PHP

Do Jeito Certo.

Tweetar

Bem vindo

Existe muita informação obsoleta na Web que desnorteia novos programadores PHP, espalhando más práticas e códigos inseguros. *PHP: Do Jeito Certo* é uma referência rápida e fácil de ler, introduzindo desenvolvedores às melhores práticas, renomados padrões de código e links para tutoriais competentes pela Web.

É importante entender que *não existe uma maneira canônica de usar PHP*. Este site introduz novos desenvolvedores PHP às melhores práticas, opções disponíveis e boas informações, que por muitas vezes, quando descobertas, já é tarde demais. Além disso leva o desenvolvedor a ter conceitos maduros sobre coisas que eram feitas durante anos e nunca foram reconsideradas. Esse site não é para dizer quais ferramentas você deve utilizar, mas ao invés disso, oferecer sugestões e múltiplas opções, e quando possível explicar as diferenças de acordo com casos de uso.

Este é um documento “vivo”, que continuará sendo atualizado com mais informações úteis e exemplos, assim que eles estiverem disponíveis.

Como contribuir

Ajude a tornar este site o melhor recurso para novos programadores PHP! Contribua no GitHub

Espalhe a palavra!

PHP: Do Jeito Certo possui banners que você pode usar em seu site. Mostre seu apoio e deixe que novos desenvolvedores PHP saibam onde encontrar boas informações!

Banners para divulgação

Começando

Use a versão estável atual (7.4)

Se você está começando com o PHP, certifique-se de começar com a versão estável mais recente PHP 7.4. O PHP 7.4 adiciona novas_funcionalidades às antigas versões 5.x. O motor foi amplamente reescrito e o PHP é agora ainda mais rápido que as versões antigas.

Você deveria tentar atualizar para a última versão estável rapidamente - PHP 5.6 não receberá atualizações de segurança após 2018. A atualização é simples e não existem muitas quebras de

compatibilidade com versões anteriores. Se você não tem certeza em qual versão uma função ou determinado recurso se encontra, você pode verificar a documentação do PHP no site php.net.

Servidor web embutido

Com a versão PHP 5.4+, você pode começar a aprender PHP sem os problemas de instalar e configurar um servidor web. Para iniciar o servidor, execute o seguinte comando no seu terminal dentro da raiz de seu projeto:

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

- Saiba mais sobre o servidor web embutido, pela linha de comando

Instalação no Mac

O macOS já vem com o PHP, mas ele é normalmente um pouco atrasado em relação à última versão estável. Existem várias maneiras de instalar a última versão do PHP no macOS.

Instalar PHP via Homebrew

Homebrew é um gerenciador de pacotes para o macOS que pode ajudá-lo a instalar facilmente o PHP e várias extensões. O repositório central do Homebrew contém "fórmulas" para PHP 5.6, 7.0, 7.1, 7.2 e 7.3. Instale a última versão com este comando:

```
brew install php@7.3
```

Você pode alternar entre as versões do PHP do Homebrew modificando a variável `PATH`. Alternativamente você pode usar o brew-php-switcher para alternar as versões PHP automaticamente.

Instalar PHP via Macports

O projeto MacPorts é uma iniciativa da comunidade de código aberto para projetar um sistema fácil de usar para compilar, instalar e atualizar softwares pela linha de comando, X11 ou Aqua no sistema operacional OS X.

O MacPorts suporta binários pré-compilados, portanto, você não precisa recompilar todas as dependências dos arquivos tar de origem, ele agiliza sua vida se você não tiver nenhum pacote instalado no seu sistema.

Neste ponto, você pode instalar `php54`, `php55`, `php56`, `php70`, `php71`, `php72` ou `php73` usando o comando `port install`, como por exemplo:

```
sudo port install php56
sudo port install php73
```

E você pode utilizar o comando `select` para trocar a versão ativa do PHP:

```
sudo port select --set php php73
```

Instalar PHP via phpbrew

phpbrew é uma ferramenta para instalar e gerenciar múltiplas versões do PHP. Isso pode ser realmente útil se duas diferentes aplicações/projetos precisam de diferentes versões do PHP e você não está usando máquinas virtuais.

Instalar PHP via instalador binário da Liip's

Outra opção popular é o php-osx.liip.ch que fornece métodos de instalação para versões PHP da 5.3 até a 7.3. Ele não substitui os binários do PHP instalados pela Apple, mas instala tudo em uma localização separada (/usr/local/php5).

Compilar do código fonte

Outra opção que lhe dá o controle sobre a versão do PHP você instalar, é compilar o código fonte. Nesse caso, certifique-se de ter instalado o Xcode ou o substituto da Apple "Command Line Tools for XCode" que pode ser encontrado no *Apple's Mac Developer Center*.

Instaladores *All-in-One*

As soluções listadas acima lidam apenas com o PHP e não fornecem coisas como Apache, Nginx ou um servidor SQL. As soluções *All-in-one* como MAMP and XAMPP vão instalar estes outros programas para você e amarrá-los todos juntos, mas a facilidade de configuração vem com um contra-ponto de flexibilidade.

Instalação no Windows

Você pode fazer o download dos binários no site windows.php.net/download. Depois de extrair o arquivo baixado, é recomendado incluir na variável PATH a pasta raiz do seu PHP (onde o arquivo php.exe está localizado). Dessa forma você pode executar o PHP de qualquer lugar.

Para aprender e desenvolver localmente, você pode utilizar o servidor web embutido do PHP 5.4+, de forma que você não precisa se preocupar em configurá-lo. Se você prefere um "pacote completo" que inclui um servidor web e MySQL, então ferramentas como Web Platform Installer, XAMPP, EasyPHP, OpenServer and WAMP irão ajudá-lo a montar rapidamente um ambiente de desenvolvimento em Windows. Dito isso, estas ferramentas serão um pouco diferentes das ferramentas em produção, portanto tenha cuidado quanto às diferenças de ambiente caso esteja trabalhando em Windows e publicando em Linux.

Caso você precise rodar seu ambiente de produção em Windows, então o IIS7 vai lhe dar mais estabilidade e melhor performance. Você pode usar o phpmanager (um plugin GUI para o IIS7) para tornar a configuração e o gerenciamento do PHP mais simples. O IIS7 vem com o FastCGI embutido e pronto para uso, você só precisa configurar o PHP como handler. Para suporte e mais recursos, existe uma área dedicada no iis.net ao PHP.

Geralmente rodar sua aplicação em ambientes diferentes para desenvolvimento e produção podem levar a bugs estranhos enquanto estiverem rodando. Se você está desenvolvendo no Windows e irá fazer deploy no Linux (ou qualquer outro ambiente não Windows) então você deveria considerar usar uma Máquina Virtual.

Chris Tankersley tem uma postagem muito útil em quais ferramentas ele usa para o Desenvolvimento PHP usando Windows.

Guia de estilo de código

A comunidade PHP é grande e diversa, composta por inúmeras bibliotecas, frameworks e componentes. É comum para desenvolvedores PHP escolher vários destes e combiná-los em um único projeto. É importante que código PHP siga (o mais próximo possível) um estilo de código comum para que desenvolvedores PHP possam misturar várias bibliotecas em seus projetos.

O Framework Interop Group propôs e aprovou uma série de recomendações de estilo, conhecidas como PSR-0, PSR-1, PSR-2 e PSR-4. Não deixe os nomes estranhos confundí-lo, estas recomendações são meramente um conjunto de regras que projetos como Drupal, Zend, Symfony, CakePHP, phpBB, AWS SDK, FuelPHP, Lithium etc. estão começando a adotar. Você pode utilizá-las para seus próprios projetos, ou continuar utilizando seu estilo pessoal.

Idealmente você deveria escrever código PHP que adere a um ou mais destes padrões. Pode ser qualquer combinação das PSR's, ou um dos padrões de código feitos pela PEAR ou Zend. Isso significa que outros desenvolvedores podem facilmente ler e trabalhar no seu código, e aplicações que implementem os componentes possam ter consistência, mesmo trabalhando com bastante código de terceiros.

- Leia sobre a PSR-0
- Leia sobre a PSR-1
- Leia sobre a PSR-2
- Leia sobre a PSR-4
- Leia sobre os Padrões de Código da PEAR
- Leia sobre os Padrões de Código do Symfony

Você pode usar o PHP_CodeSniffer para checar seu código contra qualquer uma dessas recomendações e plugins para editores de texto como o Sublime Text 2 para fazer a verificação em tempo real.

Você pode corrigir o estilo do código com uma das duas possíveis ferramentas. Uma é a PHP Coding Standards Fixer do Fabien Potencier que tem uma base de código muito bem testada. É maior e mais lenta, mas muito estável e usada por grandes projetos como Magento e Symfony. Outra opção é a php.tools, que se tornou popular pelo plugin sublime-phpfmt do sublime. Apesar de ser mais novo ela tem grandes melhorias no desempenho fazendo com que a correção do estilo de código em tempo real seja mais fluída.

O Inglês é o idioma preferido para todos os nomes simbólicos e para a infra-estrutura do código. Comentários devem ser escritos em qualquer idioma que possa ser facilmente lido por todos os atuais e futuros desenvolvedores que possam trabalhar nessa base de código.

Destaques da linguagem

Paradigmas de programação

O PHP é uma linguagem dinâmica e flexível, que suporta uma variedade de técnicas de programação. Ele evoluiu drasticamente com o passar dos anos, notavelmente adicionando um sólido modelo de orientação a objetos no PHP 5.0 (2004), funções anônimas e namespaces no PHP 5.3 (2009) e traits no PHP 5.4 (2012).

Programação orientada a objetos

O PHP possui um conjunto completo de funcionalidades de programação orientada a objetos, incluindo suporte à classes, classes abstratas, interfaces, herança, construtores, clonagem, exceções e muito mais.

- Leia sobre PHP orientado a objetos
- Leia sobre Traits

Programação funcional

PHP suporta funções de primeira classe, o que significa que funções podem ser atribuidas a variáveis. Tanto funções nativas como funções definidas por usuários podem ser referenciadas por uma variável e invocadas dinamicamente. Funções também pode ser passadas como argumentos para outras funções (funcionalidade chamada de funções de ordem superior) e funções podem retornar outras funções.

Recursão, uma funcionalidade que permite que funções realizem chamadas para elas mesmas também é suportada pela linguagem, mas a maioria dos códigos em PHP tem foco em iteração.

Novas funções anônimas (incluindo suporte para closures) também estão presentes de o PHP 5.3 (2009).

PHP 5.4 inclui a habilidade de vincular closures com o escopo de objetos e também melhorou o suporte para invocaveis (callables) tanto que elas podem ser usadas indistintamente com funções anónimas na maioria dos casos.

- Continue lendo em Programação Funcional em PHP
- Leia mais sobre Funções Anônimas
- Leia mais sobre Closures
- Mais detalhes na RFC sobre Closures
- Leia mais sobre invocáveis (callables)
- Leia sobre invocamento dinâmico de funções com `call_user_func_array`

Meta Programação

PHP suporta várias formas de meta-programação através de mecanismos como a API de reflexão e métodos mágicos. Existem vários métodos mágicos disponíveis como __get(), __set(), __clone(), __toString(), __invoke(), etc. Isso permite que desenvolvedores alterem o comportamento das classes. Desenvolvedores Ruby costumam dizer que o PHP carece de `method_missing`, mas ele está disponível com `__call()` e __callStatic().

- Leia sobre Métodos Mágicos
- Leia sobre Reflexão

Namespaces

Como mencionado anteriormente, a comunidade PHP tem muitos desenvolvedores criando muito código. Isso significa que o código de uma biblioteca PHP pode usar o mesmo nome de classe que uma outra biblioteca. Quando ambas bibliotecas são usadas no mesmo namespace, elas colidem e causam problemas.

Os *Namespaces* resolvem esse problema. Como descrito no manual de referência do PHP, os namespaces podem ser comparados com os diretórios dos sistemas operacionais, que fazem *namespace* dos arquivos; dois arquivos com o mesmo nome podem coexistir em diretórios separados. Da mesma forma, duas classes PHP com o mesmo nome podem coexistir em namespaces PHP separados. Simples assim.

É importante que você use namespace no seu código para que ele possa ser usado por outros desenvolvedores sem risco de colisão com outras bibliotecas.

Um modo recomendado de usar namespaces está descrito na PSR-4, que tem como objetivo fornecer uma convenção padrão para arquivos, classes e namespaces para parmitir um código plug-and-play.

Em outubro de 2014 o PHP-FIG (Framework Interop Group) depreciou o padrão para auto-carregamento anterior: a PSR-0. Tanto a PSR-0 e a PSR-4 ainda são perfeitamente utilizáveis. A última requer o PHP 5.3, então muitos projetos que rodam apenas em PHP 5.2 implementam a PSR-0.

Se você estiver planejando usar um padrão para auto-carregamento para uma nova aplicação ou pacote, olhe na PSR-4.

- Leia sobre os Namespaces
- Leia sobre a PSR-0
- Leia sobre a PSR-4

Standard PHP Library

A Standard PHP Library (SPL), ou Biblioteca Padrão do PHP, vem empacotada com o PHP e fornece uma coleção de classes e interfaces. Ela é composta principalmente por classes de estruturas de dados normalmente necessárias (pilha, fila, heap e outras) e iteradores que podem percorrer por essas estruturas de dados ou por suas próprias classes que implementem as interfaces SPL.

- Leia sobre a SPL

Interface de Linha de Comando

O PHP foi criado primariamente para escrever aplicações web, mas ele também é útil para criar scripts de linha de comando (CLI). Programas PHP de linha de comando podem te ajudar a automatizar tarefas comuns como testes, publicação e administração de aplicações.

Programas PHP CLI são poderosos pois você pode usar o código de sua aplicação diretamente sem precisar criar e proteger uma GUI (Interface Gráfica do Usuário) web para isso. Apenas tenha a certeza de não colocar seus scripts PHP CLI na raiz pública do seu servidor web!

Tente executar o PHP a partir da sua linha de comando:

```
> php -i
```

A opção `-i` irá mostrar a sua configuração do PHP da mesma forma que a função `phpinfo`.

A opção `-a` fornece um shell interativo, similar ao IRB do ruby e ao shell interativo do python. Também existe um número de outras opções de linha comando úteis.

Vamos escrever um programa CLI "Hello, $name" simples. Para testá-lo, crie um arquivo chamado `hello.php`, como mostrado a seguir.

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

O PHP define duas variáveis especiais baseadas nos argumentos que seu script receber. `$argc` é uma variável integer que contém a *quantidade* de argumentos e `$argv` é uma variável array que contém o *valor* de cada argumento. O primeiro argumento sempre é o nome do arquivo PHP do seu programa, no caso `hello.php`.

A expressão `exit()` é usada com um número diferente de zero para informar ao shell que o comando falhou. Códigos de saída normalmente usados podem ser encontrados aqui.

Para executar nosso script acima, a partir da linha de comando:

```
> php hello.php
Usage: php hello.php [name]
> php hello.php world
Hello, world
```

- Aprenda sobre como executar o PHP a partir da linha de comando
- Aprenda sobre como configurar o Windows para executar o PHP a partir da linha de comando

XDebug

Uma das ferramentas mais úteis no desenvolvimento de software é um depurador apropriado. Ele permite que você trace a execução do seu código e monitore os itens na pilha de execução. XDebug, um depurador de PHP, pode ser utilizado por várias IDEs para prover *breakpoints* e inspecionar a pilha. Ele também lhe permite que ferramentas como PHPUnit e KCacheGrind realizem análise de cobertura e perfis de código.

Se você perceber que você está travado, disposto a recorrer a *var_dump*/*print_r*, e ainda assim não consegue resolver o problema - talvez você devesse usar um depurador.

Instalar o XDebug pode ser complicado, mas uma das características mais importantes é a "Depuração Remota" - se você desenvolve seu código localmente e depois o testa dentro de uma VM (máquina virtual) ou em outro servidor, a "Depuração Remota" é uma característica que você vai querer manter ativa desde o início.

Tradicionalmente, você modificará o VHost ou o .htaccess do seu Apache para incluir os seguintes valores:

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

O “remote host” e o “remote port” vão corresponder ao seu computador local e a porta que você configurar para ser escutada na sua IDE. É apenas uma questão de colocar a sua IDE em modo para “escutar por conexões”, e carregar a URL:

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

Sua IDE agora irá interceptar o estado atual enquanto seu script é executado, permitindo a você definir *breakpoints* e inspecionar os valores na memória.

Depuradores gráficos deixam muito fácil o processo de percorrer o código, inspecionar variáveis e avaliar o código em tempo de execução. Várias IDE’s possuem incluso ou suportam um plugin para depurar graficamente com o XDebug. MacGDBp é gratuito tem o código fonte aberto uma GUI (Interface Gráfica do Usuário) stand-alone do XDebug para Mac.

- Aprenda sobre o XDebug
- Aprenda sobre o MacGDBp

Gerenciamento de Dependência

Existem toneladas de bibliotecas PHP, frameworks, e componentes para você escolher. Seu projeto provavelmente irá usar muitos deles — eles são as dependências do projeto. Até recentemente, o PHP não possuía uma boa forma de gerenciar essas dependências de projeto. Mesmo se você as gerenciasse manualmente, ainda assim teria que se preocupar com autoloaders. Não mais.

Atualmente existem dois sistemas principais para gerenciamento de pacotes no PHP - o Composer e o PEAR. Qual deles é o certo para você? A resposta é: ambos.

- Use o **Composer** quando estiver gerenciando as dependências de um único projeto.
- Use o **PEAR** quando gerenciar dependências do PHP para o seu sistema inteiro.

Em geral, os pacotes Composer estarão disponíveis apenas em projetos que você especificar de forma explícita, enquanto que um pacote PEAR estará disponível para todos os seus projetos PHP. Mesmo que o PEAR pareça ser o método mais fácil à primeira vista, existem vantagens em usar uma abordagem projeto-por-projeto para suas dependências.

Composer e Packagist

O Composer é o gerenciador de dependências recomendado para PHP. Liste as dependências do seu projeto em um arquivo `composer.json` e, com poucos comandos simples, o Composer irá fazer o download das dependências do seu projeto automaticamente e configurará o autoloading para você. O Composer é semelhante ao NPM no mundo node.js ou o Bunder no mundo Ruby.

Já existem várias bibliotecas PHP que são compatíveis com o Composer, prontas para usar no seu projeto. Esses “pacotes” estão listados no Packagist, o repositório oficial das bibliotecas PHP compatíveis com o Composer.

Como Instalar o Composer

A forma mais segura é fazer o download do composer seguindo as instruções oficiais. Isso vai verificar se o instalador não está corrompido ou foi adulterado. O instalador instala um arquivo binário `composer.phar` no seu *diretório atual*.

Nós recomendadmos instalar o Composer *globally* (por exemplo uma única cópia no diretório `/usr/local/bin`). Para fazer isso, rode o próximo comando:

```
mv composer.phar /usr/local/bin/composer
```

Nota: Se o comando falhar devido a permissão, use o comando `sudo` no início.

Para rodar o Composer instalado localmente, use o comando `php composer.phar`, e globalmente apenas `composer`.

Instalação no Windows

Para usuários Windows a forma mais fácil de obter e executá-lo é usar o instalador ComposerSetup, que realiza uma instalação global e configura seu `$PATH` de modo que você possa executar o comando `composer` de qualquer diretório pela linha de comando.

Como Definir e Instalar Dependências

O Composer mantém o controle de dependências do seu projeto em um arquivo chamado `composer.json`. Você pode controlá-lo na mão se preferir ou usar o próprio Composer. O comando `composer require` adiciona uma dependência do projeto e se você não tem um arquivo `composer.json`, ele será criado. Aqui está um exemplo que adiciona o Twig como uma dependência do seu projeto.

```
composer require twig/twig:^2.0
```

Outra alternativa é o comando `composer init` que guiará a criação completa do arquivo `composer.json` para seu projeto. De qualquer maneira, uma vez criado o arquivo `composer.json` você pode chamar o Composer para baixar suas dependências para o diretório `vendor/`. Isto também se aplica para projetos baixados que fornecem um arquivo `composer.json`:

```
composer install
```

Em seguida, adicione esta linha ao arquivo PHP principal da sua aplicação; isso dirá ao PHP para usar o autoloader do Composer para as dependências do seu projeto.

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

Agora você pode usar as dependências do seu projeto, e elas serão carregadas automaticamente sob demanda.

Atualizando suas dependências

O Composer cria um arquivo chamado `composer.lock` que armazena a versão exata de cada pacote baixado quando você executou `composer install`. Se você compartilhar seu projeto com outras pessoas, certifique-se que o o arquivo `composer.lock` está incluído, pois quando eles executarem o comando `composer install` eles receberão as mesmas versões que você possui. Para atualizar suas dependências, execute o comando `composer update`. Não use o comando `composer update` quando estiver fazendo deploy, use apenas `composer install`, senão você poderá terminar com versões de pacotes diferentes em produção.

Isso é muito útil quando você define as versões requeridascom flexibilidade. Por exemplo, uma versão requerida de `~1.8` significa “qualquer versão mais recente que `1.8.0`, mas menos recente do que `2.0.x-dev`”. Você também pode usar o curinga `*` como `1.8.*`. Agora o comando `composer update` atualizará todas as suas dependências para a versão mais recente que se encaixa às restrições definidas.

Notificações de Atualização

Para receber notificações sobre novas versões você pode se inscrever no libraries.io, um serviço web que pode monitorar dependências e lhe enviar alertas de atualizações.

Tratando dependências globais com Composer

O Composer também pode tratar dependências globais e seus binários. O seu uso é direto, tudo que você precisa é prefixar seu comando com a palavra `global`. Se por exemplo você quer instalar o PHPUnit e quer tê-lo disponível globalmente, você deve rodar o seguinte comando:

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

Isso irá criar uma pasta `~/.composer` onde suas dependências globais residem. Para ter os pacotes binários disponíveis em qualquer lugar, você deve adicionar a pasta `~/.composer/vendor/bin` para sua variável `$PATH`.

- Aprenda sobre o Composer

PEAR

Outro gerenciador de pacotes veterano e que muitos desenvolvedores PHP gostam é o PEAR. Ele se comporta da mesma maneira que o Composer, mas possui diferenças notáveis.

PEAR requer que cada pacote tenha uma estrutura específica, isso significa que o autor do pacote deve prepará-lo para ser usado com PEAR. Não é possível usar um projeto que não foi preparado para o PEAR.

PEAR instala pacotes de forma global, ou seja, uma vez instalados ficam disponíveis para todos os projetos no servidor. Isto pode ser bom se muitos projetos dependem dos mesmos pacotes com as mesmas versões, mas isso pode gerar problemas se houver conflitos de versões entre os projetos.

Como instalar o PEAR

Você pode instalar o PEAR baixando o instalador phar e executando-o. A documentação do PEAR tem instruções de instalação mais detalhadas para todos sistemas operacionais.

Se você usa Linux, pode conferir no gerenciador de pacotes da sua distribuição. Debian e Ubuntu, por exemplo, tem um pacote `php-pear`.

Como instalar um pacote

Se o pacote está na lista de pacotes do PEAR, você pode instalá-lo informando seu nome oficial:

```
pear install foo
```

Se o pacote está hospedado em outro canal, você precisa, primeiro, descobrí-lo (`discover`) e especificá-lo durante a instalação. Veja a Documentação Usando Canais para mais informações sobre este tópico.

- Aprenda sobre PEAR

Manuseio de dependências PEAR com Composer

Se você já está usando Composer e também gostaria de instalar algum código PEAR, você pode usar o Composer para manusear suas dependências PEAR. Este exemplo instalará um código a partir do `pear2.php.net`:

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

A primeira seção `"repositories"` será usada para o Composer saber que deve “inicializar” (ou “descobrir” a terminologia PEAR) o repositório pear. Em seguida, na seção `"require"` terá `pear` como prefixo no nome do pacote, como:

> pear-channel/Package

O prefixo “pear” é padrão para evitar qualquer conflito, por exemplo, um canal pear pode ter o mesmo nome de um pacote no vendor. Então, o nome curto do canal (ou a URL completa) pode ser usada para referenciar o canal em que o pacote se encontra.

Quando este código for instalado, ficará disponível dentro do seu diretório vendor e disponível automaticamente através do autoloader do Composer:

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

Para usar este pacote PEAR, basta referenciá-lo assim:

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

- Aprenda mais sobre o uso do PEAR com Composer

Práticas de Codificação

O Básico

PHP é uma grande linguagem que permite a programadores de todos os níveis produzirem código, não apenas rapidamente, mas eficientemente. Entretanto enquanto se avança na linguagem, nós frequentemente esquecemos do básico que tínhamos aprendido no começo (ou passado o olho) em prol de atalhos e/ou maus hábitos. Para ajudar a combater esse problema comum, esse seção é focada em lembrar aos programadores das práticas básicas de codificação no PHP.

- Continue lendo O Básico

Data e Horário

O PHP tem uma classe chamada DateTime para ajudar você com leitura, escrita, comparações e cálculos com datas e horários. Existem muitas funções no PHP relacionadas a datas e horários além da DateTime, mas ela fornece uma boa interface orientada a objetos para a maioria dos usos comuns. Ela pode tratar de fusos horários, mas isso vai além dessa breve introdução.

Para começar a trabalhar com a DateTime, converta uma string bruta de data e hora para um objeto com o método *factory* `createFromFormat()`, ou use `new DateTime` para obter a data e a hora atual. Use o método `format()` para converter um objeto DateTime de volta para uma string para saída.

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

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

Cálculos com a DateTime são possíveis com a classe DateInterval. A DateTime tem métodos como o `add()` e o `sub()` que recebem um DateInterval como argumento. Não escreva código que espera o mesmo número de segundos para todos os dia, pois tanto as alterações de horário de verão quanto as de fuso horário irão quebrar essa suposição. Em vez disso use intervalos de data. Para calcular diferenças entre datas use o método `diff()`. Ele retornará um novo DateInterval, que é bem fácil de mostrar.

```
<?php
// cria uma cópia de $start e adiciona um mês e 6 dias
$end = clone $start;
$end->add(new DateInterval('P1M6D'));

$diff = $end->diff($start);
echo "Difference: " . $diff->format('%m month, %d days (total: %a days)') . "\
n";
```

```
// Diferença: 1 mês, 6 dias (total: 37 dias)
```

Com objetos DateTime, você pode usar a comparação padrão:

```
<?php
if($start < $end) {
    echo "Start is before end!\n";
}
```

Um último exemplo para demonstrar a classe DatePeriod. Ela é usada para iterar por eventos recorrentes. Ela pode receber dois objetos DateTime, um início e um fim, e o intervalo para o qual ele retornará todos os eventos intermediários.

```
<?php
// mostra todas as quintas-feiras entre $start e $end
$periodInterval = DateInterval::createFromDateString('first thursday');
$periodIterator = new DatePeriod($start, $periodInterval, $end,
DatePeriod::EXCLUDE_START_DATE);
foreach($periodIterator as $date) {
    //mostra cada data no período
    echo $date->format('Y-m-d') . " ";
}
```

- Leia sobre a classe DateTime
- Leia sobre formatação de datas (opções aceitas para formatos de strings de data)

Design Patterns

Quando você está construindo sua aplicação web é muito útil utilizar padrões de codificação para formar a estrutura do seu projeto. Usar “design patterns” (padrões de projeto) é útil pois eles facilitam bastante na hora de gerenciar seu código e permite que outros desenvolvedores entendam rapidamente como tudo está se encaixando.

Se você utiliza um framework então a maior parte do código de alto nível e a estrutura do projeto serão baseados no framework, ou seja, uma grande parte das decisões de padrões de design do código já foram decididas para você. Mas ainda cabe a você escolher os melhores padrões a seguir no código na hora de utilizá-los no framework. Se, por outro lado, você não estiver utilizando uma framework para construir sua aplicação, então você terá que descobrir os padrões que melhor se encaixam para o tipo e tamanho da aplicação que você está construindo.

- Continue lendo em Design Patterns

Trabalhando com UTF-8

Esta seção foi originalmente escrita por Alex Cabal como PHP Melhores Práticas e tem sido usado como base para os nossos próprios conselhos sobre UTF-8.

Não existe um jeito fácil. Seja cuidadoso, detalhado e consistente.

Atualmente o PHP não possui suporte a Unicode em um nível baixo. Existem maneiras de garantir que strings UTF-8 sejam processadas OK, mas não é fácil e isto requer cavar quase todos os níveis da aplicação web desde o HTML passando pelo SQL, até o PHP. Vamos abordar para um resumo breve e prático.

UTF-8 no nível do PHP

As operações básicas com strings, como concatenar duas strings e atribuição de strings a variáveis, não preciso de nada especial para o UTF-8. No entanto a maioria das funções de strings, como `strpos()` e `strlen()`, precisam de atenção especial. Essas funções têm, frequentemente, uma `mb_*` em contrapartida: por exemplo `mb_strpos()` e `mb_strlen()`. Estas funções de string `mb_*` são disponibilizados a você por meio do Multibyte Extensão String e são projetadas especificamente para operar em strings de caracteres Unicode.

Você deve usar as funções `mb_*` sempre que operar com strings Unicode. Por exemplo, se você usar `substr()` em uma string UTF-8, há uma boa chance de que o resultado terá alguns caracteres ilegíveis. A função correta de usar seria a contrapartida multibyte, `mb_substr()`.

A parte mais difícil é lembrar de usar as funções `mb_*` em todos os momentos. Se você esquecer mesmo que apenas uma vez, sua string Unicode tem uma chance de ser ilegível durante o processamento.

Nem todas as funções de string têm um ‘mb_*` contrapartida. Se não houver uma para o que você quer fazer, então você pode estar sem sorte.

Você deve usar a função `mb_internal_encoding()` no topo de cada script PHP que você escreve (ou no topo de seu script global que seja incluído), e a função `mb_http_output()` logo após ele se seu script está enviando saída para um navegador. Definir explicitamente a codificação de suas strings em cada script vai lhe poupar muita dor de cabeça futuramente.

Além disso, muitas funções PHP que operam em cadeias de caracteres têm um parâmetro opcional que lhe permite especificar o caractere codificação. Você deve sempre indicar explicitamente UTF-8 quando for dada a opção. Por exemplo, `htmlentities()` tem uma opção para codificação de caracteres, e você deve sempre especificar UTF-8 se lidar com tais cordas. Note-se que a partir do PHP 5.4.0, UTF-8 é a codificação padrão para `htmlentities()` e `htmlspecialchars()`.

Finalmente, se você estiver criando um aplicativo distribuído e não tiver certeza de que a extensão `mbstring` será ativada, então considere o uso do pacote Composer patchwork/utf8. Isto irá usar a `mbstring` se estiver disponível, e criar fall back para funções UTF-8 que não estiverem.

UTF-8 no nível de banco de dados

Se o seu script PHP acessa o MySQL, há uma chance de suas strings serem armazenadas como strings não-UTF-8 no banco de dados, mesmo que você siga todas as precauções acima.

Para certificar-se de que suas strings irão do PHP para o MySQL como UTF-8, verifique se o banco de dados e as tabelas estão todos setados com o character set e o collation como `utf8mb4` e que você use o character set `utf8mb4` na string de conexão PDO. Veja o exemplo de código abaixo. Isto é *criticamente importante*.

Observe que você deve usar o character set `utf8mb4` para ter suporte completo de UTF-8 e não o character set `utf8`! Continue lendo para o porquê.

UTF-8 no nível do navegador

Use a função `mb_http_output()` para garantir que o seu script PHP gere strings UTF-8 para o seu browser.

O navegador então será avisado pela resposta HTTP que esta página deve ser considerada como UTF-8. A abordagem histórica para fazer isso foi a inclusão da tag `<meta>` charset na tag `<head>` da sua página. Esta abordagem é perfeitamente válida, mas definir o charset no cabeçalho `Content-type` é realmente muito mais rápido.

```
<?php
// Diz para o PHP que estamos usando strings UTF-8 até o final do script
mb_internal_encoding('UTF-8');

// Diz para o PHP que nós vamos enviar uma saída UTF-8 para o navegador
mb_http_output('UTF-8');

// A nossa string UTF-8 de teste
$string = 'Êl síla erin lû e-govaned vîn.';

// Transformar a seqüência de alguma forma com uma função multibyte
// Observe como cortamos a string em um caractere não-ASCII para fins de
demonstração
$string = mb_substr($string, 0, 15);

// Conectar a um banco de dados para armazenar a string transformada
// Veja o exemplo PDO neste documento para obter mais informações
// Observe os comandos `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
    )
);

// Armazena a nossa string transformada como UTF-8 em nosso banco de dados
// Seu DB e tabelas estão com character set e collation utf8mb4, certo?
$handle = $link->prepare('insert into ElvishSentences (Id, Body) values
(?, ?)');
$handle->bindValue(1, 1, PDO::PARAM_INT);
$handle->bindValue(2, $string);
$handle->execute();

// Recuperar a string que armazenamos apenas para provar se foi armazenada
corretamente
$handle = $link->prepare('select * from ElvishSentences where Id = ?');
$handle->bindValue(1, 1, PDO::PARAM_INT);
$handle->execute();

// Armazena o resultado em um objeto que vamos saída mais tarde em nossa HTML
$result = $handle->fetchAll(PDO::FETCH_OBJ);

header('Content-Type: text/html; charset=UTF-8');
?><!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>UTF-8 test page</title>
    </head>
```

```
    <body>
        <?php
        foreach($result as $row){
            print($row->Body);  // Isto deve emitir corretamente nossa string
transformada como UTF-8 para o navegador
         }
        ?>
    </body>
</html>
```

Leitura adicional

- Manual do PHP: Operações Strings
- Manual do PHP: Funções para Strings
    - `strpos()`
    - `strlen()`
    - `substr()`
- Manual do PHP: Funções multibyte de strings
    - `mb_strpos()`
    - `mb_strlen()`
    - `mb_substr()`
    - `mb_internal_encoding()`
    - `mb_http_output()`
    - `htmlentities()`
    - `htmlspecialchars()`
- Dicas PHP e UTF-8
- Manuseando UTF-8 com o PHP
- Stack Overflow: Quais os fatores que fazem o PHP incompatível com Unicode?
- Stack Overflow: Melhores práticas em PHP e MySQL com strings internacionais
- Como ter suporte total a Unicode em bases de dados MySQL
- Trazendo Unicode para o PHP com `Portable UTF-8`

Dependency Injection (Injeção de Dependência)

Fonte Wikipedia:

> Dependency Injection é um Design Pattern que permite retirar as dependências hard-coded e torna possível mudá-las, seja em tempo de execução ou em tempo de compilação.

Esta citação torna o conceito muito mais complicado do que realmente é. Dependency Injection fornece um componente com suas dependências, seja por injeção no construtor, chamada de método ou na definição de propriedades. É simples assim.

Conceito Básico

Demostraremos o conceito com um simples exemplo.

Temos uma classe `Database` que requer um adaptador para se comunicar com o banco de dados. Instanciaremos o adaptador no construtor e assim criamos uma forte de dependência. Isto dificulta os testes e significa que a classe `Database` está fortemente acoplada ao adaptador.

```
<?php
namespace Database;

class Database
{
    protected $adapter;

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

class MysqlAdapter {}
```

Este código pode ser refatorado para usar a Dependency Injection para desacoplar a dependência.

```
<?php
namespace Database;

class Database
{
    protected $adapter;

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

class MysqlAdapter {}
```

Agora, damos a classe `Database` a sua dependência em vez de criar dentro dela. Poderiamos também, criar um método que aceitaria um argumento da dependência e defini-la dessa forma, ou definir a propriedade `$adapter` como `public` para defini-la diretamente.

Problema Complexo

Se você já leu sobre Inversão de Dependência, então você provavelmente viu os termos *"Inversão de Controle"* ou *"Princípio da Inversão de Dependência"*. Estes são os problemas complexos que a Inversão de Dependência resolve.

Inversão de Controle

Inversão de controle é como se diz, "invertendo o controle" de um sistema para manter os controles organizacionais totalmente separados dos seus objetos. Em termos de Dependency Injection, isto significa desacoplar as dependências para controlá-las e instanciá-las em outro lugar do sistema.

Por anos, os frameworks PHP usam a Inversão de Controle, no entanto, a questão é: que parte do controle está invertendo, e onde? Por exemplo, frameworks MVC, em geral, fornecem um super objeto ou um controlador base que outros controladores devem extender para obter acesso as suas dependências. Isto **é** Inversão de Controle, no entanto, em vez de desacoplar as dependências, este método simplesmente as mudou.

Dependency Injection permite resolver de forma mais elegante este problema apenas injetando a dependência que precisamos, quando precisamos dela, sem a necessidade de quaisquer dependências no código.

Princípio da Inversão de Dependência

O Princípio da Inversão de Dependência é o "D", no S.O.L.I.D, define o princípio de design da orientação a objeto que afirma que *"Dependa de uma Abstração. Não depende de Objetos concretos"*. Simplificando, isto significa que nossas dependências devem ser classes de interfaces/contratos ou class abstratas em vez de implementações concretas. Podemos facilmente refatorar o exemplo abaixo para seguir este princípio.

```
<?php
namespace Database;

class Database
{
    protected $adapter;

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

interface AdapterInterface {}

class MysqlAdapter implements AdapterInterface {}
```

Existem vários benefícios para a classe `Database`, agora, dependendo de uma interface em vez de um objeto concreto.

Considerando que você trabalha em uma equipe e o adaptador está sendo trabalhado por um colega. No primeiro exemplo, teriamos que esperar o colega dizer que terminou o adaptador antes que pudéssemos simulá-la(mock) nos testes unitários. Agora, que a dependência é uma interface/contrato podemos facilmente simulá-la(mock) sabendo que o colega vai construir o adaptador com base neste contrato.

Um benefício ainda maior para este método é que nosso código, agora, está mais escalável. Se um ano depois decidir migrar para um banco de dados diferente, podemos escrever um novo adaptador que implementa a interface original e injetá-lo, não seria preciso nenhuma refatoração, pois o adaptador segue o contrato definido pela interface.

Containers

A primeira coisa que você deve entender sobre os Containers da Dependency Injection é que eles não são a mesma coisa que ela. Um container é um utilitário de conveniência que nos ajuda

implementar a Dedepency Injection, no entanto, eles podem ser, e muitas vezes são, uma implementação anti-pattern, Service Location (Serviço de Localização). Injetar um container DI como um Localizador de Serviço para suas classes, sem dúvida, cria uma dependência difícil de substituir no container. Isso também torna seu código menos transparente e, finalmente, mais difícil de testar.

A maioria dos frameworks têm seu próprio Container de Dependency Injection que permite ligar suas dependências em conjunto, através de configuração. O que isto significa, na prática, que você pode escrever o código da aplicação tão limpo e desacoplado como do framework foi construído.

Leitura Adicional

- Aprenda sobre Dependency Injection e PHP
- O que é Dependency Injection?
- Dependency Injection: uma analogia
- Dependency Injection: Huh?
- Dependency Injection como uma ferramenta para testes

Bancos de Dados

Muitas vezes o seu código PHP usará um banco de dados para persistir informações. Você tem algumas opções para conectar e interagir com o seu banco de dados. A opção recomendada *até o PHP 5.1.0* era usar os drivers nativos, como o mysqli, o pgsql, mssql etc.

Drivers nativos são excelentes se você estiver usando apenas *um* banco de dados em sua aplicação, mas se, por exemplo, você estiver usando o MySQL e um pouco de MSSQL, ou você precisar conectar em um banco de dados Oracle, aí você não poderá usar os mesmos drivers. Você precisará aprender um API totalmente nova para cada um dos bancos de dados — e isso pode ficar chato.

MySQL Extension

A extensão mysql para o PHP não está mais em desenvolvimento ativo e foi oficialmente descontinuada no PHP 5.5.0. Isso significa que ela será removida dentro de alguns lançamentos das próximas versões. Se você estiver usando funções que inicial com `mysql_*` como a `mysql_connect()` e a `mysql_query()` em suas aplicações você irá se deparar com uma reescrita em algum momento no futuro, por isso a melhor opção é substituir o uso do mysql pelo mysqli ou pela PDO na sua aplicação dentro de sua própria agenda de desenvolvimento, assim você não terá que correr lá na frente.

Se você estiver começando do zero então não utilize de forma nenhuma a extensão mysql: use a extensão MySQLi, ou use a PDO.

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

Extensão PDO

A PDO é uma biblioteca para abstração de conexões a bancos de dados — embutida no PHP desde a versão 5.1.0 — que fornece uma interface comum para conversar com vários bancos de dados diferentes. Por exemplo, você pode usar basicamente o mesmo código para fazer a interface com o MySQL ou SQLite:

```
<?php
// PDO + MySQL
$pdo = new PDO('mysql:host=example.com;dbname=database', 'user', 'password');
$statement = $pdo->query("SELECT some\_field FROM some\_table");
$row = $statement->fetch(PDO::FETCH_ASSOC);
echo htmlentities($row['some_field']);

// PDO + SQLite
$pdo = new PDO('sqlite:/path/db/foo.sqlite');
$statement = $pdo->query("SELECT some\_field FROM some\_table");
$row = $statement->fetch(PDO::FETCH_ASSOC);
echo htmlentities($row['some_field']);
```

A PDO não irá traduzir suas consultas SQL ou emular funcionalidades que não existem; ela é feita puramente para conectar múltiplos tipos de bancos de dados com a mesma API.

Mais importante, a `PDO` permite que você injete de forma segura entradas externas (e.g IDs) em suas consultas SQL sem se preocupar com ataques de SQL injection. Isso é possível usando instruções PDO (PDO statements) e parâmetros restritos (bound parameters).

Vamos assumir que um script PHP recebe um ID numérico como parâmetro de uma consulta. Este ID deve ser usado para buscar um registro de um usuário no banco de dados. Essa é a forma `errada` de fazer isso:

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

Esse código é péssimo. Você está inserindo um parâmetro bruto na sua consulta SQL. Isso fará você ser hackeado num piscar de olhos, usando uma prática chamada SQL Injection. Apenas imagine se um hacker passar como parâmetro um `id` inventivo chamando uma URL como `http://domain.com/?id=1%3BDELETE+FROM+users`. Isso irá definir a variável `$_GET['id']` como `id=1;DELETE FROM users`, o que irá excluir todos os seus usuários. Em vez disso, você deveria higienizar (sanitize) a entrada do ID usando parâmetros restritos da PDO.

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

Esse é o código correto. Ele usa um parâmetro restrito em uma instrução PDO. Assim a entrada externa do ID é escapada antes de ser introduzida no banco de dados, prevenindo contra potenciais ataques de SQL injection.

- Leia mais sobre a PDO

Você também deve estar ciente de que usam recursos do servidor e não é raro ter esses recursos esgotados se essas conexões não forem implicitamente fechadas, entretanto isso é mais comum em outras linguagens. Com PDO você pode implicitamente fechar as conexões pela destruição dos objetos garantindo que todas as referências a ele forem excluídas, ex. atribuindo NULL a elas. Se você não fizer isso explicitamente o PHP irá fechar todas as conexões quando seu script terminar - a não ser é claro que você esteja usando conexões persistentes.

- Leia mais sobre conexões PDO

Interagindo com o banco de dados

Quando os desenvolvedores começam a aprender PHP, muitas vezes acabam misturando a interação com o banco de dados com a camada de apresentação, usando código que pode parecer com isso:

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

Esta é uma má prática por várias razões, principalmente por ser difícil de depurar, testar, ler e ainda pode gerar na saída um monte de campos se não houver um limite.

Embora existam muitas outras soluções para fazer isso - dependendo se você preferir a OOP ou programação funcional - deve haver algum elemento de separação.

Considere o passo mais básico:

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

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

Este é um bom começo. Coloque estes dois itens em dois arquivos diferentes e você terá alguma separação limpa.

Crie uma classe para colocar este método e você terá um “Modelo”. Criando um arquivo `.php` simples para colocar a lógica de apresentação e você terá uma “View”, que é quase um MVC - uma arquitura OOP comum para a maioria dos frameworks.

foo.php

```
<?php
$db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8', 'username',
'password');

// Deixe seu modelo disponível
include 'models/FooModel.php';

// Crie uma instância
$fooList = new FooModel($db);
```

```
// Mostre a view
include 'views/foo-list.php';
```

models/FooModel.php

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

Isto é essenciamente o mesmo que a maioria dos frameworks modernos fazem, todos sejam eles um pouco mais manual. Você pode não precisar de tudo a todo momento, mas misturando muita lógica de apresentação e interação com o banco de dados pode ser um problema real se você quiser testes unitários em sua aplicação.

PHPBridge tem um grande recurso chamado Criando uma classe de dados que aborda um tópico muito similar e é ótimo para os desenvolvedores se acostumar ao o conceito de interagir com o banco de dados.

Camadas de Abstração

Muitos frameworks fornecem sua própria camada de abstração que pode ou não sobrepor o PDO. Estes muitas vezes simulam características de um sistema de banco de dados que outro banco de dados não possui envolvendo suas consultas em métodos PHP, dando-lhe a abstração real do banco de dados em vez de apenas a abstração da conexão como o PDO oferece.

Isto obviamente adiciona um pequeno peso, mas se você estiver construindo uma aplicação portátil que precise trabalhar com MySQL, PostgreSQL e SQLite, então este pequeno peso vai valer a pena pela limpeza e menos linhas de código.

Algumas camadas de abstração foram construídas utilizando o padrão de namespaces da PSR-0 ou PSR-4 para que possa ser instalado em qualquer aplicação que você queira.

- Aura SQL
- Doctrine2 DBAL
- Propel
- ZF2 Db

Templates

Os templates fornecem uma forma conveniente de separar seu controlador e a camada de domínio da sua camada de apresentação.

Eles contém geralmente o HTML de sua aplicação, mas também podem ser usados para outros formatos, como o XML.

São muitas vezes referidos como a camada de visão que faz parte do segundo componente do padrão de arquitetura de software modelo-visão-controlador (MVC)

Benefícios

O principal benefício de se utilizar templates é a clara separação que eles criam entre a lógica de apresentação e o resto da sua aplicação. Templates têm a responsabilidade exclusiva de exibir o conteúdo formatado. Eles não são responsáveis por pesquisa de dados, persistência ou outras tarefas mais complexas. Isto leva a código mais limpo, mais legível que é especialmente útil em um ambiente de equipe, onde os desenvolvedores trabalham no código do lado do servidor (controladores, modelos) e designers trabalham no código do lado do cliente (markup).

Os templates também melhoram a organização do código de apresentação. Os templates são normalmente colocados em uma pasta “views”, cada um definido dentro de um único arquivo. Essa abordagem incentiva a reutilização de código, onde grandes blocos de código são quebrados em pequenos pedaços reutilizáveis, chamados frequentemente de partials. Por exemplo, o cabeçalho e o rodapé do site de cada um pode ser definido como templates, que são então incluídos antes e depois de cada template de página.

Finalmente, dependendo da biblioteca que você usa, os templates podem oferecer mais segurança ao escapar automaticamente o conteúdo gerado pelo usuário. Algumas bibliotecas oferecem até mesmo sand-boxing, onde os criadores de templates só têm acesso à white-listed (lista branca) de variáveis e funções.

Templates Simples em PHP

Templates Simples em PHP são templates que usam código nativo do PHP. Eles são uma escolha natural já que o PHP é na realidade um linguagem de template por si só. Isso significa simplesmente que você pode combinar código PHP dentro de outro código, como HTML. Isso é benéfico para os desenvolvedores de PHP pois não há uma nova sintaxe para aprender, eles sabem as funções disponíveis e seus editores de código PHP já tem syntax highlighting and auto-completion embutidos. Além disso, estes templates tendem a ser muito mais rápido já que não é necessária a fase de compilação.

Cada framework moderno PHP emprega algum tipo de sistema de templates, a maioria usam PHP simples por padrão. Fora dos frameworks, bibliotecas como Plates ou Aura.View tornam o trabalho mais fácil, oferecendo funcionalidade modernas ao template, tais como herança, layouts e extensões.

Exemplo de um template simples em PHP

Utilizando a biblioteca Plates.

```
<?php // user_profile.php ?>

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

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

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

Exemplo de um template simples em PHP usando herança

Utilizando a biblioteca Plates.

```
<?php // template.php ?>

<html>
<head>
    <title><?=$title?></title>
</head>
<body>

<main>
    <?=$this->section('content')?>
</main>

</body>
</html>

<?php // user_profile.php ?>

<?php $this->layout('template', ['title' => 'User Profile']) ?>

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

Templates Compilados

Enquanto o PHP evoluiu para uma linguagem orientada a objetos madura, ele não melhorou muito como uma linguagem de templates. Templates compilados, como Twig ou Smarty*, preenchem este vazio oferecendo uma nova sintaxe que foi direcionada especificamente para templating. De escape automático, à herança e estruturas de controle simplificadas, templates compilados são projetados para ser mais fácil de escrever, simples de ler e mais seguro de usar. Templates compilados pode ser até compartilhados entre diferentes linguagens, Mustache vem sendo um bom exemplo disso. Uma vez que esses templates devem ser compilados há uma pequena queda de performance, no entanto, este é mínimo quando o cache apropriado é usado.

** Enquanto Smarty oferece escape automático, este recurso NÃO está habilitado por padrão.*

Exemplo simples de um template compilado

Utilizando a biblioteca Twig.

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

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

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

Exemplo de templates compilados usando herança

Utilizando a biblioteca Twig.

```
// template.php

<html>
<head>
    <title>{% block title %}{% endblock %}</title>
</head>
<body>

<main>
    {% block content %}{% endblock %}
</main>

</body>
</html>

// user_profile.php

{% extends "template.html" %}

{% block title %}User Profile{% endblock %}
{% block content %}
    <h1>User Profile</h1>
    <p>Hello, {{ name }}</p>
{% endblock %}
```

Templates - Leitura Adicional

Artigos & Tutoriais

- 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

Bibliotecas

- Aura.View *(nativo)*
- Blade *(compilado, específico do framework)*
- Dwoo *(compilado)*
- Latte *(compilado)*
- Mustache *(compilado)*
- PHPTAL *(compilado)*
- Plates *(nativo)*
- Smarty *(compilado)*

- Twig *(compilado)*
- Zend\View *(nativo, específico do framework)*

Erros e Exceções

Erros

Em muitas linguagens de programação que fazem o uso generalizado das exceções, sempre que algo dá errado uma exceção é lançada. Esta é certamente uma forma viável de fazer as coisas, mas o PHP é uma linguagem que utiliza menos exceções. Mesmo que elas existam e mais membros do núcleo de desenvolvimento estejam começando a usá-las quando trabalhando com objetos, o PHP irá na maioria das vezes tentar manter a linha de processamento independentemente do que aconteça, a não ser que ocorra um erro fatal.

Por exemplo:

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

Este é apenas um `notice error` e o PHP irá continuar a excecução. Isso pode ser confuso para quem vem de linguagens "exception-heavy", porque referência a uma variável que falta em Python, por exemplo, irá lançar uma exceção:

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

A única diferença real é que Python vai surtar sobre qualquer coisa pequena, de modo que os desenvolvedores podem ter certeza que qualquer problema em potencial ou caso extremo será capturado, enquanto o PHP irá continuar o processamento, a menos que algo extremo aconteça e neste ponto irá lançar um erro e relatá-lo.

Severidade dos Erros

O PHP tem vários níveis de severidade de erro. Os três tipos mais comuns de mensagens são erros, avisos e advertências (error, notice e warnings). Estes têm diferentes níveis de severidade; `E_ERROR`, `E_NOTICE` e `E_WARNING`. *Erros* são erros fatais em tempo de execução e são geralmente causados por falhas no seu código e precisam ser corrigidos à medida que eles causam a parada da execução do PHP. Os avisos são erros não fatais, a execução do script não será interrompida. Avisos são mensagens de conselho causadas por um código que pode ou não causar problemas durante a execução do script, a execução não é interrompida.

Outro tipo de mensagem de erro relatado em tempo de compilação são mensagens `E_STRICT`. Estas mensagens são usadas para sugerir mudanças no seu código para ajudar a assegurar melhor interoperabilidade e compatibilidade com futuras versões do PHP.

Mudando o comportamento do relatório de erros do PHP

O relatório de erros pode ser alterado nas configurações do PHP e/ou através de chamadas de função. Usando a função nativa do PHP `error_reporting()` você pode definir o nível dos erros para a duração da execução do script, passando um dos níveis de erro pré-definidos, ou seja, se você só quer ver os *Warnings* e os *Errors* - mas não *Notices* - então você pode configurar como:

```
7
```

Você também pode controlar ou não os erros que são exibidos na tela (bom para o desenvolvimento) ou ocultados e registrados (bom para produção). Para mais informações, verifique a seção Error Reporting.

Supressão de erros

Você também pode dizer ao PHP para suprimir erros específicos com o operador de controle de erro `@`. Você coloca este operador no início de uma expressão e qualquer erro resultado pela expressão será suprimido.

```
<?php
echo @$foo['bar'];
```

A saída será `$foo['bar']` se existir, mas vai simplesmente devolver um null e não mostrar nada, se a variável `$foo` ou chave `'bar'` não existir. Sem o operador de controle de erros, essa expressão poderia criar um erro 'PHP Notice: Undefined variable: foo `ou` PHP Notice: Undefined index: bar`.

Isto pode parecer uma boa idéia, mas há alguns efeitos indesejáveis. O PHP lida com expressões usando o '@' de uma forma menos eficaz do que expressões sem o '@'. Otimização prematura pode ser a raiz de todos os argumentos de programação, mas se o desempenho é particularmente importante para o seu aplicativo/biblioteca é importante entender as implicações de desempenho do operador de controle de erro.

Em segundo lugar, o operador de controle de erro engole **completamente** o erro. O erro não é exibido e não é enviado para o log de erro. Além disso, os sistemas de produção PHP não possuem nenhuma maneira de desligar o operador de controle de erro. Enquanto você pode estar certo que o erro que você está vendo é inofensivo, um erro diferente e menos inofensivo será silenciado.

Se há uma maneira de evitar o uso do operador de supressão de erro então você deve considerar isso. Por exemplo, nosso código acima poderia ser reescrito da seguinte forma:

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

Um exemplo em que a supressão de erro pode fazer sentido é onde a função `fopen()` falha em não conseguir encontrar o arquivo para carregar. Você pode verificar a existência do arquivo antes de tentar carregá-lo, mas se o arquivo for apagado após a verificação e antes da `fopen()` (que pode parecer impossível, mas pode acontecer), então a `fopen()` retornará false *e* lançará um erro. Isso é potencialmente algo que o PHP deverá resolver, mas é um caso em que a supressão de erro pode parecer a única solução válida.

Anteriormente mencionamos não há nenhuma maneira para desligar o operador de controle de erro. No entanto o xDebug tem uma configuração `xdebug.scream` que irá desativar o operador de controle de erro. Você pode definir essa opção seu arquivo `php.ini` com o seguinte.

```
xdebug.scream = On
```

Você também pode definir esse valor em tempo de execução com a função `ini_set`

```
<?php
ini_set('xdebug.scream', '1')
```

A extensão PHP "Scream" oferece funcionalidade semelhante à do xDebug, embora a configuração do Scream seja chamada `scream.enabled`.

Isso é muito útil quando você está a depuração do código e suspeita de um erro informativo é suprimida. Use o scream com cuidado e como uma ferramenta de depuração. Há muitos códigos da biblioteca PHP que podem não funcionar com o operador de controle de erro desativado.

- Operadores de Controle de Erro
- SitePoint
- xDebug
- Scream

ErrorException

O PHP é uma linguagem perfeitamente capaz de ser "exception-heavy" e requer apenas algumas linhas de código para fazer a troca. Basicamente, você pode lançar seus "erros" como "exceções", utilizando a classe `ErrorException`, que estende a classe `Exception`.

Esta é uma prática comum implementada por um grande número de frameworks modernos, como Symfony e Laravel. Por padrão Laravel irá exibir todos os erros como exceções usando o pacote Whoops! se o `app.debug` estiver ligado e em seguida escondê-los se estiver desligado.

Ao lançar erros como exceções em desenvolvimento você pode lidar com eles melhor do que o de costume, e se você ver uma exceção durante o desenvolvimento você pode envolvê-lo em uma instrução catch com instruções específicas sobre como lidar com a situação. Cada exceção que você pega instantaneamente faz da sua aplicação um pouco mais robusto.

Mais informações e os detalhes sobre como usar o `ErrorException` com tratamento de erros podem ser encontradas em Classe ErrorException.

- Operadores de Controle de Erro
- Constantes pré-definidas para manipulação de erros
- error_reporting
- Reporting

Exceções

Exceções são uma parte padrão da maioria das linguagens populares, mas elas sao frequentemente negligenciadas pelos programadores de PHP. Linguagens como Ruby usam pesadamente o sistema de Exceções, então sempre que algo de errado acontece, como um pedido HTTP que falha, ou uma consulta ao banco de dados gera um erro, ou até mesmo se uma imagem não puder ser encontrada, o

Ruby (ou suas bibliotecas que estiverem sendo utilizadas) irão disparar uma exceção para a tela, assim você saberá imediatamente que algo está errado.

O PHP por si só é bastante relaxado com isso, e uma chamada para `file_get_contents()` irá resultar apenas em um `FALSE` e um alerta. Muitos frameworks antigos, como CodeIgniter, irão apenas retornar um `FALSE`, registrar uma mensagem em seus logs proprietários e talvez deixar que você use um método como `$this->upload->get_error()` para ver o que houve de errado. O problema, aqui, é você tem que sair procurando por um erro e verificar na documentação para saber como achar o método que retorna o erro para essa classe, em vez de ter isso de forma extremamente óbvia.

Outro problema é quando as classes automaticamente disparam um erro para a tela e finalizam o processo. Quando você faz isso você impede que outro programador seja capaz de dinamicamente lidar com o erro. Exceções devem ser disparadas para que os desenvolvedores fiquem a par do erro, para então decidirem como lidar com ele. Ex:

```
<?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)
{
    // A validação falhou
}
catch(Fuel\Email\SendingFailedException $e)
{
    // O driver não pode enviar o e-mail
}
finally
{
    // Executado independentemente de se uma exceção foi acionada e antes de
retomar a execução normal
}
```

Exceções SPL

A classe genérica `Exception` fornece muito pouco contexto de depuração para o desenvolvedor; no entanto, para remediar esta situação, é possível criar uma `Exception` especializada como sub-classes da classe genérica `Exception`:

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

Isso significa que você pode adicionar múltiplos blocos de captura para lidar com diferentes Exceções. Isso pode lhe levar a criação de *muitas* exceções customizadas e algumas delas poderiam ter sido evitadas como o uso das Exceções SPL (exceções da biblioteca padrão) que estão disponíveis em SPL extension.

Se por exemplo você fizer uso do método mágico `__call()` e o método chamado for inválido, então em vez de disparar uma exceção padrão que seria muito vaga, ou criar uma exceção apenas para isso, você poderia disparar apenas um `throw new BadFunctionCallException;`.

- Leia sobre Exceções
- Leia sobre SPL Exceptions
- Aninhando exceções no PHP
- Melhores práticas com exceções no PHP 5.3

Segurança

Segurança em uma Aplicação Web

Existem pessoas ruins prontas e dispostas a invadir sua aplicação web. É importante que você tome as medidas necessárias para reforçar a segurança da sua aplicação web. Felizmente, o pessoal bacana da Open Web Application Security Project (OWASP) compilou uma lista abrangente dos problemas de segurança conhecidos e dos métodos para se proteger contra eles. É uma leitura obrigatória para o desenvolvedor preocupado com segurança.

- Leia o Guia OWASP de Segurança

Hash de Senhas

No fim, todos construímos aplicações PHP que dependem de login dos usuários. Usuários e senhas (com hash) são armazenadas em um banco de dados e posteriormente são usados para autenticar os usuários no login.

É importante que você faça adequadamente o *hash* das senhas que são armazenadas em um banco de dados. O hash da senha é irreversível, uma função executada contra a senha do usuário. Isto produz uma sequência de comprimento fixo que não pode ser revertido. Isto significa que você pode comparar um hash contra o outro para determinar se ambos foram produzidos da mesma string, mas você não pode determinar o string original. Se as senhas não estiverm com hash, e seu banco for hackeado ou acessado por alguém não autorizado, todas as contas dos usuários ficarão comprometidas. Alguns usuários (infelizmente) usam a mesma senha para outros serviços. Por isso, é importante levar segurança a sério.

Faça o hash das senhas com `password_hash`

No PHP 5.5 `password_hash` foi adicionado. Neste momento utiliza o BCrypt, o mais forte algorítimo suportado pelo PHP. Ele será atualizado no futuro para suportar mais algorítimos conforme for preciso. A biblioteca `password_compat` foi criada para ter compatibilidade para o PHP >= 5.3.7.

Abaixo um exemplo, vamos fazer o hash de uma string, e em seguida, verificamos o hash contra uma nova string. As duas string são diferentes (‘secret-password’ vs. ‘bad-password’) e por isso o login falhará.

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

$passwordHash = password_hash('secret-password', PASSWORD_DEFAULT);

if (password_verify('bad-password', $passwordHash)) {
    //Senha correta
} else {
    //Senha errada
}
```

- Aprenda sobre `password_hash`
- `password_compat` para PHP >= 5.3.7 && < 5.5
- Aprenda sobre hashing no que diz respeito à criptografia
- PHP `password_hash` RFC

Filtro de Dados

Nunca, jamais (nunca mesmo), confie em entradas externas feitas no seu código PHP. Sempre higienize (sanitize) e valide as entradas externas antes de utilizá-las no seu código. As funcões `filter_var` e `filter_input` podem higienizar textos e validar formatos (e.g. endereços de email).

Entradas externas podem ser qualquer coisa: dados de entrada de formulário `$_GET` ou `$_POST`, alguns valores na superglobal `$_SERVER` e o corpo da requisição HTTP via `fopen('php://input', 'r')`. Lembre-se, entradas externas não estão limitadas a dados de formulários enviados pelo usuário. Arquivos carregados e baixados, valores em sessões, dados dos cookies e dados de web services de terceiros também são entradas externas.

Enquanto o dado externo puder ser armazenado, combinado ou acessado posteriormente, ele continua sendo uma entrada externa. Todo momento que você processar, emitir, concatenar ou incluir dados no seu código, pergunte a si mesmo se eles foram filtrados adequadamente e se são confiáveis.

Os dados podem ser *filtrados* de maneiras diferentes baseando-se em sua finalidade. Por exemplo, quando entradas externas não filtradas são passadas para uma saída de página HTML, elas podem executar HTML e JavaScript no seu site! Isso é conhecido como Cross-Site Scripting (XSS) e pode ser um ataque bem perigoso. Um modo de evitar o XSS é higienizar todas as tags HTML da entrada, removendo as tags ou escapando-as usando entidades HTML.

Outro exemplo é ao passar opções para execução na linha de comando. Isso pode ser extremamente perigoso (e geralmente é má ideia), mas você pode usar a função embutida `escapeshellarg` para higienizar os argumentos executados.

Um último exemplo é aceitar entradas externas para determinar o carregamento de um arquivo do sistema de arquivos. Isso pode ser explorado alterando o nome e o caminho do arquivo. Você precisa remover os “/”, “../”, null bytes e outros caracteres do caminho do arquivo, dessa forma não será possível carregar arquivos ocultos, privados ou confidenciais.

- Aprenda sobre filtro de dados
- Aprenda sobre a `filter_var`

- Aprenda sobre a `filter_input`
- Aprenda sobre tratamento de null bytes

Higienização/Sanitization

A higienização remove (ou “escapa”) caracteres ilegais ou inseguros das entradas externas.

Por exemplo, você deveria higienizar entradas externas antes de incluí-las no HTML ou de inseri-las em consultas SQL brutas. Quando você usar parâmetros restritos com a PDO, ela já irá higienizar as entradas para você.

Às vezes será obrigatório permitir algumas tags HTML seguras na sua entrada quando estiver incluindo-as em um página HTML. Isso é bem difícil de fazer e muitas evitam isso utilizando outros formatos mais restritos, como o Markdown ou o BBCode, embora bibliotecas para listas brancas/whitelistening, como a HTML Purifier, existem por essa razão.

Veja sobre os Filtros de Higienização

Validação

A validação garante que as entradas externas são o que você espera. Por exemplo, você pode querer validar um endereço de email, um número de telefone ou uma idade quando for processar o envio de um registro.

Veja sobre os Filtros de Validação

Arquivos de Configuração

Quando criar arquivos de configuração para suas aplicações, as melhores práticas recomendam que um dos seguintes métodos seja seguido:

- É recomendado que você armazene sua informação de configuração onde ela não possa ser acessada diretamente ou puxada através do sistema de arquivos.
- Se você tiver que armazenar seus arquivos de configuração no diretório raiz, nomeie os arquivos com a extensão `.php`. Isso garante que, mesmo se um script for acessado diretamente, ele não será mostrado como texto puro.
- As informações nos arquivos de configuração devem ser adequadamente protegidas, ou através de criptografia ou por permissões de grupos/usuários no sistema de arquivos

Register Globals

OBSERVAÇÃO: A partir do PHP 5.4.0 a configuração `register_globals` foi removida e não pode mais ser utilizada. Isto só foi incluído como um alerta para alguém no processo de atualização de uma aplicação legada.

Quando habilitada, a configuração `register_globals` torna disponível, no escopo global da sua aplicação, vários tipos de variáveis (`$_POST`, `$_GET` e `$_REQUEST`). Isso pode facilmente levar a problemas de segurança pois sua aplicação não pode dizer de forma efetiva de onde o dado está vindo.

Por exemplo: `$_GET['foo']` poderia ficar disponível via `$foo`, o que poderia sobrescrever variáveis que não tiverem sido declaradas. Se você estiver usando PHP < 5.4.0 **garanta** que `register_globals` esteja **desligado**.

- Register_globals no manual do PHP

Relatório de Erros

O registro de erros pode ser útil para encontrar pontos problemáticos em sua aplicação, mas isso também pode expor informações sobre a estrutura de sua aplicação para o mundo exterior. Para proteger efetivamente sua aplicação dos problemas que poderiam ser causados com a exposição dessas mensagens, você precisa configurar seu servidor de formas diferentes quando em desenvolvimento versus quando em produção (no ar).

Desenvolvimento

Para mostrar erros no seus ambiente de **desenvolvimento**, configure as definições a seguir no seu `php ini`:

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

Do php.net:

> Passar o valor -1 irá mostrar todos os erros possíveis, até mesmo quando novos níveis e constantes forem adicionados em versões futuras do PHP. A constante E_ALL também se comporta desta maneira a partir do PHP 5.4.

O nível de error `E_STRICT` foi introduzido no 5.3.0 e não faz parte do `E_ALL`, contudo ele tornou-se parte do `E_ALL` no 5.4.0. O que isso significa? Que para mostrar todos os erros possíveis na versão 5.3 você precisa usar `-1` ou `E_ALL | E_STRICT`.

Reportando todos os erros possíveis em diferentes versões do PHP

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

Com essas configurações em produção, os erros continuarão sendo registrados nos logs de erros do servidor web, mas eles não serão mostrados para o usuário. Para mais informações sobre essas configurações, veja o manual do PHP:

- error_reporting
- display_errors
- display_startup_errors
- log_errors

Testes

Escrever testes automatizados para o seu código PHP é considerado uma boa prática, e leva a aplicações bem escritas. Testes automatizados são uma excelente ferramenta para garantir que sua aplicação não irá quebrar quando você estiver fazendo alterações ou adicionando novas funcionalidades, e não deveriam ser ignorados.

Existem vários tipos diferentes de ferramentas de testes (ou frameworks) disponíveis para PHP, com diferentes abordagens: todas elas tentam evitar os testes manuais e a necessidade de equipes grandes de Garantia de Qualidade Quality Assurance, ou QA) apenas para garantir que alterações recentes não interrompam funcionalidade existentes.

Desenvolvimento Guiado por Testes

Da Wikipedia:

> O desenvolvimento guiado por testes (TDD) é um processo de desenvolvimento que se baseia na repetição de um ciclo de desenvolvimento bem curto: primeiro o desenvolvedor escreve um caso de teste automatizado que falha, definindo uma melhoria ou uma nova função desejada, em seguida produz o código para passar no teste, e finalmente refatora o novo código pelos padrões aceitáveis. Kent Beck, que é creditado como quem desenvolveu ou "redescobriu" essa técnica, afirmou em 2003 que o TDD encoraja design simples e inspira confiança.

Existem vários tipos diferentes de testes que você pode fazer para sua aplicação.

Testes Unitários

Testes unitários são uma metodologia de programação que garante que as funções, as classes e os métodos estão funcionando como esperado, desde o momento que você os constrói até o fim do ciclo de desenvolvimento. Verificando como os valores entram e saem em várias funções e métodos, você pode garantir que a lógica interna está funcionando corretamente. Utilizando Injeção de Dependências e construindo classes "mock" e stubs, você pode verificar se as dependências foram utilizadas corretamente para uma cobertura de testes ainda melhor.

Quando for criar uma classe ou função, você deveria criar um teste unitário para cada comportamento que ela deveria ter. Num nível bem básico, você deveria garantir que são emitidos erros quando você envia argumentos errados e garantir que tudo funciona bem se você enviar argumentos válidos. Isso ajudará a garantir que, quando você alterar sua classe ou sua função posteriormente no ciclo de desenvolvimento, as funcionalidades antigas continuarão funcionando como esperado. A única alternativa a isso seria usar var_dump() em um test.php, o que não é o certo a fazer na construção de uma aplicação - grande ou pequena.

O outro uso para testes unitários é contribuir para projetos open source. Se você puder escrever um teste que demonstra uma funcionalidade incorreta (i.e. uma falha), em seguida consertá-la e mostrar o teste passando, os patches serão muito mais suscetíveis a serem aceitos. Se você estiver em um projeto que aceite pull requests, você deveria sugerir isso como um requisito.

O PHPUnit é o framework de testes de fato para escrever testes unitários em aplicações PHP, mas existem várias alternativas:

- SimpleTest
- Enhance PHP
- PUnit
- atoum

Testes de Integração

Da Wikipedia:

> Testes de integração (chamado às vezes de “Integração e Teste”, abreviado como “I&T”) é a fase do teste do software na qual módulos individuais do sistema são combinados e testados como um grupo. Ela acontece após os testes unitários e antes dos testes de validação. Os testes de integração recebem como entrada os módulos que foram testados unitariamente, os agrupa em grandes blocos, aplica testes definidos em um plano de teste de integração, e entrega como saída o sistema integrado pronto para os testes de sistema.

Muitos das mesmas ferramentas que são usadas para testes unitários podem ser usadas para testes de integração, pois muitos dos mesmos princípios são usados.

Testes Funcionais

Algumas vezes conhecidos também como testes de aceitação, os testes funcionais consistem em utilizar ferramentas para criar testes automatizados que usem de verdade sua aplicação, em vez de apenas verificar se unidades individuais de código se comportam adequadamente ou se essas partes conversam entre si do jeito certo. Essas ferramentas geralmente trabalham usando dados reais e simulam usuários verdadeiros da sua aplicação.

Ferramentas para Testes Funcionais

- Selenium
- Mink
- O Codeception é um framework de testes full-stack que inclui ferramentas para testes de aceitação
- O Storyplayer é um framework de testes full-stack que inclui suporte para criação e destruição de ambientes sob demanda

Desenvolvimento Guiado por Comportamentos

Existem dois tipos diferentes de Desenvolvimento Guiado por Comportamentos (BDD): o SpecBDD e o StoryBDD. O SpecBDD foca nos comportamentos técnicos, no código, enquanto que o StoryBDD foca nos comportamentos de negócio e de funcionalidades, nas interações. O PHP possui frameworks para ambos os tipos de BDD.

No StoryBDD, você escreve histórias humanamente legíveis que descrevem o comportamento da sua aplicação. Estas histórias podem então ser executadas como testes reais em sua aplicação. O framework usado nas aplicações PHP para StoryBDD é o Behat, que foi inspirado no projeto Cucumber do Ruby e implementa a linguagem Gherkin DSL para descrever o comportamento das funcionalidades.

No SpecBDD, você escreve as especificações que descrevem como seu código real deveria se comportar. Em vez de escrever uma função ou método, você descreve como a função ou o método deveriam se comportar. O PHP fornece o framework PHPSpec para esse propósito. Esse framework foi inspirado no projeto RSpec do Ruby.

Links sobre BDD

- O Behat é inspirado pelo projeto Cucumber do Ruby
- O PHPSpec é o framework para SpecBDD do PHP
- O Codeception é um framework de testes full-stack que usa os princípios do BDD

Ferramentas Complementares para Testes

Além dos testes individuais e dos frameworks guiados por comportamentos, também existe uma série de frameworks genéricos e bibliotecas auxiliares úteis para qualquer das abordagem escolhidas.

Links para as Ferramentas

- O Selenium é uma ferramenta para automação de navegação que pode ser intregada ao PHPUnit
- O Mockery é um Framework para Objetos Mock que pode ser integrado ao PHPUnit e ao PHPSpec.
- O Prophecy é um framework para Objetos Mock bastante obstinado porém poderoso e flexível. É integrado com PHPSpec e pode ser usado com PHPUnit(http://phpunit.de/).

Servidores e Publicação

As aplicações PHP podem ser publicadas e executadas em servidores web de produção de diversas formas.

Plataforma como Serviço (PaaS)

O PaaS fornece o sistema e a arquitetura de rede necessários para executar aplicações PHP na web. Isso significa não precisar de quase nenhuma configuração para publicar aplicações ou frameworks PHP.

Recentemente o PaaS se tornou um método popular para publicar, hospedar e escalar aplicações PHP de todos os tamanho . Você pode encontrar uma lista de fornecedores de PHP PaaS “Platform as a Service” na seção sobre recursos.

Servidores Virtuais ou Dedicados

Se você estiver confortável com administração de sistemas, ou estiver interessado em aprender sobre isso, os servidores virtuais ou dedicados te dão controle completo do ambiente de produção da sua aplicação.

nginx e PHP-FPM

O PHP, por meio do seu Gerenciador de Processos FastCGI (FPM), funciona muito bem junto com o nginx, que é um servidor web leve e de alta performance. Ele usa menos memória do que o Apache e pode lidar melhor como muitas requisições concorrentes. Ele é importante especialmente em servidores virtuais que não tem muita memória sobrando.

- Leia mais sobre o nginx
- Leia mais sobre o PHP-FPM
- Leia mais sobre como configurar de forma segura o nginx e o PHP-FPM

Apache e PHP

O PHP e o Apache tem um longo histórico juntos. O Apache é amplamente configurável e tem muitos módulos disponíveis para estender suas funcionalidades. Ele é uma escolha popular para servidores compartilhados e pela configuração fácil em frameworks PHP e aplicativos open source como, o Wordpress. Infelizmente, o Apache utiliza mais recursos do que o nginx por padrão e não pode lidar com tantos visitantes ao mesmo tempo.

O Apache tem várias configurações possíveis para executar o PHP. A mais comum e mais fácil para configurar é a prefork MPM com o mod_php5. Mesmo não sendo a mais eficiente em memória, é a mais simples para começar a usar. Essa é provavelmente a melhor escolha se você não quiser entrar muito profundamente nos aspectos de administração do servidor. Observe que, se você usar o mod_php5, terá que usar o prefork MPM.

Alternativamente, se você quiser extrair mais performance e estabilidade do seu Apache então você poderia se beneficiar do mesmo sistema FPM que o nginx e executar o worker MPM ou o event MPM com o mod_fastcgi ou com o mod_fcgi. Essa configuração será significativamente mais eficiente em relação a memória e muito mais rápida, mas gera mais trabalho.

- Leia mais sobre o Apache
- Leia mais sobre os Multi-Processing Modules
- Leia mais sobre o mod_fastcgi
- Leia mais sobre o mod_fcgid

Servidores Compartilhados

O PHP tem que agradecer aos servidores compartilhados por sua popularidade. É difícil encontrar uma hospedagem que não tenha o PHP instalado, mas certifique-se de que seja a última versão. Servidores compartilhados permitem que você e outros desenvolvedores publiquem sites em uma única máquina. A parte boa é que isso se tornou uma commodity barata. A parte ruim é que você nunca sabe que tipo de bagunça seus vizinhos vão criar; sobrecarregamento do servidor e abertura de falhas de segurança são os principais problemas. Evite usar servidores compartilhados se o orçamento de seu projeto permitir.

“Compilando” e Implementando sua Aplicação

Se você se pegar fazendo alterações manuais no seu esquema do banco de dados ou rodando seus testes manualmente antes de alterar seus arquivos (manualmente), pense duas vezes! A cada tarefa manual adicional necessária para implementar uma nova versão da sua aplicação as chances de

erros fatais são potencialmente maiores, Seja lidando com uma simples atualização, um processo completo de implementação ou até mesmo uma estratégia de integração contínua, a Automação de Compilação é sua amiga.

Entre as tarefas que talvez você deseja automatizar estão:

- Gerenciamento de Dependências
- Compilação e Compressão de Arquivos
- Execução de Testes
- Criação de Documentação
- Empacotamento
- Implementação

Ferramentas de Automação

Ferramentas de automação podem ser descritas como uma coleção de scripts que tratam de tarefas comuns da implementação de software. As ferramentas de automação não são parte da sua aplicação, elas agem na sua aplicação externamente.

Existem muitas ferramentas de código aberto disponíveis para ajudar você com o processo de automação, algumas são escritas em PHP, outras não. Isso não deve impedi-lo de usá-las, se elas se ajustarem melhor ao trabalho em questão. Aqui estão alguns exemplos:

Phing é o jeito mais fácil de começar com automação de implementação no PHP. Com Phing você pode controlar os processos de empacotamento, implementação e testes através de um simples arquivo XML. Phing (Que é baseado em Apache Ant) fornece um rico conjunto de tarefas geralmente necessárias para instalar ou atualizar uma aplicação web e pode ser estendido com tarefas adicionais personalizadas, escritas em PHP.

Capistrano é um sistema para *programadores intermediarios ou avançados* que executa comando de forma estruturada e repetitiva em uma ou mais maquinas. Ele é pré-configurado para implementar aplicações Ruby on Rails, entretanto pessoas estão **implementando com sucesso sistemas em PHP** com ele. Ter sucesso com uso de Capistrano depende de um conhecimento de trabalho com Ruby e Rails.

O artigo de Dave Gardner PHP Deployment com Capistrano é um bom ponto de partida para desenvolvedores PHP interessando em Capistrano.

Chef é mais que um framework de implementação, é um framework de integração bastante poderoso escrito em Ruby que não consegue apenas implementar sua aplicação mas também construir seu ambiente de servidor completo em maquinas virtuais.

Conteúdo sobre Chef para Desenvolvedores PHP:

- Serie em 3 partes sobre implementação de uma aplicação LAMP com Chef, Vagrant, e EC2
- Chef Cookbook sobre instalação e configuração de PHP 5.3 e do sistema de gerenciamento de pacotes PEAR
- Chef - série de video tutoriais por Opscode, os criadores do chef

Leitura Adicional:

- Automatize seu projeto com Apache Ant

Integração Contínua

> Integração Contínua é uma prática de desenvolvimento de software onde membros de um time integram seu trabalho com frequência, geralmente essa integração ocorre diariamente - levando a muitas integrações de código por dia. Muitos times acreditam que essa abordagem leva a reduções significativas dos problemas de integração e permite que o time desenvolva software de forma coesa e rápida.

– Martin Fowler

Existem diferentes formas de se implementar integração contínua com PHP. Recentemente o Travis CI tem feito um ótimo trabalho ao fazer da integração contínua uma realidade mesmo para pequenos projetos. O Travis CI é um sistema de integração contínua na nuvem desenvolvido pela comunidade de código livre. Esta integrado com GitHub e oferece suporte de primeira classe para muitas linguagens incluindo PHP.

Leitura Adicional:

- Integração Contínua com Jenkins
- Integração Contínua com PHPCI
- Integração Contínua com Teamcity

Virtualização

Executar seu aplicativo em ambientes diferentes de desenvolvimento e produção pode levar a aparecer erros estranhos. Também é complicado manter diferentes ambientes de desenvolvimento atualizados com a mesma versão de todas as bibliotecas usadas quando se trabalha com uma equipe de desenvolvedores.

Se você estiver desenvolvendo em Windows e implantando de Linux (ou qualquer coisa não-Windows) ou estão desenvolvendo em uma equipe, você deve considerar o uso de uma máquina virtual. Isso parece complicado, mas além dos ambientes de virtualização amplamente conhecidos como o VMware ou VirtualBox, existem ferramentas adicionais que podem ajudá-lo a configurar um ambiente virtual em algumas etapas fáceis.

Vagrant

O Vagrant ajuda a construir suas máquinas virtuais em cima de ambientes virtuais conhecidos e a configurar esses ambientes com base em um único arquivo de configuração. As máquinas virtuais base (box) podem ser configuradas manualmente, ou você pode usar um software de “provisionamento” como o Puppet ou o Chef para fazer isso por você. Provisionar o box é uma ótima maneira de garantir que as múltiplas máquinas virtuais sejam configuradas de forma idêntica e que você não necessite manter complicadas listas de comandos de configuração. Você pode também destruir (destroy) o box base e recriá-lo sem muitos passos manuais, tornando fácil criar instalações novas.

O Vagrant cria pastas compartilhadas para compartilhar seu código entre sua máquina e a máquina virtual, assim você pode criar e editar seus arquivos na sua máquina e então executar seu código dentro da máquina virtual.

Uma pequena ajuda

Se você precisa de uma pequena ajuda para inciar o uso do Vagrant existem dois serviços que podem ser úteis:

- Rove: serviço que permite que você gere configurações típicas do Vagrant, sendo o PHP uma das opções. O provisionamento é realizado com Chef.
- Puphpet: interface gráfica simples de configurar máquinas virtuais para o desenvolvimento PHP. **Altamente focada em PHP**. Além VMs local, ele pode ser usado para implantar em serviços de nuvem também. O provisionamento é feito com Puppet.
- Protobox: é uma camada em cima do vagrant e uma interface gráfica web para configuração de máquinas virtuais para o desenvolvimento web. Um único documento YAML controla tudo o que está instalado na máquina virtual.
- Phansible: oferece uma interface que ajuda a gerar Ansible Playbooks para projetos baseados em PHP.

Docker

Junto ao Vagrant, uma outra maneira fácil de obter um ambiente de desenvolvimento ou produção virtual instalado e funcionando é o Docker. O Docker ajuda você a fornecer recipientes Linux para todos os tipos de aplicações. Há muitas imagens Docker úteis que podem fornecer outros grandes serviços sem a necessidade de instalar estes serviços em sua máquina local, por exemplo, MySQL ou PostgreSQL e muito mais. Dê uma olhada no Docker Hub Registry para procurar uma lista de recipientes pré-construídos disponíveis, que você pode executar e usar em poucos passos.

Exemplo: Runnning suas aplicações PHP em Docker

Depois de instalar o docker em sua máquina, você pode iniciar um apache com suporte a PHP em uma única etapa. O comando a seguir irá baixar uma instalação apache totalmente funcional com a última versão do PHP e fornecer o diretório `/path/to/your/php/files` em `http://localhost:8080`:

```
docker run -d --name my-php-webserver -p 8080:80 -v
/path/to/your/php/files:/var/www/html/ php:apache
```

Depois de executar o `docker run` seu recipiente estará inicializado e funcionando. Se você quiser parar (stop) ou iniciar (start) o seu recipiente novamente, você pode usar o atributo `--name` fornecido e simplesmente executar o `docker stop my-php-webserver` e `docker start my-php-webserver` sem fornecer novamente os demais parâmetros mencionados acima.

Saiba mais sobre Docker

Os comandos mencionados acima mostram apenas uma maneira rápida de executar um servidor web apache com suporte a PHP, mas há muito mais coisas que você pode fazer com Docker. Uma das coisas mais importantes para os desenvolvedores de PHP é ligar seu servidor web com uma instância de banco de dados, por exemplo. Como fazer isso está bem descrito no Guia do Usuário Docker.

- Docker
- Instalação do Docker

- Imagens Docker no Docker Hub Registry
- Guia do Usuário Docker

Cache

O PHP é bem rápido por si só, mas gargalos podem aparecer quando você faz conexões remotas, ou carrega arquivos etc. Felizmente, existem várias ferramentas disponíveis para acelerar certas partes de sua aplicação, ou para reduzir o número de vezes que essas tarefas, que tomam tempo, precisem ser executadas.

Cache de Bytecode

Quando um arquivo PHP é executado, por baixo dos panos ele primeiro é compilado para bytecode (também conhecido como opcode) e, só aí, o bytecode é executado. Se o arquivo PHP não foi modificado, o bytecode será sempre o mesmo. Isso significa que o passo de compilação é um desperdício de recursos de CPU.

É aqui que entra o cache de Bytecode. Ele previne as compilações redundantes armazenando bytecode na memória e reutilizando-o em chamadas sucessivas. A configuração do cache de bytecode é feita em questão de minutos, e sua aplicação irá acelerar de forma significativa. Não existe razão para não utilizá-lo.

No PHP 5.5 o OPcache foi incluído como um cache de bytecode nativo chamado OPcache. Ele também está disponível para versões anteriores.

Caches de bytecode populares são:

- OPcache (desde o PHP 5.5)
- APC (PHP 5.4 e anteriores)
- XCache
- Zend Optimizer+ (parte do pacote Zend Server)
- WinCache (extensão para o MS Windows Server)

Cache de Objetos

Existem momentos onde pode ser benéfico fazer cache de objetos individuais no seu código, como em dados que são custosos de conseguir ou em chamadas de bancos de dados cujo resultado dificilmente se modifica. Você pode usar um software de cache de objetos para armazenar esses pedaços de dados na memória, para acessá-los posteriomente de forma extremamente rápida. Se você guardar esses itens em um data store logo que os recuperar, e depois os enviar diretamente do cache para as suas requisições posteriores, você conseguirá uma melhora significativa no desempenho além de reduzir a carga nos seus servidores de banco de dados.

Muitas das soluções populares de cache de bytecode permitem que você também faça cache de dados personalizados, assim há ainda mais razões para se beneficiar deles. Tanto o APC, quanto o XCache e o Wincache fornecem APIs para armazenar dados do seu código PHP na memória cache deles.

Os sistemas mais usados para cache de objetos em memória são o APC e o memcached. O APC é uma escolha excelente para cache de objetos, ele inclui uma API simples para adicionar seus próprios dados para seu cache de memória e é muito fácil de configurar e utilizar. A única limitação real do APC é que ele fica amarrado ao servidor onde ele está instalado. O memcached por outro lado é instalado como um serviço separado e pode ser acessado através da rede, assim você pode armazenar objetos em um data store ultra-rápido, em uma localização central, e vários sistemas diferentes podem acessá-lo.

Note que se estiver rodando o PHP como uma aplicação (Fast-)CGI dentro do seu servidor web, cada processo do PHP terá seu próprio cache, ou seja, os dados de APCu não são compartilhados entre diferentes processos. Nesse caso você deve considerar usar mecached em seu lugar, já que ele não está ligado aos processos do PHP.

Em uma configuração em rede, o APC geralmente terá um desempenho melhor do que o memcached em termos da velocidade de acesso, mas o memcached poderá escalar mais rápido e melhor. Se você não planeja ter múltiplo servidores executando sua aplicação, ou não precisar das funcionalidades extras que o memcached oferece, então o APC provavelmente é sua melhor opção para cache de objetos.

Exemplo de lógica usando APC:

```
<?php
// verifica se existe um dado salvo como 'expensive_data' no cache
$data = apc_fetch('expensive_data');
if (!$data)
{
    // dado não está no cache, faça a chamada custosa e guarde-a para usar
depois
    $data = get_expensive_data();
    apc_store('expensive_data', $data);
}

print_r($data);
```

Note que, antes do PHP 5.5, APC possui tanto um cache de objetos quanto um cache de bytecode. APCu é um projeto que tem como objetivo trazer o cache de objetos do APC para o PHP 5.5, já que o PHP agora possui um cache bytecode nativo (OPcache).

Aprenda mais sobre sistemas populares de cache de objetos:

- APCu
- Funções APC
- Memcached
- Redis
- XCache APIs
- Funções do WinCache

Recursos

Da Fonte

- Site do PHP
- Documentação do PHP

Pessoas para Seguir

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

Mentoring

- phpmentoring.org - Mentoring formal e `pessoa-para-pessoa` na comunidade PHP.

Fornecedores de PaaS PHP

- PagodaBox
- AppFog
- Heroku
- fortrabbit
- Engine Yard Cloud
- Red Hat OpenShift Platform
- dotCloud
- AWS Elastic Beanstalk
- cloudControl
- Windows Azure
- Zend Developer Cloud
- Google App Engine
- Jelastic

Frameworks

Em vez de reinventar a roda, muitos desenvolvedores PHP usam frameworks para construir aplicações web. Os frameworks abstraem muitas das preocupações de baixo nível e fornecem interfaces úteis e fáceis de utilizar para completar tarefas comuns.

Você não precisa usar um framework para todo projeto. Algumas vezes, PHP puro é a maneira certa de fazer, mas se você realmente precisar de um framework existem três tipos disponíveis:

- Micro Frameworks
- Full-Stack Frameworks
- Component Frameworks

Os micro-frameworks são essencialmente invólucros para rotear uma requisição HTTP para um callback, ou um controller, ou um método etc., da forma mais rápida possível, e algumas vezes possuem algumas bibliotecas para auxiliar no desenvolvimento, como por exemplo pacotes básicos para bancos de dados. Eles são proeminentemente usados para construir serviços HTTP remotos.

Muitos frameworks adicionam um número considerável de funcionalidades ao que está disponível em um micro-framework e são conhecidos como frameworks completos ou full-stack. Eles frequentemente possuem ORMs, pacotes de autenticação, etc.

Frameworks baseados em componentes são coleções de bibliotecas especializadas ou de propósito-único. Diferentes frameworks baseados em componentes podem ser utilizados conjuntamente para criar um micro-framework ou um framework completo.

Componentes

Como mencionado acima “Componentes” são são uma outra abordagem comum para a o objetivo comum de criação, distribuição e implementação de código compartilhado. Existem vários repositórios de componentes, os dois principais são:

- Packagist
- PEAR

Ambos repositórios possuem ferramentas de linha de comando para ajudar a instalação e processos de atualização, e foram explicados com mais detalhes na seção Gerenciamento de Dependência

Há também componentes de frameworks e componentes de fornecedores que não oferecem nenhum framework. Estes projetos fornecem outra fonte de pacotes que idealmente tem pouco ou nenhum dependências de outros pacotes, ou estruturas específicas.

Por exemplo, você pode usar o FuelPHP Validation package sem precisar usar o framework FuelPHP em si.

- Aura
- FuelPHP
- Hoa Project
- Orno
- Componentes Symfony
- A Liga Extraordinária de pacotes
- Componentes Illuminate do Laravel
    - Eloquent ORM
    - Fila

Outros Componentes Illuminate do Laravel virão com menor desacoplamento do framework Laravel. Por enquanto, apenas os componentes mais desacoplados do framework Laravel estão listados acima.

Livros

Há uma grande quantidade de livros em sobre PHP, mas alguns são infelizmente bastante antigos e já não contêm informações precisas. Há ainda livros publicados para o “PHP 6”, que não existe e nunca irá existir. A próxima grande versão do PHP chamará “PHP 7” por causa destes livros.

Esta seção tem como objetivo ser um documento vivo de livros recomendados sobre o desenvolvimento PHP em geral. Se você gostaria que o seu livro seja adicionado, envie um PR e será revistado para relevância.

Livros gratuitos

- PHP The Right Way - Este site está disponível como um livro totalmente de graça

Livros Pagos

- 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.

Comunidade

A comunidade PHP é tão diversa quanto é grande, e seus membros estão prontos e dispostos para a apoiar novos programadores PHP. Considere se juntar ao grupo de usuários PHP (PUG) de sua localidade ou participar das grandes conferências PHP para aprender mais sobre as melhores práticas mostradas aqui. Você também pode entrar no IRC, no canal #phpc em irc.freenode.com e também seguir o @phpc no twitter. Vá lá, conheça novos desenvolvedores, aprenda sobre novos assuntos e, acima de tudo, faça novos amigos. Outros recursos da comunidade inclui o StackOverflow.

Veja o Calendário de Eventos Oficiais do PHP

PHPDoc

PHPDoc é um padrão informal para comentar código PHP. Há um *monte* de diferentes tags disponíveis. A lista completa de tags e exemplos podem ser encontrados no Manual do PHPDoc.

Abaixo está um exemplo de como você pode documentar uma classe com alguns métodos;

```php
<?php
/**
 * @author Um nome <a.name@example.com>
 * @link http://www.phpdoc.org/docs/latest/index.html
 * @package helper
 */
class DateTimeHelper
{
    /**
     * @param mixed $anything Tudo que podemos converter para um objeto \
DateTime
     *
     * @return \DateTime
     * @throws \InvalidArgumentException
     */
    public function dateTimeFromAnything($anything)
    {
        $type = gettype($anything);

        switch ($type) {
            // Algum código que tenta retornar um objeto \DateTime
        }

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

    /**
     * @param mixed $date Tudo que podemos converter para um objeto \DateTime
     *
     * @return void
     */
    public function printISO8601Date($date)
    {
        echo $this->dateTimeFromAnything($date)->format('c');
    }

    /**
     * @param mixed $date Tudo que podemos converter para um objeto \DateTime
     */
    public function printRFC2822Date($date)
    {
        echo $this->dateTimeFromAnything($date)->format('r');
    }
}
```

A documentação para a classe como um todo, em primeiro lugar tem a tag @author, esta tag é usada para documentar o autor do código e pode ser repetido para documentar vários autores. Em segundo lugar é a tag @link, usada para conectar-se a um site que indica uma relação entre o site eo código. Em terceiro lugar, tem a tag @package, usada para categorizar o código.

Dentro da classe, o primeiro método tem uma tag @param usada para documentar o tipo, nome e descrição do parâmetro sendo passado para o método. Além disso, ele tem as tags @return e @throws para documentar o tipo de retorno e todas as exceções que podem ser lançadas respectivamente.

O segundo e terceiro métodos são muito semelhantes e têm uma única marcação @param, assim como o primeiro método. A diferença de importação entre o segundo eo terceiro método é bloco doc é a inclusão/exclusão da tag @return. `@return void` nos informa explicitamente que não há

retorno, historicamente omitir a declaração `@return void` também resulta na mesma ação (sem retorno).

Banners para divulgação

Espalhe a palavra com os banners *PHP: Do Jeito Certo*! Mostre à novos programadores PHP onde encontrar boa informação.

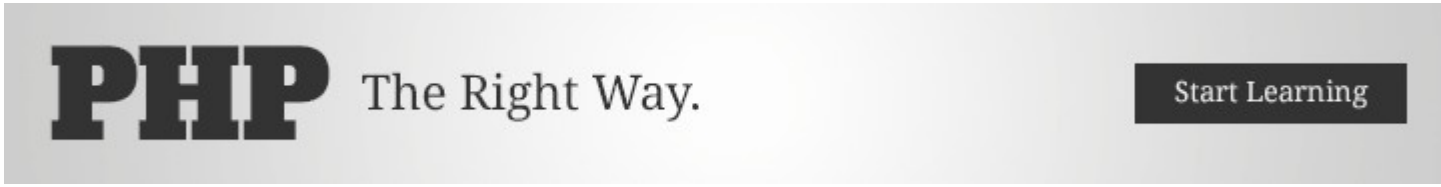
Botão 1 (120x90)


```
    <a href="http://br.phptherightway.com">
        <img src="http://br.phptherightway.com/images/banners/btn1-120x90.png"
alt="PHP: Do Jeito Certo"/>
    </a>
```

Botão 2 (120x60)


```
    <a href="http://br.phptherightway.com">
        <img src="http://br.phptherightway.com/images/banners/btn2-120x60.png"
alt="PHP: Do Jeito Certo"/>
    </a>
```

Leaderboard (728x90)




```
    <a href="http://br.phptherightway.com">
        <img src="http://br.phptherightway.com/images/banners/leaderboard-
728x90.png" alt="PHP: Do Jeito Certo"/>
    </a>
```

Retângulo Grande (386x280)


```
    <a href="http://br.phptherightway.com">
        <img src="http://br.phptherightway.com/images/banners/lg-rect-
386x280.png" alt="PHP: Do Jeito Certo"/>
    </a>
```

Retângulo Médio (300x250)


```
    <a href="http://br.phptherightway.com">
        <img src="http://br.phptherightway.com/images/banners/med-rect-
300x250.png" alt="PHP: Do Jeito Certo"/>
    </a>
```

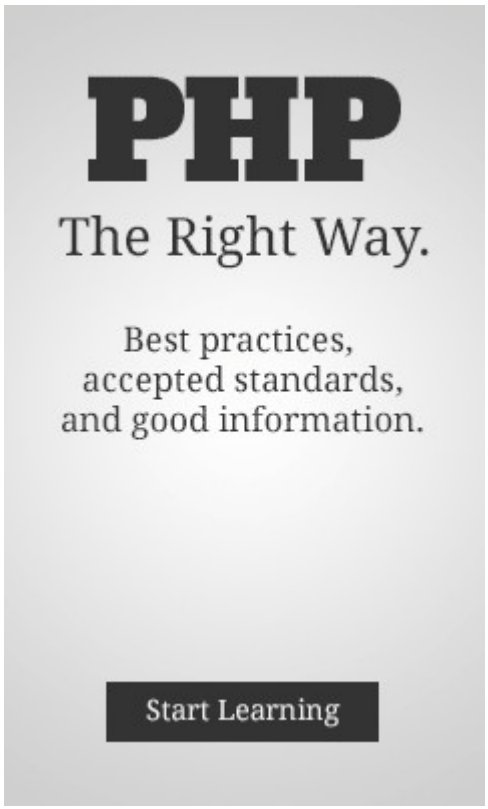
Retângulo (180x150)


```
    <a href="http://br.phptherightway.com">
        <img src="http://br.phptherightway.com/images/banners/rect-180x150.png"
alt="PHP: Do Jeito Certo"/>
    </a>
```

Botão Quadrado (125x125)


```
    <a href="http://br.phptherightway.com">
        <img src="http://br.phptherightway.com/images/banners/sq-btn-
125x125.png" alt="PHP: Do Jeito Certo"/>
    </a>
```

Retângulo Vertical (240x400)




```
    <a href="http://br.phptherightway.com">
        <img src="http://br.phptherightway.com/images/banners/vert-rect-
240x400.png" alt="PHP: Do Jeito Certo"/>
    </a>
```

https://br.phptherightway.com