



Trabalhando e manipulando datas com a classe DateTime no PHP



Yure Pereira · Follow

Published in Weyes · 9 min read · Jan 13, 2017

🕒 74

🗨 3



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

if($now >= $today) {
    echo $now->format('d/m/Y H:i:s') . ' it greater ' . $today->
        format('d/m/Y H:i:s') . PHP_EOL;
}

if($yesterday <= $tomorrow) {
    echo $yesterday->format('d/m/Y H:i:s') . ' it smaller ' . $tomo
```

```
3 $timeZone1 = new DateTimeZone('Asia/Dubai');
4 $timeZone2 = new DateTimeZone('America/New_York');
5 $timeZone3 = new DateTimeZone('America/Sao_Paulo');
6
7 $date1 = new DateTime('now', $timeZone1);
8 $date2 = new DateTime('now', $timeZone2);
9 $date3 = new DateTime('now', $timeZone3);
10
11 echo 'Asia/Dubai: ' . $date1->format('d/m/Y H:i:s');
12 echo 'America/New_York: ' . $date2->format('d/m/Y H:i:s');
13 echo 'America/Sao_Paulo: ' . $date3->format('d/m/Y H:i:s');
```

Saber trabalhar e manipular datas é algo de muita importância e praticamente imprescindível para o desenvolvimento e manutenção de sistemas computacionais, por isso o objetivo desse artigo é abordar de forma conceitual e prática como podemos fazer isso utilizando à linguagem de programação PHP de forma moderna com a classe DateTime e suas classes auxiliares.

Inicialmente vamos começar falando sobre a classe DateTime, ela é uma classe que foi introduzida à princípio no PHP a partir de sua versão 5.2, ela provém uma interface orientada a objetos que possibilita que possamos trabalhar e fazer manipulação de datas, com ela você poderá desde simplesmente pegar a data atual e fazer sua exibição de forma formatada como também fazer a realização de modificações a manipulações mais avançadas utilizando intervalos de tempo e muito mais.

Sendo assim, vamos começar a utilizá-la fazendo alguns exemplos básicos de sua utilização para ver como ela funciona, para isso inicialmente vamos aprender como podemos criar objetos de instância dessa classe trabalhando com os seus parâmetros de seu método construtor.

Exibição e formatação básica de datas

No primeiro exemplo apresentado simplesmente vou criar uma instância da classe DateTime, suprimindo seus dois parâmetros que são opcionais, onde seu primeiro parâmetro refere-se a passagem de uma string que representará a data a qual você quer obter e caso esse parâmetro não seja passado a data padrão que será obtida virá a ser a data atual do servidor de onde o script PHP está rodando, e seu segundo parâmetro é referente ao fuso horário que será configurado para o objeto de data que irá ser criado e caso ele não seja passado o fuso horário padrão que será utilizado será o atual configurado no servidor mais especificamente o que é configurado no arquivo php.ini.

```
1 <?php
2
3 $currentDateTime = new DateTime();
4
5 echo $currentDateTime->format('d-m-Y H:i:s');
```

current_date_time.php hosted with ❤ by GitHub

view raw

Nesse exemplo acima a data que será obtida será a data atual do servidor por

padrão, onde nele está sendo utilizado o método `format` do objeto da classe `DateTime` que permite que nós possamos formatar a string de data que será apresentada, nesse método é passado uma string que representará o formato da data a qual nós queremos obter, esse formato de string se chama `Format Characters` (Caracteres de formatação). Existem também alguns formatos padronizados pré-definidos como constantes na classe `DateTime` que nós podemos utilizá-los, como poderemos ver no exemplo abaixo algumas dessas constantes em uso.

```
1 <?php
2
3 $currentDateTime = new DateTime();
4
5 //COOKIE: 1, d-M-Y H:i:s T
6 echo $currentDateTime->format(DateTime::COOKIE) . PHP_EOL;
7
8 //RSS: D, d M Y H:i:s O
9 echo $currentDateTime->format(DateTime::RSS) . PHP_EOL;
10
11 //W3C: Y-m-d\TH:i:sP
12 echo $currentDateTime->format(DateTime::W3C);
```

format_current_date.php hosted with ❤ by GitHub [view raw](#)

Para obter a lista completa das constantes padronizadas pré-definidas de formatos de datas da classe `DateTime` acesse o seguinte link [DateTime Constants](#).

Contudo, você também pode criar suas próprias formatações de datas customizadas em um formato específico, utilizando os `Format Characters`, como podemos ver no seguinte exemplo.

```
1 <?php
2
3 $currentDateTime = new DateTime();
4
5 $myFormat = 'Y-m-d H:i:s';
6 $myFormat2 = 'd-m-Y H:i:s';
7 $myFormat3 = 'd-m-Y';
8
9 echo $currentDateTime->format($myFormat) . PHP_EOL;
10
11 echo $currentDateTime->format($myFormat2) . PHP_EOL;
12
13 echo $currentDateTime->format($myFormat3);
```

my_formats_date.php hosted with ❤ by GitHub [view raw](#)

Passando datas específicas e obtendo datas relativas

Com a classe `DateTime` também podemos passar uma determinada data a qual pretendemos usar, podemos fazer isso passada essa data no seu primeiro parâmetro de seu construtor no formato [Date and Time Formats](#). Isso é muito usado quando estamos trabalhando com datas obtidas de registros em banco de dados, onde seu formato é YYYY-MM-DD HH:MM:SS.

```
1 <?php
2
3 $myDateTime = new DateTime('2000-10-12 09:01:10');
4
5 echo $myDateTime->format('d-m-Y') . PHP_EOL;//12-10-2000
6 echo $myDateTime->format('d-m-Y H:i:s');//12-10-2000 09:01:10
```

my_specific_date.php hosted with ❤ by GitHub [view raw](#)

Ou também podemos definir uma determinada data e tempo usando os métodos `setDate` e `setTime` ao invés de passar nossa data pelo método construtor da classe `DateTime`, porém dessa forma teríamos mais trabalho caso nossa data esteja em formato de string, pois temos que passar cada valor da data e tempo individualmente como parâmetro, como podemos ver a seguir.

```
1 <?php
2
3 $dateTime = new DateTime();
4
5 //Date
6 $year = 2000; $month = 10; $day = 12;
```

```

7 //Time
8 $hour = 9; $minute = 1; $second = 10;
9
10 $dateTime->setDate($year, $month, $day);
11 $dateTime->setTime($hour, $minute, $second);
12
13 echo $dateTime->format('d/m/Y H:i:s');

```

using_setdate_and_settime.php hosted with ❤ by GitHub [view raw](#)

Além disso, também podemos definir uma determinada data relativamente à data atual passado uma string de *Relative Formats* como veremos no seguinte exemplo.

```

1 <?php
2
3 $now = new DateTime('now');//Valor padrão
4 $today = new DateTime('today');
5 $yesterday = new DateTime('yesterday');
6 $tomorrow = new DateTime('tomorrow');
7
8 echo 'Now: ' . $now->format('d-m-Y H:i:m') . PHP_EOL;
9 echo 'Today: ' . $today->format('d-m-Y H:i:m') . PHP_EOL;
10 echo 'Yesterday: ' . $yesterday->format('d-m-Y H:i:m') . PHP_EOL;
11 echo 'Tomorrow: ' . $tomorrow->format('d-m-Y H:i:m');

```

get_relative_formats.php hosted with ❤ by GitHub [view raw](#)

Como podemos ver no exemplo acima, é possível pegar as datas relativas a data atual, como a data de ontem (*Yesterday*), hoje (*Today*), agora (*Now*), amanhã (*Tomorrow*) e algumas outras datas com *Relative Formats* passando os como primeiro parâmetro do construtor da classe *DateTime*. Sendo que quando instanciamos essa classe e não passamos esse primeiro parâmetro, o valor *default* (padrão) que será assumido será o *Relative Format Now*, que a princípio aparenta ser semelhante ao *Relative Format Today*, pois suas datas são as mesmas, porém a diferença entre essas datas está no time (tempo) delas que não são os mesmos, onde o time da data Now é o horário atual do servidor de onde o script PHP está em execução e o time do Today é o horário inicial desse dia 00:00:01.

Usando as strings de *Relative Format* também é possível obter datas relativas a partir de uma determinada adição ou subtração de dias, semanas, meses ou anos, como se segue no seguinte exemplo.

```

1 <?php
2
3 $myDate1 = new DateTime('+ 3 days');
4 $myDate2 = new DateTime('- 1 week');
5 $myDate3 = new DateTime('+ 2 months');
6 $myDate4 = new DateTime('+ 10 years');
7
8 echo $myDate1->format('d/m/Y H:i:s') . PHP_EOL;
9 echo $myDate2->format('d/m/Y H:i:s') . PHP_EOL;
10 echo $myDate3->format('d/m/Y H:i:s') . PHP_EOL;
11 echo $myDate4->format('d/m/Y H:i:s');

```

get_date_from_relative_formats.php hosted with ❤ by GitHub [view raw](#)

Lembrando também que ao usar essa adição ou subtração de períodos usando *Relative Formats*, devemos levar em consideração a quantidade numérica que estamos usando, se ela está no singular ou plural, pois ao usar as strings de período de tempo devemos coloca-las na forma correta, por exemplo '+ 1 day', '- 3 days', '+ 1 month', '+ 10 months' e assim por diante.

Determinado o fuso horário de uma data

Determinar o fuso horário a ser utilizando é de grande importância dependendo da localização dos usuários os quais irão acessar nosso sistema, então para que possamos fazer essa configuração em uma determinada data a qual estamos utilizando, simplesmente teremos que passar um objeto da classe *DateTimeZone*, como segundo parâmetro no momento de instanciar à classe *DateTime*, onde no objeto da classe *DateTimeZone* nós configuraremos qual fuso horário iremos utilizar, isso será feito passado uma string com o nome do fuso horário no seu método construtor como primeiro e único parâmetro, como veremos no exemplo a seguir.

```

1 <?php
2
3 $timezone1 = new DateTimeZone('Asia/Dubai');
4 $timezone2 = new DateTimeZone('America/New_York');
5 $timezone3 = new DateTimeZone('America/Sao_Paulo');
6
7 $date1 = new DateTime('now', $timezone1);
8 $date2 = new DateTime('now', $timezone2);
9 $date3 = new DateTime('now', $timezone3);
10
11 echo 'Asia/Dubai: ' . $date1->format('d/m/Y H:i:s') . PHP_EOL;
12 echo 'America/New_York: ' . $date2->format('d/m/Y H:i:s') . PHP_EOL;
13 echo 'America/Sao_Paulo: ' . $date3->format('d/m/Y H:i:s');

```

[setting_timezone.php](#) hosted with ❤ by GitHub [view raw](#)

Como poderemos ver no exemplo anterior ao rodar esse script serão apresentadas três datas e horas diferentes cada uma delas correspondendo ao seu fuso horário configurado.

Caso não telhamos definido o fuso horário no momento da criação de nosso objeto da classe *DateTime* ou queremos alterá-lo, também poderemos defini-lo ou alterá-lo utilizando o método *setTimezone* a qualquer momento, da seguinte forma que veremos no exemplo abaixo.

```

1 <?php
2
3 $myTimeZone = new DateTimeZone('America/Sao_Paulo');
4 $myDate = new DateTime('+2 months', $myTimeZone);
5
6 echo $myDate->format('Y-m-d H:i:s A') . PHP_EOL;
7
8 $myNewTimeZone = new DateTimeZone('Asia/Dubai');
9 $myDate->setTimezone($myNewTimeZone);
10
11 echo $myDate->format('Y-m-d H:i:s A');

```

[changes_timezone.php](#) hosted with ❤ by GitHub [view raw](#)

Agora se você quiser configurar outros *Timezones* (Fuso horários), você pode acessar a lista completa com todos os *Timezones* suportados pelo PHP no seguinte link [List of Supported Timezones](#).

Comparação entre duas datas

Utilizando a classe *DateTime* para manipular datas, possibilita que nós possamos comparar datas de forma muito simples utilizando estruturas condicionais com expressões booleanas.

```

1 <?php
2
3 $now = new DateTime();
4 $today = new DateTime('today');
5 $yesterday = new DateTime('yesterday');
6 $tomorrow = new DateTime('tomorrow');
7
8 if ($now >= $today) {
9     echo $now->format('d/m/Y H:i:s') . ' is greater ' . $today->format('d/m/Y H:i:s') . PHP_EOL;
10 }
11
12 if ($yesterday <= $tomorrow) {
13     echo $yesterday->format('d/m/Y H:i:s') . ' is smaller ' . $tomorrow->format('d/m/Y H:i:s');
14 }
15
16 if ($today != $tomorrow) {
17     echo $today->format('d/m/Y H:i:s') . ' is different from ' . $tomorrow->format('d/m/Y H:i:s');
18 }

```

[comparison_of_two_dates.php](#) hosted with ❤ by GitHub [view raw](#)

Como podemos ver no exemplo anterior, é possível comparar se uma data é maior ou menor que outra simplesmente usando expressões booleanas comparando nossos objetos da classe da *DateTime*.

Agora se quiséssemos pegar o intervalo de tempo de diferença entre duas datas com mais precisão, podemos usar o método *diff* da classe *DateTime* a qual nos retorna um objeto da classe *DateInterval* com todos os dados referentes a diferença entre as duas datas que estão sendo usadas na comparação.

```

1 <?php
2
3 $dateTime1 = new DateTime('2016-12-01 12:10:15');
4 $dateTime2 = new DateTime('2017-01-08 08:18:09');
5
6 $dateInterval = $dateTime1->diff($dateTime2);
7
8 print_r($dateInterval);

```

taking_interval_between_dates.php hosted with ❤ by GitHub [view raw](#)

O exemplo anterior irá resultar no seguinte objeto que está sendo apresentado na imagem abaixo, onde podemos ver que nele contém todos os dados como quantidade de meses, dias, horas e etc., referentes as duas datas.

```

DateInterval Object
(
    [y] => 0
    [m] => 1
    [d] => 6
    [h] => 20
    [i] => 7
    [s] => 54
    [weekday] => 0
    [weekday_behavior] => 0
    [first_last_day_of] => 0
    [invert] => 0
    [days] => 37
    [special_type] => 0
    [special_amount] => 0
    [have_weekday_relative] => 0
    [have_special_relative] => 0
)

```

Objeto DateInterval

O objeto da classe DateInterval que é retornado também nos fornece o método *format*, que possibilita que nós possamos formatar e exibir o período de tempo que é retornado, como pode-se ver no exemplo abaixo.

```

1 <?php
2
3 $dateTime1 = new DateTime('2016-12-01 12:10:15');
4 $dateTime2 = new DateTime('2018-01-08 08:18:09');
5 $interval = $dateTime1->diff($dateTime2);
6
7 echo $interval->format('%y anos') . PHP_EOL;
8 echo $interval->format('%m meses') . PHP_EOL;
9 echo $interval->format('%d dias') . PHP_EOL;
10 echo $interval->format('%h horas') . PHP_EOL;
11 echo $interval->format('%i minutos') . PHP_EOL;
12 echo $interval->format('%s segundos') . PHP_EOL;
13 echo $interval->format('%h:%i:%s');

```

using_format_dateinterval.php hosted with ❤ by GitHub [view raw](#)

Manipulando datas

Outra funcionalidade que a classe DateTime nos fornece é a manipulação de datas, possibilitando que nós possamos fazer a adição ou subtração de intervalos de tempo em nossas datas pré-definidas no momento de sua criação, como veremos no exemplo a seguir.

```

1 <?php
2
3 $dateTime = new DateTime('2016-12-01 00:00:01');
4 echo $dateTime->format('d/m/Y H:i:s') . PHP_EOL;
5
6 //Adicionado 40 dias a nossa data
7 $dateTime->add(new DateInterval('P40D'));
8 echo $dateTime->format('d/m/Y H:i:s');

```

add_interval_date.php hosted with ❤ by GitHub [view raw](#)

No exemplo anterior usamos o método *add* para fazer a adição de 40 dias à nossa data, e como se pode observar nesse exemplo foi preciso passar um objeto da classe DateInterval que representa essa quantidade de dias que serão adicionadas a nossa data, onde também podemos observar que essa representação é feita a partir de uma string de *Period Designators* que é passada no primeiro parâmetro do construtor da classe DateInterval.

Agora no próximo exemplo será mostrado como fazer a subtração de um determinado intervalo de tempo que é algo tão simples como fazer a adição.

```

1 <?php
2
3 $dateTime = new DateTime('2016-12-01 00:00:01');
4 echo $dateTime->format('d/m/Y H:i:s') . PHP_EOL;
5
6 //Subtraindo 3 anos e 2 meses de nossa data
7 $dateTime->sub(new DateInterval('P3Y2M'));
8 echo $dateTime->format('d/m/Y H:i:s');

sub_interval_date.php hosted with ❤ by GitHub

```

[view raw](#)

Nesse exemplo foi feito a subtração de três anos e dois meses de nossa data com a utilização do método `sub`, de forma semelhante a como foi feito quando utilizamos o método `add` passando uma instância da classe `DateInterval` onde definimos o intervalo de tempo a ser subtraído a partir de uma string de *Period Designators* que agora veremos como ela funciona.

Para forma essa string de designação de período devemos colocar a primeira letra de nossa string sempre iniciando com a letra P que vez de *period* (período) posteriormente deveremos definir um valor inteiro com a duração do período seguido por um *Period Designator* (Designador de Período). A seguinte tabela mostra todos os *Period Designators* que podemos usar e suas devidas descrições.

Period Designator	Description
Y	years
M	months
D	days
W	weeks. These get converted into days, so can not be combined with D.
H	hours
M	minutes
S	seconds

Tabela de Period Designators

A seguir temos alguns exemplos de como forma essa string para definir períodos de tempo:

- P40D: Período de 40 dias
- P1Y20M3D: Período de 1 ano, 20 meses e 3 dias
- P3M10D: Período de 3 meses e 10 dias

Agora se quiséssemos definir tempo em nossa string deveremos preceder esse tempo pela letra T que vez de time (tempo) como veremos nos seguintes exemplos:

- P1YT10H55S: Período de um ano e 10 horas e 55 segundos
- PT5H30M20S: Período de tempo de 5 horas, 30 minutos e 20 segundos.
- P1Y4M10DT2H30M50S: Período de 1 ano, 4 meses, 10 dias, 2 horas, 30 minutos e 50 segundos.

Esses exemplos ficariam da seguinte forma em formato de código utilizando a classe `DateInterval`:

```

1 <?php
2
3 $dateTime = new DateTime('2016-12-01 00:00:01');
4 echo $dateTime->format('d/m/Y H:i:s'), PHP_EOL;
5
6 $dateTime->add(new DateInterval('P1YT10H55S'));
7 echo $dateTime->format('d/m/Y H:i:s'), PHP_EOL;
8
9 $dateTime->sub(new DateInterval('PT5H30M20S'));
10 echo $dateTime->format('d/m/Y H:i:s'), PHP_EOL;
11
12 $dateTime->add(new DateInterval('P1Y4M10DT2H30M50S'));
13 echo $dateTime->format('d/m/Y H:i:s');

using_period_designators.php hosted with ❤ by GitHub

```

[view raw](#)

A classe `DateInterval` também fornece outra forma de definirmos o período de tempo a qual queremos usar, utilizando seu método estático

`createFromDateString` que permite que nós possamos criar nosso período de tempo a partir de *Relative Formats* como já utilizamos anteriormente, porém nós passamos essa string como parâmetro do construtor da classe `DateTime`, mas agora passaremos essa string para o método `createFromDateString`. Como veremos no exemplo a seguir.

```
1 <?php
2
3 $dateTime = new DateTime('2016-12-01 00:00:01');
4 echo $dateTime->format('d/m/Y H:i:s'), PHP_EOL;
5
6 //P1Y10H55S
7 $dateTime->add(DateInterval::createFromString('1 year + 10 hours + 55 seconds'));
8 echo $dateTime->format('d/m/Y H:i:s'), PHP_EOL;
9
10 //PT5H30M20S
11 $dateTime->sub(DateInterval::createFromString('5 hours + 30 minutes + 20 seconds'));
12 echo $dateTime->format('d/m/Y H:i:s'), PHP_EOL;
13
14 //P1Y4M10DT2H30M50S
15 $dateTime->add(DateInterval::createFromString('1 year + 4 months + 10 days + 2 hours + 30 mi
16 echo $dateTime->format('d/m/Y H:i:s');
```

using_createfromdatedstring.php hosted with ❤ by GitHub [view raw](#)

Modificando datas

Nós podemos fazer modificações de adição ou subtração de períodos de tempo em nossas datas a partir do método `modify` da classe `DateTime`, isso de princípio aparenta ser semelhante aos métodos `add` e `sub`, mas usando um método só ou invés de dois, porém existe uma diferença em relação à esses métodos e essa diferença vem do fato que o tipo de entrada de seus parâmetros são diferentes, onde nos métodos `add` e `sub`, nós temos que passar um objeto da classe `DateInterval` para podemos fazer a manipulação de nossas datas, contudo no método `modify` o que passamos é uma string de *Relative Formats* para podemos definir qual período de tempo queremos usar para modificarmos nossa data, assim como veremos no exemplo a seguir.

```
1 <?php
2
3 $dateTime = new DateTime('2016-12-01 00:00:01');
4 echo $dateTime->format('d/m/Y H:i:s'), PHP_EOL;
5
6 //Adicionando dez dias a nossa data
7 $dateTime->modify('10 days');
8 echo $dateTime->format('d/m/Y H:i:s'), PHP_EOL;
9
10 //Subtraindo dez dias de nossa data
11 $dateTime->modify('-10 days');
12 echo $dateTime->format('d/m/Y H:i:s'), PHP_EOL;
13
14 //Adicionando 1 anos, 4 meses e 25 dias a nossa data
15 $dateTime->modify('1 year +4 months +25 days');
16 echo $dateTime->format('d/m/Y H:i:s');
```

using_modify.php hosted with ❤ by GitHub [view raw](#)

Conclusão

No decorrer desse artigo foi possível ver o quanto a classe `DateTime` pode nos ajudar quando precisamos manipular datas em nosso sistema. Vimos como é fácil criar objetos dessa classe e fazer a sua formatação de apresentação utilizando o método `format` com a string de *Format Characters*, fizemos a configuração de fuso horário de nossas datas utilizando a classe `DateTimeZone`.

Posteriormente também vimos como podemos fazer comparação entre datas utilizando expressões booleanas de forma muito simples, e também como saber qual é a diferença de intervalo de tempo de duas datas usando o método `diff` que nos retorna um objeto da classe `DateInterval` com todos esses dados para que possamos analisá-los.

Depois disso aprendemos como trabalhar com manipulação de datas fazendo adição e subtração de período de tempo usando os métodos `add` e `sub` em conjunto com à classe `DateInterval`, onde com ela definímos o período de tempo usando *Period Designators* ou se não usando *Relative Formats* com o método estático `createFromString` dessa classe.

E para finalizar vimos como fazer modificações em nossos objetos de datas usando o método `modify` de tal forma como os métodos add e sub nos permitiu, porém ao invés de passamos um objeto da classe `DateInterval`, simplesmente passamos como parâmetro uma string de `Relative Formats`.

Contudo, ainda existem muitas peculiaridades que podem ser estudadas em relação à manipulação de datas utilizando à classe `DateTime` que não foram abordadas nesse artigo, e caso você queira aprender mais sobre elas acesse sua documentação completa no seguinte link [PHP Data e Hora](#).

• • •

Se gostou do artigo dê like ❤️ e me siga, para acompanhar novos artigos. Mas se gostou muito, compartilha que ficarei muito grato. E sinta-se à vontade para fazer comentários, sobre dúvidas, críticas construtivas (ou não), melhorias e agradecimentos.

PHP Date Object Oriented Programming Web Development

74 3



Written by Yure Pereira

285 Followers · Editor for Weyes

Follow



System Analyst na Weyes Technology Solutions, Contato: franciscoyurep@gmail.com

More from Yure Pereira and Weyes

```
3 $keyarray = array(
4   'item_1',
5   'item_2',
6   'item_3',
7   'item_4',
8   'item_5',
9   'item_6',
10  );
11
12 foreach ($keyarray as $keyItem => $itemValue)
13   echo "Item index: " . $keyItem . ", Value
14     " . $itemValue . PHP_EOL;
15
16
17
```

Yure Pereira in Weyes

Trabalhando e manipulando Arrays no PHP

Quando estamos desenvolvendo aplicações para internet ou para quaisquer outras...

11 min read · Feb 13, 2017

49 4

Trabalhando e manipulando sessão no PHP



Yure Pereira in Weyes

Trabalhando e manipulando sessão no PHP

Aprenda a usar sessão no PHP de forma prática é fácil para aplica-la em suas...

8 min read · Apr 16, 2017

135 2





Yure Pereira in Weyes

Usando a função each do JQuery

Aprendendo a usar a função each do JQuery para fazer iterações em objetos.

5 min read · Apr 8, 2017

137 2

```
path\filename_1.jpg';  
path\filename_1.php';  
função include_once;  
(Spath1);  
no Spath1 não será incluído, pois ele  
(Spath1);  
ocorre include_once;  
(Spath2);  
no Spath2 não será incluído, pois ele  
});  
});
```

Yure Pereira in Weyes

Fazendo inclusão de arquivos de forma correta no PHP

Quando trabalhamos com desenvolvimento de sites ou sistemas web com PHP,...

6 min read · Feb 26, 2017

104 3

+

See all from Yure Pereira

See all from Weyes

See all from Yure Pereira

See all from Weyes

Recommended from Medium



Jacob Bennett in Level Up Coding

Use Git like a senior engineer

Git is a powerful tool that feels great to use when you know how to use it.

4 min read · Nov 15, 2022

6.6K 68



The PyCoach in Artificial Corner

You're Using ChatGPT Wrong! Here's How to Be Ahead of 99% of...

Master ChatGPT by learning prompt engineering.

7 min read · Mar 17

25K 452

+

Lists



General Coding Knowledge

20 stories · 25 saves



It's never too late or early to start something

10 stories · 7 saves



Coding & Development

11 stories · 13 saves



Stories to Help You Grow as a Software Developer

19 stories · 149 saves



Unbecoming

10 Seconds That Ended My 20 Year Marriage

It's August in Northern Virginia, hot and humid. I still haven't showered from my...

4 min read · Feb 16, 2022



Aleid ter Weel in Better Advice

10 Things To Do In The Evening Instead Of Watching Netflix

Device-free habits to increase your productivity and happiness.

5 min read · Feb 15, 2022

51K 804

22K 362



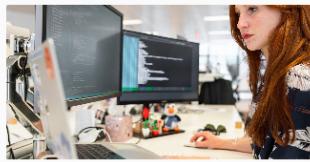
Love Sharma in Dev Genius

System Design Blueprint: The Ultimate Guide

Developing a robust, scalable, and efficient system can be daunting. However,...

9 min read · Apr 20

4.3K 32



The Coding Diaries in The Coding Diaries

Why Experienced Programmers Fail Coding Interviews

A friend of mine recently joined a FAANG company as an engineering manager, and...

5 min read · Nov 2, 2022

4.3K 92

See more recommendations

[Help](#) [Status](#) [Writers](#) [Blog](#) [Careers](#) [Privacy](#) [Terms](#) [About](#) [Text to speech](#) [Teams](#)