

Tobias Vogt

Konfiguratoren mit Magento:
Wir erstellen ein Modul

Wer bin ich?

- Tobias Vogt aus Paderborn
- 26 Jahre
- Magento seit 2008
- Wirtschaftsinformatik, zertifizierter Magento-Entwickler
- code-x GmbH (www.code-x.de)

Projekte

- www.webguys.de
- www.magento-podcast.de

Vorab

- Ein paar Vorkenntnisse sind gut
- Keine komplette von A-Z Anleitung
- Ziel: Ein grober Überblick!

Was sind Konfiguratoren?

- Mass-Customization
 - Kundenindividuelle Massenproduktion
 - Stark in der Auto-Industrie eingesetzt
- Persönliche Produkte wie
 - Grusskarten, T-Shirts, ..
- Komplexere Produkt-Logiken wie
 - Veredelung, Preis-Logiken (B2B), ..

Magento Standard Features

Artikelinformationen

- Allgemein
- Preise
- Meta Information
- Bilder
- Wiederkehrendes Profil
- Gestaltung
- Geschenkooptionen
- Badmöbel
- Lagerverwaltung
- Kategorien
- Zubehör
- Up-Selling
- Cross-Selling
- Kundenmeinungen
- Artikelschlagworte
- Schlagworte von Kunden
- Individuelle Optionen**

Hallo Meet Magento (Default)

[Zurück](#) [Zurücksetzen](#) [Löschen](#) [Duplizieren](#) [Speichern](#) [Speichern und weiter bearbeiten](#)

Individuelle Optionen

[+ Neue Option](#)

Titel * **Eingabetyp *** **Pflichtangabe** **Reihenfolge** [Option löschen](#)

Dropdown Drop-down Ja 0

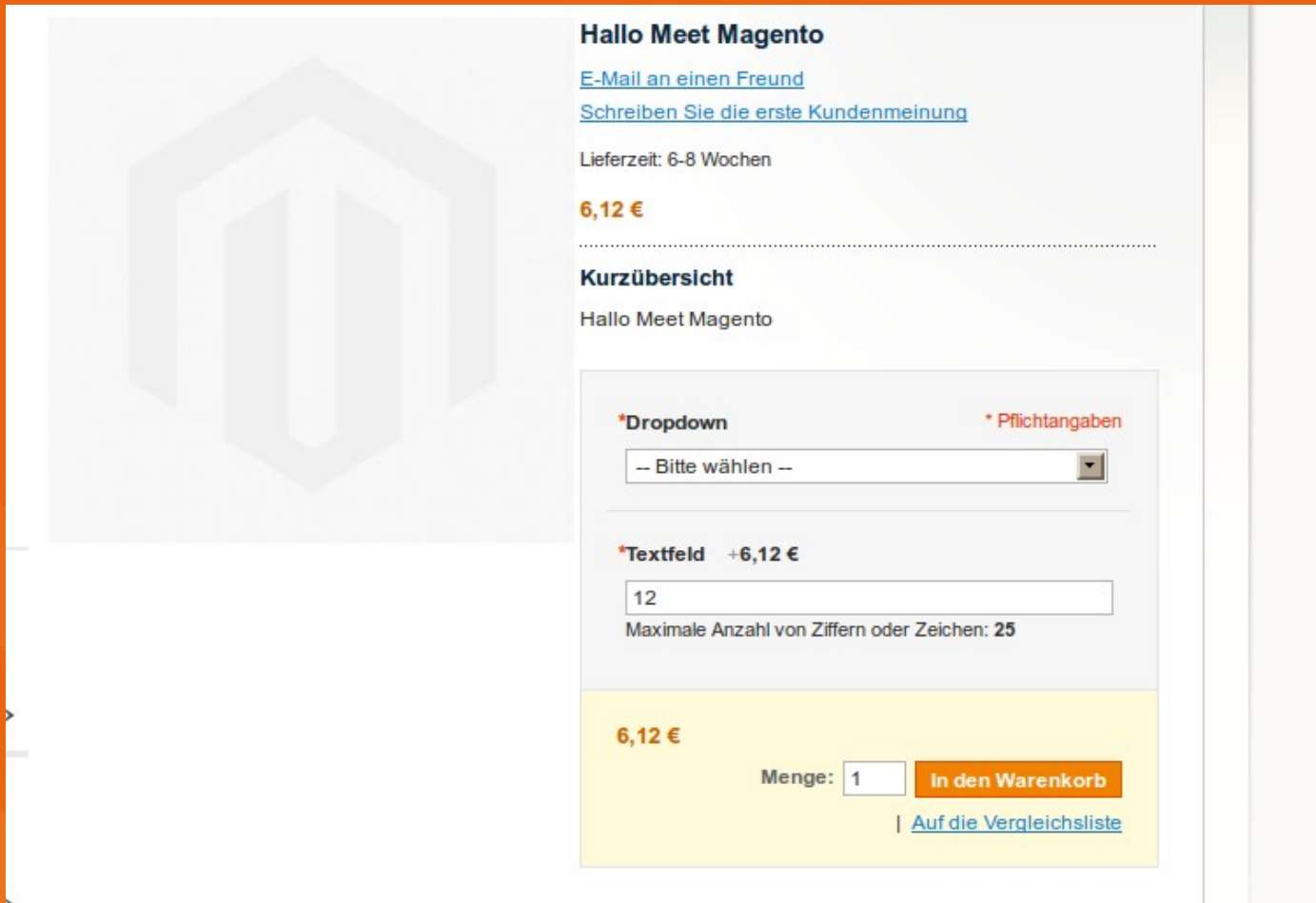
Titel *	Preis	Preisart	Artikelnummer	Reihenfolge	
Wert A	10.00	Prozent	DD-A	0	X
Wert B	12.00	Prozent	DD-B	0	X

[+ Neue Zeile](#)

Titel * **Eingabetyp *** **Pflichtangabe** **Reihenfolge** [Option löschen](#)

Textfeld Field Ja 0

Preis	Preisart	Artikelnummer	Maximale Anzahl Zeichen
6.12	Fest	TF-Text	25



Hallo Meet Magento

[E-Mail an einen Freund](#)
[Schreiben Sie die erste Kundenmeinung](#)

Lieferzeit: 6-8 Wochen

6,12 €

Kurzübersicht

Hallo Meet Magento

***Dropdown** *** Pflichtangaben**

-- Bitte wählen --

***Textfeld** **+6,12 €**

12

Maximale Anzahl von Ziffern oder Zeichen: 25

6,12 €

Menge: [In den Warenkorb](#)

| [Auf die Vergleichsliste](#)

Custom-Options im Frontend

.. und wenn das nicht reicht?

Das eigene Modul

Meet Magento Testproduct Konfigurator

[Email to a Friend](#)

[Be the first to review this product](#)

Availability: In stock

100,00 €

Text

X | Y

Qty: OR [Add to Wishlist](#)
[Add to Compare](#)

Quick Overview

Meet Magento Testproduct Konfigurator

Reine Funktion, kein „Klimmbimm“

Nützliche Events

0100 Events helfen uns Daten zu lesen und passend wieder zu schreiben

0010 weitere Events erweitern unsere Lösung um die Möglichkeit das konfigurierte Produkt auf den Wunschzettel legen zu können

- controller_action_layout_load_before
- catalog_controller_product_view
- checkout_cart_product_add_after
- checkout_cart_update_item_complete

- (wishlist_add_product)
- (wishlist_update_item)

1. Kunde ruft Artikel auf

- Event: *catalog_controller_product_view*
 - Prüfung ob Artikel konfigurierbar
 - Wenn ja: Layout-Handle vormerken
- Event: *before_load_layout*
 - Vorgemerkttes Layout-Handle hinzufügen

```
public function before_load_layout( $event ) {  
    if ( count( $this->_layoutHandlesToAdd ) ) {  
        $layoutupdate = $event->getEvent()->getLayout()->getUpdate();  
        foreach( $this->_layoutHandlesToAdd AS $handle ) {  
            Mage::helper('codex_konfigurator')->log('layout-handle:' . $handle );  
            $layoutupdate->addHandle( $handle );  
        }  
    }  
}
```

2. Konfigurator darstellen

- Eigenes Layout-Handle in Layout-XML abfangen
- catalog_product_view um Darstellung erweitern

```
<layout version="0.1.0">  
    <codex_konfigurator_hallowelt>  
        <reference name="product.info.extrahint">  
            <block type="codex_konfigurator/hallowelt"  
                template="konfigurator/hallowelt.phtml"  
                name="konfigurator.hallowelt"  
            />  
        </reference>  
    </codex_konfigurator_hallowelt>  
</layout>
```

3. Der Konfigurator

- Je nach Anforderung umsetzen
- Java-Script, Ajax, eigene Controller, ..
- Daten mit Blöcken aufbereiten

→ euer Job

4. Daten speichern

- `<form..>`-Tag von Magento nutzen
- Mittels Observer Daten aus Post-Request abfragen
- Im Quote-Item ablegen

Quote vs. Order

Mage_Sales_Model_Quote

Mage_Sales_Model_Order

- Quote ist der Warenkorb
- Quote wird bei Bestellung zu Order konvertiert
- Unterschiedliche Datenbank-Tabellen
- Konvertierung erfolgt nach Regeln im XML

mysql4-install-0.1.0.php

```
1 <?php
2
3 $installer = $this;
4 $installer->startSetup();
5
6 $setup = new Mage_Eav_Model_Entity_Setup('core_setup');
7
8 foreach( array('text', 'position_x', 'position_y') AS $cur ) {
9
10     $name = 'konfigurator_'. $cur;
11
12     $installer->getConnection()->addColumn(
13         $installer->getTable('sales_flat_quote_item'),
14         $name,
15         'TEXT NULL DEFAULT NULL'
16     );
17
18     $installer->getConnection()->addColumn(
19         $installer->getTable('sales_flat_order_item'),
20         $name,
21         'TEXT NULL DEFAULT NULL'
22     );
23 |
24 }
25
26 $installer->endSetup();
```

Dazu brauchen wir ein paar Datenbank-Felder

config.xml

```
<global>
  <fieldset>
    <sales_convert_quote_item>
      <konfigurator_text><to_order_item>*</to_order_item></konfigurator_text>
      <konfigurator_position_x><to_order_item>*</to_order_item></konfigurator_position_x>
      <konfigurator_position_y><to_order_item>*</to_order_item></konfigurator_position_y>
    </sales_convert_quote_item>
  </fieldset>
</global>
```

Quote-Item wird zu Order-Item konvertiert

config.xml

```
<events>

  <catalog_controller_product_view>
    <observers>
      <codex_konfigurator_catalog_product_view>
        <type>singleton</type>
        <class>codex_konfigurator/observer</class>
        <method>catalog_controller_product_view</method>
      </codex_konfigurator_catalog_product_view>
    </observers>
  </catalog_controller_product_view>

  <checkout_cart_product_add_after>
    <observers>
      <codex_konfigurator_checkout_cart_product_add_after>
        <type>singleton</type>
        <class>codex_konfigurator/observer</class>
        <method>checkout_cart_product_add_after</method>
      </codex_konfigurator_checkout_cart_product_add_after>
    </observers>
  </checkout_cart_product_add_after>

</events>
```

Events abfangen und verarbeiten

class ..Model_Konfigurator_Cart_ Item_Abstract {

Models nutzen

- ausschließlich über Models in das Cart-Item bzw. die Datenbank schreiben
- Übersicht & Wartung verbessern
- Keine „Magie“ im Observer da sonst Testen schwierig

- getProduct()
- load(*Mage_Sales_Model_Quote|Order_Item \$item*)
- save(*Mage_Sales_Model_Quote|Order_Item \$item*)
- validate()
- [..]

}

```

public function checkout_cart_product_add_after($event) {

    $quote_item = $event->getQuoteItem();
    /* @var $quote_item Mage_Sales_Model_Quote_Item */

    if ( !$quote_item ) {
        return;
    }

    $product = $quote_item->getProduct();
    /* @var $product Mage_Catalog_Model_Product */

    $controller = Mage::registry('controller');
    /* @var $controller Mage_Core_Controller_Varien_Front */

    try {

        $model = Mage::getModel('codex_konfigurator/hallowelt');
        $model->load( $quote_item );

        $data = $controller->getRequest()->getParam( 'helloworld', array() );
        $model->setData( $data );+

        $model->save( $quote_item );

    } catch( Codex_Konfigurator_Model_Cart_Exception $e ) {

        // Fehler abfangen

    }

}

```

Beim Add-To-Cart ebenfalls Daten speichern

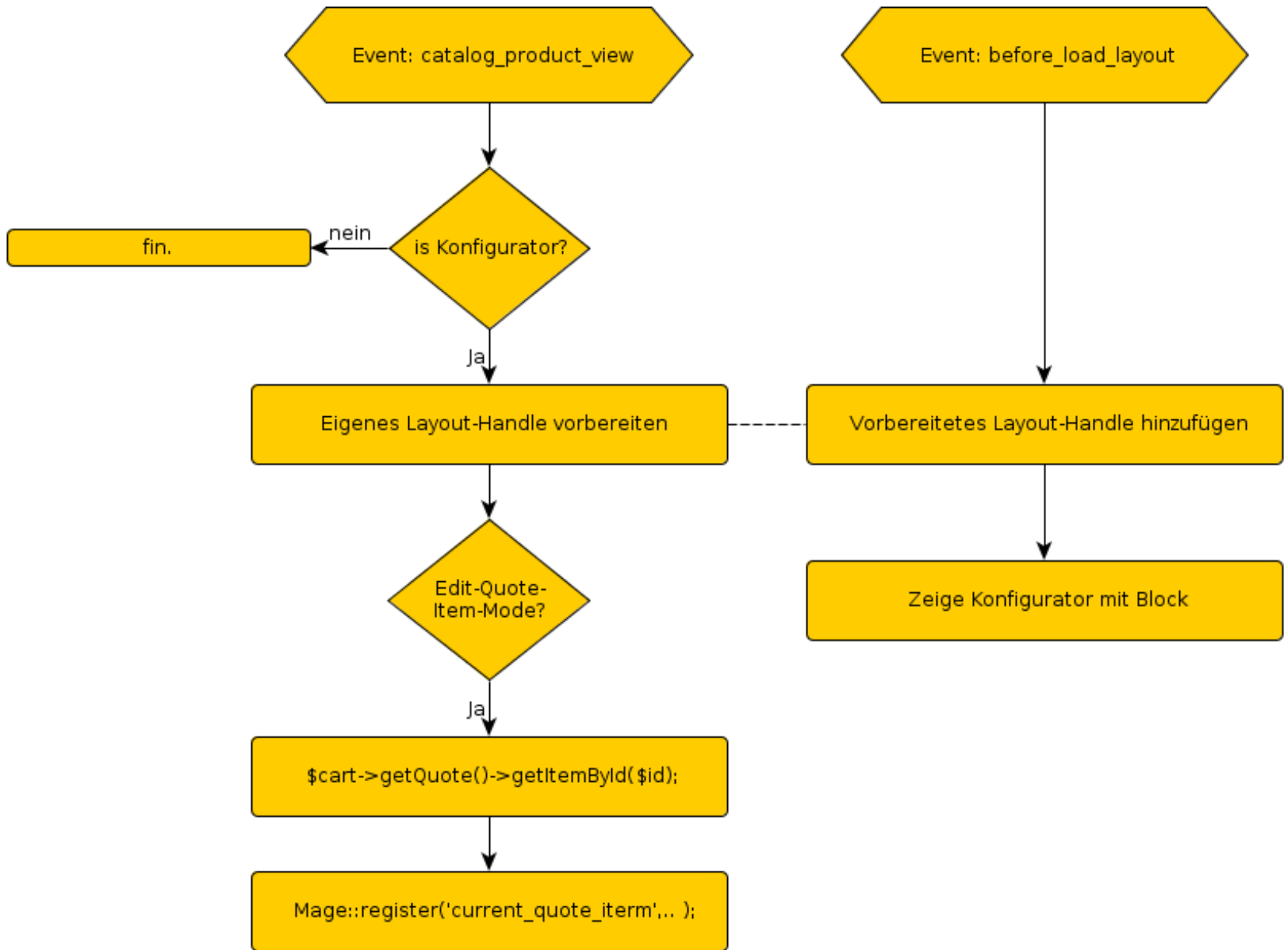
5. Bearbeiten des Quote-Items

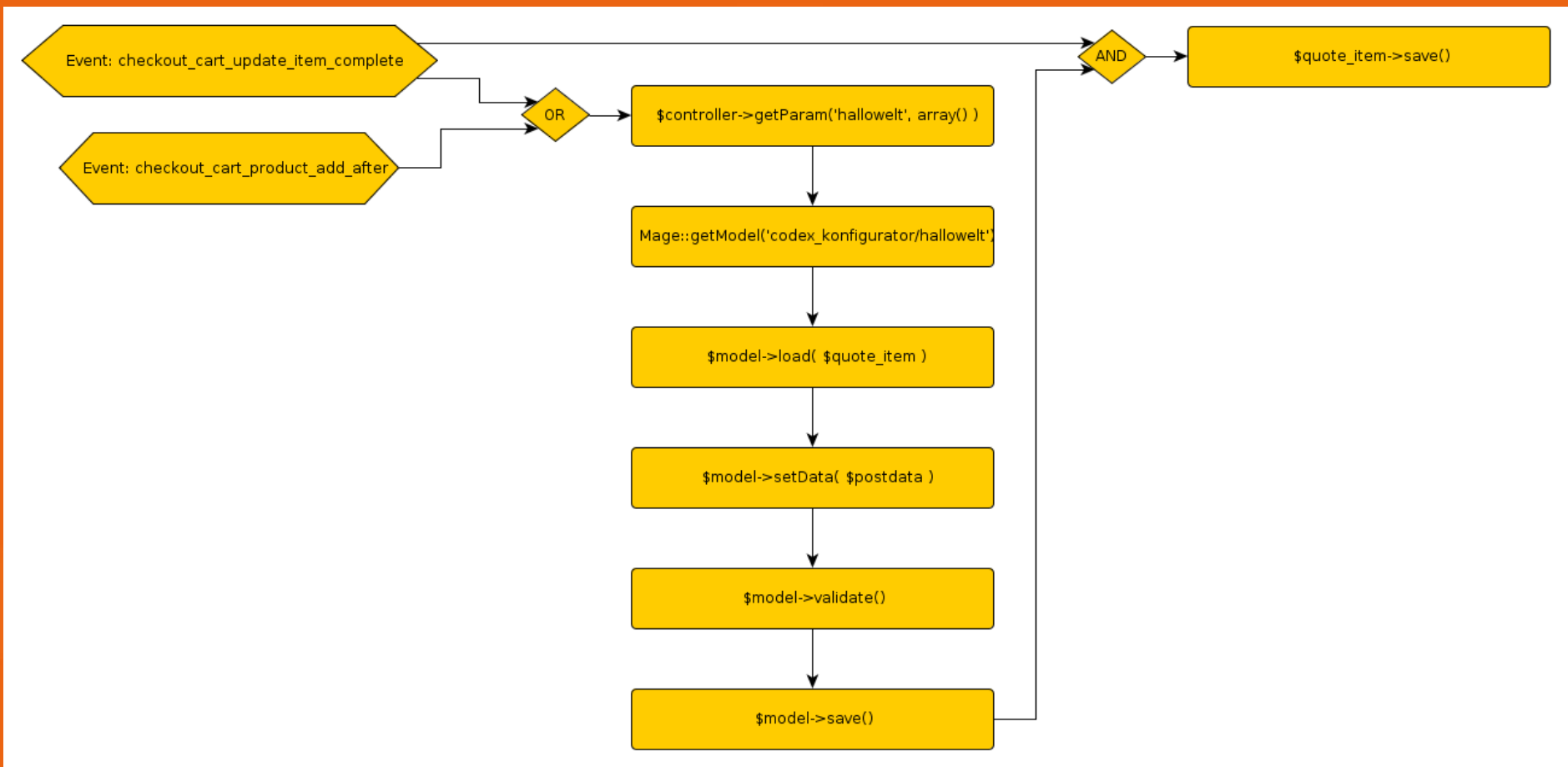
- Klick im Warenkorb auf „Bearbeiten“
- Magento baut Link wie *cart/configure/id/2614/*
- Id entspricht Quote-Item-Id

Lösung

- Im *catalog_product_view*-Event Quote-Item laden, global registrieren und im Block abfragen

???





Danke!

Beispiel unter
www.webguys.de/mm12de

Kontakt über
tobias.vogt@code-x.de

Twitter
www.twitter.com/tobi_pb