WebsiteBaker Shop-Module Bakery

Add a new payment method plugin

This brief tutorial refers to **Bakery v1.7.6** or later.
Tutorial v0.7

Files used in the plugin

A Bakery payment method plugin consists of at least 6 files and a language directory:

- **index.php**
- **info.php**: Contains payment method settings like the payment gateway URL, variables and info about the payment method.
- **gateway.php**: This file provides public information about the payment gateway.
- **post_data.php**: Prepares the data sent to the payment gateway api. Will be included into `view_summary.php`.
- **report.php**: Depending of the payment gateway this file can be named differently. After payment transaction it receives a background response from the payment gateway api to set the payment status.
- **check_payment.php**: Checks whether the payment has been canceled by the user, the payment has been completed successfully or an error has occurred during payment processing.
- **languages-directory**: Contains all localisation files of the payment method.
- **icon.php**: This icon is used by the order administration of the Bakery backend to indicate the payment method that has been selected by the customer.
- **install.php / upgrade.php**: These files are optional and can be used to execute any code on payment gateway installation or upgrade.

Follow the steps below to implement a new payment gateway plugin:

1. Sign up to a test/sandbox account of your favorite payment service provider.

2. Read the developer's guide provided by the payment service provider. Especially read carefully about…

- how data is transfered to the payment gateway api (`HTML`, `XML`, `HTTP`, `SOAP`, ...).
- how the payment gateway api confirms the successfully completed payment.
- how the customer will be directed back from the payment gateway to the shop site.

3. Check the code of the existing Bakery payment method plugins provided in the `payment_methods` directory. Choose an existing plugin that will fit best the requirements of the payment gateway you are going to implement.

4. Make a copy and change the name of the directory to the name of your payment gateway.

5. info.php

- First provide the payment gateway URL by using the variable `$payment_gateway_url`.
- If you want to provide information about the payments gateway security then you can define an additional `$security_info_url` variable, that will be used by the `gateway.php` file.
- Then define the variables `$field_1` to `$field_6`. Provide variable values like merchant id, merchant key, user id, partner id, project id, customer email or whatever your payment service provider requires for a valid authentication.

The shop admin will then automatically be asked to enter this required information in the Bakery backend.

Make sure to create a corresponding variable in the payment method language files for each value of the variable `$field_X`.

Example: `$field_1 = 'email';`
The value `'`**`email`**`'` will be converted to uppercase `'`**`TXT_EMAIL`**`'`, which you must use in the language files array as follows:
`$MOD_BAKERY[$payment_method]['`**`TXT_EMAIL`**`'] = 'E-Mail';`

Second provide information about your payment method plugin using these variables:

```
$payment_method_name    = '';
$payment_method_version = '';
$payment_method_author  = '';
$requires_bakery_module = '';
```

6. gateway.php
This file contains some rows of a html table that provide public information about your payment method.

It will be displayed as part of the payment methods page `view_pay_methods.php`, where customers get information about the available payment methods and are able to select the payment method of their choice.

7. post_data.php
This file prepares the required information for the payment gateway api and will be included into `view_summary.php`.

Depending on the variety of data and the specified method how data will be transferred to the payment gateway api you have to modify this file. You might make use of a php class or any other sample code provided by your payment service provider to ease integration. Store the generated data in a variable called `$pay_gateway_data`.

8. Return URL
The return URL is used to direct customers back from the payment gateway to your shop upon payment completion. Params can be posted back using HTTP GET or POST method. Best use the `$setting_continue_url` var.

Please see the condensed PHP code example below taken from the `post_data.php` file:

```
$post_data = array(
     // [Put additional parameters here …]
     'return'        => $setting_continue_url.'?pm='.
$payment_method,
     'cancel_return' => $setting_continue_url.'?pm='.
$payment_method.'&status=canceled'
);
```

9. report.php (The file name can differ depending on the payment gateway)
This file is optional. It receives a background response from the payment gateway api using post/get, xml or whatever the api provides. After the payment has been completed successfully it updates the database field '`transaction_status`' to '`paid`', otherwise to '`pending`'. Default ist '`none`'. As an example please refer to the `ipn.php` file at the PayPal payment method.

10. check_payment.php
This file checks whether the payment has been `'canceled'` by user, the payment has been `'completed'` successfully, the payment is `'pending'` or whether an `'error'` has occurred during payment processing. It sets the var `$payment_status` that will be used by the `view_confirmation.php` file to show the corresponding message on the confirmation page.

If your payment service provider supports the payment status `'pending'`, it sends one email to the customer and another to the shop merchant. It displays the message "*Payment transaction will be processed shortly*". The shop email will contain a notification on the pending payment.

In this case do not forget to set the two language variables:
`$MOD_BAKERY[$payment_method]['TXT_PENDING']`
`$MOD_BAKERY[$payment_method]['TXT_TRANSACTION_STATUS']`
See PayPal as an example.

In case of `'success'` it concludes the order, sends emails to customer and merchant and displays the "*Payment completed successful*" message.

11. frontend.css

Add a new class to the frontend.css style sheet at the end of the BUTTONS section. Eg:

```
.mod_bakery_bt_pay_newpaymentgateway_f {
     width: 98%;
}
```

Replace `'newpaymentgateway'` by the directory name of your payment gateway.

12. icon.png

Provide a 16x16px png icon of your payment gateway.

13. Optional: Localized payment method name

If your payment method makes use of localized names, you just have to add the var `$MOD_BAKERY[$payment_method]['TXT_NAME']` to each payment method language file.
In this example I will use "bopis":

```
EN.php:
$MOD_BAKERY[$payment_method]['TXT_NAME'] = 'Buy online, pick up in store';
DE.php:
$MOD_BAKERY[$payment_method]['TXT_NAME'] = 'Abholung im
Ladengesch&auml;ft';
NL.php:
$MOD_BAKERY[$payment_method]['TXT_NAME'] = 'Afhalen in de winkel';
```

14. Optional: install.php / upgrade.php

Use an `install.php` / `upgrade.php` file if your payment method for example needs an additional field in the customer table or for whatever you need to do on installation or upgrade.