Data i czas w PHP

Wiele współczesnych aplikacji webowych potrzebuje dokonywać operacji na dacie i czasie – od prostych logów działań użytkownika (np. moment zakupu produktu), przez systemy rejestracji, postowania treści, aż po zaawansowane funkcje, jak np. odliczanie czasu dostępu premium do usługi. PHP udostępnia kilka wbudowanych funkcji oraz klas do operowania na czasie – na egzaminie INF.03 także często pojawia się temat formatowania daty oraz przeprowadzania operacji (np. porównania) na czasie – warto więc dobrze opanować te zagadnienia!

W razie gdybyśmy oprócz niniejszej powtórki wiadomości, potrzebowali także tutoriala video wyjaśniającego krok po kroku pracę z datą i czasem w PHP, to warto zajrzeć do filmu: Przetwarzanie daty i czasu serwera w PHP.

Czas serwera kontra czas lokalny

- **Czas serwera (backend – PHP)** – to czas generowany przez serwer, zależny od jego strefy czasowej.
- **Czas lokalny (frontend – JavaScript)** – to czas użytkownika (pobierany z lokalnej maszyny klienta), który może rzecz jasna różnić się od czasu serwera.

Jeśli pracujemy w aplikacji wielojęzycznej, dostępnej w różnych strefach czasowych, to musimy wówczas *dostosować czas* do użytkownika lub przechowywać go w *UTC* i przeliczać po stronie klienta.

Funkcja time() w PHP – czas UNIX

Funkcja

`time()` w PHP zwraca aktualny czas jako liczbę sekund, które upłynęły od tzw. epoki UNIX – czyli dokładnie od północy 1 stycznia 1970 roku (czasu UTC). W praktyce, ta liczba reprezentuje tzw. “odcisk czasu” (ang. *timestamp*) i jest bardzo wygodna do przechowywania, porównywania i wykonywania obliczeń na datach.

```
echo time();
// Przykładowy wynik: 1714913625
```

Ponieważ ta funkcja zwraca czas bieżący (moment jej wywołania) i możemy np. określić ile sekund minęło od jakiegoś wydarzenia, albo kiedy coś ma się wydarzyć za np. 3 dni – wystarczy dodać odpowiednią liczbę sekund (3 × 24 × 60 × 60 = 259200):

```
$zaTrzyDni = time() + 3 * 24 * 60 * 60;
echo $zaTrzyDni;
// Przykład: 1715172825
```

To bardzo wygodne, ponieważ możemy wykorzystywać tę liczbę do dokonywania porównań:

```
if (time() > $terminWaznosci) {
    echo "Dostęp premium wygasł.";
}
```

Zalety czasu unixowego:

- Prosty format – tylko jedna liczba całkowita.
- Łatwe porównania – można używać operatorów `>`, `<`, `==`.
- Idealny do baz danych – przechowujemy znacznik czasu jako `INT`.
- Uniwersalny format – stosowany w wielu językach programowania i systemach.

Tworzenie konkretnych dat: mktime()

Funkcja

`mktime()` pozwala utworzyć znacznik czasu dla określonej daty i godziny – idealna gdy chcemy pracować z konkretnym momentem w czasie. Możemy wyznaczyć ile sekund minęło od rozpoczęcia epoki UNIX do zadanego, zdefiniowanego przez nas momentu:

```
$czas = mktime(12, 0, 0, 1, 1, 2025);
echo $czas;
// Wynik: 1767240000
```

Składnia domyślna przyjmuje format:

```
mktime(godzina, minuta, sekunda, miesiąc, dzień, rok)
```

Dzięki

`mktime()` możemy łatwo tworzyć daty przeszłe i przyszłe, aby np. porównać je z czasem aktualnym, w momencie wywołania:

```
$jutro = mktime(0, 0, 0, date("m"), date("d") + 1, date("Y"));
if (time() < $jutro) {
    echo "Jeszcze dziś!";
}
```

Takie podejście przydaje się np. przy planowaniu zadań, wyświetlaniu ważności kont, działania licznika lub pokazywaniu aktualnego dnia tygodnia.

W kolejnych częściach lekcji zobaczymy jak przekształcać ten “sekundowy” znacznik czasu na bardziej przyjazny dla człowieka format, np.

`01.05.2025 12:00` – za pomocą funkcji `date()`.

Formatowanie daty: funkcja date()

Funkcja

`date()` pozwala sformatować klasyczny znacznik czasu UNIX (czyli liczbę sekund) do czytelnej dla człowieka postaci – na przykład: `2025-05-05 14:00`.

Jako argumenty przyjmuje ona dwa parametry:

1. Łańcuch znaków – szablon formatu (np. `"Y-m-d H:i:s"`),
2. (opcjonalnie) znacznik czasu UNIX – np. `time()` lub wynik `mktime()`.

```
echo date("Y-m-d H:i:s");
// Wynik: 2025-05-05 14:00:00
```

Jeśli nie podamy drugiego argumentu, to domyślnie używana jest bieżąca (aktualna) wartość

`time()`.

Najczęściej używane literały formatujące w date():

- `Y` – pełny rok, np. 2025
- `y` – skrócony rok, np. 25
- `m` – miesiąc z zerem wiodącym (01-12)
- `n` – miesiąc bez zera wiodącego (1-12)
- `d` – dzień miesiąca z zerem wiodącym (01-31)
- `j` – dzień miesiąca bez zera (1-31)
- `H` – godzina w formacie 24-godzinnym (00-23)
- `h` – godzina w formacie 12-godzinnym (01-12)
- `i` – minuty (00-59)
- `s` – sekundy (00-59)
- `l` – pełna nazwa dnia tygodnia, np. Monday
- `D` – skrót dnia tygodnia, np. Mon
- `N` – numer dnia tygodnia (1-7, gdzie 1 to poniedziałek)

Przykłady użycia date() w kodzie PHP:

```php
// Pełna data i godzina
echo date("Y-m-d H:i:s");
// 2025-05-05 14:00:00
// Tylko dzień i miesiąc
echo date("d.m");
// 05.05
// Nazwa dnia tygodnia i godzina
echo date("l, H:i");
// Monday, 14:00
// Data z własnym separatorem
echo date("d/m/Y");
// 05/05/2025
// Polski format: 05 maja 2025 (niestandardowe podejście)
setlocale(LC_TIME, "pl_PL.utf8");
echo strftime("%d %B %Y", time());
```

Uwaga: funkcja

`strftime()` umożliwia pełne wsparcie dla języków lokalnych (np. nazw miesięcy po polsku), ale wymaga odpowiednich ustawień lokalizacji systemowej. W nowoczesnym PHP lepiej używać obiektów `DateTime`, co omówimy niebawem.

Reasumując: funkcja

`date()` pozwala w bardzo elastyczny sposób sformatować dowolną datę lub znacznik czasu UNIX. Dzięki niej w prosty sposób uzyskamy czytelny dla użytkownika format daty – zgodnie z wymaganiami aplikacji lub użytkownika końcowego. Trzeba tylko nauczyć się podstawowych literałów używanych w formatowaniu dat.

Walidacja poprawności daty: checkdate()

W PHP bardzo przydatna okazuje się funkcja

`checkdate()`, która pozwala sprawdzić, czy podana data rzeczywiście istnieje. To szczególnie ważne np. przy walidacji formularzy, gdy użytkownik wpisuje ręcznie dzień, miesiąc i rok – musimy upewnić się, że np. 29 lutego występuje tylko w latach przestępnych.

Funkcja przyjmuje trzy parametry: **miesiąc**, **dzień** i **rok** – wszystkie jako liczby całkowite.

```
if (checkdate(2, 29, 2023)) {
    echo "Data poprawna!";
} else {
    echo "Błąd! 29 lutego 2023 nie istnieje.";
}
```

W tym przypadku otrzymamy komunikat błędu, ponieważ rok 2023 nie był przestępny – luty miał tylko 28 dni. Dzięki

`checkdate()` zabezpieczymy naszą aplikację przed wprowadzeniem nierealnych dat.

Nowoczesna data i czas w PHP: obiekt DateTime

PHP, choć kiedyś opierał się głównie na funkcjach proceduralnych, dziś oferuje nowoczesne podejście do pracy z datą i czasem – a mianowicie za pomocą obiektu

`DateTime`. To znacznie wygodniejsza i bardziej rozbudowana alternatywa dla tradycyjnych funkcji takich jak `date()` czy `time()`.

Stworzenie nowego obiektu

`DateTime` jest banalnie proste:

```
$data = new DateTime();
echo $data->format("Y-m-d H:i:s");
// Przykładowy wynik: 2025-05-05 12:34:56
```

Jeśli nie podamy żadnego argumentu, PHP przyjmie oczywiście aktualną datę i godzinę. Jednak nic nie stoi na przeszkodzie, aby utworzyć obiekt także z konkretną datą:

```
$data = new DateTime("2025-12-24 18:00:00");
echo $data->format("l, j F Y H:i");
// Wynik: Wednesday, 24 December 2025 18:00
```

Dzięki metodzie

`format()` mamy pełną kontrolę nad prezentacją daty i czasu. Używamy tu tych samych znaczników (literałów formatujących) co w funkcji `date()`, lecz oczywiście wywołujemy metodę w sposób obiektowy, z użyciem zapisu `obiekt->metoda()`.

Dodawanie i odejmowanie czasu

Chcemy dodać np. 5 dni do daty? Nic prostszego:

```php
$data = new DateTime("2025-12-24");
$data->modify("+5 days");
echo $data->format("Y-m-d");
// Wynik: 2025-12-29
```

Analogicznie możemy też oczywiście odjąć czas, np. 1 miesiąc wstecz:

```php
$data = new DateTime("2025-12-24");
$data->modify("-1 month");
echo $data->format("Y-m-d");
// Wynik: 2025-11-24
```

Metoda

`modify()` przyjmuje argument w formacie tekstowym – działa więc jak “mini-parser” języka naturalnego (np. “+1 day”, “-2 weeks”, “+3 months”).

Uzyskiwanie konkretnej części daty

Zamiast wycinać fragmenty z łańcucha, używajmy metody

`format()` do pobierania konkretnych informacji z daty:

```php
$data = new DateTime();
echo "Dzień tygodnia: " . $data->format("l");   // np. Monday
echo "Rok: " . $data->format("Y");              // np. 2025
echo "Numer tygodnia: " . $data->format("W");   // np. 18
```

To bardzo wygodne, gdy potrzebujemy szybko sprawdzić jaki dziś dzień, tydzień czy rok – np. do wyświetlania aktualnej daty w nagłówku strony lub tworzenia raportów.

Różnica między dwiema datami

Jak już wiemy, obiekt

`DateTime` pozwala także na porównywanie dat. Metoda `diff()` zwraca obiekt klasy `DateInterval`, z którego możemy uzyskać m.in. liczbę dni między datami.

```php
$start = new DateTime("2025-01-01");
$koniec = new DateTime("2025-12-31");
$roznica = $start->diff($koniec);
echo "Liczba dni: " . $roznica->days;
// Wynik: Liczba dni: 364
```

Dodatkowo możemy odczytać inne dane – np. liczbę miesięcy, lat czy oznaczenie czy druga data jest wcześniejsza:

```php
echo "Lata: " . $roznica->y;
echo "Miesięcy: " . $roznica->m;
echo "Ujemna różnica? " . ($roznica->invert ? "tak" : "nie");
```

To wszystko czyni klasę

`DateTime` potężnym narzędziem do pracy z czasem w PHP – nie tylko na egzaminie INF.03, ale również w profesjonalnych aplikacjach webowych.

W kolejnej części lekcji przyjrzymy się jeszcze funkcji

`strtotime()`, która zamienia tekst na datę i jest świetna do szybkich obliczeń bez użycia obiektów.

Przekształcanie tekstu na czas: strtotime()

Funkcja

`strtotime()` to jedno z najbardziej praktycznych narzędzi PHP do pracy z datami zapisanymi w postaci łańcucha tekstowego. Pozwala zamienić *czytelny zapis daty* (np. “next Monday” lub “2025-12-24 18:00:00”) na znacznik czasu UNIX, który poznaliśmy wcześniej, czyli przypomnijmy – liczbę sekund od 1 stycznia 1970 roku. Składnia funkcji jest banalna:

```php
$timestamp = strtotime("2025-12-24 18:00:00");
echo $timestamp;
// Wynik: 1766916000 (liczba sekund od 1970-01-01)
```

Wartością zwrotną jest **liczba całkowita**, którą możemy potem używać np. w funkcji

`date()`:

```php
$data = strtotime("tomorrow");
echo date("Y-m-d", $data);
// Wynik: data jutrzejsza, np. 2025-05-06
```

Można więc traktować

`strtotime()` jako swoisty parser języka naturalnego – przyjmuje wiele fraz w języku angielskim: `"now"`, `"yesterday"`, `"tomorrow"`, `"next Monday"`, `"last Friday"`, `"+1 day"`, `"+1 week"`, `"-3 months"`. Dzięki temu możemy dynamicznie obliczać nowe daty na podstawie wpisanego tekstu – idealne w formularzach, automatycznych raportach czy nawet testach jednostkowych.

Przykład strtotime() – obliczenie różnicy w dniach

```php
$start = strtotime("2025-05-01");
$koniec = strtotime("2025-05-15");
$roznica = ($koniec - $start) / (60 * 60 * 24);
echo "Różnica: $roznica dni";
// Wynik: Różnica: 14 dni
```

Jeśli nie potrzebujemy tworzyć obiektów

`DateTime`, a wystarczą nam szybkie obliczenia, to `strtotime()` może być idealnym wyborem – prostym, szybkim i kompatybilnym z funkcją `date()`.

Obsługa stref czasowych w PHP

W przypadku aplikacji internetowych strefa czasowa może mieć kluczowe znaczenie – szczególnie jeśli nasza strona obsługuje użytkowników z różnych krajów lub gdy serwer znajduje się w innej strefie czasowej niż nasz lokalny komputer. Przykładowo: serwer

zlokalizowany w Niemczech może mieć domyślną strefę “Europe/Berlin”, natomiast nasza strona działa w Polsce (“Europe/Warsaw”).

Aby upewnić się, że funkcje związane z datą (np.

`date()`, `time()`, `strtotime()`, `DateTime()`) korzystają z poprawnej strefy czasowej, należy ją jednoznacznie ustawić.

Ustawianie strefy czasowej w PHP

Ta funkcja pozwala ustawić globalną strefę czasową dla wszystkich operacji związanych z datą i godziną w aktualnym skrypcie:

```
date_default_timezone_set("Europe/Warsaw");
echo date("Y-m-d H:i:s");
// Wynik: aktualna data i czas w strefie Warszawy
```

Najczęściej stosowane identyfikatory stref czasowych:

- `Europe/Warsaw` – Polska
- `UTC` – Czas uniwersalny (z zerowym przesunięciem)
- `Europe/London` – Wielka Brytania
- `America/New_York` – Nowy Jork
- `Asia/Tokyo` – Japonia

Odczytywanie bieżącej strefy czasowej

Możemy sprawdzić, jaka strefa czasowa obowiązuje obecnie w naszym skrypcie:

```
echo date_default_timezone_get();
// Wynik: np. "UTC"
```

Czas w bazach danych: TIMESTAMP czy DATETIME?

Gdy pracujemy z bazą danych MySQL/MariaDB, mamy do wyboru kilka typów kolumn do przechowywania daty i czasu. Najczęściej używane to

`TIMESTAMP` oraz `DATETIME`.

Typ	Opis
`TIMESTAMP`	Zapisuje czas jako liczbę sekund od 1970-01-01 00:00:00 (format UNIX). Automatycznie przelicza się na lokalny czas serwera. Idealny do porównań i operacji matematycznych.
`DATETIME`	Zapisuje datę i czas w formacie tekstowym `YYYY-MM-DD HH:MM:SS`

Typ	Opis
	. Jest bardziej czytelny, ale nie przechowuje strefy czasowej.

Wybór odpowiedniego typu zależy od konkretnego zastosowania. Jeśli zależy nam na pełnej kontroli i precyzyjnym porównywaniu czasów – używajmy

`TIMESTAMP`. Jeśli zaś zależy nam na czytelności oraz prezentacji czasu w dokładnej postaci – lepszy będzie `DATETIME`.

Podsumowanie

Przetwarzanie daty i czasu to jedno z pojawiających się zagadnień na egzaminie INF.03, więc warto solidnie je opanować. Oczywiście do egzaminu powinny wystarczyć podstawowe, klasyczne operacje przetwarzania daty i czasu – niekoniecznie obiektowe. Lecz zrealizowaliśmy solidny przegląd obu rodzajów metod, aby nie dać się niczym zaskoczyć. Podsumujmy zatem wiedzę z lekcji:

- Funkcja `time()` zwraca aktualny znacznik czasu UNIX (liczbę sekund od 1 stycznia 1970 roku), co umożliwia łatwe porównywanie i obliczenia na czasie.
- Funkcja `mktime()` pozwala wygenerować dowolny znacznik czasu na podstawie wskazanych danych (rok, miesiąc, dzień, godzina, minuta, sekunda).
- Funkcja `date()` umożliwia sformatowanie znacznika czasu do czytelnej postaci tekstowej (np. “2025-05-05 14:00”).
- Funkcja `checkdate()` pozwala sprawdzić, czy podana data rzeczywiście istnieje – idealna do walidacji np. formularzy.
- Obiekt `DateTime` oraz jego metoda `modify()` umożliwiają nowoczesną, obiektową pracę z datami, w tym m.in. dodawanie/odejmowanie dni i porównania.
- Funkcja `strtotime()` zamienia tekstowy zapis daty (np. “next Monday”) na znacznik czasu UNIX – bardzo przydatne przy “podręcznych” obliczeniach.
- Dzięki `date_default_timezone_set()` możemy jednoznacznie ustawić strefę czasową dla skryptu PHP – domyślnie wiele serwerów korzysta z UTC lub czasu serwera, co warto mieć na uwadze.
- W bazach danych stosujemy zwykle kolumny typu TIMESTAMP (dla operacji matematycznych) lub DATETIME (dla lepszej czytelności). Wybór zależy od zastosowania i potrzeb projektu.