Pagini WEB cu PHP 4

Mihai Scorțaru, Claudiu Soroiu

În acest articol vom prezenta o clasă creată în redacția GInfo și modul in care se pot aplica conceptele de introspecție și reflexie în limbajul PHP.

Introspecție și reflexie în PHP

Introspecția reprezintă acțiunea prin care se pot afla informații despre metodele și proprietățile unei clase.

Reflexia reprezintă acțiunea prin care se pot introduce în mediul de execuție și în domeniul de vizibilitate al mediului de execuție curent – clase stocate în altă parte sau clase create de utilizator pentru a fi folosite cu anumite module care au fost realizate anterior de utilizator sau de alți programatori.

Termenii de *introspecție* și *reflexie* în practică nu pot fi separați, deoarece pentru a introduce un obiect în mediul de execuție, trebuie cunoscut numele, metodele și proprietățile acestuia și nu pot fi aflate informații despre o clasă care nu este încărcată și nu este vizibilă în mediul de execuție.

În continuare vom prezenta un set de funcții cu ajutorul cărora se pot afla informații despre clasele încărcate și vom exemplifica utilizarea acestora în cadrul unei clase similare clasei *Class* din limbajul *Java*.

Funcția get_declared_classes

Această funcție returnează un tablou care conține numele tuturor claselor definite în momentul execuției unui *script PHP*.

Funcția class_exists

Această funcție primește ca parametru un șir de caractere și returnează valoarea logică **TRUE** dacă există o clasă care să aibă numele identic cu șirul de caractere primit ca parametru și valoarea logică **FALSE** în caz contrar.

Funcția get_parent_class

Această funcție are un parametru care poate fi de tip șir de caractere sau obiect.

În cazul în care parametrul este de tipul șir de caractere, atunci funcția `get_parent_class` returnează un șir de caractere care reprezintă numele clasei care este părintele direct al clasei al cărui nume este dat de parametru.

În cazul în care parametrul este de tip obiect, atunci funcția returnează un șir de caractere care reprezintă numele părintelui direct al clasei a cărei instanță este reprezentată de obiectul dat ca parametru.

Funcția is_a

Această funcție are doi parametri. Primul parametru este de tip obiect, iar al doilea este de tip șir de caractere.

Funcția `is_a` returnează valoarea logică **TRUE** dacă primul parametru este de tipul reprezentat de al doilea parametru sau de un tip derivat din cel de-al doilea parametru și valoarea logică **FALSE** în caz contrar.

De exemplu, dacă avem clasele `cls1`, `cls2` și `cls3` și `cls3` este derivată din `cls2` și `cls2` este derivată din `cls1`, atunci, dacă `obj` reprezintă o instanță a clasei `cls3`, în urma apelurilor `is_a(obj, "cls1")`, `is_a (obj, "cls2")` și `is_a(obj, "cls3")` se obține valoarea logică **TRUE**.

Funcția is_subclass_of

Funcția `is_subclass_of` are aceiași parametri cu funcția anterioară. Funcționalitatea ei diferă față de funcția precedentă prin faptul că valoarea logică **TRUE** este returnată numai în cazul în care primul parametru este o instanță a unui tip derivat din tipul al cărui nume este dat de cel de-al doilea parametru.

În condițiile exemplului anterior, în urma apelului `is_subclass_of(obj, "cls3")` se va obține valoarea logică **FALSE**.

Funcția get_class_methods

Această funcție are un parametru care poate fi de tip șir de caractere care reprezintă numele unei clase sau de tip obiect.

Funcția `get_class_methods` returnează un tablou ale cărui elemente sunt de tipul șir de caractere și care repre-

zintă numele metodelor definite în cadrul clasei cu numele primit ca parametru sau în cadrul clasei care este reprezentată de obiectul primit ca parametru.

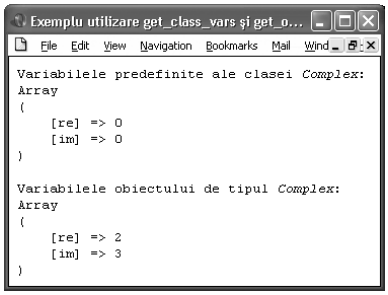

Figura 1: Exemplu de utilizare a funcţiilor get_class_vars şi get_object_vars

În cazul în care parametrul este de tip şir de caractere şi nu este definită nici o clasă cu acest nume, atunci funcţia returnează valoarea logică **FALSE**.

Funcţia get_class_vars

Această funcţie are un singur parametru de tip şir de caractere care reprezintă numele unei clase şi returnează un tablou ale cărui elemente sunt de tip şir de caractere şi care reprezintă valorile implicite ale variabilelor definite în cadrul clasei.

Tabloul rezultat în urma apelului acestei funcţii este indexat după numele variabilelor definite în cadrul clasei.

În continuare vă prezentăm un exemplu de utilizare a acestei clase, iar în figura 1 se poate observa efectul execuţiei *script*-ului:

```
<?PHP
  class Complex{
    var $re=0;
    var $im=0;

    function Complex($re = 0, $im = 0){
      $this->re = $re;
      $this->im = $im;
    }

    function add($a){
      return new Complex($a->re + $this->re,
                         $a->im + $this->im);
   }

    function toString(){
      if ($this->im==0)
        return (string) $this->re;
      if ($this->re==0)
        return $this->im.'i';
      if ($this->im<0)
        return $this->re.$this->im.'i';
      return $this->re.'+'.$this->im.'i';
    }
  }
?>

<HTML>
  <HEAD>
    <TITLE>
      Exemplu utilizare get_class_vars şi
                                  get_object_vars
    </TITLE>
  </HEAD>
  <BODY>
    <PRE>
    <?PHP
       echo "Variabilele predefinite ale clasei
                              <I>Complex</I>:\n";
      print_r(get_class_vars('Complex'));
      echo "\nVariabilele obiec-
                                  tului de tipul
                              <I>Complex</I>:\n";
      $a = new Complex(2, 3);
      print_r(get_object_vars($a));
    ?>
    </PRE>
  </BODY>
</HTML>
```

Funcţia get_class

Această funcţie primeşte ca parametru un obiect şi returnează numele clasei care reprezintă tipul obiectului primit ca parametru.

Funcţia get_object_vars

Această funcţie are un singur parametru de tip obiect şi returnează un tablou ale cărui elemente sunt de tip şir de caractere şi care reprezintă valorile variabilelor definite în cadrul acestuia.

Tabloul rezultat în urma apelului acestei funcţii este indexat după numele variabilelor definite în cadrul obiectului.

În *script*-ul al cărui rezultat se poate observa în figura 1 am utilizat şi această funcţie.

Se poate observa că funcţia `get_object_vars` se aseamănă foarte mult cu funcţia `get_class_vars`.

Funcţia call_user_method

Această funcţie realizează apelul unei metode care aparţine unei clase sau unui obiect cu parametrii specificaţi.

Funcţia `call_user_method` are mai mulţi parametri. Primul parametru este de tip şir de caractere şi reprezintă numele metodei care se va apela. Al doilea parametru este de tip obiect sau de tip şir de caractere şi reprezintă numele unei clase. În cazul în care acest parametru este de tip obiect, metoda dată de primul parametru este apelată pentru obiect, iar dacă parametrul este de tip şir de caractere, atunci metoda este apelată pentru clasa al cărui nume este dat de acest parametru. Restul parametrilor reprezintă parametrii cu care se va apela metoda al cărui nume este dat de primul parametru.

Funcţia call_user_method_array

Această funcţie are aceeaşi funcţionalitate cu funcţia anterioară cu diferenţa că are doar trei parametri, iar al treilea

parametru este un tablou unidimensional care conține parametrii cu care se va apela metoda dată de primul parametru pentru clasa sau obiectul reprezentat de cel deal doilea parametru.

Aceste ultime două funcții pot fi asociate reflexiei. Alături de acestea, pentru reflexie se mai pot utiliza funcțiile `include`, `include_once`, `require`, `require_once` care au fost prezentate în articolele anterioare și funcția `eval`.

Pentru introspecție, în afară de funcțiile prezentate în acest articol mai pot fi utilizate și funcțiile `dl`, `extension_loaded` și `get_declared_classes`.

În continuare vom prezenta funcțiile `eval`, `dl`, `get_declared_classes` și `extension_loaded`.

Funcția eval

Această funcție primește ca parametru un șir de caractere care reprezintă o secvență de cod *PHP* și realizează execuția acestuia.

Funcția dl

Această funcție realizează încărcarea în timpul execuției *script*-ului a unor extensii externe, dacă acest lucru este permis de setările interpretorului *PHP*.

Funcția `dl` primește ca parametru un șir de caractere care reprezintă calea către extensia care trebuie încărcată și returnează valoarea logică **`TRUE`** dacă s-a reușit încărcarea extensiei și valoarea logică **`FALSE`** în caz contrar.

Funcția extension_loaded

Această funcție verifică dacă o anumită extensie este încărcată. Funcția primește un parametru de tip șir de caractere care reprezintă numele unei extensii și returnează o valoare logică corespunzătoare.

Funcția get_declared_classes

Funcția `get_declared_classes` nu are nici un parametru și returnează un tablou unidimensional care conține numele tuturor claselor încărcate în mediul de execuție în momentul apelării acesteia.

Clasa _Class

În continuare vă prezentăm o clasă pe care am construit-o în vederea descoperirii de informații legate de clase prezente în mediul de execuție. Această clasă este experimentală și acoperă doar o parte a introspecției care poate fi realizată în cadrul limbajului *PHP*.

Metodele clasei `_Class` se bazează pe funcțiile prezentate la începutul acestui articol.

La realizarea acestei clase ne-am concentrat atenția doar asupra aflării de informații legate de clase, și nu asupra aflării de informații legate de obiectele care reprezintă instanțe ale unor clase. În concluzie, anumite metode nu dau același rezultat cu cel obținut în urma aplicării lor pentru obiecte.

Structura acestei clase este următoarea:

```
class _Class{
  var $classname;

  function _Class($clazz = NULL){...}
  function getType(){...}
  function &getParentClass(){...}
  function &getAncestors(){...}
  function &getAncestorsNames(){...}
  function &getOwnMethodsNames(){...}
  function &getAllMethodsNames(){...}
  function &getOwnDeclaredFields(){...}
  function &getAllDeclaredFields(){...}
}
```

În cadrul acestei implementări, datorită faptului că în cadrul limbajului *PHP* nu există o clasă din care să fie derivate toate celelalte, cum este cazul clasei `Object` din limbajul *Java*, am considerat că părintele tuturor claselor este o clasă virtuală al cărei nume este "`(null)`", iar, ca o excepție, părintele clasei virtuale `(null)` este tot clasa `(null)`.

În continuare vom detalia clasa `_Class` și vom da câteva exemple de utilizare a acesteia.

Proprietatea classname

Această proprietate reține numele clasei pentru care se efectuează toate operațiile.

Constructorul _Class

Constructorul `_Class` are un singur parametru care este opțional. Acesta poate fi de tip șir de caractere sau de tip obiect.

În cazul în care parametrul este de tip șir de caractere se va verifica existența clasei cu numele dat de acest parametru și dacă o astfel de clasă nu există și numele ei este diferit de "`(null)`" este generată o eroare.

În cazul în care parametrul este de tip obiect, se va utiliza numele clasei care reprezintă numele acestui obiect.

Dacă parametrul lipsește sau are una dintre valorile `NULL` sau "`(null)`" atunci se va considera pentru prelucrările ulterioare clasa virtuală cu numele "`(null)`".

În continuare vă prezentăm codul sursă al acestui constructor:

```
function _Class($clazz = NULL){
  if (is_string($clazz)){
    if (strtolower($clazz) == "(null)")
      $this->classname = NULL;
    else{
      $this->classname = $clazz;
      if (!class_exists($this->classname))
        trigger_error("Class $clazz not found!",
                                    E_USER_ERROR);
```

```
    }
  }
  elseif (is_object($clazz))
    $this->classname = get_class($clazz);
  elseif (is_null($clazz))
    $this->classname = NULL;
  else
    trigger_error("Invalid parameter type",
                                  E_USER_ERROR);
}
```

Metoda _Class->getType()

Această metodă returnează numele clasei asupra căreia se efectuează operațiile date de celelalte metode, nume care este dat de variabila `classname`.

Codul sursă al acestei metode este următorul:

```
function getType(){
  if (is_null($this->classname))
    return "(null)";
  return $this->classname;
}
```

Metoda _Class->getParentClass()

Această metodă returnează o referință la un obiect de tipul `_Class` al cărei constructor a fost apelat pentru părintele clasei curente.

Codul sursă al acestei metode este următorul:

```
function &getParentClass(){
  if (is_null($this->classname))
    return new _Class(NULL);
  $parent = get_parent_class($this->classname);
  if ($parent)
    return new _Class($parent);
  else
    return new _Class(NULL);
}
```

Metoda _Class->getAncestors()

Această metodă returnează o referință către un tablou unidimensional ale cărui elemente sunt de tipul `_Class` și reprezintă strămoșii clasei curente.

Această metodă a fost implementată folosind un ciclu repetitiv **`while`**. Vom folosi variabila `$cls` care la început este inițializată cu numele clasei curente și, parcurgând în sus ierarhia de clase, pornind de la clasa curentă și până la întâlnirea valorii **`NULL`** vom completa elementele tabloului a cărei referință va fi returnată ca rezultat.

Codul sursă al acestei metode este:

```
function &getAncestors(){
  if (is_null($this->classname))
    return array(new _Class(NULL));
  $ancestors = array();¤
  $cls = $this->classname;
  while($cls = get_parent_class($cls))
    $ancestors[] = new _Class($cls);
  $ancestors[] = new _Class(NULL);
  return $ancestors;
}
```

Metoda _Class->getAncestorsNames()

Această metodă returnează o referință către un tablou unidimensional ale cărui elemente sunt de tip șir de caractere și reprezintă numele strămoșilor clasei curente.

Diferența dintre această metodă și cea precedentă constă în faptul că în tabloul creat se vor stoca doar numele claselor fără a se mai apela constructorul clasei `_Class`.

Codul sursă al acestei metode este:

```
function &getAncestorsNames(){
  if (is_null($this->classname))
    return array("(null)");
  $ancestors = array();
  $cls = $this->classname;
  while($cls = get_parent_class($cls))
    $ancestors[] = $cls;
  $ancestors[] = "(null)";
  return $ancestors;
}
```

Metoda _Class->getAllMethodsNames()

Această metodă returnează o referință către un tablou unidimensional ale cărui elemente sunt de tip șir de caractere și reprezintă numele metodelor pe care clasa curentă le conține.

Tabloul construit de metoda `getAllMethodsNames` conține și numele metodelor clasei curente care au fost moștenite de la clasa părinte.

Codul sursă al acestei metode este:

```
function &getAllMethodsNames(){
  if (is_null($this->classname))
    return array();
  return get_class_methods($this->classname);
}
```

Metoda _Class->getOwnMethodsNames()

Această metodă returnează o referință către un tablou unidimensional ale cărui elemente sunt de tip șir de caractere și reprezintă numele metodelor care au fost definite în cadrul clasei curente.

Construirea acestui tablou se realizează astfel: se determină toate metodele pe care le posedă clasa curentă sub forma unui tablou unidimensional primit ca rezultat în urma apelului funcției `get_class_methods`, se determină toate metodele pe care le posedă părintele clasei curente sub forma unui tablou unidimensional și se returnează o referință spre un tablou unidimensional care conține diferența dintre cele două tablouri determinate.

Codul sursă al acestei metode este:

```
function &getOwnMethodsNames(){
  if (is_null($this->classname))
    return array();
  $clsm = get_class_methods($this->classname);
  $pcls = get_parent_class($this->classname);
  if (!$pcls) return $cls;
  $pclsm = get_class_methods($pcls);
  return array_diff($clsm, $pclsm);
}
```

Metoda _Class->getAllDeclaredFields()

Această metodă returnează o referință spre un tablou unidimensional care conține valorile implicite ale câmpurilor implicite definite în cadrul clasei curente. Acest tablou este obținut în urma apelului funcției `get_class_vars` pentru a determina numele proprietăților implicite definite în cadrul clasei curente. Acest tablou conține și proprietățile moștenite de la clasa părinte.

Codul sursă al acestei metode este următorul:

```
function &getAllDeclaredFields(){
  if (is_null($this->classname))
    return array();
  return get_class_vars($this->classname);
}
```

Metoda _Class->getOwnDeclaredFields()

Această metodă returnează o referință spre un tablou unidimensional care conține valorile implicite ale câmpurilor implicite definite în cadrul clasei curente fără a lua în considerare proprietățile moștenite de la clasa părinte.

Tabloul a cărui referință se returnează se construiește folosind diferența dintre tabloul care conține proprietățile clasei curente și tabloul care conține proprietățile clasei părinte, ca în cazul metodei `getOwnMethodsNames`.

Codul sursă al acestei metode este următorul:

```
function &getOwnDeclaredFields(){
  if (is_null($this->classname))
    return array();
  $vars = get_class_vars($this->classname);
  $pcls = get_parent_class($this->classname);
    if (!$pcls) return $vars;
  $pvars = get_class_vars($pcls);
  return array_diff($vars, $pvars);
}
```

Exemplu

În continuare vă prezentăm un exemplu de utilizare a clasei `_Class` care este definită în fișierul *class_introspector.php*.

Script-ul următor afișează pentru fiecare clasă, încărcată în mediul de execuție al interpretorului *PHP*, numele strămoșilor, metodele proprii și valorile implicite ale proprietăților acestora. În figura 2 poate fi observat efectul execuției acestui script.

```
<?PHP
  include "class_introspector.php";

  function printAncestors($b){
    $a=$b->getAncestors();
    for($i = 0;$i < sizeof($a);$i++)
      echo $a[$i]->getType().' ';
  }

  function printMethods($b){
    $a = $b->getOwnMethodsNames();
    for($i = 0;$i < sizeof($a);$i++)
      echo $a[$i].' ';
  }

  function printFields($b){
    $a = $b->getOwnDeclaredFields();
    foreach($a as $key => $keyv)
      echo "'".$key." = (".gettype($keyv).")"
                                  .$keyv."' ";
  }
?>
<HTML>
  <HEAD>
   <TITLE>Exemplu de utilizare _Class</TITLE>
  </HEAD>
  <BODY><PRE>
  <?PHP
  $a = get_declared_classes();
  for($i = 0;$i < sizeof($a);$i++){
    $b = new _Class($a[$i]);
    echo "<B>".$b->getType()."</B><BR>";
    echo "  <B>Strămoşi: </B> ";
    printAncestors($b);
    echo "<BR>";
    echo "  <B>Metode proprii:</B> ";
    printMethods($b);
    echo "<BR>";
    echo "  <B>Proprietăţi proprii
                          predefinite:</B> ";
    printFields($b);
    echo "<BR>";
    echo "<BR>";
  }
  ?>
  </PRE></BODY>
</HTML>
```

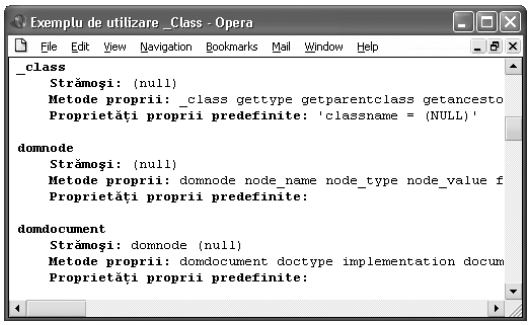

Figura 2: Exemplu de utilizare a clasei _Class