

Ομάδα: ILikeBrowsers

Συμμετέχοντες:

Γιώργος Πατσιαούρας

AM: 8100098

Έτος: 2014

Αναφορά

Εισαγωγικά:

Η συνεισφορά που ανέλαβα να πραγματοποιήσω αφορά στη πλατφόρμα Prestashop. Το Prestashop είναι μια πλατφόρμα ανάπτυξης ηλεκτρονικού καταστήματος η οποία σαν κύριο χαρακτηριστικό έχει την modular κατασκευή της. Διαθέτει πολλαπλά modules/προσθήκες για την επέκταση της λειτουργικότητας και πολλά από αυτά είναι επί πληρωμή στην ιστοσελίδα του Prestashop από διάφορους προγραμματιστές. Εγώ επικεντρώθηκα στα Prestashop modules, δηλαδή προσθήκες που κατασκευάζονται και υποστηρίζονται από την ομάδα του prestashop, βρίσκονται στο github και είναι δωρεάν καθώς έρχονται ενσωματωμένα στην κύρια διανομή του prestashop που είναι επίσης δωρεάν.

Τα modules που αποφάσισα να συνεισφέρω είναι τα Bank wire(Κατάθεση σε τράπεζα), Cash on Delivery(Αντικαταβολή) και PayPal (όλα αποτελούν τρόπους πληρωμής).

Το πρώτο ουσιαστικά εμφανίζει τα στοιχεία ενός λογαριασμού τραπεζής με σκοπό ο χρήστης χειροκίνητα να πραγματοποιήσει μια μεταφορά χρημάτων από τον δικό του λογαριασμό στο λογαριασμό του καταστήματος. Η συνεισφορά μου θα επιτρέπει στο διαχειριστή και ιδιοκτήτη του καταστήματος να προσθέτει πάνω από έναν τραπεζικούς λογαριασμούς.

Το δεύτερο αποτελεί τρόπο πληρωμής. Δηλαδή όταν ο πελάτης πραγματοποιεί την παραγγελία του να έχει τη δυνατότητα να επιλέξει αντικαταβολή και ουσιαστικά να πληρώσει για αυτήν όταν την παραλάβει. Η συνεισφορά μου είναι η επιλογή της αντικαταβολής να επιβαρύνει τον πελάτη με κάποιο ποσό. Αυτό είναι χρήσιμο γιατί πολλές μεταφορικές ή εταιρείες courier που χρεώνουν την αντικαταβολή έξτρα από την παράδοση.

Το τρίτο είναι πάλι τρόπος πληρωμής και αποτελεί την επιλογή για το χρήστη να πληρώσει μέσω του λογαριασμού του στο PayPal. Στο συγκεκριμένο module εντοπιζόταν ένα πρόβλημα κατά την μεταφορά από το eshop στη σελίδα πληρωμής του PayPal.

Για την πραγματοποίηση των αλλαγών στήθηκε ένα Prestashop τοπικά στον υπολογιστή μου χρησιμοποιώντας τοxampp(apache, mysql,etc) και μέσα στα αρχεία του prestashop και συγκεκριμένα στον φάκελο modules έγιναν clone τα repositories. Σκοπός ήταν να μπορώ να βλέπω τις αλλαγές και συγχρόνως με το development να γίνεται και το testing.

Αλλαγές:

Οι αλλαγές που προτάθηκαν ήταν οι εξής:

Διόρθωση PayPal Error 11812

Το λάθος αυτό εμφανιζόταν όταν ο χρήστης πήγαινε να ολοκληρώσει την αγορά του στο ηλεκτρονικό κατάστημα επιλέγοντας σαν τρόπο πληρωμής το PayPal. Για να γίνει η συναλλαγή ο χρήστης μεταφέρατε από την οθόνη του eshop στην σελίδα εισόδου του PayPal για να πραγματοποιήσει την αγορά του. Τα δεδομένα που αποστέλλονται στο PayPal από το eshop είναι το όνομα του προϊόντος, η περιγραφή του, το συνολικό ποσό πληρωμής και στοιχεία για την επιστροφή στο eshop. Το πρόβλημα εντοπιζόταν στο γεγονός ότι υπήρχε ένας περιορισμός από την μεριά του PayPal ότι η περιγραφή του προϊόντος έπρεπε να είναι μέχρι 127 χαρακτήρες. Το eshop είχε περιορισμό στους 100 χαρακτήρες. Μετά από αρκετό ψάξιμο στο developers guide στη σελίδα του PayPal διευκρινίστηκε ότι στην πραγματικότητα ο περιορισμός ήταν 127 single-byte characters. Οπότε οποιαδήποτε γλώσσα ξέφευγε από τους λατινικούς χαρακτήρες (π.χ. Ελληνικά ή Εβραϊκά) είχε double-byte χαρακτήρες με αποτέλεσμα το πραγματικό όριο από 100 έμοιαζε να είναι 200. Η λύση που βρέθηκε ήταν να μειωθεί το όριο από 100 σε 50 οπότε η double-characters να γίνουν 100.

Συγκεκριμένα, οι αλλαγές ήταν:

```
private function setProductsList(&$fields, &$index, &$total)
    if (isset($product['attributes']) && (empty($product['attributes']) === false))
        $fields['L_PAYMENTREQUEST_0_NAME'][$index] .= ' - ' . $product['attributes'];

    $fields['L_PAYMENTREQUEST_0_DESC'][$index] = Tools::substr(strip_tags($product['description_short']), 0, 100). '...';
    $fields['L_PAYMENTREQUEST_0_DESC'][$index] = Tools::substr(strip_tags($product['description_short']), 0, 50). '...';

    $fields['L_PAYMENTREQUEST_0_AMT'][$index] = Tools::ps_round($product['price_wt'], $this->decimals);
    $fields['L_PAYMENTREQUEST_0_QTY'][$index] = $product['quantity'];
private function setDiscountsList(&$fields, &$index, &$total)

    $fields['L_PAYMENTREQUEST_0_NAME'][$index] = $discount['name'];
    if (isset($discount['description']) && !empty($discount['description']))
        $fields['L_PAYMENTREQUEST_0_DESC'][$index] = Tools::substr(strip_tags($discount['description']), 0, 100). '...';
        $fields['L_PAYMENTREQUEST_0_DESC'][$index] = Tools::substr(strip_tags($discount['description']), 0, 50). '...';

    /* It is a discount so we store a negative value */
    $fields['L_PAYMENTREQUEST_0_AMT'][$index] = - 1 * Tools::ps_round($discount['value_real'], $this->decimals);
```

Και το Pull Request με λεπτομερή εξήγηση.

PayPal Error 11812 Fix: Non Latin Characters #336

Open giorgosp92 wants to merge 1 commit into PrestaShop:master from giorgosp92:master

Conversation 0 Commits 1 Files changed 1

giorgosp92 commented 4 hours ago

PayPal can handle up to 127 single byte characters for the description of a product. Non Latin character languages like Greek or Hebrew have characters that are double-byte. So the 100 character limit, in Greek seems like a 200 single-byte character limit and that's the reason of the Error 11812. This limits the description to 50 characters so that the double-characters will be ok.

PayPal Error 11812 Fix: Non Latin Characters b8cd48

Υποστήριξη πολλαπλών λογαριασμών στην Κατάθεση σε τράπεζα

Για την αλλαγή αυτή έγιναν οι εξής αλλαγές:

1. Ρυθμίστηκε το module ώστε κατά την εγκατάσταση και την απεγκατάσταση του να δημιουργεί και να καταστρέφει αντίστοιχα έναν πίνακα στη βάση δεδομένων ο οποίος κρατάει τα στοιχεία των λογαριασμών τραπεζών.
2. Στο διαχειριστικό του eshop, μέσα στη σελίδα διαχείρισης του module προστέθηκε ένας πίνακας στον οποίον φαίνονται οι ήδη υπάρχοντες δηλωμένοι λογαριασμοί. Ο χρήστης εδώ έχει την δυνατότητα να διαγράψει κάποιον από τους υπάρχοντες λογαριασμούς. Ακόμη η φόρμα για την καταχώρηση νέου λογαριασμού ρυθμίστηκε ώστε να κάνει εισαγωγή μια νέας γραμμής στη βάση δεδομένων στην αντίστοιχο πίνακα. Για οποιαδήποτε ενέργεια του χρήστη δημιουργήθηκε το αντίστοιχο feedback μήνυμα για να είναι σίγουρος ότι όλα πήγαν όπως έπρεπε.
3. Τέλος, έγιναν οι απαραίτητες αλλαγές στο Front end δηλαδή ο χρήστης να μπορεί να δει όλους τους λογαριασμούς και αντίστοιχα να επιλέξει ποιος του ταιριάζει.

Τεχνικές Σημειώσεις:

Εγκατάσταση του Module (εκτελείται η μέθοδος install())

```
public function install()
{
    if (!parent::install() || !$this->registerHook('payment') || !$this->registerHook('paymentReturn'))
        return false;
    return Db::getInstance()->execute('
        CREATE TABLE IF NOT EXISTS `'.DB_PREFIX_.'bankwire` (
            `id` int(6) NOT NULL AUTO_INCREMENT,
            `id_shop` INTEGER UNSIGNED NOT NULL DEFAULT \'1\',
            `id_shop_group` INTEGER UNSIGNED NOT NULL DEFAULT \'1\',
            `owner` varchar(255) NOT NULL,
            `details` text NOT NULL,
            `address` text NOT NULL,
            PRIMARY KEY(`id`)
        ) ENGINE='.MYSQL_ENGINE_.' default CHARSET=utf8');
}
```

Κατά την εγκατάσταση του Module προστέθηκε το query για δημιουργία του πίνακα στη βάση δεδομένων.

*Για την επικοινωνία με τη βάση χρησιμοποιείται η κλάση Db.php

Απεγκατάσταση του Module

```
public function uninstall()
{
    Db::getInstance()->execute('DROP TABLE '._DB_PREFIX_.'bankwire');
    return parent::uninstall();
}
```

*Με την ίδια λογική κατά την απεγκατάσταση διαγράφουμε το module.

Δημιουργία του πίνακα που εμφανίζει τις εγγραφές στη βάση δεδομένων (Λογαριασμοί)

```
public function renderList() {
    $records_table = array(
        'id' => array(
            'title' => $this->l('Id'),
            'width' => 140,
            'type' => 'text',
        ),
        'owner' => array(
            'title' => $this->l('Account Owner'),
            'width' => 140,
            'type' => 'text',
        ),
        'details' => array(
            'title' => $this->l('Details'),
            'width' => 140,
            'type' => 'text',
        ),
        'address' => array(
            'title' => $this->l('Bank address'),
            'width' => 140,
            'type' => 'text',
        )
    );

    $helper = new HelperList();
    $helper->shopLinkType = '';
    $helper->simple_header = true;
    $helper->_select = $this->getBankAccounts();
    $helper->actions = array('delete');
    $helper->identifier = 'id';
    $helper->show_toolbar = true;
    $helper->defaultOrderBy = 'id';
    $helper->title = $this->l('Existing Accounts');

    $helper->token = Tools::getAdminTokenLite('AdminModules');
    $helper->currentIndex = AdminController::$currentIndex.'&configure='.$this->name;
    $helper->table = "bankwire";
    return $helper->generateList($helper->_select, $records_table);
}
```

Δήλωση πεδίων πίνακα
χρησιμοποιώντας τη κλάση
HelperList.php

Ενέργειες για κάθε (row)
στη βάση δεδομένων

...

Οι περισσότερες εντολές από αυτές είναι τυπικές όπως το AdminToken για να επιβεβαιωθεί ότι το request είναι ασφαλές ή το simple_header που αναφέρεται στο design του πίνακα. Αξίζουν να τονισθούν οι εντολές \$helper->_select η οποία αντλεί τα δεδομένα του πίνακα από την μέθοδο getBankAccounts(). Η μέθοδος αυτή προστέθηκε για την ανάκτηση των γραμμών από τη βάση

```
public function getBankAccounts() {
    $sql = 'SELECT *
    FROM '._DB_PREFIX_.'bankwire';
    return Db::getInstance()->executeS($sql);
}
```

Στην επικοινωνία αυτή με τη βλέπουμε ότι χρησιμοποιούμε μια άλλη μέθοδο της Db.php η οποία είναι η executeS. Το όνομα προκύπτει από το executeSelect και χρησιμεύει όταν χρειαζόμαστε ανάκτηση από έναν ή παραπάνω πίνακες. Η μεταβλητή `_DB_PREFIX` είναι ένα πρόθεμα που βάζει ο χρήστης σε όλα τα tables της βάσης δεδομένων κατά την εγκατάσταση της πλατφόρμας. Όσο πιο σπάνιο είναι τόσο πιο ασφαλές θεωρείται η βάση δεδομένων.

*** Η σημειώση αφορά την εντολή `$helper->generateList` η οποία αρχικοποιεί τον πίνακα html με τα δεδομένα που έχουμε επιλέξει. Δέχετε δύο attributes, τις γραμμές από τον πίνακα της βάσης, και τα πεδία που δηλώσαμε που είναι ουσιαστικά οι στήλες του πίνακα(html).

Ακόμα προστέθηκαν δύο μέθοδοι

```
public function deleterow($id) {
    if ($id != null)
    {
        Db::getInstance()->execute('DELETE FROM '._DB_PREFIX_.'bankwire WHERE `id` = '.$id);
        $this->_html .= $this->displayConfirmation($this->l('Bank Account was deleted successfully'));
    }
}

public function checkAccounts() {
    $sql = 'SELECT COUNT(*) FROM '._DB_PREFIX_.'bankwire';
    $totalaccounts = Db::getInstance()->getValue($sql);
    error_log($totalaccounts);
    return $totalaccounts;
}
```

Η deleterow δέχεται σαν όρισμα το id της γραμμής του πίνακα της βάσης που θέλει να διαγράψει ο χρήστης και το διαγράφει. Βλέπουμε ότι με το που γίνει η διαγραφή αποθηκεύεται μήνυμα επιτυχίας για να εμφανιστεί στο χρήστη όταν φορτώσει η σελίδα.

Η checkAccounts χρησιμοποιεί μια μέθοδο της Db.php η οποία παίρνει μια μοναδική τιμή. Στη συγκεκριμένη περίπτωση παίρνουμε τον αριθμό των εγγραφών της βάσης δεδομένων. Για μελλοντικό έλεγχο αν ο χρήστης έχει δηλώσει η όχι κάποιο λογαριασμό και επομένως αν θα του εμφανιστεί μήνυμα ότι το module θέλει ρύθμιση ή όχι.

Οι μεταβλητές του τύπου `$this->l('Random test')`; Χρησιμεύουν στην εκτύπωση μηνυμάτων που μπορούν να μεταφραστούν μέσα από τη πλατφόρμα σε πολλαπλές γλώσσες.

Ακόμα αλλάχτηκε η μέθοδος `_PostProcess()` η οποία καλείται από τη μέθοδο `getContent` όταν πατηθεί το κουμπί για δημιουργία λογαριασμού.

```
if (Tools::isSubmit('btnSubmit')) {
    Db::getInstance()->execute('INSERT INTO '._DB_PREFIX_.'bankwire (id_shop, id_shop_group, owner, details, address)
VALUES
('.$this->context->shop->id.',
 '$this->context->shop->id_shop_group.',\''
 .Tools::getValue('BANK_WIRE_OWNER').'\',\''
 .Tools::getValue('BANK_WIRE_DETAILS').'\',\''
 .Tools::getValue('BANK_WIRE_ADDRESS').'\''
)') or die(Db::getInstance()->getMsgError());
}
```

Βλέπουμε ότι γίνεται έλεγχος μέσω της κλάσης Tools.php του prestashop με την οποία χρησιμοποιώντας τη μέθοδο `isSubmit` βλέπουμε αν έχει πατηθεί κάποιο κουμπί τύπου "Submit", με όνομα `btnSubmit` (έτσι λέγεται το κουμπί προσθήκης νέου λογαριασμού). Οπότε και γίνεται η εισαγωγή στη βάση δεδομένων.

Τέλος, η πιο βασική μέθοδος είναι η `getContent()`. Η μέθοδος αυτή χρησιμοποιείται και για να υποδηλώσει ότι το module αυτό έχει σελίδα επεξεργασίας. Αν δεν υπάρχει σημαίνει ότι το module είναι παθητικό και ότι ο χρήστης δεν μπορεί να το επηρεάσει εκτός από επέμβαση στον πηγαίο του κώδικα.

```
public function getContent()
{
    if (Tools::isSubmit('btnSubmit'))
    {
        $this->_postValidation();
        if (!count($this->_postErrors))
            $this->_postProcess();
        else
            foreach ($this->_postErrors as $err)
                $this->_html .= $this->displayError($err);
    }
    elseif (Tools::isSubmit('deletebankwire'))
    {
        //action button of helper list "delete" was pressed
        $this->deleterow(Tools::getValue('id'));
    }
    else
        $this->_html .= '<br />';

    $this->_html .= $this->_displayBankWire();
    $this->_html .= $this->renderForm();
    $this->_html .= $this->renderList();

    return $this->_html;
}
```

Η μέθοδος αυτή υπήρχε. Οι προσθήκες που έγιναν ήταν το:

```
elseif (Tools::isSubmit('deletebankwire'))
{
    //action button of helper list "delete" was pressed

    $this->deleterow(Tools::getValue('id'));
}
```


Που ελέγχει αν πατήθηκε το κουμπί delete για κάποια από τις γραμμές του πίνακα με τους λογαριασμούς και αν έχει πατηθεί καλεί τη μέθοδο `deleterow` περνώντας από το GET Request τη τιμή `id` και δίνοντας τη στη μέθοδο.

Η τελευταία προσθήκη είναι η `$this->renderList()`; που αρχικοποιεί το πίνακα με τις εγγραφές που είδαμε παραπάνω.

Όσον αφορά την προσαρμογή του FrontEnd προστέθηκε ένα `foreach` (Για κάθε) αντικείμενο του πίνακα που περιέχει τις εγγραφές να τυπώνει τα στοιχεία της εγγραφής

```
{foreach from=$bankAccounts item=account}
<br />
<br>-- {1 s='Owner' mod='bankwire'}: <strong>{$account.owner}</strong>
<br>-- {1 s='Details' mod='bankwire'}: <strong>{$account.details}</strong>
<br>-- {1 s='Bank Name' mod='bankwire'}: <strong>{$account.address}</strong>
{/foreach}
```

Τελική μορφή του module στο διαχειριστικό του Prestashop


 This module allows you to accept secure payments by bank wire.
If the client chooses to pay by bank wire, the order's status will change to 'Waiting for Payment'.
That said, you must manually confirm the order upon receiving the bank wire.

CONTACT DETAILS



Account owner

Details
Such as bank branch, IBAN number, BIC, etc.

Bank address

[Αλλαξε το λογότυπό](#) 

EXISTING ACCOUNTS [HelperList.php](#)

Id	Account Owner	Details	Bank address	
3	asd	asd	asd	 Delete
4	sadasd	asdas	dasdasd	 Delete

CONFIGURATION [Actions](#)

[Manage translations](#)

Ο πίνακας που γεννήθηκε από τη μέθοδο renderList χρησιμοποιώντας την HelperList.php του prestashop.

Προσθήκη λειτουργίας χρέωσης της Αντικαταβολής για ορισμένους μεταφορείς

Για την αλλαγή αυτή έγιναν τα εξής βήματα:

Αρχικά κατασκευάστηκε το back end. Με την εγκατάσταση/απεγκατάσταση του module, δημιουργείται/ καταστρέφεται ένας πίνακας στην βάση δεδομένων που σκοπό έχει να αποθηκεύει δεδομένα για ποιους μεταφορείς θα υπάρχει έξτρα χρέωση. Αυτό γίνεται αποθηκεύοντας το ID του Μεταφορέα (π.χ. ACS id=1) και το ποσό της έξτρα χρέωσης π.χ. 3€. Το συγκεκριμένο module πριν την επεξεργασία θεωρούταν παθητικό δηλαδή ο χρήστης είχε τη δυνατότητα είτε να το έχει ενεργοποιημένο είτε απενεργοποιημένο και δεν είχε σελίδα διαχείρισης. Για τις αλλαγές, δημιούργησα σελίδα διαχείρισης η οποία αποτελούνταν από μια φόρμα και ένα πίνακα. Με τη φόρμα προσθέτει ο χρήστης δεδομένα στην βάση δεδομένων και στον πίνακα βλέπει τους ήδη υπάρχοντες κανόνες. Στον πίνακα ο χρήστης έχει τη δυνατότητα να διαγράψει υπάρχοντες κανόνες.

Τεχνικά:

Προστέθηκε εξολοκλήρου η μέθοδος getContent() για να αποκτήσει το module σελίδα διαχείρισης

```

public function getContent()
{
    if (Tools::isSubmit('btnSubmit'))
    {
        if (Tools::getValue('id_carrier') && Tools::getValue('extra_fee'))
        {
            $this->newRule(Tools::getValue('id_carrier'), Tools::getValue('extra_fee'));
            $this->_html = $this->displayConfirmation($this->l('New Rule added successfully'));
        }
    }
    elseif (Tools::isSubmit('deletecashondelivery'))
    {
        //action button of helper list "delete" was pressed
        $this->deleterow(Tools::getValue('id'));
    }

    $this->_html .= $this->renderForm();
    $this->_html .= $this->renderList();
    return $this->_html;
}

```

Υπάρχουν πολλές ομοιότητες με το module Bankwire καθώς η λειτουργία τους είναι παρόμοια στο κώδικα του BackEnd.(Εισαγωγή στη βάση, Διαγραφή, Ανάκτηση)

Δυο ιδιαίτερες μέθοδοι που προστέθηκαν ήταν οι:

```

public function getCarriername($id_carrier)
{
    $results = Db::getInstance()->executeS('SELECT name FROM '._DB_PREFIX_.'carrier WHERE `id_carrier` = '.$id_carrier);
    return $results[0]['name'];
}

public function getExtrafee($id_carrier)
{
    $results = Db::getInstance()->executeS('SELECT extra_fee FROM '._DB_PREFIX_.'cashondelivery WHERE `id_carrier` = '.$id_carrier);
    return $results[0]['extra_fee'];
}

```

Οι δύο αυτές μέθοδοι χρησιμεύουν για την επικοινωνία με το Front end. Συγκεκριμένα δέχονται ως όρισμα το id του τρόπου αποστολής που επέλεξε ο χρήστης κατά την διάρκεια της παραγγελίας και επιστρέφουν η πρώτη το όνομα του μεταφορέα και η δεύτερη την έξτρα χρέωση που πρέπει να προστεθεί στο συνολικό ποσό. Το id του μεταφορέα ανακτάται από την μνήμη του περιηγητή του χρήστη. Τα αποτελέσματα των δυο μεθόδων στέλνονται στο αρχείο που ευθύνεται για την εμφάνιση στο χρήστη.

Εδώ γίνεται η προετοιμασία για την αποστολή στο αρχείο εμφάνισης payment.tpl

```

$smarty->assign(array(
    'this_path' => $this->_path, //keep for retro compat
    'this_path_cod' => $this->_path,
    'this_path_ssl' => Tools::getShopDomainSsl(true, true)._PS_BASE_URI_.modules.'/'.$this->name.'/'.
    'carrier_name' => $this->getCarriername($this->context->cart->id_carrier),
    'extra_fee' => $this->getExtrafee($this->context->cart->id_carrier)
));
return $this->display(__FILE__, 'payment.tpl');

```

Οι νέες μεταβλητές περνάνε ως carrier_name και extra_fee

Τυπώνονται στο χρήστη με τον απλό τρόπο.

```

<br />{1 s='Pay with cash on delivery (COD)' mod='cashondelivery'}
<br />{1 s='You pay for the merchandise upon delivery' mod='cashondelivery'}
<br />{1 s='For the carrier' mod='cashondelivery'} "{$carrier_name}" {1 s='There is an extra fee of' mod='cashondelivery'} {$
extra_fee}
{"$total_price"}
<br style="clear:both;" />

```

Σημαντικό:

Το module Αντικαταβολή (CashOnDelivery) δεν ολοκληρώθηκε σε πλήρη βαθμό επειδή το prestashop λειτουργεί με πολλαπλά νομίσματα για τον πελάτη. Επομένως απαιτείται μεγάλη προσοχή.

Τεχνικές Λεπτομέρειες:

Δόθηκε μεγάλη έμφαση στην αξιοποίηση των εργαλείων που παρέχονταν από την πλατφόρμα για την πραγματοποίηση των εργασιών όπως οι κλάσεις Db.php, HelperList.php, Tools.php. Αξίζει να αναφερθεί ότι παρόλο την ύπαρξη ενός βασικού Developers Guide από την ομάδα ανάπτυξης του Prestashop οι περισσότερες λειτουργίες εντοπίστηκαν μελετώντας τις μεθόδους των κλάσεων αυτών καθώς το documentation ήταν ελλιπές.

Ακόμα, τηρήθηκε σχεδόν σε πλήρη βαθμό και όπου ήταν εφικτό το συντακτικό που έμμεσα προωθεί η ομάδα ανάπτυξης. Π.χ. Στο άνοιγμα μιας μεθόδου το «{» πηγαίνει από κάτω και όχι στην ίδια ευθεία.

Για την αλλαγή στο module **Bankwire**

Έγιναν 7 commits. Κρίθηκε ότι δεν υπήρχε λόγος για περισσότερα καθώς δεν θα λειτουργούσαν οι επιμέρους αλλαγές. Συνολικά προστέθηκαν 206 γραμμές κώδικα και αφαιρέθηκαν 46. Οι 46 ήταν ουσιαστικά η προηγούμενη λειτουργικότητα.

Support for multiple Accounts #1

The screenshot shows a GitHub pull request interface. At the top, a green button says 'Open' and the text indicates 'giorgosp92 wants to merge 8 commits into Prestashop:master from giorgosp92:master'. Below this, there are tabs for 'Conversation' (0), 'Commits' (8), and 'Files changed' (4). The main content area shows a comment from 'giorgosp92' dated 'Saturday at 12:41 PM'. The comment text is: 'Hello, Added support for multiple accounts on bankwire by saving details in a database table. Now users can choose from multiple accounts depending on the Bank that they use. Thank you'. Below the comment, there is a list of 8 commits by 'giorgosp92' dated 'Jun 1'. Each commit has a green icon, a description, and a commit hash. The commits are: 1. 'This my first step and my first commit on github' (fb5000c), 2. 'Added in the creation of the database table when the module is being ...' (35acbc4), 3. 'Added unistall and insert on submit on database' (221b6d1), 4. 'Added not working table' (b719e5b), 5. 'Bankwire completed. Functions created are, New account and delete acc...' (87ea122), 6. 'Final Commit everything is ready' (f02870b), 7. 'Cleaned up old stuff' (1328f4a), and 8. 'Removed old commented values' (0a52d3f). At the bottom, there is a note: 'Add more commits by pushing to the master branch on giorgosp92/bankwire.'

Μέχρι τη στιγμή που γράφτηκε αυτό το Report δεν υπήρξε κάποια απάντηση από την ομάδα ανάπτυξης του Prestashop.

Σύνοψη:

Η εργασία αυτή με έφερε σε επαφή με τον κόσμο του Github. Παρόλο που το χρησιμοποιούσα καθημερινά για την εύρεση βιβλιοθηκών πότε δεν είχα γίνει «μέρος» του, και ποτέ δεν είχα δει την μεριά των contributors. Είναι ένα αξιοθαύμαστο εργαλείο και το git αλλά και η συνεργασία και η επαφή που προσφέρει το Github.