

Metoda GET w PHP — zasady i szablony

Jak korzystać z tej instrukcji

Instrukcja składa się z dwóch części. **Część I** wyjaśnia, jak działa metoda GET — przeczytaj ją raz, uważnie. **Część II** to zestaw dwunastu **szablonów**: gotowych schematów kodu dla wszystkich typowych sytuacji, w których strona przekazuje lub odbiera parameter.

Każdy szablon ma ten sam układ:

1. **Kiedy używać** — w jakiej sytuacji szablon się przydaje.
2. **Szablon** — kod z nazwami ogólnymi, które trzeba podmienić.
3. **Co podmienić** — tabela z listą nazw do zmiany.
4. **Jak to działa** — objaśnienie najważniejszych linii.
5. **Przykład** — ten sam szablon wypełniony dla przykładowej bazy sklep.

Baza do ćwiczeń. Przykłady korzystają z małej bazy sklep z dwiema tabelami: kategorie (id_kategorii, nazwa) oraz produkty (id_produktu, nazwa, cena, id_kategorii, opis, zdjecie). Plik sklep.sql i komplet stron z przykładami znajdziesz w archiwum GET_szablony_przyklady.zip. Treść bazy jest też w załączniku na końcu instrukcji.

Część I. Jak działa metoda GET

1. Parametr w adresie strony

Metoda GET przekazuje dane **w adresie strony**, po znaku zapytania. Dzięki temu ten sam plik PHP może pokazać różną treść — w zależności od tego, co jest w adresie.

Budowa adresu z parametrami GET

Przykład 1 — jeden parametr (strona szczegółów produktu)



Przykład 2 — kilka parametrów (wyszukiwarka)



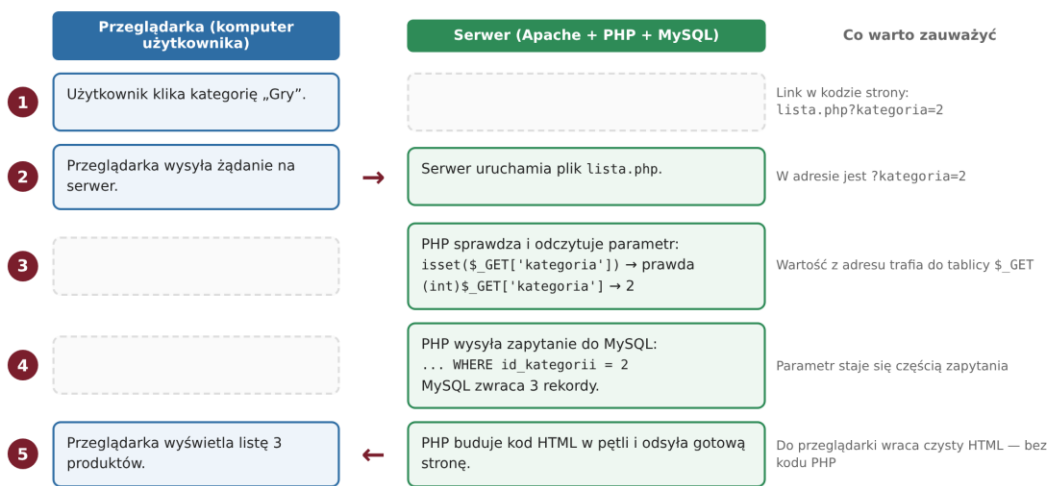
Rysunek 1. Budowa adresu z parametrami GET

Zasady budowy adresu:

- po nazwie pliku stoi znak ? — od tego miejsca zaczynają się dane,
- każdy parametr to para **nazwa=wartość**, np. `id=5`,
- kolejne parametry łączy znak &, np. `kategoria=2&cena=100`,
- parametr może mieć pustą wartość, np. `szukaj=` — liczy się wtedy samo to, że istnieje,
- w adresie nie ma spacji wokół ?, = ani &.

2. Co się dzieje po kliknięciu linku

Co się dzieje po kliknięciu linku z parametrem GET



Rysunek 2. Przebieg żądania z parametrem GET

Z rysunku wynikają trzy zasady, które warto zapamiętać:

1. **Parametr wysyła przeglądarka, a odczytuje serwer.** Kod PHP nie wie, że ktoś kliknął link — widzi tylko adres, który do niego dotarł.
2. **Ten sam plik — różna treść.** `lista.php?kategoria=1` i `lista.php?kategoria=2` uruchamiają ten sam skrypt, ale zapytanie SQL dostaje inną wartość, więc zwraca inne rekordy.
3. **Do przeglądarki wraca czysty HTML.** Kod PHP wykonuje się na serwerze; w „Pokaż źródło strony” zobaczysz tylko jego wynik.

3. Tablica `$_GET`

W PHP wszystkie parametry z adresu trafiają do tablicy `$_GET`. Odczytujesz je po nazwie parametru:

Adres	Co jest w <code>\$_GET</code>
<code>produkt.php?id=5</code>	<code>\$_GET['id'] = "5"</code>
<code>szukaj.php?kategoria=2&cena=100&szukaj=</code>	<code>\$_GET['kategoria'] = "2"</code> , <code>\$_GET['cena'] = "100"</code> , <code>\$_GET['szukaj'] = ""</code>
<code>produkt.php</code>	tablica jest pusta — <code>\$_GET['id']</code> nie istnieje

Zapamiętaj. Wartości w `$_GET` są zawsze **tekstem**, nawet gdy wyglądają jak liczby. Nazwa w nawiasie musi być **identyczna** jak w adresie — z tą samą wielkością liter.

4. Trzy zasady odbierania parametru

Każdy kod, który odbiera parametr, stosuje te same trzy zasady:

Zasada	Zapis	Po co
Najpierw sprawdź, czy parametr istnieje	<code>if (isset(\$_GET['id']))</code>	Strona może być otwarta bez parametru; bez sprawdzenia pojawi się ostrzeżenie „Undefined array key”
Liczbę zamień na liczbę	<code>\$id = (int)\$_GET['id'];</code>	Do zapytania trafi zawsze liczba, nawet gdy ktoś wpisze w adres tekst — tekst bez cyfr daje 0
Tekst wstaw bezpiecznie	apostrofy w SQL lub zapytanie przygotowane	Tekst w SQL musi być w apostrofach; wersja z <code>?</code> chroni przed SQL Injection

W zapytaniu SQL **liczbę** wstawiasz bez apostrofów (`WHERE id = $id`), a **tekst** — w apostrofach (`WHERE nazwa = '$nazwa'`).

5. Słowniczek pojęć

Pojęcie	Znaczenie
parametr	para nazwa=wartość dopisana do adresu po znaku <code>?</code>
<code>\$_GET</code>	tablica PHP z parametrami z adresu
<code>isset()</code>	sprawdza, czy zmienna (np. parametr) istnieje
<code>(int)</code>	zamienia wartość na liczbę całkowitą
<code>mysqli_num_rows()</code>	liczba rekordów w wyniku zapytania; 0 oznacza „nic nie znaleziono”
<code>href</code>	adres, do którego prowadzi odnośnik <code><a></code>
<code>method="get"</code>	formularz wysyła dane w adresie
<code>action</code>	plik, do którego formularz wysyła dane
<code>name</code>	nazwa pola formularza — staje się nazwą parametru
<code>value</code>	wartość opcji lub pola — staje się wartością parametru
<code>selected</code>	zaznacza opcję listy rozwijanej
<code>LIKE '%...%'</code>	warunek SQL „zawiera fragment tekstu”
<code>AS</code>	nadaje kolumnie w wyniku inną nazwę, np. gdy dwie tabele mają kolumnę nazwa

6. Szkielet każdej strony

Wszystkie szablony z części II wstawiasz do takiego szkieletu. Połączenie otwierasz **raz, na początku pliku**, a zamykasz **raz, na końcu** — dzięki temu każdy fragment PHP w środku strony może korzystać z `$con`.

```
<?php
// połączenie z bazą - raz, na początku pliku
$con = mysqli_connect("localhost", "root", "", "nazwa_bazy");
```

```

?>
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Tytuł strony</title>
</head>
<body>

    <!-- tutaj HTML strony i fragmenty PHP z szablonów -->

    <?php
    // zamknięcie połączenia - raz, na końcu pliku
    mysqli_close($con);
    ?>
</body>
</html>

```

7. Mapa szablonów

Nr	Szablon	Typowe zadanie
1	Link z parametrem wpisany na stałe	menu, odnośnik do konkretnej strony
2	Lista odnośników z bazy	lista kategorii, lista rekordów z linkami
3	Obraz jako odnośnik z bazy	galeria miniatur prowadzących do szczegółów
4	Strona szczegółów — jeden rekord	strona produktu, filmu, ucznia
5	Lista rekordów filtrowana parametrem	produkty z jednej kategorii
6	Jedna strona, dwa widoki	„wszystkie” albo „wybrana kategoria”
7	Formularz z polem tekstowym	szukanie po fragmencie nazwy
8	Lista rozwijana generowana z bazy	wybór kategorii w formularzu
9	Formularz z kilkoma polami	wyszukiwarka z kilkoma warunkami
10	Formularz, który pamięta wybór	wyszukiwarka po przeładowaniu strony
11	Link z kilkoma parametrami i link powrotny	przejdźcie dalej z zachowaniem wyboru
12	Sortowanie wybierane parametrem	„od najtańszych”, „od najdroższych”

Część II. Szablony

Szablon 1. Link z parametrem wpisany na stałe

Kiedy używać: gdy wartość parametru znasz z góry — np. w menu chcesz mieć odnośnik do konkretnej kategorii albo strony.

Szablon:

```
<a href="plik.php?parametr=wartosc">Treść odnośnika</a>
```

Co podmienić:

W szablonie	Zamień na
plik.php	plik, który ma odebrać parametr
parametr	nazwę parametru, którą odczytasz w <code>\$_GET['... ']</code>
wartosc	wartość parametru
Treść odnośnika	tekst widoczny dla użytkownika

Jak to działa: przeglądarka po kliknięciu przechodzi pod adres z href. Odnośnik **bez** parametru (np. href="plik.php") też jest przydatny — strona, która go odbierze, pokaże wtedy widok domyślny (szablon 6).

Przykład:

```
<a href="index.php?kategoria=2">Gry</a>
<a href="index.php">Wszystkie</a>
```

Szablon 2. Lista odnośników generowana z bazy

Kiedy używać: gdy odnośników ma być tyle, ile rekordów w tabeli — np. lista kategorii albo lista produktów, w której każdy element prowadzi do innej strony.

Szablon:

```
<ul>
    <?php
    $sql = "SELECT id, nazwa FROM tabela ORDER BY nazwa";
    $wynik = mysqli_query($con, $sql);

    while ($w = mysqli_fetch_array($wynik)) {
        echo "<li><a href='plik.php?parametr=" . $w['id'] . "'> .
    $w['nazwa'] . "</a></li>";
    }
    ?>
</ul>
```

Co podmienić:

W szablonie	Zamień na
id, nazwa	kolumnę z identyfikatorem i kolumnę z tekstem odnośnika
tabela	nazwę tabeli
plik.php	plik docelowy
parametr	nazwę parametru
,	inny znacznik, jeśli zadanie tego wymaga (np.)

Jak to działa: pętla while wykonuje się raz dla każdego rekordu. Najtrudniejsza jest linia z echo — składa się z kawałków połączonych kropką:

Fragment	Co trafia do HTML (dla rekordu id = 2, nazwa = Gry)
" <a href='plik.php?parametr="	<a href='plik.php?parametr=

Fragment	Co trafia do HTML (dla rekordu id = 2, nazwa = Gry)
<code>\$w['id']</code>	2
<code>"'>"</code>	'>
<code>\$w['nazwa']</code>	Gry
<code>""</code>	
Razem	Gry

Zapamiętaj. Cały tekst w echo jest w cudzysłowach "...", a wartość atrybutu href — w apostrofach '...'. Znaczniki i stoją **poza** pętlą, a — **w** pętli.

Przykład:

```
$sql = "SELECT id_kategorii, nazwa FROM kategorie ORDER BY nazwa";
$wynik = mysqli_query($con, $sql);

while ($w = mysqli_fetch_array($wynik)) {
    echo "<li><a href='index.php?kategoria=" . $w['id_kategorii'] . "'> " .
    $w['nazwa'] . "</a></li>";
}
```

Wynik: trzy odnośniki — Gry (index.php?kategoria=2), Książki (?kategoria=1), Muzyka (?kategoria=3).

Szablon 3. Obraz jako odnośnik z bazy

Kiedy używać: galeria miniatur, w której kliknięcie obrazu prowadzi do strony szczegółów.

Szablon:

```
<?php
$sql = "SELECT id, nazwa, plik_obrazu FROM tabela";
$wynik = mysqli_query($con, $sql);

while ($w = mysqli_fetch_array($wynik)) {
    echo "<a href='szczegoly.php?id=" . $w['id'] . "'>";
    echo "<img src='folder/" . $w['plik_obrazu'] . "' alt='" . $w['nazwa'] .
    "'>";
    echo "</a>";
}
?>
```

Co podmienić:

W szablonie	Zamień na
plik_obrazu	kolumnę z nazwą pliku obrazu
folder/	folder z obrazami (z ukośnikiem na końcu)
szczegoly.php, id	plik szczegółów i nazwę parametru
alt	tekst alternatywny — zwykle nazwa rekordu

Jak to działa: w bazie zapisana jest tylko **nazwa pliku**, np. atlas.jpg. Skrypt dokleja przed nią nazwę folderu i powstaje ścieżka folder/atlas.jpg. Znacznik stoi **wewnątrz** <a>, więc to obraz jest odnośnikiem. Jeśli zadanie wymaga podpowiedzi po najechaniu myszką, dodaj atrybut title.

Przykład:

```
while ($w = mysqli_fetch_array($wynik)) {
    echo "<a href='produkt.php?id=" . $w['id_produktu'] . "'>";
    echo "<img class='miniatura' src='img/" . $w['zdjecie'] . "' alt='" .
    $w['nazwa'] . "' title='" . $w['nazwa'] . "'>";
    echo "</a>";
}
```

Szablon 4. Strona szczegółów — jeden rekord

Kiedy używać: strona, która pokazuje jeden rekord wskazany w adresie, np. produkt.php?id=5. To druga połowa szablonów 2 i 3 — tamte tworzą linki, ten je odbiera.

Szablon:

```
<?php
if (isset($_GET['id'])) {

    $id = (int)$_GET['id'];

    $sql = "SELECT kolumna1, kolumna2 FROM tabela WHERE id = $id";
    $wynik = mysqli_query($con, $sql);

    if (mysqli_num_rows($wynik) > 0) {
        $w = mysqli_fetch_array($wynik);
        echo "<h2>" . $w['kolumna1'] . "</h2>";
        echo "<p>" . $w['kolumna2'] . "</p>";
    } else {
        echo "<p>Nie ma rekordu o takim numerze.</p>";
    }

} else {
    echo "<p>Nie wybrano rekordu.</p>";
}
?>
```

Co podmienić:

W szablonie	Zamień na
id (w \$_GET['id'])	nazwę parametru z linku
kolumna1, kolumna2	kolumny do wyświetlenia
tabela, id (w WHERE)	tabelę i kolumnę z identyfikatorem
komunikaty	teksty wymagane w zadaniu

Jak to działa:

Linia	Znaczenie
<code>if (isset(\$_GET['id']))</code>	czy w adresie jest parametr id? Jeśli nie — wykonuje się ostatni else
<code>\$id = (int)\$_GET['id'];</code>	odczyt i zamiana na liczbę; ?id=abc da 0
<code>if (mysqli_num_rows(\$wynik) > 0)</code>	czy rekord o takim numerze istnieje?
<code>\$w = mysqli_fetch_array(\$wynik);</code>	pobranie jednego wiersza — bez pętli , bo wynik ma jeden rekord

Przykład z danymi z dwóch tabel. Obie tabele mają kolumnę nazwa, dlatego przed nazwą kolumny stoi nazwa tabeli, a druga kolumna dostaje nową nazwę przez AS:

```
$sql = "SELECT produkty.nazwa, cena, opis, kategorie.id_kategorii,
kategorie.nazwa AS kategoria
FROM produkty JOIN kategorie ON produkty.id_kategorii =
kategorie.id_kategorii
WHERE id_produktu = $id";
$wynik = mysqli_query($con, $sql);

if (mysqli_num_rows($wynik) > 0) {
    $w = mysqli_fetch_array($wynik);
    echo "<h2>" . $w['nazwa'] . "</h2>";
    echo "<p>Kategoria: " . $w['kategoria'] . "</p>";
    echo "<p>Cena: " . $w['cena'] . " zł</p>";
}
```

Dla `produkt.php?id=4` wyświetli: Szachy turniejowe, Kategoria: Gry, Cena: 120.00 zł.

Szablon 5. Lista rekordów filtrowana parametrem

Kiedy używać: gdy parametr nie wskazuje jednego rekordu, tylko **grupę** — np. wszystkie produkty z wybranej kategorii.

Szablon:

```
<?php
if (isset($_GET['parametr'])) {

    $x = (int)$_GET['parametr'];

    $sql = "SELECT id, nazwa FROM tabela WHERE kolumna_filtra = $x ORDER BY
nazwa";
    $wynik = mysqli_query($con, $sql);

    if (mysqli_num_rows($wynik) > 0) {
        echo "<ol>";
        while ($w = mysqli_fetch_array($wynik)) {
            echo "<li>" . $w['nazwa'] . "</li>";
        }
        echo "</ol>";
    }
}
```

```

    } else {
        echo "<p>Brak rekordów.</p>";
    }
}
?>

```

Różnica w porównaniu z szablonem 4: wynik może mieć wiele wierszy, więc jest **pętla** while, a komunikat dotyczy pustej grupy, a nie brakującego rekordu.

Przykład: index.php?kategoria=2 z warunkiem WHERE id_kategorii = \$kategoria zwraca trzy gry: Gra planszowa Miasta, Puzzle 1000 elementów, Szachy turniejowe. Dla ?kategoria=99 — komunikat „Brak produktów w tej kategorii.”.

Szablon 6. Jedna strona, dwa widoki

Kiedy używać: gdy ta sama strona ma pokazać **wszystko**, gdy nie ma parametru, i **część**, gdy parametr jest. Typowy przykład: odnośnik „Wszystkie” (bez parametru) i odnośniki kategorii (z parametrem) prowadzą do tego samego pliku.

Szablon:

```

<?php
if (isset($_GET['parametr'])) {
    $x = (int)$_GET['parametr'];
    $sql = "SELECT id, nazwa FROM tabela WHERE kolumna_filtra = $x ORDER BY nazwa";
    echo "<h2>Wybrana grupa</h2>";
} else {
    $sql = "SELECT id, nazwa FROM tabela ORDER BY nazwa";
    echo "<h2>Wszystkie</h2>";
}

$wynik = mysqli_query($con, $sql);

if (mysqli_num_rows($wynik) > 0) {
    echo "<ul>";
    while ($w = mysqli_fetch_array($wynik)) {
        echo "<li>" . $w['nazwa'] . "</li>";
    }
    echo "</ul>";
} else {
    echo "<p>Brak rekordów.</p>";
}
?>

```

Jak to działa: warunek if wybiera tylko **treść zapytania** (i nagłówek). Wykonanie zapytania i pętla są wspólne dla obu widoków — nie trzeba ich pisać dwa razy.

Przykład: index.php pokazuje nagłówek „Wszystkie produkty” i 8 produktów, index.php?kategoria=2 — nagłówek „Produkty z wybranej kategorii” i 3 gry.

Szablon 7. Formularz z polem tekstowym

Kiedy używać: użytkownik wpisuje tekst, a strona szuka rekordów, które go zawierają.

Szablon — formularz i odbiór w jednym pliku:

```
<form action="plik.php" method="get">
    <input type="text" name="frazza">
    <button type="submit" name="szukaj">Szukaj</button>
</form>

<?php
if (isset($_GET['szukaj'])) {

    $frazza = $_GET['frazza'];

    $sql = "SELECT id, nazwa FROM tabela WHERE nazwa LIKE ?";
    $st = mysqli_prepare($con, $sql);
    $wzorzec = "%" . $frazza . "%";
    mysqli_stmt_bind_param($st, "s", $wzorzec);
    mysqli_stmt_execute($st);
    $wynik = mysqli_stmt_get_result($st);

    while ($w = mysqli_fetch_array($wynik)) {
        echo "<p>" . $w['nazwa'] . "</p>";
    }
}
?>
```

Jak to działa:

Element	Znaczenie
action="plik.php"	formularz wysyła dane do tego samego pliku, w którym jest
name="szukaj" przy przycisku	dzięki temu w adresie pojawia się szukaj= i isset(\$_GET['szukaj']) wie, że formularz został wysłany
LIKE ?	„zawiera fragment”; znak ? to miejsce na wartość
"%" . \$frazza . "%"	% oznacza „dowolne znaki”, więc %gra% pasuje do każdej nazwy zawierającej „gra”
bind_param(\$st, "s", ...)	podpięcie wartości; s — tekst (string), i — liczba całkowita

Tekst wpisany przez użytkownika nie jest liczbą, więc (int) nie pomoże. Zapytanie przygotowane (?) przekazuje go do bazy osobno od treści zapytania — nawet apostrof w tekście niczego nie zepsuje.

Wersja prosta (niebezpieczna) — krótsza, ale podatna na SQL Injection. Zastępuje pięć linii od \$sql do \$wynik:

```
$sql = "SELECT id, nazwa FROM tabela WHERE nazwa LIKE '%$frazza%'";
$wynik = mysqli_query($con, $sql);
```

Przykład: w bazie sklep fraza gra znajduje „Gra planszowa Miasta” i „Kurs programowania” (MySQL nie rozróżnia tu wielkości liter). Frazę xyz — komunikat „Nie znaleziono produktów.”.

Szablon 8. Lista rozwijana generowana z bazy

Kiedy używać: w formularzu użytkownik wybiera wartość z listy, a opcje listy pochodzą z tabeli.

Szablon:

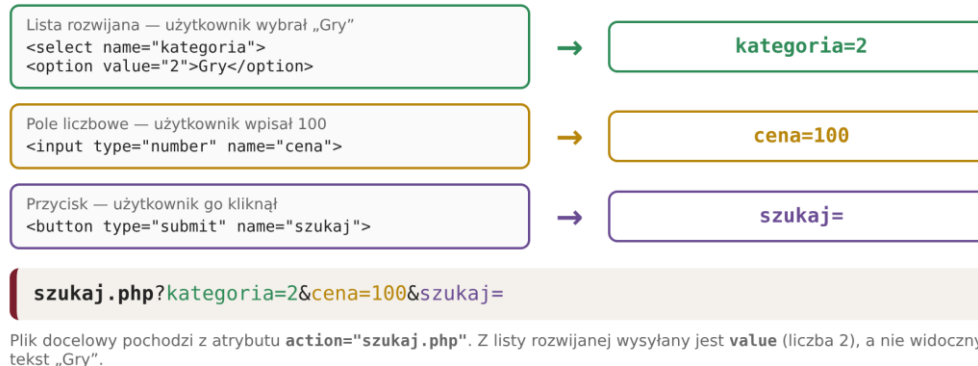
```
<select name="parametr">
  <?php
    $sql = "SELECT id, nazwa FROM tabela ORDER BY nazwa";
    $wynik = mysqli_query($con, $sql);

    while ($w = mysqli_fetch_array($wynik)) {
      echo "<option value='" . $w['id'] . "'>" . $w['nazwa'] . "</option>";
    }
  ?>
</select>
```

Jak to działa: każda opcja ma dwie wartości — value (identyfikator, który trafi do adresu) i treść (nazwa, którą widzi użytkownik). Po wybraniu „Gry” w adresie pojawi się parametr=2, a nie parametr=Gry.

Jak formularz z method="get" buduje adres

Każde pole z atrybutem `name` staje się jednym parametrem. Nazwą parametru jest `name`, a wartością — to, co wybrał lub wpisał użytkownik.



Rysunek 3. Jak formularz z method="get" zamienia pola na parametry adresu

Zapamiętaj. Ten fragment PHP stoi **wewnątrz formularza**, więc połączenie `$con` musi być otwarte wcześniej — dlatego `mysqli_connect()` umieszczamy na samym początku pliku.

Szablon 9. Formularz z kilkoma polami

Kiedy używać: wyszukiwarka z kilkoma warunkami, np. kategoria i maksymalna cena.

Szablon:

```

<form action="plik.php" method="get">
    <select name="parametr1">
        <!-- opcje z szablonu 8 -->
    </select>
    <input type="number" name="parametr2" value="200">
    <button type="submit" name="szukaj">Szukaj</button>
</form>

<?php
if (isset($_GET['szukaj'])) {

    $a = (int)$_GET['parametr1'];
    $b = (int)$_GET['parametr2'];

    $sql = "SELECT id, nazwa, kolumna FROM tabela
            WHERE kolumna_a = $a AND kolumna_b <= $b
            ORDER BY kolumna";
    $wynik = mysqli_query($con, $sql);

    if (mysqli_num_rows($wynik) > 0) {
        echo "<table>";
        echo "<tr><th>Nazwa</th><th>Kolumna</th></tr>";
        while ($w = mysqli_fetch_array($wynik)) {
            echo "<tr><td>" . $w['nazwa'] . "</td><td>" . $w['kolumna'] .
"</td></tr>";
        }
        echo "</table>";
    } else {
        echo "<p>Brak wyników.</p>";
    }
}
?>

```

Jak to działa:

- każde pole z name daje jeden parametr — adres wygląda np. tak: plik.php?parametr1=2¶metr2=200&szukaj=,
- warunki w WHERE łączysz słowem AND (wszystkie muszą być spełnione) — OR oznaczałby „wystarczy jeden”,
- operatory porównania: = równe, <= co najwyżej, >= co najmniej, < mniejsze, > większe,
- wiersz nagłówkowy <tr><th> wypisujesz **przed** pętlą, a wiersze z danymi — **w** pętli.

Przykład: kategoria Gry i cena do 130 zł (szukaj.php?kategoria=2&cena=130&szukaj=) — dwa wyniki: Puzzle 1000 elementów (39.99 zł) i Szachy turniejowe (120.00 zł).

Szablon 10. Formularz, który pamięta wybór

Kiedy używać: po wysłaniu formularza strona ładuje się od nowa i pola wracają do wartości domyślnych. Ten szablon sprawia, że pokazują to, czego szukano.

Szablon — lista rozwijana:

```
while ($w = mysqli_fetch_array($wynik)) {
    if (isset($_GET['parametr']) && $_GET['parametr'] == $w['id']) {
        echo "<option value='" . $w['id'] . "' selected>" . $w['nazwa'] .
"</option>";
    } else {
        echo "<option value='" . $w['id'] . "'>" . $w['nazwa'] . "</option>";
    }
}
```

Szablon — pole liczbowe lub tekstowe:

```
<?php
if (isset($_GET['parametr2'])) {
    $wartoscPola = (int)$_GET['parametr2'];
} else {
    $wartoscPola = 200; // wartość przy pierwszym
otwarcu
}
echo "<input type='number' name='parametr2' value='" . $wartoscPola . "'>";
?>
```

Jak to działa: warunek `isset($_GET['parametr']) && $_GET['parametr'] == $w['id']` jest sprawdzany dla każdej opcji w pętli. Prawdziwy jest tylko dla tej opcji, którą wybrano przy wyszukiwaniu — i tylko ona dostaje `selected`.

Część warunku	Znaczenie
<code>isset(\$_GET['parametr'])</code>	czy w ogóle szukano (czy parametr jest w adresie)?
<code>&&</code>	„i” — obie części muszą być prawdziwe
<code>\$_GET['parametr'] == \$w['id']</code>	czy to ta opcja, którą wybrano? (== porównuje, = przypisuje)

`isset()` stoi **pierwsze**: gdy parametru nie ma, PHP przerywa sprawdzanie i nie dochodzi do `$_GET['parametr']`, które dałoby ostrzeżenie.

Przykład: po wyszukaniu `szukaj.php?kategoria=2&cena=130&szukaj=` lista pokazuje „Gry”, a pole ceny — 130.

Szablon 11. Link z kilkoma parametrami i link powrotny

Kiedy używać: gdy link ma przekazać więcej niż jedną wartość albo gdy strona szczegółów ma prowadzić z powrotem do listy, z której przyszedł użytkownik.

Szablon — link z kilkoma parametrami generowany w pętli:

```
echo "<a href='plik.php?parametr1=" . $w['kolumna1'] . "&parametr2=" .
$w['kolumna2'] . "'>" . $w['nazwa'] . "</a>";
```

Szablon — link powrotny na stronie szczegółów:

```
// identyfikator grupy pobrany z bazy razem z rekordem
echo "<a href='lista.php?parametr=" . $w['id_grupy'] . "'>Powrót do
grupy</a>";
```

Jak to działa: parametry przekazuje się w obie strony. Strona szczegółów sama nie „pamięta”, skąd przyszedł użytkownik, ale zna z bazy identyfikator grupy rekordu (np. `id_kategorii` produktu) i może zbudować z niego link.

Przykład: na stronie `produkt.php?id=4` (Szachy turniejowe) odnośnik „Więcej z kategorii Gry” prowadzi do `index.php?kategoria=2`.

Szablon 12. Sortowanie wybierane parametrem

Kiedy używać: użytkownik wybiera kolejność wyświetlania, np. „od najtańszych”.

Szablon:

```
<a href="plik.php?sort=a">sortuj A</a>
<a href="plik.php?sort=b">sortuj B</a>

<?php
$kolejnosc = "kolumna_domyslna";           // wartość domyślna

if (isset($_GET['sort'])) {
    if ($_GET['sort'] == "a") {
        $kolejnosc = "kolumna_a";
    } elseif ($_GET['sort'] == "b") {
        $kolejnosc = "kolumna_b DESC";
    }
}

$sql = "SELECT nazwa, kolumna_a FROM tabela ORDER BY $kolejnosc";
$wynik = mysqli_query($con, $sql);
?>
```

Jak to działa: nazwy kolumny nie da się zabezpieczyć ani przez `(int)`, ani przez `?`. Dlatego wartość parametru **nigdy nie trafia do zapytania** — służy tylko do wyboru jednej z przygotowanych wcześniej, dozwolonych wartości. Każda inna wartość w adresie (np. `?sort=DROP`) daje sortowanie domyślne. `DESC` oznacza kolejność malejącą.

Przykład: `sortuj.php?sort=tanie` ustawia `ORDER BY cena` (od Płyty z muzyką filmową za 35.00 zł do Gry planszowej Miasta za 149.00 zł), a `?sort=drogie` — `ORDER BY cena DESC`.

Część III. Podsumowanie

8. Jak dobrać szablon do zadania

Treść zadania mówi...	Użyj szablonu
„odnośnik prowadzący do pliku ... z parametrem ...” wygenerowany dla każdego rekordu	2 (lub 3, gdy odnośnikiem jest obraz)
„sprawdza, czy przekazano parametr ...”, „wyświetla dane rekordu”	4
„wyświetla rekordy z wybranej kategorii / grupy”	5
„jeśli parametru nie przekazano, wyświetla wszystkie ...”	6

Treść zadania mówi...	Użyj szablonu
„formularz wysyłający dane metodą GET”, „po kliknięciu przycisku”	7 lub 9
„opcje listy generowane z bazy”	8
„po wyszukiwaniu lista pokazuje wybraną wartość”	10
„odnośnik powrotny do ...”	11
„sortowanie według ...” wybierane przez użytkownika	12

9. Bezpieczeństwo — podsumowanie

Rodzaj wartości w parametrze	Jak zabezpieczyć	Szablon
Liczba (id, cena, rok)	<code>(int)\$_GET['...']</code>	4, 5, 6, 9
Tekst (frazę, nazwisko)	zapytanie przygotowane z ?	7
Nazwa kolumny, kierunek sortowania	wybór z listy dozwolonych wartości	12

Prosta reguła wyboru metody: **GET — gdy coś pokazujesz, POST — gdy coś zmieniasz** (dodajesz, usuwasz, edytujesz rekord) albo wysyłasz poufne dane, np. hasło. Adres z parametrami GET trafia do historii przeglądarki i można go wysłać innej osobie.

10. Najczęstsze błędy

Błąd	Objaw	Poprawka
brak <code>isset()</code>	ostrzeżenie „Undefined array key” po otwarciu strony bez parametru	otocz kod warunkiem <code>if (isset(\$_GET['...']))</code>
spacja w adresie: <code>plik.php? id=5</code>	parametr nie dociera	adres bez spacji
różne nazwy w linku i w <code>\$_GET</code>	parametr „nie istnieje”	ta sama nazwa i wielkość liter
formularz <code>method="post"</code> , odczyt z <code>\$_GET</code>	formularz „nic nie robi”	metoda i tablica muszą się zgadzać
przycisk bez <code>name</code>	wyniki nigdy się nie pojawiają	<code><button type="submit" name="szukaj"></code>
tekst w SQL bez apostrofów	błąd SQL	<code>'\$tekst'</code> albo zapytanie z ?
pomyłone cudzysłowy w <code>echo</code>	błąd składni lub urwany link	zewnątrz <code>"..."</code> , w atrybucie <code>'...'</code>
<code></code> lub <code><table></code> w pętli	każdy rekord w osobnej liście	znacznik otwierający przed pętlą
pętla przy jednym rekordzie albo jej brak przy wielu	wyświetla się tylko pierwszy rekord	jeden rekord — bez pętli, wiele — <code>while</code>
<code>=</code> zamiast <code>==</code> w warunku	warunek zawsze prawdziwy	do porównania <code>==</code>
brak folderu przed nazwą pliku obrazu	ikona uszkodzonego obrazu	<code>src='img/" . \$w['plik'] . "'</code>
kolumna o tej samej	błąd „ambiguous” lub zła	<code>tabela.kolumna</code> , a w razie

Błąd	Objaw	Poprawka
nazwie w dwóch tabelach	wartość	potrzeby AS

11. Jak szukać błędów

Idź tą samą drogą, którą płyną dane:

1. **Najedź myszką na link** — na dole przeglądarki zobaczysz adres. Czy jest w nim ? i właściwy parametr?
2. **Sprawdź pasek adresu po kliknięciu.** Czy parametr dotarł i ma właściwą nazwę?
3. **Wypisz \$_GET:** dopisz na chwilę `print_r($_GET);` — zobaczysz wszystkie odebrane parametry.
4. **Wypisz zapytanie:** dopisz `echo $sql;` przed `mysqli_query()` i uruchom wypisane zapytanie w phpMyAdmin.
5. **Sprawdź liczbę wyników:** `echo mysqli_num_rows($wynik);` — 0 oznacza problem w warunku WHERE, a nie w wyświetlaniu.

Po znalezieniu błędu usuń linie kontrolne.

12. Ściąga

```
// WYSŁANIE - link z bazy
echo "<a href='plik.php?parametr=" . $w['id'] . "'>" . $w['nazwa'] . "</a>";

// ODBIÓR - jeden rekord
if (isset($_GET['parametr'])) {
    $x = (int)$_GET['parametr'];
    $wynik = mysqli_query($con, "SELECT ... WHERE id = $x");
    if (mysqli_num_rows($wynik) > 0) {
        $w = mysqli_fetch_array($wynik); // bez pętli
    } else { echo "<p>Brak rekordu.</p>"; }
} else { echo "<p>Nie wybrano.</p>"; }

// ODBIÓR - wiele rekordów
while ($w = mysqli_fetch_array($wynik)) { /* wiersz */ }

// FORMULARZ - wyniki po kliknięciu
if (isset($_GET['szukaj'])) { /* ... */ }

// ZAZNACZENIE wybranej opcji
if (isset($_GET['p']) && $_GET['p'] == $w['id']) { /* selected */ }
```

Załącznik. Baza danych sklep

Zapisz poniższy kod jako `sklep.sql` (kodowanie UTF-8) i zaimportuj w phpMyAdmin. Komplet stron z przykładami (`index.php`, `produkt.php`, `galeria.php`, `fraza.php`, `szukaj.php`, `sortuj.php`) jest w archiwum `GET_szablony_przyklady.zip`.

```
-- Baza danych "sklep" - przykłady do instrukcji „Metoda GET w PHP -
szablony”
```

-- Import: phpMyAdmin → Import → wybierz ten plik → Wykonaj

```
DROP DATABASE IF EXISTS sklep;
CREATE DATABASE sklep CHARACTER SET utf8mb4 COLLATE utf8mb4_polish_ci;
USE sklep;

CREATE TABLE kategorie (
    id_kategorii INT PRIMARY KEY AUTO_INCREMENT,
    nazwa        VARCHAR(50) NOT NULL
);

CREATE TABLE produkty (
    id_produktu  INT PRIMARY KEY AUTO_INCREMENT,
    nazwa        VARCHAR(100) NOT NULL,
    cena         DECIMAL(8,2) NOT NULL,
    id_kategorii INT NOT NULL,
    opis         VARCHAR(255) NOT NULL,
    zdjecie      VARCHAR(50) NOT NULL
);

INSERT INTO kategorie (nazwa) VALUES
('Książki'),
('Gry'),
('Muzyka');

INSERT INTO produkty (nazwa, cena, id_kategorii, opis, zdjecie) VALUES
('Atlas świata', 89.00, 1, 'Mapy wszystkich kontynentów w dużym formacie.', 'atlas.jpg'),
('Kurs programowania', 59.90, 1, 'Podręcznik dla początkujących programistów.', 'kurs.jpg'),
('Słownik angielski', 45.00, 1, 'Ponad 50 tysięcy haseł z wymową.', 'slownik.jpg'),
('Szachy turniejowe', 120.00, 2, 'Drewniana szachownica z kompletem figur.', 'szachy.jpg'),
('Puzzle 1000 elementów', 39.99, 2, 'Panorama górskiego jeziora.', 'puzzle.jpg'),
('Gra planszowa Miasta', 149.00, 2, 'Strategiczna gra dla 2-4 graczy.', 'miasta.jpg'),
('Płyta z muzyką filmową', 35.00, 3, 'Najpopularniejsze motywy z filmów.', 'plyta.jpg'),
('Koncert na żywo', 49.00, 3, 'Nagranie koncertu w wysokiej jakości.', 'koncert.jpg');
```