

# Praxisbeispiel Modulentwicklung: Integration einer Risikoprüfung

Meet Magento #2.09  
02.11.2009

## Überblick

- Profil Andreas von Studnitz
- Fallbeispiel: Aufgabenstellung und Ziele
- Umsetzung
  - Schritt 1: Erstellen der Konfiguration
  - Schritt 2: Ansatz finden
  - Schritt 3: Umsetzung im Detail
- Fazit

## Profil

- Informatikstudium an der RWTH Aachen
- 6 Jahre bei der Internetagentur team in medias
  - Projektleitung
  - Entwicklung mit Magento, TYPO3, xt:Commerce, ...
- Seit August 2009 als Magento Freelancer selbstständig
- Schwerpunkte:
  - Modulentwicklung
    - Schnittstellen
    - Backend-Erweiterung
    - Zahlungsmodule
  - Umsetzung von Shops
  - Templateentwicklung
  - Beratung

## **Fallbeispiel: Integration der Risikoprüfung von CEG in den Bestellprozess**

- Basis: Shop für Elektronikprodukte
- Ziel: Minimierung der Ausfallwahrscheinlichkeit von Zahlungen
- Methode: Anbieten von bestimmten Zahlungsarten nur bei ausreichend Bonität
  
- Anbindung der Services der CEG Creditreform Consumer GmbH (nur Bonitätsprüfung)
- Schnittstelle per HTTP
- Nahtlose Integration angestrebt, für den Endkunden unsichtbar

## Umsetzung: Ziele

- Die Schnittstelle wird aufgerufen, sobald im Bezahlprozess die Bezahlarten zur Auswahl angeboten werden sollen



The screenshot shows a payment selection interface with a progress bar at the top. The progress bar has six steps: 1. Wie möchten Sie zur Kasse gehen?, 2. Rechnungsinformation, 3. Versandinformation, 4. Versandart, 5. Zahlungsinformation (highlighted), and 6. Bestellübersicht. Below the progress bar, there are three radio button options: Lastschrift, Kreditkarte, and Auftrag. A red asterisk and the text '\* Pflichtfelder' are visible to the right of the options. At the bottom, there is a navigation bar with a left arrow and the text 'Zurück' on the left, and a blue button with the text 'Weiter' on the right.

- An CEG werden Kundendaten übermittelt, auf deren Basis ein Score-Wert berechnet wird.
- Auf Basis des Score-Wertes werden bestimmte Bezahlmodule ausgeblendet.

## Umsetzung: Ziele (2)

- Hierzu gibt es einen neuen Konfigurationsbereich:

The screenshot shows a configuration page titled "CEG Risikoprüfung" with a "Konfiguration speichern" button. The page is divided into sections: "Zugriffsdaten" and "Zahlungsmodule". Under "Zahlungsmodule", there are three modules, each with a dropdown menu for the payment method, a text input for the required score, and a text input for the maximum total sum in the cart. Module 1 is set to "payone\_elv" with a score of 949 and a maximum sum of 1700. Module 2 is set to "payone\_cc" with a score of 954 and a maximum sum of 1200. Module 3 is currently empty.

Modul	Zahlungsmethode	Erforderlicher Scorewert	Maximale Gesamtsumme im Warenkorb	Notizen
Modul 1	payone_elv	949	1700	Diese Zahlungsmethode wird nur erlaubt, wenn der erforderliche Scorewert erreicht wird und die erlaubte Warenkorb-Summe nicht überschritten wird (oder 0 ist).
Modul 2	payone_cc	954	1200	Diese Zahlungsmethode wird nur erlaubt, wenn der erforderliche Scorewert erreicht wird und die erlaubte Warenkorb-Summe nicht überschritten wird (oder 0 ist).
Modul 3				

## Vorgehensweise: Schritte

1. Einrichtung der Konfigurationsmöglichkeiten
2. Finden der Stelle im Core-Programmcode, an der die verfügbaren Bezahlarten ausgewählt werden
3. Erweiterung der bestehenden Funktionalität, um auf Basis einer Schnittstellenabfrage zu CEG die verfügbaren Bezahlarten einzuschränken

## Schritt 1: Erstellen der Konfiguration (1): neue Sektion

- Anlegen eines neuen Konfigurationsbereiches in etc/system.xml

```
<config>
  <sections>
    <ceg translate="label">
      <label>CEG Risk Check</label>
      <tab>sales</tab>
      <frontend_type>text</frontend_type>
      <sort_order>670</sort_order>
      <show_in_default>1</show_in_default>
      <show_in_website>1</show_in_website>
      <show_in_store>1</show_in_store>
      <groups>
        <payment_modules translate="label">
          <label>Payment Modules</label>
          <sort_order>20</sort_order>
          <show_in_default>1</show_in_default>
          <show_in_website>1</show_in_website>
          <show_in_store>1</show_in_store>
          <fields>
```

The diagram illustrates the XML code for creating a new configuration section. Three yellow callout boxes point to specific parts of the code:

- Name der Sektion:** Points to the `<ceg translate="label">` tag.
- Bereich:** Points to the `<tab>sales</tab>` tag.
- Neue Gruppe:** Points to the `<payment_modules translate="label">` tag inside the `<groups>` block.

## Schritt 1: Erstellen der Konfiguration (2): neue Felder unter <fields>

```
<module_1 translate="label">
  <label>Module 1</label>
  <frontend_type>select</frontend_type>
  <source_model>ceg/configSourcePaymentModules</source_model>
  <sort_order>10</sort_order>
  <show_in_default>1</show_in_default>
  <show_in_website>1</show_in_website>
  <show_in_store>1</show_in_store>
</module_1>
<minimum_score_1 translate="label">
  <label>Minimum score for module 1</label>
  <frontend_type>text</frontend_type>
  <sort_order>15</sort_order>
  <show_in_default>1</show_in_default>
  <show_in_website>1</show_in_website>
  <show_in_store>1</show_in_store>
  <comment></comment>
</minimum_score_1>
```

Name der Source-Model-Klasse

Name des Feldes

Typ des Feldes

## Schritt 1: Erstellen der Konfiguration (3): neues Source Model

- Neue Datei /app/code/local/AvS/CEG/Model/ConfigSourcePaymentModules.php

```
class AvS_CEG_Model_ConfigSourcePaymentModules
{
    public function toOptionArray()
    {
        $returnArray = array(array('value' => '', 'label' => ''));

        $modules = Mage::getModel('payment/config')->getActiveMethods();
        foreach($modules as $module) {

            $returnArray[] = array(
                'value' => $module->getCode(),
                'label' => $module->getCode()
            );
        }

        return $returnArray;
    }
}
```

Vordefinierter Standard-Name

Default-Eintrag

Alle aktive Bezahlarten auslesen

## Schritt 2: Ansatz finden

- Checkout wird über das Modul „Checkout“ (/app/code/core/Mage/Checkout/) gesteuert
- Möglicher Ansatzpunkt: Methode „\_canUseMethod“ in der Klasse „Mage\_Checkout\_Block\_Onepage\_Payment\_Methods“ (/app/code/core/Mage/Checkout/Block/Onepage/Payment/Methods.php)

```
protected function _canUseMethod($method)
{
    if (!$method || !$method->canUseCheckout()) {
        return false;
    }
    return parent::_canUseMethod($method);
}
```

- Die Methode wird für jede Bezahlart einmal aufgerufen und prüft, ob diese dargestellt werden kann

### Schritt 3.1: Erweiterung der bestehenden Klasse

1. Kopie der Klassendatei als neue Datei:

/app/code/local/AvS/CEG/Block/CheckoutOnepagePaymentMethods.php

```
class AvS_CEG_Block_CheckoutOnepagePaymentMethods  
extends Mage_Checkout_Block_Onepage_Payment_Methods
```

2. Konfiguration in etc/config.xml im Modulverzeichnis:

```
<global>  
  <blocks>  
    <checkout>  
      <rewrite>  
        <onepage_payment_methods>  
          AvS_CEG_Block_CheckoutOnepagePaymentMethods  
        </onepage_payment_methods>  
      </rewrite>  
    </checkout>  
  </blocks>  
</global>
```

Modul

Klassen-Suffix

Name der überschreibenden Klasse

### Schritt 3.2: Auslesen der festgelegten Konfiguration

```
protected $configValues = array();

protected function getConfigValues() {

    if (empty($this->configValues)) {

        for ($i = 1; $i <= 10; $i++) {

            $paymentModuleCode = Mage::getStoreConfig(
                'ceg/payment_modules/module_' . $i);
            $paymentModuleMinimumScore = Mage::getStoreConfig(
                'ceg/payment_modules/minimum_score_' . $i);
            $this->configValues[$paymentModuleCode] = array(
                'minimumScore' =>$paymentModuleMinimumScore,
            );
        }
    }
    return $this->configValues;
}
```

section/group/field

Auslesen eines Konfigurationswertes

### Schritt 3.3: Erweiterung der Methode „\_canUseMethod“

```
protected function _canUseMethod($method)
{
    $configValues = $this->getConfigValues();

    if (isset($configValues[$method->getCode()])) {

        $methodConfig = $configValues[$method->getCode()];
        $minimumScoreForMethod = $methodConfig['minimumScore'];
        $scoreForCurrentCustomer = $this->getRiskCheckResult();

        if ($scoreForCurrentCustomer <= $minimumScoreForMethod) {
            return false;
        }
    }
    return parent::_canUseMethod($method);
}
```

Wird für jede Bezahlart aufgerufen

Aufruf der Schnittstelle

Bezahlart verbieten!

(Die Methode „getRiskCheckResult“ wird hier nicht näher betrachtet, da sie nicht Magento-spezifisch ist. Hier ist die eigentliche Schnittstelle zu CEG implementiert (HTTP-Request). Sie benutzt das Quote-Objekt über „\$this->getQuote()“ und damit auch die Kundenadresse)

## Fazit

- Modulentwicklung mit Magento ist kein Hexenwerk – PHP-Kenntnisse vorausgesetzt
- Aber: eine gewisse Grunderfahrung mit Magento ist unabdingbar
- Größte Schwierigkeit ist häufig das Finden der richtigen Ansatzstelle für die eigene Funktionalität
- Magento bietet viele schöne Vorlagen und Vereinfachungen für die eigene Nutzung
- Das Modul ist bei mir auf Anfrage erhältlich, später über MagentoConnect

## Kontakt

Andreas von Studnitz  
- Professionelle Webentwicklung -  
Heussstraße 34  
D-52078 Aachen

Telefon +49 241 912 75 912

Mobil +49 170 486 0 464

E-Mail [avs@avs-webentwicklung.de](mailto:avs@avs-webentwicklung.de)

Web [www.avs-webentwicklung.de](http://www.avs-webentwicklung.de)

XING [www.xing.com/profile/Andreas\\_vonStudnitz2](http://www.xing.com/profile/Andreas_vonStudnitz2)