Pagini WEB cu PHP 4

Mihai Scorțaru, Claudiu Soroiu

Limbajul PHP oferă realizatorilor de pagini web dinamice o multitudine de clase și funcții. În acest articol vom prezenta felul în care, într-un script scris în PHP 4, se pot utiliza clase Java și felul în care se pot realiza imagini dinamic.

Interoperabilitate PHP/Java

Un aspect important al limbajului *PHP* este acela că realizatorii de pagini *web* pot folosi în *script*-uri o mare parte a facilităților oferite de limbajul *Java* prin intermediul unei biblioteci de funcții numită ***php_java***.

Această facilitate îmbunătățește foarte mult limbajul *PHP*, aducându-i acestuia o foarte mare flexibilitate.

În continuare vom prezenta modul în care se pot instanția și utiliza clase *Java* în cadrul *script*-urilor *PHP*.

Cerințe

Pentru a utiliza facilitățile oferite de limbajul *Java* în cadrul *script*-urilor *PHP* este necesară o platformă *Java* care poate fi găsită pe site-ul **http://java.sun.com** și pachetul *php_java* care conține două fișiere: biblioteca ***php_java.dll*** (pe platforme *Windows*) și pachetul ***php_java.jar*** care conține o clasă care se ocupă de încărcarea celor folosite în *script*-urile *PHP*. Această clasă este utilizată de către *PHP* pentru a executa funcții din clase *Java* și pentru a instanția clase noi.

Limbajul *PHP* poate utiliza cel puțin următoarele versiuni de *Java*: 1.2.x, 1.3.x, 1.4.2. Noi am testat cu *Java* 1.4.2.

Cele două fișiere ale pachetului *php_java* pot fi găsite în versiunea integrală a limbajului *PHP*.

Instalare

Pentru a utiliza această facilitate trebuie instalată platforma *Java* și cele două fișiere ale pachetului *php_java* trebuie copiate în directorul *extensions* care poate fi găsit în directorul în care a fost instalat limbajul *PHP*.

Configurare

Pentru a putea accesa clase *Java* în cadrul *script*-urilor *PHP* trebuie configurată biblioteca *php_java*.

Configurarea se realizează prin intermediul fișierului `php.ini` localizat pe platforme *Windows* în directorul sistemului de operare, și anume directorul care constituie valoarea variabilei sistem `%windir%`.

În acest fișier trebuie adăugată valoarea `php_java.dll` variabilei `extension`. Există mai multe posibilități pentru a face acest lucru. Una dintre ele este eliminarea caracterului "`;`" de la începutul liniei:

```
;extension=php_java.dll,
```

dacă o astfel de linie există în fișierul `php.ini`, iar o altă variantă este adăugarea liniei: `extension=php_java.dll` la sfârșitul fișierului.

În continuare trebuie setate valori pentru următoarele patru variabile:

- `java.class.path` - reprezintă o listă de directoare, arhive *zip* sau alte tipuri de pachete suportate de limbajul java (*jar*, *ejb*, *war* etc.) separate între ele prin caracterul "`;`"; această listă trebuie să conțină calea către pachetul `php_java.jar`, altfel nu se pot utiliza clase *Java*;
- `java.home` - reprezintă numele directorului în care este instalată platforma *Java*;
- `java.library` - reprezintă calea către mașina virtuală *Java* care se dorește a fi încărcată;
- `java.library.path` - reprezintă o listă de directoare în care pot fi găsite anumite pahete de clase *Java*, în plus decât cele cu care este configurată platforma *Java*; directoarele din această listă sunt separate între ele prin caracterul "`;`".

Valorile pentru aceste variabile se setează în fișierul `php.ini` în cadrul secțiunii `[Java]`. Un exemplu de setare a valorilor pentru acestea este:

```
[Java]
java.class.path = c:\php\extensions\php_java.jar
java.home = c:\j2sdk1.4.2
java.library =
          c:\j2sdk1.4.2\jre\bin\server\jvm.dll
java.library.path = c:\php\extensions\
```

Cu aceste setări făcute, mașina virtuală *Java* poate fi încărcată și clasele pot fi utilizate în cadrul *script*-urilor *PHP*.

Utilizare

Clasele *Java* se instanțiază în *PHP* folosind constructorul clasei *Java* explicat în secțiunea următoare.

Metodele și proprietățile statice sau dinamice ale claselor instanțiate se accesează folosind operatorul de indirectare "->". De exemplu, dacă avem o variabilă `$clasa` căreia i-a fost atașată ca valoare o clasă *Java*, atunci pentru a afișa reprezentarea textuală a acesteia trebuie să utilizăm metoda `toString()` (metodă specifică tuturor claselor *Java*) vom utiliza o instrucțiune de forma:

```
echo $clasa->toString();.
```

Clasa Java

Această clasă se folosește pentru a instanția o clasă a limbajului *Java* și reprezintă, de fapt, o clasă derivată din clasa *java.lang.Object* a limbajului *Java*.

Constructorul acesteia primește mai mulți parametri, dintre care primul este un șir de caractere care reprezintă numele clasei *Java* care se instanțiază, iar restul de parametri reprezintă parametrii pentru care se va apela constructorul clasei care se instanțiază.

Rezultatul apelării constructorului clasei *Java* constă într-o clasă a acestui limbaj care are tipul dat ca parametru constructorului.

În cazul în care nu se reușește instanțierea unei clase a limbajului *Java* este generată o excepție *Java*, iar pentru a verifica dacă a fost generată o astfel de excepție se folosește funcția *java_last_exception_get* care este prezentată în secțiunea imediat următoare.

În continuare vom prezenta un exemplu de utilizare a clasei *Java*:

```
<HTML>
  <HEAD>
    <TITLE>
      Java &amp; PHP
    </TITLE>
  </HEAD>
  <BODY>
    <?PHP
      $system = new Java("java.lang.System");
      echo "Versiunea Java este: ";
      echo $system->
          getProperty("java.version")."<BR>";
      echo "Sistem de operare: ";
      echo $system->getProperty("os.name")
                                        ."<BR>";
      echo "Versiunea: ";
      echo $system->getProperty("os.version")
                                        ."<BR>";
    ?>
  </BODY>
</HTML>
```

În figura alăturată poate fi observat efectul execuției acestui cod sursă care folosește constructorul *Java* pentru a obține o instanță a clasei statice *java.lang.System*.

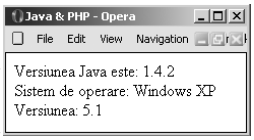


Figura 1: Clasa Java

Funcția java_last_exception_get

Această funcție returnează o clasă *Java* derivată din clasa *java.lang.Exception* care reprezintă ultima excepție care a fost generată sau valoarea logică *false* dacă nu a fost generată nici o excepție de la începutul execuției *script*-ului sau de la ultimul apel al funcției *java_last_exception_clear*.

Funcția *java_last_exception_get* nu are parametri.

În continuare prezentăm un exemplu de utilizare a acestei funcții și în figura 2 poate fi observat efectul execuției codului:

```
<HTML>
  <HEAD>
    <TITLE>
      Conversie
    </TITLE>
  </HEAD>
  <BODY>
    <FORM action = "exceptie.php" method = post>
      Introduceti un şir de caractere:
      <INPUT type = text name = "sir">
      <INPUT type = submit value = "Verificare">
    </FORM>
    <?PHP
      if (isset($_POST["sir"])){
        echo "<HR>";
        $int = new Java("java.lang.Integer");
        $sir = $_POST["sir"];
        $i = $int->parseInt($sir);
        $error = java_last_exception_get();
        if ($error){
          echo "Şirul \"".$sir.
                    "\" nu este număr întreg.";
           java_last_exception_clear();
         }
        else{
          echo "Şirul \"".$i.
                      "\" este număr întreg.";
        }
      }
    ?>
  </BODY>
</HTML>
```

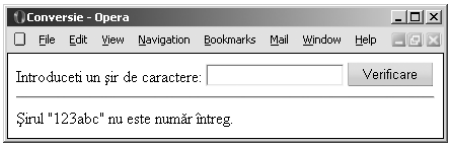


Figura 2: Funcția java_last_exception_get

Funcția java_last_exception_clear

Această funcție nu are nici un parametru și nu returnează nici o valoare, în schimb efectul aplicării ei este acela că este anulată ultima excepție *Java* care a fost generată de către mașina virtuală *Java* în timpul executării de cod.

Apelul acestei funcții este necesar, deoarece, în cazul în care a fost generată o excepție și nu a fost tratată, se poate întâmpla ca după apelul unei funcții să tratăm excepția greșită. De exemplu, în cazul următorului cod se tratează excepția greșită dacă atribuim lui *a* valoarea `abc` și lui *b* valoarea `12`, iar rezultatul nu este cel dorit:

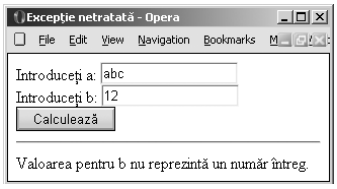


Figura 3: Excepție netratată

```
<HTML>
  <HEAD>
    <TITLE>
      Excepție
          netratată
    </TITLE>
  </HEAD>
  <BODY>
    <FORM action = "exceptie2.php" method = post>
      Introduceți a:
      <INPUT type = text name = "nr1">
      <BR>
      Introduceți b:
      <INPUT type = text name = "nr2">
      <BR>
      <INPUT type = submit value = "Calculează">
    </FORM>
    <?PHP
      if ((isset($_POST["nr1"])) &&
                    (isset($_POST["nr2"]))){
        echo "<HR>";
        $int = new Java("java.lang.Integer");
        $a = $int->parseInt($_POST["nr1"]);
        $b = $int->parseInt($_POST["nr2"]);
        $error = java_last_exception_get();
        if ($error){
          echo "Valoarea pentru b nu reprezintă
                                un număr întreg.";
           java_last_exception_clear();
         }
        else {
          echo "b^2 = ".$b*$b;
        }
      }
    ?>
  </BODY>
</HTML>
```

Dacă după linia în care variabilei `$a` i se atribuie valoarea `abc` apelăm funcția *java_last_exception_clear*, atunci codul va funcționa bine.

În exemplul 3 poate fi observat efectul executării codului de mai sus pentru valorile `abc` și `12`. Se observă că în loc să se afișeze `b^2 = 144`, se afișează o eroare.

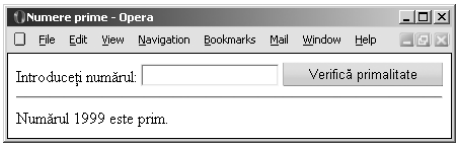


Figura 4: Tratarea corectă a excepțiilor

În continuare prezentăm un exemplu de utilizare corectă a celor două funcții de manipulare a erorilor generate de programele care rulează pe mașina virtuală *Java*, iar rezultatul execuției acestuia poate fi observat în figura 4.

```
<HTML>
  <HEAD>
    <TITLE>
      Numere prime
    </TITLE>
  </HEAD>
  <BODY>
    <FORM action = "prim.php" method = post>
    Introduceți numărul:
    <INPUT type = text name = "number">
    <INPUT type = submit value = "Verifică
                                  primalitate">
    </FORM>
    <?PHP
      if (isset($_POST["number"])){
       echo "<HR>";
       $bint = new Java("java.math.BigInteger",
                          $_POST["number"]);
        $error = java_last_exception_get();
        if ($error){
            echo "Număr invalid.";
            java_last_exception_clear();
         }
        else{
          echo "Numărul ".$_POST["number"];
          $p=20;
          $p=(int)$p;
          if ($bint->isProbablePrime($p))
            echo " este prim.";
          else
            echo " nu este prim.";
        }
      }
    ?>
  </BODY>
</HTML>
```

Importanța librăriei php_java

Pentru programatorii de pagini *web* care cunosc atât limbajul *Java*, cât și limbajul *PHP*, prezența acestei librării are o foarte mare importanță.

Limbajul *Java* are, pe lângă limbajul *PHP*, o serie de facilități care pot fi folosite cu succes în cadrul paginilor *web*.

Java oferă clase pentru prelucrarea diferitelor tipuri de fișiere, lucru care se face mult mai ușor decât în *PHP*, clase

pentru acces la resurse aflate la distanță. În plus, funcțiile scrise în *Java* sunt executate de către procesor mult mai repede decât cele scrise în *PHP*, iar utilizarea codului *Java* oferă o mai mare siguranță creatorilor de pagini *web* dinamice.

Datorită faptului că limbajul *PHP* nu este puternic tipizat, variabilor fiindu-le atașate un tip în mod dinamic, în cadrul scripturilor *PHP* nu se pot utiliza toate facilitățile oferite de platforma *Java*, cum ar fi utilizarea conversiilor explicite de tip, ceea ce îngreunează puțin lucrul cu clase *Java* în paginile *PHP*, fiind necesare clase în *Java* suplimentare, realizate de către programator, care să facă exact ceea ce trebuie.

Singurele conversii explicite de tip care se pot face sunt conversiile între tipurile de bază comune ale celor două limbaje.

Dacă realizați clase pe care doriți să le instanțiați în cadrul *script*-urilor *PHP*, să nu uitați să configurați fișierul `php.ini` corespunzător.

Prelucrarea dinamică a imaginilor

Un alt aspect important al limbajului *PHP* este acela că oferă programatorilor posibilitatea de a crea și publica în cadrul paginilor *web* imagini dinamice.

În cazul în care vă întrebați în ce condiții sunt folosite imaginile generate dinamic vă pot da un exemplu simplu, și anume, gândiți-vă la o aplicație care monitorizează accesările unei pagini *web* și afișează rezultatul sub forma unei imagini.

Pentru a profita de această facilitate în limbajul *PHP* trebuie activată mai întâi biblioteca `php_gd2`. Acest lucru se face prin adăugarea următoarei linii în fișierul de configurare *php.ini* (exemplul de configurare este pentru sistemele de operare din familia *Windows*):

```
extension = php_gd2.dll
```

În acest moment sunt disponibile funcțiile de prelucrare și generare dinamică de imagini.

Această librărie permite și adăugarea de text în cadrul imaginilor generate și oferă suport pentru imagini în mai mai multe formate.

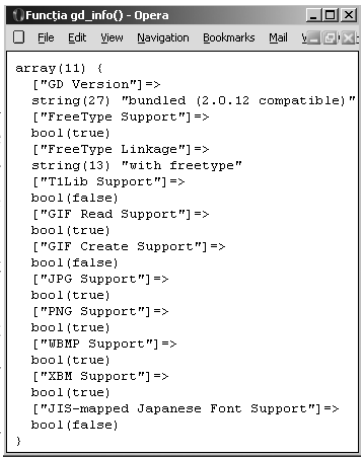
Pentru a obține imagini de calitate foarte bună, această bibliotecă oferă suport integrat pentru bibliotecile *FreeType* 1, *FreeType* 2 și *T1Lib*, care au ca scop încărcarea și desenarea *font*-urilor independent de platformă, însă nu vom discuta în cadrul acestui articol despre aceste trei biblioteci, ci numai vom aminti funcțiile oferite de limbajul *PHP* pentru accesarea lor.

În continuare vom prezenta cele mai importante funcții, oferite de limbajul *PHP,* necesare pentru prelucrarea imaginilor.

Formate de imagini suportate

Pentru a putea afla ce formate de imagini suportă biblioteca instalată (în funcție de versiunea *PHP*) se poate executa funcția `gd_info()`.

În continuare vă prezentăm un exemplu de utilizare a acestei funcții și facilitățile oferite de biblioteca `php_gd2` și în figura 5 poate fi observat efectul execuției codului.




```
<HTML>
  <TITLE>
    Funcția
          gd_info()
  </TITLE>
  <BODY>
    <PRE>
    <?PHP
      var_dump
       (gd_info());
    ?>
    </PRE>
  </BODY>
</HTML>
```

Figura 5: Funcția gd_info

Rezultatul returnat de această funcție îl reprezintă un vector cu mai multe componente.

Se poate observa că printre aceste componente se află versiunea bibliotecii și facilitățile oferite.

Formatele de imagini suportate de biblioteca *php_gd2* sunt *GIF*, *JPG*, *PNG*, *WBMP* și *XBM* (*XPM*). Se observă că biblioteca folosită de noi nu permite salvarea imaginilor în format *GIF*.

Crearea dinamică a imaginilor

Pentru a crea imagini în mod dinamic avem la dispoziție două opțiuni. Putem crea o imagine care este bazată pe o paletă de culori (pentru acest tip de imagini avem la dispoziție doar 256 de culori cu care putem opera) sau putem crea o imagine cu număr neprecizat de culori (*true color*).

Funcțiile cu ajutorul cărora se pot crea imagini sunt:

- `imagecreate` - această funcție creează o imagine goală care este bazată pe o paletă de culori; primește ca parametri două numere întregi care reprezintă dimensiunile imaginii (lățimea și înălțimea) și returnează o variabilă de tipul `resource` care reprezintă identificatorul de acces la imaginea nou creată;
- `imagecreatetruecolor` - această funcție creează o imagine care are fundalul negru și număr nedefinit de culori; primește ca parametri două numere întregi care reprezintă dimensiunile imaginii (lățimea și înălțimea) și returnează o variabilă de tipul `resource` care reprezintă identificatorul de acces la imaginea nou creată.

Trebuie avut în vedere, că pentru a afișa o imagine dinamică pe o pagină *web* trebuie realizată o pagină care are un element **`IMG`** al cărui atribut **`src`** să indice calea către *script*-ul *PHP* care creează imaginea.

În continuare vă prezentăm un exemplu de creare a unei imagini care se bazează pe o paletă de culori împreună cu o pagină web simplă în cadrul căreia este afișată imaginea.

Pagina *web* este:

```
<HTML>
  <TITLE>
    Afișarea imaginii dinamice
  </TITLE>
  <BODY>
    <IMG src = "creareimagine1.php">
  </BODY>
</HTML>
```

Script-ul **creareimagine1.php** creează o imagine cu lățimea și înălțimea de 100 de pixeli, a cărei paletă conține o singură culoare, și anume gri, și o transmite programului de navigare în format *PNG*. Se observă că am setat antetul corespunzător transmisiei unei astfel de imagini.

```
<?PHP
  header("Content-type: image/png");
  $im = imagecreate(100, 100);
  $fundal = imagecolorallocate($im,64,64,64);
  imagepng($im);
  imagedestroy($im);
?>
```

Figura 6: Funcția imagecreate

În figura alăturată poate fi observat efectul încărcării paginii *web* în fereastra programului de navigare.

Încărcarea imaginilor

Biblioteca permite, de asemenea, citirea de imagini aflate la o anumită locație și încărcarea acestora în memorie pentru prelucrare și, eventual, publicare.

Funcțiile care încarcă imagini deja existente sunt:

- `imagecreatefromgif` - această funcție primește ca parametru o cale spre un fișier, care se poate afla și la distanță, și returnează o variabilă de tipul `resource` care reprezintă identificatorul de acces la imaginea de tipul *GIF* încărcată;
- `imagecreatefromjpeg` - această funcție primește ca parametru o cale spre un fișier, care se poate afla și la distanță, și returnează o variabilă de tipul `resource` care reprezintă identificatorul de acces la imaginea de tipul *JPG* încărcată;
- `imagecreatefrompng` - această funcție primește ca parametru o cale spre un fișier, care se poate afla și la distanță, și returnează o variabilă de tipul `resource` care reprezintă identificatorul de acces la imaginea de tipul *PNG* încărcată;
- `imagecreatefromwbmp` - această funcție primește ca parametru o cale spre un fișier, care se poate afla și la distanță, și returnează o variabilă de tipul `resource` care reprezintă identificatorul de acces la imaginea de tipul *WBMP* încărcată;
- `imagecreatefromxbm` - această funcție primește ca parametru o cale spre un fișier, care se poate afla și la distanță, și returnează o variabilă de tipul `resource` care reprezintă identificatorul de acces la imaginea de tipul *XBM* încărcată;

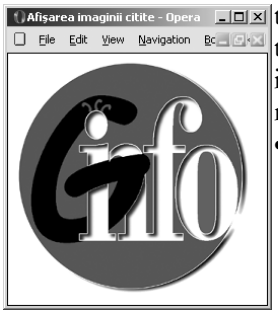


Figura 7: Încărcarea unei imagini

- `imagecreatefromxpm` - această funcție primește ca parametru o cale spre un fișier, care se poate afla și la distanță, și returnează o variabilă de tipul `resource` care reprezintă identificatorul de acces la imaginea de tipul *XPM* încărcată;
- `imagecreatefromstring` - această funcție primește ca parametru un șir de caractere care conține reprezentarea binară a unei imagini, și returnează o variabilă de tipul `resource` care reprezintă identificatorul de acces la imaginea încărcată.

În continuare vă prezentăm un exemplu de utlizare a acestei funcții:

```
<?PHP
  header("Content-type: image/png");
  $im = imagecreatefromgif("images/ginfo.gif");
  imagepng($im);
  imagedestroy($im);
?>
```

În figura 7 poate fi observat efectul execuției acestui cod dintr-o pagină *web*.

În continuare vă prezentăm câteva clase de funcții care pot fi utilizate la prelucrarea imaginilor.

Desenarea pe imagini

Pentru a desena pe o imagine creată dinamic sau încărcată dintr-un fișier avem la dispoziție mai multe funcții:

- `imagearc` - desenează pe o imagine un arc de elipsă; primește ca parametri identificatorul de acces la imaginea pe care se va desena, coordonata *x* a centrului elipsei, coordonata *y* a centrului elipsei, lungimea și înălțimea elipsei, unghiurile de start și de stop ale arcului și culoarea cu care se va desena; desenarea se face în sens trigonometric;
- `imagefilledarc` - desenează pe o imagine un arc de elipsă hașurat; are aceeași parametri cu funcția anterioară la care se mai adaugă unul care reprezintă stilul hașurii și care poate avea valorile: `IMG_ARC_PIE`, `IMG_ARC_CHORD`, `IMG_ARC_NOFILL`, `IMG_ARC_EDGED`;
- `imageline` - desenează o linie pe o imagine; primește ca parametri identificatorul de acces al imaginii pe care se va desena, coordonata *x* a primului capăt al liniei, coordonata *y* a primului capăt, coordonata *x* a celui de-al doilea capăt, coordonata *y* a celui de-al doilea capăt și culoarea cu care se va desena linia;
- `imagedashedline` - desenează o linie punctată pe o imagine; are aceeași parametri ca funcția anterioară și va folosi stilul implicit sau cel setat de utilizator;

- `imagesetthickness` - setează grosimea cu care se vor trasa liniile pentru o anumită imagine; primește ca parametri identificatorul de acces al imaginii și un număr întreg care reprezintă grosimea cu care se vor desena liniile pe imagine;
- `imageellipse` - desenează o elipsă pe o imagine; primește ca parametri identificatorul de acces al imaginii pe care se va desena, coordonata *x* a centrului elipsei, coordonata *y* a centrului elipsei, lungimea și înălțimea elipsei și culoarea cu care se va desena;
- `imagefilledellipse` - desenează o elipsă plină; are aceeași parametri cu funcția anterioară;
- `imagepolygon` - desenează un poligon pe o imagine; primește ca parametri identificatorul de acces al imaginii pe care se va desena poligonul, un tablou ale cărui elemente sunt numere întregi și reprezintă coordonatele punctelor poligonului care se va desena, numărul de puncte ale poligonului și culoarea cu care se va desena; tabloul cu punctele poligonului conține pe pozițiile pare coordonatele *x* ale punctelor, iar pe pozițiile impare coordonatele *y* ale punctelor;
- `imagefilledpolygon` - desenează un poligon plin; are aceeași parametri cu funcția anterioară;
- `imagerectangle` - desenează un dreptunghi pe o imagine; primește ca parametri identificatorul de acces al imaginii pe care se va face desena, coordonata *x* a colțului din partea stânga-sus a dreptunghiului, coordonata *y* a colțului din partea stânga-sus a dreptunghiului, coordonata *x* a colțului din partea dreapta-jos a dreptunghiului, coordonata *y* a colțului din partea dreapta-jos a dreptunghiului și culoarea cu care se va desena;
- `imagefilledrectangle` - desenează un dreptunghi plin; are aceeași parametri cu funcția anterioară;
- `imagefill` - umple cu o culoare o zonă de pe imagine care are aceeași culoare cu cea a punctului care se află la coordonatele specificate ca parametru; primește ca parametri identificatorul de acces al imaginii pe care se va face umplerea, coordonata *x* a unui punct din zona care se va umple cu culoare, coordonata *y* a punctului și culoarea cu care se va umple;
- `imagefilltoborder` - umple cu o culoare o zonă de pe imagine care este mărginită de o culoare specificată ca parametru; primește ca parametri identificatorul de acces al imaginii pe care se va face umplerea, coordonata *x* a unui punct din zona care se va umple cu culoare, coordonata *y* a punctului, culoarea punctelor cu care este mărginită zona și culoarea cu care se va umple;
- `imagesetpixel` - schimbă culoarea unui punct de pe o imagine; primește ca parametri identificatorul de acces al imaginii, coordonata *x* a punctului a cărui culoare se va schimba, coordonata *y* a punctului și noua culoarea a acestuia;
- `imagecolorat` - returnează culoarea unui anumit punct de pe imagine; primește ca parametri identificatorul de acces al imaginii, coordonata *x* a punctului și coordonata *y* a punctului;
- `imagecolorallocate` - primește ca parametri identificatorul de acces al unei imagini, o valoare pentru componenta *roșu* a unei culori, o valoare pentru componenta *verde* a unei culori și o valoare pentru componenta *albastru* a unei culori și returnează un număr întreg care reprezintă culoarea având cele trei componente; în cazul în care imaginea se bazează pe o paletă de culori, atunci se returnează indicele culorii din paletă; dacă în paleta de culori nu există o culoarea care să aibă toate cele trei componente, atunci ea este adăugată la paleta de culori a imaginii;
- `imagecolordeallocate` - primește ca parametri identificatorul de acces al unei imagini și o culoare; în cazul în care imaginea specificată ca parametru se bazează pe o paletă de culori, atunci culoarea este eliminată din paletă.

În continuare prezentăm un exemplu care folosește o parte din funcțiile prezentate anterior. În figura 8 poate fi observat efectul execuției codului din cadrul unei pagini *web*:

```
<?PHP
   header("Content-type: image/png");
   $im = imagecreatefromjpeg("image/ginfo.jpg");

  $red = imagecolorallocate($im,255,0,0);
   $blue = imagecolorallocate($im,0,0,255);
   $green = imagecolorallocate($im,0,255,0);
   $yellow = imagecolorallocate($im,255,255,0);
   $cyan = imagecolorallocate($im,0,255,255);
   $magenta = imagecolorallocate($im,255,0,255);

   imagefill($im,229,229,$cyan);
   imagefill($im,30,100,$magenta);
   imagefill($im,80,160,$yellow);

   imagefilledrectangle($im,0,0,50,100,$red);
   imagefilledellipse($im,100,100,50,50,
                                          $green);
   imagesetthickness($im,3);
   $poly = array(10,10,200,50,225,100,150,200,
                                   100,75,50,50);
   imagefilledpolygon($im,$poly,6,$blue);
   imageline($im,0,229,229,0,$red);
   imagepng($im);
   imagedestroy($im);
?>
```

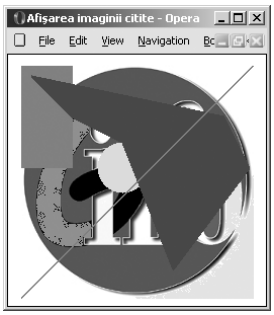


Figura 8: Exemplu prelucrare

Va urma...

În articolul următor vom continua prezentarea funcțiilor referitoare la prelucrarea imaginilor și vom începe prezentarea modului de utilizare a bazelor de date *MySQL* în interiorul *script*-urilor *PHP*.