УДК 004

WEB-ТИРКЕМЕЛЕРДИН КОНФИГУРАЦИЯ ФАЙЛДАРЫ МЕНЕН ИШТӨӨ КУРАЛДАРЫ

Саитмамат кызы Рысгул - ОшМУ, магистрант
Талантбек кызы Канзада - ОшМУ, магистрант

Аннотация. *Бул макалада коддор менен иштөө, чоң көлөмдүү маалыматтарды сактоо жана аларды web сервердик тиркемелерде иштетүү мүмкүнчүлүктөрү жана аларды программалаштыруу жолдору каралган.*

Негизги сөздөр: *require_once, PEAR, parse_ini_file (),Config, XML*

ИНСТРУМЕНТЫ ДЛЯ РАБОТЫ С КОНФИГУРАЦИОННЫМИ ФАЙЛАМИ

Саитмамат кызы Рысгул - ОшМУ, магистрант
Талантбек кызы Канзада - ОшМУ, магистрант

Аннотация *. В этой статье обсуждаются возможности работы с кодом, хранения больших объемов данных и их обработки в приложениях веб-сервера, а также способы их программирования.*

Ключевые слова: *require_once, PEAR, parse_ini_file (),Config, XML*

WORKING INSTRUMENTS WITH CONFIGURATED FILES

Saitmamat kyzy Rysgul - OshMU, master student
Talantbek kyzy Kanzada - OshMU, undergraduate

Annotation. *This article discusses the possibilities of working with code, storing large volumes of data and their processing in web server applications, as well as their programming capabilities.*

Keywords: *require_once, PEAR, parse_ini_file (),Config, XML*

Алгачкы маалыматтар. Албетте, сиз буга чейин иштеп чыккан тиркеме башка долбоордо кайрадан колдонула турган жагдайга туш болдуңуз. Албетте, алгач бул эч кандай кыйынчылыктарды жаратпайт деп ойлогонсуз. Болгону, кодду бир каталогдон башкасына көчүрүү керек! Убакыттын өтүшү менен, сиз долбоорлор бири-биринен ар кандай параметрлер боюнча айырмаланышы мүмкүн экендигин, ал тургай, анча-мынча мааниге ээ болгонун байкадыңыз. *Мисалы*, ал билдирүүлөр жөнөтүлгөн электрондук почта дареги болушу мүмкүн. Мындай учурда, редактордо көптөгөн файлдарды ачуудан жана табуу / алмаштыруу функциясын колдонуп, керектүү электрондук почтаны киргизип, алардын мазмунун өзгөртүүдөн башка арга жок. Бул макалада өзүңүздү мындай иштен кантип куткарып алсаңыз болот, ошондой эле конфигурациялык файлдарды түзүү жана окуу үчүн бир катар кошумча куралдарды сунуштайбыз.

Кодду кайра колдонуу. Компьютер адамды ашыкча жумуштан куткаруу максатында ойлоп табылган. Компьютердик технологиянын өнүгүшү адамдардын компьютерге барган сайын аз убакыт сарптоого умтула баштагандыгына алып келди. Сиз программистсиз дейли. Компьютерсиз сиз жумушсуз калмаксыз. Бирок, ошол эле учурда, сиз компьютердин жардамы менен күнүмдүк ишиңизди жөнөкөйлөтүүгө аракет кылып жатасыз, ушул максатта, мисалы, редактордогу кодду толтуруу функциясын колдоносуз. Биз сиз жараткан кодду модификациялоо иштери минималдаштырылышы үчүн уюштуруу керек деген ойго жетелегибиз келет. Бул көбүнчө форманы түзүү жана чийүү, ошондой эле электрондук почта билдирүүлөрүн жөнөтүү сыяктуу күндөлүк иш-аракеттерди

автоматташтырган код түзгөндө жасалат. Бирок, ар кандай тиркемелерде күнүмдүк иш-аракеттерди жүзөгө ашыруу функциялары эч качан 100% окшош эместигин унутпаңыз. Бир форма ар башка, ал эми электрондук почта билдирүүлөрү ар башка адресаттарга арналган. Бирок, тиркеменин деңгээлиндеги логика өзгөрүүсүз калат, функциялар бири-биринен айрым параметрлеринде гана айырмаланат. Демек, сиз өзүңүздүн максатыңыз жөнүндө так билишиңиз керек - тышкы параметрлерге коюлуучу кодду иштеп чыгуу.

Модулдук уюм. Бул көйгөйдү чечүү үчүн, колдонмоңуздун түзүмүн пландаштырууда, модулдуулук жөнүндө кам көрүшүңүз керек. Башкача айтканда, көп колдонулган функцияларды же класстарды өзүнчө файлга жайгаштырышыңыз керек, алар **require_once** аркылуу туташтырылат. Мындай учурда, арыздын файлдары ашыкча код менен толтурулбайт. Сиз журнал журналына көп жазасыз дейли. Мындай учурда, бул операцияны аткарган кодду класска же функциянын алкагына кошуп койсоңуз жакшы болмок. **PEAR** сыяктуу кээ бир булак коддорунун китепканасынан алынган даяр классты колдонсоңуз дагы жакшы болот.

Процедуралык коддун параметрлери. Кодду анализдеп, кайталанган фрагменттерди тандап, класстарга жана функцияларга бөлүштүргөндөн кийин, керектүү параметрлерди бөлүп көрсөтүү жөнүндө ойлонушуңуз керек, алардын мааниси сырттан орнотулат. Процедуралык код жөнүндө сөз болгондо, эң жөнөкөй чечим - глобалдык өзгөрмөлөрдү колдонуу, алар өзүнчө файлда аныкталууга тийиш. Бул келечекте алардын баалуулуктарын көйгөйсүз өзгөртүүгө мүмкүнчүлүк берет.

1-тизмеде электрондук почта билдирүүлөрүн жөндөө функциясы көрсөтүлөт. Анын корпусунда бир гана PHP функциясы бар - mail (). Ошентип, билдирүү жөнөтүүдө ар бир жолу алуучуну көрсөтүү зарылдыгынан арылабыз. Кийинки биз аныктаган өзгөрмө - билдирүү предметинин алдындагы префикс. Require_once аркылуу байланышкан конфигурация файлы төмөнкүлөргө окшош болушу мүмкүн.

Listing 1

```
<?PHP
  $to = 'webmaster@localhost';
  $prefix = '[Testinstallation] ';
 ?>

<?PHP
  function sendMail( $subject, $body )
  {
    global $to, $prefix;
    $subject = $prefix . $subject;
    return mail( $to, $subject, $body );
  }
  require_once 'config1.php';
  sendMail( 'Test', 'Это тестовое сообщение.' );
 ?>
```

Андан да жакшы жолу бар.

Жогоруда талкууланган ыкма натыйжалуу болсо дагы, бул мыкты чечим эмес. Сиздин колдонмо кодуңуз татаалдашып, параметрлердин саны өскөн сайын, төмөнкү көйгөйлөр келип чыгышы мүмкүн:

✓ Биз колдонгон глобалдык өзгөрмөлөр ат мейкиндигиндеги карама-каршылыктарды жаратышы мүмкүн.

✓ Эгер конфигурация файлдары программист тарабынан эмес, үйрөнчүк тарабынан оңдолсо, тутумда синтаксистик каталар пайда болушу мүмкүн, мисалы, жабык цитаталардан улам.

✓ Ар кандай өзгөрмөлөргө жетүү үчүн **$ _GLOBALS** массивине кайрылуу керек.

PHP модулдарынын ордуна, ышкыбоздор оңой түшүнүп, өзгөртө турган башка форматтар, ошондой эле php скрипттери бар. Биз эки форматты билдирет: Windows иштөө тутуму тарабынан кеңири колдонулган **ini**-файлдар, ошондой эле **XML** формат.

PHPде ini файлдарын көйгөйсүз окуган **parse_ini_file** () функциясы бар. Бул файл абдан жөнөкөй түзүлүшкө ээ. Ар бир параметрге бир гана маани берилиши мүмкүн, ал эми барабар белгиси дайындоо оператору катары колдонулат. Мурунку мисалдагы конфигурация файлы **ini** форматында ушундай көрүнөт.

```
to = "webmaster@localhost"
prefix = "[Testinstallation] "
```

Parse_ini_file () функциясына параметр катары берилген ini файлын окугандан кийин, биз төмөнкүдөй ассоциативдик массивди алабыз:

```
   Array
(
 [to] => webmaster@localhost
 [prefix] => [Testinstallation]
)
```

Тизме 2 **ini** негизиндеги почта билдирүүлөрүн тапшыруу функциясын камтыйт:

Listing 2

```
<?PHP
 function sendMail( $subject, $body )
 {
  $config = parse_ini_file( 'config1.ini' );
  $to = $config['mail_to'];
  $prefix = $config['mail_prefix'];
  $subject = $prefix . $subject;
  return mail( $to, $subject, $body );
 }
 sendMail( 'Test', 'Это тестовое письмо.' );
?>
```

Эгер сиз **parse_ini_file ()** функциясынын документтерин окуп чыккан болсоңуз, анда ал дагы экинчи параметрге өтүшү мүмкүн экендигин байкайсыз. Эгер **ini** файлын бир нече бөлүмгө же бөлүмгө бөлгүңүз келсе, анда ал талап кылынат. Бир нече электрондук почта орнотууларын сактоо керек деп коёлу. Ошондо **ini** файлы төмөнкүдөй болот:

```
    [errors]
to = "webmaster@localhost"
prefix = "[Testinstallation] "
[contact]
to = "kunde@test.de"
prefix = "[Contact] "
```

Эгерде сиз **parse_ini_file ()** чакырганда экинчи параметр катары true деп өтсөңүз, анда php бөлүм файлынан издеп, андан кийин ар бир бөлүм (errors жана contact) белгилүү бир мааниге туура келген көп өлчөмдүү массивди кайтарат:

```
    Array
(
 [errors] => Array
 (
```

```
    [to] => webmaster@localhost
    [prefix] => [Testinstallation]
  )
  [contact] => Array
  (
    [to] => kunde@test.de
    [prefix] => [Testinstallation]
  )
)
```

Ini-файлдарындагы өзгөчө маанилер. Ini-файлдарын колдонууда, кээ бир өзгөчө маанилерди саптар менен чагылдырууга болорун унутпаңыз. Варианттын маанисин чыныгы же ооба (тырмакчасыз) деп көрсөтсөңүз, мындай учурда алар автоматтык түрдө 1 санына, ал эми **false** же **no** деген сап бош сапка айланат. Тилекке каршы, бул эч кандай ката кетирбейт.

Listing 3

```
<?PHP
  $config = parse_ini_file( 'config2.ini', true );
  echo '<pre>';
  print_r( $config );
  echo '</pre>';
?>
```

Коопсуздук. Эгер сиз конфигурация файлы сырсөздөр сыяктуу купуя маалыматтарды сактоо үчүн колдонулса, анда мындай файлдын мазмуну web- браузерге түшүп калбашы үчүн кам көрүү керектигин түшүнүшүңүз керек. Жөнөкөй жол - конфигурация файлдарын сайттын түпкү каталогунан тышкары сактоо, мисалы, бул жерде: **/ etc / myApp / config**

Эгер бул мүмкүн эмес болсо, анда файл кеңейтүүсүн өзгөртө аласыз. **PHP** модулунун форматындагы конфигурация файлы үчүн ар дайым. **PHP** кеңейтүүсүн тандашыңыз керек. Бул учурда сервер php файлын талдайт жана колдонуучу бош баракты көрөт. Бул **ini** файлдары менен иштебейт, бирок **Apache** сервери маалыматтарды коргоо мүмкүнчүлүгүн берет. **Ini** файлы сакталган каталогго **.htaccess** деген файлды жайгаштырсаңыз болот, ага төмөнкү саптарды жайгаштырыңыз:

```
<Files *.ini>
Order deny,allow
Deny from all
</Files>
```

Башка каражаттар. Албетте, веб-тиркеменин жөндөөлөрүндө ийкемдүүлүктү камсыз кылуу көйгөйүнө туш болгон жалгыз гана иштеп чыгуучу сиз эмессиз. Ошондуктан, кээ бир программисттер конфигурация файлдары менен иштөөнү абстракттуу деңгээлге жеткирүүчү, ошондой эле конфигурация файлдарынын ар кандай форматтарын жазууну жана окууну жөнөкөйлөтүүчү класстык китепканаларды иштеп чыгышкан.

PEAR::Config

Конфигурациялык файлдарды окууда жана жазууда ыңгайлуу боло турган класстардын бири **PEAR :: Config [3].** Бардык **PEAR** класстары сыяктуу эле, **PEAR::** Config буйрукту колдонуп **PEAR-Installer** менен орнотулган

```
pear install Config
```

Бул класс көп форматтуу, анткени ал **XMIL, ini, Apach-Style** (XML жана ini гибриддери) форматтарындагы конфигурация файлдары, ошондой эле php массивдери менен иштейт. Бул класстын артыкчылыгы - бардык форматтар менен иштешүү үчүн **API** бирдей. Ошол. XML форматындагы конфигурация файлдары менен иштөөнүн логикасы **ini**-файлдар менен иштөөнүн логикасынан айырмаланбайт. Натыйжада, бардык форматтардын бирдей түзүлүшкө ээ болушу шарт. **PEAR:: Config** иштей турган конфигурация файлдары, **ini**-файлдары сыяктуу, бөлүмдөрдөн турат.

Келгиле, дагы бир жолу мисалыбызды өзгөртөбүз. Алгач **Config** объектисин түзүп, андан кийин анын **parseConfig ()** ыкмасын чакырабыз. Метод ар кандай файл форматтарын окууга мүмкүнчүлүк бергендиктен, ага чалып жатканда форматты көрсөткөн параметрди тапшырышыңыз керек. **Ini** форматындагы конфигурация файлдары үчүн iniFile сабы ушундай параметр катары колдонулат. Файлды окугандан кийин, биз параметрлерди массив түрүндө албайбыз, анын ордуна бардык орнотууларга мүмкүнчүлүк берген контейнер объектиси түзүлөт. Көпчүлүк учурларда варианттарды массив түрүндө алуу максатка ылайык. Бул үчүн **toArray ()** ыкмасы колдонулат. 4-тизме **ini-файлын** окугандыгын көрсөтөт:

Listing 4

```
<?PHP
  require_once 'Config.php';
  $config = new Config();
  $root =& $config->parseConfig('config2.ini', 'IniFile');
  $settings = $root->toArray();
  echo "<pre>";
  print_r( $settings );
  echo "</pre>";
?>
```

Бир караганда, бул бир аз түшүнүксүз сезилиши мүмкүн. Бирок, бул ыкманын артыкчылыгы, бир эле ыкма **PEAR :: Config** тарабынан колдоого алынган бардык файл форматтарын окуу үчүн колдонулат. Өзгөртүлгөн параметрлерди каалаган форматта сактоого болот:

```
<?php
  $root->writeDatasrc( 'config2.conf', 'Apache' );
?>
```

5-тизмеде бир катар варианттар массивге жайгаштырылган код бар, андан кийин **XML** форматында сакталат. Эгер сиз **PEAR :: Config** жөнүндө көбүрөөк билгиңиз келсе, анда **PEAR** документациясында [5] же **DevShed-Tutorial** [6] боюнча керектүү маалыматты таба аласыз.

Listing 5

```
<?PHP
  require_once 'Config.php';
  $settings = array(
    'errors' => array(
      'email' => 'webmaster@localhost',
      'prefix' => '[ERRORS]',
    )
  );
```

```
$config = new Config();
$root =& $config->parseConfig($settings, 'PHPArray');
$root->writeDatasrc( 'config2.xml', 'XML' );
?>
```

patConfiguration

PatConfiguration [7] бул конфигурация файлдары менен иштөөнүн альтернативдүү классы, бирок ал **XML** форматындагы файлдар менен иштөөгө гана арналган. Архивди жүктөп алгандан кийин, аны таңгактан чыгаруу керек. Класс өзү камтыган каталогдо жайгашкан. **patConfiguration** алдын-ала **Tag-Set**ти аныктайт, андан кийин маалыматтар менен толтурулат. Мындан тышкары, бул класс варианттын түрүн көрсөтүүгө мүмкүнчүлүк берет: бүтүн, өзгөрүлмө чекит, логикалык. **PatConfiguration** тарабынан түзүлгөн типтүү конфигурация файлы төмөнкү структурага ээ:

```
<configuration>
<path name="fehler">
<configValue name="email" type="string">webmaster@localhost</configValue>
<configValue name="priority" type="float">100</configValue>
</path>
</configuration>
```

Класс объектиси түзүлгөндөн кийин**, parseConfigFile ()** ыкмасын чакырса болот. Параметрлерге **getConfigValue ()** аркылуу кирүүгө болот. Параметр катары, бул ыкма керектүү варианттын жолун алат. Келгиле, биздин мисалга кайрылып көрөлү. Ката жөнүндө билдирүү жөнөтүлгөн электрондук почта дарегин алгыбыз келет дейли. Бул учурда, error.email жолу колдонулат. Эгер жол көрсөтүлбөсө, анда бардык параметрлер массивге өткөрүлүп берилет. 6-тизмеде файлдарды окуу үчүн колдонула турган код көрсөтүлөт.

patConfiguration 2.0.0

Учурда **PatConfiguration** программасынын көп форматтуу версиясы иштелип жатат. Балким, макала жарыяланганда, бул версия буга чейин жеткиликтүү болуп калышы мүмкүн. Бирок, иштеп чыгуучулар үчүн акыркы нусканы **snaps.php-tools.net/downloaden** сайтынан жүктөп алсаңыз болот.

Бул мисалда, буга чейин маанинин түрү тегдин ичинде көрсөтүлгөнүн байкадыңыз. Түр аттары **php settype ()** функциясында колдонулган аталыштарга окшош. Эгерде эч кандай түр көрсөтүлбөсө, анда маани сап катары чечмеленет. Көп колдонулган варианттар үчүн, айрымдарын аныктай аласыз.

```
<configuration>
<define tag="priority" type="float"/>
<define tag="email" type="string"/>
<path name="fehler">
<email>webmaster@localhost</email>
<priority>100</priority>
</path>
</configuration>.
```

GetConfigValue функциясы менен катар, **setConfigValue ()** функциясы бар, анын жардамы менен параметрдин маанисин өзгөртө аласыз. Андан кийин конфигурация файлын **writeConfigFile ()** жардамы менен кайра жазууга болот (Тизме 7ди караңыз).

patConfiguration дагы бир катар кошумча функцияларды сунуш кылат. Мисалы, тегдер менен катар атрибуттарды жана аталыштар мейкиндигин аныктоого болот (Namespace), ошондой эле тышкы файл биркага тиркелиши мүмкүн, ошондуктан параметрлер бир нече файлга бөлүштүрүлөт. Мындан тышкары, patConfiguration кэш тутумун камтыйт, анын жардамы менен конфигурация файлын бир нече жолу окуунун кажети жок.

PHP Application Tools-Homepage жана **DevChed**'тен **patConfiguration-Tutorial** жөнүндө көбүрөөк маалымат таба аласыз [8].

Listing 7

```
<?PHP
  require_once 'include/patConfiguration.php';
  $config = new patConfiguration(
    array(
      "configDir" =>"./",
      "errorHandling" =>"nice_die"
    )
  );
  $config->parseConfigFile( 'config6.xml' );
  $config->setConfigValue( 'errors.email', 'errors@localhost' );
  $config->writeConfigFile( 'config6_new.xml', 'xml', array( 'mode' => 'pretty' ) );
?>
```

Корутунду

Колдонмонун жөндөөлөрүнүн ийкемдүүлүгүн эске алуу менен, айрыкча анын компоненттери башка долбоорлордо колдонулушу керек болсо, көп убакытты үнөмдөйт. Конфигурация файлдары менен иштөөнү даяр класстардын бирине тапшырсаңыз, сиз андан дагы аз убакытты коротосуз. **PEAR :: Config** жана **patConfiguration** ортосунда тандоо тапшырмага жараша болот. **PEAR :: Config** программасынын артыкчылыгы - ар кандай конфигурация файл форматтарын колдоо, ал эми **patConfiguration XML** менен жакшы иштейт жана бир катар кошумча функцияларды камсыз кылат. Бирок, 2.0.0 версиясы пайда болгондо, бул топтом **ini** жана **wddx** файлдарын окуу үчүн бирдей **API**ге ээ болот. **PHP** массивдери учурдагы нускасында буга чейин колдоого алынган.

Шилтемелер

[1] PEAR, pear.php.net/
[2] Функция parse_ini_file, de2.php.net/manual/de/function.parse-ini-file.php
[3] PEAR::Config, pear.php.net/package/Config
[4] Введение в PEAR, PHP Magazin 1.02
[5] PEAR::Config Dokumentation, pear.php.net/manual/en/package.-configuration.config.intro.php
[6] PEAR::Config Tutorial, www.devshed.com/c/a/PHP/Configuration-Manipulation-With-PHP-Config/
[7] patConfiguration, www.php-tools.net/
[8] patConfiguration-Tutorial, www.devshed.com/c/a/PHP/Easy-Application-Configuration-With-patConfiguration