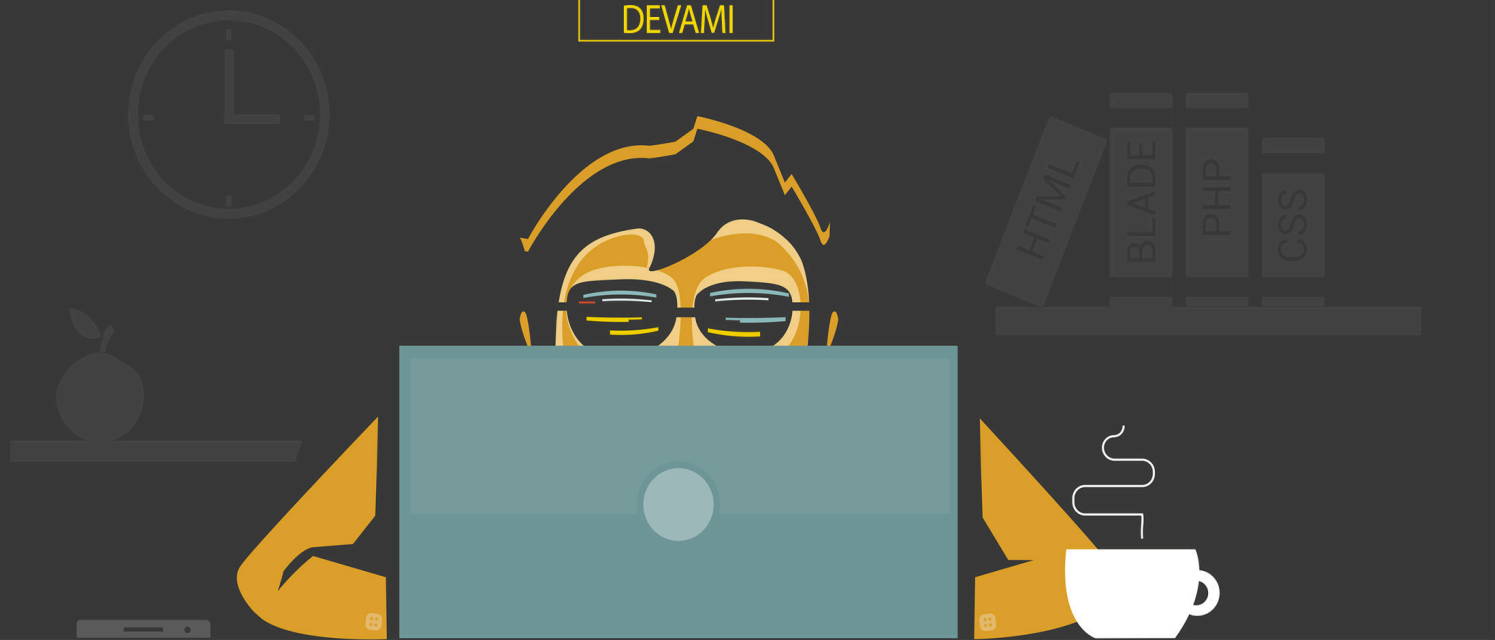
LARAVEL 5.1 GÜZELLİĞİ

Laravel 5.1 ile Harika Web Uygulamaları Geliştirme

DEVAMI

<CHUCK HEINTZELMAN & SİNAN ELDEM>

DEVAMI



Laravel 5.1 Güzelliği (Türkçe)

Laravel 5.1 ile Harika Web Uygulamaları Geliştirme

Chuck Heintzelman ve Sinan Eldem

Bu kitap http://leanpub.com/laravel51beautytr adresinde satıştadır.

Bu versiyon, 2016-01-17 tarihinde yayınlanmıştır

Bu Yazarlardan Ayrıca

Chuck Heintzelman Kitapları

Getting Stuff Done with Laravel 4

Getting Stuff Done with Laravel 4 (TR) Türkçe

Laravel4でこなすプログラム術　Getting Stuff Done

Laravel 5.1 Beauty

Sinan Eldem Kitapları

Laravel: Code Bright (TR) Türkçe

Laravel: From Apprentice To Artisan (TR) Türkçe

Implementing Laravel (TR) Türkçe

Laravel 4 Türkçe Dokümantasyon (v. 4.1) (Ücretsiz)

Laravel 4 Cookbook (TR) Türkçe

Getting Stuff Done with Laravel 4 (TR) Türkçe

Nefret Etmeyeceğiniz Uygulama Programlama Arayüzleri (API) İnşa Edin

PHP: ”Usulüne” Uygun (Ücretsiz)

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

İçindekiler

Teşekkürler

Bu kitabı incelediğiniz için teşekkürler. Umarım eğitici ve yararlı bulursunuz.

Geri Besleme

Görüşleriniz teşvik edilmektedir!

Herhangi bir yazım yanlışı, düzeltme ile karşılaşırsanız veya herhangi bir bölüm hakkında yorum veya eleştiri göndermek isterseniz lütfen LaravelCoding.com[1] ilgili sayfayı açıp bölüm hakkında bilgilendirmede bulunun.

Laravel 5.1 Öğrenmek İçin Diğer Kaynaklar

- Laravel Web Sitesi[2] - Belgeler öğrenmeye başlamak için güzel kaynaktır.
- Laracasts[3] - Jeffrey Way tarafından yayınlanan videolar benzersizdir.
- Laravel.gen.tr[4] - Türkiye'deki Laravel kullanıcıları için oluşturulmuş topluluk platformu.

[1] http://LaravelCoding.com/blog/?tag=L5+Beauty
[2] http://laravel.com
[3] http://laracasts.com
[4] http://laravel.gen.tr

Bölüm 1 - Giriş

Bölüm İçeriği

Uzun Süreli Destek

Laravel sürüm 5.1 ilk LTS (Uzun Süreli Destek) Laravel sürümüdür. Bu, 2 yıl süreli hata düzeltmeleri ve 3 yıl süreli güvenlik düzeltmeleri sağlanması anlamına gelir.

Bugün inşa uygulamaların yarın framework tarafından desteklenecek olması anlamına geldiğinden son derece önemlidir.

Niçin Bu Kitap

Laravel üstüne ilk kitabım *Getting Stuff Done with Laravel 4*[5] iyi ilgi gördü.

Şimdi Laravel 5.1 kullanılabilir durumda olduğundan önceki kitabımı Laravel 5.1'e güncellemeyi düşündüm. Laravel yeni sürümü Laravel 4'ten çok büyük değişiklikler içerir, ama Laravel 5.1 çoğunlukla geriye dönük uyumludur.

Ama *Getting Stuff Done with Laravel 4* kitabı Laravel 4'ü her yönüyle kapsayan bir kitap değil. Uygulama biraz farklı olsa da ele alınan ilkeler hala geçerli.

Önceki kitabımı dönüştürmektense, yeni özelliklerinden bazılarını vurgulamak için yeni bir kitap yazdım, ***Laravel 5.1 Güzelliği***, bu kitap [*Getting Stuff Done with Laravel 4*]'tan çok daha büyük ve iyi.

[5] https://leanpub.com/gsd-laravel-tr

GitHub ve Blog

Laravel 5.1 Beauty kitabını geliştirirken eşzamanlı olarak LaravelCoding.com[6] ve Leanpub[7]'da yayınlıyorum.

Kaynak Kodlar GitHub'ta

Bu kitapta inşa edilen uygulamanın kaynak kodları Github'ta sineld/l5beauty[8] ambarındadır. Hangi bölümü çalışıyorsanız, o bölümün dalından kodları inceleyiniz.

Bu kitabımın öncekinden farklı bir tonu vardır. Komik olmaya gerek gerek yok. *(Sanırım herkes Dayle Rees olamaz.)*

Laravel 5.1 Güzelliği Laravel'i Php frameworkleri arasında en iyi yapan özelliklerine odaklanarak uygulamanın inşası ve tasarımıyla bir gerçek dünya uygulaması inşa eder.

Uygulama Nedir?

Bu kitap boyunca, yönetimini sağlayan paneli de dahil olmak üzere basit, temiz ve güzel bir blog uygulaması inşa edeceğiz.

Benim şahsi bloğum, `LaravelCoding.com` da burada geliştirilen uygulamayı kullanmaktadır.

Kitapta Kullanılan Düzen

Bu kitap boyunca kullanılan bazı düzenler vardır.

Kod iki boşluk girintilidir

Php kodunun standart girintilemesi 4 boşluktur. Bu kitap birçok e-kitap formatında hazırlanacağından ve aygıtların birçoğunun ekranlarının küçük olmasından dolayı, çok yatay boşluk olmayacaktır. Bu kitaptaki kodlar yer kazanımı için 4 yerine 2 boşluk ile yazılmıştır.

```
for ($i = 1; $i <= 10; $i++) {
  echo "Saymayı öğreniyorum $i\n";
}
```

Ters bölü (\) ile biten satırlar devam ettirilmelidir

Eğer ters bölü ile biten satırlarla karşılaşırsanız, bu kodları bölmeden sonraki satırlarla birleştirmelisiniz anlamına gelir.

[6] http://laravelcoding.com
[7] http://leanpub.com
[8] https://github.com/sineld/l5beauty

```
$ Bu_gerçekten_çok_uzun_bir_komut_satırıdır burada_birçok devam_edilmesi\
gereken_argüman_mevcuttur
```

Üstteki satırda her ne kadar iki satır görüntülense de, siz ters bölü haricindeki herşeyi tek satırda yazmalısınız.

Buraya Dikkat Ediniz

Kodlama yaparken burayı gözden kaçırırsanız kodlarınız çalışmayabilir Şüpheniz olduğundan GitHub **l5beauty** ambarına gözatınız.

Windows, OS X (veya Linux), ve Homestead için farklı komut satırları

Windows komutları her kullanıldığında `C:` ile başlar ve `>` ile biter.

```
C:\veri\yolu>
```

OS X konsolu ve Linux konsolu kullanılırken `>` sembolü ile biter ancak veri yolunun gösterilmesi için ters bölü yerine bölü işareti kullanılır. Çoğunlukla veri yolunda (~) tilde işareti vardır.

```
~/veri/yolu>
```

Konsol genelleyici olduğunda (yani Windows, OS X veya Linux konsolu olabilir, sizin işletim sistemi tercihinize bağlı) komut satırı `%` ile bitmektedir.

```
/veri/yolu%
```

Son olarak, komut satırında, Homestead Sanal Makinesi kullanıldığında, standart dolar işareti `$` kullanılmaktadır. *(Bu kitabın büyük bölümünde Homestead Sanal Makinesi kullanılmıştır.)*

```
~/veri/yolu$
```

Homestead Sanal Makinesi ile, komut satırınız genelde veri yolundan önce bilgisayar ve kullanıcı adını birlikte gösterir. Örneğin: `vagrant@homestead:~$`, ancak kullanıcı adı ve bilgisayar adı nadiren gösterilmektedir.

Bazen veri yolu kayıptır

Windows komut satırında veri yolu atlanmışsa projenin ana dizininde bulunduğunuz varsayılmıştır.

İyi Eğlenceler

Umarım bu kitabı beğenir ve aracılığı ile Laravel 5.1 öğrenirsiniz. Anlatılanları adım adım takip edip, geliştirme ortamınızı kurduktan sonra bölüm bölüm ilerleyiniz.

Her şeyden önce iyi eğlenceler dilerim. Laravel 5.1 ile kodlama çok eğlencelidir.

Çevirenin Notu

Çalışmalarımın destekçisi, bu kitap ve bundan önceki kitaplarımda her zaman yanımda olan sevgili Eşim Bilge'ye, gözümün ışığı kızım Tuana Şeyma'ya teşekkürler.

Ayrıca bu çalışmanın destekçisi siz değerli okuyucuma teşekkür ederim.

Bölüm 2 - Gerekli Yazılım ve Bileşenler

Bu bölümde Laravel 5.1 uygulamaları geliştirirken hangi yazılımları ve bileşenleri kullanmamız gerektiğini tartışacağız. VirtualBox ve Vagrant yüklemek için talimatlar verilmektedir.

Bölüm İçerikleri

Sanal Makinelerin Yükselişi

Son birkaç yıldır, sanal makineler, kendi yerine gelmiş. Sanal Makineler (veya VM'ler) bilgisayar sisteminin (ana işletim sisteminin) farklı bir işletim sistemini taklit etmesini sağlar. Elbette, VM'ler yalnızca bir süredir etrafımızdalar ancak artan işlemci hızları ve ucuz bellekler sayesinde VM'ler her geliştiricinin masaüstünde yerini alabilir.

Laravel VM teknolojisini kucaklar ve kendi özel *"box"* 'ını (kutusunu) web uygulamaları geliştirme için gerekli uygulamalar ile dolu olarak sunar. Önpaketli geliştirme ortamının ismi **Laravel Homestead**[9] 'dir.

Laravel Homestead Hakkında

Laravel ile geliştirmenin ardındaki en temel felsefe PHP geliştirmenin hem kolay hem de eğlenceli olmasıdır. Bunu sağlamak içinse Laravel size Laravel Homestead adında geliştirme ortamını sağlar. Vagrant[10] sanal makinenin yönetiminde kullanılmaktadır. Kaputun altında VirtualBox[11] ana işletim sistemi arayüzü sağlar.

Bir **araba** bütün bu şeylerin bir arada çalışması için güzel bir metafordur. **Homestead** sürücünün araç koltuğudur, **Vagrant** aracın omurgası, ve **VirtualBox** motorudur. Vagrant ve VirtualBox

[9] http://laravel.com/docs/5.1/homestead
[10] http://vagrantup.com
[11] http://virtualbox.org

yüklendiğinde, bunlar hakkında tekrar endişeye gerek yoktur. VM ile tüm etkileşim Homestead aracılığı ile olur. *(Aynen sürülen araç gibi, omurga ve motor hakkında endişeye gerek yoktur.)*

Laravel Homestead bir sanal Ubuntu Linux kullanmanıza olanak sağlar. Web uygulaması geliştirmek için gerekli uygulamalar öntanımlı olarak yüklenmiştir. Bu VM şunları içerir:

- Ubuntu 14.04
- PHP 5.6
- HHVM
- Nginx
- MySQL
- PostgresSQL
- Node (Bower, Grunt, ve Gulp ile birlikte)
- Redis
- Memcached
- Beanstalkd
- Laravel Envoy
- Fabric + HipChat Uzantıları

En güzel tarafı ise Laravel Homestead, Windows, OS X ve Linux işletim sistemlerinin ana makinayla çakışma sorunu olmaksızın aynı ortamı kullanmasını sağlar.

Virtual Box Yüklenmesi

Vagrant onu yönetecek sanal makine sağlamak için bir arka uç sağlayıcısı gerektirir. Eğer zaten VirtualBox, VMWare, veya farklı uyumlu bir sağlayıcı[12]'ya sahipseniz bu adımı atlayabilirsiniz.

Ancak kurulu bir arka uca sahip değilseniz, VirtualBox platform paketini kullanın. Ücretsizdir ve tüm mayor platformlarda çalışır.

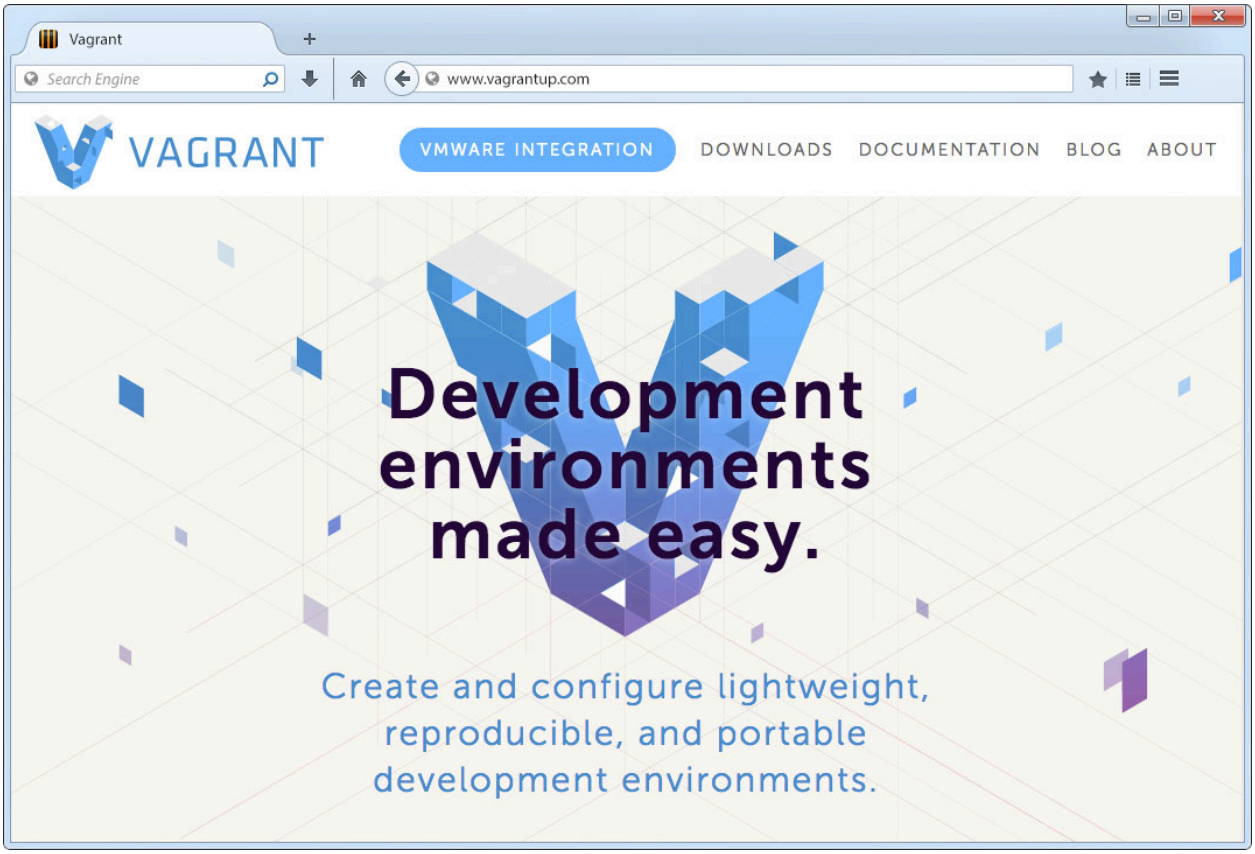
[12] https://docs.vagrantup.com/v2/getting-started/providers.html

Virtualbox Yükleme Sayfası

‘www.virtualbox.org](https://www.virtualbox.org/wiki/Downloads) adresine gidin, işletim sisteminize uygun paketi indirin ve yükleyin.

Vagrant Yüklenmesi

VirtualBox (veya farklı bir arka uç sağlayıcı) yüklediyseniz, Vagrant yüklemeniz gerekmektedir.



Vagrant Ana Sayfası

`www.vagrantup.com` adresine gidin, işletim sisteminize uygun paketi indirin ve yükleyin.

Vagrant yüklemesi tamamlandığında bilgisayarınızı yeniden başlatmalısınız. Yeniden başlatmanın ardından Vagrant'ın kurulduğundan emin olmak için konsolu (Windows'ta komut istemi, OS X veya Linux'ta terminal) çalıştırın ve sürüm denetimi yapın.

Vagrant Sürümü Denetleniyor

```
% vagrant --version
Vagrant 1.6.5
```

Vagrant Windows Yükleme Konumu

Windows'a yüklenen diğer yazılımlardan farklı olarak, Vagrant Windows Başlangıç Menüsünde bulunmaz. Kendini `C:\HashiCorp` dizinine yükler ve `C:\HashiCorp\Vagrant\bin` satırını Windows sistem path'ine (veriyolu) ilave eder.

Şeyleri Nerede Çalıştırırım?

Önümüzdeki bölümler boyunca Laravel Homestead'in kurulum ve çalıştırılmasında en genel soru *"Şu ... nerede çalıştıracağım?"* veya *"Şu ... nerede çalışır?"* olacaktır. Bu bölüm Homestead içinde Laravel ile geliştirmenin önemli bileşenlerinden kısa bir özet sunar ve sorulara cevap verir.

Web Sunucusu

Web Sunucusu Homestead Sanal Makinesi içinde çalışır.

Nginx web sayfalarını sunan web sunucusudur. Ana bilgisayar işletim sistemi web sayfalarına standart HTTP portu (80) ile 192.168.10.10 adresinden erişebilir. Ana bilgisayar ayrıca web sayfalarına 127.0.0.1 üzerinden 8000 portu ile de erişebilir.

Dosyaların Düzenlenmesi

Dosyalarınızı her zaman ana bilgisayar üzerinde düzlenleyin.

Düzenlenen sayfalar, paylaşım klasörleri aracılığı ile Homestead VM'de anında erişebilir olur.

MySQL

MySQL Homestead Sanal Makinesi içinde çalışır.

MySQL'e ana bilgisayarınızdan aşağıdaki bilgiler ile erişebilirsiniz.

Ayar Adı	Ayar Değeri
Host	127.0.0.1
Port	33060
Username	homestead
Password	secret

Memcached

Memcached bellek içi anahtar/değer önbellekleme sistemir. Homestead Sanal Makinesi içinde çalışır.

Beanstalkd

Beanstalkd basit ve hızlı bir iş kuyruğudur. Homestead Sanal Makinesi içinde çalışır.

Git or Subversion

Ana bilgisayarınızda çalıştırınız.

Her ne kadar versiyon denetleme sistemlerini her iki yerden de çalıştırabiliyor olsanız da, **şiddetle** önerilen bunu ana bilgisayarda yapmanızdır. Sürekli olarak bunu tek yerden çalıştırmak muhtemel çakışmaları önler.

Varsayalım ki, Homestead Sanal Makinesi'ne subversion yüklediniz ve sürümü 1.8. Homestead Sanal Makinesi içindeki kodu incelemeye çalışın ve ana bilgisayarda denetlemeye çalışın. Eğer ana bilgisayardaki yüklü sürüm 1.7 ise subversion sürümünü yükseltmediğiniz sürece çalışmayacaktır.

Bower

Bower web için basit bir paket yöneticidir. Ana bilgisayarınızda yüklü ise her iki yerden de çalıştırabilirsiniz.

Gulp

Gulp, Laravel Elixir'in kullandığı basit bir inşa sistemidir. Gulp ile assetlerinizi birleştirebilir, sıkıştırabilir, kopyalayabilir ve ünite testlerinizi otomatikleştirebilirsiniz.

YALNIZCA ana bilgisayarınızdan çalıştırmalısınız.

Gulp'ı ana bilgisayarınızdan çalıştırdığınızda, belirli görevler yapıldığında (LESS dosyalarının derlenmesi) growl benzeri bildirimlerler görüntülenecektir. Gulp'ı Homestead Sanal Makinesi'nde çalıştırırsanız bu bildirimlerin gösterilmesi esnasında hatalar ile karşılaşacaksınız.

Composer

Yalnızca ana bilgisayarınızdan çalıştırmalısınız.

Eğer ana bilgisayarınızın işletim sistemi OS X veya Linux ise her iki yerden de çalıştırabilirsiniz ancak Windows ise Composer'ın doğru çalışabilmesi için gerekli toplu işlem dosyalarını oluşturması gerekecektir.

Artisan

Yalnızca Homestead Sanal Makinesi içinde çalıştırmalısınız. Bunun temel sebebi veritabanı, kuyruk ve önbellekleme sürücüleri Homestead içinde yüklüdür ve ana bilgisayarınızda bunlara erişim olmayacaktır. Ayrıca `localhost` için veritabanı ayarları Homestead VM için özelleştirilmiştir ve ana bilgisayarınızda tanımlı olmayacaktır.

Konsolda komut çalıştırma kuralları

Kural şudur: *Yalnızca artisan komutlarını Homestead VM içinde çalıştırın.* Bunun dışında her komut ana bilgisayarınızın işletim sisteminde çalıştırılmalıdır.

Tekrar

Bu bölümde Laravel 5.1 ile uygulama geliştirme için gerekli birtakım yazılımlar hakkında tartıştık. Virtualbox ve Vagrant yüklemesi yaptık.

Eğer ana bilgisayarınızın işletim sistemi Windows ise sonraki bölüm olan *Windows Makine Kurulumu*'na geçiniz. Aksi takdirde sonraki bölümü atlayıp *OS X veya Linux Makine Kurulumu* bölümüne geçiniz.

Bölüm 3 - Windows Makine Kurulumu

Bu bölümde Windows makinesine Laravel Homestead için gerekli uygulamaların yüklenmesi ve kurulması için gereken adımları sırasıyla gerçekleştireceğiz. **VirtualBox** ve **Vagrant**'ın önceki bölümde anlatıldığı gibi yüklenmiş olduğu varsayılmıştır.

Ana bilgisayarınızın işletim sistemi OS X veya Linux ise sonraki bölüme geçiniz.

Bölüm İçerikleri

Windows Kurulumunun Birçok Yöntemi

Windows ile gerekli uygulamaların yüklemesi için birçok yöntem bulunmaktadır. Aşağıdaki yöntemleri size sunabilmek için birçok senaryo denedim. Bu bölüm Windows 8.1 ile oluşturuldu ancak Windows 7 ve Windows 10 ile de sorunsuz çalışabilecektir.

Aşama 1 - PHP'nin Doğal Olarak Yüklenmesi

İlk aşama PHP'yi Windows'ta çalışır duruma getirmektir.

Aşama 1.1 - PHP'yi İndir / Zip'ten çıkar

windows.php.net/download[13] adresine gidin ve en son Zip dosyasını bilgisayarınıza indirin. Ben makinem için **VC11 x64 Thread Safe** sürümünü indirdim. (Bu yazının yazıldığı zamanki en son sürüm `php-5.6.10-Win32-VC11-x64.zip`)

Bu dosyayı `C:\Php` dizini altına Zip'ten çıkarın.

Aşama 1.2 - PHP.INI güncelleme

Komut İstemini açın ve `php.ini` dosyasını oluşturun.

php.ini-development dosyasını php.ini ye kopyalayın

```
C:\Kullanıcılar\Sineld> cd \php
C:\Php> copy php.ini-development php.ini
```

Ardından `php.ini` dosyasını metin editörü ile açıp aşağıdaki satırları değiştirin.

php.ini içindeki değişiklikler

```
// eski değer
; extension_dir = "ext"
// yeni değer
extension_dir = "ext"

// eski değer
;extension =php_openssl.dll
// yeni değer
extension =php_openssl.dll

// eski değer
;extension =php_mbstring.dll
// yeni değer
extension =php_mbstring.dll
```

Artık, `C:\Php` dizini içinde, php çalıştırabiliyor olmanız gerekir.

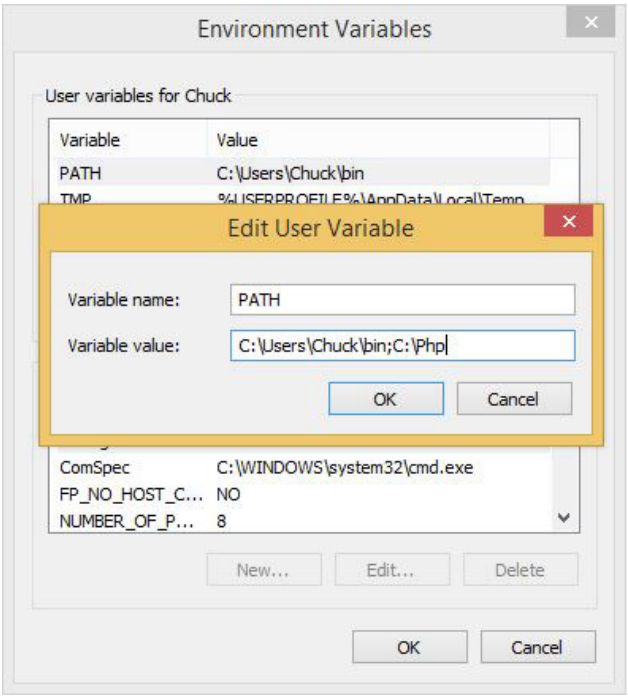
[13] http://windows.php.net/download

PHP sürümünün denetlenmesi

```
C:\Php> php --version
PHP 5.6.10 (cli) (built: Oct 30 2014 16:05:53)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.6.0, Copyright (c) 1998-2014 Zend Technologies
```

Aşama 1.3 - Veriyolu'na C:\Php eklenmesi

- *Windows Denetim Masası*'nı açın
- Sağ üst köşedeki arama kısmından *env* değerini arayınız
- **Hesabınız için ortam değişkenlerini düzenleyin** bağlantısına tıklayın
- Eğer `PATH` zaten Kullanıcı değişkeni ise, **[Düzenle...]**'yi tıklayın, sona `;C:\Php` şeklinde düzenleyin, değilse ekleme yapın.



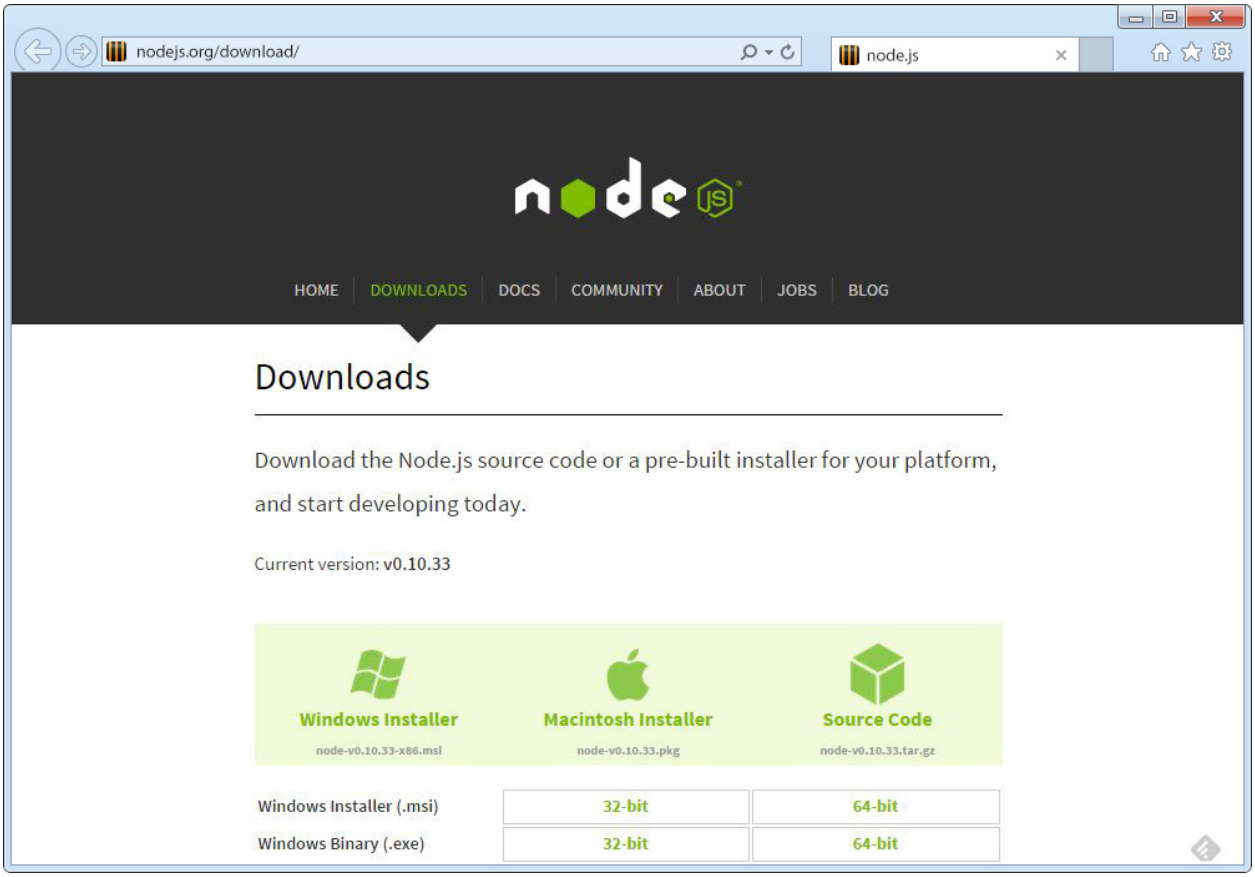
Windows Veriyoluna PHP Eklenmesi

Sonraki Komut istemini çalıştırmanızda `php` komutuna erişebiliyor olacaksınız.

Aşama 2 - Node.js'nin Yüklenmesi

Node.js'yi Windows'a doğal olarak yükleyeceğiz çünkü bu Gulp'ın Windows Komut istemi aracılığı ile doğrudan erişebilir olmasını sağlayacaktır.

nodejs.org/download[14] adresine gidin ve windows sürümünüze uygun dosyayı indiriniz. *(32-bit veya 64-bit.)*



Node.js İndirme Sayfası

Varsayılanlar doğrultusunda yüklemeyi yapınız. Yükleme tamamlanınca *yeni* bir komut istemi penceresi açıp versiyonu sorgulayarak kurulumu denetleyiniz.

node ve npm sürümleri denetimi

```
C:\Kullanıcılar\Sineld> node --version
v0.10.33

C:\Kullanıcılar\Sineld> npm --version
1.4.28
```

[14] http://nodejs.org/download

Gulp'ın evrensel yüklenmesi

```
C:\Kullanıcılar\Sineld> npm install -g gulp
C:\Kullanıcılar\Sineld\AppData\Roaming\npm\gulp -> C:\Kullanıcılar\Sineld\AppDat\
a/
Roaming\npm\node_modules\gulp\bin\gulp.js
gulp@3.8.10 C:\Kullanıcılar\Sineld\AppData\Roaming\npm\node_modules\gulp
[snip]
```

Gulp sürümünü denetlenmesi

```
C:\Kullanıcılar\Sineld> gulp --version
[10:13:44] CLI version 3.8.10
```

İsteğe bağlı Bower yüklenmesi

Windows komut isteminden Bower çalıştırmak için isteğe bağlı olarak evrensel olarak yükleyebilirsiniz. Kişisel olarak ben Homestead Sanal Makinesi içindeki bower kurulumunu kullanırım ama tercih sizin.

Node paket yönetimini (NPM), kullanarak bower'ın evrensel yüklenmesi.

Bower'ın evrensel yüklenmesi

```
C:\Kullanıcılar\Sineld> npm install -g bower
C:\Kullanıcılar\Sineld\AppData\Roaming\npm\bower -> C:\Kullanıcılar\Sineld\AppDa\
ta\
    Roaming\npm\node_modules\bower\bin\bower
bower@1.3.12 C:\Kullanıcılar\Sineld\AppData\Roaming\npm\node_modules\bower
[snip]
```

Bower sürümünün denetlenmesi

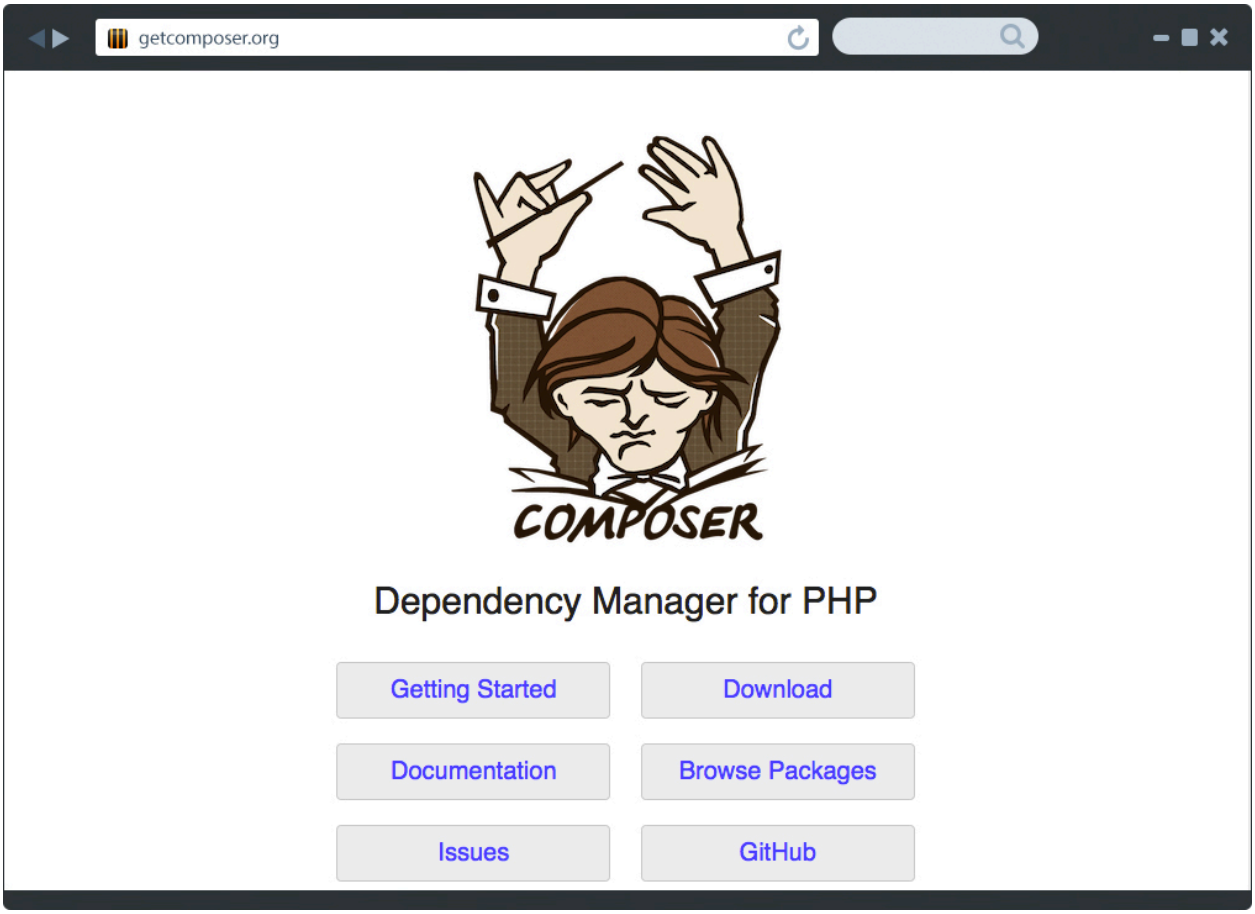
```
C:\Kullanıcılar\Sineld> bower --version
1.3.12
```

Unutmayın, bu sadece programları evrensel yükler

Eğer gulp (veya bower'ı) belirli bir proje içinde kullanacaksanız proje dizini içinde `npm install` komutu ile yükleme yapmanız gerektiğini unutmayın. (-g seçeneğini atlayarak). Buna daha sonra değinilecektir.

Aşama 3 - Composer Yüklenmesi

Composer PHP için paket yöneticidir.



Composer Web Sayfası

Windows kurulum programını indirin ve yükleyin, Composer-Setup.exe[15]. Kurulumda varsayılanları kullanın ve PHP veriyolu sorulduğunda `C:\Php\php.exe` girin.

Composer yüklendiği zaman, komut pencerelerini kapatın ve yenisini açın. Doğru yüklendiğinden emin olmak için composer sürümünü denetleyin.

Composer sürümünün denetlenmesi

```
C:\Kullanıcılar\Sineld> composer --version
Composer version 1.0-dev (b23a3cd36870ff0eefc161a4638d9fcf49d998ba)\
2014-11-21 17:59:11
```

[15] https://getcomposer.org/Composer-Setup.exe

Composer yüklenmesi veriyolunu günceller

Kurulum kişisel PATH veriyolundan `C:\Php` çıkaracak ve sistem PATH'ine `C:\ProgramData\ComposerSetup\bin` kaydını ekleyecektir.

Aşama 4 - GIT Yüklenmesi ve SSH Anahtarının Kurulması

Windows için doğal GIT uygulamasını yükleyeceğiz ve GIT BASH ile SSH anahtarımızı oluşturacağız. Herhangi bir GIT kullanımını Windows komut istemi ile yapacağız.

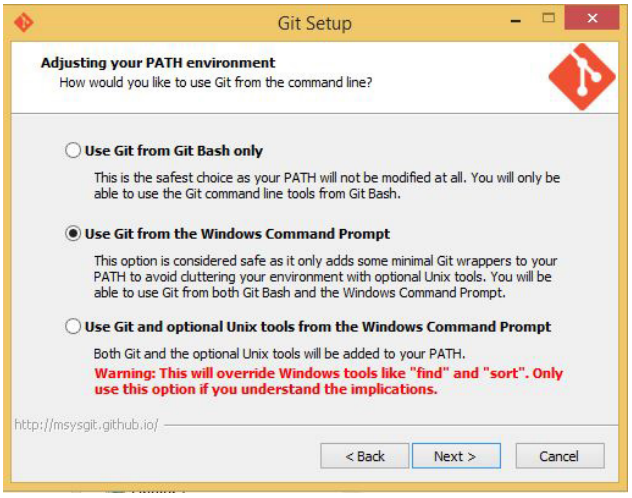
Aşama 4.1 - Git Kurulumunun İndirilmesi

git-scm.com/downloads[16] adresine gidin ve **[Download for Windows]** butonuna basın. Bu windows için Git'in son sürümünü indirecektir.

(Bu yazının yazıldığı zaman, indirilen dosyasın adı `Git-1.9.4-preview20140920.exe`'di.)

Aşama 4.2 - 'Use Git from Command Prompt' seçeneği ile kurulumu yapın

Aşağıdaki ekranı görünceye kadar indirdiğiniz dosyasının kurulumunu varsayılanlar ile yapmaya devam edin.



Git Veriyolu Seçeneği

Use Git from the Windows Command Prompt seçeneğini seçtiğinizden emin olun.

Kurulumunun kalanı için varsayılanlar ile devam edin.

[16] http://git-scm.com/downloads

Aşama 4.3 - Git Sürümünün Denetlenmesi

Git yüklendiği zaman, komut pencerelerini kapatın ve yenisini açın. Doğru yüklendiğinden emin olmak için git sürümünü denetleyin.

Git Sürümünün Denetlenmesi

```
C:\Kullanıcılar\Sineld> git --version
git version 1.9.4.msysgit.2
```

Aşama 4.4 - SSH Anahtarının Kurulması

Windows Başlangıç Menüsü'nden **Git Bash** uygulamasını bulun ve `ssh-keygen` komutunu çalıştırın. Tüm onay pencereleri boyunca [Enter] tuşuna basarak varsayılanları kullanın, bu sayede şifresiz bir SSH anahtarı oluşturmuş olacaksınız.

Git Bash içinde SSH Anahtarı Oluşturulması

```
Sineld@Windows ~
$ ssh-keygen -t rsa -C "email@adresiniz.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Sineld/.ssh/id_rsa):
Created directory '/c/Users/Sineld/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
$
```

Aşama 5 - Homestead Kutusunun Eklenmesi

Bu adımda Laravel Homestead Vagrant kutusu indirilecektir.

Windows'ta Homestead kutusunun eklenmesi

```
C:\Kullanıcılar\Sineld> vagrant box add laravel/homestead
==> box: Loading metadata for box 'laravel/homestead'
    box: URL: https://vagrantcloud.com/laravel/homestead

[snip]
```

Bu işlem yavaş bağlantıda biraz zaman alacaktır.

Aşama 6. Homestead Yüklenmesi

Şimdi `homestead` komutunun yüklenmesi için composer kullanacağız. Bu komut satısı aracı Homestead VM'in kolay denetimini sağlar.

Aşama 6.1 - Homestead'in Evrensel Yüklenmesi

Homestead 2.0'ın Evrensel Yüklenmesi

```
C:\Kullanıcılar\Sineld> composer global require "laravel/homestead"
Changed current directory to C:\Users/Sineld/AppData/Roaming/Composer
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing symfony/process (v2.5.7)
    Loading from cache

  - Installing symfony/console (v2.5.7)
    Loading from cache

  - Installing laravel/homestead (v2.0.7)
    Loading from cache

Writing lock file
Generating autoload files
```

Aşama 6.2 - Veriyolu'nun Güncellenmesi

Composer Homestead'i Composer yüklenmesinin `vendor` dizinine ekledi. (Örneğin, `C:\Kullanıcılar\Sineld\AppD` altına).

Homestead'e herhangi bir komut istemi ekranında erişebilmek için bu veriyolunu PATH değişkenine ilave edin.

Geçmişte **Aşama 1.3 - Veriyolu'na C:\Php eklenmesi** kısmında yaptığınız adımları takip ediniz, yalnızca bu sefer aşağıdaki adresi ekleyeceksiniz. (Buradaki **Sineld** kısmını kendi kullanıcı adınıza göre düzenlemeyi unutmayınız.)

Eklenecek Veriyolu

```
C:\Kullanıcılar\Sineld\AppData\Roaming\Composer\vendor\bin;vendor\bin
```

Sondaki ilave ‘vendor\bin’ dikkatinizi çekti mi?

Bunu eklediğimiz için Laravel proje dizini içinde herhangi bir vendor aracına kolaylıkla ulaşabilirisniz. Örneğin, **phpunit** her Laravel uygulamasında `vendor/bin` dizini içinde bulunur.

Aşama 6.3 - Homestead Yüklenmesini Doğrulamak

Tüm değişikliklerin etkin olması için, komut pencerelerini kapatın ve yeni birini açın. Doğru yüklendiğinden emin olmak için homestead sürümünü denetleyin.

Homestead Sürümünün Denetlenmesi

```
C:\Kullanıcılar\Sineld>homestead --version
Laravel Homestead version 2.0.7
```

Aşama 6.4 - Homestead’in Başlatılması

Homestead komutunun kurulumunu yaptıysanız ve composer bin dizinini veriyolunuza eklediyseniz Homestead’i başlatmalısınız.

Homestead’i Başlatmak

```
C:\Kullanıcılar\Sineld> homestead init
Creating Homestead.yaml file...
Homestead.yaml file created at: C:\Kullanıcılar\Sineld\.homestead/Homestead.yaml
```

Unutmayın

Homestead’i makinenizde yalnızca bir defa başlatmalısınız.

Aşama 7 - Homestead VM’i Ayağa Kaldırmak

`homestead up` komutunu kullanarak Homestead’i ayağa kaldırmak ve projelerinizi saklamak için öncelikle `Code` dizinini oluşturmalısınız.

Homestead'i ilk defa çalıştırmak

```
C:\Kullanıcılar\Sineld> mkdir Code
C:\Kullanıcılar\Sineld> homestead up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'laravel/homestead'...
==> default: Matching MAC address for NAT networking...
==> default Checking if box 'laravel/homestead' is up to date...

[snip]
```

Şu anda Homestead Sanal Makineniz çalışıyor. Eğer Windows Komut istemini kapatsanız bile VM çalışmasını sürdürecektir. Siz `homestead halt` komutunu kullanıncaya kadar çalışmasını sürdürücektir.

Homestead Sanal Makinenize giriş yapabilirsiniz, ancak Windows `homestead ssh` komutunu kullanamayacaktır, bunun için **PuTTY** kullanacağız.

Aşama 8 - PuTTY'nin Kurulması

Windows SSH istemcisi içermez, bu nedenle bizim bir SSH istemcisi indirip yüklememiz gerekecektir. Bu kitap için **PuTTY** kullanarak Homestead VM'imize bağlanacağız.

Aşama 8.1 - PuTTY İndirme ve Yükleme

putty-0.63-installer[17] dosyasını indirin. PuTTY yüklemesi yapmak için bu dosyayı çalıştırın. Varsayılan yükleme ayarlarını kullabilirisiniz.

Aşama 8.2 - SSH Anahtarının Dönüştürülmesi

Windows Başlangıç Menüsü'nden PuTTYgen'i bulup çalıştırınız. Menüden **Conversions** açın ve **Import key** seçin. Aşama 4.4'te oluşturulan `id_rsa` dosyasına seçin. **[Save private key]** butonuna basın. Evet, anahtarı şifresiz kaydetmek isteriz, bunu da aynı dizin içine, –benim için `C:\Kullanıcılar\Sineld\.ssh– id_rsa.ppk` dosya ismi ile kaydedin.

[17] http://the.earth.li/~sgtatham/putty/latest/x86/putty-0.63-installer.exe

Aşama 8.3 - Homestead PuTTY Oturumunu Ayarlayın

PuTTY'yi kurun ve **Connection | SSH | Auth** adımları ile henüz oluşturduğunuz `id_rsa.ppk` anahtarınızı tanımlayın. Session Hostname değerini `vagrant@127.0.0.1` ve portu `2222` olarak ayarlayın.

Oturumu **homestead** ismi ile kaydedin.

Oturumu ilk başlattığınızda onay penceresi karşınıza çıkacaktır ancak Homestead Sanal Makinesi'ne bağlandıktan sonra tekrar şifre sormayacaktır.

Masa üstüne kısayol oluşturmak isteyebilirsiniz. Oluşturduğunuz kısayol `"C:\Program Files (x86)\PuTTY\Putty.exe" -load homestead` adresine işaret etmelidir ve ismini **homestead** koyabilirsiniz.

PuTTY'nin Yazıtipini Değiştirmek

PuTTY varsayılan olarak *Courier New* yazıtipini kullanır, bu bana son derece çirkin görünüyor. **Window | Appearance** ayarlarına giderek yazıtipi, boyut, renk vs. değiştirebilirsiniz.

Aşama 8.4 - Homestead'e PuTTY ile Bağlanmak

Masa üstüne PuTTY için oluşturmuş olduğunuz **homestead** oturumunu başlatın ve karşınıza aşağıdakine benzer bir pencere çıkacaktır.

Homestead'in İlk Ekranı

```
Using username "vagrant".
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-11-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Fri Nov 28 04:24:01 UTC 2014

  System load:  0.0               Processes:           92
  Usage of /:   5.2% of 39.34GB   Users logged in:     0
  Memory usage: 33%               IP address for eth0: 10.0.2.15
  Swap usage:   0%                IP address for eth1: 192.168.10.10

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
```

```
    http://www.ubuntu.com/business/services/cloud

Last login: Fri Nov 28 04:24:01 2014 from 10.0.2.2
vagrant@homestead:~$
```

Aşama 9 - Laravel Yükleyicisinin Yüklenmesi

Son adım olarak Laravel Yükleyicisini yükleyeceğiz.

Laravel Yükleyicisinin Evrensel Yüklenmesi

```
C:\Kullanıcılar\Sineld> composer global require "laravel/installer =~1.1"
Changed current directory to C:\Users/Sineld/AppData/Roaming/Composer
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing guzzlehttp/streams (2.1.0)
    İndiring: 100%

  - Installing guzzlehttp/guzzle (4.2.3)
    İndiring: 100%

  - Installing laravel/installer (v1.1.3)
    İndiring: 100%

Writing lock file
Generating autoload files
```

Aşama 6.2'de PATH veriyolunuz zaten composer'ın bin dizinini içerdiğinden `laravel` komutu DOS komut istemininden erişebilir olacaktır. Bunu doğrulamak için versiyon denetimi yapın.

Laravel Yükleyicisinin Sürümünün Denetlenmesi

```
C:\Kullanıcılar\Sineld>laravel --version
Laravel Installer version 1.1
```

Tebrikler!

Şu anda Laravel 5.1 ile web uygulamaları geliştirmek için Ubuntu 64-bit sanal makineniz hazır.

Özet

Bu bölüm Laravel Homestead'i Windows makinenizde çalıştırabilmek için birtakım adımlar listesi oldu. İyi haber ise bu adımların sadece bir defa yapılması gerektiği.

Şimdi Homestead hakkında daha detaylı bilgi için **Homestead'i Kullanmak** bölümüne geçin.

Bölüm 4 - OS X veya Linux Makine Kurulumu

Bu bölümde OS X ve Linux makinesine Laravel Homestead için gerekli uygulamaların yüklenmesi ve kurulması için gereken adımları sırasıyla gerçekleştireceğiz. **VirtualBox** ve **Vagrant**'ın önceki bölümde anlatıldığı gibi yüklenmiş olduğu varsayılmıştır.

Bölüm İçerikleri

Linux ile Hafif Varyasyonları

Farklı Linux dağıtımları arasında küçük farklılıklar vardır. Özellikle, paket yöneticisi. CentOS ve Fedora paket yöneticisi olarak **yum** kullanır, Ubuntu **apt** kullanır. OS X'in App Store haricinde resmi bir "paket yöneticisi" yoktur, ama **homebrew** resmi olmayan paket yöneticisidir. Farklılıklar ne olursa olsun, OS X de dahil olmak üzere, hemen hemen tüm *nix sistemlerin özü aynıdır.

Aşama 1 - PHP Yüklenmesi

Genelde PHP sisteminizde öntanımlı olarak kurulu gelecektir. Terminal penceresinden sürümü denetleyebilirsiniz.

PHP Sürümünün Denetlenmesi

```
~> php --version
PHP 5.5.9-1ubuntu4.5 (cli) (built: Oct 29 2014 11:59:10)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2014 Zend Technologies
    with Zend OPcache v7.0.3, Copyright (c) 1999-2014, by Zend Technologies
```

Laravel 5.1 PHP 5.5.9 ve üstü sürüm gerektirir. PHP kurulu değilse veya sürümü en az 5.5.9 değilse paket yöneticisi ile yükleyebilirsiniz.

OS X Yosemite

Yosemite (bu metni yazarken) PHP versiyon 5.5.14 ile geliyordu. Yani endişeye gerek yok.

Ubuntu'da PHP Yüklenmesi

```
~> sudo apt-get install php5
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  php5
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.

[snip]
```

Aşama 2 - Node.js Yüklenmesi

Gulp kullanabilmek için Node.js yüklenmesi gerekmektedir.

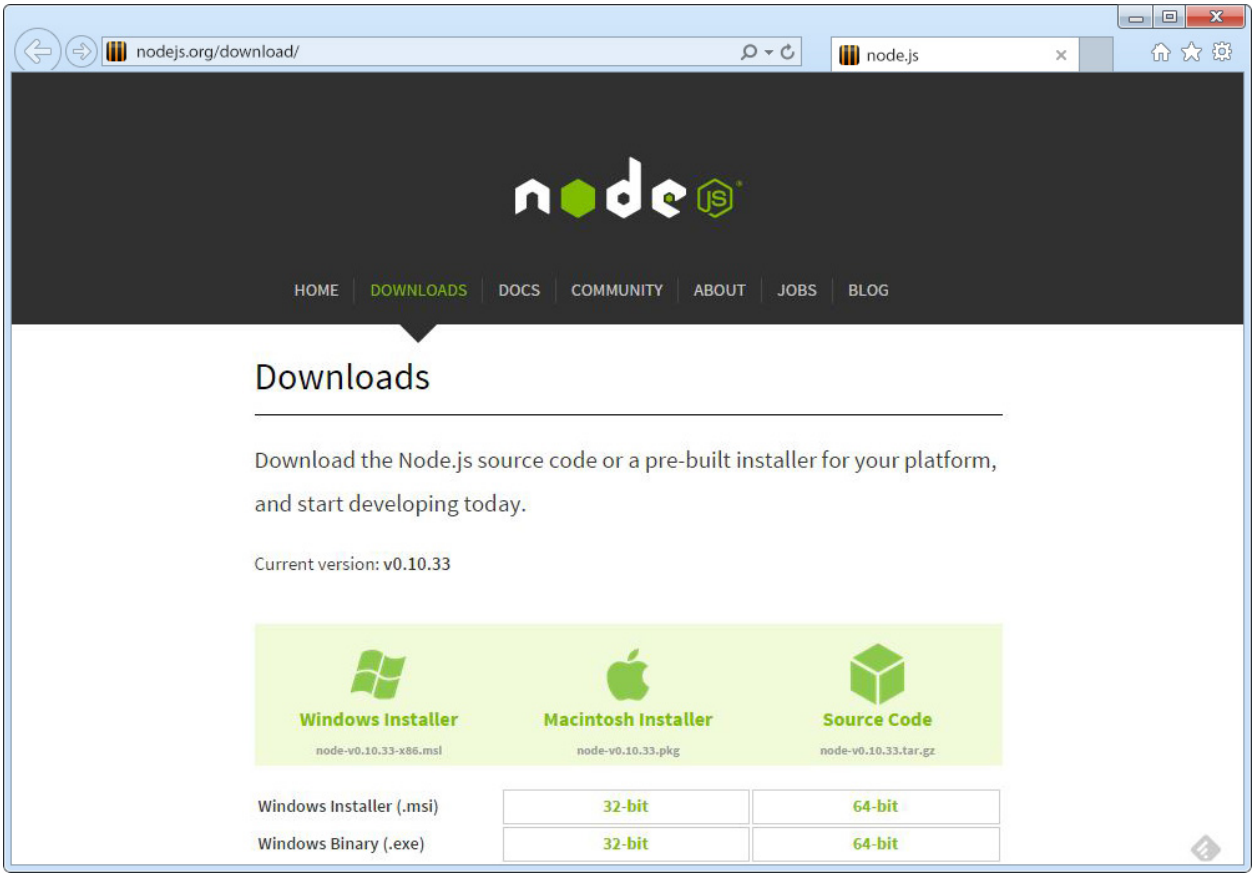
Genelde Node.js zaten kuruludur. **npm** sürümünü denetleyerek Node.js'nin kurulu olduğundan emin olabilirsiniz.

npm Sürümünün Denetlenmesi

```
$> npm --version
1.5.0-alpha-4
```

Eğer yüklü değilse yüklemenin birkaç seçeneği vardır. Yüklemek için paket yöneticinizi kullanabilirsiniz. OS X'de Homebrew ile yükleyebilirsiniz veya nodejs.org/download[18] adresine giderek işletim sisteminiz için doğru sürümü indirebilirsiniz.

[18] http://nodejs.org/download



Node.js İndirme Sayfası

Kurulumdan sonra node ve npn sürümlerini denetleyerek yüklendiğinden emin olabilirsiniz.

node ve npm sürümlerinin denetlenmesi

```
~> node --version
v0.10.29

~> npm --version
1.5.0-alpha-4
```

Aşama 3 - Gulp Yüklenmesi

Gulp hızlı Laravel geliştirmenin ayrılmaz bir parçasıdır. Node paket yükleyicisi (NPM) ile evrensel yükleyebilirsiniz.

Gulp'ın evrensel yüklenmesi

```
~> npm install -g gulp
/usr/local/bin/gulp -> /usr/local/lib/node_modules/gulp/bin/gulp.js
gulp@3.8.10 /usr/local/lib/node_modules/gulp
[snip]
```

Gulp Sürümünün Denetlenmesi

```
~> gulp --version
[10:13:44] CLI version 3.8.10
```

İsteğe bağlı Bower yüklenmesi

Linux veya OS X terminalinden Bower'a erişmek için isteğe bağlı olarak evrensel yükleyebilirsiniz. Kişisel olarak çoğunlukla içinde olduğumdan Homestead Sanal Makinesi'ndekini kullanırım.

Node paket yöneticisi (NPM) ile Bower'ı evrensel yükleyin.

Bower'ın evrensel yüklenmesi

```
~> npm install -g bower
/usr/local/bin/bower -> /usr/local/lib/node_modules/bower/bin/bower
bower@1.3.12 /usr/local/lib/node_modules/bower
[snip]
```

Bower Sürümünün Denetlenmesi

```
~> bower --version
1.3.12
```

Unutmayın, bu sadece programları evrensel yükler

Eğer gulp (veya bower'ı) belirli bir proje içinde kullanacaksanız proje dizini içinde `npm install` komutu ile yükleme yapmanız gerektiğini unutmayın. (-g seçeneğini atlayarak). Buna daha sonra değinilecektir.

Aşama 4 - Composer Yüklenmesi

Composer PHP için paket yöneticisidir. *nix sistemlerde (hem OS X hem de Linux) terminal penceresinden kolaylıkla yüklenebilir. Alternatif olarak OS X üzerine Composer'ın Homebrew ile yüklenmesine bu bölüm sonunda değinilecektir.

Composer Yüklenmesi

```
~> curl -sS https://getcomposer.org/installer | php
#!/usr/bin/env php
All settings correct for using Composer
Downloading...

Composer successfully installed to: /Users/sineld/composer.phar
Use it: php composer.phar
```

`composer.phar` indirildiğinde, onu evrensel veriyoluna taşıyınız.

composer.phar taşınması

```
~> sudo mv composer.phar /usr/local/bin/composer
```

Ve sürüm denetimi yaparak erişibilir olduğundan emin olun.

Composer sürümü denetlenmesi

```
~> composer --version
Composer version 1.0-dev (b23a3cd36870ff0eefc161a4638d9fcf49d998ba)\
2014-11-21 17:59:11
```

Homebrew ile yüklenmesi

OS X'te, eğer Homebrew kullanıyorsanız, şu yönergeler ile Composer yükleyebilirsiniz.

OS X'te Homebrew ile Alternatif Yükleme

```
~> brew update
~> brew tap homebrew/dupes
~> brew tap homebrew/php
~> brew install composer
```

Aşama 5 - SSH Anahtarının Eklenmesi

Eğer daha önceden SSH anahtarını makinenize eklemediyseniz şimdi yapmanız gerekecektir.

SSH Anahtarlarının Denetlenmesi

```
~> ls ~/.ssh
config id_rsa id_rsa.pub
```

Şayet `id_rsa` ve `id_rsa.pub` dosyalarını üstteki komut ile göremiyorsanız aşağıdaki komut ile oluşturunuz. Tüm onay pencereleri boyunca [Enter] tuşuna basarak varsayılanları kullanın, bu sayede şifresiz bir SSH anahtarı oluşturmuş olacaksınız.

SSH Anahtarlarının Oluşturulması

```
~> ssh-keygen -t rsa -C "your@email.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/sineld/.ssh/id_rsa):
Created directory '/Users/sineld/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

Aşama 6 - Homestead Kutusunun Eklenmesi

Bu adımda Laravel Homestead Vagrant kutusu indirilecektir.

Linux'ta Homestead kutusunun eklenmesi

```
~> vagrant box add laravel/homestead
==> box: Loading metadata for box 'laravel/homestead'
    box: URL: https://vagrantcloud.com/laravel/homstead

[snip]
```

Bu işlem yavaş bağlantıda biraz zaman alacaktır.

Aşama 7 - Homestead Yüklenmesi

Şimdi `homestead` komutunun yüklenmesi için composer kullanacağız. Bu komut satısı aracı Homestead VM'in kolay denetimini sağlar.

Aşama 7.1 - Homestead'in Evrensel Yüklenmesi

Homestead 2.0'ın Evrensel Yüklenmesi

```
~> composer global require "laravel/homestead =~2.0"
Changed current directory to /home/sineld/.composer
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing symfony/process (v2.5.7)
    Loading from cache

  - Installing symfony/console (v2.5.7)
    Loading from cache

  - Installing laravel/homestead (v2.0.7)
    Loading from cache

Writing lock file
Generating autoload files
```

Aşama 7.2 - Veriyolu'nun Güncellenmesi

Composer Homestead'i Composer yüklenmesinin `vendor` dizinine ekledi. (Örneğin Linux'ta `/home-/sineld/.composer` ve OS X'te `/Users/sineld/.composer` altına).

Homestead'e herhangi bir komut istemi ekranında erişebilmek için bu veriyolunu PATH değişkenine ilave edin. Bu işletim sisteminizin başlangıç script'i hangisi ise ona eklenmelidir. Genel başlangıç dosyaları şunlardır: `.bashrc`, `.bash_profile`, `.zshrc`, etc.

Başlangıç script'inizin altına şu satırı ilave edin:

Updating path in the startup script

```
export PATH="~/.composer/vendor/bin:vendor/bin:$PATH"
```

Sondaki ilave 'vendor\bin' dikkatinizi çekti mi?

Bunu eklediğimiz için Laravel proje dizini içinde herhangi bir vendor aracına kolaylıkla ulaşabilirisniz. Örneğin, **phpunit** her Laravel uygulamasında `vendor/bin` dizini içinde bulunur.

Aşama 7.3 - Homestead Yüklenmesini Doğrulamak

Tüm değişikliklerin etkin olması için, terminal pencerelerini kapatın ve yeni birini açın. Doğru yüklendiğinden emin olmak için homestead sürümünü denetleyin.

Homestead Sürümünün Denetlenmesi

```
~>homestead --version
Laravel Homestead version 2.0.7
```

Aşama 7.4 - Homestead'in Başlatılması

Homestead komutunun kurulumunu yaptıysanız ve composer bin dizinini veriyolunuza eklediyseniz Homestead'i başlatmalısınız.

Homestead'i Başlatmak

```
~> homestead init
Creating Homestead.yaml file...
Homestead.yaml file created at: /home/sineld/.homestead/Homestead.yaml
```

Unutmayın

Homestead'i makinenizde yalnızca bir defa başlatmalısınız.

Aşama 8 - Homestead VM'i Ayağa Kaldırmak

`homestead up` komutunu kullanarak Homestead'i ayağa kaldırmak ve projelerinizi saklamak için öncelikle `Code` dizinini oluşturmalısınız.

Homestead'i ilk defa çalıştırmak

```
~> mkdir Code
~> homestead up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'laravel/homestead'...
==> default: Matching MAC address for NAT networking...
==> default Checking if box 'laravel/homestead' is up to date...

[snip]
```

Şu anda Homestead Sanal Makineniz çalışıyor. Eğer terminal penceresini kapatsanız bile VM çalışmasını sürdürecektir. Siz `homestead halt` komutunu kullanıncaya kadar çalışmasını sürdürücektir.

Şimdi Homestead Sanal Makinenize `homestead ssh` komutu ile giriş yapabilirsiniz.

Homestead'e Bağlantı Kurmak

```
~> homestead ssh
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-11-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Fri Nov 28 04:24:01 UTC 2014

  System load:  0.0               Processes:           92
  Usage of /:   5.2% of 39.34GB   Users logged in:     0
  Memory usage: 33%               IP address for eth0: 10.0.2.15
  Swap usage:   0%                IP address for eth1: 192.168.10.10

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

Last login: Fri Nov 28 04:24:01 2014 from 10.0.2.2
vagrant@homestead:~$
```

Aşama 9 - Laravel Yükleyicisinin Yüklenmesi

Son adım olarak Laravel Yükleyicisini yükleyeceğiz. Bunu terminal ile ana makinenizde yapacaksınız (Homestead Sanal Makineniz'de değil).

Laravel Yükleyicisinin Evrensel Yüklenmesi

```
~> composer global require "laravel/installer =~1.1"
Changed current directory to /Users/sineld/.composer
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing guzzlehttp/streams (2.1.0)
    Downloading: 100%

  - Installing guzzlehttp/guzzle (4.2.3)
    Downloading: 100%

  - Installing laravel/installer (v1.1.3)
```

```
    Downloading: 100%

Writing lock file
Generating autoload files
```

Aşama 7.2'de PATH veriyolunuz zaten composer'ın bin dizinini içerdiğinden `laravel` komutu DOS komut istemininden erişebilir olacaktır. Bunu doğrulamak için versiyon denetimi yapın.

Laravel Yükleyicisinin Sürümünün Denetlenmesi

```
~> laravel --version
Laravel Installer version 1.2.1
```

Tebrikler!

Şu anda Laravel 5.1 ile web uygulamaları geliştirmek için Ubuntu 64-bit sanal makineniz hazır.

Özet

Bu bölüm Laravel Homestead'i OS X veya Linux makinenizde çalıştırabilmek için birtakım adımlar listesi oldu. İyi haber ise bu adımların sadece bir defa yapılması gerektiği.

Şimdi Homestead hakkında daha detaylı bilgi için **Homestead'i Kullanmak** bölümüne geçin.

Bölüm 5 - Homestead ve Laravel Yükleyicisi

Bu bölümde daha önceden yüklemiş olduğunuz iki composer paketini inceleyeceğiz: **homestead** ve **laravel** Tipik bir günlük iş akışı incelendiğinde, bir yeni laravel 5.1 projesi kurmak için altı adım vardır.

Bölüm İçerikleri

Homestead Aracı

Konsol Tanımlı

Konsoldan bir şey yapmak istendiğinde zaman bağlam önemlidir. **homestead console** Homestead VM'e SSH ile bağlanmak demektir. Windows için bu, PuTTY kullanmak demektir *(Windows Makine Kurulumu bölümünde açıklanmıştır)*. Diğer işletim sistemlerinde terminalden `homestead ssh` komutunu çalıştırabilirsiniz. Bu kitapta nerede `$` işareti görürseniz homestead konsolda olduğunuzu anlamalısınız.

OS konsol hem Windows Komut İstemi hem de terminal uygulaması demektir. (Bu kitapta göreceğiniz `%` işareti sizin işletim sisteminize özel işaretçidir.)

İşletim sisteminizin *konsolundan*, herhangi bir argüman girmeden `homestead` komutu ile kolayca homestead komutlarına erişebilirsiniz.

Homestead Komutları

```
% homestead
Laravel Homestead version 2.0.9

Usage:
  [options] command [arguments]

Options:
  --help           -h Display this help message.
  --quiet          -q Do not output any message.
  --verbose        -v|vv|vvv Increase the verbosity of messages: 1 for normal \
output, 2 for more verbose output and 3 for debug.
  --version        -V Display this application version.
  --ansi              Force ANSI output.
  --no-ansi           Disable ANSI output.
  --no-interaction -n Do not ask any interactive question.

Available commands:
  destroy     Destroy the Homestead machine
  edit        Edit the Homestead.yaml file
  halt        Halt the Homestead machine
  help        Displays help for a command
  init        Create a stub Homestead.yaml file
  list        Lists commands
  provision   Re-provisions the Homestead machine
  resume      Resume the suspended Homestead machine
  run         Run commands through the Homestead machine via SSH
  ssh         Login to the Homestead machine via SSH
  status      Get the status of the Homestead machine
  suspend     Suspend the Homestead machine
  up          Start the Homestead machine
  update      Update the Homestead machine image
```

Her gün karşılaşacağınız temel komut `homestead up`'tır, bu komutla sanal makinenizi başlatmış olursunuz.

Ortak Homestead Komutlarına Genel Bakış

Yaygın olarak kullanılan Homestead komutlarına hızlı bir bakış.

homestead up
: Sanal Makineyi başlatır. Bu VM'in gücünü açar. Eğer provision seçeneğini kullanırsanız (`homestead up --provision`) eklenmiş olan yeni web siteleri çalışmaya başlayacaktır.

homestead halt
: Sanal Makineyi durdurur. Bir diğer deyişle gücü kapatır.

homestead suspend
: Sanal Makineyi askıya alır, uyku moduna geçmesi gibi.

homestead resume
: Sanal Makineyi sürdürür.

homestead edit
: Homestead.yaml dosyasını düzenle. İşletim sisteminizin YAML dosyalarını düzenlemeye tanımlı dosya editör uygulaması ile Homestead.yaml dosyası başlatılır.

homestead status
: Homestead Sanal Makinenizin durumunu görüntüleyin.

Homestead.yaml İncelenmesi

Laravel Homestead'in yapılandırma ayarları `Homestead.yaml` dosyası içindedir. Bu dosya ana bilgisayarınızın kullanıcı hesabının ev dizininde `.homestead` dizini içindedir.

Dosyayı incelerseniz aşağıdakileri göreceksiniz.

Homestead.yaml Dosya İçeriği

```
---
ip: "192.168.10.10"
memory: 2048
cpus: 1

authorize: ~/.ssh/id_rsa.pub

keys:
    - ~/.ssh/id_rsa

folders:
    - map: ~/Code
      to: /home/vagrant/Code

sites:
```

```
    - map: homestead.app
      to: /home/vagrant/Code/Laravel/public

databases:
    - homestead

variables:
    - key: APP_ENV
      value: local
```

İşte ayarların her birinin tanımı.

ip Makineye erişmek için kullanılan iç IP.

memory
VM'in kullanacağı bellek miktarı.

cpus VM'in kullanacağı İşlemci sayısı.

authorize
Burası yerel SSH anahtarınıza işaret etmelidir.

keys Özel SSH anahtarlarınız.

folders
Paylaşım dizinleri. Bunlar ana bilgisayarınızın işletim sisteminizin Sanal Makine içinde görüntülenecek dizinleridir. Windows için `~/Code` dizini `C:\Users\sineld\Code` dizinine eşitlenir. OS X'te, bu `/Users/sineld/Code` dizinidir. Linux'ta ise genelde `/home/sineld/Code` dizinidir. Ana bilgisayardaki bu dizin içindeki dosyalardan birini değiştirdiğinizde, anında Homestead Sanal Makinesinde görünür olur.

sites Sitelerin bir listesi (her alanadının işaret ettiği veriyolu) Homestead Sanal Makinesi provision yaptığınız her seferde yeniden ayarlanacaktır.

databases
Homestead'in otomatik oluşturacağı veritabanı listesi.

variables
homestead'in kullanımına hazır edilecek ortam değişkenleri.

Yapılandırma Notu

Yapılandırma listesinde genelde tek değiştirdiğim veritabanı isminin `homestead`'den `xhomestead`'e dönüştürülmesidir. Bu sayede yeni bir Laravel uygulaması oluşturur ve veritababını oluşturmazsam hata meydana gelir. *(Aksi takdirde yeni uygulama için varsayılan veritabanı `homestead` olacağından farkında olmadan `homestead` veritabanını kullanacağım.)*

Şimdilik, **databases** dışında (ki onu da eğer isterseniz) hiç bir homestead yapılandırma değerini değiştirmeyiniz.

Homestead Sanal Makine Detayları

Anahtar	Değer
Makine Adı	homestead
IP Adresi	192.168.10.10
Kullanıcı	vagrant
Yönetici Şifresi	vagrant
Veritabanı Makinesi	127.0.0.1
Veritabanı Portu	33060
Veritabanı Kullanıcısı	homestead
Veritabanı Şifresi	secret

Homestead VM'e Yazılım Eklenmesi

Homestead Sanal Makinesi'ne yeni yazılım yüklemek istediğinizde Ubuntu'nun aracı `apt-get`'i kullanın.

Bu iki basit aşamalı bir işlemdir.

1. Ubuntu'yu güncelle
2. apt-get ile yükle

Örneğin, zip arşivleri ile ilgilenmek için kullanışlı bir araç olan **unzip**'i yükleyelim.

Önce, Ubuntu'yu Güncelle

Ubuntu Yazılımın Güncellenmesi

```
vagrant@homestead:~$ sudo apt-get update
vagrant@homestead:~$ sudo apt-get upgrade
```

Devam edebilmek için "Y" seçmek zorunda kalabilirsiniz. Eğer yükleme süresince istenirse seçeneklerden mevcut olanı veya en iyi olanı seçebilirsiniz.

Homestead VM içindeki Ubuntu güncellediyse, **unzip** yüklemesine geçiniz.

Sonra, apt-get ile unzip yükle

Homestead VM içine unzip yüklenmesi

```
vagrant@homestead:~$ sudo apt-get install unzip
```

Günlük İş Akışı

Homestead ile çalışırken günlük iş akışı üç adımdan oluşur:

Adım 1 - homestead up - Güne Homestead Sanal Makinesini başlatmakla başlayın.

Adım 2 - homestead ssh veya **PuTTY** - Homestead VM içine SSH ile erişin ve artisan komutları çalıştırın.

Adım 3 - güzel kodlar yazın - Favori editörünüz ile işletim sisteminizde güzel kodlar yazın.

İsteğe bağlı 4. Adım - homestead halt - Gün sona erdiğinde isteğe bağlı olarak `homestead halt` komutu ile makinayı kapatabilirsiniz.

Yeni Bir Laravel 5.1 Projesine Başlamak için Altı Aşama

Yeni bir Laravel 5.1 uygulamasına başlarken takip etmeniz gereken basit altı aşama vardır.

test.app adında bir proje oluşturup proje dizini olarak **test** kullanacağımızı varsayalım.

Aşama 1 - Uygulamanın Omurgasını Oluşturma

Laravel Yükleyicisi (`laravel` komutu yüklemesi önceki bölümde anlatılmıştır) kullanarak proje omurgası oluşturmak son derece kolaydır.

Yeni Bir Uygulama Omurgası Oluşturma

```
~/Code % laravel new test
Crafting application...
Generating optimized class loader
Compiling common classes
Application key [rzUhyDksVxzTXFjzFYiOWToqpunI2m6X] set successfully.
Application ready! Build something amazing.
```

Aşama 2 - Web Sunucusunun Ayarlanması

Uygulama omurgası hazır olduğuna göre homestead içindeki Nginx web sunucusuna uygulamanın public dizinini servis etmesini ayarlayabiliriz.

Homestead ortamı bunu `serve` komutu ile kolayca halleder.

Homestead içinde sanal sunucu ayarlanması

```
~/Code$ serve test.app ~/Code/test/public
dos2unix: converting file /vagrant/scripts/serve.sh to Unix format ...
 * Restarting nginx nginx                                                [ OK ]
php5-fpm stop/waiting
php5-fpm start/running, process 2169
```

`serve` komutu ile `/etc/nginx/sites-available` içinde kullanacağımız yeni bir alanadı (**test.app**) dosyası oluşturulur ve bu dosyaya `/etc/nginx/sites-enabled` içinden sembolik bağlantı verilir.

Makineyi yeniden başlattığınızda bu yapılandırma dosyası yerinde bulunacaktır.

Neden Homestead.yaml dosyasını düzenlemedik?

Diğer alternatif **test.app** sitesi için sanal olarak `homestead edit` komutu ile dosyayı açıp *sites:* bölümüne eklemek ve parametreleri ayarlamaktır. Ama bu kolay olanıdır Homestead VM'i yeniden provision etmek gerekmez.

Ancak her zaman tanımlı olmasını istediğiniz bir uygulama kuracaksanız Homestead.yaml dosyasını düzenlemek kötü bir fikir değildir.

Aşama 3 - Hosts Dosyanıza Host'un Eklenmesi

Eklediğiniz alanadı **test.app** DNS içinde var olmadığından, ana bilgisayar işletim sistemi içindeki host dosyasına kayıt eklenmelidir. Linux ve OS X için `/etc/hosts` dosyasını düzenleyin. Windows'ta bu dosya `C:\Windows\System32\drivers\etc\hosts` yolundadır. Bu dosya içinde **test.app** alanadı `Homestead.yaml` içindeki IP'ye işaret etmelidir.

Aşağıdaki satırı host dosyasına ekleyin.

test.app için host girdisi

```
192.168.10.10  test.app
```

Windows bu dosyanın düzenlenmesi için yönetici yetkisi gerektirir

Windows'ta editörünüzü (Not Defteri, Wordpad veya, Sublime Text'i) yönetici olarak çalıştırınız. Linux veya OS X'te `sudo` komutunu kullanabilirsiniz.

Linux veya OS X'te hosts dosyasının düzenlemek

```
sudo nano /etc/hosts
// veya
sudo vi /etc/hosts
```

Aşama 4 - NPM Yerel Yüklemeler

Daha sonra **gulp** kullanabilmek için tüm gerekli npm modüllerinin yerel olarak yüklenmiş olduğundan emin olun.

Eğer gulp kullanmayacağınızdan eminseniz bu aşamayı geçebilirsiniz.

Ana bilgisayarınızdaki proje dizinine gidin ve aşağıdaki komutu çalıştırın.

NPM Yerel Yüklemeler

```
~% cd Code/test
~/Code/test% npm install
npm WARN package.json @ No repository field.

> v8flags@1.0.5 install /Users/chuck/Code/test/node_modules/gulp/\
node_modules/v8flags
> node fetch.js

flags for v8 3.14.5.9 cached.

[snip]
```

Bu komut ile gulp için gerekli olan her şey projenizin `node_modules` dizini içine yüklenecektir.

Aşama 5 - Uygulamanın Veritabanının Oluşturulması

Eğer uygulamanız bir veritabanı gerektiriyorsa Homestead VM içinde **mysql** konsolu ile oluşturması çok kolaydır.

Homestead VM içinde veritabanı oluşturmak

```
$ mysql --user =homestead --password =secret
mysql> create database test;
mysql> exit;
```

Veritabanı oluşturulduysa projenizin ana dizininde bulunan `.env` dosyasını düzenleyin ve `DB_NAME` değerini doğru isimle değiştirin.

.env içindeki DB_NAME'i değiştirmek

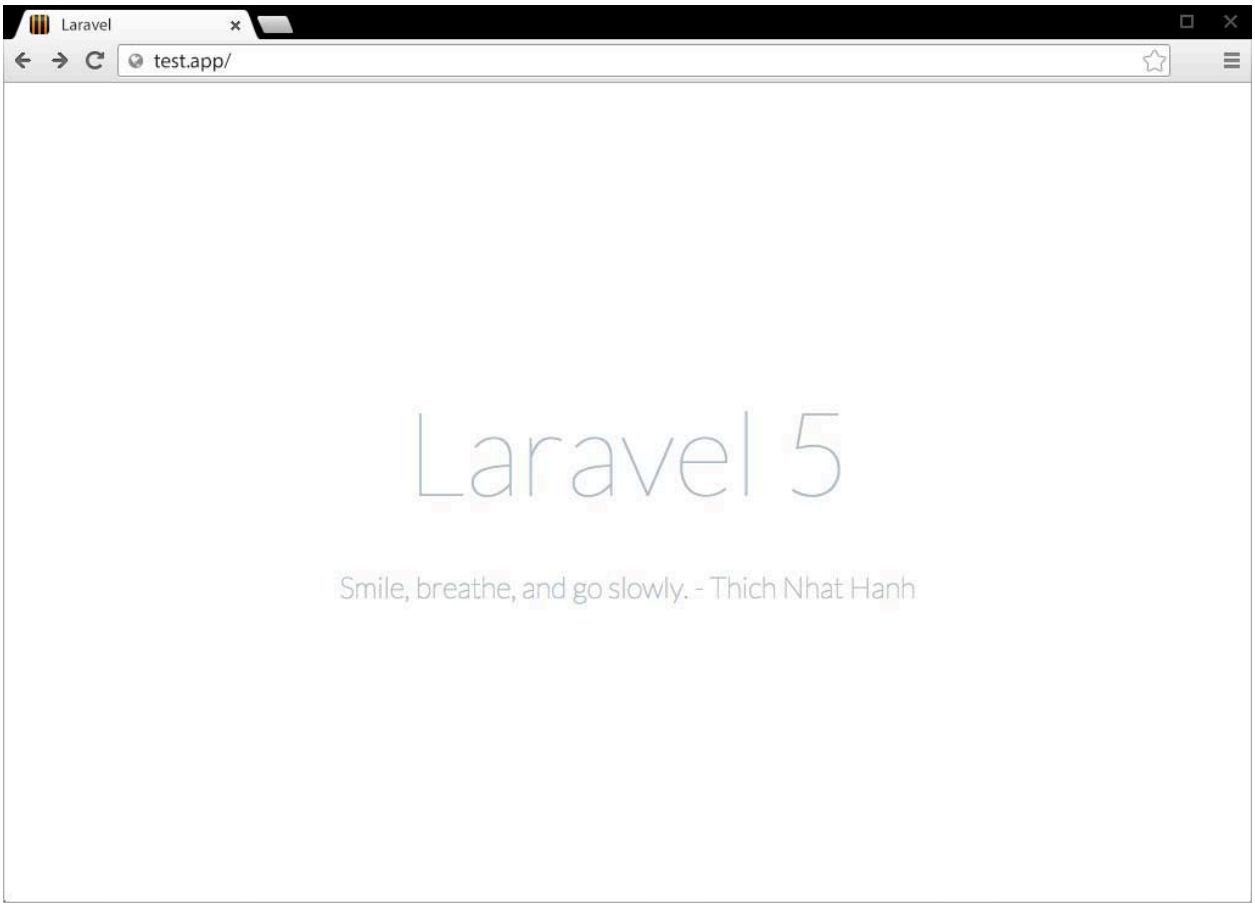
```
// Aşağıdaki satırı bulun
DB_DATABASE =homestead

// doğru değer ile değiştirin
DB_DATABASE =test
```

Kolay! Artık veritabanının migrasyonunu ve oluşturulmasını yapabilirsiniz. Buna sonraki bölümlerden birinde değineceğiz.

Aşama 6 - Tarayıcının Denetlenmesi

Tarayıcınızı `http://test.app` adresine işaret edin, aşağıdakine benzer bir ekran görmelisiniz.



Laravel Varsayılan Sayfası

Eğer farklı bir ekran ile karşılaştıysanız birşeyler ters gitmiştir.

Diğer Homestead İpuçları

Kaynak Kodlarınızı Ana İşletim Sisteminizde Düzenleyin

Buna önceki bölümlerde değinildiği halde tekrar üstünden geçmekte yarar var.Kaynak kodlarınızı her zaman ana işletim sisteminizde düzenleyin. Paylaşım dizinleri sayesinde `~/Code` dizini içindeki her şey anında Homestead Sanal Makinesinde görünür olur.

.homestead/aliases dosyasını kullanın

Homestead'i `homestead up --provision` veya `homestead provision` ile her yeniden provision ettiğinizde `.homestead/aliases` dosyası Homestead Sanal Makinesi aliases'ı günceller.

Bu takma adları (alias), fonksiyonları ve hatta ortam değişkenlerini eklemek için ideal bir yerdir.

Homestead VM'i güncel tutun

Daha önce de değinildiği gibi iki komut ile Homestead Sanal Makinenizin Ubuntu işletim sistemini güncel tutabilirsiniz.

Ubuntu'yu Güncel Tutmak

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Özet

Bu bölüm `homestead` ve `laravel` komutlarıyla ilgili detaylı bilgi verdi ve *Yeni Bir Laravel 5.1 Projesine Başlamak için Altı Aşama* konusuna değinildi.

Sonraki bölümde birtakım testler yapacağız.

Bölüm 6 - Test Etme

Bu bölümde kitap boyunca kullanacağımız ve test etmek için çeşitli seçeneklere değineceğimiz bir proje oluşturacağız. Markdown olarak biçimlendirilmiş dosyaların HTML'e döntüştürülmesi için bir sınıf oluşturacağız. Bu sınıf TDD (Test Driven Development = Test Güdümlü Geliştirme) prensiplerini kullanarak oluşturulacaktır.

Bölüm İçerikleri

l5beauty Projesinin Oluşturulması

Yeni Bir Laravel 5.1 Projesine Başlamak için Altı Aşama'yı takip ederek *l5beauty* projesini oluşturun.

Öncelikle, ana bilgisayardan uygulama omurgasını oluşturun.

Aşama 1 - Uygulama omurgasının yüklenmesi

```
~/Code % laravel new l5beauty
Crafting application...
Generating optimized class loader
Compiling common classes
Application key [rzUhyDksVxzTXFjzFYiOWToqpunI2m6X] set successfully.
Application ready! Build something amazing.
```

Ardından, Homestead VM içinde `l5beauty.app` sanal sunucusunu oluşturun.

Aşama 2 - Web sunucusunun yapılandırılması

```
~/Code$ serve l5beauty.app ~/Code/l5beauty/public
dos2unix: converting file /vagrant/scripts/serve.sh to Unix format ...
 * Restarting nginx nginx                                                [ OK ]
php5-fpm stop/waiting
php5-fpm start/running, process 2169
```

Ana bilgisayarınızda aşağıdaki satırı hosts dosyanıza ilave edin.

Aşama 3 - Host dosyasına l5beauty.app eklenmesi

```
192.168.10.10  l5beauty.app
```

Ana bilgisayarınızdan NPM paketlerinin yerel yüklenmesini başlatın.

Aşama 4 - NPM Yerel Yüklemeleri

```
~% cd Code/l5beauty
~/Code/l5beauty% npm install
|
> node-sass@2.0.1 install /Users/sineld/Code/l5beauty/node_modules/laravel-\
        elixir/node_modules/gulp-sass/node_modules/node-sass
> node scripts/install.js

> node-sass@2.0.1 postinstall /Users/sineld/Code/l5beauty/node_modules/\
        laravel-elixir/node_modules/gulp-sass/node_modules/node-sass
> node scripts/build.js

`darwin-x64-node-0.10` exists; testing
Binary is fine; exiting
```

```
gulp@3.8.11 node_modules/gulp
├── v8flags@2.0.2
├── pretty-hrtime@0.2.2

[snip]
```

Geri dönüp Homestead VM içinden proje için veritabanı oluşturun.

Aşama 5 - uygulamanın veritanının oluşturulması

```
$ mysql --user =homestead --password =secret
mysql> create database l5beauty;
Query OK, 1 row affected (0.00 sec)

mysql> exit;
Bye
```

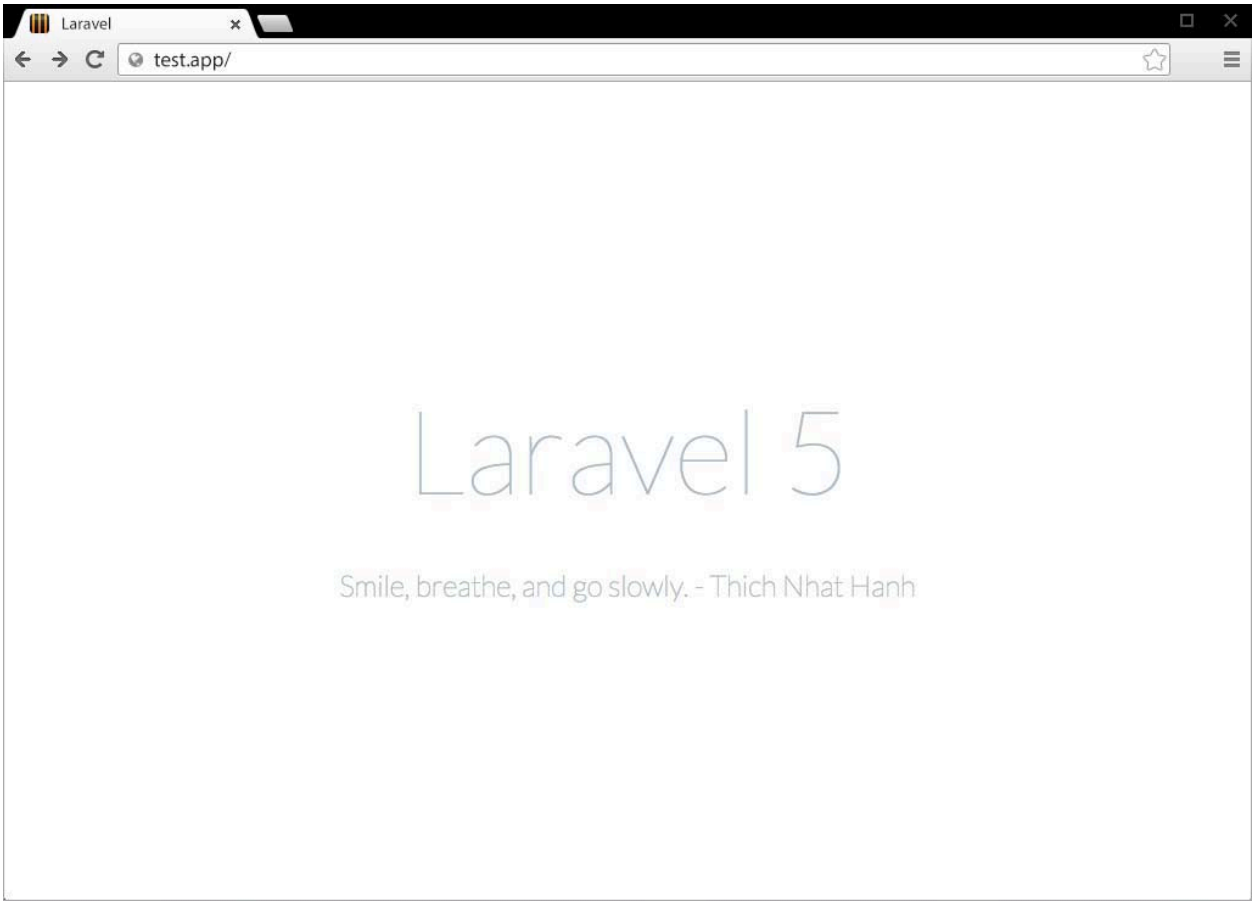
`.env` dosyasını düzenleyin ve veritabanını `l5beauty` olarak değiştirin.

DB_NAME yapılandırılmasının değiştirilmesi

```
// Aşağıdaki satırı bulun
DB_DATABASE =homestead

// doğru değer ile değiştirin
DB_DATABASE =l5beauty
```

Son olarak, herşeyin yolunda gittiğinden emin olmak için tarayıcınızı `http://l5beauty.app` adresine işaret edin.



Laravel Varsayılan Sayfası

PHPUnit'i Çalıştırma

Laravel 5.1 kutudan test edilebilir şekilde hazır olarak çıkar. Hatta uygulamanın web talebi üzerine beklendiği şekilde 200 HTTP yanıtının verildiğini onaylayan basit bir test yazılmıştır.

PHPUnit'i çalıştırmak için uygulama ana dizininde `phpunit` komutunu çalıştırın.

PHPUnit'i Çalıştırma

```
~% cd Code/l5beauty
~/Code/l5beauty% phpunit
PHPUnit 4.7.4 by Sebastian Bergmann and contributors.

.

Time: 544 ms, Memory: 10.25Mb

OK (1 test, 2 assertions)
```

Hata mı aldınız?

Eğer **komut bulunamadı** veya **izin reddedildi** hatalarından biri ile karşılaşıyorsanız bu `phpunit`'in yüklenme hatasından kaynaklıdır. `phpunit` komutu `vendor/bin` dizini içinde bulunmalıdır ve bu dizin işletim sisteminizin veriyoluna Bölüm 3 veya 4'te eklenmiştir. Sorun Laravel Yükleyicisinin phpunit ve diğer araçlar için gerekli izinleri doğru ayarlayamayan bir hatasından kaynaklıdır.

Bu hatayı gidermek için aşağıdaki aşamaları takip edin.

Aşama 1 - `vendor` dizinini tamamen silin. İşletim sisteminize uygun komut hangisi ise onunla bu dizini silin.

Aşama 2 - `composer update` komutu ile `vendor` dizinini yeniden oluşturun: Bu komutu projenizin ana dizininde çalıştırmalı ve bunu ana işletim sisteminizde yapmalısınız.

Bu kadar. `phpunit` komutunu tekrar çalıştırmayı deneyin.

Laravel 5.1'in PHPUnit Yapılandırması

Her Laravel 5.1 proje dizini içinde PHPUnit'in `phpunit`komutu çalıştırıldığı zaman kullanması için gerekli yapılandırma ayalarının saklandığı bir `phpunit.xml` dosyası mevcuttur.

`phpunit.xml` dosyası incelendiği zaman `tests` dizinine işaret etmektedir. Bu dizin içinde iki dosyaya bulunmaktadır.

1. `ExampleTest.php` - `testBasicExample() adında bir test içerir.` `ExampleTest` sınıfı diğer dosyadaki `TestCase` üst sınıfından türetilmiştir.
2. `TestCase.php` - Laravel testleri türetmek için temel sınıf.

`ExampleTest.php` içindeki `testBasicExample()` metoduna göz atın.

testBasicExample() metodu

```
1   public function testBasicExample()
2   {
3     $this->visit('/')
4          ->see('Laravel 5');
5   }
```

Test der ki, "Ana sayfayı ziyaret etğimizde, 'Laravel 5' kelimelerini görmeliyiz." Test bundan daha kolay olabilir mi?

`TestCase` sınıfı ünite testleri için, ayrıca Laravel 5.1'e özgü uygulama metodları ve özellikleri sağlar. `TestCase` ayrıca ilave işleç (assertion) metodları ve *tarayıcı* tipi testler için uzun bir liste sağlar.

Laravel 5.1 Tarayıcı Metod ve Özellikleri

Tarayıcı (*Crawler*) testleri web uygulamanızın saylarının test edilmesine olanak sağlar. Güzel olansa bu testlerin çoğunun akıcı olması ve `$this` döndürerek yukarıdakine benzer `->visit()->see()` testleri inşa etmenize olanak sağlamalarıdır.

Mevcut özelliklerin ve metodların bazıları şunlardır.

`$response`
: Web uygulaması ise döndürülen son yanıt.

`$currentUri`
: Şu anda gösterilen URL.

`visit($uri)`
: (Akıcı) GET isteğiyle URI'ı ziyaret et.

`get($uri, array $headers = [])`
: (Akıcı) İsteğe bağlı başlıklar geçerek GET isteğiyle URI'ı ziyaret et.

`post($uri, array $data = [], array $headers = [])`
: (Akıcı) Belirli URI'a POST isteği yap.

`put($uri, array $data = [], array $headers = [])`
: (Akıcı) Belirli URI'a PUT isteği yap.

`patch($uri, array $data = [], array $headers = [])`
: (Akıcı) Belirli URI'a PATCH isteği yap.

`delete($uri, array $data = [], array $headers = [])`
: (Akıcı) Belirli URI'a DELETE isteği yap.

`followRedirects()`
: (Akıcı) Son talepten gelen yönlendirmeleri takip et.

`see($text, $negate = false)`
: (Akıcı) Sayfa üzerinde görünen (veya görünmeyen) metni işle.

`seeJson(array $data = null)`
: (Akıcı) JSON içeren yanıtı işle. Eğer `$data` geçerse, ayrıca JSON'ın tam olarak eşleştiğini işler.

`seeStatusCode($status)`
: (Akıcı) Yanıtın beklenen durum koduna sahip olduğunu işle.

`seePageIs($uri)`
: (Akıcı) Şu anki sayfanın verilen URL ile eşleştiğini işle.

`seeOnPage($uri)` and `landOn($uri)`
(Akıcı) `seePageIs()` için takma ad.

`click($name)`
(Akıcı) Verilen body, name veya id'deki bağlantıya tıkla.

`type($text, $element)`
(Akıcı) Verilen metin ile bir girdi alanını doldur.

`check($element)`
(Akıcı) Sayfa üzerindeki bir onay kutusunu işaretle.

`select($option, $element)`
(Akıcı) Açılır listeden bir seçenek seç.

`attach($absolutePath, $element)`
(Akıcı) Form alanına bir dosya ekle.

`press($buttonText)`
(Akıcı) Verilen metne sahip bir buton ile formu gönder.

`withoutMiddleware()`
(Akıcı) Test için middleware'ı devredışı bırak.

`dump()`
En son yanıtın içeriğini dök.

Laravel 5.1 PHPUnit Uygulama metodları ve özellikleri

Laravel 5.1'in PHPUnit için sağladığı uygulama metodları ve özelliklerinin kısa özeti.

`$app` Laravel 5.1 uygulamasının örneği (instance).

`$code`
Artisan'dan tarafından döndürülen son kod.

`refreshApplication()`
Uygulamayı yenile. TestCase'in `setup()` metodu ile otomatik çağrılır.

`call($method, $uri, $parameters = [], $cookies = [], $files = [], $server = [], $content = null)`
Verilen URI'yı çağırır ve yanıtı döndürür.

`callSecure($method, $uri, $parameters = [], $cookies = [], $files = [], $server = [], $content = null)`
Verilen HTTPS URI'yı çağırır ve yanıtı döndürür.

`action($method, $action, $wildcards = [], $parameters = [], $cookies = [], $files = [], $server = [], $content = null)`
: Controller eylemini çağırır ve yanıtı döndürür.

`route($method, $name, $routeParameters = [], $parameters = [], $cookies = [], $files = [], $server = [], $content = null)`
: İsimli rotayı çağırır ve yanıtı döndürür.

`instance($abstract, $object)`
: Container içinde nesnenin bir örneğini (instance) kaydet.

`expectsEvents($events)`
: Verilen eylem için ateşlenecek olaylar listesi tanımla.

`withoutEvents()`
: Olay dağıtıcısını taklit et, böylece tüm olaylar susturulur.

`expectsJobs($jobs)`
: Verilen eylem için dağıtılacak görevler listesi tanımla.

`withSession(array $data)`
: Verilen diziye göre oturumu ayarla.

`session(array $data)`
: Oturumu başlatır ve dizideki oturum değerlerini ayarlar.

`flushSession()`
: Mevcut oturum içeriğini temizler.

`startSession()`
: Uygulamanın oturumunu başlatır.

`actingAs($user)`
: (Akıcı) Uygulama için sisteme giriş yapmış kullanıcıyı ayarlar.

`be($user)`
: Uygulama için sisteme giriş yapmış kullanıcıyı ayarlar.

`seeInDatabase($table, array $data, $connection = null)`
: (Akıcı) Verilen where koşulunun veritabanında var olduğunu işle.

`notSeeInDatabase($table, $array $data, $connection = null)`
: (Akıcı) Verilen where koşulunun veritabanında var olmadığını işle.

`missingFromDatabase($table, array $data, $connection = null)`
: (Akıcı) `notSeeInDatabase()` için takma ad.

`seed()`
: Veritabanına ekim yap.

`artisan($command, $parameters = [])`
: Artisan komutunu çalıştırır ve kodu döndürür.

Bu metodlardan veya özelliklerden herhangi birine test sınıfı içinden erişilebilir. Verilen `ExampleTest.php` dosyası içindeki `testBasicExample()` metodu `$this->call(...)` satırını içerir.

Laravel 5.1 PHPUnit İşleçleri (Assertions)

Laravel 5.1 web uygulamalarının testlerinin yazılması ile ilgilenmesi için PHPUnit'in standard işleçlerine (örneğin `assertEquals()`, `assertContains()`, `assertInstanceOf()`, ...) ilaveten ek işleçler sunar.

`assertPageLoaded($uri, $message = null)`
: Son sayfanın yüklendiğini işle; yüklenmezse $uri/$message ile istisna fırlat (throw exception).

`assertResponseOk()`
: İstemci yanıtının OK durum koduna sahip olduğunu işle.

`assertReponseStatus($code)`
: İstemci yanıtının verilen durum koduna sahip olduğunu işle.

`assertViewHas($key, $value = null)`
: Yanıt görünüm dosyasının, verilen bağlı veri parçasına sahip olduğunu işle.

`assertViewHasAll($bindings)`
: Görünüm dosyasının, verilen veri parçası listesine sahip olduğunu işle.

`assertViewMissing($key)`
: Görünüm dosyasının, verilen veri parçası listesine sahip olmadığını işle.

`assertRedirectedTo($uri, $with = [])`
: İstemcinin verilen URI'ya yönlendirildiğini işle.

`assertRedirectedToRoute($name, $parameters = [], $with = [])`
: İstemcinin verilen rotaya yönlendirildiğini işle.

`assertRedirectedToAction($name, $parameters = [], $with = [])`
: İstemcinin verilen eyleme yönlendirildiğini işle.

`assertSessionHas($key, $value = null)`
: Oturumun verilen anahtar(lar)/veri(ler)'e sahip olduğunu işle.

`assertSessionHasAll($bindings)`
: Oturumun verilen veriler listesine sahip olduğunu işle.

`assertSessionHasErrors($bindings = [])`
: Oturumun hata parçacıklarına sahip olduğunu işle.

`assertHasOldInput()`
: Oturumun eski girdi verisine sahip olduğunu işle.

TDD için Gulp Kullanma

Gulp[19] JavaScript ile yazılmış inşa ve otomatikleştirme sistemidir. Kaynak dosyalarının minimizasyonu gibi ortak görevlerin otomatikleştirilmesine olanak sağlar. Gulp aynı zamanda kaynak kodlarınızdaki değişiğimi izleyerek değişiklik esnasında bu görevlerin otomatik çalıştırılması ile de ilgilenir.

Laravel 5.1 Gulp görevlerinin kolay yoldan inşa edilmesi için Laravel Elixir[20] aracını içerir. Elixir Gulp'a enfes bir sözdizimi ekler. PHP için Laravel neyse Gulp için de Elixir odur.

Gulp'ın en genel kullanım alanlarından biri ünite testleridir. Burada TDD kullanarak Gulp'ın testlerimizi otomatik çalıştırmasını sağlayacağız.

Öncelikle `l5beauty` proje dizinindeki `gulpfile.js` dosyasını düzenleyin ve içeriğini inceleyelim.

Gulp'ın PHPUnit Testlerini Çalıştırmasını Sağlamak

```
var elixir = require('laravel-elixir');

elixir(function(mix) {
    mix.phpUnit();
});
```

Burada `elixir()` metodunu çağırıyor ve buna bir fonksiyon geçiyoruz. Buradaki fonksiyonun geçildiği `mix` nesnesi akış üstlenir ve birçok şey gerçekleşebilir. LESS dosyalarının CSS dosyalarına dönüştürülmesi, CSS dosyalarının bir araya getirilip birleştirilmesi ve çıktı dosyalarının sürümlendirilmiş olmasını sağlar. Tüm bu görevler `mix` nesnesi üzerinde akıcı bir arayüzde tanımlanabilir.

Biz şimdilik PHPUnit testleri çalıştıracağız.

Sonra, ana bilgisayarınızda, projenizin ana dizininde gulp'ı çalıştırıp neler gerçekleştiğini görün.

[19] http://gulpjs.com/
[20] http://laravel.com/docs/5.1/elixir

Gulp'ı çalıştırmak

```
~% cd Code/l5beauty
~/Code/l5beauty% gulp
[15:26:23] Using gulpfile ~/Code/l5beauty/gulpfile.js
[15:26:23] Starting 'default'...
[15:26:23] Starting 'phpunit'...
[15:26:25] Finished 'default' after 2.15 s
[15:26:25]

       *** Debug Cmd: ./vendor/bin/phpunit --colors --debug ***

[15:26:28] PHPUnit 4.7.4 by Sebastian Bergmann and contributors.

Configuration read from /Users/sineld/Code/l5beauty/phpunit.xml

Starting test 'ExampleTest::testBasicExample'.
.

Time: 2.07 seconds, Memory: 10.25Mb

OK (1 test, 2 assertions)
[15:26:28] gulp-notify: [Green!]
[15:26:28] Finished 'phpunit' after 4.96 s
```

Ana bilgisayarınızda bir bildirim, popup veya benzeri bir uyarı almış olmalısınız. Bildirim renginin yeşil olması herşeyin yolunda gittiğine, testlerinizin başarılı olduğuna işaret eder.

PHPUnit Başarılı

Gulp'ın ünite testlerinizi otomatik modda yapması için ana bilgisayarınızda `gulp tdd` komutunu çalıştırın.

Gulp'ı çalıştırma

```
~% cd Code/l5beauty
~/Code/l5beauty% gulp tdd
[15:29:49] Using gulpfile ~/Code/l5beauty/gulpfile.js
[15:29:49] Starting 'tdd'...
[15:29:49] Finished 'tdd' after 21 ms
```

Bu komut burada *asılı* kalarak kaynak kodlarınızdaki değişikliği izleyecek ve gerektiğinde ünite testlerinizi çalıştıracaktır.

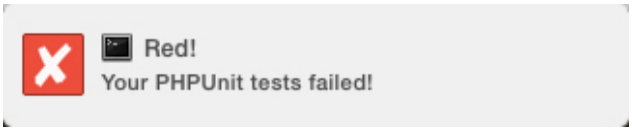
Nasıl çalıştığını görmek etmek için mevcut ünite testimizi bozalım.

`tests/ExampleTest.php` içindeki `see()` bölümünü aşağıdaki gibi değiştirin.

ExampleTest.php dosyasının bozulması

```
1   ->see('Laravel 5x');
```

Dosyayı kaydettiğiniz zaman Gulp bunu farkedecek ve PHPUnit'i yeniden çalıştıracaktır. Test başarısız olacak ve siz aşağıdakine benzer bir uyarı ile karşılaşacaksınız.



PHPUnit Başarısız

Change the line back to what it was before, save it, and again gulp will run PHPUnit. This time you should receive a notice indicating you are *"back to green"*.

Gulp'ın *tdd* modundan çıkmak için

`Ctrl+C`

Markdown Servisinin Oluşturulması

Oluşturacağımız blog uygulaması sayfaların Markdown dosyalarının düzenlenmesine izin verecektir. Markdown HTML'e dönüştürülen okuması kolay *ve* yazması kolay bir biçimdir.

Testi göstermek için TDD kullanarak, markdown metinleri HTML metinlere dönüştürecek bir servis inşa edeceğiz.

Markdown Paketlerinin Çekilmesi

Markdown'ın HTML dönüştürülmesi için etrafta birçok PHP paketi mevcut. `http://packagist.org` adresine gider ve *markdown* kelimesini aratırsanız yirmi sayfadan fazla sonuç üretecektir.

Michel Fortin tarafından oluşturulmuş paketi kullanacağız çünkü ona ait `SmartyPants` adında tırnak işaretlerini şık görünümlü küme alıntılara dönüştüren bir paketi de var.

Ana bilgisayarınızın konsolunda aşağıdaki işlemi yapıp paketi çekiniz.

Markdown ve SmartyPants Eklenmesi

```
~/Code/l5beauty% composer require michelf/php-markdown
Using version ^1.5 for michelf/php-markdown
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing michelf/php-markdown (1.5.0)
    Downloading: 100%

Writing lock file
Generating autoload files
Generating optimized class loader

~/Code/l5beauty% composer require "michelf/php-smartypants =1.6.0-beta1"
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing michelf/php-smartypants (1.6.0-beta1)
    Loading from cache

Writing lock file
Generating autoload files
Generating optimized class loader
```

SmartyPants'ı projeye dahil ederken belirtli bir sürüm kullandığımız dikkatinizi çekti mi? Bunun sebebi ben bu yazıyı yazarken paketin halen kararlı sürümü çıkmadığından stabil paketin otomatik olarak çekilmesinin mümkün olmaması.

Markdown Test Sınıfı Oluşturulması

TDD oturumunu başlatırken ilk yapmamız gereken Gulp'ı TDD modda çalıştırmaktır.

Gulp'ın TDD modda çalıştırılması

```
~/Code/l5beauty% gulp tdd
[19:41:38] Using gulpfile ~/Code/l5beauty/gulpfile.js
[19:41:38] Starting 'tdd'...
[19:41:38] Finished 'tdd' after 23 ms
```

Artık Gulp değişiklikleri izliyor ve farkeder etmek PHPUnit'i çalıştıracaktır. Hadi test sınıfımızı yazalım.

`tests` dizini içinde `Services` adında bir dizin ve bunun içinde `MarkdownerTest.php` adında bir dosya oluşturun.

İlk Test tests/Services/MarkdownerTest.php

```
1  <?php
2
3  class MarkdownerTest extends TestCase
4  {
5
6    protected $markdown;
7
8    public function setup()
9    {
10     $this->markdown = new \App\Services\Markdowner();
11   }
12
13   public function testSimpleParagraph()
14   {
15     $this->assertEquals(
16       "<p>test</p>\n",
17       $this->markdown->toHTML('test')
18     );
19   }
20 }
```

Satır 6
: Markdown nesnesinin bir örneğini (instance) saklayın

Satır 8
: `Markdowner` örneğini oluşturması için `setup()` metoduna sahip olun. (Evet, henüz yok.)

Satır 13
: Çalışacağından emin olduğumuz basit bir test.

Hata bildirimi almış olmalısınız. *(Ctrl+C yaparak Gulp'ı kapatmadı ve yeniden başlatmadıysanız.)*

Her ne kadar bildirim testin başarısız olduğunu gösterse de bazen konsola bakıp neler olup bittiğini bakıp hatayı irdemelek yararlı olabilir. Bu durumda hata çok net. `App\Services\Markdowner` nesnesi mevcut değil.

Markdowner Servisi Oluşturulması

Burada yapacağımız daha önceden içer aktardığımız php-markdown ve php-smartypants paketlerini kapsayacak basit bir servis oluşturmaktır.

`app\Services` dizini içinde `Markdowner.php` adında bir dosya oluşturup aşağıdaki içeriği ilave edin.

app/Services/Markdowner dosya içeriği

```
<?php

namespace App\Services;

use Michelf\MarkdownExtra;
use Michelf\SmartyPants;

class Markdowner
{

  public function toHTML($text)
  {
    $text = $this->preTransformText($text);
    $text = MarkdownExtra::defaultTransform($text);
    $text = SmartyPants::defaultTransform($text);
    $text = $this->postTransformText($text);
    return $text;
  }

  protected function preTransformText($text)
  {
    return $text;
  }

  protected function postTransformText($text)
  {
    return $text;
  }
}
```

Satır 3
Aduzayını (namespace) unutma

Satırs 5 and 6
Kullanacağımız sınıflar.

Satır 11
Metni dönüşümlerden geçiren toHTML() metodu.

Satır 14
Kütüphanenin Markdown Extra sürümünü kullandığımıza dikkat edin.

Satır 20
Bu durumda herşeyden önce kendi dönüşümlerimizi sonda yapmak istiyoruz.

Satır 25
`preTransformText()` gibi ancak bu sefer eğer kendi son dönüşümlerimizi ilave etmek istersek.

Dosyayı kaydettiğinizde Gulp size herşeyin yolunda gittiğini gösterir anlamda bir "YEŞİL" bildirim gösterecektir.

Eğer yeşil bildirim almazsanız geri gidip `App\Services\Markdowner` ve `MarkdownerTest` sınıfları içindeki yazım yanlışlarını düzeltin.

Birkaç Test Daha

Kuşkusuz, bu TDD için büyük bir örnek değil çünkü içinde basit bir sınıf mevcut ve hatayı gidermek için tam bir sınıf dosyası oluşturuldu. Gerçekte TDD daha fazla tekrarlama içermeli ve aşağıdakine benzer bir akışa sahip olmalıdır:

- MarkdownerTest Oluştur w/ testSimpleParagraph()
- Testler Başarısız
- Markdowner sınıfı Oluştur, testi geçmek için toHTML() metodunu sert kodla (hard-code).
- Testler Başarılı
- Markdowner sınıfını MarkdownExtra ile değiştir
- Testler Başarılı
- MarkdownerTest sınıfına testQuotes() ekle
- Testler Başarısız
- Markdowner sınıfını SmartyPants kullanacak şekilde güncelle
- Testler Başarılı

Ve benzeri. Söz konusu test olunca bizim `Markdowner` sınıfımızın bile yapısı kusurlu. Bu sınıf üzerinde *temiz* ünite testleri yapabilmek için `MarkdownExtra` ve `SmartyPants` sınıflarının *construct* sınıfına enjekte edilerek örneklendirilmesi gerekir. Bu şekilde ünite testleri taklit nesnelerini enjekte eder ve yalnızca `MarkdownExtra` sınıfının davranışını doğrular, onun çağırdığı alt sınıfları değil.

Neyseki bu kitap test üzerine değil. Aslında bu testlerimizi yazacağımız tek bölümdür.

Şimdilik yapıyı aynı bırakacak ve birkaç test daha ekleyeceğiz.

`MarkdownerTest` aşağıdaki gibi güncelleyin.

Final Contents of app/Services/Markdowner.

```
<?php

class MarkdownerTest extends TestCase
{

  protected $markdown;

  public function setup()
  {
    $this->markdown = new \App\Services\Markdowner();
  }

  /**
   * @dataProvider conversionsProvider
   */
  public function testConversions($value, $expected)
  {
    $this->assertEquals($expected, $this->markdown->toHTML($value));
  }

  public function conversionsProvider()
  {
    return [
      ["test", "<p>test</p>\n"],
      ["# title", "<h1>title</h1>\n"],
      ["Tuana'ya merhaba!", "<p>Tuana&#8217;ya merhaba!</p>\n"],
    ];
  }
}
```

Burada test sınıfımızı birçok konuşmayı aynı anda test edecek biçimde değiştirdik ve `conversionsProvider()` içine üç test ekledik. İlerlemeden önce testleriniz yeşil olmalı.

Testleriniz yeşil ise ana bilgisayarınızda, konsolda `Ctrl+C` tuşlarıyla Gulp'ı durdurun.

Diğer Test Yöntemleri

Laravel 5.1 ile test yöntemlerinin tümünün bir listesini sağlamak niyetinde değilim çünkü PHP'de testler yalnızca bir şekilde yapılmaz. Aynı şekilde, Laravel 5'te de testler tek şekilde yapılmaz.

Ama biz bazı alternatifleri ele alacağız.

phpspec

PHPUnit dışında, Laravel 5.1 ayrıca phpspec[21]'i de doğrudan destekler. Bu daha çok Davranış Güdümlü Geliştirme için bir diğer popüler PHP test aracıdır.

Phpspec üzerine birtakım notlar.

- Uygulama dosyası `vendor/bin` içindedir, bu sebeple `phpspec` komutunu projenizin ana dizininde çalıştırabilirsiniz.
- Yapılandırma dosyası projenin ana dizinindedir ve ismi `phpspec.yml`'dir.
- Phpspec'i Gulp'ten çalıştırabilmek için, Laravel Elixir `mix` nesnesi içinde kullanabileceğiniz `phpSpec()` metodunu sağlar.
- Eğer uygulamanızın aduzayını `App`'den farklı bir isme değiştirirseniz `phpspec.yml` dosyasını uygun şekilde güncellemeyi unutmayın.

Ünite Testleri

Her ne kadar PHP ile ünite testi yaparken PHPUnit standart olsa da kullanabileceğiniz başka paketler de mevcuttur.

- Enhance PHP[22] - Taklit (mock) ve koçan (stub) desteğine sahip ünite testi frameworkü.
- SimpleTest[23] - Bir diğer taklit nesneleri destekli test frameworkü.

Bütünleşme ve Kabul Testleri

Bu testler aslında uygulamanızın bir birimini test etmek yerine uygulamanızı kullanarak beklendiğini gibi çalışıp çalışmadığını doğrular. Laravel 5.1'in sağladığı akıcı test metodlarını kullanırken PHPUnit ile bazı bütünleşme testleri de yapabilirsiniz. `ExampleTest.php` basit bir örnek sunar ama bütünleşme ve kabul testleri üzerine yoğunlaşan farklı test frameworkleri de mevcuttur.

[21] http://phpspec.net
[22] https://github.com/Enhance-PHP/Enhance-PHP
[23] http://simpletest.org

- Codeception[24] - Kabul testleri üzerine en popüler framework.
- Selenium[25] - Tarayıcı otomatikleştirme.
- Mink[26] - Tarayıcı otomatikleştirme.

Davranış Güdümlü Geliştirme

BDD (Davranış Güdümlü Geliştirme) iki şekilde karşımıza çıkar: SpecBDD ve StoryBDD.

SpecBDD kodunuzun teknik yönleri üstüne yoğunlaşır. Laravel 5.1 SpecBDD için standart olan *phpspec*'i içerir.

StoryBDD işlev veya nitelik testlerine odaklanır. Behat en popüler StoryBDD frameworküdür. Buna rağmen, StoryBDD için Codeception da kullanılabilir.

Özet

Bu bölümde ilk olarak **l5beauty** adında bir proje oluşturduk. Sonra bu projeye ünite testlerinden PHPUnit testleri uyguladık. Son olarak `Markdowner` servis sınıfını hem test etmek hem de daha sonra markdown metinleri HTML'e dönüştürmek için oluşturduk.

Bu oldukça uzun bir bölüm oldu çünkü test etme başlı başına büyük bir konudur ve hakkını tek bölüm ile teslim etmek zordur ama daha önce de belirttiğim gibi testler yazmak bu kitabın asıl konusu değildir. Alt bölümlerde başka testler yazmayacağız.

Hızlı birşeylere ne dersiniz? Sonraki bölümde 10 dakikada bir blog oluşturacağız.

[24] http://codeception.com
[25] http://seleniumhq.org
[26] http://mink.behat.org

Önizleme Sonu

Her ne kadar önizleme sürümü çok geniş olsa dahi, daha sizi bu kitapta bekleyen çok şey var.

Bölüm 7 - 10 Dakidalık Blog

Bu bölümde, testlerini de tamamlayarak L5beauty projesini bloğa dönüşüreceğiz. Laravel 5'in gücüyle 10 dakikadan az bir sürede blog oluşturulabilir. Bu kez, aşağıda ayrıntılarla vakit kaybetmeden baştan sona bitireceğiz. Burada çok fazla düdük öttürmeden, ıslık çalmadan ve yönetim paneline değinmeden bloğumuzu oluşturacağız.

Bölüm 8 - Yönetim Paneline Başlama

Bu bölümde l5beauty projemizi inşasını, yönetim panelini geliştirmeye başlayarak, sürdüreceğiz. Laravel 5 yetkilendirme ve kayıt işlemleri için birtakım araçlar sunar. Yönetim panelimizin omurgasını oluşturmak için bunları kullanacağız.

Bölüm 9 - Bower Kullanma

Bu bölümde yönetim panelimizin üstüne kurulacağı birtakım destek yazılımları üzerinden çalışacağız. Yani, hangi varlıkların (assets) çekileceği ve kullanılacağına değineceğiz. İnşa edilen sistem Bower ve Gulp kullanarak otomatik olarak jQuery, Bootstrap, Font Awesome, ve DataTables'ı internetten indirecek ve birleştirecek.

Bölüm 10 - Blog Etiketleri

10 Dakidalık Blog bölümünde inşa edilen temel blog çok süslü değildi. Birçok blog platformu blog yazılarının kategorize edilmesine veya "etiketlenmesine" birçok yönden olanak sağlar. Bu bölümde l5beauty projemiz için etiketleme sistemi geliştireceğiz.

Bölüm 11 - Yükleme Yöneticisi

Bu bölümde blog yönetimi için Yükleme Yöneticisi oluşturacağız. Öncelikle, dosyaların yüklenmesi için yerel dosya sistemini kullanacağız. Sonra yapılandırmamızı, dosyaların Amazon S3 Bulut Depolama'da saklanması için değiştireceğiz.

Bölüm 12 - Yazı Yönetimi

Bu bölümde yönetim panelinin yazı işlevselliğini tamamlayacağız. Burada yeni migrasyon ile `posts` tablosunun yapısını değitirecek, ilave assets'leri çekecek ve temel Create (Oluştur), Update (Güncelle) ve Delete (Sil) metodlarını ekleyeceğiz.

Bölüm 13 - Bloğun Temizlenmesi

Bu bölümde bloğumuzun özyüzünü temizleyeceğiz. Buna yazılar listesinin görüntülendiği fihrist (index) ve belirli bir yazının gösterilmesi de dahildir.

Bölüm 14 - E-posta Gönderme ve Kuyruk Kullanma

Bu bölümde bloğumuza *Bize Ulaşın* formu ekleyeceğiz. Bunu yaparken Laravel'in posta işlevlerini irdeleyecek ve eşzamansız işleme için kuyruk yapılandıracağız.

Bölüm 15 - Yorumlar, RSS ve Site Haritası Ekleme

Bu bölümde bloğumuza yorumlar ve sosyal ağlar için bağlantılar ekleyeceğiz. Ardından Laravel 5.1 Beauty bloğumuz için RSS beslemesi oluşturacağız. Son olarak, projeyi bitiren Site Haritası ekleyeceğiz.

Bölüm 16 - Genel Özet ve Geleceğe Bakış

L5Beauty projesi tamamlandı ancak Laravel 5.1'in çeşitli özelliklerini incelemek için ilave bir bölüm daha var.