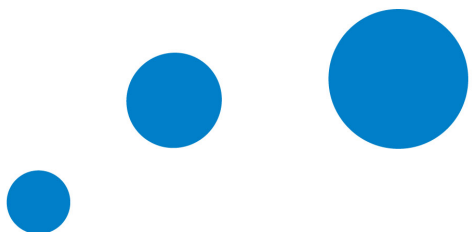


FWP Shop



FWP systems

Implementierung eines Zahlungsmoduls

Shop-Version 4.0

Inhalt

- 1 Verzeichnisstruktur
 - 1.1 Erklärung Namensgebung
- 2 Implementierung
 - 2.1 Payment Interface
 - 2.2 Implementierung der benötigten Klasse/Methoden
 - 2.3 Konfiguration des Moduls
- 3 Ablaufplan
- 4 Vorgaben
 - 4.1 Abbruch durch den Benutzer
 - 4.2 Einsatz von „AfterPayment“ Methoden
 - 4.2.1 Abbuchung
 - 4.2.2 Weiterleitung

1. Verzeichnisstruktur

Die Verzeichnisstruktur für alle Dateien des Zahlungsmoduls ist folgendermaßen aufgebaut:

```
_system/_classesphp/payment/payment[Modulname]_std.inc.php
```

[Modulname] steht für den Namen des Zahlungsmoduls und sollte „camelCased“ sein, beispielsweise paymentPaypal_std.inc.php. Nachdem die benötigte Datei angelegt wurde, kann mit der in **Schritt 2** dieser Dokumentation beschriebenen Implementierung der Klasse und Methoden begonnen werden.

2. Implementierung

2.1 Payment Interface

Die Implementierung eines Zahlungsmoduls erfordert eine Basisklasse. In dieser Klasse müssen bestimmte Methoden vorhanden sein, die durch ein Interface beschrieben werden. Das Interface lässt sich in dieser Datei finden:

```
_system/_classesphp/_interfaces/payment_iface_std.inc.php.
```

Die Klassen- und Methodenbeschreibung des Interfaces sieht folgendermaßen aus:

```
interface payment_iface {
    function prePaymentPrepareData();
    function prePaymentProcessData();
    function afterPaymentPrepareData();
    function afterPaymentProcessData();
    function checkPayment();
    function capturePayment();
}
```

Die Klasse des Zahlungsmoduls muss zwangsweise dieses Interface implementieren und die Beschreibung des Interfaces auch umsetzen, um Fehler zu verhindern.

2.2 Implementierung der benötigten Klasse und Methoden

Der Klassenname sollte der in **Schritt 1** beschriebene Modulname sein. Dem Modulnamen wird ein Präfix angefügt, der „payment“ lautet, um klar zustellen, dass es sich um ein Zahlungsmodul handelt. Hierbei ist wieder darauf zu achten, dass der Klassenname exakt dem des Modulnamens entspricht, da anhand des Namens die Klasse geladen und auf ihre Methoden zugegriffen wird. Alle in dieser Dokumentation beschriebenen Methodenaufrufe beziehen sich auf die nachfolgend dargestellte Beispielklasse.

Eine Beispielklasse, in diesem Fall für PayPal, könnte so aussehen:

```
class paymentPaypal implements payment_iface {
    public function __construct() {} // evtl. Konfiguration von Daten
    public function prePaymentPrepareData() {}
    public function prePaymentProcessData() {}
    public function afterPaymentPrepareData() {}
    public function afterPaymentProcessData() {}
    public function checkPayment() {}
    public function capturePayment() {}
}
```

In der folgenden Tabelle wird die Benutzung der einzelnen Methoden näher erläutert. Eine grafische Darstellung dieser Tabelle ist in **Schritt 3** anhand eines Ablaufplans visualisiert.

#	Methode	Beschreibung	vorher	danach
1	prePaymentPrepareData	Diese Funktion wird nach Auswahl der Zahlungsart ausgeführt und kann dazu genutzt werden bestimmte Daten vorzubereiten, die für den weiteren Ablauf des Bestellprozesses von Nöten sind. Falls nach Auswahl der Zahlungsart eine Weiterleitung zu einem Zahlungssystem gefordert ist, kann dies ebenfalls in dieser Methode implementiert werden.	-	2
2	prePaymentProcessData	In dieser Methode können Daten geprüft/bestätigt werden. Diese Funktion muss nicht zwingend einem Nutzen dienen, dies ist abhängig von der Zahlungsart und des Ziels des Zahlungsmoduls. Wenn in dieser Methode eine Überprüfung vorhanden ist und fehlschlägt, muss angeboten werden die Zahlungsart zu ändern.	1	3
3	afterPaymentPrepareData	afterPaymentPrepareData In dieser Methode kann bsp. die Abbuchung des zu zahlenden Betrages ausgeführt werden. Eine andere Möglichkeit wäre die Weiterleitung zu einem Zahlungssystem um dort benötigte Schritte auszuführen.	2	4
4	afterPaymentProcessData	Wenn in der vorher aufgerufenen Methode die Abbuchung und erfolgreiche Eintragung der Bestellung im Shopsystem bereits ausgeführt wurde, kann diese Methode vernachlässigt werden. Andernfalls, beispielsweise bei einer Weiterleitung durch die vorher aufgerufene Methode kann die Abbuchung und Eintragung der Bestellung in dieser Funktion stattfinden.	3	-

2.3 Konfiguration des Moduls

Es gibt verschiedene Konfigurationsmöglichkeiten für ein Zahlungsmodul. Die Konfigurationsdatei ist im Verzeichnis `_config/payment_types.ini` zu finden. Die Konfiguration ist folgendermaßen aufgebaut:

```
[ID des Zahlungstyps]
payment_class      = [Klassenname]
skipPrePayment    = [Vorgang vor der Bestellbestätigung überspringen?]
skipAfterPayment  = [Vorgang nach der Bestellbestätigung überspringen?]
```

Eine Beispielkonfiguration könnte so aussehen:

```
[5]
payment_class      = paymentPaypal
skipPrePayment    = 0
skipAfterPayment  = 0
```

Achtung!

Die ID des Zahlungsmoduls kann in der Datenbanktabelle `data_payment_type` eingesehen werden. Wenn die Konfiguration für das Überspringen der Bestellprozessschritte fehlt, wird der Schritt „prePayment“ automatisch übersprungen.

3. Ablaufplan



* Bei Abbruch sollte der Benutzer eine neue Zahlungsart auswählen können, damit der Bestellprozess weitergeführt werden kann. Dieser Fall trifft nur zu, wenn eine Umleitung zu einem Zahlungssystem (z.B. PayPal) stattgefunden hat. Andernfalls kann der Benutzer über die Bestellprozessanzeige zum Bearbeiten der Zahlungsart zurückspringen.

** Falls afterPaymentPrepareData() die Abbuchung des Betrags schon übernommen hat, weil keine Weiterleitung mehr nötig ist, so muss afterPaymentProcessData() nicht beachtet werden. In diesem Fall ist das Ende des Bestellprozesses bereits nach afterPaymentPrepareData() erreicht.

4. Vorgaben

4.1 Abbruch durch den Benutzer

Falls der Benutzer, also der Shop-Kunde einen Abbruch des aktuellen Vorgangs wünscht, muss zu der in **Schritt 3** beschriebenen Datei `utils/cancel_payment.php` weitergeleitet werden. In diesem Schritt sollte die Zahlungsart erneut ausgewählt werden können, damit der Benutzer seinen Einkauf fortsetzen kann. Ab diesem Punkt ist es benutzerabhängig wie der weitere Ablauf des Bestellprozesses von Statten geht.

4.2 Benutzung von „AfterPayment“ Methoden

4.2.1 Abbuchung

In Schritt 3 wird bereits die Verwendung von „AfterPayment“-Methoden erläutert. Falls die Methode `afterPaymentPrepareData` bereits die Abbuchung des Betrags oder ähnlich relevante Vorgänge durchgeführt hat, sodass die Benutzung der Methode `afterPaymentProcessData` nutzlos wäre, kann diese Methode vernachlässigt werden. Hierbei sollte nur darauf geachtet werden, dass der korrekte Statuscode als Rückgabewert angegeben wird.

4.2.2 Weiterleitung

Wenn in der Methode `afterPaymentPrepareData` zu einem Zahlungssystem weitergeleitet wurde, um dort die Abbuchung des Betrags manuell vorzunehmen, so muss `afterPaymentProcessData` aufgerufen werden um die Bestellung als erfolgreich abzuspeichern und ggf. den Betrag als bezahlt zu markieren.