Лабораторно упражнение 9

„Магически” или „вълшебни” методи
Сесии и бисквитки

1. **Магически методи .**

В PHP 5 съществуват т.нар. магически методи. Магически методи са методи на класа, които са достъпни за всички инстанции на един клас, дефинирани в рамките на класа. Името им винаги започва с "__".

Примери за магически методи са: __construct и __destruct () – методи за създаване и за унищожаване на обект и са със статус на достъп public (трябва изрично да са декларирани като публични).

Често използвани магически методи в PHP 5 са:

__construct ()
__destruct ()
__set ()
__get ()
__call ()
__toString ()
__sleep ()
__wakeup ()
__isset ()
__unset ()
__autoload ()
__clone ()

1.1. Магически методи __set() и __get() .

Пример 1.1:

```
<?php
class Customer {
public $name;
}
$c = new Customer();
$c->name = "Ivan Ivanov"; // $name is set because its public
$c->email = "ivan@abv.bg"; //assigning ivan@abv.bg to the $email //variable.
echo $c->email;
?>
```

Примерът написан на друг обектно ориентиран език, ще изведе грешка, защото се присвоява стойност на недефинирано свойство, в случая на свойство email на обект $c. Php е слабо типизиран език, т.е. могат да се създават недефинирани в класа свойства динамично и да се извличат стойности на такива недефинирани свойства. Това може да е опасно. Нарушава се капсулацията, всеки може да си създава свойства. Тази функционалност е полезна, ако все пак се контролира.

Решението е, чрез използване на магически методи __set, __get. Магическите методи са обикновени методи, но имат определен подпис и се извикват само, ако са дефинирани в класа. В

момента, когато PHP види, че някой се опитва да пише в недефинирано свойство, то той извиква магически метод __set (ако е дефиниран в класа). Магическите методи, в случая __set, __get се явяват механизъм за контрол.

В примера: за да се контролира писането в недефинирано свойство, то се извиква set – с 2 параметъра name (име) – value (стойност). Обратно с get, може да се извлича стойност на недефинирано свойство (с един параметър).

Пример 1.2:

```
<?php
class Customer {
public $name;
private $data = array();
public function __set($dt, $vl) {
$this->data[$dt] = $vl;
}
public function __get($dt) {
return $this->data[$dt];
}
}
$c = new Customer();
$c->name = "Ivan Ivanov"; // $name is set because its public
$c->__set("email", "ivan@abv.bg");
echo "The email of ". $c->name." is ".($c->__get("email"));
$c->age=23;
echo "<br> It's age is ".($c->__get("age"));
$c->__set("grade", 4.20);
echo "<br>It's grade is ".$c->grade;
?>
```

Резултат:

The email of Ivan Ivanov is ivan@abv.bg
It's age is 23
It's grade is 4.2

1.2. Магически метод __toString()

Методът __toString позволява на даден клас да бъде преобразуван до низ. От PHP 5.2.0 при преобразуването на обекти в низ без метода __toString се генерира грешка.

Пример 2.1:

```
<?php
class Customer {
private $firstName, $lastName, $email;
public function __construct($firstName, $lastName, $email) {
$this->firstName = $firstName;
$this->lastName = $lastName;
$this->email = $email;
}
public function __toString() {
```

```
return "The customer $this->firstName $this->lastName"." has e-mail
address ". $this->email;
}
}
$c = new Customer("Ivan","Ivanov","ivo@abv.bg");
echo $c;
?>
```

Резултат:

The customer Ivan Ivanov has e-mail address ivo@abv.bg

Забележете echo $c;

Обект $c от тип Customer се конвертира към string type. Автоматично се извиква магическия метод __toString(). Ако методът не беше деклариран ще се изведе фатална грешка.

Пример 2.2:

Същият пример без **метод __toString()**

```
<?php
class Customer {
private $firstName, $lastName, $email;
public function __construct($firstName, $lastName, $email) {
$this->firstName = $firstName;
$this->lastName = $lastName;
$this->email = $email;
}
public function show () {
echo "The customer $this->firstName $this->lastName"." has e-mail address
". $this->email;
}
}
$c = new Customer("Ivan","Ivanov","ivo@abv.bg");
$c->show();
//echo $c; // ще изведе грешка:
// Recoverable fatal error: Object of class Customer could not be converted to string
?>
```

1.3. Magic Methods – __isset() and __unset()

Тези методи __isset() и __unset() се извикват автоматично когато се извиква isset() и unset() за недекларирани членове на класа. Магическият метод __isset() получава един аргумент – стойността на тестваната променлива, ако променливата е установена или не.

Магическият метод __unset() получава един аргумент – името на променливата, чиято стойност е нужно да се унищожи.

Пример 3:

```
<?php
class Customer {
```

```
     private $data = array();
     public function __set($dt, $vl) {
          $this->data[$dt] = $vl;
     }
     public function __get($dt) {return $this->data[$dt];   }
     public function __isset($dt) {return isset($this->data[$dt]);}
     public function __unset($dt) {unset($this->data[$dt]);}
}
$c = new Customer();
$c->name = "Ivan Ivanov";
echo "The customer is: $c->name";
echo "<br>The result of isset is ".isset($c->name)."\n";
unset($c->name);
echo "<br>After unset, the result of isset is ".isset($c->name);
?>
```

Резултат:

The customer Ivan Ivanov
The result of isset is 1
After unset, the result of isset is

1.4. Magic method __call().

Магическият метод **__call()** има 2 аргумента: името на недекларирaн метод, извикан от програмата и втори – един масив, който съдържа списък на параметрите на недекларирания метод.

Пример 4:

```
<?php
class Customer {
     public function __call($n, $m) {
          var_dump($n);
          var_dump($m);
     }
}
$c = new Customer();
$c->My_Method("Name","email");
?>
```

Резултат:

string(9) "My_Method" array(2) { [0]=> string(4) "Name" [1]=> string(5) "email" }

1.5. Magic method __invoke

Методът __invoke се извиква, когато даден скрипт направи опит да извика обект, както се извиква функция.

Пример 5:

```
<?php
class Customer {
     function __invoke($name) {
    var_dump($name);
  }
```

```
}
$c = new Customer();
$c("Ivan Ivanov");
?>
```

Резултат:

string(11) "Ivan Ivanov"

2. Сесии и бисквитки

Сесиите и бисквитките са предназначени за съхранение на сведения за потребителите, при преход между няколко страници или при последващо посещение на сайта.

При използване на сесии, данните за сесията (сесийните променливи) се съхраняват във временни файлове на сървъра, докато данните за бисквитките (файловете с cookies) се съхраняват на компютъра на потребителя при първото поискване на web страниците, като при следваща заявка на съответните web страници, биват изпращани от браузъра на сървъра.

Използването на сесии и бисквитки е много удобно и оправдано в такива приложения като Интернет-магазини, форуми, обяви и др., за да съхраняват информация за потребителя (например такава като: username и password), така щото да не се налага на потребителя да въвежда тази информация всеки път, когато минава от страница на страница, или при следващо посещение на същия сайт.

За да се съхрани информация за вашата PHP сесия, програмистът трябва да стартира такава сесия. Сесийните променливи съдържат информация за един единствен потребител и са валидни за всички страници на едно приложение.

2.1. Сесии

2.1.1. Стартиране на сесия

Стартирането на сесия става с функция session_start, която се извиква в началото на PHP-сценария:

```
session_start();
```

Забележка: Тази функция (session_start) трябва да се напише преди <html> tag, тоест:

```
<?php session_start(); ?>
< html>
< body>
...
< /body>
< /html>
```

При стартиране на сесия (session_start), PHP интерпретаторът извършва следното:

- Проверява, съществува ли идентификатор на сесията, и ако не, то го създава. Ако идентификаторът на текущата сесия вече съществува, то се зареждат регистрираните променливи на сесията.
- Ако в заявката няма включен сесиен идентификатор, то интерпретаторът генерира уникален низ от най-често 32 шестнадесетични символа, който се използва за сесиен идентификатор:
- Създава се локално (на сървъра) сесиен файл с име sess_<идентификатор на сесията>, в който ще се съхраняват сесийните данни.

- При следващият Http отговор, интерпретаторът изпраща хедър Set_Cookie, тоест “бисквитка”, с идентификатора на сесията. Ако по някаква причина работата на браузърът с бисквитки е забранена, тогава сесийният идентификатор се разпространява заедно с адреса. Чрез тази “бисквитка” ще се разпознае клиента при следващите си обръщения към сървъра.

Пример 6:

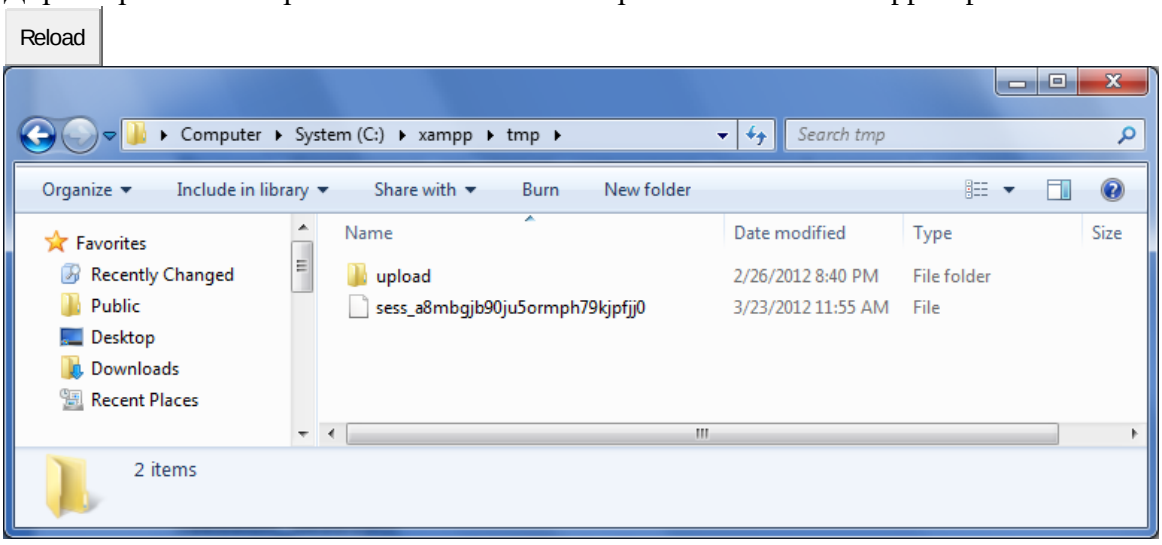
Стартиране на сесия, извеждане идентификатора на сесията и директория за съхраняване на сесийните променливи.

```
<?php session_start();?>
<html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8" />
<title>Session_example</title>
</head>
<body>
<?php
echo "Session id=".session_id()."<br>";
echo "Директорията за съхраняване на сесийните променливи е
".session_save_path();
//session_destroy();
//echo "Session id=".session_id()."<br>";
?>
<form>
<br><input type="submit" value="Reload">
</form>
<body>
<html>
```

Резултат:

Session id=a8mbgjb90ju5ormph79kjpfjj0

Директорията за съхраняване на сесийните променливи е C:\xampp\tmp



Пример 7:
Промяна на директорията за съхраняване на сесийни променливи:

```
<?php
session_save_path("F:\\session_data");
session_start();
echo "Session id=".session_id()."<br>";
session_destroy();
echo "Session id=".session_id()."<br>";
?>
<form>
<br><input type="submit" value="Reload">
```

2.1.2. Регистрация (съхраняване) на сесийни променливи

От версия PHP 4.2.0, нататък, сеансовите променливи винаги се регистрират в глобалните асоциативни масиви $_SESSION или $HTTP_SESSION_VARS :
$_SESSION['username'] = "Maria";
//добавя се елемент в асоциативния масив $_SESSION
или **$HTTP_SESSION_VARS['username'] = " Maria";**
Оттук нататък, ще използваме само $_SESSION, като най-съвременен подход за регистрация на сеансови променливи.
Дали са регистрирани сесийни променливи?
За коректност на работата на приложението е необходимо да се провери дали са регистрирани променливи на сесията.
За целта е достатъчно да се проверят елементите на асоциативния масив ($HTTP_SESSION_VARS или $_SESSION) по следния начин:
if (isset($_SESSION['username']))

Пример 8: **Създаване на елементарен брояч на страници**
Използвана е isset() function, която да провери дали е регистрирана сесийната променлива "views ". Ако е регистрирана, то увеличаваме брояча. Ако не е регистрирана - то я създаваме, със стойност 1:

```
<?php
session_start();
if(isset($_SESSION['views']))
 {
 $_SESSION['views']=$_SESSION['views']+1;
 echo "Views=". $_SESSION['views'];
 }
 else
 {
 $_SESSION['views']=1;
 echo "Views=". $_SESSION['views'];
 }
 ?>
```

```
<form>
<br><input type="submit" value="Reload">
```

Ако регистрацията е чрез използване на асоциативни масиви (в частност - $_SESSION), то за раз-регистриране на сесийните променливи се използва функция unset() по следния начин:
unset($_SESSION["username"]);

Пример 9:
Стартиране на елементарна сесия, работеща с 3 web страници (index.php, page2.php и page3.php).
Когато потребителят посети първата web страница (index.php) се открива сесия и се регистрира променлива **$username**.

index.php
```
<?php
  session_start();
  $_SESSION['username'] = "Maria";
  echo 'Hello, '.$_SESSION['username']."<br>";
?>
<a href="page2.php">Go to next page </a>
```

page2.php
```
<?php
session_start();
  echo $_SESSION['username'].' , This is page 2 of the site!';
  echo("<br>");
?>
  <a href="page3.php">Next page - 3 </a>
```

Забележка: HTML таг <a> обикновено указва link или hyperlink. Атрибут href специфицира дестинацията на връзка, например:
Visit W3Schools

page3.php
```
<?php
session_start();
echo 'Hello, '.$_SESSION['username'];
unset($_SESSION['username']); // разрегистраме променливата
echo 'Your session is unset, '.$_SESSION['username'];
```

```
  /* Сега вече името на потебителя не се извежда */
session_destroy(); // разрушаване на сесия
?>
```

Обяснение:

Създали сме три файла: index.php, page2.php и page3.php. Ако не използваме session_start(), web сървърът по никакъв начин няма да разбере, че http заявките за 3-те файла са изпратени от един и същи потребител.

В случая обаче и 3-те файла започват със session_start(), така при достъп до първия файл, интерпретаторът създава нова сесия. След поискване на 2-ра или 3-та страници, браузърът ще включи към заявката и идентификатора на сесията (чрез хедър Set_Cookie). Така session_start() функциите за 2-ра или 3-та страници вместо да инициират нова сесия ще продължат да използват наличната, поради наличие на валиден сесиен идентификатор в http заявките.

2.2. Бисквитки

"Бисквитката" е малък файл с информация, от вида "име-значение", (най-често за идентифициране на потребителя), който файл сървърът вгражда на компютъра на потребителя при (първото) поискване на дадена web страница.

Така, всеки следващ път, когато потребителят поиска същата web страница с браузъра на своя компютър, потребителят изпраща и бисквитката, като част от своето искане към сървъра.
С PHP, може едновременно да създаде и да се извлече стойността на бисквитката.

В допълнение към основната информация, която съхранява бисквитката ("име-значение"), всяка бисквитка има и набор от атрибути:

- изтичане на срока на годност,
- валиден домейн,
- валиден път на домейн и
- възможно - флаг за сигурност.

Тези атрибути помагат да се гарантира, че браузърът изпраща правилните "бисквитки", когато се подава заявка до сървъра.

2.2.1. Създаване на бисквитка

Създаването на бисквитка става с помощта на функция setcookie. Тази фунция след създаване на бисквитка я изпраща в заглавната част на HTTP отговора, който web сървърът изпраща към потребителския браузер:

bool setcookie (string name [, string value [, int expire [, string path [, string domain [, int secure]]]]])

Функция setcookie има следните аргументи:

- **name** - име на създаваната cookie;
- **value** - значение, съхранявана в cookie с име $name;
- **expire** – задава времето в секунди, след изтичането на което, бисквитката става невалидна (интервалът се добавя към текущото време time()).
- **path** – път в URL адреса, с който се асоциира бисквитката;
- **domain** – домейн, с който се асоциира бисквитката;

- **secure** – определя дали бисквитката може да се изпраща само по криптиран канал за комуникация (протокол HTPPS). По подразбиране тази директива има стойност 0, което означава възможност за достъп по обичайна заявка HTTP.

Стойността (value) на създадената бисквитка автоматично се кодира към URL-a (URLencoded), когато се изпраща бисквитката (от браузъра) и автоматично се декодира когато се получава от браузъра (за да се избегне URLencoding, може да използвате setrawcookie() вместо setcookie).

2.2.2. Задаване на срок на годност

По подразбиране cookies се създават за един сеанс на работа с браузъра, но може да им зададем по дълъг срок.

Това е много удобно за потребителя: ако предостави свои данни при посещение на даден сайт, тогава няма да му се налага да ги задава при всяко посещение на сайта.

Пример 10:

Създаване на елементарен сценарий, който да дава възможност да се отчита с помощта на cookies количеството на обръщенията на посетителя към страница.
Използване на асоциативния масив $_COOKIE.

```
<?php
 $_COOKIE["counter"];
 setcookie("counter",6);
 echo 'The page is visited by '.$_COOKIE['counter'].' times!!!';
?>
 // The page is visited by 6 times!!!
```

2.2.3. Изчистване стойността на бисквитка

Извиква се функция setcookie и се предава само името на бисквитката, чиято стойност трябва да се изчисти: setcookie("name")

Пример 11:

```
<?php
 setcookie("c_name", "Adam", time()+60); //създаване на бисквитка c_name,
със срок 1 минута
  echo $_COOKIE["c_name"];
//извеждане на бисквитка
 ?>
// Adam
```

2.2.4. Четене на достъпните бисквитки чрез $_COOKIE.

Пример 12:

```
<?php
echo $_COOKIE["user"]; // Отпечатване на стойността на бисквитка user
print_r($_COOKIE); // Отпечатване на всички бисквитки
 ?>// AdamArray ( [counter] => 6 [user] => Adam [PHPSESSID] =>
//adsug8rn7gqv7rug6n7j21q7h9 )
```