

# Tutorial



**Verkauf von konfigurierbaren Artikeln am Beispiel  
Meterware  
mit ShopPilot Enterprise**

IBO Internet Software OHG

Wehrstr. 6  
41199 Mönchengladbach

Tel. +49 (0) 2166 9989 530  
Fax +49 (0) 2166 9989 535

[ibo@shoppilot.de](mailto:ibo@shoppilot.de)  
[www.shoppilot.de](http://www.shoppilot.de)

<b>1</b>	<b>EINLEITUNG</b> .....	<b>3</b>
<b>2</b>	<b>EINRICHTEN DER FUNKTIONALITÄT AM BEISPIEL METERWARE</b> .....	<b>3</b>
2.1	Hinzufügen eines benutzerdefinierten Datenfeldes für Bestellpositionen.....	3
2.2	Bearbeiten der HTML-Vorlage für konfektionierte Ware (Detailseite).....	4
2.3	Berechnung des Preises für den Warenkorb.....	5
2.4	Darstellen der konfektionierten Ware im Warenkorb.....	5
2.4.1	Erstellen der Datei „bestellpositionen.ipf“.....	5
2.4.2	Einbau in die Warenkorbvorlage.....	6
<b>3</b>	<b>BEISPIEL</b> .....	<b>7</b>
3.1	<b>Screenshots (Kabelkonfigurator) im Konfiguratorshop</b> .....	<b>7</b>
3.1.1	Darstellung auf der Detailseite.....	7
3.1.2	Darstellung im Warenkorb.....	7

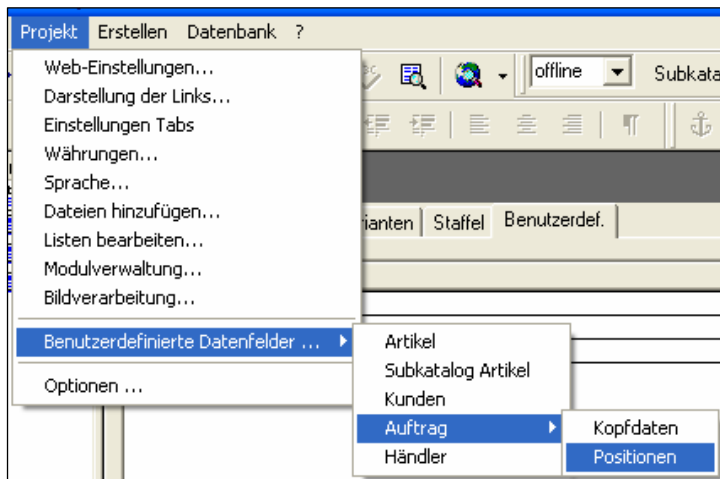
# 1 Einleitung

ShopPilot ermöglicht es konfigurierbare Artikel zu vertreiben. So ist es zum Beispiel möglich eine Farbe für Artikel vom Kunden eingeben zu lassen, eine Beschriftung für den Artikel eingeben zu lassen oder wie in diesem Beispiel beschrieben Meterware zu vertreiben.

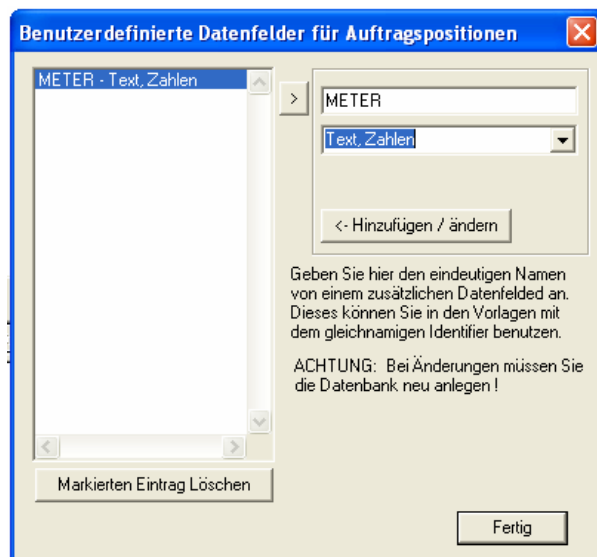
## 2 Einrichten der Funktionalität am Beispiel Meterware

### 2.1 Hinzufügen eines benutzerdefinierten Datenfeldes für Bestellpositionen

Um das Bestellen konfekzionierter Ware zu ermöglichen muss ein benutzerdefiniertes Datenfeld für Bestellpositionen eingerichtet werden, in welchem die Länge des Artikels gespeichert werden soll.



Öffnen Sie hierzu Shoppilot und navigieren Sie nach „Projekt→Benutzerdefinierte Datenfelder→Auftrag→Positionen“



Als benutzerdefiniertes Datenfeld für Bestellpositionen tragen Sie zum Beispiel „METER“ ein, und wählen als Datentyp „Text/Zahlen“.

## 2.2 Bearbeiten der HTML-Vorlage für konfektionierte Ware (Detailseite)

Um das Bestellen konfektionierter Ware zu ermöglichen muss für die Vorlage `pagetype(1)` verwendet werden. Außerdem muss das Formular um die Artikel in den Warenkorb zu legen manuell aufgebaut werden und als „action“ `__xformcart__` beinhalten.

Bauen Sie das Formular folgendermaßen auf:

```
<!--spmacro:pagetype(1)-->
<form method="post" action="__xformcart__">
```

Nun muss noch ein hidden-input Feld zum Formular hinzugefügt werden, über welches bestimmt wird, dass es sich bei diesem Artikel um einen konfektionierbaren Artikel handelt.

```
<input type="hidden" name="ITEMIDSTOFF" value="__artnr__">
```

Des Weiteren muss noch der Preis des Artikels über ein hidden-Input Feld mitgegeben werden.

```
<input type="hidden" name="price" value="__getprice::price__">
```

Damit das Modul „`getprice`“ verwendet werden kann muss dieses entweder in den `<head>`Bereich der Html-Vorlage geschrieben werden oder in eine separate `.tpl`-Datei, welche im `<head>`Bereich inkludiert wird.

Das Modul „`getprice`“:

```
<!--spmacro:module(getprice)
  sub price {
    $itemprice = ssp::get_var_article(preis, 0);
    main::mprint $itemprice;
  }
-->
```

Um es dem Kunden zu ermöglichen die Menge und die Länge der zu bestellenden Ware einzutragen, müssen noch folgende Input-Felder zum Formular hinzugefügt werden.

Eingabe der Länge:

```
<input type="text" name="METER__artnr__" value="0" size="3">
```

Eingabe der Menge:

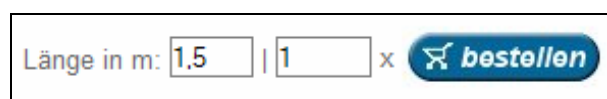
```
__anzkaufen__
```

Ein fertiges Formular in der HTML-Vorlage könnte also folgendermaßen aussehen:

*Beachten Sie das „METER“ der Name des benutzerdefinierten Feldes für Bestellpositionen ist. Wenn Sie einen anderen Namen verwendet haben, müssen Sie alle Stellen an welchen „METER“ vorkommt durch Ihre eigene Bezeichnung ersetzen.*

```
<!--spmacro:pagetype(1)-->
<form method="post" action="__xformcart__">
  <input type="hidden" name="ITEMIDSTOFF" value="__artnr__">
  <input type="hidden" name="price" value="__getprice::price__">
  Artikel: __produkt__
  Artikelnummer: __artnr__
  Preis: __preis::pav__
  Beschreibung: __dtext__
  Länge in Metern:
  <input type="text" name="METER__artnr__" value="0" size="3"> |
  __anzkaufen__ x __id__

</form>
```



Länge in m:  |  x

## 2.3 Berechnung des Preises für den Warenkorb

Die Berechnung des Preises für Meterware erfolgt in der Datei stdplacecart.ipl, welche aufgerufen wird, wenn ein Artikel in den Warenkorb gelegt wird.

In der stdplacecart.ipl erstellen wir nun ein Modul mit dem Namen „form“:  
Hier werden die Formularvariablen entgegengenommen und der Preis für den Artikel berechnet.  
Der Preis berechnet sich aus `Artikelpreis(Preis pro Meter) * Meter`.

```
<!--spmacro:module(form)
  $item = ssp::get_var_form('ITEMIDSTOFF');
  if (length($item) && ($item ne ssp::undefined)) {
    my $meter = main::qform(ssp::get_var_form("METER__$item"));
    my $price = main::qform(ssp::get_var_form("price"));
    $meter =~ s/,././g;
    if (length($meter)) {
      $main::FORM{"METER__$item"} = $meter;
      $main::FORM{"p__$item"} = $price * $meter;
    }
  }
-->
```

Hier gilt es nur das benutzerdefinierte Feld für Bestellpositionen anzupassen. Wenn Sie beim Anlegen des Feldes in ShopPilot den Namen „METER“ verwendet haben, können Sie den Code so verwenden wie oben dargestellt.

## 2.4 Darstellen der konfektionierten Ware im Warenkorb

### 2.4.1 Erstellen der Datei „bestellpositionen.ipl“

Damit die konfektionierte Ware im Warenkorb dargestellt werden kann und eine Kunde die Länge auch im Warenkorb noch ändern kann, muss die Datei bestellpositionen.ipl in die Warenkorbvorlage inkludiert werden.

Das Modul „bestellposition“ steuert die Darstellung im Warenkorb beim jeweiligen Artikel sowie die Preismodifikation, wenn die Länge eines Artikels im Warenkorb verändert wird.

In der Datei muss folgende Funktionalität hinterlegt sein:

```
<!--spmacro:module(bestellposition)
  sub printBenutzerdefinierteFelder {

    my $index = shift;
    my $txt = "";
    my $anzahl = ssp::get_var_cart('anzahl',$index);
    my $artnr = ssp::get_var_cart('id',$index);
    my $id = ssp::get_var_cart('rawid',$index);

    my $meter = ssp::get_var_cart("METER",$index);
    my $price = ssp::get_var_cart("preis",$index);

    if ($price != 0 and $meter !=0){
      $price = main::fp($price / $meter);
      #Ausgelesenen Summenpreis auf Meterpreis zurücksetzen.
    }
    if (length($meter)) {
      $meter =~ tr/././;
      $txt = qq {
        <tr><td colspan="6">$meter m zu $price pro Meter</td></tr>
      };
    }

    main::mprint $txt;
  }

  sub printBenutzerdefinierteFelder_Bearbeitbar {

    my $index = shift;

    my $anzahl = ssp::get_var_cart('anzahl',$index);
    my $artnr = ssp::get_var_cart('id',$index);
    my $id = ssp::get_var_cart('rawid',$index);
```

```

my $txt = "";

my $meter = ssp::get_var_cart("METER", $index);
my $price = ssp::get_var_cart ("preis", $index);

if ($price != 0 and $meter !=0){
    $price = ($price / $meter);
    #Ausgelesenen Summenpreis auf Meterpreis zurücksetzen.
}
if (length($meter)) {
    $meter =~ tr/./,/;
    $txt = qq {
        <tr>
            <td colspan="6">
                lfd. Meter (z.B. 1,5 für 150 cm):
                <form method="post" action="$xanzaendern" style="display:inline;">
                    <input name="$id" type="hidden" value="$anzahl">
                    <input name="ITEMIDSTOFF" type="hidden" value="$id">
                    <input name="posindex" type="hidden" value="$index">
                    <input name="price" type="hidden" value="$price">
                    <input name="METER__$id" type="text" value="$meter" size="5"
maxlength="10" class="text">
                    <input type="submit" value="&auml;ndern" class="submit">
                </form><br>
            </td>
        </tr>
    };
}
main::mprint $txt;
-->

```

## 2.4.2 Einbau in die Warenkorbvorlage

Als erstes muss die Datei `bestellpositionen.ipl` in die Warenkorbvorlage inkludiert werden. Das geschieht per `<!--spsmacro:include(scriptverz/bestellpositionen.ipl)-->`.

Im `loopitem`-Bereich, an der Stelle wo die im Warenkorb befindlichen Artikel ausgegeben werden sollen, muss eine Zeile hinzugefügt werden, in welcher die zusätzlichen Angaben der konfektionierten Artikel ausgegeben werden. Die Zeile muss folgenden Identifier beinhalten:

```
__bestellposition:::printBenutzerdefinierteFelder_Bearbeitbar__
```

### 3 Beispiel

Im Demoshop „Konfiguratorshop“, welcher im Lieferumfang von ShopPilot enthalten ist, gibt es einen Kabelkonfigurator, mit welchem man Kabel in einer selbst definierten Länge bestellen kann.

Die Dateien, welche benutzt werden um die Funktionalität im Konfiguratorshop abzubilden sind:

- scripts/bestellpositionen.ipl
- stdplacecart.ipl
- html\_de/p\_detail\_stoff.html
- html\_de/p\_cart.html

### 3.1 Screenshots (Kabelkonfigurator) im Konfiguratorshop

#### 3.1.1 Darstellung auf der Detailseite

Der Artikel wird über die normalen Standard-Identifizier angezeigt, wobei ein per `__xformcart__` selbst aufgebautes Formular verwendet wird.

Es besteht die Möglichkeit Anzahl und Länge des Artikels einzutragen.

#### 3.1.2 Darstellung im Warenkorb

Artikel	Bezeichnung	Stückzahl	Einzelpreis	Gesamtpreis
	kderr HiFi-Kabel 5mm	5 <input type="button" value="ändern"/> ✖	255,00 EUR	1275,00 EUR
lfd. Meter (z.B. 1,5 für 150 cm):		25,5 <input type="button" value="ändern"/>		
Rabattierte Summe: 1275,00 EUR				0,00 EUR
nächste Stufe : 5000,00 EUR 7 %			5 %	- 63,75 EUR
<i>Hinweis: Angebots-Artikel werden nicht rabattiert!</i>			Gesamtpreis:	1211,25 EUR
Gesamtpreis inkl. 193,39 EUR enthaltene MwSt.				

Beim Artikel wird eine zusätzliche Zeile eingeblendet, in welcher die aktuelle Länge des Artikels angezeigt wird. Es ist möglich die Länge auch im Warenkorb noch zu ändern. Der Preis berechnet sich folgendermaßen:

$$\text{Einzelpreis(EP)} = \text{Preis} * \text{Länge}$$

$$\text{Gesamtpreis(GP)} = \text{EP} * \text{Anzahl}$$