

Bestellformular im PhPepperShop

Ein beliebiges Formular im Bestellprozess einbinden. Im Moment ist nur ein Formular fix implementiert – nach der Kasse, vor der Bestellungsübersicht.

Inhaltsverzeichnis

Bestellformular im PhPepperShop.....	2
1. Dateien und Einbindung.....	2
1.1 Dateien.....	2
1.2 Einbindung in den Bestellprozess.....	2
2. Formulartemplate(s).....	2
2.1 Allgemeines.....	2
2.2 Variablen.....	2
2.3 Hidden Elements.....	3
2.4 <form>-Tag.....	3
3. Elementdefinition.....	3
3.1 Sprachunabhängige Daten.....	3
3.2 Sprachabhängige Daten.....	3
3.2.1 check_type – nach was soll geprüft werden?.....	4
3.2.2 dropdown.....	4
3.2.3 text.....	4
3.2.4 radio.....	4
3.2.5 checkbox.....	5
3.2.6 textarea.....	5
4. Ablauf des Prozesses im Bestellvorgang.....	5
5. Wie werden die Checkresultate ausgewertet?.....	6
5.1 Interpretationsarten.....	6



Bestellformular im PhPepperShop

Der PhPepperShop bietet mit dem Bestellformular einen einfach einzubindenden, dynamisch gestaltbaren und multilingual einsetzbaren Formulargenerator.

1. Dateien und Einbindung

1.1 Dateien

<code>shop/</code>	
└─ <code>USER_BESTELLUNG_1.php</code>	Bestellprozess des PhPepperShops / Einbindung
└─ <code>bestellformular/</code>	
└─ <code>nach_kasse_form.php</code>	Formulartemplate
└─ <code>nach_kasse_elements.php</code>	Elemente des Formulars (sprachunabhängig)
└─ <code>nach_kasse_de.php</code>	Elementedefinitionen in der Sprache Deutsch
└─ <code>nach_kasse_en.php</code>	Elementedefinitionen in der Sprache Englisch
└─ <code>nach_kasse_fr.php</code>	Elementedefinitionen in der Sprache Französisch
└─ <code>nach_kasse_it.php</code>	Elementedefinitionen in der Sprache Italienisch

1.2 Einbindung in den Bestellprozess

Das einzige Formular, welches bis jetzt existiert, das '*nach_kasse*'-Formular, ist in der Datei `shop/USER_BESTELLUNG_1.php` eingebunden. Nach der Kasse und vor der Bestellungsübersicht. Die Einbindung erfolgt im Bereich `darstellen == 2`. Zu Beginn des Bereichs wird überprüft ob man von der Kasse her kommt oder vom Formular und weiter unten ist die Steuerung der Anzeige des Formulars oder der Bestellungsübersicht eingefügt.

Die Anzeigesteuerung erfolgt über die in `shop/config.inc.php` definierte Konstante '`NACH_KASSE_FORMULAR`'. Standardmässig ist diese auf `false` (Boolean) gesetzt.

2. Formulartemplate(s)

2.1 Allgemeines

Wie im Kaptitel Dateien und Einbindung beschrieben, gibt es die Datei `shop/bestellformular/nach_kasse_form.php`, welches das HTML-Template des Formulars beinhaltet. Eigentlich nicht nur ein Template, sondern gleich mehrere:

<code>\$form_strings['title']</code>	Formulartitel (nicht HTML- <code><title></code>)
<code>\$form_strings['head']</code>	Formularteil vor den Elementen
<code>\$form_strings['element']</code>	Beschreibung eines Elements
<code>\$form_strings['begrueundung']</code>	Beschreibung der Begründung (falls verwendet)
<code>\$form_strings['footer']</code>	Formularteil nach den Elementen

Die Templatedefinitionen erfolgen jeweils in einem PHP-String, welcher sich meistens über mehrere Zeilen hinweg erstreckt.

2.2 Variablen

Die Variablen, welche vom Formulartemplatesystem ersetzt werden, sind in geschweifte Klammern gefasst. Einige Beispiele:

<code>{get_localized_text[formulartitel]}</code>	Lokalisierte Variable (hier <code>formulartitel</code>) einlesen
<code>{selection}</code>	innerhalb einer Elementedefinition gibt es keine Prefixes
<code>{special:name}</code>	'special:' = Prefix zur Unterscheidung

2.3 Hidden Elements

Es gibt drei versteckte HTML Input-Elemente, welche für die Anzeigesteuerung verwendet werden, und welche somit auch *immer vorhanden* sein müssen:

```
<input type="hidden" name="von_bestellformular" value="true">
<input type="hidden" name="lang" value="'.$lang.'">
<input type="hidden" name="javascript_enabled" value="'.$javascript_enabled.'">
```

2.4 <form>-Tag

Etwas speziell ist natürlich auch immer der <form ... > HTML-Tag, weil dies die Verknüpfung des Formulars mit dem Bestellprozess aus Sicht des Formulars darstellt.

```
<form name="Bestellformular" action="'.$link_zum_check.'" method="post">
```

Wichtig ist hier die PHP-Variable `$link_zum_check`. Diese Variable enthält den Ort, wo das Formular die eingegebenen Kundendaten zur Verifikation sendet. In der Formulartemplatedatei sieht man ganz am Anfang PHP-Variablendefinitionen, eine davon ist dieser Link. Der Link zurück ist via HTML-Link realisiert, auch diese Variable ist am Dateianfang definiert.

3. Elementdefinition

Ein Element, resp. Seine Definition besteht aus zwei Teilen: Einem sprachabhängigen und einem unabhängigen Teil. Definiert wird der Name des Formulars, das Label (was wird dem Shopkunden angezeigt), die Art der Eingabeüberprüfung (irgend ein Text, exakt, gar nichts, ...) und Optionen...

3.1 Sprachunabhängige Daten

Diese Daten werden in der Datei `shop/bestellformular/nach_kasse_elements.php` definiert. Ein Beispieleintrag sieht wie folgt aus:

```
$form_elements[0]['name']      = 'keine_medikamente_einnehmen';
$form_elements[0]['type']      = 'dropdown';
```

Wichtig ist, dass man die Nummer (`[0]`) pro Element erhöht, es dürfen weiter in der Nummerierung *keine Lücken* vorkommen.

Wie man sieht, ist die Elementdefinition via PHP-Arrays realisiert.

'name' Hiermit ist ein interner Name gemeint, am besten nur folgende Zeichen verwenden:
[a-zA-Z0-9_] verwenden

'type' Von welchem Typ ist das Element: *dropdown, text, radio, checkbox, textarea*

Die Textfeldoptionen sind auch sprachunabhängig, werden aber im sprachabhängigen Teil erklärt, da dort die Typen der Eingabefelder beschrieben werden. Alle weiteren Einstellungen sind sprachabhängig und befinden sich in den entsprechenden Dateien.

3.2 Sprachabhängige Daten

Die von einer Sprache abhängigen Definitionen eines Elements befinden sich für jede Sprache in einer eigenen Datei. Die Namen der Dateien lassen sich leicht einer Sprache zuordnen, der Postfix ist jeweils der ISO-639-1 Code (lowercase) der Sprache, Format: *Formularname_ISO-Code.php*. Hier einige Beispiele:

```
nach_kasse_de.php  Deutsch
nach_kasse_en.php  Englisch
nach_kasse_fr.php  Französisch
nach_kasse_it.php  Italienisch
```

Die Sprachabhängigen Daten sind folgende, zum Teil Typ-abhängige Daten:

<code>\$form_elements[0]['name']</code>	Name des Elements (interner Name)
<code>\$form_elements[0]['label']</code>	angezeigter Name (Bezeichnung / Label)
<code>\$form_elements[0]['error_msg']</code>	Fehlermeldung, wenn Check nicht bestanden wird
<code>\$form_elements[0]['check_type']</code>	Art des Checks (z.B. @text, ...)

3.2.1 check_type – nach was soll geprüft werden?

Was ist der akzeptierte Wert/Wertebereich für dieses Element. Falls ein anderer Wert vom Shopkunden gegeben wird, zeigt ihm der PhPepperShop die in *error_msg* angegebene Fehlermeldung und er wird nochmals zum Formular zurück gesendet. Arten des Checks:

- irgendwas = genau dieser Text ist gefordert
- irgendwas = alles ist erlaubt, ausser Leerstring und der angegebene String
- @text = irgend ein Text kann eingegeben werden, ein Leerstring wird als falsch erkannt
- @email = Nur eine E-Mail Adresse wird akzeptiert (nur syntaktischer Check)
- = Wenn man nichts angibt, so wird das Element nicht überprüft

Nun folgen die vom Typ des Elements abhängigen Elementdefinitionen, es gibt zu jedem Typ ein Beispiel in der sprachabhängigen Definitionsdatei.

3.2.2 dropdown

Ein Dropdown Menü besteht aus Optionen. Die Auswahl dieser Optionen wird dem Shopkunden angezeigt. Die Dropdown-Werte (dropdown_values) werden wiederum in einem Array von 0-n eingeordnet. Da ein Dropdown-Wert selbst aber noch aus mehreren Definitionen besteht, ist auch der Wert selber in einem Array definiert (array(...)). Der 'value' enthält den via POST übertragenen Wert, 'label' den angezeigten Namen. Die optionale Definition 'option_color_style' fasst einen in Hex codierten Farbwert, wie er im Web üblich ist mit vorangestelltem #. Die Farbwertzuweisung funktioniert nur im Internet Explorer richtig.

```
$form_elements[0]['dropdown_values'][0] = array('value'=>'Ja' , 'label'=>'Ja' ,
'option_color_style'=>'#33CC33');
$form_elements[0]['dropdown_values'][1] = array('value'=>'Nein', 'label'=>'Nein',
'option_color_style'=>'#FF0000');
```

3.2.3 text

Ein Textfeld muss noch wissen, wie lange das Eingabefeld sein soll 'size' und wie seine maximale Länge aussehen soll – 'maxlength'. Da das Textfeld keine sprachabhängigen Daten beinhaltet, wird die Definition in der Datei nach_kasse_elements.php gemacht:

```
$form_elements[0]['options'] = array('size'=>10, 'maxlength'=>30);
```

3.2.4 radio

Damit man nicht noch weitere Feldnamendefinitionen im Programmcode abfragen muss, wurden hier in der v.1.0 API die dropdown_values verwendet. Man hat wieder 'name' und 'label' aber danach noch eine neue Definition 'default'. Dieses Feld fasst einen booleschen Wert. Wenn es = true ist, so dient dieses Feld in der Radiobuttongroup als vorangewähltes Standardfeld. Pro Radiobuttongroup sollte es immer nur ein Feld geben, welches 'default' = true hat. Alle anderen Felder der Gruppe müssen 'default' = false haben.

In folgendem Beispiel sieht man gerade noch, wie man z.B. mittels einem einfachen -Tag eine Farbinformation für das Label integrieren kann.

```
$form_elements[0]['dropdown_values'][0] = array('value'=>'Ja' , 'label'=><font
color="#33CC33">Ja</font>' , 'default'=>true);
$form_elements[0]['dropdown_values'][1] = array('value'=>'Nein', 'label'=><font
color="#FF0000">Nein</font>' , 'default'=>false);
```

3.2.5 checkbox

Hier gilt dieselbe Syntax wie für die Radiobuttongroup. Die 'default' Definition bedeutet hier, dass die Checkbox vorangewählt ist.

Die Checkbox hat die Eigenschaft, dass sie ihren Wert nur überträgt, wenn sie aktiviert ist. Falls dies nicht der Fall ist, wird kein POST-Wert generiert und übertragen, dies nur zur Information.

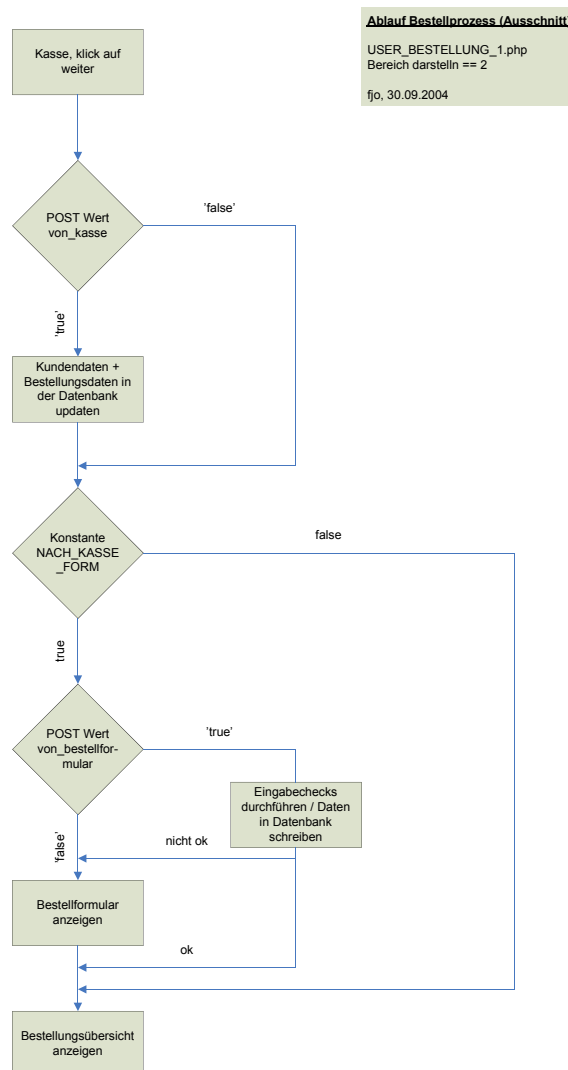
```
$form_elements[0]['dropdown_values'] = array('value'=>'Ja', 'label'=>'<font color="#33CC33">Ja</font>', 'default'=>true);
```

3.2.6 textarea

Mit einer Textarea kann auch grösserer Text abgefragt werden. Man muss hier die Anzahl Zeilen ('rows') und Spalten ('cols'), sowie die Art des Zeilenumbruchs ('wrap') definieren. Die hier möglichen Werte sind dieselben wie in der HTML 4.01 Spezifikation.

```
$form_elements[0]['dropdown_values'] = array('rows'=>10, 'cols'=>3, 'wrap'=>'physical', 'default_value'=>'Mein Text');
```

4. Ablauf des Prozesses im Bestellvorgang



5. Wie werden die Checkresultate ausgewertet?

In der Datei `shop/bestellformular/nach_kasse_elements.php` können zu Beginn der Datei zwei Variablen definiert werden: `$form_check_commit` (wie sollen die Fehler interpretiert werden) und den Array `$form_check_begrueundung`, in welchem die Details beschrieben werden, wenn die Interpretation 'begrueundung' gewählt ist.

5.1 Interpretationsarten

Der Variable `$form_check_commit` kann man zwei verschiedene Werte angeben:

- a) **einzel**: Der Check eines jeden einzelnen Elements muss erfüllt werden. Wenn auch nur einer nicht erfolgreich bestanden wird, kann der Kunde nicht weitergehen. Dies ist die Standard Formularauswertungsmethode.
- b) **begrueundung**: Ist die Wahl auf 'begrueundung' gefallen, so wird vom Kunden eine Begrueundung verlangt, wieso er bei den nicht bestanden Checks auf diese Art gewählt hat. Wenn der Shopkunde keine Checks der Elemente die im Array `$form_check_begrueundung['exclude']` eingetragen sind, verletzt, so kann er passieren. Nach soviel Prosa nun der Checkablauf noch in einer Auflistung:
 1. Ist mindestens ein Check nicht erfüllt, wenn ja:
 2. Fehlermeldungen und das Begrueundungsfeld anzeigen.
 3. Wenn der Shopkunde nun keine Begrueundung angibt und mindestens ein Check immer noch fehlerhaft ist, wird der Kunde daran gehindert weiterzufahren.
 4. Wenn der Kunde alle Checks besteht oder zumindest eine Begrueundung angegeben hat, so wird noch ein weiterer Check gestartet:
 5. Verletzt der Shopkunde noch einen Check, welcher von der Begrueundungsklausel ausgeschlossen ist (im Array `exclude` enthaltenes Element). Wenn ja, gilt für dieses Element die in a) beschriebene 'einzel' Interpretation – er *muss* diesen Check bestehen.