Laravel 4

Türkçe Dokümantasyon

(Versiyon 4.2)

Hazırlayan ve E-kitap'a dönüştüren

Sinan Eldem

Laravel 4 Türkçe Dokümantasyon (v. 4.2) (Ücretsiz)

Laravel 4 Türkiye Forumları Çeviri Ekibi tarafından yapılan çeviriler

Sinan Eldem

Bu kitap http://leanpub.com/laravel42-tr adresinde satıştadır.

Bu versiyon, 2015-08-29 tarihinde yayınlanmıştır

İçindekiler

Artisan CLI

Giriş

Artisan, Laravel içerisinde gelen CLI'ın (Command-line Interface) adıdır. Artisan size uygulamanızı geliştirirken birçok yardımcı komut sağlar. Artisan, güçlü Symfony Console bileşeni üzerinden geliştirilmiştir.

Kullanım

Tüm Kullanılabilir Komutların Listelenmesi

Tüm Artisan komutlarının bir listesini görmek için `list` komutunu kullanabilirsiniz:

```
1  php artisan list
```

Bir Komut için Yardım Ekranının Görüntülenmesi

Tüm komutların özel bir "yardım" ekranı vardır ve komut hakkındaki argüman sırası ile ayarlar gibi bilgilerin açıklanmasını sağlar. Bir yardım ekranını görüntülemek için komut adından önce `help` yazın:

```
1  php artisan help migrate
```

Yapılandırma Ortamının Belirtilmesi

`--env` anahtarını kullanarak bir komut çalıştırılırken kullanılacak olan yapılandırma ortamını belirtebilirsiniz:

```
php artisan migrate --env=local
```

Güncel Laravel Sürümünüzün Gösterilmesi

Ayrıca Laravel yüklemenizin güncel sürümünü de `--version` seçeneğini kullanarak görebilirsiniz:

```
php artisan --version
```

Laravel Cashier

Giriş

Laravel Cashier Stripe'in[1] abonelik faturalama hizmetleri için anlamlı, akıcı bir arayüz sağlar. Sizin yazmaktan ürktüğünüz klişe abonelik faturalama kodunun hemen tümünü halleder. Cashier, temel abonelik yönetimine ek olarak kuponları, abonelik takasını, abonelik "miktarlarını", ödemesiz dönemlerin iptal edilmesini halledebilir ve hatta fatura PDF'leri üretebilir.

Yapılandırma

Composer

Öncelikle, `composer.json` dosyanıza Cashier paketini ekleyin:

```
1  "laravel/cashier": "~2.0"
```

Service Provider

Daha sonra, `app` yapılandırma dosyanızda `Laravel\Cashier\CashierServiceProvic` kayda geçirin.

Migration

Cashier kullanabilmemiz için, veritabanımıza birkaç sütun eklememiz gerekiyor. Endişe etmeyin, gerekli sütunları ekleyecek bir

[1] https://stripe.com

migrasyon oluşturmak için `cashier:table` Artisan komutunu kullanabilirsiniz. Örneğin, bu alanı users tablosuna eklemek için `php artisan cashier:table users` kullanın. Bu migrasyonu oluşturduktan sonra basitçe `migrate` komutunu çalıştırın.

Model Ayarı

Ondan sonra da model tanımlamanıza BillableTrait ve uygun tarih değiştiricilerini ekleyin:

```
use Laravel\Cashier\BillableTrait;
use Laravel\Cashier\BillableInterface;

class User extends Eloquent implements BillableInterfac\
e {

        use BillableTrait;

        protected $dates = ['trial_ends_at', 'subscription_end\
s_at'];

}
```

Stripe Key

Son olarak, bootstrap dosyalarınızın birinde Stripe anahtarınızı ayarlayın:

```
User::setStripeKey('stripe-key');
```

Bir Plana Abone Olunması

Bir model olgusuna sahip olduktan sonra o kullanıcıyı verilen bir Stripe planına kolaylıkla abone edebilirsiniz:

```
$user = User::find(1);

$user->subscription('monthly')->create($creditCardToken\
);
```

Bir abonelik oluştururken bir kupon uygulamak isterseniz, `withCoupon` metodunu kullanabilirsiniz:

```
$user->subscription('monthly')
     ->withCoupon('code')
     ->create($creditCardToken);
```

Bu `subscription` metodu ilgili Stripe aboneliğini otomatik olarak oluşturacaktır, bunun yanında veritabanınızı Stripe müşteri ID'si ve ilgili diğer faturalama bilgisiyle güncelleyecektir. Eğer planınızda Stripe'de yapılandırılmış olan bir trial (deneme) varsa, kullanıcı kaydında deneme bitiş tarihi (trial end date) de otomatik olarak ayarlanacaktır.

Eğer planınız Stripe'de yapılandırılmış **olmayan** bir deneme süresine sahipse, deneme bitiş tarihini abonelikten sonra elle ayarlamak zorundasınız:

```
$user->trial_ends_at = Carbon::now()->addDays(14);

$user->save();
```

Ek Kullanıcı Ayrıntılarının Belirtilmesi

Ek müşteri ayrıntılarını geçmek isterseniz, onları `create` metoduna ikinci parametre olarak geçmek suretiyle bunu yapabilirsiniz:

```
$user->subscription('monthly')->create($creditCardToken\
, [
        'email' => $email, 'description' => 'Our First Custome\
r'
]);
```

Stripe tarafından desteklenen ek alanlar hakkında daha fazlasını öğrenmek için, Stripe'ın müşteri oluşturma dokümantasyonuna[2] bakınız.

Kredi Kartsız

Eğer uygulamanız kredi kartı olmaksızın bedava bir deneme teklif ediyorsa, modelinizde `cardUpFront` özelliğini `false` olarak ayarlayın:

```
protected $cardUpFront = false;
```

Hesap oluşturulmasında, modelde deneme bitiş tarihi ayarladığınızdan emin olun:

```
$user->trial_ends_at = Carbon::now()->addDays(14);

$user->save();
```

Aboneliklerin Takas Edilmesi

Bir kullanıcıyı yeni bir aboneliğe takas etmek için, `swap` metodunu kullanın:

[2] https://stripe.com/docs/api#create_customer

```
$user->subscription('premium')->swap();
```

Eğer kullanıcı deneme (trial) durumundaysa, deneme normal şekilde sürdürülecektir. Ayrıca abonelik için eğer bir "miktar (quantity)" mevcutsa, miktar da sürdürülecektir.

Abonelik Miktarı

Bazen abonelikler "miktar" ile etkilenir. Örneğin, uygulamanız bir hesap üzerinde kullanıcı başına ayda $10 ücretlendirme yapabilir. Abonelik miktarını kolayca artırmak ve azaltmak için `increment` ve `decrement` metodlarını kullanın:

```
$user = User::find(1);

$user->subscription()->increment();

// Aboneliğin mevcut miktarına beş ekle...
$user->subscription()->increment(5);

$user->subscription()->decrement();

// Aboneliğin mevcut miktarından beş çıkar...
$user->subscription()->decrement(5);
```

Bir Aboneliğin İptal Edilmesi

Bir aboneliğin iptal edilmesi parkta bir yürüyüştür:

```
$user->subscription()->cancel();
```

Bir abonelik iptal edildiği zaman, Cashier veritabanınızdaki `subscription_ends_at` sütununu otomatik olarak ayarlayacaktır. Bu sütun, `subscribed` metodunun ne zaman `false` döndürmeye başlaması gerektiğini bilmek için kullanılır. Örneğin, eğer bir müşteri 1 Martta bir aboneliği iptal ederse ama aboneliğin sona ermesi 5 Marta kadar planlanmamışsa, `subscribed` metodu 5 Marta kadar `true` döndürmeye devam edecektir.

Bir Aboneliğe Geri Dönülmesi

Eğer bir kullanıcı aboneliğini iptal etmiş ve bu aboneliğe kaldığı yerden devam etmesini istiyorsanız, `resume` metodunu kullanın:

```
$user->subscription('monthly')->resume($creditCardToken\
);
```

Eğer kullanıcı bir aboneliği iptal eder ve daha sonra bu abonelik tam olarak sona ermeden geri dönerse, onlara hemen fatura edilmeyecektir. Abonelikleri sadece tekrar etkinleştirilecektir ve orijinal faturalama döngüsüne göre fatura edilecektir.

Abonelik Durumunun Yoklanması

Bir kullanıcının uygulamanıza abone olduğunu doğrulamak için, `subscribed` komutunu kullanın:

```
if ($user->subscribed())
{
        //
}
```

Bu `subscribed` metodu bir rota filtresi için harika bir adaydır:

```
Route::filter('subscribed', function()
{
        if (Auth::user() && ! Auth::user()->subscribed())
        {
                return Redirect::to('billing');
        }
});
```

Ayrıca, `onTrial` metodunu kullanmak suretiyle kullanıcının hala deneme süresinde olup olmadığını (uygunsa) da tayin edebilirsiniz:

```
if ($user->onTrial())
{
        //
}
```

Kullanıcının daha önce aktif bir abone olduğunu ama aboneliğini iptal etmiş olduğunu tayin etmek için `cancelled` metodunu kullanabilirsiniz:

```
if ($user->cancelled())
{
        //
}
```

Ayrıca, bir kullanıcının aboneliğini iptal etmiş ama hala aboneliği tam sona erinceye kadar "yetkisiz kullanım süresinde (grace period)" olup olmadıklarını da belirleyebilirsiniz. Örneğin, bir kullanıcı 10 Martta sona ereceği planlanmış bir aboneliği 5 Martta iptal ederse, bu kullanıcı 10 Marta kadar "yetkisiz kullanım süresindedir". `Subscribed` metodunun bu zaman süresinde hala `true` döndürdüğüne dikkat ediniz.

```
if ($user->onGracePeriod())
{
        //
}
```

Bir kullanıcının uygulamanızdaki bir plana hiç abone olup olmadığını tayin etmek için `everSubscribed` metodu kullanılabilir:

```
if ($user->everSubscribed())
{
        //
}
```

Bir kullanıcının verilen bir plana abone olup olmadığını ID'sine dayalı olarak tayin etmek için `onPlan` metodu kullanılabilir:

```
if ($user->onPlan('monthly'))
{
        //
}
```

Başarısız Ödemelerin Halledilmesi

Şayet bir müşterinin kredi kartı süresi dolarsa ne olur? Endişeye gerek yok - Cashier sizin için müşterinin üyeliğini kolaylıkla iptal edebileceğiniz bir Webhook controller içermektedir. Sadece bir rotada bu controlleri belirtin:

```
Route::post('stripe/webhook', 'Laravel\Cashier\WebhookC\
ontroller@handleWebhook');
```

Hepsi bu kadar! Gerçekleşmemiş ödemeler bu controller tarafından yakalanacak ve halledilecektir. Bu controller üç başarısız ödeme

girişiminden sonra ilgili müşterinin aboneliğini iptal edecektir. Bu örnekteki `stripe/webhook` URI sadece örnek içindir. Kendi Stripe ayarlarınızda bu URI'ı yapılandırmanız gerekir.

Diğer Stripe Webhook'larının İşlenmesi

İşlemek istediğiniz başka Stripe webhook olaylarına sahipseniz, Webhook controller'i basitçe genişletin. Metod isminiz Cashier'in beklenen geleneğine uygun olmalıdır, burası için özel olarak, metod ismi işlemek istediğiniz Stripe webhook'un ismi ve önüne `handle` getirilmiş hali olmalıdır. Örneğin, eğer `invoice.payment_succeeded` webhook'unu işlemek istiyorsanız controllerinize bir `handleInvoicePaymentSucceeded` metodu eklemelisiniz.

```
1  class WebhookController extends Laravel\Cashier\Webhook\
2  Controller {
3
4          public function handleInvoicePaymentSucceeded($payload)
5          {
6                  // Olayı işle...
7          }
8
9  }
```

> **Not**: Webhook controller veritabanınızdaki abonelik bilgilerini güncellemeye ek olarak Stripe API aracılığıyla aboneliği de iptal edecektir.

Faturalar

`invoices` metodunu kullanarak bir kullanıcının faturalarından oluşan bir diziyi kolaylıkla elde edebilirsiniz:

```
$invoices = $user->invoices();
```

Müşterinin faturalarını listelerken, ilgili fatura bilgisini göstermek için şu helper metodlarını kullanabilirsiniz:

```
{{ $invoice->id }}

{{ $invoice->dateString() }}

{{ $invoice->dollars() }}
```

Bir faturanın indirilebilir bir PDF'sini üretmek için `downloadInvoice` metodunu kullanın. Evet, bu gerçekten bu kadar kolaydır:

```
return $user->downloadInvoice($invoice->id, [
        'vendor'  => 'Şirketiniz',
        'product' => 'Ürününüz',
]);
```