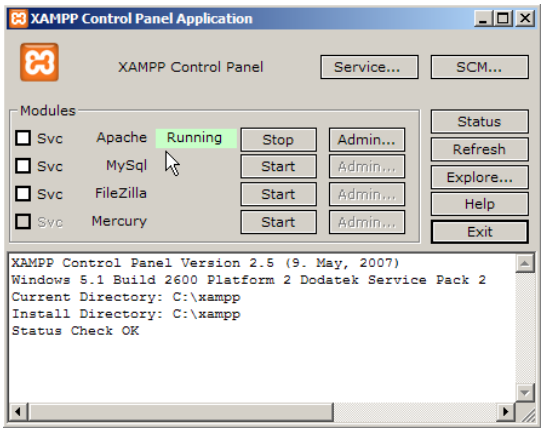
Aplikacje WWW - laboratorium

Język PHP

Celem ćwiczenia jest przygotowanie prostej aplikacji internetowej zaimplementowanej w języku PHP. Aplikacja ilustruje takie mechanizmy jak: obiektowość w PHP, obsługa formularzy oraz obsługa zmiennych sesyjnych.

Ćwiczenia można wykonać na dowolnym serwerze, na którym zainstalowano serwer HTTP (np. Apache) z obsługą PHP. Skrypty PHP należy zapisać w odpowiednim katalogu na serwerze jako pliki z rozszerzeniem `.php`. Do edycji plików należy wykorzystać dowolny edytor tekstowy. W ćwiczeniu jako przykładowe środowisko wykorzystano środowisko XAMPP (http://www.apachefriends.org/en/xampp.html)

1. Uruchom panel sterujący środowiska XAMPP i uruchom serwer Apache.

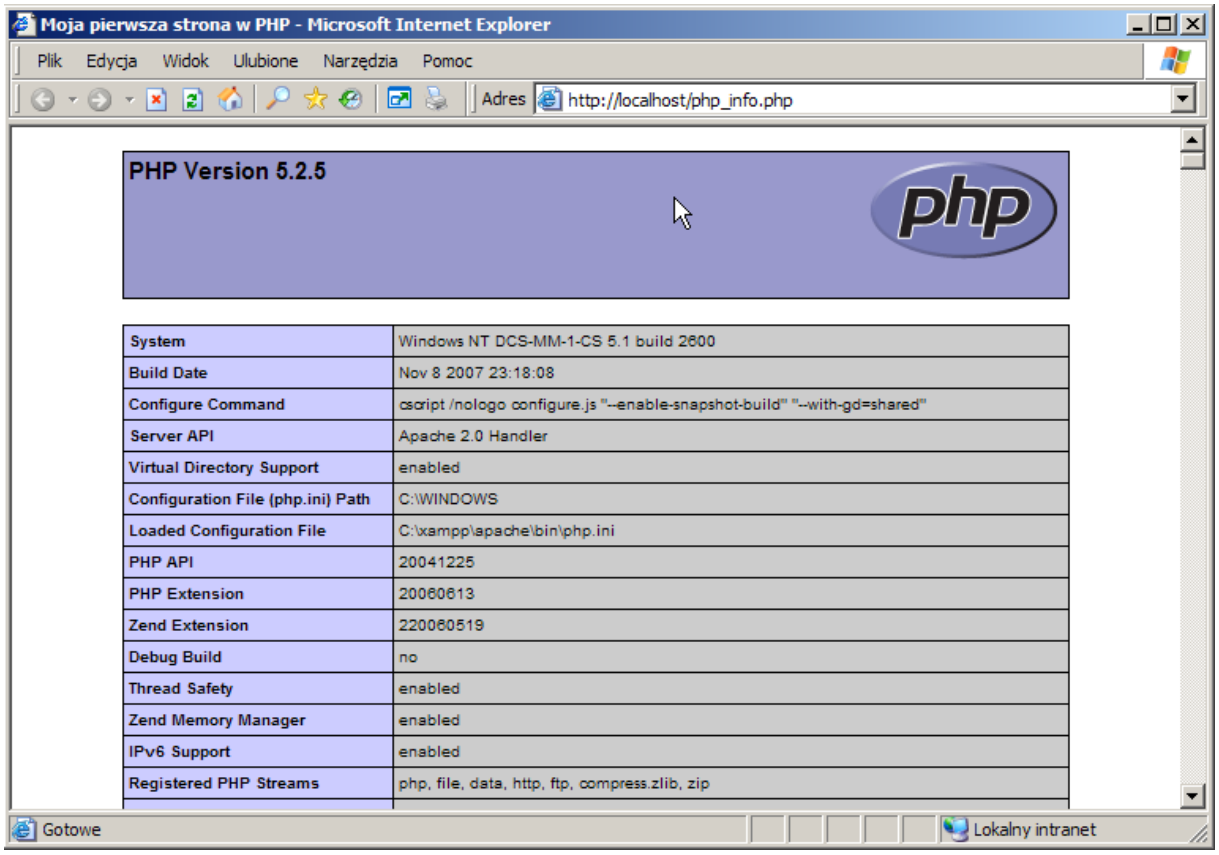


2. W katalogu `$XAMPP_HOME/htdocs` utwórz plik `php_info.php` i umieść w nim poniższy kod.

```
<html>
  <head>
  <title>Moja pierwsza strona w PHP</title>
  </head>

  <body>
    <?php phpinfo(); ?>
  </body>
</html>
```

3. Otwórz w przeglądarce adres http://localhost/php_info.php i przeanalizuj wyniki działania skryptu. Zapoznaj się ze środowiskiem. Zwróć szczególną uwagę na listę zmiennych środowiskowych udostępnianych przez serwer skryptowi PHP (sekcja Apache Environment) i listę nagłówków HTTP zawartych w żądaniu i odpowiedzi (HTTP Headers Information).



4. W pierwszej kolejności zapoznamy się z listą zmiennych udostępnianych skryptowi przez serwer. Zwróć uwagę na charakterystyczne dla PHP użycie tablicy asocjacyjnej `$_SERVER` oraz sposób iteracji po elementach takiej tablicy. Utwórz nowy plik `file01.php` i umieść w nim poniższy kod.

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>

  <h1>Zmienne serwera</h1>
  <ul>
  <?php
    foreach ($_SERVER as $zmienna => $wartosc)
    {
      echo "<li><b>".$zmienna."</b> : ".$wartosc."</li>";
    }
  ?>
  </ul>
</body>
</html>
```

6. Kolejnym krokiem będzie przejęcie danych z prostego formularza i wyliczenie sumy liczb. Utwórz plik `file02.php` i umieść w nim poniższy kod.

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>

  <form action="file02.php" method="GET">
    Podaj wartość x:  <input type="text" name="x"/>
    <input type="submit" value="wyślij"/>
    <input type="reset"  value="wyczyść"/>
  </form>

  <?php
    $x = $_GET['x'];

    $suma = 0;
    for ($i=0; $i<$x; $i++)
      $suma += $i;

    echo "suma liczb od 1 do ".$x." wynosi: ".$suma;
  ?>

</body>
</html>
```

7. Kod w pliku `file02.php` jest niepoprawny, ponieważ wykorzystuje wartość zmiennej przekazanej przez użytkownika bez jakiejkolwiek walidacji poprawności. Zmodyfikuj główny fragment skryptu PHP w poniższy sposób i przetestuj poprawność rozwiązania. Czy potrafisz wykonać to samo obliczenie bez użycia pętli?

```
<?php
  $x = $_GET['x'];

  if (is_numeric($x))
  {
    $suma = 0;
    for ($i=0; $i<$x; $i++)
      $suma += $i;

    echo "suma liczb od 1 do ".$x." wynosi: ".$suma;
  }
  else
    echo "<div style=\"color: red\">proszę podać poprawną liczbę</div>";
?>
```

8. W następnym kroku zaimplementujemy algorytm Euklidesa i wykorzystamy do walidacji poprawności danych mechanizm filtrów. Zwróć uwagę na sposób walidacji. Utwórz plik `file03.php` i umieść w nim poniższy kod.

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>
  <form action="file03.php" method="GET">
    Podaj wartość x:  <input type="text" name="x"/>
    Podaj wartość y:  <input type="text" name="y"/>
    <input type="submit" value="wyślij"/>
    <input type="reset"  value="wyczyść"/>
  </form>

  <?php
    $int_options = array("options" =>
      array("min_range" => 1,"max_range" => 1000000));

    $x = filter_var($_GET['x'], FILTER_VALIDATE_INT, $int_options);
    $y = filter_var($_GET['y'], FILTER_VALIDATE_INT, $int_options);

    if ($x && $y)
    {
      while ($x != $y)
      {
        if ($x > $y)
          $x -= $y;
        else
          $y -= $x;
      }
      echo "największy wspólny dzielnik tych liczb to ".$x;
    }
    else
      echo "<div style=\"color: red\">podaj poprawne liczby</div>";
  ?>
</body>
</html>
```

9. Podobnie jak w przypadku języka JavaScript, jednym z najwygodniejszych mechanizmów walidacji danych wejściowych są wyrażenia regularne. Poniższy przykład pokazuje, w jaki sposób można wykorzystać język wyrażeń regularnych do przeprowadzenia walidacji formatu karty kredytowej. Numer karty kredytowej to cztery grupy czterocyfrowe opcjonalnie rozdzielone spacją lub myślnikiem, a data ważności to dwie grupy dwucyfrowe rozdzielone znakiem '/'. Utwórz plik `file04.php` i umieść w nim poniższy kod.

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>
  <form action="file04.php" method="GET">
  Podaj numer karty:
    <input type="text" maxlength="19" size="19" name="numer_karty" />
  Podaj datę ważności:
    <input type="text" maxlength="5" size="5" name="data_waznosci" />
    <input type="submit" value="wyślij"/>
    <input type="reset"  value="wyczyść"/>
  </form>

  <?php
    $numer_karty = $_GET['numer_karty'];
    $data_waznosci = $_GET['data_waznosci'];

    if (!preg_match("/^(\d{4}[\s\-]?){4}$/",$numer_karty))
      echo "<div style=\"color: red\">podaj poprawny numer karty</div>";
    elseif (!preg_match("/^\d{2}\/\d{2}$/",$data_waznosci))
      echo "<div style=\"color: red\">podaj poprawną datę</div>";
    else
      echo "podano poprawny numer karty kredytowej";
  ?>
</body>
</html>
```

10. Kod w pliku `file04.php` ma dwie wady. Po pierwsze, wyrażenie regularne zaakceptuje łańcuch 16 cyfr zakończony spacją. Zmodyfikuj wyrażenie regularne w taki sposób, aby na końcu numeru karty kredytowej nie mogła się pojawić spacja. Po drugie, w przypadku popełnienia błędu zawartość formularza jest tracona – stanowi to bardzo poważną niedogodność dla użytkownika. Zmodyfikuj kod tworzący formularz HTML w taki sposób, aby nie tracić wartości wprowadzanych do pól formularza.

```
<form action="file04.php" method="GET">
  Podaj numer karty:
  <input type="text" maxlength="19" size="19" name="numer_karty"
         value="<?php echo $_GET['numer_karty']; ?>"/>
  Podaj datę ważności:
  <input type="text" maxlength="5" size="5" name="data_waznosci"
         value="<?php echo $_GET['data_waznosci']; ?>"/>
  <input type="submit" value="wyślij"/>
  <input type="reset"  value="wyczyść"/>
</form>
```

11. Kolejny przykład obrazuje sposób wykorzystania funkcji i obiektów w PHP. Rozpocznij od utworzenia pliku `file05.php` i umieszczenia w nim poniższego kodu.

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>


  <form action="file05.php" method="GET">
    Podaj długość:    <input type="text" size="4" name="x"/>
    Podaj szerokość:  <input type="text" size="4" name="y"/>
    <input type="submit" value="wyślij"/>
    <input type="reset"  value="wyczyść"/>
  </form>

  <?php
    function pole($a,$b)
    {
      return $a * $b;
    }
  ?>

  <?php
    $int_options = array("options"=>array("min_range"=>1));
    $x = filter_var($_GET['x'], FILTER_VALIDATE_INT, $int_options);
    $y = filter_var($_GET['y'], FILTER_VALIDATE_INT, $int_options);

    if ($x && $y)
      echo "pole prostokąta o wymiarach $x x $y wynosi ".pole($x,$y);
    else
      echo "<div style=\"color: red\">podaj poprawne wymiary</div>";
  ?>

</body>
</html>
```

12. Następnym krokiem będzie przepisanie powyższego skryptu w taki sposób, aby korzystał z opcji obiektowej. Zamień główny kod PHP na poniższy kod i przetestuj aplikację.

```
<?php
  class prostokat
  {
    public $dlugosc;
    public $szerokosc;

    function pole()
    {
      return $this->dlugosc * $this->szerokosc;
    }
  }
?>

<?php
  $int_options = array("options"=>array("min_range"=>1));
  $x = filter_var($_GET['x'], FILTER_VALIDATE_INT, $int_options);
  $y = filter_var($_GET['y'], FILTER_VALIDATE_INT, $int_options);

  if ($x && $y)
  {
    $moj_prostokat = new prostokat;
    $moj_prostokat->dlugosc = $x;
    $moj_prostokat->szerokosc = $y;

    echo "pole prostokąta o wymiarach ".$x."x".$y." wynosi
".$moj_prostokat->pole();
  }
  else
    echo "<div style=\"color: red\">podaj poprawne wymiary</div>";
?>
```

13. Powyższy kod łamie zasadę hermetyczności i nie wykorzystuje konstruktorów. Napraw klasę `prostokat` w następujący sposób.

```
<?php
  class prostokat
  {
    private $dlugosc;
    private $szerokosc;

    function prostokat($dlugosc, $szerokosc)
    {
      $this->dlugosc = $dlugosc;
      $this->szerokosc = $szerokosc;
    }

    function pole()
    {
      return $this->dlugosc * $this->szerokosc;
    }
  }
?>

<?php
  $int_options = array("options"=>array("min_range"=>1));
  $x = filter_var($_GET['x'], FILTER_VALIDATE_INT, $int_options);
  $y = filter_var($_GET['y'], FILTER_VALIDATE_INT, $int_options);

  if ($x && $y)
  {
    $moj_prostokat = new prostokat($x,$y);

    echo "pole prostokąta o wymiarach ".$x."x".$y." wynosi
".$moj_prostokat->pole();
  }
  else
    echo "<div style=\"color: red\">podaj poprawne wymiary</div>";
?>
```

14. Programując w PHP mamy również do dyspozycji mechanizm dziedziczenia. Poniższy przykład obrazuje sposób wykorzystania dziedziczenia do wyliczenia objętości prostopadłościanu. Zwróć szczególną uwagę na sposób wywołania konstruktora nadklasy wewnątrz konstruktora klasy. Umieść poniższy kod w pliku `file06.php`.

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>
 <form action="file06.php" method="GET">
    Podaj długość:  <input type="text" size="4" name="x"/>
    Podaj szerokość:  <input type="text" size="4" name="y"/>
    Podaj wysokość:  <input type="text" size="4" name="z"/>
    <input type="submit" value="wyślij"/>
    <input type="reset"  value="wyczyść"/>
  </form>

  <?php
    class prostokat
    {
      private $dlugosc;
      private $szerokosc;

      function prostokat($dlugosc, $szerokosc)
      {
        $this->dlugosc = $dlugosc;
        $this->szerokosc = $szerokosc;
      }

      function pole()
      {
        return $this->dlugosc * $this->szerokosc;
      }
    }

    class prostopadloscian extends prostokat
    {
      private $wysokosc;

      function prostopadloscian($dlugosc, $szerokosc, $wysokosc)
      {
         parent::prostokat($dlugosc, $szerokosc);
         $this->wysokosc = $wysokosc;
      }

      function objetosc()
      {
        return parent::pole() * $this->wysokosc;
      }
    }
  ?>

  <?php

    $int_options = array("options"=>array("min_range"=>1));
    $x = filter_var($_GET['x'], FILTER_VALIDATE_INT, $int_options);
    $y = filter_var($_GET['y'], FILTER_VALIDATE_INT, $int_options);
    $z = filter_var($_GET['z'], FILTER_VALIDATE_INT, $int_options);
```

```
    if ($x && $y && $z)
    {
      $moj_prostopadloscian = new prostopadloscian($x,$y,$z);
      echo "objętość prostopadłościanu o wymiarach ".$x."x".$y."x".$z."
wynosi ".$moj_prostopadloscian->objetosc();
    }
    else
      echo "<div style=\"color: red\">podaj poprawne wymiary</div>";
  ?>

</body>
</html>
```

15. Język PHP pozwala użytkownikowi na elastyczne raportowanie i obsługiwanie błędów pojawiających się w programie. Następny przykład obrazuje koncepcję własnej funkcji obsługi błędu. Zmodyfikuj kod w pliku `file06.php` w następujący sposób. Dodaj definicję funkcji custom_error_handler().

```
function custom_error_handler($error_code, $error_message)
{
  echo "<div style=\"{position: absolute; left: 10px; bottom: 100px;
                      color: red}\" >";
  echo "<hr><b>Error:</b> [$error_code] $error_message</div>";
  die();
}
```

a następnie, tuż przed walidacją zmiennych z formularza, wskaż przygotowaną przez siebie funkcję jako domyślną funkcję obsługi błędów:

```
set_error_handler("custom_error_handler");
```

W ostatnim kroku zamień wyświetlenie komunikatu o błędnych wymiarach na jawne wywołanie błędu:

```
else
  trigger_error("proszę podać poprawne wymiary");
```

16. Poza obsługą błędów język PHP umożliwia także zgłaszanie i przechwytywanie wyjątków. Utwórz plik `file07.php` i umieść w nim poniższy kod

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>
  <form action="file07.php" method="GET">
    Podaj wartość:  <input type="text" name="wartosc"/>
    <input type="submit" value="wyślij"/>
    <input type="reset"  value="wyczyść"/>
  </form>


  <?php

    $wartosc = $_GET['wartosc'];

    try
    {
      if (is_numeric($wartosc))
        throw new Exception("wartość numeryczna");
      elseif (is_string($wartosc))
        throw new Exception("wartość znakowa");
    }
    catch(Exception $e)
    {
      echo "Wystąpił wyjątek: ".$e->getMessage();
    }
  ?>

</body>
</html>
```

17. Istnieje także możliwość definiowania własnych wyjątków jako klas potomnych dziedziczących z klasy `Exception`. W poniższym przykładzie przygotujemy specjalizowaną klasę `NumericException` służącą do obsługi sytuacji, w której użytkownik podaje zmienną liczbową. Umieść w pliku `file07.php` poniższy kod.

```
<?php
  class NumericException extends Exception
  {
    function errorMessage()
    {
      $msg = "Błąd w linii ".$this->getLine()." w pliku ".
              $this->getFile().":".$this->getMessage();
      return $msg;
    }
  }

  $wartosc = $_GET['wartosc'];

  try
  {
    if (is_numeric($wartosc))
      throw new NumericException("wartość numeryczna");
    elseif (is_string($wartosc))
      throw new Exception("wartość znakowa");
  }
  catch(NumericException $e)
  {
    echo $e->errorMessage();
  }
  catch(Exception $e)
  {
    echo "Wystąpił wyjątek: ".$e->getMessage();
  }
?>
```

18. Ostatnie trzy ćwiczenia mają na celu przybliżenie studentom podstawowych mechanizmów wykorzystywanych w prostych aplikacjach internetowych. Pierwszy przykład obrazuje działanie mechanizmu ciasteczek (ang. *cookies*). Utwórz plik `file08.php` i umieść w nim poniższy kod. Zwróć uwagę, że ciasteczka są przesyłane w nagłówku pakietu http, a zatem polecenie przesłania ciasteczka musi się pojawić zanim wygenerowane zostanie ciało pakietu http (w tym przypadku dokument HTML). Po uruchomieniu skryptu odśwież kilkukrotnie stronę w przeglądarce.

```
<?php setcookie("last_visit",time(),time()+3600); ?>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>
  <?php
    if (isset($_COOKIE['last_visit']))
    {
      $last_visit = $_COOKIE['last_visit'];
      echo "Ostatnia wizyta :".date("d M Y H:i:s",$last_visit);
    }
    else
      echo "To Twoja pierwsza wizyta";
  ?>
</body>
</html>
```

19. Drugim popularnym mechanizmem wykorzystywanym w aplikacjach internetowych jest emulowana sesja http. Poniżej znajduje się prosta implementacja gry w „orzeł-reszka” wykorzystująca sesję do zapamiętania stanu gry. Zwróć uwagę na sposób inicjalizacji obiektów sesyjnych oraz sposób przenoszenia stanu gry pomiędzy kolejnymi żądaniami. Utwórz plik `file09.php` i umieść w nim poniższy kod.

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

<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>
  <?php
    class heads_tails
    {
      public $heads;
      public $tails;

      function heads_tails()
      {
        $this->heads = 0;
        $this->tails = 0;
      }

      function add_head() { $this->heads++; }
      function add_tail() { $this->tails++; }
    }
  ?>

  <form action="file09.php" method="post">
    <input type="submit" value="rzuć monetą"/>
  </form>

  <?php
    $result = rand() % 2;

    if (!isset($_SESSION['state']))
      $state = new heads_tails;
    else
      $state = $_SESSION['state'];

    switch($result)
    {
      case 0: $state->add_head(); break;
      case 1: $state->add_tail(); break;
    };

    $_SESSION['state'] = $state;

    echo "<h2>dotychczasowy wynik</h2>";
    echo "<ul>";
    echo "<li>orzełek: ".$state->heads."</li>";
    echo "<li>reszka: ".$state->tails."</li>";
    echo "</ul>";
  ?>

</body>
</html>
```

20. Ostatni przykład pokazuje proste wymuszenie zalogowania użytkownika oraz wskazuje, w jaki sposób szybko i wygodnie można osadzić w kodzie PHP statyczną zawartość. Utwórz plik `file10.php` i umieść w nim poniższy kod. Przetestuj działanie aplikacji.

```
<?php
  if ($_SERVER['PHP_AUTH_USER'] != "scott" ||
      $_SERVER['PHP_AUTH_PW'] != "tiger")
  {
    header('WWW-Authenticate: Basic realm="Logowanie"');
    header('HTTP/1.0 401 Unauthorized');
    echo "<h1>brak autoryzacji</h1>";
    exit();
  }

  print<<<KONIEC
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>

<body>
  <h1>Nie pieprz wieprza...</h1>

  <p>"Nie pieprz, Pietrze, pieprzem wieprza,
  Wtedy szynka będzie lepsza."

  <p>"Właśnie po to wieprza pieprzę,
  Żeby mięso było lepsze."

  <p>"Ależ będzie gorsze, Pietrze,
  Kiedy w wieprza pieprz się wetrze!"

</body>
</html>
KONIEC;
?>
```