

INSTRUKCJA DLA UCZNIÓW — KWALIFIKACJA INF.03

Ciasteczka i sesje w PHP

Jak strona internetowa „pamięta” użytkownika — cztery przykłady krok po kroku

1. Po co ciasteczka i sesje?

Protokół HTTP nie ma pamięci. Każde wejście na stronę to osobne żądanie, a serwer nie wie, że to ta sama osoba co minutę wcześniej. Żeby strona mogła „pamiętać” użytkownika, PHP daje dwa narzędzia:

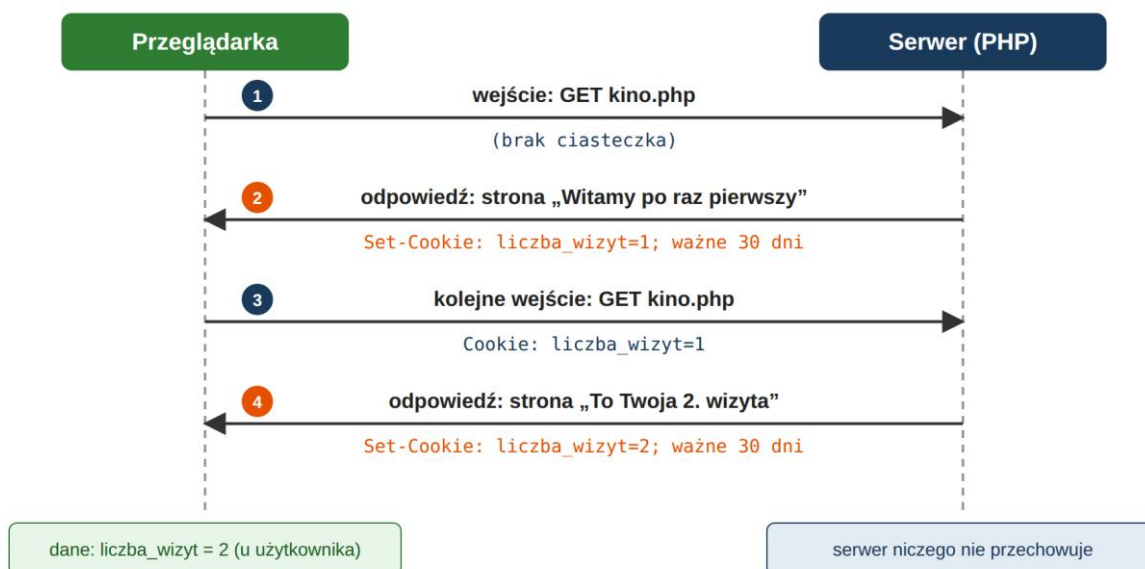
- **ciasteczko (cookie)** – mały plik tekstowy zapisany **w przeglądarce użytkownika**,
- **sesja (session)** – dane zapisane **na serwerze**; przeglądarka trzyma tylko numer sesji.

	Ciasteczko	Sesja
Gdzie są dane?	W przeglądarce użytkownika	Na serwerze (plik sesji)
Jak długo żyją?	Tyle, ile ustawimy (np. 2 godziny, 30 dni)	Zwykle do zamknięcia przeglądarki lub wylogowania
Czy użytkownik może je zmienić?	Tak – np. w narzędziach F12	Nie – widzi tylko losowy numer sesji
Tablica w PHP	\$_COOKIE	\$_SESSION
Zapis	setcookie(nazwa, wartość, wygaśnięcie)	\$_SESSION['klucz'] = wartość;
Typowe zastosowania	licznik odwiedzin, zapamiętany wybór, „Witaj ponownie”	logowanie, koszyk w sklepie, wynik quizu

2. Ciasteczka

Jak działa ciasteczko

Serwer wysyła ciasteczko razem z odpowiedzią. Przeglądarka zapisuje je u siebie i przy każdym kolejnym wejściu na tę stronę sama dołącza je do żądania. PHP odczytuje je wtedy z tablicy \$_COOKIE.



Rysunek 1. Wymiana ciasteczka między przeglądarką a serwerem (przykład A)

Najważniejsze polecenia

Polecenie	Co robi
<code>setcookie("imie", "Ania", time() + 3600);</code>	Tworzy ciasteczko imie o wartości Ania, ważne 1 godzinę.
<code>isset(\$_COOKIE['imie'])</code>	Sprawdza, czy ciasteczko istnieje (true / false).
<code>\$_COOKIE['imie']</code>	Odczytuje wartość ciasteczka – zawsze jako tekst.
<code>setcookie("imie", "", time() - 3600);</code>	Usuwa ciasteczko: czas wygaśnięcia w przeszłości.

Jak liczyć czas ważności

Trzeci argument `setcookie()` to moment wygaśnięcia, a nie długość. Funkcja `time()` zwraca bieżący czas jako liczbę sekund od 1 stycznia 1970 r., więc do niej dodajemy czas ważności w sekundach:

Czas ważności	Obliczenie	Trzeci argument
1 godzina	$60 \times 60 = 3600 \text{ s}$	<code>time() + 3600</code>
2 godziny	$2 \times 60 \times 60 = 7200 \text{ s}$	<code>time() + 7200</code>
1 dzień	$24 \times 60 \times 60 = 86\,400 \text{ s}$	<code>time() + 86400</code>
30 dni	$30 \times 24 \times 60 \times 60$	<code>time() + 30 * 24 * 60 * 60</code>
do zamknięcia przeglądarki	–	pomijamy trzeci argument

Ważne

`setcookie()` wysyła nagłówek HTTP, dlatego trzeba je wywołać, zanim strona wyśle jakikolwiek kod HTML. Najbezpieczniej umieścić skrypt z ciasteczkiem na samym początku pliku, przed `<!DOCTYPE html>`, a w treści strony wypisać tylko przygotowany komunikat.

Przykład A — Kino Pod Gwiazdami: licznik odwiedzin

Strona kina wita nowego gościa, a stałemu pokazuje, który raz ją odwiedza. Liczba wizyt jest zapisana w ciasteczku ważnym 30 dni.

• • • kino.php
<code><?php</code>
<code>// kino.php - licznik odwiedzin zapisany w ciasteczku</code>
<code></code>
<code>if (isset(\$_COOKIE['liczba_wizyt'])) {</code>
<code> // ciasteczko istnieje → użytkownik już tu był</code>
<code> // wartość ciasteczka to zawsze TEKST, więc zamieniamy ją na liczbę (int)</code>
<code> \$wizyty = (int)\$_COOKIE['liczba_wizyt'] + 1;</code>
<code> \$komunikat = "Miło Cię znowu widzieć! To Twoja " . \$wizyty . ". wizyta.";</code>
<code>} else {</code>
<code> // brak ciasteczka → pierwsza wizyta</code>
<code> \$wizyty = 1;</code>
<code> \$komunikat = "Witamy po raz pierwszy w Kinie Pod Gwiazdami!";</code>
<code>}</code>
<code></code>
<code>// zapisujemy nową liczbę wizyt; ciasteczko ważne 30 dni</code>
<code>setcookie("liczba_wizyt", \$wizyty, time() + 30 * 24 * 60 * 60);</code>
<code>?></code>
<code><!DOCTYPE html></code>
<code><html lang="pl"></code>

<head>
<meta charset="UTF-8">
<title>Kino Pod Gwiazdami</title>
</head>
<body>
Kino Pod Gwiazdami – repertuar
<?php echo \$komunikat; ?>
</body>
</html>

Co się dzieje krok po kroku:

1. **Pierwsza wizyta:** ciasteczka liczba_wizyt nie ma, więc \$wizyty = 1 i wyświetla się powitanie. setcookie() zapisuje w przeglądarce wartość 1.
2. **Druga wizyta:** przeglądarka wysyła ciasteczko z wartością "1". Skrypt zamienia tekst na liczbę, dodaje 1 i zapisuje 2.
3. **Każda kolejna wizyta:** licznik rośnie, a ciasteczko jest nadpisywane – 30 dni liczy się od ostatniej wizyty.

Przykład B — Księgarnia Zakładka: zapamiętany wybór z formularza

Użytkownik wybiera z listy ulubioną kategorię książek. Strona zapamiętuje ją na 7 dni i przy kolejnych wizytach poleca nowości z tej kategorii. Drugi przycisk usuwa zapamiętany wybór.

• • • ksiegarnia.php
<?php
// ksiegarnia.php – zapamiętanie ulubionej kategorii na 7 dni
\$kategorie = ["kryminał", "fantastyka", "reportaż"];
\$ulubiona = "";
if (isset(\$_POST['zapomnij'])) {
// USUNIĘCIE ciasteczka: czas wygaśnięcia w przeszłości
setcookie("ulubiona_kategoria", "", time() - 3600);
} elseif (isset(\$_POST['kategoria']) && in_array(\$_POST['kategoria'], \$kategorie)) {
// użytkownik właśnie wybrał kategorię w formularzu
\$ulubiona = \$_POST['kategoria'];
setcookie("ulubiona_kategoria", \$ulubiona, time() + 7 * 24 * 60 * 60);
} elseif (isset(\$_COOKIE['ulubiona_kategoria'])) {
// kolejna wizyta – odczytujemy wybór zapisany w ciasteczku
\$ulubiona = \$_COOKIE['ulubiona_kategoria'];
}
?>
<!DOCTYPE html>
<html lang="pl">
<head>
<meta charset="UTF-8">
<title>Księgarnia Zakładka</title>
</head>
<body>
Księgarnia Zakładka
<?php if (\$ulubiona != "") : ?>
<p>Polecamy nowości z kategorii: <?php echo htmlspecialchars(\$ulubiona);
?></p>
<?php else : ?>
Wybierz ulubioną kategorię, a zapamiętamy ją na 7 dni.
<?php endif; ?>

```

<form method="post">
  <select name="kategoria">
    <option>kryminał</option>
    <option>fantastyka</option>
    <option>reportaż</option>
  </select>
  <button type="submit">Zapamiętaj</button>
</form>

<form method="post">
  <button type="submit" name="zapomnij">Zapomnij mój wybór</button>
</form>
</body>
</html>

```

Dlaczego w drugim warunku używamy `$_POST`, a nie `$_COOKIE`?

Ciasteczko zapisane przez `setcookie()` pojawia się w tablicy `$_COOKIE` dopiero przy NASTĘPNYM wejściu na stronę – przeglądarka musi je najpierw dostać i odesłać. Gdybyśmy zaraz po `setcookie()` odczytali `$_COOKIE`, zobaczylibyśmy starą wartość albo nic. Dlatego w chwili wyboru korzystamy z wartości przesłanej formularzem.

`in_array()` sprawdza, czy przesłana kategoria jest na liście dozwolonych. Dzięki temu nikt nie zapisze w ciasteczku dowolnego tekstu, podmieniając wartość w formularzu.

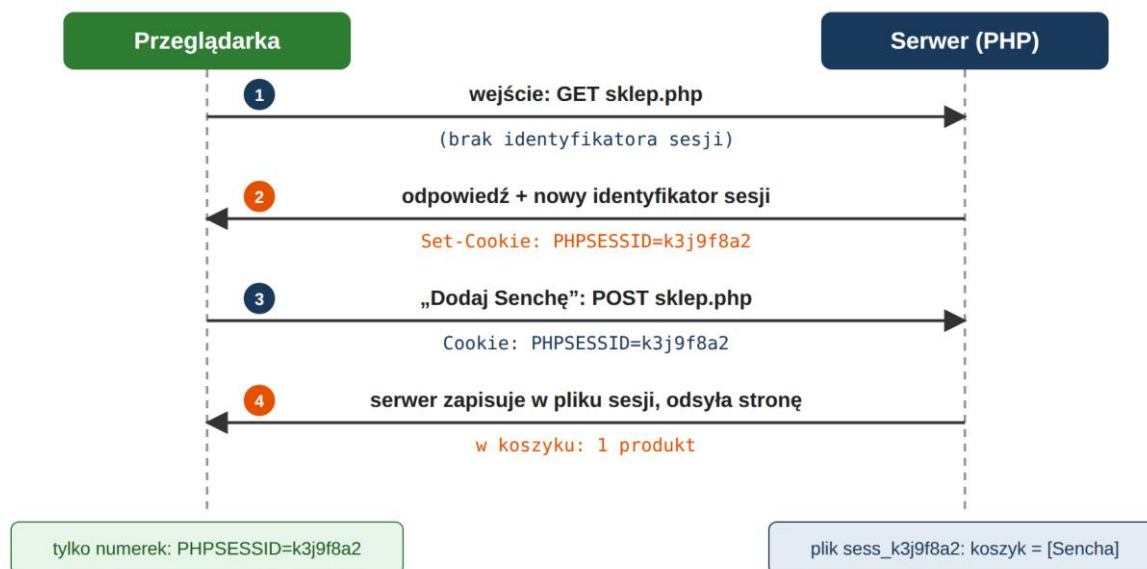
Jak podejrzeć ciasteczka w przeglądarce

Naciśnij F12 → zakładka Aplikacja (Application) → Pliki cookie → adres strony. Zobaczysz nazwę, wartość i datę wygaśnięcia każdego ciasteczka. Możesz też ręcznie zmienić wartość albo usunąć ciasteczko, żeby sprawdzić, jak strona zachowa się przy „pierwszej wizycie”.

3. Sesje

Jak działa sesja

Przy pierwszym wywołaniu `session_start()` PHP tworzy na serwerze plik sesji i losuje dla niego identyfikator. Identyfikator trafia do przeglądarki w ciasteczku `PHPSESSID`. Przy kolejnych wejściach przeglądarka odsyła ten numer, a PHP wczytuje dane z pliku do tablicy `$_SESSION`.



Rysunek 2. Sesja: dane na serwerze, w przeglądarce tylko numer (przykład C)

Polecenie	Co robi
<code>session_start();</code>	Rozpoczyna nową lub wczytuje istniejącą sesję. Na początku KAŻDEJ strony, która korzysta z sesji.
<code>\$_SESSION['login'] = "ania";</code>	Zapisuje dane w sesji. Mogą to być też tablice.
<code>isset(\$_SESSION['login'])</code>	Sprawdza, czy w sesji są zapisane dane.
<code>unset(\$_SESSION['login']);</code>	Usuwa jedną zmienną z sesji.
<code>session_unset(); session_destroy();</code>	Czyści wszystkie dane i niszczy sesję (wylogowanie).

Przykład C — Herbaciarnia Liść: koszyk w sesji

Klient dodaje herbaty do koszyka. Koszyk to tablica zapisana w `$_SESSION`, więc zawartość zostaje po odświeżeniu strony i przechodzeniu między podstronami. Przycisk „Wyczyść koszyk” opróżnia tablicę.

• • • sklep.php
<pre><?php // sklep.php – koszyk w sklepie z herbatą przechowywany w sesji session_start(); // cennik: nazwa herbaty => cena w zł \$herbaty = ["Earl Grey" => 18.50, "Sencha" => 24.00, "Rooibos" => 15.90]; // przy pierwszym wejściu koszyka jeszcze nie ma – tworzymy pustą tablicę if (!isset(\$_SESSION['koszyk'])) { \$_SESSION['koszyk'] = []; } // przycisk "Dodaj" – dopisujemy nazwę herbaty na koniec tablicy w sesji if (isset(\$_POST['dodaj']) && isset(\$herbaty[\$_POST['dodaj']])) { \$_SESSION['koszyk'][] = \$_POST['dodaj']; } // przycisk "Wyczyść koszyk" – zastępujemy koszyk pustą tablicą if (isset(\$_POST['wyczysc'])) { \$_SESSION['koszyk'] = []; }</pre>

```
// liczymy wartość koszyka
$suma = 0;
foreach ($_SESSION['koszyk'] as $nazwa) {
    $suma = $suma + $herbaty[$nazwa];
}
?>
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Herbaciarnia Liść</title>
</head>
<body>
    <h1>Herbaciarnia Liść</h1>

    <h2>Nasze herbaty</h2>
    <form method="post">
        <?php foreach ($herbaty as $nazwa => $cena) : ?>
            <p>
                <?php echo $nazwa . " - " . number_format($cena, 2, ",", " ") . " zł"; ?>
                <!-- value przycisku trafia do $_POST['dodaj'] -->
                <button type="submit" name="dodaj" value="<?php echo $nazwa; ?>">Dodaj</button>
            </p>
        <?php endforeach; ?>
    </form>

    <h2>Koszyk (<?php echo count($_SESSION['koszyk']); ?> szt.)</h2>
    <ul>
        <?php foreach ($_SESSION['koszyk'] as $nazwa) : ?>
            <li><?php echo $nazwa; ?></li>
        <?php endforeach; ?>
    </ul>
    <p>Do zapłaty: <strong><?php echo number_format($suma, 2, ",", " "); ?> zł</strong></p>

    <form method="post">
        <button type="submit" name="wyczysc">Wyczyść koszyk</button>
    </form>
</body>
</html>
```

Na co zwrócić uwagę:

- `$_SESSION['koszyk'][] = ...` – puste nawiasy `[]` dopisują nowy element na końcu tablicy.
- **Przycisk z atrybutem value** – każdy przycisk „Dodaj” ma tę samą nazwę `dodaj`, ale inną wartość. Do `$_POST['dodaj']` trafia wartość klikniętego przycisku.
- `isset($herbaty[$_POST['dodaj']])` – do koszyka trafi tylko herbata, która jest w cenniku.
- `count()` zwraca liczbę elementów tablicy, a `number_format()` zapisuje cenę jako 18,50.

Przykład D — Biblioteka szkolna: logowanie i strona tylko dla zalogowanych

Czytelnik loguje się do panelu biblioteki. Po poprawnym logowaniu w sesji zapisywany jest login i godzina logowania, a skrypt przenosi użytkownika do panelu. Panel sprawdza sesję i odsyła niezalogowanych do strony logowania.

Tabela czytelników w bazie biblioteka:

• • • czytelnicy.sql

```
CREATE TABLE czytelnicy (
  id INT AUTO_INCREMENT PRIMARY KEY,
  login VARCHAR(50) NOT NULL UNIQUE,
  haslo VARCHAR(255) NOT NULL -- miejsce na skrót z password_hash()
);
```

Konto czytelnika zakładamy formularzem rejestracji, który zapisuje hasło funkcją `password_hash()` – tak jak w pliku `rejestracja_bezpieczna.php` z Przychodni (plik `rejestracja.php` jest w paczce z przykładami).

Poleceniem `INSERT` w `phpMyAdmin` nie da się utworzyć poprawnego skrótu.

• • • logowanie.php

```
<?php
// logowanie.php - logowanie do panelu biblioteki szkolnej
session_start();

$komunikat = "";

if (isset($_POST['zaloguj'])) {
  $conn = mysqli_connect("localhost", "root", "", "biblioteka");
  mysqli_set_charset($conn, "utf8mb4");

  // zapytanie przygotowane - login trafia w miejsce ? jako dane, nie jako kod SQL
  $st = mysqli_prepare($conn, "SELECT login, haslo FROM czytelnicy WHERE login = ?");
  mysqli_stmt_bind_param($st, "s", $_POST['login']);
  mysqli_stmt_execute($st);
  $czytelnik = mysqli_fetch_assoc(mysqli_stmt_get_result($st));

  // password_verify porównuje wpisane hasło ze skrótem zapisanym w bazie
  if ($czytelnik && password_verify($_POST['haslo'], $czytelnik['haslo'])) {
    session_regenerate_id(true); // nowy identyfikator sesji
    $_SESSION['login'] = $czytelnik['login']; // SESJA ROZPOCZĘTA
    $_SESSION['czas_logowania'] = date("H:i"); // w sesji można trzymać więcej danych

    header("Location: panel.php"); // przejście do panelu
    exit;
  } else {
    $komunikat = "Błędny login lub hasło.";
  }
}
?>

<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Biblioteka - logowanie</title>
</head>
<body>
  <h1>Biblioteka szkolna - logowanie</h1>
  <p><?php echo $komunikat; ?></p>
  <form method="post">
    <label>Login: <input type="text" name="login" required></label><br>
    <label>Hasło: <input type="password" name="haslo" required></label><br>
    <button type="submit" name="zaloguj">Zaloguj</button>
  </form>
</body>
</html>
```

• • • panel.php

```
<?php
```

```
// panel.php – strona dostępna TYLKO dla zalogowanych
session_start();

// brak loginu w sesji = ktoś wszedł tu bez logowania → odsyłamy go
if (!isset($_SESSION['login'])) {
    header("Location: logowanie.php");
    exit;
}
?>
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Biblioteka – panel czytelnika</title>
</head>
<body>
    <h1>Panel czytelnika</h1>
    <p>Witaj, <strong><?php echo htmlspecialchars($_SESSION['login']); ?></strong>!</p>
    <p>Zalogowano o godzinie: <?php echo $_SESSION['czas_logowania']; ?></p>
    <p><a href="wyloguj.php">Wyloguj</a></p>
</body>
</html>
```

```
• • • wyloguj.php
<?php
// wyloguj.php – zniszczenie sesji
session_start(); // otwieramy sesję, żeby wiedzieć, którą usunąć
session_unset(); // czyścimy $_SESSION
session_destroy(); // usuwamy sesję z serwera
header("Location: logowanie.php");
exit;
?>
```

Co się dzieje krok po kroku:

1. Wpisanie adresu panel.php bez logowania: w sesji nie ma loginu, więc header() od razu przenosi na logowanie.php.
2. Poprawne logowanie: password_verify() zwraca true, w sesji zapisują się login i godzina, a header() przenosi do panelu.
3. Panel: session_start() wczytuje sesję, warunek jest spełniony, strona wita użytkownika i pokazuje godzinę logowania.
4. Wylogowanie: sesja zostaje zniszczona. Próba powrotu do panel.php (np. strzałką „Wstecz” i odświeżeniem) znów kończy się przeniesieniem na logowanie.

header("Location: ...") i exit

Funkcja header() wysyła przeglądarce polecenie przejścia pod inny adres. Tak jak setcookie(), musi być wywołana przed kodem HTML. Po niej zawsze piszemy exit, żeby reszta strony – np. treść panelu – nie została wysłana.

4. Ciasteczko czy sesja?

Sytuacja	Wybierz
Zapamiętanie czegoś na długo, także po zamknięciu przeglądarki (licznik wizyt, motyw strony, ulubiona kategoria)	Ciasteczko
Komunikat „Witaj ponownie” lub data ostatniej wizyty	Ciasteczko
Informacja, kto jest zalogowany	Sesja
Koszyk, wynik quizu, dane formularza rozłożonego na kilka stron	Sesja
Wszystko, czego użytkownik nie może zmienić ani zobaczyć	Sesja

Nigdy nie zapisuj logowania w ciasteczku

Kod `setcookie("zalogowany", "tak")` albo `setcookie("login", "admin")` to poważny błąd. Każdy może w narzędziach F12 dopisać sobie takie ciasteczko i zostać „zalogowanym” bez hasła – także jako administrator. W sesji dane leżą na serwerze, a użytkownik ma tylko losowy numer, którego nie da się zgadnąć.

5. Najczęstsze błędy

Błąd	Skutek i poprawka
<code>setcookie()</code> lub <code>header()</code> po kodzie HTML	Ostrzeżenie „headers already sent”. Przenieś skrypt na początek pliku.
<code>setcookie("x", "y", 7200)</code> – bez <code>time()</code>	Data wygaśnięcia z 1970 r., ciasteczko od razu znika. Pisz <code>time() + 7200</code> .
Odczyt <code>\$_COOKIE</code> zaraz po <code>setcookie()</code>	Nowa wartość będzie dostępna dopiero przy następnym wejściu. Użyj zmiennej z bieżącą wartością.
Brak <code>session_start()</code> na którejś stronie	Na tej stronie <code>\$_SESSION</code> jest puste – użytkownik „nie jest zalogowany”.
Brak <code>exit</code> po <code>header("Location: ...")</code>	Skrypt działa dalej i może wysłać chronioną treść.
Logowanie zapisane w ciasteczku	Każdy może podrobić ciasteczko. Używaj sesji.

6. Ćwiczenia do samodzielnego wykonania

- Pizzeria Margherita.** Formularz z polem „Twoje imię”. Po wysłaniu strona zapamiętuje imię w ciasteczku na 1 dzień i przy kolejnych wizytach wyświetla „Cześć, <imię>! Co dziś zamawiasz?”. Dodaj przycisk „To nie ja”, który usuwa ciasteczko.
- Siłownia FitLand.** Na jednej stronie wyświetl dwa liczniki odwiedzin: jeden w sesji, drugi w ciasteczku ważnym 7 dni. Odśwież stronę kilka razy, zamknij przeglądarkę, otwórz ją ponownie i opisz, czym różnią się wyniki.
- Szkolny dziennik.** Na podstawie przykładu D utwórz stronę `oceny.php` dostępną tylko dla zalogowanych uczniów. Niezalogowany użytkownik ma zostać przeniesiony na stronę logowania z komunikatem „Najpierw się zaloguj”.

4. **Quiz z historii.** Trzy pytania na trzech osobnych stronach. Odpowiedzi zapisuj w sesji, a na czwartej stronie wyświetl wynik, np. „Twój wynik: 2/3”, i przycisk „Zaczynij od nowa”, który czyści sesję.

Materiał opracowany na potrzeby zajęć z programowania aplikacji internetowych — technikum informatyczne, kwalifikacja INF.03.