Centar Informacijske Sigurnosti

Zend Framework - web razvojno okruženje

CIS-DOC-2011-03-007

Upozorenje

Podaci, informacije, tvrdnje i stavovi navedeni u ovom dokumentu nastali su dobrom namjerom i dobrom voljom te profesionalnim radom CIS-ovih stručnjaka, a temelje se na njihovom znanju i petnaestak godina iskustva u radu u informacijskoj sigurnosti. Namjera je da budu točni, precizni, aktualni, potpuni i nepristrani.

Ipak, oni su dani samo kao izvor informacija i CIS ne snosi nikakvu izravnu ili posrednu odgovornost za bilo kakve posljedice nastale korištenjem podataka iz ovog dokumenta.

Ukoliko primijetite bilo kakve netočnosti, krive podatke ili pogreške u ovom dokumentu, ili imate potrebu komentirati sadržaj molimo Vas da to javite elektroničkom poštom na adresu info@CIS.hr.

O CIS-u

CIS izrađuje pregledne dokumente (eng. white paper) na teme iz područja informacijske sigurnosti koji će biti korisni zainteresiranoj javnosti, a u svrhu **podizanje njezine svijesti o informacijskoj sigurnosti** i **sposobnosti za čuvanje i zaštitu informacija i informacijskih sustava**. Pored toga, CIS razvija i održava mrežni portal www.CIS.hr kao referalnu točku za informacijsku sigurnost za cjelokupnu javnost; izrađuje obrazovne materijale namijenjene javnosti; organizira događaje za podizanje svijesti o informacijskoj sigurnosti u javnosti i pojedinim skupinama te djeluje u suradnji sa svim medijima.

CIS **okuplja mlade** zainteresirane za informacijsku sigurnost i radi na njihovom pravilnom odgoju i obrazovanju u području informacijske sigurnosti te pripremu za **profesionalno bavljenje informacijskom sigurnošću**.

Centar informacijske sigurnosti [CIS] nastao je 2010. godine na poticaj Laboratorija za sustave i signale[LSS] Zavoda za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu, a kao posljedica 15togodišnjeg rada na istraživanju, razvoju i primjeni informacijske sigurnosti. LSS je među ostalim potaknuo osnivanje CARNetovog CERTa i sudjelovao u izradi Nacionalnog programa informacijske sigurnosti RH.

Smisao CISa je da bude **referentno mjesto za informacijsku sigurnost** za javnost, informatičare i posebno za mlade te da sustavno podiže njihovu svijest i sposobnosti u području informacijske sigurnosti.

Rad CISa podržava Ministarstvo znanosti, obrazovanja i sporta Republike Hrvatske, a omogućuju sponzori.

Prava korištenja

Sadržaj

1. Uvod

Programski jezik PHP se već više od 10 godina koristi za izradu dinamičkih web sjedišta. Inicijalno su se dinamičke stranice stvarale uz pomoć jezika HTML, a dinamičke karakteristike dodavao je programski jezik PHP koji se uklapao u postojeću HTML stranicu. Ovakvo ispreplitanje dvaju programskih jezika je predstavljalo pogodnost budući da se na jednostavan način dodavala funkcionalnost web sjedištima. Kroz inačice 3 i 4, programski jezik PHP stekao je veliku popularnost u zajednici otvorenog koda. Kako je popularnost jezika rasla bilo je neizbježno nastajanje sve većih i većih aplikacija koje su temeljene na ovom jeziku.

Ubrzo je postalo jasno da isprepletanje dvaju programskih jezika nije predstavljalo dugoročno rješenje za velike web aplikacije. Problemi se očituju u skalabilnosti i održavanju velikih projekata. Točnije, puno je teže dodavati nove funkcionalnosti i ispravljati programske greške (eng. *bug*) ukoliko se funkcijski dio koda nalazi zajedno sa HTML prikazom. Jedno svojstvo web aplikacija je često mijenjanje grafičkog sučelja, odnosno izgleda. Ukoliko se funkcijski dio koda usko veže uz HTML nije moguće zamijeniti izgled stranice bez mijenjanja postojeće funkcionalnosti koda. U cilju rješavanja ovog problema nastala su razvojna okruženja, a jedno od najpopularnijih razvojnih okruženja za programski jezik PHP je *Zend Framework*. Razvojna okruženja općenito služe za potporu izgradnje i testiranja dinamičkih web stranica, web aplikacija i usluga. Također, pomažu u razrješavanju učestalih problema koji se pojavljuju prilikom projektiranja web aplikacija. Često okruženja nude gotov skup biblioteka i alata za pristup bazi podataka, izradu i upravljanje predlošcima, upravljanje korisničkom sjednicom te olakšavaju ponovnu uporabu programskog koda.

Mnoga razvojna okruženja prate arhitektonski obrazac MVC (engl. *Model-View-Controller*) koji omogućuje razdvajanje modela podataka s poslovnim pravilima od korisničkog sučelja. Ovo svojstvo se općenito smatra dobrom praksom obzirom da modularizira kod, potiče ponovnu uporabu koda i omogućuje više sučelja koji se mogu primjenjivati. Zend okruženje je stvoreno s ciljem jednostavnijeg stvaranja i održavanja velikih web aplikacija zasnovanih na jeziku PHP. Sadrži bogat skup gotovih komponenti koje pokrivaju velik dio učestalih potreba prilikom izrade web aplikacija. MVC obrazac je ključni dio Zend okruženja i detaljnije se prikazuje u idućem poglavlju.

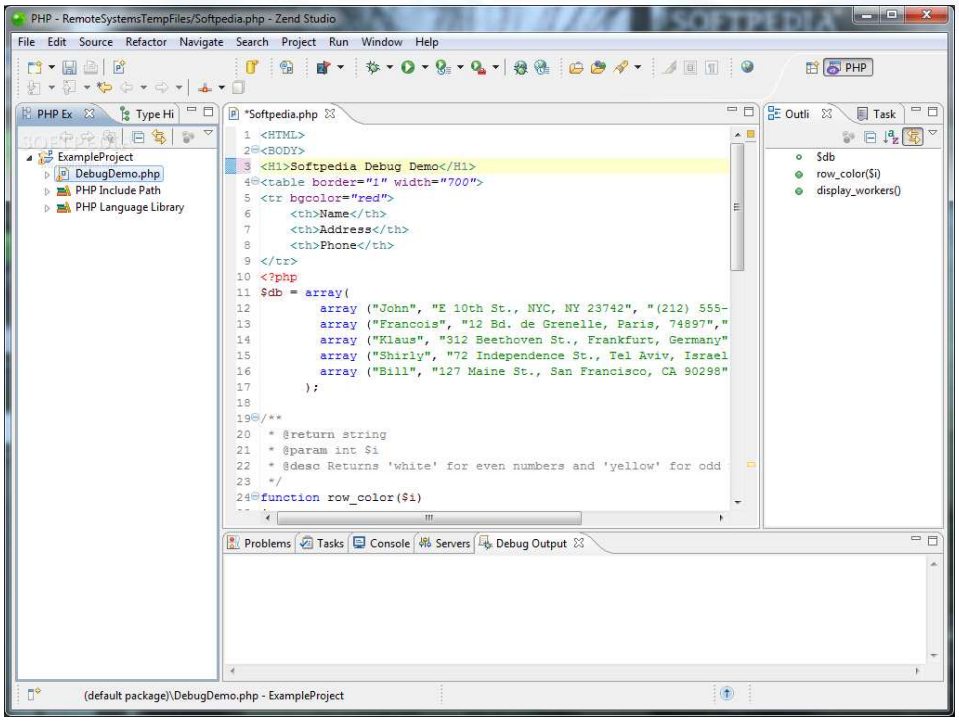
2. Zend Framework

Zend Framework (Slika 1) je razvojno okruženje otvorenog koda za izradu web aplikacija i servisa zasnovan na programskom jeziku PHP 5. Zend je u potpunosti implementiran korištenjem objektno orijentirane paradigme. Komponente ovog web razvojnog okruženja su oblikovane tako da imaju što manji broj zavisnosti prema drugim komponentama. Ovim se postiže suglasnost sa jednim od osnovnih koncepata objektno orijentiranog programiranja, a to je smanjivanje sprege (engl. *coupling*). Slaba spregnutost komponenata omogućuje programerima korištenje individualnih komponenata po potrebi. Zend zajednica ovu pogodnost naziva oblikovanje po potrebi (engl. *use-at-will design*).

Iako ih je moguće koristiti odvojeno, komponente razvojnog okruženja Zend čine snažnu i proširivu biblioteku gotovih rješenja. Cilj Zenda je pružiti programerima robusnu okolinu za razvoj vlastitih web aplikacija i usluga. On nudi MVC implementaciju visoke učinkovitosti, apstraktni sloj nad bazom podataka koji olakšava upravljanje podacima i niz komponenata koje implementiraju prikaz HTML stranica te validaciju i filtriranje korisničkog unosa. Ostale komponente kao što su Zend_Auth i Zend_Acl omogućavaju autorizaciju i autentifikaciju korisnika prilikom pristupa resursu. Prednost razvojnog okruženja Zend Framework je velika količina gotovih komponenata koje se mogu iskoristiti za izradu vlastitih web aplikacija ili servisa, a te komponente su temeljito testirane i isprobane.

Korištenjem gotovih testiranih komponenata smanjuje se vrijeme razvoja. Na primjer, validacija i/ili filtriranje korisničkog unosa je česta operacija u svim web aplikacijama. Ovisno o domeni aplikacije, korisnički unos može biti problematičan za analizu (npr. unos e-mail adrese ili JMBG). Zend sadrži niz gotovih komponenata koje implementiraju većinu poslovne logike vezane za filtriranje i validaciju unosa. Detaljniji primjeri su dani u poglavlju 4.

Glavni sponzor projekta „Zend Framework“ je tvrtka Zend Technologies ali mnoge tvrtke su pridonijele razvoju ovog web razvojnog okruženja. Ovakva potpora privatnog sektora Zend okruženju omogućava brži razvoj i kvalitetniju potporu. Dodatno, osnivači tvrtke Zend Technologies su Andi Gutmas i Zeev Suraski, koji su poznati kroz svoj rad na jezgri programskog jezika PHP. Smatra se da je njihova popularnost imala značajan utjecaj na širenje i prihvaćanje Zend okruženja u ranim fazama njegova razvoja. Tvrtke kao što su Google, Microsoft i StrikeIron udružile su se sa Zend Technologies kako bi pružili gotove komponente, odnosno sučelja prema vlastitim servisima i drugim tehnologijama koje žele pružiti korisnicima razvojnog okruženja Zend. Osim službenih partnera, Zend Framework se sastoji od velike zajednice korisnika koji međusobno razmjenjuju informacije putem IRC kanala, foruma i elektroničke pošte.

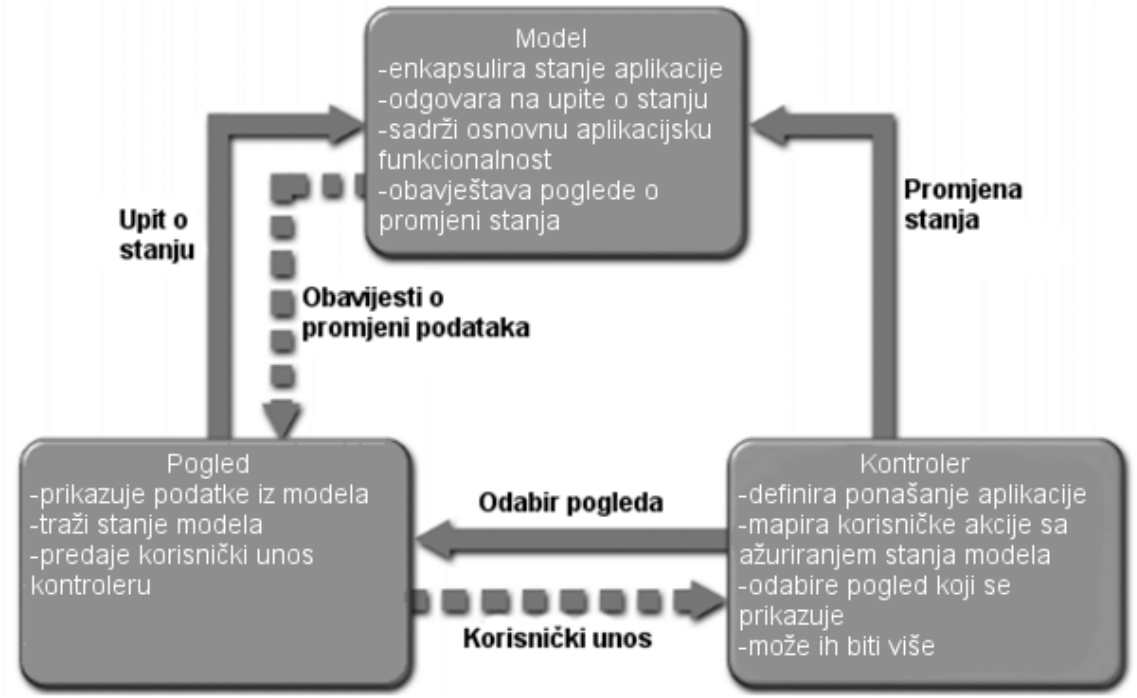


Slika 1. Zend korisničko sučelje

2.1. MVC obrazac

U programskom inženjerstvu oblikovni obrasci predstavljaju ispravno rješenje za razrede čestih problema koji se pojavljuju prilikom dizajna programske potpore. Neki od čestih problema koji se pojavljuju prilikom oblikovanja programske potpore odnose se na smanjivanje sprege između pojedinih komponenata, odvajanje sučelja i poslovne logike te neovisnost o podatkovnom sloju (bazi podataka). Oblikovni obrazac nije gotov dio programskog koda već predložak koji opisuje kako riješiti problem. Obrazac MVC (engl. *model-view-controller*) je programska arhitektura za odvajanje programskog koda koji odvaja problem od koda koji prezentira problem korisniku. Osnovno svojstvo MVC obrasca je odvajanje modela domene od prezentacijskog sloja ili grafičkog sučelja. Time se omogućuje jednostavno mijenjanje i stvaranje novih grafičkih maski bez promjene podložnog modela. MVC obrazac se sastoji od sljedećih komponenata:

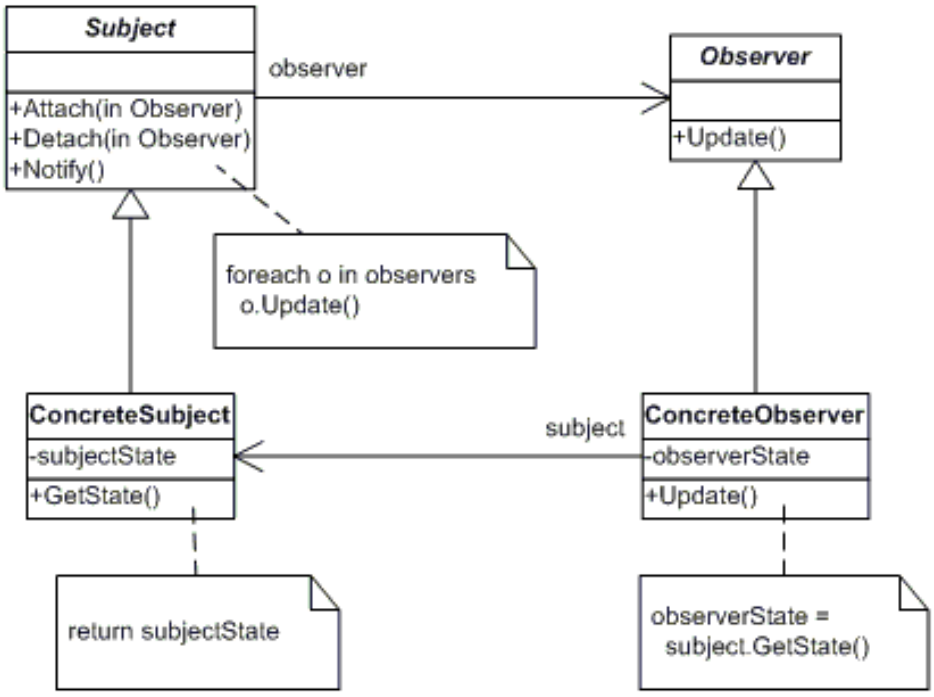
- **Model** – sadrži podatke i ponašanje koji predstavljaju problematiku aplikacije, odnosno čine model domene ili sloj poslovne logike. Model je apstrakcija podataka, i oblikuje dio stvarnog svijeta koji je potreban za aplikaciju. Zadužen je za prosljeđivanje novih podataka prema grafičkom sučelju, upravlja ponašanjem podataka te reagira na upute za promjenu stanja koja dolaze iz kontrolera.
- **Pogled** (engl. *View*) – pogledi su zaduženi za prezentaciju podataka korisniku. Pogledi mogu zatražiti podatke od modela i primiti ih. Najvažnije obilježje pogleda je njihova jednostavnost. Točnije, ne sadrže nikakvu aplikacijsku ili poslovnu logiku već samo prezentiraju podatke korisniku. Po potrebi prosljeđuju korisnički unos dalje na obradu.
- **Kontroler** (engl. *Controller*) – definira ponašanje aplikacije. Predstavlja posrednika između pogleda i modela. Korisnički unos se prosljeđuje iz pogleda prema kontroleru koji svaku akciju mapira na određen način. Za razliku od pogleda, kontroler sadrži dio aplikacijske logike i ima sposobnost utjecati na stanje modela. Točnije, preko definiranog sučelja može mijenjati podatke u samom modelu. Ovo se najčešće odnosi na obradu podataka koje je korisnik unio putem pogleda.



Slika 2. MVC obrazac

Slika 2. prikazuje komunikaciju između triju komponenata MVC obrasca. Najsloženija veza u ovom modelu je veza između modela i pogleda. Točnije, kako propagirati obavijest o promijeni

nekog skupa podataka unutar modela proizvoljnom broju pogleda. Ova veza postiže se uporabom drugog oblikovnog obrasca „Promatrač“ (engl. *Observer pattern*). Uporabom oblikovnog obrasca „promatrač“ omogućuje se slanje poruka od modela prema pogledima bez stvaranja ovisnosti o konkretnim pogledima. Slika 3. prikazuje opći dijagram razreda ovog oblikovnog obrasca. Kako se vidi iz dijagrama, neovisnost o konkretnim klasama se postiže usmjeravanjem ovisnosti prema apstraktnim klasama (na slici 3. apstraktne klase su Subject i Observer). Konkretne klase subjekta su entiteti koji čine model domene[1]. Iako Zend podržava i podupire razvoj vlastitih aplikacija korištenjem obrasca MVC, ne uvjetuje njegovu uporabu. Drugim riječima, moguće je napraviti web aplikaciju koja se sastoji od odabranih MVC dijelova, a umjesto potpunog modela koji je prikazan slikomSlika 2 dobiva se projekt koji je izgrađen isključivo od pogleda. Većina službenih primjera i knjiga podupiru razvoj aplikacije korištenjem MVC obrasca zbog djelotvornosti odvajanja korisničkog sučelja od funkcijskih dijelova aplikacije.



Slika 3. Dijagram razreda oblikovnog obrasca – promatrač trač
Izvor: *dofactory.com*

2.2. Razvoj i budućnost

Razvoj Zenda je usko vezan uz razvoj programskog jezika PHP. Točnije, nastao je u cilju proširivanja jezika i poboljšanja učinkovitosti u izradi aplikacija. Nakon što je PHP 3.0 postao javno dostupan 1998. godine, Andi Gutmas i Zeev Suraski počeli su raditi na poboljšanju njegove jezgre. Cilj im je bio poboljšati performanse složenih aplikacija i podršku za modularnu izradu komponenata. Tadašnja treća inačica Zend okruženja nije bila u stanju učinkovito raditi sa složenim tipovima podataka. Svoje poboljšanje prozvali su Zend Engine (sastavljeno od njihovih imena Zeev i Andi). Prvi puta je predstavljena 1999. godine, a nova, četvrta inačica programskog jezika PHP bazirala se na Zend Engineu. Nova inačica je donijela brojne pogodnosti kao što su:

- podrška za više poslužitelja,
- visoke performanse,
- HTTP sjednice,

[1] U većim projektima kod kojih se koristi objektno orijentirana paradigma konkretne subjekte čine posebni entiteti koji se nazivaju repozitoriji (engl. *Repository*). Repozitoriji se koriste za dohvat objekata putem definiranog sučelja kao i za njihovu pohranu. Kada se u repozitorij doda novi objekt poziva se metoda *Notify* iz nadređene klase *Subject* i šalje se poruka svim registriranim objektima (odnosno pogledima). Konkretni promatrači (odnosno pogledi) se prethodno moraju registrirati na repozitorije koje žele pratiti. Detaljnije o oblikovnim obrascima može se pronaći u [2] i [3].

- globalne poslužiteljske varijable (npr.: $_GET, $_POST, $_COOKIE, $_SERVER, $_SESSION) te
- sigurniji način upravljanja korisničkim unosom.

PHP 4 i dalje nije bio bolji po pitanju objektnog modeliranja od inačice 3. Iako PHP nije bio stvoren za objektno orijentirano modeliranje mnogi programeri su ga ipak koristili. Objektu su u četvrtoj inačici programskog jezika PHP predstavljali samo sintaksni šećer[2] za asocijativna polja[3]. Nova inačica Zend razvojnog okruženja (Zend 2) sadrži većinu obilježja objektno orijentiranih jezika kao što su:

- prosljeđivanje referenci (engl. *Pass by Reference*),
- kopiranje objekata (poznatije kao kloniranje) te
- vraćanje objekata.

PHP inačica 5 je po uzoru na Zend 2 preuzela sve karakteristike objektne paradigme te se od inačice 5 programski jezik PHP smatra potpuno objektno orijentiranim jezikom. Neke nove karakteristike, uz one koje uvodi Zend 2, su:

- destruktori,
- sakupljači smeća[4] (engl. *Garbage Collection*) i
- vidljivost varijabli objekata (standardno: public, private, protected)

Usvajanjem objektne paradigme stvorilo se plodno tlo za daljnji razvoj biblioteka gotovih funkcionalnih modula (engl. *Framework*). Autori PHP-a 5, Zeev Suraski i Andi Gutmas, su otišli jednu razinu iznad objektnog modela i predložili stvaranje gotovih i prilagodivih komponenti koje bi ubrzale razvijanje aplikacija i olakšalo njihovo održavanje. Gotove komponente bile bi oblikovane po odgovarajućim oblikovnim obrascima što smanjuje međusobnu povezanost i time povećava modularnost. Ta inicijativa je s vremenom sazrela u Zend Framework. Tijekom 2010. godine sve više web programera prelazi sa programskog jezika Java na PHP. Razlog tome je velika jednostavnost jezika PHP u odnosu na Javu. Programski jezik Java je strogo tipizirani potpuno objektni programski jezik koji se pokreće uz pomoć posebnog virtualnog jezičnog procesora. Jezik PHP se uči znatno lakše i puno je fleksibilniji u odnosu na Javu što ga čini pogodnim za početnike. Podrškom za objektno orijentiranu paradigmu i razvojem okruženja poput Zend Frameworka, PHP postaje jezik pogodan za razvoj vrlo složenih web aplikacija[5]. Autor Zend okruženja Zeev Suraski smatra da će kroz narednih nekoliko godina trend razvoja aplikacija korištenjem jezika PHP naglo narasti, a primarni razlog će biti korištenje razvojnih okruženja poput Zenda. Budućnost razvoja samog razvojnog okruženja Zend biti će usmjerena na računarstvu zasnovanom na oblacima.

[2] Sintaksni šećer – (engl. *Syntax sugar*) pojam koji označava uporabu posebnih sintaksnih pravila unutar nekog programskog jezika kako bi se olakšalo čitanje i/ili pisanje izraza.

[3] Asocijativno polje – (engl. *Associative array*) je apstraktni tip podatka koji se sastoji od kolekcije jedinstvenih ključeva i vrijednosti. Svaki ključ se veže uz jednu vrijednost i koristi se za pristup toj vrijednosti.

[4] Sakupljači smeća – (engl. *Garbage Collection*) u računarskoj znanosti predstavlja oblik automatiziranog upravljanja memorije. Karakteristični su za jezike poput Java, C#, Python i drugi, a primarna zadaća je brisanje objekata iz memorije koji se više ne koriste.

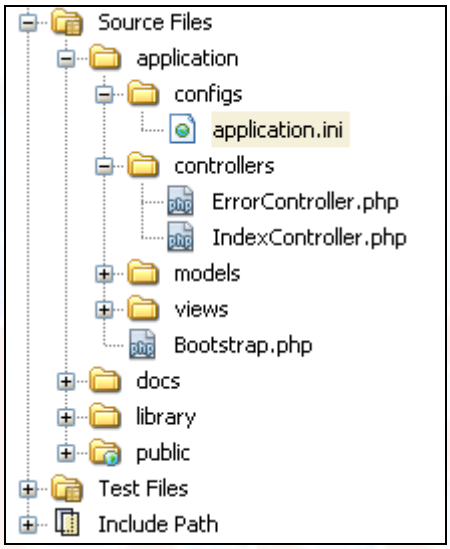
[5] Neke od najpoznatijih web aplikacija današnjice napravljeni su upravo uz pomoć programskog jezika PHP. Među značajnijim aplikacijama smatraju se Wikipedia, Facebook i White House.

3. Razvoj sigurnih PHP aplikacija koristeći Zend Framework

U nastavku se razmatra primjena Zend okruženja za izradu konkretnih funkcionalnih aplikacija, odnosno njihovih dijelova. Prije toga dan je pregled strukture Zend datoteka i način obrade korisničkih zahtjeva.

3.1. Struktura Zend datoteka

Tipična Zend Framework aplikacija se sastoji od osnovnih direktorija koji razdvajaju različite dijelove aplikacije. Raspored direktorija prikazan je na Slika 4.



Slika 4. Zend datoteke
Izvor: *sistemac.srce.hr*

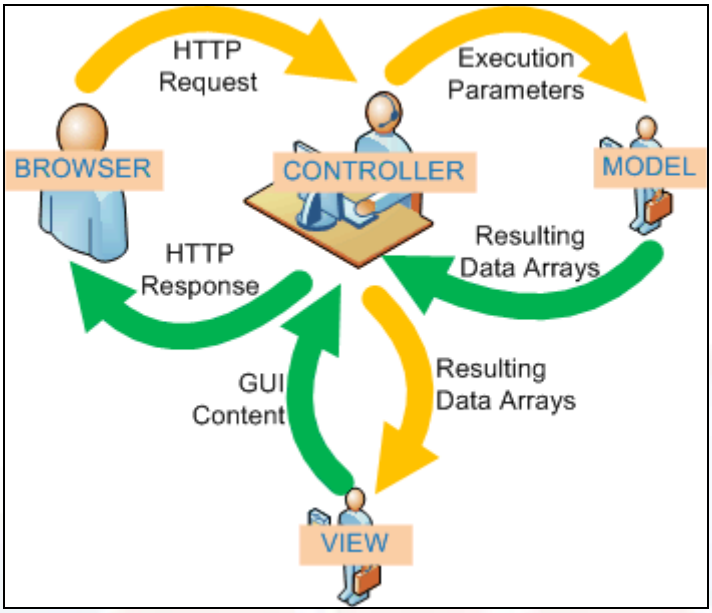
Svaka datoteka sadrži određene dijelove aplikacije, čiji opis je u nastavku:

- **Application** – sadrži programski kod potreban za pokretanje aplikacije. Unutar tog direktorija nalaze se dodatni direktoriji pomoću kojih se odvajaju pojedini dijelovi programskog koda.
- **Configs** – sadrži globalne konfiguracijske postavke aplikacije kao što su parametri za spajanje na bazu, podaci potrebni za slanje elektroničke pošte i drugo.
- **Controllers** – prateći MVC arhitekturu, izvorni kod svih kontrolera se nalazi u ovoj datoteci.
- **Models** – sadrži programski kod koji opisuje model domene, poslovna pravila i osnovnu funkcionalnost aplikacije.
- **Views** – sadrži programski kod koji čini korisničko sučelje.
- **Docs** – služi za pohranu dokumentacije.
- **Library** – direktorij namijenjen za pohranu raznih dodatnih programskih zbirki (Zend, Smarty engine[6], ORM, itd).
- **Public** – sadrži javne datoteke kao što su početna stranica (index.php), slike i CSS dokumenti koji čine korisničko sučelje.

[6] Smarty – jedan od najpopularnijih sustava za upravljanje predlošcima (engl. *template*) pisan u programskom jeziku PHP. Koristi se za razdvajanje grafičkog sučelja od ostatka programskog koda.

3.2. Obrada korisničkih zahtjeva

Osnovni modul Zend okruženja je *Zend_Controller* koji implementira glavni kontroler u MVC obrascu. Zend_Controller presreće HTTP zahtjeve (engl. *HTTP request*) te prema formatu zahtjeva poziva odgovarajuću metodu instancirane klase. Slika 5. prikazuje presretanje HTTP zahtjeva i proizvodnju odgovora.



Slika 5. Obrada HTTP zahtjeva
Izvor: *sistemac.srce.hr*

Zend okruženje svaku stranicu aplikacije smatra jednom akcijom, a akcije se grupiraju u kontrolere. Primjerice sljedeći URL zahtjev poziva akciju (metodu) *unosimena* koja se nalazi u kontroleru (klasi) *korisnik*. Ukoliko se izostavi akcija Zend poziva pretpostavljenu akciju *index*.

IP_ADRESA/public/zf-primjer/korisnik/unosimena

Modul Zend_Controller postavlja vlastita pravila u organizaciji datoteka i direktorija web aplikacije. Korisnik može podesiti svoja pravila izmjenom parametara modula Zend_Controller.

3.3. Sigurna prijava korisnika koristeći Zend

Izgradnja sustava za prijavu je vrlo čest zadatak programera. Sigurnosni aspekt sustava za prijavu je vrlo važan, a u nastavku se navodi nekoliko glavnih sigurnosnih problema. Ponuđeno rješenje nije konačno već služi za demonstraciju izrade modula korištenjem Zend okruženja. Rješenje se sastoji od obične web stranice na kojoj se nalazi HTML forma s poljima za unos korisničkog imena i lozinke. Polja su mapirana u jednostavnu MySQL bazu podataka koristeći sljedeću tablicu:

```
CREATE TABLE users (
       username VARCHAR(32) NOT NULL
     , password CHAR(32) NOT NULL
     , salt CHAR(20) NOT NULL
     , email VARCHAR(80)
     , active BOOLEAN NOT NULL DEFAULT 1
     , last_access DATETIME NOT NULL
     , PRIMARY KEY (username)
);
```

Lozinka se ne pohranjuje u izvornom obliku već se izračunava MD5[7] sažetak te se dobivena vrijednost pohranjuje u bazu. Iz tog razloga se koristi tip podataka *CHAR(32).* Poznato je da lozinku pohranjenu u sažetom (engl. *Hashed*) obliku, u teoriji, nije moguće vratiti u početno stanje. Bitno je napomenuti kako ova metoda sama po sebi ne jamči apsolutnu sigurnost korisničke lozinke budući da današnje sklopovlje može u relativno kratkom vremenu proizvesti velik broj sažetaka te se tako slabe lozinke[8] mogu pogoditi. Neki napadi koji se koriste u ovu svrhu su napad grubom silom (engl. *Brute force attack*) i napadi rječnikom (engl. Dictionary attack). U cilju poboljšanja sigurnosti MD5 sažetka koristi se salt[9] vrijednost. Konačna vrijednost atributa *password* tablice *users* dobiva se spajanjem salt vrijednosti na lozinku u izvornom obliku te stvaranjem MD5 sažetka (npr.: *MD5(CONCAT(salt,'password'))*). Ovom metodom se otežava uspješno izvođenje napada rječnikom i napada grubom silom. Za izradu primjera koriste se sljedeće Zend Framework klase:

- **Zend_Form** – koristi se za izradu stranice za prijavu korisnika.
- **Zend_Auth_Adapter_DbTable** – služi za autentikaciju korisnika uz pomoć podataka pohranjenih u bazi.
- **Zend_Session** – koristi se za pohranjivanje podataka o korisničkoj sjednici.
- **Zend_Config** – za pohranu konfiguracijskih datoteka aplikacije.
- **Zend_Db_Table** – služi za mapiranje tablica u bazi podataka s PHP klasama.

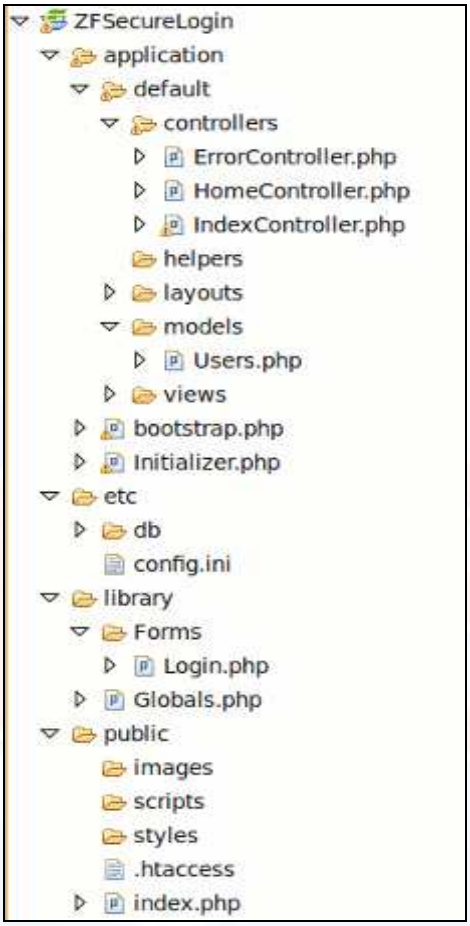
Struktura direktorija prikazana je slikom 6. Glavni direktoriji su: **application**, **etc**, **library** i **public**. Direktoriji se koriste na isti način kako je opisano u poglavlju 3.1. Klasa Initializer (na slici *Initializer.php*) je dodatak (engl. *plug-in)* za kontroler gdje se obavlja autentikacija korisnika. Ovim putem je moguće upravljati procesom autentikacije bez ikakvih izmjena u samom kontroleru. Točnije, klasa Initializer je zadužena za autentikaciju korisnika u sustav. Klasa je oblikovana po pravilima koja Zend propisuje za sve dodatke. Pravila se uglavnom sastoje od implementacije određenih sustavskih metoda u vlastite klase. Od svih metoda posebno se ističu *preDispatch* i *checkSession* koje se koriste u ovom primjeru. U nastavku je opis tih metoda i ispis izvornog koda:

[7] MD5 – (engl. *Message Digest 5*) jedan od najpopularnijih algoritama za generiranje sažetaka poruka. Kao izlaz daje 128-bitni sažetak dobiven miješanjem 512-bitnih blokova.

[8] Slabe lozinke – izraz koji se koristi za niz znakova koji se u relativno kratkom vremenu može pogoditi korištenjem raznih metoda društvenog inženjeringa ili alata za pogađanje lozinki grubom silom (engl. *Brute Force*) ili rječnikom (engl. *Dictionary attack*).

[9] Salt – niz nasumičnih bitova koji se dodaju lozinki prije proizvodnje sažetka. Koristi se za otežavanje napada pogađanje lozinke kada je poznat sažetak.

- **preDispatch** – prva metoda koju Zend poziva svaki put kada obavlja neku akciju. Iz tog razloga je ovo vrlo pogodno mjesto za ugradnju sustava za autentikaciju. Ukoliko trenutni kontroler koji obrađuje korisnički zahtjev nije *index* ili *error* kontroler poziva se metoda checkSession. Kontroleri indeks i error se izostavljaju zato što oni predstavljaju stranice koje moraju biti javno dostupne te se postupak autentikacije ne odnosi na njih.
- **checkSession** – metoda je vrlo jednostavna te provjerava je li sjednička varijabla *username* postavljena. Ukoliko nije postavljena znači da se korisnik nije prijavio na sustav i potrebno ga je preusmjeriti na stranicu za prijavu. Na kraju metode poziva se funkcija *exit*. Poziv funkcije exit je vrlo bitan jer će u suprotnom Zend nastaviti sa izvođenjem.



Slika 6. Struktura direktorija iz primjera

```
/**
 * preDispatch
 */
public function preDispatch () {
  $this->_controller= $this->_front->getRequest()->getControllerName();
  if (($this->_controller!='index') && ($this->_controller!='error')) {
    if (Globals::getConfig()->authentication->active) {
      $this->checkSession();
    }
  }
}
/**
 * checkSession
 */
private function checkSession() {
  if (empty(Globals::getSession()->username)) {
    $this->_response->setRedirect('/index/login')->sendResponse();
    exit;
  }
}
```

U primjerima prethodnih metoda koristi se posebna klasa *Globals* prilikom dohvaćanja sjednice i konfiguracijske datoteke. Ovaj pristup je pogodan kada postoji samo jedna instanca baze podataka, sjednice i konfiguracijske datoteke. Klasa *Globals* je napravljena prema oblikovnom obrascu *singleton*[10] čime se ubrzava čitanje navedenih instanci prilikom izvođenja jer se jednom dohvaćena instanca pohranjuje u memoriji.

Forma za prijavu korisnika se nalazi u direktoriju *library/Forms/Login.php*. Koristi se klasa **Zend_Form_Element_Hash** Zend biblioteke kako bi se spriječio CSRF napad[11]. Ova klasa ima posebnu metodu za postavljanje *salt* vrijednosti, *setSalt*. Sljedećim isječkom se demonstrira instanciranje klase *Zend_Form_Element_Hash*, postavljanje njene *salt* vrijednosti na pseudo slučajnu vrijednost dobivenu funkcijama *uniqid*, *rand* i *md5* te postavljanje roka trajanja na proizvedeni token. Token predstavlja posebnu vrijednost koja osigurava da HTTP zahtjev koji dolazi na sustav proizveden upravo ovom formom, odnosno da nije došao sa druge (potencijalno maliciozne) stranice.

```
...
$token = new Zend_Form_Element_Hash('token');
$token->setSalt(md5(uniqid(rand(), TRUE)));
$token->setTimeout(Globals::getConfig()->authentication->timeout);
$this->addElement($token);
...
```

Forma za prijavu se instancira metodom *loginAction*.

```
...
public function loginAction() {
  $flash = $this->_helper->getHelper('flashMessenger');
  if ($flash->hasMessages()) {
    $this->view->message = $flash->getMessages();
  }
  $this->view->form= new Forms_Login();
  $this->render('login');
}
...
```

[10] Singleton – oblikovni obrazac koji se koristi za implementaciju matematičkog koncepta jednočlanog entiteta restrikcijom instanciranja klase na samo jedan objekt.

[11] CSRF napad – (engl. *Cross-site request forgery*) oblik malicioznog iskorištavanja web aplikacije u kojem se neautorizirane naredbe prosljeđuju kroz korisnički račun koji aplikacija smatra sigurnim.

Kada korisnik popuni formu sa podacima za prijavu, ti podaci se prosljeđuju metodi *submitAction* koja ih obrađuje. Prvo se provjerava valjanost tokena, pa ukoliko token nije ispravan znači da se sustav nalazi pod CSRF napadom te se prosljeđuje na adekvatnu stranicu (koja će obraditi napad). U ovom slučaju to je stranica *csrf-forbidden* koja samo ispisuje statusnu poruku o zabrani. Više informacija o CSRF napadima može se pronaći u literaturi [2]. Za autentifikaciju korisnika koristi se adapter **Zend_Auth_Adapter_DbTable**. Provjerava se samo za aktivne korisnike (oni kojima je atribut *active* postavljen na vrijednost 1). Ukoliko je autentikacija ispravna ažurira se vrijeme zadnjeg pristupa (atribut last_access) te se korisnik preusmjerava na matičnu stranicu. Ukoliko autentikacija nije ispravna, korisnik se preusmjerava natrag na *login* stranicu sa porukom „*Authentication error.*“. Dodatno, prilikom uspješne autentikacije pokreće se metoda *Zend_Session::regenerateId* koja osvježava korisničku sjednicu. Također, sjednica se prilikom odjave briše. Ova sigurnosna mjera sprječava napade fiksacije sjednice (engl. *Session fixation attack*). Više o napadu fiksacija sjednice može se naći u literaturi [3]. Napad SQL injekcijom se nije spominjao iz razloga što klasa **Zend_Db** koja se koristi za pristup bazi podataka samostalno obrađuje opasne znakove. Dodatno, klasa *Zend_Form* posjeduje metodu za filtriranje korisničkih unosa *addValidators*. U nastavku je isječak s metodom *submitAction*.

```
…
public function submitAction() {
  $form= new Forms_Login();
  if (!$form->isValid($_POST)) {
    if (count($form->getErrors('token')) > 0) {
      return $this->_forward('csrf-forbidden', 'error');
    } else {
      $this->view->form = $form;
      return $this->render('login');
    }
  }
  $username= $this->getRequest()->getPost('username');
  $password= $this->getRequest()->getPost('password');
  $authAdapter = new Zend_Auth_Adapter_DbTable(
                              Globals::getDbConnection(),
                          'users',
                          'username',
                          'password',
                          'MD5(CONCAT(salt,?)) AND active=1'
                            );
  $authAdapter->setIdentity($username)
                        ->setCredential($password);
  $result= $authAdapter->authenticate();
  Zend_Session::regenerateId();
  if (!$result->isValid()) {
    $this->_helper->flashMessenger->addMessage("Authentication
error.");
    $this->_redirect('/index/login');
  } else {
    Globals::getSession()->username= $result->getIdentity();
    Zend_Loader::loadClass('Users');
    $users= new Users();
    $data= array ('last_access' => date('Y-m-d H:i:s'));
    $where= $users->getAdapter()->quoteInto('username = ?',
Globals::getSession()->username);
    if (!$users->update($data,$where)) {
      throw new Zend_Exception('Error on update last_access');
    }
    $this->_redirect('/home');
  }
}
…
```

4. Sigurnosni aspekti

Zend je jedno od najpopularnijih PHP razvojnih okruženja, koji se koristi za izradu složenih i jednostavnih aplikacija. Samim time potrebno je osigurati ispravan, posebice siguran, rad svih Zend komponenata. Ranjivost samo jedne komponente može kompromitirati velik broj web aplikacija (odnosno svih aplikacija koje tu komponentu koriste). Osim toga, puno češći izvor ranjivosti web aplikacija su propusti koje programeri samostalno uvode u svoje aplikacije nepažnjom ili neznanjem.

4.1. Sigurnosne ranjivosti i nedostaci Zend komponenata

Svaka komponenta Zend okruženja prolazi iscrpan proces testiranja. Testiranje se provodi nad svakom funkcionalnom programskom jedinicom (engl. *Unit Testing*[12]). Ipak, zbog moguće složenosti komponente ili procesa testiranja neke propuste jednostavno nije moguće uočiti na vrijeme. Posljedice takvih propusta potencijalno mogu biti velike. Jedan od najkontroverznijih primjera takvog propusta je ranjivost uključivanja lokalnih datoteka[13] (engl. *Local File Inclusion*) koja je uklonjena u Zend inačici 1.7.5. Kontroverzija leži u načinu na koji je navedena ranjivost uklonjena. Točnije, izdana sigurnosna zakrpa za ranjivu komponentu *Zend_View* je u nekim slučajevima korištenja mogla uzrokovati neispravan rad sustava ili usluge. Ranjivost se nalazila u metodi *setScriptPath* komponente *Zend_View*. Ukoliko bi korisnik kao ulazni parametar predao stazu skripte, pokrenuo bi se LFI napad. Navedena ranjivost je potpuno neprirodna budući da se ova naredba nikad ne pokreće iz konteksta korisničke skripte. Ipak, ta ranjivost je istaknula drugi veoma realan LFI vektor napada. Vektor napada i u ovom slučaju iskorištava neispravno filtriranje korisničkog unosa, no ovaj put u vrlo realnom kontekstu izvršavanja ranjive naredbe. Druga metoda komponente *Zend_View*, metoda *render*, dozvoljavala je kretanje prema vršnim direktorijima znakovnim nizom „../“. Time se omogućilo korištenje znakovnog niza „../../../../etc/passwd“ u svrhu prikaza osjetljivih datoteka operacijskog sustava (ili bilo kojih drugih datoteka na sustavu). U inačici 1.7.5 Zend okruženja ta se ranjivost riješila filtriranjem znakovnih nizova „../“ i „..\“. Navedeno rješenje filtrira sve pojave „opasnog“ znakovnog niza neovisno o tome da li dolazi kao korisnički unos ili se programski doziva. U nekim situacijama programeri su programski dohvaćali određene datoteke koristeći metodu *render*, no zbog novog procesa filtriranja ovo više nije bilo moguće te je uzrokovalo grešku. Kako bi se osigurala kompatibilnost s dotadašnjim skriptama dodana je zastavica kojom je moguće isključiti, odnosno uključiti filtriranje. Sljedeći isječak demonstrira ovaj proces; moguće je zastavicu postaviti prilikom instanciranja komponente *Zend_View* ili kasnije tokom izvođenja. Detaljan popis svih poznatih ranjivosti Zend komponenata nalazi se u literaturi [4].

```
// Prilikom instanciranja komponente:
$view = new Zend_View(array(
    'lfiProtectionOn' => false,
));

// kasnije tokom izvođenja:
$view->setLfiProtection(false);
```

[12] Testiranje funkcionalnih jedinica (engl. *Unit Testing*) – metoda testiranja programske podrške u kojoj se svaka funkcionalna jedinica koda testira zasebno kako bi se ustanovilo je li spremna za uporabu.

[13] Napad uključivanja lokalne datoteke – (engl. *Local File Inclusion*) iskorištava pozadinski interpreter kako bi se dobio prikaz lokalnih datoteka. Najčešća meta je datoteka /etc/passwd na operacijskim sustavima Linux/Unix.

4.2. Sigurnost korisničkih aplikacija

Zend nudi velik broj gotovih komponenata koje nude određenu funkcionalnost. Neke od tih komponenata mogu se koristiti za implementaciju sigurnosnih mjera u korisničkim aplikacijama. Često korištene komponente su:

- **Zend_Auth** – koristi se za autentifikaciju korisnika i perzistiranje podataka o korisniku,
- **Zend_Acl** – brine se za odgovarajući pristup resursima, upravlja ulogama i odlučuje koji korisnik smije pregledavati određen sadržaj.

Ipak, u većini slučajeva korisnici moraju samostalno implementirati neku funkcionalnost. Kako bi se smanjio broj ranjivosti u korisničkim implementacijama Zend nudi velik broj gotovih komponenata koje enkapsuliraju česte potrebe kao što su filtriranje sadržaja, upravljanje sjednicom, sprječavanje injekcija i druge. U nastavku se navode neke komponente s primjerima korištenja, a detaljniji opis se nalazi u literaturi [5].

4.2.1. Validacija korisničkog unosa

Zend nudi velik broj gotovih komponenti za validaciju korisničkog unosa. U nastavku se izdvajaju najčešće korišteni:

- **Alnum** – provjerava da li zadani znakovni niz sadrži isključivo slova i brojeve.
- **EmailAddress** – provjerava da li zadani znakovni niz predstavlja ispravnu *e-mail* adresu.
- **Hex** – provjerava da li su svi zadani znakovi heksadekadski.
- **Ip** – provjerava da li je znakovni niz IP adresa.
- **Regex** – provjerava da li znakovni niz zadovoljava regularni izraz.
- **StringLength** – provjerava da li je znakovni niz odgovarajuće veličine.

Glavna klasa za validaciju se zove **Zend_Validate**, a konkretnih potklasa ima više. Ime konkretne klase se dobiva dodavanjem imena validatora (npr. *Alnum*) na kraj osnovne klase (npr. *Zend_Validate_Alnum*). U nastavku je isječak programskog koda koji provjerava da li je predani znakovni niz *e-mail* adresa. Dodatno, Zend validatori imaju mogućnost javiti razlog zbog kojeg je znakovni niz odbijen.

```
$email = $this->getRequest()->getPost('email', 'none@example.com');

$validator = new Zend_Validate_EmailAddress();

if ($validator->isValid($email)) {
     // email je ispravan
} else {
     // email je neispravan; ispiši razloge
     foreach ($validator->getMessages() as $message) {
     echo "$message\n";
     }
}
```

Ponekad je potrebno obaviti složenu validaciju korisničkog unosa. Iz tog razloga moguće je kombinirati više konkretnih validatora u jednoj provjeri. U nastavku se provjerava predano korisničko ime. Zahtjev je da se korisničko sastoji od barem 6 i najviše 12 znakova te da budu isključivo alfa-numerički znakovi (slova i brojevi). Kako bi se to ostvarilo jednim validatorom instancira se općenita klasa *Zend_Validate* te se pomoću metode *addValidator* dodaju željeni validatori. Isječak ispod prikazuje validaciju korisničkog imena po prethodno navedenim pravilima.

```
// instanciraj općeniti validator
$validatorChain = new Zend_Validate();

$validatorChain->addValidator(new Zend_Validate_StringLength(6,
12))
           ->addValidator(new Zend_Validate_Alnum());

// Validacija korisničkog imena
if ($validatorChain->isValid($username)) {
     // korisničko ime je ispravno
} else {
     // korisničko ime nije ispravno; ispiši razloge
     foreach ($validatorChain->getMessages() as $message) {
           echo "$message\n";
     }
}
```

4.2.2. Siguran pristup bazi podataka

Prema projektu OWASP TOP 10, najčešći napadi usmjereni na web aplikacije su napadi umetanjem koda (injekcije). Osim toga, zbog česte ovisnosti o pozadinskoj bazi podataka najčešći oblik injekcijskog napada na web aplikacije su SQL injekcije. „Obične“ PHP aplikacije izravno pozivaju metode za pristup bazi podataka te pritom koriste neke od postojećih funkcija za sprječavanje (engl. *escaping*) injekcija. Nedostatak ovog pristupa je potreba za ručnim upravljanjem unosom prilikom svakog pristupa bazi. Ovaj proces je izvor mnogih sigurnosnih ranjivosti jer izostanak filtriranja samo jednog unosa može u potpunosti narušiti sigurnost aplikacije. Zend nudi poseban API za pristup bazi podataka. Uporabom tih komponenata moguće je transparentno filtrirati sve štetne unose. Ovisno o odabranoj klasi za pristup bazi podataka, upiti se grade drugačije. U nastavku je primjer *SELECT* naredbe koja se prosljeđuje metodom *fetchAll*. Ta metoda interno zamjenjuje upitnike u SQL naredbi sa predanim parametrima i vraća rezultat upita.

```
$sql = "SELECT id FROM _users WHERE lastname=? AND age=?";
$params = array('Smith', '18');
$res = $db->fetchAll($sql, $params);
```

Sigurniji način dohvata postiže se klasom **Zend_Db_Select**. Klasa služi za dinamičko stvaranje SELECT upita.

```
$select = $db->select()
      ->from('products',
            array('product_id', 'product_name', 'price'))
      ->where("price < $minimumPrice OR price > $maximumPrice")
      ->where('product_name = ?', $prod);
```

5. Usporedba s drugim web razvojnim okruženjima

U nastavku poglavlja se uspoređuju druga postojeća web razvojna okruženja sa Zendom. Zbog velikog broja takvih razvojnih okruženja usporedit će se samo najpopularnija razvojna okruženja. U vrijeme pisanja ovog dokumenta najpopularnija web razvojna okruženja za programski jezik PHP su CakePHP, Yiiframework i Codeigniter. U nastavku se ta razvojna okruženja uspoređuju za Zend okruženjem. Radi cjelovitosti Zend se uspoređuje s drugim popularnim web razvojnim okruženjima kao što su Rubby on Rails, JSP i ASP.NET.

5.1. CakePHP

CakePHP je web razvojno okruženje otvorenog koda izrađeno u programskom jeziku PHP. Projekt je započet 2005. godine i oblikovan je pomoću koncepata Ruby on Rails razvojnog okruženja. Iako koristi mnoge koncepte Rails okruženja, ne radi se o pokušaju prevođenja tog okruženja u jezik PHP. CakePHP se koristi za razvoj raznolikih aplikacija, a popis nekih stranica koje su izgrađene pomoću CakePHP-a može se pronaći na službenim stranicama ili u literaturi [6]. Kao i Zend, CakePHP je napravljen po uzoru na MVC obrazac. Ipak, CakePHP je znatno jednostavniji za korištenje pogotovo kod početnika. Iako oba okruženja nude veliku razinu modularnosti i uporabu gotovih komponenata, CakePHP većinu postavki i operacija definira i obrađuje samostalno dok Zend zahtjeva dublje poznavanje korištene komponente. Iz tog razloga pogodniji je za manje projekte gdje se u relativno malom vremenskom razdoblju mora razraditi funkcionalan prototip. Izrada istog projekta korištenjem Zend okruženja mogla bi trajati dulje ukoliko se odabrane komponente moraju dodatno konfigurirati ili preoblikovati. CakePHP zahtjeva znatno manje poznavanje objektnog programiranja budući da nije nužno ručno konfigurirati gotove komponente. Nedostatak okruženja CakePHP je manja fleksibilnost u odnosu na Zend. Zbog unaprijed pretpostavljenog ponašanja komponenti teško je unositi veće promjene bez mijenjanja samog izvornog koda komponente. Komponente Zend okruženja mogu promijeniti svoje ponašanje jednostavnom manipulacijom postavki putem definiranog sučelja što ih čini vrlo fleksibilnim. Iz tog razloga se Zend okruženje koristi u većim projektima gdje se promjene zahtjeva događaju često.

5.2. Yiiframework

Yiiframework je web razvojno okruženje otvorenog koda izrađeno u programskom jeziku PHP. Yii je akronim za „Yes It Is!“. Yii je projekt započet početkom 2008. godine u cilju poboljšanja PRADO[14] web razvojnog okruženja. Nedostatak PRADO okruženja je bio spor odaziv prilikom prevođenja složenih stranica weba, te nije bio pogodan za početnike i mnoge komponente se nisu mogle prilagoditi na jednostavan način. Dolaskom Yii okruženja navedeni nedostaci su uklonjeni. Kao moderno web razvojno okruženje Yii je građen korištenjem MVC obrasca. Nudi gotove komponente za obavljanje raznih funkcija kao što su pristup bazi podataka i obavljanje upita, validacija korisničkog unosa, automatizirano generiranje kompleksnih WSDL specifikacija za servise, pretpohranjivanje (engl. caching) sadržaja i drugo. Ujedno je i jedino razvojno okruženje za programski jezik PHP koje nalikuje okruženju ASP.NET. Ipak, nije primarni izbor za početnike, ali niti za veće projekte. Razlog tomu je manjak funkcionalnosti u odnosu na okruženja poput CakePHP, Codeigniter i Zend. Dodatno, Yii je relativno mlado razvojno okruženje. U vrijeme pisanja nalazi se u inačici 1.0., što je apriorni nedostatak zbog mogućih neuočenih grešaka i znatno manje zajednice korisnika u odnosu na ostala okruženja.

5.3. Codeigniter

Codeigniter je, također, web razvojno okruženje otvorenog koda pisano u programskom jeziku PHP. Cilj okruženja je omogućiti programerima razvoj složenih aplikacija korištenjem bogatih biblioteka gotovih funkcija. Prva inačica okruženja Codeigniter je izdana 2006. godine, a trenutna stabilna inačica je 2.0. Okvirno se bazira na MVC obrascu. Točnije, kontroleri i pogledi su nužni dio okruženja, ali model nije i rijetko se koristi. Ovo razvojno okruženje se ističe svojim performansama. Najbrži je od prethodno spomenutih okruženja uključujući Zend okruženje. Jedan od tvoraca programskog jezika PHP, Rasmus Ledford, na konferenciji frOSCon 2008. godine pohvalio je Codeigniter okruženje zbog visokih performansi, velike zajednice korisnika i jednostavnosti korištenja. Ipak, postoje određene mane zbog kojih se ipak preferira Zend okruženje unatoč manjim performansama. Neke od tih mana su:

- podrška za predloške (engl. *templates*),
- korisničke kontrole,
- integracija sa programskim jezikom jQuery te
- kompatibilnost sa PHP inačicom 4.

Zadnja stavka (kompatibilnost sa PHP 4) se broji kao mana iz razloga što se u program uključuje velika količina nepotrebnog koda što smanjuje preglednost i održavanje.

[14] PRADO – web razvojno okruženje temeljeno na događajima i komponentama. PRADO uvodi nove koncepte u razvoj web aplikacija kao što su događaji i svojstva. Ovi novi koncepti zamjenjuju postojeće koncepte procedura, URL-a i parametara u upitima koji se javljaju u ostalim okruženjima.

5.4. Ruby on Rails

Ruby on Rails je razvojno okruženje otvorenog koda za programski jezik Ruby. Projekt je započeo 2005. godine, a danas se smatra jednim od najboljih i najpotpunijih razvojnih okruženja za izradu web aplikacija. Kao i sva spomenuta okruženja, Rails koristi MVC oblikovni obrazac za organizaciju programskog koda. Velika prednost ovog okruženja su dodaci koji se koriste prilikom razvoja aplikacija. Neki od tih dodataka su:

- **Scaffolding** – metoda meta-programiranja koja služi za izradu dinamičkih upita prema bazi podataka. Pozadinski jezični procesor pomoću meta podataka gradi upite i vraća rezultate aplikaciji.
- **WEBrick** – jednostavan ugrađeni web poslužitelj.
- **RubyGems** – metoda upravljanja programskim paketima slična standardnim Linux alatima poput *yum* i *apt-get.*

Osim navedenih dodataka, Rails podržava velik broj tehnologija kao što su Ajax[15], SOAP[16], JavaScript, jQuery i druge. Zahvaljujući fleksibilnom načinu pristupa podatkovnom sloju lako je promijeniti pozadinsku bazu podataka. Prilikom zamjene baze podataka potrebno je izmijeniti određene konfiguracijske datoteke (dok je kod Zend okruženja potrebno prilagođavati upite).

5.5. Java Server Pages

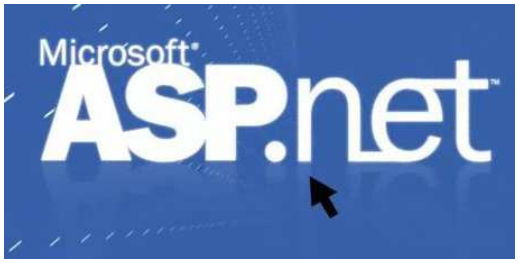
Java Server Pages ili JSP je tehnologija bazirana na programskom jeziku Java koja se koristi za izradu dinamičkih web stranica u HTML, XML i drugim formatima. Projekt je započet 1999. godine kao odgovor na postojeće tehnologije za izradu web aplikacija kao što su ASP i PHP. Koristeći bogatu biblioteku gotovih komponenata programskog jezika Java i napredna okruženja za rad, JSP predstavlja moćnu tehnologiju za izradu web aplikacija. Za razliku od spomenutih programskih jezika, Java nije interpretirajući jezik već koristi virtualni jezični procesor. Java je potpuno objektno orijentirani, strogo tipizirani[17] programski jezik. Budući da je strogo tipiziran i inzistira na poznavanju objektne paradigme nije pogodan za početnike. U zadnje vrijeme se sve više pojavljuje trend prijelaza s ove platforme na PHP i Rails zato što implicitna složenost izgradnje web aplikacije putem strogo objektnog jezika uzrokuje dulji vremenski period razvoja. Dulji vremenski period izgradnje često nije opravdan budući da je programski jezik Java znatno manje fleksibilan u odnosu na nove jezike za izgradnju web aplikacija (npr. PHP i Ruby) što stvara dodatnu prepreku za nadogradnju i dugoročno održavanje.

[15] Ajax – (engl. *Asynchronous JavaScript and* XML) tehnologija koja omogućuje transparentnu komunikaciju između klijenta i poslužitelja bez potpunog osvježavanja trenutne stranice.

[16] SOAP – (engl *Simple Object Access Protocol*) protokol koji služi za izmjenu strukturiranih informacija web usluga u računalnim mrežama. Za prijenos sadržaja se koristi jezik XML i koristi protokole na aplikacijskom sloju za prijenos podataka.

[17] Strogo tipiziranje – (engl *Strong Typing*) u računarskoj znanosti označava obilježje programskih jezika koja zahtjeva izričito definiranje tipova svih varijabli u programu. Suprotno strogom tipiziranju je slabo tipiziranje (engl. *Loose Typing*) gdje se ne zahtjeva definiranje tipova varijabli prije uporabe.

5.6. ASP.NET



ASP.NET je programsko okruženje za razvoj web aplikacija izdano pod Microsoft licencom. Projekt je razvijen 2002. godine i predstavlja nasljednika ASP okruženja. Izrađen je iznad CLR[18] okruženja što omogućuje programerima izradu aplikacija korištenjem ASP.NET okruženja i bilo kojeg programskog jezika podržanog CLR-om (npr. C#, J#, C++ ili VB.NET). Znatno popularnije okruženje u odnosu na JSP, no gotovo jednako složeno i strogo tipizirano kao i programski jezik Java. Zbog zatvorenosti izvornog koda i visoke cijene licenciranja nije primarni izbor za izradu manjih web aplikacija. Dodatna karakteristika ASP.NET okruženja je brz stupanj razvoja. Svakom iteracijom implementira se podrška za nove tehnologije kao što su *AJAX, LINQ* i *MVC*. ASP.NET je usko vezan uz poslužiteljsku platformu tvrtke Microsoft, performanse se optimiziraju na svim razinama pa tako i na poslužitelju, što omogućava kreiranje unaprijed prevedenih i pripremljenih komponenata koje se znatno brže izvode. Ukoliko se ne koristi poslužitelj s operacijskim sustavom tvrtke Microsoft, performanse se mogu smanjiti. Iz tog razloga često se ne koristi u tvrtkama koje koriste Linux/Unix poslužitelje.

[18] CLR – (engl. *Common Language Runtime*) je jezgrena komponenta Microsoft .NET okruženja. Predstavlja konkretnu implementaciju CLI (engl. *Common Language Infrastructure*) standarda koji definira okruženje za izvođenje programa. Putem CLR sustava, programski kod se prevodi u poseban oblik međukoda poznat kao CIL (*Common Intermediate Language*). Programeri mogu koristiti bilo koji programski jezik koji je podržan CLR infrastrukturom za izradu svojih aplikacija u .NET okruženju.

6. Zaključak

Programski jezik PHP stekao je veliku popularnost u zadnjih 10 godina te je postao jednim od glavnih tehnologija za razvoj web aplikacija. Velika prihvaćenost i dominacija jezika PHP vidljiva je i kroz brojna razvojna okruženja kako što je i Zend Framework. Zend pomaže u izgradnji i testiranju dinamičkih web stranica, web aplikacija i usluga. Također, pomaže u rješavanju učestalih problema koji se pojavljuju prilikom projektiranja web aplikacija. Komponente ovog web razvojnog okruženja teže što manjem broju zavisnosti prema drugim komponentama. Ovim se postiže suglasnost sa jednim od osnovnih koncepata objektno orijentiranog programiranja, smanjivanje sprege (engl. *coupling*). Osim toga, smanjivanje sprege se postiže korištenjem arhitektonskog obrasca MVC koji odvaja funkcionalni dio koda od grafičkog sučelja (što često predstavlja problem prilikom izrade novih projekata).

Zend je jedno od najpopularnijih PHP razvojnih okruženja. Može se koristi u svrhu izrade složenih i jednostavnih aplikacija. Samim time potrebno je osigurati ispravan, posebice siguran, rad svih komponenata Zend biblioteke. Ranjivost samo jedne komponente može kompromitirati velik broj web aplikacija (odnosno svih aplikacija koje tu komponentu koriste). Zahvaljujući velikoj korisničkoj zajednici takvih incidenata je malo. Veći dio ranjivosti web aplikacija ima izvorište u programskom kodu same aplikacije. Nepravilnim ili neodgovarajućim filtriranjem i validacijom korisničkog unosa stvara se potencijalna ranjivost u sustavu. Zend pomaže u ovom pogledu pružajući gotov skup komponenti za validaciju i filtriranje sadržaja. U tom pogledu uspješno se takmiči sa ostalim popularnim web razvojnim okruženjima kao što su ASP.NET i Ruby on Rails.

Ipak, zbog velikog broja gotovih komponenti koje je potrebno konfigurirati na odgovarajuć način, te zbog složenosti korištenja oblikovnog obrasca MVC, Zend nije namijenjen za početnike. Za manje projekte i za početnike pogodna su druga okruženja poput CakePHP i Codeigniter koji imaju manje naglašenu ovisnost o MVC obrascu i jednostavniji su za uporabu. Dolaskom programskog jezika PHP, Java programeri postepeno napuštaju tehnologije poput JSP i prelaze upravo na PHP. Autor Zend okruženja Zeev Suraski smatra da ća kroz narednih nekoliko godina trend razvoja aplikacija korištenjem jezika PHP naglo narasti, a primarni razlog će biti korištenje razvojnih okruženja poput Zenda.

Navedene činjenice i dosadašnji razvoj upućuje na daljnji razvoj i usavršavanje Zend okruženja. Ovo potvrđuje i službeni plan nadogradnje (engl. *Roadmap*) za inačicu 2.0. Ciljevi nove inačice su olakšati učenje jezika početnicima, zatim olakšati proširivanje standardne biblioteke (što će dodatno ubrzati razvoj aplikacija), poboljšanje performansi te druga poboljšanja same jezgre koja će olakšati održavanje. Iz navedenoga se jasno prepoznaje da Zend okruženje još nije dostiglo svoj vrhunac, iako se već sada nalazi među najpopularnijim web razvojnim okruženjima. Očekuje se da će daljnja poboljšanja osigurati da ostane među najkorištenijim web razvojnim okruženjima.

7. Leksikon pojmova

Ajax (Asynchronous JavaScript and XML)

Tehnologija weba koja omogućuje transparentnu komunikaciju između klijenta i poslužitelja bez osvježavanja trenutne stranice. Kao i druge slične tehnologije (DHTML, LAMP), Ajax nije jedna tehnologija već skupina tehnologija. Koristi kombinaciju HTML i CSS jezika kako bi prenijela informaciju. DOM (engl. Document Object Model) modelu se pristupa putem JavaScript isječaka kako bi se dinamički izmijenio sadržaj web stranice.

http://en.wikipedia.org/wiki/Ajax_%28programming%29

CIL (Common Intermediate Language)

Poseban oblik međujezika zvan bytecode koji nastaje prevođenjem CLI jezika .NET platforme. Klasični jezici poput programskog jezika C i C++ se prevode izravno u asembler, no jezici koji koriste virtualni jezični procesor se prevode u poseban međujezik. Za jezike .NET platforme to je originalno bio jezik MSIL (MicroSoft Intermediate Language), no novi naziv je CIL.

http://www.scriptol.com/programming/cil.php

http://en.wikipedia.org/wiki/Common_Intermediate_Language

CLI (Common Language Infrastructure)

Otvorena specifikacija koju je izdala tvrtka Microsoft, a standardizirala ISO i ECMA. Specifikacija opisuje izvršni programski kod i okruženje koje čini jezgru Microsoft .NET platforme te besplatne implementacije otvorenog koda Mono i Portable.NET. Specifikacijom se definiraju okruženje koje omogućuje uporabu niza programskih jezika više razine prilikom izrade programskog proizvoda. Ovim se omogućuje migracija postojećeg koda na druge platforme bez dodatne obrade i prilagodbe. Neki od podržanih jezika su C#, VB.NET i J#.

http://en.wikipedia.org/wiki/Common_Language_Infrastructure

CLR (Common Language Runtime)

Jezgrena komponenta Microsoft .NET okruženja. Predstavlja konkretnu implementaciju CLI (engl. Common Language Infrastructure) standarda koji definira okruženje za izvođenje programa. Putem CLR sustava, programski kod se prevodi u poseban oblik međukoda poznat kao CIL (Common Intermediate Language). Programeri mogu koristiti bilo koji programski jezik koji je podržan CLR infrastrukturom za izradu svojih aplikacija u .NET okruženju.

http://msdn.microsoft.com/en-us/library/8bs2ecf4.aspx

CSRF (Lažiranje zahtjeva za web stranicom)

Napad na web stranice (Cross-Site Request Forgery) koji iskorištava povjerenje web stranice/aplikacije prema legitimnom autoriziranom korisniku za izvođenje zlonamjernih radnji. Svrha napada je obično krađa povjerljivih informacija o autoriziranom korisniku, a napad se često dostavlja metodama društvenog inženjeringa. Točnije, žrtvi se dostavlja poveznika koja djeluje poznato, te kada korisnik otvori poveznicu pokreće se napad.

https://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29

CSS (Cascading Style Sheets)

Opisni programski jezik koji služi za definiranje prikaza grafičkih elemenata na web sjedištu. Služi za upravljanje prikazom HTML elemenata. Koristi se za odvajanje funkcijske logike od logike za prikaz sadržaja. CSS je postao standard za izradu grafičkog sučelja web sjedišta, te je World Wide Web Consortium's (W3C) preporuka.

http://www.w3schools.com/css/default.asp

DOM (Document Object Model)

Platformski i jezično neovisna metoda pristupa objektima u jezicima HTML, XHTML i XML. Objekti DOM modela (kao što su HTML elementi) mogu se adresirati i manipulirati neovisno o jeziku koji se koristi.

http://en.wikipedia.org/wiki/Document_Object_Model

http://www.w3.org/DOM/

E-mail (Elektronička pošta)

Predstavlja način prijenosa tekstualnih poruka putem komunikacijskih mreža, najčešće Interneta. Usluga omogućava umetanje dodatnih datoteka kao privitke (engl. attachment), a ovisno o poslužitelju usluge može postojati ograničenje na količinu, veličinu i tip datoteka. Elektronička pošta je postala standard za poslovnu komunikaciju, te je zamijenilo standardne dopise (dopisi se i dalje šalju ali putem elektroničke pošte). Nedugo nakon popularizacije elektronička pošta je postala medij za prijenos raznih zlonamjernih, štetnih programa kao što su crvi i virusi. Uporabom raznih heurističkih metoda prepoznavanja ovo se većinom spriječilo, no i dalje se dnevno razmjenjuju razne (bezopasne) spam ili junk poruke kojima je cilj reklamirati neki proizvod ili uslugu.

http://en.wikipedia.org/wiki/Email

HTTP (HyperText Transfer Protocol)

Osnovna i najčešća metoda prijenosa informacija na Webu. Predstavlja protokol na aplikacijskom sloju OSI modela, a osnovna namjena je prijenos HTML dokumenata (tj. web stranica). HTTP je request/response protokol za komunikaciju između poslužitelja i klijenta. HTTP klijent, kao što je web preglednik najčešće inicira prijenos podataka nakon što uspostavi TCP vezu s udaljenim web poslužiteljem na određenom priključku. Poslužitelj konstantno osluškuje zahtjeve na određenom mrežnom komunikacijskom priključku (tipično priljučak 80), čekajući da klijent inicira komunikaciju.

http://www.w3.org/Protocols/

http://hr.wikipedia.org/wiki/HTTP

JavaScript (Programski jezik JavaScript)

JavaScript je skriptni programski jezik, koji se izvodi u web pregledniku na strani korisnika. Napravljen je da bude sličan Javi, zbog lakšega korištenja, ali nije objektno orijentiran kao Java, već se temelji na prototipu i tu prestaje svaka povezanost s programskim jezikom Java. Izvorno ga je razvila tvrtka Netscape (www.netscape.com). JavaScript je izrađen primjenom ECMAScript standarda.

http://en.wikipedia.org/wiki/JavaScript

LFI (Uključivanja lokalnih datoteka)

Napad uključivanja lokalnih datoteka (engl. Local File Inclusion) predstavlja napad na web stranice/aplikacije koji iskorištava propuste u filtriranju korisničkog unosa kako bi se dobio prikaz proizvoljnih datoteka na poslužitelju. Uspješnim izvođenjem napada prikazuje se sadržaj ciljane datoteke, a ne originalni sadržaj web stranice. Česta meta na Unix/Linux operacijskim sustavima je datoteka /etc/passwd, te datoteka C:\WINDOWS\system32\drivers\etc\hosts na Windows operacijskim sustavima.

https://www.owasp.org/index.php/PHP_File_Inclusion

www.exploit-db.com/download_pdf/13678/

MVC (Model-View-Controller)

Arhitektonski oblikovni obrazac, odnosno arhitektura izrade programske potpore, koja omogućuje raslojavanje poslovne, podatkovne i prezentacijske logike. Točnije, odvaja kod koji reprezentira problem domene od koda koji prezentira problem korisniku (grafičko sučelje). MVC znatno utječe na organiziranost i čitljivost programskog koda te je postao standard u pisanju modernih web aplikacija.

http://msdn.microsoft.com/en-us/library/ff649643.aspx

http://csis.pace.edu/~bergin/mvc/mvcgui.html

http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

PHP (Hypertext Preprocessor)

Objektno-orijentiran programski jezik namijenjen prvenstveno za izradu dinamičnih web sjedišta. PHP je besplatan proizvod, objavljen pod PHP License licencom. Sintaksom je vrlo sličan popularnim jezicima poput C/C++, Java i Perl, a u potpunosti je implementiran u programskom jeziku C. Zbog jednostavnosti uporabe i visoke popularnosti postao je jednim od najpopularnijih jezika za izradu web sjedišta i usluga. Za razliku od jezika C/C++ i Jave koji su strogo tipizirani, PHP nema tipove podataka kao većina skriptnih jezika.

http://www.php.net/

http://www.w3schools.com/php/default.asp

http://en.wikipedia.org/wiki/PHP

Regularni izrazi (Regex)

Izrazi kojim se definira uzorak koji se koristi za pretraživanje teksta. Regularni izrazi se, kao i svi drugi matematički izrazi, sastoje od operatora i operanada. Operandi su jezici (skupovi riječi), a operatori oni već dobro poznati iz teorije skupova – unija, produkt i slično. Regularne izraze koriste mnogi uređivači teksta i pomoćni programi za pretragu i manipulaciju teksta ovisno o nekim uzorcima. Mnogi programski jezici podržavaju regularne izraze za manipulaciju stringovima.

http://www.java.hr/node/181

http://www.webmajstori.net/clanci/programiranje/regularni-izrazi-teorija-i-praksa-regularni-izrazi-u-teoriji/137/

Repozitorij (Repozitorij objekata)

Ključni entitet u domensko-orijentiranom dizajnu (engl. Domain-Driven Design). Enkapsulira svu logiku perzistencije korisničkih objekata što ostatku objekata u domeni nudi neovisnost o konkretnoj metodi perzistencije - baza podataka, datoteka i drugo. Također, osigurava sučelje za dohvat željenih objekata skrivajući konkretan način dohvata objekata.

http://geekswithblogs.net/gyoung/archive/2006/05/03/77171.aspx

http://evan.bottch.com/2007/12/06/factory-and-repository-in-the-domain/

Salt (Salt dodatak)

Kriptografska metoda koja se koristi za otežavanje napada rječnikom prilikom pogađanja lozinke. Nasumični niz bitova se dodaje lozinki prije nego što se proizvede sažetak (koristeći SHA1, MD5 ili neki drugi algoritam). Napadač mora postojeći rječnik ponovno proizvesti sa odgovarajućom salt vrijednošću (spomenuti nasumični niz bitova) što produljuje vrijeme potrebno za otkrivanje lozinke.

http://www.ucertify.com/article/salt-cryptography.html

http://www.bookrags.com/wiki/Salt_%28cryptography%29

Singleton (Oblikovni obrazac Singleton)

Konstrukcijski oblikovni obrazac koji se koristi kada je potrebno osigurati da uvjek postoji samo jedna instanca objekta za vrijeme izvođenja aplikacije. Korištenjem jedne instance je pogodno kada postoje različite konkurentne operacije (npr. zapisivanje u datoteku) jer se time sprječavaju "sudari" budući da samo jedna instanca radi upis.

http://www.dofactory.com/Patterns/PatternSingleton.aspx

http://www.fluffycat.com/Java-Design-Patterns/Singleton/

SOAP (Simple Object Access Protocol)

Protokol koji služi za izmjenu strukturiranih informacija web usluga u računalnim mrežama. Za prijenos sadržaja se koristi jezik XML i koristi protokole na aplikacijskom sloju za prijenos podataka.

http://www.w3.org/TR/soap/

http://www.w3schools.com/soap/default.asp

Unit Testing (Testiranje funkcionalnih jedinica)

Metoda testiranja programske podrške u kojoj se svaka funkcionalna jedinica koda testira zasebno kako bi se ustanovilo je li spremna za uporabu.

http://msdn.microsoft.com/en-us/library/aa292197%28v=vs.71%29.aspx

http://en.wikipedia.org/wiki/Unit_testing

URL (Uniform Resource Locator)

URL predstavlja adresu određenog resursa na Internetu. Resurs na koji pokazuje URL adresa može biti HTML dokument, slika, datoteka ili bilo koja datoteka koja se nalazi na određenom web poslužitelju.

http://en.wikipedia.org/wiki/Uniform_Resource_Locator

http://www.ietf.org/rfc/rfc1738.txt

http://searchnetworking.techtarget.com/definition/URL

8. Reference

[1] WASC Web Application Security Statistics Project, http://projects.webappsec.org/w/page/13246989/Web-Application-Security-Statistics

[2] http://www.noginn.com/2009/03/01/preventing-csrf-properly/

[3] http://en.wikipedia.org/wiki/Session_fixation

[4] Zend Security Advisories, http://zendframework.com/security/advisories

[5] Secure programming with Zend, http://www.suspekt.org/downloads/DPC_Secure_Programming_With_The_Zend_Framework.pdf

[6] Stranice izrađene pomoću CakePHP okruženja, http://book.cakephp.org/view/510/Sites-in-the-wild

[7] R. Allen, N. Lo, S. Brown, Zend Framework in Action, Manning, 2009.