PHP

Do Jeito Certo.

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 as 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*. Essa é a graça. Este site introduz novos desenvolvedores PHP às melhores práticas, opções disponíveis e boas informações, que por muitas vezes, quando descobertas, já é tarde de mais. 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 multiplas 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.

Traduções

PHP: Do Jeito Certo está (ou em breve estará) sendo traduzido em várias linguagens:

- Inglês
- Catalão (Link quebrado... estamos procurando por ele)
- Chinês
- Japonês
- Coreano
- Italian (Link quebrado... estamos procurando por ele)
- Polonês
- Português
- Russo
- Espanhol (Link quebrado... estamos procurando por ele)
- Ucraniano
- Búlgaro
- Alemão
- Turco

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

Começando

Use a versão estável atual (5.5)

Se você está começando com o PHP, certifique-se de começar com a versão estável atual do PHP 5.5. O PHP fez grandes avanços adicionando novas funcionalidades poderosas nos últimos anos. Não deixe que a pequena diferença entre os números decimais das versões 5.2 a 5.5 o enganem, elas representam *grandes* melhorias. Se você está procurando por uma função ou seu uso, a documentação no php.net terá a resposta.

Servidor web embutido

Você pode começar a aprender PHP sem os problemas de instalar e configurar um servidor web (mas você precisa do PHP 5.4+). 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 OSX já vem com o PHP, mas ele é normalmente um pouco atrasado em relação ao último estável. O Lion vem com o PHP 5.3.6, Mountain Lion com o 5.3.10 e Mavericks com 5.4.17.

Para atualizar o PHP no OSX você pode pegar o executável do PHP através de vários gerenciadores de pacote para Mac, sendo php-osx por Liip o mais recomendado.

Outra opção é compilar você mesmo. Nesse caso, certifique-se de ter instalado o Xcode ou seu substituto "Command Line Tools for Xcode" disponível no Mac Developer Center da Apple).

Para um pacote LAMP completo com GUI, tente o MAMP ou XAMPP.

Instalação no Windows

O PHP está disponível de diversas maneiras no Windows. Você pode baixar os binários e até recentemente você podia usar um instalador '.msi'. O instalador não é mais suportado e parou no PHP 5.3.0.

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 o Web Platform Installer, o Zend Server CE, o XAMPP e o 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.

Vagrant

Rodar sua aplicação em ambientes diferentes entre desenvolvimento e produção pode levar a estranhos bugs quando a aplicação estiver no ar. Também é complicado manter diferentes ambientes de desenvolvimento atualizados com as mesmas versões de bibliotécas enquanto estiver trabalhando em uma equipe de desenvolvedores.

Se você estiver desenvolvendo em Windows e publicando em Linux (ou qualquer coisa que não seja Windows) ou se está desenvolvendo em uma equipe, você deveria considerar o uso de uma máquina virtual. Pode parecer complicado, mas usando o Vagrant você poderá configurar uma máquina virtual simples com apenas alguns passos. 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 para configurar máquinas virtuais para o desenvolvimento com PHP. **Altamente focado em PHP**. Além de máquinas virtuais locais, pode ser usado para configurar máquinas em serviços de cloud. O provisionamento é feito com Puppet.

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 e PSR-2. 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 os Padrões de Código da PEAR
- Leia sobre os Padrões de Código da Zend

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.

Use o PHP Coding Standards Fixer do Fabien Potencier para automaticamente modificar seu código para que ele seja compatível com os padrões, evitando fazer as alterações na mão.

O Inglês é a linguagem preferida para todos os nomes simbólicos e para a infra-estrutura do código. Comentários devem ser escritos em qualquer linguagem que possa ser facilmente lida 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 funcções 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 teh vincluar closures com o escopo de objetos e também melhorou o suporte para invocaveis (callables) tanto que 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 varias formas de meta-programação atraves de macanismos 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()` ecallStatic().

- 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-0, que tem como objetivo fornecer uma convenção padrão para arquivos, classes e namespaces, permitindo código plug-and-play.

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

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 web para isso. Apenas tenha a certeza de não colocar seus scripts PHP CLI no seu web root público!

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 analise de cobertura de código 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 code localmente e depois testa o mesmo dentro de uma VM (máquina virtual) ou em outro servidor, a "Depuração Remota" é uma característica que você vai querer manter ativa deste 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, open-source e stand-alone XDebug GUI 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 possuia 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, pacotes Composer estarão disponíveis apenas em projetos que você especificar de forma explícita, enquanto que um pacote PEAR estaria 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 um gerenciador de dependências **brilhante** para o 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.

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

Você pode instalar o Composer localmente (no seu diretório de trabalho atual; embora isso não seja mais recomendado) ou globalmente (e.g. /usr/local/bin). Vamos assumir que você queira instalar o Composer localmente. A partir do diretório raiz do seu projeto:

```
curl -s http://getcomposer.org/installer | php
```

Isso irá baixar o `composer.phar` (um arquivo PHP binário). Você pode executá-lo com o `php` para gerenciar as dependências do seu projeto. **Por favor, Observe:** Se você passar o código baixado diretamente para um interpretador, por favor leia primeiro o código online para confirmar que ele é seguro.

Como instalar o Composer (manualmente)

Instalar o Composer manualmente é uma técnica avançada; no entanto, existem várias razões pelas quais um desenvolvedor poderia preferir esse método a usar a rotina de instalação interativa. A instalação interativa verifica sua instalação do PHP para garantir que:

- uma versão suficiente do PHP esteja sendo usada
- arquivos `.phar` possam ser executados corretamente
- permissões em certos diretórios sejam suficientes
- certas extensões problemáticas não estejam carregadas
- certas configurações no `php.ini` não estejam definidas

Como uma instalação manual não executa nenhuma dessas verificações, você precisa decidir se o custo valerá a pena para você. Se sim, segue abaixo como obter o Composer manualmente:

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

O caminho `$HOME/local/bin` (ou um diretório de sua escolha) deve estar na sua variável de ambiente `$PATH`. Isso fará com que o comando `composer` fique disponível.

Quando você vir a documentação dizendo para executar o Composer como `php composer.phar install`, você pode substituir por isso:

```
composer install
```

Como Definir e Instalar Dependências

Primeiramente, crie um arquivo `composer.json` no mesmo diretório do `composer.phar`. Aqui está um exemplo que lista o Twig como uma dependência do projeto. 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 `php composer.phar require` adiciona um 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. Execute no diretório raiz do seu projeto onde baixou o `composer.phar`:

```
{
    "require": {
        "twig/twig": "1.8.*"
    }
}
```

Outra alternativa é o comando `php composer.phar 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 `vendors/`. Isto também se aplica para projetos baixados que fornecem um arquivo `composer.json`:

```
php composer.phar 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 `php composer.phar install`. Se você compartilhar seu projeto com outros desenvolvedores e o arquivo `composer.lock` é parte da sua distribuição, quando eles executarem `php composer.phar install` receberão as mesmas versões como você. Para atualizar suas dependências, execute `php composer.phar update`. Isso é muito útil quando você define as versões requiridas. Por exemplo, a versão requerida de ~1.8 significa "qualquer coisa mais recente que 1.8.0, mas menos do que 2.0.x-dev". Você também pode usar o `*` curinga como `1.8.*`. Agora o comando `php composer.phar update` do Composer atualizará todas as suas dependências para a versão mais recente que se encaixa às restrições definidas.

Verificando suas dependências para as questões de segurança

O Security Advisories Checker é um serviço web e uma ferramenta de linha de comando, ambos examinarão seu arquivo `composer.lock` e diz se você precisa atualizar alguma das dependências.

- 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 termilogia PEAR) o repositório pear. Em seguida, na seção `"required"` 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:

```
$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. Entretando enquanto se avança na linguagem, nós frequentemente esquecemos do básico que tinhamos 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('m/d/Y') . "\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 no meio.

```
<?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('m/d/Y') . " ";
}
```

- Leia sobre a 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 comuns) 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 estiver usando uma framework, a maior parte do código de alto nível e a estrutura do projeto serão baseados na 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 utiliza-los na framework. Se, por outro lado, você não estiver utilizando uma framework para construir sua aplicação, então você terá que descobrir os padrões que se encaixam melhor para o tipo e tamanho da aplicação que você está construindo.

- Continue lendo em Design Patterns

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)
{
    // The validation failed
}
catch(Fuel\Email\SendingFailedException $e)
{
    // The driver could not send the email
}
```

SPL Exceptions

Por padrão uma exceção não tem significado nenhum, e a forma mais comum de dar um

significado é pela definição do seu próprio nome:

```
<?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 SPL Exceptions (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

Serialize

Como armazenar ou transmitir valores PHP sem perder sua naturalidade e sua estrutura? Por exemplo, você possui um objeto já com seus atributos definidos e deseja armazenar esse estado que ele se encontra agora para poder usar uma outra hora. Você pode utilizar a serialização para colocar todas as informações dentro de um texto, e quando você quiser utilizar novamente é só resgatar os valores. Mas como fazer isso?

Serializar Dados

Tendo uma estrutura de dados com valores você já pode guardar o seu atual estado. Como por exemplo em um array.

```
<?php
$array = array(1, 2, 3, 4, 5);
echo serialize($array);
// Resultado: a:5:{i:0;i:1;i:1;i:2;i:2;i:3;i:3;i:4;i:4;i:5;}
?>
```

Restaurar Dados Serializados

Para obter os valores de volta, basta utilizar a função unserialize e em seu parametro colocar o texto já serializado. Você terá como retorno a estrutura e os dados que foram serializados!

```
<?php
$arraySerialized = 'a:5:{i:0;i:1;i:1;i:2;i:2;i:3;i:3;i:4;i:4;i:5;}';
```

```
var_dump(unserialize($arraySerialized));
/* Resultado:
array
 0 => int 1
 1 => int 2
 2 => int 3
 3 => int 4
 4 => int 5 */
```

Isso pode ser utilizado para qualquer conjunto de dados, como objeto, array, entre outros. Caso o valor do parâmetro do unserialize não for texto ele dispara um E_NOTICE, para que isso não ocorra coloque o operador @ na função unserialize (@unserialize($arraySerialized))

- Leia sobre Serialize

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 como seu banco de dados. A opção recomendada *até o PHP 5.1.0* era usar os drivers nativos, como o mysql, o mysqli, o pgsql etc.

Drivers nativos são excelentes se você estiver usando apenas UM banco de dados na 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.

Uma observação adicional sobre os drivers nativos: a extensão mysql para o PHP não está mais em desenvolvimento ativo , e a situação oficial desde o PHP 5.4.0 é "Long term deprecation", ou "Obsoleta no longo prazo". Isso significa que ela será removida dentro de alguns lançamentos das próximas versões, assim, por volta do PHP 5.6 (ou o que venha depois do 5.5), ela pode muito bem ter desaparecido. Se você estiver usando `mysql_connect()` e `mysql_query()` nas 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

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. 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. Esse 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:users.db');
$pdo->query("SELECT name FROM users WHERE id = " . $_GET['id']); // <-- NO!
```

Esse código é péssimo. Você está inserindo um parâmetro bruto na sua consulta SQL. Isso fará você ser hackeado num piscar de olhos. Apenas imagine se um hacker passar como parâmetro um `id` criativo 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:users.db');
$stmt = $pdo->prepare('SELECT name FROM users WHERE id = :id');
$stmt->bindParam(':id', $_GET['id'], PDO::PARAM_INT); //<-- Higienizado automa
ticamente sanitized 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 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 referencias a ele forem excluidas, 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

Camadas de Abstração

Muitos frameworks fornecem sua própria camada de abstração que pode ou não ser baseada na PDO. Elas frequentemente irão emular funcionalidades de um sistema de banco de dados que não

estão presentes nos outros, embutindo suas consultas SQL em métodos PHP, te dando abstração real de banco de dados. Isso, naturalmente, adiciona uma pequena sobrecarga, mas se você estiver construindo uma aplicação portátil que precise trabalhar com o MySQL, o PostgreSQL e o SQLite então uma pequena sobrecarga valerá a pena em função da clareza no código.

Algumas camadas de abstração foram construídas usando o padrão de namespaces PSR-0, por isso podem ser instaladas em qualquer aplicação que você quiser:

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

Erros e Exceções

Segurança

Erros

O PHP possui vários níveis de severidade de erros. Os três tipos mais comuns de mensagens são erros, avisos e advertências. Esses possuem diferentes níveis de severidade: `E_ERROR`, `E_NOTICE` e `E_WARNING`. Erros são erros fatais de tempo de execução e são geralmente causados por falhas no seu código e precisam ser corrigidos, pois eles irão fazer com que a execução do PHP seja interrompida. Advertências não são erros fatais e a execução do script não será interrompida. Avisos são mensagens indicativas causadas pelo código que podem ou não causar problemas durante a execução do script e a execução não será interrompida.

Um outro tipo de mensagem de erro reportada em tempo de compilação é o `E_STRICT`. Essas mensagens são usadas para sugerir mudanças no seu código para ajudar a garantir uma melhor interoperabilidade e compatibilidade do seu código.

- Constantes Pré-definidas para Tratamento de Erros

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. Hash da senha é irreversível, uma função executada contra a senha do usuário. Isto produz uma seqüência 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 (inflezmente) 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

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` and `$_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
- 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_AL` 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`

Produção

Para esconder os erros no seu ambiente de **produção**, configure seu `php.ini` assim:

- display_errors: Off
- error_reporting: E_ALL
- log_errors: On

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
- 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 estórias humanamente legíveis que descrevem o comportamento da sua aplicação. Essas estórias 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 helpers 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.

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. Se o orçamento de seu projeto permitir, você

deveria evitar utilizar servidores compartilhados.

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. Ele também está disponível para versões anteriores.

Caches de bytecode populares são:

- 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 dep
ois
    $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

“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 with 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

Leitura Adicional:

- Automatize seu projeto com Apache Ant
- Maven, um framework de implementação baseado em Ant e como usá-lo com PHP

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

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 um-para-um na comunidade PHP.

Fornecedores de PaaS PHP

- PagodaBox
- AppFog
- Heroku (PHP support is undocumented but based on stable Facebook partnership link)
- 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, entre outros componentes embutidos.

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.

- Frameworks PHP Populares

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 frameworks baseados em componentes, permite que você use seus componentes com os requisistos mínimos (ou nenhum requisito). Por exemplo, você pode usar o FuelPHP Validation package sem precisar usar o framework FuelPHP em si. Este projetos são essencialmente apenas mais um repositório de componentes reusáveis:

- Aura
- FuelPHP (2.0 only)
- Laravel's "Componentes Iluminados"
- Symfony Components

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.

Veja o Calendário de Eventos Oficiais do PHP

Grupos de Usuários PHP

Se você vive em uma cidade grande, provavelmente tem um grupo de usuários PHP por perto.

Embora não exista ainda uma lista oficial de PUGs, você pode encontrar facilmente seu PUG local buscando no Google ou no Meetup.com. Se você vive em uma cidade menor, talvez não exista um PUG local; se for o caso, comece um!

Leia sobre Grupos de Usuários na PHP Wiki

Conferências PHP

A comunidade PHP também oferece grandes conferências regionais e nacionais em muitos países no mundo todo. Membros bem conhecidos da comunidade PHP geralmente palestram nesses grandes eventos, por isso eles são uma excelente oportunidade para aprender diretamente dos líderes do setor.

Encontre uma Conferência PHP

Criado e mantido por
Josh Lockhart

Colaboradores do projeto
Kris Jordan
Phil Sturgeon

Contribuintes
Este projeto não seria possível sem a ajuda de nossos contribuintes no GitHub.

Patrocinadores do projeto
New Media Campaigns