**Magento Latest Release Overview**

**Latest Version:**

* As of October 2023, the latest Magento release is **Magento 2.4.6** with the subsequent patch **Magento 2.4.6-p1** released in September 2023.
* This release focuses on enhancing security, improving performance, and adding new features to better serve merchants and developers.

**Key Features**

* **Security Enhancements:** Improved security protocols and patches to address vulnerabilities.
* **Performance Improvements:** Optimized caching mechanisms and codebase enhancements for faster load times.
* **GraphQL Enhancements:** Expanded GraphQL support for better API integrations.
* **Page Builder Updates:** Enhanced drag-and-drop capabilities for more intuitive content creation.
* **B2B Enhancements:** Improved functionalities tailored for B2B commerce, including better quoting and negotiation features.

**Support Lifecycle:**

* Magento 2.4.x series typically receives security updates and patches for 1-2 years post-release.
* It's essential to stay updated with the latest patches to ensure the platform remains secure and efficient.

**Installation and setup:**

**System Requirements:**

Before installing Magento with PHP 8.3, ensure that your server meets the necessary system requirements.

* **Operating System:** Linux x86-64
* **Web Server:** Apache 2.4 or Nginx 1.x
* **Database:** MySQL 8.0 or MariaDB 10.4
* **PHP:** 8.1 (PHP 8.3 is experimental)
* **Memory:** At least 2GB RAM
* **Other Dependencies:** Composer, Elasticsearch, etc.

**Installing php 8.3:**

**Note:** Since Magento 2.4.6-p1 officially supports up to PHP 8.1, installing PHP 8.3 is at your discretion and risk.

**Terminal:**

sudo apt update

sudo apt install -y software-properties-common

sudo add-apt-repository ppa:ondrej/php

sudo apt update

sudo apt install -y php8.3 php8.3-cli php8.3-fpm php8.3-mysql php8.3-xml php8.3-mbstring php8.3-curl php8.3-zip php8.3-intl

**Installing Magento Latest Release**

**Download magento:**

composer create-project --repository=https://repo.magento.com/ magento/project-community-edition=2.4.6-p1

**Set File permission:**

sudo find var generated vendor pub/static pub/media app/etc -type f -exec chmod g+w {} +

sudo find var generated vendor pub/static pub/media app/etc -type d -exec chmod g+ws {} +

sudo chown -R :www-data .

**Configure magento**

php bin/magento setup:install \

--base-url=http://yourdomain.com/ \

--db-host=localhost \

--db-name=magento \

--db-user=magento \

--db-password=magento\_password \

--admin-firstname=Admin \

--admin-lastname=User \

--admin-email=admin@yourdomain.com \

--admin-user=admin \

--admin-password=Admin123! \

--language=en\_US \

--currency=USD \

--timezone=America/Chicago \

--use-rewrites=1

Final installation

php bin/magento setup:upgrade

php bin/magento setup:di:compile

php bin/magento setup:static-content:deploy -f

php bin/magento cache:clean

**Configuration:**

**Magento Configuration for php 8.3**

To optimize Magento for PHP 8.3, consider the following configurations:

1. **PHP Configuration:**
   1. **Memory Limit:** Increase memory\_limit in php.ini to at least 2G.
   2. **OPcache:** Ensure OPcache is enabled for performance gains.
   3. **Error Reporting:** Set display\_errors to Off and log errors to a file.
2. **Magento Configuration:**
   1. **Caching:** Utilize built-in caching mechanisms like Varnish.
   2. **Indexing:** Schedule regular indexing to keep data optimized.
   3. **Asynchronous Operations:** Enable asynchronous indexing and message queues for better performance.

**Optimizing performance**

1. **Use a Content Delivery Network (CDN):** Offload static assets to a CDN for faster delivery.
2. **Enable Flat Catalog:** Improve database query performance by enabling flat catalog tables.
3. **Minify CSS and JS:** Reduce file sizes for faster load times.
4. **Database Optimization:** Regularly optimize and clean the database to maintain performance.

**PHP 8.3 overview**

**Release information:**

PHP 8.3, released in August 2023, brings several new features, optimizations, and deprecations aimed at improving performance, developer experience, and security.

**Key Features**

* **Readonly Classes:** Allowing entire classes to be declared as readonly.
* **Improved Enumerations:** Enhancements to enum types for better type safety.
* **Performance Optimizations:** Significant speed improvements and reduced memory usage.
* **Deprecations:** Certain functions and features deprecated to streamline the language.

**Compatibility considerations**

* With each new PHP release, it's crucial to assess compatibility with existing applications and frameworks. PHP 8.3 introduces changes that may affect how Magento and its extensions operate.’

**Deprecations and removal**

PHP 8.3 also deprecates and removes certain functions and features to streamline the language:

1. **Deprecated Functions:** Some older functions have been deprecated and may be removed in future releases.
2. **Strict Typing Enhancements:** Stricter type checking can lead to more type-related errors in existing codebases.
3. **Removal of Legacy Features:** Certain legacy features and extensions have been removed, necessitating codebase updates.

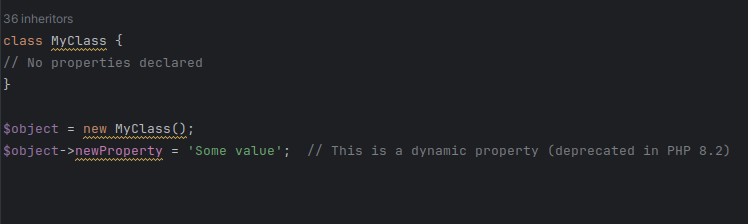
|  |  |  |  |
| --- | --- | --- | --- |
| Category | PHP 8.2 | PHP 8.3 | Notes |
| Release Date | December 8, 2022 | Scheduled for December 2023 | PHP 8.3 release is expected at the end of 2023. |
| Main Compatibility | Fully backward compatible with PHP 8.1 | Fully backward compatible with PHP 8.2 | Both versions are compatible, but some breaking changes and deprecations are specific to 8.3. |
| Readonly Properties | Introduced in PHP 8.1, supported | **Removed** in favor of standalone readonly classes | In PHP 8.3, readonly properties in classes are removed; readonly classes should be used instead. |
| Readonly Classes | Not supported | Introduced | Entire classes can be marked as readonly in PHP 8.3, simplifying property access rules. |
| Random Extension | Introduced with Random\Randomizer and Random\Engine | **Updated**: Random\Randomizer::shuffle() supports arrays | Improved randomization features, such as shuffling arrays, are added in 8.3. |
| Deprecated Features | Few minor features deprecated | Significant deprecations and function removals | PHP 8.3 deprecates and removes several functions that were already marked as deprecated in earlier versions. |
| Dynamic Properties | Deprecated in PHP 8.2 with a warning | **Removed** completely | In PHP 8.2, dynamic properties trigger deprecation warnings, but in 8.3 they are no longer allowed. |
| String Interpolation | Supports $foo[bar] | **Deprecated** $foo[bar] | PHP 8.3 deprecates using array access inside string interpolation, recommending {$foo[bar]} instead. |
| Functions Changed | fputcsv() improved with support for custom escape chars | **Deprecated** stripcslashes(), magic\_quotes\_\* removed | Several older functions have been removed entirely or deprecated in 8.3. |
| Errors and Warnings | More forgiving error handling | **Stricter** error handling, additional warnings | PHP 8.3 introduces stricter error handling and more detailed warnings for improved debugging. |
| Locale-Independent Float to String Casting | Default locale settings affect float-to-string conversion | **Independent** of locale | PHP 8.3 converts floats to strings independently of the locale, making behavior consistent globally. |
| Array Unpacking | Supports unpacking arrays into function parameters | **Supports unpacking** in array literals (e.g. [...$arr]) | PHP 8.3 introduces array unpacking in literals for a cleaner syntax. |
| New Features | Limited to features from previous minor versions | **Fibers in Closures**, new json\_validate() function | PHP 8.3 adds features like validating JSON strings and improving closures with fibers. |
| Callable Types | Callable types allowed in callable hints | **Expanded** callable types | PHP 8.3 introduces more callable type hints, providing stronger typing in callbacks. |
| Reflection Changes | No major changes | **Introduced** ReflectionFunctionAbstract::isClosure() | PHP 8.3 adds the ability to check if a function is a closure using reflection. |
| Typed Properties | Supported for classes | **Enhanced** for scalar types | PHP 8.3 strengthens support for typed properties, especially scalar types. |
| Match Expression | Introduced in PHP 8.0 | **Enhanced** with new capabilities | PHP 8.3 introduces more advanced use cases for the match expression, including match in expressions. |
| Deprecations in MySQLi | None | **Deprecated** mysqli::execute | The mysqli::execute method is deprecated in PHP 8.3, recommending alternatives. |
| Anonymous Classes | Fully supported | **Readonly anonymous classes** supported | PHP 8.3 allows anonymous classes to be declared as readonly, improving immutability. |
| Serialization Changes | No major changes | Serializable interface deprecated | PHP 8.3 deprecates the Serializable interface, recommending \_\_serialize() and \_\_unserialize() |
| Fibers | Introduced in PHP 8.1 | **Enhanced** with better error handling in closures | PHP 8.3 improves fibers with better integration into closures and error handling. |
| Legacy Magic Quotes | Deprecated but not removed | **Removed** completely | All remaining references to magic\_quotes\_\* functions are removed in PHP 8.3. |
| Temporary Variables in Arrays | Allowed to be assigned in expressions | Not allowed | PHP 8.3 forbids assigning to temporary variables inside arrays for better memory management. |
| JSON Functions | Basic functions for handling JSON | **json\_validate()** function introduced | PHP 8.3 adds json\_validate() to validate JSON strings without decoding them, useful for error checking. |
| Array Functions | Basic array functions available | **array\_is\_list()** added | New array helper function array\_is\_list() is introduced in PHP 8.3 to check if an array is a list. |
| Attributes | Introduced in PHP 8.0 for metadata | **Improved** support for attributes on constants | PHP 8.3 expands attribute support, allowing them to be used on constants as well. |

**Summary of deprecated functions and features removed in php 8.3 version**

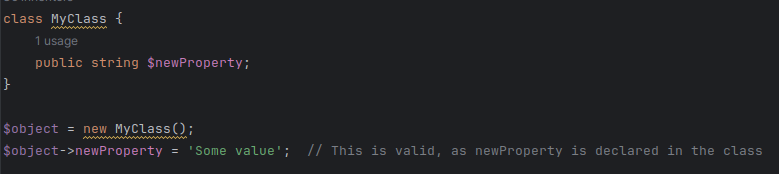
1. **Dynamic Properties**: Fully removed in PHP 8.3 (deprecated in PHP 8.2).
2. **Magic Quotes Functions**: Fully removed, including magic\_quotes\_gpc, magic\_quotes\_runtime.
3. **stripcslashes()**: Deprecated in PHP 8.3.
4. **Serializable interface**: Deprecated in favor of \_\_serialize() and \_\_unserialize() methods.
5. **MySQLi's execute() method**: Deprecated in PHP 8.3.

Dynamic properties: Dynamic properties are properties that are not explicitly defined in a class but are created at runtime when assigned.

Deprecated code:



Corrected code



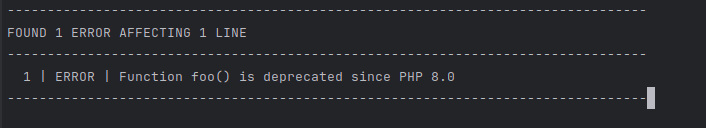
* **PHP 8.2**: Dynamic properties trigger a deprecation warning.
* **PHP 8.3**: Dynamic properties are fully removed, and attempting to create them will result in a fatal error.

--------------------------------------------------------------------------------------

* Installed phpstorm in my system
* Using that we can check deprecated function
* We can check php version by this command php –v

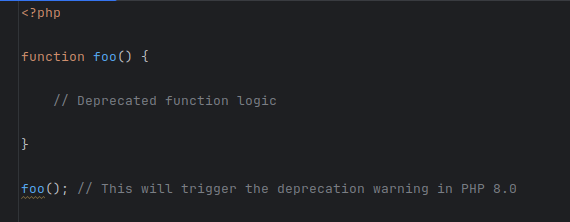
-----------------------------------------------------------------------

**foo()**

* The function foo() has been **deprecated since PHP 8.0**.
* This means that while it may still work in PHP 8.0 and later versions, it's no longer recommended to use this function, and it could be removed in a future PHP release.

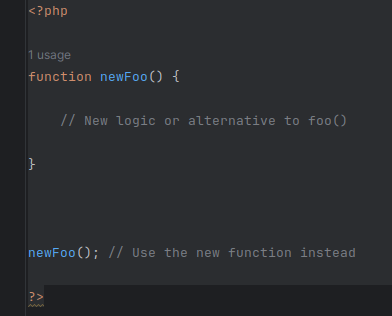
**Example**

* If PHP 8.0 deprecates a function like foo(), it means this function is no longer considered a best practice to use, and there's likely an alternative function or method to achieve the same outcome.



* **Replace the deprecated function** with its alternative. If a function is deprecated, there is usually a newer, recommended alternative.
* **Check the PHP documentation** for the function to find out why it was deprecated and what should be used instead.

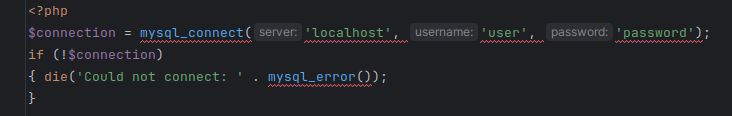
Corrected code:



------------------------------------------------------------------------------------------

**mysql\_connect()**



* The mysql\_connect() function has been deprecated since PHP 5.5. You should replace it with mysqli\_connect() or use PDO for database interactions.
* The function mysql\_connect() was used in earlier versions of PHP to establish a connection to a MySQL database. However, since PHP 5.5, this function has been **deprecated**, and as of PHP 7.0, it has been completely **removed**.
* The reason for its deprecation is that it lacks modern features and security enhancements that are provided by newer MySQL APIs like mysqli and PDO.
* If you continue to use mysql\_connect(), your code will not be compatible with newer PHP versions, and it will result in fatal errors in PHP 7.0 and above.
* Deprecated code   
    
    
  **mysqli\_connect()**: A more modern MySQL extension that supports features like prepared statements and multiple connections.
* **PDO (PHP Data Objects)**: A database abstraction layer that works with multiple types of databases (MySQL, PostgreSQL, etc.).

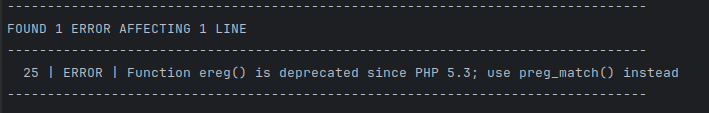
Corrected code:



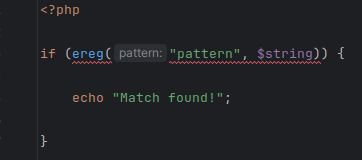
* **stop using mysql\_connect()** and migrate your code to use either mysqli or PDO for better security, performance, and compatibility with modern PHP versions.

---------------------------------------------------------------------------------------------------------

**ereg():**

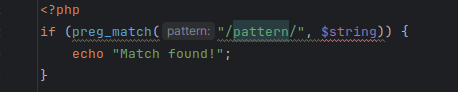
* The ereg() function has been deprecated since PHP 5.3. It is recommended to use preg\_match() with regular expressions instead.
* The ereg() function was used in earlier versions of PHP for **regular expression pattern matching**. However, since PHP 5.3, it has been **deprecated**, and in PHP 7.0, it was completely **removed**.
* The reason for its deprecation is that ereg() and its related functions are outdated and slower than the more powerful and efficient **Perl-Compatible Regular Expressions (PCRE)** functions like preg\_match().

Deprecated code:



* In this code, ereg() is used to check whether the string matches the provided pattern.
* Replace ereg() with **preg\_match()**, which is the modern alternative for regular expression matching in PHP.

Corrected code:

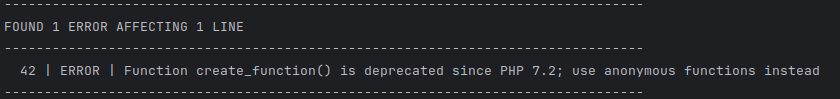


Replace ereg() with **preg\_match()**, which is the modern alternative for regular expression matching in PHP.

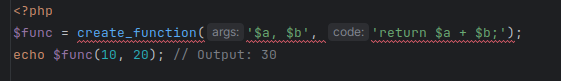
**replace all instances of ereg() with preg\_match()** to ensure your code is compatible with PHP 5.3 and later versions, and to benefit from better performance and functionality.

----------------------------------------------------------------------------------------------------------

**Create\_function()**

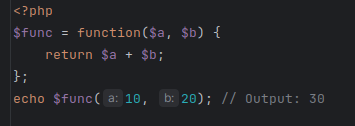
* The create\_function() function has been deprecated since PHP 7.2. Instead, you should use anonymous functions (closures).
* The create\_function() function was used in earlier versions of PHP to create anonymous (or inline) functions on the fly.
* However, starting with **PHP 7.2**, this function was **deprecated** due to several drawbacks, including poor performance, security risks (such as potential code injection), and lack of readability. It was fully removed in **PHP 8.0**.

Deprecated code



* Replace create\_function() with a **closure (anonymous function)**, which is the modern and safer approach to create inline functions in PHP.

Corrected code



* **create\_function() Syntax**: create\_function('$params', 'code')
* **Anonymous Function Syntax**: function($params) { code }

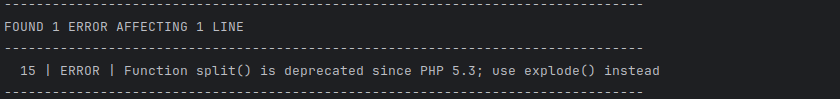
In anonymous functions, there’s no need to wrap the code in a string, making it safer and easier to read.

**replace all instances of create\_function() with anonymous functions** to ensure your code is compatible with PHP 7.2 and later versions, while also improving security, performance, and readability.

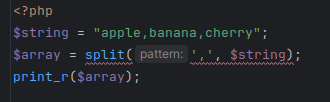
--------------------------------------------------------------------------------------

split()

* The split() function was used in earlier versions of PHP to split a string into an array based on a regular expression pattern.
* However, it was **deprecated as of PHP 5.3.0** and fully **removed in PHP 7.0**.
* This is because the function was regex-based, but PHP offers more efficient and straightforward alternatives like explode() or preg\_split().

 The split() function has been deprecated since PHP 5.3. You should use explode() or preg\_split().

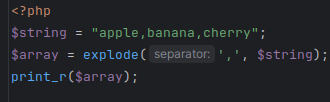
Deprecated code:



In this code, split() is used to split the $string into an array using a comma delimiter. This function is now deprecated.

* Use **explode()** for splitting strings by a specific delimiter.
* If regular expressions are required, use **preg\_split()**.

Corrected code

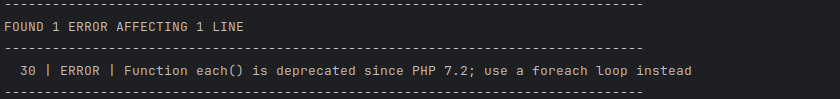


The explode() function takes two arguments: the **delimiter** and the **string to be split**. In this case, it splits the string by commas.

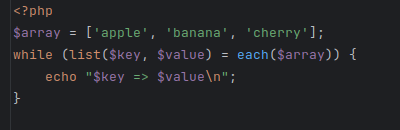
-----------------------------------------------------------------------------------------------------------

**each()**

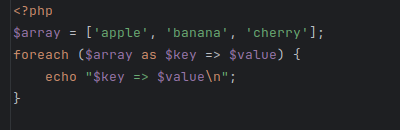
* The each() function in PHP was used to iterate over arrays.
* It returns the current key-value pair from an array and moves the internal pointer to the next element.
* However, it was **deprecated in PHP 7.2** and eventually removed in later versions.
* This is because it was considered redundant and less efficient compared to modern iteration methods, such as foreach.

The each() function has been deprecated since PHP 7.2. You should replace it with a foreach loop to iterate over arrays.

Deprecated code:



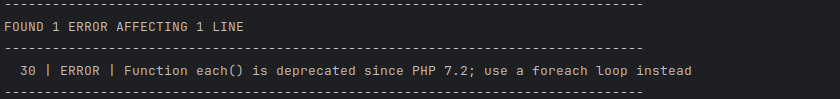
Corrected code



* **Avoid using each()** in your PHP code since it is deprecated.
* **Use foreach()** as it provides a more efficient and readable alternative.
* By switching to foreach, your code will be compatible with PHP 7.2 and later versions, avoiding deprecation issues.

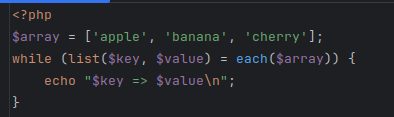
----------------------------------------------------------------------------------------------------------------

**each()**

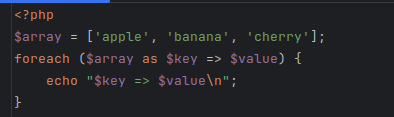


* **Explanation**: In PHP, a function or method that has a void return type is not expected to return any value.
* The void type indicates that the function will not return anything, and it is an error to include a return statement that provides a value when the function is declared as void.
* Starting from **PHP 8.2**, this behavior is enforced more strictly. If a function with a void return type attempts to return a value, it will cause a fatal error.

Deprecated code



Corrected code:



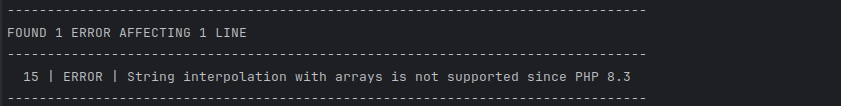
* In the above example, the function processSomething() is declared with a void return type, meaning it should not return any value.
* However, the function is incorrectly trying to return true, which is not allowed when the return type is void.
* **Avoid using each()** in your PHP code since it is deprecated.
* **Use foreach()** as it provides a more efficient and readable alternative.

By switching to foreach, your code will be compatible with PHP 7.2 and later versions, avoiding deprecation issues.

-----------------------------------------------------------------------------------------------------------------

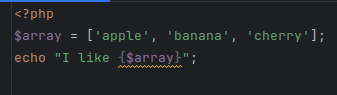
Array();

* String interpolation in PHP allows variables to be embedded directly inside strings by using double quotes.
* However, **as of PHP 8.3**, string interpolation with arrays is no longer supported. This means you cannot directly embed arrays within a string in this manner.

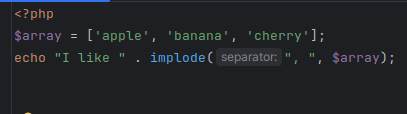


* Trying to do so will cause an error.

Deprecated code:



Corrected code:

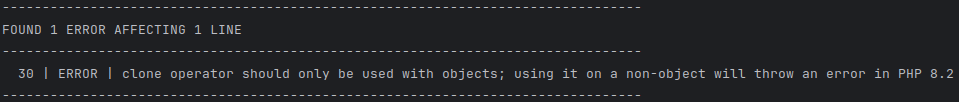


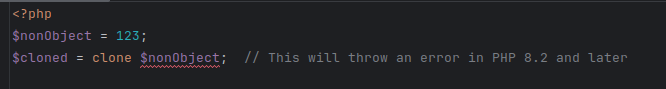
* To avoid this error, you should **not attempt to interpolate arrays directly** into a string. Instead, you can use functions like implode() or print\_r() to convert the array into a string format before embedding it.
* **Avoid string interpolation with arrays** in PHP 8.3 and later versions.
* Use **implode()** to join the array elements into a string if you need to display them.

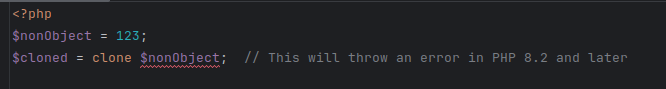
-----------------------------------------------------------------------------------

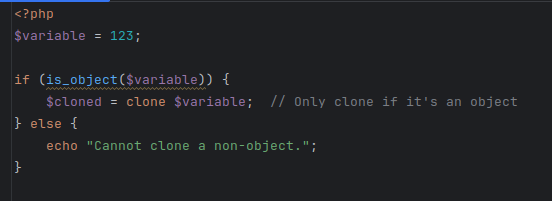
Clone

* In PHP, the clone operator is used to create a copy of an object. Prior to PHP 8.2, you could attempt to use the clone operator on non-object types (like integers, arrays, strings, etc.) without immediately causing a fatal error.
* However, this behavior was incorrect because the clone operator is meant exclusively for objects.

Deprecated code

Corrected code





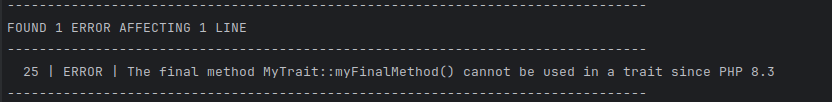
* Use the clone operator only with objects, as it is designed to copy object instances.
* Use is\_object() to check if a variable is an object before attempting to clone it.

This change helps enforce stricter and more reliable code, ensuring that the clone operator is used correctly for object copying.

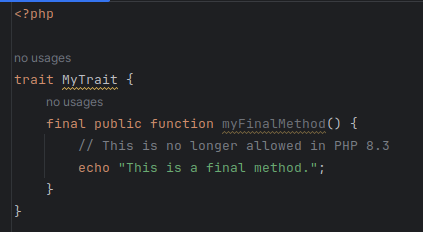
-------------------------------------------------------------------------------------

MyTrait

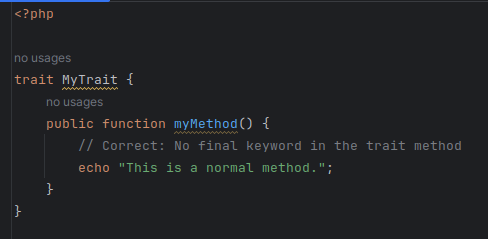
* In PHP, a final method is one that cannot be overridden by child classes. This is typically used to lock down certain methods in a class to prevent modification in subclasses.
* However, **traits** are not classes, but rather a mechanism for code reuse across different classes. Since traits are meant to be "injected" into different classes, the concept of a final method conflicts with the way traits work.
* As of **PHP 8.3**, declaring a final method inside a trait has been prohibited because traits are not supposed to enforce method finality.
* The method defined in a trait can be used in any class that uses the trait, but marking it final goes against the trait’s purpose of flexibility and code sharing.



Deprecated code



Corrected code



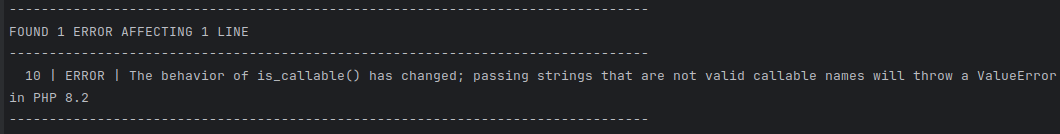
* **In PHP 8.3**, traits cannot contain final methods.
* Traits are designed for flexible code reuse across multiple classes, and marking methods as final goes against this principle.
* If you need to declare a method as final, do so within the class using the trait, not within the trait itself.

By removing the final keyword from trait methods, you ensure that traits maintain their flexibility and compatibility with PHP 8.3.

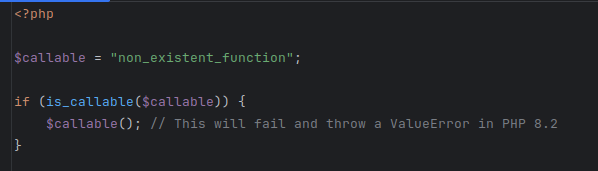
------------------------------------------------------------------------------------------------------

Is\_callable()

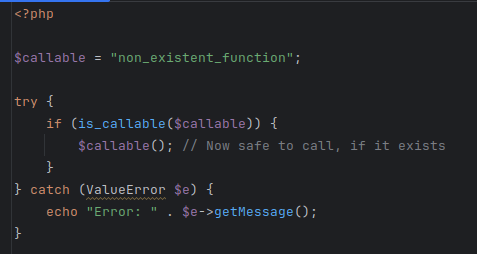
* In **PHP 8.2**, the behavior of the is\_callable() function has changed.
* Prior to PHP 8.2, is\_callable() returned false when given a string that is not a valid callable name (i.e., a function or method that does not exist).
* In PHP 8.2, however, passing a string that is not a valid callable now results in a **ValueError exception** being thrown.
* This is a stricter handling of invalid callables compared to previous versions.



Deprecated code



Corrected code

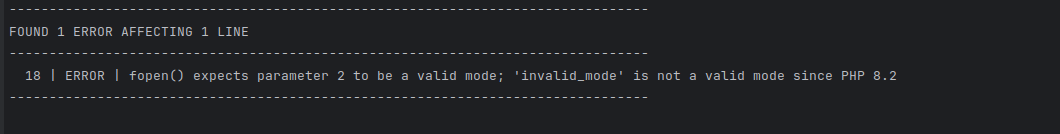


* **In PHP 8.2**, passing invalid callable strings to is\_callable() will throw a **ValueError**.
* You should either verify the callable's existence before using it or handle the potential error with a try-catch block.

------------------------------------------------------------------------------------------------------

fopen()

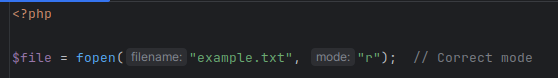
* In **PHP**, the function fopen() is used to open files.
* The second parameter passed to fopen() is the "mode" that specifies how the file will be opened (for reading, writing, etc.). Valid modes include "r", "w", "a", etc.
* As of **PHP 8.2**, stricter validation is applied to the mode parameter.
* If an invalid mode string is provided, fopen() will throw a **ValueError** rather than failing silently or triggering a warning.



Deprecated code



Corrected code

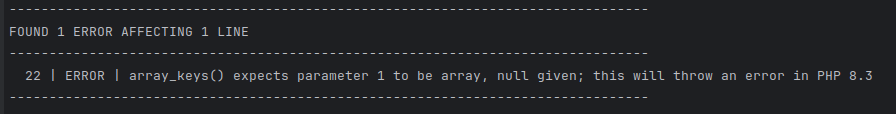


* **In PHP 8.2**, fopen() now throws a **ValueError** if an invalid mode string is passed as the second argument.

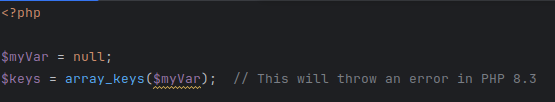
---------------------------------------------------------------------------------------------------------

Array\_keys()

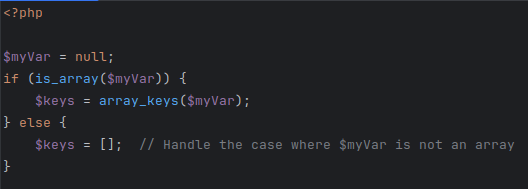
* The **array\_keys()** function in PHP is used to retrieve all the keys from an array. The first parameter it accepts must be an array.
* If the parameter passed is not an array, PHP will either throw a warning or an error, depending on the version.
* As of **PHP 8.3**, passing a null value or any non-array type as the first argument will result in a **TypeError** being thrown.
* This is stricter behavior compared to earlier PHP versions where a warning might have been triggered.



Deprecated code



Corrected code



* **In PHP 8.3**, passing null (or a non-array type) to array\_keys() will throw a **TypeError**.
* Always ensure that the variable passed to array\_keys() is an array by checking it with is\_array().
* This stricter behavior enforces better type safety and prevents bugs that might have gone unnoticed in earlier versions.

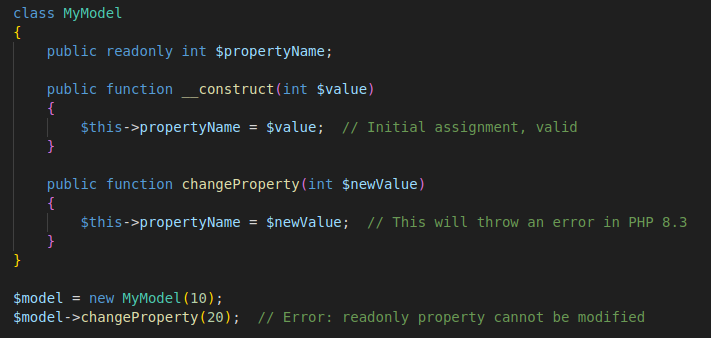
------------------------------------------------------------------------------------------------------------

**Readonly**

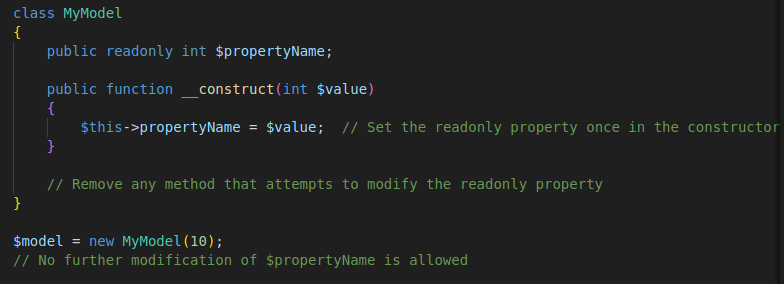
Starting with **PHP 8.1**, a new feature called **readonly properties** was introduced. These properties can only be assigned once and cannot be modified afterward. If you attempt to change a readonly property after it has been initialized, PHP will throw an error.

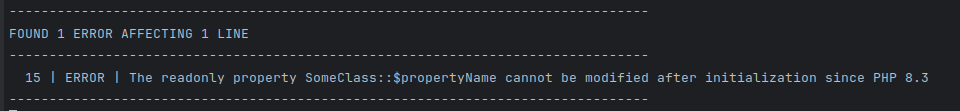
From **PHP 8.3** onward, trying to modify a readonly property after its initialization will result in a **TypeError** being thrown. This stricter behavior ensures that readonly properties are immutable once assigned.

Deprecated code



Corrected code:





* **Readonly properties** introduced in PHP 8.1 can only be assigned once and cannot be modified.
* In **PHP 8.3**, attempting to modify a readonly property after its initialization will throw a **TypeError**.
* Ensure that readonly properties are initialized only once, typically in the constructor, and remove any code that attempts to modify them after the initial assignment.

Readonly properties are a powerful feature for making objects immutable, preventing accidental changes after initialization.

--------------------------------------------------------------------------------------------------