LIVRO PHP MODERNO

Capítulo 1. O Novo PHP

A linguagem PHP está experimentando um renascimento. PHP está se transformando em uma linguagem de script moderna com características úteis, como namespaces, traits, closures , e um built-in opcode cache. O moderno ecossistema PHP está evoluindo também. desenvolvedores PHP dependem menos de estruturas monolíticas e mais em componentes especializados menores. o Composer gerente de dependência está revolucionando a forma como vamos construir aplicações PHP; que nos emancipa do jardim murado de um quadro e nos permite misturar e combinar PHP interoperável componentes mais adequados para as nossas aplicações personalizadas PHP. interoperabilidade dos componentes não seria possível sem os padrões da comunidade proposto e com curadoria do Framewok PHP Grupo Interop.
PHP moderno é o seu guia para o novo PHP, e vai mostrar-lhe como construir e implantar aplicações surpreendentes com PHP usando padrões da comunidade, boas práticas, e componentes interoperáveis.

Passado

Antes de explorarmos o PHP moderno, é importante compreender a origem do PHP. PHP é uma linguagem interpretadora de script do lado do servidor. Isto significa que você escreve código PHP, envia ele para um servidor web, e executa com um intérprete. PHP é normalmente usada com um servidor web Apache ou nginx para servir conteúdo dinâmico. No entanto, PHP também podem ser usados para construir aplicações de linha de comando poderosos (como bash, Ruby, Python, e assim por diante). Muitos desenvolvedores PHP não percebem isso e perdem uma característica muito emocionante. Não é você, apesar de tudo.
Você pode ler a história oficial do PHP na página http://php.net/manual/history.php.php. Eu não vou repetir o que já foi dito tão bem por Rasmus Lerdorf (o criador do PHP). O que eu vou dizer-lhe é que o PHP tem um passado tumultuado. PHP começou como uma coleção de scripts CGI escrito por Rasmus Lerdorf para rastrear as visitas ao seu currículo online. Lerdorf nomeou seu conjunto de scripts CGI "Personal Home Page Tools." Esta primeira encarnação foi completamente diferente a partir do PHP que conhecemos hoje. A primeiras ferramentas PHP de Lerdorf não eram uma linguagem de script; eles eram ferramentas que forneceram variáveis rudimentares e variável forma automáticainterpretação usando uma sintaxe embutida no HTML.
Entre 1994 e 1998, PHP sofreu inúmeras revisões e ainda recebeu alguns reescreve ground-up . Andi Gutmans e Zeev Suraski, dois desenvolvedores de Tel Aviv, juntoram forças com Rasmus Lerdorf para transformar PHP a partir de uma pequena coleção de ferramentas de CGI em uma linguagem de programação completa com uma sintaxe mais consistente e suporte básico para programação orientada a objetos. Eles nomearam seu PHP produto final 3 e lançaram no final de 1998. O novo apelido PHP foi uma partida de nomes anteriores, e é uma recursiva acrônimo para PHP: Hypertext Preprocessor. PHP 3 foi a primeira versão que a maioria assemelhava ao PHP que nós conhecemos hoje. Ele forneceu a extensibilidade superior a várias bases de dados, protocolos e APIs. extensibilidade do PHP 3 atraiu muitos novos desenvolvedores para o projeto.
Ao final de 1998, o PHP 3 já foi instalado em um escalonamento de 10% dos servidores web do mundo.

Presente

Hoje, a linguagem PHP está evoluindo rapidamente e é apoiada por dezenas de equipes de desenvolvedores de todo o mundo. práticas de desenvolvimento mudaram também. No passado, era prática comum para gravar um arquivo PHP, enviá-lo para um servidor de produção com FTP, e espero que ele trabalhou. Esta é uma estratégia de desenvolvimento terrível, mas foi necessário devido à falta de ambientes de desenvolvimento locais viáveis.
Hoje em dia, nós evitamos o FTP e utilizamos controle de versão em vez disso. software de controle de versão como Git ajudam a manter um histórico de código auditável que pode ser ramificada, bifurcada, e se fundiram.
Ambientes de desenvolvimento locais são idênticos aos servidores de produção graças a ferramentas de virtualização como Vagrant e provisionamento ferramentas como Ansible, Chef e Puppet.
Nós alavancagem especializada componentes PHP com o gerente de dependência Composer .
Nosso código PHP adere ao PSR - normas comunitárias geridas pelo Grupo PHP Framework Interop. Nós absolutamente testamos nosso código com ferramentas como o PHPUnit. Nós implantar nossos aplicativos com PHP FastCGI gerente de processo atrás de um servidor web como o nginx. E nós aumentamos o desempenho do aplicativo com um cache opcode.
PHP moderna engloba muitas práticas novas que podem ser desconhecidos para aqueles de vocês de novo para PHP, ou para aqueles atualizando a partir de versões mais antigas do PHP. Não se sinta sobrecarregado. Eu andarei através de cada conceito mais adiante neste livro.
Eu também estou animado que o PHP tem agora um projecto de especificação oficial - algo que faltava até 2014.

Nota

A maioria das linguagens de programação maduras têm uma especificação. Em termos leigos, a especificação é um modelo canônico que define o que significa ser PHP. Este plano é usado por desenvolvedores que criam programas que analisam, interpretar e executar código PHP. Isto não é para desenvolvedores que criam aplicativos e sites com PHP.
Sara Golemon e Facebook anunciaram o projecto de primeira especificação PHP na O'Reilly de conferência OSCON em 2014. Você pode ler o anúncio oficial sobre o funcionamento interno do PHP mailing list, e você pode ler a especificação PHP no GitHub.
Uma especificação oficial linguagem PHP é cada vez mais importante, dada a introdução de múltiplos motores de PHP concorrentes. O motor de PHP original é o Zend Engine, um PHP intérprete escrito em C e introduzido no PHP 4. A Zend Engine foi criado por Rasmus Lerdorf, Andi Gutmans, e Zeev Suraski. Hoje, a Zend Engine é a Zend empresa de A principal contribuição para a comunidade PHP. No entanto, existe agora uma segunda grande motor PHP - a Máquina Virtual HipHop do Facebook. A garante especificação de linguagem que ambos os motores de manter um grau de compatibilidade de linha de base.

Nota

Um motor de PHP é um programa que analisa, interpreta e executa o código PHP (por exemplo, o Zend Motor ou HipHop Virtual Machine do Facebook). Isto não é para ser confundido com PHP, que é uma referência genérica para a linguagem PHP.

Futuro

A Zend Engine está melhorando em um ritmo rápido, com novas funcionalidades e melhor desempenho. Eu atribuo as melhorias Zend Engine à sua nova competição, especificamente HipHop Virtual Machine do Facebook e linguagem de programação Hack .
Hack é uma nova linguagem de programação construída em cima de PHP. Introduz tipagem estática, novas estruturas de dados e interfaces adicionais, mantendo a compatibilidade com versões anteriores com código PHP tipagem dinâmica existente. Hack é voltada para desenvolvedores que apreciam PHP características rápido desenvolvimento, mas precisa da previsibilidade e estabilidade da tipagem estática.

Nota

Vamos discutir dinâmica contra tipagem estática mais adiante neste livro. A diferença entre o dois é quando os tipos de PHP são verificados. tipos dinâmicos são verificados em tempo de execução, enquanto estática tipos são verificados em tempo de compilação. Ir direto para o Capítulo 12 para mais informações.

O HipHop Virtual Machine (HHVM) é um PHP e intérprete Hack que utiliza uma só no time (JIT) para melhorar o desempenho do aplicativo e reduzir o uso de memória.
Não prevejo Hack and HHVM substituindo o Zend Engine, mas o Facebook de novo contribuições estão criando um respingo gigante na comunidade PHP. O aumento da concorrência fez com que a equipe principal Zend Engine para anunciar PHP 7, um Zend Engine otimizado Diz-se que a par com HHVM. Vamos discutir estes desenvolvimentos ainda mais no Capítulo 12.
É um momento emocionante para ser um programador PHP. A comunidade PHP nunca foi tão energizado, divertida e inovadora. Espero que este livro ajuda você firmemente abraçar o PHP práticas modernas. Há uma tonelada de coisas novas para aprender, e muitas mais coisas no horizonte.
Considere este o seu roteiro. Agora vamos começar.

Capítulo 2. Características

A linguagem PHP moderna tem muitos novos recursos interessantes. Muitos desses recursos será novo para programadores PHP atualização de versões anteriores, e eles vão ser uma boa surpresa para programadores migração para o PHP de outro idioma. Esses novos recursos tornar a linguagem PHP uma plataforma poderosa e proporcionar uma experiência agradável para os aplicações web de construção e ferramentas de linha de comando.
Algumas destas características não são essenciais, mas eles ainda facilitam nossas vidas. Alguns recursos, contudo, são essenciais. Namespaces, por exemplo, são uma pedra angular do PHP moderna normas e permitir práticas de desenvolvimento que os modernos desenvolvedores PHP exame para concedido (Por exemplo, carregamento automático). Vou apresentar cada novo recurso, explicar por que é útil, e mostrar-lhe como implementá-lo em seus próprios projetos.

Dica

Encorajo-vos a acompanhar no seu próprio computador. Você pode encontrar todo o código do texto exemplos companheiro repositório GitHub do livro.

Namespaces

Se há um moderno recurso de PHP eu quero que você saiba, é **namespaces**. introduzido em PHP 5.3.0, **namespaces** são uma importante ferramenta que organiza o código PHP em uma hierarquia, comparável a estrutura de diretórios do sistema de arquivos do seu sistema operacional. Cada componente de PHP moderno e estrutura organiza seu código globalmente sob a seu próprio fornecedor namespace exclusivo para que não entre em conflito com, ou reclamar, classe comum nomes usados por outros fornecedores.

Nota

Você não odeia quando você anda em uma loja de café e essa pessoa detestável tem uma confusão de livros, cabos e outros enfeites espalhados por várias mesas? Sem mencionar que ele é sentado ao lado, mas não usar, a única tomada disponível. Ele está perdendo espaço valioso que poderiam ser úteis para você. Figurativamente falando, essa pessoa não está usando **namespaces**. Não seja essa pessoa.

Vamos ver como um componente PHP do mundo real usa namespaces. O Framework Symfony próprio symfony / HttpFoundation é um componente PHP popular que gerencia solicitações HTTP e respostas. Mais importante, o componente symfony / HttpFoundation usa comum nomes de classe PHP como Request, Response, e Cookie. Eu garanto que você há muitos outros componentes de PHP que usam os mesmos nomes de classe. Como podemos usar a componente de PHP / HttpFoundation symfony se outro código PHP usa os mesmos nomes de classe?
Podemos usar com segurança o componente symfony / HttpFoundation precisamente porque seu código é sandboxed abaixo do namespace único fornecedor Symfony. Visite a symfony componente HttpFoundation / no GitHub e navegue até o arquivo Response.php. isto parece com a Figura 2-1.

Olhe atentamente para linha 12. Ele contém este código:

namespace Symfony\Component\HttpFoundation;

Esta é uma declaração PHP namespace, e sempre aparece em uma nova linha imediatamente após a abertura tag <?php . Esta declaração de namespace em particular nos diz várias coisas. Em primeiro lugar, sabemos que a vida da classe Response abaixo do namespace fornecedor Symfony (o fornecedor namespace é o namespace de nível superior). Sabemos que a classe Response vive debaixo o subnamespace Component. Sabemos também as vidas de classe Response abaixo ainda outra subnamespace chamado HttpFoundation. Você pode ver outros arquivos adjacentes Response.php, e você vai ver que eles usam a mesma declaração de namespace. Um espaço para nome (ou subnamespace) encapsula e organiza classes PHP relacionados, assim como um sistema de arquivos diretório contém arquivos relacionados.

Dica

Subnamespaces são separados com um caractere \.
Ao contrário do sistema de arquivos físico do seu sistema operacional, namespaces PHP são um conceito virtual e não necessariamente um mapa 1: 1 com diretórios de arquivos. Dito isto, a maioria PHP componentes que, de fato, mapeia subnamespaces ao sistema de arquivos diretórios para compatibilidade com o popular PSR-4 padrão autoloader (falaremos mais sobre isso no Capítulo 3).
Nota Tecnicamente falando, namespaces são apenas uma notação linguagem PHP referenciado pelo PHP intérprete para aplicar um prefixo do nome comum a um conjunto de classes, interfaces, funções, e constantes.

Porque nós usamos Namespaces

Namespaces são importantes porque vamos criar o código de área restrita, que trabalha ao lado código de outros desenvolvedores. Este é o conceito fundamental do componente PHP moderna ecossistema. Componentes e quadro autores construir e distribuir o código para um grande número de desenvolvedores PHP, e eles têm nenhuma maneira de saber ou controlar o que as classes,
interfaces, funções e constantes são usados juntamente com o seu próprio código. Esse problema aplica-se a seus próprios projetos em casa, também. Se você escrever componentes personalizados do PHP ou
classes para um projeto, que o código deve trabalhar ao lado de terceiros do seu projeto dependências.
Como mencionei anteriormente com o componente symfony / HttpFoundation, seu código e código de outros desenvolvedores podem usar a mesma classe, interface, função ou nomes de constantes.
Sem espaços de nomes, um nome de colisão faz com que o PHP falhe. Com namespaces, seu código e código de outros desenvolvedores podem usar a mesma classe, interface, função ou nome de constante assumindo que o seu código vive debaixo de um namespace fornecedor único.
Se você está construindo um pequeno projeto pessoal com apenas algumas dependências, nome da classe closures provavelmente não será um problema. Mas quando você está trabalhando com uma equipe de construção de uma grande

projeto com inúmeras dependências de terceiros, closures de nomes tornam-se uma preocupação real. Você não pode controlar quais classes, interfaces, funções e constantes são introduzidas no espaço global por dependências do seu projeto. Isso é por que namespacing seu código é importante.

Declaração

Cada classe PHP, interface, função e vidas constantes abaixo de um espaço de nomes (ou subnamespace). Namespaces são declaradas na parte superior de um arquivo PHP em uma nova linha imediatamente após a abertura tag <? php. A declaração namespace começa com namespace, em seguida, um caractere de espaço, então o nome do espaço para nome, e depois um fechamento ponto e vírgula ; .
Lembre-se que namespaces são muitas vezes utilizados para estabelecer um nome do fornecedor de nível superior. Este exemplo de declaração namespace estabelece o nome do fornecedor Oreilly:

```
<?php
namespace Oreilly;
```

Todos as classes PHP, interfaces, funções ou constantes declaradas abaixo desta declaração namespace de viver no namespace Oreilly e são, de alguma forma, relacionados com O'Reilly Meios de comunicação. O que se queria para organizar o código relacionado com este livro? Nós usamos um subnamespace.
Subnamespaces são declarados exactamente o mesmo que no exemplo anterior. O único diferença é que nós separar nomes de namespace e subnamespace com a \ personagem.
O exemplo a seguir declara uma subnamespace chamado ModernPHP que vive abaixo da superior Oreilly namespace fornecedor:

```
<?php
namespace Oreilly\ModernPHP;
```

Todas as classes, interfaces, funções e constantes declaradas abaixo desta namespace declaração de viver no subnamespace Oreilly \ ModernPHP e são, de alguma forma, relacionado com
este livro.
Todas as classes no mesmo espaço para nome ou subnamespace não tem que ser declarado na mesma
arquivo PHP. Você pode especificar um espaço para nome ou subnamespace no topo de qualquer arquivo PHP, e
código que do arquivo torna-se uma parte desse espaço de nomes ou subnamespace. Isto torna possível
para escrever várias classes em arquivos separados que pertencem a um namespace comum.
Dica
O namespace mais importante é o namespace fornecedor. Este é o namespace de nível superior que identifica a sua marca ou organização, e deve ser globalmente únicos. subnamespaces são menos importantes, mas eles são úteis para a organização de código do seu projeto.