# BIBLIOTEKA PEAR W TWORZENIU APLIKACJI INTERNETOWYCH

**Beata Pańczyk[1], Michał Duszyk[2]**
[1]Politechnika Lubelska, Wydział Elektrotechniki i Informatyki, Instytut Informatyki, [2]PayU S.A.

***Streszczenie.*** *Niniejszy artykuł prezentuje możliwości biblioteki PEAR w procesie szybkiego wytwarzania aplikacji internetowych w języku PHP. Na podstawie przykładów zostaną zaprezentowane zalety oraz korzyści płynące z użytkowania biblioteki PEAR. Stosowanie repozytoriów klas z niniejszej biblioteki pozwala znacznie uprościć i zoptymalizować kod w procesie programowania aplikacji PHP.*

**Słowa kluczowe**: aplikacje internetowe, PHP, repozytorium klas PEAR

## WEB APPLICATION DEVELOPMENT USING THE PEAR LIBRARY

***Abstract.*** *The paper presents the possibility of the PEAR library in the rapid production of the PHP applications. The advantages and benefits of the use of the PEAR are going to be presented. Use of this repository class library can significantly simplify and optimize the PHP code and accelerate the process of the web application programming.*

**Keywords**: web applications, PHP, PEAR classes repository

## Wstęp

W procesie wytwarzania aplikacji internetowych niezmiernie istotny jest czas poświęcony na ich budowę oraz możliwość wykorzystania sprawdzonych wzorców programistycznych. Progresja w programowaniu wiąże się z ciągłym tworzeniem narzędzi mających na celu szerokie zastosowanie w codziennym użytkowaniu, przy jednoczesnym skróceniu czasu pracy programisty. Zaczynając pisanie w języku PHP wielokrotnie można natrafić na problemy, które przeważnie rozwiązywane są poprzez tworzenie nowych funkcji od podstaw. Biblioteka PEAR umożliwia szybkie wdrożenie w aplikacjach sieciowych najczęściej wykorzystywanych funkcjonalności, dając gwarancję, że uzyskany kod będzie zoptymalizowany i bezpieczny.

PEAR (ang. PHP Extension and Application Repository) jest projektem zajmującym się katalogowaniem i udostępnianiem gotowych komponentów oraz modułów przeznaczonych dla PHP [9].

PEAR jest repozytorium dedykowanym dla programistów PHP, w którym tworzonych jest obecnie ponad 70% witryn internetowych. Źródeł drukowanych przeznaczonych wyłącznie dla tej tematyki nie jest zbyt wiele. Informacje o bibliotece można sporadycznie napotkać w książkach związanych bezpośrednio z językiem PHP [1, 4, 6].

## 1. Biblioteka PEAR

Projekt narodził się w 1999 roku dzięki Stigowi S. Bakkenowi i miał na celu ponowne wykorzystanie kodu wykonującego typowe operacje. Pomysł ten bardzo szybko znalazł wielu zwolenników, tworząc społeczność pracująca nad jego rozwojem. Przedsięwzięcie polegało na stworzeniu gotowej biblioteki kodu, utrzymaniu i rozwijaniu dystrybucji, zarządzaniu pakietami oraz promowaniu standardowych styli kodowania [10]. PEAR składa się z wielu małych komponentów, zawierających kod źródłowy lub binaria, specyficzne dla danego środowiska. Każdy z pakietów rozwijany jest przez grupę programistów jako osobny projekt, posiada ponadto własną dokumentację i zdefiniowane zależności między nią a innymi repozytoriami. Każde archiwum składa się z kodu źródłowego napisanego w PHP, zazwyczaj w stylu obiektowym.

Największą instytucją PEAR jest jej strona internetowa [9]. Witryna ta dostarcza dokumentację związana bezpośrednio z samym PEAR oraz z jego każdym komponentem. W serwisie funkcjonuje także system PEPr (PEar Proposal), w którym każdy ma możliwość zaprezentowania propozycji nowego repozytorium. System obsługuje pełen przepływ pracy (kod, przykłady, testy itp.) nad pakietem. Fora dyskusyjne są jednym z istotniejszych elementów instytucji PEAR.

Pakiety pogrupowane są w logiczne grupy (np. Authentication, File_System, Copy, Networking, XML), które pozwalają na łatwiejsze ich zestawienia. PEAR pozwala twórcom pakietów dowolnie rozwijać swoje projekty, przy zachowaniu pewnych zasad w celu zapewnienia jakości oraz łatwości dostępu dla użytkowników. Jedną z zasad jest projektowanie API swoich pakietów bez używania kodu osadzonego w innej części repozytorium PEAR. Pakiety PEAR mogą zależeć od innych pakietów PEAR, rozszerzeń oraz wersji PHP, ale nie mogą być zależne od zewnętrznego kodu PHP. Zależności zewnętrzne PHP muszą być zastąpione przez wewnętrzne. Zależności są pokazane na stronach internetowych pakietów i są również obsługiwane przez instalatora PEAR.

Poza ogólnymi zasadami PEAR definiuje także stałe reguły, które muszą być bezwzględnie stosowane przez programistów. Wymagane są standardy kodowe np. jasno określone nazewnictwo funkcji, zmiennych oraz konstruktorów.

Instalator PEAR pozwala na bardzo łatwe instalowanie, aktualizowanie oraz usuwanie pakietów. Instalator dostarczany jest (w wersji stabilnej) wraz z PHP od wersji 4.3.0. Jeśli wybrana dystrybucja PHP dostarczana jest bez instalatora PEAR lub zawiera nieaktualne wydanie, można dokonać instalacji posługując się instrukcją przedstawioną na stronie:
*http:// pear.php.net/manual/en/installation.getting.php.*

PEAR oferuje przede wszystkim:

- wysokiej jakości pakiety,
- elastyczne klasy wielokrotnego użytku,
- gotowe komponenty dla przedsiębiorstw (ang. enterprise-ready components),
- bezpieczeństwo,
- pomoc wspólnoty PEAR oraz bezpośredni kontakt z opiekunami pakietów,
- automatyczną instalację potrzebnych pakietów,
- łatwą i szybką aktualizację pakietów,
- wolne licencje, które mogą być używane bez opłat nawet w zastosowaniach komercyjnych.

## 2. Repozytoria PEAR

Najczęściej stosowane repozytoria klas PEAR to:

- HTML_QuickForm2,
- HTML_table,
- Text_CAPTCHA_Numeral,
- Image_QRCode,
- Validate_pl,
- HTML_menu,
- MDB2,
- i wiele innych.

### *HTML_QuickForm2*

Formularze są nieodłączną częścią niemal każdej strony WWW. Każdy serwis, w którym użytkownik dokonuje zakupów, lub który wymaga rejestracji, jest w nie wyposażony. Repozytorium *QuickForm2* służy do łatwego i szybkiego

tworzenia formularzy HTML. Wspomaga wiele rodzajów formularzy począwszy od okien logowania a skończywszy na polach tekstowych, notatkach, mailach, postach i innych elementach strony WWW. Obsługuje wszystkie moduły określone przez standard HTML w zakresie formularzy; zawiera także kilka niestandardowych, dodatkowych elementów. Jest to klasa definitywnie usprawniająca, a przede wszystkim przyspieszająca pracę programisty. Skraca czas poświęcony budowie formularzy, ale też ułatwia możliwość ich późniejszej rozbudowy oraz modyfikacji. Posiada wiele przydatnych opcji jak np. opcja *addRule*, która automatycznie dodaje znak ”*” sugerujący, iż miejsce przy którym się on znajduje obowiązkowo musi zostać wypełnione. Ponadto opcja ta automatycznie dodaje informację tekstową o konieczności wypełnienia danego pola.

W *QuickForm2* główną rolę odgrywa funkcja *addElement*, tworząca elementy takie jak pola tekstowe bądź ich opis. *QuickForm2* umożliwia również m.in. tworzenie przycisków typu *radio* i *checkbox*. Przy budowie serwisu internetowego, repozytorium *QuickForm2* można zastosować do budowy okna loginu, hasła oraz przycisku logowania. Przykładowy kod z wykorzystaniem tej klasy pokazano na listingu 1 [3].

**Listing 1. Zastosowanie HTML_QuickForm2 w procesie logowania**

```
<?php
function logowanie_pear() {
if ($_SESSION['uprawnienia']== 0) {

      // utworzenie obiektu za pomocą konstruktora HTML_QuickForm2
      $form = new  HTML_QuickForm2('tutorial','post',
            'action="../kontroler/funkcje.php"', FALSE);

      //zbiór pól formularza:
      $fieldset = $form->addElement('fieldset')->setLabel('Logowanie');

      //pakiet oferuje ograniczenie ciągu znaków w formularzu oraz ustawienie
      //długości formularza
      $name = $fieldset->addElement('text', 'login', array('size' => 36,
            'maxlength' => 255)) ->setLabel('Login');raniczenia znaków
            $haslo = $fieldset->addElement('password', 'haslo', array('size' => 36,
            'maxlength' => 255)) ->setLabel('Hasło');
      $fieldset->addElement('hidden','logInForm');
      $fieldset->addElement('submit', null, array('value' => 'Zaloguj'));

      echo '<input type="submit" name="dodajUzytkownika"
            onClick="show_me(1)" value="dodajUzytkownika" class="center">';
      echo "$form";}
else{
   $form=new HTML_QuickForm2('tutorial','post','action="../kontroler/funkcje.php"',
            FALSE);
      $fieldset = $form->addElement('fieldset')->setLabel('Wylogowywanie');
      $fieldset->addElement('submit', 'wylogowanie', array('value' => 'Wyloguj'));
      echo  "$form";}
}
?>
```

Klasa *Quickform2* automatycznie generuje HTML potrzebny do budowania obiektów formularza i dodawania do niego pól.

### *HTML_table*

Repozytorium *Html_table* to alternatywa dla serwisów, w których obszar tabel będzie z biegiem czasu się zwiększał a ich zawartość ma posiadać ściśle określony styl. Pakiet ten oferuje szybkie dodawanie wierszy oraz kolumny. Dodatkowo umożliwia ich uzupełnienie, poprzez pobieranie danych do tabeli bezpośrednio z bazy danych. Na listingu 2 przedstawiono wykorzystanie PEAR do budowy tabeli [3].

**Listing 2. Zastosowanie HTML_table**

```
<?php
 //dołączenie pliku z definicją klasy HTML_Table
 require_once 'HTML/Table.php';

 $data = array(
 '0' => array('Artur', 'Kowalski', 'www.wp.pl', 'a.kowal@wp.pl'),
 '1' => array('Kurt', 'Cobain', 'kurt.cobain.com', 'k.cobain@gmail.com'),
 '2' => array('Krzysztof', 'Marcin', '', ''),
 '3' => array ('Michal', 'Duszyk', '', 'michal.duszyk@gmail.com'),
 '4' => array ('Janusz','Wojcik', 'o2.pl','jw@wp.pl')
 );

 $attrs = array('width' => '600');
 $table = new HTML_Table($attrs);

 //auto powiekszanie tabeli
 $table->setAutoGrow(true);

 //wartosc ktora ma byc wstawiona w każdą pustą komórkę
 $table->setAutoFill('n/a');

 for ($nr = 0; $nr < count($data); $nr++) {
    //zawartosc nagłówka
   $table->setHeaderContents($nr+1, 0, (string)$nr);
   for ($i = 0; $i < 4; $i++) {
     if ('' != $data[$nr][$i]) {
        //zmienia zawartość komórki dla istniejącej komórki
       $table->setCellContents($nr+1, $i+1, $data[$nr][$i]);
     }
   }
 }

 $altRow = array('bgcolor' => '#FF0000');
 $table->spoko(1, null, $altRow);
 $table->setHeaderContents(0, 0, '');
 $table->setHeaderContents(0, 1, 'Imie');
 $table->setHeaderContents(0, 2, 'Nazwisko');
 $table->setHeaderContents(0, 3, 'Strona WWW');
 $table->setHeaderContents(0, 4, 'E-Mail');
 $hrAttrs = array('bgcolor' => '#808080');
 $table->setRowAttributes(0, $hrAttrs, true);
 $table->setColAttributes(0, $hrAttrs);

 echo $table->toHtml();
 ?>
```

### *Text_CAPTCHA_Numeral*

CAPTCHA (ang. Completely Automated Public Turing test to tell Computers and Humans Apart) jest bardzo powszechną techniką zabezpieczającą strony WWW przed dostępem botów. Tylko człowiek jest w stanie przejść test captcha. Technika ta chroni formularze na stronach WWW przed spamem, fora dyskusyjne i portale przed rejestracją automatu, blogi przed niechcianymi reklamami itp. Technologia ta zazwyczaj polega na rozmazywaniu tekstu bądź zadawaniu pytań matematycznych. System, po przedstawieniu zadania, oczekuje odpowiedzi, którą następnie weryfikuje. W przypadku podania poprawnej odpowiedzi, użytkownikowi zostają udostępnione odpowiednie prawa.

Repozytorium PEAR dostarcza programistom gotowe rozwiązania związane z technologią CAPTCHA. Pakiet oferuje wersję zarówno obrazkową jak i numeryczną. Listing 3 przedstawia przykład korzystania z wersji numerycznej [3]. W dobie szeroko stosowanych zabezpieczeń w serwisach WWW, rozwiązanie oferowane przez PEAR jest dobrą propozycją dla każdego programisty. Jest proste do instalacji oraz integracji, kod jest czytelny, przejrzysty, łatwy w modyfikacji.

**Listing 3. Zastosowanie kodu Captcha w wersji numerycznej**

```
<?php
function captcha() {
   $numcap = new
   Text_CAPTCHA_Numeral(Text_CAPTCHA_Numeral::TEXT_CAPTCHA_NUME
   RAL_COMPLEXITY_UNIVERSITY);
   //Text_CAPTCHA_Numeral::
   //TEXT_CAPTCHA_NUMERAL_COMPLEXITY_ELEMENTARY
   //Text_CAPTCHA_Numeral::
   //TEXT_CAPTCHA_NUMERAL_COMPLEXITY_HIGH_SCHOOL
   //Text_CAPTCHA_Numeral::
   //TEXT_CAPTCHA_NUMERAL_COMPLEXITY_UNIVERSITY

   if (isset($_POST['captcha']) && isset($_SESSION['a'])) {
      if ($_POST['captcha'] == $_SESSION['a']) {
         $errors[] = 'Ok... You might be human...';
      } else {
         $errors[] = 'You are dumb or not human';
```

```
    };

    if (!empty($errors)) {
      foreach ($errors as $error) {
        print "<h1><font color='red'>$error</font></h1><br />";
      }
    }
  }

  print '
    <form name="capter" action="index.php?page=liveExample" method="post">
     <table>
     <tr>
      <th>Jaki jest prawidlowy wynik?: ' . $numcap->getOperation() . '</th>
      <td><input type="text" value="" name="captcha" /></td>
     </tr>
     <tr>
      <th/>
      <td><input type="submit" value="udowodnij ze jestes czlowiekiem!" /></td>
     </tr></table>
    </form>';
 $_SESSION['a'] = $numcap->getAnswer();
  print $numcap->getAnswer();
?>
```

Rys. 1 przedstawia efekt działania kodu z listingu 3 a rys. 2 prezentuje przykładowy wynik działania w wersji graficznej.


*Rys. 1. Przykład Captcha – wersja numeryczna [1]*


*Rys. 2. Przykład Captcha - wersja graficzna [11]*

### *Validate_pl*

Wprowadzając dane do formularzy, użytkownik może w pełni świadomie, bądź nie, popełnić błąd. Jeśli informacje odebrane od użytkownika poddane zostaną przetwarzaniu bez weryfikacji, wówczas, w zależności od odporności aplikacji, jej administrator może mieć do czynienia z różnymi rodzajami błędów, od drukowania w przeglądarce klienta komunikatów diagnostycznych PHP czy MySQL, poprzez utratę spójności bazy danych [6], aż po ujawnienie niepowołanym użytkownikom informacji poufnych. Z tego powodu niezbędna jest walidacja wszystkich danych wprowadzanych przez użytkownika. Jest to proces bardzo żmudny i pracochłonny dlatego warto poszukać rozwiązań wspomagających sprawdzanie danych.

Portale przeznaczone do sprzedaży wysyłkowej, zarządzające finansami, generujące dokumenty takie jak faktury, muszą posiadać narzędzie sprawdzające poprawność danych. Standardowymi, dość łatwo sprawdzalnymi, są takie elementy jak: poprawność adresu mail, imię, numer ulicy, kod miasta itp. Problem pojawia się, gdy system ma sprawdzić Pesel, NIP czy REGON, które stanowią unikalne ciągi cyfr generowane za pomocą specjalnych algorytmów. Biblioteka PEAR oferuje gotowe do zaimplementowania rozwiązania w zakresie tego typu walidacji. Listing 4 przedstawia wykorzystanie komponentu *Validate_PL*, tym niemniej dostępna jest także walidacja ukierunkowana na inne języki [3].

Walidacja dostarczana przez PEAR jest doskonałym rozwiązaniem dla producentów witryn np. o charakterze handlowym czy finansowym. Podczas wprowadzania danych składających się z samych cyfr bardzo łatwo o pomyłkę. Taka walidacja jest nieodzowna jako zabezpieczenie przed ewentualnymi atakami botów podejmujących próby rejestracji czy też tworzenia kont.

**Listing 4. Funkcje do walidacji danych**

```
<?php
require_once 'Validate/PL.php';

function testNIP($NIP) {
   $numer_nip = new Validate_PL();
   return $numer_nip->nip($NIP);
}

function testREGON($REGON) {
   $numer_regon = new Validate_PL();
   return $numer_regon->regon($REGON);
}

function testPESEL($PESEL) {
  $numer_pesel = new Validate_PL();
   return $numer_pesel->pesel($PESEL, $dsfs);
}
?>
```

*Validate_PL* jest bardzo dobrym narzędziem sprawdzania poprawności danych. Jest to pakiet bardzo dokładny i łatwy w użyciu. Implementacja repozytorium jest prosta, pozwalając nawet początkującym programistom na jej zastosowanie. Główną zaletą przemawiającą za zastosowaniem tego pakietu jest koszt pracy nad implementacją własnych algorytmów.

### *Image_QRCode*

QR Code (ang. Quick Response), czyli szybka odpowiedź kodowa jest to dwuwymiarowy, alfanumeryczny, kwadratowy, matrycowy kod, początkowo stworzony z myślą o przemyśle samochodowym. Całkiem niedawno system stał się wyjątkowo popularny ze względu na swoją czytelność oraz dużą pojemność w porównaniu do standardowego kodu kreskowego. Kod składa się z czarnych modułów ułożonych w kwadracie na białym tle. Zakodowane informacje mogą składać się z różnych typów danych (np. dane numeryczne, alfanumeryczne, binarne). QR Code jest jednym z najbardziej popularnych typów kodów dwuwymiarowych, został zaprojektowany tak, aby jego zawartość możliwa była do odczytywania ze znacznie większą prędkością niż klasyczny kod kreskowy.

W ostatnich latach system stał się powszechny w reklamach i na opakowaniach. Temu stanowi rzeczy przysłużyła się popularność smartfonów, dzięki którym każdy posiada „czytnik kodów kreskowych" we własnej kieszeni.

QR kody mogą być używane w mobilnych systemach operacyjnych (np. Android, iOS, Windows Phone). Biblioteka PEAR w zakresie kodów QR udostępnia klasę, której zastosowanie prezentuje listing 5 [3].

**Listing 5. Kod QR_Code**

```
<?php
require_once 'Image/QRCode.php';
$qrcode = new Image_QRCode();
$qrcode->makeCode('test', array(
   'image_type' => 'png',
   'error_correct' => 'H', <br>
   ));
//mozna zastosowac takze rozne rozszerzenia jak jpg lub png
?>
```

Metoda *makeCode()* służy do tworzenia kodu QR. W metodzie poza informacją/adresem, które mają zostać zakodowane mogą zostać wykorzystane także atrybuty:

- *image_type* - określa format obrazu,
- *output_type* - ustawia co zrobić gdy kod został wygenerowany - 'display' or 'return'),
- *error_correct* - ustawia poziom korekcji błędów (L: 7%, M: 15%, P: 25%, H: 30%),
- *module_size* - ustawia domyślny rozmiar modułu (PNG: 4, JPEG: 8).

W listingu 5 w miejsce łańcucha 'test' użytkownik może wprowadzić dowolny ciąg znaków. Może to być adres strony, opis, numery itd. Rys. 3 przedstawia przykład kodu QR wygenerowany dla strony uczelni.

*Rys. 3. QrCode dla strony http://www.pollub.pl/pl/uczelnia/organizacja-uczelni/wydzialy/wydzial-elektrotechniki-i-informatyki [3]*

Qr Code jest nowoczesnym i niezbędnym narzędziem każdego serwisu. Podczas wpisywania adresu WWW użytkownik traci ponad 50% czasu, a Qr Code gwarantuje bezpośrednie poprawne przeniesienie na oczekiwaną stronę. Z czasem każdy serwis zobowiązany będzie stosować tego typu kody.

### *HTML_Menu*

Podstawą funkcjonowania serwisów internetowych jest jasna i intuicyjna nawigacja, oparta na odpowiednim elemencie menu. Każdy twórca strony webowej zmuszony jest dostosować do niej taki element menu, zwykle za pomocą języka HTML, CSS, JavaScript, PHP itp. We wszystkich dostępnych technologiach każda zakładka tworzona musi być od początku. Programista musi rozbudowywać kod służący do spisu zawartości serwisu, tym samym jego czytelność oraz przejrzystość się zmniejsza. Aby zagwarantować stały układ, styl, kolory, wielkości przycisków i inne wizualne elementy menu trzeba zastosować zwykle dobrze zaprojektowany arkusz CSS. Istnieje wiele możliwości stworzenia spisu strony, ale każda wymaga czasu i znajomości dostępnych technologii. Jednym z rozwiązań może być pakiet *HTML_menu*, który pozwala budować nawigację na stronie w oparciu o układy menu pokazane na rys. 4.

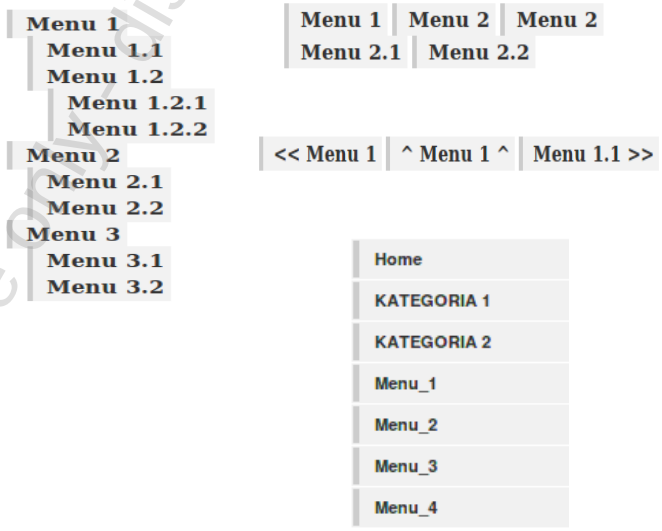
*HTML_menu* oferuje pięć grup ustawień (rys.4):

- rows,
- tree,
- urhere,
- prevnext,
- sitemap.

Rozwiązanie nie nakłada żadnych kosztów czasowych związanych z tworzeniem kodu menu. Sposób tworzenia menu prezentuje listing 6. System działania repozytorium opiera się głównie na tabelach *array*, w których programista podaje nazwę – *title* oraz adres URL strony, do której dany stopień menu ma przenieść użytkownika. W celu utworzenia podkategorii menu wystarczy zagnieździć identyczną tabelę w tabeli wyższego rzędu, podając kolejne dane takie jak *nr* tabeli np. *21=> array...* . Zagnieżdżanie podkategorii nie posiada ograniczeń ilościowych. Dzięki *HTML_menu*, można bardzo sprawnie rozbudowywać istniejące kategorie oraz dodawać nowe podkategorie.

**Listing 6. Przykład zastosowania HTML_Menu**

```
<?php
require_once 'HTML/Menu.php';
$data=array(
  1 => array(
    'title' => 'Menu 1',
    'url' => '/SKLEP/html_menu.php',
    'sub' => array(
      11 => array('title' => 'Menu 1.1', 'url' => '/SKLEP/nip.php'),
      12 => array(
        'title' => 'Menu 1.2',
        'url' => '/SKLEP/capchta.php',
        'sub' => array(
          121 => array('title' => 'Menu 1.2.1', '
                url' => '/SKLEP/widok/index.php?p=kategoria&kategoria=5'),
          122 => array('title' => 'Menu 1.2.2', '
                url' => '/SKLEP/widok/index.php?p=kategoria&kategoria=3')
        )
      )
    )
  ),
  2 => array(
    'title' => 'Menu 2',
    'url' => '/SKLEP/html_menu.php',
    'sub' => array(
      21 => array('title' => 'Menu 2.1',
            'url' => '/SKLEP/widok/index.php?p=nowyTowar'),
      22 => array('title' => 'Menu 2.2',
            'url' => '/SKLEP/widok/index.php?p=nowaKategoria')
    )
  ),
  3 => array(
    'title' => 'Menu 3',
    'url' => '/SKLEP/html_menu.php',
    'sub' => array(
      31 => array('title' => 'Menu 3.1',
            'url' => '/SKLEP/widok/index.php?p=edycjaKategoria'),
      32 => array('title' => 'Menu 3.2',
            'url' => '/SKLEP/widok/index.php?p=kategoria&kategoria=1')
    )
  )
);
// $data określa strukturę menu: rows, tree, urhere, prevnext, sitemap
$menu =& new HTML_Menu($data, 'urhere');
$menu->show();  ?>
```




*Rys. 4. Przykładowe układy w HTML_Menu [1]*

### *MDB2*

W przeszłości przechowywanie informacji w pliku tekstowym lub w prostej bazie danych było wystarczającym rozwiązaniem. Obecnie każdy programista aplikacji musi posiadać wiedzę o tym, w jaki sposób przechowywać informacje w bazach danych. Od najwcześniejszych wersji PHP, programiści zawsze mogli liczyć na silne wsparcie dla baz danych. Jednak, aż do najnowszej wersji PDO (ang. PHP Data Objects), nie było standardowego sposobu posługiwania się różnymi sterownikami baz danych dołączanych do PHP [3]. Brak jednolitego API zrodziło potrzebę stworzenia warstwy abstrakcji bazy danych (DBAL – ang. DataBase Abstraction Layer). Jednym z oferowanych rozwiązań jest PEAR: MDB2 [9].

Na tworzenie warstw abstrakcji należy spojrzeć z trzech perspektyw:

- abstrakcji dla interfejsu bazy danych,
- abstrakcji dla kodu SQL,
- abstrakcji dla typów danych.

Abstrakcja interfejsu bazy danych jest niezwykle istotna, pozwala programiście na dostęp do każdej bazy danych przy użyciu tych samych metod wywołania. Oznacza to, że połączenie z instancją bazy danych, wysyłanie i pobieranie danych, będzie identyczne, niezależnie od rodzaju używanej bazy danych.

Większość nowoczesnych baz danych obsługuje standard SQL, tak więc znacząca część SQL, będzie działać niezależnie od tego, jaka w ostateczności baza będzie używana. Jednak wiele baz wprowadziło specyficzne funkcje SQL, co zwiększa prawdopodobieństwo, iż dany kod zadziała na jednej bazie, na innej już nie. Podczas progresji RDBMS (ang. Relational DataBase Management System) mogą być implementowane funkcje niekompatybilne ze starszymi wersjami tej samej bazy danych. Chcąc napisać SQL zgodny ze wszystkimi wersjami konkretnej bazy danych, najlepszym rozwiązaniem jest użycie warstwy abstrakcji. Chociaż nie ma sposobu, aby zebrać wszystkie możliwe funkcje SQL, MDB2 zapewnia wsparcie dla najbardziej popularnych. Korzystanie z mechanizmów abstrakcji kodu SQL, dostarczanego przez MDB2, daje gwarancję, że użytkownik będzie mógł posługiwać się zaawanasowanymi funkcjami nawet, gdy baza danych nie będzie ich obsługiwała.

Kolejnym rodzajem abstrakcji jest abstrakcja typów danych [8]. Zapotrzebowanie na tego rodzaju abstrakcje wynika z faktu, iż różne bazy danych obsługują różne typy danych. Przy zastosowaniu warstwy abstrakcji bazy danych, w wielu przypadkach efektywność działania jest mniejsza. Nie jest to specyficzne dla MDB2 czy innych warstw abstrakcji baz danych, ale dla warstw abstrakcji i systemów wirtualizacji oprogramowania w ogóle. W przeciwieństwie do VMWare lub Microsoft Virtual PC, których abstrakcyjne są wszystkie systemowe wywołania, MBD2 dostarcza abstrakcję tylko w ostateczności, gdy dana funkcja jest niedostępna. Oznacza to, że wydajność w głównej mierze zależy od platformy, na której używany jest MDB2.

Projekt API MDB2 został stworzony, w celu zapewnienia maksymalnej elastyczności. Modularne podejście było stosowane w obsłudze baz danych i konkretnych zaawansowanych funkcjonalności. Każdy sterownik bazy danych jest spakowany i utrzymywany jako samodzielny moduł PEAR.

Drugi typ modułowości wbudowany w MDB2 służy do dodawania rozszerzonych funkcji MDB2. Zamiast umieszczać te funkcje bezpośrednio w MDB2 lub dodawać nową klasę implementującą te funkcje, użytkownik ma możliwość utworzenia osobnej klasy w nowym module, którą następnie wczytuje za pomocą metody *loadModule()* do rdzenia MDB2. Kiedy moduł zostanie już załadowany do MDB2, będzie można uzyskać dostęp do jego metod, jak gdyby były wbudowane w pakiet MDB2. Działanie to jest podyktowane szybkością pracy jego wewnętrznych metod, jednocześnie umożliwiając użytkownikom swobodne dołączanie własnych klas do pakietu MDB2.

Gdy użytkownik zdecyduje się na użycie pakietu MDB2 musi pamiętać, że nie zawiera on żadnych sterowników baz danych. Sterowniki te należy zainstalować oddzielnie. Standardowo instalowane są tylko klasy tworzące rdzeń MDB2.

Instalacji sterowników odpowiedniej bazy danych dokonać można np. za pomocą polecenia:

- dla MySql: Pear install MDB2_driver_mysql,
- dla SQLite: Pear install MDB2_driver_sqlite

Pełną listę aktualnych sterowników można znaleźć w [18].

Przykład kodu aplikacji pobierającej z bazy danych informacje zaprezentowany został na listingu 7 [3].

**Listing 7. Kod warstwy abstrakcji MDB2**

```
<?
require_once( 'MDB2.php' );
...
//funkcja pobierająca dane o towarze
function NowyTowar() {
   $db = MDB2::connect("mysql://sklep@127.0.0.1/sklep");
   if (PEAR::isError($db)) {
      die($db->getMessage());
   }
   $nazwa = $_POST["nazwa"];
   $producent = $_POST["producent"];
$opis = $_POST["opis"];
$parametry = $_POST["parametry"];
$cena = $_POST["cena"];
$VAT = $_POST["vat"];
$serwis = $_POST["serwis"];
$gwarancja = $_POST["gwarancja"];
$dostepnosc = $_POST["dostepnosc"];
$kategoria = $_POST["kategorie"];

$query = "insert into Towar
(nazwa, producent, opis, parametry, cena, vat, serwis, gwarancja,
  dostepnosc, id_promocja, ilosc_sprzedanych, id_kategoria )
values( '$nazwa', '$producent', '$opis', '$parametry', '$cena', '$VAT', '$serwis',
 '$gwarancja', '$dostepnosc', 0, 0, $kategoria );";

$res = $db->exec($query) or $error .= "ERROR SELECT $query<br>";
if (PEAR::isError($res)) {
   die($res->getMessage() . "<br>$query");
}
```

Zapytanie w MDB2 można wykonać za pomocą metod:

- *query*, która zwraca wynik zapytania,
- *exec*, która zwraca liczbę zmienionych przez zapytanie wierszy.

Przykład wykorzystania metody *query* przedstawia listing 8 [3].

**Listing 8. Przykładowy kod z zastosowaniem metody query**

```
<?php
//demo
$sql='select from grupy';
$demoResult=$db->query($sql);
While ($demoRow=$demoResult->fetchRow()) {
echo $demoRow[2] . '<br>';
}
$db -> disconnect();
echo "testTESTtest";
?>
```

Zmienna *$result* jest obiektem typu *MDB2_Result* a także klasą zależną od konkretnego sterownika bazy danych. Do przeglądania zbioru wyników wykorzystano metodę *fetchRow()*, która pobiera pojedyncze wiersze.

Poza metodą *fetchRow()* można wykorzystać jeszcze kilka innych metod z grupy *fetch*():*

- *fetchAll()* zwraca tablicę zawierającą wszystkie rekordy,
- *fetchOne()* bez żadnych parametrów zwraca wartość pierwszego pola z bieżącego wiersza. Jeśli zostaną przesłane jej odpowiednie parametry, za jej pomocą będzie można pobrać dowolne pole z dowolnego wiersza np. *fetchOne (1,2)* zwróci drugą kolumnę trzeciego wiersza w tabeli,
- *fetchCol($column)* zwróci pola kolumny o numerze *$column* dla wszystkich wierszy lub pierwszą kolumnę, w przypadku nieokreślenia parametru *$column*.

Znacznie łatwiej jest pobierać dane z tablicy asocjacyjnej. Repozytorium MBD2 posiada dwa zestawy metod pobierania danych za pomocą klucza: *query** oraz *get**.

Różne systemy baz danych obsługują różne typy danych. Pakiet MDB2 omija tego typu problemy poprzez stosowanie własnych, uniwersalnych zestawów typów danych. Programista może z tych typów korzystać i pozwolić, aby to pakiet *MDB2* zadbał o przenośność typów danych między systemami RDBMS, poprzez mapowanie swoich typów na typy odpowiedniego systemu baz danych.

Typy danych oferowane przez pakiet MDB2 to: *text*, *decimal*, *boolean*, *integer*, *float*, *blob*, *clob*, *date*, *time*, *timestamp* [19]. We wszystkich metodach z grupy *query(), fetch()* i *get()* można określać typ zbioru wyników, który użytkownik chce otrzymać. Pakiet MDB2 automatycznie konwertuje wartości na odpowiedni typ danych.

## 3. Wnioski

Dzięki bibliotece PEAR pisanie aplikacji PHP jest znacznie szybsze i skuteczniejsze. Ze względu na wielkość biblioteki, w artykule zaprezentowano wyłącznie jej podstawowe pakiety.

Przy prezentacji pakietów analizowana była przede wszystkim złożoność repozytorium oraz koszty czasowe nałożone na implementację konkretnej funkcjonalności przy użyciu powszechnie stosowanych technologii.

Klasy PEAR spełniają oczekiwania programistów, oferując zoptymalizowane pod kątem kodu i bezpieczeństwa gotowe rozwiązania najczęściej występujących problemów pojawiających się w trakcie tworzenia aplikacji internetowych w języku PHP.

Omawiane pakiety PEAR są łatwe w instalacji, a przez określoną strukturę i przejrzystość kodu, można łatwo i szybko przyswoić sobie ich budowę i zastosowania. PEAR jest bardzo elastyczny – oferuje nieskomplikowaną integrację z najbardziej popularnymi technologiami. Osoby początkujące, lecz posiadające doświadczenie w programowaniu, bez żadnych trudności wykorzystają zalety PEAR we własnych projektach

Korzystanie z omówionej biblioteki dodatkowo gwarantuje poprawność zastosowanego w niej kodu. Programista nie musi obawiać się konfliktów między funkcjami aplikacji, bądź poświęcać dodatkowy czas na wyszukiwanie ewentualnych błędów.

Niezaprzeczalnie ogromną zaletą PEAR jest silna i prężnie działającą społeczność, z pomocy oraz porad której może skorzystać każdy zainteresowany wykorzystaniem któregoś z pakietów. PEAR jest doskonale udokumentowany, dzięki czemu można bardzo łatwo zrozumieć zasadę jego działania i poznać zarówno podstawowe jak i bardziej zaawansowane pakiety.

Reasumując biblioteka PEAR jest idealnym rozwiązaniem do szybkiego i w pełni funkcjonalnego tworzenia systemów webowych. Gwarantuje pewny kod, wsparcie, bezpłatne użytkowanie, prostą instalację, pełną dokumentację oraz ogromny wybór spośród swoich repozytoriów.

## Literatura


[1] Ballad B., Valade J.: PHP & MySQL Web Development All-in-One Desk Reference For Dummies, United States of America 2008.
[2] Converse T, Park J., Suehring S.: PHP6 and MySQL Bible, United States of America 2009.
[3] Duszyk M.: Zastosowanie biblioteki PEAR w tworzeniu aplikacji internetowych, praca inżynierska, Politechnika Lubelska, 2012
[4] Holzner S.: PHP: The Complete Reference, United States of America 2007.
[5] Hossain Tonu M. A.: PHP Application Development with NetBeans Beginner's Guide, Birmingham 2012.
[6] MacIntyre P.: PHP: The Good Parts, United States of America 2010.
[7] Thomson L., Welling L.: PHP i MySQL. Tworzenie stron WWW. Vademecum profesjonalisty. Wydanie trzecie, Wydawnictwo Helion, Gliwice 2005.
[8] Vela Nava E. A., Heiderich M., Heyes G., Lindsay D.: Web Application Obfuscation, United States of America 2011.
[9] pear.php.net
[10] pear.11abacus.com/dev/doc/HTML/QuickForm/examples/
[11] www.captcha.net/



**Dr Beata Pańczyk**
e-mail: b.panczyk@pollub.pl


Ukończyła studia matematyczne na UMCS w Lublinie. W latach 1989-2011 pracownik naukowy (asystent, adiunkt) w Instytucie Informatyki Politechniki Lubelskiej. Tytuł doktora uzyskała w roku 1996 na Wydziale Elektrycznym PL. Temat rozprawy doktorskiej: Konstrukcja obrazu rozkładu właściwości fizycznych obiektu metodą Impedancyjnej Tomografii Komputerowej. Od roku 2011 na stanowisku starszego wykładowcy.
Obszar zainteresowań dydaktycznych i naukowych to metody numeryczne i języki programowania.


**Inż. Michał Duszyk**
e-mail: michal.duszyk@gmail.com


Jest absolwentem kierunku Informatyka Politechniki Lubelskiej. Obecnie pracownik zespołu ds. integracji systemu płatnościowego PayU z aplikacjami partnerów biznesowych spółki. Wykonywane obowiązki są ściśle związane z pracą opartą na technologii PHP i MySQL. Obszary pozostające w sferze zainteresowań to przede wszystkim Business Inteligence, e-commerce, bankowość.

*otrzymano/received: 13.05.2013* *przyjęto do druku/accepted: 12.11.2013*

# LISTA RECENZENTÓW ARTYKUŁÓW OPUBLIKOWANYCH W ROKU 2013

- prof. Javier Ballester - University of Zaragoza (Hiszpania)
- dr inż. Dariusz Bober - Uniwersytet Rzeszowski
- prof. Marian Cholewa - Politechnika Rzeszowska
- prof. Antonii Cieśla - Akademia Górniczo-Hutnicza w Krakowie
- dr inż. Bogusław Dołęga - Politechnika Rzeszowska
- prof. Zbigniew Fedyczak - Uniwersytet Zielonogórski
- prof. Stefan Filipowicz - Politechnika Warszawska
- prof. Andrzej Garbacz - Politechnika Warszawska
- dr inż. Ryszard Goleman - Politechnika Lubelska
- prof. Andrzej Gontarz - Politechnika Lubelska
- prof. Marek Gotfryd - Politechnika Rzeszowska
- prof. Stanisław Gratkowski - Zachodniopomorski Uniwersytet Technologiczny w Szczecinie
- dr inż. Konrad Gromaszek - Politechnika Lubelska
- prof. Oleksandra Hotra - Politechnika Lubelska
- prof. Zenon Hotra - Lviv Polytechnic National University (Ukraina)
- dr inż. Paweł Jabłoński - Politechnika Częstochowska
- prof. Elżbieta Jartych - Politechnika Lubelska
- prof. Wojciech Jarzyna - Politechnika Lubelska
- dr inż. Dariusz Klepacki - Politechnika Rzeszowska
- prof. Jacek Kluska - Politechnika Rzeszowska
- dr inż. Tomasz Kołtunowicz - Politechnika Lubelska
- prof. Volodymyr Kovanko - National University of Water Management and Nature Resources Use (Ukraina)
- prof. Andrzej Kotyra - Politechnika Lubelska
- prof. Sławomir Kozak - Instytut Elektrotechniki
- dr inż. Michał Kruk - Szkoła Główna Gospodarstwa Wiejskiego w Warszawie
- prof. Roman Kubacki - Wojskowa Akademia Techniczna
- prof. Wiesława Kuniszyk-Józkowiak - Uniwersytet Marii Curie-Skłodowskiej w Lublinie
- prof. Volodymyr Lytvynenko - Kherson National Technical University (Ukraina)
- prof. Ewa Majchrzak - Politechnika Śląska
- prof. Tomasz Markiewicz - Politechnika Warszawska
- dr inż. Arkadiusz Miaskowski - Uniwersytet Przyrodniczy w Lublinie
- prof. Bogdan Miedziński - Politechnika Wrocławska
- prof. Andrzej Nafalski - University of South Australia (Australia)
- prof. Andrzej Napieralski - Politechnika Łódzka
- prof. Ryszard Nawrowski - Politechnika Poznańska
- prof. Marian Noga - Akademia Górniczo-Hutnicza w Krakowie
- prof. Antoni Nowakowski - Politechnika Gdańska
- prof. Witold Pawelski - Politechnika Łódzka
- prof. Krystyn Bożydar Pawluk - Instytut Elektrotechniki
- prof. Zygmunt Piątek - Politechnika Częstochowska
- dr n. med. Sebeastian Radej - Uniwersytet Medyczny w Lublinie
- prof. Dominik Sankowski - Politechnika Łódzka
- prof. Bartosz Sawicki - Politechnika Warszawska
- dr inż. Monika Siewczyńska - Politechnika Poznańska
- prof. Jarosław Sikora - Politechnika Lubelska
- dr Aleksander Sokołowski - Politechnika Rzeszowska
- dr inż. Ernest Staroń - Państwowa Agencja Atomistyki
- prof. Adam Szelag - Politechnika Warszawska
- prof. Elżbieta Szychta - Uniwersytet Technologiczno-Humanistyczny im. Kazimierza Pułaskiego w Radomiu
- dr inż. Piotr Świszcz - Politechnika Śląska
- prof. Andrzej Wac-Włodarczyk - Politechnika Lubelska
- dr inż. Radosław Wajman - Politechnika Łódzka
- dr inż. Agnieszka Wantuch - Akademia Górniczo-Hutnicza w Krakowie
- dr inż. Paweł Wierzba - Politechnika Gdańska
- prof. Krzysztof Zaremba - Politechnika Białostocka
- prof. Krzysztof Zymmer - Instytut Elektrotechniki w Warszawie