PHP

Usulüne Uygun

SON GÜNCELLEME: 2019-03-27 07:44:26 +0000

SHARE ON TWITTER

Kodlama Uygulamaları

Temelleri

Tarih ve Saat

Tasarım Desenleri (Design Patterns)

UTF-8 ile Çalışmak

Bağımlılık Değiştirme

Temel Kavramlar

Complex Problem

Containers

Further Reading

Veritabanları

Veritabanları MySQL

Veritabanları PDO

Interacting with Databases

Soyutlama Katmanı

Arayüz Kalıpları (Templating)

Templating (Şablon)

Yalın(Düz) PHP Şablonları

Derlenen Şablonlar (Compiled Template)

Ekstra Kaynaklar

Hatalar ve İstisnalar

Hatalar

İstisnalar (Exceptions)

Güvenlik

Web Uygulama Güvenliği

Şifre Karıştırma (hashing)

Veri Süzme

Konfigürasyon Dosyaları

Register Globals

Hata Raporları

Test Etme

Test Odaklı Geliştirme

Davranış Odaklı Geliştirme

Tamamlayıcı Test Araçları

Sunucular ve Dağıtım

Platform as a Service (PaaS)

Sanal veya Özel Sunucular

Paylaşımlı Sunucu

Uygulamanızı Build Etmek ve Deploy Etmek

Sanallaştırma

Vagrant (Sanal Sunucu)

Docker

Caching

Bytecode Cache (Önbelleği)

Nesnel Önbellekleme

Documenting

Phpdoc

Kaynaklar

Çatılar (Frameworks)

Bileşenler (Components)

Books

Topluluk

Hazırlayanlar

CHAPTER 1.

Hoşgeldiniz

İnternet üzerinde PHP ile ilgili geçerliliğini yitirmiş çok fazla bilgi bulunmakta. Bu da PHP ile yeni tanışanların günümüzde kullanılmayan, benimsenmeyen yaklaşımları ve deneyimleri izlemesine, kötü yazımış kodların üretilmesine neden olmakta. Buna artık

bir son verilmeli. *PHP: Usulüne Uygun (PHP: The Right Way)*, PHP hakkında kolay anlaşılır, güncel ve pratik bilgiler, kabul görmüş kodlama standartları hakkında bilgiler ve internet üzerinde güvenilir eğitici web sitelerine bağlantılar içermektedir.

There's a lot of outdated information on the Web that leads new PHP users astray, propagating bad practices and insecure code. *PHP: The Right Way* is an easy-to-read, quick reference for PHP popular coding standards, links to authoritative tutorials around the Web and what the contributors consider to be best practices at the present time.

There is no canonical way to use PHP. This website aims to introduce new PHP developers to some topics which they may not discover until it is too late, and aims to give seasoned pros some fresh ideas on those topics they've been doing for years without ever reconsidering. This website will also not tell you which tools to use, but instead offer suggestions for multiple options, when possible explaining the differences in approach and use-case.

This is a living document and will continue to be updated with more helpful information and examples as they become available.

Çeviri

PHP: Usulüne Uygun birçok dilde çevirisi bulunmaktadır:

- English
- Deutsch
- Español
- Français
- Indonesia
- Italiano
- Polski
- Português do Brasil
- Română
- Slovenščina
- Srpski
- Türkçe
- български
- Русский язык
- Українська
- العربية
- فارسی
- ภาษาไทย
- 한국어판
- 日本語
- 简体中文

- 繁體中文

Feragatname

PHP'yi kullanmanın kuralsal, kesin bir yolu yoktur. Fakat bu site uygun özellikleri ve yararlı bilgileri içeren en iyi kullanım yöntemini göstermeye çalışmaktadır. Bu site yeni kullanıcılar için bir başlangıç ve tecrübeli kişiler için fikirlerini tazeleme ortamıdır.

Bu site, daha yararlı olmak için devamlı güncellenecek bir belgedir.

Nasıl Katkıda Bulunurum

Bu sitenin yeni kullanıcılar için yararlı olmasına yardımcı olun! GitHub Üzerinden Katkıda Bulunun

Afişler

PHP: Usulüne Uygun'un sitenize yerleştirmek için afişleri var. Bunları sitenize yerleştirerek desteğinizi gösterin, ve yeni kullanıcıların doğru bilgiye ulaşmasını sağlayın.

Afişler

Back to Top

CHAPTER 2.

Başlarken

Güncel ve Tutarlı Sürümü Kullanın (7.0)

Eğer PHP'ye başlamak istiyorsanız, PHP 7.0 ile başlayın. PHP 7.0 yeni ve PHP 5.x üzerine, bir çok yeni özelliğe sahip. Ana kodunda bir çok kısım yeniden yazıldı ve önceki sürümlerinden daha hızlı.

Yakın zamanlarda 5.x sürümleri mevcut ve 5.6 sürümü en son 5.x sürümü. Hangi fonksiyonunun hangi versiyonda olup olmadığını kontrol etmek için php.net sitesindeki dökümantasyonu kontrol edebilirsiniz.

Dahili Web Sunucusu

PHP 5.4 sürümü veya daha yeni sürümleri ile birlikte PHP kullanmaya herhangi bir web sunucusu kurmadan ve ayarlama yapmadan başlayabilirsiniz. Dahili web sunucusunu

başlatmak için aşağıdaki komutu, projenize ait dosyaların bulunduğu dizin içinde komut satırından çalıştırabilirsiniz:

```
> php -S localhost:8000
```

- Dahili web sunucusu ve komut satırı hakkında belgeler

Mac (OSX) Kurulumu

OSX ile birlikte PHP zaten kurulu halde gelmektedir fakat kurulu PHP sürümü güncel ve kararlı sürümden biraz eski olabilir. Örneğin, Lion PHP 5.3.6 versiyonu ile, Mountain Lion ise 5.3.10 versiyonu ile, Mavericks 5.4.17 ile, Yosemite 5.5.9 ile ve El Capitan 5.5.29 ile gelmektedir. Hali hazırda PHP 7.0 dahili olarak gelmemektedir.

PHP'yi OS X'de kurmak için bir kaç yöntem bulunmaktadır.

Homebrew ile Kurulum

Homebrew OS X için güçlü bir paket yöneticisidir ve PHP ve eklentilerinin kurulumunu kolayca yapabilmenizi sağlar. Homebrew PHP deposu, PHP kurulumuna izin verecek, PHP ile ilgili "formulae"leri barındıran bir depodur.

Bu noktada, siz `php53`, `php54`, `php55`, `php56` veya `php70` sürümlerini `brew install` komutu ile rahatça kurablirsiniz ve bu sürümler arasında `PATH` değişkenini değiştirerek dolaşabilirsiniz. Alternatif olarak brew-php-switcher sizin için bunu otomatik olarak yapacaktır.

Macports ile Kurulum

MacPorts projesi OS X üzerinde komut satırından çalışan, X11 üzerinde çalışan veya Aqua tabanlı açık kaynak yazılımları bile compile edip, kurup güncelleyebileceğiniz kolay kurulum için tasarlanmış bir açık kaynak projesidir.

MacPorts pre-compiled binary dosyalar sunmaktadır. Bu yüzden kaynak kodundan bütün gereksinimlerle birlikte tekrar kurmanız gerekmez, Bu sizin hayatınızı kolaylaştırır ve ekstra kurulumlar ile uğraştırmaz.

Bu noktada, `php54`, `php55`, `php56` veya `php70` sürümlerini `port install` komutu ile kolayca kurabilirsiniz.

```
sudo port install php56
sudo port install php70
```

Ve kolayca `select` komutu ile kullanmak istediğiniz sürümü aktif edebilirsiniz.

```
sudo port select --set php php70
```

phpbrew ile Kurulum

phpbrew çoklu PHP sürümlerini yönetmek için bir araçtır. Farklı versiyonlara ihtiyaç duyduğunuz uygulama/projeler için gerçekten çok yararlı olabilir ve siz sanal makine kullanmazsınız.

Liip's binary installer ile Kurulum

php-osx.liip.ch 5.3 ile 7.0 sürümleri arasında her sürüm için bir satırlık kurulum scriptleri sunan popüler bir seçenektir. Apple tarafından kurulan PHP'nin üzerine yazmak ama her bir sürüm için farklı klasörler oluşturur (/usr/local/php5).

Kaynaktan Kurmak

Diğer bir seçenek ise kaynak kodunu compile ederek kurmakdır. Bu durumda Xcode'un veya "Command Line Tools for XCode" kurulmuş olması gerekmektedir.

Hepsi Bir Arada Kurulum

Yukarıdaki listelenenler PHP'nin kendisinin kurulumuyde ve Apache, Nginx veya SQL Server gibi şeyleri desteklemiyordu. MAMP ve XAMPP gibi "Hepsi Bir Arada" çözümünde ekstra olarak bir kaç yazılım daha kurulmaktadır. Ancak kolay kurulum desteği ile gelmektedir.

Windows Kurulumu

windows.php.net/download adresinden binary indirebilirsiniz. PHP'yi çıkardıktan sonra PHP'yi heryerden çalıştırabilmek için bir sizin PHP klasörünüze (php.exe nerede ise) PATH belirlemenizi önerir.

Öğrenmek ve yerel geliştirme için hali hazırda gelen PHP 5.4+ sürümü kullanabilirsiniz ve konfigürasyonu düşünmeniz gerekmez. Eğer hepsi bir arada bir kurulum (MySQL ve bir çok ekstra araç içeren) istiyorsanız, Web Platform Installer, XAMPP, EasyPHP, OpenServer ve WAMP size Windows platformunda hızlıca bir kurulum için yardımcı olacaktır. Bu araçlar production ortamına göre biraz farklılıklar içerecektir. Eğer uygulamanızı Linux ortamında yayınlarken Windows ortamında geliştirme yapıyorsanız biraz dikkat etmeniz gerekebilir.

Eğer uygulamanızı Windows ortamında yayınlamak istiyorsanız IIS7 size daha stabil ve en iyi performansı sunacaktır. Phpmanager (IIS7 için bir arayüz eklentisi) size PHP konfigürasyonları ve yönetimi için kolaylık sağlayacaktır. IIS7 FastCGI ile gelmektedir sadece PHP'yi handler olarak ayarlamanız yeterlidir. Ekstra kaynaklar için Dedicated area on iis.net adresine bakabilirsiniz.

Genellikle uygulamalar farklı ortamlarda çalıştırıldığı için uygulama yayınlandığında farklı hatalar ile karşılaşılabilir. Eğer Windows ortamında geliştirirken projenizi Linux (ya da Windows olmayan herhangi bir yerde) ortamında yayınlıyorsanız, Virtual Machine kullanmayı düşünmelisiniz.

Chris Tankersley'in Windows ortamında kullanılacak araçlar ile ilgili çok yararlı blog yazılarıbulunmaktadır. PHP development using Windows.

Back to Top

CHAPTER 3.

Kodlama Stili Rehberi

PHP topluluğu geniş ve çeşitlidir. Sayısız kütüphane, framework ve bileşenlerden oluşmaktadır. Bunlardan birkaçını seçip herhangi bir projede kullanmak PHP geliştiricilerinin sıklıkla kullandığı bir yöntemdir. PHP kodlarında genel bir stile uymak, farklı ve çeşitli kütüphaneleri sorunsuz kullanmak açısından son derece önemlidir. PHP kodunun genel kod stiline olabildiğince uyması, geliştiricilerin kolaylıkla farklı kütüphaneleri kullanabilmesine olanak tanır.

Framework Interop Group (daha önce 'PHP Standartları Grubu' olarak bilinen) çeşitli kodlama standartları önerilerinde bulunmuş ve PSR-0, PSR-1 ve PSR-2 gibi standartları kabul etmiştir. Kabul edilmiş olan bu standartlar Drupal, Zend, Symfony, CakePHP, phpBB, AWS SDK, FuelPHP, Laravel ve Lithium gibi birçok büyük projelerde kullanılmaya başlanmış ve desteklenmiştir. Bu standartları dilerseniz kendi projelerinizde kullanabilirsiniz veya kendi kodlama stiliniz ve standartlarınız ile devam edebilirsiniz.

Diğer geliştiricilerin yazdığınız kodu daha kolay okuyup anlamaları ve üzerinden çalışabilmeleri için bu standartlardan bir veya daha fazlasına uygun kod yazmalısınız. Bunlar PSR'nin standartları ya da PEAR veya Zend'in standartları olabilir. Üçüncü partiler tarafından hazırlanmış kodlarla çalışırken birçok bileşenden oluşan uygulamalar birbiri ile daha uyumlu olabilir.

- PSR-0 hakkında
- PSR-1 hakkında
- PSR-2 hakkında
- Read about PSR-4
- "PEAR Coding Standards" hakkında
- "Zend Coding Standards" hakkında
- Read about Symfony Coding Standards

PHP_CodeSniffer eklentisini kullanarak yazdığınız kodların bu standartlardan herhangi birine uyumlu olup olmadığını kontrol edebilirsiniz. Sublime Text 2 gibi bir editörde ise bu konuda gerçek zamanlı geribildirimler alabilirsiniz.

PHP_CodeSniffer aracını bu standartlara karşı uygun olup olmadığını kotrol etmek için kullanabilirsiniz veya Sublime Text 2-3 gibi bir editör'e eklenti olarak kurabilir ve neredeyse gerçek zamanlı geribildirimler alabilirsiniz.

Kod düzeninizi aşağıdaki bir kaç aracı kullanarak kolayce düzeltebilirsiniz. Bunlardan birisi iyi test edilmiş bir kod bulunan PHP Coding Standards Fixer'dir. Diğer bir seçenek ise sublime-phpfmt olarak bir Sublime Text ektentisi olan php.tools'dur.

Ayrıca `phpcs`yi komut satırından da çalıştırabilirsiniz:

```
phpcs -sw --standard=PSR2 file.php
```

Size hataları ve nasıl düzelteceğinize dair açıklamaları gösterecektir. Ayrıca bu komutu git hook olarak da kaydedebilirsiniz. Bu kod deponuza standart olmayan hiç bir kodun girmemesinde yardımcı olacaktır.

İngilizce, kod içinde bulunan değişken, sabit, fonksiyon, yordam gibi semboller ve yapılara ait isimlendirmelerde tercih edilmektedir. Yorumlar için ise şu anda ve gelecekte kodlar üzerinde uğraşacak kişilerin anlayabileceği bir dil tercih edilmelidir. Örneğin, proje ekibinde Türkçe bilmeyen bir eleman varsa, yorum satırlarını Türkçe yerine İngilizce yazmanız daha doğru olacaktır.

Back to Top

CHAPTER 4.

Öne Çıkanlar

Programlama Yaklaşımları

PHP çeşitli programlama tekniklerini destekleyen esnek ve dinamik bir dildir. Özellikle PHP 5.0 ile katı bir nesne tabanlı model eklenmiş (2004), PHP 5.3 de "anonymous functions" ve "namespaces" (2009) desteği gelmiş ve PHP 5.4'de "traits" özelliği eklenerek yıllar içerisinde PHP, nesne tabanlı programlama konusunda kendisini önemli ölçüde geliştirmiştir.

Nesne Tabanlı (Nesneye Dayalı) Programlama

PHP, nesne tabanlı programlamanın "class", "abstract class", "interfaces", "inheritance", "constructors", "cloning", "exception" özellikleri ve daha fazlasını içermektedir.

- Nesne tabanlı programlama hakkında
- Trait'ler hakkında

Fonksiyonel Programlama

PHP, bir fonskiyonun bir değişkene atandığı "first-class" fonksiyonu desteklemektedir. Kullanıcı tanımlı ve dahili fonksiyonların ikiside bir değişkene referans edilebilir ve dinamik olarak çağırılabilir. Fonksiyon bir diğer fonksiyona parametre olarak gönderilebilir (bu özellik "High-order functions olarak bilinir.") veya bir fonksiyondan geri döndürülebilir.

Özyineleme (recursion), kendi kendini çağıran fonksiyon desteklenen bir özelliktir, ama bir çok PHP kodu iterasyona (iteration) odaklanmaktadır.

Yeni anonim fonksiyonlar (closures) PHP 5.3 ile gelmiştir. (2009)

PHP 5.4, closurelerin bir nesnenin scope'u içine bindlenmesi özelliğini getirmiş, ayrıca callable desteğini geliştirmiştir. Artık neredeyse callables ve closures neredeyse tüm durumlarda değişmeli olarak kullanılabilmektedir.

- PHP'de Fonksiyonel Programlama hakkında
- "Anonymous Functions" hakkında
- "Closure class" hakkında
- "Closures RFC" hakkında
- "Callables" hakkında
- `call_user_func_array()` ile dinamik olarak fonksiyon çağırmak hakkında

Meta Programlama

PHP, Reflection API ve Sihirli Yöntemler (Magic Methods) gibi bazı meta programlama mekanizmalarını destekler. PHP'de `__get()`, `__set()`, `__clone()`, `__toString()` ve `__invoke()` gibi Sihirli Yöntemler bulunmaktadır. Bunlar geliştiriciye sınıfların davranışlarını değiştirmelerine izin verirler. Ruby geliştiricileri genellikle PHP'de `method_missing`in (Çağırılan methodun olmaması durumu) eksik olduğunu söylerler. Ancak PHP bunu `__call()` ve `__callStatic()` Sihirli Yöntemlerini kullanarak desteklemektedir.

- Sihirli yöntemler hakkında
- Reflection hakkında
- Overloading hakkında

İsim Uzayları (Namespaces)

Daha öncede bahsedildiği üzere PHP topluluğu bir sürü geliştiriciden oluşmaktadır. Bu nedenle bazı durumlarda bir kaç farklı kütüphane aynı isimde kullanılmış olabilir. İki

kütüphane aynı isim uzayında oldukları zaman çakışırlar ve bu da fatal error gibi sorunlara veya exceptionlara neden olur.

İsim uzayları bu sorunu çözmektedir. PHP referans kılavuzunda da açıklandığı gibi, isim uzayları işletim sistemlerindeki klasörler ile karşılaştırılabilir. Aynı isimdeki iki dosya nasıl farklı klasörlerde bulunabilirsa, aynı şekilde aynı isimdeki iki sınıf farklı İsim Uzayları altında aynı isimde bulunabilmektedir.

Diğer geliştiricilerin geliştirdiği kütüphaneler ile çakışma korkusu yaşamadan geliştirme yapabilmeniz için isim uzaylarını kullanmak sizin yararınıza olacaktır.

PSR-4 İsim Uzayları konusuna değinmiştir ve birbiri ile uyumlu (tak-ve-kullan kod) standart dosyalar, sınıflar ve isim uzayları düzeni kurmayı amaçlamaktadır.

Ekim 2014'de daha önceki autoloader standardı olan PSR-0'ı eski olarak işaretledi, ve PSR-4 bunun yerine yayınlandı. Şu anda ikiside kullanılabilir ve PSR-0 silinmiş değil. PSR-4 PHP 5.3 ve üstü sürümlerini gerektirdiği için daha düşük sürümleri kullanan bir çok PHP projesi halen PSR-0'ı uygulamaktadır veya kullanmaktadır. *Luckily those PHP 5.2-only projects are starting to up their version requirements, meaning PSR-0 is being used less and less.*

Eğer yeni bir proje ya da paket için autoloader standardını kullanmak istiyorsanız PSR-4'e bir gözatmak isteyebilirsiniz.

- İsim Uzayları hakkında
- PSR-0 hakkında
- Read about PSR-4

Standart PHP Kütüphanesi

Standart PHP Kütüphanesi (The Standard PHP Library (SPL)) PHP ile paketlenmiştir ve sınıflar ve arayüzler koleksiyonu sağlar. Temel olarak ihtiyaç duyulan bazı sınıflar ve veri yapılarını içerir (stack, queue, heap, ve dahası), ve bu veri yapıları veya kendi sınıfları üzerinden ulaşılabilen yineleyiciler SPL arayüzünü oluşturur.

- SPL hakkında
- SPL video course on Lynda.com(Paid)

Komut Satırı Arayüzü

PHP web uygulamaları geliştirmek için oluşturuldu, ama komut satırı (CLI) uygulamaları içinde yararlıdır. Komut Satırından çalışan PHP uygulamaları test etme, yayınlama ve uygulama yönetimi vs gibi uygulamaları otomatize etmenize yardımcı olabilir.

CLI PHP uygulamaları güvenli bir web arayüzü oluşturmadan direk olarak çalıştığı için güçlüdür. Sadece komut satırı uygulamanızı direk ana web anadizinine koymayınız!

Aşağıdaki komutları komut satırında çalıştırmayı deneyiniz :

```
> php -i
```

`-i` özelliği PHP'nin konfigürasyonlarını `phpinfo()` fonksiyonu gibi ekrana bastıracaktır.

`-a` özelliği ise ruby'nin IRBsi veya python gibi interaktif bir kabuk sağlayacaktır. command line options'dan daha fazla özelliğe ulaşabilirsiniz.

"Hello, $isim" yazan basit bir CLI uygulaması yazalım. `merhaba.php` adında bir dosya oluşturup bunu çalıştırmayı deneyin.

```
<?php
if ($argc !== 2) {
    echo "Usage: php merhaba.php [isim].\n";
    exit(1);
}
$isim = $argv[1];
echo "Hello, $isim\n";
```

PHP kodunuz çalıştığında argümanlar için iki özel değişken oluşturmaktadır. `$argc` argümanların *sayısını* içereren sayı(integer) tipinde bir değişkendir ve `$argv` her bir argümanın *değerini* içeren değişken dizisidir. İlk argüman her zaman PHP kodunun çalıştırıldığı dosyanın adınıdır, bu örnekte `merhaba.php`dir.

`exit()` kodu kabuğun komutun hata ile sonuçlandığını düşünmemesi için sıfır olmayan bir değişken ile kullanılmalıdır. Genelde kullanılan exit komutları buradadır.

Kodumuzu çalıştırmak için aşağıdaki satırları kullanabilirsiniz :

```
> php merhaba.php
Usage: php merhaba.php [isim]
> php merhaba.php dunya
Merhaba, dunya
```

- PHP komut satırı arayüzü hakkında
- Windows'da PHP komut satırı arayüzü

XDebug

Yazılım gelişticiliğinde en yararlı araçlardan birisi de ayıklayıcıdır (debugger). Bu kodunuzu izlemenize ve yığın içeriklerini görme imkanı sağlar. XDebug, PHP'nin ayıklayıcısı, bazı geliştirme ortamlarında (IDE) yığın denetleme ve kesme noktaları için kullanılır. PHPUnit ve KCacheGrind gibi araçlar ile kodun profilini çıkarma ve analiz etme imkanı verir.

Kendinizi `var_dump/print_r` karmaşasında bulursanız ve halen çözüm bulamıyorsanız, bir ayıklayıcı kullanmalısınız.

Xdebug kurmak aldatıcı olabilir, ama en önemli özelliği "Remote Debugging"dir - eğer siz yerel bir makinede geliştirip VM veya başka bir sunucuda test ediyorsanız, "Remote Debugging" özelliğini etkin hale getirmek isteyebilirsiniz.

Genellikle, Apache Vhost veya .htaccess dosyalarınızı aşağıdaki gibi güncelleyeceksiniz :

```
php_value xdebug.remote_host=192.168.?.?
php_value xdebug.remote_port=9000
```

"remote host" ve "remote port" sizin yerel bilgisayarınızı ve portunu belirtmektedir. Geliştirme ortamınızı bu bilgilere göre konfigüre edin. Geliştirme ortamınızı "listen for connections" moduna getirin ve aşağıdaki bağlantıyı açın :

```
http://your-website.example.com/index.php?XDEBUG_SESSION_START=1
```

Geliştirme ortamınız komut yürütür gibi mevcut durumu yakalar, kesme noktaları eklemenize ve bellekteki değerini araştırmanıza izin verir.

Grafik arayüzlü ayıklayıcılar adım adım ilerlemeyi, değişkenleri seçmeyi ve çalışma zamanında kodu değerlendirmeyi kolaylaştırır. Birçok IDE Xdebug için grafiksel bir arayüzü dahili olarak ya da eklenti olarak barındırır. MacGDBp mac için açık kaynak kodlu bağımsız bir Xdebug uygulamasıdır.

- Xdebug Hakkında
- MacGDBp Hakkında

Back to Top

CHAPTER 5.

Bağımlılık (Dependency) Yönetimi

Kullanabilmeniz için tonlarca PHP kütüphanesi, yapısı(frameworks) ve bileşeni(component) bulunmaktadır. Projeniz bunlardan birkaçını kullanacaktır - bu başlıkta projenizde kullandığınız kütüphanelere bağımlılığınızdan ve bunların yönetilmesinden bahsedeceğiz. Yakın zamana kadar PHP'de bu projeniz içerisinde barındırdığınız diğer kütüphane, çatı ya da bileşeni yönetmek için uygun bir yol yoktu. Elle yönetilse bile, `autoloader` endişesi mevcuttu. Ancak şu anda bu endişeye girmenize bir neden yok.

Şu sıralarda PHP için iki büyük paket yönetim sistemi bulunmaktadır. Composer ve ve PEAR. Composer şu anda PHP için en popüler paket yöneticisi ama PEAR'da uzun zaman ana paket yöneticisiydi. Hiç kullanmasanız bile PEAR hakkında bir kaç referans bulabilirsiniz.

Composer ve Packagist

Composer PHP için **muhteşem** bir bağımlılık yönetimi aracıdır. Bağımlılık yönetimini bir kaç basit komut ile `composer.json` dosyası içerisinde yazıyoruz, Composer projenizin ihtiyaçlarını otomatik olarak indiriyor ve kurulumunu gerçekleştiriyor. Composer Node.js'deki NPM ile veya Ruby'deki Bundler ile benzerdir.

Composer ile uyumlu şimdiden projenizde kullanmanız için bir çok PHP kütüphanesi bulunmakta. Bu paketler Composer'ın resmi deposu olan Packagist'da listelenmetedir.

Composer Nasıl Kurulur

Yerel (önerilmemesine rağmen içinde bulunduğunuzu dizin için) ya da genel (örneğin: /usr/local/bin) kullanım için Composer'ı kurabilirsiniz. Global olarak kurmak için aşağıdaki örneğe bakınız:

```
curl -sS https://getcomposer.org/installer | php
mv composer.phar /usr/local/bin/composer
```

Not: Eğer `mv` ile ilgili bir izin hatası alıyorsanız `sudo` kullanabilirsiniz.

Bu komut `composer.phar` (bir PHP ikili arşivi) dosyası indirecek. Bu dosyayı `php` kullanarak çalıştırabilir ve projenizin ihtiyaçlarını yönetebilirsiniz.

Not: Kodu çevirici bir kaynağa indirdi iseniz öncelikle çevrim içi kod okuyunuz.

Windows Kurulumu

Windows kullanıcıları için ComposerSetup adında bir kurulum dosyası bulunmaktadır. Bu kurulum dosyası `$PATH` dizinine yani global dizine kurulum yapmaktadır. Kullanırken `composer` komutu ile direk çalışabilirsiniz.

Composer Nasıl Kurulur (elle)

Composer'i elle kurmak biraz üst seviye bir tekniktir. Ancak geliştiricinin etkileşimli kurulum yerine bu yöntemi kullanması için bir kaç sebep vardır. Etkilşimli kurulum aşağıdaki PHP kurulumlarını denetler:

- PHP'nin yeterli bir sürümü kontrol edilir
- `.phar` dosyası düzgün çalıştırılır.
- Dizin izinleri kontrol edilir.
- Sorunlu uzantılar yüklenmez.
- Belirli `php.ini` ayarları ayarlanır.

Bu kurulum adımlarının hiçbiri olmadan yapılan elle kurulumda, sizin için bir değerinin olup olmadığına karar vermelisiniz. Aşağıdaki komutlar ile elle kuruluma başlayalım :

```
curl -s https://getcomposer.org/composer.phar -o $HOME/local/bi
chmod +x $HOME/local/bin/composer
```

`$HOME/local/bin` dizini (ya da sizin seçeceğiniz bir dizin) sizin `$PATH` değişkeninizin içerisinde olmalı. Bu `composer` komutunu kullanılabilir yapacaktır.

Dökümantasyonda karşılaştığımız `php composer.phar install` komutunu aşağıdaki gibi kullanabilirsiniz :

```
composer install
```

Bu bölüm compeser'ın global olarak nasıl kurulduğunu varsayar.

Bağımlıklıkları Nasıl Tanımlar ve Kurarız

Composer proje bağımlılığını `composer.json` isimli bir dosya ile sağlar. Bu dosyayı elle ya da Composer vasıtasıyla güncelleyebilirsiniz. `composer.phar require` komutu projenize bağımlılığı ekler ve daha önce oluşturmamış iseniz `composer.json` dosyanızı oluşturur. Aşağıda bir örnek yapalım ve projemize Twig'i ekleyelim.

```
composer require twig/twig:~1.8
```

`composer init` komutu sana tam bir `composer.json` dosyası oluşturmak için rehber olabilir. Herhangi bir şekilde `composer.json` dosyanızı oluşturduktan sonra aşağıdaki komutu kullanarak Composer ile bağımlılıklarınızı `vendor/` dizini altına indirebilir ve yükleyebilirsiniz.

```
composer install
```

Daha sonra aşağıdaki satırları uygulamanızın ana PHP dosyasına ekleyiniz; bunları nasıl yapacağınız size zaten Composer'daki projelerde bildirilecek.

```
<?php
require 'vendor/autoload.php';
```

Şimdi projenize eklediğiniz kütüphaneleri kullanabilirsiniz ve bunlar projenizde talep dahilinde otomatik yükleniyor (autoloaded) olacak.

Bağımlılıkları Güncellemek

Composer, `php composer.phar install` komutunu ilk çalıştırdığınızda `composer.lock` adında bir dosya oluşturur ve bu dosyada her paketin o anki versiyonunu tutar. Projenizi başka geliştiricilerle paylaştığınızda, `composer.lock` dosyanızı da paylaşırsanız, geliştiriciler `php composer.phar install` komutunu uyguladıklarında sizinle aynı versiyondaki paketlere sahip olurlar. Bağımlılıkları (Dependencies) güncellemek için, `php composer.phar update` komutunu çalıştırabilirsiniz.

Esnek bir versiyon gereksinimlerini tanımlamak için çok kullanışlı. Örneğin ~1.8 versiyon gereksinimi "1.8.0'dan daha yeni ve 2.0.x-dev'den daha eski herşeyi" anlamına gelmektedir. Ayrıca `1.8.*` şekilde `*` da kullanabilirsiniz. Şimdi Composer'a `php composer.phar update` komutunu verdiğinizde kısıtlamalarda belirlediğiniz tanımlamalara göre bağımlılıklarınızı güncelleyecektir.

Güncelleme Bildirimleri

Yeni versiyon geldiğinde bildirim almak için VersionEye'a kayıt olabilirsiniz, bu web servisi `composer.json` dosyasındaki dosyaların Github veya BitBucket üzerindeki hesaplarını sizin için kontrol eder ve yeni bir release çıktığında size e-posta gönderir.

Güvenlik konusundaki bağımlılıklarınız kontrol etmek

Security Advisories Checker komut satırından ve web arayüzü olarak çalışabilen bir servistir. Sizin `composer.lock` dosyanızı algılar ve bir güncellemeye ihtiyacınız olup olmadığını tespit eder.

Global bağımlılıkları Composer ile yönetmek

Composer global bağımlılıkları ve binary hallerini yönetebilir. Kullanımı basit, Sadece ihtiyacınız her bir komutun başına `global` komutunu eklemek. Eğer PHPUnit'i global olarak yüklemek istiyorsanız, aşağıdaki komutu çalıştırabilirsiniz:

```
composer global require phpunit/phpunit
```

Bu komut bağımlılıklarınızın bulunacağı `~/.composer` klasörünü oluşturacaktır. Kurulan kütüphanelerinizi veya bağımlılıklarınızın binary dosyalarınını global olarak çalıştırmak için `~/.composer/vendor/bin` dizinini `$PATH` değişkenine ekleyebilirsiniz.

- Learn about Composer

PEAR

PEAR diğer bir paket yöneticisidir. Composer ile, bir kaç önemli fark dışında, hemen hemen aynıdır.

PEAR her bir paket kendi içerisinde özel bir yapısı olmasını gerektirir. Bunun anlamı paketi yazan geliştiricinin bunu hazırlaması gerekmektedir. Eğer hazırlanmamış ise PEAR

PEAR kurulumu geneldir. Bunun anlamı paketi bir kez kurduğunuzda kurulum yaptığınız sunucu üzerindeki bütün projelerinizde kullanabilirsiniz. Bu bir çok projenin aynı sürüm paketleri kullandığı durumlarda iyi olabilir. Ancak iki projede sürüm sıkıntılarından dolayı probleme yol açabilir.

PEAR Nasıl Kurulur

`phar` kurulumunu indirip onu çalıştırırarak kurabilirsiniz. kurulum adımları PEAR dökümantasyonunda her bir işletim sistemi için detaylı bir şekilde bulunmaktadır.

Eğer Linux kullanıyorsanız, dağıtımınızın paket yöneticisine bakabilirsiniz. Örneğin Debian ve Ubuntu'da `php-pear` paket olarak bulunmaktadır.

Bir Paket Nasıl Kurulur

Eğer PEAR paket listesinde listelenmiş bir paket ise, aşağıdaki gibi kurabilirsiniz:

```
pear install foo
```

Eğer paket farklı bir yrede barındırılıyor ise, öncelikle nereden yayınlandığını araştırmalısınız. Daha fazla bilgi için "Using channel docs" adresini kontrol edin.

- PEAR hakkında

Composer ile PEAR'ı İdare Etmek

Composer kullanırken bir yandan da PEAR kodlarınıda yüklemek isterseniz Composer ile bunu yapabilirsiniz. Aşağıdaki satırlar `pear2.php.net` adresinden yükleme yapmak için örnektir:

```
{
    "repositories": [
        {
            "type": "pear",
            "url": "http://pear2.php.net"
        }
    ],
    "require": {
        "pear-pear2/PEAR2_Text_Markdown": "*",
        "pear-pear2/PEAR2_HTTP_Request": "*"
    }
}
```

İlk bölüm olan `"repositories"` pear depolarını “initialise” ediyoruz. İkinci kısım olan require kısmında ise paketleri aşağıdaki gibi yazıyoruz:

> *pear-channel/Package*

bir pear kanalı diğer bir paket ile aynı isimde olabileceğinden dolayı, “pear” öneki çakışmaları önlemek için zorunludur.

BU kod çalıştırıldıktan sonra paketin Composer autoloader’i aşağıdaki gibi olacaktır:

> *vendor/pear-pear2.php.net/PEAR2_HTTP_Request/pear2/HTTP/Request.php*

Aşağıdaki gibi kullanabilirsiniz:

```
<?php
$request = new pear2\HTTP\Request();
```

- PEAR’ın Composer ile kullanımı hakkında daha fazla bilgi

Back to Top

CHAPTER 6.

Kodlama Uygulamaları

Temelleri

PHP her seviyeden geliştiriciye sadece hızlıca değil efektif şekilde kod üretmeye izin veren geniş bir dildir. Dilde kendimizi geliştirirken, sık sık dilin, ilk öğrendiğimiz (veya

gözden kaçırdığımız) kısayolları ve kötü davranışlarını unutuyoruz. Bu temel sorunu çözmek için bu bölümde PHP içindeki basit temelleri hatırlatmayı amaçlamaktadır.

- Temelleri'den okumaya devam edin.

Tarih ve Saat

PHP'de `DateTime` isimli bir sınıf bulunmaktadır ve tarih zamanla ilgili okuma, yazma, karşılaştırma veya hesaplama gibi işlerinizde yardımcı olmaktadır. PHP içerisinde tarih ve saat ile ilgili bir çok fonksiyon bulunmaktadır, ama `DateTime` sınıfı genel kullanım için nesne tabanlı güzel bir arayüz sunmaktadır. DateTime sınıfı zaman dilimlerini(time zones) işleyebilir ancak bu durumda kısa girişin dışına çıkmış oluruz.

İlk olarak `createFromFormat()` fonksiyonu ile işlenmemiş bir tarih saat metnini(string) çevirelim ve onu ekranda gösterelim. `format()` fonksiyonu DateTime nesnesini tekrar metne çevirmek için kullanılır.

```
<?php
$raw = '22. 11. 1968';
$start = \DateTime::createFromFormat('d. m. Y', $raw);

echo 'Başlangıç Tarihi: ' . $start->format('Y-m-d') . "\n";
```

DateTime ile hesaplamalar DateInterval sınıfı ile mümkünüdür. DateTime sınıfının `add()` ve `sub()` fonksiyonları DateInterval sınıfını parametre olarak alır. DateInterval kullanmak yerine, tarih ve saat işlemleriyle kendinizi yormayın. Gün ışığından yararlanma ve zaman dilimi değişiklikleri zaman hesaplamalarında umduğumuzdan daha karışık sorunlar ortaya çıkarır. Zaman farkını hesaplamak için `diff()` fonksiyonunu kullanabilirsiniz. Bu fonksiyon yeni bir DateInterval nesnesi döndürecektir.

```
<?php
// $start'ın bir kopyasını oluşturuyoruz ce bir ay 6 gün ekliyo
$end = clone $start;
$end->add(new DateInterval('P1M6D'));

$diff = $end->diff($start);
echo 'Fark: ' . $diff->format('%m ay, %d gün (toplam: %a days)'
// Fark: 1 ay, 6 gün (toplam: 37 days)
```

DateTime nesnesi ile standart karşılaştırma yöntemlerini kullanabilirsiniz:

```php
<?php
if ($start < $end) {
    echo "Başlangıç bitişten önce!\n";
}
```

Son bir örnekte DatePeriod sınıfını gösterelim. Yenilenen zaman dilimleri için kullanılır. İki DateTime nesnesini, başlangıç(start) ve bitiş(end), ve zaman aralığını parametre olarak alır. Sonunda bu aralıktaki kriterlere uyan bütün tarihleri geri döner.

```php
<?php
// $start ve $end arasındaki bütün perşembeleri yazdırıyoruz
$periodInterval = DateInterval::createFromDateString('first thu
$periodIterator = new DatePeriod($start, $periodInterval, $end,
foreach ($periodIterator as $date) {
    // her bir periyot için çıktı
    echo $date->format('Y-m-d') . ' ';
}
```

- DateTime hakkında
- Tarih Formatlama hakkında (kabul edilen tarih formatı seçenekleri)

Tasarım Desenleri (Design Patterns)

Uygulamanızı oluştururken kodunuzda ve projenizin yapısında genel bir desen kullanmak yararlı olacaktır. Desen kullanmak kodunuzu yönetmeyi kolaylaştırdığı ve diğer geliştiricilerin herşeyin birbiri ile uyumunu daha kolay anlayabilmesini sağladığı için yararlıdır.

Eğer yüksek seviye kodla yazılmış bir çatı kullanıyorsanız projeniz bu çatı üzerine kurulacaktır. Bu çatı ile bazı desen kararları zaten hazırlanmış olacaktır. Ama çatı üzerine kodunuzu yazarak en iyi desen kararlarını ortaya çıkarabilirsiniz. Diğer taraftan, uygulamanızı geliştirirken bir çatı kullanmıyorsanız, projenizin türü ve boyutu ile uyumlu en iyi deseni bulmalısınız.

- Tasarım Desenleri

UTF-8 ile Çalışmak

Bu bölüm orjinalde Alex Cabal tarafından PHP Best Practices sayfasında yazılmıştır ve kendi tavsiyelerimizin temelinde kullanılmıştır.

Dikkatli, Ayrıntılı ve Tutarlı Olun.

Şu anki PHP alt seviyelerinde Unicode desteklemiyor. UTF-8 metinlerin işlenmesi için çeşitli yolları var, ama bu çok kolay değil ve bu web uygulamanızın hemen hemen her seviyesinde değişikliğe neden olacak. HTML'den SQL'e PHP'ye. Size pratik bir özet sunmayı amaçlıyoruz.

PHP Seviyesinde UTF-8

İki metni birleştirmek ve bir metni bir değişkene atama gibi basit metin işlemlerinde, UTF-8 için özel bir şey yapmanız gerekmiyor. Ama `strpos()` ve `strlen()` gibi çoğu metin fonksiyonlarında özel bir dikkat gerekmektedir. Bu fonksiyonların sıksık `mb_*` benzerlerine sahiptir. Örneğin, `mb_strpos()` ve `mb_strlen()` gibi. Bu `mb_*` fonskiyonları, metinleri, Multibyte String Extension'ı ile Unicode metinler üzerinde işlem yapabilecek hale getirmektedir.

Ne zaman Unicode metinler üzerinde işlem yapacaksanız `mb_*` fonskiyonlarını kullanmalısınız. Örneğin, `substr()` fonksiyonunu UTF-8 metinler üzerinde kullanırsanız; sonuç olarak bazı bozuk karakterler ile karşılaşabilirsiniz. Kullanmanız gereken doğru fonksiyon `mb_substr()`dir.

Zor kısmı `mb_*` fonksiyonlarını hatırlamak. Eğer unutursanız Unicode metinlerde metinlerin bozuk çıkma ihtimalleri var.

Bütün metinler `mb_*` kullanmak zorunda değiller. Eğer yapmak istediğiniz şey için birden fazla yol varsa, bozuk karakterlerden kurtulmak için şansınız olabilir.

`mb_internal_encoding()` fonksiyonunu her PHP dosyanızın başında (ya da her yere include ettiğiniz dosyanın başında) kullanabilirsiniz, ve `mb_http_output()` fonksiyonunu browser'a çıktı vermek için kullanabilirsiniz. Açıkça her kodunuzda encoding bilgisini tanımlamanız ileride karşınıza çıkacak bir sürü baş ağrısından sizi kurtaracaktır.

Ek olarak, bir çok PHP fonksiyonu opsiyonel parametreler ile sizin metninizi seçtiğiniz bir encoding ile işleme tabi tutar.

Her zaman UTF-8 olarak göstermelisiniz. Örneğin `htmlentities()` karakter encoding için bir opsiyonu bulunmaktadır, ve eğer bir metin ile ilişkili iseniz UTF-8 olarak belirtmelisiniz. Ayrıca, PHP 5.4.0'da `htmlentities()` ve `htmlspecialchars()` için varsayılan karakter encoding'i UTF-8'dir.

Son olarak, eğer dağıtık bir uygulama kuruyorsanız ve `mbstring` eklentisinin açık olup olmadığından emin olamazsanız, patchwork/utf8 Composer paketini kullanmayı deneyebilirsiniz. Bu paket eğer uygunsa `mbstring` kullanır.

Veritabanı Seviyesinde UTF-8

Eğer MYSQL kullanıyorsanız, yukarıda belirtilen önlemleri alsanız bile verilerinizin UTF-8 olamyan hallerini tutabilirsiniz. Verilerinizin veritabanına UTF-8 olarak kaydedildiğinden emin olmak için, tüm tablo ve veritabanlarınızın character set ve collation değerlerinin `utf8mb4` olduğundan emin olun ve PDO bağlantı metninde `utf8mb4` character set'ini kullanın. Örneği gözden geçirin. Bu *özellikle önemli*.

Not olarak, tamamen UTF-8 desteği için `utf8mb4` kullanmak zorundasınız, `utf8` değil. Nedeni için okumaya devam edin.

Browser Seviyesinde UTF-8

PHP kodunuzun çıktısının UTF-8 formatında browser'a gittiğinden emin olmak için `mb_http_output()` fonksiyonu kullanın.

The browser will then need to be told by the HTTP response that this page should be considered as UTF-8. The historic approach to doing that was to include the charset `<meta>` tag in your page's `<head>` tag. This approach is perfectly valid, but setting the charset in the `Content-Type` header is actually much faster.

```php
<?php
// PHP'ye script sonuna kadar UTF-8 kullanacağımızı söylüyoruz
mb_internal_encoding('UTF-8');

// PHP'ye browser'a gönderilecek çıktının UTF-8 olacağını söylü
mb_http_output('UTF-8');

// Bİzim UTF-8 test metnimiz
$string = 'Êl síla erin lû e-govaned vîn.';

// Metnimizi multibyte fonsksiyon ile değiştiriyoruz
$string = mb_substr($string, 0, 15);

// Veritabanı bağlantısı
// Daha fazla bilgi için PDO örneklerine bakabilirsiniz.
// Not: `set names utf8mb4`
$link = new \PDO(
    'mysql:host=your-hostname;dbname=your-db;charset=utf8mb4',
    'your-username',
    'your-password',
    array(
        \PDO::ATTR_ERRMODE => \PDO::ERRMODE_EXCEPTION,
        \PDO::ATTR_PERSISTENT => false
    )
```

```
);

// Değiştirdiğimiz metni UTF-8 olarak veritabanına kaydediyoruz
// Veritabanı ve tablolarınız utf8bm4 formatında değil mi?
$handle = $link->prepare('insert into ElvishSentences (Id, Body
$handle->bindValue(1, 1, PDO::PARAM_INT);
$handle->bindValue(2, $string);
$handle->execute();

// Düzgün kaydedildiğini görmek için metni tekrar alıyoruz.
$handle = $link->prepare('select * from ElvishSentences where I
$handle->bindValue(1, 1, PDO::PARAM_INT);
$handle->execute();

// Birazdan bilgiyi HTML olarak bastıracağız
$result = $handle->fetchAll(\PDO::FETCH_OBJ);

header('Content-Type: text/html; charset=utf-8');
?><!doctype html>
<html>
    <head>
        <title>UTF-8 test sayfası</title>
    </head>
    <body>
        <?php
        foreach($result as $row){
            print($row->Body);  // This should correctly output
        }
        ?>
    </body>
</html>
```

Daha fazla bilgi

- PHP Manual: String Operations
- PHP Manual: String Functions
  - `strpos()`
  - `strlen()`
  - `substr()`
- PHP Manual: Multibyte String Functions
  - `mb_strpos()`

  - `mb_strlen()`
  - `mb_substr()`
  - `mb_internal_encoding()`
  - `mb_http_output()`
  - `htmlentities()`
  - `htmlspecialchars()`
- PHP UTF-8 Cheatsheet
- Handling UTF-8 with PHP
- Stack Overflow: What factors make PHP Unicode-incompatible?
- Stack Overflow: Best practices in PHP and MySQL with international strings
- How to support full Unicode in MySQL databases
- Bringing Unicode to PHP with Portable UTF-8
- Stack Overflow: DOMDocument loadHTML does not encode UTF-8 correctly

Back to Top

Bağımlılık Değiştirme

Wikipedia'da:

> *Bağımlılık değiştirme kodlanmış bağımlılıkların çalışma veya derlenme anında kaldırılabilmesine ve değiştirilebilmesine izin veren bir yazılım tasarım desenidir.*

Bu açıklama konuyu olduğundan biraz daha karmaşık gösteriyor olabilir. Bağımlılık değiştirme bir bileşenin bütün bağımlıklarının yaratılma anında (constructor aracılığıyla), metod kullanarak ya da özelliklere atama yaparak verilebilmesini sağlıyor. Böylece istenen anda bu bağımlıklar değiştirilebiliyor. Konu bu kadar basit aslında.

Temel Kavramlar

Basit bir örnek ile kavramları size anlatmaya çalışalım.

Veritabanı ile iletişim kuran bir adapter gerektiren bir `Database` sınıfı olsun. Bu `adapter`'in bir objesini constructor'da oluşturalım. Bu testi zorlaştırır.

```
<?php
namespace Database;

class Database
{
    protected $adapter;

    public function __construct()
```

```
    {
        $this->adapter = new MySqlAdapter;
    }
}

class MysqlAdapter {}
```

This code can be refactored to use Dependency Injection and therefore loosen the dependency.

```
<?php
namespace Database;

class Database
{
    protected $adapter;

    public function __construct(MySqlAdapter $adapter)
    {
        $this->adapter = $adapter;
    }
}

class MysqlAdapter {}
```

Now we are giving the `Database` class its dependency rather than it creating it itself. We could even create a method that would accept an argument of the dependency and set it that way, or if the `$adapter` property was `public` we could set it directly.

Karmaşık Problemler

Eğer daha önce Dependency Injection hakkında okuduysanız, *"Inversion of Control" (Kontrolün Ters Çevirilmesi)* ya da *"Dependency Inversion Principle" (Bağımlılığın Ters Çevirilmesi İlkesi)* kavramlarını duymuş olmalısınız. Bunlar Dependency Injection ile çözülmeye çalışılan sorunlardır.

Inversion of Control

"Kontrolün Tersine Çevirilmesi" kelime anlamında olduğu gibi, yapısal kontrolü nesnelerimizden tamamen ayrı tutarak sistemin kontrolünü tersine çevirmektir. Dependency Injection açısından bakarsak, bu başka bir yerde yöneterek ya da oluşturarak bağımlılıkların bitirilmesi anlamına gelir.

For years, PHP frameworks have been achieving Inversion of Control, however, the question became, which part of control are you inverting, and where to? For example, MVC frameworks would generally provide a super object or base controller that other controllers must extend to gain access to its dependencies. This **is** Inversion of Control, however, instead of loosening dependencies, this method simply moved them.

Dependency Injection allows us to more elegantly solve this problem by only injecting the dependencies we need, when we need them, without the need for any hard coded dependencies at all.

Dependency Inversion Principle

Dependency Inversion Principle is the "D" in the S.O.L.I.D set of object oriented design principles that states one should *"Depend on Abstractions. Do not depend on concretions."*. Put simply, this means our dependencies should be interfaces/contracts or abstract classes rather than concrete implementations. We can easily refactor the above example to follow this principle.

```php
<?php
namespace Database;

class Database
{
    protected $adapter;

    public function __construct(AdapterInterface $adapter)
    {
        $this->adapter = $adapter;
    }
}

interface AdapterInterface {}

class MysqlAdapter implements AdapterInterface {}
```

There are several benefits to the `Database` class now depending on an interface rather than a concretion.

Consider that you are working in a team and the adapter is being worked on by a colleague. In our first example, we would have to wait for said colleague to finish the adapter before we could properly mock it for our unit tests. Now that the dependency is an interface/contract we can happily mock that interface knowing that our colleague will build the adapter based on that contract.

An even bigger benefit to this method is that our code is now much more scalable. If a year down the line we decide that we want to migrate to a different type of database, we can write an adapter that implements the original interface and inject that instead, no more refactoring would be required as we can ensure that the adapter follows the contract set by the interface.

Containers

The first thing you should understand about Dependency Injection Containers is that they are not the same thing as Dependency Injection. A container is a convenience utility that helps us implement Dependency Injection, however, they can be and often are misused to implement an anti-pattern, Service Location. Injecting a DI container as a Service Locator in to your classes arguably creates a harder dependency on the container than the dependency you are replacing. It also makes your code much less transparent and ultimately harder to test.

Most modern frameworks have their own Dependency Injection Container that allows you to wire your dependencies together through configuration. What this means in practice is that you can write application code that is as clean and de-coupled as the framework it is built on.

Further Reading

- Learning about Dependency Injection and PHP
- What is Dependency Injection?
- Dependency Injection: An analogy
- Dependency Injection: Huh?
- Dependency Injection as a tool for testing

Back to Top

CHAPTER 7.

Veritabanları

Bilginin devamlılığı için bir çok zaman kodunuzda veritabanını kullanırsınız. Veritabanına bağlanmak için bir kaç yolunuz vardır. *PHP 5.1.0'e kadar* PHP'de bulunan mysql, mysqli, pgsql gibi veritabanı sürücülerini kullanabilirsiniz.

Eğer sadece bir veritabanı kullanıyorsanız yukarıda ki saydıklarımız çok iyi olacaktır. Ama MySQL yanında biraz da MSSQL kullanıyorsanız, ya da Oracle veritabanına bağlanmanız gerekiyorsa, aynı sürücüyü kullanma ihtimaliniz olmayacaktır. Her birisi için yeni bir API öğrenmeniz gerekecektir.

Diğer taraftan, mysql eklentisinin PHP'de aktif geliştirmesi duracaktır. PHP 5.4.0'dan bu yana resmi durumu "Uzun vadede önerilmemektedir" halindedir. Bunun anlamı bir kaç sürüm sonrasında silinecektir. PHP 5.6 ile gidebilir belki de. Eğer `mysql_connect()` ver `mysql_query()` kullanıyorsanız bu satırları tekrar yazmak durumunda kalacaksınız. Bu durumda en iyi seçenek mysql eklentisinin kullanımını belli bir geliştirme programı(takvimi) ile mysqli veya PDO ile değiştirmek olacaktır. *Eğer sıfırdan başlıyorsanız mysql eklentisi yerine MySQLi extension veya PDO'yu kullanın.*

- PHP: MySQL için Bir API Seçmek

Veritabanları MySQL

PHP için mysql eklentisi oldukça eski ve yerini 2 yeni eklentiye bırakmış haldedir:

- mysqli
- pdo

Sadece geliştirmesi durmuş değil ayrıca PHP 5.5.0 ile eskimiş olarak işaretlenmiş durumundadır, ve **PHP 7.0 ile birlikte silinecektir.**

Hangi modulü kullandığınızı görmek için `php.ini` dosyanıza bakmanız gerekmektedir. Diğer bir opsiyon'da seçtiğiniz editörde `mysql_*` diye arama yapmanız olabilir. Eğer `mysql_connect()` ve `mysql_query()` diye fonksiyonlar ile karşılaşırsanız `mysql` kullanımda demektir.

PHP 7.0 kullanmıyor olsanız bile, PHP 7.0 güncellemesi geldiğinde bu güncellemeyi yapmak sizin için çok zor olabilir. Halen geliştirmekte olduğunuz kendi uygulamanız için en iyi seçenek sizin mysql kullanımınızı mysqli veya PDO ile değiştirmek olacaktır.

Eğer mysql'den mysqli'ye güncellemek istiyorsanız, size basitçe şunu önerebiliriz. Uygulamanız içerisinde `mysql_*`ı arayıp `mysqli_*` ile değiştirin. Bu basitleştirilmiş çözüm mysqli ve PDO'nun sunduğu, parameter binding gibi imkanları kaçırmamıza neden olabilir.

- PHP: Choosing an API for MySQL
- PDO Tutorial for MySQL Developers

Veritabanları PDO

PDO bir veritabanı soyutlama kütüphanesidir. — PHP 5.1.0'den beri — PDO farklı veritabanları ile bağlantı için genel bir arayüz sunar. PDO SQL sorgularınızı çevirmez ya da eksik kısımlarını tamamlamamaz; Sadece farklı veritabanlarına bağlanmak için aynı API'yi kullanmanızı sağlar.

SQL injection atakları için `PDO` sorgunuzdaki yabancı girdileri kolaylıkla güvenli hale getirmenizi sağlar. Bu PDO deyimleri ve bağlı(bound) parametreler ile mümkündür.

Bir sorguya sayısal bir ID değeri ekleyelim ve veritabanından bir kullanıcının kayıtlarını çekmeye çalışalım. Bunu yapmak için aşağıdaki yanlış bir yöntemdir:

```
<?php
$pdo = new PDO('sqlite:users.db');
$pdo->query("SELECT name FROM users WHERE id = " . $_GET['id'])
```

Yukarıdaki kötü bir koddur. Kolaylıkla farkedilebilir ve kullanılabilir bir koddur. Kötü niyetli bir kişi `id` parametresinde sayı yerine `http://domain.com/?id=1%3BDELETE+FROM+users` gibi bir sorgu gönderebilir. Bu `$_GET['id']` değişkenine `1;DELETE FROM users` değerini atayacaktır ve bütün kullanıcılarınızı silecektir. Siz bunu PDO bağlı parameter ile girdiyi temizlemelisiniz.

```
<?php
$pdo = new PDO('sqlite:users.db');
$stmt = $pdo->prepare('SELECT name FROM users WHERE id = :id');
$stmt->bindParam(':id', $_GET['id'], PDO::PARAM_INT); // <-- PD
$stmt->execute();
```

Bu doğru bir koddur. Bu bir PDO deyiminde bağlı parametreleri kullanıyor ve yabancı girdilerden gelecek zararları engelliyor.

- PDO hakkında

Veritabanı bağlantısı kapatılmadığında kaynaklarınız bir süre sonra tükenecektir. Bunu PDO kullanarak önleyebilirsiniz. PDO sayesinde bağlantı nesnesi silinirken bağlantı kapatılır ve bütün referansları silinir veya NULL'a eşitlenir. Bunu yapmazsanız kalıcı bir bağlantı kullanmadığınız sürece kodunuz bittiğinde PHP tabiki bağlantıları otomatik olarak kapatacaktır.

- PDO bağlantıları hakkında

Sayutlama Katmanı (Abstraction Layers)

Birçok çatı PDO üzerine kurulmuş kendi soyutlanmış katmanını barındırmaktadır. Bunlar genellikle bir veritabanı sisteminin özelliklerini barındırmaktadır. Bazı durumlarda yetersiz kalsada uygulamanızın taşınabilir olması için bu gibi katmanlar güzel olabilir. (!)

Aşağıda uygulamalarınızda kullanabileceğiniz PSR-0 veya PSR-4 standardında bir kaç örnek bulunmaktadır:

- Aura SQL
- Doctrine2 DBAL
- Propel
- ZF2 Db
- ZF1 Db

Veritabanı ile etkilesim

Geliştiriciler PHP öğrenmeye başladıkları zaman, genellikle kendi yöntemleri ile bunu yapmaktadırlar ve kodları şu şekilde görünür:

```
<ul>
<?php
foreach ($db->query('SELECT * FROM table') as $row) {
    echo "<li>".$row['field1']." - ".$row['field1']."</li>";
}
</ul>
```

Bu kötü bir yöntemdir, ana olarak test edilmesi, debug edilmesi ve okunması zordur ve eğer bir limit konulmaz ise bütün alanları dışarıya çıktı olarak verebilirsiniz.

OOP ya da functional programming seçimlerine göre bunu yapmanın bir çok farklı yolları vardır.

En temel adımı düşünün:

```
<?php
function getAllFoos($db) {
    return $db->query('SELECT * FROM table');
}

foreach (getAllFoos($db) as $row) {
    echo "<li>".$row['field1']." - ".$row['field1']."</li>"; //
}
```

Bu iyi bir başlangıçtır. İki farklı kalemi iki arklı dosyaya koyun ve bazı temizlikler yapın.

Bir sınıf oluşturun ve içerisine bazı methodlarınızı koyun ve artık bir “Model”iniz var. Basit bir `.php` dosyası oluşturun ve sunum logic’inizi bu dosyanın içerisine koyun ve bir “View” katmanı oluşturun. Neredeyse bir MVC oluşturdunuz ki bu OOP mimarisi neredeyse bir çok framework‘de mevcut.

foo.php

```php
<?php

$db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8'

// Make your model available
include 'models/FooModel.php';

// Create an instance
$fooModel = new FooModel($db);
// Get the list of Foos
$fooList = $fooModel->getAllFoos();

// Show the view
include 'views/foo-list.php';
```

models/FooModel.php

```php
<?php
class FooModel()
{
    protected $db;

    public function __construct(PDO $db)
    {
        $this->db = $db;
    }

    public function getAllFoos() {
        return $this->db->query('SELECT * FROM table');
    }
}
```

views/foo-list.php

```php
<?php foreach ($fooList as $row): ?>
    <?php echo $row['field1'] ?> - <?= $row['field1'] ?>
<?php endforeach ?>
```

Bu aslında modern framework'lerin yaptıklarıyla aynı, sadece biraz daha manual yöntemler kullanılmaktadır. Tüm bu şeyleri her zaman yapanız gerekmez, ama çok fazla gösterim mantığı(presentation logic) ve veritabanı etkileşimi olan gerçek bir uygulama

veya problem olabilir. Bu durumda bu katmanları farklı şekillerde kullanmak isteyebilirsiniz. Örneğin unit-test için SQLite kullanırken uygulama için PostgreSQL kullanmanız gerekebilir.

PHPBridge veritabanı ile etkileşim konsepti için Creating a Data Class olarak bilinen güzel bir kaynaktır. Göz atabilirsiniz.

Soyutlama Katmanı

Bir çok framework PDO üzerine kurulu kendine ait bir soyutlama katmanı içerir. Bunlar genellikle PDO'nun bağlantı arayüzü için bir soyutlama oluşturmak yerine tüm bir veritabanı için bir soyutlama getirirler, bu da farklı veritabanları yerine bir veritabanı sistemini soyutlamalarına neden olur. Bu tabiki bu biraz yük katacaktır, ama eğer MySQL, PostgreSQL ve SQLite ile çalışabilecek taşınabilir bir uygulama insa etmek istiyorsak bu yük katlanılabilir bir yük olur.

Bazı soyutlama katmanları hali hazırda PSR-0 veya PSR-4 namespace standartlarına uygun inşa edilmiş ve siz uygulamanız için kullanabilirsiniz:

- Aura SQL
- Doctrine2 DBAL
- Propel
- ZF2 Db

Back to Top

CHAPTER 8.

Arayüz Kalıpları

Arayüz kalıpları sizin controller ve domain mantıklarınızı sunum katmanınızdan ayrımanıza imkan sağlar. Kalıplar genellikle uygulamanızın HMTL kısmıdır, ama siz ayrıca XML gibi diğer formatları da kullanabilirsiniz. Kalıplar genellikle model–view–controller (MVC) yazılım mimarisinin ikinci parçası olan, "views" olarak da adlandırılır.

Yararları

Sunum katmanındaki mantığı uygulamanın geri kalanından ayırmak template(şablon) mantığının kullanımındaki ana yarardır. Template yani şablonlar formatlanmış içeriği göstermekle görevlidirler. Verinin erişimi ya da verinin sürekliliği ya da daha karmaşık görevler için değillerdir. Geliştiricilerin sunucu tarafındaki kodla (controller, model) ilgilendiği ve tasarımcıların kullanıcı tarafındaki kodla (markup) ilgilendiği bir takım çalışması içerisinde daha okunalı ve temiz kod yazmayı sağlar.

Şablonlar sunum katmanındaki kodların organizasyonunuda geliştirir. Şablonlar genellikle "views" klasörü altında olurlar ve her biri bir tekil dosya içerisinde bulunur. Bu taklaşım büyük kod parçalarının daha küçük hallere bölünmesiyle birlikte tekrar kullanılabilirliğini de artırır. Örneğin, sizin sitenizde bir başlık ve birde alt başlık kısmı varsa, bu şablonlar her bir sayfanın başına ve sonuna eklenebilir.

Sonuç olarak, kullandığınız kütüphanelere bağlı olarak, şablonlar kullanıcı tarafından oluşturulan içeriklerde bazı kısımlarda işlemler(escaping) yaparak daha güvenli olmasını sağlar. Bazı kütüphaneler daha kapalı kutu olur ve tasarımcılar sadece izin verilen değişkenlere ve methodlara erişebilirler.

Yalın PHP Şablonları

Yalın PHP şablonları düz PHP kodlarını kullanan basit şablonlardır. Bu doğal bir seçimdir çünkü PHP'nin kendisi aslında bir şablon dilidir. Bunun anlamı siz PHP dilini rahatça HTML gibi dillerin arasında kullanabilirsiniz. Bu PHP geliştiricilerinin yararınadır çünkü başka yeni bir dil öğrenmelerine gerek yoktur, bildikleri fonksiyonları kullanabilirler ve hatta kullandıkları editörlerinin renklendirme seçeneklerini bile değiştirmeden kullanabilirler. Dahası, düz PHP şablonları tekrar derlenen şablon yöntemlerine göre derlenme aşaması olmadığından daha da hızlı çalışacaktır.

Her modern PHP framework yapısı bir çeşit şablon sistemi kullanmaktadır, bir çoğuda varsayılan olarak düz PHP kullanır. Framework'ler dışında Plates veya Aura.View gibi kütüphaneler kalıtım, tasarım modeli(layout) ve eklenti gibi modern özellikleri sağlayan düz PHP şablonları ile çalışabilmeyi sağlarlar.

Örnek bir düz PHP şablonu (Plates kütüphanesi ile):

```
<?php $this->insert('header', ['title' => 'User Profile']) ?>

<h1>User Profile</h1>
<p>Hello, <?=$this->escape($name)?></p>

<?php $this->insert('footer') ?>
```

Derlenmiş Şablonlar(Template)

PHP nesne odaklı bir dil olarak olgunlaşırken, aynı gelişimi şablon dili olma yolunda gerçekleştiremedi. Twig veya Smarty gibi derlenen şablonlama sistemleri, bu boşluğu farklı dizimleri ile doldurmaya çalışıyorlar. Otomatik sızıntıları önleme, basit kontrol yapıları ve kalıtım özellikleri ile derlenebilen şablon sistemleri kolay kod yazmanın yanında temiz, okunabilir ve daha güvenli kullanım için tasarlanmıştır. Derlenen şablonlar farklı diller arasında paylaşılabilirler bile. Buna Mustache güzel bir örnektir.

Şablonlar derlenirken performans olarak biraz sıkıntı yaratabilir ancak bu derlenmiş şablon önbelleğe alındığında çok çok küçük bir performans kaybı yaratacaktır.

Örnek bir derlenmiş şablon (Twig kütüphanesi ile):

```
{% include 'header.html' with {'title': 'User Profile'} %}

<h1>User Profile</h1>
<p>Hello, {{ name }}</p>

{% include 'footer.html' %}
```

Ekstra Kaynaklar

Makaleler & Örnekler

- Templating Engines in PHP
- An Introduction to Views & Templating in CodeIgniter
- Getting Started With PHP Templating
- Roll Your Own Templating System in PHP
- Master Pages
- Working With Templates in Symfony 2

Kütüphaneler

- Aura.View *(native)*
- Blade *(compiled, framework specific)*
- Dwoo *(compiled)*
- Latte *(compiled)*
- Mustache *(compiled)*
- PHPTAL *(compiled)*
- Plates *(native)*
- Smarty *(compiled)*
- Twig *(compiled)*
- Zend\View *(native, framework specific)*

Back to Top

CHAPTER 9.

Hatalar ve İstisnalar

Hatalar

Birçok “exception-heavy (exception-yoğun)” programlama dilinde, herhangi bir şey yanlış gittiğinde bir eception fırlatılır. Bu durumlar için kesinlikle en uygun yol budur, ancak PHP maalesef “exception-yoğun” bir programlama dili değildir. Eception fırlatma özelliği olsa da (daha çok nesnelerle çalışırken), PHP ölümcül bir hata oluşmadıkça ne olduğuna bakmaksızın işlem yapmaya devam edecek.

Örnek:

```
$ php -a
php > echo $foo;
Notice: Undefined variable: foo in php shell code on line 1
```

Bu en düşük seviyede bir hata uyarısı oluşurur ve PHP mutlu mesut çalışmaya devam eder. Bu durum “exception-heavy” programlama geçmişinden gelenlere karmaşık ve saçma gelebilir.Örneğin Python bu durumda bir exception fırlatıp çalışmayı durduracaktır:

```
$ python
>>> print foo
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'foo' is not defined
```

Aslında tek gerçek fark, Python’un küçük de olsa herhangi bir şey için çalışmayı durdurmasıdır, böylece geliştiriciler herhangi bir olası sorunun veya aykırı durumun olmadığından emin olabilirler. Bunun yanında PHP hata raporladığı ve oluşurdu bir durum olmadıkça işleme devam edecektir.

Hata Ciddiyeti

PHP farklı hata ciddiyet seviyesine sahiptir. En yaygın üç mesaj türü hatalar, bildirimler ve uyarılardır. Bunlar farklı şiddet derecelerine sahiptir; `E_ERROR`, `E_NOTICE`, and `E_WARNING`. Hatalar ölümcül çalışma zamanı hatalarıdır ve genellikle kodunuzdaki hatalardan kaynaklanır ve PHP’nin çalışmayı durdurmasına neden olacağı için düzeltilmesi gerekir. Bildirimler ölümcül olmayan hatalardır, komut dizisinin çalıştırılması durdurulmaz. Uyarılar kodun neden olduğu veya komut dosyasının çalıştırılması sırasında sorun çıkarmayabilecek öneri mesajlarıdır ve kodun çalıştırmasını durdurmaz.

Derleme zamanında bildirilen başka bir hata mesajı türü ise `E_STRICT` mesajlarıdır. Bu mesajlar ortak çalışabilirliği sağlamak ve PHP’nin yaklaşmakta olan sürümleriyle uyumluluk sağlamak için kodunuzdaki değişiklikler önermek için kullanılır.

PHP’nin Hata Raporlama Davranışını Değiştirme

Hata raporlama şekli PHP ayarlarını değiştirerek ya da fonksiyon kullanarak değiştirilebilir. PHP'nin yerleşik fonksiyonlarından `error_reporting()` ile ön tanımlı hata seviyesi değerlerini kullanarak hata seviyesini belirleyebilirsiniz. Örneğin sadece uyarı ve hataları görmek ama bildirimleri görmek istemiyorsanız aşağıdaki gibi ayarlama yapabilirsiniz:

```
error_reporting(E_ERROR | E_WARNING);
```

Ayrıca, hataların ekrana yazılıp (geliştirme ortamı için gerekli) yazılmamasını ve log dosyasına kaydedilmesini (canlı ortamlara daha uygun) ayarlayabilirsiniz. Daha fazla bilgi için Error Reporting bölümüne bakabilirsiniz.

Hata Yazdırmayı Önleme

PHP'ye, hata kontrol operatörü `@` ile ilgili hataları saklamasını da söyleyebilirsiniz. Bu işleci bir ifadenin başlangıcına koyarsınız ve ifadenin doğrudan sonucu olan herhangi bir hata gizlenir.

```
echo @$foo['bar'];
```

Yukarıdaki örnekte eğer `$foo['bar']` değeri varsa ekrana yazdırılır, ama `$foo` değişkeni yada `'bar'` elamanı null döner ve ekrana hiçbirşey yazmaz. Eğer hata kontrol operatörü kullanılmazsa, bu satır `PHP Notice: Undefined variable: foo` ya da `PHP Notice: Undefined index: bar` hatalarına sebep olur.

Bu kullanım güzel birşeymiş gibi görünse de bazı istenmeyen durumlara da yol açabilir. PHP `@` barındıran satırları `@` barındırmayan satırlara göre daha yavaş işler. Erken (alt seviye) optimizasyon tartışmalı bir konu olabilir, ancak uygulamanız veya kütüphaneniz için performans özellikle önemliyse, hata kontrol operatörünün performans etkilerini anlamak önemlidir.

İkinci önemli bir konu ise, hata kontrol operatörü oluşan hatayı **tamamen** gizler. Hata ekrana basılmadığı gibi herhangi bir log dosyasına da gönderilmez. Ayrıca PHP'nin hata kontrol operatörünü kapatma gibi bir ayarı veya özelliği de yoktur. Farkında olmadığınız daha az tehlikeli olduğunu düşündüğünüz bir hatanın farkında olduğunuz bir hatadan daha zararlı oluduğunu kabul edebiliriz.

Eğer hata kontrol operaötürün kullanmadan yazabiliyorsanız, o yolu seçmeniz daha doğrdur. Yukarıda örnek olarak kullandığımız satır şöyle yazılabilir;

```
echo isset($foo['bar']) ? $foo['bar'] : '';
```

Hata kontrol öperatörünün kullanılmasının faydalı olduğu bir durum da `fopen()` fonksiyonu kullanılmasıdır. Dosyayı okumadan önce var olup olmadığını kontrol

edebilirsiniz ama az da olsa `fopen()` çalışmadan önce dosyanın silinmesi ihtimali vardır. Bu durumda `fopen()` fonksiyonu false döner ve hata fırlatır. Bu PHP'nin çözmesi gereken bir konu ama şimdilik hata kontrol operatörünün kullanılmasının en haklı olduğu durum gibi duruyor.

Daha önce de belirttiğimiz gibi PHP'nin ayarlarında hata kontrol operatörü kapatmanın hiçbir yolu yoktur. Ama xDebug eklentisinin `xdebug.scream` ayarı ile hata kontrol operatörünü etkisiz hale getirebilirsiniz. Bu ayar değişikliğini `php.ini` dosyasında aşağıdaki gibi yapabilirsiniz.

```
xdebug.scream = On
```

Çalışma anında bu değişikliği `ini_set` fonksiyonu ile aşağıdaki gibi de yapabilirsiniz.

```
ini_set('xdebug.scream', '1')
```

"Scream" PHP eklentisi de benzer ayarı yapmanızı sağlar. Scream eklentisinin ayarı ise `scream.enabled` şeklindedir.

Bu eklenti hata ayıklama yaparken ve ya hataların ekrana yazılmasının kapatıldığını düşündüğünüz durumlarda kullanabilirsiniz. Bu şekilde kullanırken dikkatli olmalısınız çünkü bazı kütüphaneler hata kontrol operatörü kapatıldığında çalışmayabilir.

- Error Control Operators
- SitePoint
- xDebug
- Scream

ErrorException

PHP mükemmel bir "exception odaklı" programlama dili olma yeteneğine sahiptir ve geçiş yapmak için sadece birkaç satır kod gerektirir. Temel olarak, "Exception" sınıfından türeyen "ErrorException" sınıfını kullanarak hataları "Exception" olarak atabilirsiniz.

Bu, Symfony ve Laravel gibi çok sayıda modern uygulama çatıları tarafından uygulanan yaygın bir uygulamadır. Varsayılan olarak Laravel `app.debug` anahtarı açıksa, Whoops! Paketini kullanarak tüm hataları Exception olarak görüntüler, anahtar kapalıysa hataları gizler.

Hataları geliştirme aşamasında Exception olarak atarak, bunları normal sonuçtan daha iyi bir şekilde ele alabilir ve geliştirme sırasında bir Exception görürseniz, durumu nasıl ele alacağınızla ilgili özel talimatlar içeren bir "catch" ifadesi içine alabilirsiniz. Çalışma anında yakaladığınız her Exceptiom, uygulamanızı daha da sağlam kılar.

`ErrorException` hakkında daha fazla bilgiyi ErrorException Class sayfasından bulabilirsiniz.

- Error Control Operators
- Predefined Constants for Error Handling
- error_reporting
- Reporting

İstisnalar (Exceptions)

İstisnalar bir çok programlama dilinin en popüler kısmıdır, ama genellikle PHP geliştiriciler için gözden kaçmıştır. Ruby gibi dillerde İstisnalar ağır derecede vardır, HTTP isteği hata vermesi, veya veritabanı sorgusu yanlış olduğunda, veya resim bulunamadığında bile Ruby ekrana bir istisna tetikleyecektir.

PHP'nin kendisi bu konuda oldukça gevşektir, ve `file_get_contents()` çağrısı genellikle bir `FALSE` dönder ve bir uyarı gösterir. Codeigniter gibi bir çok eski PHP çatılarında sadece `false` dönecektir, hata mesajlarına ekleyecektir ve hatayı `$this->upload->get_error()` bu gibi yöntemleri kullanarak görebilirsiniz. Buradaki problem, hatanın son derece bariz olması yerine hatanın yerine bakmak ve hangi metotta olduğunu bulmak için dökümanlarına bakmak zorunda olmaktadır.

Diğer bir problem, sınıflar otomatik olarak bir hata fırlattıklarında ve işlemden çıkıldığındadır.

When you do this you stop another developer from being able to dynamically handle that error. Exceptions should be thrown to make a developer aware of an error, then they can choose how to handle this. E.g:

```
<?php
$email = new Fuel\Email;
$email->subject('My Subject');
$email->body('How the heck are you?');
$email->to('guy@example.com', 'Some Guy');

try
{
    $email->send();
}
catch(Fuel\Email\ValidationFailedException $e)
{
    // The validation failed
}
catch(Fuel\Email\SendingFailedException $e)
```

```
{
    // The driver could not send the email
}
finally
{
    // Executed regardless of whether an exception has been thr
}
```

SPL İstisnalar

Genel `Exception` sınıfı geliştiriciler için çok az ayıklama kaynağı sağlayabilir. Ancak, buna çare olarak, varsayılan `Exception` sınıfını kapsayan özelliştirilmiş bir `Exception` sınıfı oluşturabilirsiniz:

```
<?php
class ValidationException extends Exception {}
```

Birden fazla `catch` bloğu ile farklı istisnalar yakalayabilirsiniz. Bu bir sürü özel istisna yaratılmasına sebep olabilir, Bunlardan bazılarını SPL extension'ın sunduğu SPL istisnalarını kullanarak önleyebilirsiniz.

Eğer `__call()` sihirli metodunu kullanıyorsanız ve geçersiz bir metot talep edildiğinde standart istisnalar yerine ki onlar belirsizler, sadece bunun için özel bir istisna oluşturabilirsiniz, `throw new BadFunctionCallException;` bu şekilde hatayı elle oluşturabilirsiniz.

- Read about Exceptions
- Read about SPL Exceptions
- Nesting Exceptions In PHP
- Exception Best Practices in PHP 5.3

Back to Top

CHAPTER 10.

Güvenlik

Web Uygulama Güvenliği

Web uygulamalarından yararlanmaya hazır art niyetli bazı insanlar vardır. Uygulamanız için gerekli önlemleri almak adına bu katı durum önemlidir. Neyse ki, bu gibi durumlara karşı alınması gereken güvenlik önlemler ve bilinmesi gerekenler The Open Web

Application Security Project (OWASP)'de derlenmiş.Bu yazı uygulamanızın güvenliği için geliştirici tarafından okunmalı.

- Read the OWASP Security Guide

Şifre Karıştırma (hashing)

Herkes bir şekilde, şifre ile korunan kullanıcı giriş ekranı bulunan bir uygulama yapar. Kullanıcı adı ve şifre veritabanında barındırılır ve giriş sırasında kimlik doğrulaması için kullanılır.

Bu aşamada en önemli şey şifre saklanmadan önce *karıştırılmasıdır (hash)*. Şifre karıştırıldıktan sonra geri döndürülemezdir ve tek yönlüdür. Sabit uzunluktadır ve geri döndürülemez. Bunun anlamı, özgün metine ulaşılamaz ancak karıştırılmış şifreler birbiri ile karşılaştırarak doğruluğu belirlenir. Eğer şifreler karıştırılmaz ise 3. kişilerçe bir şekilde veritabanına ulaşılırsa kişilerin bilgileri tehlikeye girer. Bazı kullanıcılar (maalesefki) diğer servisler ile aynı şifreleri kullanabilir. Bu yüzden, bu konuda ciddi güvenlik önlemi almak gerekir.

`password_hash` ile Şifre Karıştırma

PHP 5.5 ile `password_hash` fonksiyonu geldi. `password_hash` şu anda PHP'nin desteklediği güçlü bir algoritmaya sahip BCrypt kütüphanesini kullanıyor. Ama gerektiğinde daha fazla algoritmayı desteklemek için güncellenecektir. `password_compat`, password_* fonksiyonlarının PHP 5.3.7'den üstünede destek verebilmek için oluşturulmuş bir kütüphanedir.

Aşağıda bir metni karıştıralım, sonra yeni bir metin ile karşılaştıralım. İki metin farklı olduğu için karşılaştırma sonucu yanlış olarak dönecektir.

```
<?php
require 'password.php';

$passwordHash = password_hash('secret-password', PASSWORD_DEFAU

if (password_verify('bad-password', $passwordHash)) {
    // Correct Password
} else {
    // Wrong password
}
```

- `password_hash` hakkında
- `password_compat` for PHP >= 5.3.7 && < 5.5

- Kriptografi açısından karıştırmayı öğrenin
- PHP `password_hash` RFC

Veri Süzme

Yabancı bir girdiye kesinlikle güvenmeyin. Yabancı girdiden gelen veriyi kesinlikle temizleyin ve onaylayın (validate). `filter_var` ve `filter_input` fonksiyonları metni temizler ve onaylar (validate). (e-posta adresi vs.)

Yabancı girdi herşey olabilir: `$_GET` ve `$_POST` ile gelen veriler, bazı `$_SERVER` değerlerinde ve `fopen('php://input', 'r')` aracılığıyla gelen HTTP isteklerinde gibi. Yabancı girdiler sadece kullanıcı form verileri ile sınırlandırılmamalı. Dosya yükleme ve indirmeler, oturum verileri, çerez verileri ve sisteminiz dışından aldığınız web servislerinden gelen verilerde yabancı girdidir.

Yabancı veriler saklanabilir, birleştirilebilir ve daha sonra erişilebilir, ancak o halen yabancı bir girdidir. Her işleminizde çıktı verdiğinizde, birleştirmenizde veya kodunuzun içine kattığınızda kendinize "Veri güvenli mi?" sorusunu tekrar tekrar sorun.

Veri amacına göre farklı şekillerde süzülebilir. Örneğin, Süzgeçten geçirilmemiş bir yabancı girdi, HTMl içerisinde kullanılabilir ve bir javascript kodunu çalıştırabilir. Bu `Cross-Site Scripting (XSS)` olarak bilinir ve çok tehlikeli bir ataktır. Bunu önlemenin bir yolu vardır. Kullanıcı tarafından oluşturulan bütün girdilerdeki HTML kodlarını `strip_tags` fonksiyonu ile silmektir veya `htmlentities`, `htmlspecialchars` fonksiyonları ile HTML özel karakterleri kaçış karakterleri ile değiştirmektir.

Diğer bir örnek komut satırına parametre göndermektir. Bu genellikle tehlikelidir. Ancak kullanılacaksa `escapeshellarg` fonskiyonu ile parametreleri temizleyebilirsiniz.

Son bir örnek ise, yabancı girdiyi dosya sisteminde bir dosyayı load etmek için kullandığımızı düşünelim. Burada bir açık oluşabilir. Burada "/", "../", null bytes, veya diğer karakterleri girdiden gelen dosya adından silmeniz gerekebilir. Aksi durumda gizli, özel veya hassas dosyalara ulaşılabilir.

- Veri Süzme hakkında
- `filter_var` hakkında
- `filter_input` hakkında
- null byte ile başa çıkma hakkında

Sterilize Etmek (Sanitization)

Sterilize etmek kural dışı ve güvenli olmayan karakterlerin yabancı girdilerden silinmesi demektir.

Örneğin, yabancı girdileri HTML içerisine ya da SQL sorgusuna göndermeden önce sterilize etmelisiniz. Eğer parametreleri gönderirken PDO kullanırsanız, o sizin için girdileri sterilize edecektir.

Bazen HTML sayfalarında sadece bazı güvenli tagların kullanılması gerekmektedir. Bu çoğu zaman çok zor bir iştir. Bunu önlemek için Markdown veya BBCode gibi kısıtlı biçimlendiriciler kullanılır. Bu nedenle HTML Purifier gibi kütüphaneler mevcuttur.

Sterilize Etme Filtreleri

Doğrulama

Doğrulama yabancı girdiler için ön bir eleme sağlar. Örneğin, bir kayıt işlemi sırasında bir e-posta adresini, bir telefonu ya da yaşı doğrulamak isteyebilirsiniz.

Doğrulama Süzgeçleri

Konfigürasyon Dosyaları

Uygulamanız için konfigürasyon dosyaları oluşturacağınız zaman, aşağıdaki yöntemleri izlemenizi öneririz:

- Konfigüreayon dosyalarınızı direk olarak erişilemeyecek ve dosya sistemi vasıtasıyla çekilemeyecek bir konumda barındırmanızı tavsiye ederiz.
- Eğer ana dizinde barındırmak zorunda iseniz onun adını '.php' uzantısı ile yazınız. Bu direk erişime izin verse bile metin olarak okunamamasını sağlar.
- Konfigürasyon dosyalarınızdaki bilgileriniz ya şifreleme yöntemleri ile ya da grup/kullanıcı sistem izinleri ile korunmalıdır.

Register Globals

NOT: PHP 5.4.0 itibariyle `register_globals` özelliği kaldırıldı ve bundan sonrada kullanılmayacak. Bu sadece uygulamanızı güncellemeniz için bir uyarı olarak eklendi.

Etkinleştirildiğinde, `$_POST`, `$_GET` ve `$_REQUEST` gibi değişkenler uygulamanızın her yerinden erişilebilir olacaktır. Bu verinin nerden geldiğini bilinemeyeceği için kolayca güvenlik açığına sebep olabiliyor.

Örneğin: `$_GET['foo']` değişkeni `$foo` olarak kullanılabilecek, daha önceden tanımlanmamış bir değişkeni etkin kılacaktır. Eğer 5.4.0'dan daha küçük versiyonlarda PHP kullanıyorsanız, `register_globals` ayarını **off** durumuna getirin.

- PHP manual'de Register_globals

Hata Raporları

Hata logları uygulamanızdaki sorunları bulmak için yararlı olabilir, ama aynı zamanda dış dünyaya uygulama yapısı hakkında bilgi gösterebilir. Bu gibi sorunlarda uygulamanızı korumak için geliştirme ve canlı ortamın farklı konfigüre edilmesi gerekebilir.

Geliştirme

Geliştirme ortamında olası her hatayı görmek için aşağıdaki satırları `php.ini` dosyanıza ekleyebilirsiniz.

```
display_errors = On
display_startup_errors = On
error_reporting = -1
log_errors = On
```

> *`-1`, PHP'nin gelecekteki sürümlerinde yeni sabitler ve seviyeler eklense de olası her hatayı gösterecektir. `E_ALL` PHP 5.4'de aynı işi görecektir. - [php.net]*

`E_STRICT` hata seviyesi 5.3.0 sürümü ile başladı ve `E_ALL` ile aynı değildir, ama 5.4.0 sürümünde `E_ALL` ın bir parçası oldu. Bu yüzden `-1` ya da `E_ALL | E_STRICT` kullanmalısınız.

PHP versionlarına göre olası tüm hata raporları

- < 5.3 `-1` or `E_ALL`
- 5.3 `-1` or `E_ALL | E_STRICT`
- > 5.3 `-1` or `E_ALL`

Canlı Ortam

Hataları canlı ortamda gizelemek için `php.ini` dosyasına aşağıdaki konfigürasyonları eklemelisiniz :

```
display_errors = Off
display_startup_errors = Off
error_reporting = E_ALL
log_errors = On
```

Canlı sistemde bu ayarlar ile, hatalar halen web server için saklanacaktır, ama kullanıcılara gösterilmeyecektir. Daha fazla bilgi için aşağıdaki linkleri kullanabilirsiniz :

- error_reporting
- display_errors
- display_startup_errors
- log_errors

Back to Top

CHAPTER 11.

Test Etme

Otomatize edilmiş testler yazmak, iyi geliştirilmiş bir uygulama elde etmiş olmanın yanı sıra, iyi yöntem olarak kabul edilir. Yeni bir özellik eklediğinizde, bir özelliği düzenlediğinizde ya da yoksaydığınızda otomatik oluşturulan testler uygulamanızın çakılmamasından emin olmak için muhteşem bir araçtır.

PHP için farklı yaklaşımlarda kullanılan bir kaç farklı test araçları (veya yapı(framework)) mevcuttur - tüm bunlar, sadece şu andaki fonksiyonaliteyi bozmadığından emin olmayı, elle yapılan testleri ve geniş kalite test ekipleri ihtiyacını önlemeyi hedefliyor.

Test Odaklı Geliştirme

Wikipedia'den:

> *Test odaklı geliştirme (TOG), geliştirme sürecinin döngüsel tekrarına dayana bir yazılım geliştirme sürecidir: ilk olarak geliştirici bir özelliği ya da fonksiyonu tanımlayan (başlangıçta başarısız) bir test oluşturur, sonra testi geçmek için az sayıda kodlar eklenir ya da çıkarılır ve sonunda yazılımın işlevini ve davranışını değiştirmeden iç yapısında yapılan değişikliklerle yeni kod kabul edilir standartlara uygun bir hale getirilir. Kent Beck, TOG tekniğini keşfeden ve geliştiren olarak kabul edilen kişi, 2003 yılında TOG'un basit tasarımı ve özgüveni teşvik ettiğini belirti.*

Uygulamanız için kullanabileceğiniz bir kaç çeşit test yöntemi vardır:

Birim Test (Unit Testing)

Unit Testing fonskiyonları, sınıfları ve metodları bir geliştirme döndüsü üzerinden tüm çalışma zamanında çalıştığından emin olmak için bir programlama yaklaşımıdır. Fonskiyonların ve metodların içindeki ya da dışındaki değişkenlerin değerlerini kontrol ederek iç mantığın doğru çalıştığını kontrol edebilirsiniz. Bağımlılık kontrolü ve sahte sınıflar ve taslaklar inşa ederek, bağımlılığı doğrulayabilirsiniz ve bu daha iyi bir test kapsamı için doğru kullanımdır.

Bir sınıf ya da metod oluşturduğunuzda, bu sınıf ya da metodun her bir davranışı için bir birim test oluşturmalısınız. Basit seviyede, yanlış bir argüman gönderdiğinizde hata vereceğinden ve doğru bir argüman gönderdiğinizde çalışacağından emin olmalısınız.

Geliştirme döngüsünde bu sınıf ya da metod içerisinde değişiklik olduğunda eski işlevleri umduğumuz gibi çalışacaktır.

Birim test için diğer bir kullanım açıkkaynağa katkıda bulunmaktır. Bozuk işlevleri göstermek ve sonrasında düzeltmek için bir test yazabilirsiniz. Testi geçebildiğini gösterebilirsiniz. Bir projeyi çalıştırdığınızda gönderilen istekler için testi bir gereklilik olarak önerebilirsiniz.

PHPUnit PHP uygulamaları için birim test yazmak için de-facto test çatısıdır. Buna ek olarak bir kaç alternatif de bulunmaktadır.

- atoum
- Enhance PHP
- PUnit
- SimpleTest

Entegrasyon Test (Integration Testing)

Wikipedia'den:

Entegrasyon testi (bazen entegrasyon ve test olarakta anılır, kısaltılmış hali "I&T"dir.) uygulamanızdaki modüllerin birleştirildiği ve bir grup olarak test edildiği bir test aşamasıdır. Birim test (Unit Test) sonrası ve doğrulama testleri (Validation Testing) öncesi oluşturulur. Entegrasyon testinde birim testi yapılmış modülleri girdi olarak alınır, daha büyük gruplar oluşturacak şekilde birleştirilir, entegrasyon test planında tanımlanan testler uygulanır ve sistem testi için hazır bir şekilde entegra sistem sunulur.

Birim test için kullanılan bir çok araç entegrasyon testi için de kullanılabilir.

Fonksiyonel Test (Functional Testing)

(acceptance testing) olarak da bilinen, fonksyonle test birimlerin birbirleri ile doğru konuşup konuşmadığını ve birimlerin tek tek düzgün çalışıp çalışmadığını kontrol etmek yerine uygulamanızı gerçekten kullanan otomatik test araçları oluşturmak için kullanılan araçlar kullanarak oluşturulur. Bu araçlar gerçek veriler ile ve gerçek kullanıcıları simule ederek çalışırlar.

Fonksiyonel Test Araçları

- Selenium
- Mink
- Codeception is a full-stack testing framework that includes acceptance testing tools
- Storyplayer is a full-stack testing framework that includes support for creating and destroying test environments on demand

Davranış Odaklı Geliştirme (Behaviour Driven Development)

İki tür Davranış Odaklı Geliştirme (Behavior-Driven Development - BDD) vardır: SpecBDD ve Story BDD. SpecBDD teknik davranışa veya koda odaklanır. StoryBDD iş veya özellik davranışlarına ve etkileşimlerine odaklanır. PHP her iki tür içinde çatıya sahiptir.

StoryBDD ile, uygulamanızın davranışlarını açıklamak için okunabilir hikayeler yazabilirsiniz. Bu hikayeler uygulamanıza karşı test olarak kullanılabilir. PHP uygulamanızda StoryBDD için Behat adındaki çatı kullanabilirsiniz. Bu çatı Ruby için yazılmış Cucumber çatısından ilham alınarak oluşturulmuş. Hikayeleri oluşturmak için Gherkin DSL dili kullanılır.

SpecBDD ile kodunuzun gerçekteki davranışlarınızı açıklayan açıklamalar yazabilirsiniz. Fonksiyon ya da metodu test etmek yerine, bu fonksiyon ya da metodun nasıl davrandığını açıklayabilirsiniz. PHP PHPSpec çatısını bu amaç için sunmaktadır. Bu çatı için Ruby için yazılmış RSpec project projesinden ilham alınmıştır.

BDD Links

- Behat, the StoryBDD framework for PHP, inspired by Ruby's Cucumber project;
- PHPSpec, the SpecBDD framework for PHP, inspired by Ruby's RSpec project;
- Codeception is a full-stack testing framework that uses BDD principles.

Tamamlayıcı Test Araçları

Bireysel testler ve davranış odaklı çatıların yanısıra, tek bir yöne yönelimi olmayan bir sürü kapsamlı çatı(framework) ve yardımcı kütüphaneler(helper library) vardır.

Araç Bağlantıları

- Selenium tarayıcı otomatize etme aracıdır. PHPUnit ile bütünleşik kullanılabilir.
- Mockery bir Mock nesne çatısıdır. PHPUnit veya PHPSpec ile bütünleşik kullanılabilir.
- Prophecy is a highly opinionated yet very powerful and flexible PHP object mocking framework. It's integrated with PHPSpec and can be used with PHPUnit.

Back to Top

CHAPTER 12.

Sunucular ve Dağıtım

PHP uygulamalarını dağıtmanın ve web üzerinde yayınlamanın birkaç yolu vardır.

Platform as a Service (PaaS)

PaaS size PHP uygulamanızı çalıştırmanız için bir ağ mimarisi ve sistem sunar. Bunun anlamı, PHP uygulamanızı veya çatınızı çalıştırmak için çok az ayarlama yapmanız gerektiğidir.

Son zamanlarda, PaaS dağıtım, barındırma ve PHP uygulamalarını ölçeklendirmede popüler bir yöntem oldu. PHP PaaS "Platform as a Service" sağlayıcıları listesini kaynaklar bölümünde bulabilirsiniz.

Sanal veya Özel Sunucular (Virtual or Dedicated Servers)

Sistem yönetimini seviyorsanız veya öğrenmek istiyorsanız, virtual veya dedicated sunucular uygulamanızın yayınlamanız için tam bir kontrol sağlarlar.

nginx ve PHP-FPM

PHP, PHP'nin dahili FastCGI Process Manager (FPM)'ı üzerinden, nginx ile hafif ve yüksek performanslı web sunucusu olarak güzel eşleşiyor. Apache'ye göre daha az bellek kullnırken daha fazla isteği cevaplıyabiliyor. Bu özellikle sanal sunucular üzerinde, ki bellek her zaman bir ihtiyaç, çok önem kazanıyor.

- nginx Hakkında
- PHP-FPM Hakkında
- nginx ve PHP-FPM Güvenli Kurulumu Hakkında

Apache ve PHP

PHP ve Apache'nin uzun bir geçmişi var. Apache çok geniş konfigürasyona ve işlevselliğini artırmak için çokça modüle sahip. PHP çatıları ve Wordpress gibi açık kaynak kodlu uygulamaların kolay kurulumu için en çok seçilen ortak sunucu uygulamasıdır. Ne yazık ki, nginx ile karşılaştırıldığında daha fazla kaynak kullanmaktadır ve aynı zamandaki kaldırabileceği ziyaretçi sayısı daha azdır.

Apache PHP'yi çalıştırmak için bir kaç konfigürasyona sahip. En yaygın ve en kolay kurulum, mod_php5 ile prefork MPM'dir. Bellek kullanımı verimli olmasada kullanımı ve çalışması en kolay olandır. Sunucu tarafı ile çok fazla uğraşmak istemiyorsanız sizin için uygun bir seçimdir. *Eğer mod_php5 kullanacaksanız, prefork MPM kullanmalısınız.*

Alternatif olarak, Apache dışında daha fazla performans ve kararlılık istiyorsanız, nginx'le aynı FPM sisteminden yararlanabilirsiniz ve mod_fastcgi veya mod_fcgid ile worker MPM veya event MPM modüllerini çalıştırabilirsiniz. Bu konfigürasyon belleği daha verimli kullanacak ve daha hızlı olacak ama kurulum aşaması daha zor olacaktır.

- Apache Hakkında
- Multi-Processing Modulü Hakkında
- mod_fastcgi Hakkında
- mod_fcgid Hakkında

Paylaşımlı Sunucu

PHP'nin popülaritesine teşekkür etmek gerekir. PHP kurulu olmayan bir barındırıcı (host) bulmak zor. Ancak son sürüm olmasına dikkat edilmeli. Paylaşımlı sunucu diğer geliştiricilerle aynı makinede uygulamanı yayınlamana izin verir. Bunun artısı daha ucuz olmasıdır. Diğer taraftan komşunuz ne karıştırıyor, ne yapıyor hiç bir zaman bilemiyorsunuz. Sunucu düşebilir ya da güvenlik açığı oluşabilir. Projenizin bütçesi dahilinde paylaşımlı sunuculardan kaçının.

Uygulamanızı Build Etmek ve Deploy Etmek

Eğer dosyalarınızı güncellemeden önce kendinizi elle (manually) veritabanı şemasını güncellerken veya testlerinizi elle (manually) çalıştırıyor buluyorsanız, iki kere düşünün!

Uygulamanızın yeni versiyonunu deploy etmek için elle (manually) yapmanız gerekecek her ekstra işlem ile büyük hatalar yapma ihtimaliniz artacaktır. İster basit bir güncellemeyle uğraşırken, ister kapsamlı bir yapı kuruyorken ve hatta sürekli entegrasyon stratejisiyle uğraşıyorken build otomasyonu arkadaşınızdır.

Otomatikleştirmek isteyeceğiniz bazı işlemler:

- Bağımlılık yönetimi
- Dosyalarınızın (assets) derleme, sıkıştırma (minification) işlemleri
- Test çalıştırma
- Dökümantasyon oluşturma
- Paketleme
- Deployment

Otomasyon Araçlarını Hazırlamak

Otomasyon araçları, yazılım deployment'larının yaygın görevlerini idare eden betik koleksiyonu olarak tanımlanabilir. Build araçları yazılımınızın bir parçası değildir, sadece yazılımınızla dışarıdan çalışır.

Bazıları PHP, bazıları farklı dillerde olmak üzere build otomasyonunda size yardımcı olabilecek bir çok açık kaynak araç bulunmaktadır. Eğer belirli bir iş için daha uygunlarsa PHP ile yazılmış olmamaları sizi durdurmasın. İşte bir kaç örnek:

Phing PHP dünyasında otomatik deployment a başlamak için en kolay yoldur. Phing ile basit bir XML dosyasından paketleme, deployment veya test işlemlerini kontrol

edebilirsiniz. Phing (Apache Ant tabanlıdır) bir web uygulamasını yüklemek ve ya güncellemek için genellikle ihtiyaç duyulabilecek geniş çaplı işlemleri hizmetinize sunar ve PHP de yazılmış özel işlemler ile geliştirilebilir.

Capistrano her seviyede programcının kullanabileceği bir ya da birden fazla makinada uzaktan düzenli ve tekrar edebilecek şekilde konut çalıştırmasılmasını sağlayan bir sistemdir. Aslında Ruby on Rails uygulamaları için ayarlanmış olmasına rağmen **PHP uygulamaları** içinde başarıyla kullanılmaktadır. Etkili kullamiö için Ruby ve Rake bilgisi gereklidir.

Dave Gardner'ın yazısı PHP Deployment with Capistrano Capistrano kullanmak isteyen PHP programcıları için güzel bir başlangıç noktası olabilir.

Chef bir dağıtım çerçevesinden daha fazlasıdır. Yalnızca uygulamanızı dağıtmakla kalmayan, tüm sunucu ortamınızı veya sanal sunucu paaketlerinizi da oluşturabilen, çok güçlü bir Ruby tabanlı sistem entegrasyon çatısıdır.

PHP kullanlar için Chef hakkında bazı kaynaklar:

- Three part blog series about deploying a LAMP application with Chef, Vagrant, and EC2
- Chef Cookbook which installs and configures PHP 5.3 and the PEAR package management system

Daha fazla kaynak:

- Automate your project with Apache Ant
- Maven, a build framework based on Ant and how to use it with PHP

Sürekli Entegrasyon (Continuous Integration)

> *Sürekli Entegrasyon, bir ekibin üyelerinin sık sık işlerini entegre ettiği, genellikle her kişinin en az günlük olarak bütünleştiği - günde birden fazla entegrasyona yol açan bir yazılım geliştirme pratiğidir. Birçok ekip bu yaklaşımın önemli > ölçüde entegrasyon sorunlarını azaltığını ve ekibin daha uyumlu bir yazılım geliştirmesini sağladığını tecrübe etti.*

– Martin Fowler

PHP için sürekli entegrasyon uygulamanın bir çok farklı yolu vardır. [Travis CI] (https://travis-ci.org/) küçük projeler için bile sürekli entegrasyonu uygulamayı kolaylaştırarar harika bir iş çıkardı. Travis CI açık kaynak topluluğu için entegrasyon hizmeti sağlayan bir uygulamasıdır. GitHub ile entegredir ve birçok kişi için PHP dahil birinci sınıf destek sunar.

Daha fazla kaynak:

- Continuous Integration with Jenkins
- Continuous Integration with PHPCI
- Continuous Integration with Teamcity

Back to Top

CHAPTER 13.

Sanallaştırma

Vagrant (Sanal Sunucu)

Geliştirme ortamı ile yayınlama ortamınız farklı ise projenizi yayınladığınızda farklı hatalar ile karşılaşabilirsiniz. Bir ekip ile çalışırken, herkesin farklı geliştirme ortamlarında çalışması ve tüm uygulamaların aynı sürümde veya güncel olabilmesi çok zorlaşıyor.

Eğer Windows bir makinede geliştirme yapıp Linux bir makinede yayınlıyorsanız veya bir ekip ile yazılım geliştiriyorsanız sanal bir makine kurmayı düşünmelisiniz. Bu ilk etapta zor gelebilir, ama Vagrant kullanarak basit birkaç adım ile sanal bir makine kurabilirsiniz. Vargant üzerindeki temel box'lar elle ayarlanabilir, veya daha önceden kurulu bir vargant box'ınız varsa bunu yapmak için Puppet veya Chef gibi "provisioning" yazılımları kullanabilirsiniz.

Provisioning, birbirleri ile özdeş birden fazla box kurulurken ve gerektiğinde silinirken, tekrarlanacak olan bir çok kurulum karmaşasından korunmak için güzel bir yöntemdir. Bir box'ı silebilir ve hiçbir manuel yöntem kullanmadan tekrar oluşturabilirsiniz. Bu güzel ve sorunsuz bir kurulumu size sağlar.

Vargant, projenizi local sunucunuz ve oluşturduğunuz sanal makine ile paylaşmanız için paylaşımlı bir klasör oluşturur. Bunun yararı, siz local sunucunuzda kodlarınızı yazarken, projenizin sanal sunucunuzda çalışmasına imkan tanımasıdır.

Bunu ufak bir örnekle açıklayalım. Farzedelim ki siz Windows tabanlı sunucunuzda kendi IDE veya Text editörünüzü kullanarak bir proje geliştiriyorsunuz. PHP, MySQL gibi araçlar ise linux sanal sunucunuzda kurulu. Proje klasörünüzü Vagrant'ın paylaşım klasörü olarak ayarlarsanız, windows ortamında geliştirmeye devam edip, çalıştırma işlemini sanal sunucu üzerinde yapabilirsiniz. O klasör iki yönlü ve senkron olarak çalışacaktır.

Birazcık Yardım

Eğer Vagrant'a başlamak için küçük bir yardım istiyorsanız, üç tane servis sizin için yardımcı olabilir:

- Rove: Bu servis size PHP'nin de aralarında olduğu dada önceden oluşturulmuş Vagrant yapıları sunar. Provizyon şef ile yapılır.
- Puphpet: PHp geliştirmelerine sanal makine kurmak için basit bir arayüzdür. **Ağırlıklı olarak PHP'ye odaklanır.** Yerel makinelerin yanında, bulutda yayınlamak içinde kullanabilirsiniz. Provizyon Puppet ile yapılır.
- Protobox: is a layer on top of vagrant and a web GUI to setup virtual machines for web development. A single YAML document controls everything that is installed on the virtual machine.

CHAPTER 14.

Docker

Back to Top

CHAPTER 15.

Caching

PHP oldukça hızlıdır, ancak dosya okuma, uzak bilgisayara bağlantı gibi işlemlerde bazı darboğazlar oluşmaktadır. Neyse ki, uygulamanızın belli kısımlarını hızlandırmanız için, veya belirli bir zamandaki kaynakların kullanımını azaltmak için çeşitli araçlar bulunmakta.

Bytecode Cache (Önbelleği)

PHP dosyaları çalıştırıldığı zaman, kodlar ilk olarak bytecode'a (aynı zamanda opcode olarak bilinir) derlenir ve daha sonra bytecode çalıştırılır. Eğer PHP değiştirilmemişse, bytecode her zaman aynıdır. Bunun anlamı derleme aşaması CPU kaynaklarını boşa zaman harcamak demektir.

Bytecode gereksiz derlemeleri önlemek için bellekte saklanacaktır ve oradan çağırılacaktır. Kurulumu ve ayarlamaları sadece dakikalar alırken uygulamanız önemli ölçüde hızlanacaktır. Kurmamanız için hiç bir neden yok.

OPcache'de dahili olarak gelen bytecode cache vardır. Ayrıca önceki versiyonlar içinde kullanılabilir.

Popüler bytcode önbellek araçları:

- APC (PHP 5.4 and earlier)
- APC

- XCache
- Zend Optimizer+ (part of Zend Server package)
- WinCache (extension for MS Windows Server)

Nesnel Önbellekleme

Veritabanından çağırmanın veya verinin getirilmesinin masraflı olduğu ve verinin değişmediği durumlar gibi kodunuzda bireysel nesneleri önbelleklemenin yararlı olabileceği zamanlar vardır. Bu verileri bellekte tutmak ve daha sonra hızlıca erişmek için nesnel önbellekleme yazılımlarını kullanabilirsiniz. Veritabanı sunucusuna binen yükü azaltarak performansı artırabilirsiniz.

Birçok popüler Bytecode Cache çözümü özel verilerinizi önbelleklemenize izin verir, *so there's even more reason to take advantage of them.* APCu, XCache, ve WinCache PHP verilerinizi önbelleklemek için bir arayüz (API) sunar.

En genel kullanılan nesnel önbellekleme sistemleri APCu ve memcached'dır. APCu mükkemmel bir seçimdir, belleğe erişmek için, basit bir arayüze sahiptir. Kolay kurulabilirdir ve kullanılabilirdir. APCu'nin bir sınırılaması sunucuda yüklü bulunmasıdır. Memcached ise ayrılmış bir servis olarak kurulur ve network üzerinden kullanılır, bunun anlamı br çok farklı sistemden erişebileceğiniz merkezi bir süper hızlı veri depolama imkanı sağlamasıdır.

PHP (Fast-)CGI olarak sunucunuzda çalışıyorsa, her işlem kendi önbelleğini kullanacaktır. APCu verileri işlemler arasında paylaşılmayacaktır. Bu durumda, memcached kullanmayı tercih edebilirsiniz, memcached PHP işleme sürecine bağlı değildir.

Bir ağ yapılandırmasında APCu memcached'dan erişim hızı açısından genellikle daha iyi performans gösterecektir, ama memcached daha hızlı ve daha fazla büyümek için uygun olacaktır. Eğer uygulamanızın ekstra özelliklerini veya birden fazla sunucuda çalışmasını önemsemiyorsanız, memcached APCu'den nesnel önbellekleme için daha iyi bir seçimdir.

APCu örneği:

```
<?php
// veri 'expensive_data' olarak önbelleklenmişmi diye kontrol e
$data = apc_fetch('expensive_data');
if ($data === false) {
    // veri önbellekte değil; sonucu sonradan çağırmak için önb
    apc_add('expensive_data', $data = get_expensive_data());
}
```

```
print_r($data);
```

Nesnel önbellekleme sistenleri hakkında daha fazla bilgi:

- APCu
- APC Fonksiyonları
- Memcached
- Redis
- XCache APIs
- WinCache Fonksiyonları

Back to Top

CHAPTER 16.

Documenting

Back to Top

CHAPTER 17.

PHPDoc

PHPDoc is an informal standard for commenting PHP code. There are a *lot* of different tags available. The full list of tags and examples can be found at the PHPDoc manual.

Below is an example of how you might document a class with a few methods;

```
<?php
/**
 * @author A Name <a.name@example.com>
 * @link http://www.phpdoc.org/docs/latest/index.html
 * @package helper
 */
class DateTimeHelper
{
    /**
     * @param mixed $anything Anything that we can convert to a
     *
```

```
     * @return \DateTime
     * @throws \InvalidArgumentException
     */
    public function dateTimeFromAnything($anything)
    {
        $type = gettype($anything);

        switch ($type) {
            // Some code that tries to return a \DateTime objec
        }

        throw new \InvalidArgumentException(
            "Failed Converting param of type '{$type}' to DateT
        );
    }

    /**
     * @param mixed $date Anything that we can convert to a \Da
     *
     * @return void
     */
    public function printISO8601Date($date)
    {
        echo $this->dateTimeFromAnything($date)->format('c');
    }

    /**
     * @param mixed $date Anything that we can convert to a \Da
     */
    public function printRFC2822Date($date)
    {
        echo $this->dateTimeFromAnything($date)->format('r');
    }
}
```

The documentation for the class as a whole firstly has the @author tag, this tag is used to document the author of the code and can be repeated for documenting several authors. Secondly is the @link tag, used to link to a website indicating a relationship between the website and the code. Thirdly it has the @package tag, used to categorize the code.

Inside the class, the first method has an @param tag documenting the type, name and description of the parameter being passed to the method. Additionally it has the @return and @throws tags for documenting the return type, and any exceptions that could be throw respectively.

The second and third methods are very similar and have a single @param tag as did the first method. The import difference between the second and third method is doc block is the inclusion/exclusion of the @return tag. `@return void` explicitly informs us that there is no return, historically omitting the `@return void` statement also results in the same (no return) action.

Back to Top

CHAPTER 18.

Kaynaklar

Kaynak

- PHP Website
- PHP Documentation

Takip Etmek için Kişiler

- Rasmus Lerdorf
- Fabien Potencier
- Derick Rethans
- Chris Shiflett
- Sebastian Bergmann
- Matthew Weier O'Phinney
- Pádraic Brady
- Anthony Ferrara
- Nikita Popov

Danışmanlık

- phpmentoring.org - PHP topluluğunda örgün, birebir danışmanlık.

PHP Servis Platformları (PaaS) Sağlayıcıları

- PagodaBox
- AppFog

- Heroku (PHP desteği dökümante edilmemiş ama tutarlı Facebook ortaklığı üzerine kurulu link)
- fortrabbit
- Engine Yard Cloud
- Red Hat OpenShift Platform
- dotCloud
- AWS Elastic Beanstalk
- cloudControl
- Windows Azure
- Zend Developer Cloud
- Google App Engine
- Jelastic

Çatılar (Frameworks)

Tekerleği yeniden icat etmektense bir çok geliştirici çatı yapısını kullanır. Çatı birçok düşük seviye sorunları soyutlar, ortak sorunları çözmek için kullanımı kolay bir arayüz sunar.

Her projede kullanmanız gerekmez. Bazen düz PHP doğru yoldur, ama bir çatı kullanmanız gerekirse 3 ana çeşidi vardır:

- Micro Frameworks (Mini Çatı)
- Full-Stack Frameworks (Yığın Çatı)
- Component-Based Frameworks (Bileşen Bazlı Çatı)

Mini Çatı `callback`, `controller`, `method` gibi yapıların HTTP isteklerine mümkün olduğunda çabuk bir yönlendirme imkanı sağlar. Geliştirme sürecine yardım etmek için basit veritabanı işlemleri gibi bazı ekstra kütüphaneler ile birlikte gelir. HTTP servisleri oluşturmak için kullanılır.

Bir çok çatı yapısında Mini Çatı bulunmaktadır ve üzerine önemli ölcüde özellikl eklenmesi ile oluşan yapılara Yığın Bazlı Çatı denir. Bu genellikle ORM'ler, Kimlik Doğrulama paketleri ile birlikte gelir.

Bileşen Bazlı Çatı özel amaçlı kütüphaneler ve özel koleksiyonlar topluluğudur. Farklı Bileşen Bazlı Çatılar mini veya Yığın Bazlı Çatı kurmak için kullanılabilir.

- Popüler PHP Çatıları

Bileşenler (Components)

Daha öncede değinildiği üzere "Bileşenler" paylaşımlı kod, yayınlama ve oluşturma aşamalarında genel bir amaç için diğer bir yaklaşımdır. Bir çok bileşen deposu vardır. Bunlardan ana iki tanesi:

- Packagist
- PEAR

İki araçta yükleme ve yükseltme işlemlerinde yardımcı olmak için komut satırında çalışan bir arayüze sahiptir. Daha fazla bilgi için Bağımlılık Yönetimi bölümüne bakabilirsiniz.

There are also component-based frameworks and component-vendors that offer no framework at all. These projects provide another source of packages which ideally have little to no dependencies on other packages, or specific frameworks.

For example, you can use the FuelPHP Validation package, without needing to use the FuelPHP framework itself.

- Aura
- FuelPHP
- Hoa Project
- Orno
- Symfony Components
- The League of Extraordinary Packages
- Laravel's Illuminate Components
  - Eloquent ORM
  - Queue

Laravel's Illuminate components will become better decoupled from the Laravel framework. For now, only the components best decoupled from the Laravel framework are listed above.

Kitaplar

Çevrenizde PHP için bir çok kitap bulunmakta ama bir çoğu eski ve artık doğru bilgi içermiyor. "PHP 6" için yazılmış kitaplar bile yok ve olmayabilir de.

Bu bölüm PHP'de geliştirme kitap önerileri için yaşayan bir döküman olmayı hedefler. Eğer kitabınızın eklenmesini istiyorsanız, bir pull-request gönderin. Bu isteğiniz değerlendirilecektir.

Ücretsiz Kitaplar

- PHP Usulüne Uygun - This website is available as a book completely for free

Ücretli Kitaplar

- Modernizing Legacy Applications In PHP - Get your code under control in a series of small, specific steps

- Building Secure PHP Apps - Learn the security basics that a senior developer usually acquires over years of experience, all condensed down into one quick and easy handbook
- The Grumpy Programmer's Guide To Building Testable PHP Applications - Learning to write testable doesn't have to suck
- Securing PHP: Core Concepts - A guide to some of the most common security terms and provides some examples of them in every day PHP
- Scaling PHP - Stop playing sysadmin and get back to coding
- Signaling PHP - PCNLT signals are a great help when writing PHP scripts that run from the command line.

Back to Top

CHAPTER 19.

Topluluk

Büyük olduğu kadar farklı(çeşitlilik açısından) PHP toplulukları mevcuttur ve bu toplulukların üyeleri yeni PHP yazılımcılarını desteklemeye hazır ve isteklidir. Yerel PHP gruplarına katılarak veya daha büyük PHP konferanslarına katılarak PHP hakkında daha iyi bilgiler edinebilir ve daha iyi örneklere ulaşabilirsiniz. PHP'nin irc.freenode.com adresindeki #phpc IRC kanalında takılabilirsin ve twitter üzerinden @phpc hesabını takip edebilirsin. Yeni geliştiricilerle tanışabilirsin, yeni konular öğrenebilirsin veya yeni arkadaşlar edinebilirsin. Google+ PHP Programmer community ve StackOverflow PHP geliştiricilerinin takıldığı diğer yerlerdir.

Resmi PHP Etkinlikleri Takvimi Hakkında

PHP Kullanıcı Grupları

Büyük bir şehirde yaşıyorsanız, yakınlarınızda bir PHP kullanıcı grubu vardır. Resmin bir liste bulunmamasına rağmen Google veya Meetup.com veya PHP.ug üzeriden bu grupları arayabilirsiniz. Daha küçük bir şehirde yaşıyorsanız, yerel bir grup olmayabilir. Bu durumda siz bir tane başlatın.

PHP Wiki'deki Kullanıcı Grupları Hakkında

PHP Konferansları

PHP toplulukları tarafından dünya üzerinde birçok ülkede bölgesel ve ulusal konferanslar düzenlemektedir. PHP topluluğundan bilinen kişiler genellikle büyük etkinliklerde konuşurlar, ve bu endüstrinin liderlerinden birşeyler öğrenmek için büyük bir fırsattır.

PHP konferansları

Back to Top

Proje Sahibi ve Yöneticisi

Josh Lockhart
Phil Sturgeon
Katkıda Bulunanlar