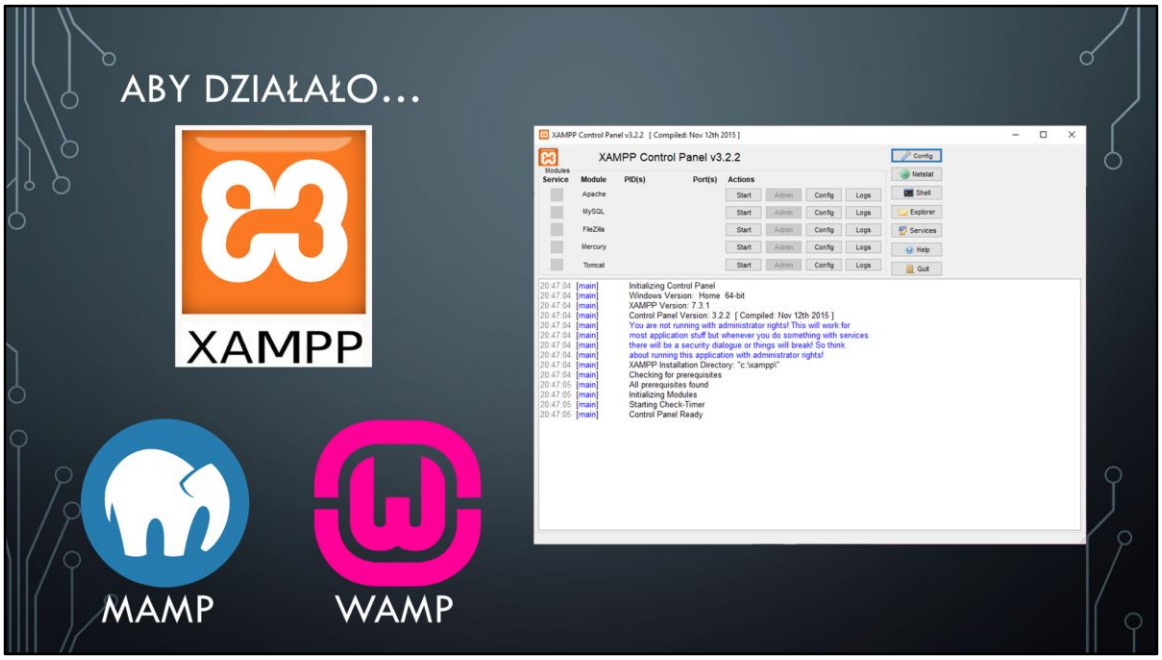
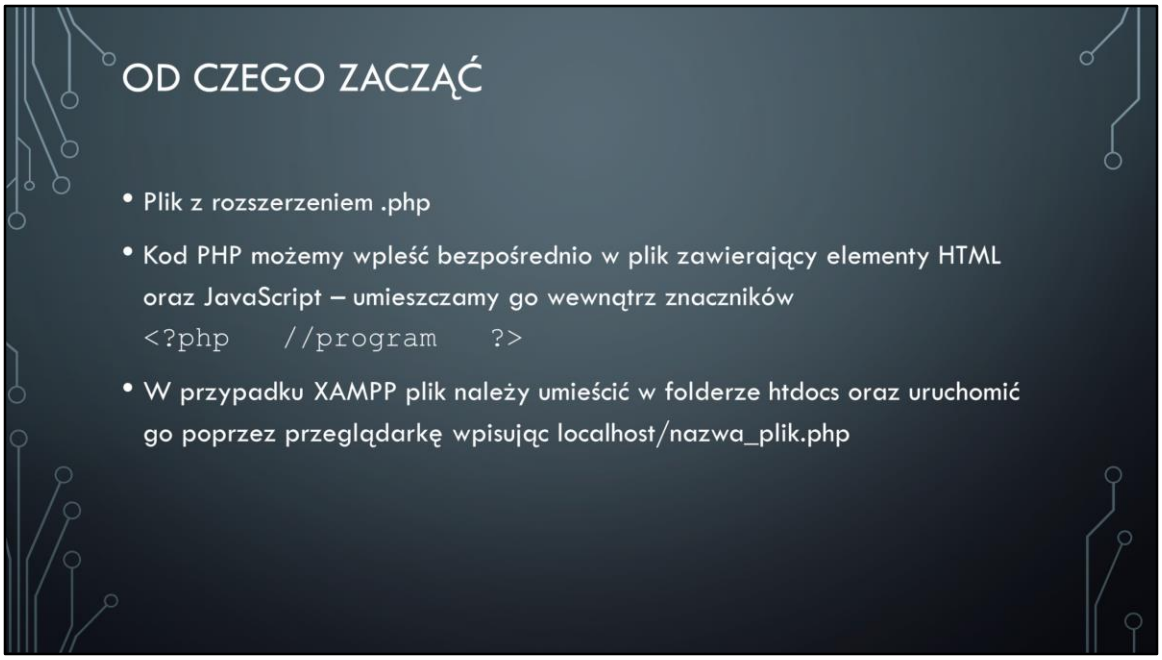

PHP
WG DEFINICJI - INTERPRETOWANY SKRYPTOWY JĘZYK PROGRAMOWANIA ZAPROJEKTOWANY DO GENEROWANIA STRON INTERNETOWYCH I BUDOWANIA APLIKACJI WEBOWYCH W CZASIE RZECZYWISTYM.
MATEUSZ RATAJCZAK

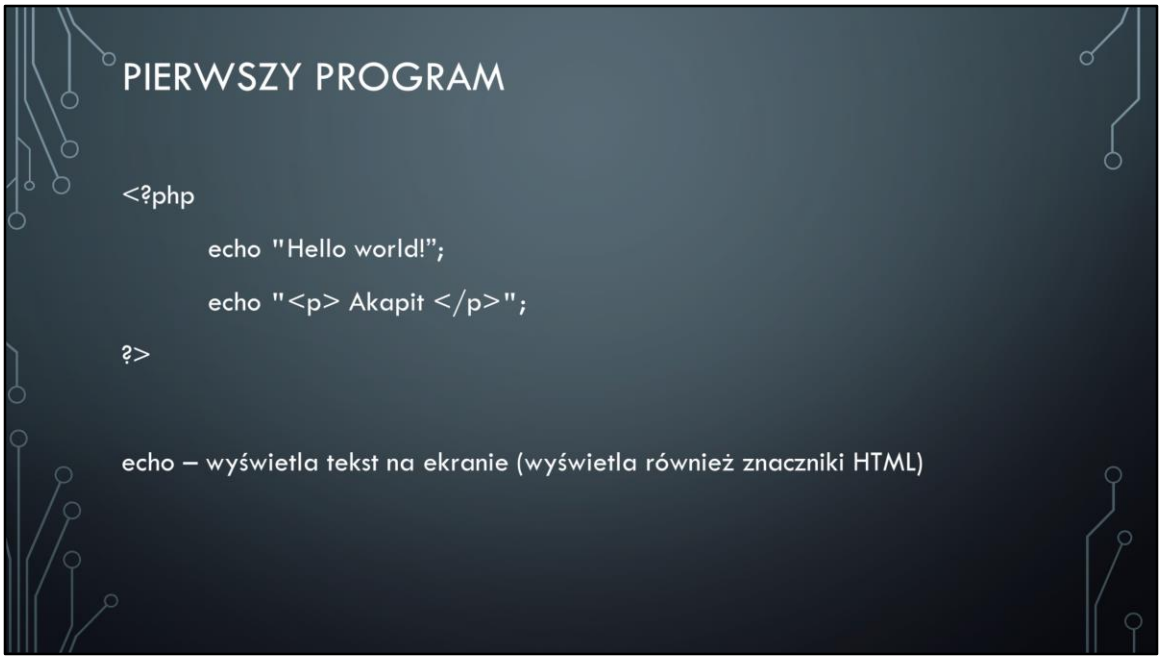


Do uruchamiania stron napisanych w HTML lub JavaScript wystarczyła nam jedynie zwykła przeglądarka. W przypadku stron zawierających kod PHP potrzebujemy serwera WWW. Osobiście, jeśli potrzebuję lokalnego serwera korzystam z XAMPP. Zapewnia on dostęp do serwera Apache oraz MySQL. Na potrzeby tych zajęć jest wystarczający. Jest on dostępny zarówno dla Windows, Linux oraz OS. Oczywiście istnieją inne lokalne serwery WWW m.in. WAMP, MAMP czy LAMP.

Jeszcze ogólna kwestia, nie zalecam kopiowania kodów z tej prezentacji ze względu na różne kodowanie znaków, lepiej przepisać te małe fragmenty niż szukać błędu typu złe kodowanie cudzysłowu.



Najważniejszą kwestią jest rozszerzenie pliku. Dowolny plik zawierający znaczniki HTML możemy przekształcić na plik PHP zmieniając jedynie rozszerzenie na .php. Po takiej zamianie strona będzie działała tak jak wcześniej, lecz będziemy mogli umieścić na niej kod w PHP, czego rozszerzenie .html nam nie zapewnia.
Kolejną ważną kwestią jest uruchomienie pliku – do tej pory wystarczyło, aby nasz plik zinterpretowała przeglądarka. Teraz musimy posłużyć się serwerem WWW lokalnym bądź zdalnym. W przypadku korzystania z XAMPP wszystkie pliki .php należy uruchamiać z poziomu katalogu htdocs, który najczęściej znajduje się: C:\xampp\htdocs dla systemu Windows.



Wcześniej wspominałem, że możemy łączyć w jednym pliku zarówno HTML jak i PHP, jednak nic nie stoi na przeszkodzie, aby stworzyć plik zawierający jedynie kod PHP. Oczywiście cały kod musi znajdować się wewnątrz „znaczników” <?php ?>
echo pozwala na wyświetlenie tekstu na ekranie, a jako tekst możemy również podać znaczniki HTML oraz kod w JS.
Każdą linię kończymy podobnie jak w C++ średnikiem.

ZMIENNE

- Zmienne tworzymy za pomocą $
- PHP nie posiada typów zmiennych
- Przykłady:
  - `$iterator = 0;`
  - `$tekst = "Hello world!";`
  - `$cena = 14.34;`

TABLICE

- Nie musimy z góry znać wielkości tablicy:

```
$tab[] = 1;
$tab[] = 2;
$tab[] = 3;
```

- Tablice indeksujemy od zera: `echo $tab[0];` wyświetli 1
- Tablice asocjacyjne:

```
tablica[klucz] = wartość;
tablica["imie"] = "Andrzej";
tablica["nazwisko"] = "Kowalski";
```

KONKATENACJA CIĄGÓW ZNAKÓW

- Umieszczanie zmiennych w tekście
  - `echo "$txt <br> Komunikat <br> $tablica[0] ";`
- Konkatencja ciągów za pomocą kropki:
  - `echo $txt . "<br> Komunikat" . "<br>" . $tablica[0];`

ELEMENTY JĘZYKA

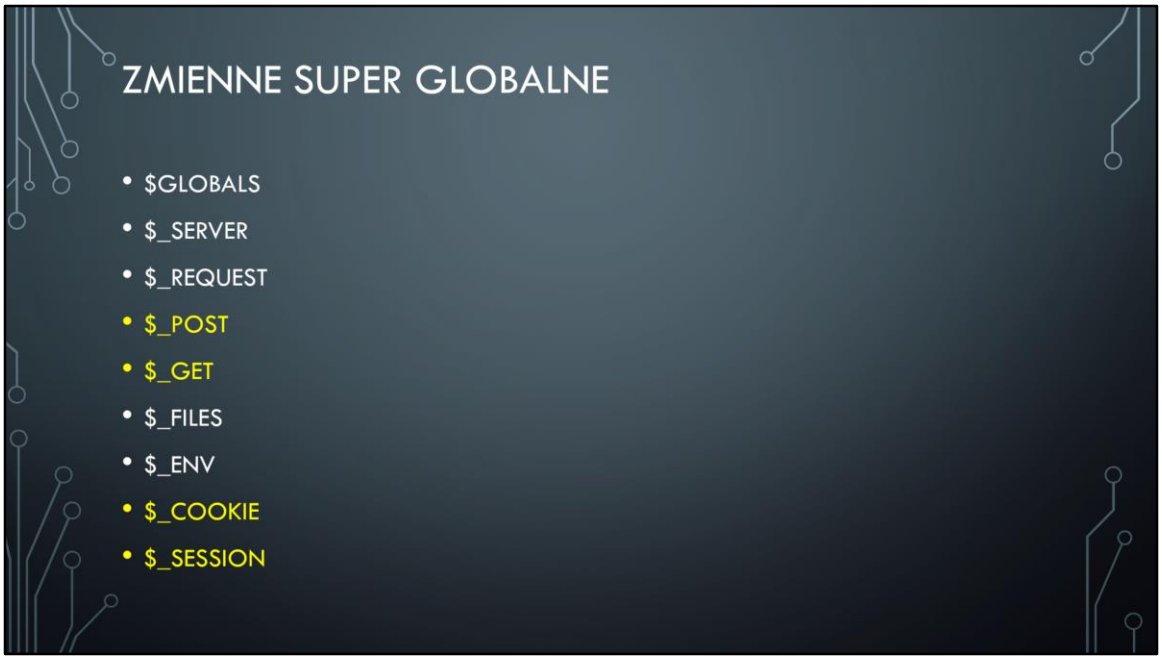
- Pętle, instrukcje warunkowe, switch podobnie jak w C++
  - Więcej: https://www.w3schools.com/php/
- Istnieje pętla foreach

```
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
  echo "$value <br>";
}
```

PRZYDATNE WBUDOWANE FUNKCJE

- date() – zwraca dzisiejszą datę, posiada wiele opcji formatowania:
  - `date("Y-m-d");` wyświetli datę w formacie YYYY-MM-DD np. 2020-03-23
  - więcej formatów: https://www.php.net/manual/en/function.date.php
  - Ustawienie naszej strefy czasowej:
    `date_default_timezone_set('Europe/Warsaw');`
- sort() – pozwala sortować tablicę

```
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);
```



Zmienne super globalne to rodzaj zdefiniowanych w języku specjalnych zmiennych. W dalszej części skupimy się na zmiennych $_POST, $_GET, $_COOKIE, $_SESSION.

ZMIENNE SUPER GLOBALNE

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

Zmienne $_POST oraz $_GET będziemy wykorzystywać do przesyłania danych pomiędzy różnymi podstronami.

Uzyskanie wartości zmiennej o nazwie imię przesłanej formularzem metodą POST.

```
echo $_POST["imie"];
```

Analogicznie w przypadku GET

FUNKCJA ISSET()

- Funkcja isSet() pozwala sprawdzić, czy dana zmienna istnieje, bądź jest ustawiona w obecnym kontekście np. czy akurat teraz jakiś formularz przesłał nam konkretną zmienną.

- Przykład:

```
if(isSet($_POST["zmienna"])) {
    echo "Zmienna istnieje i ma wartość " . $_POST["zmienna"];
}
```

ZMIENNE SUPER GLOBALNE - $_COOKIE

Przez zmienną $_COOKIE możemy uzyskać informacje zapisane w plikach cookie na komputerze użytkownika.

Tworzenie cookie:

```
<?php
setcookie("nazwa", "wartość", time() + (86400 * 30), "/");
// 86400 = 1 dzień
?>
```

Wyświetlenie wartości – zabezpieczamy na wypadek gdyby zmienna wygasła, bądź nie istniała:

```
if(isSet($_COOKIE["nazwa"])) { echo $_COOKIE["nazwa"]; }
```

Parametry tworzenia cookie:

1. Nazwa zmiennej
2. Wartość zmiennej
3. Data wygaśnięcia time() + czas w sekundach – 86400 sekund ma doba
4. „/” – ścieżka gdzie zostanie zapisane cookie

ZMIENNE SUPER GLOBALNE

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

Zmienna $_SESSION pozwala przechowywać wartości na serwerze podczas trwania sesji. Jest ona dostępna w obrębie wszystkich plików na stronie.

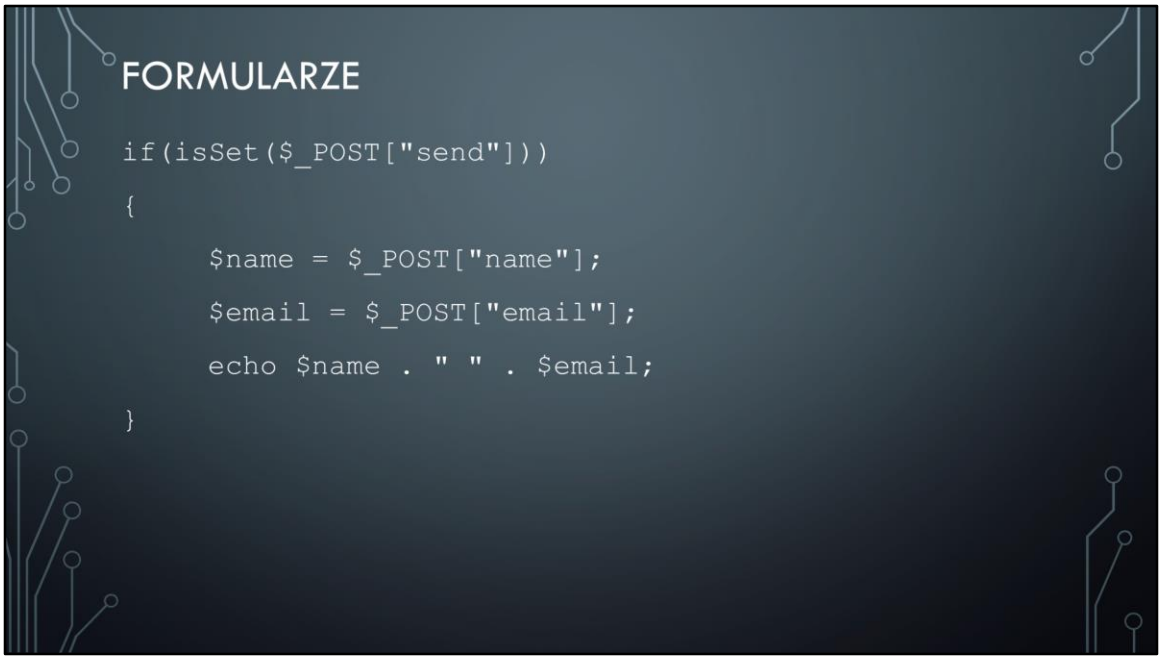
Aby wykorzystywać ten mechanizm w każdym pliku php musimy w pierwszej linii uruchomić sesje poleceniem:

```
<?php session_start(); ?>
```

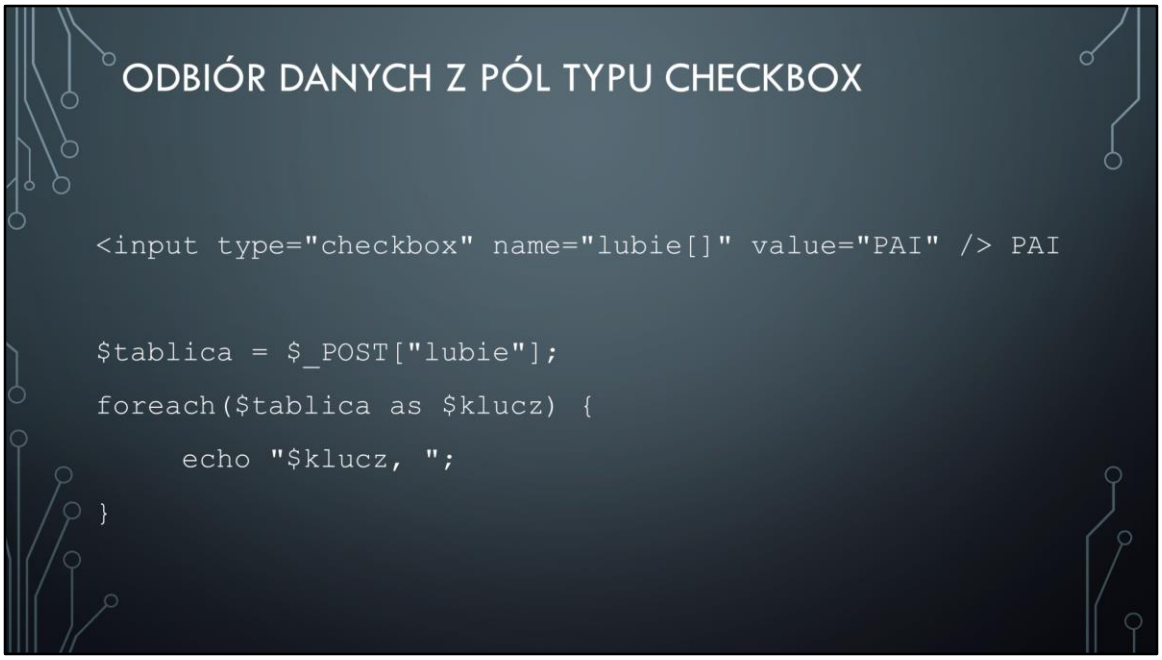
Ustawianie wartości: `$_SESSION["nazwa"] = wartość;`

Wyświetlanie: `echo $_SESSION["nazwa"];`

Teraz zajmiemy się obsługą formularzy za pomocą języka PHP.
Przyjrzyjmy się formularzowi na slajdzie szczególnie zwracając uwagę na elementy zaznaczone na żółto.
W znaczniku form atrybut action oznacza miejsce, do którego zostanie przeniesiony użytkownik po naciśnięciu przycisku type=„submit”. W naszym przypadku zostanie przekierowany na stronę welcome.php
Atrybut method oznacza w jaki sposób będziemy przesyłać dane – mamy dostęp do typu POST oraz GET. Przesyłając zmienne metodą GET przeglądarka doklei je do adresu URL, natomiast przesyłając metodą POST wartości zmiennych nie będą widoczne dla zwykłego użytkownika.
Bardzo ważne są również nazwy konkretnych pól formularza, ponieważ dzięki nim możemy uzyskać wartość jaką wpisał w to pole użytkownik. Polecam również mimo, że nie jest to wymagane, ustawić nazwę dla pola typu submit, ponieważ ułatwia to weryfikację, który formularz został teraz przesłany, gdy jeden plik odbiera różne formularze. Na następnym slajdzie przedstawiłem w jaki sposób odebrać wartości z tego formularza.



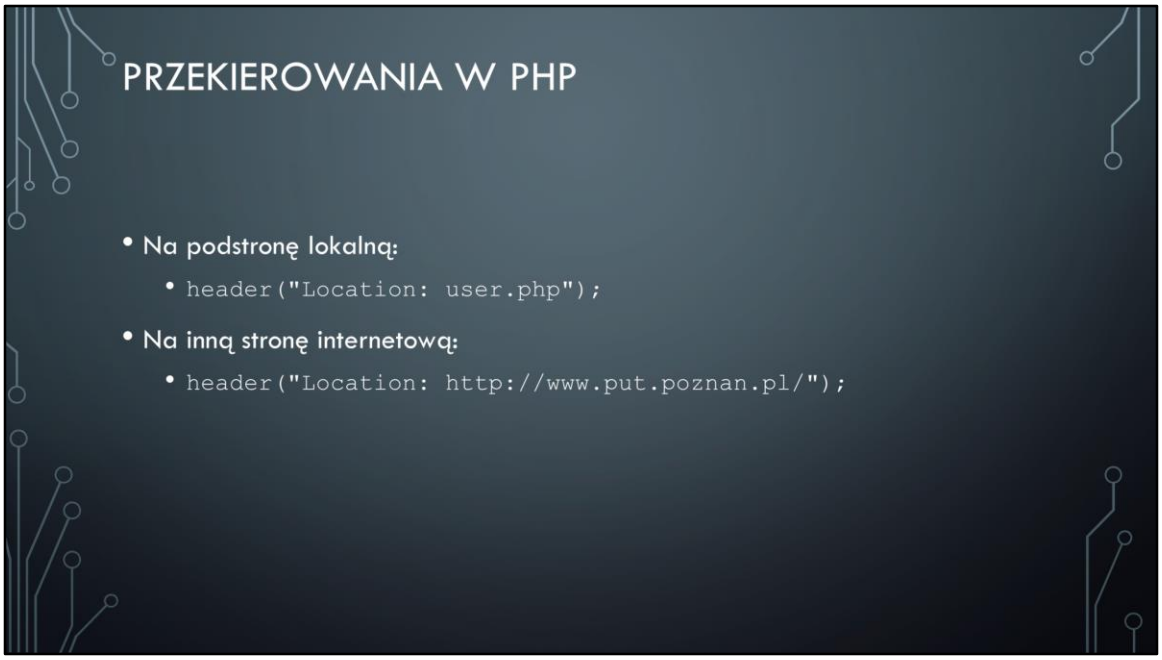
Gdybyśmy przesyłali dane używając GET należałoby zamienić zmienne globalne $_POST na $_GET. Dodatkowo, jak wcześniej wspomniałem, sprawdzam czy zmienna send jest ustawiona – skoro jest to znaczy, że ten formularz został przesłany i możemy pobrać, bądź spróbować pobrać, wartości zmiennych $_POST[„name”] i $_POST[„email”]. Oczywiście w pełni poprawnym kodzie należałoby również zabezpieczyć pobranie zmiennych funkcją isSet.
Jeszcze mały niuans – PHP nie jest czuły na wielkość liter, czyli zadziała zarówno isSet jak i isset.



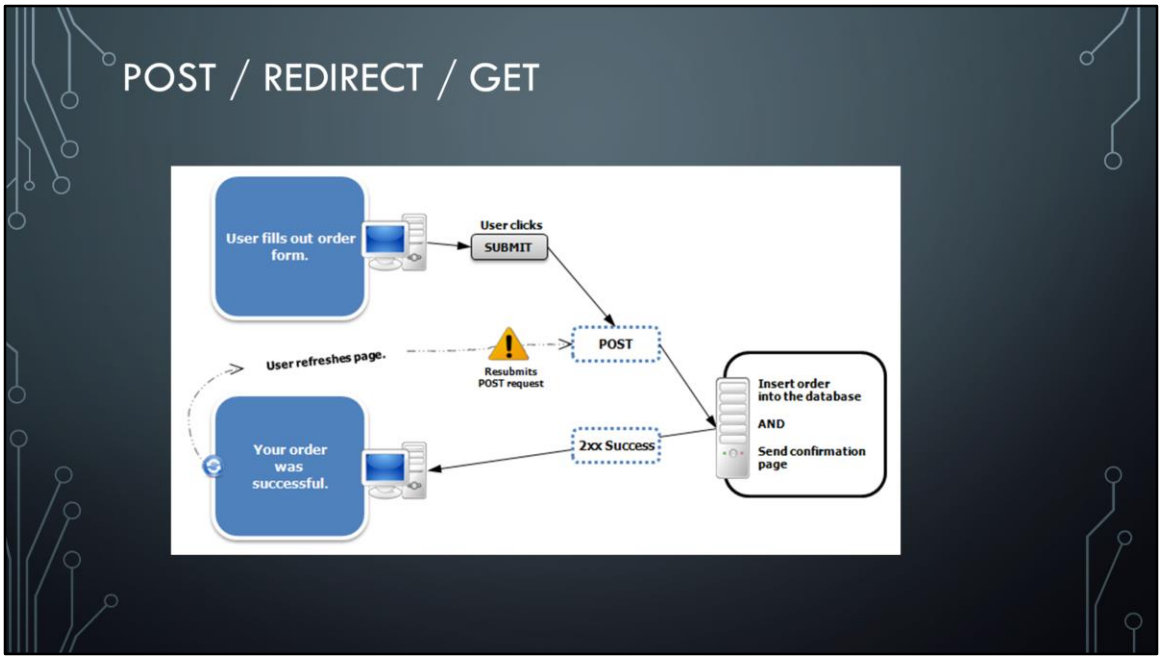
Jako nazwę pola w formularzu wpisujemy tablicę w sposób pokazany na slajdzie. Dostęp do zaznaczonych pól formularza uzyskujemy dzięki pętli foreach, która zwróci nam value zaznaczonych pól.
Typ radio traktujemy podobnie jak normalne pole np. tekstowe, jednak podobnie jak w checkbox zostanie zwrócona wartość value przycisku radio.
Ogólnie każde pole może mieć znacznik value, a jego zawartość zostanie wpisana do formularza już na początku. Możliwe jest również ustawianie jako value wartości zmiennej języka php, np. <input value=„$zmienna” … />

LISTY ROZWIJALNE ORAZ POLA HIDDEN

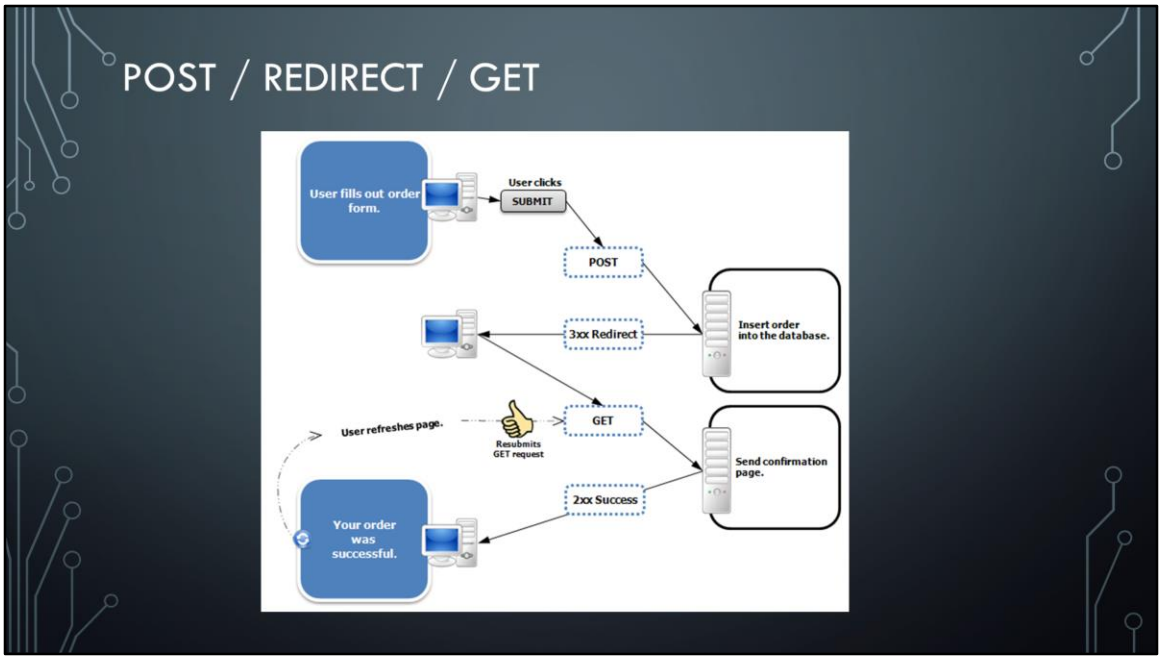
- Formularz zawierający listę rozwijalną:
  - Więcej: https://www.w3schools.com/tags/att_select_form.asp
- Pola ukryte – type="hidden" – pole formularza, którym możemy przesłać jakąś wartość. Pole jest niewidoczne dla użytkownika
  - `<input type="hidden" name="ukryte" value="123" />`



Podczas budowy stron, czasami musimy przekierować użytkownika na konkretną podstronę. Jednym z przykładów może być zastosowanie formularza logowania, tego samego zarówno dla użytkowników jak i administratorów. Wtedy po udanej weryfikacji użytkownika można zweryfikować poziom uprawnień i przykładowo przenieść zwykłego użytkownika na podstronę user.php, a osobę logującą się z uprawnieniami administratora na podstronę admin.php.
I teraz jeszcze jedna ważna rzecz – mimo, że zwykły użytkownik nigdy nie powinien mieć hiperłącza do sekcji administratora może np. podejrzeć adres i wpisać go bezpośrednio. W takich przypadkach należy na początku pliku sprawdzić, czy dana osoba ma uprawnienia do żądanej podstrony np. wykorzystując zmienne sesji, w której możemy trzymać „rolę” zalogowanego użytkownika.



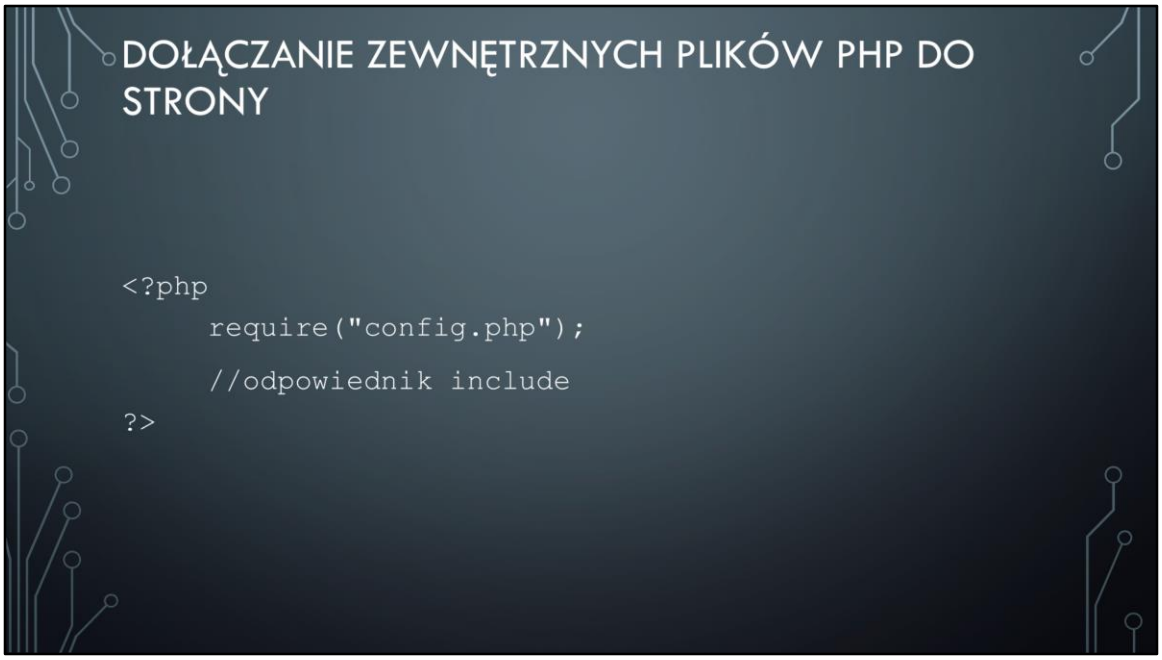
Teraz omówimy problem, z którym każdy z nas spotkał się na stronach internetowych. Mianowicie chodzi o komunikat „Potwierdź ponowne przesłanie formularza". Problem występuje, gdy użytkownik po przesłaniu formularza chce np. odświeżyć stronę, bądź kliknąć przycisk wstecz. Skonstruujmy mały przykład. Użytkownik wypełnia formularz na stronie index.php. Przetwarzanie formularza również znajduje się na tej stronie. Teraz, jeśli użytkownik odświeży stronę, pojawi się niechciany komunikat. Wynika to z faktu, że zmienne POST dalej istnieją w pamięci przeglądarki i mogą być ponownie wykorzystane. Jeśli użytkownik zatwierdzi ponowne przesłanie formularza, formularz zostanie przesłany poprawnie, lecz otrzymamy drugi raz te same dane. Problem jest bardzo poważny np. w bazach danych, kiedy dane przesyłane formularzem dodajemy do tabel. Mogą wtedy pojawić się wiersze zduplikowanych informacji.



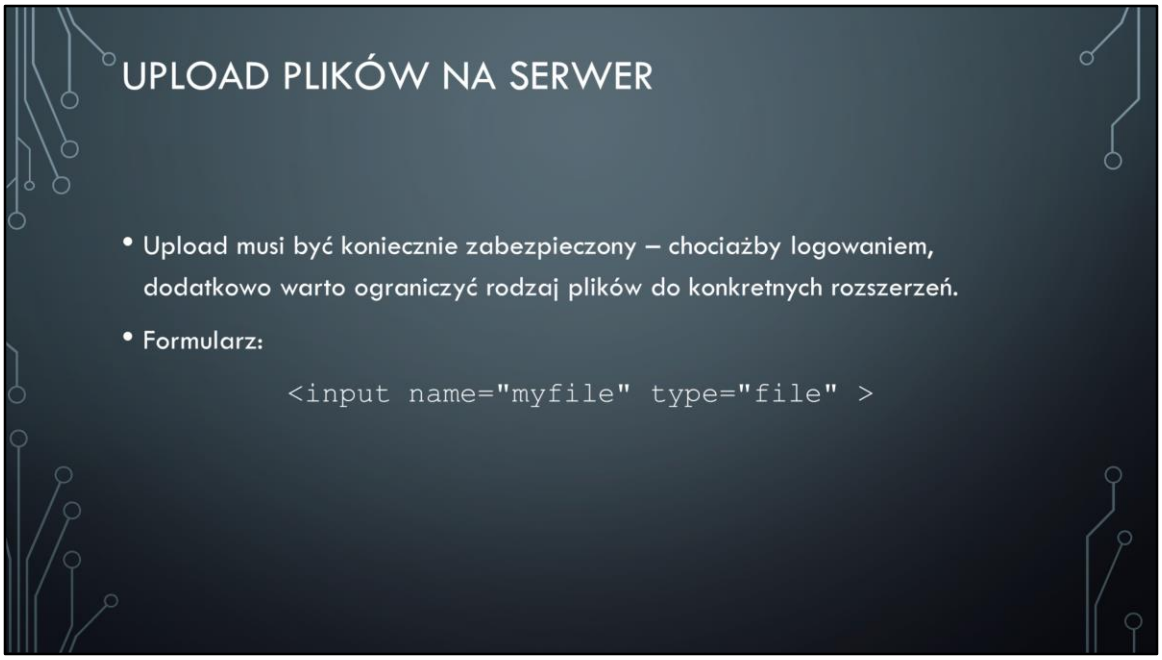
W języku PHP został skonstruowany wzorzec Post/Redirect/Get. W tym przypadku również posłużę się przykładem. Formularz znajduje się na stronie index.php, ale jego obsługa jest zaimplementowana w nowym pliku np. o nazwie logowanie.php. W pliku logowanie.php, pobieramy dane z formularza oraz je przetwarzamy. Na końcu takiego pliku umieszczamy przekierowanie, najczęściej funkcją header(), do strony, na której użytkownik miał się znaleźć po przesłaniu formularza. W takim podejściu unikami błędów wynikających z cofania bądź odświeżania strony, gdyż po przekierowaniu wartości przesłanych zmiennych POST przestają być aktywne i nie dojdzie do duplikacji danych.
Czasami podczas przetwarzania chcielibyśmy wyświetlić komunikat użytkownikowi. W tym podejściu jest to niemożliwe bo użytkownik nie widzi tej strony na której dokonujemy przetwarzania*. W takim wypadku komunikaty musimy „łapać" do zmiennych sesji i wyświetlać je na stronie do której przekierujemy użytkownika po przetwarzaniu.

* - Mówi się, że użytkownik tej strony nie widzi, jednak w przypadku wolnego łącza lub dużej liczby danych ta strona może zostać wyświetlona. Nie oszukujmy się biała czysta strona, która pojawia się np. na 10 sekund a następnie sama znika może zaniepokoić użytkownika, więc jeśli wiemy o tym warto na stronie na której

przetwarzamy również dodać jakiś tekst typu: „Twoje dane są przetwarzane – za chwilę przeniesiemy Ciebie na właściwą stronę”.



Czasami warto np. własne funkcje, umieścić w dodatkowym pliku php, a następnie dołączać go do konkretnych podstron.
Wyprzedzając już kolejny temat, warto zapisać w osobnym pliku np. config.php, poświadczenia logowania do bazy danych. W przypadku zmiany serwera, łatwiej dokonać modyfikacji danych dostępowych w jednym miejscu.



Dodając do strony internetowej pliki przez jakiś system zarządzania treścią możemy wrzucić np. zdjęcia na dwa sposoby. Pierwszym z nich jest umieszczenie zdjęcia w bazie danych. Jest to jednak z uwagi na wielkość plików przestarzałe podejście, które powoduje długi czas ładowania strony. Innym podejściem jest umieszczenie zdjęć bezpośrednio na serwerze, a w bazie danych zapisanie jedynie nazwy bądź ścieżki do danego pliku. Obecnie z takiego rozwiązania korzysta m.in. Wordpress.
Jednak na upload plików na serwer trzeba patrzeć bardzo poważnie. W końcu zezwalamy komuś wrzucić własny plik na nasz serwer. Pisząc taki skrypt bez zabezpieczeń narażamy się na niebezpieczeństwo. „Haker” wystarczy, że wrzuci plik, uruchomi go i w zależności od typu, może wykraść nasze dane bądź uszkodzić stronę.

```
$currentDir = getcwd();
$uploadDirectory = "/zdjeciaUzytkownikow/";
$fileName = $_FILES['myfile']['name'];
$fileSize = $_FILES['myfile']['size'];
$fileTmpName  = $_FILES['myfile']['tmp_name'];
$fileType = $_FILES['myfile']['type'];
if($fileName != "" and
      ($fileType == 'image/png' or $fileType == 'image/jpeg'
      or $fileType == 'image/JPEG' or $fileType == 'image/PNG'
))
{
      $uploadPath = $currentDir . $uploadDirectory . $fileName;
      if(move_uploaded_file($fileTmpName, $uploadPath))
            echo "Zdjęcia zostało załadowane na serwer FTP";
}
```

Tym razem skorzystałem z kolejnej zmiennej globalnej $_FILES. Na początku odczytuję do zmiennej ścieżkę do folderu w którym się obecnie znajdujemy. Z kolejnej linijki można wywnioskować, że będziemy umieszczać na serwerze zdjęcia w podfolderze zdjeciaUzytkownikow. Zmienna $_FILES zawiera tablicę i możemy wyciągnąć z niej zarówno nazwę pliku, jego rozmiar oraz tymczasową nazwę pliku oraz jego typ. Następnie w ramach weryfikacji sprawdzamy, czy nazwa przesłanego pliku nie jest pusta oraz czy jest to zdjęcie w formacie JPEG lub PNG. Jeśli tak to możemy za pomocą metody move_uploaded_file() wrzucić nasz plik na serwer do danej lokalizacji.

Tak jak mówiłem wcześniej taki formularz musi być zabezpieczony np. logowaniem. Tutaj użyłem sprawdzania rozszerzenia pliku, jednak nie jest to wystarczające zabezpieczenie, gdyż można przesłać tak spreparowany plik, który nasz system uzna za plik graficzny, a w rzeczywistości może zawierać wrogi kod PHP lub JS.

USUWANIE ZNAKÓW SPECJALNYCH PRZESŁANYCH PRZEZ FORMULARZ

```
function test_input($data)
{
      $data = trim($data);
      $data = stripslashes($data);
      $data = htmlspecialchars($data);
      return $data;
}
```

Funkcja trim usuwa białe znaki z początku i końca ciągu.
Funkcja striplslashes usuwa z ciągu znaki \
Funkcja htmlspecialchars usuwa znaki specjalne języka html zamieniając je na niegroźne odpowiedniki, np. < zamienia na <

ZADANIA …