www.phparch.com

November 2021
Volume 20 - Issue 11

php[architect]

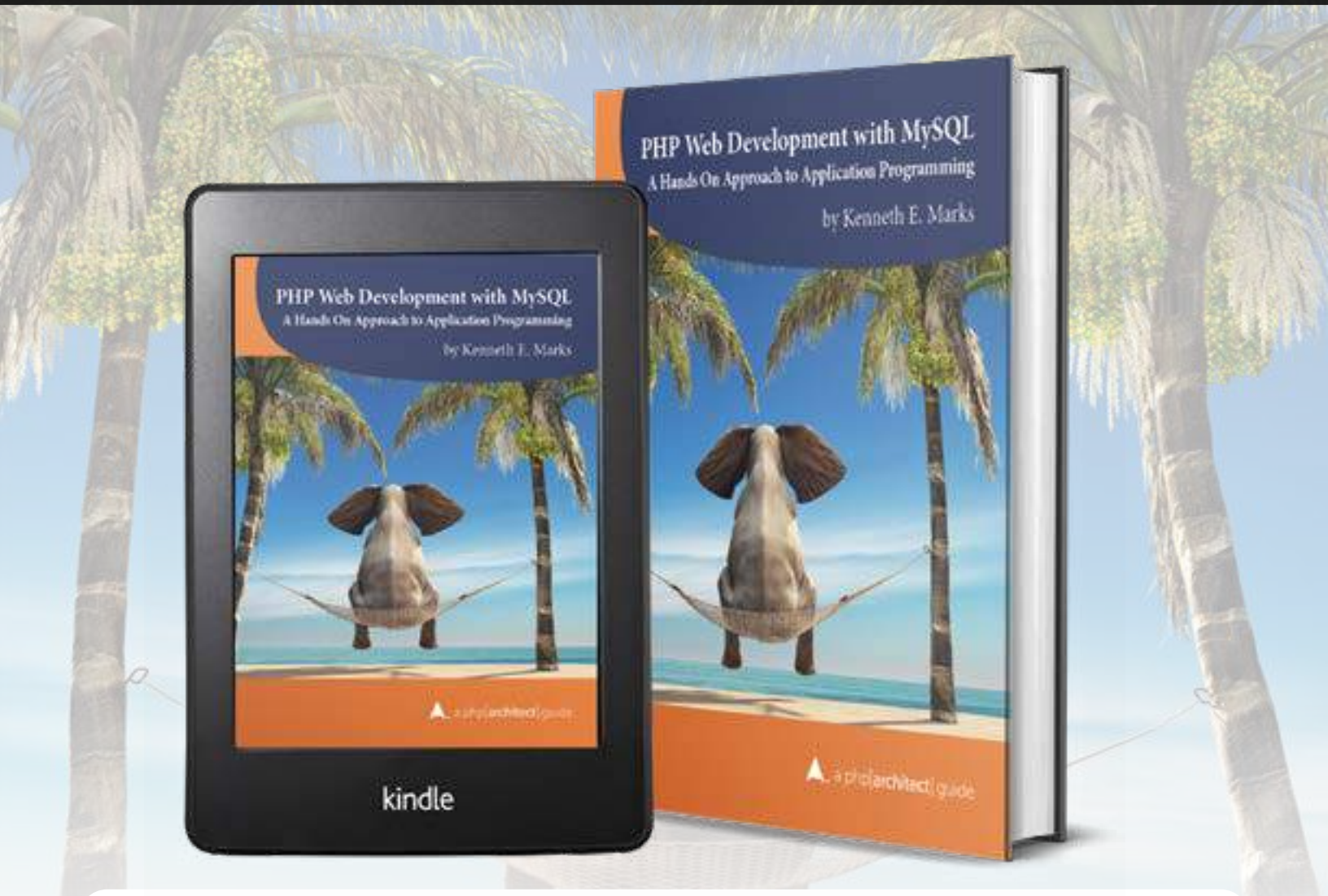
The Art of Data

Libsodium

Persisting objects to a relational database: Active Record Pattern

ALSO INSIDE

Free Sample Article



Learn how to build dynamic and secure websites.

The book also walks you through building a typical Create-Read-Update-Delete (CRUD) application. Along the way, you'll get solid, practical advice on how to add authentication, handle file uploads, safely store passwords, application security, and more.

Available in Print+Digital and Digital Editions.

Purchase Your Copy

https://phpa.me/php-development-book

Compounding Interest

Oscar Merida

Last month, we looked at how to output an amortization table to see how compounding interest affects some amount of money. Printing out such a table assumes we have a certain set of inputs. Moreover, it's tedious, even simple for a computer to do. Let's look at other ways to calculate the time value of money.

Recap

Our amortization table last month assumed we knew the principal amount, the interest rate, and the compounding period. However, there are other interesting questions we can ask when looking to make a financial decision. Last month's column posed the following three scenarios.

1. You are offered an annuity that will pay $17,000 per year for seven years (the first payment will be made today). If you feel that the appropriate discount rate is 11%, what is the annuity worth to you today?
2. You plan to borrow $189,000 now and repay it in equal monthly installments. If the annual interest rate is 6.5%, how much will your monthly payments be over 15 years?
3. What annual interest rate do you need to earn if you invest $1,500 today and have $6,000 in 10 years?

Investigation

While we could brute force the solutions, likely using a spreadsheet to do so. With a dash of research, you'll find each has a known solution and formula to calculate the answer. As we look at each one, some terms we'll use include:

- `P` Present value = what something is worth today at time zero.
- `A` Annuity value = a fixed, recurring amount that, despite its name, repeats every time period (year, month, etc.)
- `F` Future Value = a single cash value at some point after today.

And we'll usually have an interest rate (`i`) for some number of periods `n`. To figure out which formula we should use, we need only identify our knowns and unknowns.

Present Value of Annuity

> *You are offered an annuity that will pay $17,000 per year for seven years (the first payment will be made today). If you feel that the appropriate discount rate is 11%, what is the annuity worth to you today?*

In this case, we know the annuity `A`, the period `n` in years, and the annual interest rate `i`. We want to know the present value of such an annuity. With this question, we're trying to find what lump sum today has the same spending power as the seven annuities.

This is one of the more complicated formulas we'll see.

Figure 1.

$$P = A \cdot \frac{(1+i)^n - 1}{i \cdot (1+i)^n}$$

We can express it as a function like so:

```
function presentValueOfAnnuity(float $annuity,
                               float $rate,
                               int $periods) : float {
    $exp = pow( 1 + $rate, $periods);
    $p = $annuity * ($exp - 1)/($rate * $exp);
    return $p;
}
```

Calling it with our scenario should give us an answer.

```
echo presentValueOfAnnuity(17000, 0.11, 7);
80107.336498694
```

What does this tell us? It means earning $17k per year in this scenario is equivalent to having about $80,107 today. Knowing this can come in handy if we're looking at another way to earn money. When comparing more than one annuity, each with a different payout, interest rate, or number of periods, we can calculate the present value of each. The one with the highest is the one that earns us the most money over its lifetime. You can use the present value to also compare the terms of a loan. Loans are nothing more than annuities where we have to pay a fixed amount every period to someone else.

Value of Annuity

> *You plan to borrow $189,000 now and repay it in equal monthly installments. If the annual interest rate is 6.5%, how much will your monthly payments be over 15 years?*

This scenario is the reverse of the previous one. In this case, we know the present value `P`, the interest rate `r`, and the lifetime of the loan `n`. We can use the formula in Figure 2 *Note: the problem statement was unsolvable as presented last month, so assume some realistic loan term.*

Figure 2.

$$A = P \cdot \frac{i(1+i)^n}{(1+i)^n - 1}$$

Listing 2.

```
function annuityGivenPresent(float $present,
                             float $interest,
                             int $periods) : float {
    $numerator = $interest * pow (1 + $interest, $periods);
    $divisor = pow(1 + $interest, $periods) - 1;

    $a = $present * $numerator / $divisor;
    return $a;
}
```

Before we can calculate our values, we have to convert our annual interest rate of 6.5% into an effective monthly rate of `0.065 / 12 = 0.0054166`. Also, we need to change the number of years into months of our loan `12 * 15 = 180` monthly payments. Of course, we can have the computer do that for us:

Listing 3.

```
$annualInterest = 0.065;
$monthlyInterest = $annualInterest / 12;

$years = 15;
$monthlyPayments = $years * 12;

$monthly = annuityGivenPresent(
                                189000,
                                $monthlyInterest,
                                $monthlyPayments);
echo number_format($monthly);
```

Our code tells us we should expect to pay approximately $1,646 per month to pay off our loan.

Rate of Return

> *What annual interest rate do you need to earn if you invest $1,500 today and have $6,000 in 10 years?*

In our final scenario, our unknown variable is the interest rate. We know we have $1,500 and want to grow that to $6,000 in a decade. We can use the formula in Figure 3 to get a future value (F) given a present one (P).

Figure 3.

$$F = P(1+i)^n$$

We'll have to do a bit of algebra to solve for `i`. You should end up with a formula like the one in Figure 4.

Figure 4.

$$i = \left(\frac{F}{P}\right)^{\frac{1}{n}} - 1$$

Now we can write that as an equation. Our algebra indicates we need to find the n-th root in our formula. In math, the n-th root of a number is the same as raising it to the 1/n-th power. Why do we need to know that? While PHP has a square root function, we have to use `pow()`[1] with a fractional exponent to calculate this root.

```
function targetRate(float $Present,
                    float $Future,
                    int $periods) : float {
    $rate = pow($Future / $Present, 1/$periods) - 1;
    return $rate;
}
```

Now, we could assume our interest compounds annually to find our desired rate. But, most of the time, interest will compound monthly or even daily. Let's account for that. Ten years means we have `12 * 10 = 120` months to earn interest.

1 `https://php.net/pow`

We can calculate the nominal annual rate once we know the effective monthly interest rate.

We can find our monthly rate by calling our function:

```
echo targetRate(1500.00, 6000.00, 120);
// 0.011619440301923
```

That means we need to earn at least 1.16% per month to meet our target. Multiplying by 12 tells us we need to find an investment advertising an Annual Percentage Rate (APR) of 13.94%. That's a nominal rate. The effective annual rate we desire is calculated with:

```
(1 + 0.016) ^ 12 = 1.1487
```

So, we need to earn almost 15% each year. Why do APRs exist? It's a marketing trick, effectively. If you're borrowing money, a lender can advertise one annual rate. Then, the lender earns a higher effective rate by compounding it more frequently than once per year.

Next Month

Let's look at one more economic puzzle to round out the year.

You've decided to produce a rules supplement for a popular tabletop role-playing game. Since this game is old-school, you'll offer printed copies for sale at $16.

Estimated Production Costs are:

1. Writing and Editing: $825
2. Art and Layout: $225
3. Marketing: $300

Each printed copy costs $3.75 to produce. Average shipping costs are $7.40.

How many copies do you have to sell to recover your initial expenditures?

Extra Credit:

If you offer a digital-only option for $12, how many digital and how many print units do you need to sell before you make a profit? Assume 70% of your sales will be digital.

Some Guidelines And Tips

The puzzles can be solved with pure PHP. No frameworks or libraries are required.

- *Each solution is encapsulated in a function or a class, and I'll give a sample output to test your solution against.*
- *You're encouraged to make your first attempt at solving the problem without using the web for clues.*
- *Refactoring is encouraged.*
- *I'm not looking for speed, cleverness, or elegance in the solutions. I'm looking for solutions that work.*
- *Go ahead and try multiple solutions if you like.*
- *PHP's interactive shell (`php -a` at the command line) or 3rd party tools like PsySH[2] can be helpful when working on your solution.*
- *To keep solutions brief, we'll omit the handling of out-of-range inputs or exception checking.*

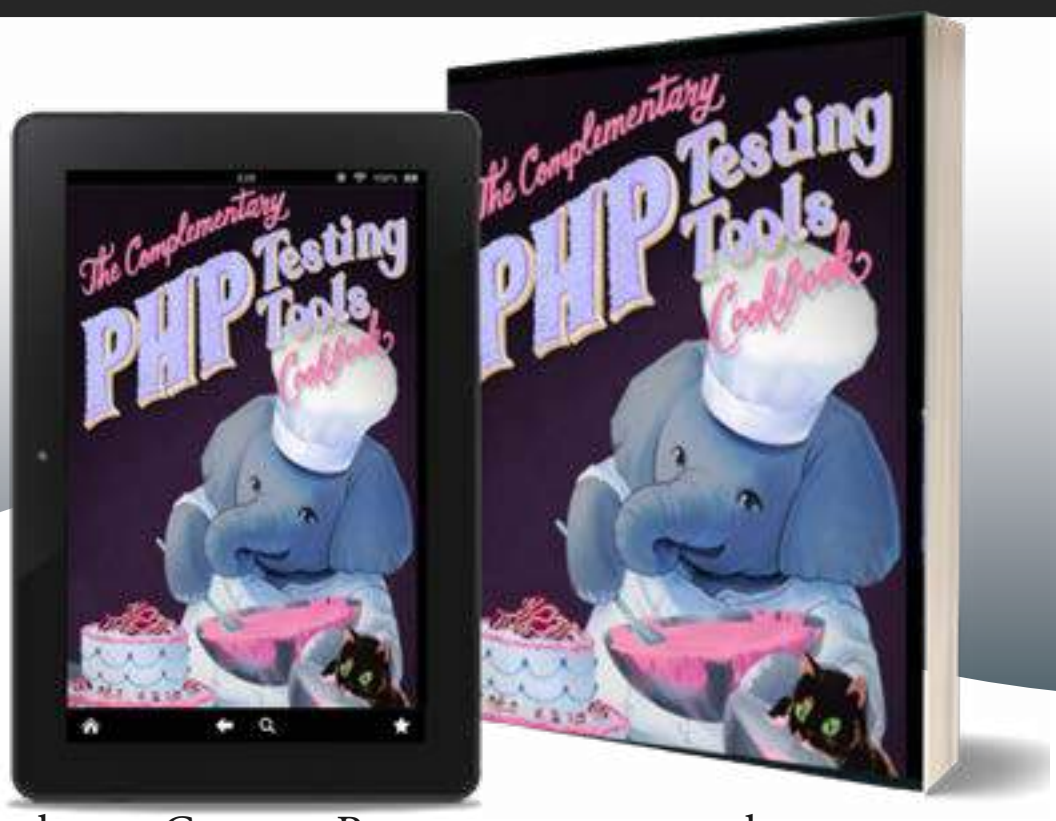
Related Reading

- *PHP Puzzles: Time Value of Money* by Oscar Merida, October 2021. http://phpa.me/puzzle-oct-2021
- *PHP Puzzles: Sending and Receiving Polybius Ciphers* by Oscar Merida, May 2021. http://phpa.me/puzzles-may-2021
- *PHP Puzzles: Factorials* by Sherri Wheeler, May 2020. https://phpa.me/puzzles-may-20

Oscar Merida has been working with PHP since version 4 was released and is constantly learning new things about it and still remembers installing an early version on Apache. When not coding or writing, he enjoys RPGs, soccer, and drawing. @omerida

2 *PsySH: https://psysh.org*

turnoff.us | Daniel Stori Shared with permission from the artist



Learn how a Grumpy Programmer approaches improving his own codebase, including all of the tools used and why.

The Complementary PHP Testing Tools Cookbook is Chris Hartjes' way to try and provide additional tools to PHP programmers who already have experience writing tests but want to improve. He believes that by learning the skills (both technical and core) surrounding testing you will be able to write tests using almost any testing framework and almost any PHP application.

Available in Print+Digital and Digital Editions.

Order Your Copy

phpa.me/grumpy-cookbook