FREE eBook

LEARNING stripe-payments

#stripe-payments

Table of Contents

About

Chapter 1: Getting started with stripe-payments

Remarks

This section provides an overview of what stripe-payments is, and why a developer might want to use it.

It should also mention any large subjects within stripe-payments, and link out to the related topics. Since the Documentation for stripe-payments is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Detailed instructions on getting stripe-payments set up or installed.

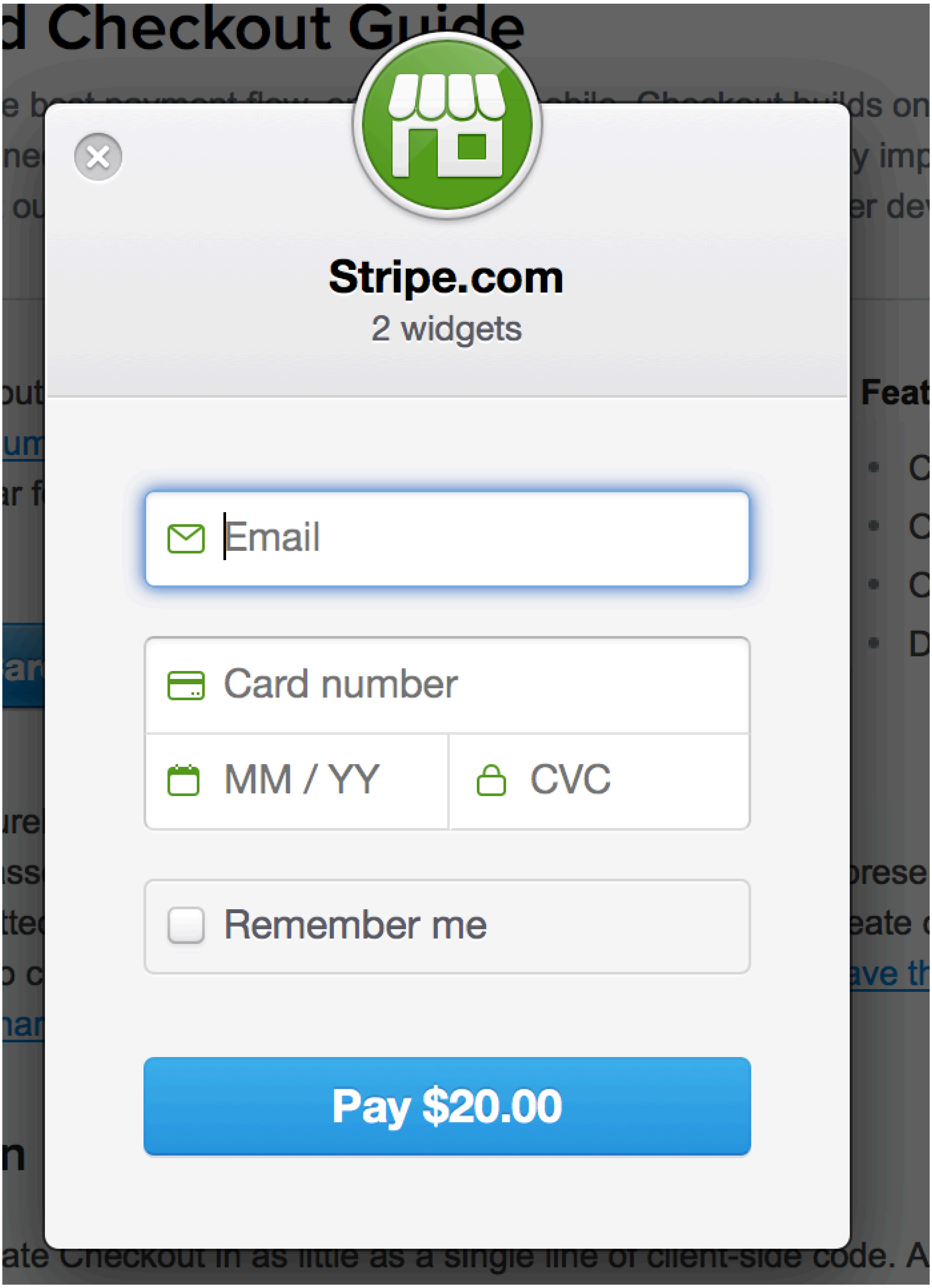
Embedded Stripe Payment Modal

Register a production/sandbox account at https://dashboard.stripe.com/register

Insert below code into your webpage where you want to have a checkout button.

```
<form action="/charge" method="POST">
  <script
    src="https://checkout.stripe.com/checkout.js" class="stripe-button"
    data-key="pk_test_6pRNASCoBOKtIshFeQd4XMUh"
    data-amount="2000"
    data-name="Stripe.com"
    data-description="2 widgets"
    data-image="/img/documentation/checkout/marketplace.png"
    data-locale="auto">
  </script>
</form>
```

Result:


Stripe.com
2 widgets
Email
Card number
MM / YY
CVC
Remember me
Pay $20.00

Hello World in Python

An example how to run stripe out of the box with wsgi from a single file.

At first, please install the python stripe API, i.e. with pip:

```
pip install --user stripe
```

Create `payment.py` which creates a WSGI webserver at port 8000 out of the box

```
html = """
<html>
<body>
     <p>%(output)s</p>
</body>
</html>
"""

form = """
<form action="" method="POST">
    <script
        src="https://checkout.stripe.com/checkout.js" class="stripe-button"
        data-key="pk_test_6pRNASCoBOKtIshFeQd4XMUh"
        data-amount="999"
        data-name="Stripe.com"
        data-description="Hello World"
        data-locale="auto">
    </script>
</form>
"""

def application(environ, start_response):
        try:
                request_body_size = int(environ.get('CONTENT_LENGTH', 0))
        except (ValueError):
                request_body_size = 0
        request_body = environ['wsgi.input'].read(request_body_size)
        post = parse_qs(request_body)
        out = ''
        if post:
                print post
                token = post.get('stripeToken', [''])[0]
                token = escape(token)
                if token:
                        import stripe
                        stripe.api_key = "sk_test_BQokikJOvBiI2HlWgH4olfQ2"
                        try:
                                charge = stripe.Charge.create(
                                        amount="999",
                                        currency="usd",
                                        source=token,
                                        description="Hello World",
                                        )
                                out = '<pre>charge: %s</pre>' % (charge,)
                        except Exception as e:
                                print 'Exception %s' % (str(e),)
                else:
                        out = 'missing in post: token'
        else:
                out = form
        response_body = html % {
                'output': out,
        }
        status = '200 OK'
        response_headers = [('content-type', 'text/html;charset=utf-8')]
        start_response(status, response_headers)
        return [response_body]

from wsgiref.simple_server import make_server
from cgi import parse_qs, escape
httpd = make_server('', 8000, application)
httpd.serve_forever()
```

Please note:

- the frontend form contains the **public key**
- the backend charge part contains the **secret key**.

Run the script

```
python payment.py
```

Navigate with your browser to

```
http://localhost:8000/
```

After clicking the Pay-Button and entering the credit card number (4242424242424242) the form is posted with the token. So the payment could be processed and finally the `charge` object will be printed into the browser, which contains:

```
...
"paid": true,
"description": "Hello World",
"status": "succeeded"
```

Resources and further reading:

- WSGI post: http://wsgi.tutorial.codepoint.net/parsing-the-request-post
- Frontend form: https://stripe.com/docs/checkout/tutorial
- Backend charge: https://stripe.com/docs/charges

Introduction to Stripe's API

A typical payment flow with Stripe can be divided in two steps:

1. Client-side, in your frontend (HTML + Javascript) code, you collect the customer's payment information using Stripe's prebuilt Checkout form or Elements form field(s). This will return a token that you then send to your server.

2. Server-side, in your backend code (in PHP, Python, Ruby, or whichever server-side programming language you prefer), you use the token in a charge creation request to actually charge the card.

The point of this 2-step flow is that your server only works with card tokens and never with raw card information. This means you never have access to card numbers, which greatly eases the burden of PCI compliance.

Stripe's documentation is pretty extensive and includes many examples and tutorials -- make sure to check it out!

Read Getting started with stripe-payments online: https://riptutorial.com/stripe-payments/topic/1672/getting-started-with-stripe-payments

Chapter 2: PHP-Stripe Connect documentation for Symfony2

Parameters

Parameter	Details
amount	required - A positive integer in the smallest currency unit (e.g., 100 cents to charge $1.00 or 100 to charge ¥100, a 0-decimal currency) representing how much to charge the card. The minimum amount is $0.50 US or equivalent in charge currency.
currency	required - 3-letter ISO code for currency.
description	optional, default is null - An arbitrary string which you can attach to a charge object. It is displayed when in the web interface alongside the charge. Note that if you use Stripe to send automatic email receipts to your customers, your receipt emails will include the description of the charge(s) that they are describing.
receipt_email	optional default is null - The email address to send this charge's receipt to. The receipt will not be sent until the charge is paid. If this charge is for a customer, the email address specified here will override the customer's email address. Receipts will not be sent for test mode charges. If receipt_email is specified for a charge in live mode, a receipt will be sent regardless of your
exp_month	required - Two digit number representing the card's expiration month.
exp_year	required - Two or four digit number representing the card's expiration year.
number	required - The card number, as a string without any separators.
cvc	usually required -Card security code. Required unless your account is registered in Australia, Canada, or the United States. Highly recommended to always include this value.

Examples

Symfony2- Stripe Integration Example

Download the Stripe API Library and place it in vendor Folder

source : [https://github.com/stripe/stripe-php][1] include the library in your controller

```
use Stripe\BalanceTransaction;
use Stripe\Charge;
use Stripe\Stripe;
require_once('../vendor/stripe/init.php');
```

set the strip key

```
 \Stripe\Stripe::setApiKey('stripe_secret_key');
```

Call the charge function for transaction

```
$card = array(
    'number' =>'cardccn',
    'cvc' =>'cardcvc',
    'exp_month' => 'expMonth',
    'exp_year' => 'expYear',
    );

$charge = Charge::create(
    array(
        'amount' => ('amount') * 100, // Amount will store in cent in Stripe Account
        'currency' => 'usd',
        'card' => $card,
        'description' => '$data['description',
        'receipt_email'=>'receipt_email'
        )
    );
```

get the details of charge

```
 $data = Charge::retrieve('ch_%');
```

Read PHP-Stripe Connect documentation for Symfony2 online: https://riptutorial.com/stripe-payments/topic/6633/php-stripe-connect-documentation-for-symfony2

Chapter 3: Stripe Add multiple card to Same User

Examples

Create customer in stripe

```
public function createCustomer($data , $token)//pass form data and token id
{
    $customer=Customer::create(array(
    "email"=>$data['email'],
    "description" => $data['name'],
    "source" => $token // obtained with Stripe.js
    ));
    return $customer['id'];
}
```

For more Information follow this Link

How to retrive customer And add cards in Stripe

```
public function addCard($cust_id, $token)
{
    $retriveResult=Customer::retrieve($cust_id);
    $tokendata = Token::retrieve($token);
    $newcard = $tokendata['card'];
    $flag = 1;

    foreach ($retriveResult['sources']['data'] as $card) {
        if($card['fingerprint'] === $newcard['fingerprint'])
        {
            $cardid = $card['id'];
            $flag = 0;
            break;
        }
    }

    if($flag)
    {
        $savecard = $retriveResult->sources->create(array("source" =>$token));
        $cardid = $savecard['id'];
    }
    return $cardid;
}
```

Read Stripe Add multiple card to Same User online: https://riptutorial.com/stripe-payments/topic/9547/stripe-add-multiple-card-to-same-user

Chapter 4: Stripe.Net Introduction

Syntax

- var stripeSubscriptionOptions = new StripeSubscriptionCreateOptions();

  //create a variable to hold options object

  ```
          stripeSubscriptionOptions.Quantity = model.update;
  ```

  //example option of quantity of seats for a subscription

  ```
          var subscriptionService = new StripeSubscriptionService();
  ```

  //create a service to make the API call

  ```
          var stripeSubscription = subscriptionService.Create(user.CustomerIdentifier,
  planId,
  ```

  stripeSubscriptionOptions);

  // service.create(string CustID, string PlanID, Object SubscriptionOptions)

  //Customer ID should be saved from your database, you can retrieve planID from stripe using a PlanService and create the options object like above. If you NuGet the Stripe.Net intellisense works for these as well.

Remarks

Somewhere in the beginning of your controller you should call

```
StripeConfiguration.SetApiKey(YOUR SECRET KEY VAR);
```

and for data safety it should be a value hidden as a secret in appsettings

if you do not set the API key you will not be able to modify subscriptions or create customers

Examples

Starting with Stripe.Net in ASP.Net Core 1.0

https://github.com/jaymedavis/stripe.net is a great starting point.
Assuming you are using MVC w/Razor you need to have a few things in your View page

```
<script type="text/javascript" src="https://js.stripe.com/v2/"></script>
```

This script calls upon the stripe.js to handle creating a token.

```
<script type="text/javascript">
        Stripe.setPublishableKey('YOUR STRIPE PUBLIC KEY');

var stripeResponseHandler = function (status, response) {
    var $form = $('#payment-form');

    if (response.error) {
        // Show the errors on the form
        $form.find('.payment-errors').text(response.error.message);
        $form.find('button').prop('disabled', false);
    } else {
        // token contains id, last4, and card type
        var token = response.id;
        // Insert the token into the form so it gets submitted to the server
        $form.append($('<input type="hidden" asp-for="stripeToken" />').val(token));
        // and re-submit
        $form.get(0).submit();
    }
};

jQuery(function ($) {
    $('#payment-form').submit(function (e) {
        var $form = $(this);

        // Disable the submit button to prevent repeated clicks
        $form.find('button').prop('disabled', true);

        Stripe.card.createToken($form, stripeResponseHandler);

        // Prevent the form from submitting with the default action
        return false;
    });
});
</script>
```

To add the token to a model make sure you change

```
$form.append($('<input type="hidden" asp-for="stripeToken" />').val(token));
```

to reflect your model. The form should look like this one.

```
 <form asp-action="confirm" method="POST" id="payment-form">
            <span class="payment-errors"></span>

            <div class="row">
                <label>
                    <span>Card Number</span>
                    <input type="text" data-stripe="number" value="4242424242424242">
                </label>
            </div>

            <div class="row">
                <label>
                    <span>CVC</span>
                    <input type="text" data-stripe="cvc" value="123">
                </label>
            </div>

            <div class="row">
                <label>
                    <span>Expiration (MM/YYYY)</span>
                    <input type="text" data-stripe="exp-month" value="12">
                </label>
                <input type="text" data-stripe="exp-year" value="2020">
            </div>

            <button type="submit">Buy Now</button>
        </form>
```

The controller takes one of the users and checks for a CustomerIdentifier which is the Id of the customer given by stripe. If this isn't saved in the database then it works on creating a customer for them

```
 public async Task<IActionResult> Index(OrderViewModel model)   //HOME PAGE beginning of 
managing subs
    {
        //get the user and their ID
        var user = await GetCurrentUserAsync();
        var userId = user?.Id;
        // If they have a customer Identifier use it
        if (!string.IsNullOrEmpty(user.CustomerIdentifier))  //eventually if user has a saved 
card as well
        {
            //Create the API call to subscription and put its response in a list
            var subscriptionService = new StripeSubscriptionService();
            IEnumerable<StripeSubscription> response = 
subscriptionService.List(user.CustomerIdentifier);

            ViewBag.Subscription = response;

            ViewBag.Customer = user.CustomerIdentifier;



        }
        ModelState.Clear();
        return View(model);
    }
 public async Task<IActionResult> Confirm(OrderViewModel model, string stripeToken)  // CREATE 
CHARGE NO CARD/ NO CUSTOMER
    {
        model.stripeToken = stripeToken;
        //get the user and their ID
        var user = await GetCurrentUserAsync();
        var userId = user?.Id;
        var planId = "YOUR PLAN ID HERE";  //plan ID only 1 atm but will need to be a dynamic 
plan ID list later
        // If they have a customer Identifier use it
        if (!string.IsNullOrEmpty(user.CustomerIdentifier))
        {
            //Create the API call to subscription and put its response in a list
            //Use the subscription options to apply a quantity to the initial subscription for 
seats
            var stripeSubscriptionOptions = new StripeSubscriptionCreateOptions();
            stripeSubscriptionOptions.Quantity = model.update;
            var subscriptionService = new StripeSubscriptionService();
            var stripeSubscription = subscriptionService.Create(user.CustomerIdentifier,
planId, stripeSubscriptionOptions);
            //save Subscriptions data here

            ModelState.Clear();
            //await SaveSubscription(stripeSubscription, user);
            await _userManager.UpdateAsync(user);
        }
        else // Customer is new and doesn't have an ID
        {
            //Create API options
            var customer = new StripeCustomerCreateOptions();

            // Add option values
            customer.Email = $"{user.Email}";
            customer.Description = $"{user.Email} [{userId}]";
            customer.PlanId = planId;

            customer.SourceToken = model.stripeToken;
            //Make the call to create the customer with the creation options
            var customerService = new StripeCustomerService();
            StripeCustomer stripeCustomer = customerService.Create(customer);

            //save the customer ID
            user.CustomerIdentifier = stripeCustomer.Id;

            //create card update options and add billing info
            var cardOptions = new StripeCardUpdateOptions();
            cardOptions.AddressLine1 = model.BillingInfo.AddressL1;
            cardOptions.AddressLine2 = model.BillingInfo.AddressL2;
            cardOptions.AddressCountry = model.BillingInfo.Country;
            cardOptions.AddressCity = model.BillingInfo.City;
            cardOptions.AddressState = model.BillingInfo.State;
            cardOptions.AddressZip = model.BillingInfo.Zip;
            cardOptions.Name = model.BillingInfo.Name;
            var cardUpdate = new StripeCardService();
            // get the customer card ID and then update the card info
            StripeCustomer customerCardGet = customerService.Get(user.CustomerIdentifier);
            var cardId = customerCardGet.DefaultSourceId;
            StripeCard Card = cardUpdate.Update(user.CustomerIdentifier, cardId, cardOptions);


            //save Subscriptions data here
            //user.ConcurrentUsers = Stripe Quantity

            ModelState.Clear();

            await _userManager.UpdateAsync(user);

        }
        ViewBag.Success = "confirm";
        return View("Success");
    }
```

This particular example includes some of the extras like taking billing information in the controller. If you want this just add it to the form on the view and create a model to hold it.

Read Stripe.Net Introduction online: https://riptutorial.com/stripe-payments/topic/4328/stripe-net-introduction

Credits

S. No	Chapters	Contributors
1	Getting started with stripe-payments	andpei, Community, Yumiko, Ywain
2	PHP-Stripe Connect documentation for Symfony2	Mohammad Fareed
3	Stripe Add multiple card to Same User	ashish bansal
4	Stripe.Net Introduction	Alex Rohr