zend
by Perforce

2021 PHP Landscape Report

Top PHP Technologies, Strategies, and Trends

www.zend.com

Introduction

Executive Summary

This report looks at the results of an industry survey of PHP development professionals and covers hot topics in PHP development, as well as PHP version and development technology adoption.

Foreword

To the reader,

PHP celebrated its 25th birthday this past year. It has come a long way from its roots as a template parsing system that would run C functions as it encountered tags. Now at version 8.0, PHP offers full object-oriented capabilities, an opt-in strict-typing system, a Just-In-Time (JIT) compiler, extensions for just about any data-base in use today, robust date and time processing tools that put other languages to shame, and far, far more. It powers content management systems, eCommerce, customer rights management, and resource manage-ment systems. APIs feeding mobile applications and communicating between businesses rely on PHP.

It's not hyperbole to say that PHP powers the web.

I began programming in PHP in late 2000. The language was free and open source, which meant I didn't need to pay for licenses in order to learn the language, nor to run a web server with it. Even then, PHP offered a ton of connectivity and integration options with databases and mail servers and more that gave instant value and ease of use to developers.

When I joined Zend in 2005, Object Oriented Programming (OOP) was just starting to take hold and stake its place in the language. The capabilities added in PHP 5, and expanded in PHP 7, helped power and create an ecosystem of libraries and frameworks that allow developers to be instantly productive and develop new features quickly.

The language today is still recognizable as the language from 20 years ago, though it offers far more and better features to allow me as a developer to write flexible, maintainable, and correct software. I can even use the language to write web servers — a possibility I'd never considered 20 years ago.

While you can still get started writing PHP just as easily as you did 20 years ago, the challenges of creating a PHP application have grown, in large part because we are creating more complex and

sophisticated applications. Security and testing have become huge concerns that will only continue to grow as we learn about the ramifications of data privacy. The types of applications have shifted from huge monoliths to APIs and microservices, which have different deployment needs and benefit from new developments in the ecosystem such as containers, async capabilities, and JIT compilation.

It's truly an exciting time to be a PHP developer and I'm eager to see what the new capabilities provided by PHP 8 will bring us in the next few years.

Enjoy the report,

Matthew Weier O'Phinney

Table of Contents

5 About the Survey

6 PHP Development Priorities

11 PHP Version and Upgrade Statistics

13 Development Technologies

16 Asynchronous Programming

17 PHP Outlook for 2021

18 Resources

About the Survey

The 2021 PHP Landscape Report from Zend by Perforce is based on the results of an anonymous survey conducted between the months of September and December of 2020.

The survey, which was promoted via social media and email, focused primarily on how PHP development teams are working with the language, their views on current trends within PHP development, and the technologies they use to develop PHP applications. The survey received a total of 670 responses.

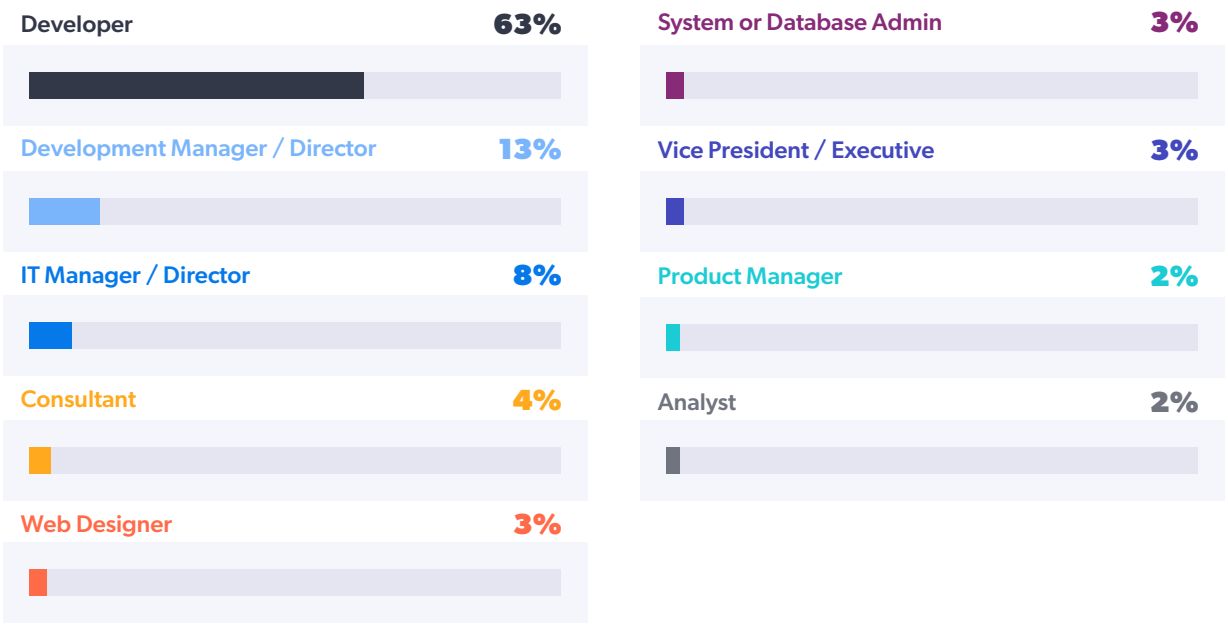
Reporting & Analysis Methodology

To help increase report readability, we removed outliers from graphs. For the purposes of this report, we defined outliers to be responses under 1%. Additionally, we rounded response percentage values to the nearest full percentage point. Year on year comparisons, where applicable, used data from the Zend 2019 PHP Landscape Survey.

Respondent Demographics

Respondents to the survey were primarily developers, who represented 63% of respondents. Development managers or directors made up the next largest segment at 13% of respondents. Rounding out the top three, IT managers or directors comprised 8% of respondents.

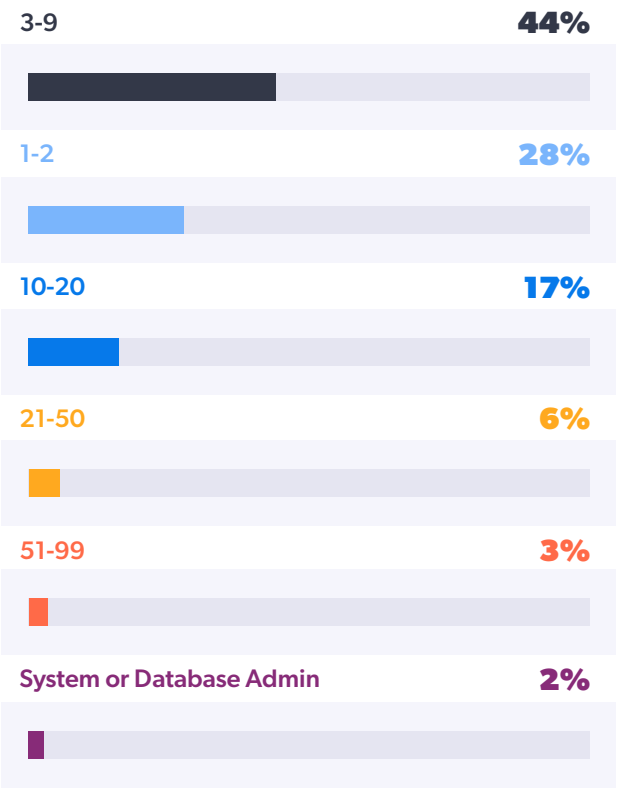
Which job title matches your current role?



Development Team Size

In our next question, we asked respondents to share the size of their development team. The largest segment of respondents, at 44%, reported working with a development team between three and nine members. The next largest segment worked on smaller teams, with 28% reporting a team size of 1-2 developers. Those working on development teams of 10-20 developers represented 17% of respondents.

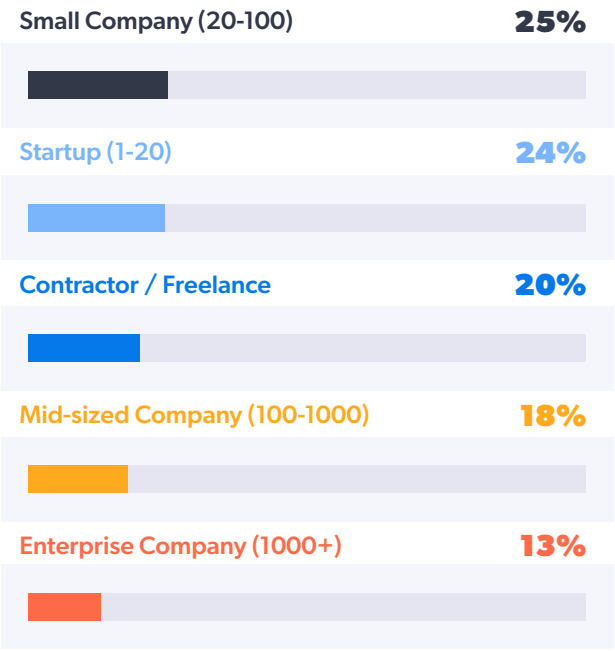
What is the size of your development team?



Company Size

In this question, we asked respondents to share the size of their company. Our survey found that respondents worked mostly for small to mid-sized companies, with 25% working for companies with 20-100 employees, 24% working for companies of with between one and 20 employees, and 20% reporting in as contract or freelance employees.

What is the size of your company?



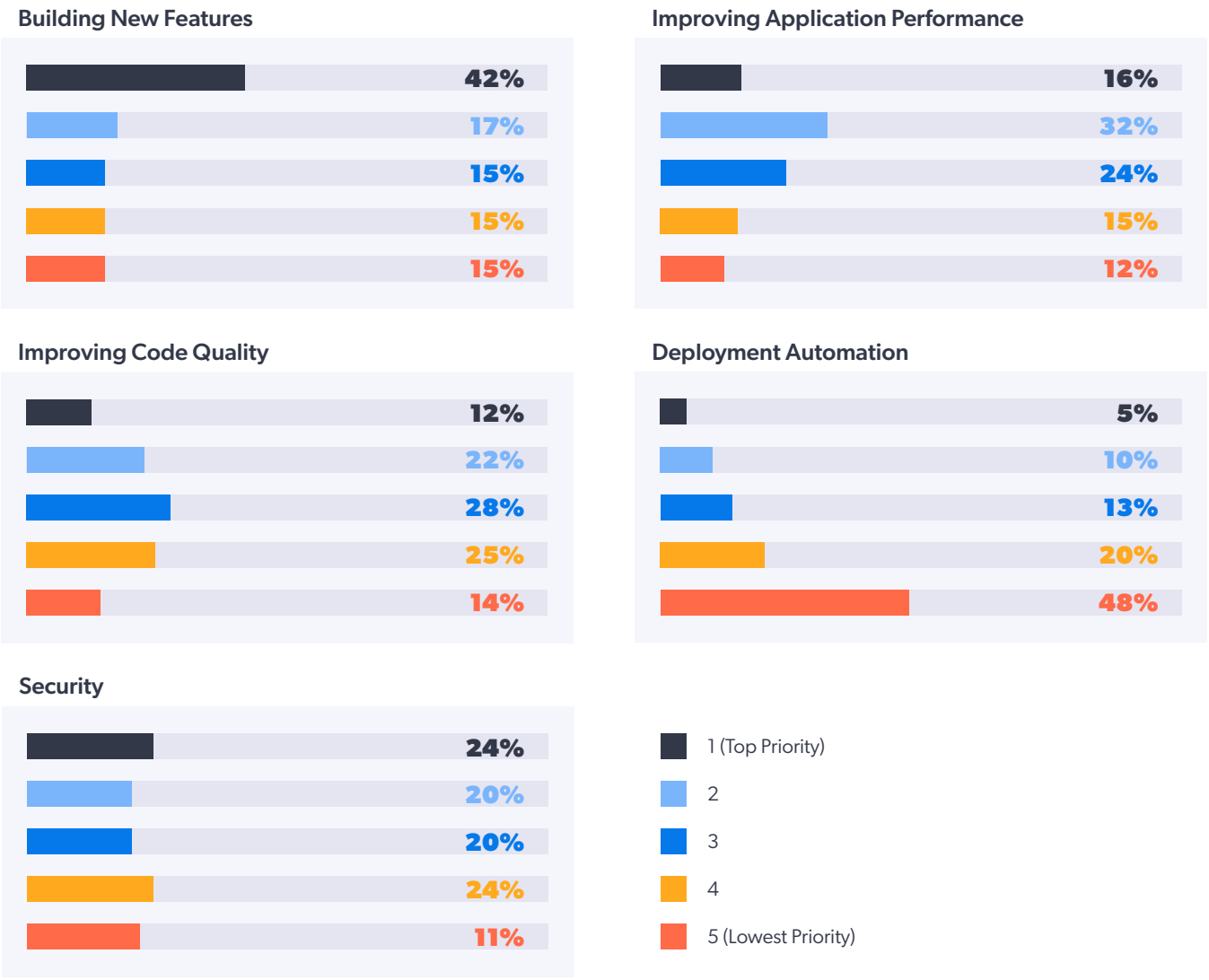
PHP Development Priorities

In this question, we asked respondents to rank their current development priorities, with choices that included building new features, improving application performance, improving code quality, deployment automation, and security.

Our survey found that the majority of development teams (42%) prioritized building new features on their current project. Security was also a priority, with 24% of teams reporting security as their top priority for ongoing development.

Among these responses, deployment automation was considered a low priority development, with 48% of respondents signaling it as their lowest priority for ongoing development.

What are your current development priorities?

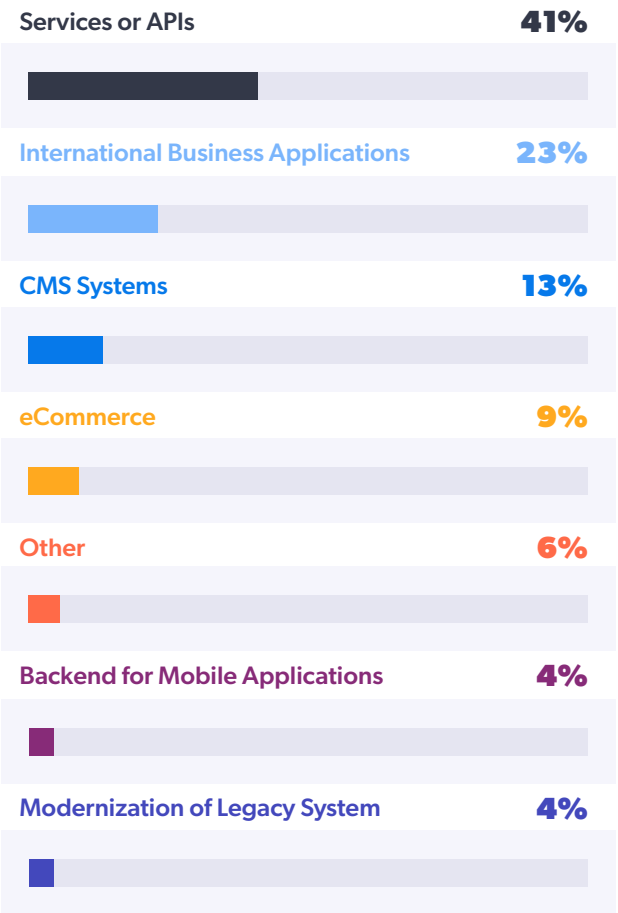


The takeaway is that developers and the companies they work for are prioritizing new features over technical debt and deployment automation. It's possible that this is because deployment automation is largely a one-time task, leading to its low priority in the survey. However, with security as the next highest priority, there is clearly a need to either run on the latest PHP versions or use supported runtimes that receive security backports. Additionally, application performance continues to be a consideration this year, which means Application Performance Management (APM) and profiling tools are a necessity for these organizations.

What Are Teams Developing With PHP?

In this question, we asked respondents to pick which type of application or system they are currently developing with PHP.

What are you using PHP for?



Our survey found that most respondents (40%) are using PHP for services or APIs. Moving down the list, internal business applications represented 23% of respondents, while CMS systems represented 13%.

Surveys in previous years found an increasing trend of developers using PHP for API development, and this survey cements API development as the leading use case for PHP. Considering that APIs are used to deliver dynamic content for web applications and mobile applications, provide communication for Internet of Things devices, and are a cornerstone of B2B integrations, this is not entirely surprising.

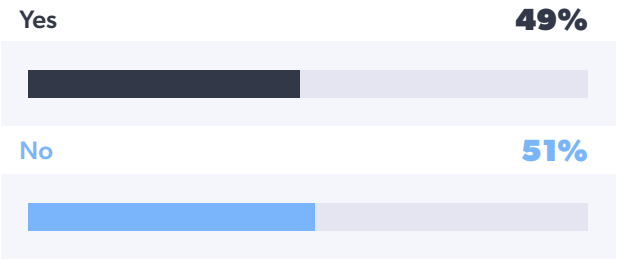
Internal business applications retain a high ranking amongst respondents. Many critical business systems leverage PHP's flexibility and speed of development, allowing them to adapt quickly to changing business requirements.

Finally, CMS systems still maintain a high profile amongst PHP users; it is rare to see public corporate presence on the web that is not using one of the several popular PHP CMS solutions, all of which provide flexibility and customization capabilities that will suit any user.

Compliance Requirements

In this question, we asked respondents whether or not their PHP applications have regulatory or industry compliance requirements.

Do you PHP applications have regulatory or industry compliance requirements (e.g., SOX, PCI, HIPPA, GDPR?)

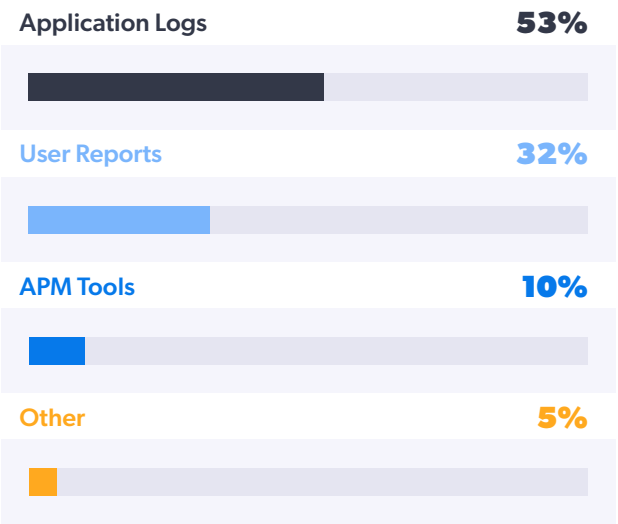


Our survey was nearly a perfect split, with 51% of respondents reporting no compliance requirements, and 49% reporting compliance requirements.

Discovering Issues in Production

In our next question, we asked respondents to weigh in with how they discover issues in their production applications.

How do you discover issues in your production application?



The majority of respondents (54%) reported using application logs as their key to finding production issues, while 32% listed user reports. APM tools rounded out the list at 10%, with the remaining respondents reporting other means of finding production issues.

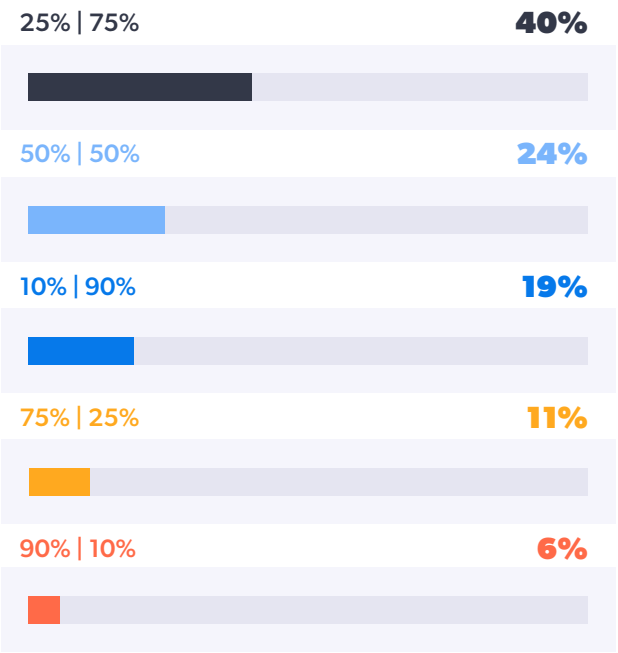
Finding and solving issues in production often requires a variety of approaches. APM tools often provide initial indications of problems, while application logs can help developers drill down to root causes. We have noticed that solutions such as the ELK stack (ElasticSearch, LogStash, and Kibana) are seeing increasing usage in a variety of application stacks; seeing a similar trend in PHP makes sense, as the same solution can be re-used in this context.

Generally speaking, receiving user reports of issues is a scenario of last resort, as it is an indication that problems are impacting customers. This may be an indication that APM tools require improvement to better assist in uncovering PHP application problems.

Maintenance vs. Feature Development

Our next question looked at how much time developers are spending on application maintenance and production bug fixes compared to how much time they spend on developing new functionalities in their application.

How much of your time is spent on maintenance and production bug/ issue resolution vs. developing new functionalities?



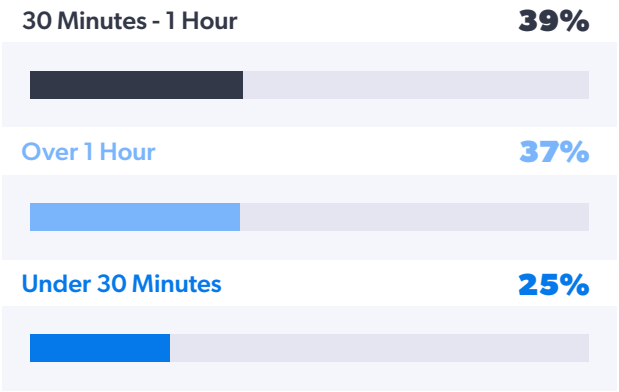
Our survey found that most respondents (40%) dedicate 25% of their time on maintenance and bug fixes, while the remaining 75% of their time is spent developing new functionalities.

Altogether, 83% of respondents reported spending at least 50% of their time on developing new functionalities.

Research and Development

Our next question looked at how much time respondents spend on performing coding-related research in an average day.

How much time do you spend per day (on average) performing coding-related research?



Our survey found that most respondents (39%) spend between 30 minutes and an hour performing code-related research, with those spending an hour or more trailing just slightly at 37%.

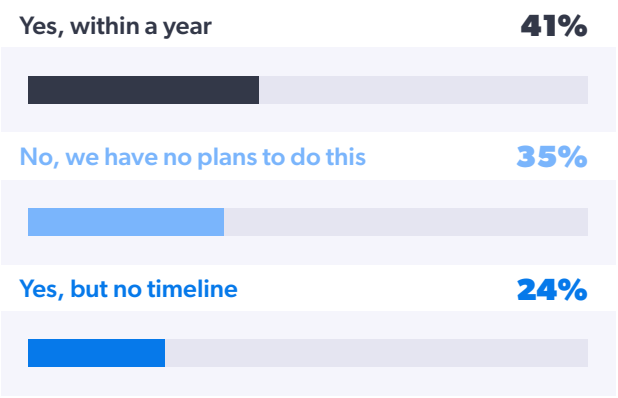
25% of respondents reported spending under 30 minutes per day performing code-related research (which could likely represent our non-development focused respondents).

Web application development has become more complex and nuanced in the past decade. Tools such as event sourcing, testing, static analysis, and strict typing often have steep learning curves, though they provide huge benefits once proficiency is reached. Additionally, concerns such as security, managing PHP version differences, and understanding the development impact of horizontal scaling can each require large amounts of research to fully understand how to architect an application correctly.

Containerization

Next, we asked respondents to share their status with regards to containerization technology adoption.

Are you currently using or planning to use any containerization technologies?



A combined 65% said they were planning on using containerization technologies. Of that 65%, 41% reported plans to use containerization technologies within the year.

Only 35% reported no current plans to use containerization technologies.

These numbers are relatively unchanged from last year's survey. Containers provide consistency and predictability in deployment, making horizontal scaling as well as deployment of microservices simpler. We expect to see container usage grow, particularly for APIs and microservices.

PHP Version and Upgrade Statistics

In this section, we look at the most-used PHP versions, how many teams plan on upgrading within the next year, and difficulties that teams encounter during the upgrade process. But first, let's consider why this information is so important to consider.

The PHP Group releases a minor (feature) or major (features that break backwards compatibility) release yearly; each receives active support for two years, and an additional year of security patches. This aggressive release cycle often falls short of standard business application lifetimes, which are typically in the 5 to 7 year range.

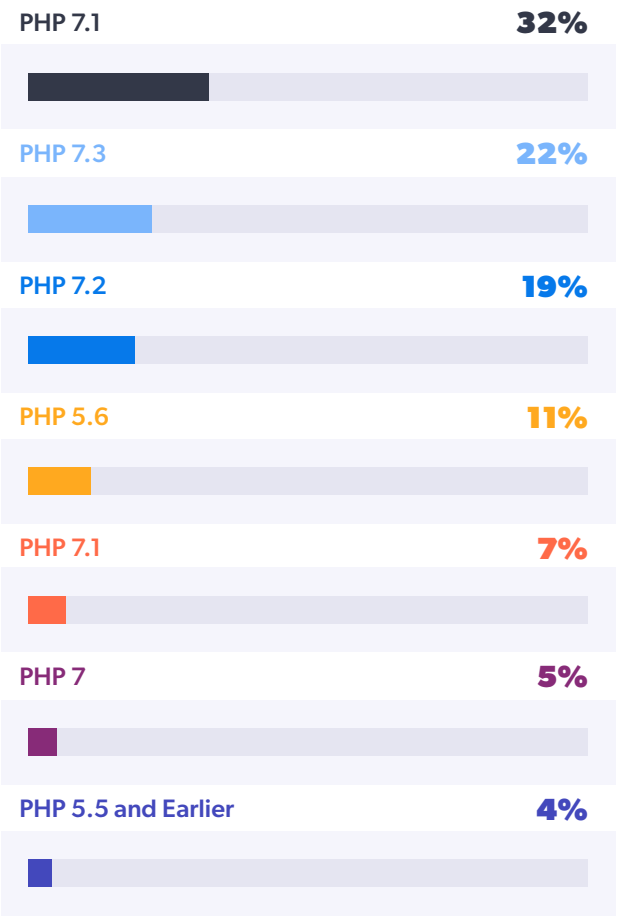
Good development practices, coupled with strong CI/CD pipelines, can allow companies to continually upgrade to newer versions of PHP. However, despite attempts to adhere to semantic versioning (which provides backwards compatibility promises), PHP often falls short, deprecating functionality during a new minor release, or tightening function signatures (generally to fix bugs) in ways that can cause subtle breakage in applications. As such, many companies will pin to a specific PHP minor release for their application; doing so helps reduce maintenance costs and allows them to focus on new features instead. They then have to weigh this approach with either security risks or the cost of a supported runtime.

With that in mind, which PHP versions are teams using most?

Most Used PHP Versions

This question asked respondents to share the PHP version they use on their current application.

Which version of PHP does your application use?



Not surprisingly, most respondents were using currently supported PHP versions, with 54% using PHP 7.3 or PHP 7.4, and 46% using PHP 7.2, 7.1, 7, 5.6, 5.5, or prior.

Perhaps reflecting the date range of the survey (which briefly preceded PHP 7.2 support EOL), 19% of respondents reported PHP 7.2 as their current PHP version.

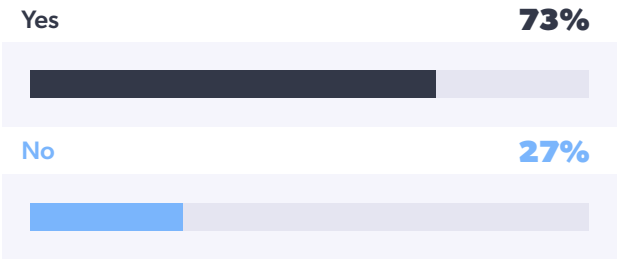
Interestingly, 11% reported still using PHP 5.6 on their current application.

> Roughly half of respondents are on versions of PHP no longer supported by the PHP Group. Companies that use older versions expose themselves to potential security risks as their runtimes remain unpatched; this can only be mitigated by purchasing commercial support that provides ongoing security patches for these versions, like ZendPHP.

PHP Version Upgrade Planning

Our next question looked at whether or not teams were planning a PHP version upgrade within the next year.

Are you planning a PHP version upgrade within the next year?



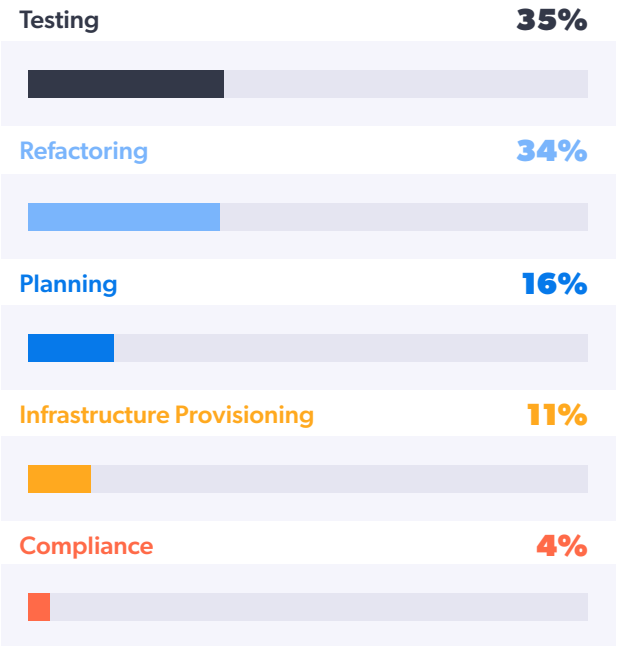
Our survey found that 73% of respondents were planning a PHP version upgrade within the next year, with the remaining 27% abstaining from an upgrade.

PHP version upgrades are not without cost; companies need to invest time in planning, testing, and refactoring whenever they do. Performing these on a yearly basis can help keep the scope of such upgrades more manageable; however, in regulated industries, adopting a new version can also trigger an audit cycle, which can introduce additional costs.

PHP Upgrade Hurdles

The next question looked at the biggest time commitments to PHP upgrades, and asked respondents to select the most-time consuming component of their last upgrade.

What was the most time-consuming component of your last PHP upgrade?



Our survey found that most respondents regarded testing (35%) or refactoring (34%) as the most time consuming elements of their last upgrade.

Planning marked the third-most time-intensive point of the upgrade process, with 16% reporting it as the most time-consuming element.

Our own experience helping customers upgrade to a new PHP version mirrors these findings. No upgrade should be performed without existing tests, and introducing tests is often the first step to any upgrade. Refactoring is generally only required when using features deprecated in the new version, but can also benefit the application when adopting new PHP features that reduce the amount of code. All PHP upgrades require careful planning and testing.

Development Technologies

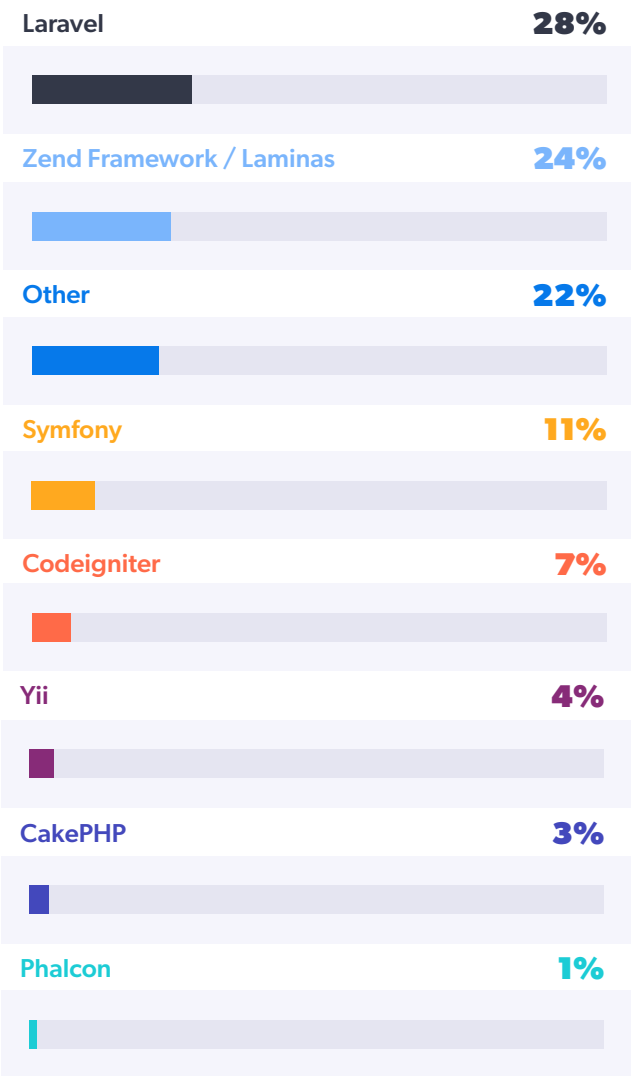
For this portion of the survey, we asked respondents to share their most-used PHP development technologies.

Specifically, we asked respondents to share their currently adopted framework, web server, and testing tools.

Most Popular PHP Framework

We first asked respondents to share which PHP framework they currently use, with choices that included Zend Framework/Laminas, Laravel, CodeIgniter, Symfony, Phalcon, Cake PHP, Yii, FuelPHP, or other.

Which PHP framework do you use?



Our survey found that most people were either using Laravel (28%) or Zend Framework / Laminas (24%).

Third most popular was Symfony at 11% of respondents, trailed by CodeIgniter at 7% andYii at 4%.

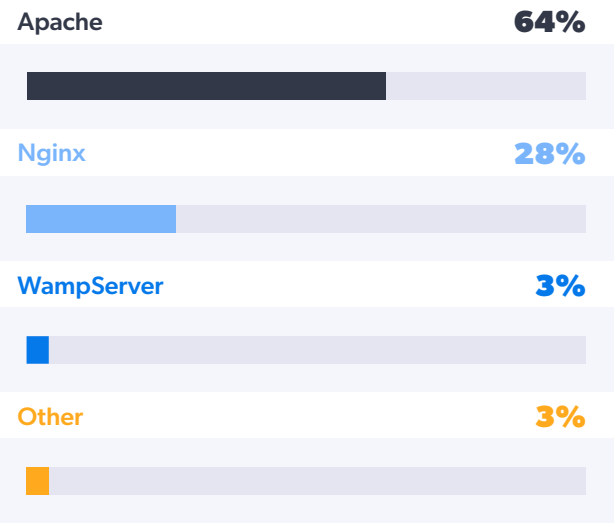
22% of respondents reported using a framework not included in our choices, with Slim and Drupal being the most popular answers provided.

The popularity of Zend Framework / Laminas can likely be attributed to the fact that its decoupled component architecture allows various packages to be used independently, then incorporated into applications developed in other frameworks. Additionally, the existence of the package manager Composer in the PHP ecosystem has led to the ability to mix and match elements of different frameworks to build out custom solutions — this has been a huge force for change in the PHP ecosystem.

Most Popular PHP Web Server

Our next question asked respondents to share the web server they use for their application.

Which PHP web server do you use?



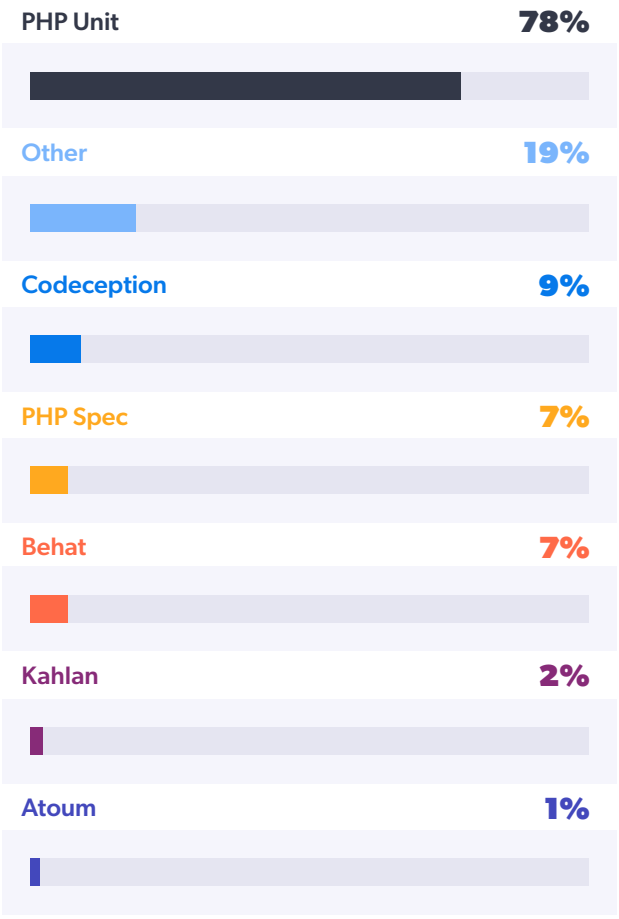
Our survey found most respondents using either Apache or Nginx, with 64% using Apache, and 28% using Nginx. WampServer was the only other statistically significant response at 3%.

Apache has been a huge force in the web ecosystem for decades and has managed to continue to outstrip competitors despite many considering it a "legacy" technology. Improvements to its performance have helped it retain its position as the web server of choice for PHP developers.

Most Popular PHP Testing Tools

Our last question for this topic asked respondents to share which PHP testing tools they use.

Which PHP testing tools do you use?



At 78%, respondents reported PHPUnit as the most popular PHP testing tool by far. Codeception came in at 9%, with PHPSpec and Behat both reported at 7%.

PHPUnit is a fundamental testing tool, allowing users to unit test applications, and provides limited integration and functional testing capabilities that can assist with overall application quality; we are not surprised to see it as the top selection. The next three most popular tools are each behavior and/or acceptance testing tools; these are used to help determine overall application correctness and are powerful tools to have in your suite, particularly if you plan to perform PHP upgrades.

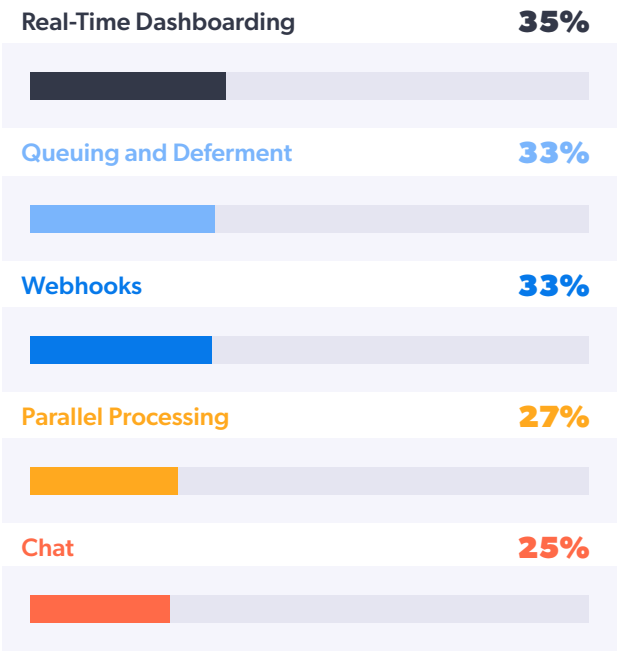
Asynchronous Programming

In previous years, we've asked if developers will be using asynchronous programming for any of their PHP applications; more than 2 in 5 indicated they have already used it or plan to. This year, we instead asked how they plan to use it and what implementations they prefer.

How Developers Are Using Asynchronous Programming

For our first question, we asked respondents to share how they use asynchronous programming in their current application.

How are you using asynchronous programming?



Our survey found respondents using asynchronous programming in multiple ways, with substantial responses for each available choice.

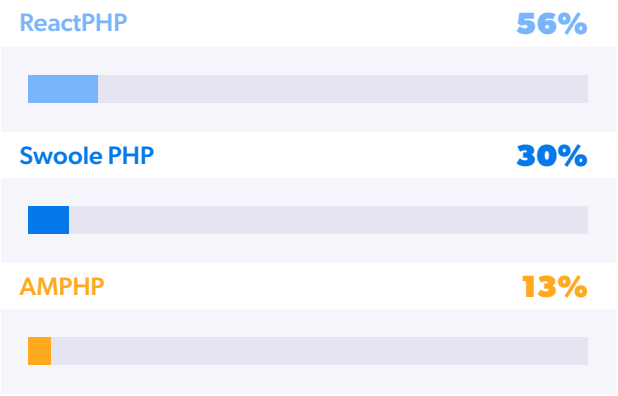
The most popular choice was real-time dashboarding, which came in at 35% of respondents. Queuing/deferment and Webhooks were next, each sharing 33% of respondents. Parallel processing was next at 27%, followed by chat usage marking 25%.

Chat and dashboards require websockets to back them, so using async for these types of applications is not surprising. We were surprised to see how evenly distributed each of these async contexts were amongst programmers; it's clear that developers have identified and have need for the various async use cases.

Most Popular Async PHP Framework

In the next question, we asked respondents to share if they are using an asynchronous PHP framework. The answers provided were Swoole PHP, AMPHP, and ReactPHP, and not applicable.

What async framework are you using?



Since an async framework is required in order to do async programming in PHP, we will only take into account the total number of respondents choosing an async framework; this gives a total of 142 respondents. Among these, ReactPHP is the most popular framework, capturing 56% of those using async. Swoole follows at 30%, then AMPHP at 13%.

With Swoole's clear advantages in speed and the ability to provide coroutine support for common operations due to its implementation as a PHP extension, we were surprised to see ReactPHP, a userland async framework, dominate the responses. That said, ReactPHP can optionally use PHP extensions (such as the "uv", "ev", or "event" extensions) to improve its performance and capabilities and provides some features, such as async filesystem access, that Swoole does not yet offer. All three options are solid choices, and we recommend keeping an eye on all of them.

PHP Outlook for 2021

As 2020 wraps up and 2021 begins, we observe several big changes in the PHP ecosystem. First, a new major version, PHP 8.0, has just dropped, providing a number of exciting features to the language. Many of these will help improve developer productivity (named arguments, constructor promotion, match expressions), reduce maintenance overhead (union types, enhanced covariance/contravariance rules, full range of type hinting capabilities), or increase performance (JIT). Features such as the Just-In-Time (JIT) compiler have yet to demonstrate their potential to the language, but could provide inroads into systems programming and machine learning.

With the introduction of PHP 8.0, we also see the sunsetting of community support for PHP 7.2. This comes on the footsteps of RHEL 8, which ships with version 7.2. Considering that any LTS operating system issued prior to this year is also on PHP 7.2 or earlier, companies depending upon a secure PHP version or a more recent PHP version will be required to either upgrade their operating system or evaluate commercially supported runtimes.

The PHP ecosystem has also diversified and matured immensely in the last few years. Testing tools and automation are gaining traction. Organizations are increasingly using containers to deploy applications. Async applications help enable real-time dashboards and data processing. As such, the average PHP developer needs more training and education to build the APIs and business-critical applications their organizations depend upon.

At this point, PHP is a stable, mature language, while still expanding its capabilities to gain a foothold in additional ecosystems, including Big Data and Machine Learning. Being a PHP developer comes with both the stability of 25 years of develop-ment and refinement as well as the excitement of all the growth and innovation to come.

QUESTIONS?

Do you have questions about the report? Want to talk to an expert about your big plans with PHP in 2021? Our experts are standing by to answer your burning questions.

TALK TO AN EXPERT

KEEP YOUR EOL PHP APPLICATIONS SECURE AND SUPPORTED

Zend can help keep your applications running on end-of-life PHP versions secure with backported bugfixes, patches, and around-the-clock support.

SEE SUPPORT OPTIONS

STREAMLINE WEB APPLICATION DEPLOYMENT

With Zend Server, your team can streamline application deployment, achieve scale, improve security, and increase performance.

TRY ZEND SERVER FREE

DEPLOYMENT SERVICES YOU CAN COUNT ON

From migrations, to audits, to implementing CI/CD, Zend Professional Services can help you realize your organization's goals.

SEE ZEND SERVICES