Univerzitet u Beogradu

Fakultet organizacionih nauka

Katedra za elektronsko poslovanje

Simulacija i simulacioni jezici

Seminarski rad

Tema: Razvoj aplikacija u Simfony okruženju

Studenti: Mentor:

Radomir Popović 643/12 Dušan Barać

Nikola Ranđelović 95/12

Uroš Pešić 125/12

Beograd, 2016

Sadržaj

O Simfony-ju 3

Instalacija Symfony paketa 5

Rad u Symfony okruženju 8

Twigovi 12

Generisanje kompletnog CRUD dokumenta 14

Zaključak 15

## O Simfony-ju

Symfony predstavlja jedan od danas najzastupljenijih framework-a za pravljenje PHP aplikacija. Framework sam po sebi predstavlja skup biblioteka, alata, funkcija, metoda, klasa sa kojima korisnik mnogo lakše razvija svoju aplikaciju. Ukratko, Symfony je jedan vid razvojnog okruženja. Primera radi, ukoliko želimo da ispišemo sve elemente neke tabele iz baze, ne moramo pisati svaku liniju koda detaljno (otvaranje konekcije, while petlja, izvlačenje pojedinačno svakog elementa, pisanje sql upita itd). Ovakav problem može se uveliko pojednostaviti pomoću navedenog framework-a.

{% for k in knjige %}

<tr>

<td>{{ k.id }}</td>

<td>{{ k.autor }}</td>

<td>{{ k.naziv }}</td>

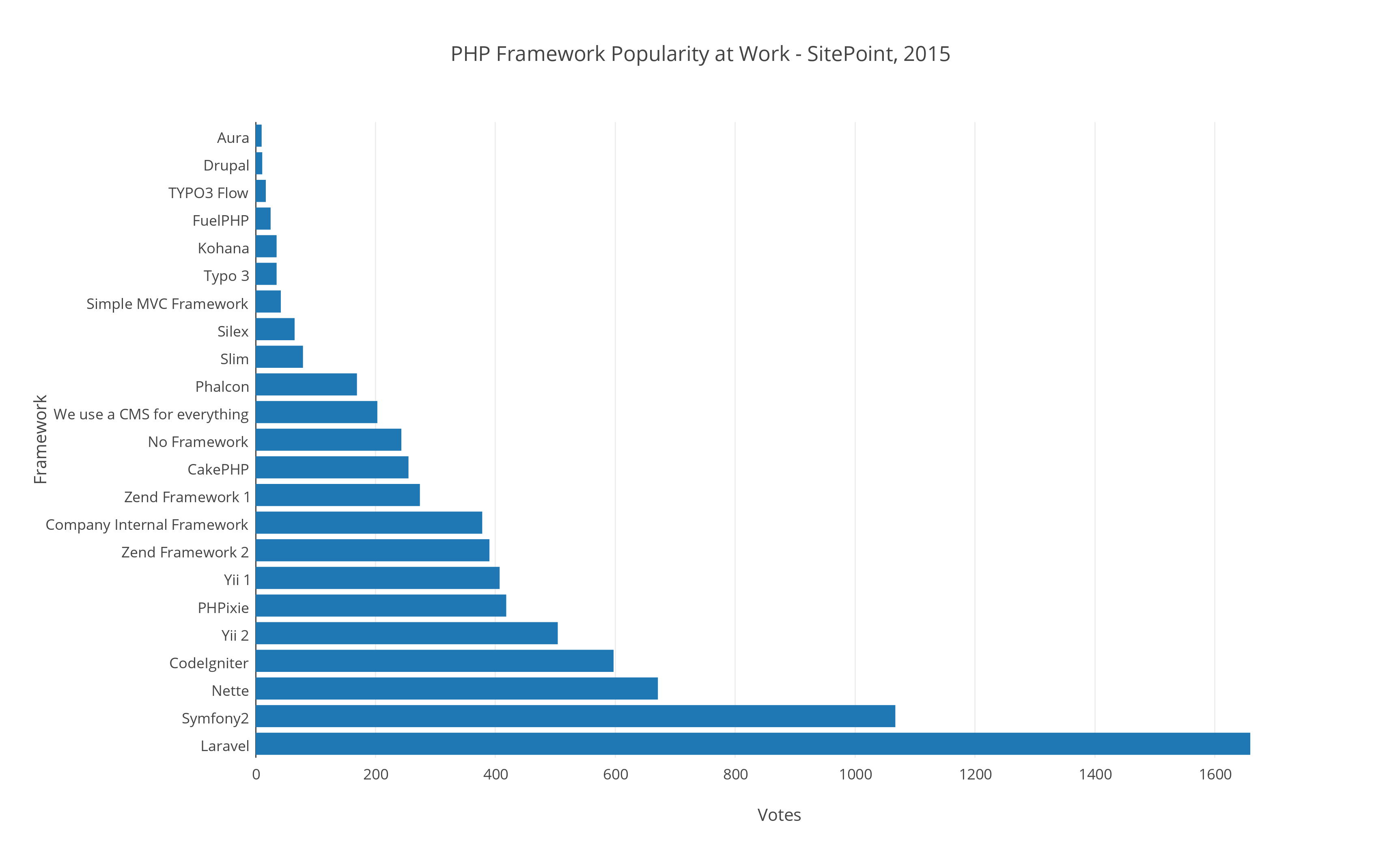
</tr>

{% endfor %}

Sintaksa izgleda malo drugačije u odnosu na pisanje regularnog php koda. Ovde je primetno korišćenje takozvanih twig-ova o kojima će biti reči kasnije.

PHP predstavlja skriptni jezik, koji se može koristiti dodatno sa HTML/CSS/Javascript tehnologijama zarad dobijanja dinamičke web stranice. Konkretno, PHP je zadužen za serverski deo i on se koristi u situacijama kada želimo da manipulišemo podacima koji nisu vidljivi korisniku.

Pored Symfony-ja postoji još mnoštvo razvojnih okruženja, a ono po čemu se Symfony izdvaja od drugih jeste brzina i pregršt mogućnosti (rutiranje, controlleri, twigovi itd), međutim kako ga ocenjuju programeri, nije najjednostaviji razvojni okvir i ukoliko se programer prvi put susreće sa framewor-cima, verovatno je najpogodniji izbor Laravel, međutim Symfony u poslednje vreme biva sve popularniji i zastupljeniji kod kompanija.



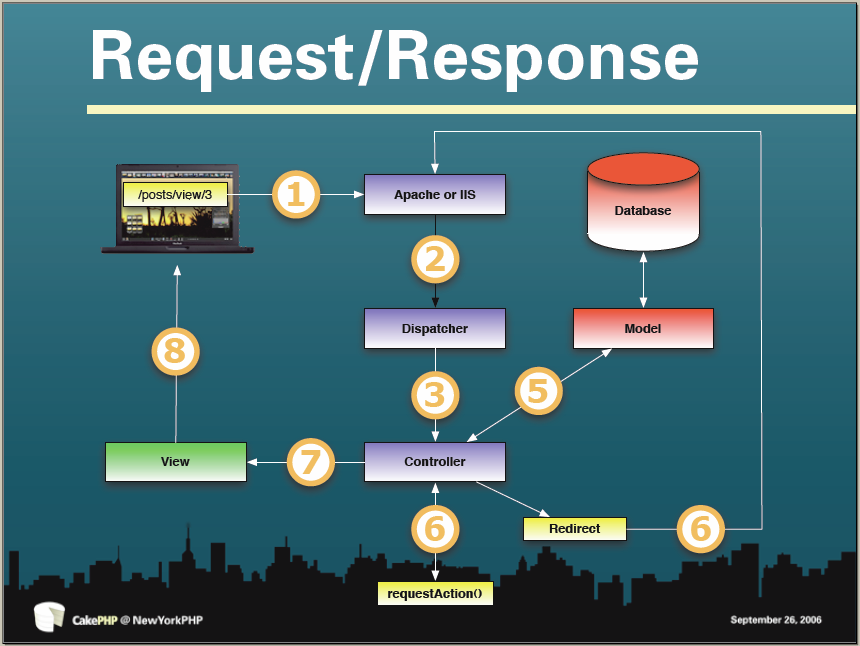
Slika 1 – Popularnost PHP Framework-a

Symfony radi na osnovu jednog od najzastupljenijih paterna danas u programiranju, a to je MVC. Pored njega, u dosta su velikoj upotrebi neki poput Observer, Adapter, Prototype, Singleton, Template itd. MVC je mikroarhitekturni patern koji deli softverski sistem u 3 dela i to :

a) view – deo vidljiv i namenjen korisniku

b) model – predstavlja stanje sistema, vidljivo programeru

c) controller – posrednik između dva prethodna entiteta. Korisnik traži zahtev, controller gleda model i po osnovu njega daje odgovor, ukoliko se desi neka promena u modelu, controller o tome obavestiti view deo – primena Observer paterna. Sam model ne mora da zna ko je controller/view.



Slika 2 – MVC Model

## Instalacija Symfony paketa

Ovde dolazimo do onog dela zašto mnogi ne vole Symfony framework. Na dosta problema može da se naiđe prilikom instalacije, neki su rešivi, neki ne, postoje alternativni načini preko takozvanog Composer alata, međutim ovde će biti reči o najjednostavnijem. Korisnici koji se susreću sa nekim poteškoćama nemaju druge, već da istraže na samom sajtu Symfony-ja kako da na pravi način pokrenu svoje aplikacije preko ovog razvojnog okruženja.

1. **korak** - skidanje Symfony framework-a sa oficijalnog sajta, putem command prompta za Windows, odnosno terminala za Mac računare.

http://symfony.com/download

1. **korak** - podešavanje vremenske zone

Na postojeći fajl AppKernel.php koji se nalazi u folderu app samog projekta dodati na kraj sledeći kod:

publicfunction \_\_construct($environment, $debug) {

parent::\_\_construct($environment, $debug);

// getridofWarning: date\_default\_timezone\_get(): Itisnotsafetorelyonthesystem'stimezone

date\_default\_timezone\_set( 'Europe/Paris' );

}

1. **korak** - osposobljavanje za rad sa bazom podataka

Potrebno je otvoriti fajl parameter.yml koji se nalazi u delu app/config, i u skladu sa parametrima postojeće baze korisnika namestiti opcije.

# This file is auto-generated during the composer install

parameters:

database\_host: 127.0.0.1

database\_port: 8889

database\_name: symfony

database\_user: root

database\_password: root

mailer\_transport: smtp

mailer\_host: 127.0.0.1

mailer\_user: null

mailer\_password: null

secret: 3e079895a1b20386c717e2f6a6f623d1320da708

**3. a** - proveriti da li radi za početak uopšte Symfony nakon navedenih koraka

php bin/console server:start

[ERROR] This command needs the pcntl extension to run.

You can either install it or use the "server:run" command instead to run the

built-in web server.

Do you want to execute server:run immediately? [Yn] [1]:

> 1

[OK] Server running on http://127.0.0.1:8000

// Quit the server with CONTROL-C.

Ukoliko radi, privremeno ugasiti server.

**4. korak** - kreiranje Bundla (radi pravljenja sopstvenih naredbi)

php bin/console generate:bundle

Welcome to the Symfony bundle generator!

Are you planning on sharing this bundle across multiple applications? [no]: no

Your application code must be written in bundles. This command helps you generate them easily.

Give your bundle a descriptive name, like BlogBundle.

Bundle name: KladionicaBundle

Bundles are usually generated into the src/ directory. Unless you're

doing something custom, hit enter to keep this default!

Target Directory [src/]:

What format do you want to use for your generated configuration?

Configuration format (annotation, yml, xml, php) [annotation]: ym

Bundle generation

> Generating a sample bundle skeleton into src/KladionicaBundleOK!

> Checking that the bundle is autoloaded: OK

> Enabling the bundle inside app/AppKernel.php: OK

> Importing the bundle's routes from the app/config/routing.yml file: OK

> Importing the bundle's services.yml from the app/config/config.yml file: OK

Everything is OK! Now get to work :).

**5. korak** - editovati routing.yml fajl koji se nalazi u okviru stvorenog bundle-a, direktorijum Resources/config

kladionica\_moja\_ruta:

path: /moja

defaults: { \_controller: KladionicaBundle:Kladionica:prikaziMetoda }

**6. korak** - napraviti novi fajl KladionicaController.php i okviru njega definisati prikaziMetoda metodu. Primer izgleda metode

$var = $this->getDoctrine()

->getRepository('KladionicaBundle:Kvota')

->findAll();

return $this->render('KladionicaBundle:Kvote:prikaz.html.twig', array("kk" => $var));

**7. korak** - napraviti fajl prikaz.html.twig koji se nalazi u folderu Kvote. Taj fajl će i biti fajl koji će biti prikazan na ekranu.

**8. korak** - pokrenuti server preko terminala, na kraju rada ga ugasiti.

/opciono/ Kada god pokrenete svoju prvu aplikaciju imaćete takozvani toolbar na dnu stranice dokle god radite (dokle god je server upaljen). Preko njega možete dobiti korisne informacije, npr. http status, na kom kontrolleru radite, http parametri kako glase, konfiguracije, rute itd.

## Rad u Symfony okruženju

Sada kada smo ustanovili kako mozemo da pokrenemo našu simfony aplikaciju, vreme je vidimo koje to sve mogućnosti Symfony okruženje pruža. U prethodnim pasusima korišćeni su neki izrazi poput entiteta, rutiranja, bundle-ova i slično. Postoje još neki koji nisu navedeni, a početnicima su nepoznati kao što su twigovi.

Najkraće rečeno, sve počinje od pravljenja našeg Bandle-a. Bandle predstavlja na neki način paket, koji u sebi sadrži određene biblioteke neophodne za manipulaciju sa podacima. U 4. koraku definisan je način na koji možemo napraviti naš sopstveni Bundle. Nije neophodno praviti ga, međutim znatno ograničava rad ukoliko nije napravljen. Bundle u sebi sadrži foldere kao što su to Controller, Resources, Tests, Entity, Repositorium. Nećemo pominjati svaki pojedinačno fajl, već samo one koji su bitni za upotrebu ovog razvojnog okruženja.

Pre nego što napravimo naš kontroler, postoje takozvana 2 front-controllera i to su: app.php i app\_dev.php. Oni predstavljaju izolovane konfiguracije Symfony web aplikacije. Sve zahteve oni prvo prihvataju.

Prilikom instaliranja bundle-a postoji već definisan default controller, međutim mi želimo da napravimo naš Controller, u ovom slučaju to će biti KladionicaController.php, sa sledećim sadržajem:

<?php

namespace KladionicaBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

class KladionicaController extends Controller

{

public function prikaziMetodaAction()

{

$var = $this->getDoctrine()

->getRepository('KladionicaBundle:Kvota')

->findAll();

return $this->render('KladionicaBundle:Kvote:prikaz.html.twig', array("kk" => $var));

}

}

?>

Iz ovog fajla možemo objasniti sledeće:

Metoda pokušava da pristupi fajlu Kvota.php i da uzme sve njegove vrednosti, objektno orijentisanim putem. A kako smo napravili fajl Kvota.php i šta on zapravo predstavlja? On predstavlja Model, to jest Entitet, i pravi se na sledeći način u okviru terminala. ()

Radomirs-MacBook-Pro:kladionicaRadomir$ php bin/console generate:doctrine:entity

Welcome to the Doctrine2 entity generator

This command helps you generate Doctrine2 entities.

First, you need to give the entity name you want to generate.

You must use the shortcut notation like AcmeBlogBundle:Post.

The Entity shortcut name: KladionicaBundle:Kvota

Determine the format to use for the mapping information.

Configuration format (yml, xml, php, or annotation) [annotation]: yml

Instead of starting with a blank entity, you can add some fields now.

Note that the primary key will be added automatically (named id).

Available types:array, simple\_array, json\_array, object,

boolean, integer, smallint, bigint, string, text, datetime, datetimetz,

date, time, decimal, float, binary, blob, guid.

New field name (press <return> to stop adding fields): domacin

Field type [string]: string

Field length [255]: 255

Is nullable [false]: false

Unique [false]: false

New field name (press <return> to stop adding fields): gost

Field type [string]: string

Field length [255]: 255

Is nullable [false]: false

Unique [false]: false

New field name (press <return> to stop adding fields): pobedjujeDomacin

Field type [string]: decimal

Precision [10]: 2

Scale:

Is nullable [false]: false

Unique [false]: false

New field name (press <return> to stop adding fields): pobedjujeGost

Field type [string]: decimal

Precision [10]: 2

Scale:

Is nullable [false]: false

Unique [false]: false

New field name (press <return> to stop adding fields): neresenIshod

Field type [string]: decimal

Precision [10]: 2

Scale:

Is nullable [false]: false

Unique [false]: false

New field name (press <return> to stop adding fields):

Entity generation

> Generating entity class src/KladionicaBundle/Entity/Kvota.php: OK!

> Generating repository class src/KladionicaBundle/Repository/KvotaRepository.php: OK!

> Generating mapping file src/KladionicaBundle/Resources/config/doctrine/Kvota.orm.yml: OK!

Everything is OK! Now get to work :).

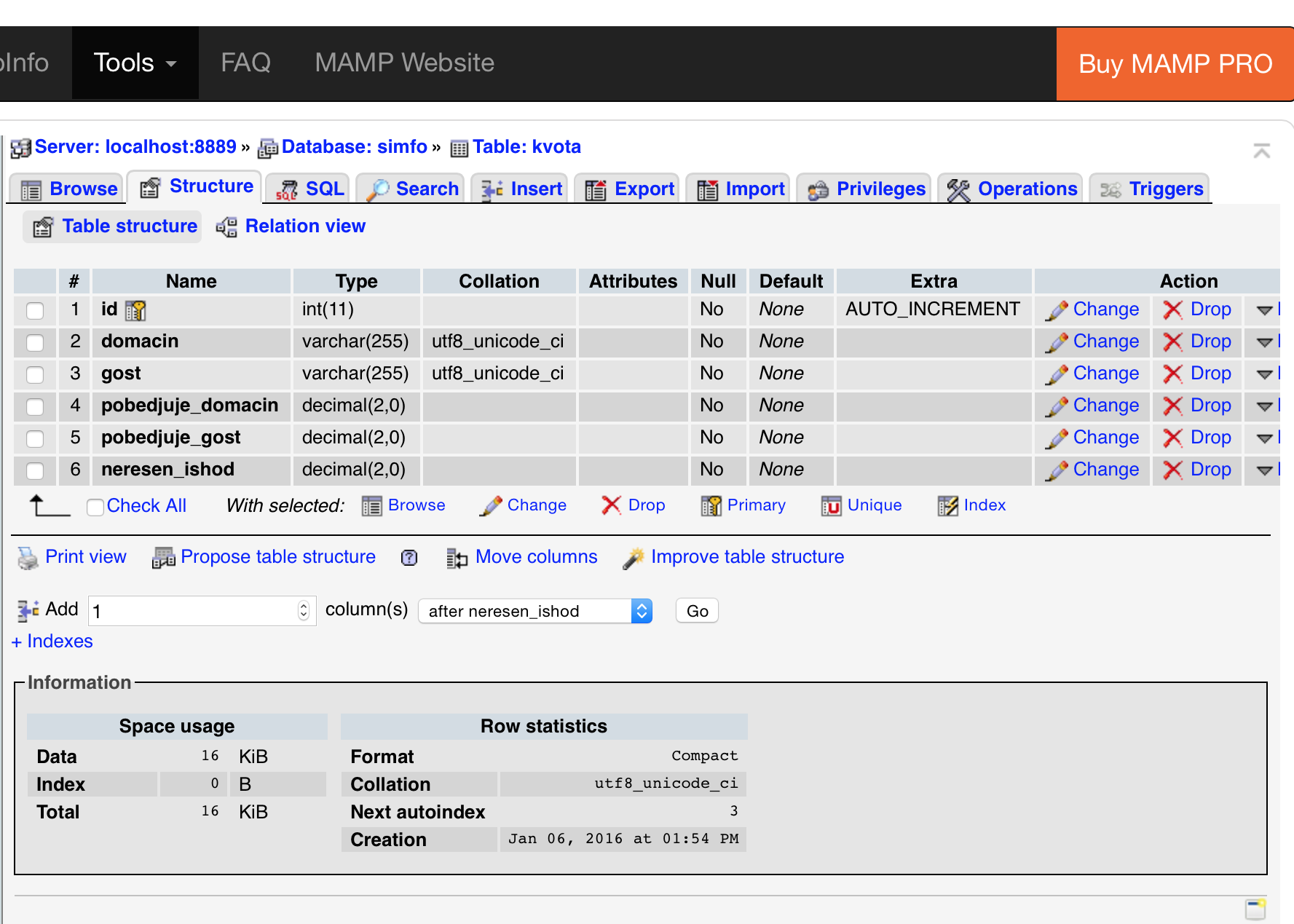
Radomirs-MacBook-Pro:kladionicaRadomir$ php bin/console doctrine:schema:update --force

Updating database schema...

Database schema updated successfully! "1" query was executed

Poslednja komanda jako je važna da bi sve ovo prethodno moglo da radi!

Šta se ovde zapravo desilo? Preko par linija koda napravljena je tabela u mysql bazi!



Slika 3 – MqSql baza

Objasnili smo prvi deo funkcije, sada sledi drugi deo:

return $this->render('KladionicaBundle:Kvote:prikaz.html.twig', array("**kk**" => $var));

Fajl prikaz.html.twig prima prethodno definisanu varijablu $var pod nazivom **kk**, koji se koristi u tom fajlu.

Želimo da otvorimo fajl prikaz.html.twig. Nećemo to uraditi standardnim putem tražeći ga preko direktorijuma, već ćemo ovde upotrebiti jednu bitnu funkcionalnost Symfoni okruženja, a to je rutiranje (rute su ovom slučaju virtuelne!).

U 5. koraku instalacije naveli smo da se dodaje sledeći kod na postojeći:

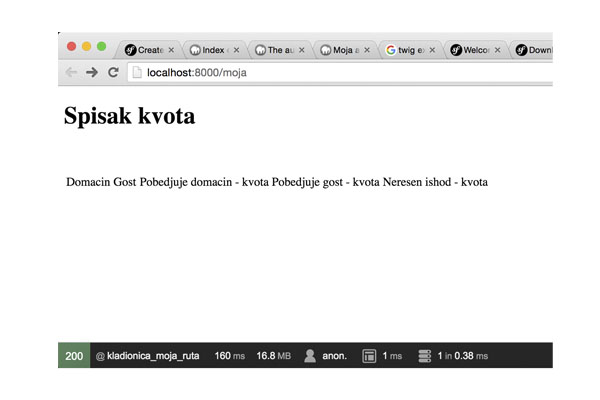
kladionica\_moja\_ruta:

path: /moja

defaults: { \_controller: KladionicaBundle:Kladionica:prikaziMetoda }

Ovde je konkretno navedeno da pristupamo metodi prikaziMetoda, koja mora biti deklarisana u okviru KladionicaController.php fajla. Može se zaključiti da je veoma bitna sintaksa ovde i da je lako pogrešiti, pogotovo kad počinjete da radite na vašoj aplikaciji. Sintaksa je ovde takva, da kada se definiše kontroler neke rute, ne piše se punim imenom, već bez sufiksa Controller.php.

Dakle naš fajl otvaramo preko navedene rute. Kada otvorimo u browser-u stranicu <http://localhost:8000/moja> (pod uslovom da je server prethodno upaljen) pojavljuje se sledeći sadržaj:



<!DOCTYPE html>

<html>

<head>

<title>Moja aplikacija</title>

<meta charset="utf-8" />

<title>{% block title %}Welcome!{% endblock %}</title>

</head>

<body>

<h1>Spisak kvota</h1>

<table style="border: 1px solid black">

<tr>

<td>Domacin</td>

<td>Gost</td>

<td>Pobedjuje domacin - kvota</td>

<td>Pobedjuje gost - kvota</td>

<td>Neresen ishod - kvota</td>

</tr>

</body>

</html>

Slika 4 - Stranica

Trenutno se ne primećuju podaci izvučeni iz baze podataka. To se čini na sledeći način:

Umesto da se stalno pravi pojedinačno ulazak u bazu, while petlja, fetch array metode i sl. uz pomoć symfonija se sve to lako prevazilazi uz pomoć symfony okruženja. Primećuju se vitičaste zagrade. To su takozvani twigovi.

{% for k in **kk**%}

<tr>

<td>{{ k.id }}</td>

<td>{{ k.domacin }}</td>

<td>{{ k.gost }}</td>

<td>{{ k.pobedjujeDomacin }}</td>

<td>{{ k.pobedjujeGost }}</td>

<td>{{ k.neresenIshod }}</td>

</tr>

{% endfor %}

## Twigovi

Twigovi predstavljaju jednu bitnu funkcionalnost ugrađenu u Symfony okruženju. Najjednostavnije rečeno, oni predstavljaju templejt sintakse koja je znatno lakša od standardne php sintakse, i logično koristi se zarad efikasnosti koda. Ovde će biti predstavljeni samo neki bitni koncepti, celokupna dokumentacija može se naći na sledećem linku (preporuka je pogledati šta su to blokovi, kako se koriste escape karakteri, zapis logičkih operacija):

http://symfony.com/doc/current/book/templating.html

Postoji tri tipa zagrade u twigovima i to:

{{ ... }} – zamena za echo

{% ... %} – zamena za izvršavanje radnji, poput petlji

{# ... #} – zamena za komentar

Primer primene prethodne tri zagrade u jednoj liniji koda:

{% set a\_variable = 'bar' %}<h1>My Webpage</h1>{{ a\_variable }}{#Ovo je mnogo lako za koriscenje#}

Prvo smo definisali a\_variable promenljivu, a potom je ispisali, a zatim napisali komentar samo vidljiv nama koliko je zaista laka primena ovih zagrada. Korišćenjem još nekih ključnih reči radi se sa twig-ovima. Primeri:

<ahref="{{path('article\_show',{'id':123,'\_format':'pdf'})}}"> PDF Version</a>

umesto

<ahref="<?phpecho$view['router']->path('article\_show',array('id'=>123,'\_format'=>'pdf',))?>"> PDF Version</a>

<imgsrc="{{asset('images/logo.png')}}"alt="Symfony!"/>

umesto

<imgsrc="<?phpecho$view['assets']->getUrl('images/logo.png')?>"alt="Symfony!"/>

Ukoliko želite da vidite da li vam je tačna sintaksa uvek možete preko terminala proveriti sa sledećom komandom:

$ php bin/console lint:twig app/Resources/views/article/recent\_list.html.twig

Jedna korisna funkcionalnost twig-ova jesu filteri. Primera radi, neku reč možemo ispisati kroz određeni filter, npr. primenićemo uvek abs filter kad god nam treba samo pozitivna vrednost:

{# number = -5 #}

{{ number|abs }}

{% set values = [1, 2] %}

{% set values = values|merge(['apple', 'orange']) %}

{# values niz sada sadrzi vrednosti [1, 2, 'apple', 'orange'] #

## Generisanje kompletnog CRUD dokumenta

Ukoliko želimo da generišemo gotove metode za create, read, update i delete operacije, bez da mi sami ručno to radimo, preko komandi u terminalu dolazimo do pravljenja automaskih ruta, controllera i html.twig fajlova koje kasnije možemo da menjamo.

/Applications/MAMP/htdocs/symfony/kladionica/src/KladionicaBundle/Resources/config/routing/kvota.yml – fajl koji sadrži crud rute

/Applications/MAMP/htdocs/symfony/kladionica/app/Resources/views/kvota – folder koji sadrži 4 crud html.twig fajla

/Applications/MAMP/htdocs/symfony/kladionica/src/KladionicaBundle/Controller/KvotaController.php – sam kontroler

Radomirs-MacBook-Pro:kladionica Radomir$ php bin/console generate:doctrine:crud

Welcome to the Doctrine2 CRUD generator

This command helps you generate CRUD controllers and templates.

First, give the name of the existing entity for which you want to generate a CRUD

(use the shortcut notation like AcmeBlogBundle:Post)

The Entity shortcut name: AppBundle

The entity name isn't valid ("AppBundle" given, expecting something like AcmeBlogBundle:Blog/Post)

The Entity shortcut name: ProbaBundle:Probaj

By default, the generator creates two actions: list and show.

You can also ask it to generate "write" actions: new, update, and delete.

Do you want to generate the "write" actions [no]? yes

Determine the format to use for the generated CRUD.

Configuration format (yml, xml, php, or annotation) [annotation]: yml

Determine the routes prefix (all the routes will be "mounted" under this

prefix: /prefix/, /prefix/new, ...).

Routes prefix [/probaj]:

Summary before generation

You are going to generate a CRUD controller for "ProbaBundle:Probaj"

using the "yml" format.

Do you confirm generation [yes]? yes

CRUD generation

Generating the CRUD code: OK

Generating the Form code: OK

Updating the routing: Confirm automatic update of the Routing [yes]?

Importing the CRUD routes:

[ERROR] The bundle's "Resources/config/routing.yml" file cannot be imported

from "app/config/routing.yml" because the "ProbaBundle" bundle is

already imported. Make sure you are not using two different

configuration/routing formats in the same bundle because it won't work.

OK

Everything is OK! Now get to work :).

## Zaključak

Smatramo da, iako savladavanje na početku ovog framework-a predstavlja na neki način izazov, kada se neke stvari usvoje, korišćenje ovog alata utiče na to, da ne samo da će se php kod pisati u i do nekoliko redova manje, već će i rad na web aplikacijama biti i zanimljiviji korišćenjem koncepta rutiranja, kao i twigova. Da ne zaboravimo, da svako ko želi da se bavi ovom vrstom posla, da će morati za svog poslodavca da nauči neki razvojni okvir, tako da pre ili kasnije, susrešće se sa navedenim, jer danas firme i koje rade preko Laravel-a uveliko prelaze na Symfony.