
Sesje w PHP

Jak strona zapamiętuje zalogowanego użytkownika

Materiał dla uczniów, wyjaśniony krok po kroku na przykładzie arkusza:

„Portal laboratorium”

Z tego materiału dowiesz się:

- czym jest sesja i po co jest potrzebna,
- jak przeglądarka i serwer „rozmawiają” ze sobą przy logowaniu,
- co dokładnie robi każda funkcja: `session_start()`, `$_SESSION`, `isset()`, `unset()`, `session_destroy()`, `header()`,
- jak napisać skrypty logowania, ochrony strony i wylogowania tak, żeby dostać wszystkie punkty.

1. Po co w ogóle jest sesja?

Wyobraź sobie, że wchodzisz na stronę laboratorium. Najpierw logujesz się na stronie **login.php**, a potem przechodzisz na stronę **wyniki.php**, gdzie widać wyniki badań pacjentów. Wydaje się to oczywiste: skoro przed chwilą podałeś login i hasło, strona powinna wiedzieć, że to Ty.

Problem polega na tym, że **serwer niczego sam z siebie nie pamięta**. Przeglądarka i serwer komunikują się za pomocą protokołu **HTTP**, a HTTP jest **bezstanowy**. To znaczy, że każde wejście na stronę jest dla serwera zupełnie nową, osobną sprawą. Serwer obsługuje żądanie, wysyła odpowiedź i od razu o Tobie zapomina.

Gdyby nie było sesji, po przejściu z login.php na wyniki.php serwer nie miałby pojęcia, że już się logowałeś. Musiałbyś podawać hasło na każdej stronie od nowa.

PRZYKŁAD Z ŻYCIA Szatnia w teatrze

Sesja działa jak szatnia. Oddajesz kurtkę (swoje dane), a w zamian dostajesz **numerek**. Kurtka zostaje w szatni, a Tynosisz przy sobie tylko numerek.

Gdy wracasz, pokazujesz numerek i szatniarz od razu wie, która kurtka jest Twoja. Nie musisz jej opisywać.

W PHP: **szatnia** to serwer, **kurtka** to dane w `$_SESSION` (np. nazwa użytkownika), a **numerek** to identyfikator sesji zapisany w ciasteczku `PHPSESSID`.

Sesja rozwiązuje więc problem „zapominalskiego” serwera. Dzięki niej serwer może zapamiętać informacje o konkretnym użytkowniku między kolejnymi stronami. Najczęściej zapamiętuje to, **kto jest zalogowany**.

ZAPAMIĘTAJ

Sesja to **schowek na dane jednego użytkownika, przechowywany na serwerze**.

Przeglądarka dostaje tylko **numer** tego schowka (identyfikator sesji) i odsyła go przy każdym kolejnym wejściu na stronę.

2. Słowniczek – pojęcia, które musisz znać

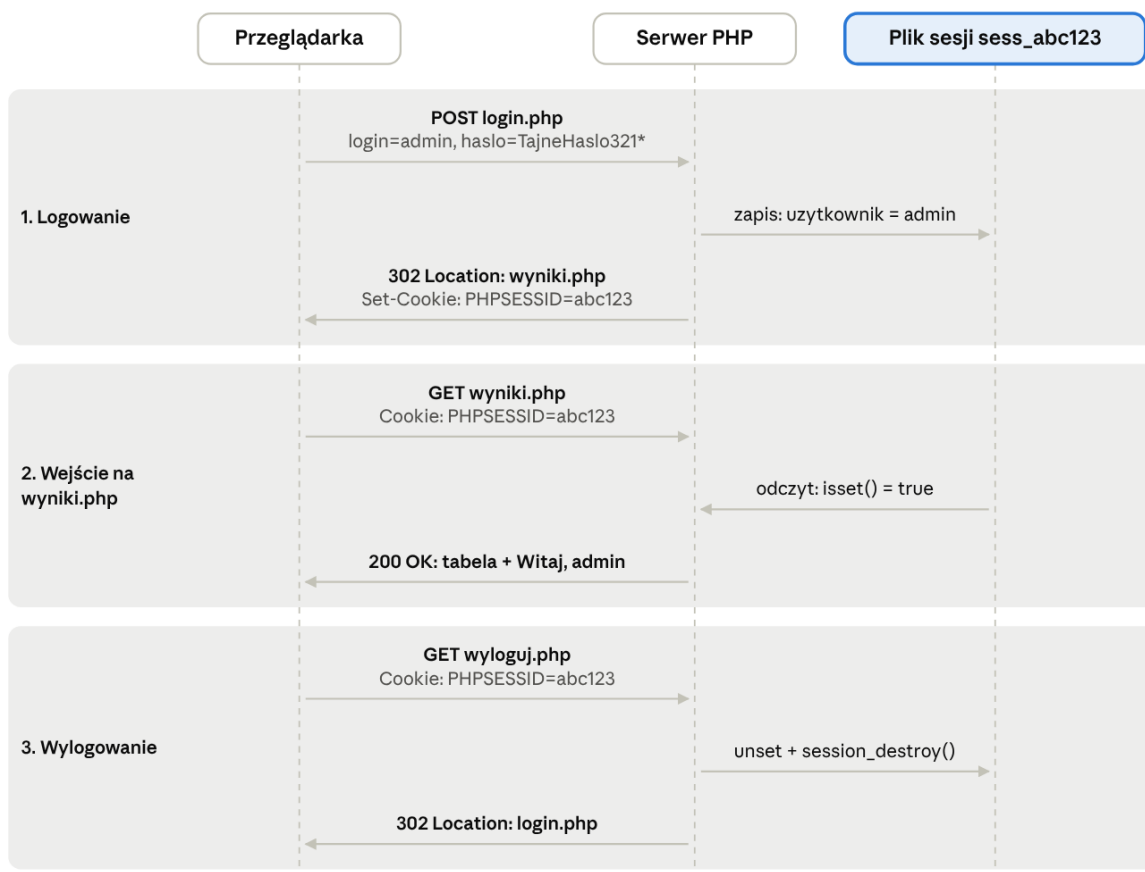
Zanim przejdziemy do kodu, upewnijmy się, że rozumiesz podstawowe pojęcia. Wrócisz do nich jeszcze wiele razy.

Pojęcie	Co to znaczy (prostymi słowami)
Przeglądarka (klient)	Program, w którym oglądasz strony: Chrome, Firefox, Edge. Wysyła prośby do serwera i wyświetla to, co dostanie.
Serwer	Komputer (u Ciebie XAMPP z Apache), który przechowuje pliki strony, uruchamia kod PHP i odsyła gotowy HTML do przeglądarki.
Żądanie (request)	Prośba przeglądarki do serwera, np. „daj mi stronę wyniki.php”. Każde kliknięcie linku albo wysłanie formularza to nowe żądanie.
Odpowiedź (response)	To, co serwer odsyła: kod HTML strony, a czasem polecenie „idź na inną stronę”.
HTTP	Zestaw zasad (protokół), według którego przeglądarka i serwer się porozumiewają. Jest bezstanowy : nie pamięta poprzednich żądań.
Ciasteczko (cookie)	Mały kawałek tekstu, który serwer każe przeglądarce zapamiętać. Przeglądarka sama odsyła go przy każdym kolejnym żądaniu do tej samej strony.
Identyfikator sesji	Długi, losowy ciąg znaków, np. <code>k3j9a8f2...</code> . To „numerek z szatni”. PHP zapisuje go w ciasteczku o nazwie <code>PHPSESSID</code> .
Zmienna sesyjna	Jedna wartość zapisana w sesji, np. <code>\$_SESSION['uzytkownik']</code> . Jest pamiętana między stronami.
Nagłówek HTTP	Dodatkowa informacja wysyłana razem z odpowiedzią, przed treścią strony. Przykład: polecenie przekierowania <code>Location</code> .
Przekierowanie	Polecenie serwera: „nie pokazuj tej strony, tylko idź na inny adres”. W PHP robi się je funkcją <code>header("Location: ...")</code> .
Superglobalna tablica	Specjalna tablica PHP dostępna w każdym miejscu kodu: <code>\$_POST</code> , <code>\$_GET</code> , <code>\$_SESSION</code> , <code>\$_COOKIE</code> .

3. Jak działa sesja – krok po kroku

Poniższy rysunek pokazuje całą „rozmowę” przeglądarki z serwerem w arkuszu Portal laboratorium. Są tu trzy etapy: logowanie, wejście na stronę z wynikami i wylogowanie. Strzałki pokazują, kto do kogo wysyła informację.

Przeglądarka nosi tylko numer sesji, dane leżą na serwerze



Rysunek 1. Przebieg sesji w arkuszu „Portal laboratorium”

Etap 1: Logowanie

1. Wpisujesz login `admin` i hasło `TajneHaslo321*`, a potem klikasz **Zaloguj**. Przeglądarka wysyła formularz metodą **POST** do pliku `login.php`.
2. Na serwerze uruchamia się `login.php`. Pierwsza instrukcja to `session_start()`. To Twoja pierwsza wizyta, więc PHP **tworzy nową sesję**: losuje identyfikator (np. `abc123`) i zakłada na serwerze pusty plik `sess_abc123`.
3. Skrypt sprawdza login i hasło. Są poprawne, więc wykonuje `$_SESSION['uzytkownik'] = $login;`. PHP zapisuje w pliku sesji: `uzytkownik = admin`.
4. Serwer odsyła odpowiedź z dwoma nagłówkami: **Set-Cookie: PHPSESSID=abc123** („zapamiętaj ten numer”) oraz **Location: wyniki.php** („idź na tę stronę”).

Etap 2: Wejście na wyniki.php

1. Przeglądarka posłusznie przechodzi na wyniki.php. Do żądania **sama dołącza** ciasteczko `PHPSESSID=abc123`. Nie musisz nic programować, przeglądarka robi to automatycznie.
2. Na serwerze wyniki.php zaczyna od `session_start()`. Tym razem PHP widzi numerkę abc123, więc **nie tworzy nowej sesji**, tylko otwiera plik `sess_abc123` i wczytuje jego zawartość do tablicy `$_SESSION`.
3. Skrypt sprawdza `isset($_SESSION['uzytkownik'])`. Wynik to **true**, bo zmienna istnieje. Użytkownik jest zalogowany, więc strona wyświetla tabelę i napis „Witaj, admin”.

Etap 3: Wylogowanie

1. Klikasz link **Wyloguj**. Przeglądarka otwiera wyloguj.php i znowu dołącza ciasteczko z numerkiem.
2. Skrypt wywołuje `session_start()` (żeby wiedzieć, o którą sesję chodzi), a potem `unset()` i `session_destroy()`. Plik sesji na serwerze zostaje usunięty.
3. Serwer przekierowuje do login.php. Numerki w ciasteczku jest teraz bezużyteczny, bo „kurtki w szatni” już nie ma. Gdy spróbujesz wejść na wyniki.php, zostaniesz odesłany do logowania.

ZAPAMIĘTAJ

Przeglądarka **nigdy nie dostaje** loginu ani innych danych z sesji. Ma tylko numer. Dane leżą bezpiecznie na serwerze.

`session_start()` działa na dwa sposoby: przy pierwszej wizycie **tworzy** sesję, przy kolejnych **wznawia** istniejącą.

4. Sesja a ciasteczko – czym się różnią?

Uczniowie często mylą sesję z ciasteczkiem, bo sesja **korzysta** z ciasteczka. To jednak dwie różne rzeczy.

Cecha	Sesja (\$_SESSION)	Ciasteczko (\$_COOKIE)
Gdzie są dane?	Na serwerze (w pliku)	W przeglądarce użytkownika
Czy użytkownik może je zmienić?	Nie, nie ma do nich dostępu	Tak, np. w narzędziach przeglądarki (F12)
Co trafia do przeglądarki?	Tylko identyfikator (PHPSESSID)	Cała wartość ciasteczka
Jak długo żyje?	Do <code>session_destroy()</code> , zamknięcia przeglądarki albo ok. 24 minut bezczynności	Tyle, ile ustawi programista (może nawet rok)
Do czego się nadaje?	Logowanie, koszyk w sklepie, dane poufne	Zapamiętanie języka, motywu, zgody na cookies
Jak zapisać?	<code>\$_SESSION['klucz'] = wartosc;</code>	<code>setcookie("klucz", wartosc, czas);</code>

UWAGA Dlaczego logowania nie robi się na ciasteczku?

Gdyby strona zapisała w ciasteczku `uzytkownik=admin`, każdy mógłby w przeglądarce zmienić tę wartość i „zalogować się” jako ktoś inny bez hasła.

W sesji jest to niemożliwe. Użytkownik ma tylko losowy numer, a to, czy jest zalogowany, wie wyłącznie serwer.

Gdzie fizycznie leży sesja w XAMPP?

Pliki sesji znajdziesz w folderze `C:\xampp\tmp`. Każdy ma nazwę w stylu `sess_k3j9a8f2...`. Jeśli otworzysz taki plik w Notatniku po zalogowaniu, zobaczysz mniej więcej coś takiego:

```
uzytkownik|s:5:"admin";
```

To znaczy: zmienna **uzytkownik**, typ **s** (string, czyli tekst), długość **5** znaków, wartość **"admin"**. Nie musisz tego umieć na egzamin, ale dobrze to zobaczyć, żeby sesja przestała być „magią”.

Jak podejrzeć ciasteczko sesji w przeglądarce?

1. Zaloguj się na swojej stronie.
2. Naciśnij **F12**, żeby otworzyć narzędzia deweloperskie.
3. W Chrome i Edge wejdź w zakładkę **Application** → **Cookies** → **http://localhost**. W Firefoxie: **Dane** (Storage) → **Ciasteczka**.
4. Zobaczysz ciasteczko **PHPSESSID** z długim losowym ciągiem znaków. To właśnie Twój „numerek z szatni”.

5. Funkcje i polecenia – każde dokładnie wyjaśnione

W arkuszu w Tabeli 2 masz cztery elementy do obsługi sesji, a w Tabeli 3 funkcję `header()`. Poniżej omawiamy każdy z nich i kilka innych, które są potrzebne w skryptach.

5.1. `session_start()` – uruchomienie sesji

```
session_start();
```

Co robi: uruchamia mechanizm sesji. Jeżeli przeglądarka przysłała ciasteczko PHPSESSID, PHP wczytuje zapisane dane do tablicy `$_SESSION`. Jeżeli nie przysłała (pierwsza wizyta), PHP tworzy nową, pustą sesję i wysyła przeglądarce ciasteczko.

Gdzie wpisać: jako **pierwszą instrukcję** w pliku, zaraz po `<?php`, i to w **każdym** pliku, który korzysta z sesji. W naszym arkuszu: `login.php`, `wyniki.php` i `wyloguj.php`.

UWAGA Dwie najczęstsze pułapki

1. Brak `session_start()` w którymś pliku. Tablica `$_SESSION` jest wtedy pusta i strona „nie widzi”, że jesteś zalogowany.
2. Coś przed `<?php`, nawet jedna spacja albo pusta linia. `session_start()` wysyła ciasteczko w nagłówku, a nagłówki muszą iść **przed** jakąkolwiek treścią strony.

5.2. `$_SESSION` – zapis i odczyt danych

`$_SESSION` to zwykła tablica asocjacyjna (z kluczami-napisami), tylko że PHP pamięta ją między stronami. Klucz wymyślasz sam. W tym materiale używamy klucza **uzytkownik**.

```
// ZAPIS – ustawienie zmiennej sesyjnej (w login.php)
$_SESSION['uzytkownik'] = $login;

// ODCZYT – pobranie wartości (w wyniki.php)
echo $_SESSION['uzytkownik']; // wypisze: admin
```

ZAPAMIĘTAJ

Klucz musi być **identyczny** we wszystkich plikach. Jeśli w `login.php` zapiszesz `$_SESSION['uzytkownik']`, a w `wyniki.php` sprawdzasz `$_SESSION['user']`, strona będzie Cię w kółko odsyłać do logowania.

5.3. `isset()` – czy zmienna istnieje?

```
isset($_SESSION['uzytkownik']) // true - zmienna jest ustawiona
                                // false - zmiennej nie ma
```

Co robi: zwraca **true** (prawda), jeśli zmienna istnieje i nie jest pusta (null), albo **false** (fałsz), jeśli jej nie ma. W praktyce pytamy: „czy użytkownik jest zalogowany?”.

Często łączy się ją z wykrzyknikiem **!**, który odwraca wynik. `!isset(...)` czytamy: „**jeśli NIE jest ustawiona**”, czyli jeśli użytkownik **NIE** jest zalogowany.

`isset()` przydaje się też przy formularzu. `isset($_POST['login'])` sprawdza, czy formularz został wysłany. Przy zwykłym otwarciu strony `$_POST` jest pusta.

5.4. unset() – usunięcie jednej zmiennej

```
unset($_SESSION['uzytkownik']);
```

Co robi: usuwa z sesji jedną, wskazaną zmienną. Sesja nadal istnieje, ale nie ma już w niej informacji o użytkowniku. Od tej chwili `isset($_SESSION['uzytkownik'])` zwróci **false**.

5.5. session_destroy() – zniszczenie całej sesji

```
session_destroy();
```

Co robi: usuwa **wszystkie** dane sesji, czyli kasuje plik sesji na serwerze, i kończy sesję.

UWAGA Dlaczego używamy unset() i session_destroy()?

`session_destroy()` kasuje plik na serwerze, ale tablica `$_SESSION` w aktualnie działającym skrypcie dalej ma stare wartości, aż skrypt się skończy.

Dlatego najpierw `unset()` (czyścimy zmienną od razu), a potem `session_destroy()` (kasujemy całą sesję). Arkusz mówi „zmienna jest usuwana (dane sesji są czyszczone)”, więc użycie obu spełnia wymagania na pewno.

	<code>unset(\$_SESSION['x'])</code>	<code>session_destroy()</code>
Co usuwa?	Jedną wybraną zmienną	Wszystkie dane sesji
Czy sesja dalej istnieje?	Tak	Nie
Działa od razu w tym skrypcie?	Tak	Nie, <code>\$_SESSION</code> zostaje do końca skryptu
Kiedy używać?	Gdy chcesz usunąć jedną rzecz, np. koszyk	Przy wylogowaniu

5.6. header("Location: ...") – przekierowanie

```
header("Location: wyniki.php");
```

Co robi: wysyła przeglądarce nagłówek z poleceniem „przejdź pod ten adres”. Przeglądarka natychmiast otwiera podaną stronę. Tak realizujemy wymagania „zostaje załadowana strona wyniki.php” i „następuje przekierowanie do strony login.php”.

Najważniejsza zasada: `header()` musi być wywołana **zanim** do przeglądarki zostanie wysłany choćby jeden znak strony: `<!DOCTYPE html>`, spacja albo `echo`. Dlatego kod logowania umieszczamy **na samej górze pliku**, przed HTML-em.

Jeśli złamiesz tę zasadę, zobaczysz ostrzeżenie:

```
Warning: Cannot modify header information - headers already sent by ...
```

NA EGZAMINIE Dlaczego „u mnie działało”?

XAMPP ma w pliku `php.ini` ustawienie `output_buffering = 4096`. PHP zbiera wtedy początek strony w buforze i nie wysyła go od razu, więc `header()` w środku HTML-a czasem „przechodzi”.

Na innym serwerze ten sam kod się wysypie. Egzaminator czyta też kod źródłowy. Pisz więc zawsze poprawnie: **logika na górze, HTML pod spodem**.

5.7. `exit()` – zatrzymanie skryptu

```
header("Location: login.php");
exit();
```

Co robi: natychmiast kończy działanie skryptu PHP. Wstawiamy go **zawsze zaraz po** `header("Location: ...")`.

Dlaczego to ważne: `header()` tylko **wysyła polecenie** przekierowania, ale nie zatrzymuje skryptu. Bez `exit()` PHP wykonałby resztę pliku i wygenerował całą stronę z tabelą wyników, nawet dla niezalogowanego użytkownika. Ktoś sprytny mógłby ją podejrzeć, np. programem `curl`.

5.8. `$_POST` – dane z formularza

Formularz w `login.php` ma `method="post"`. Po kliknięciu Zaloguj dane trafiają do tablicy `$_POST`. Kluczem jest **atrybut name** pola formularza:

```
<input type="text" name="login" id="login">      → $_POST['login']
<input type="password" name="haslo" id="haslo">  → $_POST['haslo']
```

Atrybut `id` służy do połączenia z etykietą `<label for="login">`. Atrybut `name` służy do odczytu w PHP. To dwie różne rzeczy, choć często mają tę samą wartość.

6. Konkatenacja – łączenie tekstów w PHP

Konkatenacja to sklejanie kilku napisów w jeden. W PHP robi się to **kropką** `.` (w JavaScriptcie jest to plus `+`).

```
$imie = "Jan";  
echo "Witaj, " . $imie . "!";    // wypisze: Witaj, Jan!
```

Czytaj to tak: weź tekst „Witaj, ”, **doklej** do niego zawartość zmiennej `$imie`, a potem **doklej** wykrzyknik.

Przykład z arkusza – powitanie użytkownika

```
echo "Witaj, <strong>" . $_SESSION['uzytkownik'] . "</strong>";
```

Sklejamy trzy kawałki:

Kawałek	Co to jest
"Witaj, "	Tekst plus znacznik otwierający pogrubienie
<code>\$_SESSION['uzytkownik']</code>	Zmienna sesyjna, czyli nazwa zalogowanego użytkownika (admin)
""	Znacznik zamykający pogrubienie

Wynik w przeglądarce: Witaj, **admin**

Przykład z arkusza – wiersz tabeli

```
echo "<td>" . $wiersz[0] . "</td>";  
echo "<td><a href='" . $wiersz[5] . "' target='_blank'>wynik</a></td>";
```

W drugiej linii zwróć uwagę na **cudzysłowy**. Cały tekst PHP jest w podwójnych cudzysłowach `"..."`, więc atrybuty HTML w środku piszemy w **pojedynczych** `'...'`. Gdybyś w środku użył podwójnych, PHP uznałby, że w tym miejscu tekst się kończy, i zgłosiłby błąd.

Konkatenacja a wstawianie zmiennej w klamrach

PHP pozwala też wpisać zmienną bezpośrednio w tekst w podwójnych cudzysłowach. Obie wersje dają ten sam wynik:

```
// sposób 1 – klamry {}  
echo "Witaj, <strong>{$_SESSION['uzytkownik']}</strong>";  
  
// sposób 2 – konkatenacja (ten wybieramy)  
echo "Witaj, <strong>" . $_SESSION['uzytkownik'] . "</strong>";
```

W tym materiale używamy **konkatenacji**, bo od razu widać, gdzie kończy się tekst, a zaczyna zmienna. Nie trzeba też pamiętać, kiedy klamry są potrzebne.

UWAGA Najczęstsze błędy przy konkatencji

Brak spacji w tekście: `"Witaj," . $imie` da „Witaj,Jan”. Spacja musi być **wewnątrz** cudzysłowu: `"Witaj, "`.

Plus zamiast kropki: `"Witaj " + $imie` w PHP nie skleja tekstów (to dodawanie liczb). Zawsze kropka.

Zapomniana kropka: `"<td>" $wiersz[0] "</td>"` to błąd składni. Między każdym kawałkiem musi być kropka.

7. Sesje w arkuszu „Portal laboratorium” – kod linia po linii

Trzy pliki tworzą jeden cykl. **login.php** zapisuje użytkownika do sesji, **wyniki.php** sprawdza, czy on tam jest, a **wyloguj.php** go usuwa. Każdy plik omawiamy linia po linii.

7.1. Skrypt 1 – login.php (logowanie)

NA EGZAMINIE Wymaganie z arkusza

„Jeżeli dane zostały wysłane z formularza i login to **admin**, a hasło to **TajneHaslo321***, ustawiana jest zmienna sesji zawierająca aktualną nazwę użytkownika oraz zostaje załadowana strona wyniki.php.”

login.php

```
<?php
session_start();

if (isset($_POST['login']) && isset($_POST['haslo'])) {
    $login = $_POST['login'];
    $haslo = $_POST['haslo'];

    if ($login == "admin" && $haslo == "TajneHaslo321*") {
        $_SESSION['uzytkownik'] = $login;
        header("Location: wyniki.php");
        exit();
    }
}
?>
<!DOCTYPE html>
<html lang="pl">
... (dalej zwykły HTML: nagłówek, formularz, stopka)
```

Linia kodu	Co robi i dlaczego
<code>session_start();</code>	Uruchamia sesję. Musi być pierwsza, bo za chwilę będziemy zapisywać do <code>\$_SESSION</code> .
<code>if (isset(\$_POST['login']) && isset(\$_POST['haslo']))</code>	Sprawdza, czy formularz został wysłany. Przy pierwszym wejściu na stronę <code>\$_POST</code> jest pusta i cały ten blok się pomija, więc po prostu wyświetla się formularz.
<code>\$login = \$_POST['login']; \$haslo = \$_POST['haslo'];</code>	Przepisuje dane z formularza do zmiennych o czytelnych nazwach. Za to jest kryterium R.5.10 (znaczące nazwy zmiennych).
<code>if (\$login == "admin" && \$haslo == "TajneHaslo321*")</code>	Porównuje login i hasło z danymi z arkusza. <code>&&</code> oznacza „i”: oba warunki muszą być prawdziwe. Gwiazdka <code>*</code> jest częścią hasła.
<code>\$_SESSION['uzytkownik'] = \$login;</code>	NAJWAŻNIEJSZA LINIA. Zapisuje nazwę użytkownika do sesji. Od teraz serwer pamięta, kto jest zalogowany.
<code>header("Location: wyniki.php");</code>	Każe przeglądarce przejść na stronę z wynikami.
<code>exit();</code>	Kończy skrypt. Reszta strony (formularz) nie jest już potrzebna.

Linia kodu	Co robi i dlaczego
<pre>?> <!DOCTYPE html></pre>	Dopiero tutaj zaczyna się HTML. Cała logika jest nad nim, więc header() działa bez błędów.

Co się dzieje przy **złym haśle**? Warunek jest fałszywy, więc nie zapisujemy nic do sesji i nie przekierowujemy. Skrypt idzie dalej i po prostu ponownie wyświetla formularz logowania. Arkusz nie wymaga komunikatu o błędzie.

7.2. Skrypt 2 – wyniki.php (ochrona strony i powitanie)

NA EGZAMINIE Wymaganie z arkusza

„Jeżeli zmienna sesji zawierająca aktualną nazwę użytkownika nie jest ustawiona, następuje przekierowanie do strony login.php.”

„W bloku nagłówkowym w akapicie wyświetlany jest napis „Witaj, <użytkownik>” ... Nazwa użytkownika jest zapisana w znaczniku semantycznym oznaczającym tekst ważny.” → czyli ****.

wyniki.php (fragment z sesją)

```
<?php
session_start();

if (!isset($_SESSION['uzytkownik'])) {
    header("Location: login.php");
    exit();
}
?>
<!DOCTYPE html>
<html lang="pl">
<head> ... </head>
<body>
    <header>
        <h1>Portal laboratorium</h1>
        <p><?php echo "Witaj, <strong>" . $_SESSION['uzytkownik'] . "</strong>"; ?></p>
        <p><a href="wyloguj.php">Wyloguj</a></p>
    </header>
    ...
```

Linia kodu	Co robi i dlaczego
<code>session_start();</code>	Wznawia sesję. Przeglądarka przysłała PHPSESSID, więc PHP wczytuje zapisane wcześniej <code>\$_SESSION['uzytkownik']</code> .
<code>if (!isset(\$_SESSION['uzytkownik']))</code>	Czytamy: „jeśli NIE ma zalogowanego użytkownika”. To jest „bramkarz” strony.
<code>header("Location: login.php");</code> <code>exit();</code>	Niezalogowany wraca do logowania. Dzięki <code>exit()</code> tabela z danymi pacjentów w ogóle się nie wygeneruje.
<code>echo "Witaj, "</code> <code>. \$_SESSION['uzytkownik']</code> <code>. "";</code>	Wypisuje powitanie. Nazwę bierzemy z sesji (a nie wpisujemy na sztywno „admin”), bo tak każe arkusz. <code></code> = tekst ważny, domyślnie pogrubiony.

Linia kodu	Co robi i dlaczego
<code>Wyloguj</code>	Link do skryptu wylogowania, w osobnym akapicie <code><p></code> .

ZAPAMIĘTAJ Sprawdź sam

Wyloguj się i wpisz w pasku adresu ręcznie **localhost/twoj_folder/wyniki.php**. Jeśli zostaniesz przeniesiony na login.php, ochrona działa poprawnie.

7.3. Skrypt 3 – tabela w wyniki.php (dla porządku)

Skrypt 3 nie używa sesji, ale jest w tym samym pliku i też korzysta z konkatenacji, więc warto go znać. Łączy się z bazą, wysyła zapytanie 4 i dla każdego rekordu tworzy wiersz tabeli.

wyniki.php – Skrypt 3

```
$polaczenie = mysqli_connect("localhost", "root", "", "laboratorium");
$zapytanie = "SELECT imie, nazwisko, pesel, Zlecenie.data, Wynik.data, plik
FROM Pacjent JOIN Zlecenie ON Zlecenie.idPacjenta = Pacjent.id
LEFT JOIN Wynik ON Zlecenie.id = Wynik.idZlecenia;";
$wynik = mysqli_query($polaczenie, $zapytanie);

while ($wiersz = mysqli_fetch_row($wynik)) {
    echo "<tr>";
    echo "<td>" . $wiersz[0] . "</td>";
    echo "<td>" . $wiersz[1] . "</td>";
    echo "<td>" . $wiersz[2] . "</td>";
    echo "<td>" . $wiersz[3] . "</td>";
    echo "<td>" . $wiersz[4] . "</td>";
    if (!empty($wiersz[5])) {
        echo "<td><a href='" . $wiersz[5] . "' target='_blank'>wynik</a></td>";
    } else {
        echo "<td></td>";
    }
    echo "</tr>";
}

mysqli_close($polaczenie);
```

- **mysqli_fetch_row()** zwraca wiersz jako tablicę z numerami 0, 1, 2... Używamy jej, bo dwie kolumny nazywają się tak samo (**data**). Po numerach [3] i [4] łatwo je rozróżnić.
- **!empty(\$wiersz[5])** sprawdza, czy nazwa pliku PDF nie jest pusta. Link „wynik” pojawia się tylko wtedy, gdy wynik istnieje.
- **target='_blank'** otwiera PDF w nowej karcie przeglądarki.
- **mysqli_close()** musi być **ostatnią** operacją na bazie (kryterium R.5.2). Łatwo o niej zapomnieć!

7.4. Skrypt 4 – wyloguj.php (wylogowanie)

NA EGZAMINIE Wymaganie z arkusza

„Jeżeli jest ustawiona zmienna sesji zawierająca nazwę użytkownika, to jest ona usuwana (dane sesji są czyszczone). Następuje przekierowanie do strony login.php.”

wyloguj.php

```
<?php
session_start();

if (isset($_SESSION['uzytkownik'])) {
    unset($_SESSION['uzytkownik']);
    session_destroy();
}

header("Location: login.php");
exit();
?>
```

Linia kodu	Co robi i dlaczego
<code>session_start();</code>	Też potrzebne! Bez tego PHP nie wie, którą sesję ma zniszczyć.
<code>if (isset(\$_SESSION['uzytkownik']))</code>	Arkusz mówi „jeżeli jest ustawiona”, więc sprawdzamy to warunkiem.
<code>unset(\$_SESSION['uzytkownik']);</code>	Usuwa zmienną z nazwą użytkownika.
<code>session_destroy();</code>	Czyści wszystkie dane sesji na serwerze.
<code>header("Location: login.php");</code> <code>exit();</code>	Przekierowanie stoi POZA if-em, więc działa ZAWSZE, nawet gdy ktoś wejdzie na wyloguj.php bez logowania. Arkusz wymaga przekierowania bez żadnego warunku.

UWAGA Częsty błąd

Wpisanie `header("Location: login.php")` **do środka** if-a. Wtedy niezalogowany użytkownik, który wejdzie na wyloguj.php, zostaje na pustej, białej stronie.

8. Najczęstsze błędy i jak ich uniknąć

Błąd	Co się dzieje	Jak naprawić
Brak <code>session_start()</code> w jednym z plików	Mimo poprawnego hasła strona ciągle odsyła do logowania	Wpisz <code>session_start()</code> ; jako pierwszą linię każdego pliku
Spacja lub pusta linia przed <code><?php</code>	Ostrzeżenie „headers already sent”	Plik musi zaczynać się dokładnie od <code><?php</code>
<code>header()</code> w środku HTML-a	Przekierowanie nie działa (albo działa tylko w XAMPP)	Logikę przenieś na górę, przed <code><!DOCTYPE html></code>
Brak <code>exit()</code> po <code>header()</code>	Skrypt działa dalej i może pokazać chronione dane	Zawsze <code>header(...); exit();</code>
Różne klucze w różnych plikach	Ciągle odsyłanie do logowania	Jeden klucz wszędzie, np. <code>'uzytkownik'</code>
Literówka w hasle	Nie da się zalogować	Skopiuj hasło z arkusza: TajneHaslo321* (z gwiazdką!)
Przekierowanie w wyloguj.php tylko w if-ie	Biała strona dla niezalogowanego	<code>header()</code> poza warunkiem
<code>=</code> zamiast <code>==</code> w warunku	<code>if (\$login = "admin")</code> zawsze jest prawdą, więc logowanie bez hasła	Porównanie to <code>==</code> , przypisanie to <code>=</code>
Wpisanie na sztywno „admin” w powitaniu	Działa, ale nie spełnia wymagania „pobrana ze zmiennej sesji”	<code>echo "Witaj, " . \$_SESSION['uzytkownik'] . "";</code>

9. Jak sprawdzić, czy wszystko działa

Zanim oddasz pracę, przejdź ten test. Zajmie Ci dwie minuty, a może uratować kilka punktów.

Nr	Co robisz	Co powinno się stać
1	Wpisujesz w przeglądarce adres wyniki.php, nie będąc zalogowanym	Przeniesienie na login.php
2	Logujesz się złym hasłem	Zostajesz na stronie logowania
3	Logujesz się: admin / TajneHaslo321*	Przeniesienie na wyniki.php, napis „Witaj, admin ”, tabela z 12 wierszami
4	Odświeżasz wyniki.php (F5)	Dalej jesteś zalogowany, sesja działa
5	Klikasz link „wynik” przy pacjencie	PDF otwiera się w nowej karcie
6	Klikasz „Wyloguj”	Przeniesienie na login.php
7	Ponownie wpisujesz adres wyniki.php	Znowu przeniesienie na login.php, czyli wylogowanie zadziałało

10. Sprawdź się – pytania kontrolne

Spróbuj odpowiedzieć samodzielnie, a dopiero potem sprawdź odpowiedzi na następnej stronie.

1. Dlaczego bez sesji serwer nie wiedziałby, że się zalogowałeś?
2. Gdzie są przechowywane dane zapisane w `$_SESSION`?
3. Co robi `session_start()` przy pierwszej wizycie, a co przy kolejnej?
4. Dlaczego kod logowania musi być przed `<!DOCTYPE html>`?
5. Po co jest `exit()` po `header("Location: ...")`?
6. Czym różni się `unset($_SESSION['x'])` od `session_destroy()`?
7. Co oznacza `isset($_SESSION['uzytkownik'])`?
8. Jakim operatorem łączy się teksty w PHP?
9. Czy `session_start()` jest potrzebne w `wyloguj.php`? Dlaczego?
10. Dlaczego logowania nie zapisuje się w zwykłym ciasteczku?

Odpowiedzi

1. Bo HTTP jest bezstanowy: każde żądanie jest dla serwera osobne i serwer nie pamięta poprzednich.
2. Na serwerze, w pliku sesji (w XAMPP w folderze xampp\tmp). Przeglądarka ma tylko identyfikator w ciasteczku PHPSESSID.
3. Przy pierwszej tworzy nową sesję i wysyła ciasteczko. Przy kolejnej wznawia istniejącą sesję i wczytuje jej dane do \$_SESSION.
4. Bo header() wysyła nagłówek HTTP, a nagłówki muszą zostać wysłane przed jakąkolwiek treścią strony.
5. header() tylko wysyła polecenie przekierowania. exit() zatrzymuje skrypt, żeby reszta strony (np. chronione dane) się nie wykonała.
6. unset usuwa jedną zmienną, a sesja trwa dalej. session_destroy usuwa wszystkie dane sesji i kończy ją.
7. „Zmienna uzytkownik NIE istnieje”, czyli użytkownik nie jest zalogowany.
8. Kropką (.), np. "Witaj, " . \$imie
9. Tak. Bez niego PHP nie wie, która sesja należy do użytkownika, więc nie mógłby jej zniszczyć.
10. Bo ciasteczko jest w przeglądarce i użytkownik może je zmienić, np. wpisać sobie nazwę innego użytkownika. Dane sesji są na serwerze i nie da się ich podmienić.

12. Ściąga – sesje w pigułce

Polecenie	Co robi	Gdzie w arkuszu
<code>session_start();</code>	Tworzy lub wznawia sesję	Pierwsza linia każdego pliku
<code>\$_SESSION['uzytkownik'] = \$login;</code>	Zapisuje użytkownika do sesji	login.php, po sprawdzeniu hasła
<code>isset(\$_SESSION['uzytkownik'])</code>	Czy użytkownik jest zalogowany?	wyniki.php, wyloguj.php
<code>unset(\$_SESSION['uzytkownik']);</code>	Usuwa jedną zmienną sesji	wyloguj.php
<code>session_destroy();</code>	Niszczy całą sesję	wyloguj.php
<code>header("Location: plik.php");</code>	Przekierowuje na inną stronę	Wszystkie trzy pliki
<code>exit();</code>	Zatrzymuje skrypt	Zaraz po każdym header()
<code>"tekst " . \$zmienna . " tekst"</code>	Konkatenacja, czyli łączenie tekstów	Powitanie, wiersze tabeli

ZAPAMIĘTAJ Schemat, który zawsze działa

login.php: session_start → sprawdź \$_POST → zapisz \$_SESSION → header → exit → dopiero HTML

strona chroniona: session_start → if (!isset) → header do logowania → exit → HTML

wyloguj.php: session_start → unset + session_destroy → header do logowania (zawsze) → exit