PHP Estruturado

Índice

Advertência

Esta apostila é fruto de pesquisa por vários sites e autores e com alguma contribuição pessoal. A maior fonte de pesquisa foi o sie oficial do PHP, especialmente a parte teórica e alguns exemplos. Geralmente o devido crédito é concedido, exceto quando eu não mais encontro o site de origem. Também segui recomendações de um documento muito importante atualmente, PHP do Jeito Certo (http://br.phptherightway.com/ e https://github.com/PHPSP/php-the-right-way). Caso identifique algo que seja de sua autoria e não tenha o crédito, poderá entrar em contato comigo para dar o crédito ou para remover: ribafs @ gmail.com.

É importante ressaltar que esta apostila é distribuída gratuitamente, originalmente no repositório:

https://github.com/ribafs/apostilas

Sugestões

Sugestões serão bem vindas, assim como aviso de erros no português ou no PHP. Crie um issue no repositório ou me envie um e-mail ribafs @ gmail.com.

Autor

Ribamar FS

ribafs @ gmail.com

https://ribamar.net.br

https://github.com/ribafs

Fortaleza, 11 de dezembro de 2021

Recomendações

O conhecimento teórico é importante para que entendamos como as coisas funcionam e sua origem, mas para consolidar um conhecimento e nos dar segurança precisa de prática e muita prática.

Não vale muito a penas ser apenas mais um programador. É importante e útil aprender muito, muito mesmo, ao ponto de sentir forte segurança do que sabe e começar a transmitir isso para os demais.

Tenha seu ambiente de programação em seu desktop para testar com praticidade todo o código sugerido.

Caso esteja iniciando em PHP recomendo que leia por inteiro e em sequência. Mas se já entende algo dê uma olhada no índice e vá aos assuntos que talvez ainda não domine.

Não esqueça de ver e testar também o conteúdo da pasta Anexos.

Caso tenha alguma dúvida, me parece que a forma mais prática de receber resposta é através de grupos. Temos também o Google que geralmente ajuda.

Dica: quando tiver uma dúvida não corra para pedir ajuda. Antes analise o problema, empenhe-se em entender o contexto e procure você mesmo resolver. Assim você está passando a responsabilidade para si mesmo, para seu cérebro, que passará a ser mais confiante e poderá te ajudar ainda mais. É muito importante que você confie em si mesmo, que é capaz de solucionar os problemas. Isso vai te ajudar. Somente depois de tentar bastante então procure ajuda.

Veja que este material não tem vídeo nem mesmo uma única imagem. Isso em si nos nossos dias é algo que atrai pouca gente, pois vídeos e fotos são mais confortáveis de ler e acompanhar. Ler um texto como este requer mais motivação e empenho. Lembrando que para ser um bom programador precisamos ser daqueles capaz de se empenhar e suportar a leitura e escrita.

Lembrando que a melhor documentação sobre o PHP é o site oficial, que é todo escrito.

Autodidata

Não tive a pretensão de que esta apostila fosse completa, contendo tudo sobre PHP, que seria praticamente impossível. Só para exemplificar o capítulo sobre funções nativas aborda apenas algumas das diversas funções do PHP. Um dos itens é sobre as funções matemáticas, que traz apenas algumas das in úmeras que o PHP entrega. Falo isso para ressaltar que não existe livro, curso, apostila que ensine tudo sobre PHP e isso mostra a importância de você ser alguém com espírito autodidata. Aquele programador que quando não sabe algo seja capaz de pesquisar, estudar, testar e praticamente sozinho. Este é um profissional importante para as empresas e procurado.

1 - Introdução

1.1 - O que é PHP

O PHP (um acrônimo recursivo para PHP: Hypertext Preprocessor) é uma linguagem de script open source de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento web e que pode ser embutida dentro do HTML.

O que distingue o PHP de algo como o JavaScript no lado do cliente é que o código é executado no servidor, gerando o HTML que é então enviado para o navegador. O navegador recebe os resultados da execução desse script, mas não sabe qual era o código fonte. Você pode inclusive configurar seu servidor web para processar todos os seus arquivos HTML com o PHP, e então não há como os usuários dizerem o que você tem na sua manga.

A melhor coisa em usar o PHP é que ele é extremamente simples para um iniciante, mas oferece muitos recursos avançados para um programador profissional. Não tenha medo de ler a longa lista de recursos do PHP. Pode entrar com tudo, o mais rápido que puder, e começar a escrever scripts simples em poucas horas.

Apesar do desenvolvimento do PHP ser focado nos scripts do lado do servidor, você pode fazer muito mais com ele. Veja sobre isso na seção O que o PHP pode fazer?, ou vá diretamente para o tutorial introdutório se você estiver interessado apenas em programação web.

1.2 – História

A linguagem foi criada em 1994 e o código fonte do PHP só foi liberado em 1995, como um pacote de problemas CGI criados por Rasmus Lerdorf, com o nome *Personal Home Page Tools*, para substituir um conjunto de scripts Perl que ele usava no desenvolvimento de sua página pessoal. Em 1997 foi lançado o novo pacote da linguagem com o nome de PHP/FI, trazendo a ferramenta *Forms Interpreter*, um interpretador de comandos SQL. Mais tarde, Zeev Suraski desenvolveu o analisador do PHP 3 que contava com o primeiro recurso de orientação a objetos, que dava poder de alcançar alguns pacotes, tinha herança e dava aos desenvolvedores somente a possibilidade de implementar propriedades e métodos.[4][5] Pouco depois, Zeev e Andi Gutmans, escreveram o PHP 4, abandonando por completo o PHP 3, dando mais poder à máquina da linguagem e maior número de recursos de orientação a objetos. O problema sério que apresentou o PHP 4 foi a criação de cópias de objetos, pois a linguagem ainda não trabalhava com apontadores ou *handlers*, como são as linguagens Java, Ruby e outras. O problema fora resolvido na versão atual do PHP, a versão 5, que já trabalha com *handlers*. Caso se copie um objeto, na verdade copiaremos um apontador, pois, caso haja alguma mudança na versão original do objeto, todas as outras também sofrem a alteração, o que não acontecia na PHP 4.[6]

Trata-se de uma linguagem extremamente modularizada, o que a torna ideal para instalação e uso em servidores web. Diversos módulos são criados no repositório de extensões PECL (PHP Extension Community Library) e alguns destes módulos são introduzidos como padrão em novas versões da linguagem. É muito parecida, em tipos de dados, sintaxe e mesmo funções, com a linguagem C e com a C++. Pode ser, dependendo da configuração do servidor, embarcada no código HTML. Existem versões do PHP disponíveis para os seguintes sistemas operacionais:

Windows, Linux, FreeBSD, Mac OS, OS/2, AS/400, Novell Netware, RISC OS, AIX, IRIX e Solaris.

Construir uma página dinâmica baseada em bases de dados é simples com PHP, (em parte, vale lembrar), este provê suporte a um grande número de bases de dados: Oracle, Sybase, PostgreSQL, InterBase, MySQL, SQLite, MSSQL, Firebird, etc., podendo abstrair o banco com a biblioteca ADOdb, entre outras. A Wikipédia funciona sobre um software inteiramente escrito em PHP, usando bases de dados MySQL: o MediaWiki.[6]

PHP tem suporte aos protocolos: IMAP, SNMP, NNTP, POP3, HTTP, LDAP, XML-RPC, SOAP. É possível abrir *sockets* e interagir com outros protocolos. E as bibliotecas de terceiros expandem ainda mais estas funcionalidades. Existem iniciativas para utilizar o PHP como linguagem de programação de sistemas fixos. A mais notável é a PHP-GTK. Trata-se de um conjunto do PHP com a biblioteca GTK, portada do C++, fazendo assim softwares inter-operacionais entre Windows e Linux. Na prática, essa extensão tem sido muito pouco utilizada para projetos reais.[6]

O acrônimo recursivo PHP representa um elefante, que é conhecido como o mascote da linguagem.

Observação: este volume cuida somente do PHP estruturado. PHP orientado a objetos e MVC serão abordados em outros volumes.

Licença
PHP é um software gratuito e de código aberto disponível sob a PHP License, que afirma:[7]

> Produtos derivados deste software não devem ser chamado de PHP, nem pode conter "PHP" em seu nome, sem prévia permissão por escrito da group@php.net. Você pode indicar que o software funciona em conjunto com o PHP, dizendo "Foo para PHP", em vez de chamá-lo "PHP Foo" ou "phpfoo".

Esta restrição no uso do nome *PHP* torna-o incompatível com a GNU General Public License (GPL).[8]

PHP 6 e Unicode
PHP recebeu diversas críticas por não ter suporte nativo a Unicode.[9][10] Em 2005, um projeto liderado por Andrei Zmievski foi iniciado para trazer esse dito suporte ao PHP através da incorporação da biblioteca International Components for Unicode (ICU) para poder passar-se a usar a codificação UTF-16.[11] Uma vez que isso causaria grandes mudanças tanto no código fonte como para o usuário, foi planejado lançá-la na versão 6.0 em conjunto com outros importantes recursos, então em desenvolvimento, em vez da 5.5.[12]

Entretanto, devido a falta de desenvolvedores que entendessem as mudanças necessárias e problemas de desempenho decorrentes da conversão para UTF-16, que raramente é usado em um contexto web, levou a atrasos no projeto. Como resultado, o PHP 5.3 foi lançado em 2009, sem total suporte ao Unicode, mas contendo algumas das novidades que seriam lançadas no PHP 6.0. Em março de 2010, o projeto em sua forma atual foi oficialmente abandonado, e uma versão 5.4 do PHP foi feita ainda sem total suporte a Unicode, também contendo as novidades que seriam lançadas no PHP 6.0.[13] Esperanças iniciais eram de que um novo plano seria formado para ter a integração Unicode, mas a partir de 2014 nenhum foi adotado.

Durante os anos, antes do lançamento do PHP 5.3 e 5.4, alguns livros foram publicados com base no conjunto de recursos esperado de PHP 6.0, incluindo o suporte a Unicode e os recursos que depois foram trazidos para outros lançamentos. Há, portanto, algum debate sobre se uma nova versão principal do PHP, com ou sem suporte a Unicode, deve ser chamado de "PHP 6", ou se a nomenclatura deve ser ignorado para evitar confusão.

Histórico de versões
(Wikipedia)

Versão	Data de lançamento original	Última versão	Suportada até[14][15]	Notas
1.0	8 de junho de 1995	?	?	Oficialmente chamado de "Personal Home Page Tools (PHP Tools)" (Ferramentas para página pessoal). Este foi o primeiro uso para o nome "PHP".
2.0	1 de novembro de 1997	?	?	Considerado pelo seu criador como a "mais rápida e simples ferramenta" para criar páginas dinâmicas para a Web.
3.0	6 de junho de 1998	3.0.18	20 de outubro de 2000	O desenvolvimento passou a ser feito por vários desenvolvedores em colaboração. Zeev Suraski e Andi Gutmans reescreveram toda a base do PHP nesta versão.
4.0	22 de maio de 2000	4.0.6	23 de junho de 2001	Foi adicionado um melhor sistema de análise sintática (parser) chamado de motor Zend (Zend engine).[16]
4.1	10 de dezembro de 2001	4.1.2	12 de março de 2002	Introduzidas as 'superglobais' ($_GET, $_POST, $_SESSION, etc.)[16]
4.2	22 de abril de 2002	4.2.3	6 de setembro de 2002	A register_globals passou agora a estar desativada por padrão. Dados recebidos via rede são mais inseridos no escopo de variável global, fechando possíveis brechas de segurança.[16]
4.3	27 de dezembro de 2002	4.3.11	31 de março de 2005	Introduziu sua interface de linha de comando (command-line interface - CLI), para complementar o CGI.[16][17]
4.4	11 de julho de 2005	4.4.9	7 de agosto de 2008	Adicionadas as páginas do manual para os scripts phpize e php-config.[16]
5.0	13 de julho de 2004	5.0.5	5 de setembro de 2005	Zend Engine II com um novo modelo de objeto.[18]
5.1	24 de novembro de 2005	5.1.6	24 de agosto de 2006	Melhorias na performance com a introdução de variáveis de compilação na reengenharia do motor PHP.[18] Adicionada biblioteca *PHP Data Objects* (PDO) como uma nova interface de acesso aos bancos de dados.[19][20]
5.2	2 de novembro de 2006	5.2.17	6 de janeiro de 2011	Habilitado por padrão o filtro de extensões. Suporte ao JSON nativo.[18][21]
5.3	30 de junho de 2009	5.3.29	14 de agosto de	Suporte a espaço de nomes (*namespace*), vinculação de nomes (*late static bindings*), rótulos de salto de

			2014	código (goto limitado), clausura nativa, arquivos PHP nativos (phar), coletor de lixo para referências circulares, suporte ao Windows melhorado, sqlite3, mysqlnd em substituição a libmysql como biblioteca de extensão de trabalho com MySQL, fileinfo em substituição ao mime_magic para um melhor suporte ao MIME, extensão de internacionalização, e descontinuidade da extensão ereg.[22]
5.4	1 de março de 2012	5.4.45	3 de setembro de 2015	Suporte à trait, suporte a uma versão mais curta na sintaxe de vetores. Items removidos: register_globals, safe_mode, allow_call_time_pass_reference, session_register(), session_unregister() and session_is_registered(). Servidor web embutido.[23]. Várias melhorias nas fucionalidades já existentes e na performance. Redução dos requerimentos de memória. [24]
5.5	20 de junho de 2013	5.5.38	21 de julho de 2016	Suporte para geradores, blocos finally para tratamento de exceções, OpCache (baseado em Zend Optimizer+) empacotado na distribuição oficial.[25]
5.6	28 de agosto de 2014	5.6.40	31 de dezembro de 2018	Expressões escalares constantes, funções variádicas, desempacotamento de argumento, novo operador de exponenciação, extensões da instrução use para funções e constantes, novo depurador phpdbg como um módulo SAPI e outras melhorias menores.[26]
6.x	Não foi lançada	—	—	Versão abandonada do PHP que planejava incluir suporte nativo ao Unicode.
7.0	3 de dezembro de 2015	7.0.33	10 de janeiro de 2019	Zend Engine 3 (melhorias de desempenho e suporte a inteiros de 64 bits no Windows), sintaxe de variável uniforme, processo de compilação baseado em árvore sintática abstrata, adicionado Closure::call(), consistência de deslocamento bit a bit entre plataformas, operador ?? (coalescência nula), sintaxe de escape de ponto de código Unicode, declarações de tipo de retorno, declarações de tipo escalar (inteiro, flutuante, string e boolean), operador de comparação de três vias "nave espacial" <=>, delegação de gerador, classes anônimas, API CSPRNG mais simples e disponível de maneira mais consistente, substituição de muitos "erros" internos restantes do PHP pelas exceções mais modernas e sintaxe abreviada para importar vários itens de um espaço de nomes.[27]
7.1	1 de dezembro de 2016	7.1.33	1 de dezembro de 2019	Tipo de retorno void, modificadores de visibilidade de constante de classe.[28]
7.2	30 de novembro de 2017	7.2.34	30 de novembro de 2020	Parâmetro de objeto e declaração de tipo de retorno, extensão Libsodium, substituição de método abstrato, ampliação de tipo de parâmetro.[29]
7.3	6 de dezembro de 2018	7.3.33	6 de dezembro de 2021	Sintaxe flexível Heredoc e Nowdoc, suporte para atribuição de referência e desconstrução de array com list(), suporte à PCRE2, função hrtime().[30]

7.4	28 de novembro de 2019	7.4.26	28 de novembro de 2022	Propriedades tipadas 2.0, pré-carregamento, operador de atribuição de coalescência nula, openssl_random_pseudo_bytes melhorada, referências fracas, FFI - interface de função externa, extensão hash sempre disponível, registro de hash de senha, divisão de string multibyte, reflexão para referências, remoção de ext/wddx, novo mecanismo de serialização de objeto personalizado.[31]
8.0	26 de novembro de 2020	8.0.13	26 de novembro de 2023	Compilação Just-In-Time (JIT), arrays começando com um índice negativo, semântica de linguagem mais rígida/sã (validação para métodos de traços abstratos), comparações de string para números mais sãs, strings numéricas mais sãs, TypeError em operadores aritméticos/bit-a-bit inválidos, reclassificação de vários erros de mecanismo, erros de tipo consistentes para funções internas, erro fatal para assinaturas de método incompatíveis), conversão de float para string independente de localidade, ajustes de sintaxe variável, atributos, argumentos nomeados, expressão de correspondência, promoção de propriedade do construtor, tipos em união, tipo misto, tipo de retorno *static*, operador *nullsafe*, *non-capturing catches*, expressão *throw*, extensão JSON está sempre disponível.[32]
8.1	25 de novembro de 2021	8.1.0	25 de novembro de 2024	Notação literal de inteiro octal explícita, enumerações, propriedades somente leitura, sintaxe chamável de primeira classe, *new* em inicializadores, tipos de interseção puros, tipo de retorno *never*, restrições de classe final, *fibers*.[33]

Legenda:
Versão antiga
Versão mais antiga, ainda mantida
Versão mais recente

Principais características

A linguagem PHP é uma linguagem de programação de domínio específico, ou seja, seu escopo se estende a um campo de atuação que é o desenvolvimento web, embora tenha variantes como o PHP-GTK. Seu propósito principal é de implementar soluções web velozes, simples e eficientes[34]. Características:

- Velocidade[35] e robustez[36].
- Orientação a objetos.
- Portabilidade - independência de plataforma - escreva uma vez, rode em qualquer lugar.
- Tipagem dinâmica.
- Sintaxe similar a C/C++ e o Perl.
- Open-source.
- Server-side (O cliente manda o pedido e o servidor responde em página HTML)

Importância do PHP Estruturado

O PHP orientado a objetos está muito valorizado atualmente, mas lembre que ele é apenas uma forma de escrever os programas, que pra valer são escritos com muito do PHP estruturado, como as estruturas de controle, por exemplo e as diversas funções nativas do PHP.

1.3 - Recursos e Vantagens

6 Vantagens para Usar PHP

Não é à toa que o PHP se tornou uma das linguagens de programação mais usadas no mundo. As vantagens que ela proporciona atendem às necessidades tanto de usuários iniciantes quanto experientes em programação e desenvolvimento para internet.

Veja, abaixo, 6 motivos para usá-la:

- **Fácil de Aprender**. A linguagem PHP é uma das mais acessíveis para aprender a usar. A sintaxe (as regras que regem a lógica das configurações) tem padrões fáceis de memorizar e entender. E, se você já tem alguma noção de linguagem C ou Java, então, vai se sentir em um ambiente familiar e propício ao desenvolvimento das suas habilidades
- **Alto Desempenho**. O PHP é capaz de suportar grandes quantidades de dados. Com isso, a linguagem consegue executar muitas funções e consumir muitos recursos ao mesmo tempo. E sem comprometer o desempenho e a velocidade do servidor em que está hospedado.
- **É Código Aberto**. Isso significa que a linguagem PHP é gratuita para qualquer usuário. Mais do que isso, desenvolvedores e programadores experientes com acesso ao código-fonte podem fazer atualizações e melhorias periódicas. Isso significa mais recursos, funcionalidades, estabilidade e menos bugs (erros).
- **É Multiplataforma**. É a facilidade que os usuários têm de poder usar e rodar a linguagem PHP numa variedade de sistemas operacionais. Windows, Linux (e suas distribuições) e MacOS são alguns deles. O mesmo acontece entre os navegadores: Chrome, Safari, Edge, Firefox e Opera, entre outros, são todos compatíveis.
- **Compatibilidade com Bancos de Dados**. Um banco de dados é o local onde você guarda todos os dados e informações do seu projeto (site, blog ou loja virtual) na internet. A linguagem PHP é compatível com os principais tipos de bancos de dados, como MySQL, SQLite, Firebird, Interbase e Oracle.
- **Compatibilidade com Hospedagens de Site**. A grande maioria das hospedagens de site do mercado é compatível com a linguagem PHP. É possível criar um site em PHP, fazer conexão com bancos de dados MySQL e alterar a versão do PHP a qualquer momento para aquela que você mais estiver acostumado a usar.

Os benefícios e as vantagens do PHP

- Post author
- By Elias Praciano
- Post date
- 26/02/2014
- 6 Comments on Os benefícios e as vantagens do PHP

Se você é programador iniciante e quer começar alguns projetos para web, este texto é para você – se estiver se perguntando sobre como o PHP pode ser útil e quais são as vantagens em fazer esta escolha, eu tenho uma lista com pelo menos 15 itens favoráveis ao PHP. Me acompanhe.

O PHP é gratuito
Isto significa que o seu projeto de aprendizagem começa sem ter que pôr a mão no bolso – se você se arrepender no final, pelo menos não vai ficar preso a este investimento. Você muda de direção e pronto.

Ninguém vai te cobrar royalties por desenvolver em PHP
Ao final do seu projeto, você tem o direito de distribuir o seu produto livremente. Ninguém virá correndo atrás de você, com uma turma de pitbulls advogados ameaçando te processar se você não lhe pagar taxas sobre o seu trabalho – ele é seu e só você ganha dinheiro com ele, se quiser.

Não há licenciamentos restritivos
Não há contratos ou licenças detalhando o que você pode ou não pode fazer com a linguagem. Faça o que você quiser.

A licença é permissiva a ponto de você poder, até mesmo, criar outra linguagem de programação a partir do PHP e distribuí-la e usar para os seus próprios objetivos comerciais.
O PHP pode ser usado em qualquer país, sob quaisquer leis, em qualquer tipo de aplicação. O seu contrato de uso é extremamente flexível, quando comparado ao JAVA e ao .NET.

Os custos de manutenção de um servidor são muito reduzidos
Além de ser muito fácil de instalar, em qualquer plataforma (inclusive Windows), o PHP foi projetado para rodar sobre o Linux e o Apache – ambos de código aberto e livres.
Os custos de um plano de hospedagem em um servidor Linux, Apache, MySQL e PHP (LAMP) são baixos. É possível encontrar alguns planos gratuitos na Internet, que não oferecem todas as vantagens de um plano pago, mas são ótimos para quem deseja aprender ou apenas testar um projeto.

Código maduro
Criado por Rasmus Leerdorf, em 1995, o PHP já tem 20 anos de estrada. Óbvio que isto não é grande coisa perto de uma linguagem como C, criada em 1969, por Dennis Ritchie. Contudo, a grande difusão de uso do PHP contribui para sua acelerada maturação.
Cabe dizer, aliás, que a linguagem PHP foi construída sobre a solidez da linguagem C, como quase tudo.
O PHP faz parte da mesma geração do JAVA e do ASP, tidas como linguagens comerciais (por conta de suas licenças de uso).

Atualizações consistentes
A linguagem PHP já saiu das mãos de seu criador há um bom tempo, embora ele continue sendo uma pessoa influente e importante, como se pode ver em sua página no Wikipedia. Contudo, o desenvolvimento da linguagem é mantido por várias pessoas, no mundo todo. Elas garantem que ela continue relevante para uso em qualquer site e que esteja sempre atualizada e pronta para responder os desafios que dela se pede.
Este é um dos pontos fortes dos projetos de código aberto, principalmente aqueles que se encaixam no conceito de softwares livres. Quando são relevantes, se tornam independentes da empresa ou da pessoa que os criou — sob certo ponto de vista e, talvez ironicamente, softwares livres estão mais próximos da ideologia capitalista do livre mercado do que os proprietários.

Traduzindo: você tem a certeza de que pode começar um grande projeto, por que você não vai acabar com uma linguagem abandonada e desatualizada nas mãos amanhã.

O PHP se integra a quase todos os bancos de dados usados na atualidade
Embora ele seja quase sempre associado ao MySQL, o PHP roda bem com quase todos os outros – Oracle, MSSQL, IBM DB2 etc.. só pra citar os grandes bancos comerciais. Ou seja, você não precisa migrar sua base de dados para migrar pro PHP. Custo a menos.

Muito fácil de aprender
Os conceitos básicos do PHP são muito fáceis de apreender. Além de uma extensa biblioteca disponível na internet sobre a linguagem, seus fundamentos não assustam os iniciantes e podem ser entendidos em poucas lições. A curva de aprendizagem, pra quem gosta da expressão, te favorece.

Grande quantidade de ambientes de desenvolvimento profissionais disponíveis
Um ambiente integrado de desenvolvimento ou IDE (*Integrated Development Environment*) é um conjunto de softwares que oferece ao programador todas as ferramentas necessárias para desenvolver um projeto.
Mesmo um editor de textos "simples" como o vi, pode resolver para quem deseja editar um código. Eu uso o Komodo, entre outros.

O PHP está rodando na maioria dos servidores web
Algumas estatísticas apontam para um número maior superior a 90% — mas eu gosto de ficar do lado seguro dos números, ainda que menores.
O Netcraft contou, em Janeiro de 2013, 244 milhões de servidores PHP, na Internet – o que significava mais de 39% dos servidores web.
Note que a pesquisa em questão contou apenas os servidores web. Não são somente servidores web que usam PHP. Sites de serviços tais como comércio eletrônico, redes sociais, blogs, gestão de conteúdo etc… todos eles podem usar e usam PHP.

Trata-se de uma tecnologia testada
Com algo em torno de 20 anos de vida, o PHP é uma linguagem escrita para a Internet – e moldada pela Internet.

Forjada na web, tem sido usada em projetos gigantescos, como o Wikipedia, o Yahoo, Flickr, WordPress e na interface ao usuário do Facebook.
Se você está começando um projeto e pretende que ele cresça, começar em PHP é uma boa idéia.

O suporte ao PHP é um bom negócio
Há dezenas de milhares de negócios voltados a dar suporte no PHP – treinamento, desenvolvimento de soluções, manutenção de produtos e servidores, suporte técnico etc.
Vários profissionais podem ajudar você e o seu negócio a crescer ou a encontrar soluções envolvendo a linguagem PHP.

Desenvolver em PHP é um bom negócio
Centenas de milhares de programadores conhecem PHP e muitos dominam e têm conhecimento mais aprofundado da linguagem e podem ajudá-lo em seus projetos.
Você pode contratar *freelancers* ou uma empresa para desenvolver soluções abertas pro seu negócio e, por se tratar de padrões abertos, você tem total liberdade para mudar os profissionais ou empresas envolvidas no projeto – quem chega, começa no ponto em que o outro parou.

Ao escolher o PHP para desenvolver seu projeto, você pode ter a certeza de que vai conseguir suporte profissional sempre que necessitar.

Um grande banco de classes e funções, prontas para uso
… e totalmente livres de ter que pagar *royalties*.
O PHP é uma linguagem aberta o que tem inspirado pessoas a desenvolver e disponibilizar trabalho de qualidade feito para complementar ou se encaixar em qualquer projeto, de qualquer porte.
Você pode baixar da Internet pacotes de funções prontas que ajudam realizar as mais diversas tarefas. Você só precisa implementá-las em seu código, no lugar certo – ou seja, onde você quiser. Isto causa um impacto considerável nos custos e no tempo para finalização de um projeto.

Frameworks prontos para usar
Mantendo o espírito do código aberto e da distribuição livre e irrestrita, há vários *frameworks* em PHP disponíveis que cuidam de funções comuns, como gestão de membros, suporte a administração, buscas, gestão de conteúdo etc. – o que deixa os programadores livres para focar no desenvolvimento do site, em vez de reinventar a roda com funções e objetos já existentes.
Para citar alguns, Cake PHP, Code Igniter e o Symphony estão entre os frameworks mais usados atualmente.

https://elias.praciano.com/2014/02/15-beneficios-e-vantagens-do-php/

1.4 - O que podemos fazer com PHP

A linguagem PHP pode ser empregada em praticamente qualquer utilidade que você queira fazer ou desenvolver na internet. Isso inclui desde a criação de sites até o desenvolvimento de aplicações de serviços e sistemas na web.

Abaixo, listamos algumas possibilidades do que é possível criar e gerenciar com PHP. Todas elas são melhor otimizadas numa hospedagem PHP dedicada.

1. Sites Dinâmicos
Sites dinâmicos são aqueles em que os elementos mostrados neles não ficam estáticos. Suas páginas são gerenciadas por uma aplicação hospedada em um servidor. Com o PHP, é o usuário que determina como uma página será mostrada quando for carregada em navegador. Você pode fazer isso com plataformas de publicação de conteúdos dinâmicos, como WordPress, Drupal, Joomla, Magento e OpenCart.

2. Aplicações para Internet
O PHP permite criar aplicações para qualquer tipo de finalidade na internet. Digamos que você queira incluir um formulário de contatos no seu blog. Ou um fórum de discussões para fazer com que os visitantes do seu site interajam uns com os outros. Ou, ainda, desenvolver e publicar uma galeria de imagens estilizada para sua loja virtual. Tudo isso é possível usando a linguagem de programação.

3. Plugins para WordPress
Plugins são extensões que adicionam novos recursos e funcionalidade ao WordPress, o CMS (Sistema de Gerenciamento de Conteúdo) mais usado no mundo. A plataforma é extremamente popular e sua comunidade de usuários e desenvolvedores é tão engajada que atualizações, melhorias e novos aplicativos são lançados com bastante frequência. Com o PHP, é possível criar plugins para o WordPress e cobrar por isso.

4. Sistemas para Web
Sistemas são conjuntos de dados e informações que se integram e se comunicam uns com os outros. No caso da web, são serviços que você pode desenvolver com PHP para atender a uma determinada finalidade. Pense em um sistema de cursos online com páginas de textos, vídeos, jogos interativos e telas e de logins e senhas. Ou numa rede interna de uma empresa que só os funcionários podem acessar, receber novidades, ver o planejamento das tarefas do dia e o ponto eletrônico.

https://www.weblink.com.br/blog/php/o-que-e-php-conheca/

1.5 - Cliente Servidor

O modelo cliente-servidor, em computação, é uma estrutura de aplicação distribuída que distribui as tarefas e cargas de trabalho entre os fornecedores de um recurso ou serviço, designados como servidores, e os requerentes dos serviços, designados como clientes. Wikipédia

2 - Ambiente de desenvolvimento

2.1 – Hospedagem

2.1.1 – No Desktop

Nosso desktop geralmente é o primeiro servidor que hospeda nosso código PHP. Se usando Windows instalamos o Xampp, por exemplo, que traz todas as ferramentas necessárias para usar os recursos de servidor em nosso desktop.

2.1.2 – Hospedagem free para PHP

Após dar os primeiros passos e quiser experimentar seu código em um servidor, geralmente optamos por fazer isso em um servidor gratuito. Geralmente dá trabalho de encontrar um servidor de hospedagem gratuito com suporte ao PHP. Idealmente devemos usar um pago, mas para testes e no início, vejamos alguns.

Freehostia
Meu preferido - https://www.freehostia.com/

Bom hosting free com suporte ao PHP, MySQL e cia

Pelo gerenciador de arquivos do site podemos enviar arquivos, inclusive compactados, com até 2MB e ele também descompacta

Para arquivos e pastas maiores criar uma conta de FTP. Minha sugestão é usar o Filezilla com SFTP

- Requer que você tenha um domínio para criar e usar a conta (Dica: É bom ter um domínio para testes e estas hospedagens)
- PHP - várias versões, inclusive 7.4 e 8. Configurações no php.ini
- MySQL - 1 banco com até 10MB, com phpMyAdmin e até acesso remoto
- Abrigar até 5 domínios com gerenciamento
- 250MB Disk Space
- 6GB Monthly Traffic
- 3 E-mail Accounts e com webmail
- Instalador com 1 clique: Joomla, WordPress, Laravel e outros
- Diretório web - /home/www

Freehostingeu
Este foi indicação do colega Jorge Miguel do grupo Laravel Brasil do Facebook

https://www.freehostingeu.com/

A vantagem deste é que nem domínio ele exige. Você recebe um subdomínio. Veja um que tenho lá: http://ribafs.eu3.org/

Recursos

- PHP com MySQL
- Instalação com 1 clique do Joomla, Wordpress, e outros
- Outros recursos. Confira

Cloudaccess
Se for free para hospedar um site com Joomla este é o meu preferido

Este é dedicado e especializado nos CMS Joomla e Wordpress

Você cria a conta e já recebe uma conta e site com o CMS instalado e pronto para usar

Free por 30 dias. Caso queira pode renovar. Eles avisam por e-mail

https://www.cloudaccess.net/joomla/managed-hosting.html

https://www.cloudaccess.net/wordpress/managed-hosting.html

Máquina Virtual com bons recursos, espaço de 500MB e uma IDE web para administração do projeto

Heroku
https://www.heroku.com/

Este exige conhecimento de git

Heroku runs your app in lightweight, isolated Linux containers called "dynos." The platform offers different dyno types to help you get the best results for your type of app.

Plano free:

- Aplicações não comerciais
- Projetos pessoais
- 512MB
- Custom domain - podemos hospedar lá nosso domínio pessoal

O deploy é feito com git e docker

Uma boa opção de SGBD free é o PostgreSQL ou o SQLite, que pra valer não é um SGBD mas quebra alguns galhos

Byethost
Este usei muito pouco, mas quando usei funcionou

https://byet.host/free-hosting/

Main FeaturesPHP & MySQL included with every free hosting plan

```
1000 MB (one gigabyte!) Disk Space
FTP account and File Manager
Control Panel
PHP
MySQL databases & PHP Support
Free tech support
Addon domain, Parked Domains, Sub-Domains
Free Community Access (Forums)
Clustered Servers
No ads!
Complete Features
Account Specification
```

Sites estáticos
Estes são sites que usam apenas HTML, CSS e Javascript. Nada de PHP nem MySQL

Github Pages - https://github.com
Este é o meu preferido para sites estáticos. Meu site está com ele - https://ribamar.net.br

Criar um repositório e usá-lo para abrigar um site em HTML

Existem várias outras alternativas, como

2.1.3 – Hospedagem Compartilhada

As free também são compartilhadas, mas estas são as comerciais. Recomendadas para quando estiver iniciando mas já quiser colocar um site no ar.

Existem diversas hospedagens compartilhadas a disposição e recomendo que faça uma pesquisa em grupos ou junto a colegas que já estejam usando.

2.1.4 – Hospedagem tipo VPS

Este tipo de hospedagem exige conhecimentos de Linux e de servidores. Caso queira estudar sobre eles veja estes e-books:

https://ribamar.net.br/phpecia/sobre/meus-livros/2-generica/20-servidores-vps.html

https://ribamar.net.br/phpecia/sobre/meus-livros/2-generica/21-servidores-web-tipo-vps.html

Com uma hospedagem tipo VPS temos praticamente controle total sobre o servidor dos nossos sites e aplicativos. A administração do servidor, a instalação do Apache, PHP, etc, tudo é feito por nós. Controlamos tudo, todas as configurações e o que instalamos. Caso esteja disposto a aprender vale a pena.

2.2 – Domínio

Para publicar um site na internet precisamos de dois serviços, uma hospedagem, onde guardamos os arquivos/código e banco de dados e também precisamos de um domínio.

Quando contratamos uma hospedagem, geralmente ela nos fornece um IP, com o qual podemos acessar nosso site. Mas não é conveniente passar um IP para divulgar nosso site. Então contratsmos um serviço de domínio, geralmente pagamento anual, e apontamos nosso domínio para a hospedagem.

2.3 - Servidor web embutido

A partir da versão 5.4 o PHP trouxe um servidor web embutido, simplificando assim a criação de um ambiente de programação local.

Para iniciar o servidor, execute o seguinte comando no seu terminal dentro da raiz de seu projeto:

```
cd aplicativo
php -S localhost:8000
```

Então abrir o navegador com
http://localhost:8000

Para fechar o servidor tecle Ctrl+C

Executar o código na pasta public
PHP -S localhost:8070 -t public

Executar um único arquivo
php -S localhost:8000 router.php

Acessível em qualquer interface
php -S 0.0.0.0:8000

Acessar usando uma configuração específica
php -S localhost:8000 -c php.ini

No caso precisará instalar apenas o MySQL para ter o ambiente.

2.4 - Xampp no Windows

O Xampp é um pacote que instala o ambiente de desenvolvimento PHP para Windows, Mac e Linux. É um dos pacotes mais populares e muito simples de instalar.

Site oficial
https://www.apachefriends.org/index.html

Para instalar basta usar o famoso next-next com alterações quando souber.

2.5 - Pacotes no Linux

Para o Linux Mint 20.1 ou Ubuntu 20.04 encontrará um script que instala todos os pacotes necessários ao ambiente: Apache, PHP, MySQL, Git, Composer, etc

https://github.com/ribafs/linux/blob/main/Scripts/1mint_20.sh

Também encontrará outros bons s cripts aqui:
https://github.com/ribafs/linux/tree/main/Scripts

3 – Sintaxe Básica

3.1 - Arquivo, tags e pasta

Para que um arquivo seja reconhecido como contendo código PHP requer:
- Sua extensão deve ser .php
- No arquivo em PHP o trecho de código em PHP deve ser delimitado pelas tags:
    <?php (abertura) e ?> (fechamento)
    Caso o arquivo contenha exclusivamente código PHP não devemos usar a tag de fechamento
- Uma terceira condição é que o arquivo somente funcionará se estiver no diretório web do servidor web. Exemplos: /var/www/html ou c:\xampp\htdocs. Exceto quando estivermos usando o servidor web embutido do PHP. Neste caso o arquivo pode funcionar em qualquer parte.

Exemplo usando Xampp

c:\xampp\htdocs\teste.php

Este arquivo deve ser chamado no navegador assim:

http://localhost/teste.php

Exemplo usando o servidor web embutido do PHP

c:\aplicativo\teste.php
cd c:\aplicativo
php -S localhost:8000 teste.php

Chamar pelo navegador com

http://localhost:8000

Dica: caso a porta 8000 esteja ocupada mude para outra.

3.2 - Olá mundo

O famoso hello world em PHP é muito simples:

c:\xampp\htdocs\ola.php

```
<?php
echo "Olá mundo";
?>
```

http://localhost/ola.aphp

Ao invés de echo também podemos usar print.

3.3 - Instruções e final de linha

Cada final de linha ou de instrução no PHP deve conter o ponto e vírgula (;), como no ola.php.

Podemos escrever duas instruções por linha, assim:

echo "Olá"; echo " mundo";
Mas é importante evitar fazer isso, pois torna o código mais trabalhoso de entender e corremos o risco de deixar passar algo importante sem ver.

3.4 – Comentários

Em PHP temos 4 tipos de comentários:

```
// Comentário para uma única linha

/*
Comentário para uma ou mais
linhas
*/

/**
* @author Ribamar FS <ribafs@gmail.com>
* @link http://www.phpdoc.org/docs/latest/index.html
* @package helper
* @Este é comentário do phpdoc
*/

# Comentário para uma única linha. Parece que é melhor evitar este comentário
```

3.5 – Variáveis

As variáveis tem um nome e recebem um valor. São armazenadas na memória RAM e estão disponíveis no programa durante a sua execução.

Como o nome sugere seu valor pode ser alterado durante o processamento.

Detalhes importantes sobre variáveis no PHP:

- Seu nome sempre começa com $ seguido do nome (Ex: $nome)
- Não declaramos o tipo de dados da variável. Somente em tempo de execução serão identificados
- O PHP é case sensitivo, portanto a variável $nome é diferente da $Nome
- O primeiro caractere do nome pode ser uma letra ou o caractere sublinhado '_' e o restante do nome pode conter os caracteres alfanuméricos e o caractere '_' (A-z,0-9,_).

Exemplos:

```
$nome = "João";
$x = 20;
$estado = true;
$preco = 25.30;
```

3.6 – Constantes

As constantes tem várias diferenças das variáveis.

Como o nome sugere, seu valor não muda durante a execução do programa. Caso tentemos definir um novo valor durante a execução receberemos um erro.

Seu nome não começa com $ e, por convenção, ele deve vir com todas as letras em maiúsculas.

As constantes são definidas no PHP estruturado usando a função define().
Exemplos

Como os dados de uma conexão com o banco de dados não se alteram durante a execução, é uma boa ideia usar constantes ao invés de variáveis. Isso torna nosso código mais seguro.

```
define('HOST', 'localhost');
define('PORT', 3306);
define('USER', 'root');
```

3.7 - Tipos de dados

O PHP suporta os seguintes tipos de dados:

- String
- Integer
- Float
- Boolean
- Array
- Object
- NULL
- Resource

String - uma sequência de caracteres, delimitada por " ou '

Integer - número não decimal entre -2,147,483,648 e 2,147,483,647

Float - números em ponto flutuante – também chamados double. Número escrito com o ponto como separador ou na forma exponencial

Boolean – pode ser de dois tipos, true ou false

Array – este pode armazenar valores de vários tipos em uma variável

Object – classes e objetos são os dois principais aspectos da orientação a objetos. Também é um tipo composto

Null – este é um tipo de dados especial que pode conter apenas o valor NULL. Uma variável com o valor NULL não tem nenhum valor associado a ela.

Dica: se uma variável for criada sem definir seu valor, ele recebe automaticamente NULL.

Resource - O tipo de dados especial resource não é um tipo de dados real. É o armazenamento de uma referência a funções e resources externos ao PHP.

Um exemplo comum de utilização do tipo de dados do recurso é uma chamada a uma base de dados.

https://www.w3schools.com/PHP/php_datatypes.asp

A função geettype() retorna o tipo de dados de uma variável.

Exemplo:

```
$data = array(1, 1., NULL, new stdClass, 'foo');

foreach ($data as $value) {
    echo gettype($value), "<br>";
}
```

3.8 – Expressões

Expressões são os blocos de construção mais importantes do PHP. No PHP, quase tudo o que você escreve são expressões. A maneira mais simples e ainda mais precisa de definir uma expressão é "tudo o que tem um valor".

As formas mais básicas de expressões são constantes e variáveis. Quando você digita "$a = 5", você está atribuindo '5' dentro de $a. '5' obviamente tem o valor 5, ou em outras palavras '5' é uma expressão com o valor de 5 (nesse caso '5' é uma constante inteira).

Depois desta atribuição, você pode esperar que o valor de $a seja 5 também, assim se você escrever $b = $a, você pode esperar que ele se comporte da mesma forma que se você escrevesse $b = 5. Em outras palavras, $a é uma expressão com valor 5 também. Se tudo funcionou bem isto é exatamente o que aconteceu.

Exemplos ligeiramente mais complexos para expressões são as funções.

```
<?php
function double($i)
{
    return $i*2;
}
$b = $a = 5;        /* atribui o valor cinco às variáveis $a e $b */
$c = $a++;          /* pós-incremento, atribui o valor original de $a
                    (5) para $c */
$e = $d = ++$b;     /* pré-incremento, atribui o valor incrementado de
                    $b (6) a $d e $e */

/* neste ponto, tanto $d quanto $e são iguais a 6 */

$f = double($d++);  /* atribui o dobro do valor de $d antes
                    do incremento, 2*6 = 12 a $f */
$g = double(++$e);  /* atribui o dobro do valor de $e depois
                    do incremento, 2*7 = 14 a $g */
$h = $g += 10;      /* primeiro, $g é incrementado de 10 e termina com o
                    valor 24. o valor da atribuição (24) é
                    então atribuído a $h, e $h termina com o valor
                    24 também. */
?>
```

Detalhes:
https://www.php.net/manual/pt_BR/language.expressions.php

3.9 – Operadores

Um operador é algo que recebe um ou mais valores (ou expressões, no jargão de programação) e que devolve outro valor (e por isso os próprios construtores se tornam expressões).

Operadores podem ser agrupados segundo o número de valores que aceitam. Operadores unários recebem um único valor, por exemplo ! (o operador lógico de negação) ou ++ (o operador de incremento). Operadores binários aceitam dois valores, como os operadores aritméticos + (soma) e - (subtração), além da maioria dos operadores PHP dessa categoria. Finalmente há um único operador ternário, ? :, que aceita três valores; normalmente conhecido simplesmente como "o operador ternário" (embora um nome melhor fosse operador condicional).

A lista completa dos operadores no PHP está na seção sobre Precedência de Operadores. Essa seção também explica precedência e combinações, que governam exatamente como expressões contendo vários operadores são avaliados.

https://www.php.net/manual/pt_BR/language.operators.php

Precedência de Operadores

A precedência de um operador especifica quem tem mais prioridade quando há duas delas juntas. Por exemplo, na expressão 1 + 5 * 3, a resposta é 16 e não 18 porque o operador de multiplicação ("*") tem prioridade de precedência que o operador de adição ("+"). Parênteses podem ser utilizados para forçar a precedência, se necessário. Assim, (1 + 5) * 3 é avaliado como 18.

Quando operadores tem precedência igual a associatividade decide como os operadores são agrupados. Por exemplo "-" é associado à esquerda, de forma que 1 - 2 - 3 é agrupado como (1 - 2) - 3 e resulta em -4. "=" por outro lado associa para a direita, de forma que $a = $b = $c é agrupado como $a = ($b = $c).

Operadores de igual precedência sem associatividade não podem ser utilizados uns próximos aos outros. Por exemplo 1 < 2 > 1 é ilegal no PHP. A expressão 1 <= 1 == 1 por outro lado é válida, porque o operador == tem menor precedência que o operador <=.

O uso de parenteses, embora não estritamente necessário, pode melhorar a leitura do código ao deixar o agrupamento explícito em vez de depender da associatividade e precedências implícitos.

A tabela seguinte mostra a precedência dos operadores, com a maior precedência no começo. Operadores com a mesma precedência estão na mesma linha, nesses casos a associatividade deles decidirá qual ordem eles serão avaliados.

Detalhes em
https://www.php.net/manual/pt_BR/language.operators.precedence.php

Tipos de operadores

Operadores Aritméticos
Operadores de Atribuição
Operadores bit a bit (bitwise)
Operadores de Comparação
Operadores de controle de erro
Operadores de Execução

Operadores de Incremento/Decremento
Operadores Lógicos
Operadores de String
Operadores de Arrays
Operadores de tipo

Detalhes em
https://www.php.net/manual/pt_BR/language.operators.php

Alguns exemplos

Operadores Aritméticos

Exemplo	Nome	Resultado
+$a	*Identidade*	*Conversão de $a para int ou float conforme apropriado.*
-$a	*Negação*	*Oposto de $a.*
$a + $b	*Adição*	*Soma de $a e $b.*
$a - $b	*Subtração*	*Diferença entre $a e $b.*
*$a * $b*	*Multiplicação*	*Produto de $a e $b.*
$a / $b	*Divisão*	*Quociente de $a e $b.*
$a % $b	*Módulo*	*Resto de $a dividido por $b.*
*$a ** $b*	*Exponencial*	*Resultado de $a elevado a $b. Introduzido no PHP 5.6.*

Operadores de atribuição

O operador de atribuição é o sinal =.

Exemplos:
$x = 10; // Assim estamos atribuindo o valor 10 para $x

$nome = 'Ribamar FS'; // Atribuímos Ribamar FS para $nome

Operadores Lógicos

Exemplo	Nome	Resultado
$a and $b	*E*	*Verdadeiro (**`true`**) se tanto $a quanto $b são verdadeiros.*
$a or $b	*OU*	*Verdadeiro se $a ou $b são verdadeiros.*
$a xor $b	*XOR*	*Verdadeiro se $a ou $b são verdadeiros, mas não ambos.*
! $a	*NÃO*	*Verdadeiro se $a não é verdadeiro.*
$a && $b	*E*	*Verdadeiro se tanto $a quanto $b são verdadeiros.*
$a || $b	*OU*	*Verdadeiro se $a ou $b são verdadeiros.*

Detalhes em
https://www.php.net/manual/pt_BR/language.operators.logical.php

Operadores de String

Há dois operadores de string. O primeiro é o operador de concatenação ('.'), que retorna a concatenação dos seus argumentos direito e esquerdo. O segundo é o operador de atribuição de concatenação ('.='), que acrescenta o argumento do lado direito no argumento do lado esquerdo. Veja em Operadores de Atribuição para mais informações.

```php
<?php
$a = "Olá ";
$b = $a . "mundo!"; // agora $b contém "Olá mundo!"

$a = "Olá ";
$a .= "mundo!";     // agora $a contém "Olá mundo!"
?>
```

4 - Estruturas de controle

Me parece que estas estruturas são a força da programação. A maioria delas é comum às principais linguagens de programação.

As estruturas de controle como o nome sugere, controlam o que faz cada trecho de código, se realiza ou não certo cálculo, se encerra ou não um laço.

Detalhes:
https://www.php.net/manual/pt_BR/language.control-structures.php

Qualquer script PHP é construído por uma série de instruções. Uma instrução pode ser uma atribuição, uma chamada de função, um laço de repetição, uma instrução condicional, ou mesmo uma instrução que não faz nada (um comando vazio). Instruções geralmente terminam com um ponto e vírgula. Além disso, as instruções podem ser agrupados em um grupo de comandos através do encapsulamento de um grupo de comandos com chaves. Um grupo de comandos é uma instrução também. Os vários tipos de instruções são descritos neste capítulo.

4.1 - if, else e elseif

O construtor `if` é um dos recursos mais importantes em muitas linguagens, inclusive no PHP. Permite a execução condicional de fragmentos de código. O PHP apresenta uma estrutura `if` semelhante a do C:

```
if (expr)
  statement
```

if – se
eles – se não

Se uma expressão for avaliada como **`true`**, o PHP executará a declaração, e se avaliá-la como **`false`** - ignorá-la. Mais informações sobre quais valores são avaliados como **`false`** pode ser encontrada na seção 'Conversão para booleano' .

```
if(expressao) {
        instrucaoA;
}else{
        instrucaoB;
}
```

Se expressao for true instrucaoA será executada.
Se for false instrucaoB será executada.

`elseif`, como o nome sugere, é uma combinação do `if` e `else`. Como o `else`, estende um `if` para executar instruções diferentes no caso da expressão `if` original ser avaliada como **`false`**. Entretanto, diferentemente do `else`, executará uma expressão alternativa somente se a expressão condicional do `elseif` for avaliada como **`true`**. Por exemplo, o código a seguir exibirá a is bigger than b, a equal to b ou a is smaller than b:

```
if ($a > $b) {
   echo "a é maior que b";
} elseif ($a == $b) {
```

```
    echo "a é igual a b";
} else {
    echo "a é menor que b"; // Caso a não seja maior que b nem igual, então será menor
}
```

Sintaxe alternativa

O PHP oferece uma sintaxe alternativa para algumas estruturas de controle; a saber, `if`, `while`, `for`, `foreach`, e `switch`. Em cada caso, basicamente a sintaxe alternativa é trocar a chave de abertura por dois pontos (:) e a chave de fechamento por `endif;`, `endwhile;`, `endfor;`, `endforeach;`, ou `endswitch;`, respectivamente.

Exemplos

<?php if ($a == 5): ?>
A é igual a 5
<?php endif; ?>

4.2 - Laços de repetição/loops

4.2.1 – for

Os laços `for` são os mais complexo no PHP. Possui comportamento semelhante ao C. A sintaxe do laço `for` é:

```
for (expr1; expr2; expr3)
    statement
```

A primeira expressão (*expr1*) é avaliada (executada), uma vez, incondicionalmente, no início do laço.

No começo de cada iteração a *expr2* é avaliada. Se a avaliada como **`true`**, o laço continuará e as instruções aninhada serão executadas. Se avaliada como **`false`**, a execução do laço terminará.

No final de cada iteração, a *expr3* é avaliada (executada).

Exemplos

Analise os seguintes exemplos. Todos exibem números de 1 até 10:

```
<?php
/* exemplo 1 */

for ($i = 1; $i <= 10; $i++) {
    echo $i;
}

/* exemplo 2 2 */

for ($i = 1; ; $i++) {
    if ($i > 10) {
        break;
    }
    echo $i;
}

/* exemplo 3 */
```

```
$i = 1;
for (; ; ) {
    if ($i > 10) {
        break;
    }
    echo $i;
    $i++;
}

/* exemplo 4 */

for ($i = 1, $j = 0; $i <= 10; $j += $i, print $i, $i++);
?>
```

É claro que o primeiro exemplo aparenta ser o mais simpático (ou talvez o quarto), mas pode-se achar que o uso de expressões vazias no laço `for`, seja vantajoso em algumas ocasiões.

Detalhes:
https://www.php.net/manual/pt_BR/control-structures.for.php

4.2.2 – while

Laços `while` são os mais simples tipos de laços do PHP. Possui comportamento semelhante ao C. O formato básico de uma declaração `while` é:

```
while (expr)
    statement
```

O propósito da declaração `while` é simples. Ele dirá ao PHP para executar as declarações aninhadas repetidamente, enquanto a expressão do `while` for avaliadas como **`true`**. O valor da expressão é checado a cada vez que o laço é iniciado, então, mesmo seu valor mude durante a execução das declarações aninhadas, a execução não será interrompida até o final da iteração (cada vez que o PHP executa as declarações dentro do laço é uma iteração). Entretanto, se a expressão do `while` for avaliada como **`false`** desde o início, as declarações aninhadas não serão executadas nenhuma vez.

```
<?php
/* example 1 */

$i = 1;
while ($i <= 10) {
    echo $i++;  /* the printed value would be
                   $i before the increment
                   (post-increment) */
}

/* example 2 */

$i = 1;
while ($i <= 10):
    echo $i;
    $i++;
endwhile;
?>
```

4.2.3 - do ... while

O laço `do-while` é muito similar ao laço `while`, com exceção que a expressão de avaliação é verificada ao final de cada iteração em vez de no começo. A maior diferença para o laço `while` é que a primeira iteração do laço `do-while` sempre é executada (a expressão de avaliação é executada somente no final da iteração), considerando que no laço `while` não é necessariamente executada (a expressão de avaliação é executada no começo de cada iteração, se avaliada como **`false`** logo no começo, a execução do laço será abortada imediatamente).

Só há uma sintaxe para o laço `do-while`:

$i = 0;

do {

*   echo $i;*

} while ($i > 0);

https://www.php.net/manual/pt_BR/control-structures.do.while.php

4.2.4 – foreach

O construtor `foreach` fornece uma maneira fácil de iterar sobre arrays. **O `foreach` funciona somente em arrays e objetos**, e emitirá um erro ao tentar usá-lo em uma variável com um tipo de dado diferente ou em uma variável não inicializada. Possui duas sintaxes:

```
foreach (array_expression as $value)
    statement
foreach (array_expression as $key => $value)
    statement
```

A primeira forma, itera sobre arrays informados na `array_expression`. A cada iteração, o valor do elemento atual é atribuído a `$value` e o ponteiro interno do array avança uma posição (então, na próxima iteração, se estará olhando para o próximo elemento).

A segunda forma var, adicionalmente, atribuir a chave do elemento corrente a variável `$key` a cada iteração.

https://www.php.net/manual/pt_BR/control-structures.foreach.php

4.3 – Interrupção

4.3.1 – break

`break` finaliza a execução da estrutura `for`, `foreach`, `while`, `do-while` ou `switch` atual.

`break` aceita um argumento numérico opcional que diz quantas estruturas aninhadas deverá interromper. O valor padrão é `1`, somente a estrutura imediata é interrompida.

$arr = array('one', 'two', 'three', 'four', 'stop', 'five');

foreach ($arr as $val) {

```
    if ($val == 'stop') {
        break;    /* You could also write 'break 1;' here. */
    }
    echo "$val<br />\n";
}
/* Using the optional argument. */
$i = 0;
while (++$i) {
    switch ($i) {
        case 5:
            echo "At 5<br />\n";
            break 1;  /* Exit only the switch. */
        case 10:
            echo "At 10; quitting<br />\n";
            break 2;  /* Exit the switch and the while. */
        default:
            break;
    }
}
```

https://www.php.net/manual/pt_BR/control-structures.break.php

4.3.2 – continue

`continue` é utilizado em estruturas de laço para pular o resto da iteração atual, e continuar a execução na validação da condição e, então, iniciar a próxima iteração.

O `continue` aceita um argumento numérico opcional que diz quantos níveis de laços aninhados deve pular. O valor padrão é `1`, saltando para o final do laço atual.

```
foreach ($arr as $key => $value) {
    if (!($key % 2)) { // pula membros pares
        continue;
    }
    do_something_odd($value);
}


$i = 0;
while ($i++ < 5) {
```

```
    echo "Outer<br />\n";
    while (1) {
        echo "Middle<br />\n";
        while (1) {
            echo "Inner<br />\n";
            continue 3;
        }
        echo "This never gets output.<br />\n";
    }
    echo "Neither does this.<br />\n";
}
```

https://www.php.net/manual/pt_BR/control-structures.continue.php

4.4 – switch

A declaração `switch` é similar a uma série de declarações IF na mesma expressão. Em muitos casos, se deseja comparar as mesmas variáveis (ou expressões), com diferentes valores, e executar pedaços diferentes de código dependendo de qual valor ela é igual. Esta é exatamente a serventia da declaração `switch`.

> **Nota**: Note que ao contrário de outras linguagens, a declaração continue aplica-se ao `switch` e age similarmente ao `break`. Se possuir um `switch` dentro de um laço, e deseja continuar na próxima iteração do laço externo, utilize `continue 2`.

Exemplos

```
if ($i == 0) {
    echo "i equals 0";
} elseif ($i == 1) {
    echo "i equals 1";
} elseif ($i == 2) {
    echo "i equals 2";
}


switch ($i) {
    case 0:
        echo "i equals 0";
        break;
    case 1:
        echo "i equals 1";
        break;
```

case 2:

echo "i equals 2";

break;

}

https://www.php.net/manual/pt_BR/control-structures.switch.php

4.5 – return

A declaração `return` retorna o controle do programa para o módulo que o chamou. A execução continuará na expressão seguinte à invocação do módulo.

Se chamada dentro de uma função, a declaração `return` terminará imediatamente sua execução, e retornará seus argumentos como valor à chamada da função. A declaração `return` também terminar a execução de uma declaração eval() ou um arquivo de script.

Se chamada no escopo global, a execução do script corrente é finalizada. Se o arquivo de script corrente for incluído ou requerido com as funções include ou or require, o controle é passado de volta ao script que está chamando. Além disso, se o script corrente foi incluído com a função include, o valor informado ao `return` será retornado como o valor da chamada de include. Se um `return` for chamado de dentro do script principal, sua execução será finalizada. Se o script corrente for mencionado nas opções de configuração auto_prepend_file ou auto_append_file *php.ini*, a execução do script será finalizada.

Exemplos

a.php

<?php

include("b.php");

echo "a";

?>

b.php

<?php

echo "b";

return;

?>

a.php

<?php

include("b.php");

echo "a";

?>

b.php

<?php

echo "b";

exit;

?>

https://www.php.net/manual/pt_BR/function.return.php

5 - Requisições de arquivos

5.1 - include

A declaração `include` inclui e avalia um arquivo externo no arquivo atual.

A documentação a seguir também se aplica a declaração require.

Os arquivos são incluídos baseando-se no caminho do arquivo informado ou, se não informado, o include_path especificado. Se o arquivo não for encontrado no include_path, a declaração `include` checará no diretório do script que o executa e no diretório de trabalho corrente, antes de falhar. O construtor `include` emitirá um aviso se não localizar o arquivo; possui um comportamento diferente do construtor require, que emitirá um erro fatal.

Se um caminho for definido — seja absoluto (iniciando com a letra do drive ou \ no Windows, ou / no Unix/Linux), ou relativo ao diretório atual (começando com `.` ou `..`) — o include_path será completamente ignorado. Por exemplo, se o nome do arquivo iniciar com `../`, o interpretador irá procurar pelo arquivo no diretório pai.

Para mais informações de como o PHP trabalha ao incluir arquivos e o caminho de inclusão, veja a documentação do include_path.

Quando um arquivo é incluído, o código herda o escopo de variáveis da da linha que a inclusão ocorrer. Qualquer variável disponível no arquivo que incluiu estará disponível no arquivo incluído, daquela linha em diante. Entretanto, todas as funções e classes definidas no arquivo incluído estarão no escopo global.

Exemplo básico com include

vars.php

```
<?php

$color = 'green';

$fruit = 'apple';

?>
```

test.php

```
<?php

echo "A $color $fruit"; // A

include 'vars.php';

echo "A $color $fruit"; // A green apple

?>
```

Avisos de Segurança
O arquivo remoto pode ser processado pelo servidor remoto (dependendo da extensão do arquivo e se o servidor remoto executa, ou não,arquivos PHP), mas ainda precisa produzir um código PHP válido pois será processado pelo servidor local. Se o arquivo do servidor remoto deve ser processado como um texto simples, a função readfile() é uma opção muito melhor a ser usada. Caso

contrário, deve-se ter um cuidado especial para garantir que o servidor remoto produza um código PHP válido.

https://www.php.net/manual/pt_BR/function.include.php

5.2 - require

A declaração `require` é idêntica a include exceto que em caso de falha também produzirá um erro fatal de nível `E_COMPILE_ERROR`. Em outras palavras, ele parará o script enquanto que o include apenas emitirá um alerta (`E_WARNING`) permitindo que o script continue.

Exemplo

```
foo.php:
<?php
return "foo";
?>

$bar = require("foo.php");
echo $bar; // equals to "foo"
```

https://www.php.net/manual/pt_BR/function.require.php

5.3 – require_once

A declaração require_once é idêntica a require exceto que o PHP verificará se o arquivo já foi incluído, e em caso afirmativo, não o incluirá (exigirá) novamente.

https://www.php.net/manual/pt_BR/function.require-once.php

5.4 – include_once

A declaração include_once inclui e avalia o arquivo informado durante a execução do script. Este é um comportamento similar a declaração include, com a única diferença que, se o código do arquivo já foi incluído, não o fará novamente, e o include_once retornará **true**. Como o nome sugere, o arquivo será incluído somente uma vez.

O `include_once` pode ser utilizado em casos em que o mesmo arquivo pode ser incluído e valiado mais de uma vez durante uma execução de um script em particular, neste caso, ajudará a evitar problemas como redefinição de funções, reatribuição de valores de variáveis, e etc.

https://www.php.net/manual/pt_BR/function.include-once.php

6 - Funções nativas

Lista de Funções do PHP
https://www.w3schools.com/php/php_ref_overview.asp
https://www.php.net/manual/pt_BR/indexes.functions.php
https://www.exakat.io/en/top-100-php-functions/

6.1 – Trabalhando com o Sistema de Arquivos

Diversas operações com arquivos:

- copy()

- delete()

- rename()

E muitas outras:

https://www.php.net/manual/pt_BR/ref.filesystem.php

Trabalhando com arquivos em PHP

O PHP tem muitos e bons recursos para trabalharmos com arquivos, pastas e permissões. E conhecer isso é importante na criação de programas.

Precisamos lembrar que o PHP trabalha com o sistema de arquivos sob os cuidados do Apache, Nginx ou outro servidor web suportado. E o PHP com suas funções somente pode fazer o que o servidor web puder fazer. Lembrar que o servidor web somente pode manipular arquivos e pastas na pasta web. No Xampp é no c:\xampp\htdocs, nos linux em geral é na pasta /var/www/html.

criado por Caio Filipini em 14/09/2002 1:57pm

Ao trabalhar com arquivos, no mínimo duas operações devem ser realizadas: abrir e fechar o arquivo.

Para abrir um arquivo, precisamos utilizar a função fopen(), que tem a seguinte sintaxe:

fopen(filename, mode, [use_include_path]);

filename: pode ser simplesmente um nome, ou um caminho completo. Exemplos: “arquivo.txt”, “./arquivo.dat”, “/data/data.txt”.

mode: especifica o modo de abertura, ou seja, se o arquivo deve ser aberto para leitura, escrita, etc.

Modos de abertura:

- r: abre o arquivo no modo somente leitura e posiciona o ponteiro no início do arquivo; o arquivo já deve existir;
- r+: abre o arquivo para leitura/escrita, posiciona o ponteiro no início do arquivo;

- w: abre o arquivo no modo somente escrita; se o arquivo já existir, será sobrescrito; senão, será criado um novo;
- w+: abre o arquivo para escrita/leitura; se o arquivo já existir, será sobrescrito; senão, será criado um novo;
- a: abre o arquivo para anexar dados, posiciona o ponteiro no final do arquivo; se o arquivo não existir, será criado um novo;
- a+: abre o arquivo para anexo/leitura, posiciona o ponteiro no final do arquivo; se o arquivo não existir, será criado um novo;

Obs: Além dos modos de abertura descritos acima, um arquivo pode ser aberto como binário, especificando o modo de abertura como "b".

use_include_path: este parâmetro é opcional; se for atribuído o valor 1, e não for especificado nenhum caminho (path) no nome do arquivo, ele será procurado no diretório especificado em include_path, no arquivo php.ini.

A função fopen() retorna um número inteiro, o indicador (handle) do arquivo. Este indicador será necessário quando formos realizar operações de leitura e escrita no arquivo, indicando qual arquivo iremos manipular. Portanto, para abrir um arquivo usamos a seguinte instrução:

Abrir arquivo

```
<?php
$fp = fopen("./arquivo.dat", "r"); // $fp conterá o handle do arquivo que abrimos
?>
```

Aqui estamos abrindo um arquivo já existente no modo somente leitura, e atribuindo o valor retornado por fopen() à variável $fp. $fp conterá um número inteiro positivo (o handle do arquivo) se a operação tiver sucesso; caso contrário, conterá zero.

Depois de utilizar o arquivo, é necessário que ele seja fechado. Para tanto utilizamos a função fclose():

Fechar

```
fclose(handle_arquivo);
```

No caso do arquivo que abrirmos no exemplo anterior, teríamos a seguinte instrução:

```
<?php
fclose($fp);
?>
```

Agora que já sabemos como abrir e fechar um arquivo, precisamos conhecer as funções que manipulam esse arquivo. Primeiro, daremos uma olhada nas funções para escrita de dados de um arquivo.

fwrite(): Esta função permite escrever strings em um arquivo. Se os dados forem escritos com sucesso, fwrite() retorna o número de bytes escritos; caso contrário, retorna –1 (indicando erro). Ela tem a seguinte sintaxe:

Gravar em arquivo

fwrite(handle, string);
- handle: handle do arquivo onde os dados serão escritos;
- string: string a ser escrita no arquivo;

Exemplo:

```
<?php
$fp = fopen("./dados.txt", "w");
fwrite($fp, "Hello world!"); // grava a string "Hello world!" no arquivo
fclose($fp);
?>
```

fputs(): Esta função é semelhante à função fwrite() e funciona da mesma maneira.

Depois de escrevermos os dados no arquivo, precisamos ser capazes de ler os dados que acabamos de gravar. É exatamente isso o que iremos aprender agora, funções para leitura de dados de um arquivo.

fread(): Esta função permite ler strings gravadas em um arquivo. A sintaxe é a seguinte:

fread(handle, length);
- handle: handle do arquivo de onde os dados serão lidos;
- length: tamanho em bytes do buffer de leitura;

Exemplo:

```
<?php
$fp = fopen("./dados.txt", "r");
$text = fread($fp, 20); // lê 20 bytes do arquivo e armazena em $text
fclose($fp);
?>
```

fgets(): Esta função é usada na leitura de strings de um arquivo. fgets() lê "length – 1" bytes do arquivo. Se for encontrado o final da linha e o número de bytes especificados não tiver sido atingido, fgets() terminará a leitura no final da linha (ou no final do arquivo, se for o caso). Eis a sua sintaxe:

fgets(handle, length);
- handle: handle do arquivo de onde os dados serão lidos;
- length: tamanho em bytes do buffer de leitura;

Exemplo:

```
<?php
$fp = fopen("./dados.txt", "r");
$text = fgets($fp, 20);
fclose($fp);
?>
```

fgetc(): Esta função permite ler caractere por caractere de um arquivo. Seguem a sintaxe e um exemplo de utilização:

fgetc(handle);
- handle: handle do arquivo de onde os dados serão lidos;

Exemplo:

```
<?php
$fp = fopen("./dados.txt", "r");
do {
   $char .= fgetc($fp);
} while($char);
fclose($fp);
?>
```

file(): Esta função lê um arquivo completo, e armazena cada linha do arquivo como um elemento de um array. Depois de ler todo o conteúdo do arquivo, file() o fecha automaticamente, não sendo necessária uma chamada a fclose(); Vejamos a sintaxe:

file(filename);
- filename: nome ou caminho completo de um arquivo.

Exemplo:

```
<?php
$file_lines = file("./dados.txt");
echo "Primeira linha: " . $file_lines[0];
?>
```

Além dessas funções para leitura e escrita, existe ainda uma função bastante útil, que testa se o final do arquivo foi atingido. É a função feof(), que tem a seguinte sintaxe:

feof(handle);
- handle: handle do arquivo;

Exemplo:

```
<?php
$fp = fopen("./dados.txt", "r");
while(!feof($fp)) {
   $char .= fgetc($fp);
}
fclose($fp);
?>
```

É isso aí! Existem diversas outras funções em PHP para a manipulação de arquivo, inclusive para acesso aleatório. Porém, o intuito deste tutorial é mostrar a base da manipulação de arquivos. Aqueles que estiverem interessados em aprender o assunto mais a fundo, há uma referência de todas as funções da linguagem PHP no manual, disponível em http://www.php.net.

http://www.phpbrasil.com/artigo/FygWH2oM4L0/2/trabalhando-com-arquivos-em-php
Trabalhando com arquivos em PHP
Iae pessoal, tudo beleza?

Hoje vamos aprender um pouco sobre como ler e excluir linhas de arquivos TXT usando o PHP.

É muito simples, basicamente vamos trabalhar muito com arrays e depois o salvaremos no arquivo. Muitas pessoas preferem modificar diretamente no arquivo, mas assim, caso este esteja sendo usado por outro processo, poderá ser danificado ou os dados não serem salvos corretamente. Vamos lá!

Conteúdo do meuarquivo.txt

1|Meu nome|Meu site|Olá pessoal 2|Meu nome1|Meu site|Olá pessoal 3|Meu nome2|Meu site|Olá pessoal 4|Meu nome3|Meu site|Olá pessoal 5|Meu nome4|Meu site|Olá pessoal

Lendo linhas do array:

Primeiro, vamos colocar todo o arquivo num array. Use o seguinte código:
$meuArray = file("nomedoarquivo.txt"); // coloco todo o arquivo num array

Vamos excluir a linha que tenha o primeiro valor do array igual a 3. Para isto, temos que ler cada linha do array e fazer um "explode" delas, separando os "|" como se fosse em colunas.
Depois, faremos um [cfIF[/cf] perguntando se a coluna desejada é igual ao valor que queremos: 3

for($n=0; $n < count($meuArray); $n++) { // enquanto que a variável $n for menor que o número de linhas do array, incremento ($n + 1). $cadaLinha = explode("|", $meuArray[$n]); // crio outro array que possua as colunas (de acordo com o explode) da linha $n do $meyArray if($cadaLinha[0] == 3) { echo "Meu nome: $cadalinha[1] - Site: $cadalinha[2]"; // imprimo o que desejo } } // fecho o for

Excluindo linhas do array e salvando nos arquivos:

Faremos a mesma lógica de ler as linhas, sendo que a única diferença é que ao invés de ser igual a 3, ele gravará apenas se for diferente.

$meuArray = file("nomedoarquivo.txt"); // coloco todo o arquivo num array $arrayModificado = array(); // crio um array vazio.

for($n=0; $n < count($meuArray); $n++) { // enquanto que a variável $n for menor que o número de linhas do array, incremento ($n + 1). $cadaLinha = explode("|", $meuArray[$n]); // crio outro array que possua as colunas (de acordo com o explode) da linha $n do $meyArray if($cadaLinha[0] <> 3) { // se o valor da coluna 0 for diferente de 3, executo $arrayModificado[] = $meuArray[$n]; // coloco no outro array, que será salvo, os valores do array inicial. } } // fecho o for

Depois é só salvar o array no arquivo, incluindo cada linha do array já modificado:

$bufferArquivo = fopen('nomedoarquivo.txt',"w"); // abro arquivo somente para escrita, excluo suas linhas e coloco o ponteiro no começo. for($n=0; $n < count($arrayModificado); $n++) { // enquanto que a variável $n for menor que o número de linhas do array, incremento ($n + 1). fwrite($bufferArquivo, $arrayModificado[$n]); // gravo esta linha do array no arquivo } // fecho o for fclose($bufferArquivo); // fecho o arquivo.

Você pode pensar, mas porque não faço apenas um FOR e já gravo no arquivo?
Como falei no começo, este procedimento fará com que o arquivo possa ser danificado, caso algum outro processo tente gravá-lo. Então, é melhor trabalhar com o array e apenas depois de modificado o desejado, salvá-lo no arquivo.

Com este exemplo da exclusão, você poderá ainda adaptá-lo e editar uma linha específica.

Bom, por hoje é só.

Até mais pessoal.

Fechar Arquivo

fclose(handle_arquivo);

```
<?php
fclose($fp);
?>
```

Gravando em arquivo:

```
<?php
$fp = fopen("./dados.txt", "w");
fwrite($fp, "Olá mundo do PHP!"); // grava a string no arquivo. Se não existir será criado
fclose($fp);
?>
```

Lendo arquivo:

```
<?php
$fp = fopen("./dados.txt", "r");
$texto = fread($fp, 20); // lê 20 bytes do arquivo e armazena em $text
fclose($fp);
echo $texto;
?>
```

fgets(): Esta função é usada na leitura de strings de um arquivo. fgets() lê "length \u2013 1" bytes do arquivo. Se for encontrado o final da linha e o número de bytes especificados não tiver sido atingido, fgets() terminará a leitura no final da linha (ou no final do arquivo, se for o caso). Eis a sua sintaxe:

fgets(handle, length);
- handle: handle do arquivo de onde os dados serão lidos;
- length: tamanho em bytes do buffer de leitura;

Exemplo:

```
<?php
$fp = fopen("./dados.txt", "r");
$texto = fgets($fp, 3);
fclose($fp);
echo $texto;
?>
```

fgetc(): Esta função permite ler caractere por caractere de um arquivo. Seguem a sintaxe e um exemplo de utilização:

fgetc(handle);
- handle: handle do arquivo de onde os dados serão lidos;

Exemplo:

```
<?php
$fp = fopen("./dados.txt", "r");
while (!feof($fp)){
   $char .= fgetc($fp);
}
fclose($fp);
echo $char."<br><br>";
?>
```

file(): Esta função lê um arquivo completo, e armazena cada linha do arquivo como um elemento de um array. Depois de ler todo o conteúdo do arquivo, file() o fecha automaticamente, não sendo necessária uma chamada a fclose(); Vejamos a sintaxe:

file(filename);
- filename: nome ou caminho completo de um arquivo.

Exemplo:

```
<?php
// file() lê todo o arquivo
$file_lines = file("./dados.txt");
echo "Primeira linha: " . $file_lines[0]."<br>";
echo "Segunda linha: " . $file_lines[1]."<br>";
echo "Terceira linha: " . $file_lines[2];
?>
```

Além dessas funções para leitura e escrita, existe ainda uma função bastante útil, que testa se o final do arquivo foi atingido. É a função feof(), que tem a seguinte sintaxe:

feof(handle);
- handle: handle do arquivo;

Exemplo:

```
<?php
$fp = fopen("./dados.txt", "r");
while(!feof($fp)) {
   $char .= fgetc($fp);
}
fclose($fp);
echo $char;
?>
```

Contando o Número de Linhas de um arquivo

```
<?php
// Contar o número de linhas de um arquivo, iniciando com 1
$fp = "./dados.txt";
$line_count = count (file ($fp));
echo $line_count;
?>
```

Contar palavras de um arquivo

Contar palavras repetidas em um arquivo

```
$fn = "./dados.txt";
$f_contents = preg_split ("/[\s+/", implode ("", file ($fn)));
foreach ($f_content as $palavra) {
        $ar[$palavra]++;
}
print "A seguinte palavra tem duplicatas<br>";
foreach ($ar as $palavra => $conta_palavra) {
        if (conta_palavra > 1) {
                print "Palavra: $palavra<br>Número de ocorrências: $conta_palavra<br><br>";
        }
}
```

Ler de forma inversa um arquivo, linha a linha

```
<?php
$fn = "./dados.txt";
$f_contents = array_reverse (file ($fn));
foreach ($f_contents as $linha_inversa) {
        print $linha_inversa;
}
?>
```

Ler aleatoriamente linha de arquivo

```
<?php
$fn = "./pensamentos.txt";
$f_contents = file ($fn);
srand ((double)microtime()*1000000);
$linha_aleatoria = $f_contents[ rand (0, (count ($f_contents) - 1)) ];
print $linha_aleatoria;
?>
```

Ler linha específica de arquivo

```
<?php
$fn = "./dados.txt";
$nr_linha = 38;
$f_contents = file ($fn);
$sua_linha = $f_contents [$nr_linha];
print $sua_linha;
?>
```

6.1.1 - Operações com Diretórios

https://www.php.net/manual/pt_BR/book.dir.php

Mostrando conteúdo de diretório

```
<?php
$dn = opendir ("/home/1www/");
while ($file = readdir ($dn)) {
        print "$file<br>";
}
closedir($dn);
?>
```

Excluindo arquivos do SO

```
<?php
$fn = "./dados0.txt";
// Excluindo arquivo
$ret = unlink ($fn);
if ($ret){
         die ("Arquivo excluído!");
}else{
        die ("Erro ao excluir arquivo");
}
?>
```

Copiando arquivos

```
<?php
$fn = "./dados.txt";

if (copy ($fn, "dados0.txt")){
         die ("Arquivo '$fn' copiado para dados0.txt ");
}else{
        die ("Erro ao copiar arquivo");
}
?>
```

Processando todos os arquivos de um diretório

```
<?php
$dh = dir ("/home/1www/");
while ($entrada = $dh->read()) {
        print $entrada . "<br>";
}
$dh->close();
?>
```

Referência:
http://phpbrasil.com/articles/print.php/id/310

6.1.2- Testar se arquivo pode ser lido (Readable)

https://www.php.net/manual/pt_BR/function.fileperms.php

```
<?php
if (is_readable('http://127.0.0.1/index.html')) {
  header('Location: http://127.0.0.1/index.html');
}else{
        echo "Este arquivo não pode ser lido!";
}

?>
```

6.1.3 – Compactando e Descompactando arquivos Zip

https://www.php.net/manual/pt_BR/book.zip.php
https://www.php.net/manual/pt_BR/zip.examples.php
https://www.php.net/manual/pt_BR/ziparchive.extractto.php
https://stackoverflow.com/questions/8889025/unzip-a-file-with-php

Compactação simples

Compactar arquivos da pasta arquivos com o nome teste.zip

```
<?php
$zip = new ZipArchive();
$filename = "./teste.zip";

if ($zip->open($filename, ZipArchive::CREATE)!==TRUE) {
   exit("cannot open <$filename>\n");
}

$zip->addFromString("arquivos/arquivo.txt" . time(), "#1 This is a test string added as testfilephp.txt.\n");
$zip->addFromString("arquivos/zip.txt", "#2 This is a test string added as testfilephp2.txt.\n");
$zip->addFile("arquivos/upload.php","/testfromfile.php");
echo "numfiles: " . $zip->numFiles . "\n";
echo "status:" . $zip->status . "\n";
$zip->close();
?>
```

Extraindo arquivo Zip

Extrair o arquivo teste.zip para a pasta extraídos

```
<?php
$zip = new ZipArchive;
if ($zip->open('teste.zip') === TRUE) {
   $zip->extractTo('extraidos/');
   $zip->close();
   echo 'ok';
} else {
   echo 'failed';
}
?>
```

Exemplo2

```
<?php
// Descompactar o arquivo teste.zip para a pasta extra sanitizando a entrada
```

```
$file = 'teste.zip';

// get the absolute path to $file
$path = pathinfo(realpath($file), PATHINFO_DIRNAME);

$zip = new ZipArchive;
$res = $zip->open($file);
if ($res === TRUE) {
  // extract it to the path we determined above
//  $zip->extractTo($path);
  $zip->extractTo('extra');
  $zip->close();
  echo "WOOT! $file extracted to $path";
} else {
  echo "Doh! I couldn't open $file";
}
```

Colhendo informações sobre um arquivo zip

```
<?php
// Informações sobre o conteúdo de um arquivo ZIP
$za = new ZipArchive();

$za->open('teste.zip');
print_r($za);
var_dump($za);
echo "numFiles: " . $za->numFiles . "\n<br>";
echo "status: " . $za->status  . "\n<br>";
echo "statusSys: " . $za->statusSys . "\n<br>";
echo "filename: " . $za->filename . "\n<br>";
echo "comment: " . $za->comment . "\n<br>";

for ($i=0; $i<$za->numFiles;$i++) {
    echo "index: $i\n<br>";
    print_r($za->statIndex($i));
}
echo "numFile:" . $za->numFiles . "\n<br>";
?>
```

Na pasta Anexos/Estruturado/Zip neste repositório encontrará outros exemplos
https://github.com/ribafs/apostilas/tree/main/Anexos/Estruturado

6.1.4 – Upload de arquivos

Exemplo simples

Upload para a pasta uploads

```
<html>
<head>
  <title>Basic Upload</title>
</head>
<body>
  <form action="#" method="POST" enctype="multipart/form-data">
    <input type="file" name="fileUpload">
    <input type="submit" value="Enviar">
  </form>
</body>
</html>
```

```
<?php
// O diretório 'upload' precisa ter permissão de escrita
  if(isset($_FILES['fileUpload']))
  {
    date_default_timezone_set("America/Fortaleza"); //Definindo timezone padrão

    $ext = strtolower(substr($_FILES['fileUpload']['name'],-4)); //Pegando extensão do arquivo
    $new_name = date("Y.m.d-H.i.s") . $ext; //Definindo um novo nome para o arquivo
    $dir = 'uploads/'; //Diretório para uploads

    move_uploaded_file($_FILES['fileUpload']['tmp_name'], $dir.$new_name); //Fazer upload do arquivo
  }
// https://tableless.com.br/upload-de-arquivos-com-php/
?>
```

Exemplo 2

```
<!DOCTYPE html>
<html>
<head>
 <title>Upload your files</title>
</head>
<body>
 <form enctype="multipart/form-data" action="" method="POST">
  <p>Upload your file</p>
  <input type="file" name="uploaded_file"></input><br />
  <input type="submit" value="Upload"></input>
 </form>
</body>
</html>
<?PHP
 if(!empty($_FILES['uploaded_file']))
 {
  $path = "uploads/";
  $path = $path . basename( $_FILES['uploaded_file']['name']);

  if(move_uploaded_file($_FILES['uploaded_file']['tmp_name'], $path)) {
    echo "The file ".  basename( $_FILES['uploaded_file']['name']).
    " has been uploaded";
  } else{
      echo "There was an error uploading the file, please try again!";
  }
 }
// https://gist.github.com/taterbase/2688850
?>
```

6.2 – Trabalhando com Arrays

Array/matriz - é um mapa ou conjunto ordenado de pares de chaves e valores.

Uma chave é como um índice
Um valor é como uma variável

Array vazio

$ar = array();

https://phpro.org/tutorials/Introduction-To-Arrays.html

Funções para Array

Veja também is_array(), explode(), implode(), split(), preg_split(), e unset().

Índice

array_splice — Remove uma parcela do array e substitui com outros elementos
array_sum — Calcula a soma dos elementos de um array
array_udiff_assoc — Computa a diferença entre arrays com checagem adicional de índice, compara dados por uma função de callback
array_udiff_uassoc — Computa a diferença entre arrays com checagem adicional de índice, compara dados e índices por uma função de callback
array_udiff — Computa a diferença de arrays usando uma função de callback para comparação dos dados
array_uintersect_assoc — Computa a interseção de arrays com checagem adicional de índice, compara os dados utilizando uma função de callback
array_uintersect_uassoc — Computa a interseção de arrays com checagem adicional de índice, compara os dados e os índices utilizando funções de callback separadas
array_uintersect — Computa a interseção de array, comparando dados com uma função callback
array_unique — Remove os valores duplicados de um array
array_unshift — Adiciona um ou mais elementos no início de um array
array_values — Retorna todos os valores de um array
array_walk_recursive — Aplica um função do usuário recursivamente para cada membro de um array
array_walk — Aplica uma determinada função em cada elemento de um array
array — Cria um array
arsort — Ordena um array em ordem descrescente mantendo a associação entre índices e valores
asort — Ordena um array mantendo a associação entre índices e valores
compact — Cria um array contendo variáveis e seus valores
count — Conta o número de elementos de uma variável, ou propriedades de um objeto
current — Retorna o elemento corrente em um array
each — Retorna o par chave/valor corrente de um array e avança o seu cursor
end — Faz o ponteiro interno de um array apontar para o seu último elemento
extract — Importa variáveis para a tabela de símbolos a partir de um array
in_array — Checa se um valor existe em um array
key_exists — Sinônimo de array_key_exists
key — Retorna uma chave de um array
krsort — Ordena um array pelas chaves em ordem descrescente
ksort — Ordena um array pelas chaves
list — Cria variáveis como se fossem arrays
natcasesort — Ordena um array utilizando o algoritmo da "ordem natural" sem diferenciar maiúsculas e minúsculas
natsort — Ordena um array utilizando o algoritmo da "ordem natural"
next — Avança o ponteiro interno de um array
pos — Sinônimo de current
prev — Retrocede o ponteiro interno de um array
range — Cria um array contendo uma faixa de elementos
reset — Faz o ponteiro interno de um array apontar para o seu primeiro elemento
rsort — Ordena um array em ordem descrescente
shuffle — Mistura os elementos de um array
sizeof — Sinônimo de count
sort — Ordena um array
uasort — Ordena um array utilizando uma função de comparação definida pelo usuário e mantendo as associações entre chaves e valores
uksort — Ordena um array pelas chaves utilizando uma função de comparação definida pelo usuário.
usort — Ordena um array pelos valores utilizando uma função de comparação definida pelo usuário

https://www.php.net/manual/pt_BR/ref.array.php

Trabalhando com Arrays

Algumas das funções

Um array no PHP é um mapa ordenado, que relaciona valores com chaves.

Especificando um array()

array([chave =>] valor, ...);

A chave pode ser uma string ou um inteiro.
O valor pode ser qualquer dos tipos permitidos.

Essas funções permitem a interação e manipulação de arrays de várias formas. Arrays são essenciais para armazenar, gerenciar, operar sobre um conjunto de variáveis.

Arrays simples e multidimensionais (matrizes) são suportados, e podem ser criados pelo usuário ou por outras funções. Existem diversas funções específicas para bancos de dados para preencher arrays com os dados retornados em consultas, e vários outros tipos de funções também retornam arrays.

array_fill

array_fill -- Preenche um array com valores

array array_fill (int start_index, int num, mixed value)

```
<?php
$a = array_fill(5, 6, 'banana');
print_r($a);
?>
```

array_merge

array_merge -- Funde dois ou mais arrays

array array_merge (array array1, array array2 [, array ...])

```
<?php
$array1 = array();
$array2 = array(1 => "data");
$result = array_merge($array1, $array2);
?>
```

Não esqueça que as chaves numéricas serão reordenadas!

```
Array
(
   [0] => data
)
```

Se você quer preservar os arrays e apenas concatená-los, o operador +:

```
<?php
$array1 = array();
$array2 = array(1 => "data");
$result = $array1 + $array2;
?>
```

As chaves numéricas serão preservadas e as associações originais permanecem.

array_pad

array_pad -- Expande um array para um certo comprimento utilizando um determinado valor

array array_pad (array input, int pad_size, mixed pad_value)

Exemplo 1. Exemplo de array_pad()

```
<?php
$input = array(12, 10, 9);

$result = array_pad($input, 5, 0);
// $result é array(12, 10, 9, 0, 0)

$result = array_pad($input, -7, -1);
// $result é array(-1, -1, -1, -1, 12, 10, 9)

$result = array_pad($input, 2, "noop");
// Não será expandido
?>
```

array_pop

array_pop -- Retira um elemento do final do array

mixed array_pop (array array)

```
<?php
$cesta = array("laranja", "banana", "melancia", "morango");
$fruta = array_pop($cesta);
print_r($cesta);
?>
```

array_push

array_push -- Adiciona um ou mais elementos no final de um array

int array_push (array array, mixed var [, mixed ...])

```
<?php
$cesta = array("laranja", "morango");
array_push($cesta, "melancia", "batata");
print_r($cesta);
?>
```

array_reverse

array_reverse -- Retorna um array com os elementos na ordem inversa

array array_reverse (array array [, bool preserve_keys])

```
<?php
$input = array("php", 4.0, array ("verde", "vermelho"));
$result = array_reverse($input);
$result_keyed = array_reverse($input, TRUE);
print_r($result_keyed);
?>
```

array_search

array_search -- Procura por um valor em um array e retorna sua chave correspondente caso seja encontrado. Caso contrário retorna FALSE.

mixed array_search (mixed procurar_este, array procurar_neste [, bool strict])

```
<?php
$a=array("a","b",0,"c","d");
echo "a: ".array_search("a",$a)."<br>";
echo "b: ".array_search("b",$a)."<br>";
echo "c: ".array_search("c",$a)."<br>";
echo "d: ".array_search("d",$a)."<br>";
echo "0: ".array_search("0",$a)."<br>";
echo "x: ".array_search("x",$a)."<br>";
echo "1: ".array_search("1",$a);
?>
```

```
<?php
if (array_search($needle, $array)!== FALSE) {
  //code goes here (
}
?>
```

```
<?php

function array_replace($search, $replace, &$array) {
 foreach($array as $key => $value) {
  if($value == $search) {
    $array[$key] = $replace;
  }
  }
}
?>
```

```
<?
$Projects[0] = array(123, "Text 1");
$Projects[1] = array(456, "Text 2");
$Projects[2] = array(789, "Text 3");

$search_value = "ext 3";

foreach ($Projects as $key => $row){
    foreach($row as $cell){
      if (strpos($cell, $search_value) !== FALSE){
                    echo  "<p>Project ".$key;
```

```
    }
   }
}
?>
```

array_shift

array_shift -- Retira o primeiro elemento de um array

mixed array_shift (array array)

```
<?php
$cesta = array("laranja", "banana", "melancia", "morango");
$fruta = array_shift($cesta);
print_r($cesta);
?>
```

array_sum

array_sum -- Calcula a soma dos elementos de um array

mixed array_sum (array arr)

```
<?php
$a = array(2, 4, 6, 8);
echo "soma(a) = ".array_sum($a)."<br>";

$b = array("a" => 1.2, "b" => 2.3, "c" => 3.4);
echo "soma(b) = ".array_sum($b)."<br>";
?>
```

array_unique

array_unique -- Remove os valores duplicados de um array

array array_unique (array array)

```
<?php
$input = array("a" => "verde", "vermelho", "b" => "verde", "azul", "vermelho");
$result = array_unique($input);
print_r($result);
?>
```

Exemplo 2. array_unique() e tipos

```
<?php
$input = array(4, "4", "3", 4, 3, "3");
$result = array_unique($input);
var_dump($result);
?>
```

array_values

array_values -- Retorna todos os valores de um array

array array_values (array input)

//Retorna os valores, as chaves não

```
<?php
$array = array("tamanho" => "G", "cor" => "dourado");
print_r(array_values ($array));
?>
```

array

array -- Cria um array

array array ([mixed ...])

Exemplo 1. Exemplo de array()

```
<?php
$frutas = array (
   "frutas" => array("a"=>"laranja", "b"=>"banana", "c"=>"maçã"),
   "numeros" => array(1, 2, 3, 4, 5, 6),
   "buracos" => array("primeiro", 5 => "segundo", "terceiro")
)
?>
```

Exemplo 2. Indexação automática com array()

```
<?php
$array = array(1, 1, 1, 1,  1, 8 => 1,  4 => 1, 19, 3 => 13);
print_r($array);
?>
```

arsort

arsort -- Ordena um array em ordem descrescente dos valores
mantendo a associação entre índices e valores

void arsort (array array [, int sort_flags])

```
<?php
$frutas = array("d" => "limao", "a" => "laranja", "b" => "banana", "c" => "melancia");
arsort($frutas);
reset($frutas);
while (list($chave, $valor) = each($frutas)) {
   echo "$chave = $valor\n";
}
?>
```

asort

asort -- Ordena um array em ordem crescente dos valores
mantendo a associação entre índices e valores

void asort (array array [, int sort_flags])

```
<?php
$frutas = array("d" => "limao", "a" => "laranja", "b" => "banana", "c" => "melancia");
asort($frutas);
reset($frutas);
while (list($chave, $valor) = each($frutas)) {
  echo "$chave = $valor\n";
}
?>
```

count

count -- Conta o número de elementos de uma variável

int count (mixed var [, int mode])

```
<?php
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$a[3] = 6;
$result = count($a);
// $result == 4
print $result."<br>";

$b[0] = 7;
$b[5] = 9;
$b[10] = 11;
$result = count($b);
// $result == 3;
print $result;
?>
```

Exemplo 2. Uso recursivo da função count() (PHP >= 4.2.0)

```
<?php
$food = array( 'fruits'  => array('orange', 'banana', 'apple'),
'veggie'  => array('carrot', 'collard','pea'));
// recursive count
echo count($food,COUNT_RECURSIVE);  // mostra 8
// normal count
echo count($food);                  // mostra2 2
?>

<?php
$food = array( 'fruits'  => array('orange', 'banana', 'apple'),
'veggie'  => array('carrot', 'collard','pea'));
// recursive count
echo count($food,COUNT_RECURSIVE)."<br>";  // mostra 8
// normal count
echo count($food);                  // mostra2 2
?>
```

current

current -- Retorna o elemento corrente em um array

mixed current (array array)

```
<?php
$transport = array('foot', 'bike', 'car', 'plane');
$mode = current($transport); // $mode = 'foot';
echo "Atual $mode<br>";
$mode = next($transport);    // $mode = 'bike';
echo "Atual $mode<br>";
$mode = current($transport); // $mode = 'bike';
echo "Atual $mode<br>";
$mode = prev($transport);    // $mode = 'foot';
echo "Atual $mode<br>";
$mode = end($transport);    // $mode = 'plane';
echo "Atual $mode<br>";
$mode = current($transport); // $mode = 'plane';
echo "Atual $mode<br>";
?>
```

each

each -- Retorna o par chave/valor corrente de um array e avança o seu cursor

array each (array array)

```
<?php
$foo = array("bob", "fred", "jussi", "jouni", "egon", "marliese");
$bar = each($foo);
print_r($bar);
?>

<?php
  $foo = array("Robert" => "Bob", "Seppo" => "Sepi");
  $bar = each($foo);
  print_r($bar);
?>
```

Percorrendo um array com each()

```
<?php
$fruit = array('a' => 'apple', 'b' => 'banana', 'c' => 'cranberry');
reset($fruit);
while (list($key, $val) = each($fruit)) {
    echo "$key => $val\n";
}
/* Saída:

  a => apple
  b => banana
  c => cranberry

  */
?>
```

end

end -- Faz o ponteiro interno de um array apontar para o seu último elemento

mixed end (array array)

```
<?php
 $frutas = array('melancia', 'banana', 'morango');
 print end($frutas); // morango
?>
```

key

key -- Retorna uma chave da posição atual de um array associativo

mixed key (array array)

```
<?php
 $array = array(
    'fruit1' => 'apple',
    'fruit2' => 'orange',
    'fruit3' => 'grape',
    'fruit4' => 'apple',
    'fruit5' => 'apple');

 // este ciclo  exibirá todas as chaves do array associativo
 // ausiliado pela função next()
 while ($fruit_name = current($array)) {
        echo key($array).'<br>';
                next($array);
 }
 ?>
```

```
<?php
 $array = array(
    'fruit1' => 'apple',
    'fruit2' => 'orange',
    'fruit3' => 'grape',
    'fruit4' => 'apple',
    'fruit5' => 'apple');

 // este ciclo  exibirá toda a chave do array associativo
 // onde o valor é igual a "apple"
 while ($fruit_name = current($array)) {
    if ($fruit_name == 'apple') {
        echo key($array).'<br>';
    }
    next($array);
 }
 ?>
```

next

next -- Avança o ponteiro interno de um array

mixed next (array array)

```
<?php
  $transport = array('foot', 'bike', 'car', 'plane');
  $mode = current($transport); // $mode = 'foot';
  print"$mode<br>";
  $mode = next($transport);    // $mode = 'bike';
  print"$mode<br>";
  $mode = next($transport);    // $mode = 'car';
  print"$mode<br>";
  $mode = prev($transport);    // $mode = 'bike';
  print"$mode<br>";
  $mode = end($transport);    // $mode = 'plane';
  print"$mode<br>";
?>
```

prev

prev -- Retrocede o ponteiro interno de um array

mixed prev (array array)

```
<?php
  $transport = array('foot', 'bike', 'car', 'plane');
  $mode = current($transport); // $mode = 'foot';
print"$mode<br>";
  $mode = next($transport);    // $mode = 'bike';
print"$mode<br>";
  $mode = next($transport);    // $mode = 'car';
print"$mode<br>";
  $mode = prev($transport);    // $mode = 'bike';
print"$mode<br>";
  $mode = end($transport);    // $mode = 'plane';
print"$mode<br>";
?>
```

reset

reset -- Faz o ponteiro interno de um array apontar para o seu primeiro elemento

mixed reset (array array)

```
<?php
  $array = array('primero passo', 'segundo passo', 'terceiro passo', 'quarto passo');

  // por definição, o ponteiro está sobre o primeiro elemento
  echo current($array)."<br>\n"; // "Primeiro passo"

  // pula dois passos
  next($array);
  next($array);
  echo current($array)."<br>\n"; // "passo três"
```

```
 // reinicia o ponteiro, começa novamente o primeiro passo
 reset($array);
 echo "Depois de resetado...: " . current($array)."<br>\n"; // "primeiro passo"
?>
```

sizeof

sizeof -- Apelido de count()

sort

sort -- Ordena um array pelo seu valor

void sort (array array [, int sort_flags])

```
<?php

$frutas = array("limao", "laranja", "banana", "melancia");
sort($frutas);
reset($frutas);
while (list($chave, $valor) = each($frutas)) {
  echo "frutas[".$chave."] = ".$valor."<br>";
}
?>
```

Os seguintes também são funcionalmente idênticos:

```
<?php
$arr = array("one", "two", "three");
reset($arr);
while (list($key, $value) = each ($arr)) {
   echo "Chave: $key; Valor: $value<br />\n";
}

foreach ($arr as $key => $value) {
   echo "Chave: $key; Valor: $value<br />\n";
}
?>
```

Mais alguns exemplos para demonstrar os usos:

```
<?php
/* exemplo foreach 1: somente valores */

$a = array(1, 2, 3, 17);

foreach ($a as $v) {
   echo "Valor atual de \$a: $v.\n";
}

/* exemplo foreach 2: valores (com as chaves impressas para ilustração) */

$a = array(1, 2, 3, 17);

$i = 0; /* para exemplo somente */

foreach ($a as $v) {
   echo "\$a[$i] => $v.\n";
```

```
    $i++;
}

/* exemplo foreach 3: chaves e valores */

$a = array (
    "um" => 1,
    "dois" => 2,
    "três" => 3,
    "dezessete" => 17
);

foreach ($a as $k => $v) {
    echo "\$a[$k] => $v.\n";
}

/* exemplo foreach 4: arrays multidimensionais */

$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";

foreach ($a as $v1) {
    foreach ($v1 as $v2) {
        echo "$v2\n";
    }
}

/* exemplo foreach 5: arrays dinâmicos */

foreach (array(1, 2, 3, 4, 5) as $v) {
    echo "$v\n";
}
?>
```

Exemplo de array multidimensional

```
$produto[1][codigo] = "1";
$produto[1][nome] = "João Pereira Brito";
$produto[1][email] = "joao@joao.org";
$produto[1][rua] = "Vasco da Gama";
$produto[1][numero] = "1345";

$produto[2][codigo] = "2";
$produto[2][nome] = "Antônio queiroz";
```

Exemplo de Array

```
$i=0;
while($i < $numregs){
            $codigo=pg_result($consulta,$i,codigo);
            $nome=pg_result($consulta,$i,nome);
            $venc=pg_result($consulta,$i,vencimento);
            $apartamento=pg_result($consulta,$i,apartamento);
            $pessoas=pg_result($consulta,$i,pessoas);
            $cota_agua=pg_result($consulta,$i,cota_agua);
            $cota_condominio=pg_result($consulta,$i,cota_condominio);
            $cota_reserva=pg_result($consulta,$i,cota_reserva);
```

```
        $total = $cota_agua + $cota_condominio + $cota_reserva;
        $total = number_format($total,2, ',','.');
...
$i++;
}
```

Observação
Veja exemplos no repositório na pasta
Arquivos/Estruturado/Arrays
https://github.com/ribafs/apostilas/tree/main/Anexos/Estruturado
Inclusive arrays multidimensionais de até 5 dimensões

6.3 – Trabalhando com JSON em PHP

https://www.php.net/manual/pt_BR/book.json.php

Esta extensão implementa o formato de troca de dados » JavaScript Object Notation (JSON). A decodificação é feita por um parser baseado no JSON_checker de Douglas Crockford.
Nota:

O PHP implementa uma extensção do JSON, além do especificado no » RFC 4627 - podendo também codificar e decodificar tipos escalares e **`null`**. RFC 4627 apenas suporta esses valores quando eles estão inseridos dentro de um objeto ou array.

Embora isso seja consistente com a definição expandida de "texto JSON" na nova » RFC 7159 (que pretende susceder RFC 4627) e ECMA-404, isto pode causar problemas de interoperabilidade com parsers JSON antigos que aderem estritamente a RFC 4627 quando decodificando um valor escalar.

Funções da JSON

- json_decode — Decodifica uma string JSON
- json_encode — Retorna a representação JSON de um valor
- json_last_error_msg — Retorna uma string contento a mensagem de erro da ultima chamada de json_encode() ou json_decode()
- json_last_error — Retorna o último erro ocorrido

Exemplo

```
<?php

class obj{
   public $name;
   public $age;
   public $city;
}

$myObj = new obj;

$myObj->name = "John";
$myObj->age = 30;
$myObj->city = "New York";

$myJSON = json_encode($myObj);
echo $myJSON;
```

Exemplo 2

```
<?php
$myArr = array("John", "Mary", "Peter", "Sally");
$myJSON = json_encode($myArr);
echo $myJSON;
?>
```

Exemplo com Banco de dados

```
<?php
header("Content-Type: application/json; charset=UTF-8");
$obj = json_decode($_GET["x"], false);

$conn = new mysqli("myServer", "myUser", "myPassword", "Northwind");
$stmt = $conn->prepare("SELECT name FROM customers LIMIT ?");
$stmt->bind_param("s", $obj->limit);
$stmt->execute();
$result = $stmt->get_result();
$outp = $result->fetch_all(MYSQLI_ASSOC);
echo json_encode($outp);
?>
```

Referências

https://www.w3schools.com/js/js_json_php.asp
https://zetcode.com/php/json/

6.4 – Path relativo e absoluto

É verdade que muitas aplicações orientadas a objetos usam namespace e praticamente não precisam usar os paths relativos e absolutos.
Acontece que muitos aplicativos que usam o PHP de forma estruturada usam os paths relativos e absolutos.

Mesmo em aplicativos que usam namespaces, usamos arquivos com constantes para facilitar o acesso as partes do aplicativo.

Como não é um assunto intuitivo quiz trazer aqui as minhas reflexões e experiências com a intenção de facilitar para quem tem algumas dificuldade.

Paths relativos e absolutos
Tanto na web quanto em diretórios

Para a web

Um exemplo de path absoluto

Tenho a web em /var/www/html e minha aplicação em /var/www/html/app

Na minha aplicação eu tenho o arquivo conexao.php no raiz. O path absoluto deste aquivo está abaixo, mas não podemos acessar desta forma no aplicativo. Veja abaixo que usaremos a função dirname para isso

/var/www/html/app/conexao.php

Tenho também
/var/www/html/app/classes

O arquivo /var/www/html/app/classes/Crud.php pode incluir conexao.php de forma absoluta ou relativa

Em /var/www/html/app/classes/Crud.php

Relativo

Eu estando na pasta app/classes, o path relativo para Crud.php seria ./Crud.php

./ - diretório atual
../ um nível acima
../.. dois níveis acima

De forma absoluta

require dirname(__DIR__).'/conexao.php';// Este require não pode ter o once, senão não mostra o conteúdo pela segunda vez

De forma relativa

require '../conexao.php'; // Estando em app/classes volta um nível e inclue conexao.php

Paths absolutos começam com / ou c:\
Relativos não começam com / ou usam ./

Há uma recomendação aqui:
https://gist.github.com/DaveRandom/6830e379578a66e2c82593137e79d099

"Os paths relativos dependem do diretório de trabalho atual. Esta é uma forma de estado global, e como tal deve ser evitada sempre que possível."

Referências
https://stackoverflow.com/questions/17407664/php-include-relative-path

Path em windows e linux

Unix: /full/path/to/file.php

Windows C:\full\path\to\file.php

/var/www/html/

c:\xampp\htdocs

6.5 - Sessions e Cookies

Session:
- Armezenado no servidor
- Rápido, pois já está no servidor

- O tamanhod e um session depende da configuração do php
- São mais seguros que cookies
- Disponível imediatamente no código sem precisar de reload

Cookies:
- Armazenados no computador do cliente
- Lentos, pois precisam sair do computador do cliente para o servidor
- Tamanho limitado
- Pode ser visualizado e modificado pelo cliente, o que acarreta problema de segurança
- Não está disponível enquanto a página nãor ecarrega

Quando trabalhando com dados sigilosos, como senha e númerod e cartão de crédito, cpf, etc devemos usar session. Com outros tipos de dados podemos usar cookies.

Sessões em PHP são, por padrão, armazenadas no /tmp e isso pode ser um grande problema em servidores compartilhados, visto que neste diretório qualquer usuário tem permissão de leitura e escrita.

É recomendado que regularmente regeremos o ID da sessão.

Exemplos na pasta do repositório
https://github.com/ribafs/apostilas/tree/main/Anexos/Estruturado

Arquivos/Estruturado/Sessions

6.6 – Trabalhando com Data e Hora usando a classe DateTime

https://www.php.net/manual/pt_BR/class.datetime.php

Índice

DateTime::add — Adiciona uma quantidade de dias, meses, anos, horas, minutos e segundos de um objeto DateTime
DateTime::__construct — Retorna um novo objeto DateTime
DateTime::createFromFormat — Retorna um novo objeto DateTime formatado de acordo com um formato informado
DateTime::createFromImmutable — Returns new DateTime object encapsulating the given DateTimeImmutable object
DateTime::createFromInterface — Returns new DateTime object encapsulating the given DateTimeInterface object
DateTime::getLastErrors — Retorna os avisos e erros
DateTime::modify — Altera o timestamp
DateTime::__set_state — O manipulador __set_state
DateTime::setDate — Define a data
DateTime::setISODate — Define uma data ISO
DateTime::setTime — Define o horário
DateTime::setTimestamp — Define a data e hora baseada em um timestamp Unix
DateTime::setTimezone — Define o fuso horário de um objeto DateTime
DateTime::sub — Subtrai uma quantidade de dias, meses, anos, horas, minutos e segundos de um objeto DateTime

É importante usar esta classe ao invés de **strtotime** e **date**. A classe DateTime tem algumas vantagens.

Date Interval

```
// Each set of intervals is equal.
$i = new DateInterval('P1D');
$i = DateInterval::createFromDateString('1 day');

$i = new DateInterval('P2W');
$i = DateInterval::createFromDateString('2 weeks');

$i = new DateInterval('P3M');
$i = DateInterval::createFromDateString('3 months');

$i = new DateInterval('P4Y');
$i = DateInterval::createFromDateString('4 years');

$i = new DateInterval('P1Y1D');
$i = DateInterval::createFromDateString('1 year + 1 day');

$i = new DateInterval('P1DT12H');
$i = DateInterval::createFromDateString('1 day + 12 hours');

$i = new DateInterval('PT3600S');
$i = DateInterval::createFromDateString('3600 seconds');
?>
```

Exemplos

```
$currentDateTime = new DateTime();
print_r($currentDateTime);

$hoje = new DateTime('yesterday');

$doisDiasAtraz = new DateTime('+ 2 days');

$umaSemanaAntes = new DateTime('- 1 week');
```

Formatando

```
$agora = new DateTime();

$agora = $agora->format('d/m/Y');

print $agora;
```

```
$agora = $agora->format('d/m/Y s:i:H);
```

Outros formatos:

```
print_r($now->format('jS F Y'));

print_r($now->format('ga jS M Y'));

print_r($now->format('Y/m/d s:i:H'));
```

Adicionando datas

Somar 2 dias à data atual

```
$today = new DateTime('today');

echo $today->format('Y-m-d') . PHP_EOL;
$today->add(new DateInterval('P2D')); // Somar 2 dias

echo $today->format('Y-m-d') . PHP_EOL;
```

Subtração de Datas

```
<?php
$today = new DateTime('today');

echo $today->format('Y-m-d') . PHP_EOL; // Mostra a data atual e dá uma quebra de linha

$today->modify('-2 days');

echo $today->format('Y-m-d') . PHP_EOL; // Mostra a data atual menos 2 dias
```

https://startutorial.com/view/master-php-datetime

Recebendo o Timezone atual

```
$date = new DateTime();
$tz = $date->getTimezone();
echo $tz->getName();
```

Alterando o Timezone atual

```
$date = new DateTime();
echo 'Timezone atual: '.$date->format('Y-m-d H:i:sP') . "<br>";

$date->setTimezone(new DateTimeZone('Pacific/Chatham'));
echo 'Novo Tz: '.$date->format('Y-m-d H:i:sP') . "\n<br>";
```

https://www.phptutorial.net/php-oop/php-datetime/

6.7 – Trabalhando com Strings

Uma string é uma série de caracteres, onde um caractere é o mesmo que um byte. Isso significa que o PHP possui suporte a um conjunto de apenas 256 caracteres, e, portanto, não possui suporte nativo a Unicode. Veja mais detalhes do tipo string.

> **Nota**: A partir do PHP 7.0.0, não há restrições particulares em relação ao tamanho de uma uma string em compilações de 64 bits. Em compilações de 32 bits e em versões anteriores, uma string pode ter o tamanho de até 2GB (máximo de 2147483647 bytes)

Sintaxe

Uma string literal pode ser especificada de quatro formas diferentes.

- aspas simples
- aspas duplas
- sintaxe heredoc
- sintaxe nowdoc (a partir do PHP 5.3.0)

Aspas simples

A maneira mais simples de se especificar uma string é delimitá-la entre aspas simples (o caractere `'`).

Para especificar um apóstrofo, escape-o com uma contrabarra (\). Para especificar uma contrabarra literal, dobre-a (\\). Todas as outras ocorrências da contrabarra serão tratadas como uma contrabarra literal: isso significa que outras sequências de escape que se esteja acostumado a utilizar, como \r ou \n, serão literalmente impressas em vez de ter qualquer significado especial.

> **Nota**: Diferentemente das sintaxes com aspas duplas e heredoc, variáveis e sequências de escape para caracteres especiais *não* serão expandidas quando ocorrerem dentro de uma string delimitada por aspas simples.

Exemplo

```
echo 'isto é uma string comum';
```

Aspas duplas

Se a string for delimitada entre aspas duplas ("), o PHP interpretará seu conteúdo.

Heredoc

Uma terceira maneira de delimitar strings é a sintaxe heredoc: <<<. Após este operador, um identificador é fornecido seguido de uma nova linha. A própria string é colocada em seguida e a seguir o mesmo identificador novamente para fechar a string.

O identificador de fechamento precisa começar na primeira coluna da linha. Além disso, o identificador precisa seguir as mesmas regras de nomeação como qualquer outro rótulo no PHP: deve conter somente caracteres alfanuméricos e sublinhados, e precisa começar com um caractere não numérico ou sublinhado.

Aviso

É muito importante notar que a linha que contêm o identificador de fechamento não deve conter nenhum outro caractere, com exceção do ponto e vírgula (;). Isso significa que o identificador não deve ser indentado e não deve ter nenhum espaço ou tabulações antes ou depois do ponto-e-vírgula. Também é importante perceber que o primeiro caractere antes do identificador de fechamento deve ser uma nova linha como definido pelo sistema operacional local. Isso é, \n em sistemas UNIX, incluindo o MAC OS X. O identificador de fechamento também deve ser seguido por uma nova linha.

Se essa regra for quebrada e o identificador de fechamento não estiver "limpo", ele não será considerado um identificador de fechamento, e o PHP continuará procurando por um. Se um identificador de fechamento apropriado não for encontrado antes do final do arquivo atual, um erro de interpretação será lançado na última linha.

Exemplo #1 Exemplo inválido

```
<?php
class foo {
    public $bar = <<<EOT
bar
    EOT;
}
```

```
// O identifcador não deve ser indentado

?>
```

Exemplo #2 Exemplo válido

```
<?php
class foo {
    public $bar = <<<EOT
bar
EOT;
}
?>
```

Mais detalhes:
https://www.php.net/manual/pt_BR/language.types.string.php

Substrings

substr -- Retorna uma parte de uma string

string substr (string string, int start [, int length])

Exemplo 1. Uso basico de substr()

```
<?php
$rest = substr("abcdef", 1);    // retorna "bcdef"
$rest = substr("abcdef", 1, 3); // retorna "bcd"
$rest = substr("abcdef", 0, 4); // retorna "abcd"
$rest = substr("abcdef", 0, 8); // retorna "abcdef"

// Outra opção é acessar atravéz de chaves
$string = 'abcdef';
echo $string{0};                // retorna a
echo $string{3};                // retorna d
?>
```

Se start for negativo, a string retornada irá começar no caractere start a partir do fim de string.

Exemplo 2. Usando um inicio negativo

```
<?php
$rest = substr("abcdef", -1);    // retorna "f"
$rest = substr("abcdef", -2);    // retorna "ef"
$rest = substr("abcdef", -3, 1); // retorna "d"
?>
```

Exemplo 3. Usando um length negativo

```
<?php
$rest = substr("abcdef", 0, -1);  // retorna "abcde"
$rest = substr("abcdef", 2, -1);  // retorna "cde"
$rest = substr("abcdef", 4, -4);  // retorna ""
$rest = substr("abcdef", -3, -1); // retorna "de"
?>
```

Sobrescrevendo Strings

str_replace

str_replace -- Substitui todas as ocorrências da string de procura com a string de substituição
mixed str_replace (mixed pesquisa, mixed substitui, mixed assunto [, int &count])

```
<?php
// Fornece: <body text='black'>
$bodytag = str_replace("%body%", "black", "<body text='%body%'>");

// Fornece: Hll Wrld f PHP
$vowels = array("a", "e", "i", "o", "u", "A", "E", "I", "O", "U");
$onlyconsonants = str_replace($vowels, "", "Hello World of PHP");

// Fornece: você comeria pizza, cerveja e sorvete todos os dias
$frase  = "você comeria frutas, vegetais, e fibra todos os dias.";
$saldavel = array("frutas", "vegetais", "fibra");
$saboroso  = array("pizza", "cerveja", "sorvete");

$novafrase = str_replace($saudavel, $saboroso, $frase);

// Uso do parâmetro count está disponível no PHP 5.0.0
$str = str_replace("ll", "", "good golly miss molly!", $count);
echo $count; // 2
?>
```

substr_replace
substr_replace -- Substitui o texto dentro de uma parte de uma string
string substr_replace (string string, string replacement, int start [, int length])

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo "Original: $var<hr>\n";

/* Estes dois exemplos substituem tudo de $var com 'bob'. */
echo substr_replace($var, 'bob', 0) . "<br>\n";
echo substr_replace($var, 'bob', 0, strlen($var)) . "<br>\n";

/* Insere 'bob' direto no começo de $var. */
echo substr_replace($var, 'bob', 0, 0) . "<br>\n";

/* Estes dois exemplos substituem 'MNRPQR' em $var com 'bob'. */
echo substr_replace($var, 'bob', 10, -1) . "<br>\n";
echo substr_replace($var, 'bob', -7, -1) . "<br>\n";

/* Deleta 'MNRPQR' de $var. */
echo substr_replace($var, '', 10, -1) . "<br>\n";
?>
```

Encontrar posição de string

strpos
strpos -- Encontra a posição da primeira ocorrência de uma string
int strpos (string str, string procurar [, int offset])

Exemplos strpos()

```
<?php
//$str = 'abc';
$str = 'cba';
$procurar  = 'a';
```

```
$posicao = strpos($str, $procurar);

// Note o uso de ===.  Simples == não funcionaria como esperado
// por causa da posição de 'a' é 0 (primeiro) caractere.
if ($pos === false) {
   echo "A string '$procurar' não foi encontrada na string '$str'";
} else {
   echo "A string '$procurar' foi encontrada na string '$str'";
   echo " e está na posição $posicao";
}

?>

<?php

//$email = 'ribafs@gmail.com.br';
$email = 'ribafs@gmail.com';
$usuario = substr ($email, 0, strpos ($email, '@'));
// Lembrando: substr ( string string, int start [, int length] )
$dominio = substr ($email, strpos ($email, '@')+1);
echo "Usuário '$usuario' e Domínio '$dominio'"; // o comprimento default é até o final
?>
```

Contando as ocorrências de substring em string

substr_count -- Conta o número de ocorrências de uma substring
int substr_count (string str, string conte_me)
substr_count() retorna o número de vezes que a substring conte_me ocorre na string str.

```
<?php
  $str = "Olá mundo do PHP";

  if (substr_count($str, "do") == 0)
   echo "nenhum";

  // same as:

  if (strpos($str, "do") === false)
   echo "nenhum";
?>
```

Exemplo 1. Exemplo substr_count()

```
<?php
print substr_count("This is a test", "is"); // mostra 2
?>
```

Trocando ponto por vírgula e vice-versa

Se temos campos tipo moeda, devemos exibir com vírgula e gravar no banco com ponto.
Para isso uma boa saída é usar a dupla de funções implode e explode.

Antes de exibir na tela (em consultas):

```
$f_custo_produtivo=explode(".",$f_custo_produtivo);
$f_custo_produtivo=implode(",",$f_custo_produtivo);
```

Antes de gravar no banco (inclusão e atualização):

```
$f_custo_produtivo=explode(",",$f_custo_produtivo);
```

$f_custo_produtivo=implode(".",$f_custo_produtivo);

Conversão de Strings

```
$foo = 1 + "10.5";echo $foo."<br>";            // $foo é float (11.5)
$foo = 1 + "-1.3e3";echo $foo."<br>";          // $foo é float (-1299)
$foo = 1 + "bob-1.3e3";echo $foo."<br>";       // $foo é integer (1)
$foo = 1 + "bob3";echo $foo."<br>";            // $foo é integer (1)
$foo = 1 + "10 Small Pigs";echo $foo."<br>";   // $foo é integer (11)
$foo = 4 + "10.2 Little Piggies";echo $foo."<br>"; // $foo é float (14.2)
$foo = "10.0 pigs " + 1;echo $foo."<br>";      // $foo é float (11)
$foo = "10.0 pigs " + 1.0;echo $foo."<br>";    // $foo é float (11)
```

Trabalhando com caracteres de strings

```
// Pega o primeiro caracter da string
$str = 'Isto é um teste.';
$first = $str{0};
echo $first."<br>";
// Pega o terceiro caracter da string
$third = $str{2};
echo $third."<br>";
// Pega o último caracter da string
$str = 'Isto ainda é um teste.';
$last = $str{strlen($str)-1};
echo $last."<br>";
// Modifica o ultimo caracter da string
$str = 'Olhe o mal';
echo $str{strlen($str)-1} = 'r';
```

Validação de caracteres

```
ctype_alnum
ctype_alpha
ctype_cntrl
ctype_digit
ctype_graph
ctype_lower
ctype_print
ctype_punct
ctype_space
ctype_upper
ctype_xdigit
```

ctype_alnum - Checa por caracteres alfanuméricos

```
$strings = array('AbCd1zyZ9', 'foo!#$bar');
foreach ($strings as $testcase) {
   if (ctype_alnum($testcase)) {
      echo "The string $testcase consists of all letters or digits.\n";
   } else {
      echo "The string $testcase does not consist of all letters or digits.\n";
   }
}
```

ctype_alpha - Checa por caracteres alfabéticos

```
$strings = array('KjgWZC', 'arf12');
foreach ($strings as $testcase) {
   if (ctype_alpha($testcase)) {
      echo "The string $testcase consists of all letters.\n";
```

```
  } else {
     echo "The string $testcase does not consist of all letters.\n";
  }
}
```

ctype_digit - Checa por caracteres numéricos

```
$strings = array('1820.20', '10002', 'wsl!12');
foreach ($strings as $testcase) {
  if (ctype_digit($testcase)) {
     echo "The string $testcase consists of all digits.\n";
  } else {
     echo "The string $testcase does not consist of all digits.\n";
  }
}
// Alerta: Ao executar veja que somente é válido quando todos são dígitos
// Não é indicado para testar valores decimais, com ponto ou vírgula
```

ctype_lower - Checa por caracteres minúsculos

```
$strings = array('aac123', 'qiutoas', 'QASsdks');
foreach ($strings as $testcase) {
  if (ctype_lower($testcase)) {
     echo "The string $testcase consists of all lowercase letters.\n";
  } else {
     echo "The string $testcase does not consist of all lowercase letters.\n";
  }
}
```

ctype_punct - Checa por Caracteres que não sejam espaço em branco nem alfanuméricos

```
$strings = array('ABasdk!@!$#', '!@ # $', '*&$()');
foreach ($strings as $testcase) {
  if (ctype_punct($testcase)) {
     echo "The string $testcase consists of all punctuation.\n";
  } else {
     echo "The string $testcase does not consist of all punctuation.\n";
  }
}
```

ctype_space - Checa por espaços em branco

Validação de Tipos

```
intval
is_array
is_bool
is_callable
is_double
is_float
is_int
is_integer
is_long
is_null
is_numeric
is_object
is_real
is_resource
is_scalar
```

is_string
isset
print_r
serialize
settype
strval
unserialize
unset

Cases

strtoupper($str) - tudo maiúsculo
strtolower($str) - tudo minúsculo
ucfirst($str) - Converte para maiúscula o primeiro caractere de uma STRING
ucwords($STR) - Converte para maiúsculas o primeiro caractere de cada PALAVRA

6.7 – Trabalhando com Matemática

O PHP tem uma grande quantidade de funções para trabalhar com matemática. Veja a relação

Constantes pré-definidas

- Funções Matemáticas

- abs — Valor absoluto
- acos — Cosseno Inverso (arco cosseno)
- acosh — Cosseno Hiperbólico Inverso
- asin — Seno Inverso (arco seno)
- asinh — Seno Hiperbólico Inverso
- atan2 — Tangente inversa de duas variáveis
- atan — Tangente Inversa (arco tangente)
- atanh — Tangente hiperbólica inversa
- base_convert — Converte um número entre bases arbitrárias
- bindec — Binário para decimal
- ceil — Arredonda frações para cima
- cos — Coseno
- cosh — Cosseno hiperbólico
- decbin — Decimal para binário
- dechex — Decimal para hexadecimal
- decoct — Decimal para octal
- deg2rad — Converte o número em graus ao equivalente em radianos
- exp — Calcula o expoente de e
- expm1 — Retorna exp(numero) - 1, computado de forma que é preciso mesmo quando o valor do número é perto de zero.
- fdiv — Divides two numbers, according to IEEE 754
- floor — Arredonda frações para baixo
- fmod — Retorna o resto em ponto flutuante (módulo) da divisão dos argumentos
- getrandmax — Retorna o maior valor aletório possível
- hexdec — Hexadecimal para decimal
- hypot — Calcula o tamanho da hipotenusa de um ângulo reto do triângulo Retorna a raiz quadrada de (num1*num1 + num2*num2)

- intdiv — Dividir números inteiros
- is_finite — Verifica se um valor é um número finito
- is_infinite — Descrição
- is_nan — Verifica se um valor não é um número
- lcg_value — Gerador congruente linear combinado
- log10 — Logaritmo Base-10
- log1p — Retorna o log(1 + numero), calculado de forma que o valor do número seja próximo de zero.
- log — Logaritmo natural
- max — Localiza o maior valor
- min — Encontra o menor valor
- mt_getrandmax — Retorna o maior valor aletório possível
- mt_rand — Gerador melhorado de números aleatórios
- mt_srand — Semeia o gerador melhorado de números aleatórios
- octdec — Octal para decimal
- pi — Obtém o valor de pi
- pow — Potência
- rad2deg — Converte o número em radianos para o equivalente em graus
- rand — Gera um inteiro aleatório
- round — Arredonda um número
- sin — Seno
- sinh — Seno hiperbólico
- sqrt — Raiz quadrada
- srand — Semeia o gerador de números aletórios
- tan — Tangente
- tanh — Tangente hiperbólica

Constantes Pré-definidas

Constantes da Matemáticas

Constante	Valor	Descrição
M_PI	3.14159265358979323846	Pi
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	log_2 e
M_LOG10E	0.43429448190325182765	log_10 e
M_LN2	0.69314718055994530942	log_e 2
M_LN10	2.30258509299404568402	log_e 10
M_PI_2	1.57079632679489661923	pi/2
M_PI_4	0.78539816339744830962	pi/4
M_1_PI	0.31830988618379067154	1/pi
M_2_PI	0.63661977236758134308	2/pi
M_SQRTPI	1.77245385090551602729	sqrt(pi) [5.2.0]
M_2_SQRTPI	1.12837916709551257390	2/sqrt(pi)
M_SQRT2	1.41421356237309504880	sqrt(2)

Constante	Valor	Descrição
M_SQRT3	1.73205080756887729352	sqrt(3) [5.2.0]
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)
M_LNPI	1.14472988584940017414	log_e(pi) [5.2.0]
M_EULER	0.57721566490153286061	Euler constant [5.2.0]

Somente M_PI está disponível nas versões do PHP anteriores a 4.0.0, inclusive. Todas as outras constantes foram disponibilizadas a partir da 4.0.0. Constantes marcadas com [4.0.2] foram acrescentadas no PHP 5.2.0.

Exemplo:

```
print M_PI;
```

Cuidado com Números em Ponto Flutuante

Teste em PHP

```
<?php
echo (int) ((0.1 + 0.7 ) * 10);
?>
```

Agora teste isso:

```
echo (int) ((0.2 + 0.7 ) * 10);
```

Não conclua muito apressadamente que é deficiência do PHP.

Neste momento devemos ter conhecimento de como se comportam os números, especialmente os floats, que são normalizados pelo IEEE.

Teste em Java

```
class teste {
   public static void main(String[] args) {
      System.out.println((int) ((0.1 + 0.7 ) * 10)); //Display the string.
   }
}
```

Em Java também dá o mesmo resultado do PHP, o que leva a crer que a coisa não depende da linguagem mas das normas de como foram construídos os números pelo IEEE.

O Effective Java sugere que se use int, long ou BigDecimal para representar os valores monetários. A classe BigDecimal foi desenvolvida para resolver dois tipos de problemas associados a números de ponto flutuante (floats e doubles): primeiro, resolve o problema da inexatidão da representação de números decimais; segundo, pode ser usado para trabalhar com números com mais de 16 dígitos significativos. Em compensação, utilizar BigDecimal pode tornar o programa menos legível por não haver sobrecarga dos operadores matemáticos para ela, sendo necessário usar métodos da classe. Veja, por exemplo, como você faria o programa da listagem 1 com BigDecimal:

```
BigDecimal d1 = new BigDecimal("1.95");

BigDecimal d2 = new BigDecimal("1.03");
```

System.out.println(d1.subtract(d2));

Utilizar os primitivos normalmente é mais rápido e mais prático, mas o problema fica por conta da definição das casas decimais. Você pode controlar diretamente as casas decimais, por exemplo, utilizando como unidade para os valores o centavo ao invés de real. Um int ou um long passariam a representar a quantidade de centavos presentes no valor, e não a quatidade de reais. Por exemplo:

long l1 = 195;

long l2 = 103;

System.out.println(l1 ? l2);

As variáveis acima dizem que você tem 195 centavos (e não R$ 1,95) e vai gastar 103 centavos, e não R$ 1,03. No final você ficará com 92 centavos (e não R$ 0,92).

Agora veja as recomendações do manual do PHP

O tamanho de um float depende também da plataforma e é de 64bits no formato IEEE(*). Nunca compare números em ponto flutuante em igualdades, sob pena de cometer erros.

Teste com PostgreSQL

*SELECT CAST((0.1 + 0.7)*10 AS INTEGER);*

Este sim, retorna o valor esperado.

Em Java:

System.out.println(1.95 - 1.03); // Retorna errado e em PHP retorna OK.

Em Ruby

(1.8+0.1)==(1.9) retorna false

O mesmo ocorre em Phyton.

7 – Funções definidas pelo Programador

https://www.php.net/manual/pt_BR/language.functions.php

Funções são a forma mais essencial de organização de código e de lógica na programação. Através de funções você consegue empacotar partes do seu código e separá-las em blocos lógicos auto-contidos. Assim, ao invés de ter um programa de cem linhas com uma lógica linear que segue do começo ao fim e bem mais difícil de identificar o que cada parte desse código faz, você tem menos repetição e um programa onde a lógica está muito bem dividida - e, portanto, muito mais fácil de ser entendida.

(Introdução ao Javascript da Trybe)

Além das funções nativas o PHP também suporta funções criadas pelo usuário/programador.

7.1 - Definição de uma função

```
function nome(){
    // Conteúdo/corpo
}
```

As funções em PHP começam com a palavra reservada "function", seguida do nome da função, que é seguido de (), depois abre chaves {, Abaixo tem o código da função/corpo, que será processado por ela e finalmente o fecha chaves }.

Exemplo

```
function olaMundo(){
    print 'Hello World';
}
```

7.2 - Chamando ou executando a função

olaMundo(); // Seu nome com () em qualquer parte onde queremos chamá-la

7.3 - Tipos de funções definidas pelo Usuário

- Simples, sem parâmetro nem retorno

- Com passagem de parâmetros

- Parâmetro com valor padrão

- Tipo de parâmetro

- Retornando valores

- Tipo de retorno

- Funções anônimas

Obs.: as funções que não tem retorno (return) apenas executam o código em seu conteúdo.

7.4 - Com passagem de parâmetros

```
function primeiraFuncao($nome) {
  echo "Olá " .  $nome;
}
primeiraFuncao("Anderson");
```

Neste caso a função passa o parâmetro $nome para o corpo da função.

7.5 - Parâmetro com valor padrão

Para que um parâmetro tenha um valor padrão, já passamos um valor para ele.

```
function primeiraFuncao($nome = "Ribamar FS") {
  echo "Olá " .  $nome;
}
// Caso chamemos a função sem passar parâmetro o default será passado
primeiraFuncao(); // Olá Ribamar FS
// Mas também podemos passar parâmetros, que sobrescreverão o default
primeiraFuncao("Anderson");
```

7.6 - Tipo de parâmetro

```
function primeiraFuncao(string $nome) {
  echo "Olá " .  $nome;
}
primeiraFuncao("Anderson");
```

7.7 - Retornando valores

Quando existe a palavra reservada return na função ela é seguida de um valor ou de uma variável, que se encarrega de retornar um valor da função. Ao executar a função ela retorna este valor.

```
function primeiraFuncao(string $nome) {
  return "Olá " .  $nome;
}
echo primeiraFuncao("Anderson");
```

Este tipo de função não processa seu conteúdo apenas. Ele processa e entrega um valor para o return. Veja que nesta função não usamos o echo no corpo e precisamos usar na chamada da função.

7.8 - Tipo de retorno

Agora teremos um tipo para o parâmetro e um para o retorno:

```
function primeiraFuncao(string $nome) : string{
  return "Olá " .  $nome;
}
$result = primeiraFuncao("Anderson");
echo $result;
```

7.9 - Funções anônimas

A partir do **PHP 7** também podemos utilizar funções anônimas. Este tipo de função não recebe um nome em sua declaração. Com isso podemos definir seu valor direto para uma variável, ou mesmo utilizar no PHP um conceito comum do **JavaScript** por exemplo, que são as funções de **callback**.

```
$result = function (string $nome) : string {
  return "Olá " .  $nome;
};
echo $result("Anderson");
```

Outro exemplo

```
function primeiraFuncao($nome, $function) {
  echo $function($nome);
}

primeiraFuncao("Anderson", function(string $nome) : string {
  return "Olá " .  $nome;
});
```

https://irias.com.br/blog/php-functions-tutorial-basico-par-funcoes-no-php/

7.10 – Escopo de Variáveis

Escopo de um variável são as áreas onde ela pode ser "vista".

O escopo de uma variável é o contexto onde foi definida. A maioria das variáveis do PHP tem somente escopo local. Este escopo local inclui os arquivos incluídos e requeridos.
Qualquer variável utilizada dentro de uma função é, por padrão, limitada ao escopo local da função.

Exemplo:

```
$nome = "Antônio";
function testeEscopo(){
        print $nome;
}
testeEscopo();
```

Ao executar o código acima o PHP acusará que a variável $nome não foi definida. Acontece que se definimos a variável $nome fora da função e assim a função não tem acesso a ela.
Uma forma de resolver isso é declarar a variável dentro da função com global:

```
$nome = "Antônio";
function testeEscopo(){
        global $nome;
        print $nome;
}
testeEscopo();
```

Mas esta não é uma forma recomendada. Melhor é passara como parâmetro

```
$nome = "Antônio";
function testeEscopo($nome){
        print $nome;
}
```

```
testeEscopo($nome);
```

https://www.php.net/manual/pt_BR/language.variables.scope.php

7.11 – Boas práticas

O Básico

Operadores de Comparação

Operadores de Comparação são frequentemente negligenciados em PHP, o que pode levar a muitos resultados inesperados. Um desses problemas decorre de comparações estritas (comparações entre booleanos e inteiros).

```
<?php
$a = 5;   // 5 como inteiro

var_dump($a == 5);       // comparação de valores; retorna true
var_dump($a == '5');     // comparação de valores (ignorando os tipos); retorna true
var_dump($a === 5);      // comparação de tipos e valores (integer vs. integer); retorna true
var_dump($a === '5');    // comparação de tipos e valores (integer vs. string); retorna false

/**
 * Comparações Estritas
 */
if (strpos('testing', 'test')) {    // 'test' é encontrado na posição 0, que é interpretado como o booleano 'false'
    // código...
}

vs.

if (strpos('testing', 'test') !== false) {    // true, já que uma comparação estrita foi feita (0 !== false)
    // código...
}
```

Operadores de Comparação
Tabela de Comparação

Estrutura de Controle
Estruturas Condicionais

Quando as declarações ‘if/else’ são usadas em uma função ou classe, é um equívoco comum pensar que ‘else’ precisa ser usado em conjunto para declarar resultados em potencial. Entretanto se o resultado serve para definir o valor a ser retornado ‘else’ não é necessário já que ‘return’ irá terminar a função, fazendo com que o uso de ‘else’ se torne discutível.

```
<?php
function test($a)
{
    if ($a) {
        return true;
    } else {
        return false;
    }
}

// vs
```

```
function test($a)
{
   if ($a) {
      return true;
   }
   return false;    // else não é necessário
}
```

Estrutura Condicionais

Estruturas de Decisão

Estruturas de decisão são uma excelente forma de evitar escrever intermináveis estruturas condicionais, mas existem alguns pontos sobre os quais deve-se ficar atento:

Estruturas de decisão só comparam valores, e não tipos (equivalente a '==')
Elas passam por caso a caso até que uma correspondencia seja encontrada, então default é usado (caso esteja definido)
Sem que haja um 'break', elas continuarão a executar cada caso até que encontrem um break/return
Dentro de uma função o uso de 'return' remove a necessidade do uso de 'break' já que isso encerra essa função

```
<?php
$answer = test(2);    // tanto o código para o 'case 2' quanto para o 'case 3' será executado

function test($a)
{
   switch ($a) {
      case 1:
         // código...
         break;            // break é usado para terminar a estrutura de decisão
      case 2:
         // código...       // sem o break, a comparação ira continuar em 'case 3'
      case 3:
         // código...
         return $result;    // dentro de uma função, 'return' termina essa função
      default:
         // código...
         return $error;
   }
}
```

Estruturas de Decisão
PHP Switch

Namespace Global

Quando estiver usando namespaces você pode reparar que funções internas ficam escondidas por funções que você mesmo escreveu. Para corrigir isso refira a funções globais através do uso de uma contra-barra antes do nome da função.

```
<?php
namespace phptherightway;

function fopen()
{
   $file = \fopen();    // O nome da nossa função é igual a de uma função interna.
```

```
                // Execute a função global através da inclusão de '\'.
}

function array()
{
    $iterator = new \ArrayIterator();    // ArrayIterator é uma classe interna. Usar seu nome sem uma contra-barra
                                         // tentará localizar essa função dentro do namespace
}
```

Espaço Global
Regras Globais

Strings
Concatenação

Se sua linha passar do tamanho recomendado (120 caracteres), considere concatenar sua linha
Para facilitar a leitura é melhor usar operadores de concatenação do que operadores de concatenação e atribuição
Enquanto dentro do escopo original da variável, indente quando a concatenação usar uma nova linha

```
<?php
$a  = 'Multi-line example';    // operador de concatenação e atribuição (.=)
$a .= "\n";
$a .= 'of what not to do';

vs.

$a = 'Multi-line example'      // operador de concatenação (.)
   . "\n"                      // indentando novas linhas
   . 'of what to do';
```

Operadores de Strings

Tipos de Strings

Tipos de string são uma característica constante na comunidade PHP, mas talvez essa seção possa explicar as diferenças entre os tipos de strings e seus usos e benefícios.
Aspas Simples

As aspas simples são utilizadas para indicar uma ‘string literal’. Strings literais não tentam analisar caracteres especiais ou variáveis.

Se estiver usando aspas simples, você pode digitar um nome de variável em uma string assim: 'some $thing' e você verá a saída exata some $thing. Se você estiver usando aspas duplas, o motor da linguagem tentará avaliar a variável “$thing” e então exibirá erros se nenhuma variável for encontrada.

```
<?php
echo 'This is my string, look at how pretty it is.';    // sem necessidade de interpretar uma string simples

/**
 * Saída:
 *
 * This is my string, look at how pretty it is.
 */
```

Aspas Simples

Aspas Duplas

Aspas duplas são o canivete suíço das strings, mas são mais lentas devido a interpretação das strings. Ele não só irá analisar as variáveis como mencionado acima mas também todos os tipos de caracteres especiais, como \n para nova linha, \t para identação, etc.

```
<?php
echo 'phptherightway é ' . $adjective . '.'      // Um exemplo com aspas simples que usa concatenação múltipla para
   . "\n"                                        // variáveis e escapar strings
   . 'I love learning' . $code . '!';

// vs

echo "phptherightway is $adjective.\n I love learning $code!"  // Em vez de concatenação múltipla, aspas duplas
                                                               // nos permitem utilizar strings interpretáveis
```

Aspas duplas que contém variáveis; Isto é chamado “interpolação”.

```
<?php
$juice = 'plum';
echo "I like $juice juice";    // Output: I like plum juice
```

Quando usando interpolação, são comuns os casos onde a variável pode estar colada com outro caracter. Isso fará com que o PHP não consiga interpretar essa variável pelo fato dela estar sendo camuflada.

Para corrigir esse problema envolva a variável em um par de chaves.

```
<?php
$juice = 'plum';
echo "I drank some juice made of $juices";    // $juice cannot be parsed

// vs

$juice = 'plum';
echo "I drank some juice made of {$juice}s";    // $juice will be parsed

/**
 * Variáveis complexas também serão interpretadas com o uso de chaves
 */

$juice = array('apple', 'orange', 'plum');
echo "I drank some juice made of {$juice[1]}s";   // $juice[1] will be parsed
```

Aspas Duplas

Sintaxe Nowdoc

A Sintaxe Nowdoc foi introduzida no PHP 5.3 e internamente se comporta da mesma forma que as aspas simples exceto que é adequada para o uso de strings de múltiplas linhas sem a necessidade de concatenação.

```
<?php
$str = <<<'EOD'          // iniciada por <<<
```

```
Example of string
spanning multiple lines
using nowdoc syntax.
$a does not parse.
EOD;                    // fechando 'EOD' precisa estar na sua própria linha, e no ponto mais a esquerda

/**
 * Output:
 *
 * Example of string
 * spanning multiple lines
 * using nowdoc syntax.
 * $a does not parse.
 */
```

Sintaxe Nowdoc

Sintaxe Heredoc

A Sintaxe Heredoc se comporta internamente da mesma forma que as aspas duplas exceto que é adequada para o uso de strings de múltiplas linhas sem a necessidade de concatenação.

```
<?php
$a = 'Variables';

$str = <<<EOD           // iniciada por <<<
Example of string
spanning multiple lines
using heredoc syntax.
$a are parsed.
EOD;                    // fechando 'EOD' precisa estar na sua própria linha, e no ponto mais a esquerda

/**
 * Output:
 *
 * Example of string
 * spanning multiple lines
 * using heredoc syntax.
 * Variables are parsed.
 */
```

Sintaxe Heredoc

O que é mais rápido?

Há um mito por aí que usar aspas simples em strings são interpretadas mais rápida do que usar aspas duplas. Isso não é fundamentalmente falso.

Se você estiver definindo uma string única e não concatenar valores ou qualquer coisa complicada, então aspas simples ou duplas serão idênticas. Não será mais rápido.

Se você está concatenando várias strings de qualquer tipo, ou interpolar valores em uma string entre aspas duplas, então os resultados podem variar. Se você estiver trabalhando com um pequeno número de valores, a concatenação é minuciosamente mais rápida. Com um monte de valores, interpolação é minuciosamente mais rápida.

Independentemente do que você está fazendo com strings, nenhum dos tipos vai ter qualquer impacto perceptível sobre a sua aplicação. Tentar reescrever código para usar um ou o outro é

sempre um exercício de futilidade, de modo a evitar este micro-otimização, a menos que você realmente compreenda o significado e o impacto das diferenças.
Operadores Ternários

O uso de operadores ternários é uma ótima forma de condensar seu código, mas eles são geralmente usados em excesso. Apesar de operações ternárias poderem ser agrupadas e aconselhado usar uma por linha para aumentar a legibilidade.

```
<?php
$a = 5;
echo ($a == 5) ? 'yay' : 'nay';

// vs

$b = 10;
echo ($a) ? ($a == 5) ? 'yay' : 'nay' : ($b == 10) ? 'excessive' : ':(';    // excesso de agrupamento sacrifica a legibilidade
```

Para usar ‘return’ em um operador ternário utilize a sintaxe correta.

```
<?php
$a = 5;
echo ($a == 5) ? return true : return false;    // esse exemplo irá disparar um erro

// vs

$a = 5;
return ($a == 5) ? 'yay' : 'nope';    // esse exemplo irá retornar 'yay'
```

Note que você não precisa usar um operador ternário para retornar um valor booleano. Um exemplo disto seria.

```
<?php
$a = 3;
return ($a == 3) ? true : false; // esse exemplo irá retornar true ou false se $a == 3

// vs

$a = 3;
return $a == 3; // esse exemplo irá retornar true ou false se $a == 3
```

Isso também pode ser dito para as operações (===, !==, !=, == etc).
Utilizando parênteses com operadores ternários para formato e função

Quando se utiliza um operador ternário, os parênteses podem melhorar a legibilidade do código e também incluir as uniões dentro de blocos de instruções. Um exemplo de quando não há nenhuma exigência para usar de parênteses é:

```
<?php
$a = 3;
return ($a == 3) ? "yay" : "nope"; // vai retornar yay ou nope se $a == 3

// vs

$a = 3;
return $a == 3 ? "yay" : "nope"; // vai retornar yay ou nope se $a == 3
```

O uso de parênteses também nos dá a capacidade de criar união dentro de um bloco de declaração onde o bloco será verificado como um todo. Tal como este exemplo abaixo que retornará verdadeiro se ambos ($a == 3 e $b == 4) são verdadeiras e $c == 5 também é verdadeiro.

```
<?php
return ($a == 3 && $b == 4) && $c == 5;
```

Outro exemplo é o trecho de código abaixo que vai returnar true se ($a != 3 e $b != 4) ou $c == 5.

```
<?php
return ($a != 3 && $b != 4) || $c == 5;
```

Operadores Ternários

Declaração de Variáveis

As vezes programadores tentam tornar seu código mais limpo declarando variáveis predefinidas com um nome diferente. O que isso faz na realidade é dobrar o consumo de memória do script. No exemplo abaixo, digamos que uma string de exemplo contém 1MB de dado válido, copiando a variável você aumenta o consumo de memória durante a execução para 2MB.

```
<?php
$about = 'Uma string com texto bem longo';    // usa 2MB de memória
echo $about;

// vs

echo 'Uma string com texto bem longo';        // usa 1MB de memória
```

Dicas de Performance

http://br.phptherightway.com/pages/The-Basics.html

8 - Tratamento de erros

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:

```
<?php
error_reporting(E_ERROR | E_WARNING);
```

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'] : '';

Você também pode definir esse valor em tempo de execução com a função ini_set
<?php
ini_set('xdebug.scream', '1')

8.1 – Xdebug

O Xdebug é um dos componentes de tratamento de erros para PHP mais importantes.

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 sem conseguir sair do lugar, querendo recorrer a var_dump/print_r, e ainda assim não conseguir 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.

8.1.1 - Instalação no Linux

sudo apt install php-xdebug

8.1.2 - Configuração do Xdebug para PHP 8

sudo nano /etc/php/8.0/mods-available/xdebug.ini

zend_extension=xdebug.so
xdebug.remote_enable = 1
xdebug.remote_autostart = 1
xdebug.remote_port=9000
xdebug.remote_connect_back=true
xdebug.remote_host=10.0.2.2 #este valor pode mudar, mas é o padrão do Vagrant, corresponde ao IP da máquina host

Geralmente as informações do Xdebug são bem claras e logo percebemos o erro. Mas acontece algumas vezes de isso não acontecer assim tão claramente. Nestes casos precisamos ler com atenção e calma as informações mostradas pelo Xdebug, que nos ajudarão a solucionar o problema.

Nem sempre a solução está apontada na faixa em laranja. Ele pode vir indicada logo abaixo.

8.2 – Try/catch

O PHP possui um modelo de exceções similar ao de outras linguagens de programação. Uma exceção pode ser lançada (`throw`) e capturada (`catch`). Código pode ser envolvido por um bloco `try` para facilitar a captura de exceções potenciais. Cada bloco `try` precisa ter ao menos um bloco `catch` ou `finally` correspondente.

Se uma exceção é lançada e o escopo atual não possui nenhum block `catch`, a exceção irá "desempilhar" o call stack pelas funções chamadoras até encontrar um bloco `catch` compatível. Todos os blocks `finally` encontrados pelo caminho serão executados. Se a pilha é esvaziada até o

final, sem encontrar um block `catch` compatível, o programa irá terminar um erro fatal a não ser que um manipulador global de exceções tenha sido configurado.

O objeto lançado precisa ser uma instância da classe Exception ou uma subclasse de Exception. Tentar lançar um objeto sem essa ascendência resultará em um erro fatal.

A partir doPHP 8.0.0, a palavra chave `throw` é uma expressão e pode ser utilizada em qualquer contexto de expressão. Em versões anteriores era considerado uma instrução e portanto precisava constar em sua própria linha.

8.2.1 - catch

Um bloco `catch` define como o código responde a uma exceção lançada. Um bloco `catch` define um ou mais tipos de exceções ou erros que ele pode processar, e opcionalmente uma variável para receber a exceção lançada. (A variável era requerida anteriormente ao PHP 8.0.0.) O primeiro bloco `catch` que uma exceção ou erro lançados encontram que seja compatível com o tipo objeto lançado irá processar esse objeto.

Múltiplos blocos `catch` podem ser utilizados para capturar exceções diferentes. A execução normal (quando nenhuma exceção é lançada dentro de um `try`) irão continuar a execução após o último `catch` definido em sequência. Exceções podem ser lançadas (ou relançadas) dentro um bloco `catch`. Caso contrário, a execução irá continuar após o bloco`catch` que foi acionado.

Quando uma exceção é lançada o código seguinte não é executado, e o PHP tentará encontrar o primeiro bloco `catch` coincidente. Se uma exceção não for capturada, um erro PHP fatal será lançado com a mensagem `"Uncaught Exception ..."` na ausência de uma função definida com set_exception_handler().

A partir do PHP 7.1 um bloco `catch` pode especificar múltiplas exceções usando o caractere pipe (`|`). Isto é útil quando diferentes exceções de diferentes hierarquias de classes são tratadas da mesma forma.

A partir do PHP 8.0.0, o nome de variável que captura a exceção é opcional. Se não especificada, o bloco `catch` compatível ainda executará, mas não terá acesso ao objeto lançado.

8.2.2 - finally

Um bloco `finally` pode ser especificado após ou no lugar de blocos `catch`. Códigos dentro de `finally` sempre serão executados depois do `try` ou `catch`, independente se houve o lançamento de uma exceção, e antes que a execução normal continue.

Uma interação notável ocorre entre um bloco `finally` e a instrução `return`. Se uma instrução `return` é encontrada dentro um bloco `try` ou `catch`, o bloco `finally` ainda assim será executado. Além disso a instrução `return` é avaliada no ponto que é encontrada, mas o resultado só será retornado após o bloco `finally` ser executado. Se o bloco `finally` também tiver uma instrução `return`, o valor da instrução de `finally` que será retornado.

Exemplos

Conexão

```
class Connection
{
        private $host = 'localhost';
        private $db   = 'estoque';
        private $user = 'root';
        private $pass = 'root';
        public  $pdo;
        private $port = 3306;

   // Conexão com o banco de dados
   public function __construct(){
        try {
                $dsn = 'mysql:host='.$this->host.';dbname='.$this->db.';port='.$this->port;
                $this->pdo = new PDO($dsn, $this->user, $this->pass);
                // Boa exibição de erros
                $this->pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES,false);
                $this->pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

                $this->pdo->query('SET NAMES utf8');
                return $this->pdo;
        }catch(PDOException $e){
      // Usar estas linhas no catch apenas em ambiente de testes/desenvolvimento. Em produção apenas o exit()
                echo '<br><br><b>Código</b>: '.$e->getCode().'<hr><br>';
                echo '<b>Mensagem</b>: '. $e->getMessage().'<br>';
                echo '<b>Arquivo</b>: '.$e->getFile().'<br>';
                echo '<b>Linha</b>: '.$e->getLine().'<br>';
                exit();
        }
   }
}
```

Ver
https://github.com/ribafs/phpecia/blob/main/PDO/Conexoes/connection2.php
Outros exemplos:
https://github.com/ribafs/phpecia/tree/main/PDO/Conexoes

É importante usar try/catch nas demais operações com o banco de dados.

```
   try {
      $insertStr = $crud->insertStr();

      $sql = "INSERT INTO $table $insertStr";
      $sth = $crud->pdo->prepare($sql);

      for($x=1;$x<$numFields;$x++){
         $field = $crud->fieldName($x);
                  $sth->bindParam(":$field", $_POST["$field"], PDO::PARAM_INT);
            }
      $sth->execute();

       print "<script>location='./index.php?table=$table';</script>";
   } catch (Exception $e) {
      if($e->getCode() == '22007'){
         print '<h3>Data vazia ou inválida</h3>';
      }
                  echo '<br><br><b>Código</b>: '.$e->getCode().'<hr><br>';
                  echo '<b>Mensagem</b>: '. $e->getMessage().'<br>';// Usar estas linhas no catch apenas em
ambiente de testes/desenvolvimento
                  echo '<b>Arquivo</b>: '.$e->getFile().'<br>';
                  echo '<b>Linha</b>: '.$e->getLine().'<br>';
```

```
        exit();
}
```

8.3 - IDE/Editor de código

Uma boa IDE ou um bom editor de programação ajudam bastante com o debug.
Quando começamos a digitar uma linha que contém um erro a IDE já nos alerta.
Em um editor simples somente veremos o erro quando chamarmos/executarmos o arquivo pelo navegador.
Especialmente para o iniciante uma IDE ou um bom editor de programação ajudam muito.

Meus preferidos são: Eclipse para PHP, Netbeans para PHP e VSCode.

8.4 – Outras ferramentas

- phpcs
- error_reporting()

composer require tracy/tracy

composer require filp/whoops

8.5 – Dicas sobre erros

Quando acontecer de efetuar diversas alterações e nada surtir o efeito esperado, existe uma grande chance de não estar mexendo no arquivo esperado. Verifique.

Strings

Cuidado ao trabalhar com strings. Acontece muito de ficarem espaços em branco difíceis de serem identificados.
Uma boa providência é usar a função trim() no resultado para eliminar os espaços em branco da esquerda e da direita da string resultante.

Quando usando um tipo OBJ $row->campo e receber

Attempt to read property "id" on array in /

Mude para ASSOC

$row["$campo"]

Nem sempre a mensagem do erro indica a linha corretamente. Algumas vezes o erro está na linha antereior.

Uma boa ideia é usar o navegador no modo private, para evitar cache em algumas situações.
No Firefox use Ctrl+Shift+P.

O erro

Too few arguments to function Crud::updateReg(), 0 passed in

Indica que o método Crud::updateReg() requer argumentos e não foi passado nenhum.

Erros sem Solução

Quando nos depararmos com error que não encontramos a solução, uma boa providência é limpar o cache do navegador. Como limpar o cache afeta algumas atividades do navegador, então podemos usar alternativas, como abrir um outro navegador qeu não tenha o site em cache ou também abrir no mesmo navegador em modo private.
No firefox podemos usar Ctrl+Shift+P

Muitas vezes as mensagens de erro do PHP se enganam. Veja esta:
Parse error: syntax error, unexpected token ")", expecting variable in

O problema era este:

function sqlInsert($pdo,table){

A v ariaúel table acidentalmente estava sem o $.

Cuidado com as mensagens.

Muito cuidado com = em ifs. Um if requer ==, >, <, etc e nunca =
O sinal de = significa atribuição e usado em um if vai indicar sempre verdadeira, pois sempre atribuirá.

if($table = 'unidades'){

Esta expressão acima sempre será verdadeira.

Cuidado ao lidar com funções que tenham dois ou mais argumentos. Os argumentos devem ser fornecidos rigorosamente na ordem que foram definidos na função.

Cuidado ao trabalhar com parâmetros em URL

index.php?id=<?=$row['id']?>&action=update

Acima eu estou passando uma variável action com o valor 'update'. Não use action='update', como seria de se esperar.

Para mostrar error em testes/desenvolvimento, adicionar no início do arquivo, idealmente um index.php:

```
error_reporting(E_ALL);
ini_set('display_errors', true);
```

Mudar parâmetros do php.ini:

```
ini_set('memory_limit', '5060M');
ini_set('max_execution_time', 3600);
ini_set("date.timezone", "America/Fortaleza");
```

Identificar ambientes

ini_set() e error_reporting().

```
// Ambiente de Testes
if(DEBUG) {
  error_reporting(E_ALL);
  ini_set('display_errors', 1);
} else {
// Ambiente de Produção
  error_reporting(0);
  ini_set('display_errors', 0);
  ini_set('log_errors', 1);
  ini_set('error_log', ROOT . DS .'tmp' . DS . 'logs' . DS . 'errors.log');
}
```

9 – Formulários

9.1 - No frontend com HTML 5

Formulários são a principal forma usada para colher informações dos usuários.

Formulários HTML são um dos principais pontos de interação entre um usuário e um web site ou aplicativo. Eles permitem que os usuários enviem dados para o web site. Na maior parte do tempo, os dados são enviados para o servidor da web, mas a página da web também pode interceptar para usá-los por conta própria.
Um formulário HTML é feito de um ou mais widgets. Esses widgets podem ser campos de texto (linha única ou de várias linhas), caixas de seleção, botões, checkboxes ou radio buttons. A maior parte do tempo, estes elementos são emparelhados com uma legenda que descreve o seu objetivo.

https://developer.mozilla.org/pt-BR/docs/Learn/Forms/Your_first_form

Exemplo de Formulário em HTML 5

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="utf-8"/>
    <title>Formulário de Demonstração HTML5</title>
  </head>
<body>

 <h1 align="center">Formulário de Demonstração HTML5</h1>

<form method="POST" action="cadastro.php">
<table>
<!-- Para inputs tipo text existe autocomplete="off"> – "on" por padrão, como medida de segurança. -->
<tr><td>Nome</td><td><input type="text" name="nome" autocomplete="on" size="30" maxlength="50"
placeholder="Digite o nome" required></td></tr>
<tr><td>E-mail</td><td><input type="email" name="email" size="30" maxlength="30"></td></tr>
<tr><td>Data</td><td><input type="date" required min="1938-01-01" max="2022-12-31"></td></tr>
<tr><td>Data e Hora</td><td><input type="datetime-local" placeholder="yyyy-mm-ddT19:30" min="2018-06-
07T00:00" max="2022-06-14T00:00"></td></tr>
<tr><td>Mês</td><td><input type="month" placeholder="yyyy-mm"></td></tr>
<tr><td>Semana</td><td><input type="week" min="2018-W27" max="2018-W35" required placeholder="yyyy-
Wxx"></td></tr>
<tr><td>Hora</td><td><input type="time" placeholder="hh:mm"></td></tr>
<tr><td>URL</td><td><input type="url" placeholder="http://xxxx.yy"></td></tr>
<tr><td>Cores</td><td><input type="color"></td></tr>
<tr><td>Telefone</td><td><input type="tel" pattern="^\d{2}-\d{4}-\d{4}$"
placeholder="xx-xxxx-xxxx"></td></tr>
<tr><td>Celular</td><td><input type="tel" pattern="^\d{2}-\d{5}-\d{4}$"
placeholder="xx-xxxxx-xxxx"></td></tr>
<tr><td>Faixa</td><td><input type="range"></td></tr>
<tr><td>Número</td><td><input type="number" min="10" max="15" title="min-10 max-15"></td></tr>
<tr><td>SVG</td><td>
<svg width="400" height="100">
  <rect width="400" height="100" style="fill:rgb(0,0,255);stroke-width:10;stroke:rgb(0,0,0)" />
  Seu navegador não suporta SVG tinline.
</svg>
</td></tr>
<tr><td>Canvas</td><td>
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
```

```
Your browser does not support the HTML5 canvas tag.</canvas>
</td></tr><!-- /outros exemplos: https://www.w3schools.com/html/html5_canvas.asp -->
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");

// Create gradient
var grd = ctx.createRadialGradient(75,50,5,90,60,100);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");

// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
</script>
<tr><td>Áudio</td><td>
 <audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="alarm.mp3" type="audio/mpeg">
  Seu navegador não suporta a tag audio
</audio> <!-- https://www.w3schools.com/tags/tag_audio.asp -->
</td></tr>
<tr><td>Vovó no Youtube</td><td>
<a href="https://www.youtube.com/watch?v=szYR9Td7oMQ">Vovó</a>
</td></tr>

<tr><td>Vídeo</td><td>
 <video width="620" height="440" controls>
  <source src="video.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  Seu navegador não suporta a tag video
</video> <!-- https://www.w3schools.com/tags/tag_video.asp -->
</td></tr>
<tr><td></td><td><input type="submit" name="enviar" value="Enviar"></td></tr>
</table>
</form>

<style>
:invalid {
  border: 2px solid #ff0000;
}
</style>
```

Valor default em textarea em form

```
<textarea name="comments" id="comments"><?php
  if ($missing || $errors) {
      echo htmlentities($comments);
  }?></textarea>
```

9.2 - Recebendo dados via URL

Se temos a seguinte situação:

Temos um arquivo c:\www\teste.php, com o seguinte código:

```
<?php
   echo $_GET['codigo'];
?>
```

Queremos passar uma variável código com valor 5, fazemos:

http://127.0.0.1/teste.php?codigo=5

Para receber mais de uma variável ou campo usar:
http://127.0.0.1/teste.php?codigo=5&nome="Antônio"&email="ribafs@yahoo.com"

echo "Código = ".$_GET['codigo']." Nome = ".$_GET['nome']." E-mail = ".$_GET['email'];

Também podemos passar variáveis pela URL, assim teste.php?codigo=$cod

Recebendo POST em arquivo PHP

Recebendo POST em arquivo php

```
<?php
foreach ($_POST as $key => $value) {
   // strip whitespace from $value if not an array
   if (!is_array($value)) {
      $value = trim($value);
   }
   if (!in_array($key, $expected)) {
      // ignore the value, it's not in $expected
      continue;
   }
   if (in_array($key, $required) && empty($value)) {
      // required value is missing
      $missing[] = $key;
      $$key = "";
      continue;
   }
   $$key = $value;
}
```

Exemplo

```
<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name" value="<?php echo $name;?>">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email" value="<?php echo $email;?>">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
```

```
  Website: <input type="text" name="website" value="<?php echo $website;?>">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"><?php echo $comment;?></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" <?php if (isset($gender) && $gender=="female") echo "checked";?>
value="female">Female
  <input type="radio" name="gender" <?php if (isset($gender) && $gender=="male") echo "checked";?>
value="male">Male
  <input type="radio" name="gender" <?php if (isset($gender) && $gender=="other") echo "checked";?>
value="other">Other
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

Mensagem de erro customizada com HTML 5

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
     <meta charset="utf-8"/>
     <title>Mensagem de Erro Customizada</title>
  </head>
<body>

 <h1 align="center">Mensagem de Erro Customizada</h1>

<form>
  <label for="mail">E-mail</label>
  <input type="email" id="mail" name="mail" required>
  <button>Submit</button>
</form>

<script>
var email = document.getElementById("mail");

email.addEventListener("input", function (event) {
  if (email.validity.typeMismatch) {
    email.setCustomValidity("Favor digitar um formato de e-mail válido aqui!");
  } else {
    email.setCustomValidity("");
  }
});
</script>
```

```
</body>
</html>
```

Valor default em Select

Criando combos (select's) dinâmicos com valores padrões

Uma dúvida muito comum – porém de solução bastante simples – é como criar um combo (select) dinâmico e com um valor padrão. Isso pode ser útil ao montar um formulário para edição de dados. Suponha que você possui um sistema em que o usuário escolhe uma dentre diversas opções, por meio de um combo. Para criar um formulário de edição, você deve exibir o combo, mas mostrando a seleção atual.

Para exemplificar, vou usar um simples array com alguns estados brasileiros e uma variável que armazenará o dado que deve aparecer selecionado.

A exibição do combo, juntamente com o dado selecionado, será realizada por uma função, que percorre o array de dados e compara se o registro corrente é igual ao que deve ser selecionado. Se a comparação retornar TRUE, insere-se o atributo "selected" na tag "option".

```
$estados = array( 'PR', 'SP', 'RJ', 'SC', 'RS' );
$padrao = 'PR';

echo MontaSelect( $estados, $padrao );


function MontaSelect( $dados, $selected = NULL )
{
	$str = "<select name=\"uf\" id=\"uf\">\n";
	$total= count( $dados );
	for ( $i = 0; $i < $total; $i++ )
	{
		$str .= "<option value=\"" . $dados[$i] . "\"" . ( ( $dados[$i] == $selected ) ? " selected=\"true\"" :
"" ) . ">" . $dados[$i] . "</option>\n";
	}
	$str .= "</select>\n";
	return $str;
}
```

A solução é simples, apesar de ser fonte de dúvida frequente em fóruns pela Internet.

Lembrando que é sempre bom efetuar verificação nos tipos de parâmetros da função. Não inseri isso no código, mas é recomendável usar is_array() para verificar a variável $dados, por exemplo, a fim de evitar geração de erros e warnings.

9.3 - Exemplos de forms com os vários tipos de campos

```
<h3>Exemplo de uso de formulário com PHP chamando outro script</h3>

<body onLoad="document.form1.nome.focus()">
<form name="form1" method="POST" action="forms0.php">
Nome.<input type="text" name="nome"><br>
E-mail<input type="text" name="email"><br>
<input type="hidden" name="oculto" value="OK">
Senha<input type="password" name="senha"><br>
SenhaCript<input type="password" name="senhacript"><br>
```

```
<input type="submit" value="Enviar"><br>
</form>
```

forms0.php

```
<?php
$nome=$_POST["nome"];
$email=$_POST["email"];
$oculto = $_POST['oculto'];
$senha = $_POST['senha'];
$senhacript = $_POST['senhacript'];

// É importante usar o isset com uma variável ou com $_POST["nome"] para ter certeza de que vem do form
if (isset($nome)){
        echo "O nome digitado foi " . $nome . "<br>";
        echo "O e-mail digitado foi " . $email . "<br>";
        echo "O campo oculto contém " . $oculto . "<br>";
        echo "A senha digitada foi " . $senha . "<br>";
        echo "A senha md5 digitada foi " . md5($senhacript) . "<br>";
}
?>
```

form.html

```
<form action="form.php" method="post">
Campo 1: <input type="text" name="campo1"><br>
Campo 2: <input type="text" name="campo2"><br>
<input type=submit value="OK">
</form>
```

form.php

```
<?php
echo "O valor do CAMPO 1 é: " . $_POST["campo1"];
echo "<br>O valor de CAMPO 2 é: " . $_POST["campo2"];
?>
```

Hidden

form.html

```
<form action="hidden.php" method="post">
<input type=hidden name=escondido value="valor do escondido">
<input type=hidden name=id value="111">
<input type=submit>
</form>
```

hidden.php

```
<?php
echo "Campo Hidden: " . $_POST["escondido"];
echo "<br>Oi, seu ID é: " . $_POST["id"];
?>
```

Text e Textarea

texts.html

```
<form action="texts.php" method="post">
Nome: <input type=text name=nome><br>
Email: <input type=text name=email><br><br>
Mensagem: <textarea name=mensagem cols=8 rows=3></textarea><br>
<input type=submit>
```

```
</form>
```

texts.php

```
<?php
echo "Olá " . $_POST["nome"] . " (email: " . $_POST["email"] . ")<br><br>";
echo "Sua mensagem: " . $_POST["mensagem"];
?>
```

Radio

radio.html

```
<form action="radio.php" method="post">
<B>Qual seu sistema operacional?</B><br>
<input type=radio name=sistema value="Windows 98"> Win 98
<input type=radio name=sistema value="Windows XP"> Win XP
<input type=radio name=sistema value="Linux"> Linux
<input type=radio name=sistema value="Mac"> Mac
<br><br>
<B>Qual a marca de seu monitor?</B><br>
<input type=radio name=monitor value="Samsung"> Samsung
<input type=radio name=monitor value="LG"> LG
<input type=radio name=monitor value="Desconhecido"> Desconhecido
<br><br>
<input type=submit>
</form>
```

radio.php

```
<?php
echo "Seu sistema operacional é: " . $_POST["sistema"];
echo "<br>Seu monitor é: " . $_POST["monitor"];
?>
```

Checkbox

checkbox.html

```
<form action="checkbox.php" method="post">
<B>Escolha os numeros de sua preferência:</B><br>
<input type=checkbox name="numeros[]" value=10> 10<br>
<input type=checkbox name="numeros[]" value=100> 100<br>
<input type=checkbox name="numeros[]" value=1000> 1000<br>
<input type=checkbox name="numeros[]" value=10000> 10000<br>
<input type=checkbox name="numeros[]" value=90> 90<br>
<input type=checkbox name="numeros[]" value=50> 50<br>
<input type=checkbox name="numeros[]" value=30> 30<br>
<input type=checkbox name="numeros[]" value=15> 15<br><BR>
<input type=checkbox name="news" value=1> <B>Receber
Newsletter?</B><br><BR>
<input type=submit>
</form>
```

checkbox.php

```
<?php
// Verifica se usuário escolheu algum número
if(isset($_POST["numeros"]))
{
   echo "Os números de sua preferência são:<BR>";
   // Faz loop pelo array dos numeros
   foreach($_POST["numeros"] as $numero)
```

```
    {
        echo "- " . $numero . "<BR>";
    }
}else{
    echo "Você não escolheu número preferido!<br>";
}

// Verifica se usuário quer receber newsletter
if(isset($_POST["news"]))
{
    echo "Você deseja receber as novidades por email!";
}
else
{
    echo "Você não quer receber novidades por email...";
}
```

Select

select.html

```
<form action="select.php" method="post">
<B>Qual seu processador?</B><br>
<select name=processador>
<option value=Pentium>Pentium</option>
<option value=AMD>AMD</option>
<option value=Celeron>Celeron</option>
</select><BR><BR>
<B>Livros que deseja comprar?</B><br>
Obs: segure "CTRL" para selecionar mais de um.<BR>
<select name="livros[]" multiple>
<option value="Biblia do PHP 4">Biblia do PHP 4</option>
<option value="PHP Professional">PHP Professional</option>
<option value="Iniciando em PHP">Iniciando em PHP</option>
<option value="Novidades do PHP 5">Novidades do PHP 5</option>
<option value="Biblia do MySQL">Biblia do MySQL</option>
</select><BR><BR>
<input type=submit>
</form>
```

select.php

```
<?php
echo "Seu processador é: " . $_POST["processador"] . "<BR>";

// Verifica se usuário escolheu algum livro
if(isset($_POST["livros"]))
{
    echo "O(s) livro(s) que você deseja comprar:<br>";
    // Faz loop para os livros
    foreach($_POST["livros"] as $livro)
    {
        echo "- " . $livro . "<br>";
    }
}
else
{
    echo "Você não escolheu nenhum livro!";
}
```

Adaptado de
https://www.devmedia.com.br/php-forms-manipulando-dados-de-formularios/29392

9.4 – Métodos HTTP

Existem vários tipos de métodos HTTP:

GET
POST
PUT
HEAD
DELETE
PATCH
OPTIONS

Mas abordaremos aqui somente os mais usados: GET, POST e REQUEST.

9.4.1 – GET

Envia as variáveis para o servidor através da URL.

Tem o limite do tamanho da URL controlado.

O método GET envia dados para o servidor através da URL com pares de chave e valor

/test/demo_form.php?name1=value1&name2=value2

- As requisições do GET podem ser armazenados em cache
- permanecem no histórico do navegador
- podem ser marcados nos bookmarks
- nunca devem ser utilizados quando se trata de dados sensíveis
- têm restrições de comprimento
- são utilizados apenas para solicitar dados (não para modificar)

Exemplo

```
<form action="<?php echo $_SERVER['PHP_SELF'];?>" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
<?php
if(isset($_GET['name'])){
   print $_GET['name'].'<br>';
   print $_GET['email'].'<br>';
}
?>
```

9.4.2 - POST

POST é Um array associativo de variáveis passados para o script atual via método HTTP POST quando utilizado application/x-www-form-urlencoded ou multipart/form-data como valor do cabeçalho HTTP Content-Type na requisição.

O método POST é mais indicado para oeprações seguras, pois não envia as variáveis para o servidor pela URL.

Os dados enviados pelo método POST são arzenados no body re request do HTTP

As requisições enviadas via POST nunca são armazenados em cache
não permanecem no histórico do navegador
não podem ser marcados
não têm restrições quanto ao comprimento dos dados

O maior tamanho das variáveis enviadas para o servidor depende das configurações do PHP, especificamente de post_max_size

Que pode ser alterada no php.ini ou no .htaccess
php_value post_max_size 20M

Ainda no ini_set
ini_set('post_max_size', '20M')

PHP Superglobal - $_POST

Variável superglobais são disponível em todos os escopos

Exemplo

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Nome: <input type="text" name="fnome">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $nome = $_POST['fnome'];
  if (empty($nome)) {
    echo "Name is empty";
  } else {
    echo $nome;
  }
}
?>
```

echo $_SERVER['PHP_SELF']; - Faz o form chamar a si mesmo e não outro arquivo

```
<?php
//Displays the data that was received from the input box named name in the form
if(isset($_POST['name'])){
print $_POST['name'];
}
?>

<form action="1.php" method=POST>
      Name:<br><input type="text" name="name"><br>
      <input type="submit" value="Submit">
</form>
```

9.4.3 – REQUEST

$_REQUEST

O que é?

A variável $_REQUEST é uma variável com o conteúdo das variáveis $_GET e $_POST e $_COOKIE.

Como usá-lo?

Antes de usar a variável $_REQUEST, você deve ter um formulário em html que tenha o método igual a GET e POST. Então, no php, você pode usar a variável $_REQUEST para obter os dados que deseja. Dependendo do que você escreveu para o método no formulário e usando $_REQUEST no php, $_REQUEST usará $ _Get se GET for escrito para o método e $_REQUEST usará $ POST se POST for escrito no método. A sintaxe $_REQUEST é ($_REQUEST ['nome do campo do formulário vai aqui']).

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_POST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>

</body>
</html>
```

https://www.php.net/manual/pt_BR/reserved.variables.request.php
http://www.shodor.org/~kevink/phpTutorial/nileshc_getreqpost.php
https://zetcode.com/php/getpostrequest/
https://www.tutorialspoint.com/php/php_get_post.htm
https://www.w3schools.com/tags/ref_httpmethods.asp

10 – Ferramentas

10.1 - Terminal/Prompt

Com o sucesso de ferramentas como o composer e o git, o terminal/prompt se tornou muito utilizado e importante.

Atualmente se faz muita coisa neles.

10.2 – Git

O git atualmente é uma ferramenta muito importante, especialmente se aliada ao Github.

Git é um sistema de controlo de versões distribuído gratuitamente e de código aberto, concebido para lidar com tudo, desde projetos pequenos até grandes, com rapidez e eficiência.

Git é fácil de aprender e tem uma pegada minúscula com um desempenho rápido de relâmpago. Ultrapassa ferramentas como Subversion, CVS, Perforce, e ClearCase com características como sucursais locais baratas, áreas de encenação convenientes, e múltiplos fluxos de trabalho.

Traduzido com a versão gratuita do tradutor - www.DeepL.com/Translator

Site oficial - https://git-scm.com/

Instalação no Linux

sudo apt install git

Enviar código para o servidor

Acessar a pasta onde está o código a ser enviado

```
git config --global user.name "Ribamar FS"
git config --global user.email "ribafs@gmail.com"
git init
git add .
git commit -m "Primeiro Commit"
git remote add origin git@github.com:ribafs/apostila.git
git push -f origin master
```

Para enviar novamente após alterações
remover a pasta .git
e repetir os passos acima

Somente na primeira vez que usar o git precisa enviar o user.name e user.email, nas próximas vezes não precisa usar.

O projeto está agora hospedado no GitHub.

10.2.1 – Sincronizando repositório local com remoto

Algo que me tem sido de muita utilidade é a sincronização de repositório local com o repositório remoto no servidor.

Supondo que queira sincronizar localmente o repositório
https://github.com/ribafs/apostilas

Faço a clonagem dele do servidor para meu desktop, assim:
Acesso
https://github.com/ribafs/apostilas

Clique no botão Code

E copie para a memória

git@github.com:ribafs/apostilas.git

Abro o terminal e digito

git clone git@github.com:ribafs/apostilas.git apostilas

Ao teclar enter ele fará uma cópia do repositório na minha pasta local apostilas

cd apostilas

Qualquer alteração que eu faça nos arquivos, após executar os comandos abaixo o repositório local será sincronizado com o remoto. Ambos ficarão idênticos:

git add .
git commit -m "Update"
git pull
git push

Obs.: esta sincronização funciona em repositórios simples, com apenas o master, sem branchs.

10.3 – Github

O Github é atualmente a grande rede social dos programadores. É uma boa ideia hospedar por lá alguns projetos, para que sirvam para divulgação e promoção do seu trabalho.

Site oficial - https://github.com/

Atualmente praticamente todos os meus projetos estão hospedados no Github, inclusive sites (meu site https://ribamar.net.br está lá).

Em parceria com o git formam uma dupla que não pode faltar aos programadores atuais.

Existem planos gratuitos e comerciais. Os gratuitos são bem generosos.

10.4 – Composer

O composer é um gerenciador de dependências, que permite que instalemos componentes sem sair do terminal/prompt com grande praticidade e rapidez.

Gerenciamento de Dependência

Há uma tonelada de bibliotecas PHP, estruturas e componentes para escolher. Seu projeto provavelmente usará várias delas

• estas são dependências de projetos. Até recentemente, o PHP não tinha uma boa maneira de administrar estas dependências de projeto. Mesmo se você as administrasse manualmente, ainda teria que se preocupar com os autoloaders. Isso não é mais um problema.

Atualmente existem dois grandes sistemas de gerenciamento de pacotes para PHP - Composer e PEAR. O Composer é atualmente o gerenciador de pacotes mais popular para PHP, porém pormuito tempo o PEAR foi o principal gerenciador de pacotes em uso. Conhecer a história do PEAR é uma boa idéia, já que você ainda pode encontrar referências a ele, mesmo que nunca o use.

Composer e Packagist
O Composer é o gerenciador de dependências recomendado para PHP. Lista as dependências de seu projeto em um arquivo composer.json e, com alguns comandos simples, o Composer fará automaticamente o download das dependências de seu projeto e configurará o autoloading para você. O Composer é semelhante ao NPM no mundo do node.js, ou Bundler no mundo do Ruby.

Há uma infinidade de bibliotecas PHP compatíveis com o Composer e estão prontas para serem usadas em seu projeto. Estes “pacotes” estão listados no Packagist, o repositório oficial para bibliotecas PHP compatíveis com o Composer.

Os grandes softwares, como frameworks e CMS atualmente podem ser instalados usando o composer. Exemplos:
Framework CakePHP
composer create-project --prefer-dist cakephp/app crud

Framework Laravel
composer create-project laravel/laravel crud

Instalação de apcotes
Após instalar o laravel, se quero instalar no aplicativo um pacote (laravel-acl):

cd crud
composer require ribafs/laravel-acl

Removendo pacote
composer remove ribafs/laravel-acl

Boa Prática
Quando enviamos um aplicativo para o Github, antes eliminamos a pasta vendor.

Ao baixarmos o aplicativo devemos executar:
cd aplicativo
composer update

Assim o composer refaz a pasta vendor com todas as dependências.

11 – Segurança

Os cuidados com a segurança colaboram para que os sites e aplicativos instalados no servidor sejam executados de forma esperada, rápida e sem interrupção.

11.1 - Melhorar a segurança no Desktop do programador

- Uma boa máquina, rápida, estável, com um bom espaço em disco
- Idealmente seu sistema oepracional é uma cópia do servidos: SO, distro, versão, serviços e configurações. Caso monitore mais de um servidor talvez deva ter uma cópia de cada em uma VM ou em um containers docker ou de outra forma. Para que não fique lidando com conflitos de sistema, versão, etc.
- De rpeferência com vários monitores grandes para monitorar os servidores com conforto
- Todos os softwares de que necessita

Melhorar a segurança no desktop é importante para maior segurança do servidor, pois estamos interagindo frequentemente com ele do nosso desktop e poderemos comprometê-lo.

Hábitos saudáveis como usar um sistema operacional menos inseguro e atualizado, como usando um firewall ativo e fechando tudo que pode.

Assim como também instalando boas ferramentas de monitoramento do servidor no desktop.

Usar Senhas Fortes

De que vai adiantar ter todo este trabalho de escolher uma boa hospedagem, de isntalar um sistema operacional seguro, atualizar o sistema e efetuar diversas medidas para melhorar a segurança, nada vai adiantar se usarmos senhas fracas.

Senhas fortes são grandes (8 dígitos ou mais) e usam uma mistura de algarismos, letras minúsculas, letras maiúsculas e símbolos. Quanto maior mais forte a senha.

11.2 - Princípios básicos de segurança

- Hospede seu site em servidor seguro
- Efetue backup regularmente, especialmente a cada alteração no site
    Lembre de fazer o backup do servidor com os recursos da hospedagem ou crie um snapshot
- Também faça teste de restore de vez em quando para garantir que o backup está integro
- A quantidade de cópias de backup a ser guardada depende da importância do site. Se mais importante mais cópias
- As cópias devem ser armazenadas em mídia confiável: HD e DVD, pendrive ou fitas
- Efetue atualização com frequência. Mantenha o aviso de atualização ativo para que receba um aviso por e-mail e atualize imediatamente
- Após a primeira atualização reinicie o servidor
- Acessar de forma segura usando SSH (enxuto e configurado para salvar a senha) e nunca via FTP
- Manter seu desktop seguro, usando um sistema operacional seguro no mesmo, com firewall e outros cuidados

- Use e abuse da comunidade com seus conhecimentos e generosidade para manter-se atualizado em termos de segurança e proteger seu site
- Use senhas fortes
- Use o SSL para proteger o site
- Evite instalar pacotes para desenvolvimento como gcc, make, etc e evite também instalar repositórios instáveis tanto no servidor quanto no desktop.
- Monitorar frequentemente os logs à procura de algo suspeito em todos os serviços ativos
- Use softwares tipo IDS que detectam intrusões
- Instalar um bom firewall de aplicativos como o mod_security no servidor
- Ficar bem atento, estudando, se informando sempre sobre o assunto de que cuida
- Logo após a configuração final do servidor já crie um backup ou snapshot e fique atento para criar outro logo que o servidor esteja concluído e bem configurado.
- Uma boa ideia é ter uma box no Vagrant do Ubuntu mesma versão usada no servidor em seu desktop, sendo cópia fiel e original do servidor localmente, mesma distribuição, mesma versão

A segurança tem muitas áreas, faces, recursos, cuidados, etc. Como qualquer área humana, não existe segurança perfeita. O papel de quem cuida dela é o de fazer o melhor ao seu alcance.

Algumas áreas da segurança, com a intenção de mostrar que existe praticamente uma infinidade de detalhes que precisam de atenção, pois a maior segurança de uma corrente é a do elo mais fraco. Então para manter um servidor seguro, todas as suas partes precisam de cuidados. Não é prudente cuidar das mais importantes e esquecer as menos.

- Estrutura física
- Prédio
- Instalações elétricas, fios, estabilizadores, no-breaks, etc
- Instalações de ar condicionado,
- Instalações telefônicas,
- Dispositivos contra incêndios,
- Instalações de rede: cabos, conectores, switches, etc
- Estrutura de internet: cabos, placas, roteadores, etc
- Hardware: firewalls, computadores, storages, etc
- etc

- Pessoal (que precisam de acesso ao prédio temporária ou permanentemente)
- Manutenção: pedreiros, pintores, eletricistas, técnicos em telefonia, hardware, redes, internet, etc,
- Zeladores, chefes, diretores, etc
- Programadores, técnicos em segurança, analistas, DBA, etc
- etc

- Servidores
- Sistemas operacionais
- Firewalls
- Softwares para reforçar a segurança em geral
- Softwares de virtualização
- Softwares de backup
- Softwares para monitoração local e remota (somente alguns)

Ok, é suficiente, pois a ideia nem é mostrar todos os detalhes de um servidor, nem mesmo muita coisa, mas apenas alertar para o fato que existem muitos pontos e cada um deles pode ser motivo de dor de cabeça. Cada detalhe precisa de atenção para que possamos dizer que nossa corrente tem a

força do elo mais forte, que é idêntica a de todos os elos. Me parece que isso é o ideal. Alias, acredito que tudo que façamos na vida deveria ser assim, ou seja, deveríamos fazer tudo bem feito, tanto porque é o melhor a fazer quanto assim estaremos mandando uma mensagem ao nosso subconsciente para que fique atento e passe a automatizar isso. Então com algum tempo estaremos dando uma forte atenção a tudo que fazemos sem esforço, de forma automática, pois o subconsciente está cuidando.

A segurança é influenciada pela popularidade e pela facilidade de uso

Quando mais popular relativamente menos seguro

Quanto mais amigável provavelmente menos seguro

Importante configurar atualizações automáticas pelo menos dos pacotes de segurança

Habilitar um firewall para bloquear todas as portas, exceto as necessárias

Sobre firewall

Não permita login do root ao servidor

Uma forte política de segurança nas senhas
Então, quais são as "melhores práticas" ao configurar senhas?

1. Use senhas tão longas quanto você possa gerenciar
2. Evite palavras que aparecem no dicionário (como "uvas azuis")
3. Evite substituições de números que sejam fáceis de adivinhar (como "h3ll0")
4. Não faça referência à cultura pop (como “TARDIS”)
5. Nunca use uma senha em mais de um lugar
6. Altere sua senha regularmente e use uma diferente para cada site
7. Não anote as senhas e não as compartilhe. Não com ninguém. Sempre!

https://www.plesk.com/blog/various/linux-server-security-best-practices/

Use sempre SSL

Instale e configure o ModSecurity

Aplicar política de senha forte em servidores com vários usuários

Adicionando a linha acima, a senha inserida não pode conter mais de 3 caracteres em uma sequência monotônica, como abcd, e mais de 3 caracteres consecutivos idênticos, como 1111.

Para forçar os usuários a usar uma senha com um comprimento mínimo de 8 caracteres, incluindo todas as classes de caracteres, verifique a força para sequências de caracteres e caracteres consecutivos, adicione as seguintes linhas ao arquivo /etc/security/pwquality.conf.

https://www.tecmint.com/centos-7-hardening-and-security-guide/

https://bigstep.com/blog/five-ways-to-secure-a-centos-8-server

Depois que um servidor está instalado e funcionando, o root não deve se conectar diretamente, exceto em situações de emergência. Isso geralmente requer mãos no console, então esse é o único

lugar onde o root deve ter permissão para fazer login. Para fazer isso, precisamos modificar / etc / securetty. Além disso, ninguém além do root deve ser permitido no diretório inicial do root. As configurações padrão são próximas disso, mas não são paranóicas o suficiente.

Uma vez que removemos efetivamente a capacidade do root de logar de qualquer lugar, exceto do console local, é necessário usar su e sudo. Isso oferece alguns benefícios secundários em um ambiente multi-admin.

Referências

https://geek.linuxman.pro.br/geek/ubuntu-pronto-para-guerra
https://www.thefanclub.co.za/how-to/how-secure-ubuntu-1604-lts-server-part-1-basics
https://linux-audit.com/ubuntu-server-hardening-guide-quick-and-secure/
https://hostpresto.com/community/tutorials/how-to-install-and-use-lynis-on-ubuntu-14-04/
https://wiki.centos.org/HowTos/OS_Protection

11.3 - Checklist de Segurança para Joomla

- Efetue um backup completo de todos os arquivos e do banco e restaure localmente
- Mudar prefixo das tabelas durante a isntalação. Após a instalação precisará alterar todo o banco mudando o prefixo de todas as tabelas
- Criar novo super usuário e remover o de ID 62. Lembre de usar senha forte para o super usuário
- Ativar URLs amigáveis e mod_rewrite
- Mover configuration.php para fora do public_html, usando:
require_once(dirname(__FILE__) . '/../../portal.cfg');
- Bloquear cadastro de usuários pelo site caso não tenha necessidade: Configuração Global - Sistema - Permitir Cadastro de Usuários - Não
- Alterar metatags em Configuração Global - Configurações de Meta Dados (Trocar Joomla por outra palavra)
- Adicionar a tag <head> do template (para ocultar na origem do código HTML):
<?php $this->setGenerator('Ribafs - Desenvolvimento Web'); ?>
- Instalar extensões:
- AdminTools
- Plugin osolcapcha
- com_encrypt
- jHackGuard

- Se possível/viável escolher a melhor hospedagem, não a mais barata;

- Utilizar sempre a última versão do CMS e das extensões;

- Efetue um backup completo de todos os arquivos e do banco e restaure localmente

- Efetuar backup completo com frfequência, especialmente antes de instalar novas extensões ou efetuar alterações como adição de conteúdo

- Ativar URLs amigáveis e mod_rewrite

- Bloquear cadastro de usuários pelo site caso não tenha necessidade: Configuração Global - Sistema - Permitir Cadastro de Usuários - Não

- Alterar metatags em Configuração Global - Configurações de Meta Dados (Trocar Joomla por outra palavra)

- Faça sempre o download do Joomla do site oficial - http://joomla.org

- Cheque o hash MD5 do arquivo baixado:
md5sum Joomla_3.7.5-Stable-Full_Package.zip
bd67cb02627e60bffef5e3b4ba3b2ece Joomla_3.7.5-Stable-Full_Package.zip

- Instalar os principais navegadores para testar o site:
Firefox, Chrome, Internet Explorer, Opera, Safari

- Mantenha os arquivos de configuração, logs e os diretórios de upload (repositórios de documentos, imagens e chache) fora do public_html.

- Remover desnecessários:

Arquivos

Extensões (se não precisa, remova e não simplesmente desabilite. Caso queira instale novamente)

- Sempre antes de instalar novas extensões:

- faça um backup completo do site e instale localmente

- Verifique se a extensão é confiável em:

https://docs.joomla.org/Archived:Vulnerable_Extensions_List

Uma ferramenta simples de backup é o com_simplebackup -
https://github.com/ribafs/com_simplebackup

Usar a ferramenta:

joomlascan - https://github.com/rezasp/joomscan
Um bom tutorial - http://www.100security.com.br/joomscan/
sudo apt-get install libswitch-perl

- Download
- Descompactar e acessar a pasta

Atualizar

./joomscan.pl update
Checar a atualização
svn co https://joomscan.svn.sourceforge.net/svnroot/joomscan joomscan
Varrer site procurando vulnerabilidades
./joomscan.pl -u http://www.joomla.org

Ativar o cache
Otimizar as tabelas do banco no phpmyadmin
Usando Captcha (plg_osolcaptcha) para forms adicionais.

array(true) para form vertical e false para horizontal

```
<?php
global $mainframe;
//set the argument below to true if you need to show vertically( 3 cells one below the other )
$mainframe->triggerEvent('onShowOSOLCaptcha', array(false));
?>
```

Alterar permissões de arquivos:

Alterar todos os arquivos para 644 e todas as pastas para 755 com:

```
find . -type f -exec chmod 644 {} \;
find . -type d -exec chmod 755 {} \;
```

Depois criar algumas exceções...

configuration.php – 400
index.php do site – 400
index.php do template padrão – 400
Permissões de pastas:
includes e libraries – 500

Remover templates não usados e outras extensões também.
Adicionar ao .htaccess:

```
# Block out any script trying to set a mosConfig value through the URL
RewriteCond %{QUERY_STRING} mosConfig_[a-zA-Z_]{1,21}(=|\%3D) [OR]

# Block out any script trying to base64_encode crap to send via URL
RewriteCond %{QUERY_STRING} base64_encode.*\(.*\) [OR]
# Block out any script that includes a <script> tag in URL
RewriteCond %{QUERY_STRING} (\<|%3C).*script.*(\>|%3E) [NC,OR]
# Block out any script trying to set a PHP GLOBALS variable via URL
RewriteCond %{QUERY_STRING} GLOBALS(=|\[|\%[0-9A-Z]{0,2}) [OR]
# Block out any script trying to modify a _REQUEST variable via URL
RewriteCond %{QUERY_STRING} _REQUEST(=|\[|\%[0-9A-Z]{0,2})
# Send all blocked request to homepage with 403 Forbidden error!
RewriteRule ^(.*)$ index.php [F,L]
```

Adicionar ao configuration.php:

```
  ini_set('extension', 'sourceguardian.so');
        ini_set('register_globals', 'off');
        ini_set('session.save_path', '/home/ribafs03/public_html/tmp');
        ini_set('cgi.force_redirect', 1);
        ini_set('allow_url_fopen', 0);
        ini_set('display_errors', 0);
        ini_set('allow_url_include', 0);
        ini_set('expose_php', 0);
        ini_set('magic_quotes_gpc', 0);
        ini_set('post_max_size', '262144');
        ini_set('upload_max_filesize', '262144');
```

```
ini_set('upload_tmp_dir','/home/joao/public_html/tmp');
$disfunctions = 'proc_open, popen, disk_free_space, set_time_limit, leak, tempfile, exec,
system, shell_exec, passthru, curl_exec, curl_multi_exec, parse_ini_file, show_source,
apache_get_modules, apache_get_version, apache_getenv, apache_note, apache_setenv,
disk_free_space, diskfreespace, dl, highlight_file, ini_alter, ini_restore, openlog,  proc_nice,
symlink, phpinfo';
ini_set('disable_functions', $disfunctions);
ini_set('zend_extension', '/usr/local/php52/lib/php/extensions/ioncube.so');
ini_set('zend_extension_manager.optimizer=', '/usr/local/Zend/lib/Optimizer-3.3.3');
ini_set('zend_extension_manager.optimizer_ts', '/usr/local/Zend/lib/Optimizer_TS-3.3.3');
ini_set('zend_optimizer.version', '3.3.3');
ini_set('zend_extension', '/usr/local/Zend/lib/ZendExtensionManager.so');
ini_set('zend_extension_ts', '/usr/local/Zend/lib/ZendExtensionManager_TS.so');
```

Adicionar ao php.ini (alternativa):
Este é para o caso do servidor permitir um php.ini no raiz que será visto por todas as partas recursivamente.

```
extension=sourceguardian.so
register_globals = off
session.save_path = "/home/ribafs03/public_html/tmp"
cgi.force_redirect = 1
allow_url_fopen= 0
display_errors = 0
expose_php = 0
magic_quotes_gpc = 0
memory_limit = 8388608
#open_basedir = 1
post_max_size = 262144
upload_max_filesize = 262144
upload_tmp_dir = "/home/ribafs03/public_html/tmp"
disable_functions = proc_open, popen, disk_free_space, set_time_limit, leak, tempfile, exec,
system, shell_exec, passthru, curl_exec, curl_multi_exec, parse_ini_file, show_source,
apache_get_modules,apache_get_version,apache_getenv,apache_note, apache_setenv,
disk_free_space, diskfreespace, dl, highlight_file, ini_alter, ini_restore, openlog, proc_nice,
symlink, phpinfo
zend_extension=/usr/local/php52/lib/php/extensions/ioncube.so
zend_extension_manager.optimizer=/usr/local/Zend/lib/Optimizer-3.3.3
zend_extension_manager.optimizer_ts=/usr/local/Zend/lib/Optimizer_TS-3.3.3
zend_optimizer.version=3.3.3
zend_extension=/usr/local/Zend/lib/ZendExtensionManager.so
zend_extension_ts=/usr/local/Zend/lib/ZendExtensionManager_TS.so
```

Vários dos recursos acima você precisará confirmar com o suporte do seu servidor para ver se estão disponíveis.

Reportar extensões vulneráveis e retirar extensão da lista
https://extensions.joomla.org/vulnerable-extensions/about/

Extensões corrigidas
https://extensions.joomla.org/index.php?option=com_content&view=category&id=10511&Itemid=1056

- Faça o download do site do criador

- Teste bastante localmente e somente então envie para o servidor

- Evite instalar extensões que tenham código criptografado

- Sempre que possível evite hospedar seu site em servidores compartilhados

- Use um servidor de SSL, pelo menos para o administrator

- Use o .htaccess

- Atualize para a versão 3 e última do Joomla

11.4 - Verificações antes de publicar um novo site

1 - Favicon
Uma boa personalização para o site é adicionar um favicon bem representativo.
Para sites pessoais podemos adicionar a foto do dono do site ou blog.
No site abaixo entramos com uma foto e ele retorna o favicon.ico:
http://www.degraeve.com/favicon/ ou então desenhe o seu favicon.

No caso de sites com Joomla que já tenham um favicon no template, basta sobrescrever o existente.
Para sites que ainda não tenham, adicione entre as tags <head>:
<link rel="icon" type="image/x-icon" href="/favicon.ico" />

2 - Título e Metadados
O título do site é um elemento importante em termos de SEO (Search Engine Otimization).
<title>RibaFS Portal</title>

Como também tags de descrição do site:
<meta name="description" content="Este é um site pessoal dedicado ao desenvolvimento web em PHP. Os focos são o CMS Joomla e PostgreSQL. Sobre o Joomla encontram-se tutoriais, dicas, cursos, módulos e componentes. Sobre o PostgreSQL tutoriais, dicas e cursos." />

3 - Checagem do site em vários navegadores
Como nosso site geralmente será visualizado por diversos navegadores é importante que ele seja visualizado corretamente pelo menos nos mais populares.

Checagem do ACID2 - http://www.webstandards.org/files/acid2/test.html

Checagem do CSS3 - http://www.css3.info/selectors-test/test.html

Download dos 8 navegadores mais populares - http://www.xenocode.com/browsers/

IETester - http://www.my-debugbar.com/wiki/IETester/HomePage

Testar o site em vários navegadores de vários sistemas operacionais - http://browsershots.org/ .
Entre com o site e aguarde a geração dos screenshots, que ao final poderá fazer download.

4 - Leitura completa e mais atenta
Leia com bastante atenção e procurando erros de grafia e concordância.

5 - Links
Clique em cada link para testar se estão corretos.
Certifique-se de que sua logomarca tem um link para seu próprio site (convenção comun).
Não sublinhe texto que não é link pois isso confunde.

Site que checa por links - http://validator.w3.org/checklink

6 - Checar funcionalidade
Caso tenha um form de contato, teste se está funcionando.
Peça colegas de grupos para testar seu site e não somente parentes e amigos.
Áreas importantes para testar são: contato, login, busca, etc.

7 - Desabilitando recursos
Caso seu site possa funcionar sem JavaScript desabilite o JavaScript e teste como fica o seu site.

8 - Validações
Verifique as validações se funcionam adequadamente. Valide o CSS e XHTML.
Veja este artigo: 10 razões por que seu código não valida e como corrigir isso:
http://net.tutsplus.com/articles/web-roundups/10-reasons-why-your-code-wont-validate-and-how-to-fix-it/

Validação online - http://validator.w3.org/ e http://jigsaw.w3.org/css-validator/

9 - Link para RSS
Este recurso é muito importante para quem visita o site e para atrair visitantes.
Através do RSS o visitante pode voltar ao site.

10 - Mapa do Site
Adicione ou verifique o mapa do site. Quando o visitante não encontrar um recurso do site o mapa o ajudará.

11 - Design defensivo
Crie bonitas e eficientes páginas para o erro 404, quando o usuário solicitar uma página e não encontrar.
Nessa página personalizada deve ter um pedido de desculpa, seu e-mail e a orientação de verificar se o link
está correto.

12 - Otimize
Aproveite para melhorar a performance do seu site, eliminando recursos desnecessários e melhorando outros:
imagens, flash, javascript, etc. Veja o artigo:
http://ribafs.org/portal/joomla/78-dicas/125-melhores-praticas-para-um-site-mais-rapido

13 - Backup
Lembre que o backup é muito importante para garantir as informações do seu site.

Faça regularmente backup dos arquivos e do banco de dados.

14 - CSS para Impressão
Caso seu site não tenha procure adicionar recurso para que o visitante possa ter uma página de impressão para cada página que queira imprimir.

15 - Estatísticas
Caso queira estatísticas sobre os visitantes do site - http://www.google.com/analytics/

CheckList:
Aqui encontrará um PDF para que possa imprimir e realizar um controle mais preciso:
http://www.boxuk.com/upload/website_launch_checklist_v1.pdf

Referências: http://www.smashingmagazine.com/2009/04/07/15-essential-checks-before-launching-your-website/

12 – Novidades do PHP 7 e 8

12.1 – PHP 7

Funções anônimas
Elas surgiram com o PHP 7
Veja exemplos no capítulo 7.9

PHP nasceu como uma linguagem dinâmica e fracamente tipada e passou a ter melhorias em relação à tipos com a sua evolução.

No início ele aceitava somente typehints de classes e interfaces em argumentos de métodos. Depois, com o lançamento do PHP 7, passou-se a ter suporte a typehints de tips escalares como string, int, float e bool, tipos para retorno de métodos e opção de tipagem estrita com `declare(strict_types=1)`.

No PHP 7.1 passou-se a suportar tipos nullable, ou seja, se a variável aceita nulo ou não e aos tipos void e iterable. E no PHP 7.2 o suporte ao tipo object.

Contudo, o PHP ainda não tinha suporte para tipos em propriedades. Ela quase foi incluída na versão 7.3, mas a equipe de desenvolvimento do core do PHP decidiu incluir com calma e deixaram para o PHP 7.4.

Anteriormente a isso utilizava-se uma annotation em docblock (@var EntityManagerInterface) para sinalizar que determinada propriedade era de um tipo específico. Isso auxiliava IDEs no seu autocomplete e ferramentas de análise estática.

Novidades do PHP 7

Algumas novidades

Referências
https://tableless.com.br/10-novidades-do-php-7/
https://www.treinaweb.com.br/blog/php-7-e-novidades-do-php-7-1/
https://www.php.net/manual/en/migration70.new-features.php
https://github.com/tpunt/PHP7-Reference
https://devzone.zend.com/4693/php-7-glance/
https://blog.digitalocean.com/getting-ready-for-php-7/
https://www.treinaweb.com.br/blog/novidades-do-php-7-2/
https://imasters.com.br/back-end/php-7-2-quais-sao-as-novidades-da-nova-versao-do-php
https://imasters.com.br/back-end/php-7-3-e-php-8-o-que-esperar-das-proximas-versoes

Novidades do PHP 7.1
- Nullable Types (possibilidade de um parâmetro receber um tipo específico ou null)
- Habilidade de pegar múltiplas exceções num mesmo bloco catch
- Criação de um pseudo-tipo chamado Iterable
- Habilidade de definir visibilidade para constantes de classes (public, private, protected)
- Diversas melhorias à extensão Curl, dentre elas, suporte a HTTP/2 Server Push.
- Tipo void para parâmetros e retornos de funções/métodos.

• Incrementos na utilização list()
• O suporte a mcrypt() foi removido da linguagem (tornou-se defasado devido à implementações mais atualizadas e seguras que hoje temos à disposição).

Capturando múltiplas exceções
Antes a única opção era:

```
try {
   // todo
} catch (MyException1 $e) {
   // todo
} catch (MyException2 $e) {
   // todo
} catch (Exception $e) {
   // todo
}
```

No PHP 7.1 é possível agrupar mais de uma exceção num mesmo bloco catch:

```
try {
   // todo
} catch (MyException1 | MyException2 $e) {
   // todo
} catch (Exception $e) {
   // todo
}
```

Void return

Na 7.1 é possível retornar void como sendo um tipo de retorno válido.

Exemplo:

```
class Squirtle extends Pokemon
{
   public function run() : void
   {
      // todo
   }
}
```

Funções removidas

As funções de acesso a bancos de dados mysql_* (mysql_connect(), mysql_query() entre outras) foram removidas na versão 7. A recomendação agora é usar o PDO.

Funções ereg_* foram removidas. Agora devemos usar uma função preg_*, como preg_match ou preg_replace.

Erros Fatais e Exceções
No PHP 7, erros fatais passaram a ser Exceções. Isso quer dizer que eles podem ser tratados em bloco try/catch, sem interromper a execução do script.

Construtores do PHP 4 continuará sendo possível mas recomendando o __construct();

Definir tipo de retorno para funções:

```
function nomeFuncao() : tipo
{
   // corpo da função
}
```

Por exemplo:

```
function soma($x, $y) : float
{
   return $x + $y + 1.5;
}
```

Criação de classes anônimas

```
function createObject()
{
   return new class{
      public function test()
      {
         echo "test" . PHP_EOL;
      }
   };
}

$obj = createObject();
$obj->test();
```

Abaixo algumas das funcionalidades que se tornarão obsoletas no 7.2:

- __autoload
- $php_errormsg
- create_function()
- func_overload
- (unset) cast
- parse_str() without second argument
- gmp_random()
- each()
- assert() with string argument
- $errcontext argument of error handler

Principais novidades que entrarão no PHP 7.2. Algumas das principais:

- get_class() desabilita o parâmetro nulo
- Impedir number_format() de retornar zero negativo
- Argon2 Password Hash
- Object typehint
- libsodium

Argon2 Password Hash

Argon2 é o algoritmo de hashing de senha vencedor do concurso “Password Hashing Competition” de 2015, sendo assim muito bem recomendado a sua utilização. Ao contrário do Bcrypt, que apenas possui um único fator de custo, o Argon2 é parametrizado por três fatores distintos:

1. Tempo de execução

2. Memória necessária
3. Grau de paralelismo

A função password_hash() é alterada para aceitar PASSWORD_ARGON2I como o algoritmo e aceitar o custo da memória, o custo do tempo e o grau de paralelismo como opções. O exemplo a seguir ilustram a nova funcionalidade:

```
// Argon2i com fatores de custo padrão
password_hash('password', PASSWORD_ARGON2I);

// Argon2i com fatores de custo personalizados
password_hash('password', PASSWORD_ARGON2I, ['memory_cost' => 1<<17, 'time_cost' => 4, 'threads' => 2]);
```

A primeira linguagem de programação a adotar criptografia moderna

Outra novidade que veio na versão 7.2 é referente à criptografia moderna, libsodium, que é parte da extensão principal do PHP 7.2. Sodium é uma biblioteca que facilita a utilização de criptografia, descriptografia, assinaturas, hash de senha e muito mais. Seu objetivo é fornecer todas as operações básicas necessárias para criar ferramentas criptográficas de alto nível.

O PHP continua sendo a linguagem mais popular do lado do servidor para criar sistemas. Com uma participação de mercado estimada em 80%, a linguagem de programação de vinte e poucos anos está em toda parte.

O que se tornará obsoleto no PHP 7.3?

Abaixo, algumas das funcionalidades que se tornarão obsoletas:

- Extensão WDDX;
- Alias de função mbstring usando um prefixo mb_ (por exemplo, mb_ereg);
- mb_detect_encoding() sem strict mode;
- Funções strip_tags() e fgetss();
- Função image2wbmp.

Principais novidades que entrarão no PHP 7.3

- Flexibilidade de sintaxe Heredoc e Nowdoc;
- Permitir vírgula à direita em chamadas de função e método;
- Opção para fazer json_encode e json_decode lançar exceções em erros;
- Atribuições de referências em list();
- Função is_countable().

Novidades sobre o PHP 7.3

https://imasters.com.br/php/php-7-3-conheca-as-novidades-desta-versao

Novidades do PHP 7.4

Poderemos declarar tipos estáticos para variáveis

```
class User
{
   public int $id;
   public string $nome;
   private bool $isAdmin = false;
}
```

Operador null coalescing

/ $data['name'] = 'John';

// verifica se a variável foi definida e não é nula utilizando o operador null coalesce
$name = $data['name'] ?? 'anonymous';
echo $name; // anonymous

Uma novidade muito forte e muito comentada foi a melhora no desempenho (teve seu motor remodelado), que alguns testes chegou a um ganho de 9 vezes.
Veja aqui - https://rberaldo.com.br/php-7-9-vezes-mais-rapido-que-php-5-6/

12.2 – PHP 8

Mais novidades na pasta Estruturado/Novidades do repositório

https://github.com/ribafs/apostilas/tree/main/Anexos/Estruturado/Novidades

Downloads
Esta apostila pode ser encontrada originalmente no repositório
https://github.com/ribafs/apostilas

E na seção Arquivos de alguns grupos de PHP do Facebook