordhash."',
        `user_nicename` = '".$sw_install_username."',
        `user_email` = '".$sw_install_email."',
        `user_registered` = '".$setdatetime."',
        `display_name` = '".$sw_install_username."'
    WHERE
        `user_login` = 'admin'
    ;
";
doSQL($sql);
```

Auch hier findet man wieder 2 Einträge wo Variablen dynamisch erstellt werden:

```
my $setdatetime = `date '+%Y-%m-%d %H:%M:%S'`;
my $loginpasswordhash = `$main::cfg{'CMD_PHP'} -n ./make_password.php
$sw_install_pass_clear '$sw_dest_path'`;
```

Wichtig ist natürlich das man die neuen Tabellen updated!! Der Präfix ist ja mittlerweile an die Tabellen dran gehangen worden. Man sieht auch, daß dort wieder Variablen genutzt werden, die unter der Sektion “dynamische Variablen” eingetragen wurden:

```
my $table_comments = "comments";
my $new_table_comments = $sw_software_prefix.$table_comments;
my $table_options = "options";
my $new_table_options = $sw_software_prefix.$table_options;
my $table_posts = "posts";
my $new_table_posts = $sw_software_prefix.$table_posts;
my $table_users = "users";
my $new_table_users = $sw_software_prefix.$table_users;
```

Damit sollte die Installation erfolgreich beendet worden sein!

Sektion uninstall ($processing_type eq "uninstall")

In diesem Bereich braucht man in der Regel nur was eintragen wenn man eine SQL-Datenbank für die Webapplikation benötigt hat. Hier ein Beispiel:

```
#Drop existing SQL-Tables
open(SQL_TABLES,'<'.$db_tables) or die("Unable to open file: ".$db_tables);
my @sql_table_data = <SQL_TABLES>;
chomp (@sql_table_data);
close(SQL_TABLES);
foreach my $sql_table_data_line (@sql_table_data) {
    $sql = "
        DROP TABLE IF EXISTS
            `".$sw_software_prefix.$sql_table_data_line."`
        ;
    ";
```

```
    doSQL($sql);
}
```

3. Testen des Paket Installations Skript

Das Paket Installations Skript kann man ganz einfach auf der Linux-Konsole auf Fehler testen.
Dazu gebt ihr einfach folgenden Befehl auf der Linux-Konsole ein:

```
perl Name_des_Installationsskripts
```

ein. Wenn keine Fehler (das heisst alle Variablen definiert sind usw.) im Paket Installations Skript sind wird folgende Ausgabe erscheinen:

```
[DEGUB] i-MSCP installer file - No Input Data available
```

Andernfalls werdet Ihr schon erkennen wo das Problem liegt.

4. SQL-DUMP

Den SQL- Dump sollte man mit phpmyadmin ziehen. Dabei ist darauf zu achten das man auch den Haken “Füge DROP TABLE / VIEW / PROCEDURE / FUNCTION hinzu” setzt!
Den Dump speichert man dann als

sql.sql

im Verzeichnis “sql” des Pakets ab.
Um flexibel zu bleiben durchsucht man nun den DUMP nach dem installierten „Präfix“. Im Falle von WordPress lautet er in der Regel “wp_”.
Dieser wird im kompletten Dump entfernt.
Aber geht den Dump in Ruhe durch und macht nicht einfach “suchen & ersetzen”.
Worauf ihr genau achten müsst findet Ihr hier: SQL Dump erstellen

5. Inhalt der Datei sql_tables

Dieses Datei liegt im Root-Verzeichnis des Paketes und beeinhaltet alle Tabellen, die später den Präfix erhalten sollen. Die Einträge bekommen keinen “Präfix” davor. Um eine Auflistung der Tabellen zu bekommen, kann man in phpmyadmin “**SHOW TABLES**” ausführen (vorher natürlich die richtige Datenbank auswählen). Diese Liste fügt man dann in eine Datei ein, die man “sql_tables” nennt. Jetzt nur noch die existierenden Präfixe entfernen (Bei WordPress “wp_”). Im Beispiel von WordPress sehen die Einträge wie folgt aus:

```
commentmeta
comments
links
options
postmeta
```

```
posts
term_relationships
term_taxonomy
terms
usermeta
users
```

Wie man sieht, wurde aus “wp_comments” jetzt “comments”.

6. Packen des Paketes

Die Pakete werden immer als “**tar.gz**” gepackt.
Andere Formate werden zur Zeit nicht unterstützt. Ich rate dazu die Pakete auf einem Linuxsystem zu packen.
Unter Windows könnte es da zu Problemen kommen.
Ich selbst erstelle immer einen Unterordner im Verzeichnis “**tmp**”. Bei WordPress z.B. “**/tmp/wordpress/**”. Alle Dateien und Verzeichnisse (Paketstruktur) kopiere ich dort rein. Nun wechselt man in das angelegte Verzeichnis (**/tmp/wordpress**).
Jetzt überprüft an ob keine Datei ausführbar ist. Die Berechtigungen sollten wie folgt aussehen:

- Dateien “**644**”
- Verzeichnisse “**755**”

Damit man nicht jedes Verzeichnis und jede Datei auf seine Richtigkeit überprüfen muss, kann man das ganz simple mit einem “**find**” durchführen lassen.

```
find ./ -type f -print0 | xargs -0 chmod 0644
find ./ -type d -print0 | xargs -0 chmod 0755
```

Jetzt wird das Paket gepackt. Wichtig ist das man das Paket eine Ebene höher erstellen lässt! Die Namen für das Paket sollten eindeutig und auch leicht zu erkennen sein (**wordpress_3_3_1_de.tar.gz**). »Apllikationsname_Version_Sprache.tar.gz« (Bitte keine Bindestriche und keine Punkte im Namen nutzen. Ersetzt diese mit einem Unterstrich “_“
Nun führt folgenden Befehl aus:

```
tar czvf ../wordpress_3_3_1_de.tar.gz *
```

Damit erstellt man nun ein Paket im Verzeichnis “**/tmp**” mit dem Namen “**wordpress_3_3_1_de.tar.gz**”. Nun nur noch testen ob die Arbeit korrekt durchgeführt wurde. Ladet das Paket als Admin oder Reseller hoch und testet es als User.
Bei Problemen mit Paketen könnt Ihr Euch jederzeit im Forum melden.