

OBSŁUGA ŻĄDAŃ HTTP

GET – POST – FORMULARZE – REDIRECT
UPLOAD PLIKÓW – COOKIES

Interakcja z użytkownikiem

2

- Podstawową motywacją dla aplikacji po stronie serwera jest konieczność interakcji z użytkownikiem
 - ▣ Reagowanie na akcje użytkownika
 - ▣ Obsługa danych podawanych przez użytkownika
 - ▣ Spersonalizowane treści, uzależnione od wyborów użytkownika
- Protokół HTTP dostarcza mechanizmy wymiany danych między przeglądarką i serwerem
- PHP oferuje dostęp do danych w żądaniach HTTP i możliwość sterowania odpowiedziami
 - API do obsługi żądań HTTP

Rodzaje zapytań HTTP

3

- Przeglądarki wykorzystują głównie dwa rodzaje zapytań HTTP:
 - ▣ GET – pobieranie informacji, bez skutków ubocznych
 - ▣ POST – wysyłanie informacji na serwer
- Pozostałe metody są rzadziej wykorzystywane
 - ▣ OPTIONS – negocjacje CORS (Cross-Origin Resource Sharing)
 - ▣ PUT, DELETE, HEAD – tylko poprzez XMLHttpRequest
 - ▣ TRACE – rzadko obsługiwane przez serwery
 - 405 Method Not Allowed
 - 501 Not Implemented

GET i POST

4

□ GET

- ▣ Nawigacja pomiędzy stronami za pomocą odnośników: ``
- ▣ Pobieranie zasobów, m.in.:
 - ▣ ``
 - ▣ `<link href="main.css" rel="stylesheet" .../>`
 - ▣ `<script src="main.js">`
- ▣ Obsługa formularzy: `<form method="get">`
- ▣ Redirect HTTP 303 *See other*

□ POST

- ▣ Obsługa formularzy: `<form method="post">`
- ▣ Redirect HTTP 307 *Temporary Redirect*
 - ▣ Jeśli oryginalne zapytanie zostało wysłane metodą POST

GET – dodatkowe parametry

5

- Parametry – dołączone do identyfikatora URI – mają doprecyzować zapytanie użytkownika

Parametry występują
po znaku zapytania

Parametry postaci:
nazwa=wartość

`http://example.com/search.php?query=kot`

Kolejne parametry
rozdzielone znakiem &

`/products.php?cat=laptops&producent=xyz`

GET – parametry w adresie

6

- Zapisywane w bookmarkach
 - ▣ Odnośniki bezpośrednie (ang. *permalinks*)
- Zapamiętywane w historii przeglądarki
- Zachowywane przy kopiowaniu adresu
- Mogą być zapisane w logach serwera
- Wysyłane przy każdym odświeżeniu strony
- Ograniczony rozmiar parametrów
 - ▣ Max długość adresu URL w przeglądarkach/klientach
 - Standard HTTP nie definiuje ograniczenia
 - Z empirycznych doświadczeń: ok. 2000 znaków

**Nie należy
przekazywać
danych
wrażliwych!**

GET

7

- Zapytania typu GET powinny być *bezpieczne* i co za tym idzie *idempotentne*
 - ▣ Bezpieczne
 - Nie ma żadnych skutków ubocznych po stronie serwera
 - Służy pobieraniu informacji
 - ▣ Idempotentne
 - Skutki uboczne wielu identycznych zapytań są takie same jak skutki pojedynczego zapytania
 - Zapytanie bezpieczne jest siłą rzeczy idempotentne
- Dlaczego?
 - ▣ Kliknięcie w bookmark, odświeżenie strony, wysłanie linku do innej osoby
 - ▣ Cachowanie odpowiedzi

POST

8

- Parametry przekazywane w ciele żądania HTTP

```
POST /wai/example.php HTTP/1.1
```

```
Accept: */*
```

```
Content-Length: 104
```

```
Content-Type: application/x-www-form-urlencoded;  
              charset=utf-8
```

```
Host: 192.168.199.11
```

```
tytuł=Wodospad+Skogafoss&autor=WK&ocena=9&opis=J  
eden+z+najbardziej+malowniczych+wodospad%C3%B3w+  
Islandii
```


POST

9

- ❑ Zapytanie typu POST nie musi być ani *bezpieczne* ani *idempotentne*
 - ❑ Modyfikowanie stanu po stronie serwera
- ❑ Nie-bookmarkowalne
- ❑ Nie-cachowalne
- ❑ Przy odświeżeniu strony przeglądarka powinna ostrzegać o ponownym wysyłaniu danych
- ❑ Parametry nie trafiają do logów serwera WWW
- ❑ Nie ma limitu na rozmiar parametrów

GET i POST – podsumowanie

10

□ GET

- Pobieranie informacji z serwera
 - np. wyświetlenie informacji o produkcie
- Parametry doprecyzowują zapytanie użytkownika
 - np. o który produkt chodzi?

□ POST

- Zmieniają stan po stronie serwera
- Wyzwalają logikę biznesową aplikacji
 - np. złożenie zamówienia i realizacja płatności
- Parametry są argumentami dla logiki biznesowej
 - np. jakie produkty wchodzą w skład zamówienia?

Obsługa formularzy w PHP

...i parametrów żądań w ogólności

Parametry GET

12

products.php?cat=laptops&producent=xyz

```
echo $_GET['cat'];  
echo $_GET['producent'];  
echo $_GET['sku'];
```

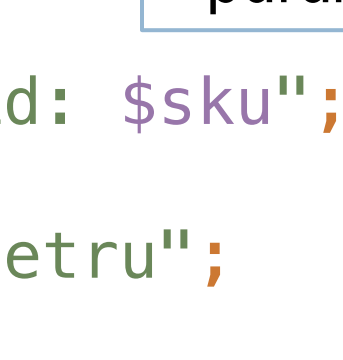


laptops

xyz

Notice: Undefined index: sku in **example.php** on line 9

```
if(!empty($_GET['sku'])) {  
    $sku = $_GET['sku'];  
    echo "Wybrano produkt o id: $sku";  
} else {  
    echo "Nie określono parametru";  
}
```



Test obecności
parametru

Tablica `$_GET`

13

- Daje dostęp do wszystkich parametrów zapytania typu GET
- Zmienna *superglobalna*
 - ▣ Dostępna z każdego miejsca w kodzie, bez używania słowa kluczowego **global**

Parametry POST

14

```
<form method="post">
  <label>
    <span>Tytuł:</span>
    <input type="text" name="tytuł"/>
  </label>

  <label>
    <span>Autor:</span>
    <input type="text" name="autor"/>
  </label>

  <textarea placeholder="Opis..."
    name="opis"></textarea>

  <input type="submit" value="Zapisz"/>
</form>
```

The diagram illustrates the mapping between the HTML code on the left and a visual representation of a web form on the right. Blue arrows connect specific code elements to their corresponding UI components:

- An arrow from the `<input type="text" name="tytuł"/>` code line points to the "Tytuł:" label and its associated text input field.
- An arrow from the `<input type="text" name="autor"/>` code line points to the "Autor:" label and its associated text input field.
- An arrow from the `<textarea placeholder="Opis..." name="opis">` code line points to the "Opis..." text area.
- An arrow from the `<input type="submit" value="Zapisz"/>` code line points to the "Zapisz" submit button.

The form itself is a light blue box containing the labels "Tytuł:", "Autor:", and "Opis...", the corresponding input fields, and a button labeled "Zapisz" at the bottom right.

Parametry POST


15

```
<input type="text" name="tytul"/>
```

```
<?php
$tytul = $_POST['tytul'];
$autor = $_POST['autor'];
$opis = $_POST['opis'];

//zapisanie do bazy danych...
?>
<div>
    <h1><?= $tytul ?></h1>
    <span>Autor: <?= $autor ?></span>

    <p><?= $opis ?></p>
</div>
```



Obsługa formularzy

16

```
<input type="text" name="tytuł"/>
```

1. GET – wyświetlenie formularza

Tytuł:

Autor:

Jeden z najbardziej imponujących wodospadów Islandii.

Zapisz

2. POST – submit

```
<?php  
$tytuł = $_POST['tytuł'];
```

3. Wyświetlenie wyników*

Wodospad Skogafoss

Autor: Waldemar

Jeden z najbardziej imponujących wodospadów Islandii.

* Docelowo z wykorzystaniem idiomu POST-REDIRECT-GET

Jakie żądanie trafiło na serwer?

17

- W wielu sytuacjach istotne jest, czy serwer otrzymał żądanie GET czy POST
- Tablica superglobalna **\$_SERVER** zawiera informacje o kontekście wywołania skryptu PHP

```
if($_SERVER['REQUEST_METHOD'] === 'POST'){  
    //żądanie POST  
}  
else {  
    //najpewniej GET  
}
```

Formularze: pole tekstowe

18

```
<input type="text" name="tytuł"/>
```

The diagram illustrates the flow of data from an HTML form element to a PHP script. At the top, the HTML tag `<input type="text" name="tytuł"/>` is shown. A blue arrow points from the `name="tytuł"` attribute to a form field. The form field is a rectangular box containing the text "Tytuł:" followed by a text input field with the value "Wodospad". Another blue arrow points from the text input field to the PHP code below.

```
<?php  
$tytuł = $_POST['tytuł'];
```

Formularze: pole hasła

19

```
<input type="password" name="pass"/>
```

The diagram illustrates the visual representation of the HTML code above. A light blue arrow points from the `<input type="password" name="pass"/>` tag to a form element. The form element consists of a light blue rectangular container. Inside this container, on the left, is the text "Hasło:". To the right of this text is a white rectangular input field with a thin grey border. Inside the input field, there are ten black dots, representing a masked password. A second light blue arrow points downwards from the bottom center of the input field.

```
<?php  
$password = $_POST['pass'];
```

Formularze: pole radio

20

```
<span>Płeć:</span>
```

```
<label>
```

```
<input type="radio"  
name="plec"  
value="mezczyzna"/>
```

Mężczyzna

```
</label>
```

```
<label>
```

```
<input type="radio"  
name="plec"  
value="kobieta"/>
```

Kobieta

```
</label>
```

Płeć:

☒ Mężczyzna

☐ Kobieta

```
<?php  
$plec = $_POST['plec'];
```

mezczyzna

Formularze: checkboxy

21

```
<span>Kategoria:</span>
<label>
  <input type="checkbox"
    name="kategorie[]"
    value="akcja"/> Akcja
</label>
<label>
  <input type="checkbox"
    name="kategorie[]"
    value="dramat"/> Dramat
</label>
<label>
  <input type="checkbox"
    name="kategorie[]"
    value="komedia"/> Komedia
</label>
```

Kategoria:

☒ Akcja

☐ Dramat

☒ Komedia

```
<?php
$kategorie =
    $_POST['kategorie'];
print_r($kategorie);
```

```
Array
(
    [0] => akcja
    [1] => komedia
)
```

Formularze: checkboxy

22

```
<span>Kategoria:</span>
<label>
  <input type="checkbox"
    name="kategorie[akcja]"
    value="true"/> Akcja
</label>
<label>
  <input type="checkbox"
    name="kategorie[dramat]"
    value="true"/> Dramat
</label>
<label>
  <input type="checkbox"
    name="kategorie[komedia]"
    value="true"/> Komedia
</label>
```

Kategoria:

- ☒ Akcja
- ☐ Dramat
- ☒ Komedia

```
<?php
$kategorie =
    $_POST['kategorie'];
print_r($kategorie);
```

```
Array
(
    [akcja] => true
    [komedia] => true
)
```

Formularze: pole ukryte

23

```
<input type="hidden" name="id"/>
```

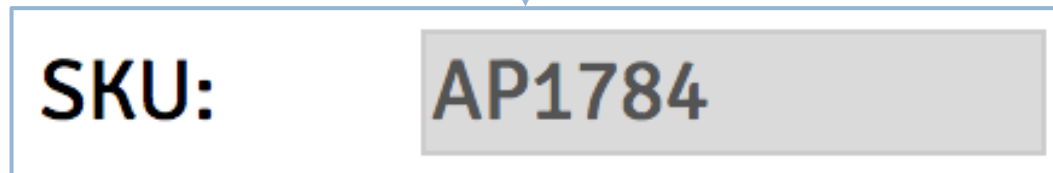
Nie wyświetla się w przeglądarce

```
<?php  
$id = $_POST['id'];
```

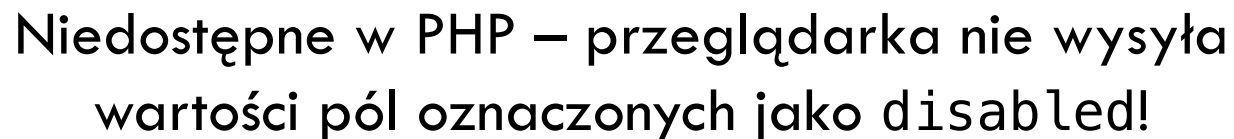
Formularze: pole nieaktywne

24

```
<input type="text" name="tytuł"  
value="AP1784"  
disabled="disabled"/>
```



A diagram showing a blue arrow pointing from the `disabled="disabled"` attribute in the HTML code above to a form field. The form field is a rectangular box with a light gray border. Inside the box, on the left, is the text "SKU:" in a bold, dark gray font. To the right of this text is a gray rectangular input field containing the text "AP1784" in a dark gray font.



A diagram showing a blue arrow pointing from the form field above to a text box. The text box is a rectangular box with a light blue border. Inside the box, the text reads: "Niedostępne w PHP – przeglądarka nie wysyła wartości pól oznaczonych jako disabled!" in a black font.

Niedostępne w PHP – przeglądarka nie wysyła wartości pól oznaczonych jako disabled!

Formularze: upload plików

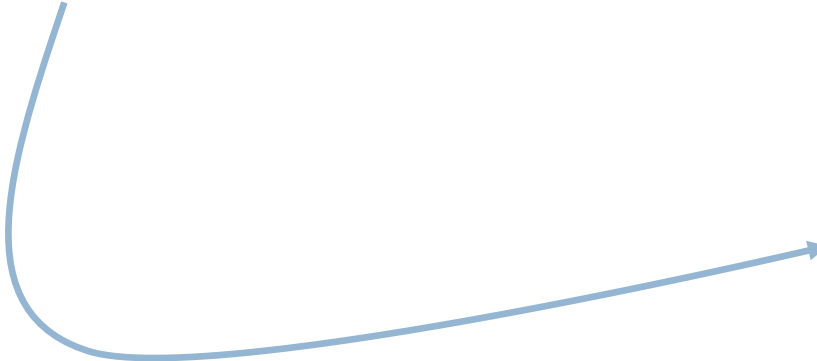

25

```
<form method="post"  
      enctype="multipart/form-data">
```

```
  <input type="file" name="zdjecie"/>
```

```
  <input type="submit" value="Wyślij"/>
```

```
</form>
```



giraffe.jpg

Upload plików

26

- Informacje o przesłanych plikach dostępne są w tablicy superglobalnej **\$_FILES**:

```
<?php
$zdjecie = $_FILES['zdjecie'];
print_r($zdjecie);
```



```
Array (
    [name] => giraffe.jpg
    [type] => image/jpeg
    [tmp_name] => /tmp/php1gzN7z
    [error] => 0
    [size] => 50094
)
```

Informacje o nadesłanym pliku

27

- name – oryginalna nazwa pliku
- type – typ MIME odczytany z nagłówek żądania HTTP (typowo określony na bazie rozszerzenia pliku)
- tmp_name – lokalizacja pliku w katalogu tymczasowym na serwerze
- size – rozmiar pliku
- error – informacja o ewentualnych błędach uploadu

```
print_r($_FILES['zdjecie'])  
Array (  
    [name] => giraffe.jpg  
    [type] => image/jpeg  
    [tmp_name] => /tmp/php1gzN7  
    [error] => 0  
    [size] => 50094  
)
```

Obsługa nadestanego pliku

28

- Pliki umieszczane są początkowo w lokalizacji tymczasowej
- Jeśli pliki mają być trwale zapisane na serwerze, należy je przenieść do lokalizacji docelowej
 - ▣ Uwaga na uprawnienia do katalogu docelowego

Obsługa nadestłanego pliku

29

```
$upload_dir = '/var/www/dev/web/upload/';

$file = $_FILES['zdjecie'];
$file_name = basename($file['name']);

$target = $upload_dir . $file_name;

$tmp_path = $file['tmp_name'];

if(move_uploaded_file($tmp_path, $target)){
    echo "Upload przebiegł pomyślnie!\n";
}
```

Sprawdzanie typu MIME pliku

30

- Nie można polegać na typie MIME odczytanym z nagłówków żądania HTTP
 - ▣ Zmylenie przeglądarki zmianą rozszerzenia pliku
 - ▣ Spreparowanie złośliwego żądania przez napastnika
- Należy sprawdzić sygnaturę pliku (ang. *magic number*):

```
$finfo = finfo_open(FILEINFO_MIME_TYPE);  
$file_name = $_FILES['zdjecie']['tmp_name'];  
$mime_type = finfo_file($finfo, $file_name);  
  
if ($mime_type === 'image/jpeg') {  
    echo 'Poprawny format.';  
}
```

31

Idiom POST-REDIRECT-GET

GET, POST i... nic więcej

32

przelew.php

```
<?php
if($_SERVER['REQUEST_METHOD']==='POST' &&
    !empty($_POST['nr_konta'] /* && ... */
) {
    $nr_konta = $_POST['nr_konta'];
    //...

    //zlecenie przelewu

    echo "Status: przyjęty do realizacji";
} else { ?>
    <form method="post">
        <input type="text" name="nr_konta"/>
        <!-- ... -->
        <input type="submit" value="OK"/>
    </form>
<?php } ?>
```


GET przelew.php

33

```
<?php
if($_SERVER['REQUEST_METHOD']==='POST' &&
    !empty($_POST['nr_konta'])/* && ... */) {
    $nr_konta = $_POST['nr_konta'];
    //...

    //zlecenie przelewu

    echo "Status: przyjęty do realizacji";
} else { ?>
    <form method="post">
        <input type="text" name="nr_konta"/>
        <!-- ... -->
        <input type="submit" value="OK"/>
    </form>
<?php } ?>
```

false

Wykonanie części else:
wyświetlenie formularza

POST przelew.php

34

```
<?php
if($_SERVER['REQUEST_METHOD']==='POST' &&
    !empty($_POST['nr_konta'])/* && ... */
) {
    $nr_konta = $_POST['nr_konta'];
    //...

    //zlecenie przelewu

    echo "Status: przyjęty do realizacji";
} else { ?>
    <form method="post">
        <input type="text" name="nr_konta"/>
        <!-- ... -->
        <input type="submit" value="OK"/>
    </form>
<?php } ?>
```

true

Sprawdzenie
poprawności
danych: true

Wykonanie
logiki
biznesowej

Użytkownik klika Odśwież (F5)

35

- Użytkownik chce sprawdzić, czy status przelewu się zmienił: klika Odśwież
- Przeglądarka ponawia poprzednie zapytanie: **POST przelew.php** z takimi samymi parametrami

```
<?php
if($_SERVER['REQUEST_METHOD']==='POST' &&
    !empty($_POST['nr_konta'])/* && ... */ } true
) {
    $nr_konta = $_POST['nr_konta'];
    //...

    //zlecenie przelewu

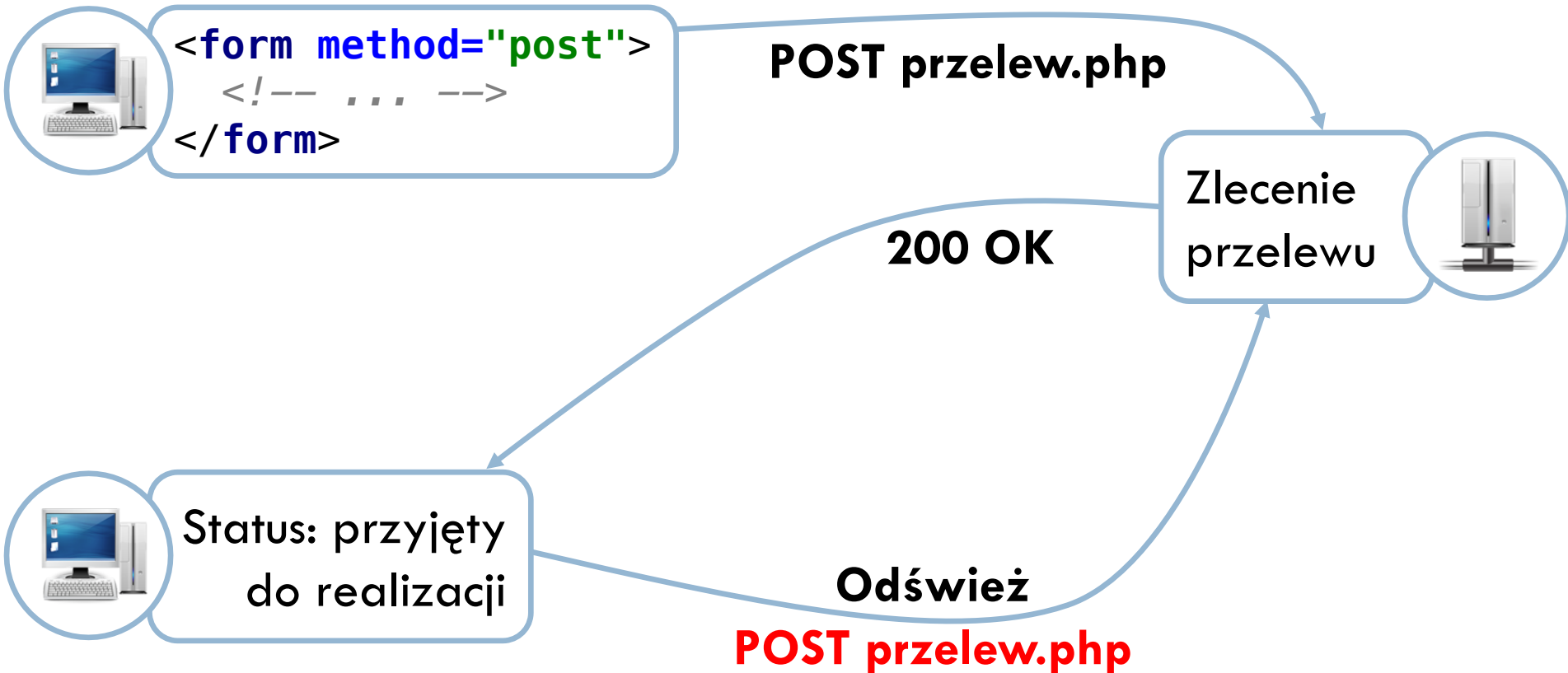
    echo "Status: przyjęty do realizacji";
} else { ?> <!-- ... --> <?php } ?>
```

Wykonanie logiki biznesowej

PO RAZ DRUGI!

Problem wielokrotnego submitu

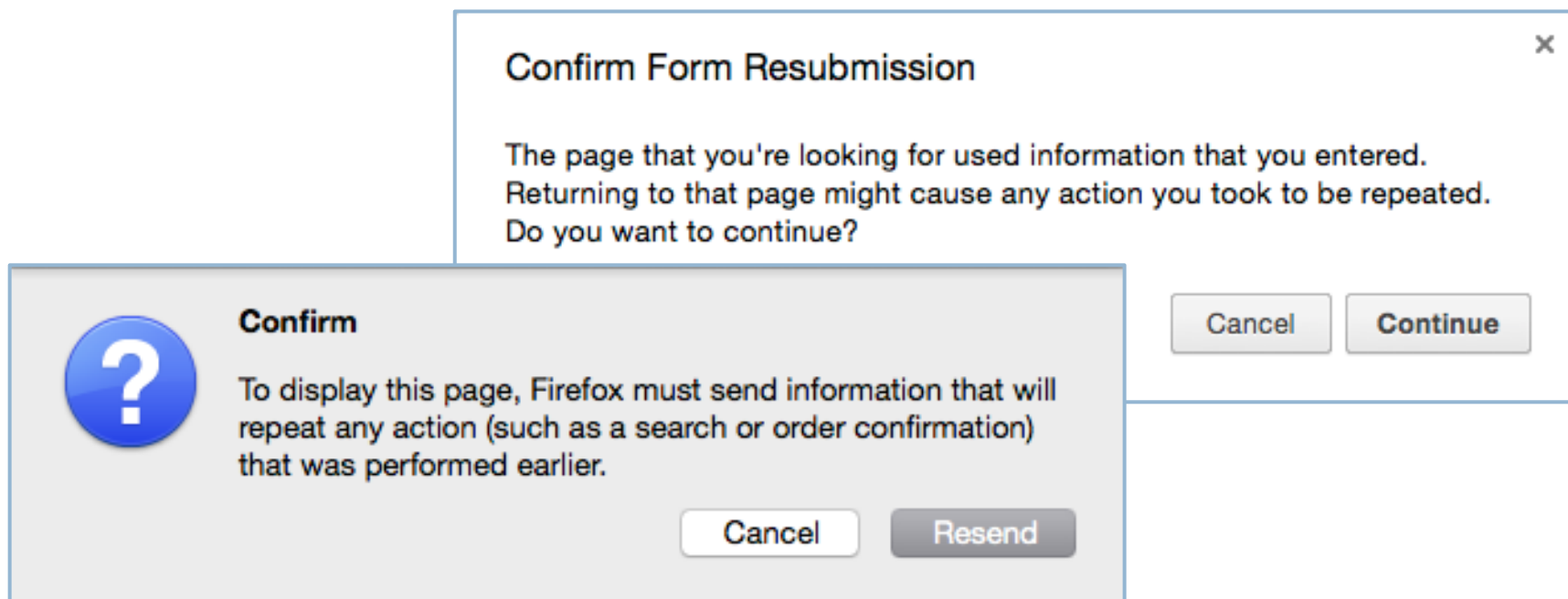
36



POST i przycisk Odśwież

37

- Współczesne przeglądarki ostrzegają przed ponownym wysłaniem zapytania POST
 - ▣ Gdyby tylko użytkownicy czytali komunikaty...
 - ▣ ...i wiedzieli jak je interpretować



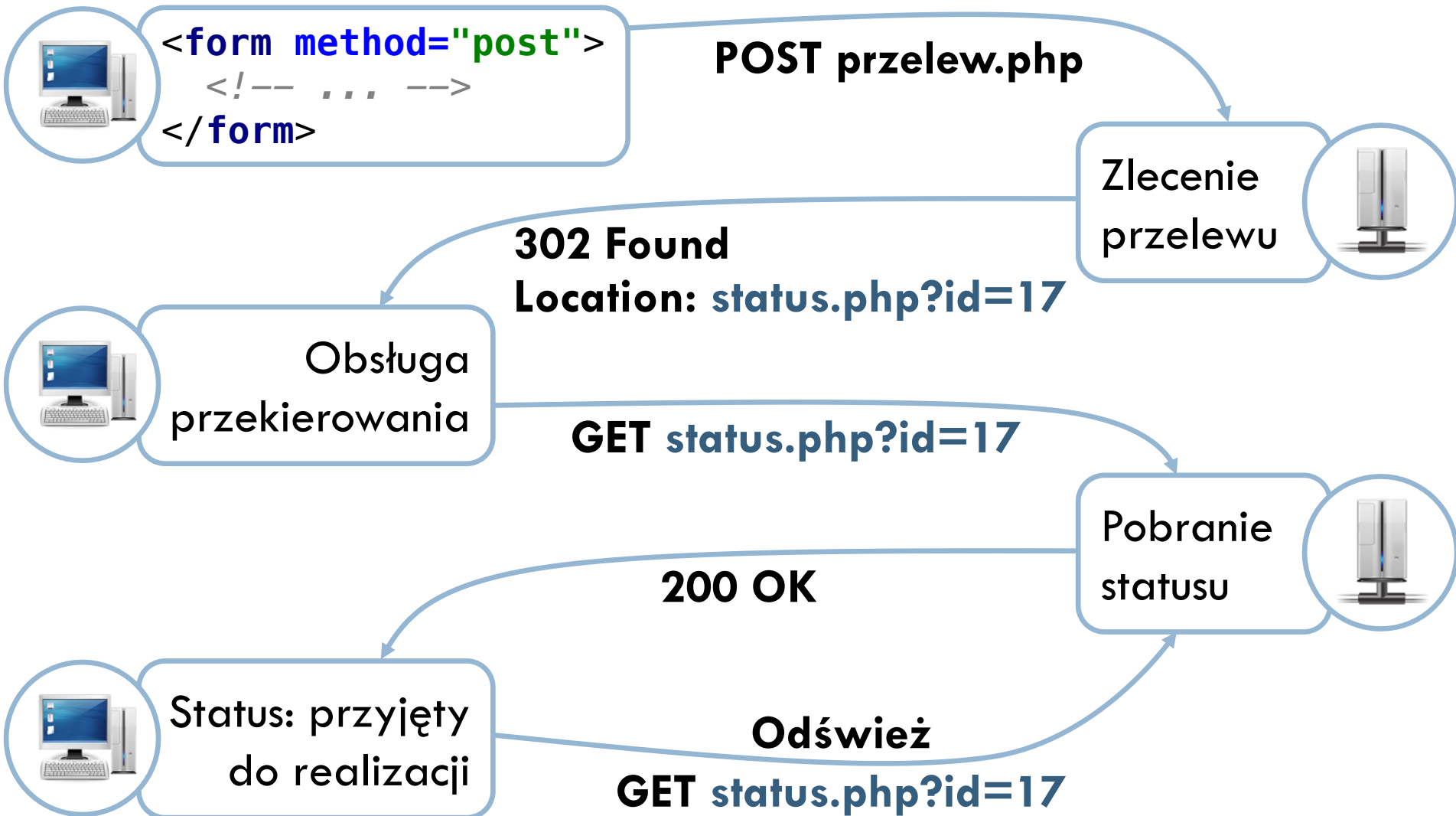
POST i zmiany stanu

38

- Żądanie POST, które zmieniło stan po stronie serwera, nie powinno zwracać informacji dla użytkownika, a jedynie odpowiedź *Redirect* (grupa 3xx) dla przeglądarki
- Przeglądarka zrealizuje przekierowanie pod adres umieszczony w nagłówku **Location**
 - ▣ Docelowa strona powinna prezentować status wykonanej operacji
- Jeśli nie nastąpiła zmiana stanu, przekierowanie nie jest konieczne
 - ▣ np. błędnie wypełniony formularz

Idiom POST-REDIRECT-GET

39



POST-REDIRECT-GET

40

przelew.php

```
<?php
if ( $_SERVER['REQUEST_METHOD'] === 'POST' &&
    !empty($_POST['nr_konta']) /* ... */
) {
    $nr_konta = $_POST['nr_konta']; // ...

    //zlecenie przelewu
    $id = /* ... */;

    $url = "status.php?id=$id";
    header("Location: $url");
    exit;
} ?>
```

Identyfikator zlecenia

Strona statusu

Przekierowanie
HTTP 302

Zakończenie skryptu

```
<form method="post">
    <!-- ... -->
</form>
```


Strona statusu

41

status.php

```
<?php
$id = $_GET['id'];
$przelew = /* pobranie z bazy danych */;
echo "Status: " . $przelew['status'];
```

Przekierowania, nagłówki i ciało odpowiedzi HTTP

42

- Przekierowania (3xx) są realizowane poprzez nagłówki odpowiedzi HTTP
 - ▣ Kod odpowiedzi, np. 302
 - ▣ Nagłówek Location

HTTP/1.1 **302 Found**

Content-Length: 0

Content-Type: text/html; charset=UTF-8

Location: status.php

Server: Apache

...

Przekierowania, nagłówki i ciało odpowiedzi HTTP

43

- Nagłówki i kod odpowiedzi muszą być ustawione przed rozpoczęciem emitowania ciała odpowiedzi (dokumentu HTML)
- Rozpoczęcie emitowania ciała odpowiedzi powoduje wysłanie nagłówków
- Próba zmiany nagłówków po rozpoczęciu emitowania ciała zakończy się niepowodzeniem i komunikatem:
„Cannot modify header information – headers already sent”

Struktura odpowiedzi HTTP

44

```
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Length: 283
Content-Type: text/html; charset=UTF-8
Server: Apache
```

nagłówki

```
<!DOCTYPE html>
<html>
  <head>
    <title>Przelew</title>
  </head>
  <body>
    <form method="post">
      <!-- ... -->
    </form>
  </body>
</html>
```

ciało

Ciało odpowiedzi HTTP

45

- Dowolna treść na standardowym wyjściu skryptu PHP jest traktowana jako ciało odpowiedzi
- Uwaga na białe znaki – one również traktowane są jako ciało odpowiedzi HTTP
 - ▣ Znaki nowego wiersza
 - ▣ Spacje
 - ▣ Znak BOM

Buforowanie odpowiedzi

46

- Ustawienie **output_buffering** w php.ini
- Ustawienie zależne od konfiguracji serwera
 - ▣ Włączenie/wyłączenie buforowania, rozmiar bufora
- Standardowe wyjście jest buforowane przed przekazaniem do serwera WWW
- Pozwala częściowo obejść problem „*headers already sent*”, ale nie stanowi rozwiązania
 - ▣ Jedynym solidnym rozwiązaniem jest prawidłowa struktura skryptów!
- Głównym celem buforowania jest poprawa wydajności komunikacji: interpreter PHP \Leftrightarrow serwer WWW
 - ▣ ...a nie zamiatanie pod dywan błędów programisty

Prawidłowa struktura skryptów PHP

47

- Wszelkie operacje zmieniające nagłówki lub kod odpowiedzi wykonywane przed kodem HTML
- Bloki PHP rozpoczynane w pierwszym wierszu pliku
 - ▣ Bez znaków nowego wiersza przed `<?php`
 - ▣ Bez znaku BOM
- Kod HTML dopiero po wykonaniu całej logiki biznesowej i logiki obsługi żądania
 - ▣ Więcej na wykładzie o przepływie sterowania w aplikacjach internetowych
- W plikach zawierających wyłącznie kod PHP (np. funkcje narzędziowe) pominięty znacznik `?>`

HTTP Cookies



HTTP Cookies

49

- Umożliwiają zapisanie informacji po stronie przeglądarki użytkownika
- Są dołączane do **każdego** zapytania, kierowanego pod adres, z którego przeglądarka je otrzymała
 - ▣ Cookies o dużym rozmiarze znacząco zwiększają rozmiar **każdego** zapytania HTTP!
- Cookies są ustawiane przez nagłówki HTTP – muszą być ustawione przed rozpoczęciem emitowania treści strony

Zapisywanie informacji w cookies

50

```
<?php
```

```
$imie = 'Waldemar';
```

```
//o zasięgu sesji przeglądarki:  
setcookie("imie_session", $imie);
```

↑
nazwa

↑
wartość

```
//z podanym czasem ważności i zakresem:  
setcookie("imie", $imie, time()+3600, "/");
```

↑
nazwa

↑
wartość

↑
czas ważności:
timestamp

↑
zakres:
cała domena

Cookies w odpowiedzi HTTP

51

HTTP/1.1 200 OK

Connection: Keep-Alive

Content-Length: 274

Content-Type: text/html; charset=UTF-8

Server: Apache

Set-Cookie: imie=Waldemar;
 expires=Thu, 24-Sep-2015 20:56:13 GMT;
 path=/

Set-Cookie: imie_session=Waldemar

X-Powered-By: PHP/5.4.45-0+deb7u1

...

Odczytywanie cookies

52

Sprawdzenie
dostępności
cookie

```
if (isset($_COOKIE['imie'])) {  
    $imie = $_COOKIE['imie'];  
}
```

Pobranie
wartości
cookie

Usuwanie cookies

53

- Nadpisanie cookie z czasem ważności w przeszłości
 - ▣ Wszystkie parametry poza wartością cookie muszą być identyczne jak w oryginalnym wywołaniu `setcookie()`

```
setcookie("imie", $imie, time() + 3600, "/");
```

```
setcookie("imie", '', time() - 3600*24, "/");
```



- Podanie **false** jako wartości
 - ▣ Wszystkie parametry poza wartością i czasem ważności muszą być identyczne jak w oryginalnym wywołaniu `setcookie()`

```
setcookie("imie", false);
```



Przykład: 01-cookies

Tablica superglobalna **`$_REQUEST`**

55

- Zawiera wszystkie parametry z tablic **`$_GET`** i **`$_POST`** (oraz opcjonalnie **`$_COOKIE`**)
- Przydatna, gdy nie ma znaczenia w jaki sposób parametr został przesłany
- Konfigurowana w `php.ini`:
 - ▣ **`request_order = "GP"`** – domyślne ustawienie
 - Najpierw GET, potem POST, parametry o tej samej nazwie nadpisują poprzednią wartość
 - ▣ **`request_order = "GCP"`** – uwzględnia również dane z *cookies*

Przeptyw sterowania

56

Przeglądarka



1. Żądanie HTTP

GET, POST, parametry żądania

Serwer HTTP



Apache

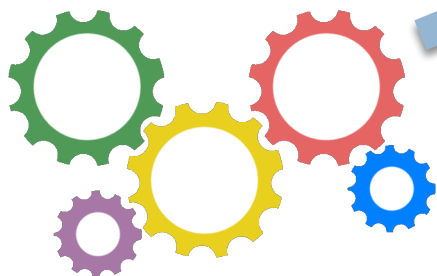
5. Odpowiedź HTTP

4. Zwrócenie dokumentu HTML
pliki cookies, nagłówki odpowiedzi

2. Lokalizacja zasobu

3. Wywołanie skryptu PHP

Parametry, nagłówki żądania,
pliki cookies



```
<!DOCTYPE html>
<html>
<head>
  <title>Strona główna</title>
</head>
<body>
  <h1>Witaj świecie!</h1>
  <p>
    Wersja PHP:
    <?php
      echo phpversion();
    ?>
  </p>
</body>
</html>
```

Skrypt PHP

57

Pytania?