

Inhaltsverzeichnis

<u>Schnittstellen</u>	1
<u>Die Grundstruktur</u>	1
<u>Ajax Schnittstelle (Adminbereich)</u>	2
<u>Die Verzeichnisstruktur</u>	5
<u>Systemverzeichnisse</u>	5
<u>Kundenverzeichnisse</u>	6
<u>Skripte (site)</u>	6
<u>Templates (skin)</u>	6
<u>Variable Dateien</u>	7
<u>Das Basisskript</u>	8
<u>Beispiel 1</u>	8
<u>Beispiel 2</u>	8
<u>Beispiel 3</u>	8
<u>Neue Seitentypen erstellen</u>	10
<u>Seitentyp erzeugen</u>	10
<u>Verschachtelte Seitentypen</u>	11
<u>Verschachtelung</u>	11
<u>Inaktive Einträge</u>	11
<u>Templates und Skripte</u>	12
<u>Das typenspezifische Template (body.html)</u>	12
<u>Das typenspezifische Skript (index.php)</u>	12
<u>Typenspezifische Reiter</u>	13
<u>Reiter festlegen</u>	13
<u>Informationsreiter erweitern</u>	13
<u>information.html (Teil 1)</u>	14
<u>information.html (Teil 2)</u>	14
<u>Informationsreiter global erweitern</u>	16
<u>Neue Reiter erstellen</u>	16
<u>Verwendung von Eingabefeldern</u>	17
<u>Globale Reiter erstellen</u>	17
<u>Zweiten Inhaltsreiter erstellen</u>	17
<u>Editorbreite einstellen</u>	17
<u>Typenspezifische Einstellungen</u>	18
<u>Eigenschaften der aktuellen Seite</u>	18
<u>Beispiel</u>	18
<u>Eigenschaften neuer Unterseiten</u>	18
<u>Zusätzliche Aktionen</u>	19
<u>Desklets erstellen</u>	19
<u>Beispiel-Desklet: Seitenstatistik</u>	20
<u>Das Template (desktop.html)</u>	20
<u>Das Skript (desktop.php)</u>	21
<u>Desklet anzeigen</u>	23
<u>Toolbar anpassen</u>	23
<u>Neuen Menüpunkt erstellen</u>	23
<u>Neuen Menü-Unterpunkt erstellen</u>	24
<u>Trenner einfügen</u>	25
<u>Globale Seitentypen</u>	26
<u>Beispiel</u>	26

Inhaltsverzeichnis

Weiterführende Möglichkeiten	27
<u>Benutzerverwaltung um eigene Reiter erweitern</u>	27
<u>Beispiel (var/lib/rights/user_profile_tab.ini)</u>	27
<u>Beispiel (Skript: var/lib/rights/kunden_daten.php)</u>	27
<u>Beispiel (Template: var/lib/rights/t/kunden_daten.html)</u>	27
<u>Vorschau</u>	29
<u>Verwendung von Captcha</u>	30
Vorbereitungen	31
Überprüfung (im Skript über PHP)	32
Überprüfung (im Template über Smarty)	33
3. Spam-Schutz einstellbar machen	35
<u>Menüleiste anpassen</u>	36
<u>Beispiel</u>	36
<u>Crons definieren</u>	37
<u>Outputfilter erstellen</u>	38
<u>Beispiel</u>	38
<u>Eigene Smarty-Funktionen erstellen</u>	39
<u>Beispiel 1</u>	39
<u>Beispiel 2</u>	40
<u>Zusätzliche Tabellen für Extrafelder</u>	42
<u>Tabellen für Extrafelder erzeugen</u>	42
<u>Ab Version 5.0.6 kann in site/_global/admin/create_table.php ein globales Skript hinterlegt werden, dass für jeden Mandanten vor dem Mandantenskript aufgerufen wird. Die Methode muss dann allerdings site_create_table_global(Site \$site, \$params) heißen</u>	43
<u>Daten in die Tabellen speichern</u>	43
<u>Ab Version 5.0.6 kann in site/_global/admin/update.php ein globales Skript hinterlegt werden, dass für jeden Mandanten vor dem Mandantenskript aufgerufen wird. Die Methode muss dann allerdings site_update_global(Site \$site, \$params) heißen</u>	45
<u>Liveserver Desklet anpassen</u>	46
Beispiel	47
<u>Cacheverhalten</u>	48
<u>Das Cachen verhindern</u>	48
Tipps und Tricks	49
<u>JavaScript</u>	49
<u>IE IMG Node kopieren</u>	49
<u>IE Bug Flickering Hintergrundbilder</u>	49
Klassen & Funktionen	50
<u>Site</u>	50
<u>destroyIDs</u>	50
<u>Beispiel</u>	50
<u>getErrorPage</u>	50
<u>Beispiel</u>	50
<u>getLanguages</u>	51
<u>Beispiel</u>	51
<u>getMediaSite</u>	51
<u>Beispiel</u>	51
<u>getPage</u>	51
<u>Beispiel</u>	52

Inhaltsverzeichnis

Klassen & Funktionen

<u>getPageld</u>	52
<u>Beispiel</u>	52
<u>getPages</u>	52
<u>Rückgabe</u>	52
<u>query</u>	52
<u>param</u>	53
<u>Beispiel 1</u>	53
<u>Beispiel 2</u>	54
<u>getPageUrl</u>	54
<u>Beispiel</u>	54
<u>getRoot</u>	55
<u>Beispiel</u>	55
<u>Ego System</u>	56
<u>clearPageLocks</u>	56
<u>Beispiel</u>	56
<u>clearTypeCache</u>	56
<u>Beispiel</u>	56
<u>file_exists</u>	56
<u>Beispiel</u>	56
<u>getAllSites</u>	57
<u>Beispiel</u>	57
<u>isEmptyContent</u>	57
<u>Beispiel</u>	57
<u>mkdir</u>	57
<u>Beispiel</u>	57
<u>urltopage</u>	58
<u>Beispiele</u>	58
<u>Page Iterator</u>	59
<u>current</u>	59
<u>Beispiel</u>	59
<u>key</u>	59
<u>Beispiel</u>	59
<u>next</u>	60
<u>Beispiel</u>	60
<u>nextPage</u>	60
<u>Beispiel</u>	60
<u>numRecords</u>	61
<u>Beispiel</u>	61
<u>Page</u>	62
<u>addChild</u>	62
<u>Beispiel</u>	62
<u>addParent</u>	62
<u>Beispiel</u>	62
<u>cleanEmptyContent</u>	62
<u>Beispiel</u>	63
<u>copyTo</u>	63
<u>Beispiel</u>	63
<u>delete</u>	63
<u>Beispiel</u>	63
<u>delParent</u>	63
<u>Beispiel</u>	63
<u>destroy</u>	64
<u>Beispiel</u>	64
<u>getAncestors</u>	64

Inhaltsverzeichnis

Klassen & Funktionen

<u>Beispiel</u>	64
<u>getChildren</u>	65
<u>Beispiel</u>	65
<u>getDescendants</u>	65
<u>Beispiel</u>	65
<u>getKeywords</u>	66
<u>Beispiel</u>	66
<u>getLanguagePage</u>	66
<u>Beispiel</u>	66
<u>getParents</u>	66
<u>Beispiel</u>	66
<u>getPath</u>	67
<u>Beispiel</u>	67
<u>getSiblings</u>	68
<u>Zusätzliche Parameter für "param"</u>	68
<u>Beispiel</u>	68
<u>getSite</u>	68
<u>Beispiel</u>	68
<u>getUrl</u>	68
<u>Beispiel</u>	68
<u>getUser</u>	69
<u>Beispiel</u>	69
<u>hasMultiParents</u>	69
<u>Beispiel</u>	70
<u>hasRights</u>	70
<u>Beispiel</u>	70
<u>move</u>	70
<u>Beispiel</u>	70
<u>newChild</u>	70
<u>Beispiel</u>	71
<u>setRightsArray</u>	71
<u>Beispiel</u>	71
<u>setUsersArray</u>	72
<u>Beispiel</u>	72
<u>undelete</u>	73
<u>Beispiel</u>	73
<u>unlinkFrom</u>	73
<u>Beispiel</u>	73
<u>update</u>	74
<u>Beispiel</u>	74
<u>updateChildren</u>	74
<u>Beispiel</u>	74
<u>updateExtra</u>	75
<u>Beispiel</u>	75
<u>updateField</u>	75
<u>Beispiel</u>	75
<u>updateParents</u>	76
<u>Beispiel</u>	76

Beispiele & Vorlagen.....77

<u>navigation.ini</u>	77
-----------------------------	----

Inhaltsverzeichnis

<u>Erweiterter Information-Reiter</u>	78
<u>Zweiter Inhaltsreiter</u>	79
<u>Eigener Reiter</u>	80
<u>information.html</u>	81
<u>Vorlage</u>	81
<u>extra.html</u>	83
<u>Vorlage</u>	83
<u>Eigenschaften neuer Unterseiten</u>	88
<u>Vorlage</u>	88
<u>Erstellen einer Unterseite</u>	91
<u>Wiki Mandanten anlegen</u>	93
<u>Dojo Schnittstelle (Adminbereich)</u>	94
<u>API</u>	96

Schnittstellen

Durch Verwendung der Skriptsprache PHP5 ist EGOTEC® sehr flexibel. Es existieren so Schnittstellen zu vielen Skriptsprachen, Datenbanken und allen Internet Standards.

EGOTEC® ist Komponenten basiert aufgebaut und lässt sich beliebig erweitern.

Es können so Schnittstellen zu beliebigen Systemen erstellt werden. Die einfachste Version einer Schnittstelle wird mit Hilfe eines FTP Uploads und einer Textdatei realisiert, die dann von einer eigens dafür erstellten Komponente verarbeitet wird.

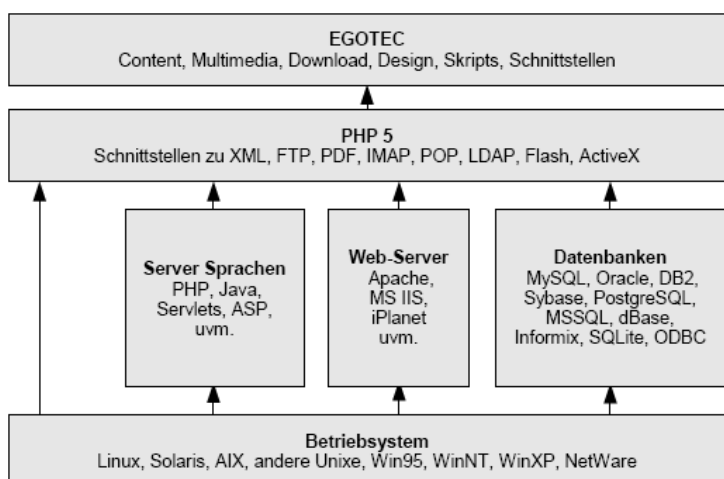
Durch die Verwendung von PHP sind aber auch höhere Schnittstellen z.B. zu Datenbanken (MySQL, MS SQL, Oracle, DB2, Sybase, PostgreSQL, dBase, Informix, InterBase, Ingres II, mSQL, ODBC, SESAM, ...) und anderen Systemen möglich (Cybercash, XML, XSLT, FTP, PDF, Verisign, Hyperwave, ICAP, IMAP, POP3, NNTP, SMTP, LDAP, Flash, Shockwave, mnoGoSearch, SSL, Corba, YAZ, YP/NIS, ...).

Auch können SOAP und XML-RPC Schnittstellen angesprochen werden. Weiterhin ist eine Schnittstelle zu Java/JavaBeans/JavaServlets möglich. Auf Windows Systemen kann zusätzlich noch COM/DCOM/ActiveX als Schnittstelle fungieren. Über einen eigens entwickelten COM/DLL Wrapper können auch beliebige DLLs eingebunden werden.

Die Einbindung können Sie selbst übernehmen. Die Schnittstellenprogrammierung ist dokumentiert und das komplette EGOTEC® liegt Ihnen bei Lizenzerwerb im Quellcode vor.

Wir übernehmen gerne die Erstellung Ihrer Schnittstellen an Hand eines Pflichtenhefts. Es wurde unter anderem eine Schnittstelle zu einer Hotelverwaltungssoftware realisiert, so dass online Buchungen vorgenommen werden. Ein weiteres Beispiel ist ein Auftrags-Tracking System mit dem der Kunde Passwort geschützt den aktuellen Stand seiner Aufträge verfolgen kann.

Die Grundstruktur



Ajax Schnittstelle (Adminbereich)

Diese Schnittstelle ist erst **ab Version 5** verfügbar.

Im Adminbereich kann die Ajax Schnittstelle verwendet werden. Als Übertragungsformatierung wird JSON verwendet.

Die Schnittstelle ist in PHP in der Datei EGOTEC/bin/json/request.php implementiert. Man kann sehr leicht per `dojo` darauf zugreifen. Es gibt eine reservierte Smarty-Variable `$ajax_url`, die die URL zur Schnittstelle mit Parametern wie `site` und `lang`, enthält.

Dem AJAX-Aufruf müssen zwei Parameter übergeben werden:

- `call_file`
Die Datei die aufgerufen werden soll (im lib-Pfad)
- `call_function`
Die Funktion die aufgerufen werden soll.

Mit dem Parameter `params` kann ein weiteres Objekt angegeben werden, das der `call_function` als assoziatives Array übergeben wird.

Beispiel auf HTML-Seite:

QuelltextJavaScript Code:

```
1. <script type="text/javascript" src="{ $local_dir }pub/dojo/dojo/dojo.js" djConfig="parseOnLoad:true,
   isDebug:true"></script>
2. <script type="text/javascript">
3. dojo.xhrPost({
4.   url: "{ /literal } { $ajax_url } { /literal }",
5.   handleAs: "json",
6.   timeout: 1000,
7.   postData: dojo.toJson({ // schreiben
8.     'call_file': 'live/check_liveserver.php',
```

```

9.     'call_function': 'check_liveserver',
10.    'params': {'AA' : 'BB'}
11.    }},
12.
13.    load: function(responseObject, ioArgs){
14.        alert (responseObject); // lesen
15.        return responseObject;
16.    },
17.
18.    error: function(response, ioArgs){
19.        alert ('error');
20.        return response;
21.    }
22.
23. });
24. </script>

```

```

<script type="text/javascript" src="{$_local_dir}pub/dojo/dojo/dojo.js" djConfig="parseOnLoad:true,
isDebug:true"></script>

```

```

<script type="text/javascript">

```

```

dojo.xhrPost({
    url: "{/literal}{$ajax_url}{literal}",
    handleAs: "json",
    timeout: 1000,
    postData: dojo.toJson({// schreiben
        'call_file': 'live/check_liveserver.php',
        'call_function': 'check_liveserver',
        'params': {'AA' : 'BB'}
    }),

```

```

load: function(responseObject, ioArgs){
    alert (responseObject); // lesen
    return responseObject;
},

```

```

error: function(response, ioArgs){
    alert ('error');
    return response;
}

```

```

});

```

```

</script>

```

Beispiel auf PHP-Seite:

QuelltextPHP Code:

```

1. <?php
2. function check_liveserver($params)
3. {
4.     return array('neu' => $params['AA'] . 'xxxxx');
5. }
6. ?>

```

```

<?php
function check_liveserver($params)

```

```
{  
  return array('neu' => $params['AA'] . 'xxxxx');  
}  
?>
```

In diesem Beispiel müsste das empfangene responseObject so aussehen: {'neu': 'BBxxxxx'}

Die Verzeichnisstruktur

EGOTEC gebraucht eine einfache und klare Verzeichnis-Struktur:

Verzeichnis	Beschreibung
bin	System: Alle ausführbaren Skripte.
lib	System: Verwendeten Bibliotheken und Skripte.
doc [optional]	Interne Dokumentationen.
setup [optional]	Installationsdateien
setup_create [optional]	Skripte zum Erstellen von Setup-Dateien. (In der Regel nicht vorhanden)
site	Mandatenspezifische Skripte
skin	Mandatenspezifische Templates
var	Speicherort für alle variablen Dateien wie Media-, Cache- und Sessiondateien. Weiter Informationen zum var-Verzeichnis finden Sie hier .

Neben diesen Verzeichnissen finden sich auf erster Ebene außerdem noch: **admin.php**, **index.php**, **rewrite.php**, **favicon.ico** und eine **.htaccess**-Datei.

Weitere Hinweise zu den Verzeichnissen erhalten Sie in folgenden Kapiteln.

Einblick in diese Verzeichnisstruktur erhalten Sie entweder über einen direkten Zugriff auf den Server (SSH, FTP, oder ähnliches) oder als Superuser über unsere WebDAV-Schnittstelle (.system-Verzeichnis)

Systemverzeichnisse

Die Verzeichnisse **lib** und **bin** bilden den Kern des Egotec-Systems. Beide Verzeichnisse werden bei einem Systemupdate vollständig abgeglichen.

Je nach Lizenz können hier unterschiedliche Unterverzeichnisse auftauchen.

Alle Änderungen an bestehenden Skripten im **bin** und **lib**-Verzeichnis werden bei einem Systemupdate **überschrieben!**

Bitte ändern Sie daher nichts in diesem Verzeichnis.

Kundenverzeichnisse

Alle kundenspezifischen Anpassungen befinden sich (bis auch einige Ausnahmen) im *site* und *skin*-Verzeichnis. Arbeiten an Templates und Funktionen geschehen fast ausschließlich nur an diesen beiden Orten. Daher bleiben diese auch bei einem Systemupdate unberührt.

Skripte (site)

Jeder Mandant besitzt ein Verzeichnis mit dessen Namen. Darunter liegen jeweils die entsprechenden Skripte und Seitentypen.

Templates (skin)

Für jedes Design existiert hier ein separates Unterverzeichnis mit dem entsprechenden Namen. Darunter befinden sich neben dem Haupttemplate mit allen benötigten CSS und JavaScript-Dateien, die typenspezifischen Templates.

Variable Dateien

Im "var"-Verzeichnis befinden sich variable Dateien des Systems, sprich: Alles, was vom System auf Verzeichnis-Ebene gesichert wird.

Verzeichnis Beschreibung

- var/backup Speicherort für Datensicherungen, die über den Admin-Bereich erstellt werden.
- var/cache Interner Cache von EGOTEC.
- var/conf Verzeichnis für lokale Konfigurationsdateien.
- var/log Verzeichnis für LOG-Dateien, wie z.B. Statistiken und Fehlermeldungen.
- var/media Verzeichnis für im Multimedia-Mandanten eingespielte Dateien. (werden über ID gespeichert)
- var/session Verzeichnis für Session-Daten der einzelnen Benutzer
- var/tmp Speicherort für temporäre Dateien

Das Basisskript

Das Hauptskript einer Site ist die site / *Mandant* / **index.php**.

Die meisten grundlegenden Funktionen einer Seite lassen sich allerdings auch schon über die entsprechenden Templates steuern, so dass Sie unter Umständen komplett auf eine index.php verzichten können. Müssen Sie aber bestimmte Aktionen für alle Seiten des Auftritts realisieren, ist die index.php die beste Stelle, um solche Skripte einzubinden.

Beispiel 1

Die index.php der Demo-Site z.B. sorgt dafür, dass bei Seiten mit leerem Inhalt direkt der Inhalt der ersten untergeordneten Seite angezeigt wird:

QuelltextPHP Code:

```
1. /* hat der Inhalt der aktuellen Seite weniger als 14 Zeichen und ist die Seite vom Typ "page"? */
2. if ((strlen($page->field['content'])<14) && ($page->field['type']=='page'))
3. {
4.     // Kinder der Seite auslesen
5.     $child = $page->getChildren();
6.     if ($child->nextPage())
7.     {
8.         $page = $child->page; // aktuelle Seite mit der Kind-Seite überschreiben
9.     }
10. }
```

```
/* hat der Inhalt der aktuellen Seite weniger als 14 Zeichen und ist die Seite vom Typ "page"? */
if ((strlen($page->field['content'])<14) && ($page->field['type']=='page'))
{
    // Kinder der Seite auslesen
    $child = $page->getChildren();
    if ($child->nextPage())
    {
        $page = $child->page; // aktuelle Seite mit der Kind-Seite überschreiben
    }
}
```

Beispiel 2

Außerdem können in der index.php auch weitere allgemein gültige Skripte eingebunden werden.

QuelltextPHP Code:

```
1. require('emotion.php'); // Skript für Emotionbilder.
```

```
require('emotion.php'); // Skript für Emotionbilder.
```

Der Pfad zu den Skripten ist dabei relativ zum aktuellen Site-Verzeichnis

Beispiel 3

Die Demo-Site bietet z.B. noch die Möglichkeit, einen zweiten Inhaltsbereich einzubinden. Je nach Seitentyp können hier unterschiedliche Templates zum Einsatz kommen. Auch ein solcher Skriptteil kann in die index.php integriert werden:

QuelltextPHP Code:

```
1. // der Pfad zu einer Template-Datei für einen zweiten Inhaltsbereich
2. $template_file = $GLOBALS['egotec_conf']['skin_dir'].$site->skin.'/$page->field['type']/body_2.html';
3.
4. if (!file_exists($template_file)) // existiert hier keine Template-Datei für einen zweiten Inhaltsbereich?
5. {
6.     // auf ein Standard-Template zurückgreifen
7.     $template_file = $GLOBALS['egotec_conf']['skin_dir'].$site->skin.'/body_2.html';
8. }
9.
10. if (file_exists($template_file)) // gibt es eine solche?
11. {
12.     $smarty->assign('typeTemplate2', $template_file); // an Smarty als Platzhalter übergeben
13. }
```

```
// der Pfad zu einer Template-Datei für einen zweiten Inhaltsbereich
$template_file = $GLOBALS['egotec_conf']['skin_dir'].$site->skin.'/$page->field['type']/body_2.html';
```

```
if (!file_exists($template_file)) // existiert hier keine Template-Datei für einen zweiten Inhaltsbereich?
{
    // auf ein Standard-Template zurückgreifen
    $template_file = $GLOBALS['egotec_conf']['skin_dir'].$site->skin.'/body_2.html';
}
```

```
if (file_exists($template_file)) // gibt es eine solche?
{
    $smarty->assign('typeTemplate2', $template_file); // an Smarty als Platzhalter übergeben
}
```

Wenn entweder das Typenspezifische- oder das Standardtemplate existiert, wird dieses dem Haupttemplate zugewiesen und kann dann vom Designer entsprechend eingebunden werden.

Neue Seitentypen erstellen

Um einen Seitentyp zu erstellen, sollte man zunächst wissen, was einen Seitentyp überhaupt ausmacht.

Ein Seitentyp...

- .. kann auf dem Informationsreiter für eine beliebige Seite eingestellt werden.
- .. kann aus einer Sammlung von verschachtelten Seitentypen bestehen (vgl. "News/Liste", "News/Eintrag")
- .. bietet in der Regel im Frontend ein bestimmtes Layout und verschiedene Funktionen an.
- .. kann im Adminbereich eigene Reiter haben, auf denen man Einstellungen vornehmen kann.
Die Reiterinhalte können unterschiedlich aussehen und verschiedene Eingabe-Elemente besitzen.
- .. kann einen angepassten "Neu"-Button besitzen (vgl. Newsliste => "Neue Nachricht") und Unterseiten automatisch von einem bestimmten Seitentyp erstellen.
- .. kann eine feste vordefinierte Sortierung der Unterseiten besitzen.
- .. kann speziell für einen Mandanten oder auch global für alle Mandanten genutzt werden.

Im ersten Schritt muss ein Seitentyp erst einmal erstellt und dem System bekannt gegeben werden.

Seitentyp erzeugen

Um einen neuen Seitentyp dem System bekannt zu machen, erstellen Sie direkt unter `site/Mandant/` ein neues Verzeichnis mit dem Namen des Seitentyps (z.B. *adressbuch*). Erstellen Sie darunter eine Datei `type.ini` mit folgendem Inhalt:

```
title = "Adressbuch"  
inactive = false
```

Der *title* ist die Bezeichnung des Seitentyps in der Drop-Down-Liste des Adminbereichs. Der *inactive*-Wert bestimmt, ob dieser Typ auswählbar sein soll.

Der neue Seitentyp sollte nun im Adminbereich sichtbar sein. Sie können diesen nun schon einer Seite zuweisen und speichern.

Verschachtelte Seitentypen

Einige Seitentypen (wie z.B. das News-Modul) setzen sich aus mehreren Seitentypen zusammen. Diese werden in der Auswahl verschachtelt dargestellt:



Verschachtelung

Seitentypen können beliebig tief verschachtelt werden. Hierzu wird pro Seitentyp einfach ein neues Unterverzeichnis mit einer type.ini erzeugt.

- site / Mandant / news
site / Mandant / news / type.ini
- site / Mandant / news / list
site / Mandant / news / list / type.ini
- site / Mandant / news / entry
site / Mandant / news / entry / type.ini
- site / Mandant / news / archive
site / Mandant / news / archive / type.ini

Inaktive Einträge

In dem Beispiel dient der Eintrag *Nachrichten* lediglich zur Strukturierung. Er kann nicht ausgewählt werden. Ein solches Verhalten erreicht man in der type.ini über den Wert *inactive=true*.

Die Seitentyp-Liste im Adminbereich wird nach einmaligem Anzeigen in einer Cache-Datei gespeichert. Um eigene Änderungen zu sehen, löschen Sie site/Mandant/**types.cache** oder leeren Sie den Cache über Extras->"Cache löschen".

Templates und Skripte

Wurde ein neuer Seitentyp angelegt und einer Seite zugeordnet, hat er zunächst keine Besonderheiten in der Ansicht. Dazu werden zwei zentrale Elemente benötigt:

Das typenspezifische Template (**body.html**)

Jeder Seitentyp kann ein eigenes Template besitzen, welches das Layout im Frontend ausmacht. Eine News/Liste zeigt z.B. eine Auflistung der neusten News untereinander, ein News/Eintrag dagegen die Nachricht selbst. Beide Layouts werden separat in den Templates der jeweiligen Seitentypen festgelegt.

Das typenspezifische Template wird stets pro Seitentyp in einer **body.html** unterhalb des Seitentyps gespeichert:

- skin / Design / news
skin / Design / news / entry
skin / Design / news / entry / body.html
- skin / Design / news / list
skin / Design / news / list / body.html
- skin / Design / news / archive
skin / Design / news / archive / body.html

Achten Sie darauf, dass ihr Seitentyp den gleichen Namen wie im Verzeichnis site/Mandant besitzt. Weitere Informationen zum Erstellen von Templates finden Sie im [Designerhandbuch](#).

Das typenspezifische Skript (**index.php**)

Für jeden Seitentyp kann ein eigenes Skript eingebunden werden. Dieses wird nur bei Aufruf des bestimmten Seitentyps geladen.

Zur allgemeiner Trennung von Funktionalität und Darstellung wird empfohlen, alle Aktionen wie: Berechnungen, Datenbankabfragen, Sortierung usw. in diesem Skript durchzuführen. Am Skriptende können alle Ergebnisse an Smarty übergeben werden und anschließend im Template positioniert werden.

Das typenspezifische Skript wird pro Seitentyp in einer **index.php** unterhalb des Seitentyps gespeichert:

- site / Mandant / news
site / Mandant / news / entry
site / Mandant / news / entry / index.php
- site / Mandant / news / list
site / Mandant / news / list / index.php
- site / Mandant / news / archive
site / Mandant / news / archive / index.php

Wird unterhalb von site/Mandant/news ebenfalls eine *index.php* erzeugt, wird diese bei allen news-Seitentypen eingebunden. Innerhalb dieser Skripte stehen bei jedem Seitenaufruf die wichtigsten Objekte \$site und \$page automatisch zu Verfügung.

Typenspezifische Reiter

Zur besseren Übersicht sind die einzelnen Eingabemöglichkeiten der Seiten auf verschiedene Reiter verteilt. Jeder Seitentyp besitzt allgemeine und spezielle Reiter, auf denen Einstellungen durchgeführt werden können. Die Reiterinhalte können unterschiedlich aussehen und verschiedene Eingabe-Elemente besitzen.

Zu unterscheiden sind

- **allgemeingültige Reiter:** Freigabe, Meta, Navigation, Workflow, Rechte, Archiv.
Diese Reiter sind für alle beliebigen Seitentypen immer gleich und standardmäßig angezeigt.
- **spezielle Reiter:** Information, Inhalt, und weitere..
Diese Reiter können angepasst bzw. je nach Seitentyp ausgeblendet werden. Ebenso können Sie eigene Reiter anlegen.

Reiter festlegen

Alle anpassbaren Reiter sind in *site/Mandant/Seitentyp/admin/navigation.ini* definiert. Hier kann pro Seitentyp eingestellt werden, welche Reiter angezeigt werden sollen und welchen Namen diese haben.

```
[information]
url = info.php
title = Information
```

```
[content]
url = edit.php
title = Inhalt
```

Die navigation.ini besteht dabei aus mehreren Sektionen, die jeweils einen Reiter darstellen. Zu jedem Reiter müssen die Werte *url* und *title* gesetzt werden. Sobald Sie eine solche *navigation.ini* erstellt haben, sollte sie zumindest über den [information] Einträge verfügen, da ansonsten wichtige Punkte wie Name, Titel und Kurzbeschreibung nicht bearbeitet werden können.

Einige Reiter (Meta, Navigation, usw.) werden unabhängig von den Angaben in der navigation.ini eingebunden. Diese können nicht Site-Spezifisch in der Datei "site/SiteName/admin/navigation.ini" angepasst werden.

Ausgenommen dem Information- und Inhaltsreiter werden alle durchgeführten Einstellungen im extra-Feld der Seite gespeichert.

Informationsreiter erweitern

Wenn nur wenige zusätzliche Eingabefelder benötigt werden, hat es sich bewährt, auf einen eigenen Reiter zu verzichten und statt dessen einfach den Informationsreiter entsprechend zu erweitern. Dazu erweitern Sie im admin-Verzeichnis des entsprechenden Typs die navigation.ini:

```
[information]
url = "info.php?tpl_name=information.html&script_name=information.php"
title = Information
```

Über den Parameter *tpl_name* wird mitgeteilt, welches Template zusätzlich auf dem Informationsreiter eingebunden werden soll (die information.html muss entsprechend im admin-Verzeichnis existieren).

information.html (Teil 1)

```
<form name="extra">
<div align="center">
  <table class="table">
    <tr>
      <td colspan="2" class="cell">
        <table border="0" cellspacing="1" cellpadding="4" class="table">
          <tr>
            <td>Mein eigenes Eingabefeld</td>
            <td><input type="text" name="extra[anzahl_news]"></td>
          </tr>
        </table>
      </td>
    </tr>
  </table>
</div>
</form>
```

Die DIV und Table-Tags mitsamt den CSS-Klassen bilden ein Grundgerüst, welches sich an das Design im Adminbereich anpasst. Innerhalb des Formulars können nun beliebig viele weitere Eingabefelder definiert werden. Der Formularname "extra" darf nicht geändert werden.

Im anschließenden JavaScript-Bereich wird das Speichern und Laden der Felder definiert:

information.html (Teil 2)

```
{literal}
<script language="javascript">
<!--
function do_load_extra()
{
  f = document.forms["extra"];
  f.elements["extra[anzahl_news]"].value = window.parent.get_extra("anzahl_news");
}

function do_unload_extra()
{
  f = document.forms["extra"];
  window.parent.set_extra("anzahl_news", f.elements["extra[anzahl_news]"].value);
}
//-->
</script>
{/literal}
```

Zunächst wird der komplette JavaScript-Teil von Smarty {literal}-Tags umschlossen, da die Template Engine geschweifte Klammern als Funktionsanfang interpretiert und eine Fehlermeldung wirft. Innerhalb dieser Tags findet damit keine Smarty-Funktionalität statt.

Um die Eingaben nun korrekt speichern und laden zu können, werden 2 vordefinierte Funktionen verwendet:

- do_load_extra()
Wird beim Laden des Reiters aufgerufen

- do_undload_extra()
Wird beim Verlassen oder Speichern des Reiters aufgerufen.

In diesen Funktionen wird über get_extra (den aktuell gespeicherten Wert der Seite auslesen) und set_extra (den Wert in die Seite schreiben) die entsprechende Werte beim Wechseln der Reiter übernommen bzw. geladen. Die Datenbankstruktur wird hierbei nicht geändert. EGOTEC legt bei Bedarf entsprechende "Felder" an. Über die Methoden get_extra und set_extra greifen Sie per Javascript auf diese Einträge zu. Die Speicherung dieser Daten erfolgt in einem objektorientierten Ansatz in der relationalen Datenbank.

Informationsreiter global erweitern

Der Informationsreiter lässt sich auch für alle Seitentypen innerhalb eines Mandanten erweitern. Dazu erstellt man unter `site/Mandant/admin` eine neue HTML-Datei **information_global.html**.

```
<form name="extra3">
<div align="center">
  <table class="table">
    <tr>
      <td colspan="2" class="cell">
        {input type="text" name="webcode" title="Webcode" short="Wird bei der Suche mit berücksichtigt"}
      </td>
    </tr>
  </table>
</div>
</form>

{literal}
<script language="javascript">
<!--

function do_load_extra3()
{
  document.forms["extra3"].elements["extra[webcode]"].value = window.parent.get_extra("webcode");
}

function do_unload_extra3()
{
  window.parent.set_extra("webcode", document.forms["extra3"].elements["extra[webcode]"].value);
}

//-->
</script>
{/literal}
```

Bei globaler Erweiterung muss im Javascriptbereich die Funktion **do_load_extra3** und **do_unload_extra3** verwendet werden. Es ist auch darauf zu achten, dass das Formular nicht wie ursprünglich den Namen "extra" sondern **extra3** erhält.

Neue Reiter erstellen

Bei mehreren Eingabefeldern oder Zugriff auf externe Daten, für die weitere Skripte benötigt werden, empfiehlt es sich einen neuen Reiter zu erstellen.

Dazu erstellt man einen entsprechenden Eintrag in der **navigation.ini**:

```
[meinReiter]
url = extra.php?tpl_name=meinTyp/admin/meinReiter.html&script_name=meinTyp/admin/meinReiter.php
title = "Mein Reiter"
```

Als url wird in diesem Fall "extra.php" aufgerufen, folgend von zwei Parametern:

- *tpl_name* (Pfad zum Template)

- *script_name* (Pfad zum Skript, ausgehen von *site/Mandant*)

Während die Übergabe des Templates zwingend erforderlich ist, muss ein Skript nicht unbedingt übergeben werden.

Die Templatedateien werden dabei nach dem gleichen Prinzip wie der erweiterte Informationsreiter erstellt, allerdings müssen die Javascript-Funktionen zur Kommunikation **do_load()** bzw. **do_unload()** benannt werden.

Verwendung von Eingabefeldern

Um Eingabefelder wie: "Auswahl einer Seite aus dem Multimediabereich" oder "Datumauswahl" verwenden zu können, bietet EGOTEC die Smarty-Funktion {input}. Damit können im Adminbereich auf eigenen Reitern leicht und unkompliziert vordefinierte Eingabefelder erstellt und positioniert werden.

Im Designerhandbuch finden Sie nähere Informationen bezüglich [Einsatz des {input}-Plugins](#).

Globale Reiter erstellen

Soll ein Reiter für ALLE Seitentypen verfügbar sein, steht Ihnen eine separate *navigation.ini* im Verzeichnis *site/Mandant/admin* zur Verfügung. Diese ist im gleichen Stil wie die typenspezifischen Navigations-Dateien zu behandeln.

In der "navigation.ini" können nach Bedarf Seitentypen ausgeschlossen werden.
ignore_types = Kommaseparierte Liste von Seitentypen die ausgeschlossen werden sollen

Zweiten Inhaltsreiter erstellen

Einen Reiter mit einem zusätzlichen WYSIWYG-Editor erstellen Sie in der navigation.ini über folgende drei Zeilen:

```
[content_2]
url = "edit.php?content=zweiter_inhalt"
title = "Inhalt 2"
```

Der Parameter *content* entspricht dem Namen der Variable (im extra-Feld der Seite), in welche der eingegebene Inhalt gespeichert wird. Dem Beispiel zu folge, kann der Inhalt mit dem Platzhalter {*page->extra.zweiter_inhalt*} im Template ausgegeben werden.

Editorbreite einstellen

```
url = "edit.php?content=zweiter_inhalt&content_width=300"
```

Über den zweiten Parameter *content_width* kann die Breite des Editors in Pixel angegeben werden (im Beispiel: 300px).

Typenspezifische Einstellungen

Für jeden Seitentyp separat können administrative Einstellungen und Optionen vorgenommen werden.

Der Großteil solcher Einstellungen erfolgt in `site/Mandant/Seitentyp/admin/index.php`.

Eigenschaften der aktuellen Seite

Für jede Seite können in `site/Mandant/Seitentyp/admin/index.php` feste Eigenschaften definiert werden. Diese werden z.B. beim Auslesen der Seite über Page-Methoden mit berücksichtigt.

Beispiel

QuelltextPHP Code:

1. // Sortierung aller bestehenden Unterseiten festlegen
2. `$page->field['children_order'] = 'desc';`

```
// Sortierung aller bestehenden Unterseiten festlegen
$page->field['children_order'] = 'desc';
```

Über das `$page` -Objekt haben Sie direkten Zugriff auf die aktuelle Seite. Werte, die Sie hier setzen, werden beim Speichern der Seite in die DB übernommen.

Vorgegebene Werte sind fest für die Seite definiert und werden durch manuelles Ändern und Speichern nicht übernommen.

Bei jedem Aufruf der Seite im Adminbereich werden die Werte wieder entsprechend diesen Angaben zurück gesetzt.

Eigenschaften neuer Unterseiten

Beim Anlegen einer neuen Seite können Eigenschaften wie Typ, Name, Titel usw. vorbelegt werden. Das macht z.B. Sinn bei News/Übersicht-Seiten, wo alle neue Unterseiten automatisch den Typ "News/Eintrag" erhalten sollen.

Letztere Einstellungen werden in einer ***index.php*** im Verzeichnis `site/Mandant/Seitentyp/admin/` durchgeführt.

QuelltextPHP Code:

1. // Unterseite soll ein Newseintrag sein
2. `$new_child['type'] = 'news/entry';`

- 3.
4. // Die Eigenschaft "Nicht in Navigation anzeigen" für alle neue Unterseiten vor belegen
5. `$new_child['nav_hide'] = 1;`

```
// Unterseite soll ein Newseintrag sein  
$new_child['type'] = 'news/entry';
```

```
// Die Eigenschaft "Nicht in Navigation anzeigen" für alle neue Unterseiten vor belegen  
$new_child['nav_hide'] = 1;
```

Mit dem *\$new_child*-Array können Sie alle Werte einer potentiellen Unterseite mit Standardwerten vorbelegen.

Zusätzliche Aktionen

Möchten Sie, dass zum Beispiel nach dem Speichern einer Seite (mit speziellem Typ) zusätzliche Aktionen durchgeführt werden, kann dafür ein zusätzliches Skript eingebaut werden:
site/Mandant/Seitentyp/admin/action_update.php.

Ebenso sind für andere Standard-Aktionen update-Skripte vorgesehen:

- *action_update.php*
- *action_delete.php*
- *action_remove.php*
- *action_move_parent.php*
- usw.

Desklets erstellen

Unter Umständen kann es sinnvoll sein, bestimmte Seitentypen mit dem Desktop zu verknüpfen. Dafür stehen auf dem Desktop *Desklets* zur Verfügung. Die sind nichts anderes als Zellen, die Informationen anzeigen.

Um eine neue Zelle zu erstellen, werden 2 Dateien benötigt:

- *site / Mandant / Seitentyp / admin / **desktop.php***
- *site / Mandant / Seitentyp / admin / **desktop.html***

In der *desktop.html* wird das Template der Zelle definiert. Die *desktop.php*-Datei bindet die Zelle ins das System ein.


```

38. }
39.
40. // Fertige Zelle hinzufügen
41. $GLOBALS['cells'][] = $cell;
42.
43. ?>

```

```
<?php
```

```

/**
 * Beispiel-Desklet
 * Seitenstatistiken ermitteln und als Deklet anzeigen
 */

/**
 * Seiten ermitteln und an Smarty übergeben
 */
// Anzahl aktiver Seiten
$alle_seiten = $site->getPages(array(),array('auth_or' => '1=1'));
$smarty->assign('aktive_seiten',$alle_seiten->numRecords());

// Anzahl inaktiver Seiten
$alle_seiten = $site->getPages(array('where' => 'inactive = 1'),array('auth_or' => '1=1'));
$smarty->assign('inaktive_seiten',$alle_seiten->numRecords());

// Anzahl gelöschter Seiten
$alle_seiten = $site->getPages(array('where' => 'deleted = 1'),array('auth_or' => '1=1','deleted' => 1));
$smarty->assign('geloeschte_seiten',$alle_seiten->numRecords());

/**
 * Desklet vorbereiten
 */
$cell = array();
$cell[0]['name'] = 'Mandant-Informationen'; // Titel
$cell[0]['id'] = 'mandant_infos'; // id-Bezeichnung
$cell[0]['icon'] = "../admin_skin/egotec/desktop/stats.gif"; // Icon

// Desklet Inhalt aus der desktop.html einbinden
if(in_array($site->name.'_'.$cell[0]['id'], $active_cells)) // Nur anzeigen wenn aktiv
{
    $cell[0]['body'] =
$smarty->fetch($GLOBALS['egotec_conf']['site_dir'].$GLOBALS['site']->name.'/page/admin/desktop.html');
} else
{
    $cell[0]['body'] = "";
}

// Fertige Zelle hinzufügen
$GLOBALS['cells'][] = $cell;

?>

```

Das Skript ermittelt zunächst über die getPages-Funktion des Site-Objekt die gewünschten Seiten-Mengen. Diese werden nach jeder Abfrage jeweils als Smarty-Platzhalter zur Verfügung gestellt.

Im zweiten Schritt werden Elemente wie Titel und Icon des Desklets bestimmt. Über die Smarty->fetch

Methode wird das Template (body.html) in das body-Element der Zelle eingebunden und angezeigt.

Zuletzt wird die fertige Zelle über `$GLOBALS['cells'] = $cell;` dem System bekannt gegeben.

Desklet anzeigen

Sobald das neue Desklet über "Ansicht->Mandanten-Informationen" aktiviert wurde, erscheint dieses auf dem Desktop-Bereich.



Mandant-Informationen	
aktive Seiten	962
inaktive Seiten	178
gelöschte Seiten	1

Nach dem Erstellen des neuen Desklets ist ein erneuter Login nötig, damit das Desklet in dem Ansicht-Menü erscheint.

Toolbar anpassen

Die Toolbar kann in `site/Mandant/Seitentyp/admin/index.php` pro Seitentyp individuell angepasst werden. Dazu steht innerhalb dieser `index.php` das Objekt `$toolbar_menu` zur Verfügung, über dessen Methoden neue Buttons erstellen können.

Neuen Menüpunkt erstellen

QuelltextPHP Code:

```
1. $toolbarMenu->addMainItem(array(
2.   'id'      => 'buttonNew2',
3.   'text'    => "<img src='\".$GLOBALS['egotec_conf']['url_dir'].
4.             \"bin/admin_skin/egotec/img/home_big.gif' border=0 style='width: 24px; height:
   24px'/><br/>".
5.             $GLOBALS['auth']->translate('Neue Unterseite'),
6.   'url'     => get_url($GLOBALS['global_conf']['url_dir'].'bin/page/action.php', array(
7.     'site'      => $site->name,
8.     'lang'      => $site->language,
9.     'field[id]' => $page->field['id'],
10.    'new_child[name]' => 'Neue Unterseite',
11.    'new_child[title]' => 'Neue Unterseite',
12.    'new_child[type]' => 'news/entry',
13.    'new_child[nav_hide]' => 5,
14.    'egoaction'  => 'new_child'
15.  )),
16.  'target' => 'alive',
17.  'alt'    => $GLOBALS['auth']->translate('Neue Unterseite'),
18.  'active' => 1
19.  ));
```

```

$toolbarMenu->addMainItem(array(
    'id'      => 'buttonNew2',
    'text'    => "<img src='\".$GLOBALS['egotec_conf']['url_dir'].
                \"bin/admin_skin/egotec/img/home_big.gif\" border=0 style='width: 24px; height: 24px' /><br/>".
                $GLOBALS['auth']->translate('Neue Unterseite'),
    'url'     => get_url($GLOBALS['global_conf']['url_dir'].'bin/page/action.php', array(
        'site'      => $site->name,
        'lang'      => $site->language,
        'field[id]' => $page->field['id'],
        'new_child[name]' => 'Neue Unterseite',
        'new_child[title]' => 'Neue Unterseite',
        'new_child[type]' => 'news/entry',
        'new_child[nav_hide]' => 5,
        'egoaction' => 'new_child'
    )),
    'target' => 'alive',
    'alt'    => $GLOBALS['auth']->translate('Neue Unterseite'),
    'active' => 1
));

```

Über die Methode **addMainItem()** wird innerhalb der Toolbar ein neuer Button erstellt. Der Methode wird dabei ein Array übergeben, das die Eigenschaften des neuen Button enthält:

Parameter Beschreibung

id	Ein eindeutiger Name für den neuen Button
text	Die Beschriftung des Button in HTML. Mit Hilfe eines -Tags können Sie also auch Bilder einbinden
url	Die URL die bei Klick auf den Button aufgerufen werden soll
target	Das Frame/Fenster in dem der Link geöffnet werden soll
alt	Ein Alternativ-Text für den Button. Dieser Text erscheint bei einem Mouse-Over
active	1 oder 0 / Gibt an, ob der Button geklickt werden kann.
image	Die URL zu einem Bild

Neuen Menü-Unterpunkt erstellen

QuelltextPHP Code:

```

1. $toolbarMenu->addItem('myButton', array(
2.     'id'      => 'mySecondButton',
3.     'text'    => 'Noch ein Button',
4.     'url'     => "",
5.     'target'  => "",
6.     'alt'     => 'Noch ein Button',
7.     'active'  => 1
8. ));

```

```

$toolbarMenu->addItem('myButton', array(
    'id'      => 'mySecondButton',
    'text'    => 'Noch ein Button',
    'url'     => "",
    'target'  => "",
    'alt'     => 'Noch ein Button',
    'active'  => 1
));

```

Über die Methode **addItem()** wird ein bestehender Button, um Unterpunkte erweitert. Dadurch entsteht ein Drop-Down-Menüs, dass sich bei Klick auf den ersten Button öffnet. Der Methode wird dabei im ersten

Parameter die ID des Buttons übergeben, zu dem ein Unterbutton erstellt werden soll.
Der zweite Parameter definiert wieder die Eigenschaften des Buttons. Über addItem() können so auch Buttons auf zweiter, dritter oder tieferen Ebenen erstellt werden.

Trenner einfügen

QuelltextPHP Code:

1. // Einen neuen Trenner hinzufügen
2. \$toolbarMenu->addMainRuler('mein_trenner');

```
// Einen neuen Trenner hinzufügen  
$toolbarMenu->addMainRuler('mein_trenner');
```

Ein Trennstrich kann über zwei Zeilen Code hinzugefügt werden.

Globale Seitentypen

Seitentypen können Mandanten-übergreifend verwendet werden, wenn diese global definiert sind.

Legen Sie dazu die 2 Verzeichnisse `site/_global/` und `skin/_global` an. Innerhalb dieser Verzeichnisse können Sie wie gewohnt Seitentypen erstellen bzw. kopieren.

Beispiel

```
site
site / _global
site / _global / adressbuch
site / _global / adressbuch / index.php
site / _global / adressbuch / type.ini
```

```
skin
skin / _global
skin / _global / adressbuch
skin / _global / adressbuch / body.html
```

Globale Seitentypen werden von Mandanten-spezifischen Seitentypen überschrieben.

Um globale Seitentypen benutzen zu können, müssen diese eventuell erst aktiviert werden.

Weiterführende Möglichkeiten

Benutzerverwaltung um eigene Reiter erweitern

Um einen eigenen Reiter in der Benutzerverwaltung zu erstellen gehen Sie wie folgt vor:

Erstellen Sie in var folgende Ordnerstruktur lib/rights/t/ und unter var/lib/rights die Datei user_profile_tab.ini. Diese Datei definiert zusätzliche Reiter welche in der Benutzerverwaltung erscheinen sollen. Die ini Datei verhält sich ähnlich der *navigation.ini* der Seitentypen.

Beispiel (var/lib/rights/user_profile_tab.ini)

```
[profile]
name = Profil
title = Profil
template = rights/t/user_profile_dlg.html
script = rights/user_profile_dlg.php
right = desktop
```

```
[kunden_daten]
name = "Kundendaten"
title = "Kundendaten"
template = rights/t/kunden_daten.html
script = rights/kunden_daten.php
right = desktop
```

Der erste Bereich [profile] sollte standardmäßig immer hinzugefügt werden, damit Benutzername und Passwort weiterhin bearbeitet werden können.

Der zweite Bereich stellt einen neuen Reiter dar. Die Parameter *template* und *script* erhalten jeweils den Pfad zur Template- und Skriptdatei. Der Parameter *right* steht für das Ansichtsrecht. Templates werden im t Verzeichnis abgelegt.

Beispiel (Skript: var/lib/rights/kunden_daten.php)

QuelltextPHP Code:

```
1. <?php
2.  $smarty->assign('aktuelles_datum',date("d.m.Y")); // Das aktuelle Datum als Smarty-Platzhalter
   übergeben
3. ?>
```

```
<?php
  $smarty->assign('aktuelles_datum',date("d.m.Y")); // Das aktuelle Datum als Smarty-Platzhalter übergeben
?>
```

Beispiel (Template: var/lib/rights/t/kunden_daten.html)

QuelltextHTML Code:

```
1. <form method="post" name="edit">
2.
3. <div align="center">
```


Verwendung von Captcha

Erklärung wie man Captcha auf PHP und Template Ebene korrekt einbindet.

Um zu verhindern, dass Bots ein Formular absenden, werden Captchas verwendet. Hierbei handelt es sich um kleine Bilder, die meist eine Folge von Zahlen, Buchstaben und Zeichen oder sogar einfachen mathematischen Rechnungen beinhalten. Dieser Code wird bei jedem Aufruf zufällig generiert und ist für Maschinen oft gar nicht oder nur schwer lesbar. Daher wird er dazu verwendet, einen echten Benutzer und eine maschinelle Eingabe zu unterscheiden.

Eine solche Captcha-Funktion wird im System von EGOTEC angeboten.

Die Captcha-Funktion benötigt das GD-Lib Modul mit integrierter Free-Type Bibliothek (<http://de2.php.net/imagettfbbox>).

Vorbereitungen

Im ersten Schritt wird die Captcha-Klasse eingebunden. Sobald das geschehen ist, steht sowohl im Skript als auch als Smarty-Variable automatisch das Objekt **\$captcha** zur Verfügung:

QuelltextPHP Code:

1. // Captcha einbinden
2. `require_once('captcha/Ego_Captcha.php');`

```
// Captcha einbinden  
require_once('captcha/Ego_Captcha.php');
```

Überprüfung (im Skript über PHP)

Mit Hilfe der Methode `check()` kann nun einfach überprüft werden, ob die Eingabe korrekt war:

QuelltextPHP Code:

```
1. if($captcha->check($modus,$name))
2. {
3.   echo 'Captcha wurde korrekt ausgefüllt';
4. }
```

```
if($captcha->check($modus,$name))
{
  echo 'Captcha wurde korrekt ausgefüllt';
}
```

Hierbei existieren zwei Parameter:

Parameter Beschreibung

modus	Art der Captcha Prüfung ("string" oder "number"). Der Wert "string" wird ausdrücklich empfohlen.
name	Prüfung, ob Captcha-Funktion aktiviert ist. Beim Wert current wird für die aktuellen Seite geprüft, ob im Backend die Captcha-Funktion aktiviert wurde. (Der Seitentyp muss entsprechend um eine [optional] Checkbox im Adminbereich erweitert werden). Wird der Parameter weggelassen, erfolgt keine Prüfung auf Aktivierung: Die Captcha-Funktionalität ist automatisch aktiv.

Einige Fallbeispiele:

- Kann man bei der aktuellen Seite Spam-Schutz einstellen und ist dieser deaktiviert, liefert diese Funktion immer **true** zurück.
- Ist der Spam-Schutz aktiviert, liefert diese Funktion **false** zurück wenn die Überprüfung fehlgeschlagen ist.
- Kann man Spam-Schutz nicht einstellen, ist dieser automatisch immer aktiviert und diese Funktion liefert **false** zurück wenn die Überprüfung fehlgeschlagen ist.

Überprüfung (im Template über Smarty)

Die check-Methode kann über das \$captcha-Objekt ebenso im Template verwendet werden.

QuelltextSmarty Code:

1. `{* Prüfen, ob für die aktuelle Seite Captcha aktiviert wurde (diese Prüfung kann auch weg gelassen werden) *}`
2. `{if $captcha.active.current}`
- 3.
4. `{* Auf Eingabe prüfen *}`
5. `{if !$captcha.check}`
6. `Die Captcha Prüfung ist fehlgeschlagen.`
7. `{/if}`
- 8.
9. `{/if}`

```
{* Prüfen, ob für die aktuelle Seite Captcha aktiviert wurde (diese Prüfung kann auch weg gelassen werden) *}
{if $captcha.active.current}
```

```
{* Auf Eingabe prüfen *}
{if !$captcha.check}
    Die Captcha Prüfung ist fehlgeschlagen.
{/if}
```

```
{/if}
```

Damit die Überprüfung auf korrekte Eingabe vom \$captcha-Objekt überhaupt durchgeführt werden kann, müssen im Template die einzelnen Elemente wie Captcha-Bild, Eingabe-Feld, reload-Button usw. positioniert werden:

QuelltextSmarty Code:

1. `{* Über die Smarty-Funktion get_captcha werden HTML-Elemente generiert *}`
2. `{get_captcha var_text="cap_text" var_image="cap_image" var_reload="cap_reload" height=60}`
3. `<table style="width:500px;">`
4. `<tr>`
5. `<td valign="top">`
6. `{ $cap_image }`
7. `</td>`
8. `<td class="{if $error_captcha}error{/if}" valign="top" width="100%" style="padding-left: 10px;">`
9. `{t}Sicherheitsprüfung{/t}`
10. `{ $cap_text }`
11. `{ $cap_reload }`
12. `</td>`
13. `</tr>`
14. `</table>`

```
{* Über die Smarty-Funktion get_captcha werden HTML-Elemente generiert *}
{get_captcha var_text="cap_text" var_image="cap_image" var_reload="cap_reload" height=60}
<table style="width:500px;">
<tr>
<td valign="top">
    { $cap_image }
</td>
```

```
<td class="{if $error_captcha}error{/if}" valign="top" width="100%" style="padding-left: 10px;">
  <span style="display:block; padding-bottom:5px;">{t}Sicherheitsprüfung:{/t}</span>
  {Scap_text}
  <span style="display:block; padding-top:5px;">{Scap_reload}</span>
</td>
</tr>
</table>
```

Weitere Informationen finden Sie im Designerhandbuch unter [{get_captcha}](#).

3. Spam-Schutz einstellbar machen

Möchten Sie im Adminbereich der Seite den Captcha-Schutz aktivierbar machen, erstellen Sie auf dem gewünschten Reiter eine entsprechende Checkbox. Diese muss zwingend den Namen **captcha_active** tragen, da auf dieses Extrafeld in der **Ego_Captcha** Klasse geprüft wird. Gültige Werte für dieses Feld im Falle einer Aktivierung sind **1** oder **true**.

Den Text "neu laden" für das Captcha können Sie in /skin/mandant/locale/sprache übersetzen. Lesen Sie dazu bitte auch Mehrsprachigkeit. Ergänzen Sie die translation.ini einfach um den Zeile neu laden = reload (z.B. für die Englische Sprache)

Menüleiste anpassen

Auch die Menüleiste (oben) kann mandantenspezifisch angepasst werden.

Dazu muss die Datei site/Mandant/admin/menü.php angelegt werden. In dieser Datei wird die Funktion `update_site_menu` definiert und als Parameter das Menü-Objekt übergeben.

Beispiel

QuelltextPHP Code:

```
1. <?php
2.
3. function update_site_menu($menu)
4. {
5.     // Erstellt einen neuen Eintrag unter dem Menu "?".
6.     $menu->addlItem('help', array(
7.         'id'      => 'mein_eintrag',
8.         'text'    => 'Mein neuer Eintrag',
9.         'alt'     => 'Mein neuer Eintrag',
10.        'active'  => 1,
11.        'image'   =>
12.            $GLOBALS['egotec_conf']['url_dir'].'bin/admin_skin/egotec/img/16x16/handbuch.png',
13.        'url'     => 'javascript:alert(1)'
14.    ));
15. }
16. ?>
```

```
<?php
```

```
function update_site_menu($menu)
{
    // Erstellt einen neuen Eintrag unter dem Menu "?".
    $menu->addlItem('help', array(
        'id'      => 'mein_eintrag',
        'text'    => 'Mein neuer Eintrag',
        'alt'     => 'Mein neuer Eintrag',
        'active'  => 1,
        'image'   => $GLOBALS['egotec_conf']['url_dir'].'bin/admin_skin/egotec/img/16x16/handbuch.png',
        'url'     => 'javascript:alert(1)'
    ));
}

?>
```

Einzelheiten über Methoden, mit welchen Sie das Menü ändern können, werden auf [dieser Seite](#) weiter erklärt.

Crons definieren

Eigene Cronjobs (Dienste) erfüllen beliebige Aufgaben und können zeitgesteuert ausgeführt werden. Über den Administrationsbereich des CMS lassen sie sich anschließend verwalten und steuern.

Ein Cronjob besteht aus einem **einfachen PHP-Skript** (site/Mandant/admin/cron_zugriffe_berechnen.php)

QuelltextPHP Code:

1. <?php
- 2.
3. // Beispiel Skript für einen Cron (Dienst)
4. function berechne_zugriffe()
5. {
6. echo 'Berechne die Zugriffe
';
7. }
- 8.
9. ?>

```
<?php
```

```
// Beispiel Skript für einen Cron (Dienst)
```

```
function berechne_zugriffe()
```

```
{  
    echo 'Berechne die Zugriffe<br />';  
}
```

```
?>
```

und einer **cron.in**, in welcher der Cron-Dienst selbst definiert wird:

```
[zugriffs_berechnungen]
```

```
section = site
```

```
name = "Zugriffe berechnen"
```

```
script = "../..../site/Mandant/admin/cron_zugriffe_berechnen.php"
```

```
function = "berechne_zugriffe"
```

Folgende Parameter sind zu setzen:

Parameter	Beschreibung
[cron_bezeichnung]	Die Bezeichnung des Crons. Diese muss eindeutig sein.
section	Gibt an, ob es sich um einen mandantenspezifischen (site) oder globalen (global) Cron handelt.
name	Die Bezeichnung des Cron
script	Pfad zum PHP-Skript, welches die durchzuführenden Aktionen beinhaltet.
function	Funktionsname, welcher aufgerufen wird (Diese Funktion wird im Skript definiert)

Die cron.ini muss im Verzeichnis site/Mandant/admin/ liegen, um vom System eingebunden und verwendet zu werden. Sobald Sie den Cron definiert haben, erscheint dieser in der Dienstverwaltung.

<u>Site und Skin Liveupdate</u>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
System Liveupdate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Benutzer- und Rechteabgleich	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<u>Zugriffe berechnen</u>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Outputfilter erstellen

Ein Outputfilter dient dazu, den HTML-Code unmittelbar vor der Ausgabe nochmal zu filtern und nachträglich anzupassen.

Eigene Outputfilter werden im Verzeichnis `/site/Mandant/plugins/smarty/` angelegt. Erstellen Sie in diesem Verzeichnis eine entsprechende PHP-Datei `outputfilter.IhrFilter.php` (wobei *IhrFilter* der Name des eigenen neuen Outputfilters ist).

Beispiel

QuelltextPHP Code:

```
1. function smarty_outputfilter_IhrFilter($tpl_output, &$smarty)
2. {
3.     //Hier wird was nach Ihren wünschen gemacht
4.     return $tpl_output;
5. }
```

```
function smarty_outputfilter_IhrFilter($tpl_output, &$smarty)
{
    //Hier wird was nach Ihren wünschen gemacht
    return $tpl_output;
}
```

Innerhalb der Funktion können Sie `$tpl_output` beliebig verändern. Die Parameter `$tpl_output`, `&$smarty` sind für die Funktion zwingend erforderlich. immer der Funktion mitgeben müssen, und am ende "return `$tpl_output`" hinzufügen, um den Seiteninhalt wieder auszugeben.

Nachdem Sie einen Outputfilter wie oben beschrieben erstellt haben, muss dieser noch **eingebunden** werden.

Soll der Filter über einen gesamten Mandanten angewendet werden, wird er in `site/Mandant/index.php` aktiviert. Um den Outputfilter nur für bestimmte Seitentypen zu aktivieren, wird dieser in `site/IhrMandant/news/entry/index.php` freigegeben. Das Aktivieren des Outputfilters erfolgt über folgende Zeile:

QuelltextPHP Code:

```
1. // Eigenen Outputfilter aktivieren
2. $smarty->autoload_filters['output'][] = 'IhrFilter';
```

```
// Eigenen Outputfilter aktivieren
$smarty->autoload_filters['output'][] = 'IhrFilter';
```

Eigene Smarty-Funktionen erstellen

Gelegentlich werden in Templates eigene Smarty-Funktionen benötigt, die in der gewünschten Form nicht verfügbar sind.

Eigene Smarty-Funktionen können Verzeichnis `/site/IhrMandant/plugins/smarty/` jeweils als `function.meineFunktion.php` definiert werden (wobei *meineFunktion* für den Namen der Funktion steht).

Beispiel 1

QuelltextPHP Code:

```
1. // benötigtes Skript einbinden
2. require_once('smarty/plugins/arg_smarty_functions.php');
3.
4. // neue Funktion definieren
5. function smarty_function_meineFunktion($params, &$smarty)
6. {
7.     if(!$param['timestamp']) // wenn der Parameter 'timestamp' nicht gesetzt wurde
8.     {
9.         $param['timestamp'] = time(); // die aktuelle Zeit verwenden
10.    }
11.    $result = date("d.m.Y",$param['timestamp']);
12.
13.    // am Ende des Skripts wird an die var-Variable das Ergebnis übergeben
14.    $smarty->assign($params['var'], $result);
15. }
16.
// benötigtes Skript einbinden
require_once('smarty/plugins/arg_smarty_functions.php');
```

```
// neue Funktion definieren
function smarty_function_meineFunktion($params, &$smarty)
{
    if(!$param['timestamp']) // wenn der Parameter 'timestamp' nicht gesetzt wurde
    {
        $param['timestamp'] = time(); // die aktuelle Zeit verwenden
    }
    $result = date("d.m.Y",$param['timestamp']);

    // am Ende des Skripts wird an die var-Variable das Ergebnis übergeben
    $smarty->assign($params['var'], $result);
}
```

Alle an die Smarty-Funktion übergebenen Parameter können innerhalb der Funktion über das Array `$param` verwendet werden. Die Beispiel-Funktion gibt ein formatiertes Datum aus. Als Parameter kann ein gewünschter Unix-Zeitstempel angegeben werden.

QuelltextSmarty Code:

1. `{* Aufruf der Funktion *}`
2. `{meineFunktion timestamp=1261132764 var="datum_formatiert"}`
- 3.
4. `{* Verwenden des Ergebnisses *}`
5. Die Funktion gab als Ergebnis das Datum `{$datum_formatiert}` zurück.

```
{* Aufruf der Funktion *}
{meineFunktion timestamp=1261132764 var="datum_formatiert"}
```

```
{* Verwenden des Ergebnisses *}
Die Funktion gab als Ergebnis das Datum {datum_formatiert} zurück.
```

Beispiel 2

QuelltextPHP Code:

```
1. /**
2. * Smarty Plugin für Ausgabe von News-Einträgen
3. *
4. * Datei: site/Mandant/plugins/smarty/function.get_news.php
5. *
6. * Gibt alle Unterseite vom Typ "news/entry" aus.
7. * Beispiel: {get_news var=VARNAME page=PAGE-Object anzahl=ANZAHL}
8. */
9.
10. require_once('smarty/plugins/arg_smarty_functions.php');
11.
12. function smarty_function_get_news($params, &$smarty)
13. {
14.     // Ist Anzahl angegeben?
15.     $params['anzahl'] ? $params['anzahl'] : 10; // Standard = 10
16.
17.     // Kinder-Seiten auslesen
18.     $kinder = $params['page']->getChildren(array('where' => "type='news/entry'",array('limit' =>
19.         $params['anzahl'])););
20.
21.     // Ergebnis an entsprechende Var-Variable übergeben
22.     $smarty->assign($params['var'], $kinder);
23. }
24.
25. /**
26. * Smarty Plugin für Ausgabe von News-Einträgen
27. *
28. * Datei: site/Mandant/plugins/smarty/function.get_news.php
29. *
30. * Gibt alle Unterseite vom Typ "news/entry" aus.
31. * Beispiel: {get_news var=VARNAME page=PAGE-Object anzahl=ANZAHL}
32. */
33.
34. require_once('smarty/plugins/arg_smarty_functions.php');
35.
36. function smarty_function_get_news($params, &$smarty)
37. {
38.     // Ist Anzahl angegeben?
39.     $params['anzahl'] ? $params['anzahl'] : 10; // Standard = 10
40.
41.     // Kinder-Seiten auslesen
42.     $kinder = $params['page']->getChildren(array('where' => "type='news/entry'",array('limit' =>
43.         $params['anzahl'])););
44.
45.     // Ergebnis an entsprechende Var-Variable übergeben
46.     $smarty->assign($params['var'], $kinder);
47. }
```

```
}  
QuelltextSmarty Code:  
  
1. {* Funktionsaufruf im Template *}  
2. {get_news page=$page anzahl=5 var="news_eintraege"}  
3.  
4. {* Ausgabe der News *}  
5. {foreach from=$news_eintraege item="news"}  
6.     {$news->field.name} <br />  
7. {foreachelse}  
8.     Leider existieren zur Zeit keine News  
9. {/foreach}
```

```
{* Funktionsaufruf im Template *}  
{get_news page=$page anzahl=5 var="news_eintraege"}  
  
{* Ausgabe der News *}  
{foreach from=$news_eintraege item="news"}  
    {$news->field.name} <br />  
{foreachelse}  
    Leider existieren zur Zeit keine News  
{/foreach}
```

Zusätzliche Tabellen für Extrafelder

Anlegen von zusätzlichen Tabellen, um Extrafelder durchsuchbar zu machen.

Das EGOTEC CMS erlaubt das erstellen beliebig vieler Attribute pro Seite resp. Seitentyp. Die Speicherung dieser Extrafelder wird in einem Tabellenfeld vorgenommen. Die Tabellenstruktur wird nicht geändert. Es ist daher ganz einfach möglich, für einen Seitentyp neue Extrafelder zu definieren. Hierfür ist nicht einmal PHP notwendig, es genügt die Definition des Extrafelds auf einem Reiter im Adminbereich. Allerdings kann nach diesen Extrafeldern nicht gesucht und nicht sortiert werden. Um dies zu ermöglichen gibt es zwei Hooks im EGOTEC CMS.

Tabellen für Extrafelder erzeugen

Legen Sie in Ihrem Mandanten das Skript *create_table.php* an. Dieses Skript muss mindestens Die Methode *site_create_table_misc* beinhalten. Diese Methode wird beim Einspielen einer Datensicherung aufgerufen. Wenn Sie also nachträglich die Tabellenstruktur ändern möchten, so legen Sie eine Datensicherung an und spielen diese wieder ein, danach ist die neue Struktur, die Sie im Skript *create_table.php* festgelegt haben, vorhanden.

Beispiel (site/demo/admin/create_table.php)

```
<?php
/**
 * Die zusätzliche Tabelle demo_de_extra für die schnelle Suche in Extrafeldern erzeugen.
 */
function site_create_table_misc(Site $site, $params)
{
    $db = new_db_connection();
    $db->createTable(
        $site->pageTable.'_extra',
        array(
            'page_id' => 'bigint',
            'extra_key' => 'varchar(255)/*bin*/',
            'extra_int' => 'bigint',
            'extra_varchar' => 'varchar(255)',
            'extra_text' => 'text',
            'PRIMARY KEY' => 'page_id,extra_key',
            'KEY extra_int' => 'extra_int',
            'KEY extra_varchar' => 'extra_varchar(16)',
        )
    );
    $db->createTable('timestamp',array(
        'timestamp_id' => 'bigint',
        'page_id' => 'bigint',
        'start_date' => 'bigint',
        'end_date' => 'bigint',
        'userid' => 'varchar(32)',
        'PRIMARY KEY' => 'timestamp_id',
        'KEY page_id' => 'page_id',
        'KEY start_date' => 'start_date',
        'KEY end_date' => 'end_date',
        'KEY userid' => 'userid'
    ));
    $db->createTable(
```

```

$site->pageTable.'_buchungen',
array(
    'page_id' => 'bigint',
    'type' => 'varchar(255)',
    'begin' => 'bigint',
    'ende' => 'bigint',
)
);
}
?>

```

Ab Version 5.0.6 kann in `site/_global/admin/create_table.php` ein globales Skript hinterlegt werden, dass für jeden Mandanten vor dem Mandantenskript aufgerufen wird.

Die Methode muss dann allerdings `site_create_table_global(Site $site, $params)` heißen.

Daten in die Tabellen speichern

Pro Mandant oder auch pro Seitentyp kann das Skript `update.php` angelegt werden. Das Skript liegt entweder im Adminverzeichnis des Mandanten (`site/demo/admin/update.php`) oder im Adminverzeichnis des Seitentyps (`site/demo/page/admin/update.php`). Die Methode die aufgerufen wird heißt `site_update_misc` resp. `site_update_page_misc`.

Beispiel `site/demo/admin/update.php`

Dieses Beispiel speichert alle Extradfelder in eine Tabelle ab. Die Bedeutung der Felder im einzelnen:

- `page_id`
die Seitenid
- `extra_key`
der Name des Extradfeldes
- `extra_int`
der Wert des Extradfeldes als Zahl
- `extra_varchar`
`extra_text`
der Wert des Extradfeldes als Text

```

<?php
/**
 * Die zusätzliche Tabelle demo_de_extra für die schnelle Suche in Extradfeldern füllen.
 */
function site_update_misc(Page $page)
{
    $db = new_db_connection();
    if ($page->extra)
    {
        $extraTable = $page->getSite()->pageTable.'_extra';
        $db->delete(array( // Zuerst werden die alten Einträge gelöscht.
            'table' => $extraTable,
            'where' => 'page_id='.$page->field['id']
        ));
        $set = array(
            'page_id' => $page->field['id']

```

```

);
foreach ($page->extra as $key => $value)
{
    if (strlen($key)>255)
    {
        continue;
    }
    $set['extra_key'] = $key;
    if (is_array($value) || is_object($value))
    { // Es können nur Ganzzahlen und Zeichenketten abgespeichert werden.
        $value = serialize($value);
    }
    $set['extra_int'] = (integer)$value;
    $set['extra_varchar'] = mb_substr($value, 0, 255);
    $set['extra_text'] = $value;
    $db->insert(array(
        'table' => $extraTable,
        'set' => $set
    ));
}
}
?>

```

Mit diesem Skript werden immer alle Extrafelder in der Tabelle demo_de_extra gespeichert. Performanter ist das folgende Skript, das die Extradaten direkt in einzelne Felder einer Tabelle speichert.

Beispiel site/demo/ressource/belegung/update.php

```
<?php
```

```

/* #####
* ## Beim Speichern auch die Tabelle      ##
* ## <mandant>_<sprache>_buchungen aktualisieren  ##
* ##### */

```

```
function site_update_ressource_belegung_misc(Page $page)
```

```

{
    // alte Einträge löschen
    $db->delete(array(
        'table' => $extraTable,
        'where' => 'page_id='.$page->field['id']
    ));

    if ($page->field['deleted'] != 1)
    {
        $set = array(
            'page_id' => $page->field['id'],
            'type' => $page->field['type'],
            'begin' => $page->extra['begin'],
            'ende' => $page->extra['ende']
        );
        $db->insert(array(
            'table' => $extraTable,
            'set' => $set
        ));
    }
}

```

```
    });  
  }  
}  
?>
```

Ab Version 5.0.6 kann in *site/_global/admin/update.php* ein globales Skript hinterlegt werden, dass für jeden Mandanten vor dem Mandantenskript aufgerufen wird.

Die Methode muss dann allerdings *site_update_global(Site \$site, \$params)* heißen.

Liveserver Desklet anpassen

Für die jeweiligen Links auf dem Liveserver Desklet können einzelne Rechte vergeben werden. Somit können Links wie "Vollständiges Update" nur für einen gewünschten Benutzerkreis freigegeben werden.

Diese müssen per Hand in die globale conf.ini eingetragen werden.

Beispiel

```
[right_incremental_update_url]
group="ed93ecd168e89bb87dbaeac27ed2d914"
role="fe89d535dbb5bc967b8eeb8320c33b8b"
```

```
[right_complete_update_url]
group="ed93ecd168e89bb87dbaeac27ed2d914"
role="fe89d535dbb5bc967b8eeb8320c33b8b"
```

```
[right_system_update_url]
group="ed93ecd168e89bb87dbaeac27ed2d914"
role="fe89d535dbb5bc967b8eeb8320c33b8b"
```

```
[right_system_upgrade_url]
group="ed93ecd168e89bb87dbaeac27ed2d914"
role="fe89d535dbb5bc967b8eeb8320c33b8b"
```

```
[right_siteskin_update_url]
group="ed93ecd168e89bb87dbaeac27ed2d914"
role="fe89d535dbb5bc967b8eeb8320c33b8b"
```

```
[right_stats_update_url]
group="ed93ecd168e89bb87dbaeac27ed2d914"
role="fe89d535dbb5bc967b8eeb8320c33b8b"
```

In jedem Block wird jeweils eine Gruppe und Rolle angegeben, welche mindestens erfüllt sein muss, um den Link zu sehen.

Cacheverhalten

Das Egotec CMS unterscheidet 3 Cache-Einstellungen: **Browser, Proxy, CMS**.

Für jede dieser Einstellung wird jeweils ein Bit verwendet.

- Bit "1": Browser
- Bit "2": Proxy
- Bit "4": CMS

Die Kombination aus den Bits ergibt das endgültige Cacheverhalten.

Beispiele

- 7 = [1+2+4] Alles cachen
- 3 = [1+2]
- 0 = nichts cachen

Das Cachen verhindern

Um das Caching für einen Seitentypen zu verbieten, reicht eine Zeile in der "index.php" des Seitentyps:

```
$page->field['cache'] = 0;
```

Hierbei wird die Cache-Eigenschaft der aktuellen Seite auf "0" gesetzt, sobald die Seite geladen wird.

Das CMS gibt (über den Header) lediglich vor, wie eine CMS-Seite gecached werden soll. Dennoch können eigene Browser-Einstellungen das Brosercacherhalten beeinflussen.

Tipps und Tricks

JavaScript

IE IMG Node kopieren

Wenn man ein IMG-Node kopieren will, muss man zuerst das "src" Attribut setzen.

Der Internet Explorer setzt sonst alle anderen Attribute auf eine Standardeinstellung zurück.

IE Bug Flickering Hintergrundbilder

Wird im Internet Explorer mit Hintergrundbildern gearbeitet, so tritt manchmal ein unerwünschtes (und störendes) Flackern auf.

Mit JavaScript lässt sich diesem Effekt ein wenig entgegenwirken.

Folgender Befehl bewirkt, dass die Bilder im IE Cache gelagert werden und somit schneller abgerufen werden:

```
document.execCommand('BackgroundImageCache', false, true);
```

Klassen & Funktionen

Egotec setzt auf eine rein objektorientierte Programmierung. Demzufolge verfügt ein Entwickler über eine Hand voll Objekte, über welche sehr einfach verschiedene Aktionen und Verarbeitungen der Seiten realisiert werden können.

An dieser Stelle werden einzelne PHP Klassen und Funktionen von Egotec aufgelistet und näher erläutert.

Die 2 zentralen Objekte, die am häufigsten in der Entwicklung verwendet werden, sind **\$site** und **\$page**.

Umfangreichere Funktionen über verfügbaren Funktionen und Klassen finden die in der [API \(Application Programming Interface\)](#).

Site

Das \$site-Objekt stellt einen einzelnen Mandanten dar. Es wird bei jedem Seitenaufruf aus der Klasse *Site* instanziiert und besitzt Eigenschaften und Methoden, die den Umgang mit dem gesamten Mandanten betreffen.

Anbei finden Sie die gebräuchlichsten Funktionen, über welche Sie bereits 90% aller Arbeiten abdecken können. Eine vollständige Liste entnehmen Sie bitte aus "lib/base/Site.php".

destroyIDs

Beispiel

QuelltextPHP Code:

```
1. // IDs, welche gelöscht werden sollen.  
2. $ids = array(123,124,125,299,300);  
3.  
4. // gewählte IDs unwiderruflich löschen  
5. $site->destroyIDs($ids);
```

```
// IDs, welche gelöscht werden sollen.  
$ids = array(123,124,125,299,300);
```

```
// gewählte IDs unwiderruflich löschen  
$site->destroyIDs($ids);
```

Seiten, die auf diesem Weg gelöscht werden, werden aus ALLEN Tabellen entfernt, d.h. sie können NICHT wieder hergestellt werden.

getErrorPage

Beispiel

QuelltextPHP Code:

```
1. // Fehlerseite ermitteln
2. $fehlerseite = $site->getErrorPage();
```

```
// Fehlerseite ermitteln
$fehlerseite = $site->getErrorPage();
```

getLanguages

Beispiel

QuelltextPHP Code:

```
1. // Alle Sprachen des aktuellen Mandanten ermitteln
2. $sprachen = $site->getLanguages();
3.
4. // Sprachen durchlaufen und ausgeben
5. foreach ($sprachen as $sprache)
6. {
7.     echo $sprache . "<br />";
8. }
```

```
// Alle Sprachen des aktuellen Mandanten ermitteln
$sprachen = $site->getLanguages();
```

```
// Sprachen durchlaufen und ausgeben
foreach ($sprachen as $sprache)
{
    echo $sprache . "<br />";
}
```

getMediaSite

Beispiel

QuelltextPHP Code:

```
1. // Das Site-Objekt des zugehörigen Media-Mandanten
2. $mm_site = $site->getMediaSite();
3.
4. // Alle verfügbaren Bilder ermitteln
5. $alle_bilder = $mm_site->getPages(array("where" => "type = 'multimedia/image'"));
```

```
// Das Site-Objekt des zugehörigen Media-Mandanten
$mm_site = $site->getMediaSite();
```

```
// Alle verfügbaren Bilder ermitteln
$alle_bilder = $mm_site->getPages(array("where" => "type = 'multimedia/image'"));
```

getPage

Beispiel

QuelltextPHP Code:

1. <?php
2. \$kontakt = \$site->getPage(20); // Das PageObjekt der Kontaktseite
3. ?>

```
<?php
$kontakt = $site->getPage(20); // Das PageObjekt der Kontaktseite
?>
```

getPageId

Beispiel

QuelltextPHP Code:

1. // Die ID der Suchseite ermitteln
2. \$suche_id = \$site->getPageId("Suche");

```
// Die ID der Suchseite ermitteln
$search_id = $site->getPageId("Suche");
```

Wenn Sie die ID einer bestimmten Seite ermitteln möchten, wird empfohlen über den Typ der Seite zu suchen.

getPages

Eine Liste der möglichen Attribute für den Parameter query und param werden auf dieser Seite weiter unten erläutert.

Rückgabe

Alle internen Funktionen, die eine Suche nach Seiten im CMS darstellen, liefern als Rückgabewert immer ein **Page_Iterator**-Objekt zurück. Dieses kann anschließend über eine foreach-Schleife durchlaufen werden. Nähere Informationen zum Page_Iterator finden Sie [hier](#).

query

Name Beschreibung

where Zusätzliche Datenbankabfragen z.B. auf den Typ oder ID der Seite. *Beispiel: array("where" => "type = 'news/entry'")*

order Sortierung der Ergebnisse. Mögliche Werte sind z.B. "name asc", "title desc", "order_field DESC". Wurde kein Wert angegeben, wird die aktuell eingestellte Sortierung aus dem Adminbereich verwendet.

fields Kommaseparierte Liste der Spalten, die bei der Abfrage gespeichert werden sollen (z.B. "id,title,name"). Standardmäßig ist hier "" für alle Werte gesetzt. Werden z.B. nur die IDs der Ergebnisse gebraucht, kann durch eine entsprechende Angabe ein Performance-Vorteil erreicht werden.

param

Sofern nicht anders angegeben werden parameter mit dem Wert "1" aktiviert. Beispiel: `array("no_navhide" => 1);`

Name	Beschreibung
no_nav_hide	Seiten mit gesetztem Anzeige-Flag "Nicht in Navigation anzeigen" ausschließen.
no_intranet	Seiten mit gesetztem Intranet-Flag "Seite nicht auf den Liveserver übertragen" ausschließen.
intranet	Nur Seiten anzeigen, die das Intranet-Flag "Seite nicht auf den Liveserver übertragen" besitzen.
search	Seiten mit gesetztem Suche-Flag "Seite von der Suche ausschließen" nicht berücksichtigen.
fulltext	Eine Volltextsuchanfrage durchführen. Wird z.B. für die Suche benötigt.
inactive	Sowohl aktive als auch inaktive Seiten suchen.
deleted	Ist standardmäßig auf "1". Bei "0" werden auch gelöschte Seiten mit angezeigt.
lang	Die gewünschte Sprache der Seiten (z.B. "de" für deutsch). Wird keine Sprache übergeben, was meist der Fall ist, dann wird die Sprache des \$site Objekts verwendet.
auth_or	Bei "1=1" werden keine Rechte berücksichtigt. D.h. es werden auch Seiten ausgegeben, auf welche der aktuell angemeldete Besucher keine Bearbeitungsrechte bzw. Ansichtsrechte hat.
has_children	Bei den einzelnen Ergebnissen (Page-Objekten) speichern, ob jeweils Kinder verfügbar sein. Das Feld has_children ist null, wenn kein Kind existiert, ansonsten enthält es eine Zahl >0

Die Funktion `getPages` bildet eine basis-Funktion. Alle verfügbaren Auslesefunktionen wie `getChildren` oder `getParents` greifen tief im System auf die `getPages` Funktion zurück. Die Parameter **query** und **param** werden weitergereicht und gelten damit auch für alle andere Funktionen.

Beispiel 1

QuelltextPHP Code:

```
1. // Alle News- und Eventseinträge holen und nach Name sortieren (Ausführliche Schreibweise)
2. $query = array(
3.   "where" => "type = 'news/entry' OR type = 'events/entry'",
4.   "order" => "name DESC"
5. );
6. $param = array(
7.   "no_nav_hide" => 1, // "Nicht in Navigation anzeigen" beachten,
8.   "deleted" => 0 // auch gelöschte Seiten anzeigen
9. );
10. // Abfrage im aktuellen Mandanten durchführen
11. $new_events = $site->getPages($query,$param);
12.
13. // Kürzere Schreibweise
14. $news_events = $site->getPages(
15.   array(
16.     "where" => "type = 'news/entry' OR type = 'events/entry'",
17.     "order" => "name DESC"
18.   ),
19.   array(
20.     "no_nav_hide" => 1 // "Nicht in Navigation anzeigen" beachten
21.   )
22. );
```

```

// Alle News- und Eventseinträge holen und nach Name sortieren (Ausführliche Schreibweise)
$query = array(
    "where" => "type = 'news/entry' OR type = 'events/entry'",
    "order" => "name DESC"
);
$params = array(
    "no_nav_hide" => 1, // "Nicht in Navigation anzeigen" beachten,
    "deleted" => 0 // auch gelöschte Seiten anzeigen
);
// Abfrage im aktuellen Mandanten durchführen
$new_events = $site->getPages($query,$params);

// Kürzere Schreibweise
$new_events = $site->getPages(
    array(
        "where" => "type = 'news/entry' OR type = 'events/entry'",
        "order" => "name DESC"
    ),
    array(
        "no_nav_hide" => 1 // "Nicht in Navigation anzeigen" beachten
    )
);

```

Beispiel 2

QuelltextPHP Code:

1. // Such-Seiten im aktuellen Mandanten suchen
2. `$suchseiten = $site->getPages(array("where" => "type = 'search'"));`
- 3.
4. // Den ersten Treffer als separates PageObjekt speichern
5. `$suchseite = $suchseiten->next();`
- 6.
7. // Namen der Sucheseite ausgeben
8. `echo $suchseite->field['name'];`

```

// Such-Seiten im aktuellen Mandanten suchen
$suchseiten = $site->getPages(array("where" => "type = 'search'"));

```

```

// Den ersten Treffer als separates PageObjekt speichern
$suchseite = $suchseiten->next();

```

```

// Namen der Sucheseite ausgeben
echo $suchseite->field['name'];

```

getPageUrl

Beispiel

QuelltextPHP Code:

1. // URL auf die Startseite erzeugen
2. `$url = $site->getPageUrl(1);`

```
// URL auf die Startseite erzeugen
$url = $site->getPageUrl(1);
```

Wenn Sie direkt im Template an einer Stelle eine URL erzeugen möchten, wird empfohlen die Smarty-Funktion `{page_url id=1}` zu verwenden.

getRoot

Beispiel

QuelltextPHP Code:

1. // Startseite als Page-Objekt auslesen
2. `$startseite = $site->getRoot();`
- 3.
4. // Name der Startseite ausgeben
5. `echo $startseite->field['name'];`

```
// Startseite als Page-Objekt auslesen
$startseite = $site->getRoot();
```

```
// Name der Startseite ausgeben
echo $startseite->field['name'];
```

Ego_System

Die "Ego_System"-Klasse dient als Container für verschieden zentrale Methoden. Diese können statisch direkt über `Ego_System::funktionsname()` aufgerufen werden.

Im Folgenden finden Sie weiter Informationen zu den einzelnen Funktionen.

clearPageLocks

Beispiel

QuelltextPHP Code:

```
1. Ego_System::clearPageLocks();
```

```
Ego_System::clearPageLocks();
```

clearTypeCache

Beispiel

QuelltextPHP Code:

```
1. <?php
2.     Ego_System::clearTypeCache();
3. ?>
```

```
<?php
    Ego_System::clearTypeCache();
?>
```

file_exists

Beispiel

QuelltextPHP Code:

```
1. <?php
2.     $file = "/pfad/zur/datei/test.php";
3.     if(Ego_System::file_exists($file))
4.     {
5.         include_once($file);
6.     }
7. ?>
```

```
<?php
    $file = "/pfad/zur/datei/test.php";
    if(Ego_System::file_exists($file))
    {
        include_once($file);
    }
```

?>

getAllSites

Beispiel

QuelltextPHP Code:

1. <?php
2. \$mandanten = Ego_System::getAllSites();
3. ?>

```
<?php
$mandanten = Ego_System::getAllSites();
?>
```

isEmptyContent

Funktion um zu prüfen, ob ein Content ohne Textinhalt ist (leere HTML-Tags u.ä. werden nicht aufgewertet)

Beispiel

QuelltextPHP Code:

1. <?php
2. if(Ego_System::isEmptyContent(\$page->field['content'])) // prüft, ob der Inhalt wirklich leer ist
3. {
4. \$page->field['content'] = "";
5. }
6. ?>

```
<?php
if(Ego_System::isEmptyContent($page->field['content'])) // prüft, ob der Inhalt wirklich leer ist
{
    $page->field['content'] = "";
}
?>
```

mkdir

Beispiel

QuelltextPHP Code:

1. <?php
2. \$pfad = 'test/verzeichnis/zum/ziel';
- 3.
4. // Verzeichnis(se) anlegen
5. Ego_System::mkdir(\$pfad);
6. ?>

```
<?php
    $pfad = 'test/verzeichnis/zum/ziel';

    // Verzeichnis(se) anlegen
    Ego_System::mkdir($pfad);
?>
```

urltopage

Beispiele

QuelltextPHP Code:

1. <?php
2. \$url = "index.php?id=162&site=demo&lang=de";
3. \$p = Ego_System::urltopage(\$url); // Aus der URL das PageObjekt gewinnen
- 4.
5. echo \$p->field['content']; // Den Inhalt der Seite ausgeben
6. ?>

```
<?php
    $url = "index.php?id=162&site=demo&lang=de";
    $p = Ego_System::urltopage($url); // Aus der URL das PageObjekt gewinnen

    echo $p->field['content']; // Den Inhalt der Seite ausgeben
?>
```

Page_Iterator

Sobald z.B. über die Funktion `getChildren()` mehrere Kinder einer Seite ausgelesen wurden, werden diese in einem `Page_Iterator` gespeichert. Dieser stellt eine Art "Container" für Page-Objekte dar. Mit Hilfe einer Schleife lassen sich anschließend die Ergebnisse einfach durchlaufen und ausgeben.

current

Beispiel

QuelltextPHP Code:

```
1. <?php
2. // Alle Seiten auslesen
3. $alle_seiten = $site->getPages();
4.
5. // Das aktuelle (erste) Element ausgeben
6. $p = $alle_seiten->current();
7.
8. // Den Namen der Seite anzeigen
9. echo $p->field['name'];
10. ?>
```

```
<?php
// Alle Seiten auslesen
$alle_seiten = $site->getPages();

// Das aktuelle (erste) Element ausgeben
$p = $alle_seiten->current();

// Den Namen der Seite anzeigen
echo $p->field['name'];
?>
```

key

Beispiel

QuelltextPHP Code:

```
1. <?php
2. // Alle Seiten auslesen
3. $alle_seiten = $site->getPages();
4.
5. // Den aktuellen Schlüssel zurückgeben
6. echo $alle_seiten->key();
7. ?>
```

```
<?php
// Alle Seiten auslesen
```

```

$alle_seiten = $site->getPages();

// Den aktuellen Schlüssel zurückgeben
echo $alle_seiten->key();
?>

```

next

Beispiel

QuelltextPHP Code:

```

1. <?php
2. // Alle Seiten auslesen
3. $seiten = $site->getPages();
4.
5. // Seiten in einer Schleife durchlaufen
6. while ($p = $seiten->next())
7. {
8.     echo $p->field['name'].'<br />';
9. }
10. ?>

```

```

<?php
// Alle Seiten auslesen
$seiten = $site->getPages();

// Seiten in einer Schleife durchlaufen
while ($p = $seiten->next())
{
    echo $p->field['name'].'<br />';
}
?>

```

Diese Funktion reagiert genauso wie **nextPage()**.

nextPage

Beispiel

QuelltextPHP Code:

```

1. <?php
2. $kinder = $page->getChildren(); // Kinder auslesen
3. while($p = $kinder->nextPage())
4. {
5.     // IDs der Kinder ausgeben
6.     echo $p->field['id']. '<br />';
7. }
8. ?>

```

```

<?php
    $kinder = $page->getChildren(); // Kinder auslesen
    while($p = $kinder->nextPage())
    {
        // IDs der Kinder ausgeben
        echo $p->field['id']. '<br />';
    }
?>

```

numRecords

Beispiel

QuelltextPHP Code:

1. <?php
2. // Kinder auslesen
3. \$kinder = \$page->getChildren();
- 4.
5. // und auf Verfügbarkeit prüfen
6. if(\$kinder->numRecords() > 0)
7. {
8. echo 'Die aktuelle Seite besitzt Kinder!';
9. }
10. ?>

```

<?php
// Kinder auslesen
$kinder = $page->getChildren();

// und auf Verfügbarkeit prüfen
if($kinder->numRecords() > 0)
{
    echo 'Die aktuelle Seite besitzt Kinder!';
}
?>

```

Page

Die Klassen *Page* beschreibt eine einzelne Seite des CMS. Bei jedem Seitenaufruf wird aus ihr das entsprechende *\$page*-Objekt zur Verfügung gestellt, mit welchem auf Eigenschaften und Methoden einer einzelnen Seite zugegriffen werden kann.

Wählen Sie die gewünschte Funktion eines Page-Objekts aus der linken Navigation aus.

addChild

Beispiel

QuelltextPHP Code:

1. <?php
2. // Die Seite mit der ID 10 unter der aktuellen Seite einhängen.
3. \$page->addChild(10);
4. ?>

```
<?php
// Die Seite mit der ID 10 unter der aktuellen Seite einhängen.
$page->addChild(10);
?>
```

addParent

Beispiel

QuelltextPHP Code:

1. <?php
- 2.
3. /* Die Seite mit der ID 15 zusätzlich unter die Startseite einhängen */
4. \$p = \$site->getPage(15); // PageObjekt der Seite auslesen
- 5.
6. /* Eltern aktualisieren */
7. \$p->addParent(1);
8. ?>

```
<?php

/* Die Seite mit der ID 15 zusätzlich unter die Startseite einhängen */
$p = $site->getPage(15); // PageObjekt der Seite auslesen

/* Eltern aktualisieren */
$p->addParent(1);
?>
```

cleanEmptyContent

Beispiel

QuelltextPHP Code:

1. // den Inhalt der aktuellen Seite vollständig leeren
2. `$page->cleanEmptyContent();`

```
// den Inhalt der aktuellen Seite vollständig leeren
$page->cleanEmptyContent();
```

copyTo

Eine Liste der möglichen Attribute für den Parameter `query` und `param` werden unter [getPages](#) erläutert.

Beispiel

QuelltextPHP Code:

1. // Die aktuelle Seite kopieren (unter die ID 430)
2. `$page->copyTo(430);`

```
// Die aktuelle Seite kopieren (unter die ID 430)
$page->copyTo(430);
```

delete

Beispiel

QuelltextPHP Code:

1. // Die aktuelle Seite + alle Unterseiten löschen
2. `$page->delete(true);`

```
// Die aktuelle Seite + alle Unterseiten löschen
$page->delete(true);
```

Diese Funktion "löscht" Seiten, d.h. sie befinden sich anschließend im Papierkorb und können von dort wiederhergestellt werden.

delParent

Beispiel

QuelltextPHP Code:

1. <?php
2. /* Die Seite mit der ID 15 (liegt unter der Startseite ID1) aushängen */
3. \$p = \$site->getPage(15); // PageObjekt der Seite auslesen
- 4.
5. /* Eltern aktualisieren */
6. \$p->delParent(1);
7. ?>

```
<?php
/* Die Seite mit der ID 15 (liegt unter der Startseite ID1) aushängen */
$p = $site->getPage(15); // PageObjekt der Seite auslesen

/* Eltern aktualisieren */
$p->delParent(1);
?>
```

destroy

Beispiel

QuelltextPHP Code:

1. // Löschseite bestimmen (ID = 199)
2. \$to_delete = \$site->getPage(199);
- 3.
4. // Die gewünschte Seite + Unterseiten zerstören (= endgültig löschen)
5. \$to_delete->destroy(true,true);

```
// Löschseite bestimmen (ID = 199)
$to_delete = $site->getPage(199);
```

```
// Die gewünschte Seite + Unterseiten zerstören (= endgültig löschen)
$to_delete->destroy(true,true);
```

Diese Funktion zerstört eine Seite endgültig. D.h. sie kann nicht mehr über den Papierkorb wiederhergestellt werden.

getAncestors

Eine Liste der möglichen Attribute für den Parameter query und param werden unter [getPages](#) erläutert.

Beispiel

QuelltextPHP Code:

1. // Alle Vorfahren-Elemente (beliebiger Generation) ermitteln
2. \$vorfahren = \$page->getAncestors();

```
// Alle Vorfahren-Elemente (beliebiger Generation) ermitteln
$vorfahren = $page->getAncestors();
```

Unter "Vorfahren" versteht die Funktion die jeweils direkt übergeordnete Seite.

Eine Seite "Produkte" die z.B. unter "Startseite->Service->Dokumente" eingehängt ist, würde über diese Funktion einen Page_Iterator mit 3 Page-Objekten (Startseite, Service, Dokumente) erzeugen.

getChildren

Eine Liste der möglichen Attribute für den Parameter query und param werden unter [getPages](#) erläutert.

Beispiel

QuelltextPHP Code:

```
1. <?php
2. // Abfrage Parameter definieren
3. $query = array(
4.     'where' => "type = 'news/entry'", // muss ein News/Eintrag sein
5.     'limit' => 5, // limitierte Anzahl: 5
6.     'order' => 'name desc' // sortiert nach name abwärts
7. );
8.
9. // Alle Kinder auslesen
10. $page->getChildren($query);
11. ?>
```

```
<?php
// Abfrage Parameter definieren
$query = array(
    'where' => "type = 'news/entry'", // muss ein News/Eintrag sein
    'limit' => 5, // limitierte Anzahl: 5
    'order' => 'name desc' // sortiert nach name abwärts
);

// Alle Kinder auslesen
$page->getChildren($query);
?>
```

getDescendants

Eine Liste der möglichen Attribute für den Parameter query und param werden unter [getPages](#) erläutert.

Beispiel

QuelltextPHP Code:

```
1. // Alle Unterseiten vom Typ News-Eintrag ermitteln
2. $news = $page->getDescendants(array('where' => "type = 'news/entry'"));
```

```
// Alle Unterseiten vom Typ News-Eintrag ermitteln
$news = $page->getDescendants(array('where' => "type = 'news/entry'"));
```

getKeywords

Wurde keine Sprache übergeben, wird die aktuelle Sprache der Seite verwendet.

Beispiel

QuelltextPHP Code:

```
1. <?php
2.
3. // Alle englischen Keywords der aktuellen Seite anzeigen
4. $keywords = $page->getKeywords('en');
5.
6. ?>
```

```
<?php
```

```
// Alle englischen Keywords der aktuellen Seite anzeigen
$keywords = $page->getKeywords('en');
```

```
?>
```

getLanguagePage

Eine Liste der möglichen Attribute für den Parameter param werden unter [getPages](#) erläutert.

Beispiel

QuelltextPHP Code:

```
1. // Die englische Seite der aktuellen Sprache ermitteln
2. $page_en = $page->getLanguagePage("en");
3.
4. // Den Titel der englischen Seite ausgeben
5. echo $page_en->field['name'];
```

```
// Die englische Seite der aktuellen Sprache ermitteln
$page_en = $page->getLanguagePage("en");
```

```
// Den Titel der englischen Seite ausgeben
echo $page_en->field['name'];
```

getParents

Eine Liste der möglichen Attribute für den Parameter query und param werden unter [getPages](#) erläutert.

Beispiel

QuelltextPHP Code:

```
1. <?php
2.
3. // die Eltern der aktuellen Seite auslesen
```

```

4. $parents = $page->getParents();
5.
6. // den ersten Treffer als PageObjekt speichern
7. $parent = $parents->nextPage();
8.
9. // den Name des Eltern-Elements ausgeben
10. echo $parent->field['name'];
11.
12. ?>

```

<?php

```

// die Eltern der aktuellen Seite auslesen
$parents = $page->getParents();

// den ersten Treffer als PageObjekt speichern
$parent = $parents->nextPage();

// den Name des Eltern-Elements ausgeben
echo $parent->field['name'];

```

?>

Die Funktion kann auch **mehrere Eltern-Seiten** ausgegeben, wenn eine Seite mehrfach eingehängt wurde. Die Reihenfolge der Elemente ist dabei zufällig.

getPath

Eine Liste der möglichen Attribute für den Parameter query und param werden unter [getPages](#) erläutert.

Beispiel

QuelltextPHP Code:

```

1. // Den aktuellen Pfad ermitteln
2. $pfad = $page->getPath(true,array(),2);
3.
4. // aktuellen Pfad ausgeben
5. foreach ($pfad as $p)
6. {
7.     echo $p->field['name'] . ' => ';
8. }

```

```

// Den aktuellen Pfad ermitteln
$pfad = $page->getPath(true,array(),2);

```

```

// aktuellen Pfad ausgeben
foreach ($pfad as $p)
{
    echo $p->field['name'] . ' => ';
}

```

getSiblings

Eine Liste der möglichen Attribute für den Parameter query und param werden unter [getPages](#) erläutert.

Zusätzliche Parameter für "param"

Name Beschreibung

next Nur die nachfolgenden Geschwister-Seiten ab der aktuellen Seite anzeigen.

reverse Die Richtung ändern. Ist der Wert "next" gesetzt und der Parameter "reverse" ebenfalls aktiv, werden nur die vorhergehenden "linken" Geschwister anzeigen.

Beispiel

QuelltextPHP Code:

1. // alle Geschwister der aktuellen Seite ermitteln
2. \$geschwister = \$page->getSiblings();

```
// alle Geschwister der aktuellen Seite ermitteln
$geschwister = $page->getSiblings();
```

getSite

Beispiel

QuelltextPHP Code:

1. <?php
2. \$aktuelle_site = \$page->getSite();
3. ?>

```
<?php
$aktuelle_site = $page->getSite();
?>
```

getUrl

Beispiel

QuelltextPHP Code:

1. <?php

```

2.
3. // Alle Parameter sammeln
4. $url_parameter = array(
5.     'modus' => 'news',
6.     'font' => 13
7. );
8.
9. // URL mit $_GET-Parametern erzeugen
10. $meine_url = $page->getUrl($url_parameter);
11. ?>

```

```
<?php
```

```

// Alle Parameter sammeln
$url_parameter = array(
    'modus' => 'news',
    'font' => 13
);

```

```

// URL mit $_GET-Parametern erzeugen
$meine_url = $page->getUrl($url_parameter);
?>

```

getUser

Beispiel

QuelltextPHP Code:

```

1. <?php
2. // Benutzer auslesen
3. $usr1 = $page->getUser();
4.
5. // Vor- und Nachname ausgeben
6. echo $usr1['extra']['vorname'] . " " . $usr1['extra']['name'];
7. ?>

```

```
<?php
```

```

// Benutzer auslesen
$usr1 = $page->getUser();

```

```

// Vor- und Nachname ausgeben
echo $usr1['extra']['vorname'] . " " . $usr1['extra']['name'];
?>

```

Hinweis: Damit in diesem Beispiel Vor- und Nachname ausgegeben werden können, müssen entsprechenden Einstellungen beim Benutzerprofil hinterlegt sein.

hasMultiParents

Rückgabewert ist entweder *true* oder *false*.

Beispiel

QuelltextPHP Code:

```
1. if ($page->hasMultiParents())
2. {
3.   echo "Die aktuelle Seite ist unter mehreren Seiten eingehängt";
4. }
```

```
if ($page->hasMultiParents())
{
    echo "Die aktuelle Seite ist unter mehreren Seiten eingehängt";
}
```

hasRights

Beispiel

```
?if ($page->hasRights('desktop'))
{
    echo "Der aktuelle Benutzer darf diese Seite bearbeiten";
}
```

move

Beispiel

QuelltextPHP Code:

```
1. // Eine falsch eingehängte Seite definieren
2. $falsch_page = $site->getPage(200);
3.
4. // ...und unter die Startseite einhängen
5. $falsch_page->move(166,1);
```

```
// Eine falsch eingehängte Seite definieren
$falsch_page = $site->getPage(200);
```

```
// ...und unter die Startseite einhängen
$falsch_page->move(166,1);
```

Um eine Seite korrekt zu verschieben, muss bei dieser Funktion vorher die Eltern-ID ermittelt werden (siehe Funktion *getParents*).

newChild

Beispiel

QuelltextPHP Code:

```
1. // Tabellenfelder sammeln
2. $new_field = array(
3.   'name'   => "Meine neue Seite",
4.   'title'  => "Meine neue Seite",
5.   'type'   => 'page',
6.   'inactive' => '0',
7.   'nav_hide' => '1'
8. );
9.
10. // Extrafelder sammeln
11. $new_extra = array(
12.   'max_anzeige' => "10",
13.   'zusatz_beschreibung' => "Eine komplette neue Seite"
14. );
15.
16. // Neue Seite erzeugen
17. $neu = $page->newChild($new_field,$new_extra);
18.
19. // Die neu erstelle Seite kann anschließend sofort verwendet werden
20. echo $neu->field['name'];
```

```
// Tabellenfelder sammeln
$new_field = array(
  'name'   => "Meine neue Seite",
  'title'  => "Meine neue Seite",
  'type'   => 'page',
  'inactive' => '0',
  'nav_hide' => '1'
);
```

```
// Extrafelder sammeln
$new_extra = array(
  'max_anzeige' => "10",
  'zusatz_beschreibung' => "Eine komplette neue Seite"
);
```

```
// Neue Seite erzeugen
$neu = $page->newChild($new_field,$new_extra);
```

```
// Die neu erstelle Seite kann anschließend sofort verwendet werden
echo $neu->field['name'];
```

setRightsArray

Beispiel

QuelltextPHP Code:

```
1. // die neuen Rechte
2. $new_rights = array();
```

```

3.
4. // Ansichtsrechte
5. $new_rights['view'][] = array(
6.   'group_id' => 'fd93ecd168e89bb87dbaeac27ed2d911', // Gruppen-ID
7.   'role_id' => 'de867a054c19774d7fee0ae917d17f27' // Rollen-ID
8. );
9.
10. // Bearbeitenrechte
11. $new_rights['edit'][] = array(
12.   'group_id' => 'fd93ecd168e89bb87dbaeac27ed2d911', // Gruppen-ID
13.   'role_id' => 'de867a054c19774d7fee0ae917d17f27' // Rollen-ID
14. );
15.
16. // Gruppen-Rollen auf die aktuelle Seite anwenden
17. $page->setRightsArray($new_rights);

```

```

// die neuen Rechte
$new_rights = array();

```

```

// Ansichtsrechte
$new_rights['view'][] = array(
  'group_id' => 'fd93ecd168e89bb87dbaeac27ed2d911', // Gruppen-ID
  'role_id' => 'de867a054c19774d7fee0ae917d17f27' // Rollen-ID
);

```

```

// Bearbeitenrechte
$new_rights['edit'][] = array(
  'group_id' => 'fd93ecd168e89bb87dbaeac27ed2d911', // Gruppen-ID
  'role_id' => 'de867a054c19774d7fee0ae917d17f27' // Rollen-ID
);

```

```

// Gruppen-Rollen auf die aktuelle Seite anwenden
$page->setRightsArray($new_rights);
Beispiele für Rechte sind: 'view','edit','release','remove' und 'child'.

```

setUsersArray

Beispiel

QuelltextPHP Code:

```

1. // Benutzer-Array sammeln
2. $new_users['edit'][] = array('user_id' => $auth->getId());
3. $new_users['remove'][] = array('user_id' => $auth->getId());
4.
5. // Benutzer für die aktuelle Seite eintragen
6. $page->setUsersArray($new_users);

```

```

// Benutzer-Array sammeln
$new_users['edit'][] = array('user_id' => $auth->getId());
$new_users['remove'][] = array('user_id' => $auth->getId());

```

```

// Benutzer für die aktuelle Seite eintragen
$page->setUsersArray($new_users);

```

Über "\$auth->getid()" wird die BenutzerID des aktuell angemeldeten Benutzers ausgelesen. In dem Beispiel wird diese ID zunächst in einem Array in Kombination mit dem gewünschten Recht gesammelt und anschließend einer Seite zugewiesen.

undelete

Beispiel

QuelltextPHP Code:

```
1. <?php
2.
3. // gelöschte Seiten finden
4. $geloschte_seiten = $site->getPages(array(),array('inactive' => 1));
5.
6. // und wieder herstellen
7. foreach ($geloeschte_seiten as $s)
8. {
9.     $s->undelete();
10. }
11. ?>
```

<?php

```
// gelöschte Seiten finden
$geloschte_seiten = $site->getPages(array(),array('inactive' => 1));
```

```
// und wieder herstellen
foreach ($geloeschte_seiten as $s)
{
    $s->undelete();
}
```

?>

unlinkFrom

Beispiel

QuelltextPHP Code:

```
1. <?php
2. // PageObjekt der Seite mit ID=20 holen
3. $meine_seite = $site->getPage(20);
4.
5. // und diese Seite aus einem Ast entfernen
6. $meine_seite->unlinkFrom(2);
7. ?>
```

<?php

```
// PageObjekt der Seite mit ID=20 holen
$meine_seite = $site->getPage(20);
```

```
// und diese Seite aus einem Ast entfernen
$meine_seite->unlinkFrom(2);
?>
```

update

Beispiel

QuelltextPHP Code:

```
1. // Aktuelle Einstellungen und extra-Daten
2. $field = $page->field;
3. $extra = $page->extra;
4.
5. // Einstellungen ändern
6. $field["name"] = "Neuer Name";
7. $extra["max_news"] = "100";
8.
9. // Einstellungen speichern
10. $page->update(array("field" => $field, "extra" => $extra));
```

```
// Aktuelle Einstellungen und extra-Daten
$field = $page->field;
$extra = $page->extra;
```

```
// Einstellungen ändern
$field["name"] = "Neuer Name";
$extra["max_news"] = "100";
```

```
// Einstellungen speichern
$page->update(array("field" => $field, "extra" => $extra));
```

updateChildren

Beispiel

QuelltextPHP Code:

```
1. // Liste der neuen Unterseiten
2. $unterseiten_ids = array(45,56,123,345);
3.
4. // Unterseiten der aktuellen Seite aktualisieren
5. $page->updateChildren($unterseiten_ids);
```

```
// Liste der neuen Unterseiten
$unterseiten_ids = array(45,56,123,345);
```

```
// Unterseiten der aktuellen Seite aktualisieren
$page->updateChildren($unterseiten_ids);
```

Alle bereits eingehängten Unterseiten werden durch die neuen ersetzt.

updateExtra

Beispiel

QuelltextPHP Code:

```
1. // Wichtig! Das aktuelle extra-Feld sichern
2. $extra = $page->extra;
3.
4. // Neuen Wert einbauen / ändern
5. $extra['mein_wert'] = "test1";
6. $extra['mein_wert2'] = "test2";
7.
8. // Die aktuelle Seite mit dem neuen (erweiterten) extra-Feld aktualisieren
9. $page->updateExtra($extra);
```

```
// Wichtig! Das aktuelle extra-Feld sichern
$extra = $page->extra;
```

```
// Neuen Wert einbauen / ändern
$extra['mein_wert'] = "test1";
$extra['mein_wert2'] = "test2";
```

```
// Die aktuelle Seite mit dem neuen (erweiterten) extra-Feld aktualisieren
$page->updateExtra($extra);
```

Diese Funktion überschreibt das komplette extra-Feld der entsprechenden Seite. Möchten Sie also lediglich neue Felder hinzufügen /ändern muss das ursprüngliche extra-Feld erweitert und mitgegeben werden.

updateField

Beispiel

QuelltextPHP Code:

```
1. // Wichtig! Das aktuelle field-Feld sichern
2. $field = $page->field;
3.
4. // Neuen Wert einbauen / ändern
5. $field['name'] = "Neuer Name";
6. $field['title'] = "Neuer Titel";
7.
8. // Die aktuelle Seite mit dem geänderten field-Feld aktualisieren
9. $page->updateField($field);
```

```
// Wichtig! Das aktuelle field-Feld sichern
$field = $page->field;
```

```
// Neuen Wert einbauen / ändern
```

```
$field['name'] = "Neuer Name";  
$field['title'] = "Neuer Titel";
```

```
// Die aktuelle Seite mit dem geänderten field-Feld aktualisieren  
$page->updateField($field);
```

updateParents

Beispiel

QuelltextPHP Code:

```
1. <?php  
2.  
3. // Liste der neuen Eltern-IDs  
4. $neue_eltern = array(1,50,1023);  
5.  
6. // Die aktuelle Seite neu einhängen  
7. $page->updateParents($neue_eltern);  
8.  
9. ?>
```

```
<?php
```

```
// Liste der neuen Eltern-IDs  
$neue_eltern = array(1,50,1023);  
  
// Die aktuelle Seite neu einhängen  
$page->updateParents($neue_eltern);
```

```
?>
```

Beispiele & Vorlagen

In diesem Bereich finden Sie Vorlagen und Beispiele für das Entwicklerhandbuch.

navigation.ini

Über die **navigation.ini** werden dem System neue Reiter kenntlich gemacht. Anbei finden Sie alle erdenklichen Beispiele.

Erweiterter Information-Reiter

[information]

url = "info.php?tpl_name=information.html"

title = Information

[content]

url = edit.php

title = "Inhalt "

Diese zwei Einträge werden standardmäßig in einer navigation.ini (falls eine vom Benutzer angelegt wurde) verwendet. Über den Parameter *?tpl_name=information.html* kann der Information-Reiter erweitert werden. Hierbei muss jedoch eine information.html im admin-Verzeichnis vorliegen.

Soll der Standard-Information-Reiter angezeigt werden, kann das "?tpl_name=information.html" einfach wegelassen werden.

Zweiter Inhaltsreiter

[content_2]

url = "edit.php?content=content_2"

title = "Inhalt 2 "

Über den Parameter content wird der Name der Variable angegeben, unter welcher der eingepflegte Inhalt gespeichert werden soll. Der Editor-Inhalt solcher Reiter wird immer im extra-Feld der Seite gespeichert (im Beispiel unter "content_2").

Eigener Reiter

```
[mein_eigener_Reiter]
```

```
url = "extra.php?tpl_name=page/admin/meinReiter.html&script_name=page/admin/meinReiter.php"
```

```
title = "Mein Reiter"
```

```
ignore_types = "type1, type2..."
```

```
mandatory = "Inputfeld_name1, Inputfeldname2...."
```

ignore_types: kommaseparierte Liste mit den Seitentypen bei denen diese Reiter ausgeblendet werden (optionale Eigenschaft)

ab Version 5.2

mandatory: kommaseparierte Liste mit den Namen der Pflichtfelder auf diesem Reiter (optionale Eigenschaft)

ab Version 5.2

Die Beispiele können gerne nach Belieben angepasst werden.

information.html

Eine Vorlage zur Erweiterung des Informationsreiters.

Vorlage

QuelltextHTML Code:

```
1. <form name="extra">
2. <div align="center">
3.   <table class="table">
4.     <tr>
5.       <td colspan="2" class="cell">
6.         <table border="0" cellspacing="1" cellpadding="4" class="table">
7.           <tr>
8.             <td>Mein eigenes Eingabefeld</td>
9.             <td><input type="text" name="extra[myField]"></td>
10.          </tr>
11.        </table>
12.      </td>
13.    </tr>
14.  </table>
15. </div>
16. </form>
17.
18. {literal}
19. <script language="javascript">
20. <!--
21.
22. var f = document.forms["extra"];
23.
24. function do_load_extra()
25. {
26.   /* Beim Betreten des Reiters das Feld laden */
27.   f.elements["extra[myField]"].value = window.parent.get_extra("myField");
28. }
29.
30. function do_unload_extra()
31. {
32.   /* Beim Verlassen des Reiters das Feld sichern */
33.   window.parent.set_extra("myField", f.elements["extra[myField]"].value);
34. }
35. //-->
36. </script>
37. {/literal}
```

```
<form name="extra">
<div align="center">
  <table class="table">
    <tr>
      <td colspan="2" class="cell">
        <table border="0" cellspacing="1" cellpadding="4" class="table">
          <tr>
            <td>Mein eigenes Eingabefeld</td>
            <td><input type="text" name="extra[myField]"></td>
```

```

        </tr>
    </table>
</td>
</tr>
</table>
</div>
</form>

{literal}
<script language="javascript">
<!--

var f = document.forms["extra"];

function do_load_extra()
{
    /* Beim Betreten des Reiters das Feld laden */
    f.elements["extra[myField]"].value = window.parent.get_extra("myField");
}

function do_unload_extra()
{
    /* Beim Verlassen des Reiters das Feld sichern */
    window.parent.set_extra("myField", f.elements["extra[myField]"].value);
}
//-->
</script>
{/literal}

```

extra.html

Ein Grundgerüst beim Erstellen von neuen Reitern.

Vorlage

QuelltextHTML Code:

```
1. <form name="extra" onsubmit="return false">
2. <div align="center">
3. <table class="table">
4.   <tr>
5.     <td colspan="2" class="cell">
6.       <table border="0" cellspacing="1" cellpadding="4" class="table">
7.         <tr>
8.           <td>
9.             {t}Tiefe{/t}
10.          </td>
11.          <td>
12.            <select name="tiefe">
13.              <option value="1000">{t}unbegrenzt{/t}</option>
14.              <option value="1">1</option>
15.              <option value="2">2</option>
16.              <option value="3">3</option>
17.              <option value="4">4</option>
18.              <option value="5">5</option>
19.              <option value="6">6</option>
20.              <option value="7">7</option>
21.              <option value="8">8</option>
22.              <option value="9">9</option>
23.            </select>
24.          </td>
25.        </tr>
26.        <tr>
27.          <td colspan="2">
28.
29.            {* Einfacher Text *}
30.            {input type="text" name="anzahl_news" title="Anzahl"}
31.
32.            {* Link *}
33.            {input type="link" name="mm_link" title="Link"}
34.
35.            {* Checkbox *}
36.            {input type="checkbox" name="test_checkbox" title="Checkbox" short="wenn
gewünscht bitte aktivieren"}
37.
38.            {* Farbpicker *}
39.            {input type="color" name="farbe" title="Farbe" short="Wählen Sie eine Farbe"}
40.
41.            {* Datum / Uhrzeit *}
42.            {input type="datetime" name="input_datetime" title="Datum" short="Bitte geben Sie ein
Datum ein, hier mit Uhrzeit"}
43.            {input type="datetime" name="input_datetime_2" title="Datum" short="Bitte geben Sie
ein Datum ein, hier ohne Uhrzeit" only_date=1}
44.
```

```

45.         {* Bild *}
46.         {input type="image" name="input_image" title="Bild" short="Wählen Sie eine Bild"}
47.
48.         </td>
49.     </tr>
50. </table>
51. </td>
52. </tr>
53. </table>
54. </div>
55. </form>
56.
57. {literal}
58. <script language="javascript">
59. <!--
60.
61. var f = document.forms["extra"]
62.
63. function do_load()
64. {
65.     // Text/Link
66.     f.elements["tiefe"].value = window.parent.get_extra("tiefe")
67.     f.elements["extra[anzahl_news]"].value = window.parent.get_extra("anzahl_news")
68.     f.elements["extra[mm_link]"].value = window.parent.get_extra("mm_link")
69.
70.     // Checkbox
71.     if(window.parent.get_extra("test_checkbox") == "checked") {
72.         f.elements["extra[test_checkbox]"].checked = true;
73.     } else {
74.         f.elements["extra[test_checkbox]"].checked = false;
75.     }
76.
77.     // Farbpicker
78.     f.elements["extra[farbe]"].value = window.parent.get_extra("farbe")
79.
80.     // Datum/Uhrzeit
81.     f.elements["extra[input_datetime]"].value = window.parent.get_extra("input_datetime")
82.     f.elements["extra[input_datetime_2]"].value = window.parent.get_extra("input_datetime_2")
83.
84.     // Bild
85.     f.elements["extra[input_image]"].value = window.parent.get_extra("input_image")
86. }
87.
88. function do_unload()
89. {
90.     // Text/Link
91.     window.parent.set_extra("tiefe", f.elements["tiefe"].value)
92.     window.parent.set_extra("anzahl_news", f.elements["extra[anzahl_news]"].value)
93.     window.parent.set_extra("mm_link", f.elements["extra[mm_link]"].value)
94.
95.     // Checkbox
96.     if(f.elements["extra[test_checkbox]"].checked) {
97.         window.parent.set_extra('test_checkbox', 'checked');
98.     } else {
99.         window.parent.set_extra('test_checkbox', 'unchecked');

```

```

100. }
101.
102. // Farbpicker
103. window.parent.set_extra("farbe", f.elements["extra[farbe]"].value)
104.
105. // Datum / Uhrzeit
106. window.parent.set_extra("input_datetime", f.elements["extra[input_datetime]"].value)
107. window.parent.set_extra("input_datetime_2", f.elements["extra[input_datetime_2]"].value)
108.
109. // Bild
110. window.parent.set_extra("input_image", f.elements["extra[input_image]"].value)
111.
112. }
113. //-->
114. </script>
115. {/literal}

```

```

<form name="extra" onsubmit="return false">
<div align="center">
<table class="table">
  <tr>
    <td colspan="2" class="cell">
      <table border="0" cellspacing="1" cellpadding="4" class="table">
        <tr>
          <td>
            {t}Tiefe{/t}
          </td>
          <td>
            <select name="tiefe">
              <option value="1000">{t}unbegrenzt{/t}</option>
              <option value="1">1</option>
              <option value="2">2</option>
              <option value="3">3</option>
              <option value="4">4</option>
              <option value="5">5</option>
              <option value="6">6</option>
              <option value="7">7</option>
              <option value="8">8</option>
              <option value="9">9</option>
            </select>
          </td>
        </tr>
      </table>
    <tr>
      <td colspan="2">
        {* Einfacher Text *}
        {input type="text" name="anzahl_news" title="Anzahl"}
        {* Link *}
        {input type="link" name="mm_link" title="Link"}
        {* Checkbox *}
        {input type="checkbox" name="test_checkbox" title="Checkbox" short="wenn gewünscht bitte
aktivieren"}

```

```

        {* Farbpicker *}
        {input type="color" name="farbe" title="Farbe" short="Wählen Sie eine Farbe"}

        {* Datum / Uhrzeit *}
        {input type="datetime" name="input_datetime" title="Datum" short="Bitte geben Sie ein Datum
ein, hier mit Uhrzeit"}
        {input type="datetime" name="input_datetime_2" title="Datum" short="Bitte geben Sie ein
Datum ein, hier ohne Uhrzeit" only_date=1}

        {* Bild *}
        {input type="image" name="input_image" title="Bild" short="Wählen Sie eine Bild"}

    </td>
</tr>
</table>
</td>
</tr>
</table>
</div>
</form>

```

```

{literal}
<script language="javascript">
<!--

var f = document.forms["extra"]

function do_load()
{
    // Text/Link
    f.elements["tiefe"].value = window.parent.get_extra("tiefe")
    f.elements["extra[anzahl_news]"].value = window.parent.get_extra("anzahl_news")
    f.elements["extra[mm_link]"].value = window.parent.get_extra("mm_link")

    // Checkbox
    if(window.parent.get_extra("test_checkbox") == "checked") {
        f.elements["extra[test_checkbox]"].checked = true;
    } else {
        f.elements["extra[test_checkbox]"].checked = false;
    }

    // Farbpicker
    f.elements["extra[farbe]"].value = window.parent.get_extra("farbe")

    // Datum/Uhrzeit
    f.elements["extra[input_datetime]"].value = window.parent.get_extra("input_datetime")
    f.elements["extra[input_datetime_2]"].value = window.parent.get_extra("input_datetime_2")

    // Bild
    f.elements["extra[input_image]"].value = window.parent.get_extra("input_image")
}

function do_unload()
{
    // Text/Link

```

```
window.parent.set_extra("tiefe", f.elements["tiefe"].value)
window.parent.set_extra("anzahl_news", f.elements["extra[anzahl_news]"].value)
window.parent.set_extra("mm_link", f.elements["extra[mm_link]"].value)

// Checkbox
if(f.elements["extra[test_checkbox]"].checked) {
    window.parent.set_extra('test_checkbox', 'checked');
} else {
    window.parent.set_extra('test_checkbox', 'unchecked');
}

// Farbpicker
window.parent.set_extra("farbe", f.elements["extra[farbe]"].value)

// Datum / Uhrzeit
window.parent.set_extra("input_datetime", f.elements["extra[input_datetime]"].value)
window.parent.set_extra("input_datetime_2", f.elements["extra[input_datetime_2]"].value)

// Bild
window.parent.set_extra("input_image", f.elements["extra[input_image]"].value)

}
//-->
</script>
{/literal}
```

Eigenschaften neuer Unterseiten

Beim Erstellen von Unterseiten im Adminbereich, können verschiedene Eigenschaften der neuen Seite beeinflusst werden.

Vorlage

QuelltextPHP Code:

```
1. <?php
2.
3. /**
4. * Seiteneinstellungen
5. */
6.
7. $page->field['children_order'] = 'desc';
8.
9. /**
10. * Eigenschaften neuer Unterseiten
11. */
12.
13. $toolbarMenu->editMainItem(array(
14.   'id'      => 'buttonNew',
15.   'text'    => "<img src='".$GLOBALS['egotec_conf']['url_dir'].
16.               "bin/admin_skin/egotec/img/new_page.gif' border=0 style='width: 24px; height:
17.               24px'/><br/>".
18.               $GLOBALS['auth']->translate('Neuer Typ'),
19.   'url'     => get_url($GLOBALS['global_conf']['url_dir'].'bin/page/action.php', array(
20.     'site'       => $site->name,
21.     'lang'       => $site->language,
22.     'field[id]'  => $page->field['id'],
23.     'new_child[name]' => 'Neuer Typ',
24.     'new_child[title]' => 'Neuer Typ',
25.     'new_child[type]' => 'kfz_typ/entry',
26.     'new_child[nav_hide]' => 5,
27.     'egoaction'  => 'new_child'
28.   )),
29.   'target' => 'alive',
30.   'alt'    => $GLOBALS['auth']->translate('Neuer Typ'),
31.   'active' => 1
32. ));
33. /**
34. * Neue Seite auf gleicher Ebene
35. */
36.
37. // wie oben, Unterschied: Parent auslesen und angeben
38. $parent = $page->getParents()->nextPage();
39. if ($parent && $parent->hasRights('child'))
40. {
41.   $toolbarMenu->editMainItem(array(
42.     'id'      => 'buttonNew',
43.     'text'    => "<img src='".$GLOBALS['egotec_conf']['url_dir'].
44.               "bin/admin_skin/egotec/img/new_page.gif' border=0 style='width: 24px; height:
45.               24px'/><br/>".
```

```

45.         $GLOBALS['auth']->translate('Neuer Typ'),
46.     'url'     => get_url($GLOBALS['global_conf']['url_dir'].'bin/page/action.php', array(
47.         'site'         => $site->name,
48.         'lang'         => $site->language,
49.         'field[id]'    => $parent->field['id'],
50.         'new_child[type]' => 'kfz_typ/entry',
51.         'new_child[nav_hide]' => 5,
52.         'egoaction'    => 'new_child'
53.     )),
54.     'target' => 'alive',
55.     'alt'    => $GLOBALS['auth']->translate('Neuer Typ'),
56.     'active' => 1
57. ));
58. }
59.
60. ?>
61.

```

<?php

```

/**
 * Seiteneinstellungen
 */

```

```
$page->field['children_order'] = 'desc';
```

```

/**
 * Eigenschaften neuer Unterseiten
 */

```

```

$toolbarMenu->editMainItem(array(
    'id'     => 'buttonNew',
    'text'   => "<img src='".$GLOBALS['egotec_conf']['url_dir'].
                "bin/admin_skin/egotec/img/new_page.gif" border=0 style='width: 24px; height: 24px'/><br/>".
                $GLOBALS['auth']->translate('Neuer Typ'),
    'url'    => get_url($GLOBALS['global_conf']['url_dir'].'bin/page/action.php', array(
        'site'         => $site->name,
        'lang'         => $site->language,
        'field[id]'    => $page->field['id'],
        'new_child[name]' => 'Neuer Typ',
        'new_child[title]' => 'Neuer Typ',
        'new_child[type]' => 'kfz_typ/entry',
        'new_child[nav_hide]' => 5,
        'egoaction'    => 'new_child'
    )),
    'target' => 'alive',
    'alt'    => $GLOBALS['auth']->translate('Neuer Typ'),
    'active' => 1
));

```

```

/**
 * Neue Seite auf gleicher Ebene
 */

```

```
// wie oben, Unterschied: Parent auslesen und angeben
```

```

$parent = $page->getParents()->nextPage();
if ($parent && $parent->hasRights('child'))
{
    $toolbarMenu->editMainItem(array(
        'id'      => 'buttonNew',
        'text'    => "<img src='". $GLOBALS['egotec_conf']['url_dir'].
                    "bin/admin_skin/egotec/img/new_page.gif" border=0 style='width: 24px; height: 24px'/><br/>".
                    $GLOBALS['auth']->translate('Neuer Typ'),
        'url'     => get_url($GLOBALS['global_conf']['url_dir'].'bin/page/action.php', array(
            'site'      => $site->name,
            'lang'      => $site->language,
            'field[id]' => $parent->field['id'],
            'new_child[type]' => 'kfz_typ/entry',
            'new_child[nav_hide]' => 5,
            'egoaction' => 'new_child'
        )),
        'target' => 'alive',
        'alt'    => $GLOBALS['auth']->translate('Neuer Typ'),
        'active' => 1
    ));
}
?>

```

Erstellen einer Unterseite

Um für die aktuelle Seite eine neue Unterseite zu erstellen, kann folgende Vorlage verwendet werden:

QuelltextPHP Code:

```
1. // field-Werte definieren
2. $field = array(
3.   'name' => 'Neuer Name',
4.   'title' => 'Neuer Titel',
5.   'short' => 'Neue Kurzbeschreibung',
6.   'type' => 'entry', // Der Seitentyp
7.   'inactive' => 0, // 0 = Seite ist aktiv; 1= Seite ist deaktiviert
8.   'nav_hide' => 1, // 1= Nicht in Navigation; 2=Intranet; 3 = 1+2; 4= von Suche ausschließen; 5=1+4;
   6=4+2; 7=1+2+4
9. );
10.
11. // extra-Werte definieren
12. $extra = array(
13.   'anzahl_news' => 10,
14.   'kurzbeschreibung_anzeigen' => true
15. );
16.
17. // Unterseite erzeugen
18. $page->newChild($field,$extra);
```

// field-Werte definieren

```
$field = array(
  'name' => 'Neuer Name',
  'title' => 'Neuer Titel',
  'short' => 'Neue Kurzbeschreibung',
  'type' => 'entry', // Der Seitentyp
  'inactive' => 0, // 0 = Seite ist aktiv; 1= Seite ist deaktiviert
  'nav_hide' => 1, // 1= Nicht in Navigation; 2=Intranet; 3 = 1+2; 4= von Suche ausschließen; 5=1+4; 6=4+2;
  7=1+2+4
);
```

// extra-Werte definieren

```
$extra = array(
  'anzahl_news' => 10,
  'kurzbeschreibung_anzeigen' => true
);
```

// Unterseite erzeugen

```
$page->newChild($field,$extra);
```

Eine andere Möglichkeit:

QuelltextPHP Code:

```
1. $schild = $page->newChild(array(
2.   'name' => "Neuer Name",
3.   'title' => "Neuer Titel",
4.   'type' => 'page', //Der Seitentyp
5.   'inactive'=> 0, // 0 = Seite ist aktiv; 1= Seite ist deaktiviert
```

```
6. 'nav_hide' => 1 // 1= Nicht in Navigation; 2=Intranet; 3 = 1+2; 4=von Suche ausschließen; 5=1+4;
   6=4+2; 7=1+2+4
7. ), array(
8.   'a' => 12,
9.   'b' => "blub"
10. ));
```

```
$child = $page->newChild(array(
  'name' => "Neuer Name",
  'title' => "Neuer Titel",
  'type' => 'page', //Der Seitentyp
  'inactive'=> 0, // 0 = Seite ist aktiv; 1= Seite ist deaktiviert
  'nav_hide' => 1 // 1= Nicht in Navigation; 2=Intranet; 3 = 1+2; 4=von Suche ausschließen; 5=1+4; 6=4+2;
  7=1+2+4
), array(
  'a' => 12,
  'b' => "blub"
));
```

Wiki Mandanten anlegen

Beim Anlegen eines neuen Wiki (=Mandant), müssen bestimmte Skripte und Templates in die entsprechenden Ordner des jeweiligen Mandanten kopiert werden.

Entpacken Sie dieses ZIP-Archiv [wiki_skin.zip](#) in den skin-Ordner des Mandanten und dieses ZIP-Archiv [wiki_site.zip](#) in den site-Ordner des Mandanten. Achten Sie anschließend darauf, dass die Berechtigungen der Dateien stimmen.

Die erste Seite des Wiki-Mandanten (ID 1) muss den Seitentyp "Wiki" bekommen.

Dojo Schnittstelle (Adminbereich)

Diese Schnittstelle ist erst **ab Version 5** verfügbar. Die Javascript Bibliothek Dojo erlaubt es unter anderem vorgefertigte Dialoge und Felder darzustellen. Im Adminbereich werden mit Dojo die Javascript Funktionen **alert()**, **confirm()** und **prompt()** simuliert. Hierzu öffnet sich jeweils ein Dojo Dialog, welches die Funktionalität der genannten Funktionen nachahmt.

Alle drei Funktionen erben voneinander. Dies bedeutet dass die Parameterübergabe bei allen drei Funktionen identisch ist.

alert() wird zu **dialog.alert()**
confirm() wird zu **dialog.confirm()**
prompt() wird zu **dialog.prompt()**
window.open() wird zu **dialog.open()**

Den confirm und prompt Funktionen wird ein Parameter übergeben, welches ein Objekt sein muss. Nur **dialog.alert()** kann optional auch ein String als Parameter übergeben werden.

QuelltextJavaScript Code:

```
1. dialog.alert('Das ist ein Alert!');
```

```
dialog.alert('Das ist ein Alert!');
```

Ansonsten sieht ein übergebener Parameter standardmäßig wie folgt aus:

QuelltextJavaScript Code:

```
1. dialog.alert({  
2.   text : 'Das ist ein Alert!'  
3. });
```

```
dialog.alert({  
  text : 'Das ist ein Alert!'  
});
```

Bei der Funktion **dialog.open()** muss als erster Parameter der Titel und als zweiten Parameter die Ziel URL angegeben werden. Optional kann mit einem dritten Parameter eingestellt werden, ob die Ziel-URL in einem iframe geöffnet werden soll, damit man verschiedene Probleme (ID-Konflikte, Styles,...) aus dem Weg gehen kann.

Beispiel:

QuelltextJavaScript Code:

```
1. dialog.open("Verlorene_Seiten", "../desktop/lost_pages.php", true);
```

```
dialog.open("Verlorene_Seiten", "../desktop/lost_pages.php", true);
```

Bei **dialog.confirm()** und **dialog.prompt()** will man allerdings die Benutzeraktion auswerten, weshalb noch folgender Parameter notwendig ist:

QuelltextJavaScript Code:

```
1. dialog.confirm({  
2.   text : 'Das ist ein Confirm!',
```

```

3.   onConfirm : function()
4.   {
5.       // Aktion die ausgeführt wird wenn der Ok Button geklickt wurde
6.   }
7. });

```

```

dialog.confirm({
    text : 'Das ist ein Confirm!',
    onConfirm : function()
    {
        // Aktion die ausgeführt wird wenn der Ok Button geklickt wurde
    }
});

```

QuelltextJavaScript Code:

```

1. dialog.prompt({
2.   text : 'Das ist ein Prompt!',
3.   onConfirm : function(response)
4.   {
5.       // Aktion die ausgeführt wird wenn der Ok Button geklickt wurde
6.       // Die Variable response beinhaltet den eingegebenen Text
7.   }
8. });

```

```

dialog.prompt({
    text : 'Das ist ein Prompt!',
    onConfirm : function(response)
    {
        // Aktion die ausgeführt wird wenn der Ok Button geklickt wurde
        // Die Variable response beinhaltet den eingegebenen Text
    }
});

```

Alle optionalen Parameter im Überblick:

- **title:** Der Titel des Dialogs (Standard: keiner)
- **buttons:** Ok und Cancel Buttons anzeigen (Standard: Ja, außer bei **dialog.alert()**)
- **confirm:** Text für den Ok Button (Standard: Ok)
- **cancel:** Text für den Cancel Button (Standard: Cancel)
- **onConfirm:** Funktion die aufgerufen wird wenn der Ok Button geklickt wurde
- **onCancel:** Funktion die aufgerufen wird wenn der Cancel Button geklickt wurde
- **timeout:** Zeit in Milliekunden bis der Dialog automatisch ausgeblendet wird (Standard: 3000)
Wenn nur **true** übergeben wird, ist der Standardwert 3000 Millisekunden.

Wenn 0 übergeben wird, wird der Dialog nicht automatisch ausgeblendet.

- **additionalElement:** ein zusätzliches HTML Objekt
- **doAfterHide:** Es kann eine Referenz auf eine Funktion übergeben werden, die nach dem Schließen eines Dialoges (Alert,Confirm,Prompt) ausgeführt wird.

Je nach Einsatzort dieser Funktionen kann ein **window.parent** Aufruf notwendig sein.

API

Erstellt mit
EGOTEC®
Internet:
www.egotec.com
© EGOTEC
GmbH

EGOTEC GmbH
Alte Neckarelzer Straße 24
D-74821 Mosbach

Tel: +49 (0)6261 /
6743-0
Fax: +49 (0)6261
/ 6743-29
E-Mail:
info@egotec.com