



NAPREDNO WEB PROGRAMIRANJE

PHP

PHP I SERVER

- PHP skripta često komunicira sa aplikacijama na serveru kao što su baze podataka, slanje elektroničke pošte i slično
- PHP se koristi i za napredne funkcije, a ne samo za generiranje web sadržaja
- Pokazati ćemo upotrebu serverskih biblioteka kao što je sažimanje datoteka, automatsko pokretanje PHP skripti upotrebom cron-a
- Također ćemo pokazati primjer kriptografije
- Za sažimanje koristiti ćemo Zlib biblioteku koja je dostupna na www.zlib.net



PHP I SERVER

- Zlib je često već ugrađen u konfiguraciju servera pa ga nije potrebno instalirati
- Nakon instalacije biblioteke PHP može čitati i pisati u komprimirane datoteke
- Većina funkcija su kao standardne funkcije: **fopen()**, **fwrite()**, **fclose()**, itd.
- U sljedećem primjeru radimo backup baze podataka
- Svi podaci iz tablica u bazi podataka spremaju se u .txt datoteku i nakon toga sažimaju



PHP I SERVER

```
<?php
```

```
/*Skripta prima podatke iz baze podataka, sprema ih u txt datoteku i sažima pomoću biblioteke zlib */
```

```
//Naziv baze podataka
```

```
$db_name = 'wordpress';
```

```
//Direktorij za backup
```

```
$dir = "backup/$db_name";
```

```
//Ako direktorij ne postoji stvori ga
```

```
if (!is_dir($dir)) {
```

```
    if (!@mkdir($dir)) {
```

```
        die("<p>Ne možemo stvoriti direktorij
```

```
$dir.</p></body></html>");
```

```
    }
```

```
}
```



PHP I SERVER

//Trenutno vrijeme

```
$time = time();
```

//Spoj na bazu podataka

```
$dbc = @mysqli_connect('localhost', 'root', '', $db_name) OR  
die("<p>Ne možemo se spojiti na bazu  
$db_name.</p></body></html>");
```

//Pokaži sve tablice iz baze podataka

```
$r = mysqli_query($dbc, 'SHOW TABLES');
```

//Radimo backup ako postoji barem jedna tablica

```
if (mysqli_num_rows($r) > 0) {
```

//Poruka da se radi backup

```
echo "<p>Backup za bazu podataka '$db_name'.</p>";
```



PHP I SERVER

```
//Dohvati ime svake tablice
while (list($table) = mysqli_fetch_array($r,
MYSQLI_NUM)) {
    //Dohvati podatke iz tablice
    $q = "SELECT * FROM $table";
    $r2 = mysqli_query($dbc, $q);
    //Ako postoje podaci
    if (mysqli_num_rows($r2) > 0) {
        //Otvori datoteku
        if ($fp = gzopen ("$dir/{$table}_{$time}.sql.gz", 'w9')) {
            //Dohvat podataka iz tablice
            while ($row = mysqli_fetch_array($r2,
MYSQLI_NUM)) {
```



PHP I SERVER

```
// Zapiši podatke razdvajajući ih sa zarezom
    foreach ($row as $value) {
        $value = addslashes($value);
        gzwrite ($fp, "$value' ");
    }
    // Novi redak za svaki redak iz baze
    gzwrite ($fp, "\n");
} // Kraj while petlje
// Zatvori datoteku
gzclose ($fp);
// Ispiši da je backup uspješno izvršen
echo "<p>Tablica '$table' je pohranjena.</p>";

} else { // Ne možemo stvoriti datoteku
```



PHP I SERVER

```
        echo "<p>Datoteka $dir/{\$table}_{\$time}.sql.gz se  
ne može otvoriti.</p>";  
        break; //Prekini while petlju  
    } // Kraj gzopen()  
  
    } //Kraj mysqli_num_rows()  
  
    } //Kraj while petlje  
  
} else {  
    echo "<p>Baza $db_name ne sadrži tablice.</p>";  
}  
  
?>
```



PHP I SERVER

- Cron je servis na Unix serverima koji omogućuje automatsko izvođenje zadataka
- Cron aplikacija je stalno pokrenuta i izvodi narudžbe kako joj ih zadamo
- Ove narudžbe su spremljene u datoteci **crontab**
- To je lista za izvođenje koja sadržava linije kao što je:

```
30 23 * * * wget -q http://www.ferit.hr
```

- Svaka linija mora imati šest polja odvojenih razmakom (minute, sati, dani, mjeseci i dan u tjednu (dani od 0 do 6))
- Zvjezdica znači da vrijednost nije ograničena



PHP I SERVER

- Gornja naredba kaže da se naredba izvede u 23:30 svaki dan u mjesecu, svaki mjesec u godini
- Za više mjeseci možete staviti na 1-6 (od prvog do 6 mjeseca u godini)
- Ili lista odvojena zarezima (1,2,5)
- Šesti parametar je `wget` naredba da otvori stranicu www.ferit.hr, `-q` znači u pozadini
- Cron možete postaviti pomoću CLI-a na serveru ili preko cPanel sučelja
- Preko CLI-a crontab datoteka se otvara naredbom `crontab -e`



PHP I SERVER

- Crontab datoteku zatim uređujete s nekim od editora
- Za trenutni sadržaj datoteke `crontab -l`
- Trenutni sadržaj potrebno je kopirati u novu datoteku jer ćemo ga inače prebrisati
- Prvo je potrebno stvoriti datoteku u koju dodamo sljedeći sadržaj (svaki petak u 21:10)
10 21 * * 5 curl http://localhost/db_backup.php
- Nakon toga potrebno je spremiti crontab sa nazivom `mojcron` i reći serveru gdje je lokacija datoteke:
- U CLI utipkati **`crontab /path/to/mojcron`**



PHP I SERVER

- Izmjene možete provjeriti sa `crontab -l`
- Crontab datoteka je jedinstvena za svakog korisnika i instrukcije će se izvoditi kao da to radi korisnik
- Potrebno je voditi brigu o restrikcijama koje ima korisnik
- Na Windows serverima umjesto cron možete koristiti `scheduled tasks`



PHP I SERVER

- Često je potrebno da se zbog sigurnosti kriptiraju i dekriptiraju podaci
- Biblioteka **OpenSSL** pruža takvu mogućnost
- Za korištenje sa PHP-om potrebno ju je instalirati ako već ne dolazi sa konfiguracijom servera
- Pokazati ćemo primjer kriptiranja podataka i spremanje u session varijable
- Nakon što se kriptiraju podaci u binarnom obliku funkcijom **base64_encode()** ih pretvaramo u tekstualni oblik



PHP I SERVER

- Kriptiranje kreće s time da odaberemo algoritam kriptiranja
- OpenSSL dolazi sa mnoštvom algoritama koje možete saznati u dokumentaciji
- Potrebno je napraviti inicijalizacijski vektor to jest **IV**
- Za ovu metodu inicijalizacijski vektor treba biti dužine 16 bajta što se postiže :

```
$cipher = "AES-128-CTR";
```

```
$iv_length = openssl_cipher_iv_length($cipher);
```

```
$encryption_iv = random_bytes($iv_length);
```



PHP I SERVER

```
<?php
```

```
/*Kriptiranje podataka i spremanje u session varijable*/  
session_start(); ?>
```

```
<?php
```

```
//Ključ za enkripciju, 256 bitni
```

```
$encryption_key = md5('jed4n j4k0 v3l1k1 kljuc');
```

```
//Podaci za enkripciju
```

```
$data = 'Ovo su podaci koje želimo kriptirati';
```

```
//Odaber cipher metodu AES
```

```
$cipher = 'AES-128-CTR';
```

```
//Stvori IV sa ispravnom dužinom
```

```
$iv_length = openssl_cipher_iv_length($cipher);
```

```
$options = 0;
```



PHP I SERVER

```
// Non-NULL inicijalizacijski vektor za enkripciju
// Random dužine 16 byte
$encryption_iv = random_bytes(iv_length);
// Kriptiraj podatke sa openssl
$data = openssl_encrypt($data , $cipher,
    $encryption_key, $options , $encryption_iv );
// Spremi podatke
$_SESSION['podaci'] = base64_encode($data);
$_SESSION['iv'] = $encryption_iv;

// Ispiši kriptirane podatke
echo '<p>Kriptirani podaci su ' . base64_encode($data) .
'</p>';
?>
```



PHP I SERVER

- Proces dekripcije je jako sličan

```
<?php
```

```
/*Dekripcija podataka sa OpenSSL*/
```

```
session_start();
```

```
//Postoje li kriptirani podaci
```

```
if (isset($_SESSION['podaci'], $_SESSION['iv'])) {
```

```
//Stvori ključ
```

```
$decryption_key = md5('jed4n j4k0 v3l1k1 kljuc');
```

```
//Odaber cipher metodu AES
```

```
$cipher = 'AES-128-CTR';
```

```
$options = 0;
```

```
//Dohvati IV i kriptirane podatke
```

```
$decryption_iv = $_SESSION['iv'];
```

```
$data= base64_decode( $_SESSION['podaci'] );
```



PHP I SERVER

// Dekriptiraj podatke:

```
$data = openssl_decrypt($data , $cipher,  
    $decryption_key, $options , $decryption_iv );
```

// Ispiši podatke

```
echo '<p>Dekriptirani podaci su "' . trim($data) . '".</p>';  
} else {  
    echo '<p>Nema podataka.</p>';  
}  
?>
```



PHP I SERVER

- Spomenuti ćemo na kraju kako pokretati naredbe na serveru
- Prvi način je **`exec(command, $output);`**
- Funkcija prima naredbu i u niz `$output` sprema dobivene rezultate
- Funkcija **`system(command);`** daje izlaz
- Funkcija **`passthru(command);`** daje izlaz u binarnom obliku
- Izlaz daju i funkcije **`shell_exec()`** i **`backticks()`**
- Zbog sigurnosti preporuča se koristiti **`escapeshellarg()`** ili **`escapeshellcmd()`** kako bi se mogle koristiti samo hardkodirane naredbe



PHP I XML

- XML (eng. Extensible Markup Language) se koristi za spremanje podataka ili dijeljenje podataka između računala
- XML predstavlja informacije, ali i metapodatke o informacijama
- XML je kao i HTML zasnovan na Standard Generalized Markup Language (SGML)
- Nadgledan je od strane W3C konzorcija i nema predefinirane oznake kao HTML
- XML dokumenti se mogu uređivati u bilo kojem uređivaču teksta
- Za više detalja pogledati literaturu



PHP I XML

- Za parsiranje XML-a koriste se različite biblioteke koje trebaju provjeriti je li dokument:
 - Dobro formatiran
 - Je li validan
 - Dohvatiti podatke
- Parser razdvaja dokument na dijelove prema oznakama i dohvaća ih
- Postoje dva tipa parsera: event-based and tree-based
- Prvi se pokreće na neki događaj kao što je XML oznaka za otvaranje u XML datoteci i čita sadržaj do kraja. Primjer takvog parsera je [Expat](#)



PHP I XML

- Drugi tip parsera čita podatke i sprema ih u stablo kojim se može manipulirati
- Koristi se u DOM sustavima, primjer takvog parsera je [SimpleXML](#)
- Dati ćemo primjere sa oba tipa parsera
- Expat ima 4 koraka:
 1. Stvori novi parser
 2. Odredi funkcije za upravljanje događajima
 3. Parsiraj datoteku
 4. Oslobodi resurse koje je koristio parser



PHP I XML

```
<?php
/*Expat parser*/
//Funkcija koja upravlja oznakom za početak
function handle_open_element($p, $element, $attributes) {
    //Ovisno o oznaci radi sljedeće
    switch ($element) {
        //Oznake su address: book, title, author, year, chapter, and
        //pages!
        case 'BOOK': //Za knjigu stvori div
            echo '<div>';
            break;
        case 'CHAPTER': //Za poglavlje stvori p
            echo "<p>Poglavljje {$attributes['NUMBER']}: ";
            break;
        case 'COVER': //Pokaži sliku
            //Informacije o slici
            $image = @getimagesize($attributes['FILENAME']);
```



PHP I XML

```
//Prikaži sliku u HTML-u
    echo "<img src=\"{" . $attributes['FILENAME'] . "\"
$image[3] border=\"0\"><br>";
    break;
case 'TITLE': //Naslovi su h2
    echo '<h2>';
    break;
//Ostalo samo ispiši
case 'YEAR':
case 'AUTHOR':
case 'PAGES':
    echo '<span class="label">' . $element . '</span>: ';
    break;
} //Kraj switch
}
```



PHP I XML

```
// Funkcija za rukovanje oznakom za kraj  
function handle_close_element($p, $element) {  
    // Ovisno o oznaci radi sljedeće  
    switch ($element) {  
        // Zatvori HTML oznake  
        case 'BOOK':  
            echo '</div>';  
            break;  
        case 'CHAPTER':  
            echo '</p>';  
            break;  
        case 'TITLE':  
            echo '</h2>';  
            break;
```



PHP I XML

```
//Dodaj novi red za ostalo  
    case 'YEAR':  
    case 'AUTHOR':  
    case 'PAGES':  
        echo '<br>';  
        break;  
    } //Kraj switch  
}
```

```
//Ispiši sadržaj  
function handle_character_data($p, $cdata) {  
    echo $cdata;  
}  
//Stvori parser  
$p = xml_parser_create();
```



PHP I XML

```
//Postavi funkcije za rukovanje
xml_set_element_handler($p, 'handle_open_element',
'handle_close_element');
xml_set_character_data_handler($p,
'handle_character_data');

//Pročitaj datoteku
$file = 'books.xml';
$fp = @fopen($file, 'r') or die("<p>Ne možemo otvoriti
datoteku '$file'.</p></body></html>");
while ($data = fread($fp, 4096)) {
    xml_parse($p, $data, feof($fp));
}
//Zatvori parser
xml_parser_free($p);
?>
```



PHP I XML

- Primjer sa SimpleXML parserom

```
<?php
```

```
/*SimpleXML parser*/
```

```
//Učitaj datoteku
```

```
$xml = simplexml_load_file('books.xml');
```

```
//Iteracija kroz XML
```

```
foreach ($xml->book as $book) {
```

```
//Ispiši naslov
```

```
echo "<div><h2>$book->title";
```

```
//Provjeri izdanje
```

```
if (isset($book->title['edition'])) {
```

```
echo " (Izdanje {$book->title['edition']})";
```

```
}
```

```
echo '</h2>';
```



PHP I XML

```
//Ispiši autora
foreach ($book->author as $author) {
    echo "<span class=\"label\">Autor</span>: $author<br>";
}

//Godina izdanja
echo "<span class=\"label\">Godina izdanja:</span> $book-
>year<br>";
if (isset($book->pages)) {
    echo "<span class=\"label\">Stranice:</span> $book-
>pages<br>";
}

//Ispiši poglavlja
if (isset($book->chapter)) {
    echo 'Sadržaj<ul>';
    foreach ($book->chapter as $chapter) {
        echo '<li>';
```



PHP I XML

```
if (isset($chapter['number'])) {  
    echo "Poglavlje {$chapter['number']}: ";  
}  
echo $chapter;  
if (isset($chapter['pages'])) {  
    echo " ({$chapter['pages']} stranica)";  
}  
echo '</li>';  
}  
echo '</ul>';  
}
```



PHP I XML

```
//Naslovnica
if (isset($book->cover)) {

    //Dohvati sliku
    $image = @getimagesize ($book->cover['filename']);

    //Prikaži sliku u HTML-u
    echo "<img src=\"{".$book->cover['filename']}\" $image[3]
border=\"0\" /><br>";

}

//Zatvori div
echo '</div>';
}
?>
```



PHP I XML

- RSS (eng. Really Simple Syndication) se koristi za prikaz sadržaja web stranice
- Lista sadrži naslov, opise i slično
- Korisnici mu mogu pristupiti pomoću RSS klijenta to jest i sa preglednicima
- RSS je XML datoteka sa već uspostavljenim oznakama i počinje sa oznakom rss
- Na idućem slajdu je primjer RSS-a



PHP I XML

```
<?php
/*RSS primjer*/
//Pošalji zaglavlje
header('Content-type: text/xml');
//Početni RSS kod
echo '<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0">
<channel>
<title>Primjer jednostavnog RSS-a</title>
<description>Ovdje postaviti opis stranice ili
sadržaja<description>
<link>www.ferit.hr</link>';
```



PHP I XML

```
//Napravi podatke
$data = array(
    0 => array('title' => 'Diplomski studij', 'description' =>
'Opis diplomskog studija', 'link' =>
'http://www.etfos.unios.hr/studiji/sveucilisni-diplomski-
studij/', 'pubDate' => '1487833804'),
    1 => array('title' => 'Preddiplomski studij',
'description' => 'Opis studija', 'link' =>
'http://www.etfos.unios.hr/studiji/sveucilisni-preddiplomski-
studij/', 'pubDate' => '1487833715'),
    2 => array('title' => 'Stručni studija', 'description' =>
'Stručni studij je...', 'link' =>
'http://www.etfos.unios.hr/studiji/strucni-studij/', 'pubDate' =>
'1487833622')
);
```



PHP I XML

```
//Prolaz kroz podatke
foreach ($data as $item) {
//Svaki zapis
echo '<item>
<title>' . htmlentities($item['title']) . '</title>
<description>' . htmlentities($item['description']) .
'...</description>
<link>' . $item['link'] . '</link>
<guid>' . $item['link'] . '</guid>
<pubDate>' . date('r', $item['pubDate']) . '</pubDate>
</item>
';
}
//Zatvori oznake
echo '</channel>
</rss>
';
```



DEBUGIRANJE I TESTIRANJE U PHP-U

- Nećemo se detaljnije baviti alatima za debugiranje nego ćemo ih samo navesti:
 - Advanced PHP Debugger (www.php.net/apd/)
 - DBG (www.php-debugger.com/dbg/)
 - Xdebug (www.xdebug.org)
- Za korištenje ovih alata trebate imati pristup serveru kako biste ih mogli instalirati
- Na primjer pomoću Xdebug možete provjeriti stanje svih varijabli, izlaze iz funkcija itd.
- Zajedno sa Webgrind možete raditi profiliranje i pronaći dijelove koda koji sporije rade



DEBUGIRANJE I TESTIRANJE U PHP-U

- Malo detaljnije ćemo obraditi **unit testing**
- Ovakav način testiranja koristi pretpostavku da ćete napisati testove koji provjeravaju dijelove Vašeg koda
- Prednosti su:
 - Smanjenje pogreški u kodu
 - Pomažu poboljšati dizajn koda
 - Pomažu pri stvaranju dokumentacije
 - Manja je vjerojatnost da ćete napraviti pogrešku pri izmjeni koda



DEBUGIRANJE I TESTIRANJE U PHP-U

- Za unit testiranje vrijedi sljedeće:
 - Lako se pišu i izvršavaju
 - Trebaju biti što jednostavniji
 - Provjeravaju radi li kod prema očekivanju
 - Ne koriste se za validaciju korisničkog unosa podataka
 - Ne koriste se za rukovanje pogreškama koje se mogu dogoditi pri radu stranice sa korisnicima
- Iz svega navedenog vidimo da se testovi koriste za provjeru pojedinih dijelova programa kako bi se smanjila mogućnost pogreške i neočekivanog ponašanja programa



DEBUGIRANJE I TESTIRANJE U PHP-U

- Testiranje je najlakše provesti sa već gotovim bibliotekama
- Jedna od najboljih je PHPUnit (**www.phpunit.de**)
- Također se koristi i Simpletest (**www.simpletest.org**)
- Pokazati ćemo primjer sa bibliotekom PHPUnit
- Prije korištenja moramo ju instalirati
- Dostupna je putem PEAR (**<http://pear.php.net>**)



DEBUGIRANJE I TESTIRANJE U PHP-U

- Instalacija se vrši preko CLI-a unutra PHP direktorija, potrebno je otvoriti u direktorij u kojem se nalazi PHP osim ako niste podesili PHP u path varijable
- Za XAMPP PHP se nalazi u <C:/xampp/php>
- PHPUnit se može instalirati koristeći Composer:
composer require --dev phpunit/phpunit
- Za provjeru je li uspješno instaliran možete provjeriti naredbom:
vendor/bin/phpunit --version
- Testove pokrećete pojedinačno naredbom:
vendor/bin/phpunit PravokutnikTest.php
- Sve testove u direktoriju tests pokrećete naredbom:
vendor/bin/phpunit tests



DEBUGIRANJE I TESTIRANJE U PHP-U

- Unit testovi se provode pomoću tvrdnji (eng. Assertions), to je koncept koji kaže provjeri je li ovo istina. PHPUnit ima niz funkcija za tvrdnje:
 - **assertContains()**
 - **assertCount()**
 - **assertEquals()**
 - **assertFalse()**
 - **assertFileExists()**
 - **assertGreaterThanOrEqual()**
 - **assertNull()**
 - **assertRegExp()**
 - **assertSame()**
 - **assertTrue()**



DEBUGIRANJE I TESTIRANJE U PHP-U

- Postoje i druge funkcije koje možete pronaći na PHPUnit stranicama
- Tvrdnje su vrlo jednostavne i s njima provjeravamo neke činjenice
- Nalazi li se student u nizu prijava na ispit:
`assertContains($student, $prijava);`
- Tvrdnja će pasti ako se ne nalazi
- Provjera je li OIB studenta jednak vrijednosti:
`assertEquals('12345678910', $ student);`



DEBUGIRANJE I TESTIRANJE U PHP-U

- Test se stvara nasljeđivanjem klase **TestCase**
`class NekaKlasaTest extends TestCase {`
`}`
- Prema konvenciji, nova klasa se imenuje tako da se postojećoj klasi koju testirate na kraju doda **Test**
- Unutar klase stvara se javna metoda za svaki test, naziv metode počinje sa test:

```
function testNesto() {  
    // Ovdje dolazi kod testa  
}
```



DEBUGIRANJE I TESTIRANJE U PHP-U

- Za potrebe testiranja napraviti ćemo klasu Pravokutnik

```
<?php
```

```
class Pravokutnik {  
    public $sirina = 0;  
    public $visina = 0;
```

```
    //Konstruktor
```

```
    function __construct($w = 0, $h = 0) {  
        $this->sirina = $w;  
        $this->visina = $h;  
    }
```

```
    function Dimenzije($w = 0, $h = 0) {  
        $this->sirina = $w;  
        $this->visina = $h;  
    }
```



DEBUGIRANJE I TESTIRANJE U PHP-U

```
//Računa površinu
function Povrsina() {
    return ($this->sirina * $this->visina);
}
//Računa opseg
function Opseg() {
    return ( ($this->sirina + $this->visina) * 2 );
}
// Provjera je li kvadrat
function Kvadrat() {
    if ($this->sirina == $this->visina) {
        return true;
    } else {
        return false;
    }
}
} // Kraj klase
```



DEBUGIRANJE I TESTIRANJE U PHP-U

- Radimo jednostavan test:

```
<?php
```

```
require('Pravokutnik.php');
```

```
//Definiramo test klasu
```

```
class PravokutnikTest extends TestCase {
```

```
// Testiramo metodu Povrsina
```

```
function testPovrsina() {
```

```
//Stvaramo novi pravokutnik
```

```
$r = new Pravokutnik(8,9);
```

```
//Tvrdnja koja provjerava izračun
```

```
$this->assertEquals(72, $r->Povrsina());
```

```
}
```

```
}
```



DEBUGIRANJE I TESTIRANJE U PHP-U

- Nakon stvaranja test se pokreće preko CLI-a
- Naredba je:
`vendor/bin/phpunit PravokutnikTest.php`
- Kao rezultat dobiti ćemo ispis da smo proveli jedan test i jednu tvrdnju
- Ako test ne prođe dobiti ćemo da test nije prošao
- Ako za više testova želimo koristiti iste postavke objekta koristimo `setUp()` metodu
- U njoj postavljamo željene vrijednosti
- Suprotno je `tearDown()`



DEBUGIRANJE I TESTIRANJE U PHP-U

- Primjer složenijeg testa za klasu Pravokutnik

```
<?php
```

```
require('Pravokutnik.php');
```

```
class PravokutnikTest extends TestCase {
```

```
    // Za spremanje objekta Pravokutnik
```

```
    protected $r;
```

```
    // Stvorimo novi objekt sa postavkama
```

```
    function setUp():void {
```

```
        $this->r = new Pravokutnik(8,9);
```

```
    }
```

```
    // Testiramo metodu Povrsina
```

```
    function testPovrsina() {
```

```
        $this->assertEquals(72, $this->r->Povrsina());
```

```
    }
```



DEBUGIRANJE I TESTIRANJE U PHP-U

```
//Testiramo metodu Opseg  
function testOpseg() {  
    $this->assertEquals(34, $this->r->Opseg());  
}  
//Testiramo metodu Kvadrat  
function testKvadrat() {  
    //U ovom slučaju ne smije biti kvadrat  
    $this->assertFalse($this->r->Kvadrat());  
  
    //Napravi kvadrat i testiraj opet  
    $this->r->Dimenzije(5,5);  
    $this->assertTrue($this->r->Kvadrat());  
}
```



DEBUGIRANJE I TESTIRANJE U PHP-U

```
//Testiramo Dimenzije  
function testDimenzije() {  
    $w = 5;  
    $h = 8;  
    $this->r->Dimenzije($w, $h);  
    $this->assertEquals($w, $this->r->sirina);  
    $this->assertEquals($h, $this->r->visina);  
}  
  
}
```



DEBUGIRANJE I TESTIRANJE U PHP-U

- Primjer testiranja upisa u datoteku

```
<?php
```

```
class FileException extends Exception {
```

```
    function Detalji() {
```

```
        //Vraća poruku ovisno o kodu
```

```
        switch ($this->code) {
```

```
            case 0:
```

```
                return 'Nema imena datoteke';
```

```
                break;
```

```
            case 1:
```

```
                return 'Datoteka ne postoji';
```

```
                break;
```

```
            case 2:
```

```
                return 'To nije datoteka';
```

```
                break;
```



DEBUGIRANJE I TESTIRANJE U PHP-U

case 3:

```
    return 'Nije dozvoljeno pisati u datoteku';  
    break;
```

case 4:

```
    return 'Niste dali ispravan mod';  
    break;
```

case 5:

```
    return 'Podaci se ne mogu zapisati';  
    break;
```

case 6:

```
    return 'Datoteka se ne može zatvoriti';  
    break;
```

default:

```
    return 'Nemamo informacija o datoteci';  
    break;
```

```
}
```

```
}}
```



DEBUGIRANJE I TESTIRANJE U PHP-U

```
class ZapisUDatoteku {  
    // Za spremanje pokazivača na datoteku  
    private $_fp = NULL;  
    // Poruka o pogrešci  
    private $_message = "";  
    // Konstruktor  
    function __construct($file = null, $mode = 'w') {  
        // Datoteka i mod se zapisuju u poruku  
        $this->_message = "Datoteka: $file Mod: $mode";  
        // Postoji li ime datoteke  
        if (empty($file)) throw new FileException($this->  
_message, 0);  
        // Postoji li datoteka  
        if (!file_exists($file)) throw new FileException($this->  
_message, 1);  
    }  
}
```



DEBUGIRANJE I TESTIRANJE U PHP-U

```
//Je li datoteka
    if (!is_file($file)) throw new FileException($this-
>_message, 2);
    //Može li se pisati
    if (!is_writable($file)) throw new FileException($this-
>_message, 3);
    //Provjera moda
    if (!in_array($mode, array('a', 'a+', 'w', 'w+'))) throw
new FileException($this->_message, 4);
    //Otvori datoteku
    $this->_fp = fopen($file, $mode);

}
```



DEBUGIRANJE I TESTIRANJE U PHP-U

```
function Pisi($data) {  
    if (@!fwrite($this->_fp, $data . "\n")) throw new  
FileException($this->_message . " Podaci: $data", 5);  
}  
function Zatvori() {  
    if ($this->_fp) {  
        if (@!fclose($this->_fp)) throw new  
FileException($this->_message, 6);  
        $this->_fp = NULL;  
    }  
}  
//Destruktor  
function __destruct() {  
    $this->close();  
}  
}
```



DEBUGIRANJE I TESTIRANJE U PHP-U

○ Slijedi kod za testiranje koji pokrećemo sa
phpunit C:/xampp/htdocs/pr2/unit/ZapisUDatotekuTest.php
<?php

```
require('ZapisUDatoteku.php');
```

```
class ZapisUDatotekuTest extends TestCase {  
    private $_fp = NULL;  
    private $_file = 'nekadatoteka.txt';  
    private $_data = 'testni podaci za upis u datoteku';
```

```
// Otvori datoteku za pisanje
```

```
function setUp():void {  
    $this->_fp = fopen($this->_file, 'w');  
}
```



DEBUGIRANJE I TESTIRANJE U PHP-U

```
// Zapiši i testiraj:  
function testPisi() {  
    fwrite($this->_fp, $this->_data);  
    $this->assertEquals($this->_data,  
file_get_contents($this->_file));  
    // $this->assertEquals('podaci koji se ne nalaze u  
datoteci', file_get_contents($this->_file));  
}  
  
// Zatvori datoteku  
function tearDown():void {  
    fclose($this->_fp);  
}  
  
}
```

